DYNAMIC CLUSTER MANAGEMENT AND RESOURCE UTILIZATION USING JINI TECHNOLOGY

Shiva Prasad Mahato

Department of Computer Engineering, Khwopa Engineering College

Abstract

With the commencement of Electronic Transaction Act, Nepal has taken further step in the field of information and communication technology. With government offices nowadays starting to use computers; there lies ahead many challenges to maximize the utilization of computing resources offered by each computer and minimize the overall cost. With many computers, so many idle resources are being wasted unnecessarily. Jobs can be distributed out to idle servers or even to idle desktops. Many of these resources remain idle during office hours off or even during office hours with many users utilizing the computing as well as memory resources. The proposed model not only utilizes resources to optimum but also makes the architecture more modular, adaptive and then provides dynamic fail over recovery and linear scalability. This approach is useful in a place which requires clusters to set up to perform resource intensive works like data processing or computing works. This model can be realized using JINI/Java Space technology which is open source technology and hence, can be cost effective as compared to other proprietary solutions. The motivating factor of this paper is to understand and identify the architectural constraint in the existing distributed application.

Keywords: *JINI, Java Space, Cluster, Space Based Architecture, Grid Computing, DCMRU*JT

1. Introduction

As the size of organization increases, with it increases the challenge of managing the diversity of existing resources. A little foresightedness may result saving huge cost for the organization. Necessarily, a question arises regarding management of computer clusters for carrying out computational task with more efficiency. This paper shall focus on the prototype cluster management and resource utilization leveraging JINI. The research intends to follow a new architectural design for computational Grid.

The research also tries to setup a cluster management system that performs resource sharing effectively. Traditional architectures normally focus on client-server or peer to peer interaction model but the current research focus shall be upon a completely new architecture "Space based Architecture" (Nati Shalom, 1999).

The space based idea has several advantages compared to its counterparts. Space based architecture is said to be more robust because one agent failing will not bring down the whole system as the case with client-server model. Replication and mirroring of persistent space permit communication regardless of network failure. Communication between peers is anonymous and asynchronous which makes computers in the cluster to work together to solve a problem collectively. These attributes of space based architecture enables us to make an adaptive cluster. The research will particularly focus on managing clusters in an adaptive manner, where the increase or decrease in the number of peers won't create any problem to the overall space. The followed approach will be based on one of the services of "JINI" (R. and S. Agrawal, 2002) the "Javaspace" (Batheja and Parashar, 2002, Hawick and James, 1998).

*Corresponding author: Shiva Prasad Mahato
Department of Computer Engineering, Khwopa Engineering College,Libali-8, Bhaktapur, Nepal
Email:shiv_mahato@yahoo.com
(Received: Nov 20, 2017 Accepted: May 5, 2018)

JINI technology is a service oriented architecture that defines a programming model which both exploits and extends the ability of java technology to enable the creation of distributed systems consisting of federations of well behaved networked services and clients. JINI technology can be used to build adaptive network systems that are scalable, evolvable and flexible as typically required in dynamic distributed systems. JINI enables computers to find each other and use each other's services on the network without prior information about each other or the protocols used. For the JINI to be selfhealing, lease is utilized. Every resource must be leased, that is, it must be periodically conformed that the registered resource is alive or that there is still interest in a resource. The JINI technology has following advantages

- JINI is a distributed computing network environment that offers, "Network plug and play"
- It also supplies a middleware layer to link services and clients from a variety of sources
- It is a set of APIs and network protocols that can help us build and deploy distributed systems that are organized as federations of services

The JavaSpaces technology is a highlevel tool for building distributed applications, and it can also be used as a coordination tool. It is written in the Java language and is a simple, fast and unified mechanism for dynamic communication, co-ordination and sharing of objects. A marked departure from classic distributed models that rely on message passing or RMI, the JavaSpaces model views a distributed application as a collection of processes that cooperate through the flow of objects into and out of one or more spaces. The dominant model of computation in distributed computing is the ClientServer model. This model is based on the assumption that local procedure calls are the same as remote procedure calls. JavaSpace overcome the problems of synchronization, latency and partial failure, inherent in distributed systems, by providing loosely coupled interactions between the components of distributed systems. Communication between processes on different physical machines is asynchronous and free from the main limitation of the traditional client/server model, when client/server communication requires simultaneous presence on network both parts client and server. Sender and receiver in javaspace don't need to be synchronized and can interact when network is available. In a distributed application, javaspace technology acts as a virtual space between providers and requesters of network resources or objects. This allows participants in a distributed solution to exchange tasks, requests, and information in the form of Java technology based objects (Rybicki,T. and Domaszewicz, J. , 2005). The javaspace transactional management and notify feature makes it easier to build a dynamic cluster management framework. In particular, it addresses the dynamic cluster problem where nodes can depart and join the cluster at any time.

2. Literature Review

Grid computing systems necessitate that services running in same or different memory spaces or potentially in different machine, are able to interconnect and communicate to provide the real time computation. For a basic communication between services, Java language supports sockets, which require the client and server to implement in applications-level protocols to encode and decode messages for exchange; unfortunately this kind of mechanism and the design of such protocols is awkward and error-prone.

Besides sockets, Microsoft had introduced a new technology called COM which helped to inter relate the distributed components, but COM also was not capable of fulfilling the real demand of pervasive computing because this technology basically relies on Microsoft services. There are also some other technologies such as CORBA, Salutation, and e-Speak, but these also could not answer the current demand of computation amongst service grids in the business and scientific communities. EJB, J2EE and also the .NET of Microsoft can be used in server-side grids but are not fully exploited for universal computing. As services grids come into widespread use, more research is taking place to establish a linkage for global grid services. Moreover, these computational grids are being used to solve large-scale problems in science and engineering; most of

which are focused on defining low-level services. Obviously, the future of scientific computing relies not only on powerful processing power, a huge data base, fast and advanced networks, but also on the approach of service oriented computing. Recently, Grid computing has started to impress Web service technology to define the standard interface for business purposes. This can be said an additional progress in Grid computing. But this is just at the starting phase of development, and there are a lot of works yet to be completed to address the current demands from e-business communities. Followings are some issues that are yet to be resolved to achieve the ultimate goal of Grid Computing.

i. Deployment Issue: Deploying the resource (services) in Grid should be done easily.
ii. Accessibility issue: Grid resources should not be complicated to access by the clients or end users. Consumers don't want to configure the software while utilizing it.

This paper aims to overcome these shortcomings by using the concept of service oriented approach. There is no doubt about the capabilities of River infrastructure for developing distributed applications. This is one of the smartest technologies in distributed computing. There are some critics that river infrastructures (lookup service, RMID etc) are hard to run. This paper sets to explore easy way for development of river services. Consequently, a new model of programming is designed (Bishu Gautam, 2010). This model is named as DCMRUJT model, and a demo-system was implemented based on this model.

Computing resources have passed the series of evolution which started from a tiny program executing in the single computer to the large virtual world so as to form the cloud computing resources as of today. This evolution of computing infrastructure implies that the increasing tendency of resource utilization often change the generally accepted notion of resource sharing and resource allocation. Cloud computing, a new prototype for solving complex and large-scale problems, is getting diverse attention from varied fields of science and technology recently. Though computational cloud like services widely known as Grid services are already being used to solve large-scale problems in science and engineering, most of them have focused on defining low-level services. These services have high potential for implementing the service level functionality which can be incorporated into a cloud computing framework in order to leverage the prevailing computing infrastructure in e-business communities. It is obvious that the future of scientific computing relies not only on powerful processing power, a huge database, fast and advanced networks, but also on the approach of architecture in order to support the entire infrastructure. Furthermore, the technological impact in society and the market economy will be another deciding factor for the sustainability of cloud computing technology. These new computing solutions are mostly service oriented. Service oriented solution adopted in cloud computing environments is not a new thing. Few years back, Grid computing started to architect the solution in service oriented architecture to impress Web service technology and define the standard interface for business purpose. However, the solution of grid computing still requires lots of effort to meet the demand of e-business in the market and the demand of the end users. Furthermore, the service level security in the cloud services is not carried out sufficiently. This certainly put high risk for the enterprises who do not want to sideline their clients with security concerns.

Software development history has started from 1940s with no architectural concept, if not, with very minimal architectural concept mainly targeted to solve the numerical problem of military project and the solutions implemented at that time were relatively simple compared to the current large-scale systems. Software architecture has got wide attention only after 1990 as the complexities of the software components tremendously increased within the time. This section will discuss the complexity of the software in order to deepen the discussion on the significance of architectural style not only for software architecture but also for the distributed application.

3. Methodology

In order to perceive the essence of architectural styles adopted in the distributed application and

understand its inherent problems, this paper analyzes and evaluates the prevalent styles from an architectural perspective. For this purpose, the research analyzes the widely used architectural style in the field of distributed applications and further deepens common understandings of internal architecture of those applications. Rather than advocating in any single style, the research utilizes the multiple styles in order to form a hybrid style so as to induce the most significant properties of the architecture used in those styles. This research advocates 'space based architecture' an architectural style for the understanding of distributed application that can be extended up to the architecture for enterprise services and demonstrates how this style can be used to enhance the architectural design of simple distributed applications. A detailed literature review about existing architectural style was done during the whole research period and then an application based on DCMRUJT infrastructure was conceived in order to demonstrate the space based architectural style for the enterprise services. Though a lot of software architects often ignore how the realization of their architecture will behave, this research emphasizes the importance of prototyping the instance as per the architectural style given by the architect.

The domain of this research lies on architectural design rather than source code structure though they might be related; the research will not discuss, except few major characteristics, the trivial structural design of the whole system in the paper. This research suggests that the architecture must utilize the sound technological environment that complies with the best practices, standard, smart and must be guided with the simplicity and are within an architectural principle. Furthermore, the style should provide the guidance for future implementation so that the contribution made by the research would be utilized in the future endeavors too. The motivating factor to undertake such initiatives is associated with service-oriented concept. On the basis of Service Oriented Architecture, an instance will be created which is named DCMRUJT, and this will demonstrate how a thorough research of architectural style can be implemented to develop space based application.

Further, a few DCMRUJT services will be developed emphasizing on the scalability of component and the reduction of dependencies among the services. DCMRUJT maintains the principle of encapsulating legacy system by providing the simple methodology of interfacing the underlying software components, and the way of enhancing them should be the well defined service. This well defined service is required to retain and be within development principles which were practiced during the development of DCMRUJT and emphasize loose coupling of the components thereby reducing the dependencies of the components participated in the foundation for the architecture. DCMRUJT does not only encapsulate the legacy service but also utilize the resources contributed by the previous architecture. The overall architectural style presented by DCMRUJT never tries to replace the prevalent architecture but tries to leverage and show the guideline for the next generation applications by utilizing hybrid architecture style.

Today's large diversity of network and computing resources with heterogeneous resources could lead to mismatch in network resources and volume of data transferred. Traditional distributed file systems like NFS, AFS, and Coda fail to address this issue. Hence this work tries to address this shortcoming by modeling data into persistent distributed objects (Kumar, et.al, 2000). The core to the solution approach would be the tuple space, servers and clients consuming exposed service. The approach will work to potentially maximize the consumption of resources to optimum in the service cluster leveraging a javaspace cluster.

3.1 DCMRUJT Framework

In this section, the application framework DCMRUJT and its design decision on the basis of architectural qualities presented in the previous part shall be portrayed. In order to choose the appropriate style, goals and requirement are set as a precondition. The research has followed a goal based approach so that it will assist the design decision. The research application should have following properties:

- The framework should allow seamless distribution of storage and computation
- It should provide real time cluster adaptability

- The framework should be highly fault tolerant
- The resources in the cluster should be highly available
- The master and the nodes communication should occur in a seamless manner
- Data should be replicated throughout the cluster
- The framework should provide linear scalability for overall system.

3.2. JavaSpace Architecture and Self Sufficiency

Further improvement to the architecture can be achieved by appending space based architectural node or element in the hybrid architecture style. Space based style supports architecture to be composed of shared memory space by providing self-sufficient node which increases the scalability of the system. Instead of further layering the system, the research provides self-sufficient node such that each client can interact with available node. In this architecture same program code will instantiate multiple times on the same machine or on multiple machines dynamically which will increase end-to-end performance of the system. System that is entirely constructed on layered based architecture often restrict knowledge of neighboring layer which decrease the visibility of deep rooted layer from developer perspective. The research does not prohibit layer based architecture fully in case it requires encapsulating of legacy services and communication with cross-organizational services. In such a case, the framework should implement a wrapper for the legacy service in order to utilize the legacy resource, however, by limiting layer and providing space based architecture, the framework does not require additional data layer as required in layered based traditional architecture. Further, these layers can be virtualized by grouping the application logic and business logic together into a single computational unit that is supported with space based architecture thereby virtualizing the underlying tires. Scaling can be achieved by running multiple instances of such units on multiple machines. Self sufficient instance of javaspace running in different nodes of the network form the backbone of DCMRUJT architecture.

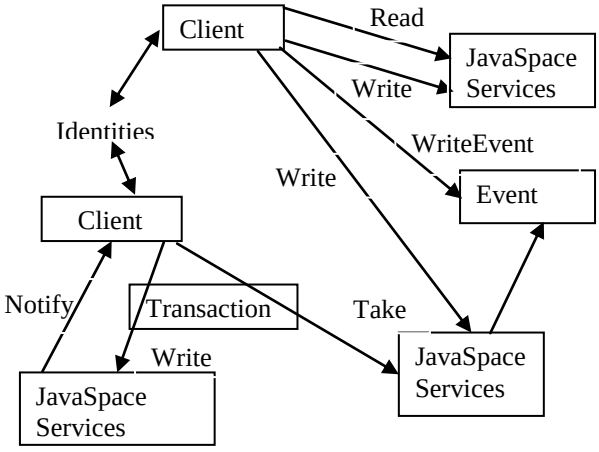

Fig 1 JavaSpace Architecture.

3.3. Distributed Data Structure in JavaSpace

Using JavaSpace it is also possible to organize objects in form of a tree structure. Since remote processes may access these structures concurrently, they are called distributed data structures. A channel in javaspace terminology is a distributed data structure that organizes messages in a queue. Several processes can write messages to the end of the channel, and several processes can read or take messages from the beginning of it (Charles, E.Hughes, et.al., 1999). A channel is made up of two pointer objects, the head and the tail, which contain the numbers of the first and the last entry in the channel .It is possible to use several such channels, giving all Actors associated with a space the possibility to handle messages in a FIFO fair manner. Channels may also be bounded, meaning that an upper limit can be set for how many messages a channel may contain.

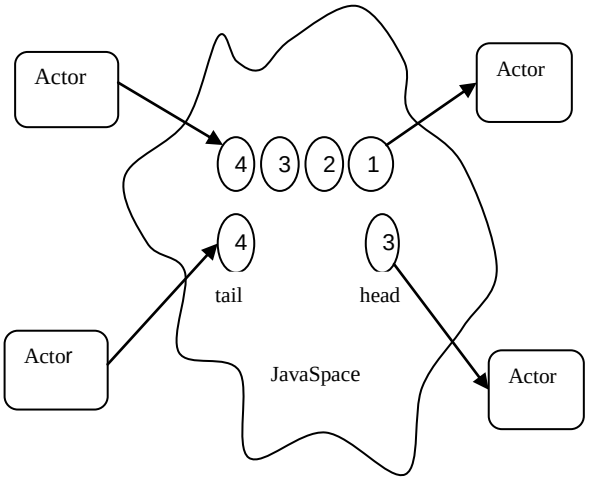

Fig 2 Channel

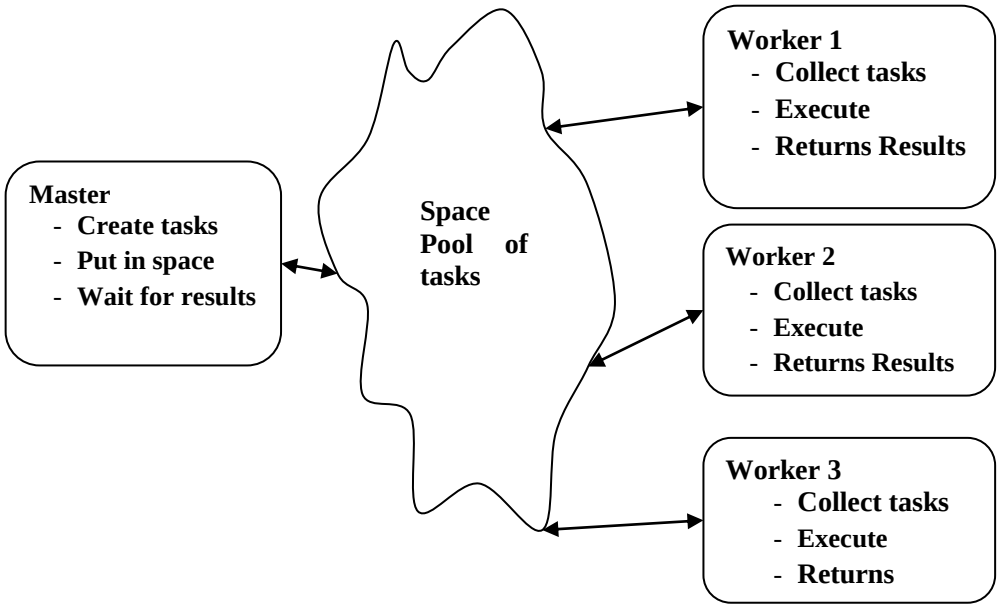

Fig 3 JavaSpace & Distributed Data Structure

3.4. Master Worker pattern

The Master-Worker Pattern (sometimes called Master-Slave pattern) is used for parallel processing and is the basis pattern to work with javaspace. It follows a simple approach that allows applications to perform simultaneous processing across multiple machines or processes via a Master and multiple Workers. The Master distributes out pieces of work to the javaspace", and these works are read, processed and written back to the javaspace by the end workers. In a typical javaspace environment there are several "spaces", several masters and several workers; these workers are normally designed to work in a generic way, i.e. they can take any unit of work from the javaspace and process it.

4. Experimented Results

As shown in this fig 4, experimental result of object querying latency with single javaspace instance, as latency decreases as the number of object increases in the space. This result has been taken from the experiment of DCMRUJT framework within the standalone computer.

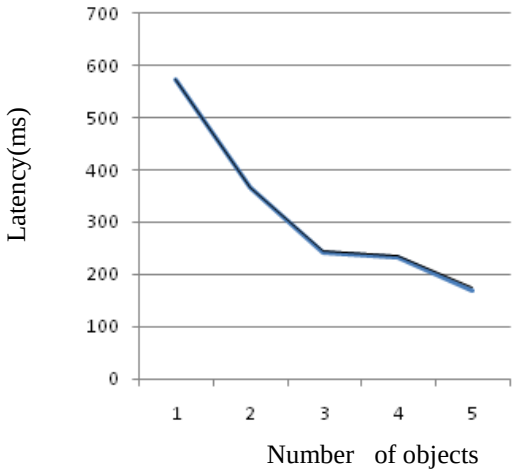

Fig 4 Object querying latency with single JavaSpace instance

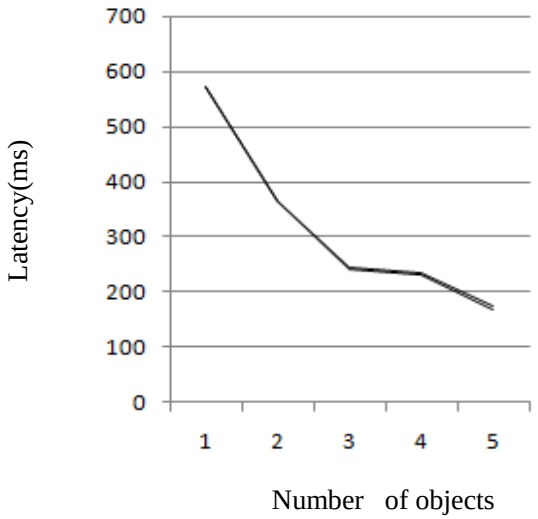

Fig 5 Object querying latency with multiple javaSpace Instance

As shown in above Fig 5, experimental results of object querying latency with multiple javaspace instance, as latency decreases as the number of object increases in the multiple instance of javaspace. This result also showed that multiple instance of javaspace decreases the latency. This result has been taken while experiment DCMRUJT framework in two computers.

Table 1 DCMRUJT framework

Qualities (DCMRUJT)	**Evaluation**
Performance	H
Portability	M
Reusability	H
Reliability	H
Self-Healing Ability	M
Scalability	H
Simplicity	H
Transparency or Visibility	H
Integrability	H
Latency	L
Varaibility	M
Sub-setability	H
Feasibility	H
Business Applicability	H

This Table 1 shows the concluded result of DCMRUJT framework whose architectural properties like performance shows that as the multiple instance of javaspace increases the performance of DCMRUJT framework increases.

5. Conclusion and Recommendation

In order to achieve the success and improvements of distributed application, an understanding of the architectural elements and the key principles of the architecture is vital. Without identifying the architectural constraints, application developed on ad-hoc basis often violates the architectural principles which may deviate from the targeted goal in long term.

This work of designing DCMRUJT application was motivated to propose a model which shows how an improved architectural style can be derived and how to apply that style to identify the new and broken features of the application. The DCMRUJT application framework is developed after analyzing the architectural constraints of distributed applications. Hence the overall results so far are quite convincing that DCMRUJT has achieved its objectives in developing a modular distributed framework.

Recommendation

Since the JCMRU model uses lookup service for communication between client and services making it a centralized model hampering the scale. It is one of the inherent limitations in the model that it has followed, which has to be mitigated to make the framework scale for bigger systems as well. One of the areas in this framework which still needs to be addressed is the security. Kerberos based security could be a possible integration to the framework in future. An application or tool is needed to be developed on top of this framework which could help manage the overall resources with Graphical User Interface.

References

[1] A classification and comparison framework for software architecture description languages (2000). IEEE Transactions on software engineering, VOL.26, NO.1.

[2] Ahmar Abbas (2003). Grid Computing: A Practical Guide To Technology And Applications. Charles River Media publisher.

[3] Arvind Kumar, et.al. (2002). Object oriented Framework for Adaption in a DFS. Java/JINI

Technologies and High-performance computing (vol : 4863):101-104.

[4] Bass et.al. , L. Clements & Kazman (2003). R.software architecture in Practice. Addison-wesley.

[5] Bishnu Prasad Gautam (2010). A model for the development of Universal Browser for proper utilization of computer resources available in service cloud over secure Network. , IMECS, Hongkong.

[6] Cameron laird (1999). Javaspace promises distributed computing breakthrough.

[7] Charles, E.Hughes, et.al. Space-based middleware for loosely coupled distributed systems. JAVA/JINI technologies and high performance pervasive computing (vol: 4863):70-78.

[8] Complexities of Software. Retrieved August 30, 2017 from http://www.dwheeler.com/sloc/redhat71-v1/redhat71sloc.html

[9] Gang Chen; Zhonghua Yang; Hao He; Kiah Mok Goh (2002); Coordinating multi-agents using JavaSpaces. Sch. of Electr. & Electron. Eng., Nanyang Technol. Univ., Singapore.

[10] Gupta, R.; Talwar, S.; Agrawal, D.P. (2002). Jini home networking: a step toward pervasive computing. Volume: 35, Issue: 8, 34 – 40

[11] Juan Joses Sanchez Penas (2006). From software architecture to formal verification of a distributed system. PHD Thesis, University of A Coruna.

[12] Jyothi batheja and Manish Parashar (2003). A framework for adaptive cluster computing using Javaspace. Volume 6, Issue 3, 201–213

[13] K.A. Hawick and H.A. James (2002). Dynamic Cluster configuration and Management using javaspace. Proceeding CLUSTER '01 Proceedings of the 3rd IEEE International Conference on Cluster Computing.

[14] Matt Bishop (2002). Computer Security: Art and Science. Addison Wesley Professional.

[15] Nati Shalon (2007). Space based architecture and the end of tier based computing. Gigaspaces Technologies.

[16] R.S. Sandhu (1993). Lattice-based access control models. IEEE Computer, 26(11):9-19

[17] Rybicki, T., Domaszewicz, J. (2005). MobileSpaces - JavaSpaces for Mobile Devices. Inst. of Telecommun. Warsaw Univ. of Technol.

[18] Sutedja, I.E.; Vun, N.; Hsu Wen Jing; Fasbender (2000).Applications of Jini technology in wireless mobile computing environment. IEEE, Conference: TENCON 2000. Proceedings, Volume: 2

.