International Journal of Computer Theory and Engineering, Vol. 2, No. 1 February, 2010
1793-8201

Grid Operating System: Making Dynamic Virtual Services in Organizations

Sanjeev Puri and. Dr. Qamas Abbas

Abstract—The structure of a grid system should be such that even a small personal computer can avail the facility of many supercomputers at a time. A grid is formulated by the number of supercomputers are used and participated in the computations. Grid computing has no straightforward way to control and administer grids dynamically. Grid operating systems bear the promise to become the new frontier in management of complex distributed computing systems and services that will offer for a single node: abstraction from hardware, and secure resource sharing with illusion dynamically by integrating grid capabilities into the kernel. It will integrate existing host operating system with a grid through an interoperating interface with expert dynamic OS on different versions of Grid virtual machine implementing grid nodes. Its goal is the creation of parallel processing pervasive grid computing platform that facilitates the rapid deployment and easy maintenance of grids of preferring peer to peer topology.

Index Terms—Grid operating systems, distributed computing, host operating system, Expert dynamic OS, Grid virtual machine.

I. Introduction

Grid is a type of parallel and distributed system [1] that enables the sharing, selection, and aggregation of geographically distributed "autonomous" resources dynamically at runtime depending on their availability, capability, performance, cost, and users' quality-of-service requirements.

Today Grid middleware is used to address the complexity of GRID environments and to help users in using GRID resources in an integrated way. This role in conventional computers is played by operating systems. Now it is time to develop a GRID operating system that may offer an integrated support for efficient management of local and remote resources available on a GRID environment to which a machine is connected. Without an operating system, Grids can fail the goal to enter mainstream computing and will not exploit all their functionality. As a conventional operating system provides an abstraction layer on top of the underlying physical resources of a computer, a GRID operating system must be designed to provide a virtual machine interface layered over the distributed, heterogeneous, autonomous, and dynamically available resources that compose a GRID. Resource sharing is the main objective of Grids and operating systems is the more appropriate environment for providing GRID users access to resource sharing facilities in a secure and transparent way.

A GRID OS should

- Provide simple connection to the GRID, Tolerating node failures and allowing application checkpoint
- Offer access to GRID resources, and Resource distribution transparency: Offering processes transparent access to all resources, and resource sharing between processes whatever the resource and process location.
- Define policies for providing local resource to a GRID.
- High performance; High availability.
- Scalability: Dynamic system reconfiguration, node addition and eviction, transparently to applications.

Grid operating systems support properties and provide functionalities that are usually addressed at middleware level to enable seamless integration and management of distributed resources while providing a uniform interface to applications and services. We believe that the Grid infrastructure must absolutely reduce the burden on the application developer investing on the open source operating systems and extending them towards Grid, simplifying the life of the high-level Grid services implementers because they could rely on the native services of the operating system kernel for tasks such as resource or process management.

A Grid is assumed to be made of an uncountable number of computers that are called Grid nodes (or simply nodes). Grid OS aims to be a first step towards the creation of a true open source operating system for Grid platforms supporting distributed resources, by embedding some important basic services or functionalities directly into the operating system kernel Grid OS aims at making VO management easy for administrators and work within VOs easy for users. The cost of administering and operating a VO (e.g., adding or removing nodes, changing access policy, authenticating and authorizing users) should be minimized to a bounded value rather than simply increase with the number of users and resources participating in the VO.

Deployment of Grids with existing Grid middleware [2, 3] involves the installation of multiple layers of software. Mathews et al [7] have highlighted similar issues. Multiple software layers in a Grid do not ensure fault tolerance. For example, with the popular cluster execution service Condor [8], a centralized cluster middleware can be liable to complete failure if a central server crashes. Active research is being pursued into more robust, flexible and fault tolerant Grid architectures, by converging Grid and Peer to Peer (P2P) topologies. However no Grid as of yet, has shown the advantages of such convergence.

It is clear that in order to facilitate the adoption of Grid computing to new domains and make it user-friendly for existing users' latent drawbacks in its architecture must be addressed. Our proposed Grid Operating system, aims at developing a pervasive Grid computing dynamic platform which addresses the drawbacks of the existing infrastructure, leading to a fault tolerant, flexible and easy to use stacks for rapid deployment of Grids. Grid OS aims to transparently Grid-enable all types of modern computations from traditional batch oriented to interactive, without requiring customization to applications in order to deploy them over a Grid. It is an operating system with built in support for Grid Computing. It is also an integrated Grid stack, that builds on and extends existing Grid technologies with expert dynamic OS interface with local host OS to enable rapid deployment of Grids, and enabling "plug and play" Grid computing on a fault tolerant resource discovery architecture.

II. Previous Work

Primary motivation for this work comes from the Anatomy of the Grid: Enabling Scalable Virtual services in Organizations, edited by Ian Foster et al [1], in which the authors discuss the challenges in the operating system interfaces for Grid architectures. The book discusses various principles but stops short of implementation details. While there has been little work on Operating System interfaces, there has been tremendous development in Grid middleware. Projects like Globus and Legion provide elaborate software infrastructure for writing Grid applications. These tools and libraries have to cope with the existing operating system services that are not designed for high-performance computing. As a result, they are forced to implement some commonly used high-performance optimizations like multiple TCP streams and TCP buffer size negotiation that more suitably should be implemented in the operating system's kernel. These tools, though quite different, often use the same set of low-level services like resource management, process management, and high-performance I/O.

Recently two major efforts in the direction of Grid operating systems (Grid OS) have been launched: Vigne and XtreemOS. The Vigne Grid Operating System [9] is a Grid OS which aims to relieve users and programmers from the burden of dealing with the highly distributed and volatile resources of computational Grids. Vigne focuses on three issues: i) Grid level single system; images to provide abstractions for users and programmers to hide physical distribution of Grid resources; ii) self-healing services to tolerate failure and reconfigurations in the Grid and; iii) self-organization to relieve administrators from manually configuring and maintaining Vigne OS's services. Vigne plugs onto the Kerrighed Cluster system [10] which supports cluster middleware level issues. However Kerrighed has some limitations which would limit wide scale deployment. Kerrighed does not tolerate node failure; clusters cannot be bigger than 32 nodes and provide no symmetric multiprocessing and 64 bit architecture support.

The aims of XtreemOS [11] are Linux based and open source to develop an OS level Grid solution with support for Grid enabling applications and providing self-healing services for large scale dynamic Grids. XtreemOS focuses additionally on small-scale mobile devices as well as supporting applications ranging from eScience, finance, and eCommerce to multimedia.

Apple xGrid [12] is a part of the Apple Mac OSX operating system, which enables an organization to create a Compute Grid or compute cluster. Apple xGrid is perhaps one of the first common-user oriented Grid computing systems. Jobs submitted by a user to an Apple xGrid system are divided into independent tasks by the 'Controller', a machine set up to coordinate the computations on the Grid. Furthermore, xGrid has not been deployed in environments with large numbers of machines in multiple domains which would give a true indication of its scalability. Apple xGrid is not self-organizing, which might be the single most important hurdle to its transition towards a universal Grid platform.

III. Virtual Services In Organizations

A virtual service can be seen as a temporary or permanent coalition of geographically dispersed entities (individuals, groups, organizational units or entire organizations) that pool resources, capabilities and information to achieve common objectives. There usually will be legal or contractual arrangements between the entities. The resources can be physical equipment such as computing or other facilities, or other capabilities such as knowledge, information or data. In an organization, information is stored and services and applications are executed by a set of computers in a Grid.

Key components of a virtual services are an administrator of the organizations, who is authorized to manage VO membership and policies, a set of participating users (called Grid users) in different participating domains, a set of roles which users/resources can play in the VO, a set of rules/policies on resource availability and access control, an (renewable) expiry time of the VO. The main responsibilities of node-level management include: translating from grid identities into local identities; granting or denying access to resources, checking limitations of resource usage (CPU wall time, disk quotas, memory, etc.); protecting and separating of resource usage by different users; logging and auditing of resource usage, etc.. A VO and its implementation by an operating system can reside in several stages of VO lifecycle: VO identification, VO formation, VO operation, VO evolution, and VO dissolution. In each stage a set of security threats to the overall system exists.

Grid OS, that is to say, the operating system is fully Grid-enabled. Once the Grid OS system has been installed on a machine, this machine is ready to participate in a VO with no need to install additional system software. Modifications to Linux to natively support VOs are done with a careful design to keep backward compatibility while providing build-in VO management interfaces [4, 5] that are as secure and simple to use as possible. System services and utilities such as login and shell programs, together with libraries, are extended in a modular approach so as to favor VO-level resource sharing requirements while keeping maximal transparency to users.

Access security in Grid OS will be policy driven. This means that for each resource (which includes VOs, applications, hosts, etc., in fact anything that requires protection) there will be a policy specifying who can access it and what they can do with it. In the case of a resource such as a file, the who could be a list of individuals and/or VOs, and the what could be read, write or execute actions similar to the conventional Linux file permissions (with a VO being considered as a sort of group). However, in a distributed and VO-based environment access will typically involve more than one entity, each with its own policies. The idea to monitor the operating system running on a PC is to execute the backdoor and the monitored OS in different virtual machines on top of a virtual machine monitor. The main issue to be tackled in the implementation is the extraction of OS state from the memory.

A. Application Management

As all layers will be integrated, the system will be able to offer information about the progress of the job, accurate monitoring of the used resources, error information, etc. In the current Grid world, given that the managers for the different layers are not integrated, a lot of information is lost in the way and the one that survives it is not correlated making it very difficult to use. For instance, in current Grid systems it is difficult to know why an application failed, when and with exactly what resources it run, etc. The integration of all services in a single OS will remove the lack of integration and offer users an execution environment with plenty of monitoring information and a powerful control of execution.

As the computational system are very large number in nature so it is planned in the present work to allocate the type of programming in a particular node hence when a user desires to avail the grid facility; the host local OS should handover the problem to the expert dynamic OS when software is loaded. The other types of a program which is complex in nature and requires the participation of many nodes. The host local OS computer evaluates the problem and transfers the modules to the participated computers. The third types of software used to such that it is divided in modules equal to the number of different grid OS and all the participating computers processing paralleled, then the responses of each computer are integrated in the host local OS node and which transfers the result to the originating PC interface with expert dynamic OS. However, when multiple users launch applications on the same cluster, it may happen that the workload exceeds the cluster capacity. To avoid this situation, a solution is to execute a batch system on top of the grid operating system. When an application is launched with the fork-delay capability enabled, its processes are queued if the cluster is overloaded. When a process terminates its execution, the global scheduler resumes the execution of the delayed processes, if any. At any time, if the cluster load is too high, the global scheduler may decide and use grid stacks that only suspend the execution of a very few or no processes.

B. Data Management

It should support extended meta-data, hierarchical names (the traditional directory structure), private, shared and collaboration data, and data archives. It should also support named Grid pipes, used by workflows where different processes produce data and some others consume it, the various processes being located on different nodes. Access rights should be managed in a manner such that file access could be granted to Grid users according to VO policies.

IV. Architecture Of Grid Os

Super peer paradigms have recently gained popularity as they enable Grids to integrate some of the advantages of peer to peer systems making a Grid infrastructure more robust, scalable and fault tolerant [14]. The toolkits require a common set of services from the underlying operating system. The key principle in Grid OS is to provide modularity. The modules provide a policy-free API which can be used to develop high level services like GridFTP. Grid OS provides a basic set of abstract dynamic services that are common to prevalent Grid software infrastructures with minimal Core Operating System Changes.

Architectural components of Grid OS are designed to be self configuring and plug-and-play in order to facilitate the rapid deployment of a Grid e.g. adding a node to a site involves a simple sign-in peer, adding a site to a remote region, involves registration process with a remote peer. The OS can be seen from two perspectives: First an integrated Grid Stack allowing rapid deployment of Grids, whilst making administration of Grids in an operating system which provides built in support for Grid computing. There is overlap between both kernel and use modes.

If an organization chooses to use the stack configuration, they can easily unload the kernel space modifications and use Grid computing from a user and middleware level. The lowest layer in Grid OS is the kernel layer and includes modules which facilitate Grid enabling of interactive application and fine resource management in Grids. Grid OS however makes use of process migration which transfers the execution context of processes to nodes where enhanced processing capabilities are available. Process migrations allow the transparent Grid enabling of existing applications without any need to modify them.

Support for dynamic virtualization [15, 16] is another central feature of the kernel layer with using expert dynamic OS. Grid OS aims to investigate hardware based virtualization in order to use a virtualization engine using fuzzy logic algorithm which enables the rapid creation and destruction of on-demand virtual machines. Both the QoS Management and kernel level process checkpointing modules allow users to regulate resource usage of applications and to autonomously migrate them to different nodes within a site.

The User interface layer will also contain services which extend existing cluster middleware like Condor, to be self-healing, self-configuring and fault tolerant. Other crucial components in this layer include the Resource controller for interactive applications as well as the fault tolerance module which wraps the kernel checkpointing functionality and makes it available to the user applications. The Security module builds on the kernel level virtualization engine and allows users to configure its behavior. The Middleware level includes components which allow Grid OS nodes to self organize into sub-Grids. Interoperability with the existing Grid infrastructure through standardized interfaces both in terms of resource discovery and authentication and authorization will also be provided.

V. Modules Description

High-performance network I/O module are accessed and analyzed data in peta bytes is the feature of grid OS. These networks have high bandwidth and large round trip time. . Gridos io and Gridos ftp are kernel modules that handle both network and file system I/O, thus double copying can be avoided.

The delay can be measured by calculating the round trip time (RTT) using ping or trace route. The TCP slow start and Congestion avoidance algorithms determine the size of the congestion window. The kernel buffers are allocated depending on the maximum size of the congestion window.

Different communication methods differ in usage of network interfaces, low-level protocols and data encodings and may have different quality of service requirements. The communication module also provided multi-threaded communication which is used in implementing the FTP module. Grid middleware have to locate and allocate resources according to application requirements. They also have to manage other activities like authentication and process creation that are required to prepare a resource to use. Gridos rm provides higher-level issues like co-allocation, online-control etc. A global PID (GPID) for every process in Grid OS and provides communication primitives which can be used on top of Gridos comm for processes to communicate among themselves [6].

The additional modules are gridos ftp server, gridos ftp client are based on client server grid architecture behavior. There are two thread pools. The first pool of threads is I/O or cache-miss thread pool. These threads populate the buffers asynchronously at the request of listener threads and gridos ftp common includes parsing and handling of FTP request and its responses.

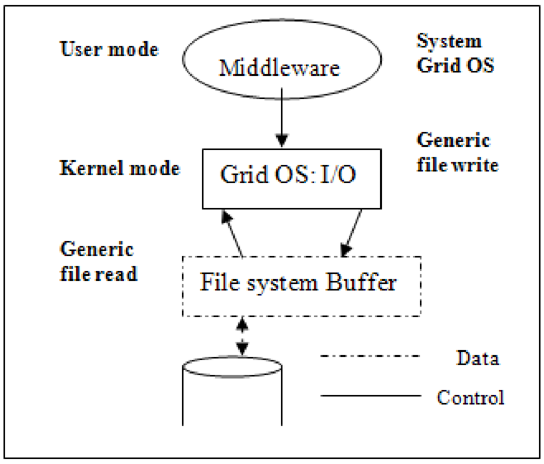

Fig.1 Reading and Writing to a file

VI. Implementation

The globus IO module implementation is divided into two APIs, one each for the network and the file system. The network API includes functions to read and write data from a Gridos managed socket.

- Gridos io async read: used to read data from a Gridos managed socket in non-blocking mode
- Gridos io write: This function writes data to the Gridos managed socket
- Gridos io buffer setopt: This function sets options for buffer management. The options include setting of TCP send and receive buffer sizes, maximum TCP buffer size etc.
- Gridos io buffer getopt: This function returns the current buffer management options.

Program 1: Illustrate through a program of file write function listing

```
int Gridos_io_file_write(const char *buf, const char
*dest, int size)
{ struct file *f = NULL;
int flags, len;
mm_segment_t  old;
int mode = 0600;
flags = O_WRONGLY;
if(!dest) {
printk(KERN_ERR "Destination file name is NULL\n");
return -1;
}
f = file_open(dest, flags, mode);
if (!f || !f->f_op || !f->f_op->write) {
printk(KERN_ERR "File (write) object is NULL \n");
return -1;
}
f->f_pos = 0;
old= get_fs();
set_fs(KERNEL_DS);
len = f->f_op->write(f, buf, size, &f->f_pos);
set_fs(old);
if (f->f_op && f->f_op->flush) {
lock_kernel();
f->f_op->flush(f);
unlock_kernel();
}
fput(f);
printk(KERN_INFO "Wrote %d bytes\n", len);
return len;
}
```

In library wrapper, there are three ways of controlling Gridos behavior from user interface layer :

1. Through system call sys Gridos
2. Using ioctl on Gridos device
3. Using /proc interface

A. Dynamic Interactive Applications of Grid

The Grid OS mechanism of Grid enabling dynamic interactive applications are multi threaded in nature. The kernel of Grid OS will support thread migration, an extension of the concept of process migration [19], which is popular in cluster middleware. The Grid OS Kernel will be capable of migrating a single thread to another system, which will be selected by the resource controller as the best available site for execution. Grid OS also provides for checkpointing of remotely migrated processes in order to save execution states and restarts them in the case of an event. The created checkpoints will also be exported to the parent node of the process at regular intervals. However, the frequency of exports will be less than the frequency of checkpointing on the local machine, in order to contain the network cost incurred when transferring process checkpoints from local host OS interfacing with expert dynamic OS. The following formula has been used to calculate future checkpoint intervals and is itself a function of previous intervals.

$$\boxed{\text{Int} = W*\text{Int}-1 + (1-W)*\text{Int}-2;}$$

Where the value of W is dictated by site level policy. Each new checkpointing interval is a function of previous checkpointing intervals along with a constant, W, which determines the importance a subGrid needs to give to the most recent checkpoint interval.

B. Resource Management and Security

Modern Cluster middleware provides “all or nothing” resource control: a node is either completely available for processing, or it is not. To allow for fine control over users’ resources, Grid OS provides dynamic virtualization to control resource usage and to provide security from foreign computation, by creating virtual machines which use the amount of resources the user is willing to provide to external users interact with dynamic expert OS and makes use of a QoS management module for local computations. Our resource control algorithm is a dynamically distributed peer-to-peer (P2P) network compute and data aware algorithm and considers both network connectivity and computational capability in scheduling decisions. It can be applied both at the local site and large-scale levels.

C. Resource Discovery Service Structure

We introduce a super peer architecture [6] in which lowest tier is a machine level granularity sub-Grid, which consists of machines that have good network connectivity between them, analogous to a traditional cluster. Each sub-Grid is represented by a super-peer, which is the most available machine within the vicinity of the sub-Grid. The regions are represented by a region peer. A virtual organization (VO) in this system can be at any level: it can consist of individual machines or be an aggregation of entire sub-Grids or of entire regions. Interactive applications will be handled at a machine-level VO, whereas large-scale Grid applications will require aggregations of entire sub-Grids. At the top-most tier the granularity is in terms of sub-Grids, and these are grouped into regions depending on geographical proximity of the super peers where dynamic expert OS in upper layer interface with lower layer host OS’ so it is as easy with this grid pipelining service structure for resource discovery.

The main features of adaptive grid OS will be works as dynamic resource discovery service i.e.

1) To improve the network usage, by allowing a resource request to propagate to peers in close proximity, thus limiting the overall network traffic, and improving response latency.
2) To improve the quality of results, by propagating the request until a suitable resource has been found, while limiting the network traffic as much as possible.
3) To provide a scalable and efficient framework dynamically grouping nodes into sub-Grids, and clustering sub-Grids into regions.
4) Once the requesting machine has a list of the machines within the sub-Grid, the resource controller determines the suitability of the discovered nodes to execute the user application.
5) The job request is forwarded to them and then the resource controlling and scheduling process takes place within the new sub-Grid. If the region cannot satisfy the resource requirements it then contacts other regions in a Peer to Peer manner.
6) A self-healing behavior is crucial in widely distributed architectures such as a Grid environment. To make sub-Grids self-healing, a distributed leader election algorithm [13] is deployed to elect a new super peer in a sub-Grid.

VII. Conclusions And Future Work

Grid operating system which provides extensive, flexible services for Grid architectures and it also has planned to port Globus libraries to Grid OS thus providing a complete software infrastructure for Grid architectures. Grid OS is not only aimed at adapting dynamic grid computing for frequently related to the set up and administration of Grids but also it is based on dynamic virtualization engine for Grid OS to provide security and resource management to resource owners and privacy to resource users. The creation of Grid applications and the lack of general fault tolerance within the Grid infrastructure are also issues of concern. Grid OS is a step towards a “Plug and Play” pervasive Grid dynamic computing environment. It is designed to support all types of modern computations, including batch and interactive and dynamic support the creation of Grids of any architecture. The main contribution of this paper is that it presents a dynamic structure for the development of adaptive Grid OS to extend the discovery service to enable self-healing and self organizing behavior. Furthermore we propose that the system should embed the capability for interoperability with existing and emerging Grid infrastructures interface with expert dynamic OS interact with lower host layer local OS by making the system compliant to evolving standards in Grid computing.

REFERENCES

[1] A. Iamnitchi and D. Talia, "P2P Computing and Interaction with Grids", *Future Generation Computer Systems*, North-Holland, vol. 21, no.3, pp. 331-332, 2005.

[2] I. Foster, C. Kesselman**.**, **“**Globus: A Metacomputing Infrastructure Toolkit”, *Intl J. Supercomputer Applications*, 11(2):115-128, 1997

[3] E. Laure et al., Middleware for the Next Generation grid Infrastructure, Proceedings of the Computing in High Energy Physics Conference, pages 826, 2004.

[4] I Foster and C. Kesselman, editors. *The Grid: Blueprint for a new computing infrastructure*. Morgan Kaufmann, 1999.

[5] I. Foster, C. Kesselman, and S Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *International Journal of High Performance Computing Applications*, 15(3):200–222, 2001.

[6] OpenMOSIX,http://openmosix.sourceforge.net

[7] B. Mathews, “Towards a Knowledge Grid: Requirements for a Grid OS to support Next Generation Grids”, Core Grid Workshop on NGN, Belgium, 2005

[8] Litzkow, M. Livny, & M. Mutka, Condor – A Hunter of Idle Workstations, Proceedings of the 8th Int. Conference of Distributed Computing Systems, June 1988, pages 104-111.

[9] Jeanvoine, E., “Using Overlay Networks to Build Operating System Services for Large Scale Grids”, The Fifth International Symposium on Parallel and Distributed Computing, July 2006 Page(s):191–198

[10] Vallee, G., “A new approach to configurable dynamic scheduling in clusters based on single system image technologies”, International Parallel and Distributed Processing Symposium, 2003. 22-26 April 2003 Page(s):8

[11] C. Morin, “XtreemOS: A Grid Operating Sytem making your Computer ready for Participating in Virtual Organizations”, 10th IEEE International Symposium on Object/component/service-oriented Real-time distributed Computing (ISORC 2007)-To appear

[12] ApplexGrid http://www.apple.com/acg/xgrid

[13] Stoller, S.D., Leader election in asynchronous distributed systems, IEEE Transactions on Computers, Volume 49, Issue 3, March 2000 Page(s):283 - 284

[14] A. Iamnitchi, I. Foster, J. Weglarz, J. Nabrzyski, J. Schopf, M. Stroinski, in: Grid Resource Management (ed.), A Peer-to-Peer Approach to Resource Location in Grid Environments, Kluwer Publishing, 2003.

[15] Barham, P. et al. “Xen and the art of virtualization”. In Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles (Bolton Landing, NY, USA, October 19 - 22, 2003) SOSP '03. ACM Press, New York, NY, 164-177

[16] KernelVirtualMachine,http://kvm.qumranet.com/kvmwik

A. Author : Sanjeev Puri , Member of IACSIT, MSc. (CS) & MCA from MD University, Rohtak, India-passed in 2001-02, MPhil (CS) from VMU, Salem, PhD(CS) perusing currently from Integral university, Lucknow, U.P., India. The author’s major field of study is grid computing. He has the 8+ yrs academic experience worked as Ass. Professor as well as industrial experience as SE of C/C++ platform in Goyal computers, Lucknow Now He is ASS. PROFESSOR, SITM, Deptt. of Computer Science & Engg., UP Technical University, Lucknow, U.P., India. His articles have published in IJCIT: A Comprehensive Framework for Value Service Innovation Using Knowledge Reengineering, IJEE: An Adaptive Service Oriented Architecture for Value-Added Mobility Services, IJGDC: Real-time Open Decisive Network Infrastructure for Smart Grid. Mr. Sanjeev Puri is associate organization member of AIMA, New Delhi, Best faculty award at SRMCEM, UPTU, Lucknow, India in 2003. Best Course Coordinator by NIEC in 2006(e-mail:purispuri_2005@rediffmail.com)

B. Author : Dr. Qamar Abbas , PhD (CS) from University of U.P., India. The author’s major fields of study are distributed computing and software engineering. He has the 22+ yrs academic experience worked as Professor as well as 5 yrs industrial experience. Now He is DIRECTOR, AITM, UP Technical University, Lucknow, U.P., India from 3+ yrs. He has also the Research Advisor of Integral University, Lucknow. His so many articles have published in international journals (22) and conferences (08) as well as books of Java. Dr. Qamar Abbas is member of computer societies in India and RDC member of other renowned universities.

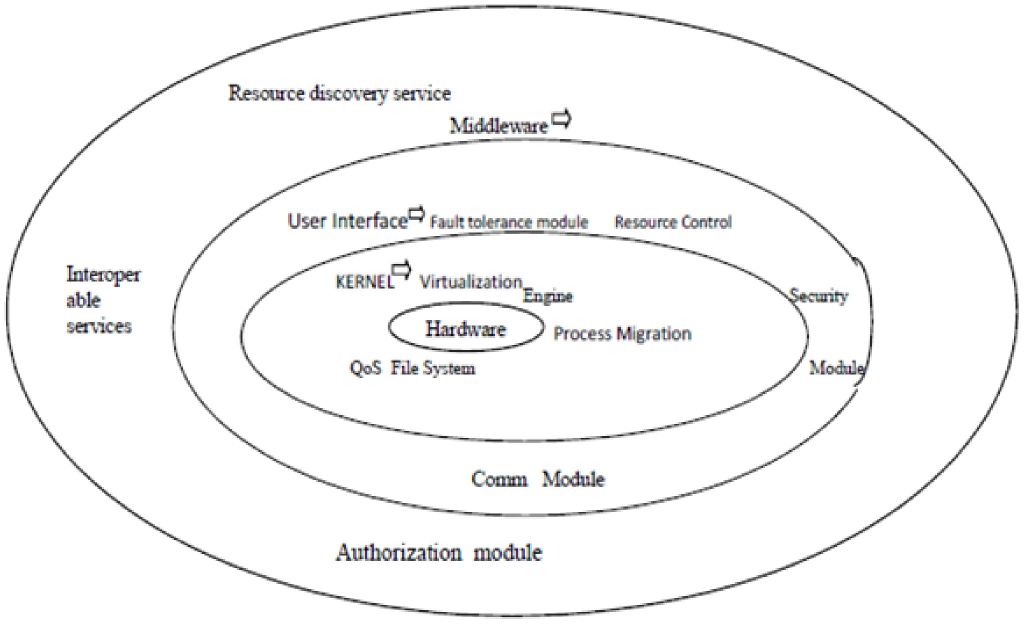

Fig.2 Grid OS Architecture

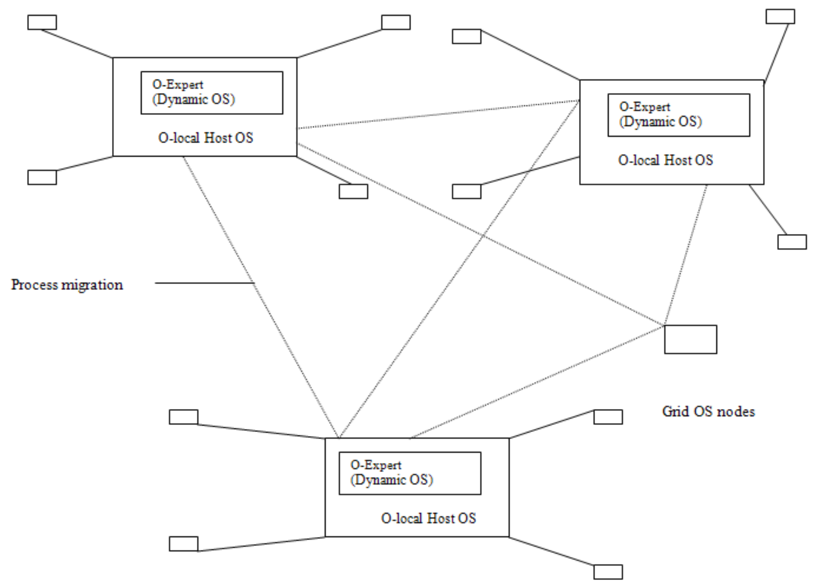

Fig.3 Dynamic Virtualization of Grid OS