Peer-to-peer Discovery of Computational Resources for Grid Applications

Adeep S. Cheema, Moosa Muhammad, and Indranil Gupta

Abstract—**Grid applications need to discover computational resources quickly, efficiently and scalably, but most importantly in an expressive manner. An expressive query may specify a variety of required metrics for the job, e.g., the number of hosts required, the amount of free CPU required on these hosts, and the minimum amount of RAM required on these hosts, etc. We present a peer-to-peer (p2p) solution to this problem, using structured naming to enable both (1) publishing of information about available computational resources, as well as (2) expressive and efficient querying of such resources. Extensive traces collected from hosts within the Computer Science department at UIUC are used to evaluate our proposed solution. Finally, our solutions are based upon a well known p2p system called Pastry, albeit for Grid applications; this is another step towards the much-needed convergence of Grid and p2p computing.**

Index Terms—**Application Scheduling, Grid Computing, Peer-to-peer, Range Queries**

I. INTRODUCTION

GRID computing has come a long way since its inception [1]. One of today's revolutionary fields, the Grid is a vast service for sharing resources such as computational power and data storage for typically high-end applications. Grid computing has an expanding user group that includes physicists, bio-informaticians, astronomers, geologists, the medical community etc. BioGrid, for instance, is a project for developing a Grid specialized in biology and medical science [2].

Grid computing is characterized by its innovative applications and high performance orientation. Grid deployments facilitate large scale, flexible, secure and coordinated resource sharing among a dynamic collection of virtual organizations such as individuals, institutions and independent resources. This is in fact the anatomy of the Grid [3].

Peer-to-Peer (p2p) computing is another active research area that shares several computing interests with the Grid.

A. Cheema was with the University of Illinois – Urbana-Champaign, Urbana, IL 61801 USA. He is now with Microsoft Corporation, Redmond, WA 98052 USA. (e-mail: cheema@engineering.uiuc.edu).

M. Muhammad was with the University of Illinois – Urbana-Champaign, Urbana, IL 61801 USA. He is now with Motorola, Inc. (e-mail: mmuhamma @motorola.com)

I. Gupta is with the Computer Science Department, University of Illinois – Urbana-Champaign, Urbana, IL 61801 USA. (e-mail: indy@cs.uiuc.edu).

However, these communities have remained separate for a while due to different user communities and different research foci. P2p computing, although hampered by a lack of legitimate applications, has acquired a strong footing in certain avenues such as search and storage scalability, decentralization, fault tolerance, anonymity etc. [6]. These avenues are becoming increasingly relevant for Grid computing as it continues to evolve.

The goal of this research is to design a protocol for resource discovery in Grid applications. There are a few well-established characteristics that effective resource discovery protocols for Grid applications should demonstrate. The following characteristics have served as guidelines for the proposed protocol:

• Robustness – Grid applications often desire precision and accuracy. A robust protocol should locate resources when they are present and meet the expectations of the application.

• Scalability – A typical Grid deployment has a vast number of resources and an effective resource discovery protocol must be scalable.

• Efficiency – The protocol should locate resources while consuming a minimal amount of bandwidth. Resources on the Grid have several dynamic properties and a resource discovery scheme should be able to handle this volatility.

• Practicality – Grid applications often specify resources based on multiple criteria and arbitrary scales. These aspects should be addressed in a practical, deployable solution.

This paper is organized as follows. Section II presents related work, Section III models a Grid while Section IV presents a DHT-based scheme for resource discovery. Section V contains experimental data and finally we conclude with Section VI.

II. RELATED WORK

This paper primarily explores two threads of existing work including former research on resource discovery as well on multiple attribute queries.

Initial Grid papers [1][3] describe a version of the Grid that uses GRIP, A Grid Resource Information Protocol to register resources on centralized Grid Index Information Servers. Virtual computing toolkits such as the Globus Toolkit also rely on centralized means of storing resource advertisements. Such centralized protocols, like Matchmaking, have been successfully deployed on Grid like environments such as Condor.
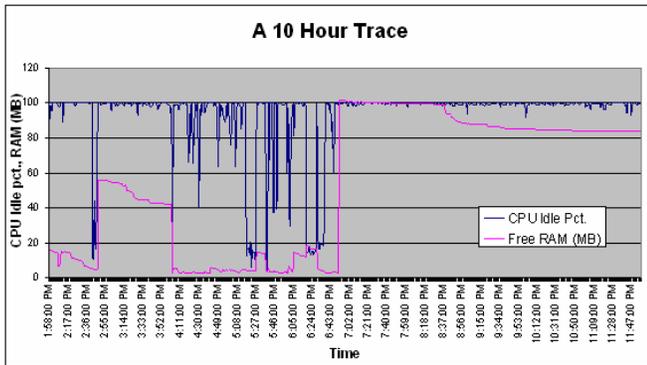
Fig. 1. A 10 Hour Trace: This plot depicts a sample 10 hours trace taken from an NFS server.
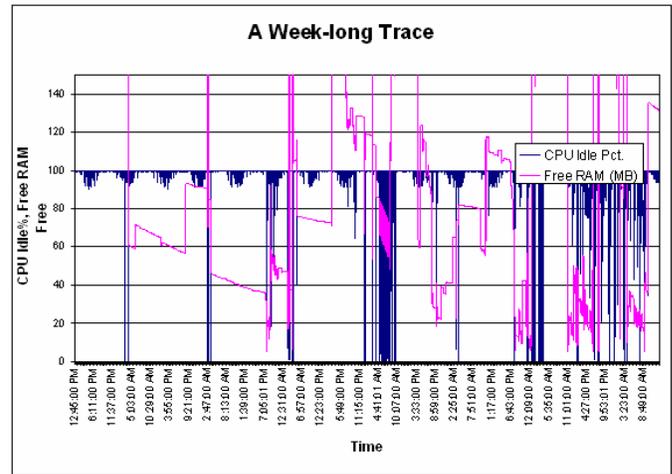

Fig. 2. A Weeklong trace: This plot is from a weeklong trace of a Linux workstation and depicts several key characteristics including (bi-diurnal) temporal patterns, maintenance patterns, intermittent high load, low average load etc.

Several resource discovery schemes have also been suggested for specific solutions and environments that tend to rely on unique properties of the domain. Finally, ideas on the convergence of the Grid and p2p system relevant to resource discovery have been put recently.

[7] proposes a modified version of Chord for distributed resource indexing on the Grid incorporating some interesting Distributed Hash Table (DHT) layer optimizations.

Since DHTs only support efficient single keyword lookups, a naïve approach to resolving a range query is to issue separate point queries to nodes that correspond to each possible value within the query range. This approach becomes quite expensive for a typical sized range query. Efficiently supporting range queries in DHT-based systems has been posed as an open problem with no solution. [8] proposes an adaptive protocol to address this problem in this domain. It utilizes Range Search Trees for content registration and query resolution. Each level of the RST corresponds to a different data partitioning granularity. Registrations are aggregated at different levels to facilitate queries with different range lengths. Queries are decomposed to a small number of sub-queries for efficient resolution. This scheme is conceptually similar to the Prefix Hash Tree (PHT) mechanism of supporting range queries in [9], with the difference of providing multiple levels to store contents instead of just using the leaf nodes for this purpose. The authors of [10] investigate the issue of extending P2P-based DHT systems such as CAN to allow range queries by using Space Filling Curves as hash functions.

III. MODELING A GRID

A. Observing Workstations

Prior to designing a resource discovery protocol, it was critical to analyze the nature of these resources in order to represent and model them effectively. This goal was achieved by actively monitoring 6 candidate machines for a period exceeding 3 weeks in an effort to model the availability of individual resources these workstations. This duration amounts to more than 3000 machine hours of activity. These measurements, taken every minute, will be referred to as traces

or availability traces through this paper. The experiments presented in Section V are driven by these very traces.

The candidate machines selected included an NFS server, 2 multimedia cluster workstations and 3 other workstations present in the Computer Science domain at the University of Illinois. These machines demonstrate workload characteristics similar to workstations on distributed computational Grids. These characteristics include low average load, relatively short periods of high load interspersed with almost negligible load (typical of batch jobs), temporal patterns as well as maintenance related patterns.

Data collected from these machines gives a minute-by-minute representation of CPU usage, RAM availability and disk space availability for each machine. Figure 1 is a snippet of this data, representing a 10 hour span on one particular workstation.

B. Analysis and Conclusions

Resource availability traces were helpful in establishing several key facts that became central to the design of the protocol. Firstly, it was evident that in order to best represent a resource on the Grid, we would have to accurately and efficiently store what the machine has to offer in terms of idle CPU cycles, megabytes of RAM and gigabytes of disk space. Out of these parameters, CPU utilization tends to be very dynamic and bursty as can be seen in Figure 2. Despite the bursts of high-utilization, the average load on the CPU is less than 4%, which suggests that these spare cycles could be harvested for Grid applications. Disk space on the flip side is fairly stable and does not fluctuate much. The trend for disk space, not plotted, is best described as linear with a small negative slope, with occasional small positive jumps. Available RAM lies in the middle of the spectrum and varies a reasonable amount with time. We end up giving it less importance than CPU availability however because all modern

TABLE I
STATIC AND DYNAMIC ATTRIBUTES OF WORKSTATIONS

Static Part	Dynamic Part
OS Configuration	CPU Idle %
CPU Speed	Available RAM
RAM	Available Disk Space
Total Disk Space	

workstations have virtual memory, which increases the amount of RAM available on demand.

This analysis forms the basis of the resource discovery protocol presented in the next Section.

IV. P2P RESOURCE DISCOVERY

A. Distributed Hash Tables

Distributed Hash Tables are systems that allow key based insertion, lookup and deletion of objects in a distributed setting. Such peer-to-peer DHTs form the basis of several applications such as file sharing, storage, multiplayer games etc. [38] Pastry [4] is one such DHT that has been adapted for several different applications. Each node in Pastry has a unique identifier (Node ID). When presented with a message and a key, a Pastry node efficiently routes the message to the node with a Node ID that is numerically closest to the key, among all currently live Pastry nodes, which can be visualized as a ring. Each Pastry node keeps track of its immediate neighbors in the Node ID space, and notifies applications of new node arrivals, node failures and recoveries. Pastry takes into account network locality and is completely decentralized, scalable, and self-organizing; it automatically adapts to the arrival, departure and failure of nodes. The expected number of routing steps is $O(\log N)$, where N is the number of Pastry nodes in the network.

B. The Naming Problem

Any workstation on a distributed computational Grid can be summarized using two sets of attributes shown in Table I.

The naming problem consists of mapping resources to points on a DHT given that each resource is represented by a collection of attributes. Moreover, the static part attributes of a resource description are discrete and infinite while dynamic attributes are continuous and thus inherently lie on an infinite scale.

The tactic adopted by the suggested protocol involves combining the static and dynamic part of a resource's attributes into a Resource ID, which serves as a key for DHT operations. Moreover, this encoding should allow for the efficient and practical discovery of these mapped resources.

C. Representing Resources

Both the static and dynamic parts of resource attributes can be translated into a format more suitable for indexing. This does come at the cost of reduced precision although the specific trade-off is variable and can be adapted depending on requirements.

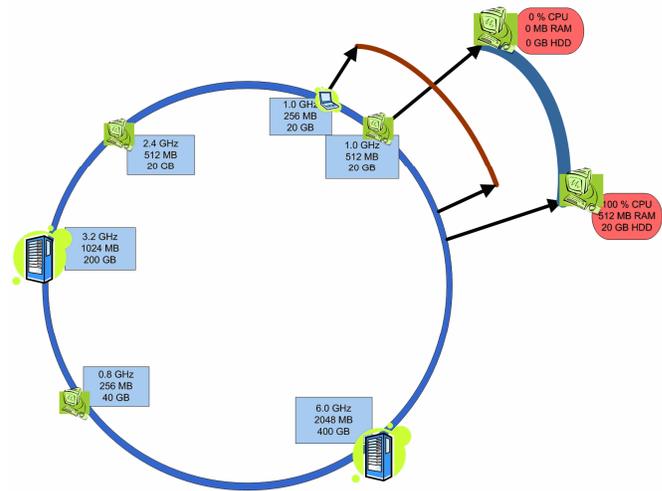Tables II and III depict sample representations for both sets

Fig. 3. Ring Mappings: This is a pictorial depiction of the proposed protocol. The blue ring represents the DHT and neon green Computers represent workstation on the Grid. Blue rectangles (light gray in B&W) are static attributes and red ovals (dark gray in B&W) are dynamic attributes.

TABLE II
ENCODING OF STATIC WORKSTATION ATTRIBUTES

Attribute	Bits	Range	Resolution
OS Configuration	8	0-255	1
Max CPU Speed	7	0.1-12.6 GHz	100 MHz
Max RAM	10	16-16352 MB	16 MB
Max Disc Space	7	10-1260 GB	10 GB

TABLE III
ENCODING OF DYNAMIC WORKSTATION ATTRIBUTES

Attribute	Bits	Quantum
CPU Idle %	16	0.006 %
RAM Free %	10	0.097 %
Disk Free %	6	1.58%

of resource attributes for our proposed system. The suggested encoding of the static part of resource attributes represents the set of workstation configurations most likely to be encountered in current Grid deployments. Dynamic attributes have the property that they are continuous. They can however be discretized by representing them as a percentage using a specific number of bits per attribute which results in the encoding and precision shown in Table III. This representation is based on the trends seen in workstation traces from Section III. It is safe to encode dynamic characteristics as percentages simply because the percentage value can be combined with the static representation of the workstation to arrive at the exact status of the resource.

D. An Abstract Description

The fundamental concept behind the suggested resource discovery protocol lies in exploiting the scalability and efficient indexing capabilities of peer-to-peer distributed hash tables. Given the static attributes of a resource, our goal is to place that information at a point on the ring representing the DHT. Given that a part of this description is dynamic, each different state of availability of a certain machine must be
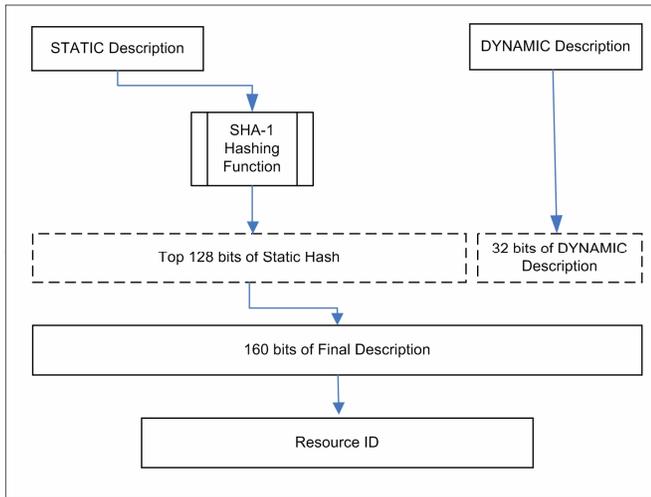
Fig. 4.   Arriving at a Node ID: Attribute descriptions translate into a Resource ID.

Fig. 5. An Insertion Screenshot.

Fig. 6. An Update Screenshot.

Fig. 7. A Query Screenshot.

placed on the ring as well. In the proposed system, this is realized by representing resources as potentially overlapping arcs instead of as individual points on the DHT ring. The beginning of each such arc represents a resources static attribute set and the length of the arc signifies the spectrum of dynamic states the very resource can exist in based on the availability of individual attributes.

A key issue here is that the DHT ring contains only a finite number of nodes while there are an infinite number of dynamic attribute configurations. This is however addressed by the provided encoding scheme. Another notable fact is that the nodes comprising the DHT can have an arbitrary relation with the actual machines on the Grid. Essentially, each machine on the Grid can choose to host anywhere from 0 to a large number of virtual nodes providing complete flexibility.

This abstract model together with the concrete representation of workstation resources provides the framework for the proposed protocol.

E. Initialization and Maintenance

1. Insertions

Insertion of resources into the suggested discovery protocol has two basic requirements. Firstly, this data should be as uniformly distributed as possible over distributed hash table. This allows for greater scalability. Secondly, each static resource configuration should be represented as an arc representing various degrees of attribute availability.

These requirements are realized by hashing the 32 bit static part of resource attributes (using a SHA-1 hashing function) resulting in a 160 bit hash. The entire 32 bits of the dynamic part of resource attributes are then appended to the top 128 bits of this static hash. As depicted in Figure 4, the result is a 160 bit Resource ID that fulfills both prior criteria and can thus be mapped to a specific node on the p2p distributed hash table. It is possible that the derived Resource ID may not map to a particular node. In that case the underlying DHT routing scheme, such as Pastry described earlier in Section IV, automatically routes to the node numerically closest to the

calculated Resource ID.

Insertions are carried out with the help of an INSERT message, which is routed from the node in the p2p network attempting to insert a resource to the node that has a Node ID closest to the calculated Resource ID based on the attributes of resource. Figure 5 shows how a resource workstation with its particular static and dynamic attributes maps to a Resource ID given by <0x3FFF2B..> in a 160 bit node namespace.

2. Updates

Updates are an essential aspect of the protocol given the volatile nature of dynamic resources attributes. Updates are initiated by resources either periodically at rate URATE or when they observe a significant change, parameterized in our system as an array UCHANGE[], which specifies, as an absolute percentage, how much each dynamic attribute must change for a resource to force an update. For instance, if UCHANGE[CPU] is 8.7 then a resource workstation will only issue an update if its CPU's idle percentage varies by at least 8.7%.

Updates are implemented with using an UPDATE message, which is routed from an arbitrary node in the p2p ring to a freshly computed Resource ID. Figure 6 shows a sample UPDATE issued by a workstation due to the change in available RAM from 18.96% to 6.94% of its maximum.

It is possible for this update scheme to create stale data in case there is churn in the system or if a resource maps to a different node after an update. This can be avoided by having nodes periodically flush resource entries that have not been updated recently or by sending REMOVE messages to prior node mappings for each UPDATE message.

3. Queries

Searching for computational resources through the protocol is message based, and essentially involves locating the node on the DHT that is currently hosting the desired resource, expressed in terms of static and dynamic attributes. There are several search heuristics and a particular one should be picked

depending on the requirements of the Grid application. Figure 7 depicts two sample (single-shot) queries.

3.1 Single-shot Searching

This search heuristic involves calculating the Resource ID of the desired resource based on the protocol's attribute encoding and hashing function. This Resource ID is then used to route to a node (with the closest Node ID), which issues a REPLY if it has information about the desired resource. If the update frequency of dynamic resource attributes is large enough, most single searches will work correctly. A high frequency of updates can also overcome the effects of churn.

Singe-shot searching is desirable when the Grid application implements local strategies for searching. A Single-shot search would then query for a particular kind of resource and report if the search was successful or not giving the calling application the freedom to take any subsequent action.

3.2 Recursive Searching

Recursive searching is a TTL (time to live) restricted search that continuously seeks out nodes likely to know about the desired resource by progressively tuning search parameters at each hop. The layout of resources on the DHT together with the format of Resource ID's permits the tuning of the least significant bits of the address, representing the dynamic attributes of the resource. Such tuning can help locate resources, which may not match exactly, but are close approximations of the original requirements.

Recursive searching is slightly more efficient than single-shot searching since it uses a distributed/global search strategy and does not report back to the application at every failed attempt thus saving on return messages. Its functionality can however be simulated with multiple single-shot queries.

3.3 Parallel Searching

For most Grid applications, it is acceptable to exceed the requirements requested in a query in case an exact match cannot be found. Parallel searching is a formidable strategy that can be used to hasten such alternative searching if so desired by the application.

Parallel searching simply involves initiating multiple searches in addition to a basic search for the exact requested parameters. The additional searches are spawned with the intention of seeking resources that have better attributes than what is actually requested by the application. Parallel searching cuts down on response time when there are a limited number of resources or high contention.

V. EXPERIMENTS AND RESULTS

Experimental results based on software simulations of the suggested resource discovery protocol are presented in this Section. The protocol simulation is layered over the FreePastry implementation of the Pastry DHT.

The trace-based simulation engine is highly parameterized and offers a number of benchmarks. A typical simulation starts

TABLE IV
SIMULATION PARAMETERS

Number of Nodes	Number of Resources	UCHANGE[] (%)	Query Rate
1000	20000	{10, 10, 10}	1

TABLE V
STATIC RESOURCE ATTRIBUTES

Attribute	Bits	Distribution
OS Configuration	8	Uniform over entire range
Max CPU Speed	7	Uniform over 0.1-3.6 with 1% outliers.
Max RAM	10	Uniform over [32, 64, 128, 256, 512, 1024] with 1% outliers.
Max Disk Space	7	Uniform over 10-400 GB with 1% outliers.

by creating a set of nodes, NUM_NODES, which comprise the DHT required by the protocol. NUM_MACHINES parameterizes the number of resources, represented as computer workstations that are introduced into the system, after being read from an ASCII file. The simulation runs for a period in minutes given by NUM_MINUTES. Queries are performed on the system at a rate specified by QUERYRATE, which is a number specifying the duration between queries in minutes. Queries arrive at a constant rate unless specified otherwise. Update frequency for resources is parameterized using URATE, which is the number of minutes a resource will wait before issuing an update and UCHANGE[] which holds the percentage change required for forcing an update for each of the dynamic attributes of the resources i.e., CPU Idle %, RAM Free % and Disc Free %. Both these update parameters have been previously described in Section IV.

Table IV contains the default values for simulation parameters, which have been used throughout the following experiments unless specified otherwise.

Additionally, the simulation uses an extensive set of workstation resource availability traces introduced in Section III. These traces determine how the dynamic attributes of resources in the system change. The static parts of resource attributes are generated with realistic assumptions. The distributions for these static attributes are provided in Table V.

A. Update Frequency

Updates have a significant impact given the dynamic nature of resource availability. Figure 8 shows how the number of updates varies with the UCHANGE[] parameter (introduced in Section III). The clustering of points drops rapidly as UCHANGE[] increases and is linear after the 10% point on the X-axis which suggests that any value that is of UCHANGE[] that is 10% or greater will exhibit better performance. This result can be verified from the availability traces which depict how most fluctuations in availability have magnitude less than 10%.

Number of updates is linear after the 10% point when plotted against the update threshold parameter. The 10% update threshold here implies that an update was triggered if
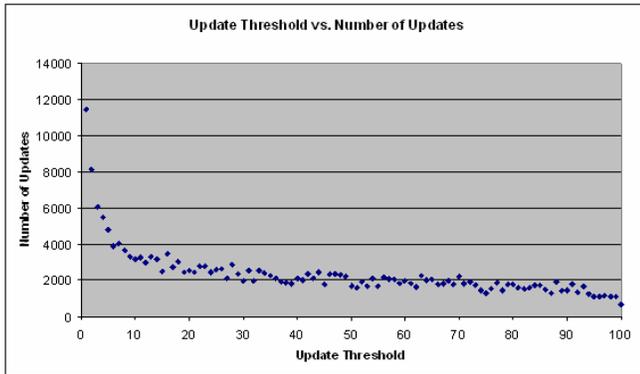
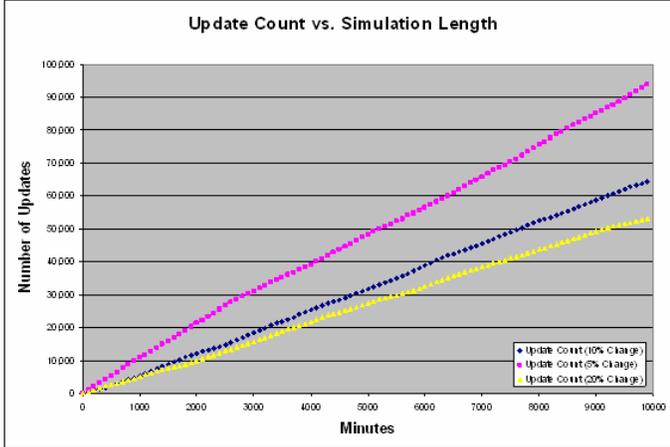Fig. 8. Update Threshold vs. Number of Updates.

Fig. 9. Update Count vs. Simulation Length. The rate of updates is constant.
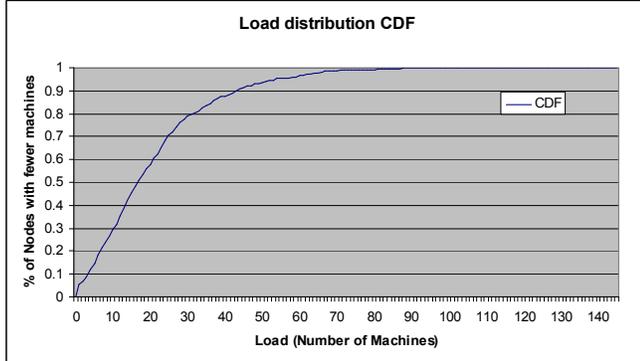

Fig. 10. Load CDF: CDF of load over DHT nodes.

any dynamic attributed experienced an absolute 10% change.

Figure 9 shows how the rate of updates is constant and independent of the length of the simulation. The large difference in slope between the lines representing 5% and 10% update thresholds ties in with Figure 8.

B. Scalability

Figure 10 shows a CDF demonstrating how nodes are hashed based on combined static and dynamic parts of their attributes in the given scheme. This plot is based on static attributes of 20,000 machines where each individual attribute is based on the realistic distributions from Table V. The somewhat asymmetric nature of the CDF can be attributed to these distributions, which drastically bring down the number of possible combinations of static resource attributes. This
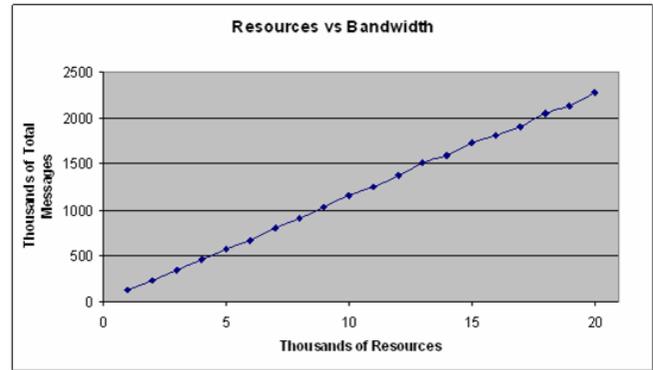

Fig. 11. Resources vs. Bandwidth: Number of messages increases linearly with number of resources.
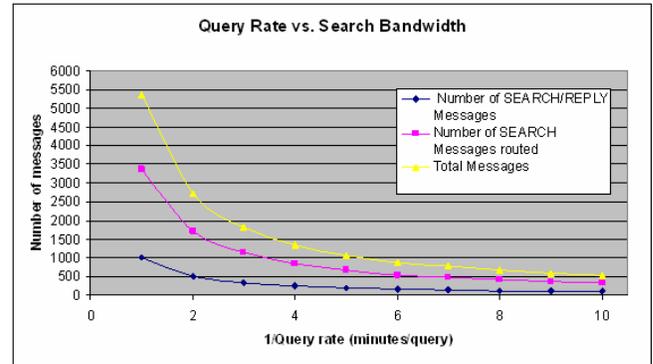

Fig. 12. Query Rate vs. Search Bandwidth: The number of search messages falls sharply with an increasing query rate

consequently results in a skewed distribution of nodes. For instance, 30% of the nodes show in the simulation stored 70% of the data. It is also important to note that that the worst-off node (with the maximum amount of load) has 0.55% absolute load as opposed to an ideal 0.1%.

Figure 11 demonstrates how the bandwidth consumed by the system scales linearly with the number of resources injected into the DHT.

C. Searching

Figure 12 shows how the amount of search related bandwidth varies with the inverse of query rate i.e., with the duration between queries. These experiments suggest that the system does not deteriorate with increased rates of querying. Additionally, the number of search messages increases linearly with the number of queries.

D. Availability: Short-term Queries

Figure 13 shows the relationship between the desired quality of a resource and its availability. Each point on the graph is the result of a single-shot search, which returns the number of machines that match the dynamic CPU attributes from the X-axis for a randomly chosen static description. Due to the skewed distribution of machines over the DHT, these single-shot searches are comprehensive i.e., they return all machines of that static configuration, which also meet the dynamic criteria. The percentage of these nodes is plotted on the Y-axis.
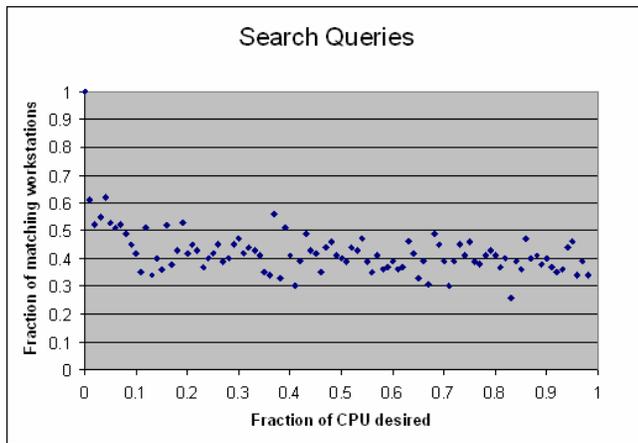
Search Queries

Fig. 13. Search Queries: Cluster plot representing queries plotted with respect to the number of resources that were fetched together with the quality of resources desired.
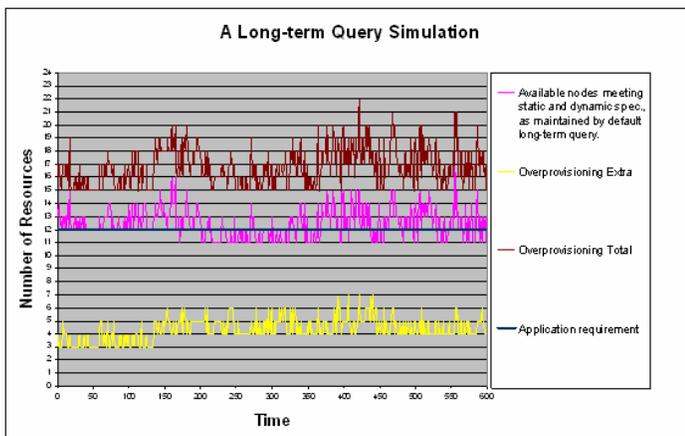
A Long-term Query Simulation

Fig. 14. A Long-term Query: This ten hour simulation shows how an application requiring a certain number of resources can perform over-provisioning. The application meets its quota by performing parallel searches for resources that exceed its requirements.

E. Availability: Short-term Queries

Figure 14 shows a ten hour simulation of a Grid application using the suggested protocol for resource discovery. The application has a static requirement of 12 nodes of a certain specification shown as the dark straight line on the plot. It tries to meet its requirement by conducting a search on the desired static and dynamic specification. This simulation shows that if that particular resource is limited, then it is possible that the application may not meet its requirement. In this case it is ideal for the application to overprovision i.e., seek out higher quality resources. A combination of true matches and better matches can then be used to satisfy the requirements of the application.

VI. CONCLUSION

The resource discovery protocol introduced in this paper supports expressive queries which allow searches on multiple machine attributes and continuous values of these attributes. Trace-based experiments show that the suggested protocol is rigorous, scalable, efficient and practical. This work provides a complete framework for resource discovery in Grid applications.

Possible future directions include studying a deployment of the suggested protocol and addressing the skewed distribution of resources over nodes perhaps by incorporating a load balancing scheme.

REFERENCES

[1]   I. Foster, The Grid: A New Infrastructure for 21st Century Science, Physics Today, Vol. 55 #2, p. 42, 2002
[2]   The Biogrid Project, http://www.biogrid.jp/
[3]   I. Foster, C. Kesselman and S. Tuecke, The Anatomy of the Grid Enabling Scalable Virtual Organizations, International Journal on Supercomputer Applications, 15(3), 2001.
[4]   A. Rowstron and P. Druschel, Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems, IFIP/ACM International Conference on Distributed Systems Platforms 2001, pp. 329-350
[5]   J. Ledlie, J. Shneidman, M. Seltzer, et al. et al, Scooped, again, In Proceedings of International Workshop on Peer-to-Peer Systems 2003, pp. 129-138
[6]   I. Foster and A. Iamnitchi. On death, taxes and the convergence of peer-to-peer and grid computing, In Proceedings of International Workshop on Peer-to-Peer Systems 2003, pp. 118-128
[7]   Y. Meng et al, A DHT-Based Grid Resource Indexing and Discovery Scheme, Singapore-MIT Alliance Annual Symposium 2005
[8]   J. Gao and Peter Steenkiste. An Adaptive Protocol for Efficient Support of Range Queries in DHT-based Systems. In Proceedings of The IEEE International Conference on Network Protocols 2004
[9]   S. Ratnasamy, J. Hellerstein, and S. Shenker. Range Queries over DHTs. Technical Report IRB-TR-03-009, Intel Corp., 2003
[10] A. Andrzejak and Z. Xu. Scalable, Efficient Range Queries for Grid Information Services. In Proceedings of the IEEE Conference on Peer-to-Peer Computing 2002
[11] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao, OceanStore: An Architecture for Global-Scale Persistent Storage, In Proceedings of the Ninth international Conference on Architectural Support for Programming Languages and Operating Systems 2000..