Resource Discovery in Grid Computing

Ch E V T V Laxmi Research Scholar (Karpagam University), Associate Professor, Department of Computer Science, Raghu Engineering Collge, Visakhapatnam,Andhra Pradesh. email:elaxmi2002@yahoo.com.
K.Somasundaram,Professor, Department of Computer Science and Engineering,Jaya Engineering College, CTH Road, Prakash Nagar,Thiruninravur,Thiruvallur - Dist, Tamilnadu.

Abstract-This paper identifies the issues in resource management and scheduling in the emerging grid computing context and briefly discusses techniques for scheduling using computational economy concept. In the context of Grid scheduling, it involves four main stages: resource discovery, resource selection, schedule generation and job execution. This article provides a brief overview on working of scheduling in the grid computing and its resource management, important factors considered in resource management, comparison of different resource discovery mechanisms and future outlook of resource discovery in the grid environment.

Index Terms- Daemons, Grid Computing, Resource discovery, Resource Management, Resource Scheduling, Scheduler,

1 INTRODUCTION

Grid computing is an integrated computer network linking large geographically distributed and heterogeneous computer systems and resources, which eliminates the need for dedicated servers for job computations but uses distributive resources collectively to enhance computational power. It is a type of distributed computing different from other types like cloud computing and cluster computing. In this case, the resources are scattered over a wide span of area belonging to different individuals or companies, which are used to solve different segments of one single problem whose solution is gotten only after combining all the individual solutions of the segments computed by the different allocated resources. Instead of using dedicated resources from only one individual/organization for computational purposes, grid provides a platform that enables resources from different organizations to be shared so as to enable problems to be solved according to the demand of the users. This helps in reducing costs, increasing the computational speed, efficiency, flexibility, scalability and performance. Examples of some resources that can be shared in grid networks are storage devices, operating systems, communication devices, bandwidth, simulation, software's etc. To handle the different resource tasks such as allocation, assignment, authentication, authorization etc. efficiently there is need for a good resource management planning.

2 RELATED WORK

Grid scheduling involves four main stages: resource discovery, resource selection, schedule generation and job execution.

2.1 Resource discovery

The goal of resource discovery is to identify a list of authenticated resources that are available for job submission. In order to cope with the dynamic nature of the Grid, a scheduler needs to have some way of incorporating dynamic state information about the available resources into its decision-making process.

A Grid a scheduler should always know what resources it can access, how busy they are, how long it takes to communicate with them and how long it takes for them to communicate with each other. With this information, the scheduler optimizes the scheduling of jobs to make more efficient and effective use of the available resources. A Grid environment typically uses a pull model, a push model or a push–pull model for resource discovery. The outcome of the resource discovery process is the identity of resources available (Ravailable) in a Grid environment for job submission and execution.

2.2 The pull model

In this model, a single daemon associated with the scheduler can query Grid resources and collect state information such as CPU loads or the

available memory. The pull model for gathering resource information incurs relatively small communication overhead, but unless it requests resource information frequently, it tends to provide fairly stale information which is likely to be constantly out-of-date, and potentially misleading. Figure 1.1 shows the architecture of the model.

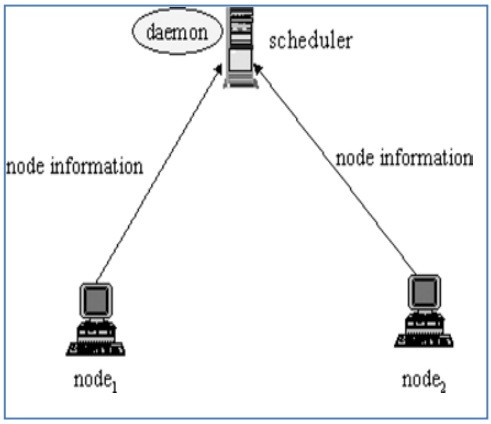

FIGURE 1.1 THE PULL MODEL FOR RESOURCE DISCOVERY

2.3The push model

In this model, each resource in the environment has a daemon for gathering local state information, which will be sent to a centralized scheduler that maintains a database to record each resource's activity. If the updates are frequent, an accurate view of the system state can be maintained over time; obviously, frequent updates to the database are intrusive and consume network bandwidth. Figure 1.2 shows the architecture of the push model.

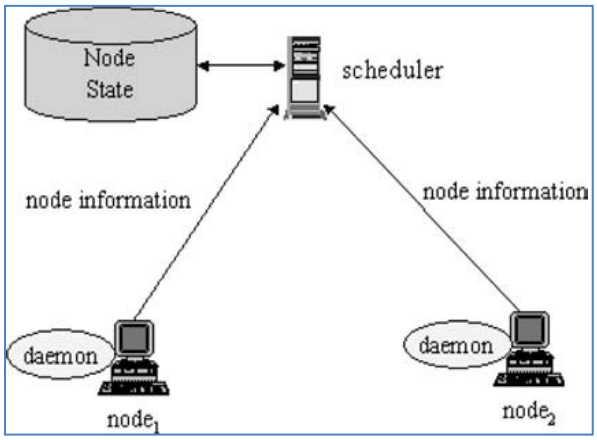

FIGURE 1.2 THE PUSH MODEL FOR RESOURCE DISCOVERY

2.4 The push–pull model

The push–pull model lies somewhere between the pull model and the push model. Each resource in the environment runs a daemon that collects state information. Instead of directly sending this information to a central scheduler, there exist some intermediate nodes running daemons that aggregate state information from different sub-resources that respond to queries from the scheduler. A challenge of this model is to find out what information is most useful, how often it should be collected and how long this information should be kept around. Figure 1.3 shows the architecture of the push–pull model.

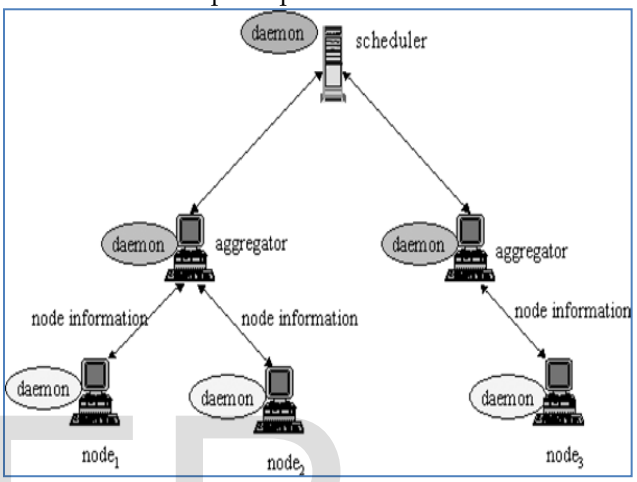

FIGURE 1.3 THE PUSH–PULL MODEL FOR RESOURCE DISCOVERY

3 PROPOSED APPROACH

The push–pull model lies somewhere between the pull and push model. In this each environment runs a daemon that collects state information, and this information is not directly forwarded to central scheduler, there exist intermediate nodes which are running daemons, that aggregate state information from different sub resources. In the push-pull model, the state of information which is collected and it is not stored therefore, whenever the resource information are needed, a daemon has to run to gather the information to find the state of resources. Therefore whenever we need the state of resource information we have to run daemon which is wastage of time and waste of memory because of running daemons. If the collected state of resource information is stored persistently in the database, the time to search the state of resource information will be reduced as well as wastage of memory comes down as the daemons are not running frequently. Figure 1.4 shows the

architecture of the push-pull model with node state database. The maintenance of resource information in the database can be implemented by using any database such as MYSQL, ORACLE even it can be implemented by using XML and XML DOM and SAX parsers.

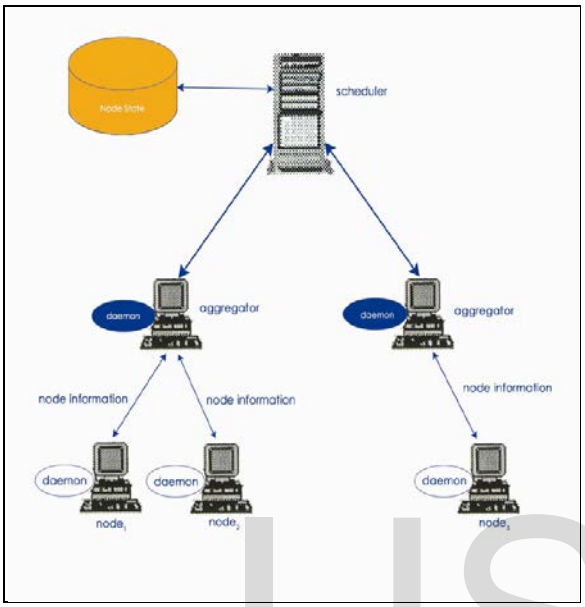

FIGURE 1.4 THE ARCHITECTURE OF THE PUSH-PULL MODEL WITH NODE STATE DATABASE.

4 MAINTENANCE OF RESOURCE INFORMATION

In the proposed architecture of the push-pull model with node state database. The collected node states are stored in a database, which are collected by running daemons on each nodes, which will be send to a centralized scheduler and it is connected to the database where all the node states are stored. The proposed architecture of the push-pull model with node state database can be maintained by the web database i.e., by using XML. The following XML structure shows the maintenance of the resource information as web database:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<node_states>
        <node>
        <node_id>1</node_id>
        <Resources>
                <cpu>
        <cpu_speed>1.8GHz</cpu_speed>
        <cpu_load>50%</cpu_load>
                </cpu>
                <memory>
        <ram_size>256MB</ram_size>
        <ram_usage>50%</ram_usage>
                </memory>
                <devices>
<Input_devices>Keyboard</Input_devices>
<output_devices>printers</output_devices>
                </devices>
                </Resources>
        </node>
        <node>
        <node_id>2</node_id>
        <Resources>
                <cpu>
        <cpu_speed>2.6GHz</cpu_speed>
        <cpu_load>70%</cpu_load>
                </cpu>
                <memory>
        <ram_size>512MB</ram_size>
        <ram_usage>50%</ram_usage>
                </memory><devices>
        <Input_devices>Pen
Input</Input_devices>
        <output_devices>Computer Output
Microfilm (COM)</output_devices>
                </devices>
        </Resources></node>
        <node>
        <node_id>3</node_id>
        <Resources><cpu>
        <cpu_speed>1.2GHz</cpu_speed>
        <cpu_load>40%</cpu_load>
                </cpu> <memory>
        <ram_size>512MB</ram_size>
        <ram_usage>30%</ram_usage>
                </memory>
        <devices>
<Input_devices>Barcode reader</Input_devices>
<output_devices>Projector</output_devices>
                </devices>
        </Resources>
        </node>
</node_states>
```

5 RESOURCE SCHEDULING

The grid resource scheduling process can be defined as the process of matching a query for resources, described in terms of required characteristics, to a set of resources that meet the expressed requirements. To make information available to users quickly and reliably, an effective and efficient resource scheduling mechanism is

crucial. Generally grid resources are potentially very large in number with various individual resources that are not centrally controlled. These resources can enter as well as leave the grid systems at any time. For these reasons resource scheduling in large-scale grids can be very challenging.

5.1 Research on Novel Dynamic Resource Management and job scheduling in grid computing (RNDRM).

This scheduling model is based on Heap Sort Tree (HST) for computing the available computational power of the nodes (resource) as well as whole grid system. Here the resource with largest available computational ability among the whole grid system is selected to be the root node of the HST and it is ready for the scheduler to submit a job. The algorithm design for job scheduling is well suitable for the complex grids environment and it is based on agents.

5.2 Experimental Analysis

In the proposed architecture of the push-pull model with node state database. The collected node states are stored in a database, i.e., by using XML. Now we give an example to explain the Research on NovelDynamic Resource Management and job scheduling in grid computing (RNDRM) to choose one resource from three possible candidates. The assumed parameters associated with each resources are stored in web database which are represented as table in table 1. The resource selection process is used to choose the resources from the resource list (Rselected) for a given job. Since all resources in the list Rselected could meet the minimum requirements imposed by the job, we implemented Research on Novel Dynamic Resource Management and job scheduling in grid computing (RNDRM) to choose the best resources to execute the job.

Node State	
Node	
Node_	Resources

id	CPU		Memory		devices	
1	Cpu speed (GHz)	CPU Load (%)	Ram Size (MB	Ram Usage (%)	Input Devices	Output Devices
	1.8	50	256	50	Keyboard	Printers
2	2.6	70	512	60	Pen Input	Computer Output Microfilm (COM)
	1.2	40	512	30	Barcode reader	Projector

TABLE 1. THE RESOURCE INFORMATION

The resource selection algorithm should take into account the current state of resources and choose the best one based on a quantitative evaluation. A resource selection algorithm that only takes CPU and RAM into account could be designed as follows:

$$Evaluation_{resource}=\frac{Evaluation_{CPU}+Evaluation_{RAM}}{W_{CPU}+W_{RAM}}$$

$$Evaluation_{CPU}=W_{CPU}(1-CPU_{load})*\frac{CPU_{speed}}{CPU_{min}}$$

$$Evaluation_{RAM}=W_{RAM}(1-RAM_{usage})*\frac{RAM_{SIZE}}{RAM_{MIN}}$$

Where

W_{CPU}- the allocated to CPU speed;
CPU_{load}-current CPU load;
CPU_{speed}-real CPU speed;
CPU_{min}-minimum CPU speed;
W_{RAM}-the weight allocated to RAM;
RAM_{usage}-current RAMusage;
RAM_{size}-original RAM size;
RAM_{min}-minimum RAM size.

Now we give an example to explain the algorithm used to choose one resource from three possible candidates. The assumed parameters associated with each resource are given in Table 2.

CPU speed (GHz)	CPU load(%)	RAM size(MB)	RAM usage(%)	CPU speed (GHz)
Resource1	1.8	50	256	50
Resource2	2.6	70	512	60
Resource3	1.2	40	512	30

TABLE 2. THE RESOURCE ALLOCATION TABLE

Let us suppose that the total weighting used in the algorithm is 10, where the CPU weight is 6 and the RAM weight is 4. The minimum CPU speed is 1 GHz and minimum RAM size is 256 MB.

Then, evaluation values for resources can be calculated using the three formulas:
$Evaluation_{resource1} = (5.4+2)/10=0.74$
$Evaluation_{resource2} = (4.68+3.2)/10=0.788$
$Evaluation_{resource3} = (4.32+5.6)/10=0.992$
From the results we know Resource3 is the best choice for the submitted job.

6 CONCLUSION

Computational Grids are quickly rising as a practical means by which to execute new science and develop new applications. The effective and efficient exploitation of Grid computing facilities needs highly advanced and protected resource management systems. Efficient resource sharing and accessing cannot go without the assurance of high trustworthiness. In this paper we have discussed three different models for grid resource discovery architecture inspired by three different philosophies. Grid environment typically uses a pull model, a push model or a push-pull model for resource discovery. The outcome of the resource discovery is availability of the resources in grid environment for job submission and execution. In this paper, we have proposed push-pull model with node state database where resource information are stored in the database. The proposed approach stores the node states, which are collected by running the deamons, in the database.

7 References

[1] *Ian Foster and Carl Kesselman, "The Grid: Blueprint for a New Computing Infrastructure," Elsevier Inc., Singapore, Second Edition, 2004.*
[2] *Raksha Sharma, Vishnu Kant Soni, Manoj Kumar Mishra, Prachet Bhuyan A Survey of Job Scheduling and Resource Management in Grid Computing World Academy of Science, Engineering and Technology 40 2010*
[3] *Hamscher, V., Schwiegelshohn, U., Streit, A. and Yahyapour, R. Evaluation of Job-Scheduling Strategies for Grid Computing. GRID 2000, 191–202, 17–20 December 2000, Bangalore, India. Lecture Notes in Computer Science, Springer-Verlag.*
[4] *Srinivasan, S., Kettimuthu, R., Subramani, V. and Sadayappan, P. Characterization of Backfilling Strategies for Parallel Job Scheduling. ICPP Workshops 2002, 514–522, August 2002, Vancouver, BC, Canada. CS Press.*
[5] *DAGManager, http://www.cs.wisc.edu/condor/dagman/.*
[6] *Ghare, G. and Leutenegger, S. Improving Small Job Response Time for Opportunistic Scheduling. Proceedings of 8th International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2000), San Francisco, CA, USA. CS Press.*
[7] *Raman, R., Livny, M. and Solomon, M. Matchmaking: Distributed Resource Management for High Throughput Computing. Proceedings of the 7th IEEE International Symposium on High Performance Distributed Computing, July 1998, Chicago, IL, USA. CS Press.*
[8] *Enterprise Edition policy, http://www.sun.com/blueprints/0703/817-3179. pdf.*
[9] *N1GE 6 Scheduling, http://docs.sun.com/app/docs/doc/817-5678/ 6ml4alis7?a=view.*
[10] *PBS Pro, http://www.pbspro.com/.*
[11] *Figueira, M., Hayes, J., Obertelli, G., Schopf, J., Shao, G., Smallen, S. ,Spring, N., Su, A. and Zagorodnov, D. Adaptive Computing on the Grid Using AppLeS. IEEE Transactions on Parallel and Distributed Systems, 14(4): 369–382 (2003).*
[12] *NWS, http://nws.cs.ucsb.edu/.*
[13] *Dail, H., Berman, F. and Casanova, H. A Decoupled Scheduling Approach for Grid Application Development Environments. Journal of Parallel Distributed Computing, 63(5): 505–524 (2003).*
[14] *Abramson, D., Giddy, J. and Kotler, L. High Performance Parametric Modeling with Nimrod/G: Killer Application for the Global Grid? Proceedings of the International Parallel and Distributed Processing (IPDPS 2000), May 2000, Cancun, Mexico. CS Press.*
[15] *Gerasoulis, A. and Jiao, J. Rescheduling Support for Mapping Dynamic Scientific Computation onto Distributed Memory Multiprocessors. Proceedings of the Euro-Pa '97, August 1997, Passau, Germany. Lecture Notes in Computer Science, Springer-Verlag.*
[16] *Goux, Jean-Pierre, Kulkarni, Sanjeev, Yoder, Michael and Linderoth, Jeff. Master-Worker: An Enabling Framework for Applications on the Computational Grid. Cluster Computing, 4(1): 63–70 (2001).*
[17] *Cactus, http://www.cactuscode.org/. 300 GRID SCHEDULING AND RESOURCE MANAGEMENT*
[18] *Spooner, D., Jarvis, S., Cao, J., Saini, S. and Nudd, G. Local Grid Scheduling Techniques using Performance Prediction, IEE Proc. – Comp. Digit. Tech., 150(2): 87–96 (2003).*
[19] *Young, L., McGough, S., Newhouse, S. and Darlington, J. Scheduling Architecture and Algorithms within the ICENI Grid Middleware. Proceedings of the UK e-Science All Hands Meeting, September 2003, Nottingham, UK.*
[20] *YarKhan, A. and Dongarra, J. Experiments with Scheduling Using Simulated Annealing in a Grid Environment. Proceedings of the 3rd International Workshop on Grid Computing (GRID 2002), November 2002, Baltimore, MD, USA. CS Press.*