International Journal of Engineering Research and Development
e-ISSN: 2278-067X, **p**-ISSN: 2278-800X, www.ijerd.com
Volume 6, Issue 11 (April 2013), PP. 47-54

A Novel Approach Inresource Management and Job Scheduling For Computational Grids

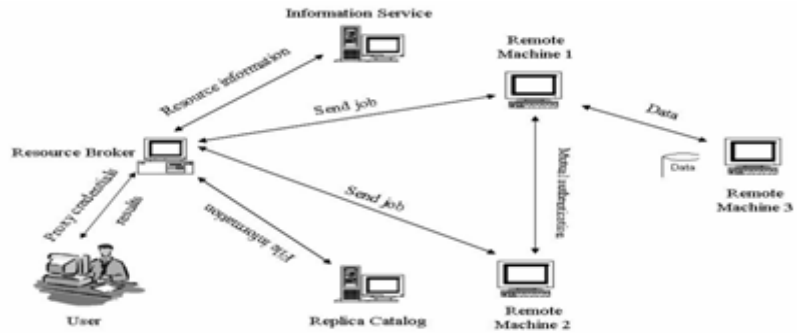
V.DayaSagar Ketaraju[1], Dr.M.V.L.N.Raja Rao[2], Dr.G.V.S.N.R.V.Prasad3
[1]Associate Professor Dept. of computer science and Engineering atNalanda Institute of Engineering & Technology,Siddartha Nagar,Kantipudi, Sattenapalli, Guntur Dist,A.P.India.
[2]Professor & Head Dept. of Information Technology at Gudlavalleru Engineering College,Gudlavalleru, Krishna Dist., A.P. India.
[3]Professor & Head Dept. of Computer Science & Engineering at Gudlavalleru Engineering College,Gudlavalleru, Krishna Dist., A.P. India.

Abstract:- Grid and P2P computing environments, the resources are usually geographically distributed in *multiple administrative domains*, managed and owned by different organizations with different policies, and interconnected by wide-area networks or the Internet. This paper introduces a novel approach in resource management and Job scheduling for computational grids we proposed a several algorithms for scheduling jobs in effective time consuming manner. The resource management and scheduling systems for Grid computing need to manage resources and job execution depending on either resource consumers' or resource provider's requirements, and dynamically adapt to changes in resource availability. The management of resources and scheduling of jobs in such large-scale distributed systems is a complex undertaking. In order to prove the effectiveness of resource brokers and associated scheduling algorithms, their performance needs to be evaluated under different scenarios such as varying number of resources and users with differentrequirements. In a Grid environment, it is hard and even impossibleto perform scheduler performance evaluation in a *repeatable* and *controllable* manner as resources andusers are distributed across multiple organizations with their own policies. To overcome this limitation, weareusing the *Grid Simulator*. This toolkit supportsmodelling and simulation of large volume of Grid resources,andapplication models. It provides primitives for creation of application tasks, mapping of tasks to resources,and their management. We have used the *Grid Simulator* to create a resource broker that simulates Nimrod-G for design and evaluation of deadline and budget constrained job scheduling algorithms with cost and time optimizations.

Keywords-Grid Computing, Grid Simulator, Job Scheduling Algorithms

I. INTRODUCTION

Internet having a powerful computers and high-speed networks as low-cost commodity components are changing the way we do large-scale parallel and distributed computing. The interest in coupling geographically distributed (computational) resources is also growing for solving large-scale problems, leading to what is popularly called the Grid [1] and peer-to-peer (P2P) computing [2] networks. These enable sharing, selection and aggregation of suitable computational and data resources for solving large-scale data intensive problems in science, engineering, and commerce. The working of the Grid computing environment is shown in Figure 1. The Grid consists of four layers of components: fabric, core middleware, user-level middleware, and applications [3].



The services they provide include security and access management, remote job submission, storage, and resource information. The user-level middleware provides higher-level tools such as resource providers, application development and adaptive runtime environment. The user essentially interacts with a resourceprovider that hides the complexities of Grid computing[4, 5]. The resource provider discovers resources that the user can access using information services, negotiatesfor access costs using trading services, maps tasks to resources (scheduling), starts job execution, and finally gathers the results. It is alsoresponsible for monitoring and tracking application execution progress along with adapting to thechanges in Grid runtime environment conditions and resource failures.

Apart from the centralized approach, two other approaches that are usedin distributed resource management are: *hierarchical* and *decentralized* scheduling or a combinationof them [6]. A Grid resource provider, called Nimrod-G [5], has been developed that performs schedulingof parameter sweep, task-farming applications on geographically distributed resources. It supportsdeadline and budget-based scheduling driven by market-based economic models.

The GridSimulator supports modelling and simulation of a wide range of heterogeneous resources,such as single or multiprocessors, shared and distributed memory machines such as PCs, workstations,SMPs, and clusters with different capabilities and configurations. It can be used for modelling andsimulation of application scheduling on various classes of parallel and distributed computing systemssuch as clusters [11], Grids [1], and P2P networks [2].

II. RELATED WORK.

Simulation include simulation languages (e.g. Simscript [12]), simulation environments (e.g. Parsec [13]), simulation libraries (SimJava [14]), and application specific simulators (e.g. OMNet++ network simulator [15]). While a large body of knowledge and tools exists, there are very few tools available for application scheduling simulation in Grid computing environments. The notable ones are: Bricks [16], MicroGrid [17], SimGrid [18], and our Grid Simulator.The Bricks simulation system [16], developed at the Tokyo Institute of Technology in Japan, helps in simulating client-server like global computing systems that provide remote access to scientific libraries and packages running on high-performance computers. It follows centralized global scheduling methodology as opposed to our work in which each application scheduling is managed by the users' own resource provider.

The MicroGrid emulator [17], undertaken at the University of California at San Diego (UCSD), is modeled after Globus [19]. It allows execution of applications constructed using the Globus toolkit in a controlled virtual Grid emulated environment.

The SimGrid toolkit [18], developed at UCSD, is a C language based toolkit for the simulation of application scheduling. It supports modelling of resources that are *time-shared* and the load can be injected as constants or from real traces. It is a powerful system that allows creation of tasks in terms of their execution time and resources with respect to a standard machine capability. Using SimGrid APIs, tasks can be assigned to resources depending on the scheduling policy being simulated. Hence, our GridSimulatorextends the ideas in existing systems and overcomes their limitations accordingly.Finally, we have chosen to implement GridSimulator in Java by leveraging SimJava's [14] basic discrete event simulation infrastructure.

III. PROPOSED SYSTEM

GridSimulator provides a comprehensive facility for simulation of different classes ofheterogeneous resources, users, applications, resource providers, and schedulers. It can be used to simulate application schedulers for single or multiple administrative domains distributed computing systems such as clusters and Grids. Application schedulers in the Grid environment, called resource providers, perform resource discovery, selection, and aggregation of a diverse set of distributed resources for an individual user. In contrast, schedulers, managing resources such as clusters in a single administrative domain, have complete control over the policy used for allocation of resources. This means that all users need to submit their jobs to the *central*scheduler, which can be targeted to perform global optimization such as higher system utilization and overall user satisfaction depending on resource allocation policy or optimize for high priority users.

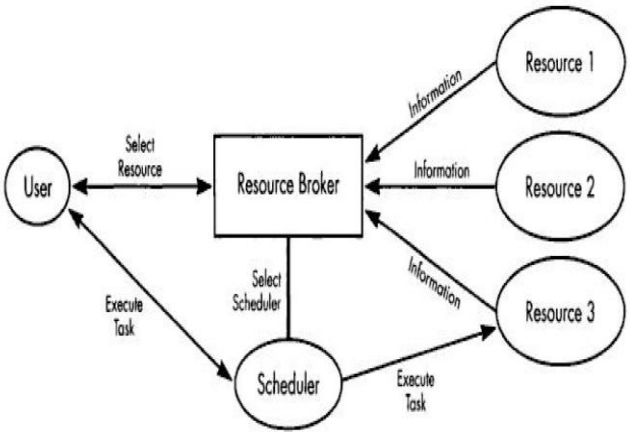


Fig2: Grid Simulator simulates different classes of heterogeneous resources

IV. IMPLEMENTATION

4.1Resource modelling and Scheduling

In the GridSimulator, we can create Processing Jobs (PJs) with different speeds (measuredin MIPS). Then, one or more PJs can be put together to create amachine. Similarly, one or more machines can be put together to create a Grid resource. Thus,the resulting Grid resource can be a single processor, shared memory multiprocessors (SMP), or adistributed memory cluster of computers. These Grid resources can simulate time- or space-sharedscheduling depending on the allocation policy. The space-shared systems use resource allocation policies such asfirst-come-first-served (FCFS), back filling, shortest-job-first-served (SJFS), and so on. The GridSimulator resources can send, receive, or scheduleevents to simulate the execution of jobs. A resource consists of two shared or distributed memoryPJs each with a MIPS rating of 1, for simplicity. FourJobs that represent jobs with processingrequirements equivalent to 10, 7.5, 8.5, and 9.5 MI (million of instructions) arrive in ST(simulation times)0, 4, 7,and 10 respectively. The way GridSimulator schedules jobs to PJs is shown schematically in Figure 3fortime-shared resources and Figure 4for space-shared resources.

4.1.1. Simulation of scheduling in time-shared resources

The Grid resource simulator uses internal events to simulate the execution and allocation of PJs'share to Grid jobs. When jobs arrive, time-shared systems start their execution immediately andshare resources among all jobs. Whenever a new Grid job arrives, we update the processing time ofexisting Gridlets and then add this newly arrived job to the execution set.We schedule an internal eventto be delivered at the earliest completion time of the smallest job in the execution set. It then waits forthe arrival of events.A complete algorithm for simulation of time-share scheduling and execution is shown in Figure 5.

If a newly arrived event happens to be an internal event whose tag number is the same as the mostrecently scheduled event, then it is recognized as a job completion event. Depending on the numberof Gridlets in execution and the number of PEs in a resource, GridSimulator allocates the appropriate PEshare to all Gridlets for the event duration using the algorithm shown in Figure 6. It should be notedthat Gridlet's sharing the same PE would get an equal amount of PE share. The completed Gridlet issent back to its originator (broker or user) and removed from the execution set. GridSimulator schedulesa new internal event to be delivered at the forecasted earliest completion time of the remainingGridlets.Figure 7illustrates the simulation of the time-share scheduling algorithmand the Gridlets' execution.

When Gridlet1 arrives at time 0, it is mapped to PE1 and an internal event to be delivered at time 10 isscheduled since the predicted completion time is still 10. At time 4, Gridlet2 arrives and it is mappedto PE2. The completion time of Gridlet2 is predicted as 12.5 and the completion time of Gridlet1 isstill 10 since both of them are executing on different PEs. A new internal event is scheduled, whichwill still be delivered at time 10. At time 7, Gridlet3 arrives, which is mapped to PE2. It shares the PEtime with Gridlet2. At time 10, an internal event is delivered to the resource to signify thecompletionof Gridlet1, which is then sent back to the broker. At this moment, as the number of Gridlets is equalsthe number of PEs, they are mapped to different PEs.

Algorithm: Time-Shared Grid Resource Event Handler()

1. Wait for an event
2. If the external and Gridlet arrival event, then:
BEGIN /*a new job arrived*/

a. Allocate PE Share for Gridlets Processed so far
b. Add arrived Gridlet to Execution Set
c. Forecast completion time of all Gridlets in
Execution Set
d. Schedule an event to be delivered at the smallest
completion time
END
3. If event is internal and its tag value is the same as the
recently scheduled internal event tag,
BEGIN /*a job finish event*/
a. Allocate PE Share for Gridlets Processed so far
b. Update finished Gridlet's PE and Wall clock time
parameters and send it back to the
Resource Provider
c. Remove finished Gridlet from the Execution Set
and add to Finished Set
d. Forecast completion time of all Gridlets in
Execution Set
e. Schedule an event to be delivered at the smallest
completion time
END
4. Repeat the above steps until the end of simulation
event is received

Figure 5. An event handler for simulating time-shared resource scheduling.

Algorithm: PE Share Allocation(Duration)

BEGIN
1.Identify total MI per PE for the duration and the
number of PEs that process one extra
GridletTotalMIperPE = MIPSRatingOfOnePE()*Duration
MinNoOfGridletsPerPE = NoOfGridletsInExec/ NoOfPEs
NoofPEsRunningOneExtraGridlet= NoOfGridletsInExec %
NoOfPEs
2. Identify maximum and minimum MI share that
Gridlet get in the Duration
If(NoOfGridletsInExec<= NoOfPEs), then:
MaxSharePerGridlet = MinSharePerGridlet = TotalMIperPE
MaxShareNoOfGridlets = NoOfGridletsInExec
else /* NoOfGridletsInExec>NoOfPEs */
MaxSharePerGridlet = TotalMIperPE/MinNoOfGridletsPerPE
MinSharePerGridlet =
TotalMIperPE/(MinNoOfGridletsPerPE+1)
MaxShareNoOfGridlets = (NoOfPEs–
NoOfPEsRunningOneExtraGridlet)* MinNoOfGridletsPerPE
END

Figure 6. PE share allocation to Gridlet in time-shared GridSimulator resource.

4.1.2. Simulation of scheduling in Space-shared resources

The Grid resource simulator uses internal events to simulate the execution and allocation of PEs to Grid let jobs. When a job arrives, space-sharedsystems start its execution immediately if there is a free PE available,otherwise, it is queued. During the Gridletassignment, job-processing time is determined and the event is scheduled for delivery at the end of the execution time. Whenever a Gridlet job finishes, an internal event is delivered to signify the completion of the scheduled Gridlet job. The resource simulator then frees the PE allocated to it and checks if there are any other jobs waiting in the queue. If there are jobs waiting in the queue, then it selects a suitable job depending on the policy and assigns it to the PE which is free. A complete algorithm for simulation of space-share scheduling and execution is shown in Figure 7.

If a newly arrived event happens to be an internal event whose tag number is the same as the most recently scheduled event, then it is recognized as a Gridlet completion event. If there are Gridlets in the submission queue, then depending on the allocation policy (e.g. the first Gridlet in the queue if FCFS policy is

used), Grid Simulator selects a suitable Gridlet from the queue and assigns it to the PE or a suitable PE if more than one PE is free. See Figure 8 for an illustration of the allocation of PEs to Gridlets. The completed Gridlet is sent back to its originator (broker or user) and removed from the execution set. GridSimulator schedules a new internal event to be delivered at the completion time of the scheduled Gridlet job. Figure 9illustrates simulation of the space-shared scheduling algorithm and Gridlet execution.

When Gridlet1 arrives at time 0, it is mapped to PE1 and an internal event to be delivered at time 10 is scheduled since the predicted completion time is still 10. At time 4, Gridlet2 arrives and it is mapped to PE2. The completion time of Gridlet2 is predicted as 12.5 and the completion time of Gridlet1 is still 10 since both of them are executing on different PEs. A new internal event to be delivered at time 12.5 is scheduled to signify the completion of Gridlet2. At time 7, Gridlet3 arrives. Since there is no free PE available on the resource, it is put into the queue.

The simulation continues, i.e. the Grid Simulator resource waits for the arrival of a new event. At time 10 a new event is delivered which happens to signify thecompletion of Gridlet1, which is then sent back to the broker. It then checks to see if there are any Grid lets waiting in the queue and chooses a suitable Grid let (in this case Gridlet2, based on FCFS policy) and assigns the available PE to it. An internal event to be delivered at time 19.5 is scheduled to indicate the completion time of Gridlet3 and then waits for the arrival of new events. A new event is delivered at the simulation time 12.5, which signifies the completion of Gridlet2, which is then sent back to the broker. There is no Grid let waiting in the queue, so it proceeds without scheduling any events and waits for the arrival of the next event.

A new internal event arrives at the simulation time 19.5, which signifies the completion of Gridlet3.This process continues until resources receive an external event indicating the termination of simulation. A schematic representation of the arrivalof the Grid lets, internal events delivery, and sending them back to the broker is shown in Figure 6. A detailed statistical data on the arrival, execution start, finish, and elapsed time of all Gridlets are shown in Table I. For every Grid resource, the non-Grid (local) workload is estimated based on typically observed load conditions depending on the time zone of the resource

Algorithm: Space-Shared Grid Resource Event Handler()

```
1. Wait for an event and Identify Type of Event
     received
2. If it external and Gridlet arrival event, then
BEGIN
/* a new job arrived */
If( number of Gridlets in execution< number of
PEs in the resource)then
Allocate PE to the Gridlet()
      {
/* It should schedule an Gridlet completion event */
        }
else
Add Gridlet to the Gridlet Submitted Queue
END
3. If event is internal and its tag value is the same
     recently scheduled internal event tag,
BEGIN
/* a job finish event */
      • Update finished Gridlet's PE and Wall clock
      time parameters and send it back to the
      Resource provider
      • Set the status of PE to FREE.
      • Remove finished Gridlet from the Execution
      Set and add to the Finished Set.
      • If Gridlet Submitted Queue has Gridlets in
      waiting, then
Choose the Gridlet to be Processed()
/* e.g., first one in Q if FCFS policy is used */
Allocate PE to the Gridlet()
/* It should schedule a Gridlet completion event */
END
 4. Repeat above steps until end of simulation event is
 received
```

Figure 7. An event handler for simulating space-shared resource scheduling

Algorithm: Allocate PE to the Gridlet(Gridletgl)
BEGIN
1. Identify a suitable Machine with Free PE
2. Identify a suitable PE in the machine and Assign to
the Gridlet

3. Set Status of the Allocated PE to BUSY
4. Determine the Completion Time of Gridlet and Set
and internal event to be delivered at the completion
time
END

Figure 8.PE allocation to the Gridlets in the space-shared GridSim resource.

V. SCHEDULING SIMULATION EXPERIMENTS

To simulate application scheduling in GridSimulator environment using the economic Grid broker requiresthe modeling and creation of GridSimulator resources and applications that model jobs as Gridlets. In thissection, we present resource and application modeling along with the results of experiments withquality of services driven application processing.

5.1. Resource Modelling

We modelled and simulated a number of time- and space-shared resources with different characteristics,configurations, and capabilities from those in the WWG testbed. We have selected the latest CPUmodels IntelXEIONServer, Sun Netra 20, Intel VC820 (800EB MHz, i5 third generation), and SGI Origin3200 1X 500 MHz R14k released by their manufacturers Compaq, Sun, Intel, and SGI, respectively.To enable the users to model their applicationprocessing requirements, we assumed the MIPS rating of the PEs to be the same as the SPEC rating.

Table II shows characteristics of resources simulated and their PE cost per time unit in G$ (Griddollar). These simulated resources resemble the WWG testbed resources used in processing parameter sweep application using the Nimrod-G broker [24]. The PE cost in G$/unit time does not necessarily reflect the cost of processing when PEs have different capabilities. The brokers need to translate the cost into G$ per MI (million instructions) for each resource. Such translation helps in identifying the relative cost of resources for processing Gridlets on them.

Table II. WWG testbed resources simulated using GridSimulator

Resource Name	Characteristics of simulated resourceVendor, resource type, node OS,no.of Pes	Eqivalet Resources in WWG (host name,location)	A PE MIPS ratin g	Type of Resource Manager	Price (G$ PE/time unit)	MIPS per G$
R0	Compaq, AlphaServer,CPU, OSF1, 4	grendel.vpac.org, VPAC, Melbourne, Australia	515	Time-shared	8	64.37
R1	Sun, Ultra, Solaris, 4	hpc420.hpcc.jp,AIST, Tokyo, Japan	377	Time-shared	4	94.25
R2	Sun, Ultra, Solaris, 4	hpc420.hpcc.jp,AIST, Tokyo, Japan	377	Time-shared	3	125.66
R3	Sun, Ultra, Solaris, 4	hpc420.hpcc.jp,AIST, Tokyo, Japan	377	Time-shared	3	125.66
R4	Intel, Pentium/VC820,Linux, 2	barbera.cnuce.cnr.it,CNR, Pisa, Italy	380	Time-shared	2	190
R5	SGI, Origin 3200,IRIX, 6	onyx1.zib.de,ZIB, Berlin, Germany	410	Time-shared	5	82
R6	SGI, Origin 3200,IRIX, 6	onyx1.zib.de,ZIB, Berlin, Germany		Time-shared	5	82
R7	SGI, Origin 3200,IRIX, 6	mat.ruk.cuni.cz,Charles U., PragueCzech Republic	410	space-shared	4	102.5
R8	Intel, Pentium/VC820,Linux, 2	marge.csm.port.ac.uk,Portsmouth, U.K.	380	Time-shared	1	380
R9	SGI, Origin 3200,IRIX, 6	green.cfs.ac.uk,Manchester, U.K.	410	Time-shared	6	68.33
R10	Sun, Ultra, Solaris, 8	pitcairn.mcs.anl.gov,ANL, Chicago, U.S.A.	377	Time-shared	3	125.66

Table II. Simulate Grid Resources using Grid Simulator.

5.2. Application modelling

We have modeled a task farming application that consists of 100 jobs. In Grid Simulator, these jobs are packaged as Gridlets whose contents include the job length in MI, the size of the job input and output data in bytes, along with various other execution related parameters when they move between the broker and resources. The job length is expressed in terms of the time it takes to run on a standard resource PE with a SPEC/MIPS rating of 100. Gridlets processing time is expressed in such a way that they are expected to take at least 100 time units with a random variation of 0–10% on the positive sideof the standard resource. That means the Gridlet job length (processing requirements) can be at least10 000 MI with a random variation of 0–10% on the positive side. This 0–10% random variation in theGridlet job length is introduced to modelheterogeneous tasks similar to those present in the real worldparameter sweep applications

5.3. DBC scheduling experiments with cost-optimization—for a single user

In this experiment, we performed scheduling experiments with different values of DBCs for a single user. The deadline is varied in simulation time from 100 to 3600 in steps of 500. The budget is varied from G$ 5000 to 22 000 in steps of 1000. For this scenario, we performed scheduling simulation for the DBC *cost-optimization* algorithm. The number of Gridlets processed, the deadline utilized, and the budget spent for different scheduling scenario is shown in Figures 21–24.

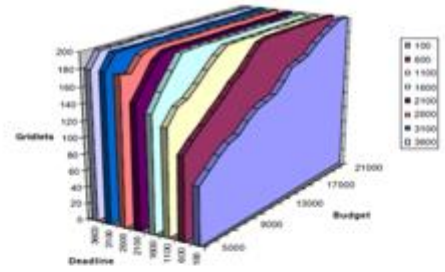


Figure 21. Number of Gridlets processed for different budget limits with a fixed deadline for each.

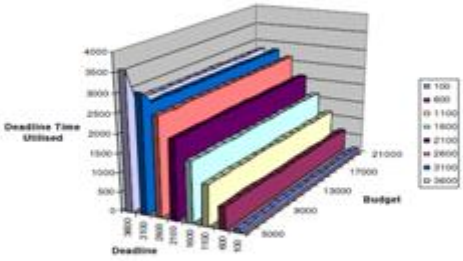


Figure 23. Deadline time utilized for processing Gridlets for different values of deadline and budget.

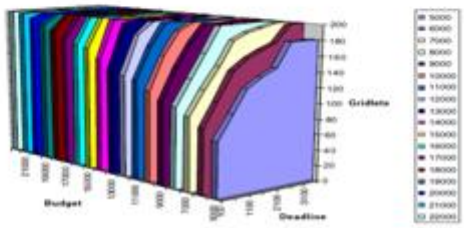


Figure 22. Number of Gridlets processed for different deadline limits with a fixed budget for each.

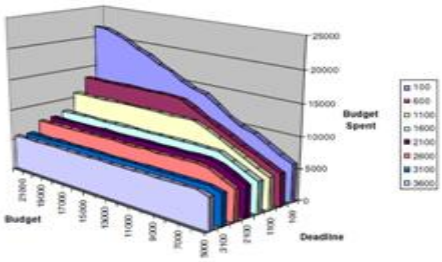


Figure 24. Budget spent for processing Gridlets for different values of deadline and budget.

VI. CONCLUSION AND FUTUREWORK

We discussed an object-oriented toolkit, called Grid Simulator, for resource modelling and scheduling simulation. Grid Simulator simulates time- and space-shared resources with different capabilities, time zones, and configurations. It supports different application models that can be mapped to resources for execution by developing simulated application schedulers. Grid Simulator scales with them due to its concurrent implementation. We have developed a Nimrod-G like economic Grid resource broker simulator using GridSimulatorand evaluated a number of scheduling algorithms based on deadline and budget based constraints. This helped us in evaluating performance and scalability of our scheduling policies with differentGrid configurations such as varying the number of resources, capability, cost, users, and processing

REFERENCES

1. Foster I, Kesselman C (eds.). *The Grid: Blueprint for a Future Computing Infrastructure*. Morgan Kaufmann: San Mateo, CA, 1999.
2. Oram A (ed.). *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. O'Reilly, 2001.
3. Baker M, Buyya R, Laforenza D. The Grid: International efforts in global computing. *Proceedings of the International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet*, Rome, Italy,31 July–6 August 2000.
4. Abramson D, Giddy J, Kotler L. High performance parametric modeling with Nimrod/G: Killer application for the global Grid? *Proceedings International Parallel and Distributed Processing Symposium (IPDPS 2000)*, Cancun, Mexico, 1–5 May 2000. IEEE Computer Society Press, 2000.
5. Buyya R, Abramson D, Giddy J. Nimrod/G: An architecture for a resource management and scheduling system in a global computational Grid. *Proceedings 4th International Conference and Exhibition on High Performance Computing in Asia-Pacific Region (HPC ASIA 2000)*, Beijing, China, 14–17 May 2000. IEEE Computer Society Press, 2000.
6. Buyya R, Abramson D, Giddy J. An economy driven resource management architecture for global computational power Grids. *Proceedings of the 2000 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2000)*, Las Vegas, NV, 26–29 June 2000. CSREA Press, 2000.
7. Buyya R, Giddy J, Abramson D. An evaluation of economy-based resource trading and scheduling on computational power Grids for parameter sweep applications. *Proceedings of the 2nd International Workshop on Active Middleware Services (AMS 2000)*, Pittsburgh, PA, 1 August 2000. Kluwer Academic Press, 2000

Authors

V.DayaSagarKetaraju received , the M.Tech. in Computer Science and Engineering from Javaharlal Nehru Technological University, Kakinada, AndhraPradesh, India in 2010. Presently working as associate professor in Dept. of Computer Science and Engineering at Nalanda Institute of Engineering & Technology, Kantipudi, sattenapalli, Guntur, Andhra Pradsesh, INDIA. His current research area is Database management systems, Grid Computing, computer networks

Dr.M.V.L.N.RajaRao received the Ph.D in Computer SciencefromAndhraUniversity, Visakapatnam, Andhrapradesh, India in2007. Presently working as Professor &Head in the Dept.of Information Technology, Gudlavalleru Engg. College, Gudlavalleru, Krishna Dist ,Andhrapradesh,India. His current researcharea is Dataware house and datamining, computer networks, Grid computing, software Engineering & testing.

Dr.G.V.S.N.R.V.Prasad receivedPh.D in Computer Science & Engineering from Javaharlal Nehru Technological University, Kakinada,Andhra Pradesh, Indian in2013. Presently working as Professor & Head in the Dept. of Computer Science & Engineering at Gudlavalleru Engineering College,Gudlavalleru, Krishna Dist., A.P. India.His current research area is Dataware house anddata mining, computer networks, Grid computing, software Engineering & testing and Intrusion detection techniques