International Journal of
Emerging Research in Management &Technology
ISSN: 2278-9359 (Volume-2, Issue-4)

Research Article | **April 2013**

Survey Report of Job Scheduler on Grids

Patel Sohil K.
PG Student,C.S.E Department,
Parul Institute of Technology, Vadodara, India.

Abstract—

This report is a Survey of the architecture of schedulers for grid systems. The schedulers being used in various grid systems have been designed for the applications that the grid is designed to handle. The survey presents a detailed study of the schedulers that are in use, it describes the architectural model in each case and it compares the features of the available schedulers. The survey contains the architectures of the available schedulers in detail and the architecture of the scheduler for the global grid as a conceptual model. A scheduler on a grid has to deal with a continuous load of diverse applications. It has a dynamic, geographically distributed and heterogeneous set of resources. It is designed to map jobs to resources with certain specified performance goals. The survey describes the methods of obtaining information about the resources, the performance goals that a scheduler may attempt to achieve and the process of mapping jobs to resources.

Keywords— Jobs, Scheduling, Resource, Scheduler, Agent.

I. INTRODUCTION

"A computational grid is a hardware and software infrastructure that provides dependable consistent, pervasive and inexpensive access to high-end computational capabilities" [1]. Today the users visualize a grid in a number of different ways. The common grids are research grids. A number of academic-research organizations have set up grids both for high performance computing as well as for grid research. Most of such grids are an inter-connection of clusters, usually through a dedicated wide area network. In general a cluster consists of homogeneous, dedicated and tightly coupled workstations that may be used to process high-performance jobs. A grid, on the other hand, consists of heterogeneous and intermittently available workstations. Whereas a cluster is expected to be located at a single laboratory or office, the workstations on a grid may be widely distributed geographically.

The structure of a grid is heterogeneous and dynamic. Moreover it is expected to be used for a wide variety of applications. Scheduling of jobs on a grid or a cluster is the task of allocation of jobs to the available nodes. The objectives of grid scheduling are fast turn-around time and maximum throughput [2]. Scheduling in a cluster is a relatively simpler task. On a grid, which is dynamic, heterogeneous and geographically spread out and which caters to highly diverse applications, scheduling is a NP-complete problem [3]. For any user, whether on a cluster or a grid, the turn-around time is required to be low. But the objectives of a service provider on the grid may be a high throughput in grids with a high load. For lightly loaded grids, the objective may be to distribute the jobs to all the nodes in an equitable manner. Scheduling has to take into account the objectives of the service provider, the needs of the user and the requirements of the applications.

A. Scheduling Systems

Grids have evolved from a natural progression of parallel processing systems, distributed processing systems and a meta-computing system. Parallel processing systems can be classified into two types:

1. Shared Memory Processing systems
2. Clusters of workstations

Centralized systems designed to cater to the needs of a large number of users are usually multi-processor systems with a shared memory. A distributed system consists of heterogeneous systems connected in parallel. A grid is a dynamic and geographically spread-out distributed system. There are some common properties that a scheduler has to satisfy, whether it is for a parallel or for a distributed system or for a grid. A scheduler is designed to satisfy one or more of the following goals:

- Maximizing system throughput
- Maximizing resource utilizations
- Maximizing economic gains
- Minimizing the turn-around time for an application.

In addition a scheduler may be designed to follow a specific or an adaptable scheduling policy. Optimization of scheduling process can be attempted when performance goals and scheduling policies have been defined. Since Grid technology is developed as an extension of technologies used in clusters and distributed systems the next section describes the characteristics of scheduling in clusters.

B. Cluster Scheduling

A cluster consists of machines connected together through a high-speed network. A cluster consists of homogeneous, dedicated and tightly coupled workstations. A cluster is managed by a single administrative entity. A cluster has a server, which interacts with the users called clients. The server also has information about the all computer nodes called Agents in the cluster. A client may be connected to the server through Internet. The client has to inform the server about the requirements of the application. The inter-dependency of the processes of the application has also to be specified either through a script or through a GUI interface for application building. A cluster can run both sequential as well as parallel jobs. According to [4], "The key to making cluster computing work well is the middleware technologies that can manage the policies, protocols, networks, and job scheduling across the interconnected set of computing resources." The two main goals of cluster scheduling are to obtain maximize the utilization and throughput. The schedulers most commonly used by the high performance community are Maui scheduler [5] and PBS scheduler[6]. These schedulers allow the administrator to set the following parameters:

- To set multiple queues for different job classes
- To set priorities and scheduling policies for each queue.
- To treat interactive jobs differently from non-interactive jobs.

Even though the facility to set the parameters is useful to the administrator, if the clusters are served with continuously varying applications, optimization of scheduling remains a desirable but an elusive goal.

C. Grid System Taxonomy

Grid systems can be classified, into three types.

- Computational grids
- Data grids
- Service grids

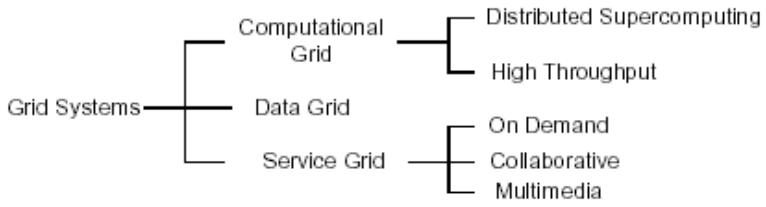

Figure 1: Grid Systems Taxonomy

The aggregate computational power of a computational grid is larger than the computational power of any single constituent of the grid. High Performance computing problems, which require large computational power, use the grids for parallel computing. [7].A data grid provides large data repositories such that both the data storage as well as the users may be distributed over a large geographical area. Users may mine the data for studies in different ways. As an example, the astronomers, as a loose group, have been able to create a large data grid. Large telescope systems continuously scan the outer space and feed the data into large data repositories. These are then used by astronomers from around the world [7].Service grids provide services not available at a single site from a single machine. Examples of such grids are On Demand computing, Multimedia computing or Collaborative computing. Grids may also be classified on the basis of administrative domains covered by it. An IntraGrid may be controlled by a single, multi-location organization. Sharcnet is an example of such a grid. This consists of big clusters at 4 universities connected by a dedicated Internet.An Extra Grid may couple two or more grids. ERP systems of today use such grids whereby a business entity may be connected to its suppliers and its sales organizations. Global Grid is the kind of grid being conceptualized by the Global Grid Forum. Such a grid would have a large number of resource providers connected through Internet to the users. Middleware for such a grid is being developed by a number of research groups of GGF and at various Universities.

1) *Resource Management System*:

A scheduler is a component of the Resource Management System (RMS) on a grid. Figure 2 shows the architecture of a general RMS. Resource Information from the sensors of the service-provider forms one input of RMS. The other input is the Resource Requests from the users. The resource information inter-acts with the Discovery module of RMS. The State Estimation system converts the information, through the Naming system to a Resource Information database. The Resource Monitoring system maintains the Historical Data and the present Resource Status databases. The Dissemination system may be used to convey the resource information to users on the grid. The Request Interpreter may format the request, or in some cases like the TITAN system, it may generate estimates of execution time for each task of the job request. The Resource Broker will negotiate for the resources. It may use Resource Reservation module to save information about the reserved resources in the Reservations database. The Scheduling system will put the job in the Job Queues. Then the Execution Manager sends the job for execution to the allocated resources. The Job Monitor

continuously monitors the job, as it is being executed. It stores the state of the job at checkpoints in the Job Status database.

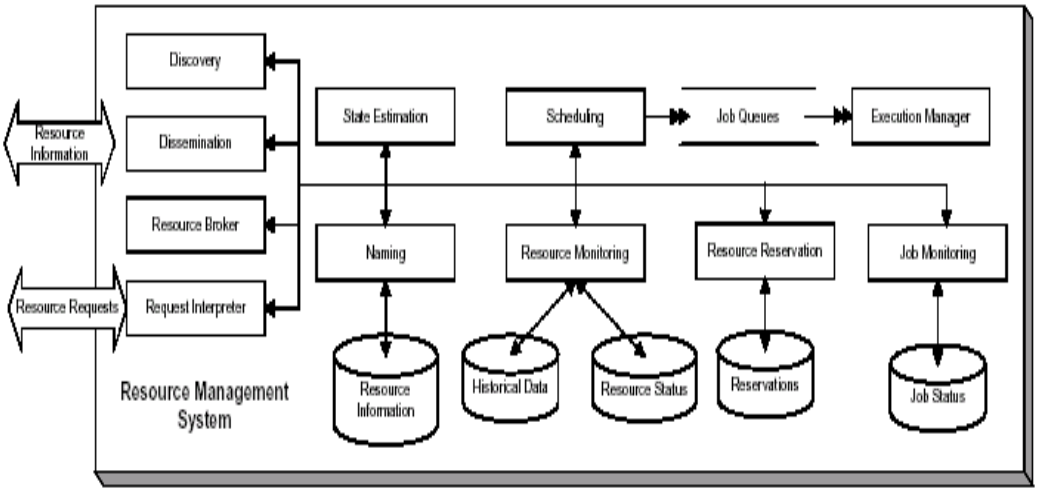

Figure 2: Resource Management System Abstract Structure

D. Grid Scheduling Problem

[1] defines the responsibilities of a grid scheduler as the "discovery of available resources for an application, mapping the resources to the application subject to some performance goals and scheduling policies and loading the application to the resource in accordance with the best available schedule."In addition a grid scheduler must provide the facility of check pointing so that a job can be migrated, along with its state, to a different node. Migration may be required in case a pre-emptive scheduling policy is considered due to failure of some nodes or some part of the network. Facilities of suspension of processing, resuming the processing or aborting a job are also usually provided in grid schedulers.

II. GRID SCHEDULING

Grid Scheduling is used to "efficiently and effectively manage the access to use grid resources by the users" [7]. According to [8] grid scheduling problem consists of three main components.

- Consumer(s).
- Resources(s).
- Policy.

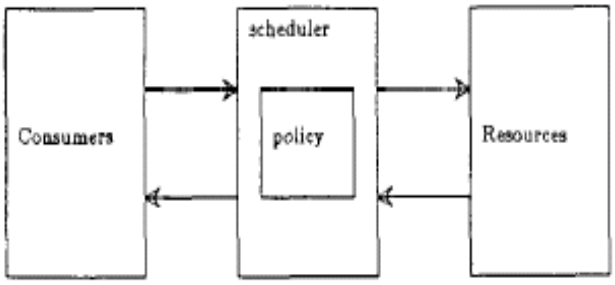

Figure 3: Scheduling system

The policy chosen for implementing a Scheduler would affect both the Users as well as the Service Providers.

A. Grid Scheduling Approaches

1) First-Come-First-Serve (FCFS): The job arrival times and the arrival rates may vary for a grid. A grid scheduler that follows the FCFS policy will allocate the jobs to compute nodes in the order of their arrival. The jobs, which arrive when all the resources are in use, will wait in a Wait queue. "This strategy is known to be inefficient for many workloads as a large number of jobs waiting for execution can result in unnecessary idle time of some resources" [9].

2) *Backfilling*: Backfilling refers to the case, where a task at the back in the Wait queue, is chosen on the basis of some criterion to be processed before the first task in the Wait queue is taken up for processing. Sometimes when a compute node becomes free, it may not be possible to allocate the first task in the wait queue due to dependencies among

the tasks. In such a case rather than keeping the node idle, an out-of-order task may be allocated to the free node. Two common variants are Easy and Conservative backfilling. In Conservative backfilling, the out-of-order task must be chosen such that the first job in the Wait queue is not delayed. "The performance of this algorithm depends upon a sufficiently large backlog." [9].

3) *Gang Scheduling:* In this approach the main concept is to "add a time-sharing dimension to space sharing using a technique called gang scheduling or co scheduling. This technique virtualizes the physical machine by slicing the time axis into multiple virtual machines. Tasks of a parallel job are co-scheduled to run in the same time-slices (same virtual machines) "[10].

4) Genetic Algorithms (GAs): GAs are useful for optimization problems, in cases where the number of parameters, which affect the performance are large. GAs begins with a potential set of solutions, called the population of the search space and a fitness function, appropriately defined for the problem on hand. GAs then try to obtain an optimum solution by using the processes of cross-over and mutation. "GAs are widely reckoned as effective techniques in solving numerous optimization problems because they can potentially locate better solutions at the expense of longer running time. Another merit of a genetic search is that its inherent parallelism can be exploited to further reduce its running time" [11].

5) *Directed Acyclic Graphs Scheduling*: A job, which is to be run on a parallel system of computational nodes, can be represented by a weighted-node and weighted-edge DAG. The weight of a node is the time that it would take to be processed. The directed edges define the dependencies. The weight of the edges may represent the communication delays."The objective of DAG scheduling is to minimize the overall program finish-time by proper allocation of the tasks to the processors and arrangement of execution sequencing of the tasks. Scheduling is done in such a manner that the precedence constraints among the program tasks are preserved. The overall finish-time of a parallel program is commonly called the schedule length or makespan" [11].

6) *Priority Queuing*: Priority Queuing has two sub types such as Head of Line (HOL) for a fixed priority and Dynamic Queue (DQ) where each arriving job is allocated a priority. Its priority level determines the position of the job in the queue[10].Priority Queuing Methods: Three Issues.While implementing priority queuing, two issues have to be considered to avoid pitfalls.

- Starvation: When high priority processes continuously enter the queue, lower priority processes may not be able to get processing time. If pre-emption is allowed, even if a low priority process were to start getting processed, it may be pushed out as soon as a high priority process was to come in. In such a situation, the lower priority processes are said to face starvation.Solution: Increase the priority level of a process depending upon the waiting time in the queue of the process. Thus over a period of time an initially low-priority process may be able to beat a high-priority process, which has just entered the queue.
- Deadlocking: A high priority process may need some resource, which can be created only when a lower priority process is executed. However since the high priority process is supposed to be processed first, a deadlock may occur, in that neither the high priority nor the low priority process may be executed.Solution: Priority Inversion: The priority of the lower-priority process may be temporarily boosted so that it may be executed, before the high priority task is processed.

B. Types of Jobs on Grid

A grid has heterogeneous dynamically available resources. In addition it handles a great diversity of jobs. A grid may have three different categories of jobs defined by [12]:

1) Local jobs: The node or a cluster may have to process local jobs. Usually such jobs have a higher priority on locally available resources.

2) Batch jobs: considered for batch processing: such jobs may be required to be processed as the resources become available. The jobs may have to follow a queuing discipline, as specified by the administrator.

3) Reserved jobs: Reserved time-slot jobs: such jobs may have to be processed as the highest priority jobs, during the pre-specified time slot. Or these jobs may have pre-specified deadlines.

Each job may consist of a number of processes. The processes may have a variety of inter-dependencies. If the interdependency is not serial in character, it may be possible to run different processes of a single job in parallel on different nodes. In addition a grid scheduler may handle a large variety of jobs from a multiplicity of users. Thus on a node may be mapped processes from multiple jobs in succession.The jobs may be inter-active or non-interactive. These may be real time jobs having deadlines or batch jobs. The jobs may have different levels of priority. Moreover each site

on the grid may have a different scheduling policy. Such diverse jobs on intermittently available resources make the grid scheduling highly complex.

C. Grid Scheduler Performance Criteria

Since grid applications have great diversity, it is not possible to specify a single test of performance criteria, which can be applied to all cases. Depending upon the environment, some of the following criteria may be used to evaluate the performance of a scheduler[13].

- Improvement from the User's perspective
- Improvement from the Service Provider's perspective
- Minimum overhead of the scheduler
- Fault tolerance
- Scalability
- Applicable to a wide diversity of applications and grid environment.

III. CLASSIFICATION OF GRID SCHEDULERS

Many scheduling systems for grid environments have been designed. However each of the schedulers has been designed for a specific grid structure and each has some unique features.

A. Organizational structure based Scheduler

The grid schedulers can be classified into three categories based on its organizational structure given by [14].

1) Hierarchical Schedulers: A hierarchical structure consists of local schedulers and higher-level schedulers, which work in concert. However such a system may not be able to provide local autonomy for setting local scheduling policies.

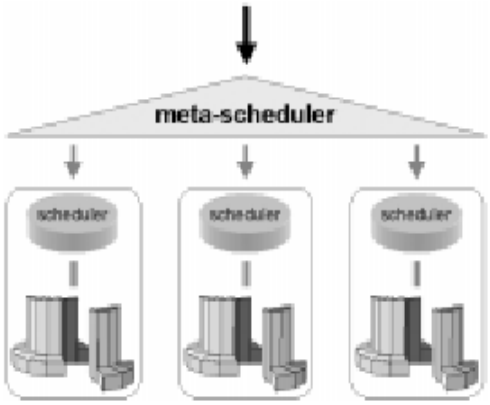

Figure 4: Hierarchical scheduling

2) *Centralized schedulers:* In a centralized scheduler, all applications are added to a common queue of a centralized scheduler. Each site has no queue and scheduling function is not performed at the sites. Since the central scheduler has information about all the jobs and all the resources, conceptually it should be able to provide an optimal solution. However such a scheduler is slow and not scalable as the grid grows.

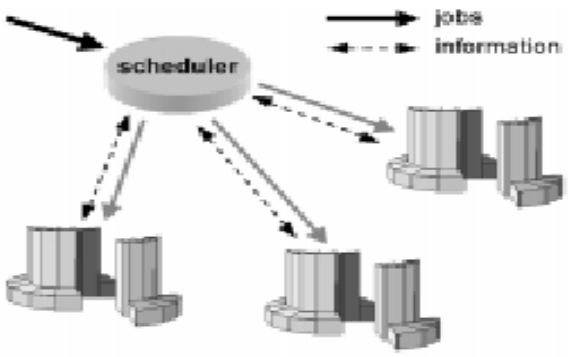

Figure 5: Centralized scheduling

3) *Decentralized Schedulers:* A decentralized scheduler has local queues at each site. It responds to requests for service from local users and from other schedulers.Such a system has no knowledge about user requests received at other schedulers. Nor does it know about the resources at other sites. So the scheduling may not be optimal. But such a system is fault-tolerant. It can also use scheduling policies as required by the local administrator.

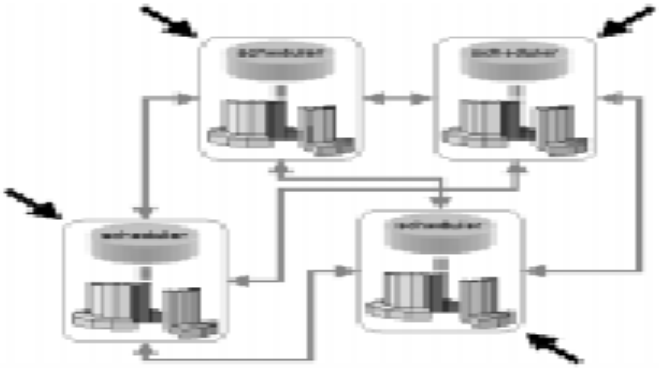

Figure 6: Decentralized scheduling

B. Application / Resource based Scheduler

1) *User-centric scheduler:* The User-centric Performance goals can be defined as follows: [15]

- Minimize execution time
- Minimize the waiting time in the ready-to-process queue
- Maximize speed-up on the user's own platform i.e. minimize the turn-around time
- Minimize process slow-down, defined as the ratio of turn-around time to the actual execution time.

Application level scheduler AppLeS [15] is a User-centric scheduler.

2) *Service-provider centric scheduler:* The Service-centric Performance goals can be defined as follows: [15]

- Maximize throughput. (Throughput is the number of jobs processed per unit of time.)
- Maximize utilization. (Utilization is the percentage of time a resource is busy)
- Minimize flow time. (Flow time or session time is the sum of completion time of all jobs.)

A service-provider centric scheduler does not try to obtain detailed information about the characteristics of the application and it tries to optimally use the computing power of the grid. Condor [2] is an example of a service-provider centric scheduler.

3) *Economy based scheduler:* A third case of scheduling is called Economy-based. The scheduler mimics the market place where the user's application is to execute at a particular total maximum cost or it is required to meet certain deadlines. The user is interested in satisfying the requirements of the application at the minimum possible cost. The goal of the service-provider is to obtain the highest price for its services. Thus each resource is characterized by its cost and its computational capacity characteristics. The scheduler tries to provide the service to the user at the lowest cost while providing maximum returns to the service provider. Nimrod-G [21] is a scheduling system based on such market-like considerations.

C. Other characterizing features

Each of the class of schedulers described in above Section may have some of the following characteristics also.

1) *Online vs. Batch Schedulers:* Online schedulers allocate a node to an application as soon as the application is received. A Batch scheduler puts the received applications in a queue. At regular scheduling events, it allocates the applications to nodes, by following its scheduling policy. The overhead of online scheduler is much higher than the overhead of batch schedulers. Most of the practical schedulers are batch schedulers.

2) *Preemptive / Non-preemptive based:* Preemptive schedulers allow a job, which is being processed at one node to be rescheduled to a different node. However rescheduling adds a great deal of overhead, since the entire state of a partially processed task has to be transferred. Most of the practical schedulers are non-preemptive type.

IV. EXISTING GRID SCHEDULERS

Many of the available grid schedulers are centralized. They assume that the jobs are received at a central point and the scheduling process is controlled by the scheduler, which has complete information about all the machines. Condor, Nimrod and GrADS are examples of the centralized schedulers.

A. Condor

The Condor High Throughput Computing System is a combination of dedicated and opportunistic scheduling. "Opportunistic scheduling involves placing jobs on non-dedicated resources under the assumption that the resources might not be available for the entire duration of the jobs" [2].

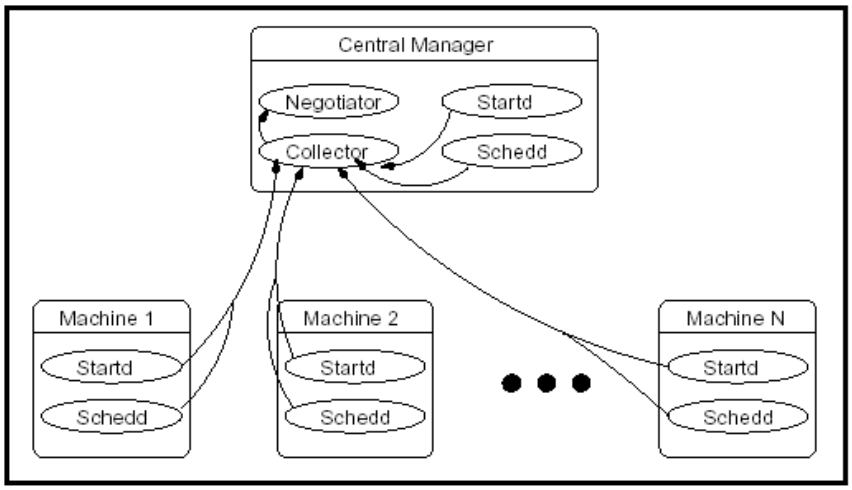

Figure 7: Architecture of a condor pool with no jobs running.

As shown in Figure 7, the architecture of Condor uses a daemon called the collector, for centrally maintaining the list of Condor resources. Periodically updates are sent to the collector by Condor daemons on computing machines. The collector runs on the machine called the Central Manager. The Central Manager consists of negotiator, which tries to find a match between resource requests, and resource offers [2]. Each computational resource within the pool is represented by a demon called startd. Another demon called Schedd, handles the user jobs. All the jobs are submitted to Schedd. The main functionality of Schedd is to maintain a job queue, to publish resource request and to negotiate for available resources. Condor, assumed that jobs, to be processed by a grid, would be independent and non-real time jobs. Resources are assumed to be independent workstations. The information about the computing resources only specifies the availability of the resource. When it is used for grid processing, the resource is not to be used for processing local jobs. Thus it is not time-shared. However as soon as a local job is loaded on a resource, the job brought through the grid will be pre-empted that is it would be moved to another resource available in the pool. Check pointing is used for storing the state of the processing so that, if required, the job may be moved. Condor does not take into account the overhead of transferring a job. Condor scheduling is designed to increase the utilization of workstations within a single domain [2].

B. Condor-G

Condor-G is a combination of Condor and Globus toolkit [16]. The Globus toolkit supports resource discovery and resource access in multi-domain systems. Condor-G allows a user to harness multi-domain resources as if they all belong to a single domain. It also uses authentication, authorization and secure file transfer facilities provided by the Globus tool-kit.

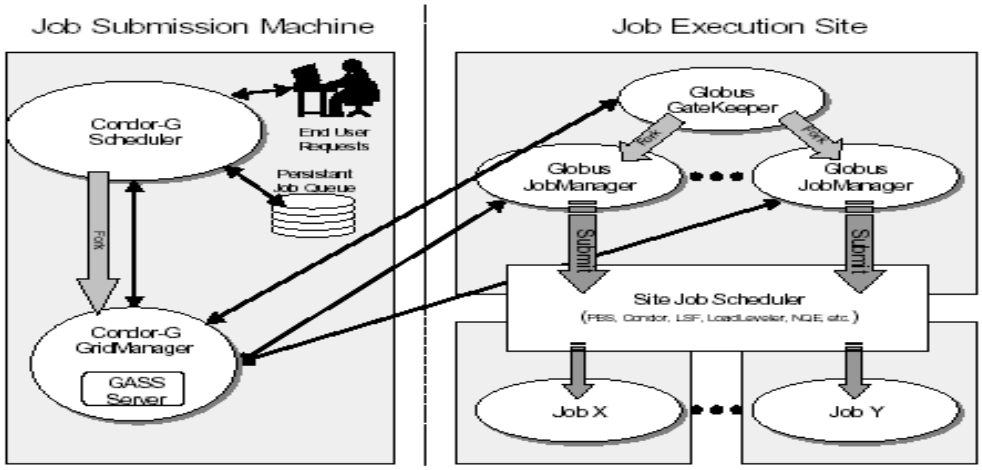

Figure 8: Remote execution by Condor-G on Globus- managed resources

As shown in the Figure 8, Condor-G uses Globus GASS (Global Access to Secondary Storage) file server, G.S.I (Grid Security Infrastructure) and GRAM (Grid Resource Allocation and Management) protocol. The user accesses Condor-G

scheduler from the user desktop. A local Grid-Manager daemon is created along with a GASS server. The Grid Manger gets authenticated with the Globus tool-kit at the job execution site. A Globus Job Manager daemon is created at the job-site. The Grid Manager and the Job Manager cooperatively get the job processed by using the available grid resources. The Condor-G scheduler at the user's desktop maintains a persistent job queue, so that failure of any part of the Grid would make available to the user the status of the job and state of the processed part up to the last checkpoint.

C. AppLeS Scheduler

AppLeS stands for Application Level Scheduler [17]. It has been designed for meeting performance goals, which may be specified by the application. Each application has its own scheduling agent. The agent monitors available resources and generates a schedule for the application.AppLeS is based on a client/server model. It is also designed for a single domain where resources are time-shared.The resource discovery and prediction is implemented through NWS. The user has to specify the necessary information about the application as well as the performance goal that the scheduler may attempt.A number of possible schedules are developed along with the expected performance index for each case. The schedule, which maximizes the performance goal, is selected and implemented. The scheduler adaptively learns from every cycle of implementation to refine its working.

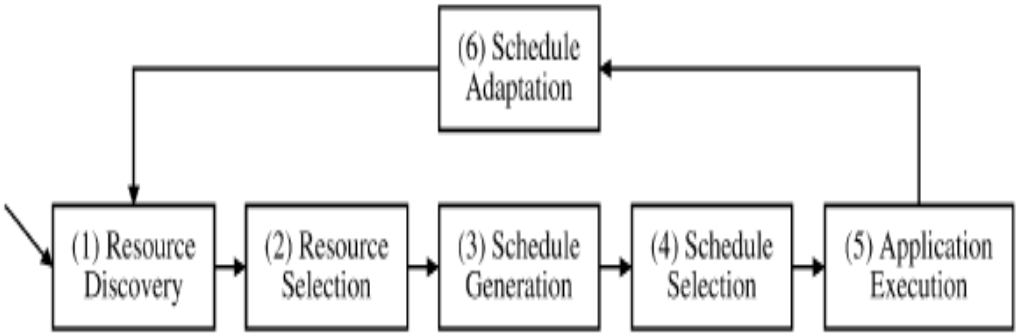

Figure 9: Steps in the AppLeS methodology

D. Nimrod-G Resource Broker

Nimrod-G was developed at Monash University, Australia. Nimrod mimics the model of a market for a single domain [21]. Nimrod-G has been obtained by redesigning Nimrod for operation with Globus tool kit as shown in figure 10. The resource discovery and job allocation over a grid is implemented through the Globus tool-kit whereas Nimrod handles the market mechanism.Each Nimrod application has a specified budget and a deadline, before which it should be completed.

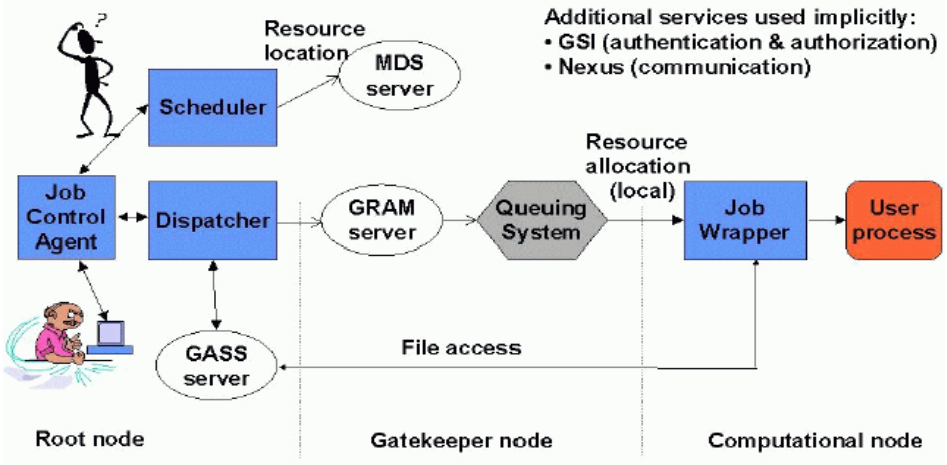

Figure 10: Nimrod/G and Globus Components Interactions

Each resource has a price, which must be paid if the resource is to be used. The scheduler is designed to complete the application within the deadline at the minimum cost. Similarly each resource tries to maximize the gain by providing its services. If the budget or the deadline should be exceeded, the user is informed about it. Every application in Nimrod-G has an associated Job Wrapper, which acts as a mediator between the resource and the application. It would also be used for sending the required information to the Resource Accounting system.Nimrod-G is part of a framework called Grid Architecture for Computational Economy (GRACE). GRACE includes a global scheduler called a broker. The broker works with other components like bid-manager, directory-server, and Globus tool-kit to maximize the performance goals.

E. GrADS Scheduler

The Grid Application Development Software (GrADS) project aims at making it simple for users to use grid resources [7]. After the application has been prepared in the GrADS program preparation system, it is delivered to the Scheduler/Service Negotiator system. The scheduler is a part of the GrADS execution environment. When the object containing the job with a performance contract is delivered to the Scheduler/Service Negotiator, it will broker the allocation and scheduling of grid resources for the job. Thereafter the dynamic optimizer is activated to adapt the program to the available resources. It will also insert sensors for monitoring the status of the job. The real time monitor verifies that the requirements of the performance contract are satisfied. In case there is a violation, the execution may be interrupted. Either the optimizer may adapt the program or new resources may be negotiated or both the steps be taken to ensure compliance with the performance contract. Thus the closed loop system of GrADS scheduler ensures Quality of Service (QoS).

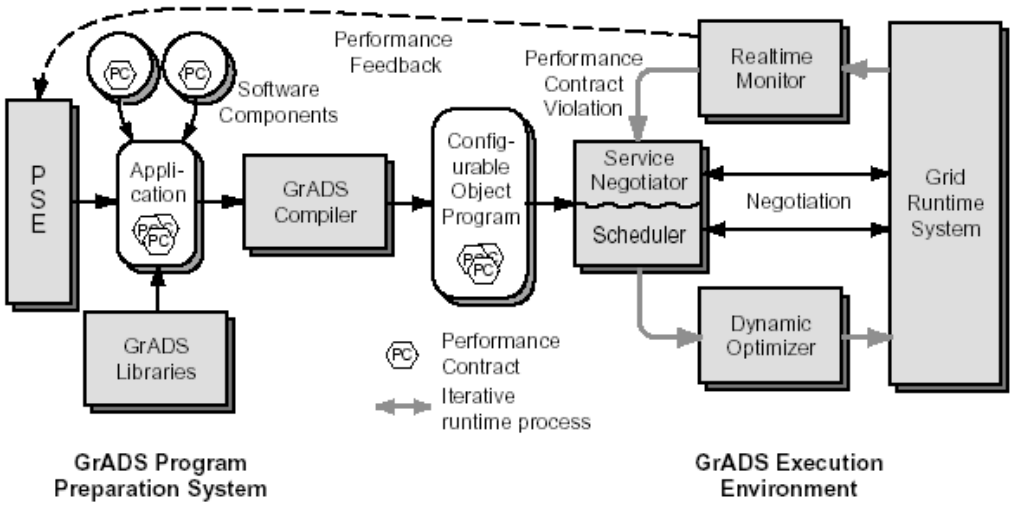

Figure 11: GrADS Program Preparation and Execution Architecture

F. The Legion Scheduler

The Legion scheduler is a part of the object-based grid middleware being developed at University of Virginia. "The Legion design encompasses ten basic objectives: site autonomy, support for heterogeneity, extensibility, ease-of-use, parallel processing to achieve performance, fault tolerance, scalability, security, multi-language support, and global naming" [19].The Legion scheduler is customized for each application and it aims at minimizing the turn-around time. Since a customized scheduler is used, Legion system can cater to a diverse set of jobs on heterogeneous resources.The Legion system consists of objects, different instances of which interact with one another. Thus an application is an object. The scheduler will use instances of this object on different resources as shown in the figure 12.

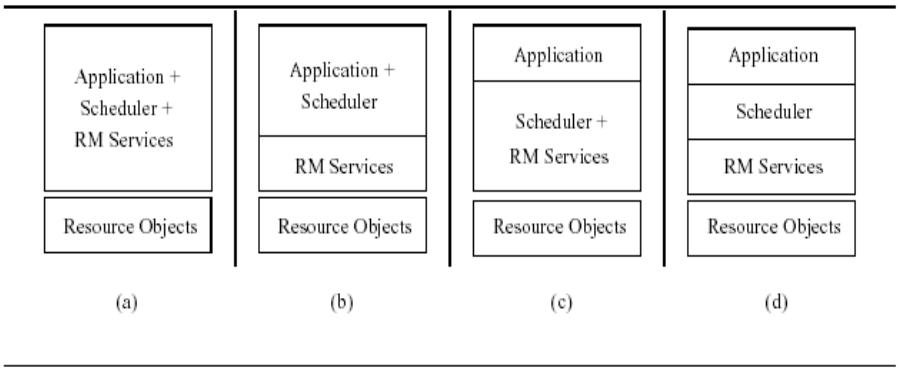

Figure 12: Choices in Resource Management Layering

Legion has a centralized Resource State Information Base. It has computational resources and storage resources. It does not yet have network resources as a part of the database.

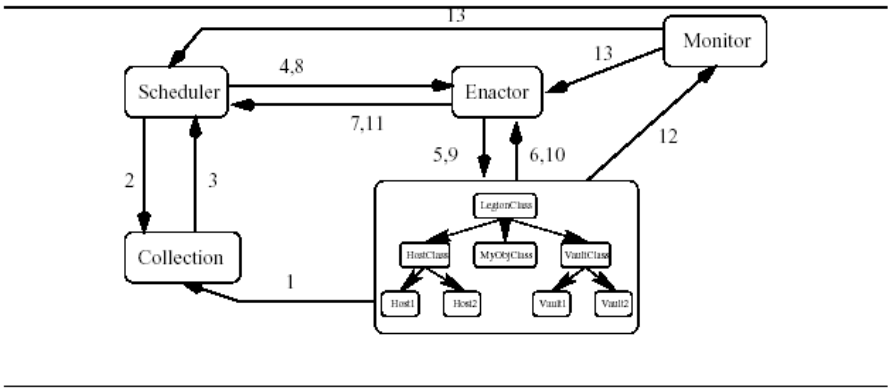

Figure 13: Use of the Resource Management Infrastructure

The customized scheduler, as shown in Figure 13, is a part of the application object. It interacts with the resource database to work out a schedule, which meets the performance goals of the application. The scheduler passes the schedule to the Legion Enactor. The Enactor makes the reservations and inter-acts with the scheduler, if any changes are required. Then the Enactor places the objects on the selected resources and monitors the status of the application. Legion system is designed to be secure and scalable.

G. TITAN

The TITAN architecture "employs a performance prediction system (PACE) and task distribution brokers to meet user-defined deadlines and improve resource usage efficiency" [20].Iterative heuristic algorithms are applied for performance prediction for job schedulers in Grid environments. Workload managers, Distribution Brokers and Globus Interoperability providers comprise the TITAN system's hierarchy, with Globus forming the highest level in the hierarchy. The algorithms of Deadline Sort, Deadline Sort with Node Limitation and the Genetic Algorithm have been tested. The Genetic Algorithm approach is found to be less sensitive to the number of processing nodes and to minor changes in the characteristic of the resource. Moreover it can take into account a large number of parameters of the resource system. The genetic algorithm is able to converge fast to balance the three objectives of makespan, idle time and the QoS metric of deadline time. TITAN uses Performance Analysis and Characterization Environment (PACE) for the predictor and brokers to meet performance objectives of a scheduler.

V. CONCLUSIONS

A global grid will consist of heterogeneous computational resources connected through high-speed network. It may also have many data repositories and code repositories. A scheduler system will be the interface between a user and the grid resources.

A. Performance Goals for a Scheduler for the Global Grid

For a global grid, a scheduler will have the characteristics of being hierarchical, batch-type and non-preemptive type. The schedulers have been developed from the perspective of efficient resource usage (Condor and Condor-G) or from the perspective of applications (AppLeS, Legion scheduler, GrADS) or from the perspective of a grid as market (Nimrod). Since the Globus tool-kit is becoming pervasive, schedulers like Condor-G or Nimrod-G may find wider acceptance. Most probably on a worldwide grid, an economy-based scheduler turns out to be the final choice.An open worldwide market in grid may lead to a user having a choice of being able to use resources from a number of resource providers. Hence the scheduling process is likely to be governed by the perspective of the user. In general a user may be interested in a short turn-around time. Or the user may be interested in having the processing completed within a specified deadline and cost. Such schedulers have not yet been built.

B. Architecture of a Scheduler for the Global Grid

In the survey a number of schedulers have been described. The architecture of each of the schedulers has also been described. But the architecture of a Global Grid scheduling system is not yet available in the published literature. But the ideas for such a system in a conceptual form can be garnered from a study of the Survey. A user will be able to approach a number of Resource Providers for processing application. The scheduling agent of the user application will negotiate with a service provider on the basis of the deadline and the cost. Once the User's Agent and the Resource provider's scheduling agent agree, the processing for the application would begin. It is possible that user's agent may do the transaction through a Broker rather than dealing with the Resource Provider directly. An effective accounting system and many components of such a scheduler are a subject of continuing research.

ACKNOWLEDGMENT

I am very thankful to faculty members and to my friends for their support. And at this moment, I would like to express my appreciation to my family members for their unlimited encouragement and support.

REFERENCES

[1] Foster, C. Kesselman, et al, "Computational Grids, The Grid: Blueprint for a New Computing Infrastructure", Morgan-Kaufman, San Fransisco, 1998.
[2] D Wright, "Cheap Cycles from the Desktop to the Dedicated Cluster: Combining Opportunistic and Dedicated Scheduling with Condor", Proceeding of HPC Revolution 01, Illinois, 2001.
[3] Rajkumar Buyya, Baikunth Nath, et al,"Nature's Heuristics for Scheduling Jobs on Computational Grids", *The 8th IEEE International Conference on Advanced Computing and Communications (ADCOM 2000),* 2000, India.
[4] J. H. Abawajy, S.P. Dandamudi,"Parallel Job Scheduling on Multi-Cluster Computing Systems", *Proceedings of IEEE International Conference on Cluster Computing (Cluster 2003),* pp.11-18, 2003.
[5] D. Jackson, Q. Snell, et al,"Core Algorithms of the Maui Scheduler", In D.G. Feitelson and L. Rudolph, editor, Proceedings of 7th Workshop on Job Scheduling Strategies for Parallel Processing, volume 2221 of Lecture Notes in Computer Science, pp. 87–103. Springer Verlag, 2001.
[6] Mark Goldenberg, Paul Lu, et al,"TrellisDAG: A System for Structured DAG Scheduling", *In Proc. of the 9th International Workshop on Job Scheduling Strategies for Parallel Processing*, volume 2862 of Lecture Notes in Computer Science, pp.21–43. Springer, 2003.
[7] R Buyya, D Laforenza, et al,"Grids and Grid Technologies for Wide-area Distributed Computing", *The Journal of Concurrency and Computation: Practice and Experience, v.14, Issue 13-42,* 2002.
[8] T.L. Casavant, J.G. Kuhl,"A Taxonomy of Scheduling in General-Purpose Distributed Computing Systems", *In IEEE Transaction on Software Engineering, v.14 n.2,* pp.141-154, 1988.
[9] U. Schwiegelshohn, A. Streit, R. Yahyapour, et.al.,"Evaluation of job-scheduling strategies for grid computing", *In Proceedings of 1st IEEE/ACM International Workshop on Grid Computing (Grid 2000),* v.1971, pp.191–202, 2000.
[10] Yanyong Zhang, Hubertus Franke, et al,"An Integrated Approach to Parallel Scheduling Using Gang-Scheduling, Backfilling, and Migration",*In IEEE Transactions On Parallel and Distributed Systems, v. 14,* pp.236-247, 2003.
[11] Y. K. Kwok, I. Ahmad,"Static Scheduling Algorithms for Allocating Directed Task Graphs to Multiprocessors", *ACM Computing Surveys, v.31 n. 4,* pp. 406-471, 1999.
[12] V. Subramani, et al,"Distributed Job Scheduling on Computational Grids using Multiple Simultaneous Requests", *11th IEEE International Symposium on High Performance Distributed Computing (HPDC-11),* pp.359-368, 2002.
[13] Francine Berman. High-performance schedulers," In Ian Foster and Carl Kesselman, editors, The Grid: Blueprint for a New Computing Infrastructure", pp.279–309, Morgan Kaufmann, San Francisco, CA, 1999.
[14] Volker Hamscher, Uwe Schwiegelshohn, et al,"Evaluation of Job-Scheduling Strategies for Grid Computing"' GRID 2000, pp.191-202, 2002.
[15] Francine Berman,"High-performance schedulers" *In Ian Foster and Carl Kesselman, editors, The Grid: Blueprint for a New Computing Infrastructure,* pp.279–309, Morgan Kaufmann, San Francisco, CA, 1999.
[16] James Frey, Todd Tannenbaum, et al,"Condor-G: A Computation Management Agent for Multi-Institutional Grids", *Journal of Cluster Computing, v.5,* pp.237-246, 2002.
[17] Francine Berman, Richard Wolski, Henri Casanova, et al,"Adaptive Computing on the Grid Using AppLes",*In IEEE Transactions On Parallel and Distributed Systems, v.14,* pp.369-382, 2003.
[18] Francine Berman, Andrew Chien, Keith Cooper, et al,"The GrADS Project: Software support for high-level Grid application development",*The International Journal of High Performance Computing Applications, v.15 n.4,* pp.327–344, 2001.
[19] S. J. Chapin, D. Katramatos, et al,"The Legion Resource Management System. Proceedings of the Job Scheduling Strategies for Parallel Processing",pp.162-178, Springer-Verlag, 1999.
[20] D. P Spooner, SA Jarvis, et al,"Local Grid Scheduling Techniques using Performance Prediction", *IEE Proceedings - Computers and Digital Techniques,* v.150 n.2, pp.87-96, 2003.
[21] R. Buyya, D. Abramson, J. Giddy,"Nimrod/G: An Architecture for a Resource Management and Scheduling System in a Global Computational Grid. Proceeding of the HPC ASIA'2000", *the 4th International Conference on High Performance Computing in Asia- Pacific Region, Beijing, China, IEEE Computer Society Press,* USA, 2000