Coordinated load management in Peer-to-Peer coupled federated grid systems

Rajiv Ranjan · Aaron Harwood · Rajkumar Buyya

Abstract This paper proposes a coordinated load management protocol for Peer-to-Peer (P2P) coupled federated Grid systems. The participants in the system, such as the resource providers and the consumers who belong to multiple control domains, work together to enable a coordinated federation. The coordinated load management protocol embeds a logical spatial index over a Distributed Hash Table (DHT) space for efficient management of the coordination objects; the DHT-based space serves as a kind of decentralized blackboard system. We show that our coordination protocol has a message complexity that is logarithmic to the number of nodes in the system, which is significantly better than existing broadcast based coordination protocols.

The proposed load management protocol can be applied for efficiently coordinating resource brokering services of distributed computing systems such as grids and PlanetLab. Resource brokering services are the main components that control the way applications are scheduled, managed and allocated in a distributed, heterogeneous, and dynamic Grid computing environments. Existing Grid resource brokers, e-Science application work-flow schedulers, operate in tandem but still lack a coordination mechanism that can lead to efficient application schedules across distributed resources. Further, lack of coordination exacerbates the utilization of various resources (such as computing cycles and network bandwidth). The feasibility of the

R. Ranjan (✉)
Service Oriented Computing (SOC) Research Group, School of Computer Science and Engineering,
The University of New South Wales, Sydney, Australia
e-mail: rajiv@unsw.edu.au

A. Harwood · R. Buyya
Clouds Lab and P2P Group, Department of Computer Science and Software Engineering,
The University of Melbourne, Melbourne, Australia

A. Harwood
e-mail: aharwood@csse.unimelb.edu.au

R. Buyya
e-mail: raj@csse.unimelb.edu.au

proposed coordinated load management protocol is studied through extensive simulations.

Keywords Grid computing · Grid scheduling · Peer-to-Peer grids

1 Introduction

Distributed systems, including computational grids, P2P systems, Planetlab and cross-company workflows, involve participants who are topologically and administratively distributed over various control domains in the Internet. Participants in these resource sharing environments can be organized based on a federated model. In a federated organization [3, 8, 12, 17], every autonomous provider pools his resources together for common benefit of the community. As a result, every participant gets access to a much larger pool of resources.

1.1 Tragedy of commons

Distributed resource sharing systems such as computational grids and PlanetLab often exhibit the classical economics paradox called "tragedy of commons" during period of high service and resource demand. A study undertaken in [8] confirms that PlanetLab environment often experiences the problem of flash crowds where a growing number of users simultaneously request "slices" on common set of nodes to host their distributed systems experiments. Such bursty behavior of users often leads to sub-optimal system performance. Furthermore, the users who cannot successfully finish their experiments due to competing or conflicting requests in the system tend to retry their experiments that further aggravates the situation.

Another, motivating example is the way multiple Grid brokers schedule jobs on distributed Grid resources. A majority of existing approaches to Grid scheduling are non-coordinated. Brokers such as Nimrod-G [1], Condor-G [7] perform scheduling related activities independent of the other brokers in the system. They directly submit their applications to the underlying resources *without* taking into account the current load, priorities, or utilization scenarios of other brokers. Clearly, this leads to over-utilization, or a bottleneck, on some valuable resources while leaving others largely underutilized. Furthermore, these brokers do not have a coordination mechanism and this exacerbates the load sharing and utilization problems of distributed resources because sub-optimal schedules are likely to occur.

1.2 Centralized approaches

One of the possible ways to solve the coordination problem among Grid brokers and users, who operate in distributed fashion over the Internet has been to host a coordination service on a centralized machine, wherein every consumer is required to submit his resource demands to the coordination service. Similarly, resource providers update their resource usage status periodically with the coordination service. The centralized resource allocation coordination service performs system wide

load-distribution primarily driven by resource demand and availability. However, this approach has several design limitations including: (i) single point of failure; (ii) lacks scalability; (iii) high network communication cost at links leading to the coordination service (i.e. network bottleneck, congestion); and (iv) computational power required to serve a large number of participants.

1.3 Unstructured decentralized approaches

The coordinated scheduling protocols adopted by NASA-Scheduler [20] and Condor–Flock P2P [5] are based on general broadcast and limited broadcast communication mechanisms, respectively. Similarly, scheduling coordination in Tycoon [13] is based on a decentralized and isolated auction mechanism, where a user can end up bidding across every auction in the system (broadcast messaging). The OurGrid [3] system coordinates load-information among the sites based on a complete broadcast messaging approach. Specifically, OurGrid utilizes JXTA search [10] primitives as regards to resource discovery and message routing. Hence, these unstructured decentralized approaches have the following limitations: (i) high network overhead; and (ii) scalability problems.

1.4 Proposed approach: structured decentralized

In this work, we propose that the role of the centralized coordinator to be distributed among a set of machines based on a P2P network model. A P2P routing substrate such as the Chord or Pastry DHT [18, 21] can be utilized as the basis for overall system decentralization and management. DHTs are inherently self-organizing, fault-tolerant, and scalable. Specifically, we consider organizing resource brokers (and users in the case of PlanetLab) and distributed resources based on a DHT overlay. In the proposed approach, resource brokers post their resource demands by injecting a *Resource Claim* object into the DHT based decentralized coordination space, while resource providers update the resource supply by injecting a *Resource Ticket* object. These objects are mapped to the DHT-based coordination services using a spatial hashing technique [23]. The details on spatial hashing techniques is discussed in Sect. 6. The decentralized coordination space is managed by a software service (a component of our Grid broker service) called the coordination service, which undertakes activities related to decentralized load-balancing, coordination space management, etc. More details on how the coordination service in implemented as a software component of Grid broker service can be found in Sect. 5.

1.5 Our contributions

The main *contributions* of this paper include: (i) a global coordination protocol for load-management between distributed Grid brokers; and (ii) a proposal for utilizing the DHT based spatial index for managing complex coordination objects and decentralizing the protocol. We now summarize some of our findings:

- resource claim and ticket object injection rates have significant influence on the coordination delay experienced by distributed users in the system

- proposed coordination protocol is highly effective in curbing the number of scheduling negotiation iterations undertaken on a per job basis, the redemption, and notification message complexity involved with the coordination protocol is $O(1)$
- In a federation of n heterogeneous Grid resources, on average the number of messages required to successfully map a job to a resource is $O(\log n)$

2 Paper organization

The rest of this paper is structured as follows: Sect. 3 sets the proposed coordination protocol in context with related work. Section 4 gives an overview of the Grid system and scheduling model that we consider in this paper. Next, Sect. 5 discusses the key elements of the coordination protocol. Section 6 summarizes d-dimensional spatial index that forms the basis for mapping the coordination objects. In Sect. 7, we present the finer details on the coordination protocol-based resource allocation and load-balancing algorithms. Message complexity analysis is presented in Sect. 8. In Sect. 9, we present various experiments and discuss our results. We end this paper with concluding remarks in Sect. 10.

3 Related work

The main focus of this section is to compare the novelty of the proposed work against the current state-of-the-art. Coordinated management of resources in grids and other distributed systems is a widely studied research problem. Several research projects such as Bellagio [4], Tycoon [13], NASA-Scheduler [20], OurGrid [3], Sharp [8], Shirako [12], and Condor–Flock [5] have proposed federated models for inter-networking compute and storage resources in grids. These systems support a varying degree of global coordination with respect to load-management. Furthermore, based on the network and communication models, these approaches have different scalabilities.

The Shirako [12] system presents a mechanism for on-demand leasing of networked Grid resources. Shriako's leasing architecture builds upon the Sharp framework for secure resource peering and distributed resource allocation. The Shirako system does not define how different brokers in the system connect with each other, as the system grows to a large number of participants. The novel contribution of the proposed work lies in this domain, i.e. efficient protocol for enforcing global coordination among distributed brokers in the system. Next, Bellagio [4] is a market-based resource allocation system for federated Grid infrastructures. With the centralized auction approach adopted by the Bellagio system, a best-case communication overhead of $O(c)$ is exhibited if the auction is limited to c participants and $O(n)$ if not. In contrast, the proposed coordinated load-management protocol is based on a DHT-based d-dimensional space. The approach inherits the underlying features of DHT-based networks such as scalability, self-organization, and not prone to single point of failure.

Tycoon [13] is a distributed market-based resource allocation system. Application scheduling and resource allocation in Tycoon is based on a decentralized isolated

auction mechanism. In the worst case, a scheduler or a broker can end-up bidding across all n sites in the system. Hence, on a per job basis a broker can generate $O(n)$ messages in the system. Since Tycoon is based on a distributed auction, it has a best-case communication overhead of $O(nc)$, based on the assumption that an auction is limited to c participants. In contrast using the proposed approach a broker needs to undertake close to logarithmic number messages (that we show later) on a per job basis. The system [20] models a Grid broker architecture and studies three different distributed job migration algorithms. Scheduling in the Grid environment is facilitated through coordination between site specific LRMS and the GS. System-wide load coordination algorithms such as the sender-initiated, receiver-initiated, and symmetrically-initiated algorithms are based on complete broadcast messaging between participants' GSes, thereby clearly incurring $O(n)$ messages on a per job basis. We improve on these load coordination algorithms by reducing the number of messages to a logarithm of the number of GSes in the system.

Condor–Flock [5] presents a Grid scheduling system that consists of Internet-wide Condor work pools organized on the Pastry overlay. The site managers in the overlay coordinate the load-management by announcing available resources to all the sites whose Identifiers (IDs) appear in the routing table. Hence, in a network of n condor sites, $O(n \log_b n)$ messages are generated per resource status change on per site basis. Similarly, the load-information coordination in OurGrid [3] is also based on complete broadcast messaging. OurGrid system is implemented using the JXTA [10] P2P substrate. In contrast, our coordinated load-management protocol utilizes a d-dimensional spatial index over a DHT space for deterministic lookups and coordination. This gives the system ability to produce controllable number of messages and guarantees a deterministic behavior with respect to number of routing hops. In Table 1, we analytically compare the performance of the proposed coordination protocol with respect to the current state-of-the-art.

Table 1 Coordination technique comparison

System name	Network model	Message complexity (per job)
NASA-Scheduler [20]	Unstructured Decentralized	$O(n)$
Condor–Flock P2P [5]	Structured Decentralized	$O(n \log n)$
Shirako [12]	Centralized	$O(1)$
Our-Grid [3]	JXTA	$O(n)$
Tycoon [13]	Unstructured Decentralized	$O(n)$
Bellagio [4]	Centralized	$O(1)$
Proposed work	*Structured Decentralized*	$O(\log n)$

4 Models

4.1 Grid scheduling model

The proposed coordinated load management protocol derives from the *Grid-Federation* [17] resource sharing model. Grid-Federation aggregates distributed resource brokering and allocation services as part of a single and cooperative resource sharing environment. The Grid-Federation, $G_F = \{R_1, R_2, \ldots, R_n\}$, consists of a number of Grid sites, n, with each site having to contribute its local resources to the federation. In this work, we assume $R_i = (p_i, x_i, \mu_i, \emptyset_i, \gamma_i)$, which includes the number of processors, p_i, processor architecture, x_i, their speed, μ_i, installed operating system type, $\emptyset_i$, and underlying interconnect network bandwidth, γ_i. Refer to Table 2 for the notations that we utilize in the remainder of this paper.

Resource brokering, indexing and allocation in the Grid-Federation is facilitated by a Resource Management System (RMS) known as the Grid Federation Agent (GFA). Figure 1 shows an example Grid-Federation resource sharing model consisting of Internet-wide distributed parallel resources (clusters, supercomputers). The GFA service is composed of three software modules including a Grid Resource Manager (GRM), Local Resource Management System (LRMS), and Grid Peer. The GRM component of a GFA exports a Grid site to the federation and is responsible for coordinating federation wide application scheduling and resource allocation. The GRM is responsible for scheduling the locally submitted jobs in the federation. Further, it also manages the execution of remote jobs in conjunction with the LRMS. Additionally, LRMS implements the following methods for facilitating federation wide job submission and migration process: answering the GRM queries related to local job queue length, expected response time, and current resource utilization status x.

The Grid peer module in conjunction with indexing service performs tasks related to decentralized resource lookups and updates. It is the Grid peer component, which interacts with the DHT overlay. A Grid Peer service generates two basic types of objects with respect to coordinated grid brokering: (i) a claim, a object sent by a

Fig. 1 GFAs 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 ticket/claim object posted by Grid sites and GFA service
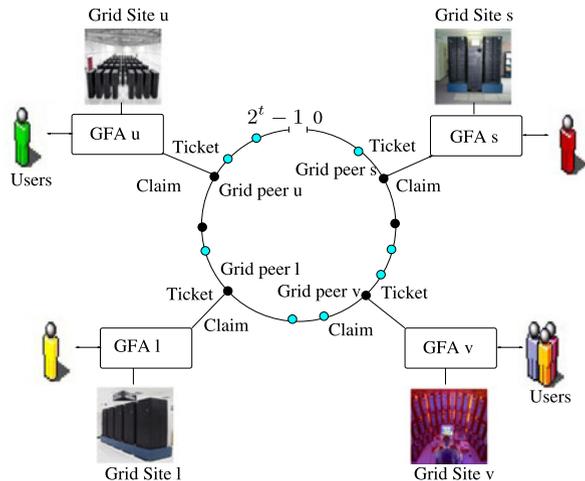
Table 2 Notations

Symbol	Meaning
Resource	
n	Number of Grid Federation Agents (GFAs) in the Grid network
G_i	ith GFA in the system
R_i	Configuration of the ith resource in the system
ρ_i	Resource utilization for resource at GFA i
x_i	Processor architecture for resource at GFA i
p_i	Number of processors for resource at GFA i
ϕ_i	Operating system type for resource at GFA i
μ_i	Processor speed at GFA i
γ_i	Network inter-connection bandwidth at R_i
Job	
$J_{i,j,k}$	ith job from the jth user of kth GFA
$p_{i,j,k}$	Number of processor required by $J_{i,j,k}$
$\phi_{i,j,k}$	Operating system type required by $J_{i,j,k}$
$\alpha_{i,j,k}$	Communication overhead for $J_{i,j,k}$
$x_{i,j,k}$	Processor architecture type required by $J_{i,j,k}$
Index	
$r_{i,j,k}$	A claim posted for job $J_{i,j,k}$
U_i	A ticket issued by the ith GFA/broker
dim	Dimensionality or number of attributes in the Cartesian space
$f_{\min}$	Minimum division level of d-dimensional index tree
M_c	Random variables denoting number of messages generated in mapping a claim object
T_c, T_t	Random variables denoting number of disjoint query paths undertaken in mapping a claim and ticket object
Network	
λ^{in}	Total incoming claim/ticket arrival rate at a network queue i
λ^{out}	Outgoing claim/ticket rate at a network queue i
μ_n	Average network queue service rate at a Grid peer i
μ_r	Average query reply rate for index service at GFA i
λ_t^{in}	Incoming ticket rate at a index service i
λ_c^{in}	Incoming claim rate at a index service i
λ_a^{in}	Incoming query rate at a DHT routing service i from the local index service
$\lambda_{\text{index}}^{\text{in}}$	Incoming index query rate at a index service i from its local DHT routing service
K	Network queue size

GFA service to the P2P overlay-based coordination space for locating the resources matching a user's application requirements; and (ii) a ticket, is an update object sent by a Grid site owner about the underlying resource conditions. Since, a Grid resource

is identified by more than one attribute, a claim or ticket 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* [19]. Grid peer component of GFA service is responsible for distributed ticket publication, claim subscription, and logical d-dimensional index space management.

In order to study the effectiveness of the proposed protocol with respect to load management, we consider coordinated scheduling of synchronous parallel applications [14] on a federated Grid system of parallel resources (such as cluster or supercomputer). A job $J_{i,j,k}$ consists of the number of processors required, $p_{i,j,k}$, the job length, $l_{i,j,k}$ (in terms of instructions), and the communication overhead, $\alpha_{i,j,k}$.

4.2 Job model

To capture the nature of parallel execution with message passing overhead required by the parallel applications, we considered a part of total execution time as the communication overhead and remaining as the computational time. We consider the network communication overhead $\alpha_{i,j,k}$ for a parallel job $J_{i,j,k}$ to be randomly distributed over the processes. In other words, we do not consider the case, e.g. when a parallel program written for a hypercube is mapped to a mesh architecture. We assume that the communication overhead parameter $\alpha_{i,j,k}$ would scale the same way over all the clusters depending on γ_i. The total data transfer involved during a parallel job execution is given by

$$\Gamma(J_{i,j,k}, R_k) = \alpha_{i,j,k} \times \gamma_k.$$

The time for job $J_{i,j,k} = (p_{i,j,k}, l_{i,j,k}, \alpha_{i,j,k})$ to execute on a parallel resource R_m is,

$$T(J_{i,j,k}, R_m) = \frac{l_{i,j,k}}{\mu_m \, p_{i,j,k}} + \frac{\Gamma(J_{i,j,k}, R_k)}{\gamma_m}.$$

5 Coordination protocol

We start this section with a description of the communication, coordination and indexing models that are utilized to facilitate the P2P coordination space. Thereafter, we look at the composition of objects and access primitives that form the basis for coordinating the application schedules among the distributed GFAs/brokers.

### 5.1 Layered design of the coordination protocol

The OPeN architecture proposed in the work [22] is utilized as the base model in architecting and implementing the proposed protocol. The OPeN architecture consists

of three layers: the *Application* layer, *Core Services* layer, and *Connectivity* layer.
Grid Services such as resource brokers work at the Application layer and insert objects via the Core services layer. In the context of a GFA, the GRM software module operates at the Application layer.

We have implemented the Coordination service as a sub-layer of the Core services layer. The Coordination service accepts the application-level objects such as claims/tickets. These objects encapsulate coordination logic, which in this case is the provisioning information (current resource status and resource requirements) that is required by coordination service to undertake efficient load-balancing across distributed Grid resources. These objects are managed by the coordination service. The calls between the Coordination service and Resource discovery service are made through the standard publish/subscribe technique. The Resource discovery service is responsible for managing the logical index space and communicating with the Connectivity layer. As shown in Fig. 2, Core services layer is managed by the Grid Peer module of a GFA service.

The Connectivity layer is responsible for undertaking key-Based routing in the DHT space such as Chord, CAN, Pastry, etc. The actual implementation protocol at this layer does not directly affect the operations of the Core services layer. In principle, any DHT implementation at this layer could perform the desired task. In this paper, our simulation environment models the Chord substrate at the Connectivity layer. Chord hashes the peers and objects (such as fields, logical indices, etc.) to the circular identifier space and it guarantees the average lookup complexity as $O(\log n)$

Fig. 2 Layered design of the coordination protocol
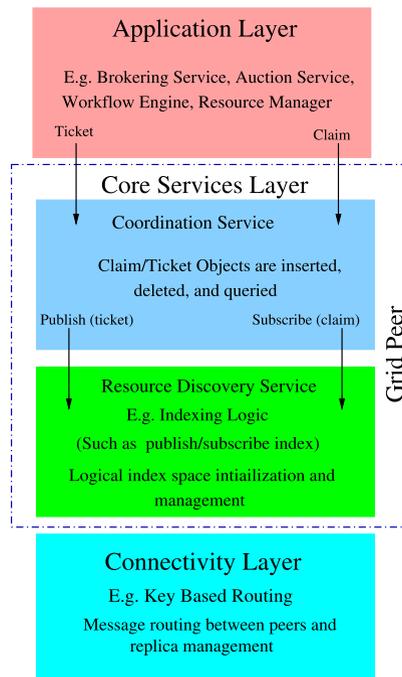
steps with high probability. Each peer in the Chord network is required to maintain routing table state of $O(\log n)$ other peers, where n is the total Grid network size.

5.2 Coordination objects

This section gives details about the resource claim and ticket objects that form the basis for enabling decentralized coordination mechanism among the brokers/GFAs in a Grid environment. We start with the description of the components that form the part of a Grid-Federation resource ticket object.

5.2.1 Resource ticket

Every GFA in the federation publishes its resource ticket to the Coordination space through the Core services layer (shown in Fig. 2). A resource ticket object U_i, or update query, consists of a resource description R_i, for a resource i. E.g.:

Resource Ticket: Total-Processors = 100 && Processor-Arch = Pentium && Processor-Speed = 2 GHz && Operating-System = Linux && Utilization = 0.80.

5.2.2 Resource claim

A resource claim or look-up object encapsulates the resource configuration needs of a user's job. In this work, we focus on the application types whose needs can be satisfied by compute and storage resources that are available within Grid and Planet-Lab environments. Users submit their application's resource requirements to the local GFA. A GFA aggregates the characteristics of a job, including number of processors, their architecture, operating system type, with constraint on maximum speed, and resource utilization into a resource claim object, $r_{i,j,k}$. For example,

Resource Claim: Total-Processors $\geq$ 70 && Processor-Arch = Pentium && 2 GHz $\leq$ Processor-Speed $\leq$ 5 GHz && Operating-System = Solaris && 0.0 $\leq$ Utilization $\leq$ 0.90.

The GRM module of a GFA passes the resource ticket and claim object to the co-ordination service operating at the Core services layer. Recall that the Core services layer is managed by the Grid peer module. It is the Grid peer module, which interacts with the DHT based overlay network. The operation between Grid peer module and DHT based overlay is transparent to the GRM. In other words, a GRM is not aware of how the Grid peer module is routing, searching, and matching the objects in the system. It is the responsibility of a Grid peer module to implement specific communication and data organization methods, which can provide desired functionality to the Application layer services, i.e. a GRM. In order to efficiently route, search, and match the d-dimensional claim and ticket objects, the Grid peer module embeds a logical spatial data-structure over the DHT overlay space. The finer details on this spatial data-structure is given in Sect. 6.

The resource ticket and claim objects are spatially hashed to an index cell i in the d-dimensional coordination space. Similarly, coordination services in the Grid
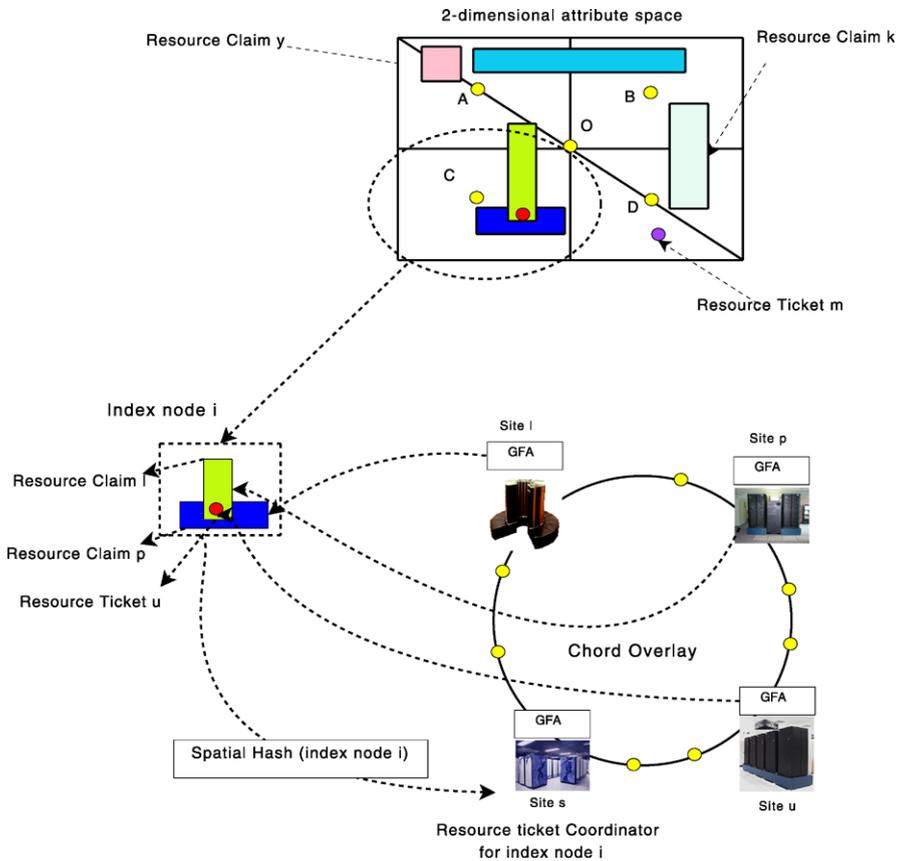
Fig. 3 Resource allocation and application scheduling coordination across Grid sites

network hash (through the Connectivity layer) themselves into the space using the overlay hashing function such as SHA-1. In Fig. 3, resource claim objects issued by site p and l are mapped to the index cell i, which is currently hashed to the site s. In this case, site s is responsible for coordinating the resource sharing among all the resource claims that are mapped to the cell i. Subsequently, site u issues a resource ticket (shown as dot in Fig. 3) that falls under a region of the space currently required by users at site p and l. In this case, the coordination service of site s has to decide which of the sites (i.e. either l or p or both) be allowed to claim the ticket issued by site u. This load-distribution decision is based on the load-balancing objective that it should not lead to over-provisioning of resources at site u.

In Table 3, we show an example list of claim objects that are stored with a coordination service at time $t = 900$ seconds. Essentially, the claims in the list arrived at a time ≤ 900 and are waiting for a suitable ticket object that can meet its resource configuration requirements while Table 4 depicts the list of ticket objects that have arrived at $t = 900$ seconds. Following the ticket arrival event, the coordination service undertakes a procedure that divides this ticket object among the list of claims.

Table 3 Claim list

Time	Claim ID	$\mu_{i,j,k}$	$p_{i,j,k}$	$x_{i,j,k}$	$\phi_{i,j,k}$
200	Claim 1	>800	50	Intel	Linux
350	Claim 2	>1200	20	Intel	Linux
500	Claim 3	>700	10	Sparc	Solaris
700	Claim 4	>1500	1	Intel	Windows XP

Table 4 Ticket list

Time	GFA ID	μ_i	p_i	p_i avail	x_i	ϕ_i
900	GFA-8	1400	80	75	Intel	Linux

Based on the resource attribute specification, only Claim 1 and Claim 2 matches the ticket's resource configuration. As specified in the ticket object, there are currently 75 processors available with the GFA 8, which is less than the sum of processors required by Claim 1 and 2 (i.e. 70). Hence, in this case the coordination service, based on a First-Come-First-Serve (FCFS) queue processing scheme, first notifies the GFA that has posted Claim 1 and follows it with the GFA responsible for Claim 2. However, Claims 3 and 4 have to wait for the arrival of tickets that can match their required resource configuration.

The coordination service notifies a claimer for resources by issuing a soft state pass that is redeemable for a lease at the ticket issuer GFA in the system. The soft state pass specifies the resource type R_i for which access should be granted over a duration (expected running time of an application). GFAs on behalf of local site issue tickets for resources and post to the coordination space. Our coordination protocol can leverage the Sharp framework [8] with respect to secure resource exchanges. In Sharp, all exchanges are digitally signed, and the GFAs/brokers endorse the public keys of the GFAs in the system.

6 *D*-dimensional coordination object mapping and routing

The 1-dimensional hashing provided by a standard DHT is insufficient to manage complex objects such as resource tickets and claims. DHTs generally hash a given unique value/identifier (e.g. a file name) to a 1-dimensional DHT key space and hence they cannot directly support mapping and lookups for complex objects. Management of those objects whose extents lie in d-dimensional space can be done by embedding a logical index structure over the 1-dimensional DHT key space.

We now describe the features of the P2P-based spatial index that we utilize for mapping the d-dimensional claim and ticket objects over the DHT key space. Providing background work and details on this topic is beyond the scope of this paper; here we only give a high level picture. The spatial index that we consider in this work assigns regions of space to the Grid peers. If a Grid peer is assigned a region of d-dimensional space, then it is responsible for handling query computation associated with the claim and ticket objects that intersect that region, and for storing the objects
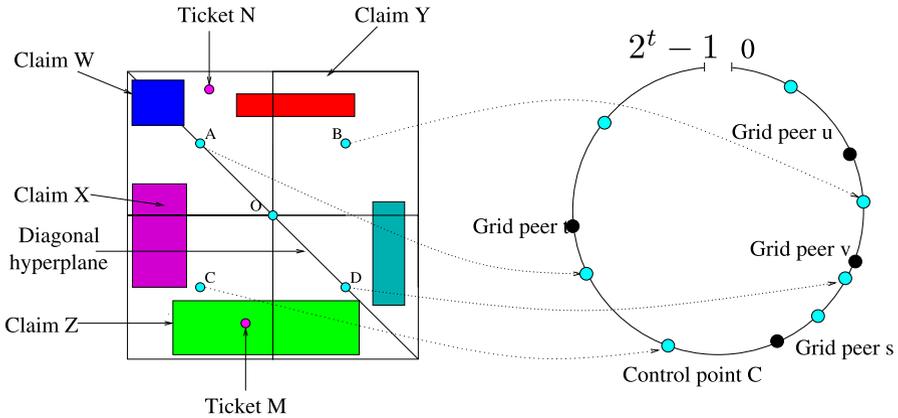
Fig. 4 Spatial resource claims $\{W, X, Y, Z\}$, cell control points, point resource tickets $\{M\}$ and some of the hashings to the Chord, i.e. the d-dimensional coordinate values of a cell's control point is used as the key and hashed onto the Chord. *Dark dots* are the grid peers that are currently part of the network. *Light dots* are the control points hashed on the Chord. For this figure, $f_{\min} = 2$, dim $= 2$

that are associated with the region. Figure 4 depicts a 2-dimensional Grid resource attribute space for mapping claim and ticket objects. The attribute space has a grid-like structure due to its recursive division process. The index cells resulting from this process remain constant throughout the life of the d-dimensional attribute space and serve as entry points for subsequent mapping of claim and ticket objects. The number of index cells produced at the minimum division level is always equal to $(f_{\min})^{\dim}$, where dim is the dimensionality of the Cartesian space. Details on recursive subdivision techniques can be found in the article [23]. Every Grid peer in the network has basic information about the Cartesian space coordinate values, dimensions, and minimum division level.

Every cell at the $f_{\min}$ level is uniquely identified by its centroid, termed a *control point*. Figure 4 depicts four control points A, B, C, and D. DHT hashing method such as the Chord method is utilized to hash these control points to the overlay so that the responsibility for managing a index cell is associated with a Grid peer in the system. In Fig. 4, control point B is hashed to the Grid peer s, which is responsible for managing all claim and ticket object that are stored with that control point. For mapping claim objects, the search strategy depends whether it is a DPQ or DRQ. For a DPQ type claim, the mapping is straight forward since every point is mapped to only one cell in the Cartesian space. For a DRQ type claim, mapping is not always singular because a range look-up can cross more than one cell. To avoid mapping a range claim 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 utilized. This mapping involves feeding claim candidate index cells as inputs into a mapping function, F_{map}. This function returns the IDs of index cells to which given claim should be mapped. Spatial hashing is performed on these IDs (which returns a keys) to identify the current Grid peers responsible for managing the given keys. A Grid peer service uses the index cell(s) currently assigned to it and a set of known base index cells obtained at initialization as the candidate index cells.

Similarly, mapping the ticket also involves the identification of the cell in the Cartesian space. A ticket is always associated with an event region and all cells that are contained fully or partially within the event region [11] will be selected to receive the corresponding ticket. The calculation of an event region is also based upon the diagonal hyperplane of the Cartesian space.

7 Proposed algorithms

In this section, we provide the description of the algorithms that have been developed for: (i) resource allocation and (ii) decentralized coordination. In Table 2, we present the model parameters required to discuss the decentralized resource allocation and coordination in federated grid environments. In Fig. 5, we summarize these steps using distributed interaction diagram.

Fig. 5 Example of the process of job scheduling, resource allocation, and decentralized coordination: (1) *user* submits a job to the local broker, *GFA1*. Following this, *GFA1* posts or subscribes a resource claim to the coordination space corresponding to that job; (2) all the *GFA*s in the system sends or publish the resource ticket or Resource Update Query (RUQ) to the P2P coordination space; (3) In the P2P coordination space, resource ticket of *GFA3* matches with the resource claim of *GFA1*; *GFA1* receives the notification message of resource matching (*GFA3*) from the coordination space; (4) *GFA1* submits job *GFA3*; job is executed at *GFA3* and *GFA3* notifies *GFA1* of completion

Algorithm 1 RESOURCE ALLOCATION IN COORDINATION SPACE

1: **PROCEDURE**: Match
2: Input: Ticket u_i from Resource R_i
3: **begin**
4:     $index \leftarrow 0$
5:     $ClaimList_m \leftarrow \Phi$
6:     **while** $ClaimList[\text{index}] \neq null$ **do**
7:         $r_{i,j,k} \leftarrow ClaimList[\text{index}]$
8:         **if** $r_{i,j,k} \cap u_i \neq null$ **then**
9:             $ClaimList_m \leftarrow ClaimList_m \cup r_{i,j,k}$
10:        $index \leftarrow index + 1$
11:    **end**
12:    **return** $ClaimList_m$
13: **end**
14: **PROCEDURE**: Event Resource Claim Submit
15: Input: Claim $r_{i,j,k}$
16: **begin**
17:    $ClaimList \leftarrow ClaimList \cup r_{i,j,k}$
18: **end**
19: **PROCEDURE**: Event Resource Ticket Submit
20: Input: Ticket u_i from Resource R_i
21: **begin**
22:    $ClaimList_m \leftarrow \text{Match}(u_i)$
23:    $index \leftarrow 0$
24:    **while** R_i is not over-provisioned **do**
25:        Send notification of match event to resource claimer $ClaimList_m[\text{index}]$
26:        Remove $ClaimList_m[\text{index}]$
27:        $index \leftarrow index + 1$
28: **end**
29: **end**

7.1 Resource allocation

The details of the decentralized resource allocation algorithm (refer to Algorithm 1) that is undertaken by the P2P coordination space is presented here. When a resource claim object, $r_{i,j,k}$ arrives at the coordination service, it is added to the existing claim list, *ClaimList* by the coordination service (lines 16–19). When a resource ticket object, u_i arrives at the coordination service, the list of resource claims that overlap or match with the submitted resource ticket object in the d-dimensional space is computed (lines 21–25). The overlap signifies that the job associated with the given claim object can be executed on the ticket issuer's resource subject to its availability.

In order to get the matches, the coordination service first, selects the claim objects in the *ClaimList* in first come first serve order (lines 4–5); then from this list, the number of claims that overlap with the ticket are selected to the *ClaimList_m* (lines 6–13). The resource claimers are notified about the resource ticket match until the ticket

issuer is not over-loaded (lines 25–30). The coordination procedure can utilize the dynamic resource parameters such as the number of available processors, queue length, etc. as the over-loading indicator. These over-loading indicators are encapsulated with the resource ticket object by the GFAs.

7.2 Coordination algorithm

This section provides the description of the algorithm (refer to Algorithm 2) for coordinating allocation of distributed Grid resources. When a GFA, G_j intends to submit a job, $J_{i,j,j}$ for execution, it compiles a resource claim object, $r_{i,j,j}$ for the job and posts it to the P2P coordination space (lines 1–6). On the other hand, the GFA, G_i compiles the resource ticket object, u_i for resource, R_i and publishes it to the P2P coordination space periodically depending on the ticket injection rate. Besides, whenever the resource condition changes such as a task completion event happens, the GFA also posts a ticket object for the corresponding resource immediately (lines 7–12).

Once there is a match between a ticket object, u_i and a claim object, $r_{i,j,j}$, the coordination service sends a ticket redemption request for job, $J_{i,j,j}$ to ticket issuer GFA, G_i (lines 13–17). The request message contains the information that job, $J_{i,j,j}$ is submitted by GFA, G_j. After notifying the resource claimer GFA, the coordination service un-subscribes the resource claim for that task from the P2P coordination space.

When the ticket issuer GFA, G_i receives the notification of match from the coordination service, it sends a *Reply* to the resource claimer GFA, G_j. If the task can be executed in the local resource at that time, it sends a positive reply; otherwise it sends a negative reply to the claimer GFA (lines 18–26). If the *Reply* is 'accept', then the claimer GFA, G_j transfers the locally submitted job, $J_{i,j,j}$ to the ticket issuer GFA and un-subscribes the claim object from the coordination space to remove duplicates (lines 27–32). However, if the ticket issuer GFA fails to grant access due to local resource sharing policy (i.e. *Reply* is 'reject'), then the claimer GFA reposts the resource claim object for that task to the coordination space for future notifications (lines 33–34).

8 Complexity analysis

Lemma 1 *In a federation of n heterogeneous Grid resources, on average a job $J_{i,j,k}$ requires $O(\log n)$ messages to be sent in the network in order to locate a node that can successfully complete the job without being overloaded.*

Scheduling a job $J_{i,j,k}$ in the coordinated federation involves the following steps: (i) posting the resource claim object to DHT based coordination space; (ii) receiving the notification message from the coordination space when a resource ticket object hits the claim object; (iii) contacting the GFA (site authority) about the claim-ticket match, the contacted GFA performs certain checks such as security, resource availability. Hence, the total number of messages produced in successfully allocating a job to a resource is summation of the number of messages produced in these three steps.

Algorithm 2 DECENTRALIZED COORDINATION

1: **PROCEDURE**: Event Job Submit
2: Input: Job $J_{i,j,k}$
3: **begin**
4:     Encapsulate claim object $r_{i,j,k}$ for job $J_{i,j,k}$
5:     Subscribe $r_{i,j,k}$ to coordination space
6: **end**
7: **PROCEDURE**: Event Resource Status Changed
8: Input: Resource R_i
9: **begin**
10:     Encapsulate ticket object u_i for resource R_i
11:     Publish u_i to coordination space
12: **end**
13: **PROCEDURE**: Event Ticket Redeemption Request
14: Input: Task $J_{i,j,k}$, GFA G_i
15: **begin**
16:     Send ticket redemption *Request* for task $J_{i,j,k}$ to
          GFA G_i; *Request* message implies G_j
17: **end**
18: **PROCEDURE**: Event Ticket Redeemption Reply
19: Input: Task $J_{i,j,k}$, GFA G_j
20: **begin**
21:     **if** $J_{i,j,k}$ can be executed in local resource **then**
22:         Send *Reply* "accept" to GFA G_j
23:     **else**
24:         Send *Reply* "reject" to GFA G_j
25:     **endif**
26: **end**
27: **PROCEDURE**: Event Ticket Redeemption Reply Action
28: Input: Task $J_{i,j,k}$, GFA G_i, *Reply*
29: **begin**
30:     **if** *Reply* is "accept" **then**
31:         Send Ji, j, k to accepting GFA G_i for execution
32:         Unsubscribes the claim object for $J_{i,j,k}$
33:     **else**
34:         Subscribe claim object $r_{i,j,k}$ to coordination space
35:     **endif**
36: **end**

We denote the number of messages generated in mapping a resource claim by a random variable M_c. The distribution of M_c is function of the problem parameters including query size, dimensionality of search space, query rate, division threshold, and data distribution. 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 $O(\log n)$ routing hops using the Chord method. Based on the above discussion, in order to compute the worst case message lookup and routing complexity one additional random variable T_c is considered. T_c denotes the number of disjoint query path undertaken in mapping a claim object. The spatial index described in Sect. 6 maps a resource claim can be utmost mapped to 2 index cells. Note that the number of index cells to which a resource claim can be mapped is dependent on the spatial index. Providing background information on this topic is beyond scope of this paper, interested readers may refer to the paper [15] for further information. With the Chord method, every disjoint path will undertake $E[T_c] \times (\log_2 n)$ routing hops with high probability. Hence, the expected value of M_c is given by:

$$E[M_c] = E[T_c] \times (\log_2 n),$$

substituting $E[T_c]$ with the value 2 and adding 2 for messages involved with sending notification and negotiation,

$$E[M_c] = 2 \times (\log_2 n) + 2,$$

ignoring the constant terms in the above equation we get

$$E[M_c] = O(\log n). \tag{1}$$

The above equation shows that scheduling message complexity function growth rate is bounded by the function $O(\log n)$.

Lemma 2 *In a federation of n GFAs/brokers, each broker having m jobs to schedule then total scheduling messages generated in the system is bounded by the function* $O(m \times n \times \log n)$.

This lemma directly follows from Definition 1. Since a job in the system requires $O(\log n)$ messages to be undertaken before it can be successfully allocated to resource, therefore, computing the scheduling message complexity for m jobs distributed over n GFAs/brokers is straightforward.

Lemma 3 *In a federation of n heterogeneous resources if GFAs/brokers posts p resource ticket object over a time period t, then the average-case message complexity involved with mapping these tickets to Grid peers in the network is bounded by the function* $O(E[T_t] \times p \times \log n)$.

The proof for this definition directly follows from Lemma 1. The procedure for mapping the ticket object to the Grid peers is similar to the one followed for a claim object. A ticket object is always associated with an event region, and all index cells that fall fully and partially within the even region will be selected to receive the corresponding ticket object. The number of disjoint query path taken to map a ticket object is denoted by random variable T_t with mean $E[T_t]$.

9 Performance evaluation

In this section, we validate the effectiveness of the proposed coordination protocol using a trace driven simulation.

9.1 Simulation model

In our message queuing model, a Grid peer node (through its Chord routing service) is connected to an outgoing network queue (see label 6 in Fig. 6) and an incoming link from the Internet (see label 5 in Fig. 6). The network messages that are delivered through the incoming link (effectively coming from other Grid peers in the overlay) are processed as soon as they arrive. If the destination of the incoming messages is the local index service then they are put into the index message queue (see label 2 in Fig. 6). Otherwise, the messages are routed through the outgoing queue to the next successor Chord service, which is more closer to the destination in the Chord key space. Furthermore, the Chord routing service (see label 4 in Fig. 6) receives messages from the local index service (see label 3 in Fig. 6). Similarly, these messages are processed as soon as they arrive at the Chord routing service. After processing, the 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 queuing and processing) encountered in an outgoing queue are given by the M/M/1/K [2] queue steady state probabilities.

9.2 Experimental setup

Our simulation infrastructure is created by combining two discrete event simulators namely *GridSim* [6], and *PlanetSim* [9]. The experiments run a Chord overlay with a 32 bit configuration, i.e. the number of bits utilized to generate Grid peer and object (claim/ticket) ids. The GFA/broker network size is fixed to $n = 100$. The network queue message processing rate is fixed at $\mu_n = 5{,}000$ messages per second and network message queue size is fixed at $K = 10^4$. The GFAs inject resource claim and

Fig. 6 Network message queuing model at a Grid peer i
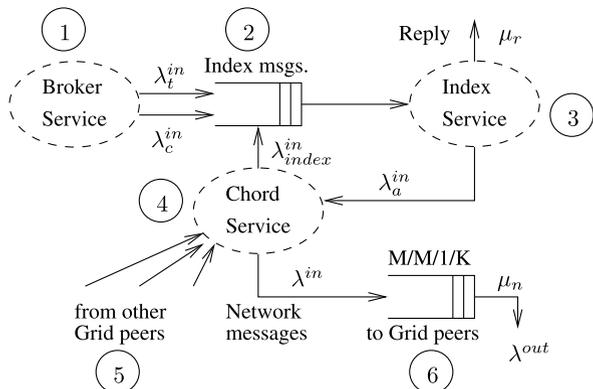
ticket objects based on the exponential inter-arrival time distribution. The value for resource claim inter-arrival delay, $\frac{1}{\lambda_c^{in}}$, is distributed over [60, 600].

The GFAs inject resource ticket objects with mean delay, $\frac{1}{\lambda_t^{in}}$, distributed over [120, 600] in steps of 120 seconds. The spatial extent of both resource claims and ticket objects lie in a 4-dimensional attribute space, i.e. dim $= 4$. These attribute dimensions include the number of processors, their speed, their architecture, and operating system type. The distributions for these resource dimensions are obtained from the Top 500 supercomputer list.[1] The GFAs/brokers encode the metric *"number of available processors"* at time t with the resource ticket object U_i. A coordination service utilizes this metric as the indicator for the current load on a resource R_i. In other words, a coordination service will stop sending the notifications as the number of processors available with a ticket issuer approaches *zero*. f_{min} of the logical d-dimensional spatial index is set to 4. We generate the workload distributions across GFAs based on the model given in the paper [14]. The processor count for a resource was fed to the workload model based on the resource configuration obtained from the Top 500 list. For more finer details on the experiment configuration, the readers are referred to the technical report [16].

9.3 Results and observations

In our simulation, we vary the resource claim inter-arrival delay over the interval [60, 600] and the resource ticket (update) inter-arrival delay over the interval [120, 600] in steps of 120 seconds. The graphs in Figs. 7 and 8 show the performance of the proposed coordination protocol in terms of scheduling efficiency and network overhead perspective, respectively. Please note that, labels in Figs. 7 and 8, denote the measurements obtained at different values of $\frac{1}{\lambda_t^{in}}$.

9.3.1 Scheduling efficiency perspective

As a measurement of scheduling performance, we use the following metrics namely, average coordination delay, average response time, average number of ticket redemption negotiations, and total scheduling messages.

The scheduling performance metric coordination delay sums up the latencies for: (i) a resource claim to reach the index cell; (ii) waiting time until a resource ticket matches with the claim; and (iii) the notification delay from coordination service to the relevant GFA. CPU time for a job is defined as the time a job takes to actually execute on a processor or set of processors. The average response time for a job is the delay between the submission and arrival of execution output. Effectively, the response time includes the latencies for coordination and CPU time. Note that these measurements are collected by averaging the values obtained for each job in the system.

Figure 7(a) depicts the results for average coordination delay in seconds with increasing resource claim mean inter-arrival delay, $\frac{1}{\lambda_c^{in}}$ and for different inter-arrival
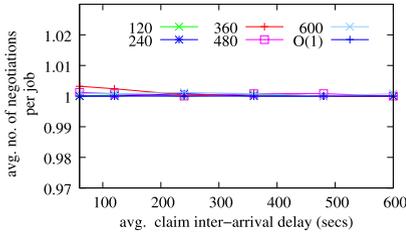

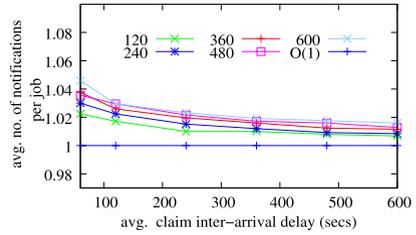
[1]Top 500 Supercomputer List, http://www.top500.org/.
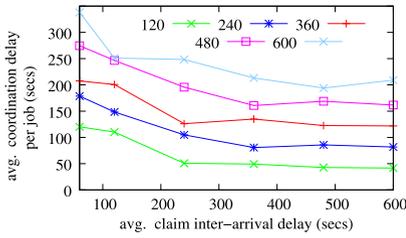
(a) Average claim inter-arrival delay vs. Coordination delay
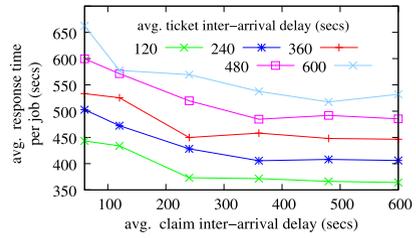
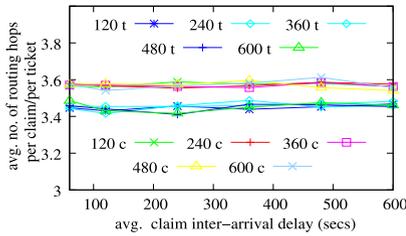(b) Average claim inter-arrival delay vs. Response time

(c) Average claim inter-arrival delay vs. Number of negotiations
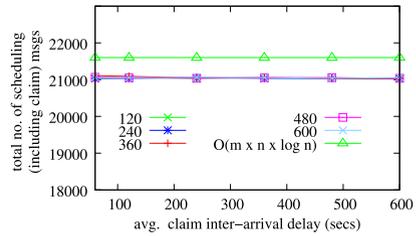
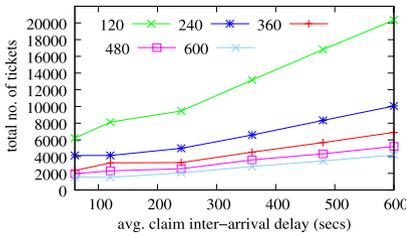(d) Average claim inter-arrival delay vs. Number of notification

Fig. 7 Effect of $\frac{1}{\lambda_c^{in}}$ and $\frac{1}{\lambda_t^{in}}$ on different performance metrics (scheduling efficiency perspective)
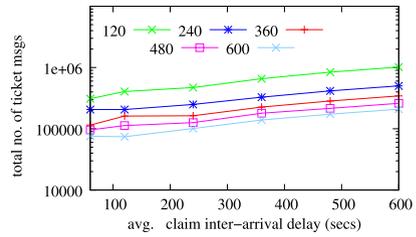

(a) Average claim inter-arrival delay vs. Average routing hops

(b) Average claim inter-arrival delay vs. Total scheduling message count

(c) Average claim inter-arrival delay vs. Total ticket count

(d) Average claim inter-arrival delay vs. Total ticket message count

Fig. 8 Effect of $\frac{1}{\lambda_c^{in}}$ and $\frac{1}{\lambda_t^{in}}$ on different performance metrics (Network perspective)

delays of resource tickets (information update delay). The results show that at higher inter-arrival delay of tickets, the jobs in the system experience increased coordination delay. This happens due to the fact that in this case, the resource claim objects for the jobs have to wait for longer period of time before they are hit by ticket objects. As the CPU time (processing time) of a job is not dependent on the resource ticket publish frequency, the average response time function for a job (see Fig. 7(b)) shows the similar growth as the coordination delay function with the changes in $\frac{1}{\lambda_t^{in}}$. However, for a fixed $\frac{1}{\lambda_t^{in}}$, the results show that at higher claim inter-arrival delays, the jobs in the system face comparatively lesser coordination delays. The main reason for this behavior of the coordination protocol is that in this case, the resource claim objects experience less network traffic and competing requests, which lead to an overall decrease in the coordination delay across the system (see Fig. 7(a)).

The proposed coordinated scheduling approach is also highly successful in reducing the number of negotiations undertaken for the successful submission of a job (see Fig. 7(c)). Note that our simulation results show that the negotiation and notification complexity of the proposed coordination protocol has similar bounds as that of a centrally coordinated system, such as the Bellagio. Essentially, with a centrally coordinated protocol the negotiation and notification message complexity is $O(1)$. Simulation results show that GFAs in the system receive on an average 1 coordination notification per job (see Fig. 7(d)) and undertake on an average, close to 1 negotiation per job basis (see Fig. 7(c)). This suggests that the negotiation and notification complexity involved with our scheduling technique is $O(1)$. The negotiation and notification complexity involved with broadcast based scheduling coordination protocols such as [3, 20] is bounded by the function $O(n)$. This shows that our approach is significantly better than the existing broadcast based or centralized coordination protocols.

In case of decentralized setting (i.e. P2P spatial index), a claim object can be mapped to more than one control points based on its spatial extent or query window size. If the control points are assigned to different Grid peers in the network then the problem of duplicate or conflicting notification can arise for a given claim (i.e. a claim is hit by more than one ticket object at more than one Grid peer at the same time). This problem is quite evident in case decentralized spatial index (see Fig. 4). On the other hand, under the centralized setting, the coordination space is hosted and managed by a single machine in the system and all the control points, claims and tickets are stored with it. Therefore, the problem of duplicate or conflicting notifications does not occur here and the notification complexity involved with the centralized approach is bounded by the function $O(1)$. Hence, this proves that our decentralized approach offers same performance as the centralized approach.

9.3.2 Network perspective

Here, we analyze the performance overhead of the DHT-based space as regards to facilitating coordinated scheduling among distributed GFAs. In particular, we measured the following: (i) number of routing hops undertaken per claim and ticket object basis; (ii) total number of claim and ticket objects generated in the system; and (iii) total number of messages generated for the successful mapping of the coordination objects, receiving notifications, and removing duplicate claim objects.

Figure 8(a) shows the number of routing hops, undertaken at different claim and ticket injection rate across the GFAs in the system. Suffixes "t" and "c" are used in the labels in Fig. 8(a) to represent the values for ticket and claim objects, respectively. Recall that the values shown here have been averaged for all the jobs over all the GFAs in the system. As expected, with an increase in the injection rate of the objects, the number of routing hops is not changed significantly. We observed that the average routing hops for mapping the ticket objects is around 3.6, while for the claim objects it is around 3.4. This shows that the routing hops for mapping claim/ticket object is as expected in a Chord based DHT space, i.e. bounded by the function $O(\log n)$. Since $\log_2(100) = 6.64$, it can be easily shown that $c_1 \times 3.6 = 6.64$, where c_1 is a constant.

In Fig. 8(b), we show the total number of messages produced as a result of successfully mapping and executing all the jobs in the system. The measurement shown in this figure includes both mapping and scheduling messages. Mapping messages for a job equals to the number of messages that are required to map a claim object for that job to the coordination space. According to Sect. 8, Lemma 2, the total number of messages produced for all the jobs (refer to Fig. 8(b)) satisfies the theoretical bound on message generation, i.e. $O(m \times n \times \log n)$, where $m = 25$ (number of jobs submitted at a GFA) and $n = 100$. The negotiation and notification complexity involved with broadcast based scheduling coordination protocols such as [3, 20] is bounded by the function $O(m \times n^2)$.

As expected, the number of claims generated during the simulation is constant, irrespective of the claim and ticket inter-arrival delay (we do not show the plot here due to space constraint). This also shows that the number of messages produced as a result of mapping and removing claim objects in the system, remains the same at different claim or ticket inter-arrival delay. However, for the same setup, the number of tickets generated show a linear growth with the increase in claim inter-arrival delay. Figure 8(c) depicts the total number of ticket objects, posted by all GFAs in the system with respect to increasing claim and ticket inter-arrival delay. In Fig. 8(c), we can see that as the claim and ticket inter-arrival delay increase, the number of messages generated during simulation period increases. For instance, when the ticket inter-arrival delay is 120 seconds and the claim inter-arrival delay is 360 seconds, 13,000 tickets as well as 650,000 ticket messages are generated in the system (see Fig. 8(d)). Thus, if the GFAs publish tickets at relatively faster rate, the message overhead of the system increases substantially. But in this case, the average coordination delays per job also decreases moderately (see Fig. 7(a)). Therefore, the ticket inter-arrival delay should be chosen in such a way that a balance between coordination delay and message overhead can co-exist in the system.

10 Conclusion

In this paper, we present a DHT based coordination protocol for efficiently managing the load in federated Grid computing systems. A d-dimensional spatial index forms the basis for distributing claim and ticket objects over a structured Grid broker overlay. Our simulation shows that: (i) the resource claim and ticket object injection rate has significant influence on the coordination delay experienced by distributed users

in the system; (ii) the proposed coordination protocol is highly effective in curbing the number of scheduling negotiation iteration required to be undertaken on a per job basis, the redemption and notification message complexity is bounded by the function $O(1)$; and (iii) on average the number of messages required to successfully map a job to a resource is bounded by the function $O(\log n)$.

One limitation with our approach is that the current index can map a resource claim object to at most 2 index cells. In some cases, this can lead to the generation of unwanted notification messages in the system and may be to an extent sub-optimal load-balancing as well. In our future work, we are going to address this issue by constraining the mapping of a resource claim object to an index cell. Another way to tackle this problem is to make the peers currently managing the same resource claim object to communicate with each other before sending the notifications.

References

1. Abramson D, Buyya R, Giddy J (2002) A computational economy for grid computing and its implementation in the Nimrod-G resource broker. Future Gener Comput Syst 18(8):1061–1074
2. Allen AO (1978) Probability, statistics and queuing theory with computer science applications. Academic Press, San Diego
3. Andrade N, Cirne W, Brasileiro F, Roisenberg P (2003) OurGrid: an approach to easily assemble grids with equitable resource sharing. In: JSSPP'03: proceedings of the 9th workshop on job scheduling strategies for parallel processing, Lecture Notes in Computer Science. Springer, Berlin
4. Auyoung A, Chun B, Snoeren A, Vahdat A (2004) Resource allocation in federated distributed computing infrastructures. In: OASIS '04: 1st workshop on operating system and architectural support for the on-demand IT infrastructure, Boston, MA, October, 2004
5. Raza Butt A, Zhang R, Hu YC (2003) A self-organizing flock of condors. In: SC '03: proceedings of the 2003 ACM/IEEE conference on supercomputing. IEEE Computer Society, Los Alamitos, CA, USA
6. Buyya R, Murshed M (2002) Gridsim: a toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. Concurr Comput Pract Exp 14(13–15):1175–1220
7. Frey J, Tannenbaum T, Livny M, Foster I, Tuecke S (2001) Condor-G: a computation management agent for multi-institutional grids. In: 10th IEEE international symposium on high performance distributed computing (HPDC-10 '01), 2001. IEEE Computer Society, Los Alamitos, pp 237–246
8. Fu Y, Chase J, Chun B, Schwab S, Vahdat A (2003) SHARP: an architecture for secure resource peering. In: SOSP '03: proceedings of the nineteenth ACM symposium on operating systems principles, Bolton Landing, NY, USA. ACM Press, New York, pp 133–148
9. Garca P, Pairot C, Mondjar R, Pujol J, Tejedor H, Rallo R (2005) Planetsim: a new overlay network simulation framework. In: Software engineering and middleware, SEM 2004, Linz, Austria, Lecture Notes in Computer Science. Springer, Berlin, pp 123–137
10. Gong L (2001) JXTA: a network programming environment. IEEE Internet Comput 5(3):88–95
11. Gupta A, Sahin OD, Agrawal D, El Abbadi A (2004) Meghdoot: content-based publish/subscribe over p2p networks. In: Proceedings of the 5th ACM/IFIP/USENIX international conference on middleware, Toronto, Canada. Springer, Berlin, pp 254–273
12. Irwin D, Chase J, Grit L, Yumerefendi A, Becker D, Yocum KG (2006) Sharing networked resources with brokered leases. In: 2006 USENIX annual technical conference, Boston, MA, USA, pp 199–212
13. Lai K, Huberman BA, Fine L (2004) Tycoon: a distributed market-based resource allocation system. Technical report, HP Labs
14. Lublin U, Feitelson DG (2003) The workload on parallel supercomputers: modeling the characteristics of rigid jobs. J Parallel Distrib Comput 63(11):1105–1122
15. Ranjan R, Harwood A, Buyya R (2006) 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

16. Ranjan R, Harwood A, Buyya R (2008) Coordinated load management in peer-to-peer coupled federated grid systems. Technical report GRIDS-TR-2008-2, Grids Laboratory, CSSE Department, The University of Melbourne, Australia

17. Ranjan R, Harwood A, Buyya R (2007) A case for cooperative and incentive based coupling of distributed clusters. Future Generation Computer Systems. Elsevier, Amsterdam (in press). Available online 15 June 2007

18. Rowstron A, Druschel P (2001) Pastry: scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In: Middleware'01: proceedings of IFIP/ACM international conference on distributed systems platforms. Springer, Heidelberg, pp 329–359

19. Samet H (1989) The design and analysis of spatial data structures. Addison-Wesley, Reading

20. Shan H, Oliker L, Biswas R (2003) Job superscheduler architecture and performance in computational grid environments. In: SC '03: proceedings of the 2003 ACM/IEEE conference on supercomputing. IEEE Computer Society, Los Alamitos, pp 44–51

21. Stoica I, Morris R, Karger D, Kaashoek MF, Balakrishnan H (2001) 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, San Diego, California, USA. ACM Press, New York, pp 149–160

22. Tanin E, Harwood A, Samet H (2005) A distributed quad-tree index for peer-to-peer settings. In: ICDE'05: proceedings of the international conference on data engineering. IEEE Computer Society, Los Alamitos, pp 254–255

23. Tanin E, Harwood A, Samet H (2007) Using a distributed quadtree index in peer-to-peer networks. VLDB J 16(2):165–178