World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:2, No:9, 2008

# Grid-HPA: Predicting Resource Requirements of a Job in the Grid Computing Environment

M. Bohlouli, M. Analoui



***Abstract*—**For complete support of Quality of Service, it is better that environment itself predicts resource requirements of a job by using special methods in the Grid computing. The exact and correct prediction causes exact matching of required resources with available resources. After the execution of each job, the used resources will be saved in the active database named "History". At first some of the attributes will be exploit from the main job and according to a defined similarity algorithm the most similar executed job will be exploited from "History" using statistic terms such as linear regression or average, resource requirements will be predicted. The new idea in this research is based on active database and centralized history maintenance. Implementation and testing of the proposed architecture results in accuracy percentage of 96.68% to predict CPU usage of jobs and 91.29% of memory usage and 89.80% of the band width usage.

***Keywords*—**Active Database, Grid Computing, Resource Requirement Prediction, Scheduling,

## I. Introduction

IN Grid systems, QOS[1] is not limited to network bandwidth but extends to the processing and storage capabilities of the nodes. Thus the focus is on the degree a Grid can provide end-to-end QOS rather than providing only QOS on the network. When a Grid job has QOS requirements, it may be necessary to negotiate a *service level agreement* (SLA) to enforce the desired level of service. Resource reservation is one of the ways of providing guaranteed QOS in a Grid with dedicated resources. [6] One of the major issues in resource reservation is correct matching of available resources with required resources, this results effectively utilize various resources in the system, such as CPU cycle, memory, communication network, and data storage. [2] Correct matching operation depends on percept job resource requirements. This can be obtained by two methods. In the first method, environment can ask resource requirements from user. This is not proper in QOS. Another method is that architecture itself predicts resource requirement. job resource requirements prediction algorithms operate on the principle that jobs with similar characteristics have similar resource requirements. Thus, we maintain a history of jobs that have executed along with their respective resource requirements. [4] To estimate a given jobs' resource requirements, we identify similar applications in the history and then compute a statistical estimate (such as the mean and linear regression) of their runtimes. We use this as the predicted resource requirements.

Prediction can be done in different ways such as centralized prediction and decentralized prediction. Also it can be done in different locations such as system scheduler, resource manager or gate keeper of each site in Grid computing. [3]

Prediction of job resource requirements is based on decentralized method in most of the recent researches. Decentralized methods have several disadvantages. The most important disadvantages are:

- Necessity of large number of interchange and transmission of information for prediction between sites and broker.
- Increase of errors
- Relatively longer time for prediction
- Limitation of history in each site
- Saving replica jobs in the sites and inability to remove repeated jobs.

Saving executing information of sites in the scheduler will increase accuracy percentage in finding the similar job, because the number of available jobs in History will increase accordingly. Therefore prediction will be more close to actuality and it is not necessary to have large number of transmission in the network. In the proposed method, we consider that database will update with specific threshold. Exactly similar jobs will be deleted from the history in updating process. In proposed method active database is used.

## II. Prediction supplies

In the proposed method we use Java language to implement grid architecture. Also we use SQL-Server software to implement database. The suggested architecture in this research is named "Grid-HPA"[2]. The new idea which is implemented in this research is that resources' attributes are saved in an active database. Unlike the ordinary databases, active databases are able to show reaction to environment and do activities according to procedures and conditions.

In implemented active database, there is a table which is specified to save users requests. This table does reaction according to random events such as entry of a new request,

M. Bohlouli is with the Department of Computer Engineering, Iran University of Science and Technology (IUST) , Narmak SQ, Tehran , Iran. Phone number: +98 - 914 – 3030160; fax number: +98 – 411 – 3808179; e-mail: mbohlouli@comp.iust.ac.ir

M. Analoui, is with Department of Computer Engineering, IUST, Narmak SQ, Tehran, Iran, Phone number: +98 – 21 - 73913329
e-mail: analoui@iust.ac.ir

[1] Quality of Service

[2] Grid History Based Prediction Architecture

finishing a turned over job, introducing a new resource or introducing available resource and acts as required. Therefore by using active databases, match level is not necessary and according to different events "Match Algorithm" will be executed.

If a resource finishes executing a job it will return results to broker again and the customer is not involved in this. After receiving results, broker will send results to the customer.

This is true when the customer's job is not broken down into sub jobs, otherwise it should wait until return of all of sub jobs' result, then it sends total result to the customer. Since broker is an interface between customers and resource, it can break down job to sub jobs and turn over each sub job to one resource. This is one of the most important advantages of this method. After receiving all results of sub jobs, it will return total result to the main customer. In fact different parts of a job which can be executed separately execute in different sites, therefore the result can be ready rapidly.

When a new resource is introduced, in fact it is inserted in related database. Thus there should do activities to insert the record. Since there is a new idle resource, the broker should check the list of jobs to see if a job without a resource matches this new idle resource. (Fig. 1)

```
On insert
   If have a job
        Then Match_Resouce (New_resource)
```

Fig. 1 Connecting new resource semi code

When connection of a resource disconnects first it should be distinguish that current job is finished or no. if it is not finished it should be put in priority and find another proportional resource as soon as possible; otherwise it just will be removed from the list of resources. (Fig. 2)

```
On delete
        If not finish job
              Then Match_Job (job)
```

Fig. 2 Disconnecting a resource semi code

When a job is entered, insert occurs in the database; if there is a free resource, match algorithm should be executed until distinguishing that is it possible to specify a job to resources or not. (Fig. 3)

```
On insert
        If have a free resource
              Then Match_Job (job)
```

Fig. 3 Entering a new job semi code

When determination of a job is declared status of resource should be changed from busy to free. So up-date event will occur in the database and it should check that if a result received it is sent to the customer. Return function will return result to the related customer.(Fig. 4)

```
On update
        If have a result
              Then return (result)
```

Fig. 4 Finishing a job semi code

When the main customer's job divides into several sub jobs, divider will define a virtual customer into the broker until it returns the result of a job to virtual customer, not actual customer. Whenever virtual customer received all of subjects' result it will send the total result to the main customer.

Also collapse of a resource deletes it from the list of the resources. If the previous state was busy, the job of the collapsed resource will be turned over to other appropriate resource. (Fig. 5)

```
On delete_resource
        If state=busy
              Then Match_Job (job)
```

Fig. 5 Collapse of a resource semi code

In the proposed method we suppose that executed information and used resources are saved in broker and are in concentrated form. The act of prediction will be done in grid's broker. In addition to execution time, band width and used memory to execute the job will be predicted in this research.

## III. The Proposed Method Algorithm

For ease of use we have divided the attributes of each job into two groups: the attributes that will predict are called decision attributes and attributes that exploit about each job and checks the similarity among jobs are named conditional attributes. [4] Decision attributes include quantity of usage of CPU, quantity of usage of memory and quantity of required band width. Conditional attributes are consisting of size of file, number of defined variables and their type and number of repeated loops.

One of the steps is calculation of the degree of relationship of the decision attributes to each of the conditional attributes. This act is done according to several tests and specifies their equation exactly. Suppose each of conditional is called $C_i$ and each of decided attributes is called $D_i$. The primary saved database for 5 jobs with 4 condition attributes and 3 decision attributes is summarized in Table I. It should be mentioned that the saved database is much larger than what is presented in Table I as representative.

The proposed method algorithm is:

1. All of available numbers in history will be normalized according to equation (1).

TABLE I
A SAMPLE OF SAVED INFORMATION SYSTEM IN HISTORY

| | $C_1$ | $C_2$ | $C_3$ | $C_4$ | $D_1$ | $D_2$ | $D_3$ |
|---|---|---|---|---|---|---|---|
| $Job_1$ | 1 | 35 | 50 | 20 | 10 | 11 | 90 |
| $Job_2$ | 15 | 20 | 90 | 18 | 15 | 12 | 88 |
| $Job_3$ | 10 | 40 | 110 | 16 | 11 | 11 | 78 |
| $Job_4$ | 19 | 39 | 123 | 19 | 11 | 15 | 76 |
| $Job_5$ | 20 | 30 | 129 | 29 | 17 | 14 | 78 |

$$X_{i,j}^{New} = \frac{X_{i,j}^{old} - Av}{\sigma} \quad (1)$$

In equation (1), Av is average of numbers and σ is diversion. Both of these quantities calculate from equation (2), (3), i denotes the column number i.e. a attribute of the job and j denotes the row number or the job itself. After implementation of step 1 the table I will be changed to table II.

$$Av = \frac{\sum_{i=1}^{n} X_{ij}}{n} \quad j = 1...k \quad (2)$$

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{k}(X_{ij} - Av)^2} \quad (3)$$

Variable n in equation (2) and (3) shows the number of available jobs in the database. All of available databases are normalize by the mentioned equation.

TABLE II
CONCLUDED TABLE AFTER IMPLEMENTATION OF STEP 1 IN HISTORY

| | $C_1$ | $C_2$ | $C_3$ | $C_4$ | $D_1$ | $D_2$ | $D_3$ |
|---|---|---|---|---|---|---|---|
| $Job_1$ | -2.23 | 2.24 | -2.23 | -2.35 | -2.24 | -2.2 | 2.23 |
| $Job_2$ | 0.37 | -2.2 | -0.45 | -2.2 | 1.38 | -0.8 | 1.34 |
| $Job_3$ | -0.54 | 1.08 | 0.41 | -2.1 | -1.01 | -1.53 | -0.83 |
| $Job_4$ | 0.97 | 0.86 | 0.89 | -0.64 | -0.85 | 1.6 | -1.09 |
| $Job_5$ | 1.00 | -0.17 | 1.00 | 1.95 | 1.48 | 0.86 | -0.69 |

2. We normalize the condition attributes of the current job which we want to predict its decision attributes using equations (1), (2) and (3).
3. To predict the required CPU for a job, first we should know which condition attribute extracted for jobs can affect the CPU performance. Based on carried out experiments, it was indicated that these attributes are the number of the existent loops in the program codes, the number calls in the program codes and the number of variables in the program codes. The distance of the current job with all jobs in the database should be found in order to calculate the used CPU based on mentioned condition attributes. Also the distance should be known for finding the condition attributes saved capacity for jobs in the History and for implementation in equation 4 and to identify the condition attributes of the current job.

$$D_{cpu_j} = \sqrt{\begin{aligned}&(Loops - Loops_j)^2 \\ &+(Calls - Calls_j)^2 \\ &+(Variables - Variables_j)^2\end{aligned}} \quad (4)$$

In equation (4) variable *loops* show the number of available loops in job and *calls* show number of available calls in job and *variables* show number of available variables in the job and variables *loops, Calls and Variables* denotes the current Job and j starts from 1 to n (The number of jobs in database).

4. After calculating all of $D_{cpu}$, we sort them ascended by Dcpu.
5. Among all of the available jobs in the sorted list, we select √n job from beginning of the list.
6. We consider average quantity of selected jobs in the list for the $job_i$.

Prediction algorithm is like the one for used band width with difference that in step 3 we are using equation (5) instead of equation (4).

$$D_{BW_j} = \sqrt{(FileSize_i - FileSize_j)^2} \quad (5)$$

Also we use this method for prediction of required memory with difference that equation (4) in step 3 will change to equation (6)

$$D_{Mem_j} = \sqrt{\begin{aligned}&(Variable_i - Variable_j)^2 \\ &+(FileSize_i - FileSize_j)^2\end{aligned}} \quad (6)$$

Note that in predicting required Memory in step 4 of mentioned algorithm we should sort the list ascended by Dmem and for required Band width by Dbw. In mentioned algorithm we suppose that database consists of n record (n>=4). When the number of available record is increased accuracy of prediction will increase too.

## IV. IMPLEMENTATION AND TESTING

In implementation step, prediction engine is implemented in the scheduler and database also holds the executed jobs in scheduler.[4] When a job is given to broker for execution, prediction engine will predict required resources of job such as CPU, memory and Band width using similarity algorithm, described in section 2 and available database in the scheduler. Then based on the available resources' database of the sites, Match resource algorithm will be executed if there is free and matched resource. After execution of a job in any site, the result of execution and information about the used resources in execution will be sent to the broker and the broker will give the result to the user and also will save execution information in History. In each saving time in History, database of executed jobs will be updated and exact similar jobs will be deleted from History and only one of them will be saved in History. Also with specific threshold the broker connects the sites and requests information about any changes in their

resource abilities. Then with received information about changed resources the broker updates available resources database. Therefore every time the last and accurate information about resources is in the broker's database.

If the number of executed jobs saved in History increases, the accuracy percentage of prediction will also increase in this architecture. Obviously the number of the executed jobs increases with very high rate in grid computing. Therefore in a short time the predicted values get close considerably to the actual values. If the exact similar jobs are not deleted from History, the size of History will be enlarged with unused information in a short time. The algorithm for deleting the exactly similar jobs from History is as follows:

1. The value of the decision attributes in Table I should be normalized using equation (1), (2) and (3). The results of this step which is done using previous algorithm in table II.
2. All three values of CPU, Memory and Band width are summed for each job.
3. The jobs are sorted based on the results.
4. For two successive jobs in the History, the differences of decision attributes are calculated and add together. If the result is less than $5\times10^{-5}$ two jobs are exactly the same or are very similar.
5. The step 4 will be repeated for the condition attributes. If the result is less than $5\times10^{-5}$ two jobs are the same and one should be deleted from History. Otherwise nothing will happen.

From deleting the same jobs, it may seem that this is very time consuming and it is repeating the same procedure, but this is not true, because this will be done for each job only once and only when it is inserted in the History a job is investigated to sort it in the already determined list of jobs. This algorithm is applied only for two prior and posterior jobs. Therefore the date in the resulted History will be very brief and useful.

## V. Conclusion

The prediction accuracy for executed time in Caltech group's method [3] (decentralized) is 86.47% while prediction accuracy in this research for is 96.68%. In the proposed method the numbers of jobs are 53. These standard jobs includes the algorithm of calculation of Π, Nipper value, Fibonacci series, factorial of large numbers and other algorithms which in the testing of Alchemi are also used. [7] Primitive jobs in the proposed method were tested upon introduced algorithm in Caltech group conclude that accuracy in decentralized method in this test is 80, 36% by used jobs while accuracy of proposal method with same condition is 96.68%. Also the speed of finding answer in the centralized method is faster than similar decentralized methods. With due attention to result in same condition we conclude that centralized method has more prediction accuracy and less error and more speed than decentralized method.

Proposed prediction accuracy will be increased if available jobs increase in history and also if more related attributes for prediction used in equation (4), (5) and (6) will increase prediction accuracy. The average prediction accuracy for used CPU is 96. 68% and the average prediction accuracy for used memory is 91.29% and the average prediction accuracy for required band width is 89. 80%.

Figure (6) shows predicted values for required CPU in 20 samples of jobs and also values of used CPU after executing 20 jobs. Also figures (7) and (8) show these values for Memory and Band width. Error percent is calculated for each of 53 used job in this research. Then calculated error will is used for calculating average error predictor. The average of prediction engine error for predicting used CPU is 3.32% and for used memory is 8.71% and required band width is 12.6%. Accuracy percentage estimated by using equation (7) for each job, then we should calculate the average of results.

$$AccrPer=(\frac{ActValue-PValue}{ActValue})\times100 \quad (7)$$

In equation (7) the ActValue variable shows acquired value from act and PValue variable shows predicted value and percent of error prediction is equal to AccPer.

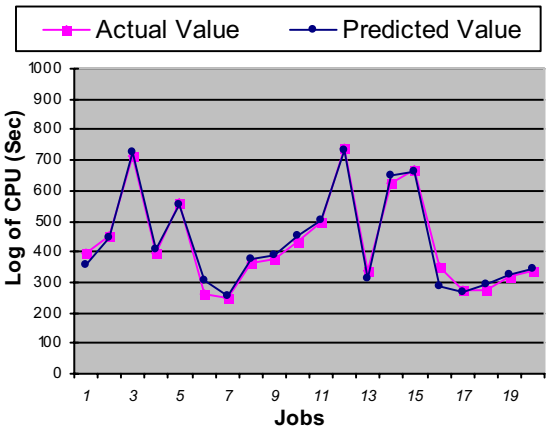


Fig. 6 Actual & Predicted CPU for 20 test cases

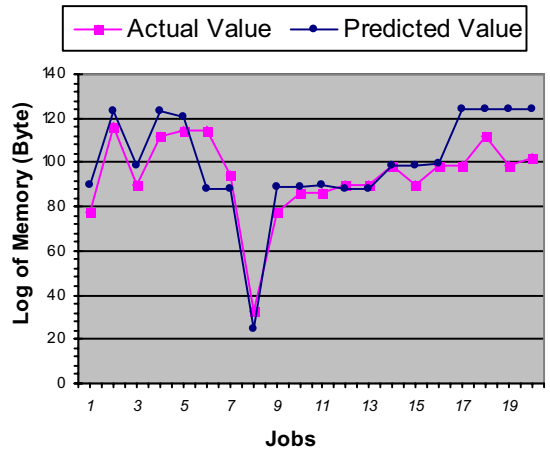


Fig. 7 Actual & Predicted Memory for 20 test cases

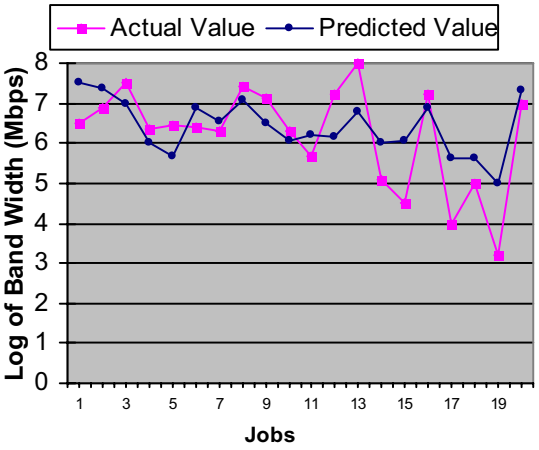


Fig. 8 Actual & Predicted Band width for 20 test cases

## ACKNOWLEDGMENT

We would like to thank Prof. B. Schnor from Potsdam University for her constructive comments on an earlier version of this paper. Also we thank Dr. M. Moazzen from University of Tabriz for his helps.

## REFERENCES

[1] I. Foster, and C. Kesselman, The Grid: Blueprint for a New Computing Infrastructure. Morgan Kafmann Publisher Inc., 1999.
[2] C. Weng, X. Lu, "Heuristic scheduling for bag-of-tasks applications in combination with QoS in the Computational Grid", Elsevier, Vol. 21 Pages 271-280, 2005.
[3] A. Ali, A. Anjum, J. Bunn, R. Cavanaugh, F. Lingen, R. McClatchey, M. Mehmood, H. Newman, C. Steenberg, M. Thomas and I. Willers, "Predicting the Resource Requirements of a Job Submission", Computing in High Energy Physics (CHEP04) , Switzerland, 2004.
[4] S. Krishnaswamy , S. Wai Loke and A. Zaslavsky "Estimating Computation Times of Data-Intensive Applications" IEEE Distributed Systems Online Vol. 5, No. 4, pp. 127-136, April 2004.
[5] P. Keyani, N. Sample and G. Wiederhold, "Scheduling Under Uncertainty: Planning for the Ubiquitous Grid", proc. of 5th international conf. on coordination models and languages, pp. 300-316, 2002.
[6] K. Krauter, R. Buyya and M. Maheswaran, "A Taxonomy and survay of Grid Resource management Systems", Software- practice & experience, vol. 32, pp. 135-164, 2002.
[7] A. Luther, R. Buyya, R. Ranjam. and S. Venugopal, " Alchemi: A .Net-based Grid Computing Framework and its Integration into Global Grids", Technical Report, GRIDS-TR-2003-8, Grid Computing and Distributed Systems Laboratory, University of Melbourne, Australia, December 2003.
[8] M. Roehrig, W. Ziegler and P. Wieder, "Grid Scheduling Dictionary of Terms and Keywords", Vol. GFD-I11 of global Grid Forum Documents, Global Grid Forum, 2003.
[9] Y. Gao, H. Rong and J. Huang, "Adaptive grid Job scheduling with Genetic Algorithms", Future Generation Computer Systems, Vol. 21, pp. 151-161, Jan. 2005,
[10] R. Buyya, D. Abramson, and J. Giddy, "Nimrod-G: An Architecture for a Resource Management and Scheduling System in a Global Computational Grid," *Proc. Fourth Int'l Conf. High-Performance Computing*, Asia-Pacific Region, IEEE CS Press, Los Alamitos, Calif., 2000.
[11] R. Buyya, D. Abramson, and J. Giddy, "Economy-Driven Resource Management Architecture for Computational Power Grids," *Proc. Int'l Conf. Parallel and Distributed Processing Techniques and Applications*, CSREA Press, 2000.
[12] R. Buyya et al., "Economic Models for Management of Resources in Peer-to-Peer and Grid Computing," *Proc. SPIE Int'l Conf. Commercial Applications for High-Performance Computing*, SPIE, Bellingham, Wash., 2001.
[13] O. Elgerd, *Electric Energy Systems Theory: An Introduction*, 2nd ed., McGraw Hill, New York, 1982.
[14] I. Foster and C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit," *Int'l J. Supercomputer Applications*, vol. 11, no. 2, 1997, pp. 115–128.
[15] H. Casanova and J. Dongarra, "NetSolve: A Network Server for Solving Computational Science Problems," *Int'l J. Supercomputer Applications and High Performance Computing*, vol. 11, no. 3, Fall 1997.
[16] I. Foster et al., "A Security Architecture for Computational Grids," *Proc. 5th ACM Conf. Computer and Communications Security*, ACM Press, New York, 1998.
[17] R. Buyya, J. Giddy, and D. Abramson, "A Case for Economy Grid Architecture for Service-Oriented Grid Computing," *10th IEEE Int'l Heterogeneous Computing Workshop*, IEEE CS Press, Los Alamitos, Calif., 2001.

**M. Bohlouli** was born October 21, 1979, at Marand, East Azarbaijan, Iran. He received his early education at the elementary and secondary schools in his home town and from then on his life has been devoted to a search for knowledge. He continued his studies at the Iran Islamic Azad University, Qazvin Branch and gained a degree in Computer hardware engineering there, in 2003. Then he graduated with a M.Sc. in Computer Systems Archiectures from Iran University of Science and Technology (IUST), Tehran, Iran in 2007. His major research interests are parallel and distributed systems, Grid computing and Robotics.
He is University Lecturer at Nabi Akram University, Tabriz, Iran. He is a Visiting Lecturer at a few universities – Daneshvaran University, Islamic Azad University of Iran, Traktorsazi University. He expects to be a Ph.D. student at Potsdam University in the Fall of 2008.