AWERProcedia
Information Technology
& Computer Science

2 (2012) 336-341

2nd World Conference on Innovation and Computer Sciences 2012

Comparison of Distributed Technologies for Defining a Distributable and Interoperable ETL Framework

Mohd Syazwan Abdullah[a *], Mohammad M. I. Awad[b], Abdul Razak Saleh[c], Abdul Bashah Mat Ali[d], Azizah Ahmad[e]

[a,b,c,d,e]School of Computing, College of Arts and Sciences, Universiti Utara Malaysia, 06010 Sintok, Kedah, Malaysia

[a,b,c,d,e] ITU-UUM Center of Excellence for Rural ICT Development, Universiti Utara Malaysia, 06010 Sintok, Kedah, Malaysia

Abstract

Extraction, Transformation and Loading (ETL) are major functionalities in data warehousing. Lack of component distribution and interoperability are the main problems in the ETL area, because ETL components are tightly-coupled in the traditional ETL framework. This paper explores and discusses five popular distributed technologies for the purpose of highlighting the best technology that is capable of overcoming the distribution and interoperability gaps of the traditional ETL framework. Based on the comparison of distributed technologies discussed in this paper, several benefits can be obtained when SOA is used in redefining the ETL framework such as: distribution, interoperability, reusability, portability, and compatibility with legacy systems. These advantages and other SOA specifications are discussed in this paper.

Keywords: Distributed Systems; Data Warehousing; ETL; SOA.

Selection and/or peer review under responsibility of Prof. Dr. Dogan Ibrahim.

1. Introduction

Tight coupling of ETL components in the traditional ETL framework has leaded to difficulties in reusing, maintaining, and extending the ETL framework. This paper compares and discusses potential distributed technologies could be used to distribute the Extraction, Transformation and Loading components so as to achieve distribution and interoperability of these ETL components. A distributed system is a collection of independent entities that cooperate to solve a problem that cannot be individually solved [1-3].

ADDRESS FOR CORRESPONDENCE: Mohd Syazwan Abdullah, School of Computing, College of Arts and Sciences, Universiti Utara Malaysia, 06010 Sintok, Kedah, Malaysia

For computing systems, a distributed system has been characterized as a collection of computers that do not share common memory or a common physical clock, that communicate by a messages passing over a communication network, where each computer has its own memory and runs its own operating system. Typically the computers are loosely coupled and they cooperate to address a problem collectively [4].

The distributed software can also be termed as middleware. In other words, the middleware is the distributed software that drives the distributed system, while providing transparency of heterogeneity at the platform level [1]. Various primitives and calls to functions defined in various libraries of the middleware layer are embedded in the user program code. In addition, there exist several libraries to choose from to invoke primitives for the more common functions of the middleware layer [4]. Furthermore, there are several standards such as Object Management Group's (OMG), Common Object Request Broker Architecture (CORBA), Remote Method Invocation (RMI) and the Remote Procedure Call (RPC) mechanisms for distributed computing [1, 3]. Currently, deployed commercial versions of middleware often use CORBA, DCOM (distributed component object model), EJB and RMI technologies [5-7].

RPC is designed to be as similar to making local procedure calls as possible. The idea behind RPC is to make a function call to a procedure in another process and address space either on the same processor or across the network on another processor without having to deal with the concrete details of how this should be done besides making a procedure call [1-3, 5]. Before an RPC call can be made, both the client and the server both have to have stubs for the remote function that are usually generated by an interface definition language (IDL). When an RPC call is made by a client the arguments to the remote function are marshalled and sent across the network and the client waits until a response is sent by the server. There are some difficulties with marshalling certain arguments such as pointers [1-3, 5], since a memory address on a client is completely useless to the server so various strategies for passing pointers are usually implemented two rules: (1) disallowing pointer arguments and (2) copying what the pointer points at and sending that to the remote function.

CORBA usually consists of an Object Request Broker (ORB), a client and a server. An ORB is responsible for matching a requesting client to the server that performs the request, using an object reference to locate the target object. When the ORB examines the object reference and discovers that the target object is remote, it marshals the arguments and routes the invocation out over the network to the remote object's ORB. The remote ORB then invokes the method locally and sends the results back to the client via the network [6, 8]. There are many optional features that ORBs can implement besides merely sending and receiving remote method invocations including looking up objects by name, maintaining persistent objects, and supporting transaction processing. A primary feature of CORBA is its interoperability between various platforms and programming languages [6].

Distributed Component Object Model (DCOM) is the distributed version of Microsoft's COM technology which allows the creation and use of binary objects/components from languages other than the one they were originally written in, it currently supports Java (J++), C++, Visual Basic, JScript, and VBScript. DCOM works over the network by using proxy's and stubs. When the client instantiates a component whose registry entry suggests that it resides outside the process space, DCOM creates a wrapper for the component and hands the client a pointer to the wrapper. This wrapper, called a proxy, simply marshals methods calls and routes them across the network [6, 9]. On the other hand, DCOM creates another wrapper, called a stub, which unmarshals methods calls and routes them to an instance of the component [5, 10].

RMI is a technology that allows the sharing of Java objects between Java Virtual Machines (JVM) across a network. An RMI application consists of a server that creates remote objects that conform to a specified interface, which are available for method invocation to client applications that obtain a remote reference to the object [10]. RMI treats a remote object differently from a local object when the object is passed from one virtual machine to another. Rather than making a copy of the implementation object in the receiving virtual machine, RMI passes a remote stub for a remote object.

The stub acts as the local representative, or proxy, for the remote object and basically is, to the caller, the remote reference. The caller invokes a method on the local stub, which is responsible for carrying out the method call on the remote object. A stub for a remote object implements the same set of remote interfaces that the remote object implements. This allows a stub to be cast to any of the interfaces that the remote object implements [11]. However, this also means that only those methods defined in a remote interface are available to be called in the receiving virtual machine [10].

Service Oriented Architecture (SOA) is not a solution for a certain problem, but it is a framework architecture which adds some features to traditional software frameworks [11]. Typically, "SOA is a software architecture that starts with an interface definition and builds the entire application topology as a topology of interfaces, interface implementations, and interface calls" [11]. The strength of SOA is that it orchestrates all of the distributed components of a software by an orchestration point that centralizes the management of the distributed components. Section 2 compares the distributed technologies briefed in this section.

2. Comparison of Distributed Technologies

For the purpose of highlighting the best technology that is capable of overcoming the distribution and interoperability gaps of the current ETL framework, table 1 and 2 show the comparison of different distributed technologies discussed in sections 1 regarding components distribution and interoperability and the features that are dependent on the distribution and interoperability features.

Table 1. Distributed Technologies based on Components Distribution and Interoperability

	Components Distribution	Components Interoperability	Central Components Orchestration
RPC	Supported	Supported	Not supported
CORBA	Supported	Supported	Not supported
DCOM	Supported	Supported	Not supported
RMI	Supported	Supported	Not supported
SOA	Supported	Supported	Supported

Table 1 compares the distribution and interoperability features among ETL components. In addition, it compares the central orchestration among the ETL components. Distribution and interoperability are supported by all the five distributed technologies as shown in the table. On the other hand, the central orchestration among the distributed components is completely supported only in SOA, while the other four technologies do not support central orchestration, and the interoperability is done directly among the components. This causes difficulties to manage and orchestrate many components sufficiently.

Table 2 shows a comparison among distributed technologies based on components portability, extensibility, and legacy compatibility. Components reusability is supported without limitations by SOA, while it is supported by RPC if the used programming languages are compatible with each others. CORBA supports components reusability for all programming languages but with lots of configuration steps if the languages of the component are different, while DCOM supports reusability only within the same infrastructure. This infrastructure includes: programming languages, distribution technology, and configuration files. RMI supports reusability only for components that are built based on java programming language.

Components portability enables the distributed components to be executable in any programming

environment and then to be stand alone components that can be plugged as portlets to web portals [12]. The portability is supported only in SOA due to the web services availability in which every component can be encapsulated in a web service that is standalone and pluggable to any SOA based portal. All distributed technologies support extensibility [5], while only SOA supports components compatibility with legacy systems due to the web services involvement in SOA.

Table 2. Distributed Technologies based on Components Portability, Extensibility, and Legacy Compatibility

	Components Reusability	Components Portability	Components Extensibility	Compatibility with Legacy Systems
RPC	Limited support	Not supported	Supported	Not supported
CORBA	Limited support	Not supported	Supported	Not supported
DCOM	Limited support	Not supported	Supported	Not supported
RMI	Limited support	Not supported	Supported	Not supported
SOA	Unlimited support	Supported	Supported	Supported

Based on the discussion about distributed technologies in this section, it can be concluded that several benefits can be obtained when SOA is used in redefining the ETL framework and these are: distribution, interoperability, reusability, portability, and compatibility with legacy systems [11]. Furthermore, since one of the major roles of SOA is to enhance the features of computerized frameworks and models [11], and ETL framework is a computerized framework, therefore, SOA can play a central role in enhancing the ETL framework features regarding component distribution and interoperability.

3. ETL Framework with Interoperable Distributed Components

Based on the comparison conducted in section 2, SOA is used in our study to restructure the ETL framework. This section briefly discusses this enhanced ETL framework, which is shown in Fig. 1. In the data layer (bottom left of of Fig. 1), the data stores are exactly similar to the traditional framework. The business layer (top left of Fig. 1), however is built based on SOA, which includes four main parts and these are:

A. Service Orchestration Point (also called Directory Service or Service Registry): It describes the services available in its domain which are Extraction, Transformation, and Loading. Those three services are called Service Providers and register themselves in the Orchestration Point.

B. Service Providers: each of them is a component that performs a service in response to a consumer request. The framework has three Service Providers which are Extraction, Transformation, and Loading services.

C. Service Consumers: each of them is a component that consumes the result of a service supplied by a provider. The main Service Consumer in the framework is the client that represents ETL administrators. In addition, the three Service Providers can be Service Consumers to other services. For example, the Transformation service can request some functions to be done by the Extraction service in case that the Extraction and the Transformation are executed in one patch.

D. Service Interface: it defines the programmatic access of the three services, and establishes the identity of the service and the rules of the service invocation.

4. Discussions and Conclusions

Loosely-coupled components in the ETL framework are more flexible than those of tightly-coupled ETL framework. In the traditional framework, the ETL components are tightly-coupled to each other, sharing semantics, libraries, and often sharing state. This limits distribution, interoperability, reusability, portability, and compatibility of the ETL framework, which leaded to rebuilding a new ETL framework that achieves loose coupling among ETL framework components. The rebuilt ETL framework was developed based on SOA because the comparison among distributed technologies has shown the strength of using SOA as the best distributing technology.

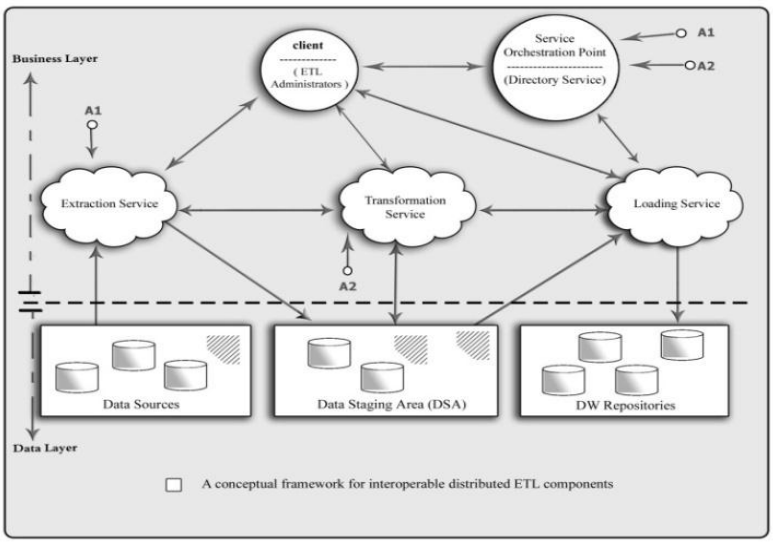

Fig. 1. A framework for interoperable distributed ETL components based on SOA

References

[1] A. D. Kshemkalyani and M. Singhal, *Distributed Computing Principles, Algorithms, and Systems*, 1st ed. Cambridge, UK: Cambridge University Press, 2008.
[2] D. Du and C. Raghavendra, *Distributed Network Systems*, 2nd ed. Califorina, USA: Springer, 2005.
[3] G. Coulouris, J. Dallimore, and T. Kindberg, *Distributed systems: concepts and design*, 3rd ed. Shanghai,China: China Machine Press, 2001.
[4] P. Wehrle, M. Miquel, and A. Tchounikine, "A Model for Distributing and Querying a Data Warehouse on a Computing Grid," *Proceedings of 2005 11th International Conference on Parallel and Distributed Systems,* 2005.
[5] A. S. Tanenbaum and M. Van Steen, *Distributed systems*, 2nd ed. New Jersey, USA: CiteSeer, 2002.
[6] V. Issarny, C. Kloukinas, A. Zarras, and M. Architectures, "Management Group's Common Object Request Broker (CORBA)," *Microsoft's Distributed Component Object Model,* 2008.
[7] G. S. Blair, G. Coulson, P. Robin, and M. Papathomas, "An architecture for next generation middleware," *Proceedings of the IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing,* pp. 191-206, 2009.
[8] P. Wehrle, M. Miquel, and A. Tchounikine, "A Grid Services-Oriented Architecture for Efficient Operation of Distributed Data Warehouses on Globus," *Proceedings of 21st International Conference on Advanced Networking and Applications,* 2007.
[9] G. R. Voth, C. Kindel, and J. Fujioka, "Distributed application development for three-tier architectures: Microsoft on Windows DNA," *IEEE Internet Computing,* vol. 2, pp. 41-45, 1998.
[10] F. Bertrand, R. Bramley, A. Sussman, D. E. Bernholdt, J. A. Kohl, J. W. Larson, and K. B. Damevski, "Data

redistribution and remote method invocation in parallel component architectures," *Proceedings of 19th IEEE International Parallel and Distributed Processing Symposium.,* 2005.

[11] M. Barai, Binildas, and V. Caselli, *Service Oriented Architecture with Java*, 1st ed. Birmingham, UK: Packt Publishing, 2008.

[12] T. Priebe and G. Pernul, "Towards integrative enterprise knowledge portals," *Proceedings of the twelfth international conference on Information and knowledge management.* pp. 216-223, 2003.