Data Transfer and Sharing within Web Service Workflows

Donglai Zhang

September 2013

A dissertation submitted to the School of Computer Science of
The University of Adelaide for the degree of Doctor of Philosophy

Supervisors:

Dr. Andrew L. Wendelborn

Dr. Paul D. Coddington

Contents

Abstract

With the development of distributed systems, it is more and more common for users to harness different resources to implement a larger task to meet their requirements.

Among the different approaches to distributed resource coordination, workflows based on Service Oriented Architecture (SOA) is an important case, as SOA provides a framework that is designed for loosely coupled applications. This thesis introduces the research work that we have carried out in distributed computing environments to improve the performance of data transfer and sharing in a web service workflow.

In a distributed environment, we explore how to improve the efficiency of data moving between services in a web service workflow. Data movement in a web service workflow can be categorized into two classes: data transfer between any two nodes in the workflow and the intermediate data sharing between different web services. We initially explored ways to improve the data transfer performance between two web service nodes, then improved the data sharing performance via study of the data sharing relationship between applications composed in a workflow.

We carried out the initial part of the research work based on the CIMA (Common Instrument Middleware Architecture) web service interface, which has been used by multiple academic organizations as an interface to distributed scientific instruments and applications. With the related experiments, we explore how the data generated by instruments can be transferred efficiently between different web service nodes. In the rest of the research, we study the data sharing relationships between different web service applications. By proposing the Web Service Data Forwarding (WSDF) framework, we allow intermediate data to be forwarded directly from the data generator to its consumer without going via a third party (the workflow engine).

We have implemented prototype systems for our proposed ideas. We also tested these systems in different environments to demonstrate the performance improvement that is expected from the WSDF approach.

Thesis Declaration

I, Donglai Zhang certify that this work contains no material which has been accepted for the award of any other degree or diploma in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text.

I give consent to this copy of my thesis when deposited in the University Library, being made available for loan and photocopying, subject to the provisions of the Copyright Act 1968.

I also give permission for the digital version of my thesis to be made available on the web, via the University's digital research repository, the Library catalogue and also through web search engines, unless permission has been granted by the University to restrict access for a period of time.

Donglai Zhang

Acknowledgments

Firstly, I would like to thank my wife and my family to support me with my study. Without your help, I will not have a chance to even start this work.

Secondly, great appreciation to my supervisors, Dr. Andrew Wendelborn and Dr. Paul Coddington, for their invaluable guidance and assistance with my study. Both of them are great supervisors. They are very experienced within the area of this research work and provided significant amount of guidance and feedback that was critical to the project. With their help, I not only gained research experiences, but also learned the fundamental methodologies for future research work as a researcher.

Third, I would thank my friends: Kewen, Paul. M, Peter, Wei, and Yidong — thanks for your help during my study.

Chapter 1

Introduction

In the 21st century, information technology has had a significant impact on every aspect of our life. Powerful IT services also bring new chances and challenges for business, manufacturing, government and other aspects in our society [49], particularly for scientific research [86].

Nowadays, scientific research is extended to an even larger scale. A common scenario is that research works are getting more and more complex and heavily rely on the support of information technology for data generation, storage and analysis. Research projects, such as the Large Hadron Collider (LHC) [38] project which has been called "the largest research project in history", have gathered researchers on a global scale to collaborate on a particular research project. These projects not only generate enormous amount of data, often referred to as data deluge [86, 90], but also bring new challenges to data storage and data processing.

To process and store the vast amount of data, researchers need corresponding processing power and software. The data processing procedure can be very complex and often involves multiple steps of operation. These tasks are often completed by workflows, which are composed of multiple independent processing procedures. While each processing procedure can be used individually, they can also be gathered as a group of procedures for a particular task. Theoretically, an individual organization may hold all processing procedures, including the data generation. In many cases, the different procedures in a large, complex scientific workflow are likely to take place at different

research organizations or institutions. Thus, collaboration between different organizations is vital. Quite often, different procedures in a research project can be built into a large workflow and the execution of the workflow can be automated for the researchers.

There are two important and challenging aspects of this collaboration. First, the collaboration is cross-institutional and distributed, which means the programs in the workflow are remotely located and under different administrative domains. To overcome the hurdles brought by distributed applications, Web Service Architecture (WSA) [92] was proposed and has been widely used in distributed systems. Web service is one of the most popular forms of the distributed architecture and it is widely used in cross domain collaborations. The second challenge is the data transfer between different applications in the system. As the basic elements (e.g. instruments, data, computational procedures and data storage capacity) are remotely located in a collaboration system and connected by networks (e.g. Wide Area Network), the data transfer, particularly for large scale data transfer, can be another important issue that affects the efficiency of remote research cooperation.

Improving the data transfer speed within a data oriented collaboration, particularly under the web service architecture, is the primary goal of this research work. To the best of our knowledge, Service Oriented Architecture (SOA) [103] and SOAP-based [113] web services have not commonly been used to serve scientific data, particularly for real-time access to data generated by sensors and scientific instruments such as synchrotrons [144]. One of the concerns is the data transfer efficiency of the web service approach [98, 143]. In our work, we propose innovative approaches to increase the data transfer throughput for scientific research. Our research work involves two major stages. In the first stage, we focus on improving data transfer between two different applications in the system. In this stage, we try to improve the data transfer speed between two directly connected application nodes. In the second stage, we take all applications in the workflow as a whole and analyze the relationship between different applications to improve the overall data transfer performance in the system.

The rest of this chapter will give a brief outline of our research. First, we review the data transfer issues within a web service workflow and point out the necessity of this

research work. Then, the network connections, as the fundamental infrastructure for any distributed systems, are discussed. We also give some information of the network connections used in our work, as these connections will have significant impact on the research outcome. After that, we give an overview of the information about our research approach and outcomes.

1.1 Web Service Workflow for Remote Cooperation

There are different workflow definitions as the word is widely used under many disciplines, such as manufacturing, office and information systems [111]. The workflow activities can either cooperate explicitly or implicitly to complete a task. In an information system, a workflow is activities that involve a group of tasks that cooperate with each of them and are carried out by separate processing entities [134]. The functionality or computational capacities provided by these entities often appear as services. A workflow composed of web services is referred to as web service workflow.

Our research work is carried out in a distributed environment, where different applications in a workflow are remotely located. In this work, we focus on improving the data transfer performance in workflows that are built on web service according to its properties of web services.

Research work often requires processing a large amount of data on a scale from several hundred megabytes to Terabytes or even Petabytes. Researchers need comprehensive information technology support to process data in such scale, including both software and hardware. Some research work involves multiple research institutions where they need to share/transfer data between them. As different organizations are remotely located, the access to software and hardware belonging to different organizations is mainly via the Internet [50]. Further, as they are under different administration, organizations often expose these resources to other users as services. Via these services, users can access both software and hardware provided by an institution. Multiple services can be composed into a new application, which is often called a workflow. When a workflow is invoked, the services composed in the workflow will be invoked individu-

ally and cooperate with each other as specified in the workflow. The final result will be returned after the completion of the workflow.

A workflow is often composed of operations from remotely located organizations. The invocation and data need to be sent between remote institutions via networks that connect the users and the providers. These operations are often exposed to outside users via the web service interface under the Service-Oriented Architecture [103]. Web service is a protocol based computational model, which provides a unified interface for its users.

The invocations to the services are transferred into a wire format, which can be understood by applications/services under different administration domains. This information needs to be sent from the client to the service via the network, and it often takes a significant part of the time for the whole workflow. Furthermore, any service invocation actually involves two different types of information, the information for invoking the operation and the parameters passed in for the operation. The number and total size of messages related to operation invocation is relatively small and steady; on the other hand, the size of the parameters and the results of the services can be quite large. Our interest is to improve the overall performance of a workflow by reducing the data (parameter or result) transfer time, particularly in a remotely connected distributed environment.

Network connections are the underlying infrastructure of distributed systems. Networks are generally classified as Wide Area Network (WAN) and Local Area Network (LAN). In the next section, we will give a general review of distributed environments and a brief introduction to the network connections applied in our network connections.

1.1.1 Wide Area Network and Local Area Network

The development of computer systems is accompanied by the desire of sharing data between different computers. The very early network research work started from Advanced Research Projects Agency Network (ARPANET) [2]. The initial idea is to support communication and information exchange between computers. Today, the largest

network in the world is Internet [50].

Wide Area Network (WAN) connects remote computers. The Internet is the network in which we carry out the research work of web service workflows. The performance of WAN is often affected by two factors: the bandwidth of a remote connection and network latency. Generally, the overall bandwidth of the network is not small. For example, the Australia's Academic and Research Network (AARNet) has a 10 Gigabit/second remote connection to US. However, as the connection is shared by many users, the average bandwidth for each user can be very limited. Latency is the other issue that affects data transfer performance in a WAN. Our experiments show that the network latency between Australia and US is about 110 milli-seconds compared to less than 1 milli-second in a local area network (as in the Adelaide University local network). Larger latency means a network application needs to wait longer before it can send out the next batch of data. More discussions about the impact of latency on data transfer are given in Chapter 4. Local area network provides access to local resources and in most cases with higher effective bandwidth and smaller latency, so it provides better data transfer performance over WAN.

1.1.2 Network Connections for Research Work

Network connections used in our research work involve both LAN and WAN. The network connection is an important factor in our research work, as we focus on running workflows that involve a large amount of data. The network performance between different service providers and clients will significantly affect the performance of the whole workflow.

Remote resources in the WAN are used when the required resources are not available in a local network. If multiple resources are remotely located and there are data exchanges between them, the performance of the workflow will be further affected. However, if they are hosted within a LAN provided by a third party, it can improve the performance of the workflow. For example, instead of providing web services deployed in their own institutions, a group of researchers in the same region set up and

run their services in a regional research centre. In this case, the data transfer speed between different services are typically much higher than the ones between remotely located services.

Cloud – a new distributed computing infrastructure, is actually an emerging type of third party service hosting. *Cloud computing* provides large scale, flexible IT services via Internet. Users of cloud can create multiple cooperative service instances in the cloud to improve performance. Pioneer cloud services providers are Amazon [33], Google [61] and Microsoft [44]. In Chapter 9, we illustrate how we create service instances in a scientific cloud and compare the performance with similar instances in the normal distributed environment.

In this work, we choose to deploy services in the same LAN where it appropriate to get the best performance of the whole workflow.

1.2 Distributed Data Transfer in Workflow

Improving the performance of data transfer in a web service workflow is the main target of this research. From the previous sections, we have figured out that the underlying network infrastructure will bring significant impact on the performance of data transfer. Further, in the context of workflow, performance improvement is not limited to the network level. Within a workflow, data is shared by applications, how to avoid unnecessary data transfer and select the best path for data transfer between services is yet another aspect of our research, which improves the data transfer performance from the workflow level.

In this section, we will first review the requirements for data transfer in a distributed workflow and analyze the data transfer issues within a workflow context.

1.2.1 Data Transfer Requirements

Distributed data transfer is a fundamental function provided by a distributed system. Remote resource sharing and cooperation often involves a large amount of distributed

data transfer and sharing. Within a distributed environment, there are three reasons for data transfer between remote applications:

- ◇ The task is inherently distributed. For example, in the Argo project [34], researchers deploy sensor networks to observe the temperature and salinity of the ocean. The data is collected automatically and reported via satellite. In this case, the data collecting sensors are remotely located from any of its users. The collected data need to be transferred to a data storage centre where it can be accessed and processed later by its users. As these nodes are deployed all over the ocean, a distributed computing model is adopted.

- ◇ The task utilizes remotely located resources. For example, to finish a computational task more quickly, a researcher can utilize the computational power from other organizations.

- ◇ Users from different organizations cooperate to share resources: computational power, software, data, and storage capacity. Researchers around the world often collaborate on research experiments (e.g. CERN Large Hadron Collider project [24, 38]). Large Hadron Collider (LHC) is a large particle accelerator used by physicists to study the smallest known particles. Physicists use the LHC to recreate the conditions just after the Big Bang by colliding the two beams head-on at very high energy [38]. The project will generate enormous amount of data to be stored and analyzed by physicists all over the world. The storage, transfer, sharing and analysis of these data involves many institutions.

When research work involves distributed resources, such as remotely located instruments and distributed processing facilities, it is important to provide an appropriate middleware for collaborative applications. For example, in the synchrotron project [144, 55], a client is expected to download and analyze the data generated from the remote instruments at near real time speed, which could be a challenge for the applications. We use CIMA [81, 94] as the middleware framework to build workflows that support the collaboration between instrument and its clients. CIMA is lightweight. It

provides a general platform and its web service based interface is very intuitive. A potential bottleneck for a CIMA based workflow is that the data transfer speed between data generator and data consumer can be very slow. We need to improve the data transfer speed between these applications, so people can run experiments remotely, access the data and analyze it in workflows in near real-time, which allows them to modify the experiment as it's running based on the results.

1.2.2 Data Transfer in a Workflow

In a distributed system, data transfer means sending data from one computer node directly to another one. In the context of workflow, however, there is more than direct data transfer between computer nodes.

Figure 1.1 shows two web service workflows. Both of them provide the same functionality: the *workflow engine* is the control center of the workflow and there are three web services in the workflow. The input data is processed sequentially by *Web_service_1*, *Web_service*_2, and *Web_service*_3. The result is returned to the workflow engine. There are two data transfer models in the workflow. First, the direct data transfer between two different nodes: data sent from workflow engine to Web_service_1 in both figure *a* and figure *b*; data sent from Web_service_2 to Web_service_3 *(*in figure *b)*. Second, the two workflows have implemented the data sharing between different web services via different paths: in figure *a,* intermediate data is transferred between individual service and the workflow engine; in figure *b,* intermediate data is directly sent to the successive service. The two different approaches have different paths for data transfer, and will therefore have different overall performance. The question is how to coordinate different applications within a workflow, so that the data transfer time can be minimized.

Our research is carried out following two directions. Initially, we focused on improving the direct data transfer performance between a web service and its client. In later work, we proposed Web Service Data Forwarding Framework (WSDF) for more efficient data sharing between services within a web service workflow. As shown in

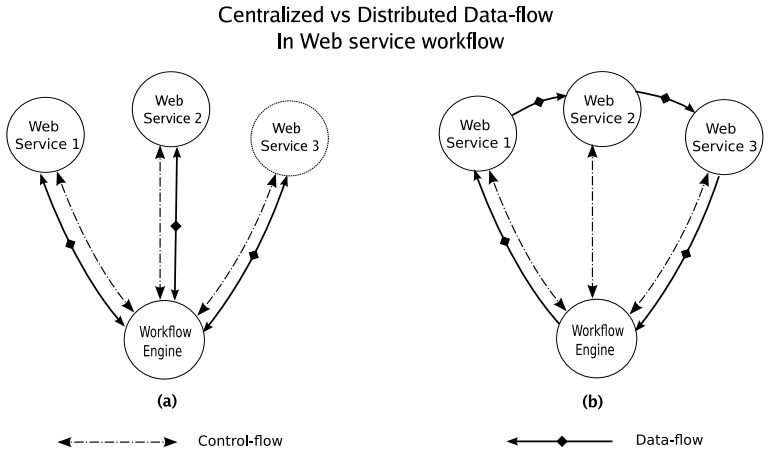


Figure 1.1: Data Flow in Web Service Workflow

Figure 1.1, there are two diagrams. Figure (a) indicates the data flow in a normal web service workflow – each web service sends the intermediate result back to the workflow engine before it is forwarded to the next web service; figure (b) indicates the distributed data flow in the WSDF framework: intermediate result data is directly sent to the next web service without going through the workflow engine, which improves data transfer efficiency.

1.3 Distributed Data Transfer and SOAP

Web service is one of the popular interfaces for distributed collaboration, as the web service architecture is based on public protocols [113, 103, 97] and is widely used in the distributed computing community. As the applications in a distributed system are remotely located, data transfer and sharing plays a vital role in distributed systems.

1.3.1 Distributed Data Transfer

In a distributed environment, data is often generated and processed at different locations, users need to send these data as efficiently as possible to other users. There are different data transfer tools available that can be used, such as HTTP [9], scp [12, 132], sftp [12], GridFTP [88, 78] and SOAP message with attachments [85]. However, there are different selection criteria that need to be taken into account in deciding which to use.

In a loosely coupled environment, the XML based web service is a good choice for inter-organization communication. Web service uses Simple Object Access Protocol (SOAP), the *de facto* format in web service for message exchange. XML [6] encoded SOAP message inherits the advantages brought by XML language, such as platform independent and extendable.

Web service considers independence between entities and standardization of interoperability as priorities in the design of architecture. Data transfer efficiency, on the contrary, is not its primary concern. For example, SOAP is a protocol designed for structured information exchanging [113] and is used by the web service as the primary communication protocol. A web service utilizes XML language to compose a SOAP envelope to transfer the invocation message and parameters between the clients and the server of a web service. However, this approach is not ideal for binary data. Theoretically, the binary data can be converted into characters by using Base64 [119], or other algorithm to convert these data into characters, however, this increases the size of the data set and consumes more bandwidth for data transfer and CPU power for format conversion between characters and binary data.

While the web service interface is ideal for loosely coupled services to exchange messages, it is not good to transfer large binary data via the XML composed SOAP. Actually, when transferring binary data between applications, most protocols (e.g. HTTP, ftp, etc.) have better performance than web service with SOAP. Therefore, finding the right way to improve the binary data transfer between the service participants is one of our research interests.

1.3.2 Data Transfer Performance Improvement with SOAP

In our research work for data transfer performance improvement, we improve data transfer performance in several ways.

1. With high latency network connection, the network transfer throughput can be improved by applying TCP tuning [144]. In a distributed environment, as the data source is often remotely located (e.g. inter continental) from its destination, the

latency can be high between the client and data source. For example, in our experiment, the data sender and the receiver are located remotely, with high latency (about 110 milliseconds) between the client and data source.

TCP tuning is directly related to the TCP protocol [28], in which the client and its server exchange information about their *receive windows* and *send windows* when they are first connected. The size of the windows (or buffers) is the device's buffer size for sending or receiving data. The default buffer size is relatively small for a high bandwidth, high latency network connection, which means that the sender has to wait for the receiver to send response after the former sent out the data. By increasing the size of the buffers, the sender can keep sending out data while the feedback from the client has not been returned. The experiments result has shown that the overall performance has been increased about 35%(for details please refer to Chapter 4).

2. Use *pull* model to replace *push* model.

   With the CIMA interface [81, 94], which is a web service interface, the default data transfer model is push model: on the server side, instrument generated data is sent to the client. This has limitations that the whole communication between the client and the server is controlled by the server and the client has no chance to initialize the data transfer (more details are given in Chapter 4).

   The web service interface on the server side is not very efficient for binary data transfer. The data transferred from the server to the client is limited by the capacity of network connection between the server and the client. The client does not have the flexibility to select when and how to download the data. We propose the *pull* [77] model in retrieving data from the server side.

   In this model, the generated data on the server side is not sent to the client directly, instead it is accessed by using other communication protocol, such as HTTP. Once the data is generated, it is saved on a local file system and served by a HTTP server. The URLs pointing to these data are passed to the CIMA server. Meanwhile, web service interface of CIMA still sends messages to its clients. However, it only

sends meta-data information, i.e. the HTTP links to these data.

There are multiple advantages in the pull model compared to the push model. First, comparing with sending the binary data directly to the client in the push model, in the pull model, the data can be saved on hard disk and retrieved by the client when it is needed. Second, instead of using SOAP messages to transfer these data, clients can use other communication protocols (e.g. HTTP) to transfer these data in a more efficient way. Finally, in a pull model, the client can get multiple URLs pointing to the available data resources. Based on this information, clients can apply a multiple threaded data connection to download these data according to capability, while not interrupting the communication between the web service and the client, as we show in the following section.

3. Apply concurrent data downloading techniques in data transfer. While the pull model can give the flexibility to the client to decide when and how to download the data, the throughput of the data transfer can be further improved by applying multi-threaded concurrent downloading. Computers with multi-core CPU and larger memory allow us to efficiently built multiple data connections between the data sender and its receiver, so the data downloading can be done concurrently. This allows the data downloading procedure to maximize the usage of the bandwidth of the connection between the sender and the receiver.

   Our work has shown that this is a very useful approach, according to our experiments, the performance can be improved up to two to three times. The other advantage of using multi-threaded downloading is that compared to the TCP tuning approach, which requires root access to change the size of the buffer, multi-threaded downloading is an easier approach, as it does not need root access.

In our research work, we also combined the above approaches to improve the overall performance improvement. According to our experiments, the data transfer performance can be improved by more than ten times (see Chapter 4 for details) compared to the standard CIMA web service interface for binary data transfer.

1.4 Web Service Data Transfer with Attachment

While our research work outlined in the previous section brings significant benefit to the overall performance of the data transfer between the client and the data source, it also makes the system more complicated. In particular, it introduces an extra HTTP server into the web service based CIMA system. This brings more complexity to the whole system, which means the service provider has to maintain a separate HTTP system while maintaining the web service system. The client also needs to understand two different systems to write the corresponding code for each of them.

An approach is to use the Web Service with Attachment (WS-Att) [20] mechanism to send large binary data from one point to the other.

When applying WS-Att with the pull model, the CIMA interface sends the location of the generated data back to the user. Users then invoke the web service interface to download the attachments. With this approach, the service provider only needs to maintain one system – web service system. The client can also establish multiple threads to connect to the service provider and concurrently download data from the service provider. Finally, according to our experiments, the data transfer performance of WS-Att is very similar to HTTP.

1.4.1 SOAP with Attachment

Web services exchange information by sending XML based SOAP messages. However, some contents are simply too large to be sent in this way. The web service community has provided the web service attachment specification [20] to send these content as an attachment to its related SOAP envelope.

There are different formats of WS-Att. SOAP Messages with Attachments (SwA) is the earliest version of WS-Att, which combines the MIME multipart/related message with the SOAP message [85]. With the SOAP message, references are defined to point to the binary part of the SOAP message. With SwA, the attached binary data is separated from the XML infoset. XML binary Optimization Package (XOP) [115] was released to unify both the binary data and the SOAP message into the XML infoset. The

SOAP Message Transmission Optimization Mechanism (MTOM) is the implementation of XOP within the SOAP 1.2 specification [113].

Our research tests APIs of different implementations (XFire [27] and Axis2 [23]) of SwA and MTOM . We also test both LAN and WAN. The WAN tests are carried out on both continental (a few thousand kilometres) and inter-continental (tens of thousands of kilometres) scale. We also use multiple threads to improve the throughput with different APIs. Further, we conduct GridFTP [6] tests between different locations. Our experiments show that, by applying web service with attachment, the data transferring performance can be quite competitive compared to the performance of GridFTP, and sufficient for the purpose of transferring files from our synchrotron experiment to clients.

1.4.2 Web Service Attachment Performance

The primary target of our WS-Att performance testing is to figure out what kind of performance improvement can be achieved by using web service with attachment to transfer binary data.

First, we compare the performance difference between SOAP and WS-Att for binary data transfer. Since the binary data will be sent in the binary format with WS-Att, we expect its performance will be better than the SOAP approach that converts the binary data to text. Further, as there are multiple WS-Att specifications available, we also compare the performance between different specifications. In our work, the comparison is based on the Java implementations of these specifications (see Chapter 5 for details). Finally, we compare the WS-Att with other binary data transfer protocols to validate its performance as a general binary data transfer method. We selected GridFTP as the protocol to compare with. GridFTP [88, 78, 95, 120] is an extension of the ftp protocol [19] proposed by Open Grid Forum [11]. It is designed to provide secure, robust, fast and efficient transfer of large data set that can be used in normal distributed environments [63]. As GridFTP has been widely used for fast transfer of large scientific binary data sets, we try to figure out the performance between different implementations of WS-Att and

GridFTP (e.g. GridFTP implementation in Globus Toolkit [105]). The details and result of the comparison are shown in Chapter 5. When carrying out the experiments, we also considered the possible impact brought by other elements in the experiments. We use multiple threads for web service attachment transfer. Different data sets have been used to verify the impact of file size variation. According to the distances between the data sender and the receiver, the experiments are also classified as local network, where the latency is the smallest, intra-continental network and inter-continental network, which has the highest latency.

Up to now, we have described research works carried out to improve the data transfer performance between two nodes under the web service architecture. As we pointed out in section 1.2.2, there are actually two data transfer issues between cooperating applications within a distributed environment: sending data directly from one application to the other and sharing the information between these applications. In the following section, we will give the overview and motivation of our work for the second type of data exchange: data sharing between different applications within a workflow.

1.5 Data Sharing in Web Service Workflow

1.5.1 Workflow Classification

Workflow systems can be classified into two categories according to the location of the workflow engine, from where the control messages are sent – centralized workflow management systems (e.g. Kepler, Taverna and Triana [26, 123, 127, 52]) and decentralized workflow management systems [141]. A centralized workflow management system can monitor and interrupt each step directly from a single workflow engine. With the decentralized workflow, both data and control are distributed on different services, so that workflow functions are fulfilled through the direct communication and coordination among the relevant services [141].

1.5.2 Workflow Data Sharing

Within a distributed workflow, there are multiple cooperating applications. If a result from one of these applications is used by a different application in the same workflow, we define this scenario as *data sharing* in the workflow. When different applications in a workflow share a large amount of data, the overall performance of the workflow will be affected by this data sharing. To reduce the impact of data sharing and improve the workflow efficiency, we need to either reduce the transfer time between any two given nodes (as we have shown in the previous sections), or reschedule and optimize the data transfer path (or both), so that the data transfer between atomic services can be minimized. For the later approach, our research focuses on the collaboration between automated applications.

Before we take any further steps, we need to have a look at the distributed computing context. In such a context, applications/services in the workflow need some communication protocol which will best fit the distributed environment. Web service is a good interface to be used for this purpose and it is the main focus of our study. We try to transfer the data via the most efficient path between the web services in the workflow without sacrificing the functionality of the whole workflow. By efficient, we mean the new approach will only introduce least amount of requirement for extra management and programming work.

Another important issue raised by distributed workflow is data sharing between legacy applications in a workflow. Most of these legacy applications don't have a unified interface, the data sharing between them can cause difficulties for workflow builders. *GriddLeS* [8] is a software package which provides an extra layer in the system in charge of coordinating data transfer details from applications. One of its motivations is to utilize legacy programs without changing much code, particularly in a workflow environment. By using *GriddLeS*, all data transfers in a workflow between different components are viewed and treated as conventional file I/O operations. GriddLeS is a library that provides a rich set of interprocess communication facilities. A user can use this library to implement data sharing between workflow components in a Grid environment. Grid-

dLeS introduces the *GridFiles* mechanism (an abstraction of local files, GridFTP server and GridBuffer) for interprocess communication between components in a workflow. When a component in the workflow invokes read/write operation, the GriddLeS system will first intercept this operation and hand it over to the File Multiplexer (FM). A FM contains multiple clients – *Remote File Client, Local File Client, Grid Buffer Client, GNS client and GRS Client,* it will pick a corresponding client (according to the system configuration) to delegate the read/write operation for that component. On the other end of the communication pipeline, another component of the workflow will also use FM and invoke the corresponding client for data exchanging. We will not pursue this aspect further in this thesis, where we focus on web service based approaches.

A service provider provides services for its clients. The relationship between a service and its clients is called client-server model. The communication only happens between two entities. This model reflects the relationship between the client and the service provider: the service responds to the request from its clients. When a service, e.g. a web service, is involved in a workflow and the generated result from the current service is used by a consecutive service in the workflow, the entities involved in a client request have changed: there are three applications in total rather than two of them. In a centralized workflow system (e.g. a Kepler based workflow), the result is first sent back to its client before being forwarded to the next service. In a decentralized workflow model, the data can be forwarded from one service to another directly, however, since decentralized workflows have other problems (as we discuss in Chapter 2), centralized workflow systems are much more commonly used. According to these reasons, our work focuses on improving the data transfer efficiency of web service workflow which has a centralized workflow engine.

In a web service workflow, ideally the data is forwarded from the current service to the next service that is going to use the data as its input data. In the current approach, the data needs to be sent to the client, which is often a workflow engine, and then forwarded to next service. This uses more network bandwidth and the client could also be overloaded, especially when multiple workflows are running on the same workflow engine. If data to be shared between different services can be forwarded directly between them,

the data transfer time can be reduced. This decentralized data sharing between adjacent web services in a workflow is another research area in our work.

1.5.3 Web Service Workflow Data Sharing Review

Web service is based on service oriented architecture. In a distributed environment, service oriented architecture allows individual applications to be independent from other applications, i.e., do not need to know the implementation details of other services. At the same time, a service, e.g. a web service, can register and publish information about itself and search/understand the interface information of other web services.

Individual web services work under the client-server model: the client sends a service request, including parameters, to the server, the server processes the request and returns the result to the client. However, in a workflow, as services often work on a set of data in a pipeline style, the relationship is not limited to the client-server model, but also requires exchanging data between different services.

There has been some research work looking at how the data can be transferred to the next participant in the workflow without going via the client. However, these implementations all have some limitations.

For example, the *proxy* model is suggested in [84] and [89]. Also, in [83], a hybrid architecture is built. A *proxy* is a piece of middleware *closely coupled* to a functional service as a gateway. It has three major functions: delegating the invocation of the functional services; managing input/output data storage and sending the result data between workflow components. A set of Application Programming Interfaces (APIs) are given to interact with the proxy service. This research work has pointed out some research targets of data sharing problem between web services in a centralized workflow, such as result data storage, forward and retrieving. The APIs given in the paper are Java language specific and do not fit web services written in the sense of programming language neutrality. A similar but different model is introduced in [89]; in this paper, the *trigger* also acts as a proxy to delegate the invocation of the web service and result data forwarding to its consumer. By sitting closely to the web service, the *trigger* acts as a

buffer of input parameters and waits until all parameters arrive, then it will trigger the corresponding service. After getting the output of the service, the results are sent only to where they are needed by the *trigger*. In this paper, no standard description is provided for the *trigger* service and we expect different implementations will not be compatible.

The general drawback of the proxy models is that they address the data sharing problem from an application level. Either the trigger or the proxy service is defined as normal function service, so the programmer needs to maintain these services for themselves. This approach will be ad hoc and error prone.

Our research target is to build an architecture, in which the model of web service can better fit the requirement for data transfer in workflows. In particular, the new model should provide data sharing functionality between services at server level and hide the details of data sharing from users. The basic requirement is to support third party data transfer, i.e. the client can invoke a specific service (*current service*) and guide the service to forward the result to a different service (*successor service*) that is going to use the result data as its input [1].

In the proxy model [83], the burden of implementing data transfer is on the programmers. We propose a new framework to free programmers from these details. In this new framework, mechanisms are built to achieve the following:

1. Provide a *data forwarding* mechanism in the web service server level. Previous web service is built on bidirectional model. In a centralized web service workflow, the data is sent from one service to another while under the control of the client. This data forwarding mechanism should be a basic functionality provided by the underlying web service server. The data forwarding should be carried out between the data generator and consumer directly following the client's instruction.

2. The forwarded data should be saved on the *successor service* as a resource. After the data has been saved on the successor service, the client (workflow engine) still decides when and how to invoke the operation of the service which uses the saved data.

[1] Alternatively, the result can be saved on the local machine as a resource and only retrieved and transferred to the successor service when it is needed, which is a *lazy* approach.

1.5.4 Web Service Data Forwarding (WSDF) Framework

WSDF is our proposed new framework for efficient data sharing between different applications in a workflow.

One of the important ideas of WSDF is to keep the intermediate result of a workflow on the server side. We introduce the stateful workflow concept and the WSDF framework is based on this concept.

In a *stateful* workflow, each atomic service is *stateful*. The de facto standard for representing the state of web service is Web Service Resource Framework (WSRF) [100], which provides a framework such that a compliant web service is stateful and the state information of a particular web service instance is a resource. A stateful workflow keeps the state of the intermediate results of web services in the workflow and shares them within the workflow. What we want is to have one data result generated from the current service to be transported to and saved on the successor service, which is stateful.

We propose the WSDF framework, which is based on the idea of stateful workflow, to allow efficient data sharing between services. In this framework, atomic services involved within a composite service are WSRF services. A WSDF server, built on WSRF server, hosts atomic services and is responsible for forwarding the result data of the current service to the successor service. The information used by the WSDF server hosting the current service to transfer result data is called *resource forwarding information*. A *resource forwarding information* schema is also defined (see Chapter 6 for details). If a client invokes the current service while embedding this resource forwarding information in the invocation request, the server first retrieves this information from the invocation request; after the functional service is finished, it forwards the result to the successor service as specified by the resource forwarding information. The successor service accepts data sent and stores it as a resource before the invocation of this service.

1.5.5 WSDF Implementation

Based on these framework principles, we built a complete prototype system to prove the proposed concepts, see Chapter 6 for details. The prototype contains WSDF server,

WSDF service and service client.

According to the definition of the WSDF framework, a WSDF server is first of all a stateful web service server. The most popular implementation of a stateful web service server is WS-core [105], as part of the *Globus 4* software [29], which is built on Axis1.1 [3]– an open source SOAP engine. Based on the source code obtained from WS-core project, we rebuilt the SOAP engine for WSDF service. The updated SOAP engine is called WSDF-axis. Two extra functions have been implemented within the WSDF server: first, when the server receives an request, it will check the request message and extract *resource forwarding information* and store them temporarily; second, after the functional service is finished, the WSDF server can forward the intermediate result according to the resource forwarding information. If there is no resource forwarding information in the request, the WSDF server will return the result back to the client as a normal web service server.

Computational web services often need to transfer binary data between different applications. The binary data can either be transferred by applying encoding schemes such as *Base64* [119] or by using web service with attachment [85]. Our implementation supports web service with attachment, as it will save both network resources as well as computational resources and is much faster than using encoding schemes such as Base64.

We implemented some example WSDF workflows (including both services and their client) and measured their performance with normal web service workflows with the same functionality.

1.5.6 WSDF Performance Testing

Based on the WSDF prototype system, we carry out performance tests in a simulated environment to compare data transfer speed of WSDF workflow with the normal web service workflow.

To avoid the impact brought by the changing performance of the Internet, we built a WAN emulating system with the WANem [43] software (for details, refer to 8.1.2.1).

With this emulating system, we tested the WSDF prototype with both inter-continental and intra-continental network conditions. We also tested the local network performance of WSDF within our Ethernet LAN.

In comparing the performance of normal web service workflows in the same testing conditions, the WSDF workflows have shown significant advantage for data transfer efficiency. In particular, the result shows significant improvement has been achieved by WSDF workflows in long distance data transfers (see Chapter 8 for detailed information).

The other distributed environment in which we carried out WSDF testing is the cloud environment - an emerging distributed infrastructure.

1.5.7 WSDF in the Cloud

Cloud is an emerging infrastructure for distributed computing. Cloud service providers provide IT services for their customers via network connection. Cloud services are flexible and are often provided in large scale. Cloud providers also provide data storage capacity for its users. The cloud provides an easy and flexible environment, where users can get resources according to demand: not only for hardware resources, but also software as resources. Cloud services are usually classified into three different types: Infrastructure as a Service (IaaS), Software as a Service (SaaS), and Platform as a Service (PaaS) [116]. We are expecting the cloud environment to play an important role in hosting services, particularly for WSDF services.

We believe the cloud environment is especially good for WSDF workflow: servers can be closely located in the cloud compared to when they are located in a normal distributed environment. An ideal environment for WSDF framework is that the services are located next to each other in a local area network, in order to get high bandwidth and low the latency connection between the services, while the client is far from these services. The cloud can provide an environment exactly matching this requirement. For example, collaborators from different organizations provide different services of a workflow, they can embed the separate service into separate operating system images

and submit to the cloud. Before the workflow that involves these services is invoked, the instances of these services should be activated in a virtual local network in the cloud.

If these instances are hosted by individual organizations, hardware and administration people have to be assigned to manage these resources. In the cloud, these are managed by the cloud system, users don't need to worry about these administration issues. They can start and stop their instances without notifying the administrators; built in necessary software in the image; and set the network connections according to their current requirement.

We carried out the performance tests in the *ScienceCloud* [40] for both WSDF prototype workflow and web service workflow. The performance improvement is compared to the improvement the same WSDF workflow achieved within a normal distributed environment. The result has shown that the cloud infrastructure is a better platform for WSDF workflows. See Chapter 9 for more details.

1.6 Thesis Content Outline

The rest of the thesis is composed of nine chapters. In Chapter 2, we give a broad review of the background and related work of our research. Chapter 3 gives more details on distributed workflows used in eScience research. Chapter 4 discusses how to improve data transfer in web service with SOAP messages, including the details of TCP tuning, push/pull model and multi-threaded concurrent web server transfer. Chapter 5 reports the data transfer performance improvement by applying web service with attachments. In Chapter 6, the concepts of web service data forwarding (WSDF) framework is proposed. Chapter 7 reports the implementation details of how we built a WSDF system prototype. Chapter 8 presents WSDF performance test results under the simulated testing environment and analysis of these results. Chapter 9 reports the WSDF testing results within the cloud environment. Finally, in Chapter 10, we conclude our work and suggest future research targets.

Chapter 2

Research Background

Our research work is to improve the data transfer performance for scientific collaborative work in a distributed environment using web services. In this chapter, we will discuss general background information related to this work.

2.1 Introduction

Distributed computing is an important computational model built on distributed systems, which is composed of multiple autonomous computers connected via networks.

The primary goal to build computer networks is to set up a new communication channel and share information. Nowadays, distributed systems can provide much more resources than just information. Users can utilize distributed resources (e.g. memory, CPU, data storage capacity) via a distributed computing model. For example, Amazon S3 [32] storage service provides data storage capacity for its clients via a standard web service interface. In this case, a user is not retrieving information (e.g. HTML pages) from a remote server, but harnesses the remote data storage capacity. Similarly, a client can invoke a remote web service by sending a remote request. When the response is sent back, the client has actually utilized the remote server's hardware, such as memory and CPU, as well as software. In this thesis, we use *resource* to represent software, hardware and other facilities provided by a computer node in a distributed system that benefits the remote user.

In the rest of this chapter, we will discuss the distributed infrastructures and the programming models used in our research work. We first review the networks that connect individual computers, then discuss the computational models for distributed computing, such as Remote Procedure Call (RPC) and Web services. We then discuss the higher level distributed infrastructures, such as the *Grid* and the *Cloud*. Finally, we discuss web services related techniques in more detail.

2.2 Distributed Environment Networks

In a distributed environment, individual computers are connected by computer networks. Without these networks, individual computers will not be able to share any information, nor any resource. The characteristics of the networks have significant impact on the development of the distributed system and will affect the performance of the system.

To build a distributed system with computers, first of all, we need these computer nodes to be connected, so the computers can communicate with each other. The network connection not only means lower level connections, i.e. physical connection, but also higher level connections, i.e. different computers can exchange messages by applying to common protocols (e.g. TCP/IP, Ethernet – IEEE 802.3).

According to their geographical perspective, networks are categorized as Wide Area Network (WAN) and Local Area Network (LAN) (as discussed in 1.1.1). Compared to a WAN connection, a LAN connection has less round trip time (RTT), and for the same bandwidth, it will be more efficient to transfer a large amount of data over a LAN. If the same resource can be found in a LAN environment, to improve the performance, we should try to avoid using the same resource from a WAN connection. We investigated data transfer over both LAN and WAN in our studies in this thesis.

A distributed computer system is composed of computer nodes and networks. The computational nodes can also affect the performance of the distributed system. For example, each computer node has its own operating system, the requirements for building a computer system in which the computer nodes use different operating systems will be different from those where the nodes have the same operating system. A distributed

system under a single administration domain will also be different from a distributed system that is under different domains of management.

2.3 Distributed Resource Sharing

Once a distributed system is connected by using networks, applications in the system can share resources between the involved computers. Resource sharing is based on the basic message passing mechanism provided by the underlying network system.

2.3.1 Information Sharing

Information (including data) sharing is built on the top of basic network connection and it is the basic format of distributed resource sharing. Web pages, files and raw data can be exchanged via the network. The network protocols involved here are basically communication protocols, such as TCP/IP [28] and HTTP [9]. FTP protocol and its implementations [39, 19] provide an efficient way for high performance data transfer.

2.3.2 Computational Resource Sharing in Distributed Environment

With the maturing of computer networks, distributed systems have been developed to utilize the computational power on remote computers. Distributed computing models started by following the procedure invocation as it is on a single machine.

On a single computer, a program procedure (the caller) can invoke another procedure (the callee) on the same machine. After the callee finishes the execution, the result is returned to the caller. If the caller and the callee are on different computers, the invocation can utilize computational power from a different machine and the overall computational power of the application can be significantly improved. For example, local area network is also used to build clusters to share the computational power between computers in the same cluster. In a WAN environment, people can build Internet scale distributed computing applications, e.g., large scale web service workflow, to utilize the distributed resources.

Built on the information sharing level, different programming models have been developed to utilize distributed resources. *Remote procedure call* (RPC) [136] simulates the procedure call syntax used with local applications, as if the remote procedure is a local one. RPC is procedure oriented, and is the first programming model designed for distributed computing. Similarly, Remote Method Invocation (RMI) supports object oriented remote method invocation.

With RPC (as well as RMI), the underlying message transport mechanism has been tightly coupled with the higher level method invocation, which introduces inflexibility to the whole system. Further, RPC is language dependent, which means the client and server have to use the same language to built a distributed application. This significantly narrows the interoperability between applications.

Distributed computing is carried out within a heterogeneous environment, where applications developed with different languages running on different operating systems coexist. To solve these problems, standard abstract programming interfaces are defined to ensure the interoperability between applications. CORBA [4, 135] is an architecture that is designed to allow programs written in different languages to interact with each other by following the common interface defined by CORBA. However, the implementation of CORBA was initially programming language specific. CORBA was later extended to support interoperation between different vendors. It also has many characteristics of a Service Oriented Architecture (SOA), such as naming service. So it is also considered as a service oriented architecture. Different from web service, CORBA defines its own standards and protocols, whereas web service techniques, such as SOAP and WSDL, are built on existing standards and protocols (e.g. HTTP, XML) and are more popular. With CORBA, one needs to use interface definition language (IDL) to define a standard interface and then use language specific compiler to generate code. This procedure heavily relies on the compiler and is more complicated comparing to building a web service by using XML language.

Web service [103] is another distributed programming model which is built on service oriented architecture (see section 2.5.1 for details). Web services are based on publicly defined protocols (e.g. SOAP, WSDL and UDDI). These approaches ensure

that web services are programming language and platform independent.

Many programming languages support the programming models we discussed here. Some programming languages are designed to be a distributed programming language – such as Erlang [60]. General purpose languages, e.g, C/C++ and Java, often provide specific libraries (e.g. Java RMI) to support network connection and therefore distributed computing.

Finally, remote resource cooperation has been supported by other related techniques. With the development of broadband network connections, particularly with higher bandwidth (e.g. the Australian Academic and Research Network (AARNet) [1] has 10 Gigabits direct data connection between Australia and the U.S.), and more stable Internet connection, it is possible to have the resource sharing extend to an even wider scope. New system level applications such as Google File System (GFS) [112] and Hadoop Distributed File System (HDFS) [93] are technologies that are designed to utilize the distributed resource within an even larger scale, by using commodity hardware. Also, development of distributed computing infrastructures, such as the *Grid* and the *Cloud* also provide better distributed environment for distributed computing.

2.4 Distributed Computing Infrastructures

The traditional environment for distributed computing refers to any computers connected by networks. While these networks of computers are the foundation for distributed computing, they need extra functionality to support distributed computing. In this section, we are going to review two distributed computing infrastructures: the *Cloud* and the *Grid.*

With the development of distributed computing, the trend for research work in this area is to build more sophisticated infrastructures that are particularly designed to improve distributed resource sharing. For example, the Grid [106] and the Cloud [35] are important infrastructures that have been developed based on the previous outcomes of distributed computing research.

Traditional distributed systems can support remote service invocation, but only lim-

ited to existing service instances. In most circumstances, a cloud service provider can dynamically create a desired resource instance for a user [110]. In a grid, users search to find an existing instance within a grid. The more fundamental difference between a cloud and a grid is: a cloud is constructed and managed by a single organization and it provides a unique interface for all users to interact with this cloud; on the other hand, in a Grid, as different organizations join the Virtual Organization (VO) to share their resources, a more generic infrastructure is built to accommodate heterogeneous resources belongs to individual management organizations. For example, in the traditional distributed environment, such as the Internet environment, there is no general authorization and authentication mechanism for users to access resources belonging to different organizations. In a cloud, the cloud provider defines its own authorization and authentication mechanism for all users to interact with this provider, but not between users of the cloud or between different clouds. Within a *Grid*, basic authorization and authentication mechanisms are provided for any participants to interact with each other, which is a key characteristic of the Grid [11, 106].

2.4.1 Grid Infrastructure

In [106], Ian Foster points out that "the real and specific problem that underlies the Grid concept is coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations". Grid computing addresses the requirement of the utilization of resources from different organizations by defining the concept of Virtual Organization (VO). A virtual organization is composed of a group of institutions to share resources between themselves according to a set of predefined rules. The main target of the *Grid* is coordinated resource sharing and problem solving based on grid protocols. These standard protocols, interfaces and middleware are vital to enable heterogeneous resources across multiple organizations to appear as uniform resources to applications. The Open Grid Forum (OGF) [11] is an open organization for defining these protocols.

These protocols define different aspects of the grid infrastructure:

◇ Architecture. For example, the *Open Grid Services Architecture (OGSA)* and

OGSA Profile Definition are protocols that define the fundamental architecture for the Grid infrastructure. The OGSA defines how the whole infrastructure is layered and elements within each layer.

◇ Security. Security is an important aspect in Grid infrastructure. OGF has defined multiple protocols in this area, e.g. the *Global Grid Forum Certificate Policy Model [46]* and *Grid Security Infrastructure Message Specification*[47]. The certificate policy (*CP*) was developed for the Open Grid community to reduce the cost and time needed to build a Grid public key infrastructure (PKI) and increase policy and technical interoperability in the Grid community. It is a set of rules that indicates the applicability of a certificate issued by conforming certification authority (CA) to its community of users and/or class of application with common security requirements. These security protocols provide the possibility for applications in a grid to interact with each other in a secure way.

◇ Data. Grid provides a platform for efficient data sharing and transfer between applications. The GridFTP [63, 88] protocol defines an improved data transfer protocol in the Grid; the Storage Resource Manager Interface [70] defines an interface to deal with multiple heterogeneous storage and file systems for managing, replicating and accessing files in distributed environments; the Conceptual Grid Authorization Framework and Classification [58] introduces the basic concepts and models of authorization by giving a conceptual grid authorization framework and proposed authorization mechanisms within the framework.

◇ Compute. Grid provides mechanisms for applications to share computational resources. *Job Submission Description Language (JSDL) Specification* [76] specifies the semantics and structure of an XML language that is used to describe the requirements of computational jobs for submission to resources, particularly in Grid environments. *Web Services Agreement Specification* is a web services protocol for establishing agreement between two applications, such as between a service provider and consumer, using an extensible XML language.

◇ Others. There are some protocols focus on management and applications.

2.4.2 Cloud Infrastructure

Cloud computing is another emerging distributed computing infrastructure. There are many different definitions [35, 125, 139] for cloud computing. The common agreement about cloud computing is that the cloud provides IT related resources, including hardware, software and network, as services over the Internet [37, 79].

Cloud computing provides services that are classified into three main categories [125, 139]: *Infrastructure as a Services* (IaaS), *Platform as a Service* (PaaS) and *Software as a Service* (SaaS).

1. IaaS providers create *Virtual Machine* (VM) based computational instances (including the corresponding operating system and software) as well as storage capability. Users send requests to IaaS service provider, often via a web service interface. In response, IaaS provider initializes virtual machine instances by using images specified by the client and allocates network resources for them. IP addresses can be either pre-allocated or dynamically located by service providers. A client will then be able to control and run applications on the provided instances.

2. A PaaS provider creates and maintains a platform for users to build their own applications and run in the cloud infrastructure maintained by the cloud provider. A PaaS is tightly coupled with a particular cloud. Normally, a cloud provider specifies the programming languages to build applications that are supported in the cloud. For example, *Google App Engine* – a cloud service provided by Google, supports Java and Python [61] based applications. Programmers write programs with the supported languages and convert them into *APP engine* compatible applications by using specific libraries provided by Google. The applications are then submitted to the *APP Engine* platform to be hosted. Users of these applications can access them over the network connection. However, these applications, in most cases, can only be run with a particular PaaS provider, which is not flexible. The advantage of PaaS is the owner of the application can specify the detailed

environment in which the application is run without worrying about how it is implemented. For example, the number of CPU cores and memory size are located to each of the application instance can be set by application owner and the cloud provider will provide corresponding hosting environment.

3. SaaS represents cloud applications that are developed and run by the service providers as on-line services. A representative provider of SaaS is *Salesforce* [67], a company that provides online software services (e.g. Customer Relationship Management–CRM) for its clients. Google *Document* [62] is another SaaS service that provides online, free word processing functionality.

The cloud environment provides not only computational power, but also data storage and file management systems. Distributed file systems, such as Google File System (GFS) [112] and Hadoop Distributed File System (HDFS) [93] are standard back end file manage software in most cloud infrastructures. This helps to store the users' data in the cloud and can be used in the applications that are developed for the cloud. When a large amount of data needs to be processed within the cloud, these data will not need to be transferred into the cloud again which saves data transfer time and network bandwidth.

Both the *Grid* and the *Cloud* aim to improve the environment for resources sharing in a distributed environment. In *Grid*, the users can find the desired resources within the VO. The *Cloud* will create instance of the resources according to the user's requirement. Clouds currently do not have standard protocols and interfaces for interoperability between them, where Grid has. If different Cloud providers are VO members in a *Grid,* then one cloud provider can interact with the other one for extra resources (e.g. virtual machine instances) via the standard Grid interface.

Both *Grid* and *Cloud* are service-oriented architectures and have made use of web services since they are a standard approach to SOA using standard web protocols.

2.5 Service Oriented Architecture and Web Service

Computer nodes, as well as other resources (e.g. distributed sensors), are managed by different organizations and connected in a distributed system. A common problem with distributed computing resources sharing is that the clients need to know what resources are provided by the remote providers. Under the traditional programming models, such as RPC, there is no standard mechanism for clients to get information about the remote procedures. The service oriented architecture is designed to separate the interface definition from its implementation and web service is the most important implementation of this architecture.

2.5.1 Service Oriented Architecture (SOA)

Service oriented architecture is designed to reflect the relationships in a distributed environment, where applications (*services* in the context of SOA) are built on different operating systems by using different programming languages and are managed by separate organizations. There are three components in a service oriented architecture, the service provider, a service information provider (service broker) and service consumer, as shown in Figure 2.1 [68]. The service provider develops a service, which is accessible for other users, and publishes the description of the service in a repository service (e.g. UDDI, in the case of web service) [103, 16]. Users first query the *service broker* to find a particular service, then it contacts the service provider for further detailed information, finally, users can use these information to build applications and interact with the service on the fly.

Service-oriented computing (SOC) is a computing model based on the service-oriented architecture. It uses autonomous, platform-independent services as fundamental elements to support loosely coupled, configurable, interoperable and distributed applications [129]. SOC also utilizes services to support fast, easy, and low-cost composition of applications. Distributed computing always faces the complexities and difficulties brought by the heterogeneity of a distributed system. Service oriented computing abstracts the computational resources as *services* without worrying about the programming

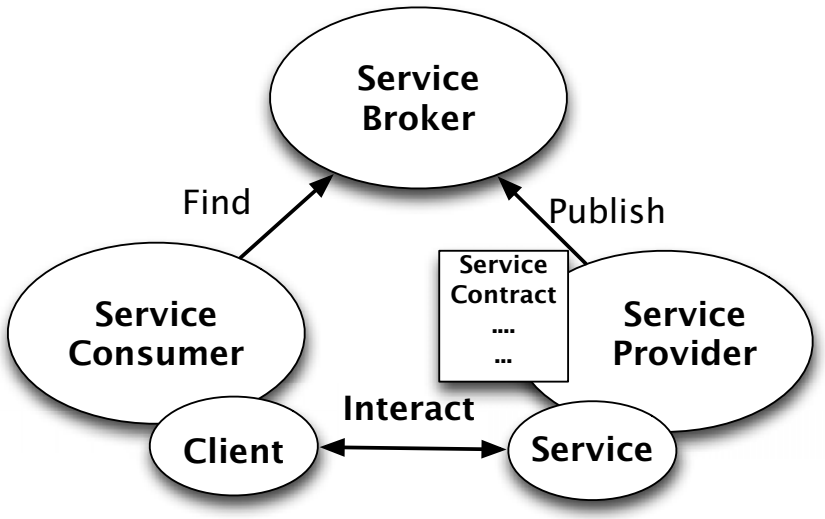


Figure 2.1: Service Oriented Architecture

languages, the platforms on which they are running and implementation details of these services. The interface of a service, which is vital for its users, is published and available for public query. Web services is currently the most promising technology for this model. Within the web service architecture [92], a service provides computational services to its users through a request/response interface; a web service registry repository (UDDI) provides service information for user enquiry. A client invokes a specific service by sending messages (typically, SOAP messages) to the service instance.

2.5.2 Web Services

Resources in a distributed system are located within different organizations and platforms and loosely coupled with each other. Service oriented architecture provides a platform to support loosely coupled services. Web service architecture is built on service oriented architecture to support programming language neutral, platform independent and location transparent services [130].

In this section, we will review the web service related techniques, particularly the protocols often used in the web service architecture: Simple Object Access Protocol (SOAP) [113] and Web Service Description Language (WSDL)[97, 96].

2.5.2.1 Web Service Architecture

Web service architecture inherits the basic components defined by service oriented architecture. In web service architecture, there are three major parts within the whole architecture. First, a web service is the provider of a specific service; second, a web service consumer – a client, interacts with the service; third, a service *web service broker* from where a client can search and get the basic information for a particular service. Typically, the information broker hosts a Universal Description, Discovery and Integration (UDDI) repository [16], another web service to provide this functionality. The UDDI provides a way of providing a storage, discovery and advertisement service to the web service information.

Detailed information of a specific web service is exposed by the web service provider, formatted by using a standard description language (typically the XML-based WSDL, see section 2.5.2.2 for details). In most cases, a web service client first checks the service information provider for basic information (e.g. URL location) of a web service, and then retrieves further information by querying the WSDL information from the web service site.

Another advantage of web service is that it is easier to utilize legacy code. A service can be built to wrap a legacy software with a standard interface without being affected by the details of its implementation. Further, web services can be used with other services to compose a web service workflow. Overall, web services provide a way for dynamically locate, compose and invoke services.

2.5.2.2 Web Service Description Language (WSDL)

Web Service Description Language (WSDL) [97] describes the web service interface in a standard way. In a WSDL description, *ports* refer to a set of service functions implemented by the current service. Each *port type* represents a specific operation, and bindings define the implementation details of communication and transportation. For example, HTTP can be specified as the transport layer for a web service.

The current WSDL language is based on a client/server model, i.e. the communica-

tions are between the client and the server only. Our research work focuses on web service cooperation within a workflow, in which case the data transfer can happen between three different applications. To meet our requirement, we propose to extend WSDL to support the representation of third party data transfer. For details, see Appendix A.

2.5.2.3 Single Object Access Protocol (SOAP)

To exchange structured information between web services, both the client and the service need to agree on a common data structure to represent messages shared between them. SOAP is the most important protocol for this purpose. It is a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment [113]. It harnesses XML technologies to support construction of messages used in distributed computing. SOAP messages are often exchanged over HTTP [97, 96]. Although these are the choices often used in a web service, web services are transport agnostic. As web services provide a very flexible framework, theoretically, almost any combination of communication and transport protocols can be used to replace these two protocols. To specify these information and make it known to users, the service provider only needs to set the corresponding information in the WSDL file.

Even though a web service may use another protocol rather than SOAP for information exchange, there is still a high probability that the communication protocol is based on XML documents, such as XML-RPC [71]. In general, web services are typically designed to exchange XML based documents between service processes using standard communication and representation protocols. This helps the web service to achieve better interoperability. In particular, the Web Services Interoperability (WS-I) organization [18], as part of OASIS [66], is dedicated to promote the best practices for web service standard across platforms, operating systems, and programming languages. The WS-I basic profile [54] standard has been proposed by the WS-I, to establish best practice for web service interoperability [54]. As the WS-I can support high interoperability between different applications in a web service, it will further encourage the use of web services in a distributed system.

While the SOAP based web service can provide various advantages, on the other

hand, it has its own disadvantages. For example, as SOAP is encoded by using XML, it is less efficient for binary data to be transferred between web services applications with SOAP. While it is not a big problem for normal business usage, it can impose a significant burden on data intensive applications, such as scientific computing. In [99], Chiu, et al. analyzed the possible performance degrading of SOAP for high-performance scientific computing. SOAP messages are encoded using XML, which requires that all self-described data be sent as ASCII strings, for example using Base64 encoding [119].

As we have stated, SOAP is one of the communication protocols that can be used to format messages between web service applications. In the following section, we will review how to use the functions provided by the lower level communication protocol (i.e. HTTP) to format messages between web service applications.

2.5.3 RESTful Approach of Web service

In our research work, we focus on the performance of the data transfer based on the SOAP message format. However, not all web services use SOAP for information exchange. The emerging RESTful web services [131, 104] utilize the functions, *GET, PUT, POST,* etc, provided by the HTTP [9] protocol to compose XML based messages.

These messages can provide similar functionality to SOAP messages. At the same time, they have significantly reduced the size and complexity of invoking web services. Particularly for developers who have been familiar with web programming techniques, they do not need to learn a new package of SOAP APIs.

With the SOAP approach, a separate SOAP engine needs to be built to interpret the HTTP messages according to the SOAP specification. With the RESTful approach, the messages passed between applications are interpreted at the HTTP level. When HTTP is the only transport protocol used by all participants in a web service invocation/workflow, this approach will improve the efficiency of message exchanging, particularly for the binary data as part of the message (to covert binary data into a SOAP message often requires more resources). The negative impact of the RESTful approach is that it will limit transport protocol to HTTP, which loses the flexibility of selecting other transport

protocols. Unlike SOAP messages, the RESTful approach is not a specification, i.e. there is no standard format for the syntax of the XML content for each request, therefore, it is not a standardized approach.

2.5.4 Stateful Web Service and WSRF

Web service is based on communication between client and server. A client sends out a request and waits for the response. After the server processes the request, it will send back the result to the client without keeping the result. A basic web service does not maintain the state of a web service instance after the invocation. However, under certain circumstances, the state of a service is desired to be maintained. For example, a client needs to utilize a computational web service by invoking a service operation for multiple times and the previous results are used as the input of the later operations. The intermediate result can be saved on the service provider side, without sending back to the client. When this result is used by the next service invocation, it can be directly retrieved from the service side. This approach is particularly good for the services that can return a large intermediate results. By saving the intermediate result on the service, the whole procedure can avoid the unnecessary data transfer between the client and the service.

Web Service Resource Framework (WSRF) is a framework used to keep the state of a web service. The framework is composed of five separate specifications that provide the normative definition of the framework: WS-ResourceProperties, WS-ResourceLifetime, WS-RenewableReferences, WS-ServiceGroup, and WS-BaseFaults [66]. The *WS-ResourceProperties* document schema, an XML document description, is defined to describe the information or data (e.g. information for a shopping chart) saved on the server side. A resource described in this way is called a WS-Resource [100]. WSRF also defines resource lifetime management interface. *WS-ResourceLifetime* (WSRF-RL) defines the life time of a resource. It defines operations such as *Destroy* (destroy a resource) and *SetTerminationTime* (set termination time of a resource) to manage the life time of resource properties. More information about other specifications of the WSRF

can be found at [66].

For the WSRF specification, there are different implementations, such as WS-core [45](part of the Globus [29] project), Apache Muse [48] and WSRF.net [48, 53].

2.6 Distributed Resources Collaboration

We have reviewed a few basic elements within the distributed environment. Even though these approaches have significantly improved the ability for people to use distributed resources, researchers are still expecting even larger scale resources collaboration for better performance. One strategy is to build web service workflows for dynamic collaboration.

In a distributed system, services are loosely coupled and web service is one of the most promising ones to support distributed computing. A workflow can dynamically integrate different services together into a larger service to provide extra functionality for larger tasks. In the following chapter, we will review the web service workflow and how it is used in e-Science [59].

Chapter 3

E-Science and Web Service Workflow

In the 21st century, the new trend in many research disciplines is to generate, store, process, and access large amount of data by using modern IT technologies. For example, the famous high energy physics project – the CERN Large Hadron Collider [24] - is claimed to be the largest research work in the world. E-Science [109, 72] is an approach that helps researchers to have better access to the distributed resources by using state of the art computing technology. According to John Taylor, Director General of Research Councils Office of Science and Technology, *"E-Science is about global collaboration in key areas of science, and the next generation of infrastructure that will enable it. It will change the dynamic of the way science is undertaken*. With the development of distributed computing techniques, researchers are looking for better ways to integrate different remote resources for even larger scale resource sharing. Workflows have been used as a powerful tool for such coordination. It enhances scientists' ability to carry out their research. Large scale workflow, particularly the web service workflow, is our primary research area. In this chapter, we first define the concept of e-Science and related pioneer projects; then we review the concept of workflow and how workflows are used in e-Science; finally, we discuss the data transfer and sharing in these workflows.

3.1 E-Science

IT technologies were widely applied to scientific research before *e-Science* was proposed. However, the comprehensive application of IT technologies as one of the most important methodologies in research, has triggered the definition of the *e-Science* concept. The term e-Science originated in the UK, to denote the application of modern computing and collaboration technology to scientific research. In Australia, this approach is termed as e-Research, since it is applicable to research in more areas than just science [21].

3.1.1 E-Science Projects

A typical e-Science project often utilizes applications that use a web browser (which can be implemented by using programming techniques, such as JSP [10]) or other GUI interfaces to interact with users. Beside that, two important aspects are often seen within an e-Science project: first, extensive computational power used to process data and second, capacity to store, retrieve and catalog data. Distributed (sometimes high performance) computing infrastructures(e.g. Grid or Cloud) are harnessed to support computational tasks. Data management tools, such as relational databases (MySql, ORACLE and PostgreSQL [51, 14, 13]), distributed file systems (e.g. Hadoop [93, 140]) and other file management tools (e.g. Storage Resource Broker – SRB and iRODS [42, 30]) form another important aspect to support e-Science activities.

Researchers from different disciplines might not be experts in IT, however they often use such state of the art technologies for their research work. They may also compose workflows to put these elements together to create new functions. To support collaboration for a large project between different organizations, a web service interface is widely adopted in this context.

Here we review a few e-Science projects:

◇ In UK, Council for the Central Laboratory of the Research Councils (CCLRC) [36] is a government body that carries out civil research in science and engineering. CCLRC's web services based multidisciplinary data portal uses an XML

metadata [1] model of scientific data to explore and access the content of data resources within CCLRC's main laboratories in UK and other facilities in Europe. CCLRC Scientific Meta-Data Model (CSMD) catalogs the data according to the study the data belongs to. CSMD is an XML format data schema to describe the metadata that can be applied to scientific data which is further reflected onto a relational database schema [133]. CCLRC uses a relational database to store scientific metadata and the real data is saved into the SRB data storage software [42] (SRB was superseded by iRods [30]). When searching/retrieving specific data, the researcher can search the database by providing the necessary metadata information, such as project name, date, data analysis software, etc, via web/web service interface, and retrieve the real data via web, web service interface or even Grid interfaces. Researchers have used the functionality provided by this mechanism to manage environmental data [91, 137, 133]. The MeCat project [73] is another example of using CSMD catalog to manage scientific data. MeCat is a joint project between the Australian Synchrotron [55] and ANSTO [74]. The goal of the project is to improve metadata management for experimental data and publication. The size of experimental data to be generated is about 2 TB annually. *eMinerals* project [17] is another project that applies both the Grid and CSMD catalogs technology. The primary object is building a simulation environment for computational researches in the mineral research area. The project uses CSMD model to store data for complex materials simulation.

◇ GRANI (Grid-enabled Archive of Nanostructural Imagery) project for the Australian Nanostructural Analysis Network Organisation (NANO). GRANI provides the NANO research community with a scalable, distributed data management solution and a secure collaborative environment to ensure high speed access to and seamless sharing of their data, instruments, analytical services and expertise. The project links the databases that host the data, particularly the images captured from different instruments (e.g. microscopy), and links them to HPC facilities for im-

[1] The data about the experiment and the data generated from the experiment. For example, the time, location, equipment specification, and the classification and storage of the generated data.

age analysis and 3D construction via a web portal interface. One particular reason is the increasing amount of data they need to process. For example, in the NANO community, increasing speed of image generation and the higher resolution leads to more and much larger images.

◇ Another e-Science project is Open Microscopy Environment (OME) [118]. This project has analyzed and adopted the technology from previous projects. However, it does not use a cross-disciplinary data model, such as CSMD. Instead, the OME data model [75], a very specific data model defined by the microscopy community, is applied and data is stored by using a MySQL[51] database. The project also uses computational power provided by Grid infrastructure.

All these projects must solve two main issues: computational power and data. With the rapid increases of computational power in these years, most research projects have found that it is relatively easy to find computational power to meet their requirement. On the contrary, storage, transferring and management of project related data is an even critical issue they need to handle. Scientific experiments often generate an enormous amount of data, often distributed. They need to be collected, transferred, analyzed, classified, and stored into different catalogues. For each step, not just hardware needs to be provided (e.g. large data storage capability for the Terabytes of data per day generated at CERN [24, 38]), but also software. For cooperation between different participants in a project, the data need to be annotated with metadata and must be transferred efficiently to meet the processing requirements.

In our research work, we study the collection and transfer of distributed data. Scientific instruments are often used to generate large amounts of data that needs to be accessed in a distributed environment, since instruments, the data storage, the servers for data processing and the scientists that need to access the data are typically not co-located. For example, in a synchrotron, each experiment can generate hundreds of Gigabytes of data that needs to be accessed in a day for a single experiment and processed by researchers who are usually located far from the instrument used for the experiment. We used the data generated from these synchrotron experiments as the source of distributed

data. Even though our work was carried out on a specific case, the target of the research is to find a general approach for distributed data collection and transfer for scientific instruments. To provide this generality, we need a general purpose middleware between the data users and any instruments. With such a middleware, it will also help to have the instrument appear as a first level element in a distributed computing environment.

3.1.2 Instruments in e-Science

In e-Science, helping scientists to access remotely located resources and facilities, such as instruments and sensors, is another problem that has triggered many research initiatives. There are many different middleware that are designed to support such functionality. We list a few:

- ◇ Grid enabled Remote Instrumentation with Distributed Computation and Control (GridCC) [7] has extended the existing Grid environment to distributed remote instruments and provided a complete set of middleware for these instruments to utilize the functionality provided by the existing Grid infrastructure, such as locating and utilization of computational power and data storage capacity in the Grid. However, it is too complicated to install and use, particularly for relatively simple instruments.

- ◇ Real-time CORBA with TAO (The ACE [2] ORB[3]) [135], is a middleware built on CORBA to facilitate high performance computing related real-time communications between distributed applications.

- ◇ Common Instrument Middleware Architecture (CIMA) [94, 80, 81] aims to convert the scientific instruments and sensors into first-class Grid elements and integrates these facilities into the Grid environment. The CIMA interface can be seen as a software agent of these data sources. CIMA uses SOAP as the standard communication protocol based on an HTTP communication layer. CIMA uses *parcels* (XML encoded) to send data, including its metadata. This is fine for small

[2]The ADAPTIVE Communication Environment (ACE(TM))
[3]Object Resource Broker

size data, but will be quite inefficient for large binary data. Compared to TAO CORBA, web service interface is a more open and flexible approach, as the web service has a better separation of the content from communication. The design requirements of CIMA are interoperability and efficiency.

The design of systems to process the data should not only focus on the requirements specific to an individual instrument, but also be able to accommodate possible changes to the way the instrument is used in the near future. In our research work, we would like to use a middleware to allow the workflow to collect distributed data. The middleware will be ideally lightweight to avoid having significant performance degradation and at the same time, it should be flexible so that it can be used within a group of loosely coupled applications without losing generality. Thus, CIMA is selected as the instrument data collection middleware in our project. CIMA is a light weight middleware with web service interface for instrument control and data collection. CIMA is designed to be used for general purpose instruments, so by using CIMA for our experiments, we can be sure that the approach of our research work can be applied more generally to most other scientific instruments.

Workflow is the organizer that puts all individual elements together and is responsible for coordinating the execution of these procedures. By building a workflow that contains both the instrument and the data processing web services, a researcher can simplify his/her task to carry out the experiment.

3.2 Workflow

Workflow is an important tool for researchers to utilize and coordinate distributed resources. For example, a workflow associated with a synchrotron experiment can start the experiment, capture data from an instrument, annotate this data with metadata, then process and store it in a data repository. When there are remote applications involved in the workflow, e.g. a client side application located in a remote place, the data can be sent to the client for further processing.

In this section, we define the types of workflows, look at some workflow examples, and introduce the important notions of centralized and decentralized workflow.

3.2.1 Workflow Definition

A workflow is composed of multiple independent processing procedures. The advantage of workflow is that the individual procedures can be used to build different workflows. These procedures/services don't need to be tightly coupled with a specific workflow. On the other hand, they can be shared by different workflows. To expose these procedures to different services, the most popular method is web service.

By defining a workflow, new functionality is created. Each individual service can only provide a certain processing functionality. The workflow provides enhanced functionality after individual services are composed into it.

We define web service workflow as a *composite* web service, as shown in Figure 3.1. A composite web service is different from an atomic web service. In the context of our research work, it provides a more sophisticated functionality which provides the combination of multiple atomic web service functionality. At the same time, a composite web service is different from normal web services in the way that it communicates with its client. A normal web service has only one input and one output, including both data and control information. Within our *composite* web service definition, unlike the single web service, multiple interactions can happen between the client and the web services that compose workflow, as shown in Figure 3.1.

3.2.2 Workflow System

A workflow system involves procedures/services, workflow engine and workflows. A procedure is a program that takes an input and returns result. Procedures are often exposed to the outside as services. Workflows can be composed by using a specific workflow language, such as Business Process Execution Language (*BPEL*) for Web Service (*BPEL4WS*) and Simple conceptual unified flow language (*Scufl*). BPEL is a workflow language for web service orchestrating in a web service workflow [138].

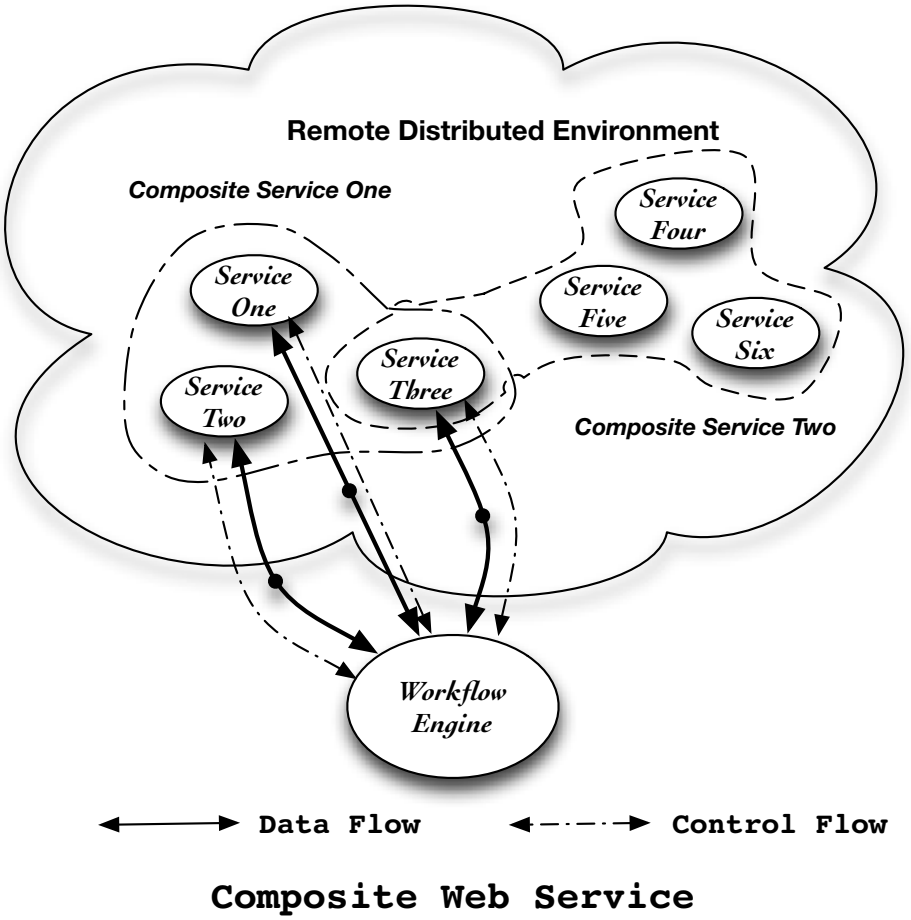


Figure 3.1: Composite Web Service

Scufl is a workflow language for users to create and run workflows using the *Freefluo* workflow enactment engine [127, 128] .

In a workflow system, the workflow engine takes the workflow file as input, interprets the file and runs the workflow for the user. A genuine workflow engine often provides different programming models (e.g. synchronization between individual procedures in a workflow) to support the execution of a workflow.

There are multiple popular scientific workflow systems, such as *Kepler* [26], *Taverna* [127] and *Triana* [52]. Each one has its own workflow engine as well as supported computational models. Take *Kepler* for example, it is built on the computational models provided by Ptolemy II [15], which is a software framework developed as part of the Ptolemy project. The models include synchronous data flow (SDF), continuous time (CT), process network (PN), and dynamic data flow (DDF), and some others [15].

The models are described by using Ptolemy's XML based Modeling Markup Language (*MoML*). Once a workflow file is passed in to the Kepler workflow engine, it is interpreted and run by the underlying workflow engine.

Workflow systems can be divided into two different categories, centralized and decentralized, according to the location of the workflow engine, i.e. the location where the control messages are created [141]. With a centralized workflow management system, the user of the workflow can directly control and monitor each step within the workflow engine. However, with the decentralized workflow, both data and control are distributed on different services, so that workflow functions are fulfilled through the direct communication and coordination among the relevant services [141].

There are two different communication models between the embedded web services, in terms of collaboration between participant services in the workflow: orchestration and choreography [101, 122]. Orchestration indicates different participants of a workflow do not have a direct relationship between them. Orchestration is the typical communication model with centralized workflow. Any messages from one service are sent to the workflow engine, before they can be forwarded to a second web service. In the choreography model, which is often seen within a decentralized workflow model, one application involved in a workflow can directly communicate with a different application in the same workflow, without going through the workflow engine. In an orchestration model, the workflow engine can have better control of the whole workflow compared to a choreography model, but as all messages will go through the workflow engine, it can become a bottleneck for the whole workflow.

With a decentralized workflow model [141, 142, 121], a workflow is controlled by different instances of the workflow engine on separate nodes at different stages of the workflow execution. One of the problems is that the user can lose control of the workflow. As the workflow engine is located on a different host computer when the workflow is executed. Each of the workflow providers hosts a workflow engine to interpret the workflow and execute one or multiple steps according to the the instructions of the workflow. Then the instructions of the workflow and the control of the workflow will be forwarded to the next workflow engine. As there might be different workflow engines in-

volved in the execution of the workflow, connections between a current workflow engine and the client can be broken and the client can lose control of the workflow. Suppose a workflow is composed of multiple computational services to sequentially process a set of data. With a decentralized workflow, each service is invoked by a workflow engine running on the same node of that service. After the processing of the current service, The workflow instructions for the next service will be sent to the next workflow engine, together with the result data, if any. The control of the workflow is also transferred to the next workflow engine. A client can build a connection with the current workflow engine to monitor the progress of the workflow. However, if this connection is broken, then the current workflow engine will either continue to invoke the service while the client has no idea of the current progress of the workflow, or wait until the client is reconnected. In a centralized workflow model, on the contrary, the client can always stop the workflow engine in the first place.

To the best of our knowledge, this issue has not been discussed, either in [141] or other research. The other limitation for decentralized workflow is that the workflow engine needs to be installed on each participant host. To install workflow engine to host a service, even with a single service, will cost extra resource on the hosting machine. In a distributed environment, it will also be hard to have every host to install an extra workflow engine.

With the latter situation, as far as we know, no such a protocol has yet been defined. Further, there is no report on any prototype workflow system that is built on this model. Therefore, the choreography model is often theoretically suggested, but there is no concrete evidence as a result of research work that proves the feasibility or performance improvement of this model. In our research work, we take the centralized workflow as the default model to build workflows. In the following chapters, we discuss data transfers under this model and address ways to minimize unnecessary data transfers between the central engine and applications in the workflow.

3.3 Data Transfer and Sharing within Workflow

3.3.1 Data Transfer with e-Science workflow

Scientific workflow often involves large scale data transfer and sharing between applications involved in the workflow. A primary goal of e-Science is to provide scientists a friendly and collaborative environment for their research. Some research work, such as material structure research, particularly relies on experiments. To meet this requirement, instrument middleware, such as CIMA [80] is used to help the scientists to conduct their experiments.

CIMA has two major components [80], instrument representative (IR) and consumer application. The IR component can be installed on the instrument side and streams real-time data to the remote consumer application. In a workflow, the two components are represented by two individual applications in a workflow.

3.3.2 Web Service Data Transfer

Web service is a communication framework for distributed computers exchanging messages. Web service is platform and programming language neutral, which is very important to build an open collaborative environment. Web service workflow facilitates the composition of multiple web services into a composite web service. CIMA as a middleware provides a web service interface between the user and the experiment. Further, it can be composed into a web service workflow as a normal web service provider for instrument related research.

Web service uses SOAP [113] message as the *de facto* communication protocol. The XML language based SOAP message has advantages for normal character-based data transmission, as the message is structured and is machine readable. However, it has difficulties with binary content transmission. The normal way of transferring binary data via SOAP message is to use binary encoding algorithms, such as Base64 [119], to convert binary file content into characters, embed these characters into a SOAP envelop and send to the receiver. The implementation of this approach is relatively easy and straight-

forward. The disadvantages of such implementation are more CPU time and memory usage as well as more bandwidth required for data transmission. An alternative approach, using a structured XML binary representation, is discussed in [98]. We believe this will bring some benefit, but the approach has significant limitations. First, some binary data is not structured, therefore can not be presented in this way. Second, legacy programs are used to analyze binary data that is not in XML format. To utilize these programs, we need to provide raw binary data rather than other formats. Last, conversion between XML formatted data and raw binary data will always consume CPU time and use more memory, degrading system performance. To avoid encoding the binary information, another common approach is to use a separate communication protocol for the data transfer, while using the web service as the controlling interface [144]. This can be effective, but introduces another system which is separate from the web service interface. This will make the programming and maintenance more difficult.

To overcome these difficulties, SOAP with Attachment (SwA) [85] and MTOM [114] (as the application of XOP [115]) have been introduced to send the binary information as the attachment within the web service framework. With SwA, the whole message is treated as an MIME [107] package. With MTOM approach, all content is part of the XML infoset and part of the whole XML infoset, which is nominally Base64 binary encoded, can be optimized and represented as binary format where it is feasible.

3.3.3 Web Service Data Sharing

In a web service workflow, data is often shared between different applications. In the orchestration model as discussed in 3.2.2, data from one service can only be shared by a different service in the workflow via the centralized workflow engine. If the data can be shared by the second service in a choreography model, then the overall data transfer will be significantly reduced.

To improve the data transfer/sharing performance within a web service workflow, we need to investigate both data transfer between services and the sharing between them. In the following chapter, we start to report our work on improving the workflow perfor-

mance by increasing the data transfer performance between two applications within a web service workflow. In later chapters we indicate how to share data between services to reduce the time taken for web service workflows.

Chapter 4

Web service Data Transfer with SOAP

Data transfer between the applications in a distributed environment can affect the overall performance of the whole system significantly. Compared to local communication, the cost of transferring data between different nodes is much larger. Therefore, it is vital to reduce the overall time used for the data transfer. Here, we present our research work that has been carried out to improve the data transfer efficiency. Our work primarily focuses on data transfer between SOAP web service in a distributed environment.

4.1 Introduction

Web service is an important component of distributed computing systems. They are based on open protocols and the related information about the service is published by the service provider using WSDL.

Web service is widely used by different distributed collaborations and users. The advantage of web service is that it has defined open protocols that enable different programming languages and operating systems to communicate with each other using standard communication protocols. The primary communication protocol used by web service is HTTP and the communication format is SOAP message, which is based on XML. web service uses the client-server model, in which the client sends a request to the server and the server processes the request and returns the response message which includes output data. Clients and servers exchange information and data by embedding

them into SOAP messages. As the SOAP is built on the XML language and is structured information, it is good for communications between different participants, i.e., SOAP messages are ideal to meet the requirements for applications running in a loosely coupled distributed environment. However, this model has not adequately considered issues of large data transfer, which are particularly important for eScience applications.

In this chapter, we first describe our approaches to improve the overall data transfer performance between web services from the following aspects: TCP tuning, communication model (push vs. pull), combining SOAP message with other communication protocols and multiple-thread connection. After that, we apply web service with attachment, which provides efficient binary data transfer capacity while supporting a web service interface, as the primary way for binary data transfer and compare the performance with a mainstream high-speed data transfer tool – GridFTP[88]. Our investigation indicates that the overall performance of data transfer within an e-Science related environment can be significantly improved by using this approach.

4.2 Research Background and Motivation

This research work started from an interdisciplinary e-Research project with synchrotron users in Australia [81]. The primary motivation of the project is to provide an improved environment in which these researchers can conduct collaborative research across international groups and facilities [144]. The initial experiments we are targeting are X-ray micro-diffraction fluorescence probes, which will be installed at the Canadian Light Source (CLS) [57] and the Australian Synchrotron [55]. Our investigations are currently using UNI/XOR, a similar beamline at the Advanced Photon Source (APS) at the Argonne National Laboratory in the US [144].

During the research work, we found that one of the most important issues for us is to improve the data transfer speed between the instruments and its data users to meet the requirement for the collaborative work.

4.2.1 Synchrotron Project and CIMA

Conducting collaborative work for remote users is currently an expensive and time-consuming process. For example, current practice for researchers using the APS is similar to the situation for most large, shared scientific instruments. The data is collected and stored at the instrument site, often with the scientist being present to oversee the experiment. Researchers return home to analyze their data (often taken on a removable disk drive) after the experiment is completed [144]. The data analysis may take many days for a personal computer to process. The data and its metadata are typically stored in an *ad-hoc* fashion, which is often only understood by its creator and makes it very hard for other collaborators or researchers to understand. For international collaborators in particular, this is expensive in requiring visits to the remote site; furthermore, slow data transfer is always a problem. Also, it is very difficult, with ad-hoc and uncoordinated data and metadata storage, to undertake any comparative analysis with earlier experiments [144].

To improve the distributed research environment for collaborative researchers, we adopt, adapt, extend and enhance the Common Instrument Middleware Architecture (CIMA) [94] model. CIMA is an NSF Middleware Initiative project that is developing a consistent and re-usable middleware framework to support instruments and sensors within an e-Research infrastructure. It was initially proposed by researchers from Indiana University and was adopted and adapted by some universities from Australia, such as University of Adelaide, James Cook University and Sydney University [81, 94]. We use CIMA as the platform to carry out our research work on SOAP based data transfer, as we believe it provides a very general platform and the web service based interface is relative simple. CIMA has been installed at the UNI/XOR and ChemMatCARS beamlines of the APS.

CIMA structure has two parts [94]–client and server. The Instrument Representative (IR), also called *sender,* acts as the server in CIMA framework. A Channel Sink, also called a *receiver*, is the client part. Both of them have a web service interface. The IR represents the instruments that are installed at its back end and the receivers represent

the instrument users. When the framework is built, the receiver initialize the interaction by sending a web service request to IR to register itself with an instrument, e.g. a sensor, or CCD frame generation on the server side. It also has a web service interface to receive the information sent from the IR. After the registration, once there is any data that meets the registration requirements, the IR will send them to the registered receiver, which is a push model.

4.2.2 Motivation for the Research Work

The CIMA framework provides multiple functions via the web service interface, such as instrument monitoring, data downloading, and remote instrument control [94]. Our work has focused so far on data downloading and metadata information extraction from these downloaded data. These have a primary requirement of data downloading in as close as possible to real time, so that the outputs of the experiment can be remotely monitored, and the experimental procedure or inputs can possibly be modified while the experiment is in progress. Within the whole process, from data generation to receiving of the data by receiver, metadata generation, and data/metadata storage in a digital repository, the most time consuming part is the data transfer between the IR and the receiver, as the data needs to cross a high latency network–in our case, from US to Australia. We have considered possible ways of increasing data transfer performance while keeping the web service based framework.

The initial work on our synchrotron project was aimed at determining whether image data from the UNI/XOR 34ID-E end-station at APS can be downloaded and stored at Adelaide, Australia, in real-time. Different cameras used on this instrument generate different file sizes (from about 500 KB to 8.5 MB) at different rates, but in each case the download speeds required are similar, a few mega bits per second. The instrument produces SPE [69] (a proprietary format) files generated by the camera CCD. We first tried downloading from APS using *scp (secure copy)*, but found that the data transfer speed was not fast enough to provide real-time download for some experiments, i.e. the time to download a file from the file server at the UNI/XOR beamline at APS to the

storage system in Adelaide was slower than the time taken by the instrument to generate a new file. We also found that data transfer rates varied considerably (by up to a factor of 10) over the length of an experiment (typically a few days), so in some experiments the download could sometimes keep up with data generation at the instrument and sometimes not.

However, we improved the the downloading speed by using a multi-threaded download client that allowed for concurrent downloads of multiple files. Using multiple concurrent data streams allowed total data download throughput to be 4 to 8 times higher. We then moved to using CIMA web service for data download, and found that we were again no longer able to obtain real-time download.

We believe there are two reasons: firstly, the data transfer speed for a single file was slower using the web service than using *scp*, and secondly, existing implementations of CIMA senders and receivers did not allow the use of multiple concurrent data downloads to improve data transfer throughput. This provided the motivation for our research work to explore alternative options for both the CIMA architecture and specification, and for the implementation of CIMA senders and receivers, in order to optimize data transfer performance.

CIMA is intended to be a general framework for interfacing to any type of scientific instruments, encompassing a wide variety of data generation speeds, data file sizes, network connectivity, data management workflows, and experimental requirements. Here we focus on results for a particular type of experiment, however the issues, approaches and performance comparisons presented are more generally applicable.

4.3 Web Service Data Transfer with SOAP Message

In an ideal e-Science environment, researchers from different organizations, different locations and even different disciplines could join together to carry out a common task. Web service architecture (WSA), as a loosely coupled environment for distributed applications and data sharing, is a good choice for research collaborations. Different applications within the e-Science ecosystem share their information by exchanging SOAP

messages, which is platform and programming language neutral. However, as e-Science applications often involve large binary data transfer between applications, the efficiency of binary data transfer is vital to the overall performance of these applications.

4.3.1 Data Transfer Performance Analysis

To improve the overall performance of data transfer with SOAP messages, first we need to analyze the relevant aspects that might impact performance; then we propose the possible ways that could lead to performance improvement.

The data transfer between two participant nodes in a web service context, especially within an e-Science related research environment, involves various elements. These aspects include but are not limited to the following factors: network connection and its configuration; data transfer model – push vs. pull; transferring protocols, e.g. plain http vs. SOAP; as well as using multiple data connections for data transfer. Our research is to figure out the best approach for data transferring within the e-Science context by looking into these aspects.

- ◇ Network connection and its configuration within the e-Science environment. For example, within an e-Science workflow, remote service provider from a different organization could be located in a different continent. Latency between these services can be relatively large. How to configure the network setting in the client and server side respectively, so we can reduce the impact of this latency is very important. Our research work focuses on how to take advantage of TCP tuning in the web service based workflow to reduce this impact.

- ◇ Data transfer models: the *push* and *pull*. In the push model, the server makes the decision on when and how to send the data to the client; on the contrary, in the pull model, the client will start the procedure of downloading data. Within our research work, we take the benefit of web service and the combination of web service with other communication protocols to make the decision of how to build our own model for workflow data transfer. For details, refer to section 4.4.2.

- ◇ The selection of communication protocol. For information to be better understandable to other systems, we use the web service interface by passing SOAP messages to transfer these information. For raw data, we prefer using other protocols such as http, TCP or even BitTorrent to transfer them.

- ◇ Multiple network connections to improve data transfer performance. With the development of commodity computer hardware, multi-core computers have become mainstream and the bandwidth of network connections is also improving. Our research work also try to figure out how concurrent downloading can improve the performance of the overall downloading efficiency.

Within the following sections, we describe the details of improving data transfer performance from the above aspects.

4.4 SOAP Message Data Transfer Improvements

As the messages that carry the web service payload use SOAP, which is an important message passing format in e-Science, we investigated how to improve the data transfer rate while using SOAP. There are several aspects of the SOAP message that we have explored: TCP tuning, push vs. pull model and multiple connections for data transfer.

4.4.1 TCP Tuning

According to the TCP specification [28], when a client is connected to a server, they exchange information about their *receive window* and *send window.* The window size indicates the device's buffer size for sending or receiving data. On any computer system, changing this setting needs administrator's right and some knowledge of networking. However, for data intensive e-Science workflows, the optimization is an important step to improve the performance of the whole system when the workflow involves large data transfer.

The default TCP configurations has buffer size set to 64k bytes and fits slow network connections and connections with small latency and, under most circumstances, network

related applications only sending small amounts of data work well with the default TCP buffer size. However, for long distance, high latency connections, after sending out the data in the buffer, the sender needs to wait for the response from the receiver before sending out next batch of data. TCP tuning is to optimize the data transfer by increasing the kernel buffer sizes on both the sender and the receiver's side. The application's TCP buffer sizes on both sides should also be increased [28]. For example, the *java.net.Socket* class provides methods such as

$$void\ \ setReceiveBufferSize\,(int\ \ size)$$

to set the *SO_RCVBUF* option to the specified value the *Socket* object [65] used in an application. This enables the sender to keep sending out more data while waiting for receiver's feedback. The total time, from when the initial data is sent until the sender receives any feedback, will be about the Round Trip Time (RTT). The *ping* program in most systems can be used to get the RTT, which is as twice the delay time. During this time, the overall data the sender can send is limited by the minimum bandwidth of any component of the network between the two end points of the data transfer. The best data throughput is obtained if the TCP *buffersize* is set to this data size [28].

$$buffersize = 2 * bandwidth * delay \tag{4.1}$$

If we use the RTT to replace *2 * delay,* then equation 4.1 should be changed into:

$$buffersize = bandwidth * RTT \tag{4.2}$$

The TCP tuning has been widely used at system level [28]. However, to the best of our knowledge, no web service level applications have been modified to take the advantage of TCP tuning. Our work indicates that web service applications can also benefit from this optimization, as explained in section 4.5.2.

4.4.2 Push/Pull Model

In a client-server data transfer system, *push-based* (referred to as *push* model) data delivery is when a server sends data to its clients without receiving any requests from the client [77]. On the other hand, if the client first sends a request for data downloading, then it is called a *pull* model. Within our research context, we build our experiments with CIMA (refer to section 3.1 for details). When data is generated from the server side, it will be sent from the server to the client. It is natural to apply the *push* model to send data from the server to the client. However, there are problems using the push model which we believe can be solved using the pull model, as now described.

- ◇ Normally, the data is sent to the client when they are generated, e.g. in an experiment, on the server side. Therefore, the client has to download these data when they are generated. As data downloading needs a quite significant amount of resources, the client needs to dedicate its network bandwidth as well as computational resources for data downloading, which is not flexible for the client. On the contrary, the server is often more dedicated on certain tasks, so if the download can be activated when the client is idle, it will be better for the client to select the right time to balance their resources. If we can apply *pull* model, the client can choose to download these data when it is relatively free.

- ◇ If the connection between the client and server can't meet the speed of the data generated, the network will be overloaded by the data transfer. The *pull* model can control the speed of the data downloading, therefore avoiding network congestion and potential for very slow or incomplete data transfer.

- ◇ Some clients may only be interested in a subset of data. For example, the client might only want to download a small part of the output from an experiment to check if it is working correctly, or to download a particular data set from a large data repository. Using *push* model will send all data to the client, ignoring the client's preference. With a *pull* model, the server can forward the meta-information of the data sets or a small part of the whole data set, and allows the client to decide which data set to collect and initialize the downloading procedure.

◇ The client may be able to select a preferred protocol. For example, if a server provides both GridFTP and HTTP service for data transfer. The client can pick the one it preferred to download the data.

4.4.3 Selection of Communication Protocols

Primarily, web service uses SOAP messages to transfer data. However, for binary data transfer, using SOAP messages requires converting the binary data into characters, which is not efficient, since it requires more data to be transferred. For example, for the Base64 encoding algorithm, there is a 33% increase of data transferred. We can use HTTP or other communication protocols such as *sftp* or *BitTorrents* if it is necessary. *Sftp* [19] is an extension of ftp and is designed for large data transfer between different nodes in a network, but it is not used as widely as HTTP. *BitTorrents* [56] is another peer-to-peer (P2P) communications protocol for file sharing. It will be very efficient when there are large number of users sharing a particular file or data set. For e-Science related data sharing, the users are basically limited to the organizations that are involved in the collaborative work, so there is no advantage to turn to this protocol. Compared with sftp and BitTorrents, HTTP is more widely used and supported. It can also support binary data transfer without having to use an encoding that increases the data transfer size. We believe this is a better choice over other communication protocols.

4.4.4 Sending Data as Attachment

Web service has provided a better communication model for distributed and loosely coupled systems. However, sending binary data to the receiver by using the SOAP format is not efficient. The web service community has introduced another specification – *Web Service with Attachment* (WS Attachment) to minimize the binary data transfer time under the web service architecture. With this specification, the *attachment* is sent as part of the HTTP POST call, but outside of the SOAP envelope. The motivation for using SOAP with attachments is to remove the binary part from the XML payload, and into the HTTP request as multipart/related MIME content. The performance of using

SOAP with attachments will be discussed in chapter 5, where a significant improvement of performance was observed.

4.4.5 Multiple Data Transfer Connections

With the popularity of the off-the-shelf multi-core CPU computers with memory size up to a few gigabytes, concurrent data transfer between sender and receiver can offer a significant advantage. For example, within a large binary data transfer process, disk reading and writing will consume considerable amount of time. By using multiple data connections, at a certain time point, while some threads are focusing on data transferring via the network, other threads can process disk I/O operation or other activities. We carried out comprehensive data transfer tests with this method and were able to improve the overall performance for data transfer between SOAP message based web service applications.

4.5 Improving Data Transfer in an e-Science Context

Our research project is based on the context of an interdisciplinary e-Research project that is to serve the synchrotron users in Australia. The project's primary motivation is to provide an ideal research environment in which researchers can improve the efficiency of the collaboration in the group and maximize the utilization of the facilities.

Our research was initially targeted at X-ray micro-diffraction fluorescence probes, which would be installed at Canadian Light Source (CLS) and the Australian Synchrotron. As these facilities are accessible via the Internet, theoretically, the researchers can monitor and control the X-ray experiments. Our experiments tested remote monitoring and data access using the CIMA architecture, which is a common middleware architecture for making instruments and sensor networks to be controlled and communicate in a standard way (for more details about CIMA, please refer to [102])**.**

The X-ray experiment in this project generates different sized files, ranging from 500KB to 8.5MB, at different rates. The average speed expected for data downloading is a few megabits per second with an inter-continental connection [144]. One priority

of our research is to enable the real-time download of the experiment data from where it is generated. Initial testing showed that this is challenging as the distance between the users (Adelaide, Australia) and the location where the experiments are carried out (APS, Canada) is several thousand kilometers. The latency between these two locations is also very high - more than 200 milli-seconds.

To prove the concepts we proposed for better data transfer between different nodes in the e-Science context, we applied the suggested methods (see section 4.4) on the CIMA based interface as our test bed.

4.5.1 Experimental Framework

We build the experiment by setting the data provider in the University of Indiana, USA and the client in the University of Adelaide. We chose these locations, as they provided a good example of downloading data from remote facilities. We carried out the data download experiments via CIMA web service between these universities as Indiana University is close to APS and provides the computer and network resources we need to carry out the experiment. The distance between Adelaide and Indiana is approximately 16,000 kms and the RTT time is about 225 milli-seconds.

To carry out the experiment under the CIMA architecture, we need a *sender* and a *receiver* as the data generator and data consumer respectively. We developed our basic implementations of a CIMA Instrument Representative (as the *sender* or IR) and a Channel Sink (as the *receiver*), based on the WSDL definitions of the CIMA web service. On the sender side, a separate program is used to emulate the creation of data files in a real synchrotron experiment, by copying some synchrotron data (CCD image files) of a given size to a directory where they are accessed by the CIMA IR program. The wait time between copying each data file to the directory can be configured in the experiment emulation program, and was set to be the actual time taken to generate files of that size in a real UNI/XOR experiment. The IR is written in C++ and used gSOAP [64] to generate the web service interface, since this is what was used in the Indiana University implementation of CIMA, which is the one deployed at the APS. The receiver

program was written in Java according to the WSDL description, and extracts the data file from the messages sent by the sender and saves it locally. XFire [27] was used as the web server, since it is believed to provide better performance than Apache [27]. While this was not a full implementation of CIMA, as the data is not generated from a real instrument, it fully implemented all aspects of transfer of binary image data [144].

For hardware, the computer used in Adelaide has dual 2.8 GHz Intel Xeon CPUs with 2 GB memory and running with Linux 2.6.18 kernel, and the server machine in Indiana is also a Linux box with a 2.4 GHz Intel Pentium 4 CPU, 1 GB memory. The kernel version is 2.6.19.

We used *iperf* [31] to measure network performance between the sender and the receiver, as it is a widely accepted tool for this purpose. Results from using *iperf* showed that the maximum data transfer speed between Adelaide and Indiana was about 20 Mbits/sec in either direction. The transfer rate from Australia to the US was fairly constant, however the transfer rate from the US to Australia was highly variable, and would sometimes decrease to around 2 Mbits/sec, particularly in the afternoon (Adelaide time). This is presumably because most of the data traffic flows from the US to Australia. We therefore decided to do all our experiments with the sender at Adelaide and the receiver at Indiana, to allow better performance comparisons when experiments were run at different times [144].

In the following sections, we test the proposed methods to improve the overall performance of the data downloading. We carried out these experiments respectively to verify the real improvement.

4.5.2 TCP Tuning

First, we optimize the performance of data downloading by applying TCP tuning to the SOAP based web service. In the testing environment, the sender and the receiver are remotely located – the University of Indiana (US) and the University of Adelaide (Australia). This leads to the fact that the network latency is high. By using *ping* to measure the RTT between the nodes, the average value was found to be 225 milliseconds. We

also figured out that the minimum bandwidth between the two nodes is 100 Mbits per second.

According to equation 4.2, the buffer size should be equal to or greater than:

$$BWmin * RTT = 100Mbits/s * 0.225s = 22.5\ Mbits = 2.9\ Mbytes \tag{4.3}$$

Based on the value we get from 4.3, we set the buffer size to 8M bytes, which is significantly larger than the minimum value as shown above.

To improve the data download performance using TCP tuning requires modification to be done for both the sender and receiver, and at the application level as well as kernel level. It is difficult to continually change the kernel settings at both sender and receiver machines, so we set the TCP buffer size in the kernel to be 8M bytes (significantly larger than the recommended minimum value) and then varied the buffer size at the application level. The TCP tuning requires modification to be done on both the client and server side. On both sides, we first adjust the buffer size in the kernel to 8M bytes, then adjust the buffer size in the applications accordingly. On the IR side, the gSOAP library, which is a C/C++ based web service library, is used to compose the send service. The gSOAP library has provided *SOAPBUFLEN* as a parameter for the programmer to set the buffer size in the program. We changed the value of this parameter from 64K bytes to 8M bytes. On the client side, we use a Java based web service application, which is created by using XFire [27] web service library, to download the data from the client. The XFire library, however, unlike the gSOAP library, does not provide an interface for the programmers to revise the buffer size from the application level. We have modified the code of XFire to allow the buffer size to be set to a specified value. In this experiment, it is also set to 8M bytes. Following these steps, we have set both the client and server side to 8M bytes for buffer size. During the experiments we tested different data downloading performance by applying various buffer sizes. These are carried out by varying the sender's buffer size in gSOAP.

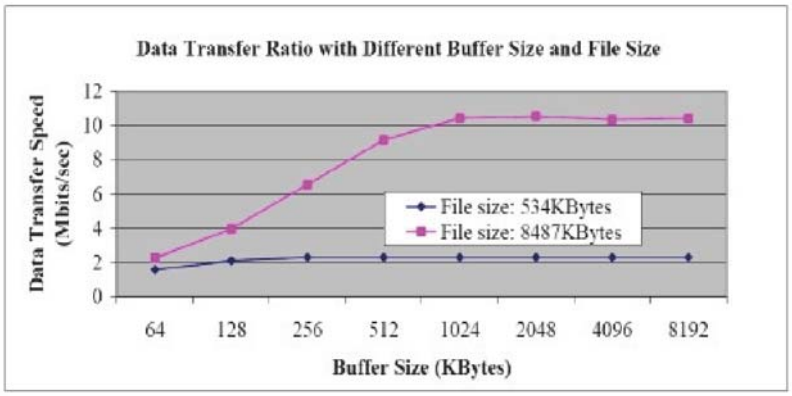


Figure 4.1: Data transfer speeds for different buffer sizes and file sizes using SOAP with Base64 encoding (push model).

With the experiment, we use 0.5 MB and 8.5 MB files, which are two typical file sizes generated from the CCD experiment as the data to be transferred. Testing results for TCP tuning are as shown in Figure 4.1.

The result has shown that for small size data files, the files with size 0.5 MB in this case, when the sender buffer size increase from 64K to 128K Bytes, the data transfer speed increased from 1.56 MBits/sec to 2.09 MBits/sec, i.e. about 34% increase; when the send buffer size increase from 128K to 256K Bytes, the data transfer speed increased to 2.29 MBits/sec, about 10% increase more. After that, buffer size increase does not bring much benefit. Once the buffer size is large enough to fit the current file in, the performance of the data transfer is not limited by the buffer size. However, the increased buffer size still have larger impact for bigger files, 8.5 MB in this case, which provides an improvement of 3 to 5 times for network transfer performance. The experiment indicates that after the buffer size is increased, the overall data transfer performance has been increased. The reason for the performance difference between different files is because small files spend more time on local I/O operation and other overheads.

4.5.3 Combination of Pull Model with HTTP

In the original CIMA architecture, after the client is registered with the server, it can receive generated data from the server as a receiver, i.e. in the push model. To take the advantage of pull model, as discussed in section 4.4.2, we suggest an alternative way to

initialize the real data transfer.

In our new approach, the sender will save the data first, and provide the data downloading mechanism by sending a data reference to the receiver, allowing the receiver to decide when, and even how, to retrieve the data. The data reference can be a URL, which includes both the transfer protocol (such as HTTP or GridFTP) and the detailed location of the data.

In the new model, the receiver can decide when to retrieve the data from the sender after it receives a data reference from the sender. The operation can be selective and decided by the receiver rather than the sender. If more than one protocol and location is provided by the sender, which implies that the sender can support multiple protocol data transferring, the receiver is able to select the most suitable for effective transfer. For example, if the sender has provided both HTTP and GridFTP downloading services, the receiver can decide which one to use on the basis of available certificates and downloading tools.

The initial step of the whole procedure will still be carried out under the CIMA architecture. As the sender can communicate with the receiver in a standard way, allowing the sender to pass metadata information which includes a reference to the data, rather than the real data, to the receiver. In our experiments, we use the CIMA standard interface to pass a reference to the receiver and HTTP protocol for data transfer. We will refer to this as a "WS+HTTP" pull model. Another advantage of the pull model is that it provides an easy way to implement multiple concurrent file downloads, which can significantly improve total data transfer throughput (section 4.5.4).

In the experiment, we not only compared different transfer models for data transfer, but also investigated how the file size and send buffer size impacts on the performance. The implementation of the "WS + HTTP" pull model works as shown in Figure 4.2.

The receiver program has two function blocks within it. One is the CIMA SOAP listener which listens to the SOAP message from the sender, which has the same structure as in the push model, except for two things: first, the <FileLocation> element in the SOAP message is an URL; second, the <FileContent> element in the SOAP message is left blank. On the contrary, in push model, the <FileContent> will be filled with char-

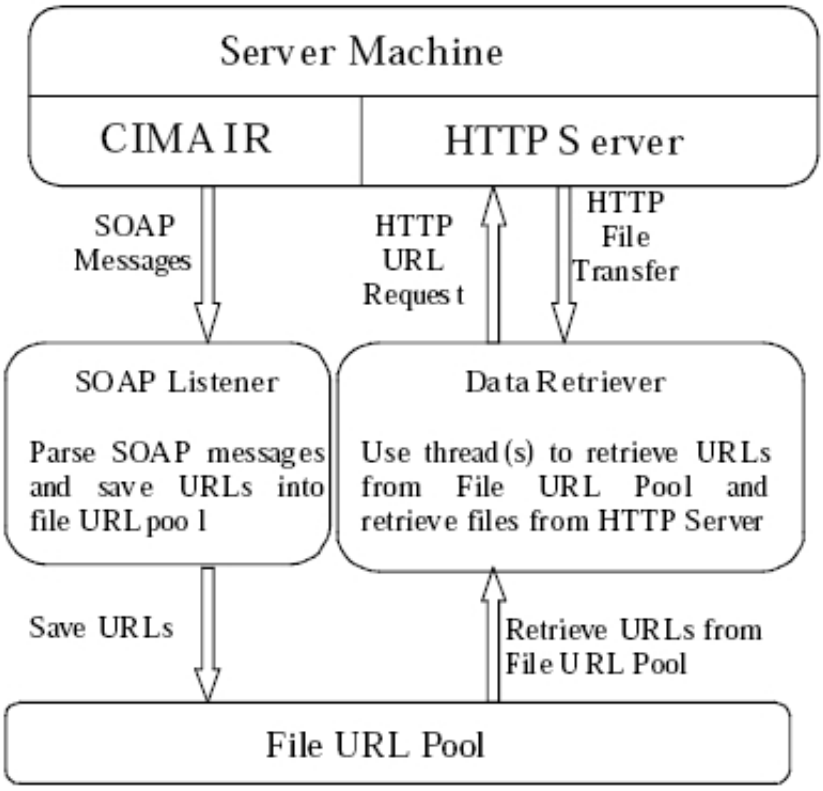


Figure 4.2: CIMA under Pull Model

acters encoded from binary data. The other block in the program is the data retriever, which is responsible for downloading data from the remote sender using an HTTP client. On the server side, Apache web server (version 2.0.54) was used to send the binary file to the receiver according to client's HTTP request. Apache has a configuration "SendBufferSize Bytes" which allows us to vary the send buffer size. Figure 4.1 shows the effects of this change, with significant performance improvement for large file size. Table 4.4 compares the results for different file sizes and send buffer sizes for the standard CIMA web service (push model) and the pull model (WS + HTTP). Here, all "WS + HTTP" measurements used only one thread to download the given file references. "WS + HTTP" always has a better performance than pure SOAP based web service. With big buffer size and big file size, it has the best performance over the existing CIMA web service method: about 2.9 times faster than the latter. With small buffer size and small file size, it is about 2 times faster than the latter.

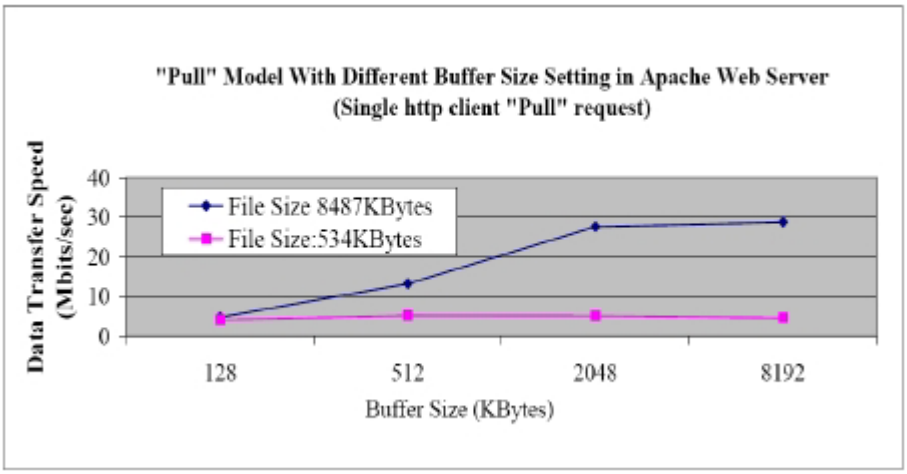


Figure 4.3: SOAP Message for Binary Data Transfer with Different Buffer Size

Num	Model	Send Buffer Size(Kbytes)	File Size(Kbytes)	Data Transfer Speed(Mbits/sec)
1	WS + HTTP (pull model)	8192	8487	28.83
2	WS + HTTP (pull model)	128	8487	4.80
3	Web Service (push model)	8192	8487	10.43
4	Web Service (push model)	128	8487	3.95
5	WS + HTTP (pull model)	8192	534	4.64
6	WS + HTTP (pull model)	128	534	4.17
7	Web Service (push model)	8192	534	2.29
8	Web Service (push model)	128	534	2.09

Figure 4.4: The Combination with HTTP in the Pull Model

As we have discussed, the pull model has provided significant flexibility over the push model in the e-Science related operations for binary data transfer. The *de facto* communication model in web service is push model and the *de facto* message format is SOAP message. SOAP message is composed of XML elements, which is good for character representation, but not really good for binary data. With pull model, we can also introduce other communication protocols which do not need to encode the binary data, therefore improve the overall binary data performance.

4.5.4 Pull Model with Concurrent HTTP Clients

The pull model provides a good chance to apply multiple threads data connections between the server and the client. In a pull model, as the client gets multiple different data references from the server, it can improve the downloading data transfer speed by using multiple data connections.

Generally, a single network connection between the client and server will not be able to fully utilize the bandwidth between the source node and destination node. Since the multiple core CPU is becoming standard for off-the-shelf computers, multiple network connections can reduce the time for data transfer between nodes. With single client and server connection, when the data transfer program is waiting for the disk operation – reading/writing from/to the disk, the network is idle and waiting for other parts of the program to finish. With multiple data transfer connections, the system has a better chance to utilize more resources and it is easy for the client to control the overall data transfer via the number of the threads.

This could be done using the push model by implementing multi-threaded CIMA senders and receivers to handle concurrent data transfers, however using the pull model only requires a simple multi-threaded receiver to invoke concurrent HTTP requests, which web servers can already handle. Multi-core processors are becoming standard, and this will improve the performance of downloading and processing of multiple concurrent data files.

An advantage of the pull model is that the client can easily be built to implement

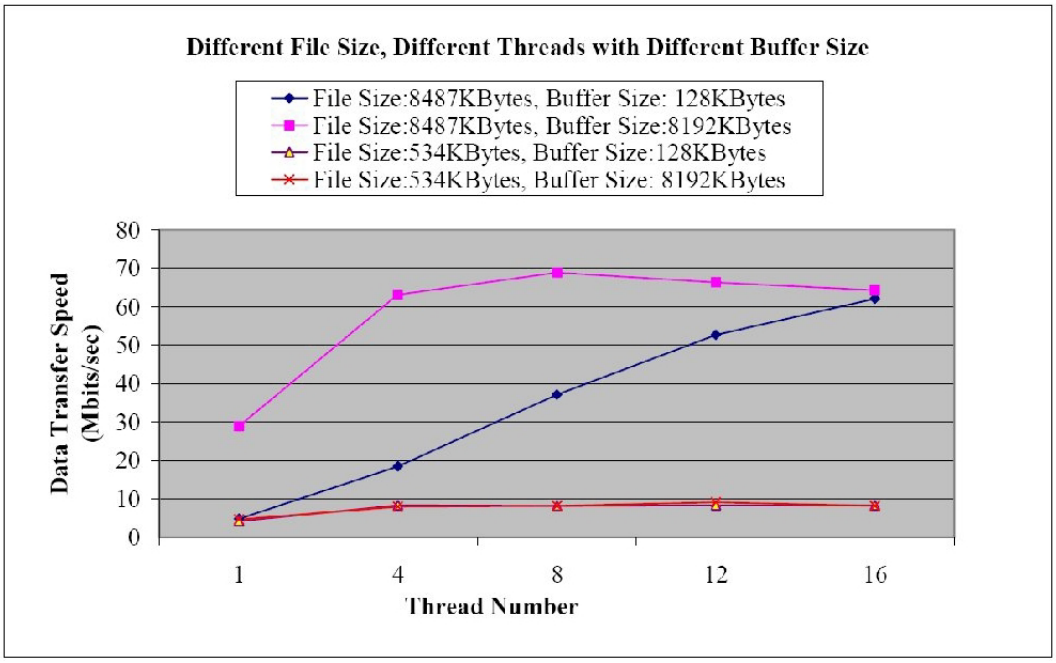


Figure 4.5: Effect of using different numbers of multiple concurrent downloads (numbers of threads) for different file sizes and different buffer sizes

multiple concurrent downloads (see Figure 4.2). We have tested the "WS + HTTP" pull model with a multi threaded client that can do multiple concurrent downloads, and using the Apache web server on the server side. Figure 4.5 shows the result of using multiple threads, each of which runs a concurrent file download via HTTP, for large and small file sizes and large and small (default) buffer sizes.

Figure 4.5 shows that with no tuning of the buffer size, using many threads can also significantly improve data transfer throughput, particularly for large file sizes, to the point where it is comparable to the performance for a large buffer size. Using a large buffer size, performance of data transfer has been increased by more than 2.3 times for large files, compared to performance with single thread. The performance of transferring small size files is also increased to nearly twice the rate for single thread data transfer. So the multiple threads client is desirable even after we applied TCP tuning and transfer model optimization, and maximum performance can be gained using fewer threads. Note that using a pull model with TCP tuning and multiple threads gives a throughput of around 64 Mbits/sec for the largest file size, which is 16 times faster than

the performance of the current CIMA implementation with data transfer done using a push model, with SOAP web service and XML, and no TCP tuning.

4.6 Conclusions

In the e-Science research context, web service is a good distributed computing model for communications between different participants. As web service primarily uses SOAP as its *de facto* message format, it is not efficient to send binary data between applications, particularly for long-distance, high latency network.

Our research work investigates ways in which we could improve performance of long-distance, intercontinental data transfers for web service. In our project, we use CIMA for its flexibility in supporting many instrument types. We must work within constraints imposed both by the networking infrastructure of a given facility, and by the need to achieve real-time data transfer over long distances, in order to support a useful collaborative environment in which researchers can guide the experiment as it is taking place. We observed that the standard SOAP packaging methods used by CIMA web service provide inadequate performance, compared to earlier results obtained using multiple transfer streams.

We have described an experimental investigation of techniques to improve performance in the context of our project, which represents a general e-Research scenario of current interest. In the longer term, it may be preferable to make use of protocols designed for efficient transport of files in binary XML format, such as described in [98]. However with existing file formats, data processing software, and lack of standards in this area, it is not yet feasible to adopt such an approach for most experiments.

Our experiments show that, in the context of web service SOAP communication of large binary files over long distances, it is not efficient to encode the binary file into characters and package them into the SOAP message. In the case of a large SOAP message transferred over a high latency network, the performance can be very poor unless TCP tuning techniques are used. It is necessary to not only tune the TCP buffer size from the kernel level, but also to increase the buffer size at the application level,

which in our implementation involved changing the value in gSOAP (for the web service wrapper) and in the web server (XFire or Apache).

We introduced a new method of transferring binary data between sender and receiver in CIMA, only sending the reference (URL) to the file. There are two advantages here: first, we avoid overloading the client when the data transfer is faster than the processing capacity of the client; second, after the client has the reference(s), it can select when and how to retrieve the data. If the sender provides multiple protocols for data downloading, e.g. GridFTP, HTTP and ftp, the receiver can select the most suitable one.

The client can also use multiple threads in some circumstances. Our results show that a "pull" model data transfer is generally faster than web service data transfer provided by existing CIMA implementations. A larger file can get better performance improvement compared with small files, as a small file needs proportionately more I/O operations for the same amount of data. Multiple threads for data transfer can also increase the total transfer throughput when adopted by the "pull" model, e.g. using HTTP. In short, the use of a "pull" model, the use of multiple data transfer threads, and the use of suitable TCP tuning can each significantly improve the performance of data transfers.

Applying these techniques together can increase performance even more. The current CIMA implementation, can provide real-time data download for our test case only in times of maximum network performance between the US and Australia, however we have observed that the network performance can vary by up to a factor of 10. Using the pull model with multiple HTTP transfers provides a 16 times improvement over the current CIMA implementation, which more than compensates for the observed variation in network performance, and means that continuous real-time data transfer is feasible.

The other interesting point is that the best performance of data transfer with/without TCP tuning is very similar when multiple threads transfer is applied. According to Figure 4.5, the best data transfer rate with TCP tuning and multiple threads is similar to the performance of multiple threads data transfer without TCP tuning: both around 65 Mbits/sec. We believe the reason is when there are multiple threads, even though each of them only has a buffer of the default size (which is often relatively small), as the number of threads is large, the overall buffer size is increased significantly. Therefore,

the network bandwidth can be fully utilized, as with TCP tuning. The final result meets this expectation.

Chapter 5

Web Services Data Transfer with Attachment

In Chapter 4, we discussed data transfer with web service by using SOAP messages. We carried out data transfer experiments under the CIMA infrastructure to improve the overall performance of binary transfer between applications. With these experiments we improve the binary data transfer by using TCP tuning, pull model and multi-thread concurrent HTTP downloading. While these approaches accelerate the overall data transfer speed between different application with low bandwidth, high latency network connection, there are still drawbacks with these approaches, and the most significant one is that we need to introduce other communication models other than web service into the web service architecture. For example, even though HTTP protocol is widely supported for binary data transfer, it will be better if we could have the complete system under the unique web service interface, including binary data transfer.

In this chapter, we are going to describe how we apply web services data transfer with attachment (WS-Att), another specification introduced by the web service community, , to send the binary data between web service applications. Before our work, the performance of implementations based on this specification is unclear. In our research work, we carried out experiments to test the performance of data transfer with web service attachment. In particular, we compared the performance of web service attachment with GridFTP – a well known and often used data transfer tool in the distributed environment.

5.1 Introduction

Web service is based on standard protocols and is widely used as a communication model between applications from different domains. SOAP is the standard format for message exchanging in web service architecture. However, this format is not ideal for binary data transfer, as it requires binary data to be converted into characters and will use more network bandwidth.

We have shown some performance improvements on web service data transfer in previous chapter. These methods include TCP tuning, using pull model to replace push model and applying parallel concurrent downloading via multiple connections simultaneously. While these approaches have brought significant performance improvements for data transfer, we also notice that these steps will make the programming and maintenance more difficult. For example, in previous experiments, we save the data generated by scientific instruments on the server side, and provide data access to these data via http service. A web service is used as the controlling interface for communication and sending meta information (URLs) to clients, so they can organize parallel downloading later. This means we have actually deployed two different mechanisms in the system. First, we need to keep the web service system to send the meta information and also need to maintain a web server to process data downloading requests from clients. This increases the complexity of the whole system.

Research work in [117] has also proposed to send reference information, but as the attachment to the message receiver. The message receiver retrieves the references from the attached information and then retrieves the binary content. These references can use different communication protocols such as HTTP, GridFTP or BitTorrent. The performance of this approach has yet to be fully investigated. Each of these approaches can potentially improve transfer performance compared to standard SOAP messaging. However, there is a fundamental disadvantage in that they introduce extra encoding APIs, and system builders have to deal with two different systems, one outside the standardized web services framework. They also have to set up additional servers, with associated firewall ports.

Different from the approach in chapter 4, the web service attachment has provided the opportunity to keep both meta information exchanging and the binary data transfer under the web service interface, while avoiding sending the binary data via SOAP format messages. We carried out data transfer experiments to study the performance of the web service attachment with Java implementations. The result will be compared with another data transfer tool – GridFTP, which is widely used to enable high-speed transfer of binary files.

5.2 Web Service with Attachment (WS-Att)

The most common format for information exchanging in a web service is SOAP. As a SOAP message is XML encoded and packaged into a SOAP envelope, the binary data has to be converted into characters before they are sent. The most popular approach of transferring binary data is to convert the binary data by using Base64 [119] encoding algorithm, which increases the data size by 33%, and increases the usage of CPU time, memory, and network transfer time.

In Chapter 4, we initially used SOAP messages, the structured message, to transfer all information, the negative impact of this kind of structured message is that it has not provided an efficient way of representing binary data. Although we can use http, sftp or GridFTP protocols for binary data transfer, as we have done with the pull model, however, this approach will increase the complexity of the whole system. For example, if we select http for binary data transfer, we need to maintain two separate subsystems simultaneously at the server side: a web service system for sending the URL information to the client, and a http system to respond to the client request for binary data. At the same time, the programmer also need to write code for two separate systems. This is potentially error prone and inefficient for maintenance.

The web service community also realized the inefficiency of sending encoded binary data via SOAP messages, the SOAP Messages with Attachments (SwA) was introduced to bind the MIME multipart/related message with the SOAP message [85]. With the SOAP message, references are defined to point to the binary part of the SOAP message.

However, with SwA, the attached binary data is not part of the XML infoset, i.e. there are two separate infosets: XML infoset and binary infoset. To keep the consistency of the infoset, XML binary Optimization Package (XOP) [115] has been released. The SOAP Message Transmission Optimization Mechanism (MTOM) is the application of XOP within SOAP 1.2 [113]. With XOP, all content, including the binary parts, are transferred as part of the XML infoset. The binary part is transformed into characters by a Base64 encoding algorithm, or can be optimized as binary content. These parts can be moved to the outside of the SOAP envelop, and pointed to by a reference element within the XML part. This reference element must have an "xop:include" element with a "href" property. The "href" points to the optimized content as a pointer. In practice, the optimized XML content is the original binary content that has never been converted into Base64 encoded XML infoset.

SwA and MTOM have the same wire format, hence are expected to have the same performance. However, conceptually, web service attachments with XOP/MTOM are treated as a complete XML infoset. In a word, all this effort is to make the transferring of binary data more efficient while keep the interface as simple as possible.

The other format of WS-Att is Direct Internet Message Encapsulation (DIME) [108] proposed by Microsoft. As in SwA, different parts of the SOAP messages are separated by using delimiting boundaries, the processing program will need to scan through the whole package to get the desired part, by applying DIME, this problem can be solved. DIME was never an official W3C standard and was superseded by XOP/MTOM specification [5].

Within our research work, we use different web service engines and APIs that support SwA and XOP/MTOM to test the performance improvement brought by web service attachment. These APIs include the libraries provided by XFire[27] and Axis2 [23] that support WS-Att. These two APIs are used as they are both open source projects and are all widely used. By comparing with other popular large data transfer tools, we find that WS-Atts, especially by using concurrent data downloading technology, has provide a very competitive model to support general purpose binary data transfer with web service interface.

5.3 GridFTP

GridFTP is a protocol that extends the File Transfer Protocol (FTP) and provides a general-purpose mechanism to meet the requirement for secure, reliable and high -performance data movement [78, 88]. The Globus striped GridFTP [78] implements a server and client within this framework. According to [78], the testing result has shown that GridFTP can achieve very high speed for data downloading.

The Globus implementation of GridFTP has been widely used in the grid ecosystem. Our research work is to investigate whether web service with attachment using MTOM/XOP provides good performance for binary data transfer, by comparing it with GridFTP, which is known to provide excellent performance.

5.4 Experiment Environment for WS-Att

We now describe our test environment, and other aspects of our tests of data transfer using web service with attachment.

5.4.1 Experiment Location

In order to gain some insight into the effects of different network latencies, we ran experiments between the University of Adelaide and different locations at varying distances. Firstly, within a 100 Mbits/s Ethernet local area network (LAN); then between the University of Adelaide and the University of Sydney, both in Australia (intra-continental), a distance of approximately 1,500km with a round-trip time (measured using ping) of around 22 msec; and also between the University of Adelaide and Indiana University in the USA (intercontinental), a distance of approximately 16,000 kms with a round-trip time of around 225 msecs. For the experiments carried out to Sydney, the connection backbone is the AARNet [1] multi gigabits connection. The connection between AARNet and the Indiana University is multi gigabits shared backbone.

5.4.2 Multiple Threads WS-Att

The experiments are carried out with varying numbers of threads. Each thread (client) builds a connection to the remote web service that can send data to the client. With multiple threads, there will be multiple connections to the service provider and the data transmission is carried out concurrently. Thus, there normally is better performance from these multiple connections operating in parallel, with the server establishing a separate thread for each connection. Usually, we ran an experiment with 1, 2, 4, 8, 16, 24 and 32 threads. When using multiple threads, the sender creates the threads for file sending. A pool of files is also created (for our tests, the same files each time). The required number of files is divided between the number of threads, and different files are sent in parallel from the pool to the receiver. Increasing the number of concurrent data transfers typically improves overall data transfer performance.

5.4.3 Facilities

The computers used for the tests have the following specifications. First, a Linux box at the University of Adelaide with kernel version 2.6.18 as the sender. GridFTP server (version 2.5) is installed on this machine. This machine has a dual core 2.8 GHz Intel Xeon CPU with 2GB memory and is connected with a 100Mbit/s Ethernet card to the LAN in campus. For on-campus testing, we use another Linux machine with kernel version 2.6.19 as the receiver, with GridFTP client (version 3.22) installed on this client machine. It has an Intel dual core 2.13GHz CPU and the memory size is 2GB. The receiver in Sydney is a Linux box with kernel version 2.6.16. It also has 2GB memory with two Intel(R) Pentium(R) D 3.00GHz CPUs. Finally, the experiments carried out between Adelaide and Indiana. The receiver in Indiana is a Linux box, kernel version is 2.6.20, with an Intel(R) Pentium(R) 4 CPU 2.40GHz and 1GB memory. The TCP buffer size used on these machines has the same setting as the default setting, i.e. no TCP tuning was done. In our experiments, the binary attachment is the dominant payload of the whole package and the SOAP part can be ignored. Each test consisted of transferring between 32 files (for large file sizes) and 3200 files (for the smallest file sizes) and

measuring the download time in order to determine the total data transfer throughput (in Mbits/sec). Due to the variability of network performance on the shared networks that we used for our tests, all the tests were run five times, and the results plotted in the figures are the average of the five results.

5.4.4 File Sizes and APIs

We used files of several different sizes for the tests, usually 10K, 100K, 1M, 10M, 40M and 100M bytes.

We have tested different WS-Att specifications using two Java-based implementations. They are XFire [27](version 1.2.2) and Axis2 [23] (version 1.1.1). Axis2/Java is a later version based on Axis, developed by Apache. Many new features have been added and it is claimed to be more efficient. Both of them support the MTOM specification.

5.4.5 GridFTP Experiment Setting

For purposes of comparison, we ran tests, with the same test files and network infrastructure, using GridFTP. For the web services with XOP/MTOM tests, multiple transfers are achieved by having several threads each sending an individual file, that is, several files are sent in parallel. GridFTP transfers have a parallelism parameter which is applied to the transfer of a single file; each thread transfers a part of the one file.

1. The physical location of GridFTP experiments is same as the WS-Att, as shown in section 5.4.1.

2. The version of GridFTP server is 2.5 and the GridFTP client is 3.22. The server is installed on the Linux box, which works as sender, and the client machines have been installed on other machines.

3. With GridFTP we use the file sizes range from 10K bytes, 100K bytes, 1M bytes, 10M bytes, 40M bytes and 100M bytes.

4. Transfers are carried out with levels of parallelism (that is, number of threads) of 1, 2, 4, 8, 16, 24 and 32.

In our previous work, we used pull model to retrieve the data from data provider. The data provider holding the data files, instead of sending the data files directly to the receiver, will send references (often URLs) to the receiver via web service interface. The client side has two concurrent running threads, one dedicated to listening to the server and receiving the references. The other one will take the references and download the data from the service provider. As these two threads are concurrently running, the total extra time for data retrieval is from the server sending the references until the client passes the references to the other thread. With our current experiments, the latter thread will take much longer time relative to the extra time taken in the first thread, as we suppose the data to be downloaded is much larger than the URLs where the data are saved. Therefore, in this experiment, the time taken to receive and process references is negligible compared to data transmission time.

5.5 Experiment Result and Analysis for WS-Att vs. GridFTP

We firstly carried out experiments comparing different kind of WS-Att specifications by using Axis2, which supports SwA, MTOM/TRUE and MTOM/FALSE. Setting MTOM to true signifies that the Base64 binary encoded part will be optimized during the processing of the XML infoset before sending it to the receiver. Here, optimization means that it is sent directly in binary format as an attachment, and not character encoded. On the contrary, by using MTOM/FALSE, the web service will send Base64 encoded binary data to the receiver.

For tests with different web service attachment specifications, we expect to see very similar, if not the same, performance for SwA and MTOM/TRUE. When MTOM is TRUE, the web service program actually sends the binary data directly as an attachment to the receiver, which is the same behavior as SwA. The only difference between them is the representation of the reference part within the SOAP envelop. Figures 5.1, 5.2, and 5.3 show, as expected, that the SwA and the XOP/MTOM has very similar performance.

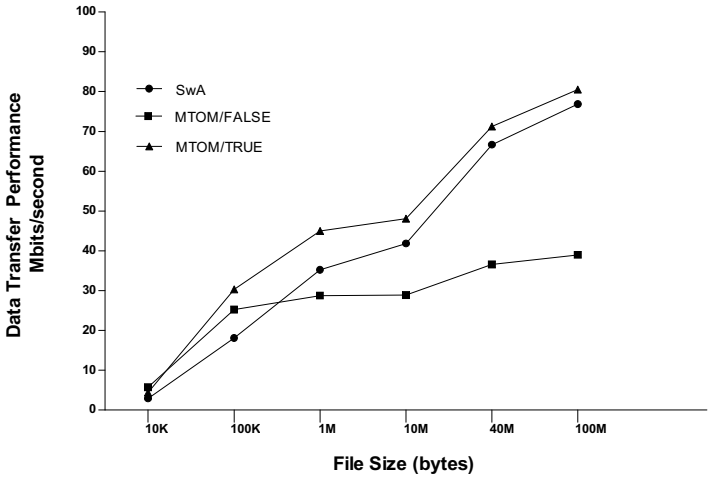


Figure 5.1: Comparison of different web service with attachment specifications over LAN using Axis2

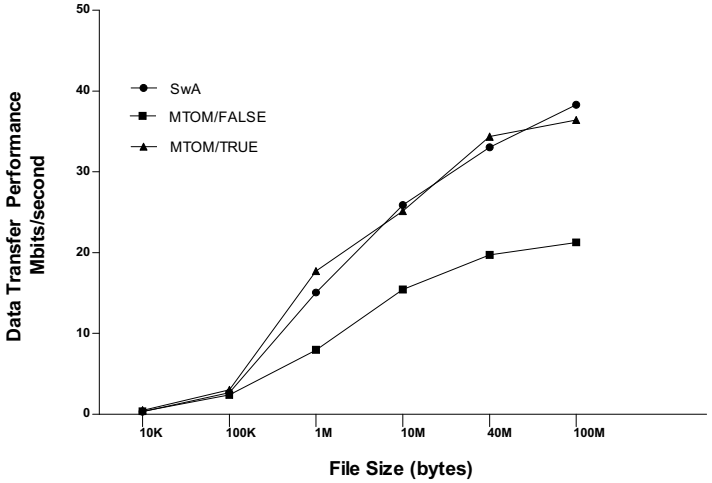


Figure 5.2: Comparison of different web service with attachment specifications over continental WAN between Adelaide and Sydney using Axis2.

The other aspect of these diagram is that, as file size increases, overall performance increases as well. This is partly because file I/O operation constitute a relatively greater overhead for small files, thereby degrading the data transfer performance.

Figure 5.1 shows that in a LAN environment, by running Axis2, the performance comparison between different web service data transfer methods.

Figure 5.2 shows that in a WAN environment, by running Axis2, the performance of web service with attachment by applying MTOM to true.

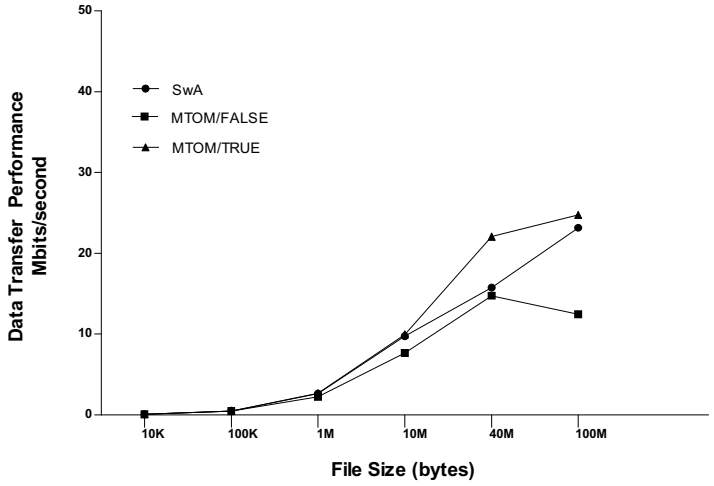


Figure 5.3: Comparison of different web service with attachment specifications over inter-continental WAN between Adelaide and Indiana using Axis2.

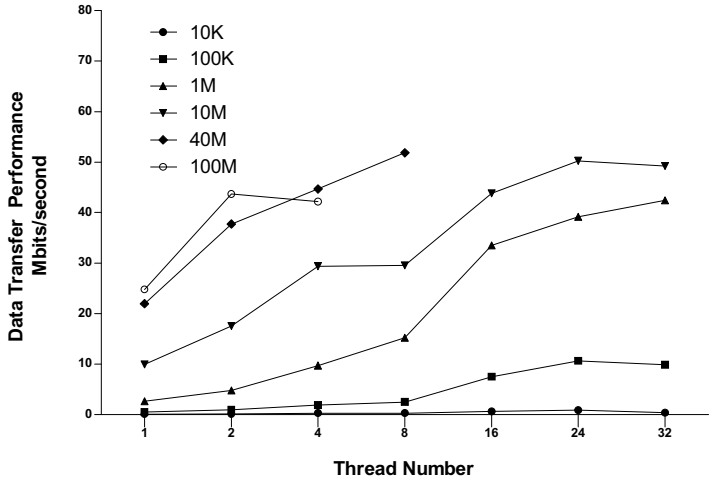


Figure 5.4: Performance of Axis2 Web service with attachment using MTOM and multiple threads on intercontinental WAN (Adelaide to Indiana)

◇ MTOM/TRUE on Inter-continental Connections

We also carried out the experiments within the WAN: inner-continental as well as inter-continental.

Figure 5.4 and 5.5 show the performance, with Axis2 and XFire respectively, of the intercontinental, high-latency link between Adelaide and Indiana. In this case, we use only XOP/MTOM with MTOM set TRUE to cause files to be sent as attachments within the XML info set. We also use various numbers of threads; we expect this to

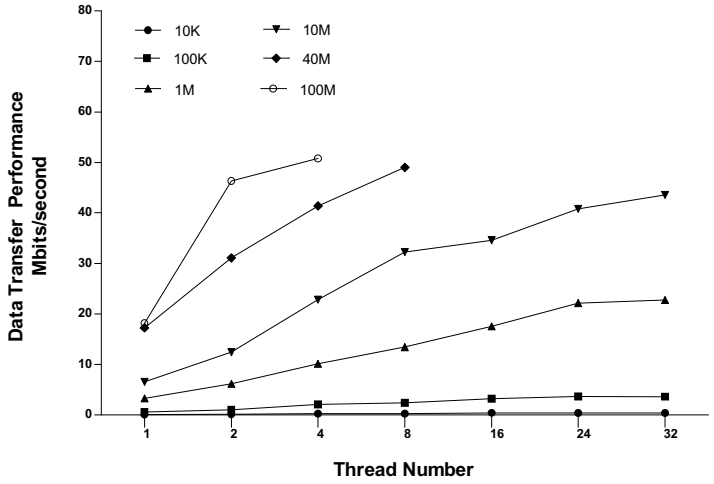


Figure 5.5: Performance of XFire Web service with attachment using MTOM and multiple threads on intercontinental WAN (Adelaide to Indiana)

improve performance, especially as the file size increases. That is because each thread involved in transfer of data will ask the TCP kernel for a separate buffer and build up a separate connection to the receiver. Thus, when there are multiple threads, the total performance of transferring will be increased correspondingly. As before, we see that as file size increases, the performance improves. Even with small files, performance has been increased correspondingly, although the actual transfer performance is still very low and less than 20 Mbits/second. When file size increases to 10 MB, with Axis2 the performance is nearly 50Mbits/sec and with XFire it is above 40Mbits/sec. With quite large files of 40MB and 100MB, initially performance is even better. However, and very soon, it reaches a limit. We were unable to complete transfers with files of size 40MB using more than eight and 100MB using more than four threads. We could not find a way to avoid Java out of memory exceptions. Both XFire and Axis2 have provided file caching mechanism to allow streaming of large attachments on the receiver side, but we have not found a similar mechanism to control the size of memory to be used for sending data on the sender side. When the testing program uses more threads with large files, the system breaks before reaching peak network performance. For smaller file sizes, as the receiver can accommodate a reasonable number of threads, the data transfer performance improves significantly by applying multiple threads.

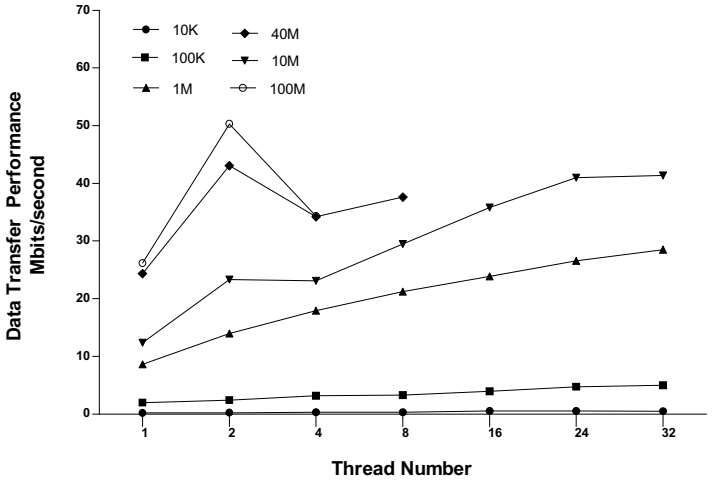


Figure 5.6: Performance of XFire Web service with attachment using MTOM and multiple threads on continental WAN (Adelaide to Sydney).

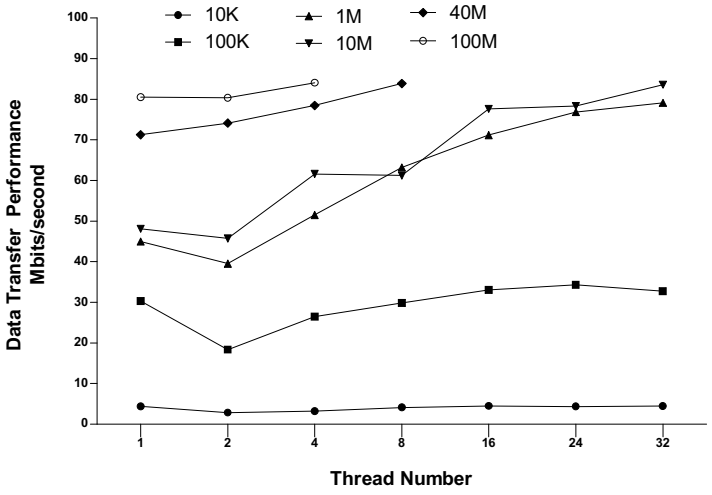


Figure 5.7: Performance of Axis2 Web service with attachment using MTOM and multiple threads on LAN

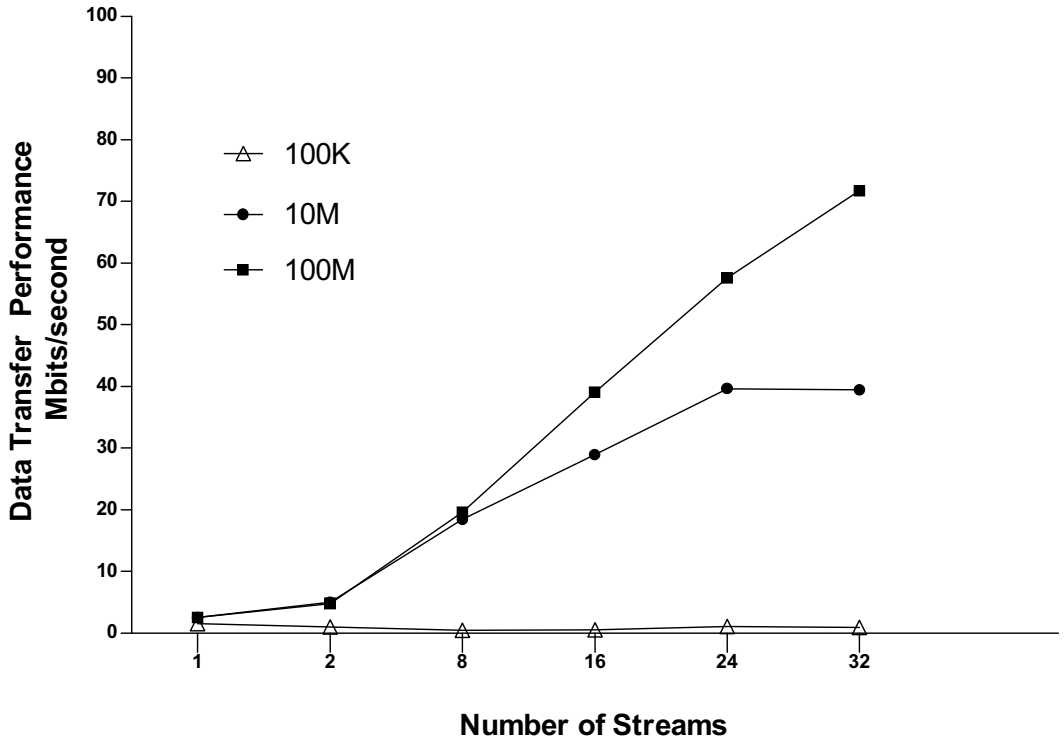


Figure 5.8: GridFTP performance within 100Mbits/sec in local area network

Figures 5.6 and 5.7 show results achieved with similar tests carried out for the continental network (Adelaide to Sydney) using XFire, and over an local area network with Axis2 in the University of Adelaide. Similar trends and limitations are observed, but with slightly higher peak transfer performance for the continental results, and significantly higher for the LAN results, as expected.

Our experiments showed that Axis2 has slightly better performance (usually around 10-20%) than XFire for binary data files using web service attachments with MTOM.

Figure 5.9 shows that within a LAN the performance of GridFTP does not change dramatically as the numbers of streams increases. There is actually a small decrease in performance (about 3% in this case). The reason for that has been attributed to the 'seek' operations on receiver side in [78]. GridFTP sends data in small chunks. Before sending, the original data is splitted into small pieces and sent to the other side via concurrent parallel data connections. On the other end, these small chunks will be reassembled into one unique file. This could improve the performance for large file transfer via wide area

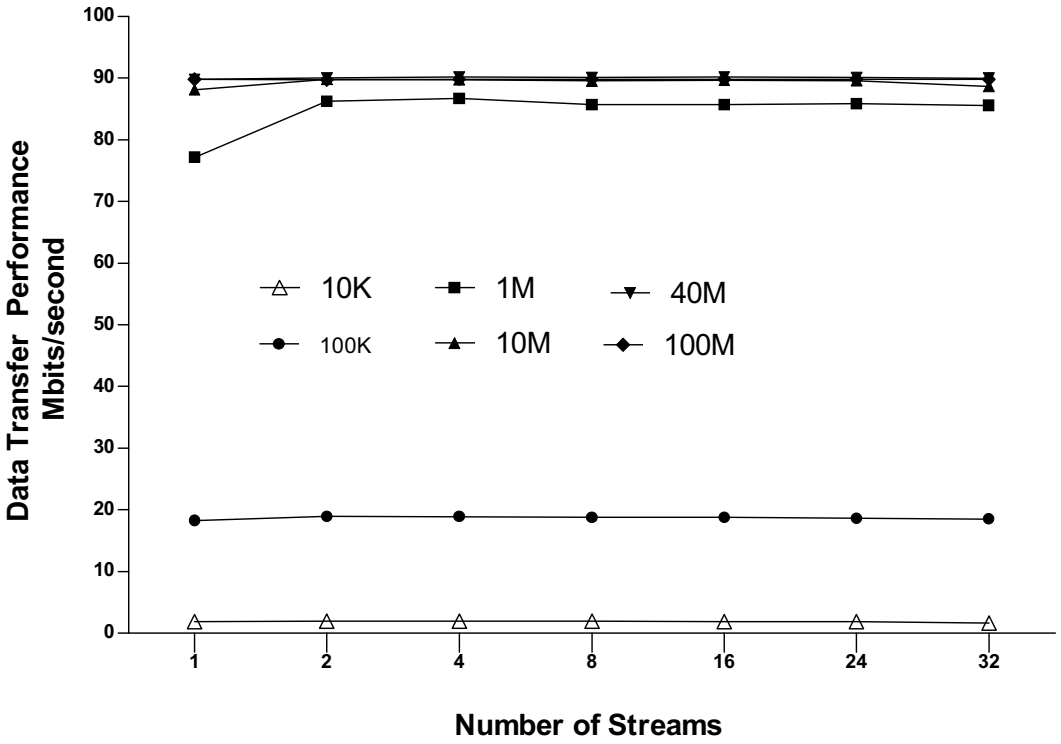


Figure 5.9: GridFTP performance within 100Mbits/sec in local area network

network. However, the performance can actually be degraded in LAN. In a LAN, as the network speed is very fast, time saved by using parallel data transfer is actually less than the time consumed in reassembling the small chunks.

Figure 5.4 shows a performance of over 50Mbits/sec for 10M bytes files when applying web service with attachment, comparing with GridFTP less than 40Mbits/sec performance. However, when files are larger than that, the GridFTP performance increases, and it is faster than any web service attachment implementation for files larger than about 10 MBytes. GridFTP peaks at about 70Mb/sec, compared to 50Mb/s for the web service attachments implementations.

As mentioned above, we also ran tests using GridFTP. Figure 5.8 shows the results of GridFTP transfers carried with varying degrees of parallelism over the inter continental link to Indiana. The performance of GridFTP is significantly better than the web service performance in the case where a single file is downloaded at a time, i.e. where a single thread is used for the web service data transfer, except in the case of small files (less

than 100 KBytes) where the performance of GridFTP was very poor. Where multiple files can be downloaded concurrently using multiple threads, the performance of web services using MTOM is very similar to that of GridFTP, except for large files where Axis2 and XFire run into memory limits.

5.6 Conclusion

Within an e-Science environment, it is desirable to have facilities such as instruments discoverable and usable via web services interfaces. Such instruments often generate large amount of binary data. Sometimes, the data is saved in a few large files, but in most cases, these data are saved in a pool of files. How to transfer such data using standard web service protocols is an important question, especially comparing transfer efficiency with HTTP and other communication protocols.

Our experiments show that the web service with attachment, especially with its XOP/MTOM specification, keeps performance of binary data transfers via web service interface at a reasonable level, especially for large number of small files (less than 10Mbytes). For very large files, GridFTP has its advantages. For a 100Mbits/sec local area network, both of them nearly reach the limitation of the bandwidth, about 80 Mbits/sec for web service with attachment and 90 Mbits/sec for GridFTP, provided the files are bigger than 1M bytes. The overall performance of web service with attachment can be improved by applying multiple threads for concurrent data transfers. More parallel data streams can be used to improve GridFTP transfer performance when transferring big files. Of the most used APIs, Axis2 has slightly better performance than XFire. The advantage of GridFTP is that it has better performance when applied to single files or very large files, but the system builders have to maintain a separate GridFTP system for data transfer. For situations with multiple moderate-sized files (tens of MBytes or less), where multi-threading can be used to improve throughput, Axis2 and XFire implementations of web services with attachments using MTOM give good performance that is comparable to GridFTP, and offer the advantage that data transfer can be done within the web services framework, rather than an external mechanism.

Chapter 6

WSDF Definition

In a distributed environment, data is shared between different nodes of a workflow. To improve the performance of data transfer, we need to consider two aspects of this issue. First, the direct data transfer between two different nodes, i.e. data is directly sent between two applications. Second, select/build the best path for data to be sent from one application to the other one with the minimum cost of time and bandwidth (please refer to 1.2.2 for more discussion on this). In this chapter, we are going to introduce a novel approach to efficiently share data between web service components within a workflow.

6.1 Introduction

In a distributed system, Service Oriented Architecture (SOA) [103] has been regarded as an appropriate framework for distributed components, as they are loosely coupled. Distributed services, e.g. web services, are provided as resources to clients. To accomplish more complicated tasks, different atomic services can be integrated into a service workflow. For instance, different web services can be composed to form a web service workflow.

There are two different types of workflows according to the location of the workflow's control point. If a workflow has a centralized control point, it is classified as a centralized workflow; otherwise, a decentralized workflow. Centralized workflow nor-

mally has a centralized workflow engine, and the workflow engine makes decisions on when and how to invoke the services involved in the workflow. The centralized workflow engine is also the hub to exchange data between different atomic services [26, 52, 127]. Centralized workflows are stable and easy for administration. However, in a centralized architecture, the workflow engine can be the bottleneck of the whole system, especially when there is significant amount of data transferred between services involved in the workflow. This could cause higher network resource consumption and lead to degraded performance of the whole system. For example, often a group of web services for a specific research purpose are deployed within a local network within an institution. Users from different campuses, cities or even continents, utilize these services by invoking a workflow that is composed of these services to process their data. Under this situation, data transfer between different services can be very inefficient in going through the centralized workflow engine.

There are two types of information flows in a workflow, control flow and data flow [122] (see Figure 1.1). The service that is being called is the current service. The service that is after the current service and is to use the result of the current service is called a successor service. Figure 1.1 (a) shows a centralized orchestration [122] model. For each service, the data flow is bidirectional. Input and output data of the web service shares the same network connection. The same applies to control flow. Different services all talk to the same workflow engine and exchange data via this engine. In Figure 1.1 (b), the control flow is still centralized, however, the data flow is decentralized. Data is sent from one service to another without going via the workflow engine and used by the successor service which is invoked later by the control flow. The data exchange between different services is in the choreography model [82], under which the workflow engine can avoid being the bottleneck in the workflow.

Previous research in this area focuses on either extending functional web services with extra capabilities [84, 87] or reconstructing the workflow [122, 126]. These implementations are limited to the application level and the service server does not provide the underlying mechanism for direct data sharing between atomic services of a workflow.

Within a workflow, atomic services are integrated into a composite service, as shown

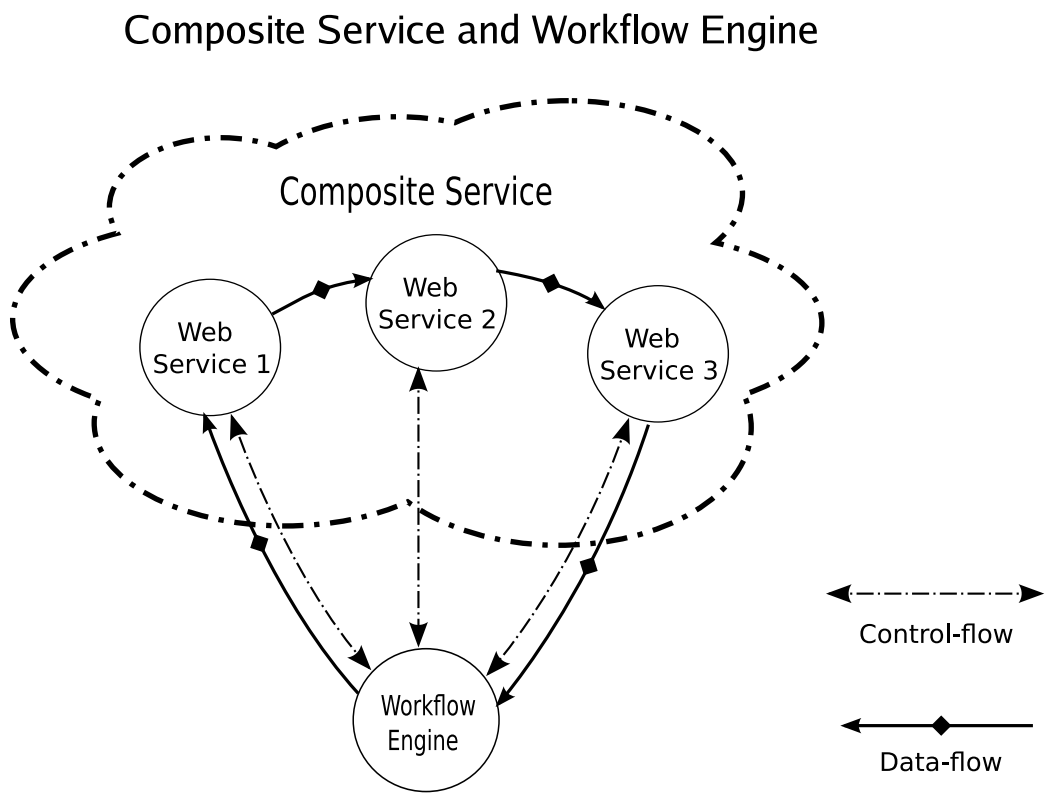


Figure 6.1: Composite Service in Distributed Environment

in Figure 6.1. The composite service acts as a normal web service in that both are invoked by a client, process input parameters and return the result to the client without saving it. In Figure 6.1, there is only one data input flow and one data output flow between the workflow engine and the composite service. Any intermediate result is within the scope of the composite service. By keeping this result within the composite service, we avoid returning it to the workflow engine, therefore, avoiding the overhead of third party data transfer. We argue that the underlying problem for this model is resource sharing across atomic web services. To keep the intermediate result of a workflow, we introduce the stateful workflow concept.

In a stateful workflow, each atomic service is stateful. The de facto standard for representing the state of web services is Web Service Resource Framework (WSRF) [22], which provides a framework such that a compliant web service is stateful and the state information of a particular web service instance is a resource. A stateful workflow keeps the state of the intermediate result of web services and shares them in the composite ser-

vice. What we want is to have one data result generated from the current service to be transported to and saved on the successor service, which is stateful. However, there is no mechanism provided within the web service framework to forward a data result from one service to the other as in the push model, or, alternatively, as in the pull model, retrieve data from current service by the successor service. In current practice, if the successor service is a stateful service, the result data can be forwarded from the current service to the successor service by adding a function from the application level. But this will lead to the situation that the data transfer depends on the specific implementation of that service. New mechanisms for result data sharing between stateful services should be built to free web service workflow developers from being required to implement their own data transfer functions.

We propose the WSDF framework, which is based on the idea of stateful workflow, to allow efficient data sharing between services. In this framework, atomic services involved within a composite service are stateful web services. A WSDF server, built on stateful web service server, hosts atomic services and is responsible for forwarding the result data of the current service to the successor service. The information used by current service server to transfer result data is called *resource forwarding* information. A resource forwarding information schema is also defined. If a client invokes the current service while embedding this resource forwarding information in the invocation request, the server first retrieves this information from the invocation request; after the functional service is finished, it forwards the result to the successor service as specified by the resource forwarding information. The successor service accepts data sent and stores it as a resource before the invocation of this service. Based on these framework principles, we built a complete prototype system to prove the proposed concepts. Comparing with the normal web service workflow system, significant data transfer speed improvement has been achieved by WSDF workflows in long distance data transfers.

In the following session, we will a more complete definition of the WSDF framework.

6.2 Web Service Data Forwarding Framework

We propose the WSDF framework to address the result data sharing issue between services within a centralized workflow by introducing the concept of stateful workflow. Within the WSDF framework, workflow is considered to be stateful because the result data of atomic service is kept within the composite service (as shown in Figure 6.1). A WSDF server forwards result data from current service to successor service according to the resource forwarding information.

6.2.1 Stateful Workflow

Web service workflows are composed of atomic web services. From the client's point of view, atomic services integrated in a workflow can be viewed as a composite service. But there is difference between them when it comes to the execution cycle and the state of the service. For each atomic service, an invocation cycle only involves a single invocation of a service operation. For a workflow, however, there are multiple invocations within an execution cycle. Each invocation represents a unique stage in the cycle, and the status of the workflow changes after the invocation: current service is executed and new intermediate data is generated. In the case of an atomic service, stateful means the state of a specific service instance can be kept; we define stateful workflow, however, to mean that the intermediate data is preserved between successive services in the same workflow. In a stateful workflow, all atomic services need to be stateful and all intermediate data is directly shared between atomic services, as shown in Figure 6.1, web service 1 can directly send intermediate data to web service 2 – a stateful service, without sending it back to the workflow engine.

In this research work, we use WSRF as the basic specification to build stateful web services, as the WSRF based system has provided the necessary mechanisms that can readily provide the functionality to properly support stateful web services in WSDF framework. On the other hand, the WSDF framework is not limited to the WSRF specification, any other specification(s) or model(s) that provide stateful web service can be used to build WSDF framework.

We also considered the question of whether or not it is possible to work with non-WSRF services. We believe that this could be done, but with considerable difficulty. Essentially, the problem is one of designing and implementing a distributed encoding of both the forwarding information between services and of the data that is to be forwarded, as well as providing a mechanism for storing the data on the server side and providing a reference to it, which can be passed between the web services and the client workflow engine. We use WSRF, since it provides a high-level mechanism [22] for managing the data on the server side as state information, in a standardized way that is generally available to web services anywhere on the Internet. In order to provide a mechanism for non-WSRF services, it would be necessary to design an alternative encoding, and in particular to provide support for it within arbitrary distributed web services. As this would be time-consuming and difficult, we focus our work on the more tractable solution using WSRF.

To maintain state in the workflow, the other key issue is to share result data between different atomic services. The WSRF specification enables web services to be stateful, but it doesn't directly address the data sharing issue and it is hard for multiple stateful web services to share their resources in a standard way. If the successor service is a WSRF service, the data can be forwarded from current service to successor service by adding functions from the application level. However, this approach is *ad hoc*, and will depend on the implementation of the forwarding function, which can be written in many different ways. It is desirable for web service developers to have a better standardized programming environment to build their workflows and this is our aim in this work. The WSDF framework implements the data forwarding between services from the server level, i.e., when the control flow invokes the current service with the resource forwarding information, the server, rather than any application service, will take the responsibility to forward the data and this is transparent to the application service.

6.2.2 Resource Forwarding Information

Within the WSDF framework, the control flow of a workflow is centralized. The workflow engine sends a service invocation request to an atomic service while embedding the resource forwarding information in the request message. This forwarding information includes where and how the generated result can be forwarded to the successor service. To separate this message from parameters used by the current service (application service), the namespace, wsdf, is defined to distinguish the resource forwarding information: *http://cs.adelaide.edu.au/2008/05/wsdf*.

An XML schema for resource forwarding information is also defined within WSDF framework, as shown in Figure 6.2. The *targetNamespace* of this element is the wsdf namespace. The element includes *serviceURL* which is the URL of the successor service. The *createOperationURL* and *setOperationURL* are the URLs for creating an Endpoint reference (EPR) of a resource instance and setting a resource with the generated EPR on successor service respectively. An EPR conveys the information for both accessing a web service endpoint and identifying messages sent to and from web services of an individual service instance [124]. The Endpoint reference is defined by the Web Services Addressing (WS-Addressing) specification [124] which specifies XML elements to identify web service endpoints and to secure end-to-end identification in messages.

To forward result data to a successor service as a web service attachment [85], the *attachResourceForward* element is defined. This is especially useful when the current service generates a big data set as result. If the data is saved as a resource in memory, it could exhaust the memory of the server and degrade the server's performance. An alternative is to save the data into a temporary file. The property *FILE_NAME_AS_RESOURCE* is to indicate whether or not the result data is real data or a file reference (i.e. file name) that points to the real data. The property, *ATTACHMENT_FORMAT*, indicates the format used by the server to forward result data. It is the client's responsibility to inform current service if a temporary file name has been saved as a resource by setting *FILE_NAME_AS_RESOURCE*. If this property is set to *true*, the server will treat the

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://cs.adelaide.edu.au/2008/05/wsdf">
  <xs:element name="wsdf_information">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="serviceURL"
                    type="xs:string">
        <xs:element name="createOperationURL"
                    type="xs:string">
        <xs:element name="setOperationURL"
                    type="xs:string">
        <xs:element name="attachResourceForward"
                    type="xs:string">
            <xs:attribute name="FILE_NAME_AS_RESOURCE"
                          type="xs:string"/>
            <xs:attribute name="ATTACHMENT_FORMAT"
                          type="xs:string"/>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Figure 6.2: XML schema for *resource forwarding* information

resource as a file name and forward the file content to the successor service. Otherwise the resource content will be directly forwarded.

6.2.3 Successor Service

In the same way as the current service, the successor service should also be a stateful web service, in this case, a WSRF service. By default, it should provide *setResource-Property* operation to set the content of a resource [22]. It should also provide a *create* operation to create a resource instance of the service and return an EPR that points to the created instance. The web service client can use the default resource operation *setResourceProperty* to set the content of a property, however, to support data sent via web service attachment, a service needs to define separate *setAttachAsResource* operation, as it is not provided by WSRF framework.

6.2.4 WSDF Architecture

The WSDF framework is built on WSRF. It includes web service engine, WSDF client and WSDF services. WSDF inherits its resource management mechanism from WSRF. Every WSDF service is first of all a WSRF service. A functional service within a WSDF framework consumes a resource and generates a response to a client request. It also provides basic services such as creating the resource reference and setting the resource.

Web service engine in WSDF framework. The primary encoding specification for a WSRF Web service is SOAP [113]. A WSDF SOAP engine is an extended WSRF SOAP engine that understands the resource forwarding information. When the WSDF engine receives request message from a client, it reads the resource forwarding information from the request message and saves it temporarily. Then it invokes the functional service. When the service finishes, the WSDF engine checks to see if the result received needs to be forwarded to any successor service. If so, it will create the resource instance(s) and set content of the instance on the successor service(s). Data forwarding between different services within WSDF uses web service invocation to keep the whole system within a standard web service framework. The URLs that are used to compose these invocations should be the ones that the web service server obtained from the request message. If the resource forwarding information contains multiple successor services, the result will be forwarded to each of them separately. The WSDF SOAP engine also supports web service with attachment [92, 85] to transfer large data efficiently.

WSDF application services. A WSRF implementation provides resource management mechanisms. The WSDF application service can utilize standard WSRF resource management functions such as *setResourceProperty*. It needs to implement the *create* interface for creating a resource instance and the *setAttachAsResource* interface to support Web service with attachment.

WSDF client. The WSDF service client composes a WSDF request based on a basic WSRF request. The difference is that if a WSDF service client needs the result to be forwarded to the successor service, it should know the URLs to create and set resource on the latter service. These URLs are used to compose the resource forwarding message

element as part of the SOAP envelope of the service invocation request.

In the following chapters (chapter 7 and 8), we address the implementation and performance testing of the WSDF framework. In chapter 7, we give details on how we built a prototype WSDF workflow engine and a simple WSDF workflow. In chapter 8, we illustrate and analyze the performance improvement achieved with WSDF workflow.

Chapter 7

WSDF Implementation

The WSDF framework is designed on a centralized control flow model, which provides the maximum control of the workflow execution for the client. At the same time, based on the stateful workflow concept, it provides an environment in which data sharing between different services can be carried out in a distributed way.

In previous discussion (section 3.2.2), we illustrated basic elements within a workflow system. These elements include workflow engine, service providers (e.g., applications run on servers), and clients of services. A workflow (e.g. web service workflow) composes these elements together to form a new workflow system. In this project, as there is no existing WSDF service engine, WSDF services and clients, we need to build them for this research work. Further, we compose a WSDF workflow based on these components.

In this chapter, we give the details of the WSDF framework implementation by addressing the details of implementing the workflow engine, WSDF service and WSDF client in the context of stateful web services. Among all of these components, the WSDF service is the primary component we try to implement.

WSDF services are provided by stateful atomic web services, therefore the WSDF workflow engine should support stateful web services. We choose WS-core [45] as the stateful web service server to build the WSDF service server. WS-core is the web service server of the Globus [29] software toolkit, which implements WSRF and has been widely used by the research community. By default, the WS-core does not support

web service with attachment as resource. Our WSDF implementation has extended its functionality to support web service with attachment to improve large binary data transfer speed for computational services. Then, example WSDF services and WSDF clients are created. Finally, we compose a WSDF workflow based on these components.

7.1 WSDF Server Implementation

WSDF service engine is a SOAP based web service engine that supports stateful web services. As a SOAP processor, the WSDF server follows the Resource Forwarding Information schema defined by the WSDF framework.

In our implementation, we built the WSDF web service server based on WS-core, which is built on Axis1.1 [3], which is a commonly used SOAP web service engine. Based on the source code obtained from the WS-core project, we built the SOAP engine for the WSDF service. We refer to the modified SOAP engine as WSDF-axis. The WSDF web service extends the functions of a WSRF server and provides the ability to interpret resource forwarding information embedded in the invocation request message, save this information temporarily, and forward result data after it is generated by a back-end application program. After data forwarding, result data is saved as a resource on the successor service and referenced by an Endpoint Reference (EPR) generated by that service. The EPR will be returned to the current service and finally to the client (workflow engine) for the next invocation in the workflow.

If the result data is sent to successor service running on a different WSDF server, the whole system implementation uses a *push* model, since result data is sent to the successor service. The WSDF server can also perform well in a *pull* model if the client set the forwarding information to the current server, then the result data will be saved as a local resource for future usage by different servers in a *pull* model. Of course, this will need to extend the current WSDF server to support data retrieving requests from the successor service.

The WSDF server also supports forwarding intermediate data to multiple successor services . If there are multiple destinations within the resource forwarding information,

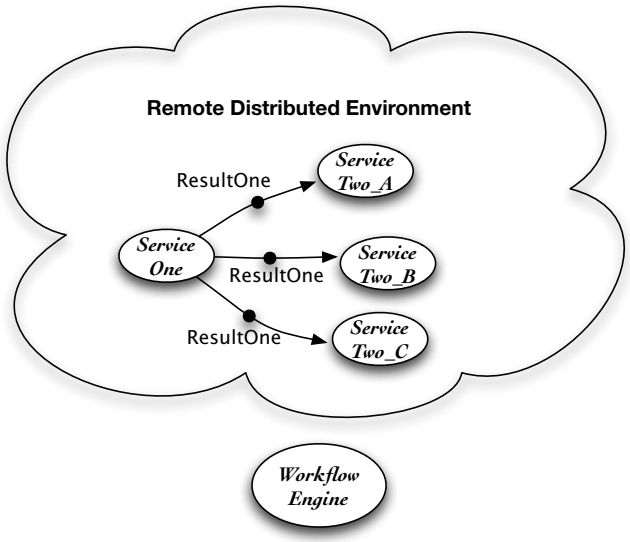


Figure 7.1: Multiple Destination Data Forwarding

the server can forward result data to different receivers. In Figure 7.1, *Service One* has generated *ResultOne* as the intermediate data of the workflow, as the resource forwarding information contains three result data destinations: *Service Two_A, Service Two_B and Service Two_C, ResultOne* is forwarded by *ServiceOne* to all of the destinations.

In particular, if the information contains information of both the successor service and the current service, then both *pull* and *push* model could be supported simultaneously. Even though it is not implemented in our current work, but this can be easily supported.

Finally, if the client does not specify any resource forwarding information in the request, the WSDF server will return the result directly back to the client as a normal web service.

7.1.1 WSDF as a SOAP engine

WSDF framework provides the result data forwarding functionality from the SOAP envelope level. Therefore, the extension of the WS-core also takes place on the SOAP level.

There are two major differences between a WSDF SOAP engine (referred to as WSDF engine) and a WS-core SOAP engine (referred to as WS-core engine).

First, WSDF engine needs to process extra resource forwarding information. With the WS-core engine, after setting environment context according to the resource specified in the SOAP header, it directly invokes the back-end application according to the information it retrieved from the SOAP body. The WSDF engine will need to retrieve the resource forwarding information from the SOAP body, and save it temporarily before invoking the back-end application. The resource forwarding information is saved into a linked list or similar data structure in the current service instance for later usage.

Second, WS-core engine only sends the result data back to the client, the WSDF engine needs to forward the result data to the next service provider. The WSDF engine needs to compose and send the resource creation and resource storage invocations to the next service. Different from a static web service invocation which often uses web service stubs generated from web service WSDLs, these invocations are created on the fly by using information retrieved from the saved resource forwarding information. The returned value for the WS-core is also different from the WSDF server. WS-core returns real data. The WSDF server sends EPRs that pointing to the resources back to the client if the intermediate data is forwarded.

7.1.2 Attachment Support for Resources

While the WSDF engine forwards the result data, we need to consider how to forward the binary result data back to the client efficiently. With our WSDF implementation, we provide two data forwarding choices.

First, embedding data in the SOAP envelope. Second, forwarding the data as attachment in binary format. The server is implemented to be able to support either way to forward the result data. But the choice of using which format, as we believe, should be under the control of the client, i.e. the client/workflow should make the decision according to the environment, including the functionality of the *successive service*. When the *current service* sends the intermediate data to the *successive service*, if the latter

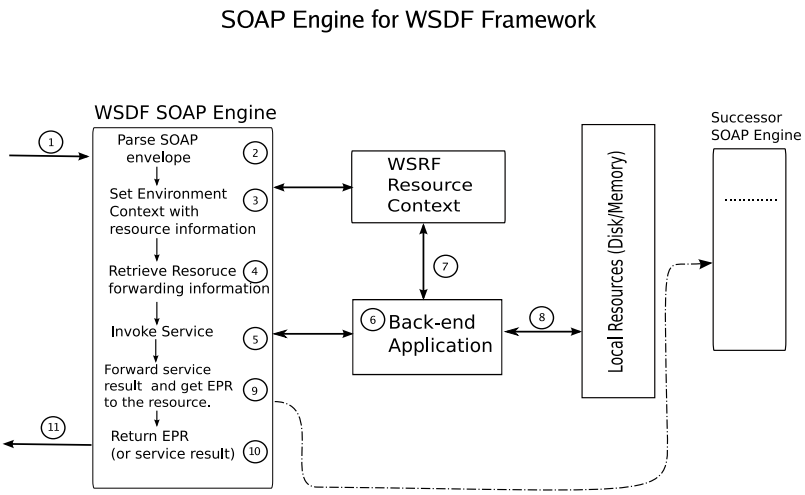


Figure 7.2: SOAP Engine Service Invocation In WSDF Framework

is implemented to save the attachment of the request as resource, then the attachment approach should be applied; otherwise, use the normal approach.

The WSDF framework should also provide a way for the client to inform the server which way it has selected. As shown and briefly discussed in section 6.2, we introduce an attribute *ATTACHMENT_FORMAT* of the *attachResourceForward* element in the data forwarding XML schema. If this attribute is not empty, that means the client wants the server to forward the intermediate result to the *successive service* as attachment. Further more, the content of the *ATTACHMENT_FORMAT*, such as MTOM, specify what specification the attachment should follow.

In our implementation, when the WSDF engine reads out the attachment related information, it will save them in the context of the current service instance, particularly the *ServletContext* (as on the Axis engine). Either the service application and the WSDF engine can get the information by querying the context.

7.1.3 Procedures in WSDF engine

Figure 7.2 shows the internal procedures of how the WSDF-axis engine works when a WSDF service is invoked on the server side.

Steps in Figure 7.2:

◇ Step 1: input SOAP message including resource forwarding information, resource

information and service information. The resource information refers to the resource that to be used by current service, that has been saved on current server.

◇ Step 2: parse SOAP envelope.

◇ Step 3: set the context of current service instance. The context refers the resource context that required to get the resource content.

◇ Step 4: retrieves resource forwarding information from the SOAP message envelope and saves information temporarily for future usage. It could contain information for multiple successor services, as shown in Figure 7.1 (SOAP Request with resource forwarding Information).

◇ Step 5: invokes the back-end application.

◇ Step 6, 7 and 8: by interacting with the WSRF resource context created in step 3, the back-end application retrieves the resource property value of current service instance, carries out the functional processing and finally resets the resource property in the context. The result is then returned to the SOAP engine.

◇ Step 9: SOAP engine carries out the resource forwarding task. It first checks the resource forwarding information. If there is information about one or more successor services, the result will be forwarded to each of them respectively and saved as a resource. Two steps are involved within the creation and setting of the resource. First, the SOAP engine sends a request to successor service to create a new resource instance on that service and receives the EPR of that instance. Second, the engine sends a second request to the successor service to set the result as resource referred by the previous created EPR. The SOAP envelope within the request is composed by the SOAP engine using information that contained within the resource forwarding information. No web service stub is needed.

◇ Step 10: if the result from current service has been forwarded to successor service, returns EPR to the client; else, returns the result to client.

Based on the functionality provided by the WSDF server, programmers can build the WSDF services run on the servers.

7.2 WSDF Service

WSDF is built on atomic stateful web services. In our implementation, the WSDF service is first of all a stateful web service, which provides the functionality to process the resource provided by the client or other services. We compare the difference between WSDF services, normal web services and stateful web services.

7.2.1 Extra Operations

A WSDF service is first of all a stateful web service. It is different from a normal web service, in which only uses the parameters passed in by the client and processes these values directly. For stateful web services, there are two different types of parameters, one is the normal parameter, directly sent from the client as in the normal web service; in addition, there is extra resource information that is stored in the request, which represents a different set of parameters that has been saved on the current service side and is to be used by this service. These two types of parameters are represented and stored differently: this is best represented by using stateful web services.

A WSDF service provides two extra operations besides the computational service. First, a *createResource* operation, for the client of a WSDF service to create a resource instance on that service before the functional service is called. By invoking the *createResource* operation, a resource instance is created for this service and an EPR (details can be found in section 7.2.2) will be returned to the client. Second, a resource property setting operation *setAttachAsResource*. The operation for the client to set the resource *SetResourceProperty* [100], as a default operation for WSRF service, can be used to set the content of a resource. However, to enable the WSDF service to support web service with attachment for binary data transferring between different services, a separate resource property operation *setAttachAsResource* has to be implemented by the service provider according to their specific implementation for the service. An invocation to

```
<ns1:RgbWSDFAttReference
    xsi:type="ns2:EndpointReferenceType"
    ......
    xmlns:ns2="http://www.w3.org/2005/08/addressing">
  <ns2:Address xsi:type="ns2:AttributedURI">
    http://129.127.10.133:8010/wsrf/services/RgbWSDFAttService
  </ns2:Address>
  <ns2:ReferenceParameters
    xsi:type="ns2:ReferenceParametersType">
    <ns1:RgbWSDFAttKey>25626358</ns1:RgbWSDFAttKey>
  </ns2:ReferenceParameters>
</ns1:RgbWSDFAttReference>
```

Figure 7.3: Endpoint Reference (EPR) example

the *setAttachAsResource* operation has the binary data to be sent embedded in the attachment part of the invocation. When the operation is invoked, it gets the data from the attachment of the invocation message and sets it as the resource. If the attachment is large, as it is in our example workflow, the *setAttachAsResource* operation is implemented in the way that it saves the attached data in a temporary file and location of the file is saved as the resource property.

7.2.2 Endpoint Reference (EPR)

The *createResource* operation creates an EPR, which is a reference that points to a created resource on a stateful web service. Figure 7.3 illustrates an EPR example pointing to a resource instance of a WSDF service implementation.

There are two elements within each EPR. First, the URI which indicates the location of the web service, in this example it is:

http://129.127.10.133:8010/wsrf/services/RgbWSDFAttService

Second, a reference parameter, which in this example is called *RgbWSDFAttKey* (which stands for *Key of RGB WSDF service with Attachment support*) with a uniquer integer value (25626358, in this case), which is used for distinguishing the current data resource instance from other resource instances of the same service.

7.3 WSDF Client

A client in the WSDF framework is responsible for invoking individual web services. If the client is actually a workflow engine, then it also needs to coordinate different individual web service clients in the workflow to share the intermediate data between them.

When the workflow engine acts as a client, it invokes the current WSDF service, and needs to embed the *resource forwarding information* into the SOAP message that is used to invoke the functional service.

In a normal web service invocation, users of the web service are either provided with a client side stub by the service provider, or they can generate the stub by using the stub generating tools by web service application APIs. The stub generating tools process the WSDL of the target web service and generate language specific client side stub code. For example, the Axis and the gSOAP SOAP servers provide client side code in Java and C/C++ respectively.

In WSDF framework, the WSDL has been extended to support the resource forwarding in WSDF workflow. The existing stub generation APIs can not be applied to generate corresponding client side code. Currently, when the WSDF client sends an invocation request to the target service, developers need to compose these SOAP envelopes explicitly in the client applications. In the future, stubs could be built automatically from WSDL by using a WSDF-compliant stub generator, as with current tools to easily convert applications to web services, which will benefit developers.

At the same time, the WSDF client is actually behaving as the workflow engine, the resource from a single client can be reused by a second client within this workflow engine. The workflow engine needs to be able to read out the EPR of the resource and pass it to the next client for usage.

7.4 Building WSDF workflow with WSDF Framework

In our research work, we developed a simple example to demonstrate the data forwarding mechanism within the WSDF framework. This workflow is called RGB WSDF workflow. To test the performance of the system, we also built a normal web service workflow, which provides the same functional service but composed of normal web services that have to pass all data via the client..

7.4.1 WSDF Service Operations

The service used to compose workflows is called RGB (The name is an acronym stands for red, green and blue). Both the WSDF RGB service and the normal RGB web service take the content of a *.bmp* image file as input and changes the color of the pixels in the file, as shown in Figure 7.4.

The WSDF RGB service provides *createResource, setAttachAsResource* and *convert* operations. The *createResource* and *setAttachAsResource* operations are implemented to meet the WSDF specification. The *convert* operation is to take a given file, as the attachment of a request, and convert the color of the pixels in the file: red to green; green to blue and blue to red. When the request for convert operation is received, the server will process the previously saved resource data and generate corresponding result. The same services are deployed on three WSDF servers.

7.4.2 Workflow Processing Steps

For our test, there are total three WSDF compliant servers, each server has a RGB service deployed. A workflow which is composed of three segments of service invocations is built to carry out the computation. The task of the workflow is to have a 16 bit .bmp image processed by three RGB services (RGB_A, RGB_B and RGB_C) one after another and return only the final image back to the client. The returned image should be identical to the original file. There are three parts within the workflow, each part contains a RGB service client, which is responsible for invoking the service on one server.

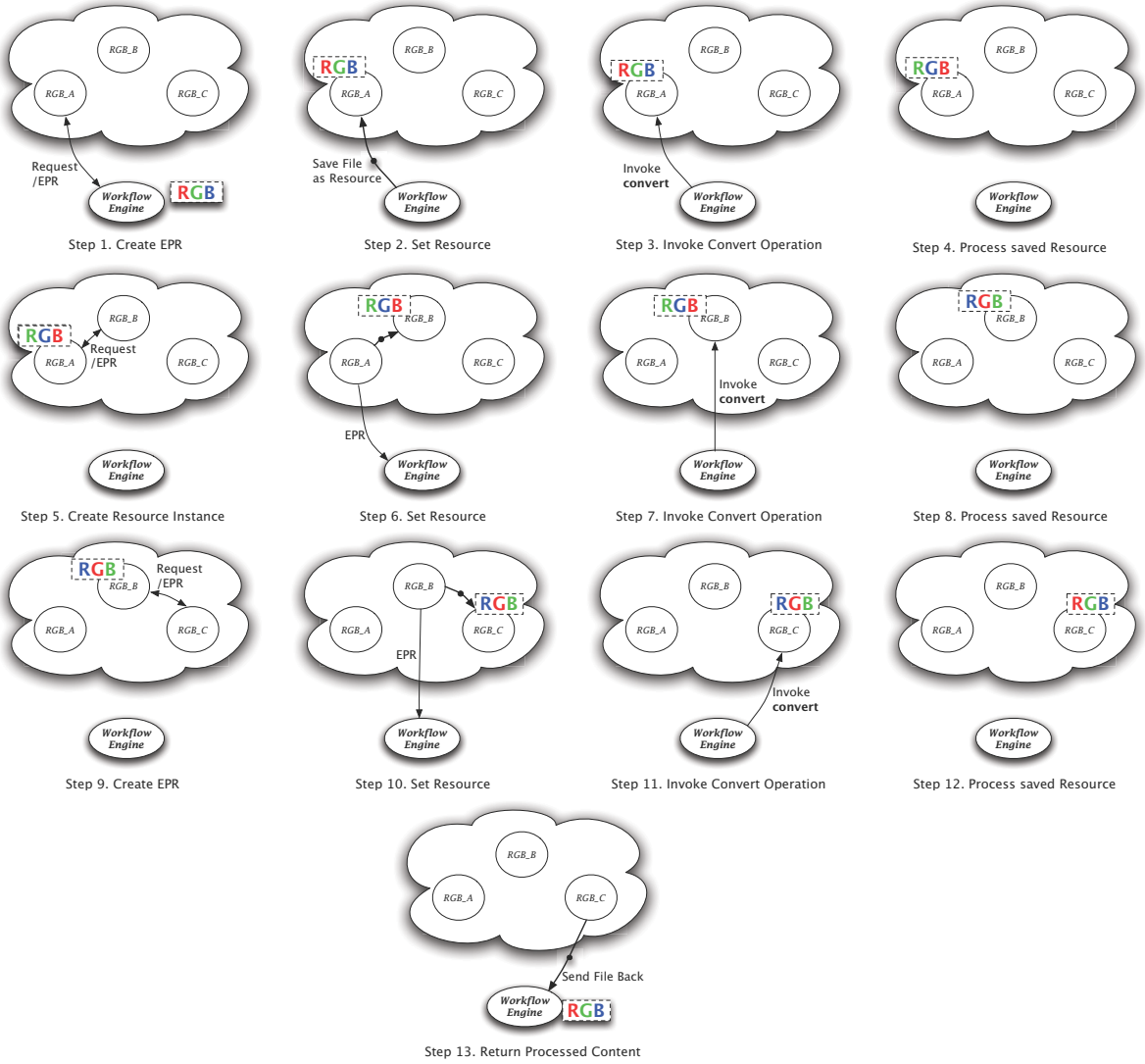


Figure 7.4: RGB Workflow Steps

1. In the first part, as shown in Figure 7.4, from step 1 to step 6, the client first invokes the *createResource* operation and generates a resource instance on the server side (step 1). An EPR pointing to that data resource instance is returned to the client. Then, the client invokes *setAttachAsResource* operation to send the .bmp file as a web service attachment to the server. The server saves the content of this attachment into a file and set the file name as the resource. As shown in step 2, the solid circle in the middle of the arrow represents the real data (.bmp file) is sent as an attachment of the *setAttachAsResource* request. Finally, the client invokes the convert operation (step 3 to step 6). Before the convert invocation, the

client needs to be aware of the following:

(a) The current client invocation is part of the whole workflow, and the data generated from current service will be the input of the next RGB service in the same workflow. The client needs to pass the forwarding information of the second RGB service to the current service. So the client needs to composed this information into the *resource forwarding information* part of the convert operation request, as shown in Figure 7.5. In the diagram step 3, we use unfilled circle to represents the resource forwarding information.

(b) The returned value is not the binary data in the .bmp file, rather it is the EPR that referring to the created resource.

During the conversion, the operation will retrieve the file name from the resource, read in the binary content of the file and convert the red, green and blue colors respectively, as shown in step 4. Then write the result back to another temporary file and save this new file name as the resource. The file name is returned to the SOAP engine for it to forward the file to RGB_B . After forwarding the result as resource of RGB_A, the SOAP engine will return the endpoint reference of this resource to the client and the first part of the workflow finishes. The two steps involved are shown in step 5 and 6.

2. In the second part, the service on RGB_B is invoked (step 7 to step 10).

The workflow engine is the centre for information exchanging between different clients of the web services. As the second client gets the EPR of data resource on the second service, it does not need to invoke createResource and setAttachAs-Resource operations, which have been carried out by service RGB_A. Instead, it can directly invoke the convert operation on RGB_B. Meanwhile, as the workflow needs the third service to process the result of the second one, the client needs to encode the third service information into the convert request as resource forwarding information, so the second server can forward the result generated from RGB_B to RGB_C. As with the first service, the EPR of the resource on the third service will be returned to the second segment of the workflow and passed on to

the third part in the workflow.

3. In the third part of the workflow, the client will invoke the convert operation directly as the resource required for that operation has been saved on RGB_C by RGB_B (step 11 and 12). If this is the last step in the workflow, and the client does not want the result generated by RGB_C to be forwarded to another service, it can invoke the convert operation directly without composing any other forwarding information into the request. As or more accurately , the WSDF server on which RGB_C resides, can not find any resource forwarding information in the request envelope, after carrying out the operation, the WSDF engine will return the result data directly back to the client as web service attachment (step 13).

On completion of the whole workflow, the client will get a converted .bmp file. Figure 7.5 shows a segment of request code with web service resource forwarding information. The XML element with the name *forwardInfo* is defined within the wsdf namespace and contains the necessary information for the WSDF server to compose a SOAP envelope on the fly and forward the result data.

The value of <wsdf:namespace>, http://rgbwsdfatt.com, is the namespace of the successor service to be invoked for data forwarding, and the serviceURL of the successor service is

$$http://129.127.10.133:8010/wsrf/services/RgbWSDFAttService$$

The URLs for createResource and setAttachAsResource operations are given in the wsdf:createOperationURL and wsdf:setOperation URL respectively.

The wsdf:attachmentResourceForward element indicates the data should be forwarded to the successor service(s) as web service attachment. This element has two attributes *fileNameAsResource* and *attachmentFormat*. FileNameAsResource shows the resource property on current server is going to be saved as a file and file name is saved as the resource property. Therefore, for the WSDF server to forward the data, it should not forward the resource directly, but to take the resource property as a file name and

```
   .......
<ns1:convert xmlns:ns1="http://rgbwsdfatt.com">
     <inRgbWSDFAttStr xsi:type="xsd:string"
          foo
     </inRgbWSDFAttStr>
     <wsdf:forwardInfo
        xmlns:wsdf="http://cs.adelaide.edu.au/2008/05/wsdf">
        <wsdf:namespace>
             http://rgbwsdfatt.com
        </wsdf:namespace>
        <wsdf:serviceURL
             http://129.127.10.134:8010/wsrf/services/RgbWSDFAttService
         </wsdf:serviceURL>
         <wsdf:createOperationURL>
              http://rgbwsdfatt.com/RgbWSDFAttPortType/createRgbWSDFAtt
        </wsdf:createOperationURL>
        <wsdf:setOperationURL>
             http://rgbwsdfatt.com/RgbWSDFPortType/set
        </wsdf:setOperationURL>
        <wsdf:attachmentResourceForward
             fileNameAsResource="true" attachmentFormat="MTOM"/>
      </wsdf:forwardInfo>
      <wsdf:forwardInfo
        xmlns:wsdf="http://cs.adelaide.edu.au/2008/05/wsdf">
          <!-WSDF information of the second successor
               service->
          ........
      </wsdf:forwardInfo>
      ......
</ns1:convert>
```

Figure 7.5: Request with Resource Forwarding Information

```
.......
<ns1:convert xmlns:ns1="http://rgbwsdfatt.com">
     <inRgbWSDFAttStr xsi:type="xsd:string">
        foo
     </inRgbWSDFAttStr>
</ns1:convert>
```

Figure 7.6: Request Without Resource Forwarding Information

forward real data from the file. The attachment format is shown in the other attribute attachmentFormat which is *MTOM* in this example. If there are multiple successor services, the client can attach corresponding forwardInfo about these services and the server will forward the data to them as well.

Not all services need the result data to be forwarded (e.g. if the current service is the last service within a workflow). Figure 7.6 is a request without any resource forwarding information . As there is no forwarding information, it does not have wsdf:forwardInfo element in the SOAP envelope as shown in Figure 7.6 which is only a normal web service invocation request message. With this request, the WSDF-axis engine will simply return the generated result back to the client as a normal web service server does.

In this chapter, we gave the implementation details of the proposed WSDF framework. In the next chapter, we will describe the tests we carried out based on the WSDF implementation and compare the performance of WSDF workflow with normal web service based workflow.

Chapter 8

WSDF Testing

In previous chapters, we have introduced the proposal and implementation of WSDF framework. The next step in our research work was to compare the performance of our implementation of WSDF with normal web service workflows to see if WSDF can improve data transfer performance.

Web service workflows are carried out in a distributed environment. Within different circumstances, the performance will vary for the same workflow (e.g. LAN vs. WAN). Even if the workflow is run in WAN, the performance will be significantly different, comparing the the performance in a city scale with the one an inter-continental environment. Further, the distributed environment, in which the workflow runs, can change dramatically from time to time (e.g., the Internet), therefore, the performance of the whole workflow can vary significantly. To avoid these uncertainties, we build our simulation environment to carry out the experiment, which will allow us to run experiments under a variety of scenarios with different specified network performance.

A simulation environment is important to reflect the character of a distributed environment, and more importantly, it can be used to reproduce the outcome and guarantee the environment is stable. The Internet environment is fundamentally dynamic, which means an experiment carried out at some stage, is unlikely to be run under the same conditions if it is repeated in the future. A simulated environment is ideal to provide a constant, reproducible environment, in which the parameters can be set as desired. The impact of the variations in the environment will be avoided.

The other characteristic of the testing work is that our testing is not only based on traditional distributed environment, but also carried out in the cloud environment. Cloud computing is new computing model emerged in recent years. Cloud computing provides large scale, flexible IT related services via web or web service interfaces. The advantage of the cloud has attracted more and more researchers in the distributed computing area, including the workflow users, to carry out their tasks in the cloud. We believe the design of the WSDF framework matches the characteristics of the cloud very well. Therefore, we are expecting to get very positive results from the WSDF experiments in the cloud as well.

For the first step, we will carry out the experiments in the simulated environment and analyze the performance. In Chapter 9, we carry out the experiments in the cloud environment. Finally, we can compare the result from the cloud and the result from the normal distributed environment.

In this chapter, we will first address the testing methodology in our experiments, then discuss the tests carried out in the simulated distributed environment.

8.1 Testing Methodology

8.1.1 Basic Service Time Consumption (BST)

One purpose of this test is to compare the performance difference between a WSDF workflow (see Chapter 6) with a normal web services workflow. From the previous analysis, we have seen that the WSDF framework can save data transferring time between different services within the workflow by distributed data transfer. However, the functional services between the WSDF service and normal web service will be the same. We use Basic Service Time (BST) to eliminate the variation in processing time for the computational service. The processing time differs according to the service selected, but for a particular service, the execution time for the computation will be the same in each approach. The BST refers to the time taken by the service for the computational work, which is algorithm specific, and excluding any input/output operations. In our analysis,

the comparisons between WSDF and normal Web service workflows are based on their execution time excluding the BST.

8.1.2 Distributed Environment for Testing

8.1.2.1 WAN environment

The performance of Internet varies from time to time. To better evaluate the WSDF framework, we would like to carry out the performance test in a reproducible way within a controlled, simulated environment. We use WANem [43] to build a WAN emulation environment on which to run the experiments. The emulator is set as a gateway between client and servers, as well as gateways between servers. By applying different configurations, different network routes in the gateway, we can test the performance of two workflows under different simulated environments.

Figure 8.1 shows the emulation of a network environment in which three web services are located within a single local Ethernet network and the workflow engine is remotely located to these web services in the Internet. To achieve that, we install five Linux boxes in a single Ethernet network. The WANem Gateway, as shown in the Figure 8.1, is a Linux box that runs the WANem software. Host 1, Host 2 and Host 3 represent the servers that host the services. The solid lines represent the real network connections in the Ethernet. The connections between service providers and the workflow engine go via WANem Gateway (gateway), in order to emulate the remote connection as shown by the dashed line. We can change the network settings on the gateway to emulate different network conditions. For example, if we want to emulate an inter-continental connection between the workflow engine and the web service providers, we can set the delay time on the gateway to half of the round trip time (RTT) between the client and the service provider, then add the gateway to the route of both providers and the workflow engine. We can also configure the bandwidth between the different participants in the workflow by configuring the gateway.

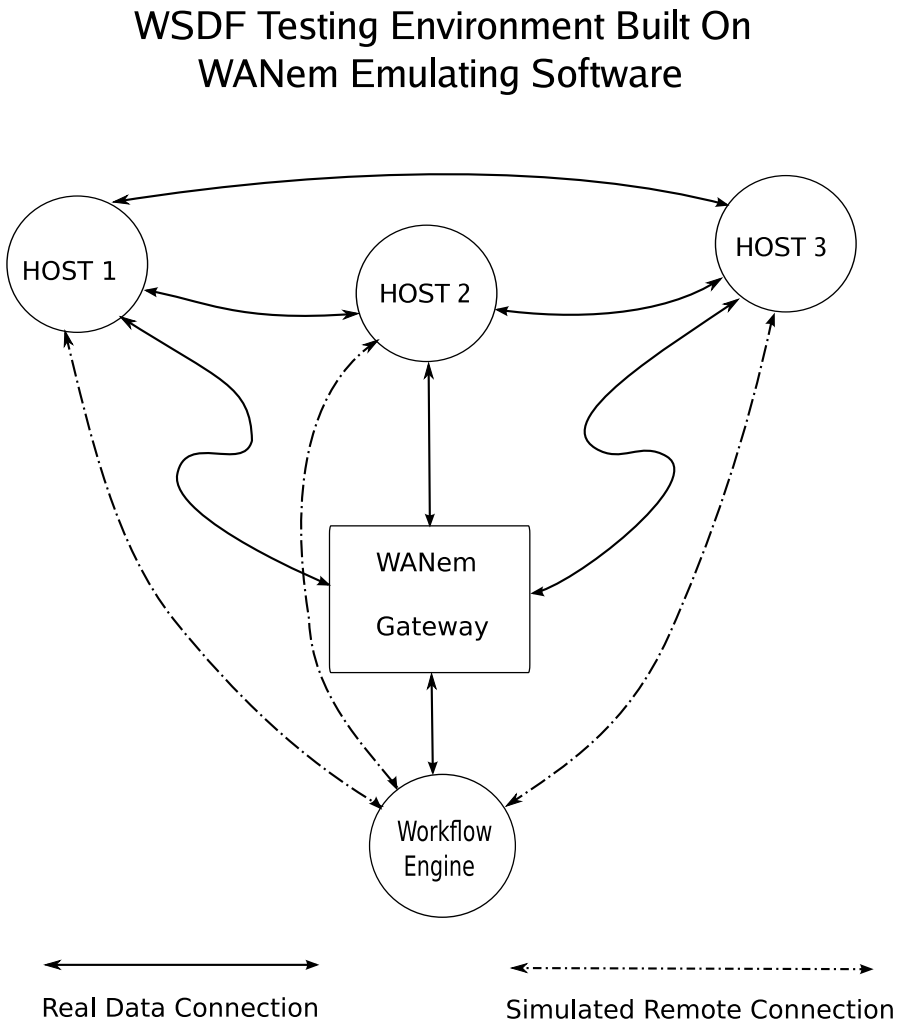


Figure 8.1: Remote Network Emulation with WANem

8.1.2.2 Cloud Environment

There are different types of cloud environment, but to compare the WSDF performance with its performance we get from the simulated distributed environment as shown in section 8.1.2.1, we need to use the Infrastructure as a Service (IaaS) cloud.

We use the *Science Clouds* [40] which is a public cloud provider for academic community and cost free. The *ScienceClouds* is an instance of Nimbus cloud management software [40], hosted by the University of Chicago. It runs Nimbus cloud management software to provide compute cycles in the cloud for its users.

8.1.3 Latency and Bandwidth Settings for Workflow Environment

The complete experiment has two different situations. First, in a normal distributed environment, we use the WANem to simulate the distributed environment for both the WSDF services and normal web services. We can set different latency and bandwidth for different distances. For example, use the latency value between two nodes to set the gateway between corresponding nodes in the simulated environment. Second, for cloud computing, as there is no cloud provider near us in Australia, we can not apply the controlled, simulated environment to test the WSDF framework. Since even using the WANem gateway, the results will still be dependent on the latency and variability of the shared wide area network. We carried out the cloud related experiments in the real distributed environment.

In a real world workflow system, clients and services are usually distributed at different physical locations. This often means the network properties (e.g, bandwidth and latency) vary between different parts of the whole workflow. The latency and bandwidth between client and services, as well as between services, can affect the network performance benefits brought by WSDF framework. In the simulated environment, our experiments are supposed to be similar to the real environment, so we set different latency and bandwidth to emulate inter-continental, intra-continental and local area network respectively.

As shown in Figure 8.1, a Linux box that has been installed with WANem software works as the gateway that emulates the environment. We set the latency and bandwidth of the gateway via the web interface provided by this gateway. Any two participants in the workflow, e.g. participants A and B, that need to have specified bandwidth and latency condition between them, should add this gateway on its path to the other participant. For example, with A, it should add the gateway on the path to B and vice versa. We configure the simulated environment according to the environment information from our previous experiments [25]. In these experiments, the network latency for inter-continental connection in the workflow is about 220ms and the bandwidth is 20Mbits/sec. To reflect these facts, we set the inter-continental network latency to

110ms for the WANem Gateway (as shown in Figure 8.1), as both the sender and the receiver are set to route via the gateway, and set the bandwidth to 20Mbits/sec for inter-continental connections. The intra-continental network latency is about 20ms, therefore, we set the latency to 10ms (for a similar reason as in the inter-continental case) on the WANem Gateway for intra-continental connections. The bandwidth for these connections is set to 40Mbits/sec. For local network, we do not set any latency and the bandwidth is 100Mbits/sec. By setting these, we can investigate three different network environments.

8.1.4 Services

Different services will take different processing time as well as I/O time. Even though we have removed the impact of processing time by introducing BST time, different service I/O time will still affect the results of the experiments. To avoid these differences, we will only use the RGB service (see section 7.4) in all different experiments.

A workflow can involve multiple services. Within a WSDF web service workflow, if each service uses the result from the previous service, only the first and the last service needs to send the data between the workflow engine (i.e. client) and the services. Other services can forward their data to their successor services. Normal web service workflow, on the other hand, needs to send data between web services and their workflow engine. The result is that, to the client or workflow engine, the data needs to be transferred is related to the numbers of web services invoked. In the example RGB workflow it is linear in the number of web services. The more services in a workflow are involved, the more efficient we expect the WSDF framework should be. We will test our WSDF framework with four different workflows. They are composed of 3, 6, 9 and 12 services respectively.

8.1.5 Data Size

The performance improvement of WSDF workflow can be affected by the input data size of the workflow. While not considering the extra time introduced by resource cre-

ation and management in the WSDF framework, the data transfer performance improvement ratio should be a constant value, which is basically decided by the number of web services involved within the workflow and the connections between the client and the services. In our experiments, we try to find out the impact of this these extra time. For different input data sizes, this extra time is a constant value. However, it will take a larger proportion of the total time for data transfer with small input data, and a relatively small proportion comparing with larger input data.

To find out the impact brought by the data size that a workflow processes as input, we provide different size data as input for the workflow. We have tested different data sizes to show that our system's performance for both small and large data sizes. The data size in our experiments ranges from 100KB to 2GB. Note that, our WSDF-axis engine is built on WS-core, which is written in Java. The maximum *int* value (2G), which is used by the Axis server in the WS-core as the maximum number of bytes it can receive for each request. We use 100K, 500K and 1M byte files as small size files; 5M, 10M 50M and 100M bytes files as middle size files; 500M, 1G and 2G bytes files as large size files.

8.2 Experiment Environment

We now describe our test environment, and other aspects of our tests of data transfer using web services.

Based on the testing methodology discussed in the previous section, we carried out both normal web service based RGB workflow and WSDF framework based RGB workflow under the simulated distributed system. All tests were run 10 times, and the results plotted in the figures are the average value.

8.2.1 Experiments in Emulated Distributed Environment

The computers used for the tests have the following specifications. We have four Linux boxes and all of them have two Intel(R) Pentium(R) 4 CPU 2.80GHz with 1 GB memory.

Three of them are used as servers and one of them is used as client or workflow engine. We also use a Linux box as the gateway to emulate remote network connection.

8.2.2 Experiments in Cloud Environment

We also carried out the experiment in a cloud environment by using cloud provided by the Science Cloud [40]. The Science Cloud is physically located in Chicago University. It provides a stateful web service interface to its users, by implementing a WSRF web service. Users can download the client side program and invoke the service. Detailed information for cloud testing environment can be found in chapter 9.

8.3 Theoretical Analysis

In this section, we are going to give the detailed results of the experiments, based on the testing methodology and experiment environment we built. Further, we will give both theoretical and experiment analysis.

Our work of analyzing the performance of WSDF workflow can be categories in two types: theoretical and experimental.

- ◇ Theoretical analysis. Build equations to calculate the I/O time used by normal web service workflow and WSDF workflow, then compare the difference between them.
- ◇ Experimental analysis. Carry out experiments in two different environments. First, in a simulated distributed environment, then within the cloud environment, which is illustrated in Chapter 9.

8.3.1 Theoretical Data Transfer Time Analysis

WSDF framework has time saving advantages over normal web service workflow, particularly for those workflows related with large data processing.

WSDF workflow provides the same functional services as normal web service workflow does, so there is no time saving from the functional services part. However, the

WSDF workflow can save time when the services exchange intermediate data. The performance comparison between different workflows will be focused on the data transfer.

We compare the total time used for network data transfer (referred to as transfer time) in both workflows and analyze the advantage of WSDF framework. First of all, we need to give the definition of the parameters to be used in the analysis work.

T : the overall transfer time for normal web service workflow.

$T^{'}$: the overall transfer time for WSDF workflow.

Overall transfer time represents the sum of various times used in the workflow for transferring data.

As there are multiple services in a workflow, the input and output data size for each service can vary. The network connections between the client and the services as well the ones between different services can be different.

DI_i , DO_i: represents input and output data of the *i*th service respectively.

$BW_{C,i}$: the bandwidth of the network connection between client and the *i*th service. (C for Client, BW for Bandwidth)

$BW_{i,i+1}$: the bandwidth of the network connection between the *i*th and *i+1*th service.

We suppose a workflow is composed of *n* atomic services and each *successor service* always uses the result data generated from its previous service.

Equation 8.1 calculates the overall transfer time in a normal distributed environment. T is the sum of transfer time used for input data DI_i and output data DO_i to transfer between the client and the server via network connection for each service. The bandwidth of the *i*th connection is $BW_{C,i}$. [1]

$$T = \sum_{i=1}^{n} (DI_i/BW_{C,i} + DO_i/BW_{C,i}) \tag{8.1}$$

In the WSDF framework, for the workflow under the same assumption, the overall transfer time is composed of three parts:

1. First, transfer time for input data from the client to the first service (i.e., sends out

[1]For simplicity reasons, we ignore other factors that might affect the transfer time, such as latency. Here, we suppose these factors only have a relatively small affect to the overall transfer time.

data from the workflow engine to the composite service).

2. Second, total transfer time for services (from the first to the *n-1*th service) to transfer output data from the *current service* to its *successor service*.

3. Finally, the output data transfer time from the *n*th service to the client (i.e., send back data from the composite service to the workflow engine).

$$T' = DI_1/BW_{C,1} + \sum_{i=1}^{n-1}(DO_i/BW_{i,i+1}) + DO_n/BW_{C,n} \tag{8.2}$$

To compare the difference between T and T', we make another assumption to simplify the equation: the output data from the *current service* is of equal size of the input data of the *successor service* (i.e. DO_i equals DI_{i+1}). Then we get the following:

$$T - T' = \sum_{i=1}^{n-1}(DO_i/BW_{C,i} + DO_i/BW_{C,i+1} - DO_i/BW_{i,i+1}) \tag{8.3}$$

Equation (8.3) illustrates the difference between the two frameworks. For both WSDF and normal Web service workflows, from the first to the *n-1*th service, each service transfers the same intermediate data DO_i from one service to the next, but via different network paths.

$DO_i/BW_{C,i} + DO_i/BW_{C,i+1}$ represents the transfer time under normal web service framework: the intermediate data DO_i is first sent from the *i*th service to the client via a network with bandwidth $BW_{C,i}$, and then sent from the client to the *i+1*th service via a network with bandwidth $BW_{C,i+1}$, i.e., the data transfer goes via a third party (the workflow engine); within a WSDF framework, the transfer time is $DO_i/BW_{i,i+1}$, which indicates that the intermediate data DO_i is sent from the *i*th service to the *i+1*th service directly via a network connection between different services with bandwidth $BW_{i,i+1}$. As mentioned in section 1, if the servers that host services in a workflow are located in a LAN and connected by high bandwidth network, then the bandwidth $BW_{i,i+1}$ is much larger than the $BW_{C,i}$ which is the bandwidth between server and client and the latency will be much smaller.

We define the ratio of data transfer time saving from WSDF to be:

$$P = \frac{T - T'}{T} \tag{8.4}$$

To further simplify the formulas, we make the following assumption: that bandwidths between the client and all the services are the same (represented by $BW_{C,S}$) and bandwidths between all services are the same ($BW_{S,S}$). Based on these assumptions, the performance improvement is given by:

$$P = \frac{\sum_{i=1}^{n-1}((DO_i + DI_{i+1})/BW_{C,S} - DO_i/BW_{S,S})}{\sum_{i=1}^{n}(DI_i + DO_i)/BW_{C,S}} * 100 \tag{8.5}$$

Within a workflow, the output data $DO_i(i \in (1, n-1))$ of one service is often used as the input data $DI_{i+1}(i \in (1, n-1))$ of the next service. If we use DO_i to replace DI_{i+1}, then equation 8.5 can be simplified to:

$$P = \frac{\sum_{i=1}^{n-1}((2 * DO_i/BW_{C,S}) - (DO_i/BW_{S,S}))}{\sum_{i=1}^{n}(DI_i + DO_i)/BW_{C,S}} * 100 \tag{8.6}$$

In our experiment, the WSDF workflow is built from *n* instances of ***RGB*** services, where all the input data and the output data in a workflow have the same size, represented by D. In this case, equation 8.6 becomes:

$$P = \frac{D * \sum_{i=1}^{n-1}(2/BW_{C,S} - 1/BW_{S,S})}{D * \sum_{i=1}^{n}(2/BW_{C,S})} \tag{8.7}$$

By simplifying the equation, we get:

$$P = \frac{(n-1) * (2/BW_{C,S} - 1/BW_{S,S})}{n * (2/BW_{C,S})} \tag{8.8}$$

In equation 8.8, if $BW_{S,S}$ is much larger than $BW_{C,S}$, then we can further simplify the equation to:

$$P = \frac{n-1}{n} \tag{8.9}$$

I.e., the maximum transfer time saving ratio P of WSDF framework is nearly $(n - 1)/n$. Of course, this value is an estimate of the maximum performance improvement

we can get under certain circumstances.

8.4 Results Analysis

In section 8.3.1, we have given the best possible performance improvement for data transfer that can be achieved by WSDF framework. As these theoretical analysis are based on a few assumptions that remove the less important aspects in the analysis (e.g., we ignore the data transfer latency that affects the transfer time as discussed in 8.3.1), to check the actual performance improvement with a real implementation of the WSDF framework, we carried out experimental tests to confirm the theoretical analysis.

We have carried out comprehensive experiments to compare WSDF workflow performance with web service workflows. According to our experiments, the performance of WSDF workflow shows great advantages on time saving for data transfer in both local network scale and Internet scale.

The detailed information for these experiments, including web services used in the tests, file sizes, selection of network bandwidth, and simulated environment setting up, can be found in section 8.1 and 8.2. These following figures (Figure 8.2, 8.3 and 8.4) show the total time consumption within the normal distributed environment.

Figure 8.2 shows the different performance of workflow in a LAN with different data size and number of web services. Figure 8.3 and Figure 8.4 show the performance of workflow in intra-continental environment and inter-continental environment respectively. In all these figures, we can see that the performance of WSDF service based workflow is better than Web service based workflow when the file size is larger than 5M bytes. However, the WSDF workflow has worse performance when the file size is less than 1M bytes: the worst is in a LAN and improved is in an inter-continental environment. The performance downgrading in small file size, we believe, is caused by the time required for resource management in WSDF framework as we discuss in the next section.

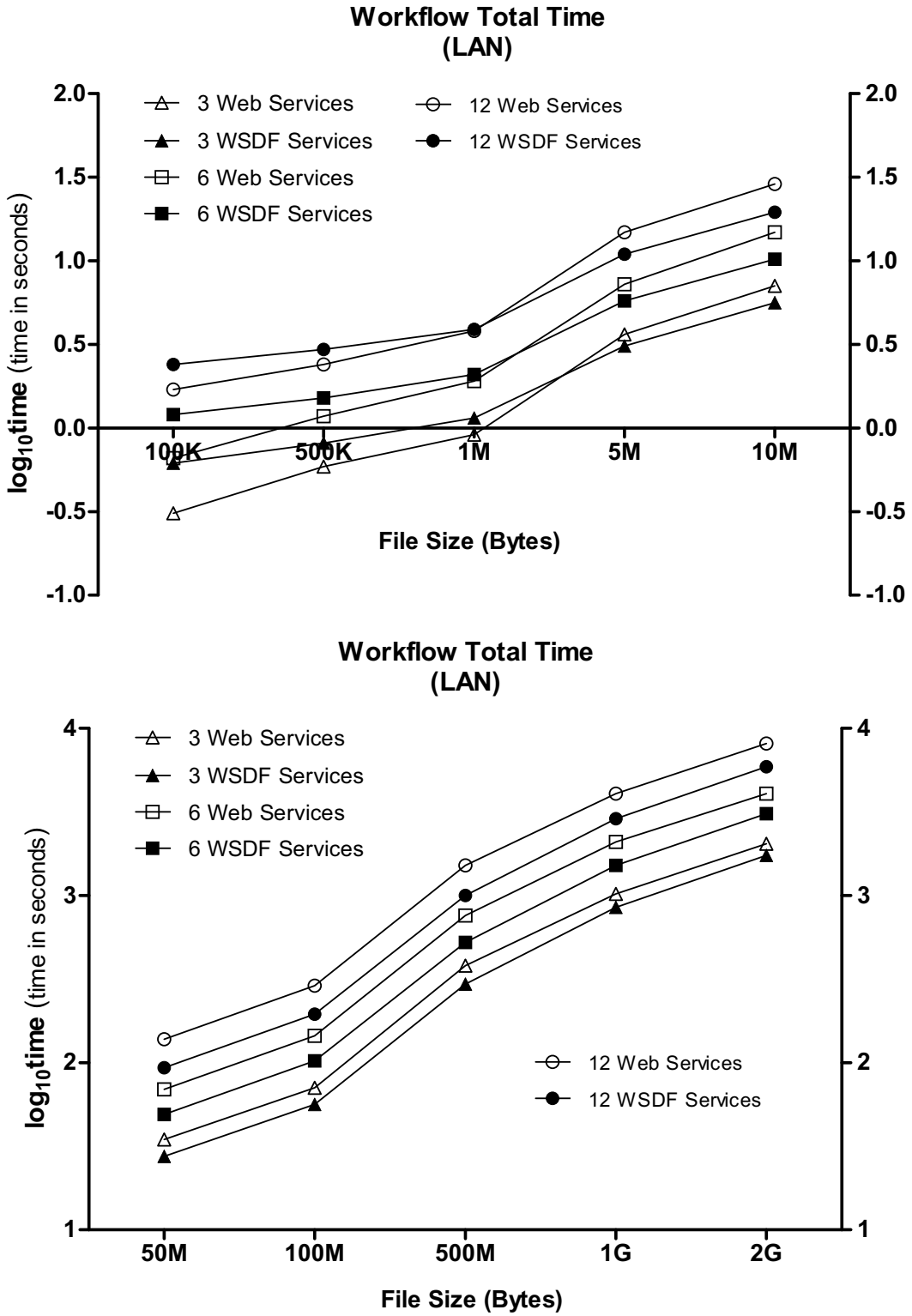


Figure 8.2: Web service vs. WSDF workflow (LAN)

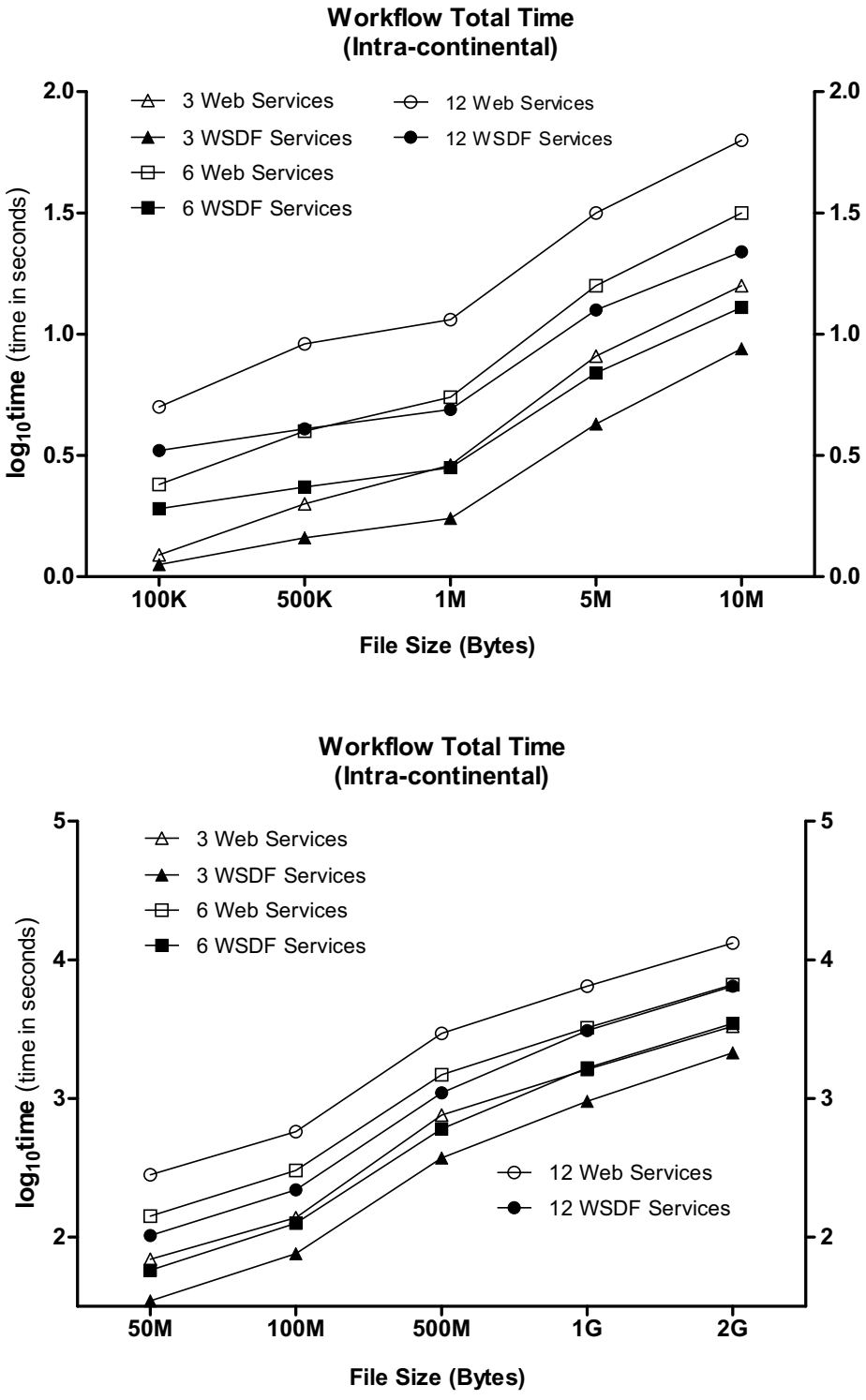


Figure 8.3: Web service vs. WSDF workflow(Intra-Continental)

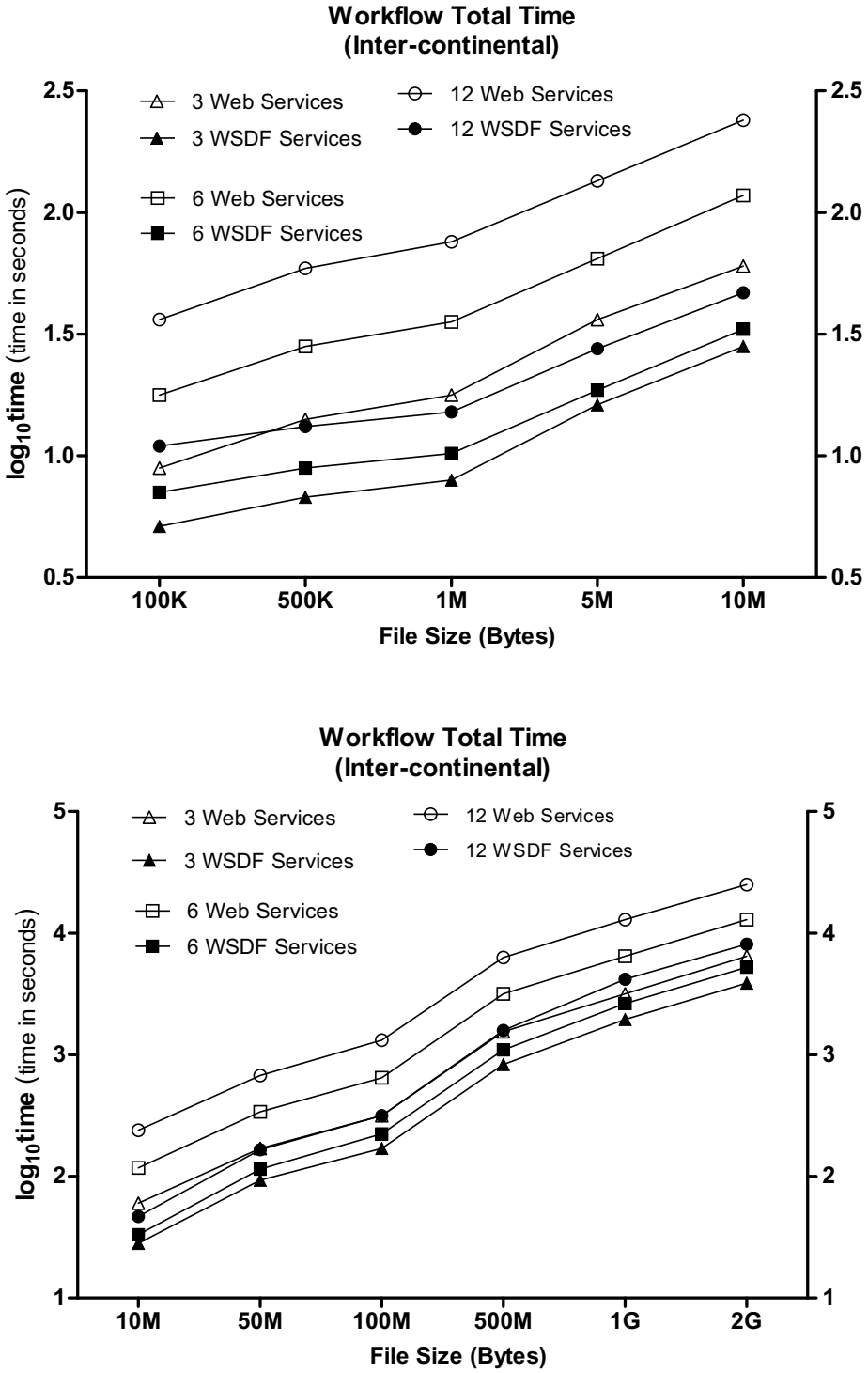


Figure 8.4: Web service vs. WSDF workflow(Inter-Continental)

8.4.1 Data Transfer vs. Resource Management

In a LAN environment, as shown in Figure 8.2, WSDF workflows have time saving advantages over web service workflows in most cases. In remote environments (Figure 8.3, 8.4), WSDF workflows show clear advantages over web service workflows.

WSDF framework saves time from data transfer, however, it requires some extra time for resource creation and management. In a LAN environment, this extra time could be higher than the time saved from data transfer by WSDF workflow (e.g. the experiments that use 100KB files with 3 services in Figure 8.2).

In a remote environment, for which network connection conditions between services are much better than the ones between the workflow engine and the services, then the time saved from data transfer is much larger compared to the extra time introduced by resource management in WSDF. Therefore, WSDF workflows have shown much more significant advantage.

8.4.2 Impact of Data Size

According to the experiment results, data size affects the performance improvement obtained by using WSDF workflows.

In these workflows, the time used for resource creation and management for each data set is the same, and according to equation 8.4, the time saving ratio on data transfer is also the same for data sets in different sizes. Comparing with large data sets, when workflows process small ones, the time saved on data transfer is relatively small, but the time used for resource creation and management is the same. This leads to the fact that the impact for resource instance creation and management is much more significant on small data sets. With the same number of services, a WSDF workflow can save more time when it is processing larger data. For example, in intra-continental environment, as shown in Figure 8.3, a WSDF workflow with 12 services can save about 33% transfer time of web service workflow with the input of a 100KB file, 60% with input file of 1MB and 70% for input file of 10MB. The percentage of performance improvement becomes steady when the input size increases further, as the resource creation time is relatively

small compared to the whole processing time.

Figure 8.5 shows the time saving of WSDF workflows over web service workflows in an inter-continental environment. The file size varies from 5M bytes to 50M bytes. All diagrams in this figure show the same trend: when the file size is big enough and the client and services are remotely located (more discussion about this in the next section), the WSDF workflows show consistent performance advantage over web service workflows. With the 5M bytes file, for example, the experiment results show that the WSDF workflows get 56%, 72%, 77% and 81% performance improvement comparing with the web service workflows.

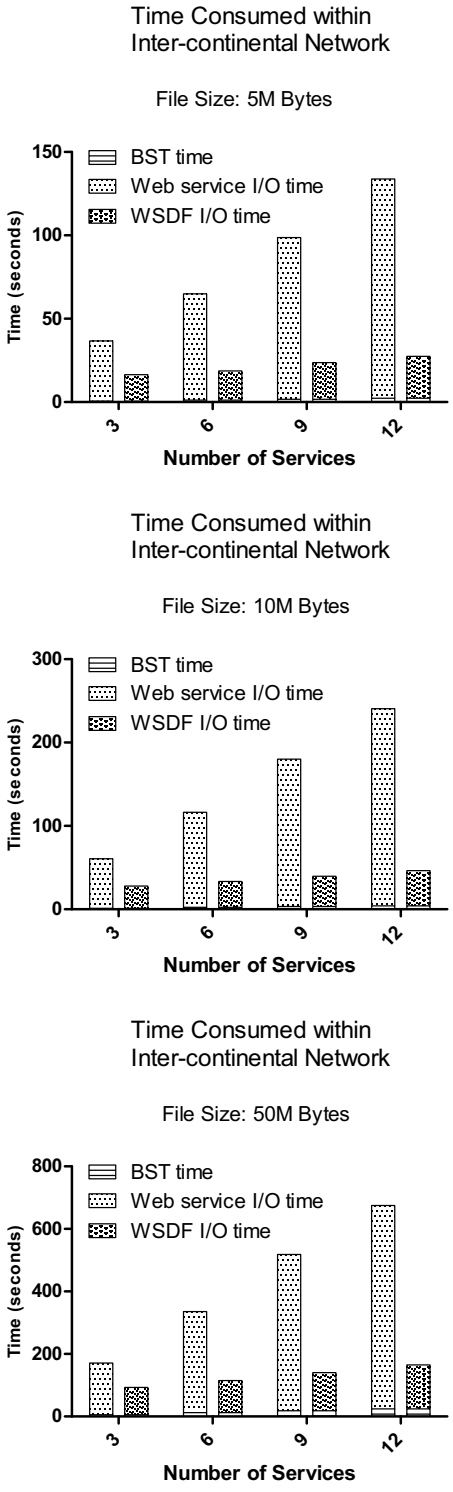


Figure 8.5: WSDF Performance Improve in Normal Distributed Environment

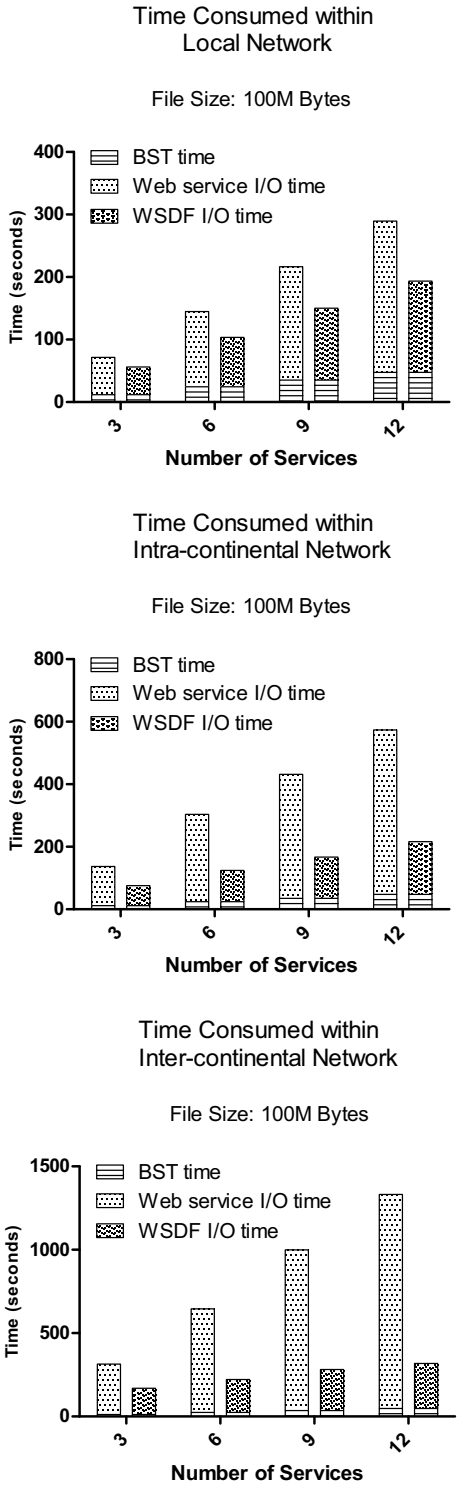


Figure 8.6: Time Saving Comparison In Distributed Environment

8.4.3 Network Connection Between Services and Client

Network connection between the client (or workflow engine) and the services plays an important role that affects performance improvement.

Comparing the time saving for local, intra-continental and inter-continental services, inter-continental WSDF workflow has the most improvement and local network WSDF workflow has the least improvement ratio. Equation 8.8 can be rewritten as:

$$P = \frac{(n-1)}{n} * (1 - 0.5 * \frac{BW_{C,S}}{BW_{S,S}}) \tag{8.10}$$

If we replace the bandwidth variables with the values we set in the experiments, then in a workflow with six services, the performance improvement for the workflow will be 42% in a LAN; 67% in an intra-continental environment and 75% in an inter-continental environment.

Figure 8.6 shows the time saving of the WSDF workflow in different environments. From the figure we can see that in an inter-continental environment, the WSDF workflow saves higher percentage of time comparing with it does in a LAN or an intra-continental environment. According to the real test results, in a LAN, the network time saving for RGB workflow with six services for a 100MB file is about 34%; in an intra-continental environment, the time saving is about 64%; and in an inter-continental environment, it is about 68% and they match the expected values.

The advantage of WSDF is that some data transfer are carried out between services involved within a workflow directly. Furthermore, an ideal environment for WSDF is that the servers are hosted within a LAN, in which the data connect is much better than the connections between the client and services. When the distance between the client and services are increased, it often leads to the decreasing of the effective bandwidth between the them. The slower the network connection from the client, the higher performance improvement from the WSDF framework.

8.4.3.1 Number of Services

The other factor that will affect the performance improvement of a WSDF workflow is the number of services involved within a workflow (see Figure 8.7). In Figure 1.1, there are three web services hosted on the remote servers are invoked in a workflow. In Figure 1.1 (a), three data transfers happen between the client and server, all of them are bi-directional. In Figure 1.1(b) and Figure 6.1, three WSDF services are hosted on remote servers, with a total of two data transfers between the services and the client with single direction data transfer. This means that for a workflow with three services invocation, the WSDF service workflow needs two single way data transfer between the client and the composite service, on the other hand, a normal web service workflow needs six single direction data transfers. If there are more services involved within a workflow, the WSDF based workflow still needs two single direction data transfer, only the data transfer between different services, which are all hosted in the remote LAN environment, will be increased. For normal web services, the number of data transfers between client and server for normal web services will increase in proportion to the number of services. From this point, when there are more services involved in the WSDF workflow, the performance of the workflow should also be increased accordingly.

We can also find the same conclusion from theoretical analysis. As in equation (8.9), the more services involved within a WSDF workflow, the higher ratio of time will be saved by the WSDF workflow, therefore, higher performance will be achieved.

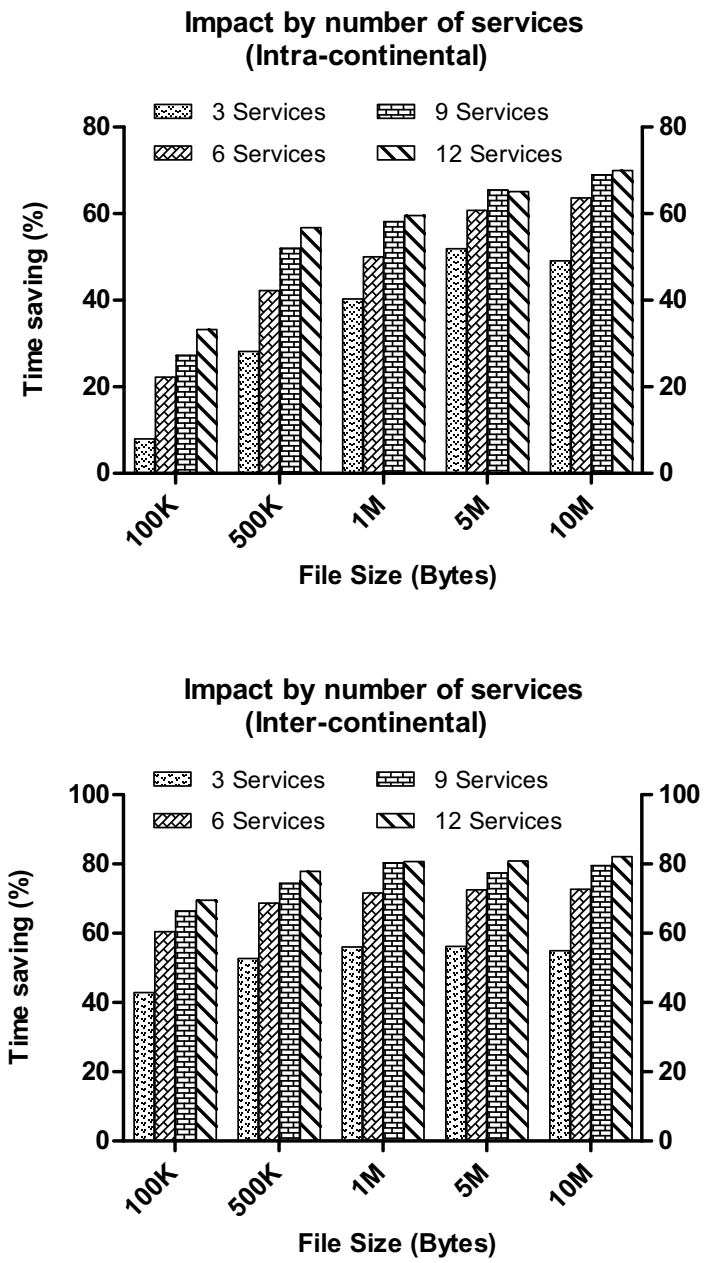


Figure 8.7: Increased Performance Improvement with More Services in a WSDF Workflow

8.4.4 Comparison with Theoretical Results

In section 8.3.1, we have made some assumptions to simplify the equations. In our real tests, some of these assumptions are true, e.g. the output data of the *current service*

has the same size as the input data of the *successive service;* and some of them, e.g. the latency of the network and the overhead of the resource management in WSDF workflow, have been ignored. In this section, we will compare the theoretical results based on our assumptions and figure out if these assumptions are reasonable.

Figure 8.6 shows the performance of both workflow frameworks under different frameworks. In the LAN environment, the network time saving for RGB workflow with six services for a 100MB input file is about 34%; in the intra-continental environment, the time saving is about 64%; and in the inter-continental environment, it is about 68%. If we put the related experimental data into equation 8.7 or 8.10 (section 8.4.3), the estimated transfer time saving for local service is 42%, for intra-continental service is 67% and for inter-continental service is 75%, which is reasonably close to our experimental results, as we are not considering the overhead brought by WSDF framework. The workflows with 3, 9 or 12 services present a similar trend, where the slower the connections between client and the service providers are, the bigger improvement could be achieved by WSDF framework.

This result meets our expectation: with WSDF workflows, as most data transferred does not need to be sent back to the client via a low bandwidth network, it takes less time for the workflow to complete. The result has also demonstrated that the implementation of our prototype WSDF workflow system as well as the experiments carried out has been successful.

In this chapter, we have shown our work on WSDF performance testing, including the setting up of a simulated environment and the results we got from the tests. We also provided theoretical analysis of expected WSDF performance improvement. Finally, we compared the real performance of WSDF framework with the expected result. It turns out that the WSDF tests carried out in the simulated environment we created have closely met the expected theoretical performance improvement. In the next chapter, we extend the testing environment to a cloud based testing environment and verify if the WSDF framework can meet the expected performance in a different environment.

Chapter 9

WSDF Testing in the Cloud

In Chapter 8, we carried out the WSDF testing within a given simulated distributed environment. In this chapter, we are going to report the WSDF tests carried out in a cloud environment. In a cloud environment, IT services, such as computational power and data storage capacity, can be provided by the cloud provider to the client according to the latter's request in a timely manner. The configuration of this cloud environment follows the specific configuration set by the client.

9.1 Introduction

Cloud is an emerging distributed computing model that has shown great potential, as discussed in section 1.5.7. In a traditional system, the administration is not completely automatic. Human intervention and communication is necessary to provide service hosting services. For example, a service provider asks the system administrator to maintain the service by providing a Linux box that has necessary service image installed on the machine. A system administrator will take care of the machine, including providing necessary network configuration and firewall setting for these machines. With very large data storage, which possibly exceeds the capacity of the single machine, it may take some time for system administrators to provide sufficient storage, which may require procuring and installing additional disks and could potentially introduce uncertainty as well as delay of the deployment of the whole workflow. On the other hand, the cloud

is built to essentially eliminate these problems: the resource instantiation and allocation are controlled by resource management software to achieve the highest efficiency. These services almost cover all requests that a workflow might require. Furthermore, within a cloud, it is not necessary to talk to a system administrator, since the provisioning or resources is all automated.

We see cloud as an ideal environment in which WSDF framework can be very well applied. Among the three different cloud models: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS), we choose IaaS platform to carry out our experiment. Within an IaaS platform, clients can provide their own disk images to run their workflow, which makes it easier for them to specify the run-time environment that is often vital in scientific research work. IaaS also provides the flexibility to move from one provider to the other, without necessarily being bound to a specific service provider. Finally, by using IaaS, we can use the exact same web services, including both web server and applications as in Chapter 7, in the cloud environment.

Cloud is an ideal environment for users from different disciplines and different organizations to collaborate in their work, such as running a workflow.

For a collaboration between different organizations, the participants need to find a common place to carry out their tasks. It will always be a burden for any single research institution to provide and manage computational and/or storage capacities as well as network resources to host workflows for the community. The cloud provider, which steps in as a third party dedicated service provider, can provide a dynamic service for the participants in the collaboration. For most cloud providers, they not only provide computational power, but also provide data storage capacity.

To the WSDF framework, the cloud provides another type of distributed environment, in which the data transfer performance can be improved to its maximum level. The cloud can be seen as a place, where all resources are gathered, including the computational power for running services and data storage. The client from a remote location sends data to the cloud and processes them and gets the result data back to his/her personal computer. The cloud, which is normally remotely located from the client, can provide resources that can be configured to be tightly connected, e.g. the computa-

tional nodes can be potentially located in a single data centre, and connected by high bandwidth network connection, which will deliver small latency and high throughput between different service connections. In a WSDF framework, the ideal scenario for performance improvement is that the services are closely located (high bandwidth, low latency), while they are far from the client (low bandwidth and high latency). From this point, the cloud architecture fits the WSDF framework very well.

We carried out the cloud experiments from a remote cloud (i.e. the *Science Clouds* [40]), measured the performance of the WSDF framework in the cloud, and compared the performance improvement under cloud with the improvement we got from the normal distributed experiments we obtained in chapter 8.

9.2 Testing Methodology

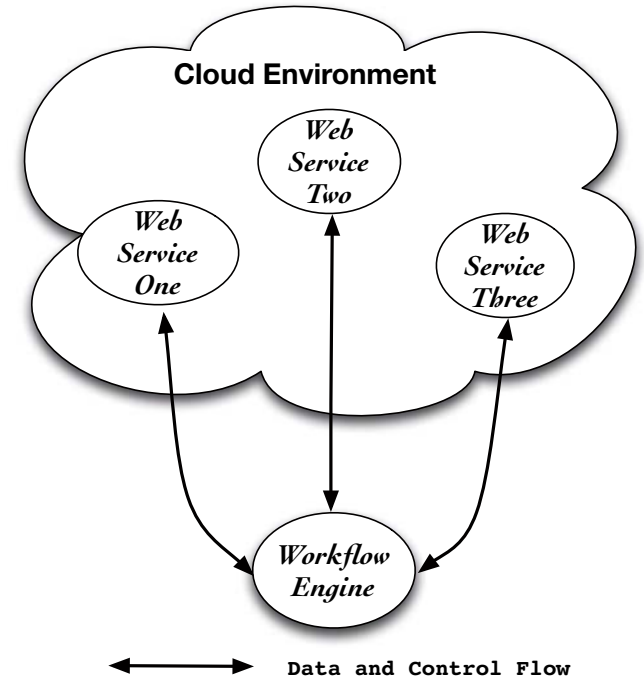
We have used the similar methodology applied in Chapter 8 for the tests carried out in the cloud.

These testing methods include Basic Service Time (BST) consumption, to compare the performance difference between a WSDF workflow with a normal web services workflow in the cloud environment. These time saving comes from data transfer part of the whole workflow.

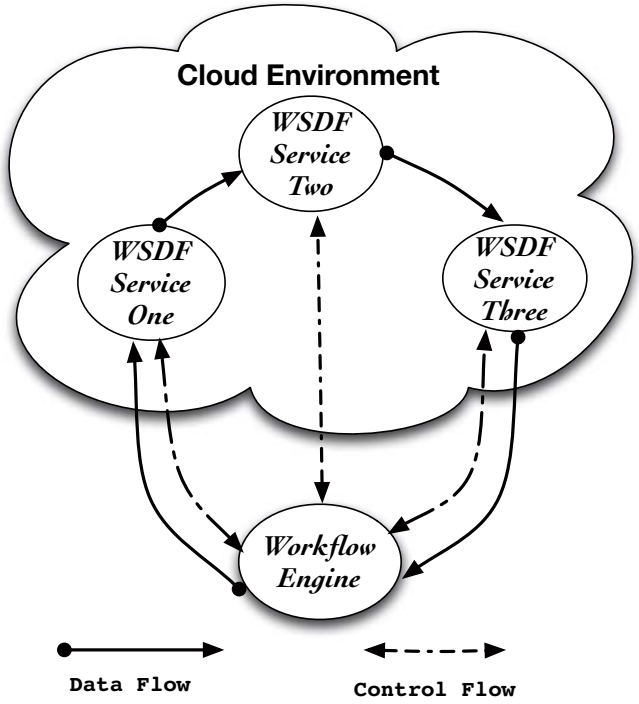
In previous experiments, we used a simulated WAN environment and ran the WSDF services in this environment. But in the case of cloud, we do not have the necessary hardware and software to host a cloud environment in this simulated WAN. We actually use a real cloud provider, the Science Clouds, which is remotely located on a different continent, to carry out the experiment.

9.2.1 Cloud Provider

We use the Science Clouds [40] for our cloud based WSDF experiments. Science Clouds is a public cloud provider for academic community and it is cost free. The Science Clouds is an instance of Nimbus cloud management software, hosted by the University of Chicago. The Nimbus cloud provides IaaS (Infrastructure as a Service) type services



(a) Web Service Workflow



(b) WSDF workflow

Figure 9.1: Workflows in Cloud

and computational cycles for its users. The Nimbus cloud software has also been deployed by other academic institutions such as University of Florida, Purdue University and Masaryk University.

9.2.2 Number of Services and Data Size

In order to compare with the performance improvement that the WSDF has achieved in a traditional distributed environment, we use the same RGB services, which have been used in the simulated environment (see section 7.4) in the cloud based tests.

Similar to other environments, a workflow in the cloud can involve multiple services. The more services in a workflow are involved, the more efficient we expect the WSDF framework should be. In the cloud testing environment, we will test our WSDF framework with four different workflows. They are composed of 3, 6, 9 and 12 services respectively. As in Chapter 8, we use 100K, 500K and 1M byte files as small size files; 5M, 10M 50M and 100M bytes files as middle size files; 500m, 1G and 2G bytes files as large size files.

9.3 Experiment Environment

Based on the testing methodology discussed in the previous section, we carried out both normal web service based RGB workflow and WSDF framework based workflow in the cloud environment. Due to the variability of network performance between the client and the cloud provider, all the tests were run 30 times, and the results plotted in the figures are the average value.

We carried out the experiment in a cloud environment by using cloud provided by the Science Clouds [40]. The Science Clouds has provided a stateful web service interface, which is built on the Globus implementation [29] of WSRF specification, to its users. Clients can download the client side program and invoke the service to create/maintain their resources.

The virtual machine instances run images we built based on a simple cloud image, called *hello-world* [41]. The hello-world image is provided by Science Clouds as a

Linux operating system and its size is about 2 Gigabytes. The new image is named *wsdf-hello-world* image and is 10 Gigabytes in size.

The data flow of the web service workflow and the WSDF service workflow is shown in Figure 9.1.

The reason we enlarge the image size is that often large data processing is involved in scientific workflows, and this is also the case in our experiment. The Science Clouds does not provide data storage services, such as S3 in Amazon [32], so if we carry out the experiments with the original image, it is very easy to exceed the available storage. To avoid the overflow, we enlarge the *hello-world* image to 10 Gigabytes. The new image is submitted to the cloud and saved into the user's repository by using a client side tool provided by Nimbus cloud management software.

While instantiating server instances in the Science Clouds, the user can specify the required environment (e.g. memory size, number of cores) for their instances. We use the default configuration file provided by Science Clouds to set up instances. We initialize three virtual machines vm01, vm02 and vm03. Each of these virtual machines has 2 CPU cores and 3 gigabytes of memory. These three virtual machines share the same image that we created with both web service and WSDF service installed on that image.

The firewall setting for the virtual machine is pre-configured on the *wsdf-hello-world* image. It allows the three virtual machines to communicate with each other directly. It also allows input/output from the client desktop in Adelaide University. By using *iperf* to test the connection between the virtual machines, we found the average result is about 910Mbits per second. The network connection between the cloud virtual machine and the client desktop in Adelaide University was also tested and the average result by using default configuration with *iperf* is 54.0 Mbits/sec.

On the cloud side, a Linux box with kernel version 2.6.18 is used as the workflow engine.

9.4 Cloud Experiment Result and Analysis

In this section, we report the experiment results collected from the cloud environment. It includes a comparison of the total time consumed by the workflows and the impact of changing the data size, and the number of services involved in the workflow.

9.4.1 Total Time Consumed

Figure 9.2 shows the total time taken for web service workflow and WSDF workflow within the Science Clouds environment. From the figure we can see, with different file sizes and different web services involved that the WSDF workflow always has significant advantages over normal web service workflow. This has been the same case as the experiments we had in the simulated distributed environment.

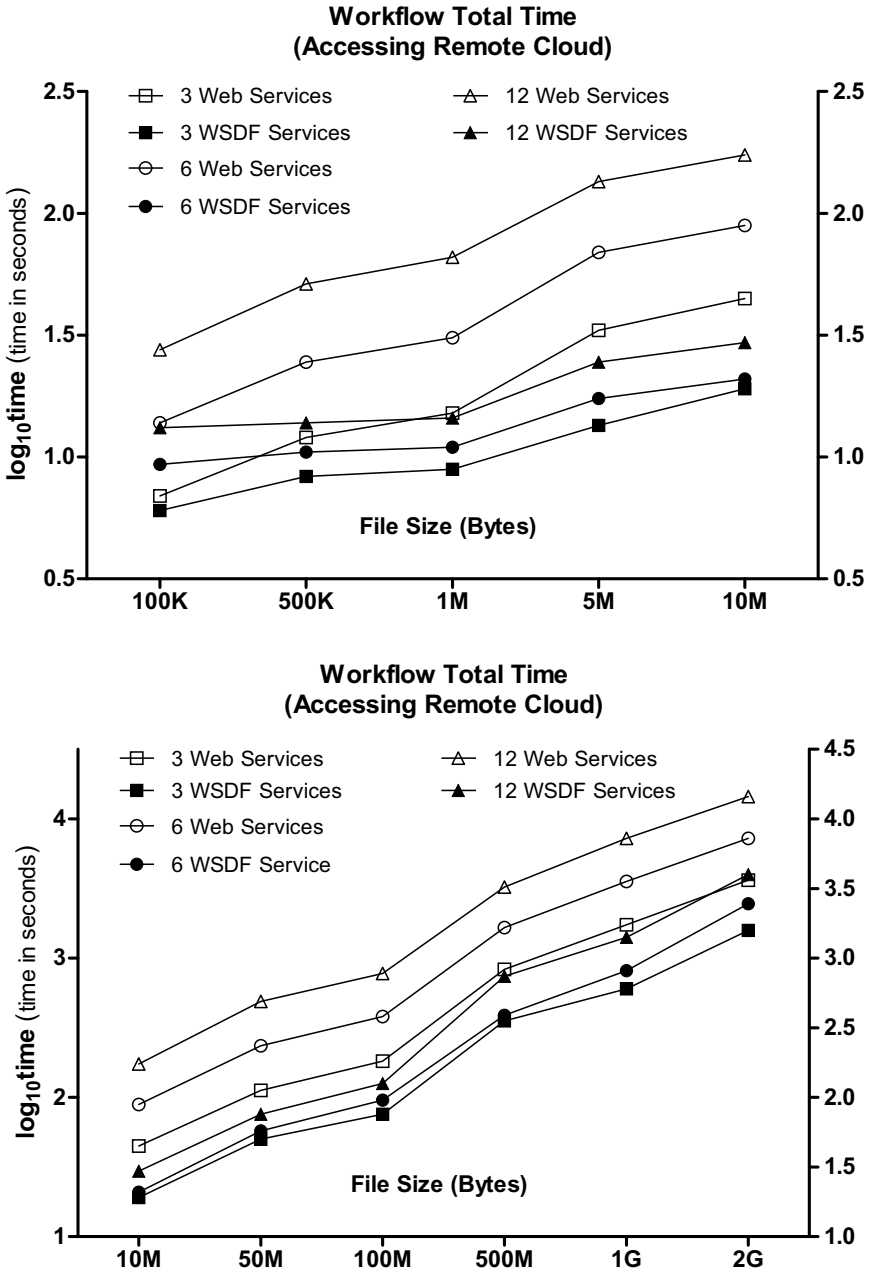


Figure 9.2: Total time consumption in cloud

9.4.2 Data Size

In Figure 9.3, 9.4 and 9.5, the performance of WSDF workflow vs. normal web service workflow is shown. Further, we also apply different number of web services within each workflow. The BST time (Basic Service Time) represents the time used by the web

service for functional processing.

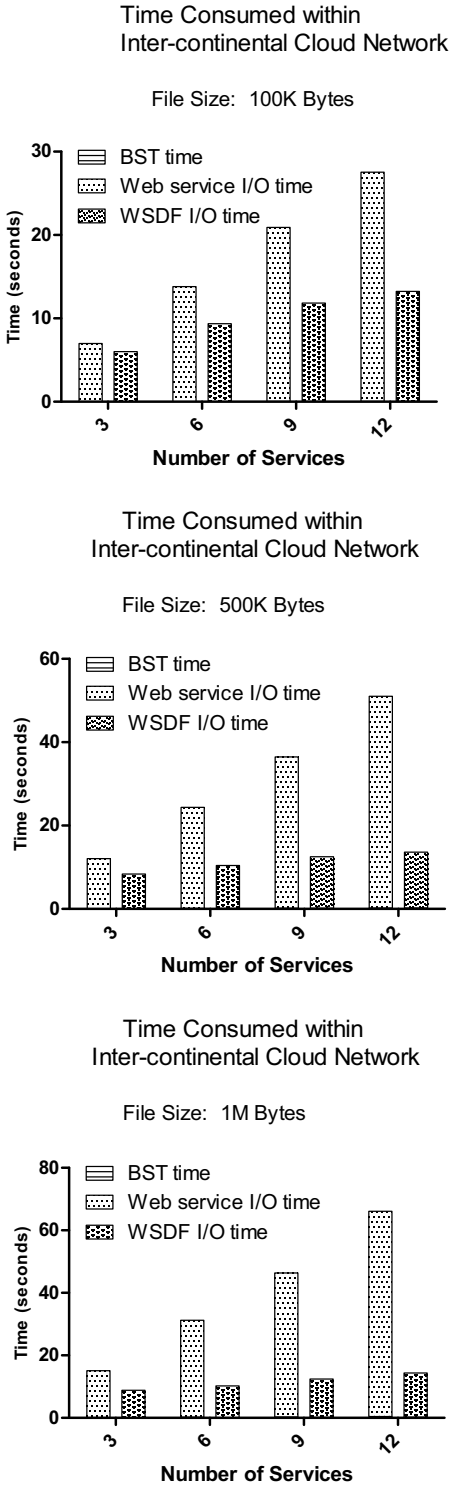


Figure 9.3: WSDF vs. Normal Web service Performance in Cloud (small file)

In Figure 9.3, the experiment is based on files with sizes range from 100K bytes to 1M bytes. As we can see from this figure, the BST takes only little percentage of the total processing time, as the data size is very small. Majority of the time used by both workflows is for transferring data between different participants in the workflow. According to the result, when there are three web services are involved within the processing, the 100K bytes workflow gets 14% time saving on data transfer, the 500K bytes file gets 31% of improvement and for a 1M bytes input file, the transfer time has been saved up to 42%. It shows the following trend: the larger the file, the higher improvement.

As discussed in section 8.4.2, comparing WSDF workflows with normal web service workflows, the time saving ratio on data transfer is kept unchanged for data sets in different sizes. The overall time saving for small file is less than that of the large file. In the WSDF workflow, a WSDF service manages the result generated from current service and forwards it to the successor service and saves it on the latter server as a resource. This means the successor service needs to create a resource reference for the result and manage the resource, which introduces extra time cost. As the resource creation and management time is a constant value for different data sets, it will take relatively higher percentage of the time saved by small data sets. When the input data size increases, the ratio of time used in resource management is decreasing quickly, so the overall performance improvement increases.

In Figure 9.4, the performance improvement of WSDF in the cloud with medium size input is given. For workflow with 3 RGB services, the performance improvement for workflow with 5M bytes is 59%, with 10M bytes file is 60% and with 50M bytes file is about 58%. This means the performance improvement by increasing the file size is relatively small and become reasonably constant and this also applies to the larger files as shown in Figure 9.5.

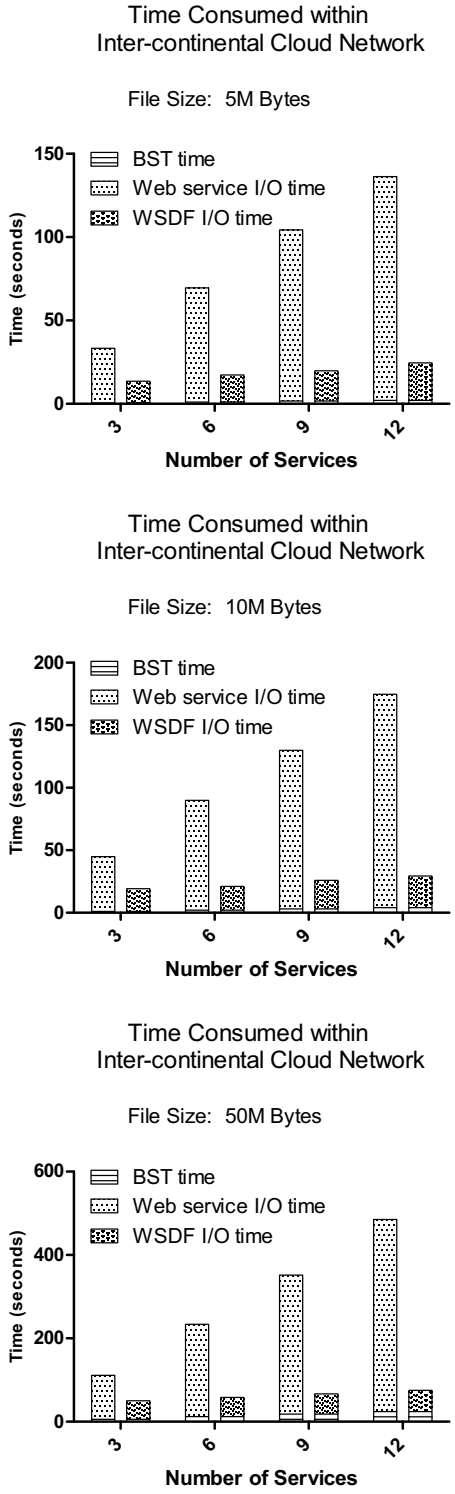


Figure 9.4: WSDF vs. Normal Web service Performance in Cloud (medium file)

Number of Services and Network Connection

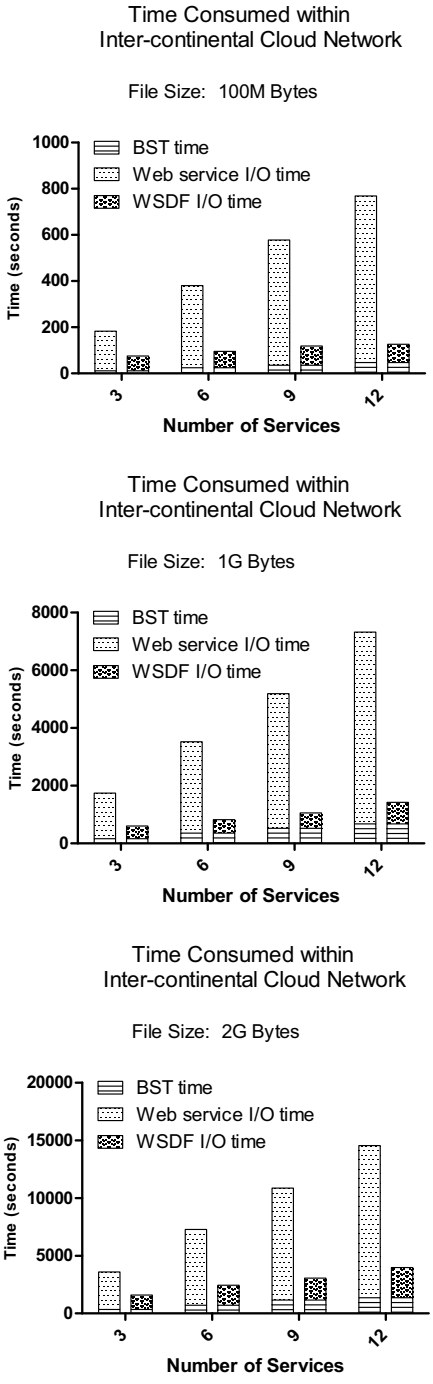


Figure 9.5: WSDF vs. Normal Web service Performance in Cloud (large file)

The other factor that will affect the performance improvement of a WSDF workflow is the number of services involved within a workflow. As shown in Figure 9.1, there are three web services hosted in the cloud that are invoked in a workflow. In Figure 9.1 (a),

there are three data transfers that happen between the client and server, all of them are bi-directional. In Figure 9.1 (b), the cloud hosts three WSDF services.

Similar to the situation in section 8.4.3.1, there are a total of two data transfers between the services and the client with single direction data transfer. This means that for a workflow that invokes three services, the WSDF service workflow needs two single way data transfer, on the other hand, a normal web service workflow needs six single direction data transfer. If there are more services involved within a workflow, the WSDF workflow still needs only two single direction data transfer, only the data transfer between different services, which are all hosted in the cloud, will be increased. For normal web services, the number of data transfers between client and server will increase in proportion to the number of web services. Because of these reasons, when there are more services involved in the WSDF workflow, its performance advantage will be more significant.

According to the experiment records, in Figure 9.4, with file size of 10M bytes, when there are three services involved, the time saving of WSDF workflow is 60%, when there are 6, 9 and 12 services involved, the time savings are 78%, 82% and 85% respectively. According to our experiment, the same trend also happens with different file sizes.

In section 8.4.3 and 8.4.4, we have given the expected performance improvement value by replacing the number of services and the network connection parameters into the proposed theoretical equations (8.8 and 8.10). The theoretical analysis also can be applied to the experiments carried out in the cloud environment. According to the network capacity we obtained when the experiments were carried out, the network bandwidth between the client and the servers was 54.0Mbits/sec, and the connections between different servers were 910Mbits/sec. If there are total six services in the workflow, by using equation 8.10, the theoretical result will be:

$$P = \frac{(n-1)}{n} * (1 - 0.5 * \frac{BW_{C,S}}{BW_{S,S}})$$

$$= \frac{(6-1)}{6} * (1 - 0.5 * \frac{54.0}{910.})$$

$$= \frac{5}{6} * (1 - 0.5 * \frac{54.0}{910.}) = 81\%$$

By applying all the numbers we get the expected value for 3, 6, 9 and 12 services are: 65%, 81%, 86% and 89%. Comparing with the real performance from the cloud experiments: 59%, 78%, 82% and 85%, the performance basically meet our expectation.

9.4.3 WSDF Performance Improvement Comparison

The WSDF framework is designed for improvement of the data transfer performance of web service workflows to be carried out in a distributed environment. Ideally, the services should be located near each other and connected via high-bandwidth, low latency network, while the client can be located remotely from the services and is responsible for services invocations. We have introduced two different environments for WSDF workflow testings: a simulated environment and a cloud environment. Both of these two environments have provided a similar environment in the way that the services are closely located and connected via fast local network, while the clients are remotely located via a relatively slow network connection. Therefore, we are expecting they will have the similar trend when comes to the performance improvement.

In Chapter 8, we have shown the experiment result of WSDF performance tests within simulated environment by using WANem[43] simulation software. Part of the result has been shown in Figure 8.5.

After the experiments in the cloud, we compare the performance improvement between the cloud environment and the simulated distributed environment. Figure 8.5 shows the time consumed within our previous work by using simulation environment.

The performance of the WSDF workflow in the cloud is similar to the WSDF workflow in the simulated distributed environment for inter-continental connections, and the data transfer performance improvement shows the performance difference is trivial. For example, with a 5M bytes file as the input file size, comparing with normal web services, the WSDF get 56%, 72%, 77% and 81% performance improvement in the simulated environment, as shown in Figure 8.5. And the same WSDF workflow in the cloud gets

60%, 76%, 82%, and 83% for 3, 6, 9, and 12 services.

For a workflow with 6 services, the different file sizes: 5M, 10M, 50M and 100M bytes. In the normal distributed environment: 73%, 73%, 68% and 68% performance improvement. And in cloud, the same WSDF workflow gets 76%, 78%, 79% and 80% performance improvement. The overall performance improvement of the cloud is normally higher than a single service provider.

In the cloud environment, the performance has a higher percentage of improvement, compared with the simulated environment. The reason for that is according to equation 8.8 and 8.10, after replacing the number of services and the bandwidth related variables, we are expecting that the cloud will bring higher performance (see section 9.4.2 and 8.4.3). The expected performance improvement for a workflow that invokes 6 web services in the cloud is about 81% with a simplified estimation, and that ratio in the simulated distributed environment is only 75%. And the real results basically match the expected values. Note that, in the cloud, the bandwidth of network connection between the client and the services is 54Mbits/sec, which is actually much higher than the bandwidth value of 20Mbits/sect set in the simulated environment. Under this condition, the cloud has provided much higher data transfer rate for the web service workflows. However, as the network connections between different services in the cloud is about 910Mbits/sec, which is much higher than the bandwidth between the services in the simulated environment, the overall performance improvement is still higher than the ones in the simulated environment.

We believe that, theoretically, there is no inherent data transfer improvement differences when the WSDF workflow are run in a normal distributed environment versus they are run in a cloud environment. However, as the cloud allows users to organize different resources (e.g. computational power, data storage and network capacity) in a flexible way, and often provides more tightly coupled resources connection (e.g. computational powers are often on a single stack or on nearby stacks), it is actually an ideal environment for WSDF workflows.

Chapter 10

Conclusion and Future Work

This conclusion covers two different aspects of the research work in the past years. First, we give the conclusions we have drawn from the research work, and second, we outline the research methodologies we learned and used in the research work.

The primary goal of this research is to improve the data transfer efficiency with e-Science applications, particularly for those that involve large data sets transfer over wide area network using web services. From this research work, we proposed novel approaches to improve the data transfer performance for data movement within data intensive workflow, as well as more efficient data sharing in a web service workflow. These approaches can be put into two categories: data transfer from one web service node to another; and data sharing between different participants in a workflow. First, we address how we utilize different approaches to improve the data transfer performance between two nodes; second, we show how to improve the overall performance of the workflow by providing optimized data sharing between applications within a workflow.

10.1 Data Transfer

The target of the first part of our research work was to improve the data transfer performance between the data sender and receiver in a web service based workflow. So the first step of our work is to improve the web service data transfer efficiency between two nodes.

During this period of the research work, we have investigated multiple optimization approaches to improve the performance of data transfer between different web service nodes, listed as follows:

- ◇ TCP tuning. The purpose of TCP tuning is to maximize the data throughput for a single TCP connection. With a high latency network, by increasing the buffer size to a certain level according to the round trip time and bandwidth, the data transfer ratio can be significantly increased (see section 4.5.2 for details). According to our experiments, the data transfer performance improvement arising from TCP tuning can be up to 30%.

- ◇ Parallel downloading with pull model. Here, we apply a combination of a pull model and parallel connections with a HTTP server. In our approach, the web service that generates the real data, does not directly send them to its receiver, but only forwards a HTTP reference of the data (which is saved on the server side HTTP server) to its clients. A client can use these references, with the pull model, to retrieve data hosted by the HTTP server. This gives the client the flexibility to choose the data it is interested in and the best time to download them. Further, this approach can be implemented in a parallel way, which will further improve the data downloading performance. Overall, up to 16 times data transfer performance increase can be achieved compared to single thread SOAP connection.

The approaches listed above have significantly improved the performance of the data transfer between two nodes of a web service workflow. However, each of the above two approaches has its own limitations. With TCP tuning, the resetting of the TCP buffer size can not be implemented at the normal user level and has to be applied to both end of the application; for the second approach, the pull model needs extra programming to manage a different download model outside of the web service framework. To simplify, we proposed an improved approach by using web service with attachment (WS-Att). With WS-Att, data is transferred between web services in binary format as attachments to the response of a service invocation, as all interactions between applications are under the unified web service interface, which simplifies the programming and administration

task, as well as reducing the total cost. However, we also need to figure out if the performance of WS-Att is competitive, particularly comparing to other approaches, such as GridFTP, which is a popular tool for high performance data transfer.

The experiments we carried out have shown that the data transfer performance when applying WS-Att is almost as good as the approaches we applied at the first step (TCP tuning, pull + HTTP, as summarized above and described in detail in Chapter 4). Further, the experiments show that WS-Att, especially with its XOP/MTOM specification, keeps performance of binary data transfer at a very high level, yet in a way that is much easier to use. Compared to GridFTP, the overall performance is similar: both of them can reach the data transfer speed of more than 80 MB per second. For a large number of small files (less than 10 MB), WS-Att has actually surpassed the performance of GridFTP. For very large files, GridFTP clearly has its advantages and can reach 90 MB per second performance. Detailed information on the performance testing can be found in Chapter 8.

Further, we conclude that: parallel concurrent data downloading provides similar functionality as TCP tuning (as discussed in 4.4). Whilst WS-Att has better performance compared with SOAP for binary data transfer, the overall performance of the system can be improved even more by applying multiple threads concurrently downloading data. Actually, GridFTP is implemented by making use of parallel data connection mechanism for data transfer. Both GridFTP and WS-Att have achieved very high performance without TCP tuning. Each one of the parallel connections has a buffer as the window for network data transfer. When multiple connections co-exist, the overall size of the network buffer has been increased and it provides similar functionality to TCP tuning.

At this stage, we have explored different approaches that can be used to improve web service data transfer performance. For the remainder part of our research, we focus on improving data sharing efficiency between web service applications.

10.2 Data Forwarding

In a web service workflow, e.g. an e-Science workflow, a set of data can be processed by multiple services in a certain order. In terms of the relationship between data generator and consumer, data is sent from generator to its consumer. However, in implementation, this might be different from the path via which data is actually transferred between different applications. For example, in the case of centralized workflow, data is transferred via the centralized workflow engine and forwarded to the next application. The motivation for proposing the Web Service Data Forwarding (WSDF) framework is to improve the efficiency of data sharing between services within a centralized workflow. In Chapters 6, 7, 8 and 9, a novel framework (WSDF) of improving data sharing performance was proposed, a complete implementation of the framework was developed and performance experiments were carried out and the results analyzed in order to show the advantages of our framework. Finally, the improvement was analyzed.

In the WSDF framework, data generated from a current service is directly sent to the successor service without going through the centralized workflow engine. Only the control information, such as invocation of the web service and resource references, goes via the workflow engine. This direct distributed data sharing between a current service and its successor service helps the workflow to save data transfer time. To support this model, we build WSDF services as stateful web services, so that the workflow engine can retrieve the data from the successor service later for the invocation on the same service.

As well as providing improved data transfer performance, our approach offers a good fit to the web service programming model and hence makes it easier to create web service workflows that provide good performance. Below we discuss some of these design considerations behind the WSDF framework.

10.2.1 Utilization of Web Service

WSDF and other early research works (e.g. *GriddLes [8])* focus on improving the data transfer efficiency of workflow. For example, both *WSDF* and *GriddLes* have been fo-

cusing on improving the data transfer efficiency between distributed applications. *Griddles* provides a transparent data transfer mechanism to help workflow developers to build workflows without worrying about the underlying implementation details. The efficiency of building workflow is improved significantly. The WSDF framework focuses on improving data transfer efficiency between web service applications by applying the concept of *resources* while providing unified web service interface.

However, there are significant differences when we compare these two approaches:

- ◇ *WSDF* tries to address the issue that all the data transfer has to go via the centralized workflow engine in a web service workflow system. *GriddLes* is not web services based, so it does not have the same issue that *WSDF* is tackling.

- ◇ *GriddLeS* supports multiple implementations by providing different clients, which could be very complex and error prone when the number of clients goes up. In the contrast, our *WSDF* approach uses a unified interface for data transfer, which provides a simplified interface.

- ◇ Our work is based on web services framework and utilizes stateful web services for implementation. *GriddLeS* does not take particular advantage from web service architecture.

- ◇ *GriddLeS* is a lower level approach compared with *WSDF*. The *GriddLeS* system needs more control of the computer system to allow it to trap system calls for reading and writing operations. *WSDF* approach updates the existing *WS-Core* which is a standalone web service engine and requires less system control.

- ◇ The GriddLeS system focus on re-utilization of legacy code. Our work looks to develop new applications.

The WSDF framework is built for web service based workflow. While web service may introduce extra overhead by applying XML based coding for messages, it has significant advantages in terms of level of abstraction, in that we can utilize advanced mechanisms brought by the XML language, such as *namespace* abstraction. For example, in our

research work, we define the namespace *wsdf* to distinguish the normal parameters from the *resource forwarding information*. Any input information under the *wsdf* namespace represents the destination of the result of the current service invocation. In this way, we avoid possible ambiguity between two types of information: input data as parameters for the functional service and the destination to where the result of the functional service is forwarded.

To keep the web service data transfer efficient, particularly for large binary data set, we apply web service with attachment (WS-Att), rather than other possible approaches, such as GridFTP or HTTP. By using this approach, the WSDF framework can maintain the advantages of a high level of abstraction, and still efficiently maintain the functionality required.

In establishing requirements for the WSDF framework (as shown in Chapters 6), we showed that each WSDF service should maintain its state. The design choice for WSDF is that we build the WSDF service as a stateful web service based on the WSRF framework. Again, by selecting this approach, we unified the whole system under a unified web service interface.

10.2.2 A Generalized Approach

There are multiple research works that have focused on trying to improve data sharing and transfer speed within a web service workflow. The common approach of these studies is to focus on application level solutions, as they are relatively easy to implement. However, this also introduces problems. For example, the implementation of data transfer can be *ad hoc,* and is often a burden for programmers. We try to develop a new approach that provides improved capability while keeping the same programming model and making use of the strengths of this model in our design and implementation.

After some research, we believe the web service is a good programming model to start with. Our WSDF proposal requires the WSDF server to provide data transfer functionality between services as a basic element of the server, which involves more effort, as we need to upgrade the existing web service engine to a WSDF service engine. How-

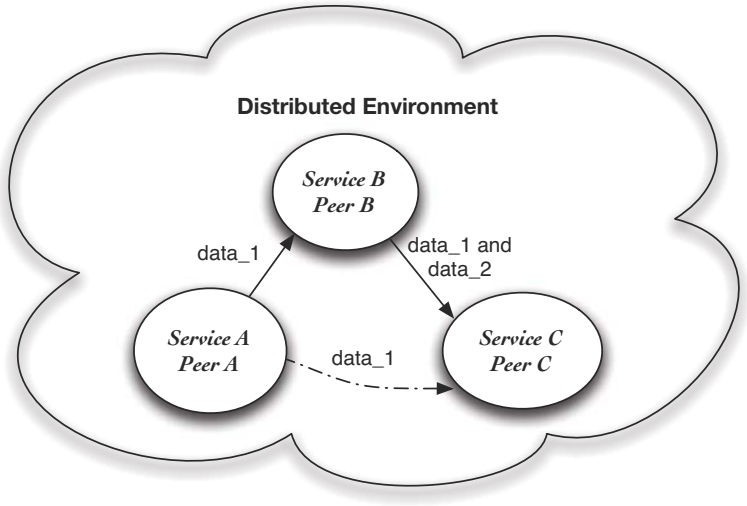


Figure 10.1: Data Transfer with WSDF in Decentralized Workflow

ever, we believe that this effort leads to a better solution that significantly reduces the burden on programmers.

10.2.3 WSDF with Decentralized Workflows

Decentralized workflow frameworks typically apply a peer-to-peer infrastructure to achieve direct communication and coordination among relevant peers for both data and control[142, 141, 121]. The traditional client-server architecture and centralized workflow engine is not used. While a decentralized workflow is running, data, as well as the control of the workflow, passes from one peer to the next workflow engine on a different peer. It is claimed [142, 141] that this peer-to-peer infrastructure can be used to provide genuinely decentralized workflow support, which removes the centralized data repository and control engine from the system.

WSDF was designed to tackle issues relevant to a centralized workflow, not anticipating a decentralized workflow system. However, if a mature decentralized workflow system appears, *WSDF* can nonetheless be used in such a system. For example, *WSDF* could be used to store intermediate result on peers that host workflow engines. In fact, when there are three or more peers in a workflow and there is data transfer between them, then the *WSDF* can be used to optimize data transfer.

Figure 10.1 shows how the data transfer can be optimized by using *WSDF* within a decentralized workflow. There are three services: service *A, B and C* in a decentralized workflow. They are hosted on Peer *A, B and C* respectively and will be invoked sequentially. Service *A* is first invoked and generates intermediate result *data_1; Service B* is then invoked without using *data_1* and generates *data_2*. Finally, *Service C* is invoked and both *data_1* and *data_2* are used. With a normal decentralized workflow, the intermediate result *data_1* is sent to *Service B* and then sent to *Service C*. The solid lines in the diagram show how *data_1* is transferred in this way. If *WSDF* services are applied, however, *data_1* is treated as a resource, i.e., it can be directly sent to *Service C* and store there to avoid unnecessary data transfer. The dashed line shows how *data_1* is transferred. After *Service B* is carried out and the control of the workflow is handed over to peer *C*, the invocation of *Service C* will retrieve *data_1* as local resource.

10.2.4 WSDF and Workflow Execution

The *WSDF* framework provides support for workflow execution monitoring and debugging in either a centralized or decentralized environment. However, the underlying system, a centralized or decentralized workflow, has a significant impact on this issue. A centralized workflow system will generally provide a better environment for workflow monitoring as the execution is always directly controlled by the centralized workflow engine and the execution is immediately reported to the central control point.

10.3 Future Work

In our research work, we have introduced the WSDF framework and tested it with a cloud environment in the *ScienceClouds*, an academic cloud environment provided by Chicago University. The performance was similar to or even better than the WSDF experiment in a normal distributed environment. Due to the limitations of the size of our experimental cloud environment, we have not obtained comprehensive results for a large scale cloud environment, such as Amazon [33] or Azure [44]. Further, the maximum size of the data used in the experiment is about 2 GB, which is relatively small in the

modern cloud environment. A comprehensive testing of WSDF performance in a large scale commercial cloud environment will be very interesting. At the same time, we can also investigate how a distributed system file system (e.g. Amazon S3 [32]) will affect the performance of a workflow.

With the current implementation of WSDF framework, there is no convenient application programming interface (API) existing to build WSDF client requests. A programmer has to write the WSDF SOAP envelope explicitly in the code. Our next target is to build programmer friendly APIs to remove the burden from programmers. To further help scientists who are not IT experts to build scientific workflows quickly and easy, we plan to provide a higher-level programming model which can be exposed to users via a GUI based interface (e.g. workflow workbench), that could handle the programming of WSDF data transfer automatically.

From an even broader prospective, I would like to expand work into the Cloud. As an emerging distributed computing infrastructure, cloud is often described as the next big trend of distributed computing. Cloud services provided by some cloud providers (such as Google and Amazon) have been widely accepted and show a good perspective for the future. Within the cloud, distributed data transfer and sharing is still an important issue. Further, the resource optimization is also an attractive area for future research work.

Bibliography

[1] AARNet Home Page. http://www.aarnet.edu.au/. Visited July, 2007.

[2] ARPANET. http://en.wikipedia.org/wiki/ARPANET.

[3] Axis1.1/java. http://ws.apache.org.

[4] CORBA. http://www.corba.org/. Visited Sep, 2007.

[5] Dot net framework developer center. http://msdn2.microsoft.com/en-us/library/ms951268.aspx.

[6] Extensible Markup Language (XML). http://www.w3.org/XML/.

[7] Grid enabled remote instrument with distributed control and computation. Visited July 2007.

[8] Griddles homepage. http://www.csse.monash.edu.au/ davida/griddles/. Visited Sep, 2007.

[9] HTTP - Hypertext Transfer Protocol. http://www.w3.org/Protocols/.

[10] JavaServer Pages Technology.

[11] Open Grid Forum. http://www.ogf.org/.

[12] OpenSSH Homepage. http://www.openssh.org/.

[13] Oracle Homepage.

[14] PostgreSql.

[15] Ptolemyii homepage. http://ptolemy.eecs.berkeley.edu/ptolemyII/. Visited Sep, 2007.

[16] UDDI Version 3.0.2. `http://www.uddi.org/pubs/uddi_v3.htm`.

[17] Usable grid infrastructures: practical experiences from the eminerals project.

[18] Web service Interoperability.

[19] FILE TRANSFER PROTOCOL (FTP), 1985.

[20] Attachments Profile Version 1.0. `http://www.ws-i.org/`, August 2004.

[21] An Australian e-Research Strategy and Implementation Framework. Technical report, Department of Education, Science and Trainging, April 2006.

[22] Oasis web service resource framework. `http://www.oasis-open.org`, 2006.

[23] Axis2/java. http://ws.apache.org/axis2/index.html, 2007.

[24] CERN homepage. `http://public.web.cern.ch/public/Welcome.html`, August 2007.

[25] Escience-grid homepage. http://wwww.escience-grid.org.uk, July 2007.

[26] Kepler homepage. `http://www.kepler-project.org/`, 2007.

[27] Xfire homepage. http://xfire.codehaus.org/, July 2007.

[28] Enabling High Performance Data Transfers. `http://www.psc.edu/networking/projects/tcptune/`, 2008.

[29] Globus Project Homepage. `http://www.globus.org/`, May 2008.

[30] Introduction to irods, 2008.

[31] iperf. `http://iperf.sourceforge.net/`, 2008.

[32] Amazon simple storage service (amazon s3). http://aws.amazon.com/s3/, 2009.

[33] Amazon web services. http://aws.amazon.com, 2009.

[34] Argo Project. http://www.argo.ucsd.edu/, 2009.

[35] Cloud computing. http://en.wikipedia.org/wiki/Cloud_computing, 2009.

[36] Council for the Central Laboratory of the Research Councils, 2009.

[37] Gartner Highlights Five Attributes of Cloud Computing. http://www.gartner.com/technology/home.jsp, 2009.

[38] LHC-THE LARGE HADRON COLLIDER. http://lhc.web.cern.ch/lhc/, 2009.

[39] ncftp Software, 2009.

[40] Science clouds. http://www.scienceclouds.org/, 2009.

[41] Science clouds marketplace. http://www.scienceclouds.org/marketplace/, 2009.

[42] Storage resource broker. http://www.sdsc.edu/srb/index.php/Main_Page, 2009.

[43] WANem The Wide Area Network emulator. http://wanem.sourceforge.net/, March 2009.

[44] Windows azure platform. http://www.microsoft.com/windowsazure, 2009.

[45] WSCORE. http://dev.globus.org/wiki/Java_WS_Core, 2009.

[46] https://www.ggf.org/documents/GFD.16.pdf, 2010.

[47] https://www.ggf.org/documents/GFD.78.pdf, 2010.

[48] Aapache Muse (WSRF implementation), 2010.

[49] The data deluge. http://www.economist.com/node/15579717, 2010.

[50] Internet. http://en.wikipedia.org/wiki/Internet, 2010.

[51] Mysql homepage, 2010.

[52] Triana Project, 2010.

[53] WSRF.net, 2010.

[54] http://www.ws-i.org/, 2011.

[55] Australian synchrotron. http://www.synchrotron.org.au/, 2011.

[56] Bittorrent homepage. http://www.bittorrent.com/, 2011.

[57] Canada light source. http://www.lightsource.ca/, 2011.

[58] Conceptual Grid Authorization Framework and Classification. www.ogf.org/documents/GFD.38.pdf, November 2011.

[59] e-Science. http://en.wikipedia.org/wiki/E-Science, November 2011.

[60] ERLANG official web site. http://www.erlang.org/, 2011.

[61] Google APP Engine. http://code.google.com/appengine/, 2011.

[62] Google Doc. http://docs.google.com, November 2011.

[63] GridFTP. http://www.globus.org/toolkit/docs/, July 2011.

[64] gsoap. http://gsoap2.sourceforge.net/, 2011.

[65] Java API. http://download.oracle.com/javase/1.4.2/docs/api, July 2011.

[66] Oasis open homepage. `http://www.oasis-open.org/home/index.php`, 2011.

[67] Salesforce. `http://www.salesforce.com`, November 2011.

[68] Service Oriented Architecture Diagram. `http://www.w3.org/`, November 2011.

[69] Spe file format. `www.gbs-elektronik.de/`, 2011.

[70] The Storage Resource Manager Interface. `www.ggf.org/documents/GFD.129.pdf`, November 2011.

[71] XML-RPC Specification. `http://www.xmlrpc.com/spec`, November 2011.

[72] `http://www.nesc.ac.uk/nesc/define.html`, 2012.

[73] `www.ands.org.au/guides/dcb/felzmann-synchrotron.pdf`, 2012.

[74] `www.ansto.gov.au`, 2012.

[75] `http://www.openmicroscopy.org/site`, 2012.

[76] Anjomshoaa. A, Brisard. F, Drescher. M, Fellows. D, Ly. An, McGough. S, Ovoca. D, and Savva. A. Job submission description language (jsdl) specification, version 1.0, 2008.

[77] Swarup Acharya, Michael Franklin, and Stanley Zdonik. Balancing push and pull for data broadcast. *SIGMOD Rec.*, 26(2):183–194, 1997.

[78] William Allcock, John Bresnahan, Rajkumar Kettimuthu, and Michael Link. The Globus Striped GridFTP Framework and Server. In *SC '05: Proc. of the 2005 ACM/IEEE conference on Supercomputing*, page 54, Washington, DC, USA, 2005. IEEE Computer Society.

[79] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A view of cloud computing. *Commun. ACM*, 53:50–58, April 2010.

[80] Ian M. Atkinson, Douglas du Boulay, Clinton Chee, Kenneth Chiu, Paul Coddington, Andrea Gerson, Tristan King, Donald F. McMullen, Romain Quilici, Peter Turner, Andrew Wendelborn, Mathew Wyatt, and Donglai Zhang. Developing cima-based cyberinfrastructure for remote access to scientific instruments and collaborative e-research. In *ACSW '07: Proceedings of the fifth Australasian symposium on ACSW frontiers*, pages 3–10, Darlinghurst, Australia, Australia, 2007. Australian Computer Society, Inc.

[81] Ian M. Atkinson, Douglas du Boulay, et al. Developing cima-based cyberinfrastructure for remote access to scientific instruments and collaborative e-research. In *Australian Symposium on Grid Computing and Research, Conferences in Research and Practice in Information Technology*, page Vol. 68. Australian Computer Society, Inc., 2007.

[82] Adam Barker, Paolo Besana, David Robertson, and Jon B. Weissman. The benefits of service choreography for data-intensive computing. CLADE '09, pages 1–10. ACM, 2009.

[83] Adam Barker, Jon Weissman, and Jano van Hemert. Orchestrating data-centric workflows. In *CCGRID '08: Proc. of the 2008 Eighth IEEE Int. Symposium on Cluster Computing and the Grid*, pages 210–217, Washington, DC, USA, 2008. IEEE Computer Society.

[84] Adam Barker, Jon B. Weissman, and Jano van Hemert. Orchestrating data-centric workflows. In *CCGrid, 2008*.

[85] J. Barton, S. Thatte, and F. Nielsen. *SOAP Messages with Attachments*. W3C, December 2000. `http://www.w3.org/TR/SOAP-attachments`.

[86] Gordon Bell, Tony Hey, and Alex Szalay. Beyond the Data Deluge. *Science*, 323:1297–1298, 2009.

[87] B. Benatallah, Q.Z. Sheng, and M. Dumas. The Self-Serv environment for Web services composition. *Internet Computing, IEEE*, 7(1):40–48, Jan/Feb 2003.

[88] Steven Tuecke Bill Allcock, Lee Liming. GridFTP: A Data Transfer Protocol for the Grid. 2003.

[89] Walter Binder, Ion Constantinescu, and Boi Faltings. Decentralized orchestration of composite web services. In *ICWS '06: Proc. of the IEEE Int. Conference on Web Services*, pages 869–876, Washington, DC, USA, 2006. IEEE Computer Society.

[90] Judith A. Blake and Carol J. Bult. Beyond the data deluge: data integration and bio-ontologies. *J. of Biomedical Informatics*, 39(3):314–320, 2006.

[91] Lisa Blanshard, Rik Tyer, Glen Drinkwater, Ananta Manandhar, Shoaib Sufi, Kerstin Klesse van Dam, and Martin Dove. Case Study: Using Web Services for the Management of Environmental Data. In *Proceedings of the 2004 IEEE International Conference on Services Computing*, pages 132–136, Washington, DC, USA, 2004. IEEE Computer Society.

[92] D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, and D. Orchard. Web Services Architecture. `http://www.w3.org/`, February 2004.

[93] Dhruba Borthakur. *The Hadoop Distributed File System: Architecture and Design*, 2004.

[94] All Bramley, Kenneth Chiu, Tharaka Devadithya, Nisha Gupta, Charles Hart, John C. Huffman, Kianosh Huffman, Yu Ma, and Donald F. Mcmullen. Instrument monitoring, data sharing and archiving using common instrument middleware architecture. *Journal of Chemical Information and Modeling*, 46, 2006.

[95] John Bresnahan, Michael Link, Rajkumar Kettimuthu, Dan Fraser, and Ian Foster. Gridftp pipelining. volume 0, Madison, WI, 2007. In Proceeding of TERAGRID 2007 Conference.

[96] R Chinnici, Curbera Francisco, Meredith Greg, and Weerawarana Sanjiva. Web Services Description Language (WSDL) Version 1.2, 2002.

[97] R Chinnici, Moreau. J, Ryman. A, and Weerawarana Sanjiva. Web Services Description Language (WSDL) Version 2.0, 2007.

[98] Kenneth Chiu, Tharaka Devadithya, Wei Lu, and Aleksander Slominski. A binary xml for scientific applications. *e-science*, 0:336–343, 2005.

[99] Kenneth Chiu, Madhusudhan Govindaraju, and Randall Bramley. Investigating the limits of soap performance for scientific computing. In *HPDC '02: Proc. of the 11th IEEE Int. Symposium on High Performance Distributed Computing*, page 246, Washington, DC, USA, 2002. IEEE Computer Society.

[100] K Czajkowski, Ferguson. D, Foster. I, Frey. J, Graham. S, Maguire. T, Snelling. D, and Tuecke. S. From Open Grid Services Infrastructure to WS-Resource Framework: Refactoring and Evolution, 2004.

[101] David Liu Dept and David Liu. Analysis of integration models for service composition. In *Proceedings of Third International Workshop on Software and Performance*, pages 158–165. ACM Press, 2002.

[102] Tharaka Devadithya, Kenneth Chiu, Kianosh Huffman, and Donald F. McMullen. The common instrument middleware architecture: Overview of goals and implementation. *e-science*, 0:578–585, 2005.

[103] Thomas Erl. *Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2005.

[104] Roy T. Fielding. Software architectural styles for network-based applications. Technical report, 1999.

[105] Ian Foster. Globus toolkit version 4: Software for service-oriented systems. In Hai Jin, Daniel Reed, and Wenbin Jiang, editors, *Network and Parallel Computing*, volume 3779 of *Lecture Notes in Computer Science*, pages 2–13. Springer Berlin / Heidelberg.

[106] Ian Foster, Carl Kesselman, and Steven Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *Int. J. High Perform. Comput. Appl.*, 15(3):200–222, 2001.

[107] N. Freed and Borenstein N. *Multipurpose Internet Mail Extensions*. IETF, November 1996. http://www.ietf.org/rfc/rfc2045.txt.

[108] Jeannine Hall Gailey. Sending files, attachments, and soap messages via direct internet message encapsulation. *MSDN Magazine*, December 2002. `http://msdn.microsoft.com/`.

[109] Henry Gardner, Chris Johnson, Geoff Leach, and Pascal Vuylsteker. escience curricula at two australian universities. In *ACE '05: Proc. of the 7th Australasian conference on Computing education*, pages 211–216, Darlinghurst, Australia, Australia, 2005. Australian Computer Society, Inc.

[110] Jeremy Geelan. Twenty-one experts define cloud computing. `http://cloudcomputing.sys-con.com/node/612375/`, 2011.

[111] Dimitrios Georgakopoulos, Mark F. Hornick, and Amit P. Sheth. An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure. *Distributed and Parallel Databases*, 3(2):119–153, 1995.

[112] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The Google file system. *SIGOPS Oper. Syst. Rev.*, 37(5):29–43, 2003.

[113] Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, Henrik Nielsen, Anish Karmarkar, and Yves Lafon. *SOAP Version 1.2*. W3C, April 2007. `http://www.w3.org/TR/soap12/`.

[114] Martin Gudgin, Noah Mendelsohn, Mark Nottingham, and Herve Ruellan. *SOAP Message Transmission Optimization Mechanism*. W3C, January 2005. Visited July, 2007.

[115] Martin Gudgin, Noah Mendelsohn, Mark Nottingham, and Herve Ruellan. *XML-binary Optimized Packaging*. W3C, January 2005. Visited July, 2007.

[116] Brian Hayes. Cloud computing. *Commun. ACM*, 51:9–11, July 2008.

[117] Steffen Heinzl, Markus Mathes, Thomas Friese, Matthew Smith, and Bernd Freisleben. Flex-swa: Flexible exchange of binary data based on soap messages with attachments. In *ICWS '06: Proc. of the IEEE Int. Conference on Web Services (ICWS'06)*, pages 3–10, Washington, DC, USA, 2006. IEEE Computer Society.

[118] H. Hochheiser and I.G. Goldberg. Quasi-hierarchical, interactive navigation of images and meta-data in the open microscopy environment. In *3rd IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, pages 1272 – 1275, Arlington, VA, 2006. IEEE Computer Society.

[119] Ed. Josefsson, S. The base16, base32, and base64 data encodings. http://www.ietf.org/rfc/rfc3548.txt.

[120] Nicholas T. Karonis, Michael E. Papka, Justin Binns, John Bresnahan, Joseph A. Insley, David Jones, and Joseph M. Link. High-resolution remote rendering of large datasets in a collaborative environment. *Future Gener. Comput. Syst.*, 19:909–917, August 2003.

[121] Peter M. Kelly, Paul D. Coddington, and Andrew L. Wendelborn. Lambda calculus as a workflow model. *Concurrency and Computation: Practice and Experience*, 21(16):1999–2017, 2009.

[122] D. Liu, K.H. Law, and G. Wiederhold. Data-flow distribution in ficas service composition infrastructure. In *In Proceedings of the 15th International Conference on Parallel and Distributed Computing Systems*, 2003.

[123] Bertram Ludascher, Ilkay Altintas, Chad Berkley, Dan Higgins, Efrat Jaeger, Matthew Jones, Edward A. Lee, Jing Tao, and Yang Zhao. Scientific workflow management and the kepler system: Research articles. *Concurr. Comput. : Pract. Exper.*, 18(10):1039–1065, 2006.

[124] Gudgin. M, Hadley. M, and Rogers. T. Web Services Addressing 1.0 - Core, 2006.

[125] Peter Mell and Timothy Grance. The nist definition of cloud computing. `http://csrc.nist.gov/.../SP800-145.pdf`, 2011.

[126] Mangala Nanda, Satish Chandra, and Vivek Sarkar. Decentralizing execution of composite web services. In *OOPSLA 04'*, pages 170–187, New York, NY, USA, 2004. ACM.

[127] Tom Oinn, Matthew Addis, Justin Ferris, Darren Marvin, Martin Senger, Mark Greenwood, Tim Carver, Kevin Glover, Matthew R. Pocock, Anil Wipat, and Peter Li. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(17):3045–3054, 2004.

[128] Tom Oinn, Mark Greenwood, Matthew Addis, M. Nedim Alpdemir, Justin Ferris, Kevin Glover, Carole Goble, Antoon Goderis, Duncan Hull, Darren Marvin, Peter Li, Phillip Lord, Matthew R. Pocock, Martin Senger, Robert Stevens, Anil Wipat, and Chris Wroe. Taverna: lessons in creating a workflow environment for the life sciences: Research articles. *Concurr. Comput. : Pract. Exper.*, 18(10):1067–1100, August 2006.

[129] Michael P. Papazoglou, Paolo Traverso, Schahram Dustdar, and Frank Leymann. Service-Oriented Computing: State of the Art and Research Challenges. *Computer*, 40:38–45, 2007.

[130] Mike P. Papazoglou. Service -oriented computing: Concepts, characteristics and directions. *Web Information Systems Engineering, International Conference on*, 0:3, 2003.

[131] Cesare Pautasso, Olaf Zimmermann, and Frank Leymann. Restful web services vs. "big"' web services: making the right architectural decision. In *Proceeding of the 17th international conference on World Wide Web*, WWW '08, pages 805–814, New York, NY, USA, 2008. ACM.

[132] Jan Pechanec. How the SCP protocol works. `http://blogs.sun.com/janp/entry/how_the_scp_protocol-works`.

[133] L. Roberts, L. J. Blanshard, R. P Tyer, and K. Kleese Van Dam. Enabling effective collaboration through a web-enabled data infrastructure, 2004.

[134] Marek Rusinkiewicz and Amit Sheth. Specification and execution of transactional workflows. pages 592–620, 1995.

[135] Douglas C. Schmidt, David L. Levine, and Sumedh Mungee. The Design of the TAO Real-Time Object Request Broker. *Computer Communications*, 21:294–324, 1997.

[136] R. Srinivasan. Rpc: Remote procedure call protocol specification version 2. Technical report, United States, 1995.

[137] Shoaib Sufi and Brian Mathews. *CCLRC Scientific Metadata Model: Version 2*, August 2004.

[138] T.Andrews, F.Curbera, H. Dholakia, Y.Goland, J.Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S.Weerawarana. The open grid services architecture, version 1.0. Informational document, Global Grid Forum, January 2005.

[139] Luis M. Vaquero, Luis Rodero-Merino, Juan Caceres, and Maik Lindner. A break in the clouds: towards a cloud definition. *SIGCOMM Comput. Commun. Rev.*, 39:50–55, December 2008.

[140] Tom White. *Hadoop The definitive Guide*. O'Reilly, 2009.

[141] Jun Yan, Yun Yang, and G.K. Raikundalia. Swindew-a p2p-based decentralized workflow management system. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 36(5):922 –935, sept. 2006.

[142] Yun Yang, Ke Liu, Jinjun Chen, J. Lignier, and Hai Jin. Peer-to-peer based grid workflow runtime environment of swindew-g. In *e-Science and Grid Computing, IEEE International Conference on*, pages 51 –58, dec. 2007.

[143] Y Ying, Y.and Huang and D. W. Walker. A performance evaluation of using soap with attachments for e-science. In *Proc. of the UK e-Science All Hands Meeting, 2005*, 2005.

[144] Donglai Zhang, Paul Coddington, and Andrew Wendelborn. Data transfer performance issues for a web services interface to synchrotron experiments. In *SOCP '07: Proc. of the 2007 workshop on Service-oriented computing performance: aspects, issues, and approaches*, pages 59–66, New York, NY, USA, 2007. ACM Press.

Appendix A

Compare the standard WSDF grammer with the suggested WSDL Grammar with Third-party Data Forwarding Element:

```
<wsdl:definitions ....>
    <wsdf:portType ....  > *
      <wsdl:operation name="nmtoken"
           parameterOrder="nmtokens">
        <wsdl:input name="nmtoken"?
         message="qname"/>
        <wsdl:output name="nmtoken"?
         message="qname"/>
        <wsdl:fault name="nmtoken"
         message="qname"/>
      </wsdl:operation>
    </wsdl:portType >
    ......
</wsdl:definitions>
```

Figure 10.2: Standard WSDL Grammer For a Request-Response Operation

The following is a suggested WSDL grammer used in this work.

```
<wsdl:definitions ....>
    <wsdl:portType ....  > *
      <wsdl:operation name="nmtoken"
         parameterOrder="nmtokens">
        <wsdl:input name="nmtoken"?
         message="qname"/>
        <wsdl:output name="nmtoken"?
         message="qname"/>
        <wsdl:fault name="nmtoken"
         message="qname"/>
        <wsdl:forward name="nmtoken"?
                      message="qname" />
      </wsdl:operation>
    </wsdl:portType>
  </wsdl:definitions>
```

Figure 10.3: Suggested WSDL Grammer with Third-party Data Forwarding Element