

Grid Computing : A Virtual Organization

V.K. Saxena*

Abstract

It is a great challenge for the scientific community to provide information online with low prices in this era of competition. In today's world computer users are globally located. The grid is essentially a heterogeneous collection of computational and storage resources, thereby leading to many related challenges. These challenges require dealing with diversity in the terms of local resources, dynamic nature of the local resources, creation and management of services and maintaining the Quality of Service (QoS). Since grid is inherently a parallel and distributed system, the key issues regarding design of the grid, data locality and availability, implementation, scalability, anatomy, privacy, maintenance, fault tolerance, security, etc come into picture and need to be addressed. These issues demand new technical approaches for the grid environment.

Keywords: Computational Grid, Virtual Organization (VO), Scheduling, Fault Tolerance Security, Economics.

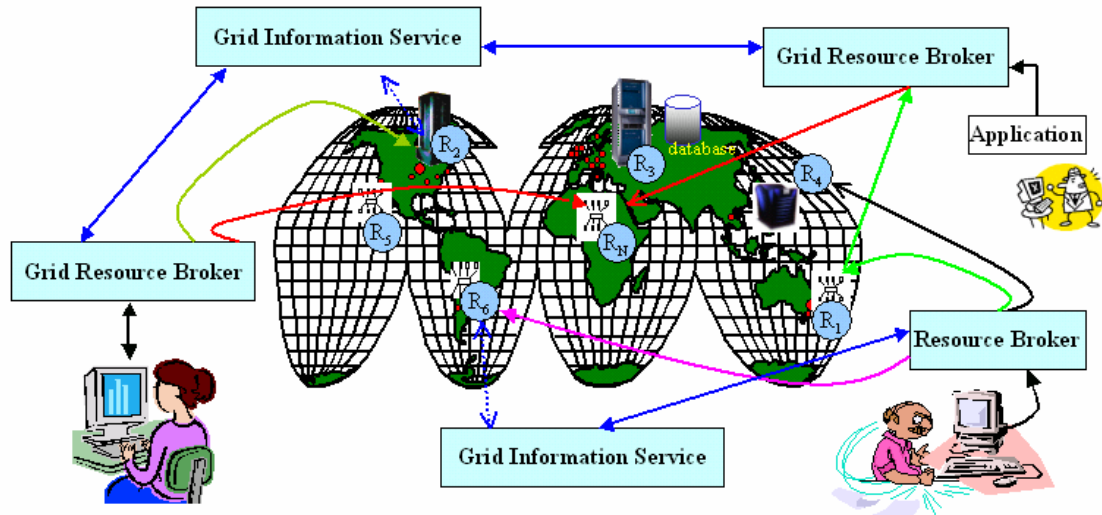
1. Introduction: Grids are finding applications in various fields in the form of Data grids, Computational grids, Science grids, Bio grids etc. To define what is a grid and who do not qualify as a grid, it can be established that a grid is a system that

- (a) has coordinated resource sharing that are not subjected to centralized control
- (b) uses standard, open, general purpose protocols and interfaces and
- (c) delivers non-trivial qualities of service.

Going by this definition cluster computing, web services etc do not fall into the category of grids. This is because clusters are owned by a single organization whereas web only provides a standard infrastructure for data exchange between two distributed applications and does not account for the aggregation of resources unlike the grid. As well web is a client - server computing system that is not intended to solve large scale problems in a distributed manner. One more question of paramount importance is what are the problems best suited to be solved over the grid? For the purpose, nature of the application, its domain and its suitability over the grid needs to be explored i.e., whether the application demands high throughput or distributed supercomputing or data intensive or collaborative engineering etc. Since a grid is a distributed system, the programming models and tools also need rethinking so as to develop suitable algorithms and software architecture for mapping over the complex grid architecture. As the grid involves heterogeneous resources, a method regarding the resource management should also be in place with efficient security measures taken to safeguard the applications due to the interaction not only between two entities but many entities, involved in collective operations.

*System Engineer, Institute of Computer Science, Vikram University, Ujjain (M.P.) India

2. Typical grid computing environment



3. Issues in computational grid scheduling : This section covers and analyzes the available computational grid models, their features and what more is expected of them.

3.1 Computational grid models: Since scheduling on a grid is a NP hard problem, a number of models addressing one or the other issues related to scheduling have been proposed in literature. There are many approaches that concentrate only on the scheduling aspects while the others may focus on aspects like reliability, security and/or fault tolerance. Thus there are a number of models each addressing one issue or the other.

In the centralized super scheduler architecture, a job super scheduler is designed to schedule the jobs for the individual nodes. In that model a few issues remain unaddressed. Since the super scheduler does not have the control over the resources of the distributed computing centers it depends on the individual local batch-queuing systems to initiate and manage job execution. But this structure proves to be a bottleneck due to centralized queuing and dispatch. What should be the super scheduling algorithm is not specified. How the interaction between super scheduler and local scheduler will take place is an important issue but has not been addressed. Even a policy of job selection and migration and the destination choice for the transferred jobs is not there.

In the distributed super scheduler architecture some super scheduling policies are proposed but the notable point is that the allocation criterion is not biased keeping in mind the nature of the job. It is not specified how the approximate execution time for the job is checked at various nodes / clusters. The model only checks for the approximate waiting time for the job awaiting execution in the local queue rather than estimating the execution time at that node in addition to the waiting

time. How much is the resource requirement is not specified. Further the super scheduler scalability and fault tolerance issues need to be addressed.

The TITAN architecture basically deals with the local scheduling at the node level and is operative with Globus Toolkit only, making its use restrictive. It accounts for the submission of the whole job to a group of nodes and then finding its suitability for that job. If the request is not met at the current broker level it is passed on to the next broker up in the hierarchy level. Thus it does not account for the simultaneous assessment of the feasibility. Further, resources are accessed irrespective of the fact that they meet the job specialization or not making the search space wider. Using PACE they calculate the task's execution time but how the mapping of the task to nodes is done is not specified.

Further it must be noted that the grid scheduling algorithms schedules the job over the grid resources but what is the scenario for the local scheduling at the node level remains unaddressed. Whether the scheduler has any control over the scheduling policies of the local nodes are equally important and should also be considered. One factor that influences the local scheduling policies is the priority assigned to the job since then the preemptive or non preemptive nature of the scheduling policies locally will effect the turnaround time of the job.

Some of the current grid schedulers Nimrod-G, GRaDS, Condor-G, Legion etc assume one entry point into the grid as the new job which has entered will make itself known to the resource selector which will schedule it on a processor. This single entry point serves as a bottleneck. They have a limitation in the fact that even the information gathering system is centralized here. Due to this, scalability of the system also gets restricted. The description of resources is also coarse grained in many available schedulers and they mostly check for the availability of the system or the workstation and does not provide any information about the attributes of the workstation like the number of processors and their speed, OS type, slot available for each process, available memory for execution etc. While distributing the job over the nodes they even do not consider the nature of the job, which is a critical attribute for any job to be executed on a heterogeneous environment like the grid. Further, they all assume to have control over the scheduling policy of the individual nodes, which is in fact not always possible.

3.2 Role of the end system: End system plays a major part in the grid system as today's end systems are relatively small and they are connected to networks by interfaces and with operating system mechanisms that are originally designed for reading and writing slow disks. Thus these end systems needs to be developed supporting high performance networking grid architecture.

3.3 Job pre processing requirements: Once the grid comes into existence there may be a number of virtual organizations (VO) forming the grid. Whenever a query enters the grid it enters through the corresponding virtual organization only. Therefore resource management also needs to be addressed as it is difficult

to have common grid architecture since they are created to cater to different needs but at least a basic set of services need to be identified for ex. Querying, Submitting and Monitoring. Any process to succeed on a grid might proceed by

- *Obtaining the necessary authentication credentials (connectivity layer protocols):* The user should be authenticated for entering the grid. This can be done by the use of password or certificates. For this, a password may be given to the authorized users keeping in mind the security of the user keys. Security measures like PKINIT and Kerberos, used for wide area networks may be useful for the purpose. The credential processing service may handle the processing and validating authentication tokens. In addition to the above, since various organizations and resources are involved in the problem solving, the grid should require only a single sign in facility for the user.
- *Querying information (collective services):* Since a grid is a dynamic system it should be updated to keep track of the changes occurring inside it. A replica catalog could be used to determine availability of the resources like computers, storage systems, networks and the location of required input files. This information should be dynamic and should be updated from time to time, whenever a job completes its execution or any resource participates or quits the grid. Mechanism of discovering resources on the grid is itself a challenging task for the designer. In the old schemes some centralized services, for e.g. Condor matchmaker were used that contained all the information. But it had the problem of scalability and single point failure. Later decentralized schemes were used e.g., MDS – 2 where the grid information is stored and indexed by index servers that communicate via registered protocols. The best approach is to have the information available to all the virtual organizations about the resource status so that accordingly load balancing can be done. Thus we require that each virtual organization should have the latest updated resource information for efficient grid utilization.
- *Submitting requests (resource protocols):* Keeping in mind the heterogeneity of the grid the request should be submitted to appropriate computers, storage systems, and networks to initiate computations, move data and so on and so forth. Here it is worth mentioning that high selectivity in the resources has an advantage of the best possible allocation but may lead to lower probability match. As well high regionalism may lead to non uniform distribution of the tasks but will reduce the communication cost. Selectivity depends on the nature of the job whereas regionalism is governed by how much communication overhead and network delays we can tolerate. Both these factors depend on the existing load structure on the grid. Thus it should be decided that what level of selectivity and regionalism is required in the grid? After submitting the request the authorization service may evaluate the service requirements of the task by gathering information about the requestor and the target. In addition access control should also be exercised as the users are working in the shared environment since there can be a number of resources over which the owner wants to have its complete authority.
- *Monitoring resources and computations (resource protocols):* The progress of the various computations and data transfers needs to be monitored necessarily. This could be done by using means such as heartbeat or check pointing, notifying the user when all are completed, and detecting and responding to failure conditions. This brings into

picture, security and fault tolerance of the grid. The grid should be able to safeguard the information about a participant's data from the other. Since large numbers of virtual organizations are part of the grid, it means large numbers of resources are to be monitored and tracked for various failures that may take place. These failures may range from submission failure to hardware/software failures. The grid should be able to meet these failures gracefully in a reactive or proactive manner. Apart from the basic monitoring services offered by the grid one more factor that needs proper attention is the QoS (Quality of service) offered by the grid. The common specifications in this regard could be whether events are delivered reliably or unreliably and whether they are in order or out of order, network delays involved robustness of the system etc.

3.4 Allocation requirements: Since the grid is a distributed environment certain points need to be noted regarding the allocation aspects. Some of these could be

- *Services:* The first important thing is that how many services the Grid has been designed to cater to i.e. is the grid designed to address a single issue or it involves many like minimum turn around time as well as real time with fault tolerance?
- *Scalability:* Till what extent the grid is scalable?
- *Topology:* What are the entry point's restrictions for the job i.e. does the grids have the ability for multipoint entry of the job? In other words whether the job services are centralized or distributed?
- *Nature of the job:* Once the job is submitted it should be sent to the suitable resources and the result should be sent back to the calling resource. To harness the advantages of a grid, proper load balancing should be taken into account. It is worth mentioning here that whether the load balancing and distribution of the task over the grid takes into account the nature of the job? Or in other words how to find the best possible resource for the task of a given nature?
- *Effect of existing load:* Most previous task allocation algorithms have assumed assignment of modules of only a single task to nodes. Therefore the execution time of a particular module on a particular processor cannot be taken as the cumulative time taken by it.
- *Number of modules allocated:* Should there a limit to the modules assigned to a particular processor? Here we need to keep in mind that now memory is no longer a constraint because of the integration technology becoming better and better.
- *Load balancing:* If the tasks are spread over the grid to exploit the parallelism then what should be the load balancing strategy? Is it the same for every task? Or otherwise how a strategy is adopted when the nature of the task changes.
- *Parallelism:* Even the parallelism of the job is considered to be at the fine grain level or the coarse grain level should be noted carefully. Whether this parallelism is assumed to be attained after the job submission or is inherent in the job with already preprocessed form, as mentioned earlier, with the job submitted in the ready to execute parallel form?
- *Interactive task handling:* The grid should be able to handle many tasks which are interactive in nature. Here a module can be allocated by knowing the outcome of the

task till one stage. The grid should be able enough to support these kinds of jobs efficiently. One possible solution to this problem is to treat the task in the same way as a modular task, have a Job Precedence and Dependence Graph (JPDG) with the user interaction dependent data being treated as dependence between two stages or modules.

- *Job migration policy:* Is the grid able to support real time tasks and what should be the preemptive strategy if they are entertained? What should be the migration policy in case a job needs to be moved over to another node because of the preemptive action or on account of the node failure?
- *Channel load:* The task allocation models do not consider the load on the channel, whereas in a practical grid like situation there is a possibility of traffic on the channel resulting in further delay on the data traveling.
- *Checkpoints location:* In case of failure check where should the checkpoints be placed and how should the checkpoint activity be performed? How the checkpoint activity effects the system performance should be carefully considered.
- *Redundant resource selection:* What should be the degree of redundancy in the form of task replication or resource replication? In case of failure what should be the criterion of node selection having task replica? How the system performance is effected by it needs to be studied properly.
- *Restricted access:* Almost all the grid scheduling models assume that any job can be allocated to the suitable resource. Normally it is not the case in real life scenario as a local organization may not wish to give access to all of its resources to the third party directly as it may wish to exercise some control over its resources for security and protection reasons or for any restricted access. For the purpose, allocation can be done via a local centralized system resulting in two tier architecture of the grid. The first tier is the super-scheduler, which allocates the job to a virtual organization or an organization inside the virtual organization, which then schedules the job internally.

3.5 Real time systems: For real time jobs the condition becomes much more complex since now the job has to be seen from the point of view that whether it is possible to schedule it or not. Thus for these type of jobs, the requirements are that the jobs should have predictable end time. For the case of composite jobs, a complex set of sub jobs must be orchestrated in such a way as to respect any dependence between sub jobs. The real time processing also demands co scheduling i.e. scheduling of multiple resources for the same precise time. Proper brokering to select best resource is equally important. What are the Service Level Agreement (SLA) requirements needs to be taken care of and renegotiated as compute and other resources may fail unpredictably or the sub jobs may fail due to user error or high priority jobs may be submitted. Thus SLA should also be constantly added, altered or withdrawn and hence scheduling would need to be continual dynamic and uncertain process. Finally, strategies should be decided for the jobs that do not meet the deadline i.e., whether they should be deferred or accepted with whatever best the grid could offer to it.

4. Reliability and fault tolerance in computational grid: Apart from all the issues relating to maintaining the QoS and secured communication we always wish to have a

system in place that is reliable and able to digest system failures. Significantly incorrect performance of the computers may lead to several devastating effects. A fault tolerant system is one, which continues to perform even in the presence of hardware and software faults. A fault is a physical defect, imperfection, or flaw that occurs within some hardware or software component whereas an error is the manifestation of a fault and is any deviation from accuracy or incorrectness. Specifically, faults are the cause of errors and errors are the cause of failures.

Whenever a task enters the grid for execution the failure chances may spread from the application failure at the point of submission to the resource failure to the node failure. Faults can be the result of many things viz. specification mistake (incorrect algorithms, architectures etc.) hardware failures (hot crash, network partition etc.), software failure (numerical exception, failed application etc.), implementation mistakes (inefficient algorithm), component defects, external disturbances (radiation, electromagnetic waves, interference etc.), performance failures (application not completing within a specified time etc.) or some other failures (machine rebooted by the owner, excessive CPU load, decreased priority by the local resource for the current task etc.) At the level of grids depending on the type of grid it may be prone to either or all of the faults.

- *Proactive approach*: One approach towards handling the faults is to avoid their occurrence. Fault avoidance can be used in the case of grids to prevent the occurrence of faults and includes design reviews, testing or other ensuring measures. This is actually a proactive approach which mostly deals with the analysis, design and testing to check the possible failures so that they do not occur.
- *Proactive approach using agents*: Even when the system is thoroughly tested for the above mentioned parameters there is always a possibility of failure. The proactive approach using multiple agents can be used here by continuously monitoring the system by checking some important parameters for their values to be within the permissible limit. The system continues doing so if the checked value lies within the permissible limit else it will take a note of it by prompting the system to take necessary action that will prevent the damage. We may have one useful approach in which we may use agents to check for various possible failures depending on few noted parameters like Mean Time to Failure (MTTF), Mean Time Between Failures (MTBF), system uptime, age etc., and give a score to the concerned resource. Accordingly, whenever the system feels that the resource or a computation is entering a danger zone where a failure may take place, it preempts the application to start afresh on the same or different available resource. For example the Memory Usage Monitor agent (MMA), keeps a check on the memory usage and if it feels that the usage is exceeding the threshold it reinitiates the job on a different node. Similarly we can have a different number of agents to keep watch on the hardware, application operating system, network or response time.
- *Reactive approach*: Since faults can be in many forms and can occur at any time so the grid should be able enough to cope with such a scenario by using the fault masking approach in which it should detect and locate the fault and reconfigure the system or restart from the checkpoint. These checkpoints should be placed concurrently during the program execution rather than blocking the program and then check pointing. Further, fault containment approaches should be used to isolate the

fault from propagating throughout the system. For e.g. if a node has crashed a message should be propagated to isolate that node from the healthy resources.

- *Redundancy*: Introduction of redundancy by various means is one of the measures taken for increasing the fault tolerant appetite of the grids. This is done by duplicating the resources i.e., hardware redundancy or by having application run at more than one place in a different manner i.e., software redundancy or having the information stored at more than one places as backup or information redundancy or by having same application run at the same time at different resources or time redundancy.

Primary- backup technique is one of the proposed approaches that can be used for the grids. The client may interact with one replica and can start interaction with the other one if the first replica fails. This requires the two replicas to be consistent with each other. Therefore the primary should not acknowledge the client till it is sure that all the back up's have been updated. This approach is best suited for crashes and message loss but not for arbitrary failures. For example, if the failure is due to faulty algorithm then even the back ups are not of any use. Rather it will increase the complexity of the system.

- *Recovery from failure*: In addition to the above considerations of the types of failures to which the grid is prone one has to keep in mind the heterogeneous nature of the grid as well as the tasks for designing the recovery mechanism. This is because of the fact that each task has its own failure semantics having failure definitions and strategies to handle it. For e.g., a real time application may desire to finish in a specific time. For e.g. if it is not done due to node failure, it could be restarted on the same or other resource or even replicated to meet further failures whereas for some other task it may not work for which the reason of failure was improper algorithm. Thus, we need to have a *task specific failure* handling rather than the conventional failure independent policies in which the failures are rather homogeneous in nature, which need almost the same treatment as in the case of, distributed transactional databases where the recovery method is logging and rollback. the failure handling should be *user specific* as well i.e., it should be able to support the recovery method suggested by the user in the form of exceptions. Considering these facts we may have a task level recovery techniques that are used at the task levels to undue the effect of failure or workflow level techniques specifying the failure recovery at the application level. At the task level it includes *retrying* in the hope that the same cause of failure may not happen again, *check pointing* so that we do not need to start from the beginning but at the marked checkpoint using checkpoint libraries such as Dome, Fail-safe PVM, CoCheck etc., for parallel working platforms and Libckpt, Condor checkpoint library for standalone machines and adding redundancy as *replication* with more than one tasks running at parallel guaranteeing at least one successful execution. At the workflow level it includes *alternative task* in which we have more than one available implementation scheme for a certain task or *workflow redundancy* with many versions of tasks running in parallel.
- *Grid reliability modeling* : More the fault tolerance of the system, more reliable it is. The system reliability can be determined and modeled by either combinatorial approach by using probabilistic techniques or by using Markov modeling by representing the grid as a collection of states and state transitions.

5. Dynamic nature of the grid and security: Since the Virtual organizations comprises of a group of individuals and associated resources and services but not located within a single administrative domain for security reasons, a variety of issues relating to certification, group membership, and authorization also need to be addressed. These may range from security of the application to the safety of the data involved with it. As well, since the constituents of grid itself are changing, the grid should be able enough to live up by adapting to the security requirements of this dynamic environment.

- *Dynamic nature of the VO:* The virtual organizations themselves are dynamic due to the fact that even new services may be deployed or removed in the virtual organization at any time. It may range from a long lived collaboration for e.g., scientific grid while the other may involve a very short lived collaboration including two individuals sharing some documents for the purpose of a proposal making. Thus the user should be able to create new services dynamically without administrator intervention. These services should be capable enough to coordinate and interact securely with other services. For this purpose, the grid needs to establish a trust domain among the participants as well as its own resources.
- *Security specification:* Here the security issue is more complicated as compared to the simple client-server mechanism since in this case the parallel computation ranges to a large number of processors involving even a large number of resources supporting them. Adding to complexity is the fact that the resources may be in different administrative domains. We need to establish trust between these domains. In order to establish trust the two entities need to find out a common set of security mechanisms that both understand. Integration of GSI (Grid Security Infrastructure) over the OGSA (Open Grid Services Architecture) enables the use of Web Services techniques to express and publish policy [16] allowing applications to determine automatically what security policies and mechanisms are required of them along with the interface specification, token formats etc., and gather them to participate in the grid for problem solving. The dynamic nature of the grid even makes it a difficult job as the trust can't be ascertained prior to the application execution.
- *Protection of applications and resources:* Security of the data and application over the grid is very important since the grid involves the participation of many organizations with data flowing between these organizations. Data security is important even when staged on the grid resource. Proper measures should be ascertained to safeguard data used in the grid against active and passive intrusions for the purpose of message protection. One of the possible solutions is to encrypt the data before transmission. As well provisions should be there that the data should be secured even from the current resource on which it is used which is not in fact the owner of that data. The operating systems may help for this problem by defining a set of permissions as in the case of UNIX. For the case of a single organization grid the cluster resource management such as Sun Grid Engine (SGE) could be used to provide mechanism for the shared use and within the completion deadline of the tasks.
- *Compliance with the existing security standards:* We should not forget that the participants in a VO also have their own security arrangements in which they have invested a good enough sum. So the grid security measures should comply with them and interoperate rather than replacing them. Whatever security mechanisms are




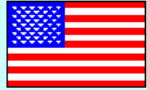

incorporated should not be on the static basis in the form of patches. The grid should be dynamic in the sense of adapting to the new security measures changes. Since the grid participants have different trust domains the credential conversion service may be used for both the parties to comply with. In addition Audit trail may be used to securely log the events that occur in the virtual organization of the grid.

The Globus Toolkit for example uses a common credential format based on X.509 identity certificates, which in conjunction with an associated private key forms a unique credential set that a grid entity may use to authenticate itself to the other grid entities. The Transport Layer Security (TLS) based protocol is used to perform authentication and then provide message protection. The Kerberos Certificate Authority (KCA) and PKINIT provide translation from Kerberos to GSI and vice versa for the purpose of credential conversions.

6. Grid Economics: Since grid enables its users to share resources within and between organizations it offers attractive value proposition in terms of efficiency and flexibility. But this sharing comes with a price thus bringing all the financial issues related with the grid business into the picture. From the business point of view the grid manager would try to extract the maximum value out of the available resources. Some of the core issues related to grid economics could be

- *Avoid the tragedy of the commons:* Since the grid is a shared resource one should always keep in mind that “who gets what, when and how much?” If the grid resources are allowed to be accessed in a free uncontrolled manner, the use could result in someone holding the resources for an undesired amount of time thus prohibiting the others to take advantage of the grid resources. This is called the tragedy of commons in which individually rational action of the members have a negative effect on the overall population. One possible solution to this problem is to restrict the grid access to the members who actually need it. This can be achieved by pricing the access rights, taxation on over use, access regulation for congestion control etc.
- *Discover and communicate dynamic value:* Resources on the grid have dynamic value which depends on the user needs. The value could be considered zero in the case we consider never exhausting resources. But in case of grid the resources are scarce. Thus a mechanism should be incorporated for dynamic price discovery depending on the user urgency requirements.
- *Use real money:* Since a number of geographically distributed organizations may participate in grid, a proper convertibility of the currency into value of resources
- should be ascertained.
- *Guarantee property rights:* Once the user is paying for the grid usage, a guaranteed QoS (Quality of Service) should be provided to the user which could be agreed upon prior to the grid use.
- *Use futures market:* Since the grid is a scarce resource the grid use should be promoted on the reservation basis rather than on the spot participation only. This will further help in reducing the price volatility.
- *Establish trust:* The system should be fully secured to ensure trust between the participants based on legal and financial consequences.

Many grid projects

<ul style="list-style-type: none">■ Australia<ul style="list-style-type: none">■ Nimrod-G■ GridSim■ Virtual Lab■ ..new coming up■ Europe<ul style="list-style-type: none">■ UNICORE■ UK eScience■ EU Data Grid■ EuroGrid■ Dutch DAS■ XW, JaWS■ Japan<ul style="list-style-type: none">■ Ninf■ DataFarm■ Korea...<ul style="list-style-type: none">■ N* Grid	  	<ul style="list-style-type: none">■ USA<ul style="list-style-type: none">■ Globus■ Legion■ AppLeS■ NASA IPG■ Condor-G■ NetSolve■ AccessGrid■ and many more...■ Public Forums<ul style="list-style-type: none">■ Global Grid Forum■ Grid & CCGrid conferences	 
---	--	---	---

7. **Conclusion:** Since the field of grid computing is quite young, much work is still to be done to establish that the same protocols applies equally well to all types of grids as the requirements for the grid are also dependent on the nature of the services it is designed to provide. We have tried to throw light on various issues and challenges keeping in mind the heterogeneity and dynamic nature of the grid. The requirements were modularized to give a proper insight into the requirements.

The issues rose started from the expectations from the end system in a grid environment, which are quite different from that of the other situations. Since the end system plays a major role, these requirements were addressed first. Other issues start bothering ever since the job submission is taken into account, ranging from the credential obtaining to resource discovery to submission of the job to monitoring the progress of the task over the grid. These issues were also considered. Since the job needs to be allocated, allocation issues were taken care of for the ordinary and time specific jobs. Taking into account the complex nature of the grid every discussion is incomplete if the system reliability is not concerned. So the fault tolerance and reliability issues were addressed. Light was thrown on the factors deciding the fault tolerance of the grid by discussing various possible faults and appropriate recovery methods. Various problems that may arise in allocation due to dynamic nature of the grid were considered. Since the grid is heterogeneous and dynamic in nature its security requirements are also different. These security threats and the attitude of the grid towards these requirements were discussed. Since the grid involves sharing of resources by various participants financial issues gains

importance. Key issues related to grid economics were raised and discussed. Thus we can say that the next big challenge for the grid is going to be dominated by business related issues along with the technical aspects. For a computational grid to be fully functional, these issues are to be taken care of by the research community. These are open issues and provide a good piece of work for the developers and researchers.

8. References:

1. Anderson, A.H., (2004) An Introduction to Web Services Policy Language. [Proceedings of Fifth IEEE International Workshop on](#) Policies for Distributed Systems and Networks, pp. 189-192.
2. Casanova H.(2002), Distributed Computing Research Issues in Grid Computing, ACM SIGACT News, Volume 33, Issue 3 (September 2002), pp. 50-70.
3. Dierks, T. and Allen C., (1999) The TLS protocol version 1.0, IETF. <http://www.ietf.org/rfc/rfc2246.txt>.
4. Foster, What is the Grid? A Three Point Checklist. Grid Today, Vol. 1, No. 6, 22 July 2002.
5. Foster, Ian., Kesselman, C. Computational Grids, (1998) The Grid: Blueprint for a Future Computing Infrastructure. Morgan Kauffman, pp. 1-29.
6. Huda Mohammad Tanvir, Schmidt W. Heinz, Peake Ian D. (2005), An Agent Oriented Proactive Fault-tolerant Framework for Grid Computing, Proceedings of the First International Conference on e-Science and Grid Computing (e-Science'05), IEEE, 2005.
7. Hwang S. and Kesselman C, (2003) Grid Workflow: A flexible Failure Handling Framework for the Grid, Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing (HPDC'03), pp. 126-137.
8. Johnson Barry, (1990), "Reliability and Fault Tolerance Issues in Intelligent Computing Systems". 5th IEEE International Symposium on Intelligent Control, Vol. 1, 1990, pp. 267-272.
9. Shan, Olikier, Biswas, (2003) Job Superscaler Architecture and Performance in Computational Grid Environments, ACM/IEEE Conference on [Supercomputing, 2003](#), 15-21 Nov. 2003, pp. 44 – 44.
10. Vidyarthi D.P., Tripathi A.K., Sarkar B.K., (2001) Allocation aspects in Distributed Computing Systems, IETE Technical review, Vol 18, pp. 449-454.
11. Welch Von, Siebenlist Frank, Foster Ian, Bresnahan John, Czajkowski Karl, Gawor Jarek, Kesselman Carl, Meder Sam, Pearlman Laura, Tuecke Steven, (2003) Security for Grid Services. Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing (HPDC'03), 1082-8907/03.
12. Wiriyaprasit, Sirappa., Muangsin, Veera, The Impact of Local Priorities Policies on Grid Scheduling Performance and an Adaptive Policy-based Grid Scheduling Algorithm. Proceedings of the Seventh International Conference on High Performance Computing and Grid in Asia Pacific Region (HPCAsia '04), pp. 343-346.
13. Zhang X., Zagorodnov D., Hiltunen M., Marzullo K., Schlichting R, (2004), Fault Tolerant Grid Services Using Primary-Backup: Feasibility and Performance. IEEE International Conference on Cluster Computing pp. 105-114