Distributed Systems and Recent Innovations: Challenges and Benefits

Krishna Nadiminti, Marcos Dias de Assunção, and Rajkumar Buyya
Grid Computing and Distributed Systems Laboratory
Department of Computer Science and Software Engineering
The University of Melbourne, Australia
http://www.gridbus.org

1. Introduction

The World Wide Web is used by millions of people everyday for various purposes including email, reading news, downloading music, online shopping or simply accessing information about anything. Using a standard web browser, the user can access information stored on Web servers situated anywhere on the globe. This gives the illusion that all this information is situated locally on the user's computer. In reality, the Web represents a huge distributed system that appears as a single resource to the user available at the click of a button.

There are several definitions and view points on what distributed systems are. Coulouris defines a distributed system as "a system in which hardware or software components located at networked computers communicate and coordinate their actions only by message passing" [1]; and Tanenbaum defines it as "A collection of independent computers that appear to the users of the system as a single computer" [2]. Leslie Lamport – a famous researcher on timing, message ordering, and clock synchronization in distributed systems once said that "A distributed system is one on which I cannot get any work done because some machine I have never heard of has crashed" reflecting on the huge number of challenges faced by distributed system designers. Despite these challenges, the benefits of distributed systems and applications are many, making it worthwhile to pursue.

Various types of distributed systems and applications have been developed and are being used extensively in the real world. In this article, we present the main characteristics of distributed systems and look at some of the challenges that are faced by designers and implementers of such systems, and also introduce an example distributed system.

2. Main Features and Benefits of a Distributed System

A common misconception among people when discussing distributed systems is that it is just another name for a network of computers. However, this overlooks an important distinction. A distributed system is built on top of a network and tries to hide the existence of multiple autonomous computers. It appears as a single entity providing the user with whatever services are required. A network is a medium for interconnecting entities (such as computers and devices) enabling the exchange of messages based on well-known protocols between these entities, which are explicitly addressable (using an IP address, for example).

There are various types of distributed systems, such as Clusters [3], Grids [4], P2P (Peer-to-Peer) networks [5], distributed storage systems and so on. A cluster is a

dedicated group of interconnected computers that appears as a single super-computer, generally used in high performance scientific engineering and business applications. A grid is a type of distributed system that enables coordinated sharing and aggregation of distributed, autonomous, heterogeneous resources based on users' QoS (Quality of Service) requirements. Grids are commonly used to support applications emerging in the areas of e-Science and e-Business, which commonly involve geographically distributed communities of people who engage in collaborative activities to solve large scale problems and require sharing of various resources such as computers, data, applications and scientific instruments. P2P networks are decentralised distributed systems, which enable applications such as file-sharing, instant messaging, online multi-user gaming and content distribution over public networks. Distributed storage systems such as NFS (Network File System) provide users with a unified view of data stored on different file systems and computers which may be on the same or different networks.

The main features of a distributed system include [1] [2]:

- Functional Separation
  Based on the functionality/services provided, capability and purpose of each entity in the system.
- Inherent distribution
  Entities such as information, people, and systems are inherently distributed. For example, different information is created and maintained by different people. This information could be generated, stored, analysed and used by different systems or applications which may or may not be aware of the existence of the other entities in the system.
- Reliability
  Long term data preservation and backup (replication) at different locations.
- Scalability
  Addition of more resources to increase performance or availability.
- Economy
  Sharing of resources by many entities to help reduce the cost of ownership.

As a consequence of these features, the various entities in a distributed system can operate concurrently and possibly autonomously. Tasks are carried out independently and actions are co-ordinated at well-defined stages by exchanging messages. Also, entities are heterogenous, and failures are independent. Generally, there is no single process, or entity, that has the knowledge of the entire state of the system.

Various kinds of distributed systems operate today, each aimed at solving different kinds of problems. The challenges faced in building a distributed system vary depending on the requirements of the system. In general, however, most systems will need to handle the following issues [1] [2]:

- Heterogeneity
  Various entities in the system must be able to interoperate with one another, despite differences in hardware architectures, operating systems, communication protocols, programming languages, software interfaces, security models, and data formats.
- Transparency
  The entire system should appear as a single unit and the complexity and interactions between the components should be typically hidden from the end user.

- Fault tolerance and failure management
  Failure of one or more components should not bring down the entire system, and should be isolated.
- Scalability
  The system should work efficiently with increasing number of users and addition of a resource should enhance the performance of the system.
- Concurrency
  Shared access to resources should be made possible.
- Openness and Extensibility
  Interfaces should be cleanly separated and publicly available to enable easy extensions to existing components and add new components.
- Migration and load balancing
  Allow the movement of tasks within a system without affecting the operation of users or applications, and distribute load among available resources for improving performance.
- Security
  Access to resources should be secured to ensure only known users are able to perform allowed operations.

Several software companies and research institutions have developed distributed computing technologies that support some or all of the features described above.

3. Distributed Computing Technologies in Practice

Over the years, technologies such as CORBA and DCOM have provided the means to build distributed component-based systems. Such technologies allow systems to interoperate at the component level, by providing a software layer and protocols that offer the interoperability needed for components developed in different programming languages to exchange messages. However, such technologies present scalability issues when applied to, for instance, the Internet and some restrict the developer to a specific programming language. Hence, approaches based on Web protocols and XML (eXtensible Markup Language) have been proposed to allow interoperable distributed systems irrespective the programming language in which they are developed.

Web Services are based on XML and provide a means to develop distributed systems that follow a Service Oriented Architecture (SOA). Services are described in an XML-based dialect (WSDL). In a similar fashion, the request and reply messages exchanged in such systems are formatted according to the Simple Object Access Protocol (SOAP). SOAP messages can be encoded and transmitted by using Web protocols such as the Hypertext Transfer Protocol (HTTP). Various industrial technologies and application platforms such as .NET from Microsoft, J2EE from Sun, WehSphere from IBM are targeted at supporting the development of applications based on Web Services.

Along with Web Services, Grid computing is another emerging paradigm for creating wide-area distributed applications. Web Services are foundation technologies that can be used in building many types of distributed systems and applications including Grid systems. Web Services are in the core of the current implementations of Grid technologies such as Globus from Argonne National Laboratory in USA and the Gridbus from the University of Melbourne, Australia. Grid computing scales from an enterprise/organisation to a global level. Global Grids are established over the public

Internet infrastructure, and are characterized by a global presence, comprise of highly heterogeneous resources, present sophisticated security mechanisms, focus on single sign-on and are mostly batch-job oriented.

To enable global Grids, one requirement is that current enterprise and campus Grids are able to interoperate. Enterprise and campus Grids consist of resources spread across an enterprise and provide services to users within that organisation and are managed by a single administrative domain. Such Grids are more concerned with cycle stealing from unused desktops and use virtualization of resources in order to provide better means to manage and utilize them within an enterprise. For example, Oracle 10g uses a virtualization approach to split data storage from the database transaction and process layer. However, scalability and the design of security mechanisms are not as difficult as they are for global Grids.

4. Alchemi: An example distributed system

In a typical corporate or academic environment there are many resources which are generally under-utilised for long periods of time. A “resource” in this context means any entity that could be used to fulfil any user requirement; this includes compute power (CPU), data storage, applications, and services. An enterprise grid is a distributed system that dynamically aggregates and co-ordinates various resources within an organisation and improves their utilisation such that there is an overall increase in productivity for the users and processes. These benefits ultimately result in huge cost savings for the business, since they will not need to purchase expensive equipment for the purpose of running their high performance applications.

The desirable features of an enterprise grid system are:

- Enabling efficient and optimal resource usage.
- Sharing of inter-organisational resources.
- Secure authentication and authorization of users.
- Security of stored data and programs.
- Secure communication.
- Centralised / semi-centralised control.
- Auditing.
- Enforcement of Quality of Service (QoS) and Service Level Agreements (SLA).
- Interoperability of different grids (and hence: the basis on open-standards).
- Support for transactional processes.

Alchemi [6] is an Enterprise Grid computing framework developed by researchers at the GRIDS Lab, in the Computer Science and Software Engineering Department at the University of Melbourne, Australia. It allows the user to aggregate the computing power of networked machines into a virtual supercomputer and develop applications to run on the Grid with no additional investment and no discernible impact on users.
The main features offered by the Alchemi framework are:

- Virtualization of compute resources across the LAN / Internet.
- Ease of deployment and management.
- Object-oriented "Grid thread" programming model for grid application development.
- File-based "Grid job" model for grid-enabling legacy applications.

- Web services interface for interoperability with other grid middleware.
- Open-source .Net based, simple installation using Windows installers.

Alchemi Grids follow the master-slave architecture, with the additional capability of connecting multiple masters in a hierarchical or peer-to-peer fashion to provide scalability of the system. An Alchemi grid has three types of components namely the Manager, the Executor, and the User Application itself.

The Manager node is the master / controller whose main function is to service the user requests for workload distribution. It receives a user request, authenticates the user, and distributes the workload across the various Executors that are connected to it. The Executor node is the one which actually performs the computation. Alchemi uses role-based security to authenticate users and authorize execution. A simple grid is created by installing Executors on each machine that is to be part of the grid and linking them to a central Manager component.

More information about Alchemi can be found at http://www.alchemi.net/

Conclusion

As we have noted thus far, distributed systems have been an important part of peoples' lives as a result of innovations in the recent past in the area of Web-based applications, and will continue to make a serious impact in the future. Emerging technologies such as Grids will drive the next wave of innovation enabling the creation of applications that deliver IT as the 5th utility after water, electricity, gas, and the telephone. In conclusion, distributed computing is a very broad area with vast potential to improve efficiency of business processes and quality of life!

References

1. G. Couloris, J. Dollimore, and T. Kinberg, *Distributed Systems - Concepts and Design*, 4th Edition, Addison-Wesley, Pearson Education, UK, 2001.
2. A. Tanenbaum and M. Van Steen, *Distributed Systems: Principles and Paradigms*, Prentice Hall, Pearson Education, USA, 2002.
3. R. Buyya (editor), *High Performance Cluster Computing*, Prentice Hall, USA, 1999.
4. I. Foster and C. Kesselman (editors), *The Grid: Blueprint for a Future Computing Infrastructure*, Morgan Kaufmann Publishers, USA, 1999.
5. R. Subramanian and B. Goodman (editors), *Peer-to-Peer Computing: Evolution of a Disruptive Technology,* Idea Group Inc., Hershey, PA, USA, 2005.
6. A. Luther, R. Buyya, R. Ranjan, and S. Venugopal, Peer-to-Peer Grid Computing and a .NET-based Alchemi Framework, In *High Performance Computing: Paradigm and Infrastructure*, L Yang and M. Guo (eds), Wiley Press, New Jersey, USA, June 2005.