A Scalable, Robust, and Decentralised Resource Discovery Service for Large Scale Federated Grids

Rajiv Ranjan, Lipo Chan, Aaron Harwood, Rajkumar Buyya, Shanika Karunasekera
P2P Networks Group and GRIDS Laboratory
Department of Computer Science and Software Engineering
The University of Melbourne, Victoria, Australia
{rranjan,lipoc,aharwood,raj,shanika}@csse.unimelb.edu.au

***Abstract*— Efficient Resource discovery mechanism is one of the fundamental requirement for Grid computing systems, as it aids in resource management and scheduling of applications. Resource discovery involves searching for resources that match the user's application requirements. Various kinds of solutions to Grid resource discovery have been developed, including the centralised and hierarchical information server approach. However, these approaches have serious limitations in regards to *scalability, fault-tolerance and network congestion*.**

To overcome such limitations, we propose a decentralised Grid resource discovery system based on a spatial publish/subscribe index. It utilises a Distributed Hash Table (DHT) routing substrate for delegation of d-dimensional service messages. Our approach has been validated using a simulated publish/subscribe index that assigns regions of a d-dimensional resource attribute space to the grid peers in the system. We generated the resource attribute distribution using the configurations obtained from the Top 500 Supercomputer list. The simulation study takes into account various parameters such as resource query rate, index load distribution, number of index messages generated, overlay routing hops and system size. Our results show that grid resource query rate directly affects the performance of the decentralised resource discovery system, and that at higher rates the queries can experience considerable latencies. Further, contrary to what one can expect, system size does not have a significant impact on the performance of the system, in particular the query latency.

I. Introduction

Recently, Internet-scale services including distributed resource brokering [10], distributed gaming, content distribution networks, peer-to-peer (P2P) storage, and distributed auctions have received significant research interest both from researchers and industry. Concurrently, resource sharing platform such as grids [9] and PlanetLab [6] have emerged as the defacto means for hosting these distributed services. One of the main challenge involving a planetary scale deployment of these services is locating the appropriate set of nodes that match the service's requirement.

An efficient resource discovery mechanism is a mandatory requirement of Grid systems (such as desktop grids, resource brokers, work-flow engines), as it aids in resource management and scheduling of applications. Traditionally, Grid resource brokering services such as Nimrod-G [3], Condor-G [10], Tycoon [15], Grid workflow engine [7] used services of centralised and hierarchical information services (such as R-GMA [25], Hawkeye [24], MDS-2,3,4 [8]). However, these approaches have several design limitations including: (i) single point of failure; (ii) lack scalability; (iii) high network communication cost at links leading to the information server (i.e. network bottleneck, congestion); and (iv) computational power required to serve a large number of resource lookup and update queries on the machine running the information services. Recent studies conducted by Zhang et al. [25] verified that existing systems including R-GMA, MDS, and Hawkeye fail to scale beyond 300 concurrent users i.e. the throughput begins to decline below acceptable levels. As regards to response time performance metric, MDS-2 performs the worst, superseded by R-GMA and Hawkeye.

Several grids (e.g., APACGrid [1], TeraGrid, ChinaGrid, and CoreGrid) have been setup in different countries to serve e-science applications. The APAC (Australian Partnership for Advanced Computing) Grid interconnects various grid sites distributed across Australian institutions and universities. The APACGrid uses a hierarchical information service MDS-2. The VPAC (Victorian Partnership for Advance Computing) which a part of the APACGrid hosts the centralised GIIS (Grid Index Information Service) (a component MDS-2), while the remaining grid sites run the GRIS (Grid Resource Information Service) that connects to the VPAC GIIS. A grid resource broker that wishes to access the APACGrid has to contact the VPAC GIIS, as contacting one of the other Grid sites running a GRIS would only allow access to information about that particular resource. The ChinaGrid is also organised using the hierarchical model. This isolation in resource information organisation among grids results in grid users getting access to only a small pool of resources. Further, the resource brokering services undertake scheduling decisions based on the resource information gathered from isolated indexing services. Hence, they often tend to formulate conflicting application schedules.

We expect that in near future the number of grid users and grid resources to continue to grow. In order to tackle this growth, we need to design scalable infrastructure solutions. We envisage decentralisation of grids as a viable way to realise an efficient grid computing infrastructure. Decentralisation can be accomplished through an Internet-wide grid resource lookup system along the same lines as the Domain Name Service (DNS). In other words, there is a need to build a scalable grid resource information service that will allow and promote all existing grid resources to combine together into a single cooperative system. Such a system would solve the problems associated with centralised or hierarchical organisation, resource fragmentation and conflicting application schedules. Fig. 1 shows such a Grid computing environment organisation based on a decentralised resource discovery system.

One of the possible ways to overcome the limitation of a centralised or hierarchical approach, is to partition the resource index space across the set of dedicated database servers [16]. For achieving fault-tolerance these database servers can be replicated across multiple machines. Further, the index space can be partitioned across servers based on attribute types

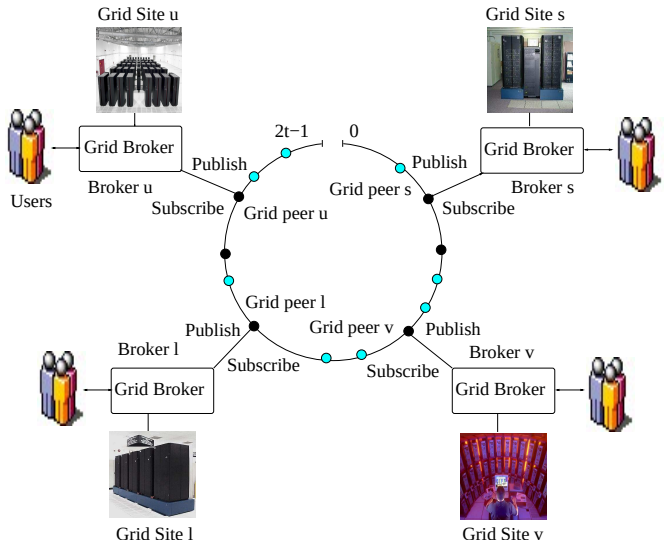


Fig. 1. Grid brokers and Grid sites with their Grid peer service and some of the hashings to the Chord ring. Dark dots are the Grid peers that are currently part of Chord based Grid network. Light dots are the publish/subscribe object posted by Grid sites and Grid brokering service.

and values. However one of the major drawbacks of this scheme is that satisfying a range query would require sending simultaneous messages to set of servers. This might prove costly in terms of the number of messages generated in the system. Further, if the number of users increases rapidly then upgrading the hardware infrastructure can prove to be an expensive process.

An other possible way to tackle this problem is to organise the index space using a Distributed Hash Table (DHT) method. In this case, commodity machines such as a desktops can be used to host the DHT and indexing services. DHTs are inherently self-organising, fault-tolerant and scalable. Further, DHT services are light-weight and hence, do not warrant an expensive hardware infrastructure. A majority of Google's data center services are hosted by the commodity machines, and this is a case in point.

In this work, we present a grid resource discovery service using the DHT-based spatial publish/subscribe index in [14]. The proposed grid resource discovery service organises data [18] by maintaining a logical d-dimensional publish/subscribe index over a network of distributed grid brokers/grid sites. These brokers create a Chord overlay [21], which collectively maintain the logical publish/subscribe index to facilitate a decentralised resource discovery process. We present more details about the publish/subscribe index in Section III. Fig. 1 depicts the proposed resource discovery system involving grid brokers and grid Sites (shown as dark coloured blocks on the Chord ring). Resource brokering services such as a Grid-Federation Agent (GFA) [17], Condor-G etc. issue a Resource Lookup Query (RLQ) by subscribing for a publication object that matches a user's application requirement. Grid resource providers update their resource status by publishing information at periodic intervals through a Resource Update Query (RUQ).

The main contributions of this work include: (i) extension of the DHTs with grid resource discovery capability; (ii) a decentralised grid resource discovery system based on a spatial and peer-to-peer publish/subscribe index; and (iii) extensive simulations for evaluating the feasibility and performance of the proposed resource discovery system. We now summarise some of our findings:

- The Resource query rate i.e. RLQ and RUQ rate directly affects the performance of the decentralised resource discovery system. At higher rates, grid resource queries can experience considerable latencies.
- Contrary to what one can expect, the system size does not have a significant impact on the performance of the system, in particular the query latency.

The rest of the paper is organized as follows. In section II, we present a brief overview of a Grid resource brokering service model and its indexing requirement. Section III presents details about the underlying d-dimensional publish/subscribe index that we leverage for this work. In section IV, we summarise the average message and routing hop complexity involved with routing of RLQ/RUQ objects. Section V presents the simulation model that we utilise for evaluating the performance of grid resource discovery system. In section VI, we present various experiments and discuss our results. Section VII summarises current state of art in resource discovery system design. We end this paper with summary and our future vision in Section VIII.

II. Grid resource brokering service and queries

A. Grid System Model

A Grid resource brokering service is one that performs: "scheduling of jobs across grid resources such as computational clusters, parallel supercomputers, and desktop machines that belong to different administrative domains". Brokering in computational grids is facilitated by specialized grid schedulers/brokers such as the Grid Federation Agent, Nimrod-G, Condor-G and workflow engine [7]. In general, a broker service requires two basic types of queries: (i) an RLQ, a query issued by a broker service to locate resources matching the user's application requirements; and (ii) an RUQ, is an update query sent to a resource discovery service by a grid site owner about the underlying resource conditions. Since, a grid resource is identified by more than one attribute, an RLQ or RUQ is always d-dimensional. Further, both of these queries can specify different kinds of constraints on the attribute values. If a query specifies a fixed value for each attribute then it is referred to as a *d-dimensional Point Query* (DPQ). However, in case the query specifies a range of values for attributes, then it is referred to as a *d-dimensional Window Query* (DWQ) or a *d-dimensional Range Query* (DRQ). In database literature, a DWQ or an DRQ is also referred to as a *spatial range query*.

In this paper, we consider a grid system model that aggregates distributed resource brokering and allocation services [17] as part of a generalised resource sharing environment, which is referred to as the *Grid-Federation*. The grid brokering model aggregates topologically and administratively separated computational grid resources such as clusters, supercomputers, and desktops. Resource brokering, indexing and allocation in the Grid-Federation is facilitated by a new Resource Management System (RMS) known as the Grid Federation Agent (GFA). More details about general Grid-Federation brokering, and the resource owner's local resource allocation services can be found in the articles [17].

TABLE I
NOTATIONS

Symbol	Meaning
n	number of Grid Federation Agents (GFAs)/peers in the Grid network.
c_i	resource access cost at GFA i.
p_i	number of processors at GFA i.
$u_{i,j}$	i^{th} user from j^{th} GFA/resource.
$J_{i,j,k}$	i-th job from the j-th user of k-th GFA.
$r_{i,j,k}$	an RLQ for $J_{i,j,k}$.
U_i	an RUQ for the i-th GFA/peer/resource.
$p_{i,j,k}$	number of processor required by $J_{i,j,k}$.
$b_{i,j,k}$	assigned budget to $J_{i,j,k}$.
$d_{i,j,k}$	assigned deadline to $J_{i,j,k}$.
ρ_i	resource utilisation for resource at GFA i.
x_i	processor architecture for resource at GFA i.
ϕ_i	operating system type for resource at GFA i.
λ^{in}	total incoming RLQ/RUQ arrival rate at a network queue i.
λ^{out}	outgoing RLQ/RUQ rate at a network queue i.
μ	average network queue service rate at a Grid peer i.
μ_r	average query reply rate for index service at GFA/peer i.
dim	dimensionality or number of attributes in the Cartesian space.
λ_u^{in}	incoming RUQ (publish) rate at a application service i.
λ_l^{in}	incoming RLQ (subscribe) rate at a application service i.
λ_a^{in}	incoming query rate at a Chord routing service i from the local application service.
K	network queue size .
M	random variable denoting number of of messages generated in mapping an RLQ or RUQ.
T	random variable denoting number of disjoint query path undertaken in mapping an RLQ or RUQ.
λ_{index}^{in}	incoming index query rate at a application service i from its local Chord routing service.

In general, compute grid resources have two types of attributes: (i) static attributes–such as the type of operating system installed, network bandwidth (both Local Area Network (LAN) and Wide Area Network (WAN) interconnection), processor speed and storage capacity (including physical and secondary memory); and (ii) dynamic attributes–such as processor utilization, physical memory utilization, free secondary memory size, current usage price and network bandwidth utilization.

Every GFA in the federation publishes its local resource information with the decentralised resource discovery system. An RUQ or a publish object consists of a resource description R_i, for a cluster i. Refer to Table- I for the notations and model parameters that we use in rest of the paper. R_i includes information about the CPU architecture, number of processors, RAM size, secondary storage size, operating system type, resource usage cost etc. In this work, $R_i = (p_i, x_i, \mu_i, \phi_i, \rho_i, c_i)$ which includes the number of processors, p_i, processor architecture, x_i, their speed, μ_i, their utilization, ρ_i, installed operating system type, ϕ_i, and a cost c_i for using that resource, configured by the site owner. A site owner charges c_i per unit time or per unit of million instructions (MI) executed, e.g. per 1000 MI. A GFA publishes the R_i into distributed resource discovery system by encapsulating it into an RUQ object, U_i.

A job in the Grid-Federation system is written as $J_{i,j,k}$, to represent the i-th job from the j-th user of the k-th resource. A job specification consists of the number of processors required, $p_{i,j,k}$, processor architecture, $x_{i,j,k}$, the job length, $l_{i,j,k}$ (in terms of instructions), the budget, $b_{i,j,k}$, the deadline or maximum delay, $d_{i,j,k}$, and operating system required, $\phi_{i,jk}$. A GFA aggregates these job characteristics including $p_{i,j,k}$, $x_{i,j,k}$, $\phi_{i,j,k}$ with a constraint on maximum speed, cost and resource utilization into an RLQ object, $r_{i,j,k}$ and sends it as a subscription object to the resource discovery system. More details about the job model can be found in the paper [17].

B. An Example RUQ and RLQ

Every GFA periodically sends an RUQ to the distributed resource discovery system. The publish, or resource update object includes a resource description set R_i:

Publish: Total-Processors= 100 && Processor-Arch="pentium" && Processor-Speed= 2 GHz && Operating-System = Linux && Utilization= 80 && Access-Cost=1 Dollar/min.

Note that, the above RUQ is a DPQ. However, an RUQ can also be compiled as a DRQ depending on a grid site configuration. As jobs arrive the GFAs (on behalf of the Grid-Federation users) issue an RLQ to the distributed resource discovery system to acquire information about active resource providers in the system. An RLQ has the following semantics:

Subscribe: Total-Processors $\geq$ 70 && Processor-Arch="pentium" && 2 GHz $\leq$ Processor-Speed $\leq$ 5GHz && Operating-System = Solaris && 0.0 $\leq$ Utilization $\leq$ 90 && 0 Dollar/min $\leq$ Access-Cost $\leq$ 5 Dollar/min .

III. P2P-BASED SPATIAL PUBLISH/SUBSCRIBE INDEX

In this section, we describe the features of the P2P-based publish/subscribe index that we utilise for our grid resource discovery system. Providing background work and details on this topic is beyond the scope of this paper; here we only give a high level picture.

In this work, we utilise the spatial publish/subscribe index proposed in the work [14]. The publish/subscribe index uses a logical d-dimensional domain space for mapping subscription and publication objects. The MX-CIF Quad-tree spatial hashing technique [19] is used to hash the logical d-dimensional index onto a DHT network.

The publish/subscribe index utilises a content-based approach. It builds a d-dimensional cartesian space based on the grid resource attributes, where each attribute represents a single dimension. The logical d-dimensional index assigns regions of space to the grid peers in the resource discovery system. If a grid peer is assigned a region (cell) in the d-dimensional space, then it is responsible for handling all the activities related to the RLQs and RUQs associated with the region. More details on this spatial hashing technique can be found in the article [22].

The cartesian space has a tree structure due to two types of division process, explained as follows:

A. Minimum division (f_{min})

This process divides the cartesian space into multiple index cells when the d-dimensional publish/subscribe index is first created. The cells resulted from this process remain constant throughout the life of the publish/subscribe domain and serve as entry points for subsequent RLQ (subscribe) and RUQ (publish) processes. The number of cells produced at the minimum division level is always equal to $(f_{min})^{dim}$, where dim is dimensionality of the cartesian space. Every grid peer in the network has basic information about the cartesian space coordinate values, dimensions and minimum division level.

B. Load division

This process is performed by the cells (at f_{min}) when their storage capacities are undermined by heavy RLQ workload. An overloaded cell subdivides itself to produce multiple child cells, which collectively undertake the workload. This is a dynamic process that is repeated by the child cells, if they also become overloaded. This growing process introduces the parent-child relationship, where a cell at level *m* is always a child of a particular cell at level *m-1*. To minimise the amount of information that needs to be known by the cells for correct routing, the parent-child relationship is limited at one level. It means that every cell only has a direct relationship with its child cells. Note that, the maximum depth (f_{max}) of the distributed index tree is curbed by constraining the load division process after a certain number of executions. Although such a constraint provides controllable performance benefits, it may lead to query load-imbalance in some cases.

C. Query mapping.

This action involves the identification of the cells in the cartesian space to map an RLQ or RUQ. For mapping RLQs, the search strategy depends whether it is a DPQ or DRQ. For a DPQ type RLQ, the mapping is straight forward since every point is mapped to only one cell in the cartesian space. For a DRQ type RLQ, mapping is not always singular because a range look-up can cross more than one cell. To avoid mapping a range RLQ to all the cells that it crosses (which can create many unnecessary duplicates) a mapping strategy based on diagonal hyperplane of the cartesian space is utilised. This mapping involves feeding an RLQ candidate index cells as inputs into a mapping function, F_{map}. This function returns the IDs of index cells to which given RLQ should be mapped (refer to Algorithm 1). Spatial hashing is performed on these IDs (which returns keys for Chord space) to identify the current grid peers responsible for managing the given keys. A grid peer service use the index cell(s) currently assigned to it and a set of known base index cells obtained at initialisation as the candidate index cells.

Similarly, the RUQ/publish process also involves the identification of the cell in the cartesian space using the same algorithm. An RUQ is always associated with an event region and all cells that fall fully or partially within the event region will be selected to receive the corresponding RUQ. The calculation of an event region is also based upon the diagonal hyperplane of the cartesian space.

Algorithm 1 Subscribing or publishing

1: *index cell(s)* = F_{map}(*candidate index cells*, subscription or publication)
2: **if** *index cell* is not null **then**
3: *ID* = spatial_hash(*index cell*)
4: **end if**
5: Lookup *grid peer* through Chord routing network based on *ID*, to either store the subscription or match the publication to stored subscriptions.

D. Query routing.

Using the query mapping policies, the resource discovery service searches for a cell (from minimum division) in the cartesian space that overlaps with area sought by an RLQ. When this cell is found, the service starts the RLQ mapping process by contacting the peer (in the network) that owns the cell. When the cell receives an RLQ, two cases are considered:

- In the first case, the cell has undergone a load division process and it routes the RLQ to the child cell that is responsible for the region in which the RLQ is mapped.
- In the second case, the cell has not undergone any load division process. Hence, there will be no further routing and the cell keeps the RLQ for future event notification.

IV. Message complexity and routing hop analysis

In this section, the complexity analysis for message and routing hop is presented. We denote the number of messages generated in mapping a DRQ by a random variable M. The distribution of M is function of the problem parameters including query size, dimensionality of search space, query rate, division threshold and data distribution. As the dimensionality increases, the order of the tree increases and each tree node has more children. If the height of the tree is kept constant, then increasing the cartesian space dimensions does not increase the maximum hop length. However, constraining the maximum height of the tree, may lead to load imbalance at some Grid peers. Note that, the derivation presented in this paper assumes that the Chord method is used for delegation of service messages in the network.

Essentially, a control point at the f_{min} level of the logical d-dimensional cartesian space can be reached in $\Omega(1/2 \log_2 n)$ routing hops with high probability (using the Chord method). Since each Grid peer at f_{min} level of the index tree controls its division with the child cells, therefore every control point owner can maintain a cache of IP address for its child cells. The child cells are created as a result of dynamic load division process. Hence, the number of routing hops required to delegate an index message beyond the f_{min} reduces to $O(1)$. However, under high churn conditions when the grid peer membership changes, the Chord stabilisation process and transfer of index keys delays the caching of IPs. During such periods the cache miss can occur and in this case the routing may have to be done using the standard Chord method. Though, we consider grid sites to be well provisioned and well connected to the Internet. Hence, we do not expect a highly dynamic behaviour (high join, leave, and failure rate) in contrast to the traditional P2P file sharing systems.

Based on above discussion, in order to compute the worst case message lookup and routing complexity one additional random variable T is considered. T denotes the number of disjoint query path undertaken in mapping an RLQ or RUQ. In the worst case, every disjoint query ends up at the maximum allowed depth of the tree i.e. f_{max}. Hence every disjoint path would undertake $\Omega(1/2\ log_2 n + f_{max} - f_{min})$ routing hops with high probability. Hence, the expected value of M is given by:

$$E[M] = \Omega(E[T] \times (1/2\ log_2\ n + f_{max} - f_{min}))$$

V. Simulation Model

In this section, we present simulation model for evaluating the performance of our resource discovery system. The proposed model is applicable to large networks of the scale of the Internet. The simulation model considers the message

queuing and processing delays at the intermediate peers in the network. In a centralised system, the index look-up latency is essentially zero, assuming the computation delay due to processing of local indices is negligible. For the peer-to-peer system, assuming negligible computation delay for index processing logic at intermediate peers, the time to complete an RLQ or RUQ is time for the query to reach all the cells (including both parent and child cells) that intersect with the query region.

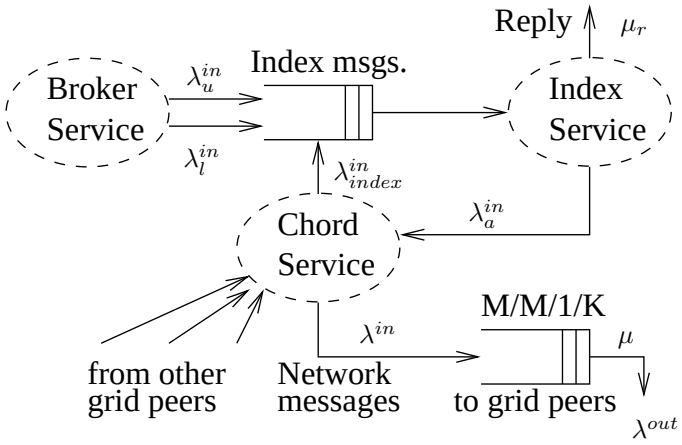


Fig. 2. Network Message queueing model at a Grid peer i

In our message queueing model, a grid peer node (through its Chord routing service) is connected to an outgoing message queue and an incoming link from the Internet (as shown in Fig. 2). The network messages delivered through the incoming link (effectively coming from other grid peers in the overlay) are processed as soon as they arrive. Further, the Chord routing service receives messages from the local publish/subscribe index service. Similarly, these messages are processed as soon as they arrive at the Chord routing service. After processing, Chord routing service queues the message in the local outgoing queue. Basically, this queue models the network latencies that a message encounters as it is transferred from one Chord routing service to another on the overlay. Once a message leaves an outgoing queue it is directly delivered to a Chord routing service through the incoming link. The distributions for the delays (including queueing and processing) encountered in an outgoing queue are given by the M/M/1/K [4] queue steady state probabilities.

Our simulation model considers an interconnection network of n grid peers whose overlay topology can be considered as a graph in which each peer maintains connection to a $\log_2 n$ other grid peers (i.e. the Chord overlay graph). As shown in Fig. 2, every grid peer is connected to a broker service that initiates lookup and update queries on behalf of the users and site owner. We denote the rates for RLQ and RUQ by λ_l^{in} and λ_u^{in} respectively. The queries are directly sent to the local index service which first processes them and then forwards them to the local Chord routing service. Although, we consider a message queue for the index service but we do not take into account the queuing and processing delays as it is in microseconds. Index service also receives messages from the Chord routing service at a rate λ_{index}^{in}. The index messages include the RLQs and RUQs that map to the control area currently owned by the grid peer, and the notification messages arriving from the the network.

VI. Experimental evaluation

In this section, we perform simulations to capture the interplay among various grid resource query and P2P network parameters and their contribution to the overall performance of grid resource discovery system.

A. Experimental setup

We start by describing the test environment setup.

1) Broker network and index simulation:: Our simulation infrastructure was modeled by combining two discrete event simulators namely *GridSim* [5], and *PlanetSim* [11]. GridSim offers a concrete base framework for simulation of different kinds of heterogeneous resources, services and application types. The core of GridSim is based on the *SimJava* [12], a discrete event simulation package.

PlanetSim is an event-based overlay network simulator. It can simulate both unstructured and structured overlays. However, in this work we utilise the services of the Chord implementation of the PlanetSim. To enable event time synchronisation between PlanetSim and GridSim, we modified the basic PlanetSim classes including *Node, Network and EndPoint* to extend the core *GridSim* class. We model the resource brokering service (i.e. a GFA inside the GridSim) that initiates RLQs and RUQs on behalf of users and resource providers. Every GFA connects to a local publish/subscribe index service which runs on a Chord node in the PlanetSim. Every instance of the index service in the network is responsible for managing and indexing the logical d-dimensional data. Our simulation considers message queueing delay, processing delay, and packet loss at the intermediate overlay Chord nodes.

2) Simulation configuration: This section explains the distributions for simulation parameters.

Network configuration: The experiments were conducted using a 32 bit Chord overlay i.e. 32 bit node and key ids. The network size, n, was fixed at 128 broker nodes/grid sites for Exp-1. In Exp-2, the system size is scaled from 100 to 500 in steps of 100. The network queue message processing rate, μ, at a grid peer was fixed at 500 messages per second. We vary the value for network message queue size, K, as 10^2, 10^3, and 10^4 in Exp-1. While in Exp-2, we fixed K to 10^4. In Exp-2 we basically simulate a large message queue size such that no message is dropped by the resource discovery system.

Query rate configuration: We vary the RLQ rate, λ_l^{in}, and RUQ rate, λ_u^{in}, from 1 to 100 queries per simulation second. At every step the RLQ rate is always equal to the RUQ rate. In Exp-2, the RLQ and RUQ rate are fixed at 1 query per second for different system sizes.

Publish/subscribe index configuration: The minimum division, f_{min} of the logical d-dimensional publish/subscribe index was set to 3, while the maximum height of the index tree, f_{max}, was also limited to 3. This means we basically do not allow the partitioning of index space beyond the f_{min} level. In this case, a cell at a minimum division level does not undergo any further division. Hence, no RLQ/RUQ object is stored beyond the f_{min} level. The index space resembles

a grid-like structure where each index cell is randomly hashed to a grid peer based on its control point value. The publish/subscribe cartesian space had 6 dimensions including resource access cost, c_i, processor speed, m_i, processor utilisation, ρ_i, processor architecture, x_i, and operating system type, ϕ_i. Hence, this configuration resulted to 729 (3^6) grid index cells at the f_{min} level. On an average, 7 index cells are hashed to a grid peer in a network comprising of 128 grid sites.

Indexed data distribution: We generated an uniform resource type distribution using the resource configuration obtained from the Top 500 Supercomputer list [2]. The list included 22 distinct processor types, so in our simulated grid resource index space, the probability of occurrence of a particular processor type is $1/22$. We utilised the resource attributes including processor architecture, its number, its speed, and installed operating system from the Supercomputer list. The values for c_i and ρ_i were fabricated. The values for c_i and ρ_i were uniformly distributed over the interval $[0, 10]$ and $[5, 80]$ respectively. Every RLQ was constrained such that it always subscribed for the operating system type, processor architecture, maximum number of processors required which was also available on the local site. An RLQ is thrashed from the system, once it matches with an RUQ. Following this, a match event notification is sent to the concerned broker service. A load of 200 RLQ and 200 RUQ objects is injected into the resource discovery system by a broker service over the simulation period during Exp-1. While in case of Exp-2 we configured a broker service to inject only 50 RLQ and 50 RUQ objects.

B. Effect of query rate

The first set of experiments measured the RLQ/RUQ query performance with an increasing incoming query rate across the grid peers in the broker network. We started from a RLQ/RUQ rate of 1 query per second and increased it till 100 queries per second. We configured the other input parameters as following n=128, f_{min}=3, $f_{max} = 3$, μ=500, and dim=6. All the broker nodes join the system at the same time, stabilise their finger tables and initialize their logical index space. Over the simulation period, we do not consider a grid peer join or leave activity. We identified six metrics to measure the RLQ/RUQ query performance including latency, $\%$ of successful RLQs, response time, routing hops, total number of messages generated for mapping RLQs/RUQs, and the total number of messages in the system over the simulation period. Measurements for parameters including latency, response time, routing hops is averaged over all the broker service in the system. While the measurements for the remaining parameters are computed by summing up their values across the broker services.

Fig. 3 and Fig. 4 show the plots for these parameters with an increasing query rate across the system. Fig. 3(a) shows results for the average RLQs/RUQs latency, Fig. 3(b) shows results for the $\%$ of successful RLQs and Fig. 3(c) shows the average response time for the RLQs across the system. These measurements were conducted for different values of network message buffer capacity i.e. K. Results show that for lower values of K (i.e. 10^2, 10^4) network drops significant number of messages (refer to Fig. 4(a)). Fig. 4(a) shows total number of messages generated in the system over the simulation period for different query rates and message queue sizes. Hence, for these message queue sizes successful RLQs/RUQs encounter compratively lower traffic hence leading to almost same latency (refer to Fig. 3(a)) and response time (refer to Fig. 3(c)). But the downside of this is that at higher rates significantly larger number RLQs are dropped by the system (refer to Fig. 3(b)). However, this is not true for the case when the network has higher buffering capability (i.e. K = 10^4), in this case the messages encounter significantly more traffic thus worsening the queuing and processing delays. Second, with a larger message queue size system experiences much higher query success rates (refer to Fig. 3(b)).

Our results state that the average number of routing hops for RLQs and RUQs remain constant irrespective of the query rate in the system (refer to Fig. 4(b)). The main reason for this is the same value for both the index tree depth parameters i.e. f_{min}=f_{max}. Thus, we do not allow the partition of index cell or load distribution between grid peers beyond f_{min}. With different query rates the height of the distributed index tree remained constant, hence leading to a similar number of routing hops. Fig. 4(c) shows the results for the total number of messages generated in the system for all RLQs/RUQs. As expected the number of messages generated for the RLQs/RUQs remained constant since the data distribution was same for all query rates.

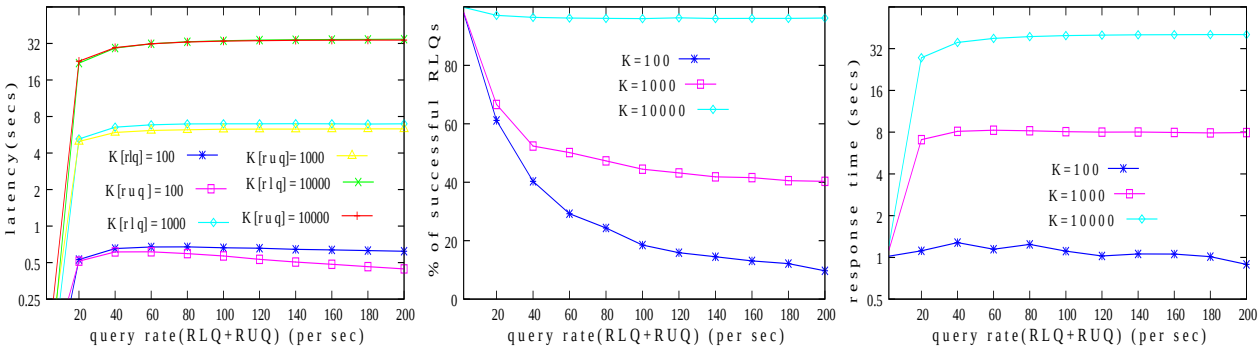
Thus it is evident that at higher query rates, the messages experience greater queuing and processing delays. This can be directly observed in the RLQ/RUQ latencies which have significantly larger values at moderately higher query rates. Further, resource discovery system performance is directly governed by the underlying network's message processing capability.

C. Effect of system size

In our second experiment, we examine the resource discovery system's scalability in terms of the number of participating Grid sites. We used the same resource distribution as before, but scaled it such that the probability of occurrences of particular resource types remained constant. We started from a system size of 100 and increased it till 500. We fixed the RLQ/RUQ rate to 1 query per second across the grid peers in the broker network. We configured the other input parameters as following f_{min}=3, $f_{max} = 3$, μ=500, K=10^4 and dim=6. All the grid peers join the system at the same time, stabilise their finger tables and initialize their logical index space. Over the simulation period, we do not consider a grid peer join or leave activity.

Fig. 5(a) shows the growth of the RLQ/RUQ latency as a function of increasing Grid network size. As expected, the query latencies do not increase significantly, because the growth rate of latency is a logarithmic function of the Grid network size n. That is on average an RLQ or RUQ encounters $\Omega(1/2 \log_2 n)$ grid peers before being finally mapped. Similarly, in Fig. 5(b) we observed that the number of routing hops undertaken RLQ/RUQ increased slightly with the system size. At the system size of 100, the RLQs/RUQs undertook 4.12 routing hops on an average. For a system size of 500, the average query path increased to 5.39 hops i.e. increased by about 30%.

Fig. 5(c) shows the results for the number of messages generated for RLQs/RUQs and Fig. 6 shows the results for the total number of messages generated as the system scaled

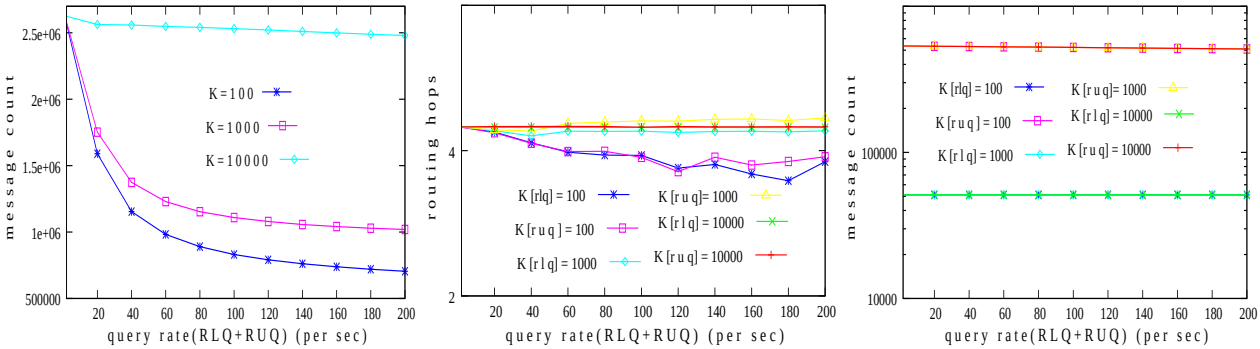


(a) query rate (RLQ + RUQ) (per sec) vs. lookup latency (secs)

(b) query rate (RLQ + RUQ) (per sec) vs % of successful RLQs

(c) query rate (RLQ + RUQ) (per sec) vs response time (secs)

Fig. 3. Simulation: Effect of query rate

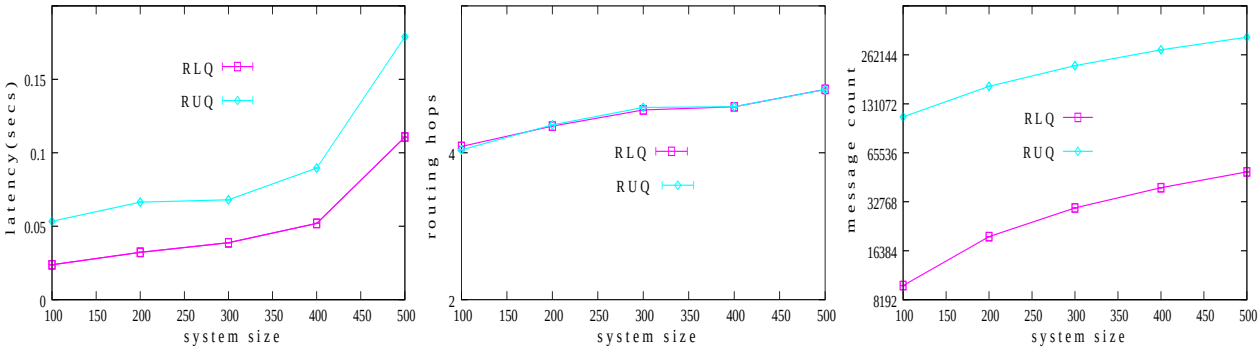


(a) query rate (RLQ + RUQ) (per sec) vs message count

(b) query rate (RLQ + RUQ) (per sec) vs. routing hops

(c) query rate (RLQ + RUQ) (per sec) vs. message count

Fig. 4. Simulation: Effect of query rate



(a) system size vs. lookup latency (secs)

(b) system size vs. routing hops

(c) system size vs. message count

Fig. 5. Simulation: Effect of system size

from 100 to 500 sites. As expected the number of messages generated for RLQs/RUQs increased with system size. A system compromising of 100 Grid sites produced 109007 RUQ messages, which increased to 336579.4 messages when the system scaled to 500 grid sites (refer to Fig.5(c)). We observed a similar growth for RLQ messages as well with an increase in the system size. The total messages generated (including RLQ and RUQ) increased significantly as the system scaled from 100 to 500 sites (refer to Fig. 6). Further, in this case we observed 575% increase in the total number of messages generated in the system.

We conclude that contrary to what one may expect, the grid system size does not have a significant impact on the performance of the resource discovery system, in particular the query latency and the number of message routing hops.

VII. Related work

The approach [13] involved a drawback of generating a large volume of network messages due to flooding. This system

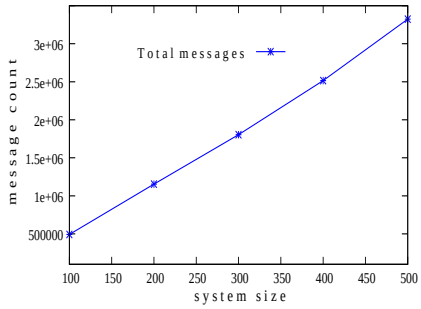


Fig. 6. system size vs message count

can not guarantee to find the desired resource even though it exists in the network due Time to Live (TTL) field associated with query messages. SWORD [16] system creates a separate search segment for each attribute and hence the query routing needs to be augmented with external techniques for resolving d-dimensional queries. In contrast, our resource discovery system utilises a spatial publish/subscribe index that hashes a d-dimensional index space to a 1-dimensional key space of Chord overlay. The publish/subscribe index does not require any additional query resolution and load-balancing heuristic. JXTA Search [23] does not apply any index for organising the distributed data. A cross-domain search operation in JXTA involves a query broadcast to all the advertisement groups using the query group membership information. The OurGrid system utilises JXTA for organising its brokering service. In contrast, our resource discovery is based on a deterministic routing substrate Chord. Our system does not require a broadcast primitive for data discovery in a grid network, hence is more efficient in terms of number of messages generated in the system. Squid [20] system applies Hilbert space filling curves for mapping a d-dimensional index space to a 1-dimensional key space. Squid maps these contiguous d-dimensional indices to the overlay key space. The approach has issues with index load balance which is fixed using external technique. In contrast, our proposed resource discovery system utilises a spatial publish/subscribe index that does not need any external load-balancing.

VIII. Summary and future work

In this paper, we presented a decentralised grid resource discovery system. It utilises a peer-to-peer spatial publish/subscribe index for organising d-dimensional grid resource data. We analysed experimentally how the query arrival rate and grid system size affects the system performance.

Currently, we are building the proposed resource discovery system using the FreePastry API. We intend to use the resource discovery system for organising our Alchemi-based desktop grid system. Alchemi is a .Net based implementation that enables coupling of desktop machines running flavors of the Windows operating systems. In future work, we plan to evaluate other spatial indices including Space Filling Curves, MX-CIF Quad-Tree, and R-Tree in organising the resource discovery system. Further, we also intend to develop a grid application scheduling coordination model based on a spatial publish/subscribe index.

References

[1] *APAC Grid http://grid.apac.edu.au/.*

[2] *Top 500 Supercomputer List. http://www.top500.org/.*

[3] D. Abramson, R. Buyya, and J. Giddy. A computational economy for grid computing and its implementation in the Nimrod-G resource broker. *Future Generation Computer Systems (FGCS) Journal, Volume 18, Issue 8, Pages: 1061-1074, Elsevier Science, The Netherlands, October*, 2002.

[4] A. O. Allen. *Probability, Statistics and Queuing Theory with computer science applications*. Academic Press, INC., 1978.

[5] R. Buyya and M. Murched. Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. *Journal of Concurrency and Computation: Practice and Experience;14(13-15), Pages:1175-1220*, 2002.

[6] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman. Planetlab: an overlay testbed for broad-coverage services. *SIGCOMM Comput. Commun. Rev.*, 33(3):3–12, 2003.

[7] T. Fahringer, A. Jugravu, S. Pllana, R. Prodan, C. Seragiotto, and H. Truong. Askalon: a tool set for cluster and grid computing: Research articles. *Concurr. Comput. : Pract. Exper.*, 17(2-4):143–169, 2005.

[8] S. Fitzgerald, I. Foster, C. Kesselman, G. von Laszewski, W. Smith, and S. Tuecke. A directory service for configuring high-performance distributed computations. In *Proc. 6th IEEE Symp. on High Performance Distributed Computing*, pages 365–375. IEEE Computer Society Press, 1997.

[9] I. Foster and C. Kesselman. The grid: Blueprint for a new computing infrastructure. *Morgan Kaufmann Publishers, USA*, 1998.

[10] J. Frey, T. Tannenbaum, M. Livny, I. Foster, and S. Tuecke. Condor-G: A computation management agent for multi-institutional grids. In *10th IEEE International Symposium on High Performance Distributed Computing (HPDC-10 '01), 2001*, pages 237 – 246, Washington, DC, USA, 2001. IEEE Computer Society.

[11] P. Garca, C. Pairot, R. Mondjar, J. Pujol, H. Tejedor, and R. Rallo. Planetsim: A new overlay network simulation framework. In *Software Engineering and Middleware, SEM 2004, Linz, Austria*, pages 123–137, 2005.

[12] F. Howell and R. McNab. Simjava: A discrete event simulation package for java applications in computer systems modeling. *First International Conference on Web-based modeling and Simulation, San Diego, CA, 1998, SCS Press: San Diego, CA*, 1998.

[13] A. Iamnitchi and I. Foster. A peer-to-peer approach to resource location in grid environments. pages 413–429, 2004.

[14] L. Chan and S. Karunasekera. Designing Configurable Publish-Subscribe Scheme for Decentralised Overlay Networks. In *The IEEE 21st International Conference on Advanced Information Networking and Applications (AINA-07)*, May 2007.

[15] K. Lai, B. A. Huberman, and L. Fine. Tycoon: A distributed market-based resource allocation system. *Technical Report, HP Labs*, 2004.

[16] D. Oppenheimer, J. Albrecht, A. Vahdat, and D. Patterson. Design and implementation tradeoffs for wide-area resource discovery. In *Proceedings of 14th IEEE Symposium on High Performance, Research Triangle Park, NC*, Washington, DC, USA, July 2005. IEEE Computer Society.

[17] R. Ranjan, R. Buyya, and A. Harwood. A case for cooperative and incentive based coupling of distributed clusters. In *Proceedings of the 7th IEEE International Conference on Cluster Computing (CLUSTER'05)*, Washington DC, USA, 2005. IEEE Computer Society.

[18] R. Ranjan, A. Harwood, and R. Buyya. A study on peer-to-peer based discovery of grid resource information. *Technical Report, GRIDS-TR-2006-17, Grid Computing and Distributed Systems Laboratory, University of Melbourne, Australia*, 2006.

[19] H. Samet. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley Publishing Company, 1989.

[20] C. Schmidt and M. Parashar. Flexible information discovery in decentralized distributed systems. *In the Twelfth International Symposium on High Performance Distributed Computing (HPDC-12), June*, 2003.

[21] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 149–160, New York, NY, USA, 2001. ACM Press.

[22] E. Tanin, A. Harwood, and H. Samet. A distributed quadtree index for peer-to-peer settings,. In *ICDE'05: Proceedings of the International Conference on Data Engineering*, pages 254–255, 2005.

[23] S. Waterhouse, D. M. Doolin, G. K., and Y. Faybishenko. Distributed search in p2p networks. *IEEE Internet Computing*, 06(1):68–72, 2002.

[24] S. Zanikolas and R. Sakellariou. A taxonomy of grid monitoring systems. *Future Generation Computer Systems (FGCS) Journal, Volume 21, Issue 1, Pages: 163-188, Elsevier Science, The Netherlands, January*, 2005.

[25] X. Zhang, J. L. Freschl, and J. M. Schopf. A performance study of monitoring and information services for distributed systems. *In the Twelfth International Symposium on High Performance Distributed Computing (HPDC-12), June*, 2003.