

Performance Prediction-based versus Load-based Site Selection: Quantifying the Difference

Seung-Hye Jang, Valerie Taylor, Xingfu Wu, Mieke Prajugo
Department of Computer Science
Texas A&M University, College Station, TX 77843
{jangs, taylor, wuxf, mprajugo}@cs.tamu.edu

Ewa Deelman, Gaurang Mehta, Karan Vahi
USC Information Sciences Institute
4676 Admiralty Way, Marina Del Rey, CA 90292
{deelman, gmehta, vahi}@isi.edu

Abstract

Distributed systems are available and provide vast compute and data resources to users. With the availability of multiple resources, one of the major issues to be addressed is site selection. Users have access to many resource sites from which to select for execution of applications. In this paper, we quantify the advantages of using performance prediction to select sites as compared to using load information, which is the widely used method. The quantification is based upon two case studies. The first case study involves a large-scale scientific application, called GEO LIGO, for which the experimental results indicate an average of 33% performance improvement as compared to a load-based method. The second case study involves a web-based, educational application, called AADMLSS, for which the results indicate an average of 10% performance improvement as compared to a load-based method.

Key Words: Site Selection, Application Performance Prediction, Grid Computing

1. Introduction

Distributed systems, such as the Distributed Teragrid Facility [7], the Grid 2003 [3], and the European Data Grid [14] are available and provide vast compute and data resources. Large-scale applications such as cosmology, ocean modeling and gravitational-wave physics often require computational and/or data grids to obtain the needed compute and/or storage resources necessary for execution. For the case of web-based applications, distributed systems allow for server replication to avoid a single server becoming a bottleneck, thereby causing significant performance degradation. One of the major issues to be addressed with distributed systems is the selection of sites. In this paper, we quantify the advantage of using performance prediction versus load information for site selection. The goal is to select the site that results in the smallest execution time of the application.

There are many site selection systems that utilize load information [12][19][20][21][22]. Kalaiarul and Collier use load information for web server site selection using

mobile agent technology [22]. In [19], Badidi et. al. use load information for server selection in a distributed object computing environment. In these load-based site selection systems, the underlying assumption is that selecting a site with the smallest load will give the best performance in terms of application execution time.

We present a site selection method that uses application performance predictions, based upon historical data. It is well known that site selection based on the processor speed or memory does not guarantee minimum execution time. The performance of an application is often affected by many aspects such as interactions between the dataflow and computations in the application with the system architecture. Significant work has been done in the area of performance prediction for parallel applications [13][16]. This work indicates that the use of historical data results in good predictions of the execution times of parallel applications. Our work uses the Prophecy infrastructure [15] to predict application performance on different sites; Prophecy uses historical data to generate analytical performance models for predictions.

To quantify the difference between load-based and performance prediction-based site selection methods we use two case studies. The first case study uses a pulsar search application, called GEO LIGO [1][5], on the Grid2003, for which the results indicate an average of 33% reduction in execution time using application performance prediction as compared to load-based site selection. The second case study involves a web-based, educational application, called AADMLSS (African-American Distributed Multiple Learning Styles System) [11]. In this case study, AADMLSS is replicated across multiple servers, and our prediction method considers the network performance as well as the server performance. The results indicate an average of 10% performance improvement as compared to load-based selection.

The remainder of the paper is organized as follows. Section 2 provides an overview of the Prophecy infrastructure that is used for the predictions. Section 3 describes the first case study using GEO LIGO pulsar search application executed on the Grid2003. Section 4 presents the second case study using the web based

application, called AADMLSS, and Section 5 summarizes the work and describes the future work.

2. Prophecy: Performance Predictions

The performance predictions used for site selection are obtained from the Prophecy infrastructure, which archives the performance data and uses this history to generate analytical performance models for predictions [15][17]. Prophecy consists of three components: data collection, databases, and data analysis. In this paper, we focus on the data analysis component, which includes the model builder.

The model builder is used to formulate predictions based upon the archived performance data. Currently, Prophecy includes three methods for developing analytical models for predictions: (1) curve fitting, (2) parameterization of the application code, (3) coupling of the kernel models that comprise the application [18]. In this paper, we utilize the curve fitting method for developing the predictions. Curve Fitting is a method that uses optimization techniques, such as multivariate regression, to develop a model. For this method, Prophecy uses the empirical data found in the database and the computational complexities of the application to generate the model. The application inputs are used as variables for the curve fitting method.

3. Case Study I: Site Selection in Grids

In the computational grid case study, we use application performance predictions to select a site from available Grid2003 [3] sites for the GEO LIGO pulsar search application. This application was developed by collaborators from the German-English Observatory (GEO) [1] and the Laser Interferometer Gravitational-wave Observatory (LIGO) [5], hence called GEO LIGO. The pulsar search is a process of finding celestial objects that may emit gravitational waves. Our experiment focuses on the application computing F-statistics for known pulsars using pre-generated Fourier Transform files. The F-statistics generation requires intensive computation and storage resources.

3.1. Testbed

Figure 1 shows the overview of the testbed consisting of the Pegasus meta-scheduler [8] and the Prophecy prediction system [15]. Pegasus maps an abstract workflow description onto the available grid resources. The abstract workflow indicates the logical transformations that are to be performed, the order and the

logical data files that transformations consume and produce. The abstract workflow does not include information about where to execute the transformations nor where the data is located. Pegasus uses various grid services to find the available sites, the needed data and the executables that correspond to the transformations. Originally, Pegasus mapped a workflow onto the available sites using random allocation. To improve the Pegasus site selection, we provide a uniform interface between Pegasus and Prophecy to enable Pegasus to make better mappings between sites and workflows using the Prophecy application predictions.

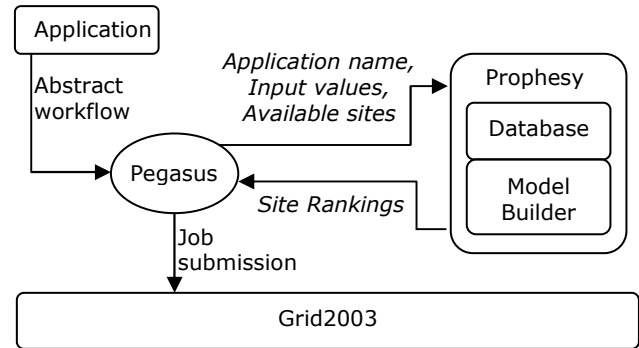


Figure 1. Testbed Overview.

Pegasus sends to Prophecy information such as the application name, the input parameter names and their values, and the list of available sites. When the query is received by the Prophecy server, Prophecy returns the rankings of the available sites using predicted execution times at each site. The training set for this case study included historical data from all sites.

We executed the GEO LIGO application on the Grid2003 using grid middleware and collected the performance data. Grid2003 is a collaborative operational grid between three major projects: International Virtual Data Grid Laboratory (iVDGL) [4], Grid Physics Network (GriPhyN) [2], and Particle Physics Data Grid (PPDG) [6]. Grid2003 consists of 30 participating sites totaling over 3200 CPUs. Table 1 shows the site specifications of the selected Grid2003 sites used in this case study. We had access to a limited number of sites, as many of the sites are reserved for the major experiments within the high energy physics community.

3.2. Training Set

We identified two input parameters that significantly affect the execution time of the GEO LIGO pulsar search. The parameters are AlphaBand and FreqBand, which together determine the search scope and the

Site Name	CPUs	Batch	Compute Nodes		
			Processors	Cache Size	Memory
alliance.unm.edu (UNM)	436	PBS	1 X PIII 731 GHz	256 KB	1 GB
atlas.iu.edu (IU)	400	PBS	2 X Intel Xeon 2.4 GHz	512 KB	2.5 GB
pdsfgrid3.nersc.gov (PDSF)	349	LSF	2 X PIII 650-1.8 GHz 2 X AMD 2100+ - 2600+	256 KB	2 GB
atlas.dpcc.uta.edu (UTA)	158	PBS	2 X Intel Xeon 2.4 - 2.6 GHz	512 KB	2 GB
nest.phys.uwm.edu (UWM)	296	CONDOR	1 X PIII 1GHz	256 KB	0.5 GB
boomer1.oscer.ou.edu (OU)	286	PBS	3 X Intel Xeon 2 GHz	512 KB	2 GB
cmsgrid.hep.wisc.edu (UWMadison)	64	CONDOR	1 X Intel Xeon 2.8 GHz	512 KB	2 GB
cluster28.knu.ac.kr (KNU)	104	CONDOR	1 X AMD Athlon XP 1700+	256 KB	0.8 GB
accdc.ccr.buffalo.edu (Ubuffalo)	74	PBS	1 X Intel Xeon 1.6 GHz	256 KB	3.7 GB

Table 1. Selected Grid2003 Sites (Batch: local scheduler).

Parameters		Prediction-based		Load-based			Random		
Alpha	Freq	Selected Site	Time (sec)	Selected Site	Time (sec)	Diff (%)	Selected Site	Time (sec)	Diff (%)
0.0065	0.002	PDSF	3863.66	UWMadison	9435.80	59.05%	UWMilwaukee	48065.83	60.09%
0.0085	0.001	IU	2850.39	UWMadison	11360.28	74.91%	KNU	7676.56	62.87%
0.0075	0.009	IU	22090.17	PDSF	20197.88	-9.37%	UNM	77298.13	71.42%
0.0055	0.009	IU	16216.25	UTA	27412.45	40.84%	UWMadison	31555.10	48.61%
0.0005	0.009	PDSF	1365.51	Ubuffalo	3226.00	57.67%	UWMilwaukee	16009.82	91.47%
0.0075	0.003	PDSF	6723.30	IU	7343.37	8.44%	KNU	8287.77	18.88%
0.0065	0.007	PDSF	13561.01	PDSF	13561.01	0.00%	UNM	52379.31	74.65%
0.0085	0.004	PDSF	10121.27	Ubuffalo	19649.22	48.49%	IU	11158.72	9.30%
0.0035	0.005	PDSF	5241.28	Ubuffalo	20799.05	74.80%	UWM	51936.49	89.91%
0.0065	0.009	IU	19184.36	UWMadison	24995.94	23.25%	OU	23441.16	18.16%
0.0045	0.009	IU	13278.68	UTA	20453.30	35.08%	UWMadison	14137.44	6.07%
0.0085	0.009	IU	25021.39	UWMadison	26246.68	4.67%	OU	31538.22	20.66%
Average						32.62%			52.01%

Table 2. Site Selection for GEO LIGO with Measured Execution Times.

frequency range of the pulsar search. We collected performance data using different input values and archived the data in the Prophecy database. The initial training data was collected from nine Grid2003 sites and consists of 60 individual runs at each site. Upon completion of each additional GEO LIGO execution, the performance data is added to the database, and this updated archival of performance data is used to generate successive performance models.

From the initial training set, we observed that given different input values, the site with the smallest execution time changes. This observation demonstrates that selecting a static site to run this application does not minimize the application execution time.

3.3. Experimental Results

The experimental results entail a comparison of the performance prediction-based site selection with load-based and random selection methods. Load-based selection method collects and compares site loads before the job

submission and selects the least loaded site for execution. We utilize Ganglia [9] to monitor and collect the loads of the Grid2003 sites. Random selection uses the random number generator function `rand` with the seed value 1000 to identify a random site to be used for execution.

Table 2 gives the experimental comparisons of the three selection methods: the application performance prediction-based method, the load-based method, and the random method. The results in the table include the site selected for execution by each selection method, the measured time of GEO LIGO executed on the selected site, and the percent difference as compared to the prediction-based selection method. Negative values indicate better site selection than that resulting from the performance prediction; and positive values indicate that the performance prediction has the better selection corresponding to a smaller execution time.

The results indicate that the performance prediction-based method provided efficient site selection versus load based and random in all but one case; for this one case the best site was off by 9%. The application prediction-based

site selection performed an average of 33% better (or smaller) in execution time than the load-based site selection. The results indicate that the load-based method might select a slower site with the least load, which results in the non-optimal site selection. In comparison to the random selection, the results indicate 52% improvement using prediction. The problem with the random method is that the results will be different for a different seed value. It is noted that the overhead for obtaining the predictions was negligible and occurs prior to execution of the application.

4. Case Study II: Web Server Selection

For our second case study, we use the performance prediction-based method to select the web server that has the minimum predicted server response time. The estimation of the server response time is based on application performance as well as network performance predictions.

Relying on a single web server to handle a large number of user requests is undesirable; the use of multiple server replicas to respond to high user requests has been known to offer many advantages including improved service availability and short response time for a requested service. The application, called AADMLSS (African-American Distributed Multiple Learning Styles System) [11], is used to evaluate the use of application performance for server selection. AADMLSS is an online educational system that incorporates the use of culture and the integration of sophisticated instructional tools into its learning environment as an attempt to improve a student’s learning experience and academic performance.

4.1. Testbed

Figure 2 gives the server selection process used with AADMLSS. The process starts with the user connecting to an AADMLSS server, which connects to the Prophecy system for server selection. Upon receiving a request from the ADDMLSS server, Prophecy retrieves performance data

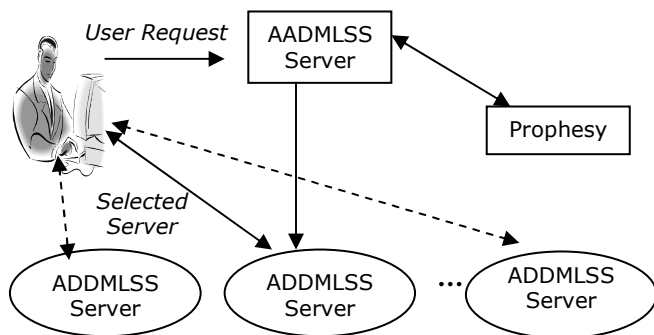


Figure 2. AADMLSS Site Selection Overview.

from the database and generates an analytical model for server response time. Prophecy returns to the server a ranked list of the sites that contain a replica of AADMLSS. The ranking is based upon estimates of the server response time. Subsequent user requests are handled by the selected server corresponding to the site ranked number one.

The server response time is computed using estimated network delay and server access time. The network delay is defined to be the time required for the user to access the concept material from the selected server. The network delay is determined by sending different packet sizes from each candidate server to the user machine. Four different packet sizes are carefully chosen to minimize the overhead, and the corresponding performance data are archived in the Prophecy database. Subsequently, Prophecy generates an analytical model for the network performance using statistical analysis. The server access time is defined as the time required for the server to access the requested files, for which the estimate is based upon historical data.

We evaluated the performance prediction-based site selection using three servers with replicas of ADDMLSS. The servers are located at Texas A&M University (TAMU), Auburn University (AU), and Boston University (BU). Table 3 provides the specifications of each server replica. The hardware specifications are fairly different among the three servers.

SPECS	Loner (TAMU)	Tina (BU)	Interact (AU)
CPU Speed	997.62 MHz	1993.56 MHz	697.87 MHz
Bus Speed	205 MB/s	638 MB/s	214 MB/s
Memory	256 MB	256 MB	256 MB
Hard Disk	30 GB	40 GB	10 GB

Table 3. Specifications of the Testbed.

Similar to the case study I, we have compared our prediction-based method to two other selection methods: load-based selection and random selection. For the load-based selection, we use the standard UNIX resource monitoring utility `uptime` to identify the site with the least load. Three sets of experiments were conducted using 1) all three servers, 2) two servers, consisting of a local server (Loner) and a remote server (Interact), to focus on the impact of network performance for the resource selection, and 3) two remote servers (Tina and Interact). With the third experiment, the two remote servers have comparable network performance; this experiment is used to investigate the impact of server performance for the site selection. The users of ADDMLSS in this study are located in the Department of Computer Science at Texas A&M University. Hence, Loner is considered to be local, and the remaining sites are remote. The results from all three experiments are given in the next section.

Concepts	Using All Servers		Using Local and Remote		Using Remote Servers	
	(Load-Prediction)(%)	(Random-Prediction)(%)	(Load-Prediction)(%)	(Random-Prediction)(%)	(Load-Prediction)(%)	(Random-Prediction)(%)
3/0/0 D	6.21	14.05	9.91	10.24	3.13	4.03
3/0/1 D	12.13	21.94	13.04	15.06	4.26	5.97
3/0/2 N	14.02	25.83	18.06	19.16	7.02	8.28
3/0/3 N	18.12	23.52	20.54	21.29	8.64	9.02
3/1/0 N	8.05	12.04	9.81	9.58	3.25	4.94
3/1/1 N	7.31	12.25	7.02	7.91	3.27	4.10
3/1/2 N	12.60	18.74	11.35	12.15	3.93	5.97
3/1/3 N	10.96	19.11	10.47	10.36	3.64	4.08
3/2/0 N	7.93	12.58	8.56	8.67	3.15	3.32
3/2/1 N	8.05	14.25	8.75	9.75	4.39	5.20
3/2/2 N	9.14	15.97	10.06	10.92	5.80	5.97
3/2/3 D	9.79	20.58	10.15	10.50	6.52	6.95
3/3/0 D	8.94	13.64	8.41	9.56	4.39	5.64
3/3/1 D	8.26	16.74	8.58	8.08	4.16	5.20
3/3/2 D	9.21	15.21	8.31	7.95	4.81	5.73
3/3/3 D	9.97	19.36	10.21	10.19	5.02	5.58
Average	10.04	17.24	10.83	11.34	4.71	5.62

Table 4. Site selection for AADMLSS and Performance Comparisons with Load-based and Random.

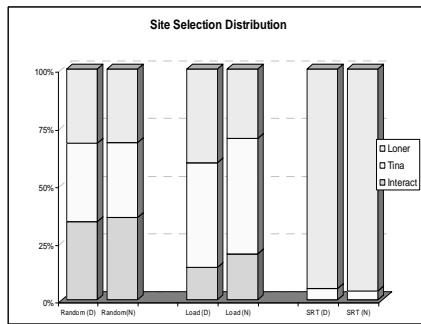


Figure 3. Three-Server Site Selection Distribution.

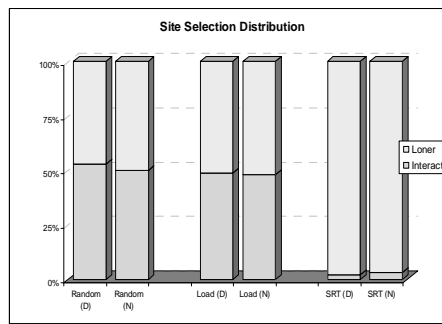


Figure 4. Two-Server Site Selection Distribution (local and remote)

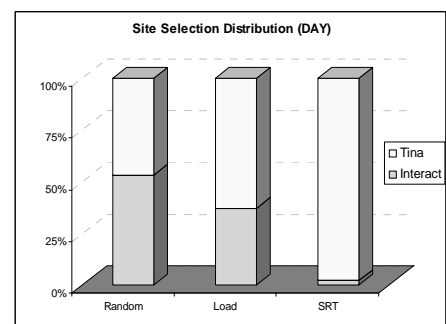


Figure 5. Two-Server Site Selection Distribution (remote servers)

SRT: Server Response Time (our prediction-based method), D: Daytime, N: Nighttime

4.3. Experimental Results

The results from the experiments are given in Table 4 and Figure 3-5. The results in Table 4 are presented in terms of percentage difference between our performance prediction-based method and load-based method; similar with respect to the random selection method. The first column in Table 4 gives the educational concept number that the user is accessing. The term **D** and **N** denote the time the experiment was conducted, for which **D** corresponds to Daytime and **N** corresponds to Nighttime. We decided to distinguish the two since network performance can vary depending on the time of day. The average file sizes for the different education concepts ranged between 6 KB and 60,654 KB. The next two columns in Table 4 indicate the results using all three servers; performance prediction-based method outperforms the load-based and random selection

methods by an average of 10% and 17% respectively. Figure 3 indicates the performance prediction-based method selects the local machine (**Loner**) for the majority of the time when using all three servers. These results indicate that network delay is a dominating factor in identifying an optimal site to allocate to the user.

To examine the impact of network delay on the server response time in detail, we conducted the two-server experiments using a local server (**Loner**) and a remote server (**Interact**). **Interact** was chosen as the second site since it has comparable server specifications to **Loner**. The two middle columns in Table 4 demonstrate that the performance improvement obtained by using the performance prediction-based method is better by an average of 10% and 11% with respect to the load-based and the random-selection methods, respectively. Since the server load between the two replicas in this experiments

were similar, there is a negligible difference in the performance improvements between using the load-based and the random selection methods. In addition, the performance prediction-based method selects the local server (Loner) for the majority of the time as shown in Figure 4, hence confirming our assumption made above regarding network delay between the user and the server as an important part in selecting a good server resource.

The two-server experiments using two remote servers highlight the impact of the server specifications in terms of hardware characteristics. The last two columns in Table 4 show that the performance prediction-based method performs better by an average of 4% and 5% compared to the load-based and the random selection methods, respectively. The site selection distribution in Figure 5 indicates that the performance prediction-based method selects Tina, which has better server hardware specifications between the two remote servers. The results indicate that server performance also has influential impact on the overall service response time, although the impact was not as large as in the case of network delay.

5. Summary and Future Work

In this paper, we quantify the advantages of using application performance prediction with site selection using two case studies. The first case study, which using the GEO LIGO application, indicates an average of 33% performance improvement as compared to using load information and 52% performance improvement as compared to a random method. The second case study, which using AADMLSS, indicates a 10% performance improvement as compared to a load based method. Future work is focused on handling the case with no training set and using performance predictions for queue wait time predictions.

Acknowledgement

This work is supported in part by the following NSF ITR grants (EIA-0085952, ACI-0086044 and ANI-0225642).

6. References

- [1] GEO 600 homepage, <http://www.geo600.uni-hannover.de/>.
- [2] Grid Physics Network (GriPhyN), <http://www.griphyn.org/>.
- [3] Grid2003, <http://www.ivdgl.org/grid2003/>.
- [4] International Virtual Data Grid Laboratory (iVDGL), <http://www.ivdgl.org/>.
- [5] Laser Interferometer Gravitational Wave Observatory (LIGO), <http://www.ligo.caltech.edu/>.
- [6] Particle Physics Data Grid (PPDG), <http://www.ppdg.net/>.
- [7] TeraGrid, <http://www.teragrid.org/>.
- [8] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, K. Blackburn, A. Lazzarini, A. Arbre, R. Cavanaugh, and S. Koranda, Mapping Abstract Complex Workflows onto Grid Environments, *Journal of Grid Computing*, vol. 1, no. 1, pages 25-39, 2003.
- [9] M. Massie, B. Chun, and D. Culler, The Ganglia Distributed Monitoring System: Design, Implementation, and Experience, *Parallel Computing*, vol. 30, issue 7, 2004.
- [10] M. Prajugo, Performance-directed Site Selection System of AADMLSS, *Master's Thesis, Department of Computer Science, Texas A&M University*, 2004.
- [11] N. Parks, T. Simmons, K. Sapp, and J. Gilbert, Culturally Influenced E-Learning: An Introduction to AADMLSS, *Proceedings of E-Learn World Conference on E-Learning in Corporate, Government, Healthcare & Higher Education*, pages 1960-1965, 2003.
- [12] P. Dinda, A Prediction-based Real-time Scheduling Advisor, *Proceedings of the 16th International Parallel and Distributed Processing Symposium (IPDPS2002)*, 2002.
- [13] R. Gibbons, Historical Application Profiler for Use by Parallel Schedulers, *Lecture Notes on Computer Science*, pages 58-75, 1997.
- [14] The European Data Grid, <http://eu-datagrid.web.cern.ch>.
- [15] V. Taylor, X. Wu, and R. Stevens, Prophecy: An Infrastructure for Performance Analysis and Modeling of Parallel and Grid Applications, *ACM SIGMETRICS Performance Evaluation Review*, vol. 30, issue 4, March 2003.
- [16] W. Smith, I. Foster, and V. Taylor, Predicting Application Run Times Using Historical Information, In *Proceedings of the IPPS/SPDP '98 Workshop on Job Scheduling Strategies for Parallel Processing*, 1998.
- [17] X. Wu, V. Taylor, and R. Stevens, Design and Implementation of Prophecy Automatic Instrumentation and Data Entry System, *the 13th IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS2001)*, 2001.
- [18] X. Wu, V. Taylor, J. Geisler, and R. Stevens, Isocoupling: Reusing Coupling Values to Predict Parallel Application Performance, *the 18th International Parallel and Distributed Processing Symposium (IPDPS2004)*, 2004.
- [19] E. Badidi, R. Keller, P. Kropf and V. Dongen, Dynamic Server Selection in Distributed Object Computing Systems, *Distributed Computing on the Web (DCW'98)*, 1998.
- [20] S. Agrawal, J. Dongarra, K. Seymour, and S. Vadhiyar, NetSolve: Past, Present, and Future - A Look at a Grid Enabled Server, *Making the Global Infrastructure a Reality*, Berman, F., Fox, G., Hey, A. eds. Wiley Publishing, 2003.
- [21] F. Hao, E. Zegura and M. Ammar, "Supporting server selection in differentiated services networks", *IEEE Infocom 2001*.
- [22] D. Kalaiarul and M. Collier, Transparent and Scalable Client-side Server Selection using Netlets, *IEEE International Conference on Open Programmable Network Architectures, IEEE-OPENARCH' 2003*, 2003.