行政院國家科學委員會專題研究計畫  成果報告

適合臺灣中小企業的自由軟體  ⚫  ⚫小世界網格  ⚫物流驢

子：第一階段

計畫類別：個別型計畫
計畫編號：
執行期間： 20年 0月 00日至 20年 0月 00日
執行單位：國立高雄第一科技大學運籌管理系

計畫主持人：陳彥銘
共同主持人：楊政穎

報告類型：精簡報告

處理方式：本計畫可公開查詢

中華民國 20年 0月 00日
國科會自由軟體專案

成果報告

適合臺灣中小企業的自由軟體 — “小世界網格”物流驢子：

：“第一階段”
NSC 93-2218-E-327-006

陳彥銘
國立高雄第一科技大學運籌管理系
楊政穎
國立虎尾科技大學資訊工程系

Department of Engineering and Applied Science
National Science Council, Taiwan

2005/10/15
行政院國家科學委員會補助專題研究計畫
●成果報告     □期中進度報告

適合臺灣中小企業的自由軟體 ─ “小世界網格”物流驢子: “第一階段”

計畫類別：■ 個別型計畫     □ 整合型計畫
計畫編號：NSC 93-2218-E-327 -006
執行期間：93年10月1日至94年9月30日

計畫主持人：陳彥銘
共同主持人：楊政穎
計畫參與人員：

成果報告類型（依經費核定清單規定繳交）：■精簡報告     □完整報告

本成果報告包括以下應繳交之附件：
□赴國外出差或研習心得報告一份
□赴大陸地區出差或研習心得報告一份
□出席國際學術會議心得報告及發表之論文各一份
□國際合作研究計畫國外研究報告書一份

處理方式：除產學合作研究計畫、提升產業技術及人才培育研究計畫、列管計畫及下列情形外，得立即公開查詢
□涉及專利或其他智慧財產權，□一年□二年後可公開查詢

執行單位：國立高雄第一科技大學

中華民國94年10月15日
I. Abstract

Full perspectives were covered in this report. Proposed solution was reckoned on appealing Meta-modeling methodology having insight into the foundation of B2B integration problems.

Two expected objectives were fulfilled by building a logi-donkey system: 1. Peer Groups Discovery and Message Exchange, 2. Exchange RosettaNet XML documents. A logi-donkey system was built in java language on top of a well-established JXTA platform. A real B2B RosettaNet XML documents, 3A4, was exchanged successfully based on our system. We are seeking the future improvement and further application of this project works.

II. Introduction

This system is a grid data exchange platform based on data/service federation and semantic integration over the internet. In other words, this system not only develops open source software but also makes full use of open source protocol and data.

This system stores all parameters suitable for the framework to initiate an executable application. All data access will go through a standard SQL interface, for example, JDBC connection.

In terms of application data, this system will only be served as a virtual integration point of heterogeneous external open source data by the help of another subsystem, Web Invocation Gateway subsystem. All application data are represented as XML format and their access will go through the SF subsystem and leave a cache copy whenever it is necessary.

This Logi-Donkey System (LD) System consists of three subsystems:
1. Data and Repository Framework (DF)
2. System Framework (SF)
3. Quality Framework (QF)

1.2 Overview

This system is a grid data exchange platform based on data/service federation and semantic integration over the internet. In other words, this system not only develops open source software but also makes full use of open source protocol and data.
Traditionally, to build a real world application for a model designer, he or she has to fully understand the problem first and then match one or many of the theoretical problems to the real-world ones. Since most of the solution algorithms are formulated on theoretical bases, they are suitable for implementing an elegant computation scheme. However, there is a wide gap between the real world and the theoretical problems. For example, if one tries to model the bank operation corresponding to a TSP formulation, he or she will face the aforementioned problems. We attempt to form a bridge to connect the theoretical problems at one end and the real-world problems at the other end. The theoretical problems have their own semantic representation in a well-structured format. However, the real-world problems involve both implicit and explicit semantic representation, which is complicated enough to be noticed at once.

Instead of modeling all kinds of real world problems one by one, we need a systematic approach to tackle the problems and overcome the problems associated with them. There are two standard approaches for this purpose, namely generalization and abstraction. Generalization is a completion process while abstraction is a simplification process. A very intuitive generation approach involves sorting out similar as well as different features and then working on every feature. Each problem instance has its own requirements, so that it is necessary to disable some unwanted features to satisfy individual needs. Generalization usually applies to the situation in which there is a small amount of deviation and the problem needs to be studied very carefully. Since each feature has to be considered individually, this increases the complexity of the process. As the problem domain gets larger and broader, all unnecessary features start to tangle together, and the software system may behave incorrectly.

This system is a first trial of grid data exchange platform based on data/service federation and semantic integration over the internet. In other words, this system not only develops open source software but also makes full use of open source protocol and data.

Grid computing is a way of organizing computing resources so that they can be flexibly and dynamically allocated and accessed, often to solve problems requiring many organizations’ resources. The resources can include central processors, storage, network bandwidth, databases, applications, sensors and so on. The objective of Grid computing is to make resources available so they can be more efficiently exploited.
The advantage of sharing is clearest when the need for resources is unpredictable, short-term, or changes quickly, or where it is simply larger than any single organization’s capability to provide for it. For example, some problems that might take days to solve on a single installation’s processing resources can be reduced to a few minutes with the right kind of parallelization and distribution on additional allocated computational resources.

The application objectives for logi-donkey system include:

- Peer Groups Discovery and Message Exchange
  
  Each peer belongs to a peer group. Each peer is able to discover its global group. The majority of the time, however, we function within a particular community and can assume the presence of a limited number of appropriate peers. Discovered group announcements, like peers, are to be cached locally.

- Exchange RosettaNet XML documents
  
  Instead of simply being displayed, individual documents can be extracted from the message. Since inter-peer messages are nothing more than XML documents, this provides various interesting intersections with Web Services and SOAP service.

  Computation is always a big issue in the underlying theoretical frameworks. A very active research area is devoted to federate the computation effort with other service providers in order to overcome the exhaustive computation. This distributed computation framework becomes the core of our collaborative grid data exchange platform. Because of the complexity in computation, sometimes the fidelity of semantics needs to be sacrificed in order to take computational advantage from the existing algorithm. This approximation can be done in the problem domain or in the computation domain.

  This meta-modeling platform provides a framework to construct a meta-model as well as facilitate the modern software engineering with an increased flexibility of the meta-models. In order to demonstrate the meta-modeling methodology and its applications, we will apply the methodology to a real-world problem.

### III. Literature Review

Grid technologies have been widely adopted in scientific and technical computing (Foster 2004). Grid technologies and infrastructures support the sharing and coordinated use of diverse resources in dynamic, distributed virtual organizations (Chervensak 2001). The creation, from geographically distributed components operated by distinct organizations with differing policies, of virtual computing systems is sufficiently integrated to a health information federation. Grids can be characterized as a set of heterogeneous system federated over a wide-area network (Foster, Kesselman & Tuecke 2001). In contrast to the general Internet, such systems are usually interconnected using special high speed, wide-area networks in order to get the bandwidth required for their applications.

While the notion of grid computing is simple enough, the practical realization of grids poses a number of challenges. Key issues that need to be dealt with are security, heterogeneity, reliability, application composition, scheduling, and resource management (Buyya 2002). We try to elaborate the cross platform federation without sacrificing flexibility, scalability, reliability, and extensibility.
Implementing a grid data exchange engine is not a big problem but using it is. It is often difficult to design and develop a software system to solve a specific real-world problem. Many factors stand in the way to interfere the development process. Among these factors are human intervention, complex environment and translation of the semantics into manageable languages. Furthermore, the users often have ambiguous requirements and multi-faceted demands for the software system. To compensate these, it is necessary for the software system developers to go through the interactive process between human experts and software systems back and forth many times. We can image that it is difficult to meet the users' expectations.

Among the modern software engineering tools, the object-oriented software engineering is the dominant method. This system will take advantage of modern methods to define a meta-modeling methodology in order to overcome the limitations of traditional methods.

Very large scaled information systems typically are deployed as centralized or distributed architectures. In a centralized architecture, a single server is responsible for processing all user requests. Centralized architectures simplify administration and coordination of a service, and they often require fewer resources. However, centralized architectures also have several disadvantages. A centralized server represents a bottleneck in processing that can delay response time. It may represent a single point-of-failure, and for adversaries, a single point-of-attack, without which the entire service fails. Finally, even if the server is resourceful and available, the routing function provided by the communications layer may fail to provide a path from some clients to the server.

Grid federation is gaining more and more attention during the past few years in a variety of R&D efforts. Any federated approach towards the creation of a B2B data exchange network environment should be capable of providing uniform ways for accessing authentic, physician-generated information that is physically located in different enterprise information systems (Brailsford, Lattimer, Tarnaras & Turnbull 2004). Grid computing makes extreme demands on distributed programming because they are typically large-scale, and they exploit wide-ranging networks consisting of a variety of protocols and systems which may span organizational boundaries. A type of architecture based on grid technologies is described below.

The logi-doneky grid architecture is shown in figure 2. There are two services in this architecture. Each service acts as a gatekeeper to actual information providers.

![Logi-doneky grid architecture](image-url)
The grid client shown in Figure 2 need not be an end-user system, but could be an agent acting on its behalf, and there may be many such clients or agents, acting independently with no central control over the components shown. To make use of resources, a client first uses information sources located within service brokers to discover those resources needed for execution of a task. Multiple sources may need to be consulted to locate all the resources needed for a computation. Assuming the discovery and allocation steps were successful, the client then sends the input data and executables, receiving a reference to the execution in return. These actions may be accomplished in several stages or as one consolidated action, depending on the nature and complexity of the task. As resources are allocated, the resource manager may need to update the information in the registry to enable reasonable bids for resource allocation from other clients. Lastly, the client monitors execution of the task using the reference it previously received. It can retrieve the results, or be sent status of the task as it progresses.

In order to fulfill the two objectives of this logi-donkey system at the first stage of development: (1) Peer Groups Discovery and Message Exchange, and (2) Exchange RosettaNet XML documents, the following architecture are proposed accordingly, an architecture are proposed as Figure 3.

![Logi-donkey architecture diagram](image)

Figure 3 Logi-donkey architecture

Each piece of shared content managed by Content Management Service is referenced by a unique content identifier, using a 128-bit MD5 checksum generated from the content data. In addition, each shared content item has an associated content advertisement which provides meta-information describing the content, including the content name, length, mime type, id, and description. Content advertisements are stored as XML documents as this example:

```xml
<?xml version="1.0">
<!doctype jxta:contentAdvertisement>
<jxta:contentAdvertisement>
  <name>3A4.xml</name>
  <cid>md5:1a8baf7ab82e8fe8fe2a2d9e7eb7a83</cid>
  <type>text/html</type>
  <length>23983</length>
  <description>rosettanet purchase order</description>
</jxta:contentAdvertisement>
```

The broker discovers resources, establishes their cost and capability, and then prepares a schedule to map
requesting jobs to resources. It identified a Service Broker first and assigned a job to it. A job has a task specification that specifies a list of operations to be performed. To process a job, the broker dispatcher deploys its Agent on that resource. The agent executes a list of commands specified in the job’s task specification. A typical task specification contains necessary commands and medical records from one of the agent resources, execution of the commands, and finally copying results back to the user. It can also contain special commands for accessing the records from the remote database. The data broker looks at the replica catalogue for a list of sites providing logi-doneky services, checks the status of those sites, and selects a suitable site. The infectious distribution fetch command can then request the logi-doneky service provider for a patient record and sent to outbreak detection federation. They should be deployable in forms flexible enough to meet the needs of incidents of various scopes and complexities; and provide useful and reliable information. Moreover, technologies, shared data, and terminology are standardized from a variety of agencies.

IV. System Ability

A logi-doneky system was built in java language on top of well-established JXTA platform. Almost every P2P application introduces a different protocol, replicating already done work and causes unnecessary incompatibilities [Konomi et. al.]. Since JXTA protocol is a good and prevalence one, we adopt it as our building block.

A. Configuration and Login

After starting the program from ant run command, configuration window appears. Figure 4 to Figure 6 show the screen shots of configuration windows. Figure 1 shows the peer name of the basic peer setting. Figure 5 shows the communication settings. Figure 6 shows the security settings of peer login.

Figure 4 Screen shot of basic peer settings.
Figure 5 Screen shot of communication settings.

Figure 6 Screen shot of security settings.

B. Man Window

After authentication process, the logi-donkey main window will appear automatically as showing on Figure 7.
C. Peer Groups Discovery and Message Exchange

Each peer belongs to a peer group. By default, each peer is a member of the global "NetPeerGroup" group, analogous to a world without area codes, where every phone number is globally unique. The majority of the time, however, we function within a particular community and can assume the presence of a limited number of appropriate peers. Discovered group announcements, like peers, are cached locally.

We share a test 3A4 XML document. A shared document icon will appear on all peers as showed in Figure 8.
D. Exchange RosettaNet XML documents

Instead of simply being displayed, individual documents can be extracted from the message as showed in Figure 9. Since inter-peer messages are nothing more than XML documents, this provides various interesting intersections with Web Services (via XML-RPC and SOAP), syndication (via RSS), Instant Messaging a la Jabber, and more.

![Figure 9 Screen shot of XML shared document display.](image-url)

**Conclusion**

Logi-doneky architecture based on grid technologies jointly replicates a logical service at many points in a network. Accordingly, some or all of the associated data also may be replicated. Because of replication, distributed applications offer opposite trade-offs. They are more likely to be available and with lower latency, have no bottlenecks, have no single points of failure or attack, and allow multiple routes to service. However, grid architectures entail designs that are more complicated, may be temporarily inconsistent, and may be difficult to
Proposed architecture achieved a common understanding of spatial data and processes yet effective supply chain integration. Grid federation enables responders to rapidly collect, process, and distribute information for more informed planning and better outcomes. Grid technology is likely to possess an increasingly importance role in public information exchange. GIS and RFID wireless technologies offer a promising solution to many of the technical challenges of using emergent technologies in accelerating the responsiveness for supply chain integration and management.

Reference


