Future Generation Computer Systems 27 (2011) 32–39

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

LAG: Achieving transparent access to legacy data by leveraging grid environment

Yuhui Deng [a,*], Frank Wang [b]

[a] Department of Computer Science, Jinan University, Guangzhou, 510632, PR China

[b] Cambridge-Cranfield High Performance Computing Facility, Cranfield University Campus, Bedfordshire MK430AL, United Kingdom

ARTICLE INFO

Article history:
Received 30 September 2007
Received in revised form
2 July 2010
Accepted 10 July 2010
Available online 16 July 2010

Keywords:
Legacy data
Grid
Architecture
Grid service
Web service

ABSTRACT

The world today is experiencing an explosive growth of data generated by information digitization. Due to the unprecedented advance in software and hardware, large amounts of data gradually becomes legacy data and inaccessible. This is building a digital black hole, and it is becoming a big challenge to access, process, and preserve the legacy data. Grid provides flexible, secure, and coordinated resource sharing among dynamic collections of individuals, institutions, and resources. It allows users and applications to access the aggregated resources in a transparent manner. This paper proposes a Legacy Application Grid (LAG) architecture. This architecture deploys diverse legacy applications in a grid environment and provides a transparent access to the remote LAG users who want to access the legacy data. In contrast to the existing methods which attempt to tackle legacy data and legacy applications, we wrap a display protocol into grid services. The service provider, who wants to deploy any legacy applications, just needs to deploy the protocol based grid service, describe and pass the parameters of those legacy applications to the service. Compared with the traditional approaches, the method proposed in this paper is very cost-effective because it avoids converting legacy data from one format to another format or upgrading legacy applications one by one. An implemented prototype validates that the LAG architecture trades acceptable performance degradation for a transparent and remote access to legacy data.

1. Introduction

According to a new report from IDC, 161 exabytes of digital information were created and copied in 2006. The growth will continue to increase exponentially. The amount of information in 2010 will surge more than six fold to 988 exabytes which amounts to a compound annual growth rate of 57% [1]. The explosive data is normally stored in autonomous repositories distributed across the Internet and varies in representation from structured (e.g. relational database) to semi-structured (e.g. e-mail and HTML pages) and unstructured formats (e.g. image and video) [2]. The ubiquitous Internet has provided an easy access to a large number of autonomous and heterogeneous information sources [3]. However, due to the unprecedented development of software and hardware, large amounts of legacy data is becoming a big challenge which we have to face when accessing the digital information. (Legacy data is the data which has been inherited from applications, software, languages, platforms, and techniques earlier than current technology.) The National Archives of the United Kingdom, which holds 900 years of written material, has more than 580 terabytes of data in legacy data formats that are no longer commercially available. Some digital documents held by the national archives had already been lost forever, because the programs which could access them no longer exist [4]. There are two reasons which cause the problem. The first one is the range of proprietary data formats that proliferated during the early digital revolution. The different data formats do no work together, which makes interoperability a big problem. The second one is that the data formats employed by software companies are not only incompatible with that of the rival companies, but also between different generations of the same program (e.g. Microsoft) [4,5].

The growing legacy data has propelled research on how to access, process, and preserve the legacy data. Some research efforts have been invested in tackling the growing challenge. Saving data in one format with one program makes it difficult to open in another program without sacrificing some information. Extensible Markup Language (XML) [6] offers portability and ease of machine processing. The wide spread and growing maturity of XML technologies bring new opportunities to tackle the legacy data. Chidlovskii and Fuselier [7] investigated data conversion from the rendering-oriented HTML markup into a semantic-oriented XML annotation defined by user-specific DTDs or XML Schema descriptions. They applied a supervised learning framework to the conversion task according to which the transformations are learned from a set of training examples. The data which are in proprietary formats such as PDF, MS Word, etc. have to be first converted to a standard format like HTML, and then the layout HTML annotations will be converted to the semantic XML.

* Corresponding author.
E-mail addresses: tyhdeng@jnu.edu.cn, yuhuid@hotmail.com (Y. Deng).

0167-739X/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2010.07.004

Kuikka et al. [8] developed a syntax directed approach to transform the XML documents from one structure to another. The aim is to automate a transformation between two grammars that have common parts, although the grammars and names of elements may differ. Other driving forces for the research on the legacy data are from industry. Open XML is a data format developed by Microsoft. This format can be adopted to save files from programs such as Word, Excel and Powerpoint. The open XML is an open international standard under independent control and is free for access [4]. Working with three partners, Microsoft also released a translation program which allows users to save Word documents in the ODF format favoured by the free Open Office application [5]. With support from Microsoft, the National Archive of the United Kingdom will be able to read older data formats in the format they were originally saved by running emulated versions of the older Windows operating systems on modern PCs. For example, if a Word document was saved using Office 97 under Windows 95, then the National Archives will be able to open that document by emulating the operating system and the corresponding software on a modern machine [4].

The legacy data and legacy application is normally in a one to one correspondence, which indicates that a specific legacy data format can only be accessed by the corresponding legacy application. If the legacy applications can be upgraded, the corresponding legacy data will be solved as well. A lot of research efforts have been invested in tackling the legacy applications. Generally, the existing solutions can be classified into three categories [9]. The first one is redevelopment. This method rewrites or reconstructs the existing legacy applications. The common activities include parsing the system and analyzing its syntax to obtain an abstract syntax tree representation of the source code, extracting the interface fragments from the system, and performing control flow analysis [10,11]. The redeployment method requires shutting down the legacy applications either during development or during the replacement [9]. The second one is wrapping. This approach surrounds the legacy component with a new interface. Thiran et al. [12] proposed and designed a generic and technology independent R/W wrapper architecture. The wrapper allows a smooth transition from the legacy and deficient databases to modern architectures, and makes the integration of a legacy database into current large applications easier. The third one is migration. This solution moves legacy applications to a new environment, while retaining the original system's data and functionality. Wrapping and migration are normally employed to reuse legacy applications. One or more approaches could be involved when tackling the legacy applications. Bi et al. [11] investigated a hybrid approach of wrapping and migration for the reuse of legacy applications from its original environment to the Internet-based platform based on a thin client using Java RMI.

However, for those companies and organizations that have a large number of diverse and legacy data, the conversion of all legacy data or the upgrading of all legacy applications could take a lot of time and money, and raise many technical problems. For personal users who have a little legacy data or just want to use the legacy data temporarily, it is not cost-effective to buy a software package to convert the data or upgrade the corresponding legacy application.

Grid is a flexible, secure, coordinated resource sharing among dynamic collections of individuals, institutions, and resources [13–16]. The objective is to virtualize resources, and allow users and applications to access shared resources in a transparent manner. In recent years, the research community has been very active in the area of investigating techniques in tackling the legacy applications in a grid environment. Kacsuk et al. [17] proposed a new approach to deploy legacy codes as grid services without modifying the original code. A workflow oriented grid portal is designed to apply the legacy code based grid services to complex business processes. Huang et al. [18] designed a wrapping and data mapping technique for converting the existing legacy code (e.g. libraries of scientific and mathematical software written in C language) into composing computational services within a grid environment. Bodhuin and Tortorella [19] designed a tool which can automatically transform the source code of legacy applications and make them compatible with the web or grid technologies. GEMLCA [20] is a front end OGSI grid service layer which surrounds the target host environment and executes legacy applications through the OGSI grid service. Plantikow [21] proposed a data management system architecture and discussed approaches for the integration of legacy applications and grid scheduling with the proposed architecture. An integrator is designed to instruct the VFS driver to add a new logical file system view. Such views are used to provide the legacy applications with input data and to collect the results. Each legacy program is run inside a jail/sandbox such that it only accesses its logical view. LGF [22] is a two-tier architecture in which the interface layer is decoupled from the legacy layer. This enables many benefits that surpass the performance penalty due to the additional interposition layer. The LGF enables semi-automatic virtualization of legacy codes as grid services. McGough et al. [23] proposed the use of a standards based job submission and monitoring system. This approach enables us to deploy legacy applications into the existing resources within a Grid, to map applications into Grid applications, and to use a web based portal to expose these applications to the end users.

In this paper, we propose a Legacy Application Grid (LAG) which is based on an existing grid environment (GT4) [15,24] consisting of a Monitoring and Discovery System (MDS) [25], a certificate authentication centre and several grid service providers. In contrast to the existing methods, we wrap a display protocol into grid service, which is registered in a MDS, instead of directly putting legacy applications into grid service or converting the legacy data from one format to another format. The service provider who wants to deploy any legacy applications just need to deploy the protocol based grid service, describe and pass the parameters (e.g. application name) of those legacy applications to the grid service. Therefore, all kinds of legacy applications can be deployed in the LAG without modifying the source code and the GUI. LAG users who want to access any legacy data can locate and discover the required legacy application in LAG, and then employ the application to access the corresponding legacy data transparently. The LAG can be maintained by companies or organizations. Thus, the method is very cost-effective for both the companies and personal users who want to access legacy data, because they just need to pay for what they use. A system prototype is constructed to investigate the overhead involved in the data access in LAG. The experimental results illustrate that the LAG can provide transparent access to the legacy data with acceptable performance degradation.

The remainder of the paper is organized as follows. Section 2 gives a brief description about the background knowledge of display protocols. Overview of the LAG architecture and workflow are introduced in Section 3. Section 4 describes how to build a legacy service, manage lifecycle and state, and service registration. Section 5 illustrates the prototype system and evaluates the system performance. Section 6 concludes the paper with remarks on the contributions of the paper. There is also a brief discussion in Section 6.

2. Background

Generally, there are two major display protocols which offer graphics sharing and guarantee network transparency: Virtual Networking Computing (VNC) [26] and X Window System (X11).

The VNC system is a simple protocol working in a client/server model. This protocol offers remote accesses to graphical user interfaces (GUI) and works at frame buffer level. The key point at the display side is putting a rectangle of pixel data at a given x, y position. Therefore, the encoding is simply an x, y coordinate. This gives a position in the frame buffer from which the client can copy the rectangle of pixel data. This encoding is typically used when the user moves a window across the screen or scrolls a window's contents. VNC sends compressed bitmaps across the network. X11 is a display protocol which provides a standard toolkit to build GUI for network computers. As a user application on top of operating systems, X11 is network transparent and operating system independent. X11 also adopts a client/server model which can be run on the same computer or on different ones with different architectures and operating systems [27,28]. The X11 server provides display and I/O services (e.g. keyboard and mouse) to applications. The X11 client leverages these services to manipulate applications. Therefore, the X11 server sends user input to and accepts output requests from client programs through communication channels. The X11 server machine can run a small program that makes a connection to the remote client machine and starts the client applications. Alternatively, the remote X11 client is also able to connect to an X11 server. Manipulating graphics in X11 is at relatively high-level elements including lines, rectangles, curves, and fonts. This is different to the systems which read and write individual pixels. X11 can direct graphics to windows or pixmaps which work in different ways. Drawing to a window generates a visible result, while drawing to a pixmap simply updates the pixmap memory which is invisible. Pixmaps can be used to store frequently drawn images and as temporary backing-store for pop-up menus. Furthermore, it can animate a series of images by placing the images into pixmap memory and then sequentially copying them to a visible window [27].

3. System overview

3.1. Architecture of LAG

A grid environment may consist of hundreds or even thousands of geographically distributed and heterogeneous resources to match the requirements imposed by all kinds of grid applications [14,16]. Grid is a Virtual Organization (VO) consisting of cooperating geographically distributed and diverse resources that combine into a logical community with only minimal administrative requirements. LAG is based on an existing Grid environment GT4. Therefore, LAG inherits the characteristics of GT4. The main goal of LAG is providing legacy applications as resources which can be located and employed by LAG users to access the legacy data transparently.

LAG is composed of Service Providers (SPs) which offer diverse legacy applications, MDS, CA centre, and LAG users. The Grid Security Infrastructure (GSI) [29] provides security functions including single or mutual authentication, confidential communication, authorization, and delegation. The security of LAG is within the GSI framework of GT4. Each service provider has to request and receive a host certificate from the CA, thus allowing joining the LAG. In order to access the legacy applications in LAG, a user has to request and receive a user certificate from the CA as well. Fig. 1 illustrates the architecture of LAG which consists of three sub-VOs. The three sub-VOs are connected with each other through three MDSs. A downstream and upstream mechanism is employed by the MDSs to exchange information automatically and efficiently. The interaction between LAG users and the LAG is mediated through a LAG service published on the Internet. When a user wants to access the LAG, firstly, it sends a request to the LAG service with its requirements. Secondly, the LAG service redirects

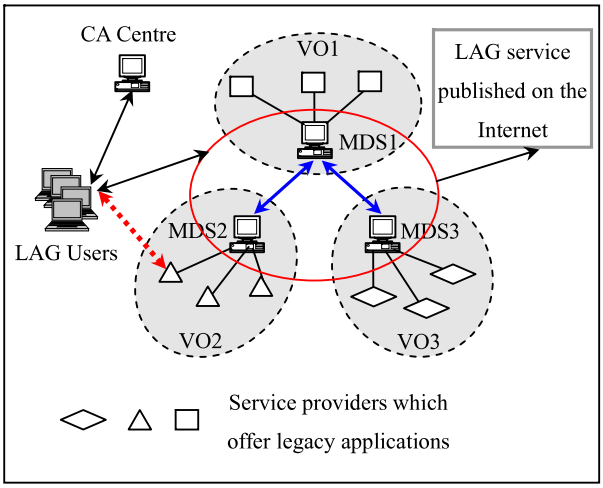

Fig. 1. Architecture of the legacy application grid.

the request to an MDS which is close to the user. Within the LAG environment, the users can achieve the following functions through the VO interface: (1) obtain a security certificate from the certificate authority. (2) With the help of MDS, search and discover appropriate legacy applications in terms of their requirements. (3) Employ the located legacy application to access the legacy data transparently and remotely.

3.2. Workflow of LAG

Recently, some research efforts have been invested in solving the legacy applications in gird environment or service oriented architecture [17–23,30]. However, the methods are either costly or inflexible (e.g. converting the legacy data from one format to another format, wrapping or modifying the legacy application one by one). The main goal of LAG is allowing users to locate and employ an appropriate legacy application to access the corresponding legacy data transparently and remotely without modifying the source code and GUI of the legacy applications. Therefore, we do not use the traditional methods including redeployment, wrapping, and migration. We develop a display protocol based grid service which is called a legacy service in the remaining discussion of this paper. The legacy service can be deployed in the LAG at the service provider side. The protocol can be adopted by LAG users to launch an application remotely and redirect the original GUI of the legacy applications to the users. Some parameters such as application name of the legacy applications are passed to the legacy service, which enables the LAG users to locate the legacy applications.

In a typical grid environment especially the LAG, the service providers could have diverse and heterogeneous platforms and operating systems. The X11 offers a very good opportunity to access the heterogeneous service providers which provide legacy applications in LAG, because almost all modern operating systems and architectures support X11. Unfortunately, the network traffic between the X11 client and X11 server is not secured, whereas grid environment requires high security due to its geographic distribution and crossing domains. The data traffic between the X11 server and X11 client is secured by a secure shell and GSI protocol in our implementation. Fig. 2 illustrates that the X11 server and X11 client are deployed on LAG user and service provider, respectively. Service is becoming a basic application pattern of grid because the service offers a standard means of interoperating between different applications running on a variety of platforms. The X11 client is wrapped in a legacy service. Each service provider in the LAG has to lunch the legacy service

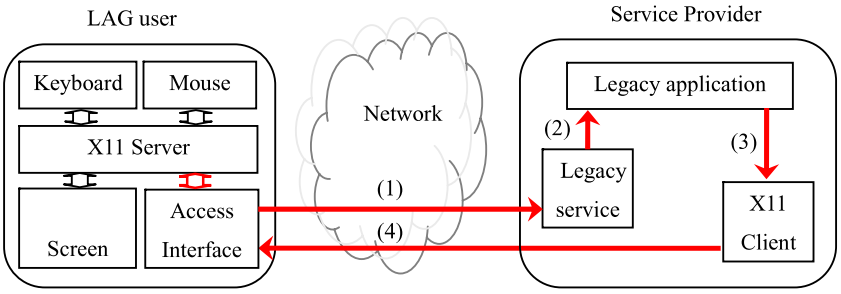

Fig. 2. X11 server and client deployed on LAG user and service provider.

which is registered in a MDS. The available legacy applications are published in the legacy service by setting the application related parameters. The LAG users can locate the published legacy applications through the legacy service with the help of MDS. When a user submits a request with corresponding requirements to the MDS, the MDS will return a list of all matched legacy applications associated with their Universal Resource Identifier (URI) by querying the available legacy service. The legacy applications can be geographically distributed and transparent to the user. If the user chooses a legacy application from the list, the remaining steps are labelled with a sequence number as defined in the following descriptions (see Fig. 2). (1) The URI is employed by the user to locate the service provider which offers the legacy application chosen by the user. A connection between the service provider and the user will be initialized and established. The user then sends a command to request the legacy application which is published with the legacy service. (2) When the legacy service receives the command, it will launch the requested legacy application. (3) The requested application creates a new process and registers itself with the X11 client to display its GUI. (4) The X11 client sends the GUI stream to the user with a secured network connection. The LAG user can then display the GUI on its screen as if it is processed locally.

There are two major jobs in a typical Grid environment. The first one is a batch job which stores output/error streams into remote files. The files can be retrieved after the job is completed. Batch jobs are normally adopted when multiple jobs are launched in parallel, or when the execution time is expected to be very large. The second one is an interactive job which provides immediate feedback to Grid users. A Grid job submission normally involves a client, a gatekeeper, and a job manager. The gatekeeper is a remote service that receives requests from clients, and performs mutual authentication with the clients. After authenticating and authorizing, it starts a job manager running under the credentials of the authenticated user. Therefore, a job manager is spawned by the gatekeeper upon receiving each request. The job manager processes job specifications sent by the clients, most of which result in a job submission to a local scheduler. It also provides a mechanism through which the client can check the status of a job or cancel it [31]. The key point of job submission in Grids is sending programs to the resources to execute. LAG works in a different way. The reason is because legacy applications sometimes are tightly coupled with the running environment. It would be a challenge to decouple the legacy applications from its hardware platform and switch to a new platform where the legacy data resides. As mentioned before, LAG users leverage the legacy service to locate the published legacy applications with the help of MDS. After this is done, the users who have legacy data to process will send a request to the legacy service to start the corresponding legacy application. When the application is launched, X11 protocol will take over the whole process. Therefore, in contrast to the Grid job submission, it is unnecessary for LAG to send the programs (legacy applications) to the resources to execute.

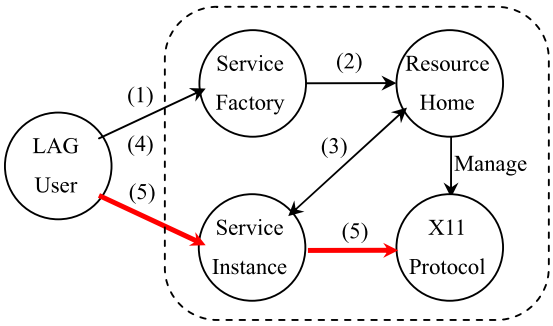

Fig. 3. Components and workflow of the legacy service.

4. Legacy service

4.1. Building a legacy service

As discussed in Section 3.1, the LAG user locates the available legacy applications through a legacy service with the help of MDS. The legacy service consists of four parts that are implemented by four classes: A legacy service factory, a legacy service instance, a legacy service home, and the X11 protocol (see Fig. 3). When dealing with multiple resources, the WSRF specifications recommend employing the factory/instance pattern that is a well-known design pattern in software design, and especially in object oriented languages. The factory/instance pattern adopts one service (the service factory) in charge of creating the resources and another one (the service instance) to actually access the information contained in the resources. Because the SPs offer resources for hundreds of thousands of LAG users with different requirements, the factory/instance pattern is employed in the legacy service. The resource home is responsible for registering and updating the MDS when it is necessary. (e.g. When a new legacy application is added to the legacy service, the resource information in the SP has to be updated in MDS.)

Fig. 3 illustrates the components and workflow of the legacy service. When an LAG user obtains the URI of a requested legacy application through MDS, the major steps are defined in the following descriptions. (1) The URI is employed by the LAG user to locate the legacy service factory of the SP chosen by the user. (2) The legacy service factory invokes a createResource method to create and initialize a new resource assigned with a unique ID through the legacy resource home. (3) The legacy resource home which is responsible for all the legacy resources (X11 protocol) will create a new legacy service instance associated with an URI. (4) Once the createResource call finishes, a WS-Addressing EndPointReference (EPR) [32] containing the URI of the created legacy service instance and the ID of the allocated legacy resource (X11 protocol) will be sent to the user. (5) The legacy service instance will employ the legacy resource home to find the exact legacy resource (X11 protocol) and provide methods to operate the newly created legacy resource.

Table 1
System configurations of the testbed.

	Computer 1	Computer 2	Computer 3
CPU	Intel 550 MHz	Intel 1.6 GHz	Centrino duo 1.66 GHz
Memory	128 MB	256 MB	1 GB
Network	100 Mbit/s	100 Mbit/s	100 Mbit/s
OS	Red Hat (Kernel 2.4.21)	Red Hat (Kernel 2.4.21)	Red Hat (Kernel 2.4.21)
Middleware	Globus Toolkit 4	Globus Toolkit 4	Globus Toolkit 4

4.2. State and lifecycle management of the legacy service

Grid service must provide their users with the ability to access and manipulate state. WSRF adopts Web Service Resource (WS-Resource) consisting of a resource document and a corresponding web service to implement a stateful service. Because web service is stateless, the resource document employs an XML schema to capture state information for a WS-Resource due to the portability and ease in machine processing. The web service can check and alter states contained in the resource document.

The WS-Resource is adopted in the legacy service to describe and access any state of the legacy resources. Two kinds of important state information are managed by the legacy service. The first one is the information of the resource utilization. A service provider in the LAG is supposed to provide legacy applications for hundreds of LAG users. The service provider could become a system bottleneck due to the overload of data traffic. Therefore, the service provider monitors the utilization of the hardware resources periodically and stores the state. The stored state will be accessed by the MDS when the MDS locate the available legacy applications in terms of the query requirements. The applications residing in the service providers which could be overloaded will be filtered by the MDS and are invisible to the LAG users. Because each instance of the legacy service corresponds to a process of a legacy application, the second state information is the process. A launched process in an operating system is identified by a unique number called Process ID (PID). The legacy service tracks the PID and records the corresponding information for three reasons. The first, the service provider is able to monitor which user is running which process. When it is necessary, the process can be killed due to some reasons (e.g. security). The second, the service can calculate which user used which legacy application and how long the user used it. The information is important for charging the user appropriately. The third, the service provider is able to calculate how many users are currently using the licensed legacy application, thus limiting the simultaneously running processes.

WSRF lifetime management is employed to describe the resources that are destroyable via grid services interfaces. The lifecycle management of the legacy service is composed of two cases. An instance of legacy service is created every time when a LAG user initiates a connection with a service provider. In the first case, a legacy service instance allocated for a specific user can be destroyed immediately by means of the user's requirement. For example, when a user finishes data access or wants to terminate the data access immediately. In the second case, when a user requests some legacy resources with specified time period, a scheduled destruction is adopted to calculate the destruction time. If the termination time is arrived, a message will be sent to the user. If the user does not want to renew the legacy service instance, the instance will be destroyed and the allocated resources will be reclaimed.

4.3. Legacy service registration

A typical grid environment (e.g. GT4) is a VO. The interaction between the VO and grid users is mediated through MDS which provides a virtual interface between the diverse resources and maintains a single logical view. Therefore, all grid services have to be registered in the MDS. There is typically one MDS per VO, but in a large VO, several MDSs are normally organized in a hierarchy. MDS provides service discovery, execution supervision, and monitoring of resource status information. A very important objective of LAG is enabling authorized LAG users to locate the required legacy applications across the geographically distributed service providers in LAG, and then employ the legacy applications to access the corresponding legacy data. Therefore, in order to discover the legacy applications which are published in the legacy service, the legacy service has to be registered in MDS.

The purpose of the service container is to shield the application from environment specific run-time settings, control the lifecycle of services, and dispatch of remote requests to service instances. A grid service provided by a service provider is first registered in the local service container, and then registered in the MDS container. There are two important services which are employed by a container to monitor and discovery resources in grid environment. One is container registry service which keeps tracks of all services running in the local container. Another one is default index service which can collect local resource information and remote resource information with the help of the upstream and downstream mechanism.

After the X11 protocol is wrapped as a legacy service, it is important to register the service in MDS to make it available to the users. The registration process can be divided into four steps: (1) Creating a default instance for service provider. (2) Updating the instance by including the newly added legacy service. (3) Registering the instance in the default index service of the local service container of service provider. (4) The MDS container receives information from the registered downstream or upstream connection and registers the default instance of the new service provider. Finally, the LAG users can discover all available resources by querying the MDS.

5. Performance evaluation

We constructed a system prototype with three computers which were connected through a 100 M switch. All computers were installed with Redhat and Globus Toolkit 4. Table 1 shows the system configurations of the three computers. All the performances reported in this paper are based on the average of 100 measurements.

5.1. Evaluation of service registration

In the test of this section, the three computers all play the role of service provides to investigate the impact on the registration process of the different computer platforms. The legacy service can be registered in the service provider locally or remotely by the clients who want to deploy legacy applications in LAG. We will explore the exact overheads involved in the registration process by registering locally which excludes the network overhead.

As discussed in Section 4.3, the registration process can be divided into 4 steps. We will investigate the overheads of the first three steps. The fourth step actually depends on the poll interval of MDS. We set the poll interval as 60 s which indicate that the MDS checks the downstream or upstream connections every 60 s. There are four legacy applications which were updated in the second step of our test. Fig. 4 illustrates the average registration overheads of

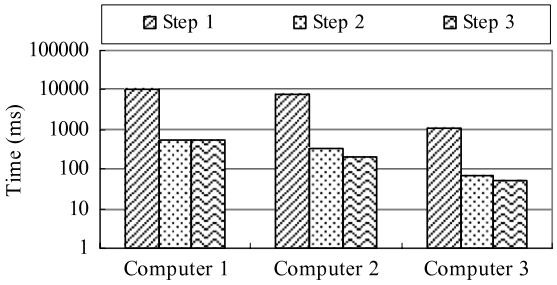

Fig. 4. Average registration overhead.

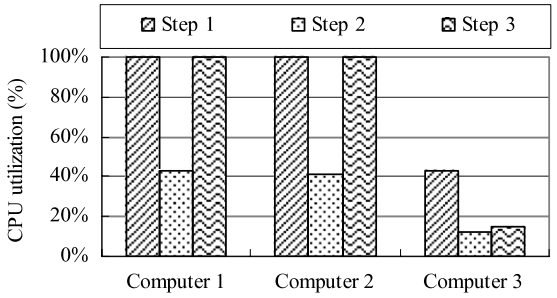

Fig. 5. CPU utilization.

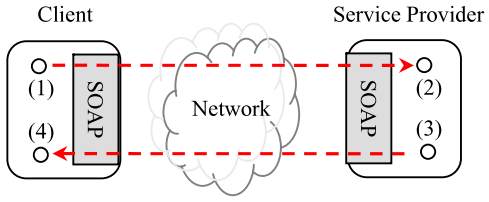

Fig. 6. Test scenario of a grid service invocation.

the three steps on three computers, respectively. Fig. 5 shows the processing power consumed by the three steps on three different computers. The above tests demonstrate that the registration is very resource consuming (e.g. the CPU utilization of computer 1 and computer 2 both reach 100% when dealing with the step 1 and step 3) which is mainly caused by the SOAP messages.

By using grid service, it is easy to construct a heterogeneous and Internet-scale system which guarantees interoperability. The core of grid service is Extensible Markup Language (XML) [6] which offers portability and ease of machine processing, because both the Web Services Description Language (WSDL) [33] and Simple Object Access Protocol (SOAP) [34] are based on XML. Due to the involved XML, requests and replies become much larger and parsing the XML messages on both the sender and the receiver side incurs additional overhead. Tian et al. [35] discovered that sending 589 bytes of content involves additional 3363 bytes by using web service. Mani and Nagarajan [36] reported that XML's way of representing data takes more than 400% overhead compared with the way adopted by binary. Therefore, it is very important to explore the exact overheads involved in the processing of XML.

Fig. 6 shows the test scenario of a typical grid service invocation. When a client needs to invoke a grid service, it wraps the required data into SOAP messages and sends the messages with HTTP protocol through network to the service provider. The service provider then analyzes the SOAP messages, extracts the data, and uses the data to invoke the corresponding service. The results of the invocation are transferred back to the client with SOAP messages through the network. Finally, the client extracts the response data from the SOAP messages. In our experiment, in order to eliminate the network overhead, the client software which is used to deploy legacy services is located in the same computer with the service provider. The overheads involved in the three steps of the registration is measured by two fine granularity timers (Timer 1 and Timer 2) which reside in the source code of the client software and service provider.

Table 2
Registration overheads and the corresponding SOAP overheads.

	Step 1 (ms)		Step 2 (ms)		Step 3 (ms)	
	Overall	SOAP	Overall	SOAP	Overall	SOAP
Computer 1	10 140	10 131	533	532	522	466
Computer 2	8 110	8 106	331	330	195	160
Computer 3	1 063	1 061	68.2	68	51	33

Table 3
Query overhead of a single request submitted to MDS.

	Query time (ms)	CPU utilization (%)
Computer 1	1460	62
Computer 2	1105	34
Computer 3	821	8

The major steps of test are labelled with the sequence number as defined in the following descriptions (see Fig. 6). (1) Timer 1 in the source code of client software is started when the client starts executing a function provided by the service provider. (2) When the service provider obtains the data from the SOAP messages which are from client and begin to invoke the required service, the Timer 2 starts to work. (3) When the service provider finishes the required operation and starts to wrap the information into SOAP messages, Timer 2 stops working. (4) After extracting the data from the SOAP messages sent by the service provider, the Timer 1 is stopped. According to the above descriptions, it is easy to calculate the overheads of processing the SOAP messages at the client side and the service provider side as ((4)–(1))–((3)–(2)).

Table 2 shows the overall overheads and the corresponding SOAP overheads involved in the three steps of the registration. Basically, the processing overhead of SOAP messages is a big portion of the overall overhead ranging from 64.7% to 99.9%. Table 2 also depicts that the overheads are decreased with the increase in performance of different hardware platforms. The above tests demonstrate that the registration is resource consuming. Fortunately, the overhead is transparent to the LAG users, and the registration does not happen frequently. It may delay the registration process when the service is available, but it is unlikely to interfere with the users.

5.2. Evaluation of service query

The query response time seen by the LAG users is an important metric in evaluating the performance of resource discovery of the LAG. The measured query time is the time between when a user submits a query request to the LAG through network and when the last byte of the response is delivered to the user. We employed average query time to measure the performance.

We employed the three computers listed in Table 1 as MDS and used another computer to investigate the query overhead, respectively. Table 3 shows the query overhead of a single request to the MDS. The first column illustrates the query time with different computers. The second column depicts the CPU utilization of the three MDSs when they are processing the query request. The trend is that the query time seen by the user and the CPU utilization of MDS is decreased with the improvement of the hardware platform of computers. According to Table 3, we believe that to a great extent, the query time is determined by the

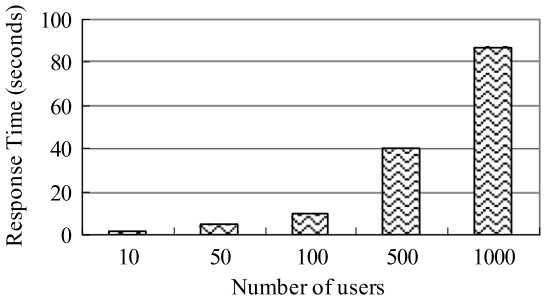

Fig. 7. Average query time.

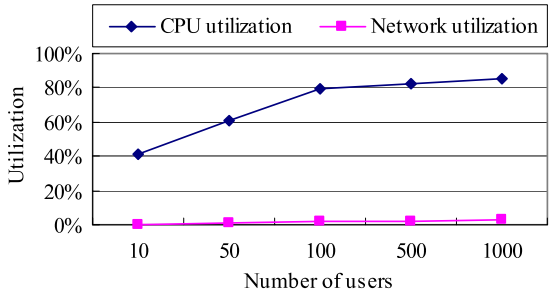

Fig. 8. CPU and network utilization.

processing power of MDS. The reason is that the large processing overhead of the query request is caused by the SOAP messages sent by the users.

The LAG is designed to support hundreds of thousands of users. However, if hundreds of thousands query requests go to a particular MDS simultaneously, the MDS could become a potential system bottleneck. The I/O traffic travelling network and LAG is also a big concern. Therefore, it is important to investigate how many simultaneous query requests a MDS can support, and the corresponding network overhead involved in the query process. We used computer 3 which has the highest performance as the MDS and other computers as client machines. In the experiment, each client machine can simulate many different users with different connections to the MDS.

Fig. 7 illustrates that the average query time of 10, 50, 100, 500, and 1000 users are 1.5, 5.2, 9.8, 39.8, and 86.5 s, respectively. The query time here is the time between when the first user submits a query request and when the last user receives the query result. Fig. 8 depicts the CPU utilization of the MDS and the network utilization when 10, 50, 100, 500, and 1000 users simultaneously send query requests to the MDS, respectively. It shows that 10, 50, 100, 500, and 1000 simultaneous query users incur 41%, 61%, 79%, 82%, 85% CPU utilization of the MDS. The 85% CPU utilization of MDS results in a significant performance penalty, i.e., 86.5 s of query time. Fig. 7 also illustrates that the network is not a performance bottleneck. Even when 1000 users send query requests to the MDS simultaneously, the network utilization is about 3%. The reason is that the query requests are all small messages. The above tests illustrate that the potential performance bottleneck of LAG is the processing power of MDS.

Nielsen [37] presented that 10 s is the limit for users to keep their attention on the task. Anything slower than 10 s needs a percent-done indicator as well as a clearly signposted way for the user to interrupt the operation. Delays of longer than 10 s are only acceptable during natural breaks in the user's work, for example when switching tasks. Therefore, the 86.5 s of query time is unacceptable when the MDS serves 1000 users. A solution to tackle the potential performance bottleneck is distributing the query requests across multiple MDSs (see Fig. 1). Therefore, the query traffic taken over by each MDS will be decreased with the increase in number of MDS.

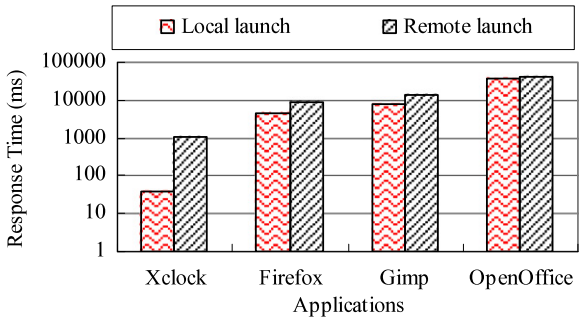

Fig. 9. Local and remote launch time of different applications.

5.3. Evaluation of service access

The objective of LAG is achieving transparent access to the legacy data by leveraging a grid environment. Therefore, it is very important to measure the performance between starting an application locally and in the LAG environment. The launch time of an application which is employed to access the corresponding legacy data is a very important metric to measure the performance of LAG. Local launch time denotes the time spent starting an application locally in the service provider. Remote launch time measures the time between when a user submits a request to a service provider to launch an application and when the application is successfully displayed on the screen of the client machine. Local launch time is measured as a baseline performance. Remote launch time is the performance of the LAG.

In our experiment, we employed computer 3 as a service provider and computer 2 as a client machine. Four different applications were launched from the service provider locally and remotely. The three applications including Firefox, Gimp, and Openoffice are normally adopted to access three typical data formats, i.e., web pages, pictures, and documents. The Xclock which is a simple application (timer) is employed to evaluate the overhead of very light applications involved in the LAG. Fig. 9 shows the local and remote launch time of the four different applications, where the *Y* axis is in logarithmic scale. As expected, launching applications in the LAG environment is longer than launching them locally. Compared with the local launch time, the remote launch times of the four different applications are increased 2610%, 103%, 83.2%, and 21.2%, respectively. The leftmost bar of Fig. 9 illustrates that the performance of Xclock is degraded significantly. The reason is that the local launch time of Xclock is too small, which makes the network transfer time become relatively big (e.g. The network transfer time of Xclock takes 96% of the remote launch time.). It is very interesting to observe that the performance degradation of heavy applications is alleviated. It is because that the network overhead is relatively small compared with the large local launch time. (e.g. Openoffice spends only 17.5% of the remote launch time on network transfer.)

The legacy service introduces a fixed amount of overhead on each instance. The less data being transferred, the higher this overhead is relative to the data transfer portion of the legacy application. The test results demonstrate that it is more interesting to deploy heavy applications than light applications with the legacy service since the involved overhead is relatively too high for light applications.

6. Discussions and conclusions

Legacy data is growing explosively due to the rapid development of software and hardware. The legacy data and legacy program is normally a one to one correspondence, which indicates that a specific data format can only be accessed by the corresponding

program. In contrast to the traditional approaches, this paper proposed and designed a LAG architecture which can deploy diverse legacy applications in it. By employing the legacy applications deployed in LAG, the LAG users can access the legacy data like local access. A salient feature is that a display protocol is wrapped as a legacy service which could enable service providers to deploy any applications on heterogeneous platforms and operating systems. This approach avoids converting the legacy data from one format to another format or wrapping the legacy application in grid service one by one, because it is not cost-effective. Furthermore, the LAG is a win–win design, because some out of date computers can be used to host some old legacy applications which can be adopted by users to access the corresponding legacy data.

A prototype was constructed and evaluated. The experimental results demonstrate that it is feasible to build such a LAG architecture in terms of the required network bandwidth and processing power. The main goal of LAG is to make the software functionality available to all people who need it and who are authorized to use it at anytime and anywhere. Please note that the LAG is a general architecture, the current applications can also be deployed in LAG besides the legacy applications. By taking advantage of the LAG, the LAG users do not have to pay a full licence for a specific software package if they just need to use the software to access or process some data temporarily.

Service composition can significantly accelerate rapid application development, service reuse, and complex service consummation. Multiple atomic services can also be automatically combined together as a composite service in terms of requirements. If the legacy service is designed as an atomic service, the integration of legacy applications would not be a problem. This will be explored in our future work.

Acknowledgements

We would like to thank the anonymous reviewers whose insightful and constructive comments have significantly enhanced this paper. Thanks also to Adrian Ciura who constructed the initial platform used in this work. In addition, we are grateful to Prof. Peter Sloot for giving us the opportunity to clarify our thoughts.

References

[1] The expanding digital universe: a forecast of worldwide information growth through 2010. IDC white paper—Sponsored by EMC. March, 2007. http://www.emc.com/about/destination/digital_universe/.
[2] A.P. Sheth, V. Kashyap, T. Lima, Semantic information brokering: how can a multi-agent approach help? in: Proceedings of the Third International Workshop on Cooperative Information Agents III, 1999, pp. 303–322.
[3] Y. Deng, F. Wang, Opportunities and challenges of storage grid enabled by grid service, ACM SIGOPS—Operating Systems Review 41 (4) (2007) 79–82.
[4] Warning of data ticking time bomb. July, 2007. http://news.bbc.co.uk/1/hi/technology/6265976.stm.
[5] Word works with open file styles. February, 2007. http://news.bbc.co.uk/1/hi/technology/6323575.stm.
[6] Extensible Markup Language. http://www.w3.org/XML/.
[7] B. Chidlovskii, J. Fuselier, Supervised learning for the legacy document conversion, in: Proceedings of the 2004 ACM Symposium on Document Engineering, 2004, pp. 220–228.
[8] E. Kuikka, P. Leinonen, M. Penttonen, Towards automating of document structure transformations, in: Proceedings of ACM Symposium on Document Engineering, 2002, pp. 103–110.
[9] J. Bisbal, D. Lawless, B. Wu, J. Grimson, Legacy information systems: issues and directions, IEEE Software 16 (5) (1999) 103–111.
[10] N.F. Schneidewind, How to evaluate legacy system maintenance? IEEE Software 12 (1) (1998) 34–42.
[11] Y. Bi, M.E.C. Hull, P.N. Nicholl, An XML approach for legacy code reuse, Journal of Systems and Software 61 (2) (2002) 77–89.
[12] P. Thiran, J. Hainaut, G. Houben, D. Benslimane, Wrapper-based evolution of legacy information systems, ACM Transactions on Software Engineering and Methodology 15 (4) (2006) 329–359.
[13] I. Foster, C. Kesselman, S. Tuecke, The anatomy of the grid: enabling scalable virtual organizations, International Journal of High Performance Computing Applications 15 (3) (2001) 200–222.
[14] Y. Deng, F. Wang, A heterogeneous storage grid enabled by grid service, in: File and Storage Systems, ACM SIGOPS Operating Systems Review 41 (1) (2007) 7–13 (special issue).
[15] Y. Deng, F. Wang, N. Helian, S. Wu, C. Liao, Dynamic and scalable storage management architecture for grid oriented storage devices, Parallel Computing 34 (1) (2008) 17–31.
[16] Y. Deng, F. Wang, A. Ciura, Ant colony optimization inspired resource discovery in P2P grid systems, Journal of Supercomputing 49 (1) (2009) 4–21.
[17] P. Kacsuk, A. Goyeneche, T. Delaitre, T. Kiss, Z. Farkas, T. Boczko, High-level grid application environment to use legacy codes as OGSA grid services, in: Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing, 2004, pp. 428–435.
[18] Y. Huang, I. Taylor, D.W. Walker, R. Davies, Wrapping legacy codes for grid-based applications, in: Proceedings of International Parallel and Distributed Processing Symposium, 2003.
[19] T. Bodhuin, M. Tortorella, Using grid technologies for web-enabling legacy systems, in: Proceedings of the Eleventh Annual International Workshop on Software Technology and Engineering Practice, 2003, pp. 186–195.
[20] T. Delaitre, A. Goyeneche, P. Kacsuk, T. Kiss, G.Z. Terstyanszky, S.C. Winter, GEMLCA: grid execution management for legacy code architecture design, in: Proceedings of the 30th Euromicro Conference, EUROMICRO04, 2004, pp. 477–483.
[21] S. Plantikow, K. Peter, M. Högqvist, C. Grimme, A. Papaspyrou, Generalizing the data management of three community grids, Future Generation Computer Systems 25 (3) (2009) 281–289.
[22] B. Bali, M. Bubaka, M. Wegiel, LGF: a flexible framework for exposing legacy codes as services, Future Generation Computer Systems 24 (7) (2008) 711–719.
[23] A.S. McGough, W. Lee, S. Das, A standards based approach to enabling legacy applications on the grid, Future Generation Computer Systems 24 (7) (2008) 731–743.
[24] I. Foster, Globus toolkit version 4: software for service-oriented systems, in: IFIP International Conference on Network and Parallel Computing, in: LNCS, vol. 3779, Springer-Verlag, 2006, pp. 2–13.
[25] Monitoring and Discovery System. http://www.globus.org/toolkit/mds/.
[26] T. Richardson, Q. Stafford-Fraser, K.R. Wood, A. Hopper, Virtual network computing, IEEE Internet Computing 2 (1) (1998) 33–38.
[27] R.W. Scheifler, J. Gettys, The X window system, ACM Transactions on Graphics 5 (2) (1986) 79–109.
[28] X Window System. http://www.x.org/wiki/.
[29] V. Welch, Globus toolkit version 4 grid security infrastructure: a standards perspective, 2004. http://www.globus.org/toolkit/docs/4.0/security/GT4-GSIOverview.Pdf.
[30] D. Kuebler, W. Eibach, Adapting legacy applications as web services, 2002. http://www.ibm.com/developerworks/library/ws-legacy/?n-ws-1312.
[31] M. Feller, I. Foster, S. Martin, GT4 GRAM: a functionality and performance study, in: Proceedings of the 2007 TeraGrid Conference, 2007.
[32] WS-Addressing. http://msdn2.microsoft.com/enus/library/ms951225.aspx.
[33] Web Services Description Language. http://www.w3.org/TR/wsdl.
[34] Simple Object Access Protocol. http://www.w3.org/TR/soap/.
[35] M. Tian, T. Voigt, T. Naumowicz, H. Ritter, J. Schiller, Performance considerations for mobile web services, Computer Communications 27 (11) (2004) 1097–1105.
[36] A. Mani, A. Nagarajan, Understanding quality of service for web services. January, 2002. http://www-106.ibm.com/developerworks/library/ws-quality.html.
[37] J. Nielsen, Usability Engineering, Morgan Kaufmann, San Francisco, 1993.

Yuhui Deng is a professor at the computer science department of Jinan University. Before joining Jinan University, he worked at EMC Corporation as a senior research scientist from 2008 to 2009. He worked as a research officer at Cranfield University in the United Kingdom from 2005 to 2008. He received his Ph.D. degree in computer architecture from Huazhong University of Science and Technology in 2004. He has authored and co-authored two book chapters and more than 20 refereed academic papers. He is on the editorial board of International Journal of Grid and High Performance Computing and a book titled Grid Technologies and Utility Computing: Concepts for Managing Large-Scale Applications. He has served as a committee member for several professional conferences in the field. He is also a reviewer of several academic journals. His research interests cover green computing, data storage, computer architecture, Grid Computing, performance evaluation, etc.

Frank Wang is the director of Centre for Grid Computing, Cambridge-Cranfield High Performance Computing Facility (CCHPCF), Cranfield University. He is Chair in e-Science and Grid Computing. He is on the editorial board of IEEE Distributed Systems Online, International Journal of Grid and Utility Computing, International Journal of High Performance Computing and Networking, and International Journal on Multiagent and Grid Systems. He is on the High End Computing Panel for the Science Foundation Ireland (SFI). He is the Chair (UK & Republic of Ireland Chapter) of the IEEE Computer Society.