Journal of Computer Science 6 (3): 336-340, 2010
ISSN 1549-3636
© 2010 Science Publications

Object Based Middleware for Grid Computing

[1]S. Muruganantham, [2]P.K. Srivastha and [3]Khanaa
[1]Department of Information and Technology, Bharath University, Chennai, India
[2]Department of Electronics, Indian Institute of Technology, Chennai, India
[3]Department of Electronics, Bharath University, Chennai, India

Abstract: Problem statement: "Grid" computing has emerged as an important new field, distinguished from conventional distributed computing by its focus on large-scale resource sharing, innovative applications and, in some cases, high-performance orientation. The role of middleware is to ease the task of designing, programming and managing distributed applications by providing a simple, consistent and integrated distributed programming environment. Essentially, middleware is a distributed software layer, which abstracts over the complexity and heterogeneity of the underlying distributed environment with its multitude of network technologies, machine architectures, operating systems and programming languages. **Approach:** This study brought out the development of supportive middleware to manage resources and distributed workload across multiple administrative boundaries is of central importance to Grid computing. Active middleware services that perform look-up, scheduling and staging are being developed that allow users to identify and utilize appropriate resources that provide sustainable system and user-level qualities of service. **Results:** Different middleware platforms support different programming models. Perhaps the most popular model is object-based middleware in which applications are structured into objects that interact via location transparent method invocation. **Conclusion:** The Object Management Group's CORBA platform offer an Interface Definition Language (IDL) which is used to abstract over the fact that objects can be implemented in any suitable programming language, an object request broker which is responsible for transparently directing method invocations to the appropriate target object and a set of services such as naming, time, transactions, replication which further enhance the programming environment.

Key words: Middleware, distributed, services, performance, object based, transparent, integration, pervasive, collaborative visualization

INTRODUCTION

The middleware refers to a distributed platform of interfaces and services that reside 'between' the application and the operating system and aim to facilitate the development, deployment and management of distributed applications. The main function of this platform is to mask the inherent heterogeneity of distributed systems and provide a standard set of interfaces and services, which distributed applications, can assume present in any participating language, operating system or machine environment. Note that these object-based platforms are not the only referent of the term middleware in common use. For example, platforms geared toward database access in a heterogeneous environment are also often referred to as middleware.

The Open Grid Services Architecture (OGSA) has recently emerged as a 'second generation' distributed computing approach to Grid middleware that is taking Grid support forward from an era of ad-hoc platforms to a more architected approach built on service-orientation and web services technologies. This new approach promises a more unified and principled approach to the support of Grid applications. It augments generic web services standards by defining a specific abstract notion of 'Grid service' and also defines Grid-specific architectural elements such as, service factories and registries, naming and referencing conventions for service instances, support for stateful services, soft-state-based garbage collection of service instances, event notification from services and version management (Coulson *et al*., 2002; 2004; Foster *et al*., 2001; Parlavantzas *et al*., 2003). However, despite these advances, OGSA and indeed the web services technologies on which it is based, are still deficient in many areas of distributed computing support which, we believe, are key to the successful hosting of large-scale, next generation, Grid applications.

Corresponding Author: S. Muruganantham, Department of Information and Technology, Bharath University, Chennai, India

MATERIALS AND METHODS

We are particularly concerned with applications that exhibit properties such as high levels of heterogeneity in terms of both networking and end-systems, real-time interactive collaboration employing multiple media-types, large scale, complexity and dynamic configuration, QoS-sensitivity and adaptability to changes in environmental conditions. An illustrative example of such an application is a world-wide collaborative visualization session involving large numbers of scientists who join and leave the session dynamically and are connected by a variety of access networks and end-systems and involving multiple media such as visualization data, live sensor output, vector graphics and video. We contend that such applications fundamentally over-stretch the state-of-the-art in existing Grid support. More specifically, our analysis is that current platforms have three major areas of deficiency in terms of advanced application support:

Integration with advanced network services: One of the attractions of OGSA is its simple SOAP-based model of interaction. However, advanced applications often require more sophisticated communications services in terms of, for example, QoS management and, especially, different 'interaction types' RPC, reliable/ unreliable messaging, publish-subscribe, tuple-space-based interaction, peer to- peer based interaction, media-streaming, reliable group interaction, workflow interaction, distributed voting or auction protocols and various transactional styles

Architectural framework: OGSA focuses on interoperability through the use of 'ubiquitous' Web protocols and associated abstractions. However, this focus on interoperability needs to be complemented with a strong internal platform architecture that supports the integration of diverse system elements in terms of both breadth and depth

Complexity management: As Grid applications become increasingly large, complex and long-lived, there emerges a strong need for their sophisticated management. It is becoming recognized that the scale and complexity of such systems demand a self-managing or autonomic approach. Linking back to the previous two points, it is crucial that such self-management is applicable to the architecture of the whole system including communication services. We argue that this implies an open and programmable approach to system construction. In our current research we are addressing these deficiencies through a pervasive component-based approach that integrates middleware and networking functionality. Component technologies have already been adopted successfully in Grid research to promote structure and re-use at the application level. But here we propose the use of component technology not only for applications but also throughout the platform architecture in terms of both breadth and depth as outlined above. The aim of the research is to develop and apply a lightweight component model that imposes minimal overhead and can be used to build even low-level, system-oriented, functionality. The component model should also be system and language independent and API-neutral, so that it can be used to construct arbitrary application-level distributed programming environments as required. A particular goal is to apply lightweight component based technology to construct an extensible family of open and programmable overlay networks, thus providing an approach that is network-centric, offers a strong architecture for the system infrastructure and facilitates self-management through the inherent openness of component-based structures. This approach also promises other important benefits: i) a extensible range of interaction types, such as those listed above, can be made available and selected according to the application domain and ii) it facilitates dynamic reconfiguration of communications as context changes. To validate our approach and provide focus for practical experimentation we are using selected collaborative visualization-based applications and scenarios. Collaborative visualization is highly appropriate for this purpose because of its inherent properties as outlined above. It is also data and compute intensive which makes it an ideal case study for an infrastructure that aspires to manage both network and end-system resources in an integrated manner.

RESULTS AND DISCUSSION

Grid toolkit components: The aim of Grid toolkit is to provide support in each of four 'domains' that we identify as key in underlying the provision of Grid services. These domains are as follows:

- Service binding: This area provides sophisticated communication services beyond SOAP: i.e., support for QoS management and for different interaction types
- Resource discovery: This provides service and more generally, resource, discovery services, allowing for the use of multiple discovery technologies to maximize the flexibility available to applications. Examples of alternative

technologies are SLP or UPnP for more traditional service discovery, GRAM for CPU discovery in a Grid context and P2P protocols for more general resource discovery

- Resource management: This comprises both coarse grained distributed resource management as currently provided by services such as GRAM and fine-grained local resource management that is required to build end-to-end QoS
- Grid security: This supports secure communication between participating nodes orthogonally to the interaction types in use.

These four domains of middleware functionality are implemented in Grid toolkit as independent, horizontal, frameworks each of which is highly configurable and reconfigurable (Fig. 1). As such, they are directly available to application services and can also be combined to provide more complex middleware capabilities. For example, service bindings can integrate with Grid security to produce secure communication channels. In the remainder of this study, we examine in detail the service binding and resource discovery frameworks.

The ancestry of grid toolkit: The Grid Toolkit is an instantiation of the generic OpenORB middleware platform and hence follows the philosophy of building systems using components, component frameworks and reflection. The generic architecture of Grid Toolkit is also strongly influenced an Open ORB-based, web services-based, mobile computing framework called ReMMoC. ReMMoC provides inspiration and a code base for specific aspects of Grid Toolkit.

In particular, the four domains discussed above are each implemented in terms of Component Frameworks (CFs) that are configurable and dynamically reconfigurable by means of ‘plug-in’ components. The generic architecture of Grid Toolkit is also strongly influenced an OpenORB-based, web services-based provides inspiration and a code base for specific aspects of Grid Toolkit. In particular, it contributes a prototype service binding CF.

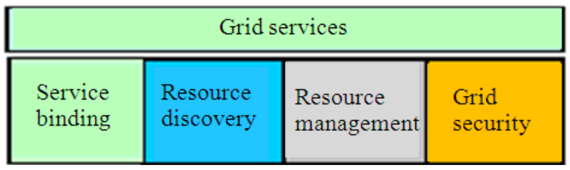

Fig. 1: The grid toolkit vision

The grid toolkit architecture: Grid toolkit is built in terms of OpenCOM derived CFs, the architecture of which is shown in Fig. 2.

The CFs behave as standard OpenCOM components; but, in addition, each implements the ICFMetaArchitecture interface which provides operations to inspect and dynamically reconfigure the CF’s internal structure. To ensure that dynamic changes to the framework are ‘valid’, each CF exports a receptacle named IAccept; from here different validation strategies can be plugged into the framework so that once a change is made, the plug-in checking strategy is executed and if invalid the framework rolls back to its previous state. By default, the local graph is checked against a set of XML-based architectural descriptions of valid component configurations. Alternatively, more or less complex strategies can be plugged in architectural style rules. Turning now to the wider picture, the subset of the Grid Toolkit architecture that deals with service binding and resource discovery is illustrated in Fig. 3.

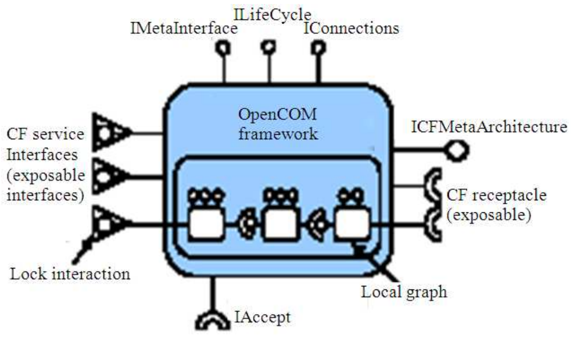

Fig. 2: The component framework model

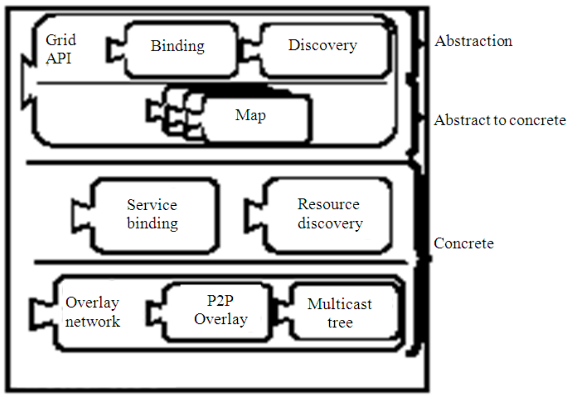

Fig. 3: The grid toolkit architecture

This depicts a three-layer architecture that is composed of: (i) abstract middleware, (ii) abstract to concrete mappings and (iii) concrete middleware. Each of these layers in turn consists of multiple CFs. This renders the architecture inherently configurable and extensible so that components implementing specific functions can be plugged in when and where required. In addition, applications requiring only minimal middleware functionality need only utilize parts of Grid Toolkit. This is especially important for execution on devices with limited resources.

The abstract middleware layer consists of a "Grid Service API" CF that is built in terms of web services abstractions. In particular, abstract service interactions are described in terms of WSDL so that services can be invoked irrespective of the interaction type underlying the service. This is achieved by exploiting WSDL's approach of breaking interactions down into individual messages: any conceivable service operation, from the user's perspective, can be described abstractly in terms of input or output messages. Discovery of both services and resources also forms part of the abstract middleware layer. Again, WSDL is used to abstract over different modes of interaction with service and resource discovery mechanisms. This is relatively straightforward for service discovery protocols because all of these tend to be based on advertisement of service types with service attributes. The abstract to concrete mapping layer then takes the abstract information submitted through the abstract middleware layer and maps it to the interfaces of the currently exposed concrete middleware implementation(s) in the layer below. This mapping is based on ReMMoC principles. Unlike ReMMoC, the Grid API framework allows multiple mapping components to be maintained in order for different services to be simultaneously hosted, each of which can use multiple service bindings. This was not necessary in ReMMoC as it was exclusively a client side framework that did not itself support remotely accessible services. Finally, the concrete middleware layer is composed of three CFs, organized in two layers. The top layer is composed of CFs to support concrete service binding and resource discovery. The service binding CF provides a set of available interaction type implementations. These are implemented as component personalities and plugged into the framework. Multiple personalities can operate in parallel to support the required level of persistence in hosting services. That is, when a service is hosted over one binding type it need not be shut down if an alternative binding must be used by another service. The binding framework exposes its network requirements to the underlying overlay framework using the exposed receptacle technique illustrated in Fig. 2. The discovery CF similarly allows multiple discovery technologies to be plugged into the framework at any one time. Resource discovery requests and advertisement of resources can be executed in parallel over each of the plugged-in personalities so that Grid applications can maximize the number of resources that are found, find them more quickly and can distribute their resources to a greater audience. The discovery framework utilizes the underlying overlay framework to enhance discovery and we intend to investigate the addition of alternative resource discovery technologies into the framework. Underpinning the Service Binding and Discovery CFs, the role of the Overlay CF is to provide overlay network services to the higher-level CFs: to route packets through virtual networks that are tailored to support the various service interaction types.

CONCLUSION

The existing Grid middleware does not provide the necessary level of support for complex Grid applications such as distributed collaborative visualization. We believe that an open component-based platform, which integrates middleware and networking functionality, is needed to support the sophisticated communication requirements of applications of this type. For this purpose, the service binding and resource discovery architectures of Grid Toolkit allow multiple interaction and discovery types to be simultaneously hosted over multiple overlay network configurations. Our ReMMoC-derived Grid Toolkit implementation initially provided us with a base of three binding protocols and two discovery technologies.

REFERENCES

Coulson, G., G.S. Blair, M. Clark and N. Parlavantzas, 2002. The design of a highly configurable and reconfigurable middleware platform. ACM Distribut. Comput. J., 15: 109-126.

Coulson, G., P. Grace, G.S. Blair, L. Mathy and D. Duce *et al*., 2004. Towards a component-based middleware framework for configurable and reconfigurable grid computing. Proceeding of the 13th IEEE International Workshop on Emerging Technologies Infrastructure for Collaborative Enterprises, June 14-16, IEEE Computer Society, Washington DC., USA., pp: 291-296. http://portal.acm.org/citation.cfm?id=1034650

Foster, I., C. Kesselman and S. Tuecke, 2001. The anatomy of the grid: Enabling virtual organizations. Int. J. High Perform. Comput. Appli., 15: 200-222. DOI: 10.1177/109434200101500302

Parlavantzas, N., G. Coulson and G.S. Blair, 2003. An extensible binding framework for component-based middleware. Proceeding of the 7th International Conference on Enterprise Distributed Object Computing, Sept. 16-19, IEEE Computer Society, Washington DC., USA., pp: 252. DOI: 10.1109/EDOC.2003.1233854