Computational Grid as an Appropriate Infrastructure for
Ultra Large Scale Software Intensive Systems

Babak Rezaei Rad
Department Of Computer Engineering
Science and Research Branch
Islamic Azad University
Tehran, Iran
rezaei.b@srbiau.ac.ir

Fereidoon Shams Aliee
Information Systems Architecture Research Center
Electrical & Computer Engineering Department
Shahid Beheshti University
Tehran, Iran
f_shams@sbu.ac.ir

Abstract—Ultra large scale (ULS) systems are future software intensive systems that have billions of lines of code, composed of heterogeneous, changing, inconsistent and independent elements that are dispersed through worldwide global networks. Huge scale of this heterology in addition to distinctive characteristics like decentralization of operation, management, deployment and evolution, continuous evolving, people participation in system operation and normal failures that dominate in ULS systems face them to new challenges that today software architectures and developers have less or no concerns about them. To face these challenges there must be an appropriate infrastructure with enough or at least extendable features and capabilities to support ULS distinctive characteristics. In this paper we propose computational grid as an appropriate infrastructure for ULS systems that not only supports some of ULS characteristics itself, but also can exploit service oriented architecture, fault tolerant approaches, distributed management algorithms and many other mechanisms in order to extend its fundamental capabilities to adapt itself with ULS systems. We explain what kind of grid is appropriate for ULS and expound how it provides solutions for each of the ULS particular characteristics.

Keywords- ULS, SoS, Infrastructure, Grid, OGSA.

I. INTRODUCTION

Ultra large scale (ULS) systems first introduced in a report produced by Carnegie Mellon University’s Software Engineering Institute in June 2006 [1], refer to future software systems that are likely to have billions of lines of code and operate at levels of complexity more similar to cities. The ULS study began by trying to answer a question posed by the US Army: “Given the issues with today’s software engineering, how can we build the systems of the future that are likely to have billions of lines of code?”[2].

A ULS system comprises a dynamic community of interdependent and competing organisms which compete for resources and may have conflicting goals. System enforces rules and policies that utilize resources effectively in a fashion that the system achieves its mission goals.

Future ULS systems are special kind of today's systems of systems (SoSs) that have particular characteristics in addition to SoSs characteristics. SoS is defined as a set or arrangement of systems that results when independent and useful systems are integrated into a larger system that delivers unique capabilities [3]. These systems are collections of independent systems that offer more functionality than simply the summation of the system parts. Development of such systems is evolutionary and behaviors of the system are not localized to any component. The principal purposes of the system are fulfilled by behaviors of components that are geographically distributed, operational and managerial independent and maintain their existence independent of the SoS.

In fact each ULS system is a SoS but not all SoS systems are ULS systems. The scale of a ULS system makes it more and more complex. Progressive evolution of human civilization and increasing of population and available resources confront future ULS systems with problems that are more complex and larger in scale than today's systems [4]. Future ultra large scale software intensive systems are ultra complex distributed systems in the new age of communication.

ULS systems inherit Characteristics of SoS systems such as operational and managerial independence of elements, evolutionary development, geographic distribution and emergent behaviors that are not localized to any element. Particular characteristics of ULS systems in addition to these characteristics are outcome of their scale.

A ULS system acts in complexity and scale of an ecosystem. The concept of an ecosystem connotes complexity, decentralized control, hard-to-predict effects of certain kinds of disruptions, difficulty of monitoring and assessment, and the risks in monocultures, as well as competition with niches, robustness, survivability, adaptability, stability, and health [1].
ULS systems are decentralized in operation and management, continuously evolve and have heterogeneous, inconsistent and changing elements. People not only use ULS systems but also participate as their elements. Software and hardware failures are the norm rather than the exception. These characteristics are beginning to emerge in today’s systems of systems; in ULS systems they will dominate [1].

A ULS system is like a city. A city is built on an infrastructure which provides fundamental and basic facilities such as power supply, telecommunication, transportation and traffic control and management network, police and emergency services and so on. One of the most important challenges of ULS systems is finding an appropriate infrastructure to design, implement and deploy such systems. Such an infrastructure must have inherent capabilities and features in its essence or at least ability to extend its fundamental capabilities to support ULS systems.

In this paper we propose computational grid as an appropriate infrastructure that has the potential to support future ULS software intensive systems. The mentioned potential comes from some fundamental features in addition to the ability to extend these features exploiting other technologies such as service oriented architecture.

The concept of computational grids is gaining popularity with the emergence of the internet as a medium for global communication and the wide spread availability of powerful computers and networks as low-cost commodity components [11]. Some grid projects are Globus [12], NetSolve [13], AppLses [14], Nimrod/G [15] and Unicore [16].

The rest of the paper is organized as follows. First we introduce the concept of computational grid and its architecture. In the next section we describe how the concept of virtual organization in grids can cover virtual systems of systems as the most similar type of today’s SoSs to the future SoS systems (i.e. ULS systems). Then we describe how computational grids can adapt themselves with distinctive characteristics of ULS systems. Service oriented architecture (SOA) helps to meet some requirements of ULS systems. In section 5 we illustrate how open group service architecture (OGSA) can help grids to adapt SOA with dynamic nature of ULS systems. Finally section 6 concludes the paper.

II. AN INTRODUCTION TO COMPUTATIONAL GRIDS

A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive and inexpensive access to high-end computational capabilities [5]. It provides an infrastructure for the application of several computers to a single problem at the same time. In such an infrastructure computing becomes pervasive and individual users (or client applications) gain access to computing resources (processors, storage, data, applications, and so on) as needed with little or no knowledge of where those resources are located or what the underlying technologies, hardware, operating system, and so on are [6].

Grid is a distributed environment and because it is an infrastructure for distributed computing it extends distributed applications capabilities with new properties such as various types of resources, applications, user groups and interactions. Grid does not need homogenous hardware, software, data and application resources. Resources and users often increase, decrease or change.

Grid is a layered infrastructure (figure 1) and each layer performs a function. The lowest layer is the network, which connects grid resources. Above the network layer lies the resource layer: actual grid resources, such as computers, storage systems, electronic data catalogues, sensors and telescopes that are connected to the network. The middleware layer provides the tools that enable the various elements (servers, storage, networks, etc.) to participate in the grid. The middleware layer can be thought of as the intelligence that brings the various elements together – the “brain” of the grid! The highest layer of the structure is the application layer, which includes applications in science, engineering, business, finance and more, as well as portals and development toolkits to support the applications. This is the layer that users of the grid will “see”. [7]

III. GRID AS AN APPROPRIATE INFRASTRUCTURE FOR VIRTUAL SYSTEMS OF SYSTEMS

A class of today’s systems of systems defined based on the amount of management control that is possible, called "virtual" system of systems is closest to the future ULS systems [1]. Virtual systems lack a central management authority. Indeed, they lack a centrally agreed upon purpose for the system of systems. Large scale behavior emerges, and
may be desirable, but the supersystem must rely upon relatively invisible mechanisms to maintain it [1].

Although ULS systems have more characteristics in addition to virtual SoS characteristics, there are many common characteristics between these two classes of software systems that arise because of operational and managerial independence of system elements.

Grid can be an appropriate infrastructure for such virtual SoSs. One of the key values of grids is in the underlying distributed computing infrastructure technologies that are evolving in support of cross-organizational application and resource sharing in a word, virtualization across technologies, platforms, and organizations [6]. This kind of virtualization is achievable through open standards which provide transparent access to grid resources in addition to technologies and standards in the areas of scheduling, security, accounting, management and so on.

Grid computing is defined in a spectrum of virtualization. Figure 2 illustrates this spectrum [6]. The lowest level of this spectrum is a single system portioning. Moving up this spectrum, homogenous and then heterogeneous resources can be virtualized. At the highest level there are heterogeneous organizations that are loosely coupled. Loosely coupling means that each component system is independent but when these independent systems enroll into a grid, they make a virtual SoS.

Grid computing is defined in a spectrum of virtualization. Figure 2 illustrates this spectrum [6]. The lowest level of this spectrum is a single system portioning. Moving up this spectrum, homogenous and then heterogeneous resources can be virtualized. At the highest level there are heterogeneous organizations that are loosely coupled. Loosely coupling means that each component system is independent but when these independent systems enroll into a grid, they make a virtual SoS.

Figure 2. Grid Virtualization Spectrum [6]

The spectrum of systems is covered by different grid topologies: intragrids, extragrids and intergrids.

An intragrid covers a single organization with a shared common security domain and shared data on a private network. An intragrid provides a relatively static set of computing resources and the ability to easily share data between grid systems. This type of grids is appropriate for enterprise vertical applications and internal jobs management [6].

By bringing together two or more intragrids, extragrids connect multiple organizations using remote/WAN connections. Resources in extragrids are more dynamic and the grid is more reactive to failures. External trusted partners can use extragrids to integrate and establish a relation of trust.

Finally an intergrid is a dynamic integration of applications, resources and services with patterns, customers and any authorized organization via internet/WAN [6]. This type of grid that includes dispersed management and multiple heterogeneous organizations can be an infrastructure for ULS systems. Figure 3 illustrates these three types of grids.

Figure 3. Intragrid, Extragrid and Intergrid [6]

IV. GRID ADAPTATION WITH ULS CHARACTERISTICS

ULS systems have some characteristics that distinguish them from other classes of software systems. In this section we illustrate how computational grid can adapt itself with these characteristics as a common infrastructure for ULS systems. These characteristics are as follows:

A. Operational and managerial independence of elements

Component systems of ULS systems are independently useful and they maintain their existence independent of the whole system [1].

Components in a grid system have the same characteristic. Each machine can work independently and when it wants to use grid resources or provide some services to other grid components, should enroll as a member of the grid and perform some sort of identification and authorization procedure to join the grid.

Donor software is responsible for enrolling and authentication into the grid. After using grid resources or service providing, the machine can logout and work independently. When a machine needs grid resources uses donors to enroll and the donors bubble up some measurement information about how busy the machine is to the management system of the grid. Thus machine can manage its resources by itself independently and determine resource assigned for internal and grid applications [6].

B. Evolutionary development

The ULS is not created fully formed but it comes to existence gradually [1].

To support evolutionary development in grids, service oriented grids are introduced. The open grid services
architectural (OGSA) is an SOA for the grid [9]. OGSA combines grid computing with web services using grid services. Grid services are grid version of web services. Using grids as the infrastructure for ULS systems enables them to add new grid services whenever needed. We explain more about OGSA in section 5.

Meanwhile like a ULS system, there is no need to create and build a grid as a whole immediately and machines can enroll into the grid gradually and can logout whenever they decide.

C. Geographic distribution

ULS components are distributed through global world community [1].

Grid is also a distributed environment that is distributed via internet/WAN. ULS system components can use an intergrid which uses internet to communicate the other components from everywhere in the world.

D. Decentralization

The scale of ULS systems means that they will necessarily be decentralized in data, and management [1].

Intergrids are also large ultra grids and their management is not completely possible through centralized approaches. To manage distributed data in ULS systems data grids are useful. Data grids focus on providing secure access to distributed, heterogeneous pools of data and can include resources such as federated databases. Within a federated database, a data grid makes a group of databases available that function as a single virtual database [6]. Data grids also harness data, storage, and network resources in distinct administrative domains, respect local and global policies governing how data can be used, schedule resources efficiently and provide high-speed and reliable access to data [6].

Usually grid management is distributed to match the topology of grid. The grid may be organized in a hierarchy consisting of clusters of clusters. These clusters divide grid to different levels. Higher levels management components (resource brokers) divide management activities to subactivities and send them to lower level components in the clusters. Lower level components take management subactivities and send their local management decisions to higher level components.

Although Hierarchical control is possible for today very large systems, it will be challenged at the scale of ULS systems and likely require different control models [1]. This is true for ultra large intergrids, thus many new control and management algorithms are introduced for intergrids that can be used for ULS systems if we use grids as their infrastructure. These new methods such as agent based management systems, [11, 17, 18], active network and policy based management, [10, 19] performance aware management [20] and so on can improve fundamental management approaches.

E. Heterogeneous, inconsistent and changing elements

ULS systems are made from components from variety of sources, under different hardware and software technologies and platforms. A key aspect in ULS systems is the integration of such dynamic components on the fly. These components may be different versions of the same software element, inconsistent in their design, implementation and usage [1]. Maintaining Integrity of ULS systems needs interoperability among a variety of changeable dynamic components while system is running.

Using grids the infrastructure for ULS systems makes it possible to use heterogeneous components. The grid itself uses middleware components and subsystems for integration. When the grid consists of heterogeneous machines, there may be multiple executable program files, each compiled for the different machine platforms on the grid. A nice feature provided by grid systems is to register these multiple versions of the program so that the grid system can automatically choose a correctly matching version to the grid machine that will run the program [6].

Another problem in grids is adaptation to the changing characteristics of grid environment. Performance and availability of grid resources vary over time. Resources may be overloaded, become unavailable and new resources may become available. Adaptation methods and techniques for grid environments [21, 22, 23, 24] can help to encounter problems with changing elements of ULS systems.

One of the most appropriate solutions to integrate different systems and applications as a whole is service oriented architecture. It is possible to combine SOA and grid computing using OGSA. OGSA can improve interoperability that is one of the most requirements in future ULS systems. We describe OGSA More in section 5.

F. Erosion of people/system boundary

People are not just users of a ULS system; they are also parts of its overall behavior. This means that system capabilities must help people to accomplish their current objectives and aware of new possibilities [1].

From a grid user perspective, user is someone that can use or maintain the grid and must have to enroll into the grid and install a software tool to use grid capabilities. Using this software a user can use grid resources as well as donating to them. Grids usually provide command-line as well as GUI tools enable users to see how their applications are processing and the ability to recover them using grid features or arrange to reserve a set of resources in advance for their exclusive or high-priority use. Grids have some tools to profile the grid's workload capacity sufficiently to have reliable statistics about the grid's ability to serve the reservation [6].

G. Normal failures

Because the physical underpinnings of a ULS system will be vast, hardware failures will no longer be an exception and because software components will be stressed beyond their designed-for capabilities, software may also behave in
undesirable ways [1]. Finally because of the scale in a ULS system, we can imagine that something will always be failing but like a city that crimes and accidents always happen and behaved with police, fire and emergency services and do not crash the whole city, there must be some capabilities in a ULS system to deal with failures.

If we choose grid as the infrastructure for ULS systems, it provides some degree of recovery for subjobs that fail. Jobs may fail due to programming errors, hardware or power failures, communications interruptions and excessive slowness. It is not always possible to automatically determine the failure reason, application problem or grid infrastructure problem thus grids categorize job failures in some way and automatically resubmit jobs to succeed. In some systems, the user is informed about any job failures and must issue a command to rerun the failed jobs [6]. Grids provide some API’s to monitor and recover jobs.

On the other hand fault tolerance is an important requirement in grid computing. As the dependability of individual grid resources may not be able to be guaranteed, thus many fault tolerant techniques are adapted for grid environment and applications [25, 26, 27, 28] that can help ULS systems.

V. ADVANTAGES OF OPEN GRID SERVICE ARCHITECTURE FOR ULS SYSTEMS

One of the panaceas for ULS systems introduced in [1] is service oriented architecture. A service oriented architecture is a framework for integrating business processes and supporting IT infrastructure as coarse-grained, discoverable, and self-contained components (services) that can be reused and combined to address changing business priorities [29]. SOA can provide a flexible architecture to integrate existing applications as a whole. It can help evolutionary development of large scale systems and provides a uniform context for cooperation of heterogeneous systems and applications. Web services are key enablers of SOA.

A web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described using a machine processable web service definition language (WSDL). Other systems interact with the web service in a manner prescribed by its description using simple object access protocol (SOAP) messages, typically conveyed using HTTP with an XML serialization in conjunction with other web related standards [29]. Universal description discovery and integration service (UDDI) is an XML-based distributed directory that enables web services to list themselves, as well as dynamically discover each other [30].

Discovery and invocation of persistent services using UDDI is not enough for dynamic ULS systems. SOA must be modified to support transient services to help ULS systems. The OGSA model adopts a factory design pattern to create transient grid services called "Grid Services" [6]. A grid service can be dynamically created and destroyed unlike a web service, which is often presumed available if its corresponding WSDL file is accessible to its client.

Meanwhile OGSA adds capabilities for discovery of more information about a service’s state, execution environment and some additional semantic details with lifetime management to web services that are crucial to the construct systems on the fly [30]. On the fly construction is an important requirement for the future ULS systems.

The open grid services architecture enables the integration of services and resources across distributed, heterogeneous, dynamic virtual organizations whether within a single enterprise or extending to external resource sharing and service provider relationships [9]. Figure 4 illustrates OGSA grid service. The service consists of data elements and various required and optional interfaces with potential instantiation via different implementations, possibly in different hosting environments.

Using OGSA to create service grids can help ULS systems to combine grid and service oriented architecture capabilities together.

VI. CONCLUSION AND FUTURE WORKS

Ultra large scale systems need an infrastructure that must be capable of supporting their particular characteristics. We proposed computational grids as the appropriate infrastructure for this type of software intensive systems. Computational grids are defined in a spectrum of virtualization. The highest level of this spectrum is intergrid that is used as virtual systems of systems infrastructure. Because virtual SoSs are the most similar systems to ULS systems with many similar characteristics, intergrid can be an appropriate infrastructure for ULS systems. To ensure appropriateness of this type of grids for ULS systems, we tried to find features and capabilities of intergrids that adapt them to particular characteristics of ULS systems. Table 1 illustrates these adaptations in brief.
Converging SOA and computational grids can help ULS systems to benefit from their emergent capabilities. Open grid service architecture provides these emergent benefits. Trying to improve semantic technologies such as ontology web language for services (OWL-S) and so on, combined with OGSA to enable semantic interoperability in computational grids as the infrastructure for ULS systems is necessary. The minimum result of such an improvement is the ability to discover and compose services on the fly. Semantic interoperability is inevitable for future ultra large scale software intensive systems and can help to dominate many challenges such as integration, adaptable structure and online modification.

**REFERENCES**


---

**TABLE 1. GRID ADAPTATION WITH ULS CHARACTERISTICS**

<table>
<thead>
<tr>
<th>ULS Characteristic</th>
<th>Grid Adaptation Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operational and managerial independence of elements</td>
<td>Independence of grid components and using of donors to enroll into grid.</td>
</tr>
<tr>
<td>Evolutionary development</td>
<td>Open grid service architecture and using of transient grid services.</td>
</tr>
<tr>
<td>Geographic distribution</td>
<td>Worldwide distribution of grid components through WAN/Internet.</td>
</tr>
<tr>
<td>Decentralization</td>
<td>Data grids, distributed management methods like hierarchical, policy based, agent based management and so on.</td>
</tr>
<tr>
<td>Heterogeneous, inconsistent and changing elements</td>
<td>Heterogeneity of grid components, middleware, self-adaptation methods and OGSA.</td>
</tr>
<tr>
<td>Erosion of people/system boundary</td>
<td>Command-line and GUI monitor and workload manager for grid users.</td>
</tr>
<tr>
<td>Normal failures</td>
<td>Recovery methods, replication and fault-tolerant grids.</td>
</tr>
</tbody>
</table>