Node Selection for Placement of Migratory Tasks in Wide-Area Shared Computing Environments

Vinit Padhye, Devdatta Kulkarni, and Anand Tripathi [*]
University of Minnesota, Minneapolis MN 55455

Abstract

In a wide-area shared computing environment such as the PlanetLab the available resource capacities at nodes can fluctuate significantly. A node selected based on its available resource capacities for executing a task may become unsuitable for it within a few minutes or hours timeframe. This motivates the need to find another node and relocate the task. In this paper we present a comparative evaluation of different strategies for selecting nodes in the PlanetLab environment for the placement and relocation of migratory tasks. We have developed a system for monitoring PlanetLab nodes for their available resource capacities. Using this system we characterize the distribution of the number of nodes that meet a given resource capacity requirement, and the distribution of the duration for which a node satisfies the requirement.

1 Introduction

Scheduling of tasks in a large-scale wide-area shared computing environment such as the PlanetLab [1] presents several challenging problems. The available resource capacities at a host can fluctuate significantly in such environments as shown by the study presented in [3]. That study indicates that typically the available resource capacity at a host may change significantly within an hour. A host selected to execute a task with some given resource availability considerations can become overloaded during the task execution due to the load placed by other applications. The work presented in [3] argued for the need of supporting dynamic relocation of tasks based on resource availability. In an earlier work [6] we developed mechanisms for autonomic relocation of tasks in the PlanetLab environment for building resource-aware migratory services using the mobile agent based programming model [5]. We implemented mechanisms for a task, programmed as a mobile agent, to monitor its host execution environment and autonomically migrate to another host if the available resource capacity at the current host falls below some required threshold. Issues related to the mechanisms for task migration in our environment have been presented in [6], and therefore we do not discuss them here.

The focus of this paper is on the investigation and evaluation of different approaches for selection of target nodes for placement and dynamic relocation of tasks with some given resource requirements. An application may consist of multiples tasks, which may have different resource capacity requirements. Such an application may need to schedule its tasks, and for each such task it needs to select a node that meets the task's resource requirements. Moreover, such a task needs to be able to relocate itself autonomically to a different node when the current host fails to meet its resource requirements.

There are two problems that need to be addressed for selecting a node for dynamic placement and relocation of a task. The first problem is to identify the set of nodes whose currently available resource capacities satisfy the requirements of the task. We refer to such a set of nodes as the *eligibility set* for a given requirement. In our investigation we were interested in finding the distribution of the eligibility set size for a given resource capacity requirement because it determines the probability of finding a suitable target host when a task is to be scheduled or relocated. The *eligibility period* of a node is defined as the contiguous period for which it remains in the eligibility set. The expected duration for which a node selected randomly from the eligibility set, at an arbitrary point in time, would satisfy a task's requirement is half of the expected value of the eligibility period for that requirement. In this paper we study the distribution of eligibility periods and eligibility set sizes exhibited by a large collection of PlanetLab nodes observed over several days for a range of resource requirements. The second problem is to select one of the members in the eligibility set as the

[*] This work was supported by National Science Foundation grants 0834357 and 0708604

target host for the placement of the task to be scheduled or relocated. Our focus here is on the investigation of several alternate strategies for the placement of tasks on eligible nodes.

In the context of the first problem, the requirements of a task could be stated in terms of CPU capacity, memory, and bandwidth. To assist in the selection of nodes for the placement and autonomic relocation of tasks in the PlanetLab environment, we have developed a service for continuous monitoring of PlanetLab nodes for their available resource capacities. One can query this service to obtain the set of eligible nodes that satisfy a given resource requirement. We have used this service to study the behavior of PlanetLab nodes in terms of their eligibility periods and eligibility set sizes for a spectrum of resource requirements. The distribution of eligibility periods indicates how long a randomly selected node is likely to meet the given requirement of a task. The expected value of the eligibility set size is an indicator of the average number of tasks of a given resource capacity requirement that can be scheduled in the system.

In [6] we presented our observations of eligibility periods and eligibility set sizes solely based on CPU capacity requirements. We present here the results of our study of node availability behavior when both CPU and memory requirements are considered together for identifying the eligibility set. Our investigation was driven by the following key questions: How does the eligibility periods of the nodes and the size of the eligibility set are affected when both CPU and memory requirements are considered together in comparison to the behavior when only the capacity requirement for CPU or memory is considered in isolation? Second, we wanted to investigate whether any particular dimension of resource requirement, i.e. CPU or memory, dominates. Third, we also wanted to determine how the eligibility set size for a given resource capacity requirement varies over time.

In regard to the second problem mentioned above, we observed that typically the eligibility set for a given resource requirement contains nodes with a significant variation in their available capacities. There are various approaches for selecting a node as a target host from the set of eligible nodes, considering that the nodes may have varying levels of available resource capacity. For example, one may select a node that has the largest idle capacity available, or another option is to select a node whose available capacity is closest to the task's requirement.

The node selection and task placement strategies mentioned above can be viewed in some sense as analogous to memory allocation schemes such as *best-fit* and *worst-fit*. While selecting the target host, we may also need to consider, whether two or more tasks of the same application can be placed on one host if it satisfies the cumulative resource requirements of those tasks. We refer to this as *colocation*. For some applications, the colocation of tasks may not be a suitable option. For example, in case of a replicated service, it may be required that no two service replicas be placed on the same host. The policy of whether to allow colocation of tasks can affect the behavior of the node selection and placement strategies. In this paper we present and evaluate four different strategies for node selection and placement, and evaluate them under the policies of allowing and disallowing colocation of tasks.

In the next section we address the problem of identifying eligible nodes for a given requirement. We present the data and analysis of node eligibility periods and the distribution of eligibility set sizes for different capacity requirements for CPU and memory. In Section 3 we present different approaches for selecting a target node from the set of eligible nodes. We present here the comparative evaluation of these approaches for different application loads and requirements.

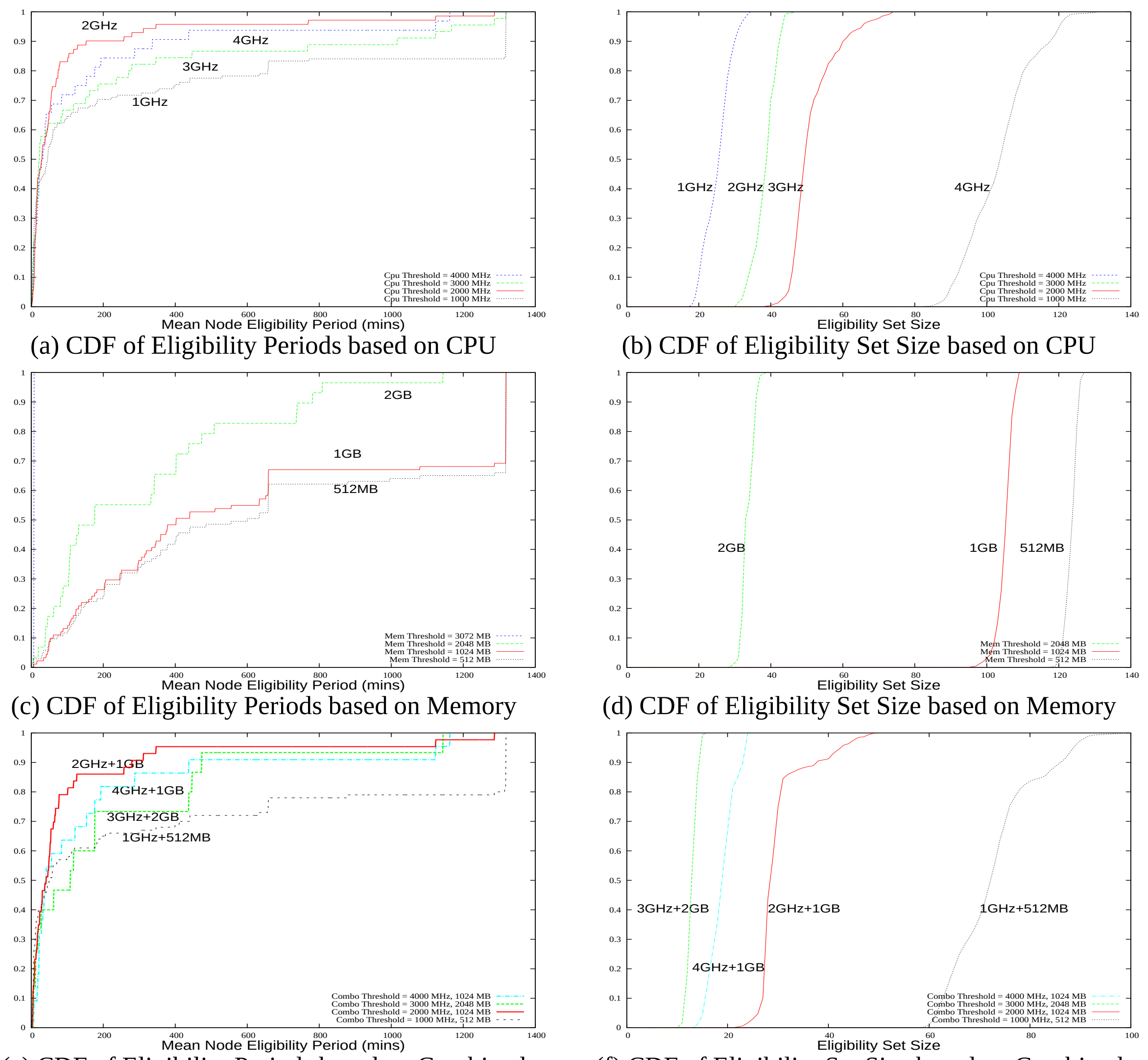
2 Monitoring of PlanetLab Nodes for Eligibility Sets

In this section, we describe the methods of identifying the set of nodes which satisfy the given resource requirement of a task, and present the results of our study of node availability for different resource requirements.

In order to find a target host for relocation and placement of a task, we need to first identify the eligibility set for a given resource requirement. For this purpose we have developed a service for monitoring PlanetLab nodes for their resource utilization. This monitoring service collects the data about resource consumption of every monitored node by probing its *SliceStat* [4] data every 10 seconds in order to obtain an accurate estimate of the node's resource utilization behavior over time. For selecting the nodes for inclusion in the eligibility set for the given requirement for a resource (such as CPU or memory), we consider the average value of a node's idle capacity for that resource over 5 minutes. We also take into account the standard deviation of a node's idle capacity over 5 minutes to mitigate the effects of fluctuations in the node's idle capacity. If C is the average idle capacity on a node and δ is its standard deviation, then for a given resource requirement R we select the node if

CPU	1GHz, 2GHz, 3GHz, 4GHz
Memory	512MB, 1GB, 2GB, 3GB
CPU+Memory	(1GHz + 512MB), (2GHz + 1GB) (3GHz + 2GB), (4GHz + 1GB)

Table 1: Capacity requirements used in experiments



(a) CDF of Eligibility Periods based on CPU

(b) CDF of Eligibility Set Size based on CPU

(c) CDF of Eligibility Periods based on Memory

(d) CDF of Eligibility Set Size based on Memory

(e) CDF of Eligibility Periods based on Combined

(f) CDF of Eligibility Set Size based on Combined

Figure 1: Cumulative distributions of Eligibility Periods and Eligibility Set Size for CPU, Memory, and Combined Requirements

it satisfies the following condition:

$$C - 2 * \delta > R \quad (1)$$

A node is dropped from the eligibility set if the idle capacity at that node falls below the resource requirement R. When considering the CPU and memory requirements together for selecting nodes, we select a node only if it satisfies the above condition for both CPU and memory requirement. We drop a node from the eligibility set, if either the available CPU capacity or available memory capacity on that node falls below the corresponding requirement given by R, the requirement threshold. The eligibility period of a node for a given resource requirement is measured as the time between the node's entry in the eligibility set for that resource requirement and its departure from the eligibility set. A node may enter and leave the eligibility set multiple times during the observation period. Thus a node may have multiple values of eligibility periods. For such nodes, we consider the average value of their individual eligibility periods. In [6], we have presented the data in terms of the measures discussed above for CPU requirements only. In this paper, we study the behavior of node availability for memory requirements as well as conjoined requirements for CPU cycles and memory. The first goal of the study presented here was to compare the distribution of eligibility periods for CPU requirements with that of memory requirements. The key questions in this study were: How the size of eligibility set varies over time? How does the behavior of nodes vary for CPU and memory requirements? Do the

	Dataset-1 (75 hours – Nov 18-21, 2009)						Dataset-2 (97 hours – December 1-4, 2009)					
	Eligibility Period (minutes)			Unique Nodes	Eligibility Set Size		Eligibility Period (minutes)			Unique Nodes	Eligibility Set Size	
	Avg	Median	Std Dev		Avg	Std Dev	Avg	Median	Std Dev		Avg	Std Dev
1GHz CPU	315	46	472	138	103	9.29	522	145	874	134	64	15.3
2GHz CPU	103	34	221	74	51	6.24	367	50	553	92	35	6.4
3GHz CPU	218	31	376	47	38	3.15	423	356	412	54	30	4.02
4GHz CPU	163	40	284	35	25	3.72	799	359	1362	30	21	3.0
1GB Memory	650	438	494	109	105	2.7	1061	1022	578	105	84	8.4
2GB Memory	335	284	281	39	34	1.75	910	787	564	35	20	5.0
2GHz+1GB	119	48	256	50	30	4.8	392	53	552	61	20	5.06
3GHz+2GB	218	108	305	16	13	1.0	518	577	502	11	5	1.7

Table 2: Eligibility Period and Set Size Statistics

nodes show more availability in terms of larger eligibility period and eligibility set size for memory requirements than those for CPU requirements? The second goal of this study was to determine whether the node availability is dominated by either the CPU requirement or the memory requirement, when both CPU and memory requirements are considered together.

In these experiments we monitored about 200 PlanetLab nodes for their available resource capacities at different periods over the past one year. Table 1 shows the capacity requirements used in these experiments. Figure 1 shows the CDFs of eligibility periods and the eligibility set sizes for the dataset observed for 75 hours during November 18–21, 2009. Table 2 presents statistics such as average eligibility period and set size for the two datasets: Dataset-1 is the one mentioned above and Dataset-2 is observed for 97 hours during December 1–4, 2009. During the period for which the Dataset-1 was collected, the monitored PlanetLab nodes were highly loaded while in the case of Dataset-2 they were relatively lightly loaded. Both the datasets correspond to same set of 200 monitored nodes.

From Table 2, we observe that for most of the requirements typically the median values for the eligibility periods tend to be less than the average values. The standard deviation also tends to be high, comparable to the average values. This indicates that some nodes tend to exhibit significantly large eligibility periods. This also indicates that the available resource capacities at a node may fluctuate significantly, and there is a large variation of eligibility periods across the nodes. In this table, the *Unique Nodes* column gives the number of nodes that became eligible during the entire duration of the observation. The CDF graphs in Figure 1(a, c, e) and the statistics in Table 2 for a requirement correspond to the unique nodes for that requirement. For example, in case of 2GHz CPU requirement the average eligibility period of 103 minutes is the average of 74 nodes' average eligibility periods. Similarly, in Figure 1(a) the CDF for 2GHZ is for the average eligibility periods of 74 nodes, whereas for 3GHz the distribution given is for 47 nodes.

Comparing the eligibility periods for CPU and memory requirements in Figure 1(a, c) and Table 2 we observe that typically nodes show high eligibility periods for memory requirements, as indicated by their average and median values. For example the median value for 1GB requirement is 438 minutes and that for 2GB is 284 minutes. From Figure 1, we observe that the distributions of eligibility periods and eligibility set size for a combined CPU+Memory (with up to 2 GB memory) requirement tend to be close to the distribution for the corresponding CPU requirement. In these cases the eligibility of nodes for combined requirements was largely dependent on the availability of the idle CPU capacity. However, in all experiments we found that for high memory requirements (e.g. 3GB), extremely few nodes were eligible. Also, in some other experiments for datasets observed in January 22 through February 1, 2010, we found that for 2GB requirement the median values for the eligibility periods sharply declined to 7-8 minutes. In such cases, the eligibility periods in the combined requirements were dominated by memory.

From the CDFs for the eligibility set sizes in Figure 1(b, d, f) and the data in Table 2, we observe that for a given requirement the eligibility set size varies very little. There is always some constant number of nodes that can satisfy a given requirement. For example, in case of the 4GHz CPU requirement there were always more than 18 nodes available, and for 2GHz at least 36 nodes were in the eligibility set. This is an indicator of how many tasks of a given requirement can be successfully scheduled in the system. We also find that the eligibility set sizes decrease with the increasing capacity requirements. However, one cannot draw such a generalization for eligibility periods, which in some cases increase for larger requirements. We found that in those cases fewer nodes became eligible but they remained in the set for a long time.

3 Node Selection and Task Placement

In this section, we present and evaluate different strategies for selecting a node from the eligibility set as the target host for a given task. There are different approaches for selecting a node for hosting a given task based on the available resource capacity of that node and the task's requirement. The selection of target host also depends on the colocation policy. Based on these considerations, we developed and evaluated the following four strategies for selecting a node for task placement.

1. Highest-Available: In this strategy the highest available capacity node is selected from the eligibility set for the given requirement. The motivation behind this strategy is that selecting the node with the highest available capacity maximizes the likelihood that the node will continue meeting the resource requirement of the given task for a long period of time, thereby reducing the number of times the task may need to be relocated. On the other hand, picking the node that has the highest available capacity for a low requirement task may not be always a good option because it may render that node ineligible for tasks with higher capacity requirements. This will lead to migration failures of high requirement tasks. This strategy can be considered analogous to the *worst-fit* strategy in memory management.

2. Lowest-Available: In this strategy, a node is selected from the set of eligible nodes which has the lowest available resource capacity. This strategy helps avoiding the problem noted above for the *Highest-Available* strategy. A disadvantage of this strategy is that placing a task on the node with available capacity closest to the task's requirement may lead to frequent migrations of the task. This strategy can be considered analogous to the *best-fit* strategy in memory management.

3. Lowest-with-Slack: This is a variation of the *Lowest-Available* strategy. In this strategy a node is selected from the eligibility set with the available capacity closest to the task's requirement with some additional slack. Thus if a task's requirement for CPU capacity is 2GHz and the slack is 500MHz, then the node with available capacity closest to 2.5GHz is selected. The motivation behind this strategy is that by selecting a node with some additional available capacity the likelihood that the task would need to be relocated is reduced.

4. Random: The approach here is to select the target node randomly from the eligibility set. Because the capacity of the node selected by any strategy can fluctuate significantly, we wanted to find how the random selection would perform in comparison to the others.

The performance of these strategies can be measured in mainly two aspects: the first measure is the number of times a task needs to be migrated because the current host no longer satisfies the task's requirement, and the second measure is the time for which a node meets the task's requirement. We define the time between the placement of task on a node and it's relocation to another node as the *residency time* of that task. A task may fail to relocate itself to another node because of not finding any eligible node, or in case of the no-colocation policy if all the eligible nodes are already hosting some tasks of the same application. If a task fails to migrate, it would still continue its execution on the current node, however, that node would no longer fully satisfy the task's requirement. If the task fails to migrate, it makes periodic attempts to migrate till it is successful or the current node becomes eligible again. We call the fraction of the residency time for which the node is satisfying the task's requirement as the *goodness factor*. The performance of the placement strategies can be evaluated in terms of the following measures: (1) Number of migrations, (2) Average residency time of tasks, and (3) Goodness factor.

We present here our evaluation of these strategies based on the above measures. The mobile agent-based task migration system [6], which we have implemented over the PlanetLab, could be used to implement and evaluate the strategies discussed above. However, we wanted a comparative evaluation of these strategies, and that required evaluating them for exactly the same conditions of node resource availability. We could not directly use the prototype system to evaluate these strategies because the evaluation experiments would interfere with each other if conducted simultaneously using the prototype system, In order to circumvent this problem, we used the trace-driven simulation approach. We developed a simulator of the prototype system, implementing the details of its task migration and node selection policies. We integrated the placement strategies in the simulator. We used the data collected by our monitoring service to perform a trace-drive simulation of various placement strategies.

We present here the results of these simulations and our observations about the behavior of different placement strategies for the two representative datasets. Table 3 shows how different strategies performed on the Dataset-1 and Dataset-2. In this experiment we simulated execution of 60 mobile agent-based migratory tasks, comprising of 20 tasks for each of the three requirements of 2GHz, 3GHz, and 4GHz CPU capacity. We used the task colocation policy. Since the study presented in Section 2 indicates that memory availability tends to be high, we considered only CPU capacity requirements in these simulations.

From the results shown in Table 3 for colocation policy, we observe that for Dataset-1 all four strategies performed comparably, however, the *Random* strategy shows slightly better performance. Dataset-1 corresponds to the period when the load on the monitored nodes was high. For the Dataset-2, which corresponds

Dataset-1 (75 hours – November 18-21, 2009)									
	2GHz Requirement Migratory Tasks			3GHz Requirement Migratory Tasks			4GHz Requirement Migratory Tasks		
Placement Strategies	T hrs:min:sec	N	Goodness Factor	T hrs:min:sec	N	Goodness Factor	T hrs:mins:sec	N	Goodness Factor
Highest-Available	0:29:22	147.8	0.99	0:35:2	114.2	0.92	0:52:34	79.2	0.89
Random	0:38:31	107.2	0.98	0:38:30	110.3	0.92	0:57:38	75	0.92
Lowest-Available	0:24:25	179.2	0.98	0:41:58	103.5	0.94	0:55:19	76	0.87
Lowest-with-Slack	0:36:29	117.2	0.99	0:36:50	115.2	0.82	0:35:48	122	0.81
Dataset-2 (97 hours – December 1-4, 2009)									
Highest-Available	34:42:1	1.75	0.99	36:8:58	1.6	0.99	31:00:0	2.15	0.99
Random	4:53:22	20.5	0.99	10:12:2	8.3	0.99	8:40:38	10.2	0.99
Lowest-Available	0:35:19	183	0.99	2:27:41	36	0.99	2:24:49	36	0.99
Lowest-with-Slack	8:11:2	10.7	0.99	2:48:10	34.8	0.99	3:59:21	23.5	0.99

Table 3: Performance of Placement Strategies (T – Avg Residency Time per task; N – Avg Migrations per task)

to a lightly loaded environment, the *Highest-Available* strategy clearly outperformed the others. Because in case of Dataset-1 the performance difference between *Random* and *Highest-Available* is not much, we recommend that the latter can be used in all cases.

We also simulated the same workload under the no-colocation policy and observed that no single strategy performed consistently better than the others. However, for tasks with high capacity requirements the *Highest-Available* strategy performed better than the others, whereas for tasks with low capacity requirements the *Lowest-Available* strategy performed the best. This conforms with the motivation behind these two strategies, as for tasks with low capacity requirements it is desirable to select the node with closest available capacity, while for tasks with higher capacity requirements selecting the node with highest capacity is a more desirable option.

4 Related Work

The CoMon project [4] has investigated monitoring of PlanetLab nodes for their resource consumption. CoMon provides node-level statistics such as the number of active slices, per slice utilization of CPU, memory, and bandwidth. A number of research projects have analyzed this data for characterizing the resource utilization [3, 2]. The work in [2] presents statistical methods for resource discovery and for characterization of nodes based on their resource usage. The focus of the work in [3] was mainly on the characterization of resource availability of the PlanetLab nodes. In [7], analysis of the CoMon data is presented for characterizing node failures and availability. In contrast to these previous works, our focus is on online monitoring and selection of PlanetLab nodes for dynamic placement and relocation of tasks.

5 Conclusion

Our study finds that the number of nodes available for a given requirement does not vary much. The available CPU capacity tends to have more variation as compared to available memory. The eligibility periods for CPU requirements tend to have significantly smaller median values as compared to their average values, and their standard deviations tend to be greater than the averages. For combined requirements with 1 GB memory needs, we found that node eligibility was mainly determined by the CPU capacity needs. In our evaluation of different placement strategies, we find that under the colocation policy the *Highest-Available* strategy outperforms others in a lightly loaded environment. In a heavily loaded environment all strategies perform equally well. Therefore, one can use the *Highest-Available* strategy in all cases.

References

[1] BAVIER, A., BOWMAN, M., CHUN, B., CULLER, D., KARLIN, S., MUIR, S., PETERSON, L., ROSCOE, T., SPALINK, T., AND WAWRZONIAK, M. Operating System Support for Planetary-scale Network Services. In *NSDI'04: Proc.of the 1st Symp. Networked Systems Design and Implementation* (2004), pp. 19–19.

[2] CARDOSA, M., AND CHANDRA, A. Resource Bundles: Using Aggregation for Statistical Wide-Area Resource Discovery and Allocation. In *The 28th IEEE International Conference on Distributed Computing* (June 2008), pp. 760–768.

[3] OPPENHEIMER, D., CHUN, B., PATTERSON, D., SNOEREN, A. C., AND VAHDAT, A. Service Placement in a Shared Wide-Area Platform. In *ATEC '06: Proceedings of the Annual Conference on USENIX '06 Annual Technical Conference* (Berkeley, CA, USA, 2006), USENIX Association, pp. 26–26.

[4] PARK, K., AND PAI, V. S. CoMon: A Mostly-scalable Monitoring System for PlanetLab. *SIGOPS Oper. Syst. Rev.* (2006), 65–74.

[5] TRIPATHI, A., KARNIK, N., VORA, M., AHMED, T., AND SINGH, R. Mobile Agent Programming in Ajanta. In *International Conference on Distributed Computing Systems (ICDCS'99).*

[6] TRIPATHI, A., PADHYE, V., AND KULKARNI, D. Resource-Aware Migratory Services in Wide-Area Shared Computing Environments . In *Proceedings of the IEEE Symposium on Reliable Distributed Systems (SRDS'09)* (2009), pp. 51–60.

[7] WARNS, T., STORM, C., AND HASSELBRING, W. Availability of Globally Distributed Nodes. In *Proceedings of the IEEE Symposium on Reliable Distributed Systems* (2008), pp. 279–284.