A Decentralized and Cooperative Workflow Scheduling Algorithm

Rajiv Ranjan, Mustafizur Rahman, and Rajkumar Buyya
Grid Computing and Distributed Systems (GRIDS) Laboratory
Department of Computer Science and Software Engineering
The University of Melbourne, Australia
{rranjan, mmrahman, raj}@csse.unimelb.edu.au

***Abstract*— In the current approaches to workflow scheduling, there is no cooperation between the distributed workflow brokers and as a result, the problem of conflicting schedules occur. To overcome this problem, in this paper, we propose a decentralized and cooperative workflow scheduling algorithm. The proposed approach utilizes a Peer-to-Peer (P2P) coordination space with respect to coordinating the application schedules among the Grid wide distributed workflow brokers. The proposed algorithm is completely decentralized in the sense that there is no central point of contact in the system, the responsibility of the key functionalities such as resource discovery and scheduling co-ordination are delegated to the P2P coordination space. With the implementation of our approach, not only the performance bottlenecks are likely to be eliminated but also efficient scheduling with enhanced scalability and better autonomy for the users are likely to be achieved. We prove the feasibility of our approach through an extensive trace driven simulation study.**

I. Introduction

Workflow scheduling algorithm is a process of finding the efficient mapping of tasks in a workflow to the suitable resources so that the execution can be completed with the satisfaction of objective functions such as execution time minimization as specified by Grid users. Therefore, the efficiency of the workflow scheduling algorithm directly affects the performance of the system with respect to delivered Quality of Service (QoS), utilization, and system performance.

In the current approaches to workflow scheduling, there is no cooperation between the distributed workflow brokers and as a result, the problem of conflicting schedules can occur. For example, consider a Grid environment (as shown in Fig. 1) that consists of n number of resources and m number of workflow brokers. Users submit their scientific applications to the workflow brokers. These brokers generate the schedules based on the resource information obtained from the Grid Information Services (GIS). A schedule is effectively mapping of the set of tasks in the workflow to the set of available resources. However, if workflow broker 1 and workflow broker 2 query the GIS at the same time, they will get the similar information about the resource availability pattern. Based on this information, workflow broker 1 and workflow broker 2 will generate the same mapping for the tasks in their locally submitted workflows, which will lead to conflicting schedules. Hence, both the system and the application suffer from degraded performance.

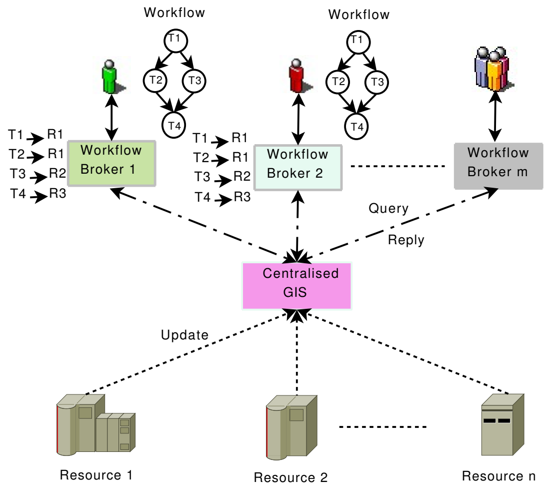


Fig. 1. Existing workflow scheduling approach.

Other major drawback involved with current workflow scheduling is that the existing workflow brokers rely on the centralized (refer to Fig. 1) or semi-centralized hierarchical resource information services such as MDS-2,3,4 [3]. Current studies have shown that [19] existing centralized model for information services do not scale well as the number of users, brokers and providers increase in the system. Hence in case, the centralized links leading to these services fail, then no broker in the system can undertake scheduling related activities due to the lack of up-to-date resource information. In addition, considering the sheer dynamism of Grid computing environment, any scheduling decision that is based on static resource information, would certainly be sub-optimal.

Further, Grids [4] are heterogeneous and dynamic environments consisting of computing, storage and network resources with different capability and availability. In [10], it is shown that the dynamic scheduling algorithms based on heuristics adapt to the changing resource conditions of Grids by performing just-in-time scheduling (generating schedules that maps the tasks dynamically). But the workflow brokers running these dynamic algorithms, still need to be coordinated in order to avoid any conflict and generate schedule globally.

To overcome the limitation of exiting approaches, we propose a fully decentralized and cooperative workflow scheduling algorithm based on Peer-to-Peer (P2P) coordination space.

A P2P coordination space [8], [11] provides a global virtual shared space that can be concurrently and associatively accessed by all participants in the system and the access is independent of the actual physical or topological proximity of the objects or hosts. New generation routing algorithms, which are more commonly known as the Distributed Hash Tables (DHTs) [15], [13] form the basis for organizing the P2P coordination space. In the proposed approach, workflow brokers post their resource demands by injecting a *Resource Claim* object into the DHT-based decentralized coordination space, while resource providers update the resource information by injecting a *Resource Ticket* object. These objects are mapped to the DHT-based coordination space using a spatial hashing technique [16]. Once a resource ticket matches with one or more resource claims, the coordinator space sends *notification* messages to the resource claimers such that it does not lead to the overloading of the concerned resource ticket issuer. Thus, this mechanism prevents the workflow brokers from overloading the same resource.

The main **contributions** of this work include: (i) a novel decentralized and cooperative workflow scheduling algorithm and (ii) extensive simulation based study to prove the effectiveness of the proposed approach.

The rest of this paper is organized as follows. In the next section, we describe the related work that focused on decentralized workflow scheduling. Section 3 provides a brief description of the system models used in our scheduling approach. In Section 4, we present the details of our workflow scheduling algorithm. The simulation model, experimental setups, and the findings of the experiments performed are discussed in Section 5. Finally, we conclude the paper with the direction for future work.

II. Related Work

The main focus of this section is to compare the novelty of the proposed work with respect to the existing decentralized scientific workflow scheduling strategies. In [18], a workflow enactment engine with a just-in-time scheduling system using tuple space is proposed to manage the execution of scientific workflow applications. In this system, every task has its own scheduler called Task Manager (TM) which implements a scheduling algorithm and handles the processing of tasks. The TMs are controlled by a Workflow Coordinator (WC). Besides, an event-driven mechanism with subscription-notification methods supported by the tuple space model is used to control and manage scheduling activities. In this work, although the task managers are working in a distributed fashion, they communicate with each other through tuple space which is implemented based on a client-server based centralized technology. Further, the WC in this system does not communicate with other WCs, managing workflow applications in the Grid. In contrast to this work, we propose a completely decentralized workflow coordinator based on a scalable P2P network model.

Yao et al. [17] have extended the aforementioned architecture with an additive Reinforcement Learning Agent (RLA) to perform the Decentralized Dynamic Workflow Scheduling using Reinforcement Learning (DDWS-RL) algorithm. DDWS-RL enabled TMs query information from the RLA and make decision on resource selection at the time of task execution. Thus, RLA is used in the tuple space to facilitate the scheduling algorithm to be more efficient. However, this approach also involves the same architecture as the other approach, stated above, which is based on centralized client-server model with respect to resource discovery and coordination space management. In contrast, with our approach there is no central component that can prove to be a bottleneck.

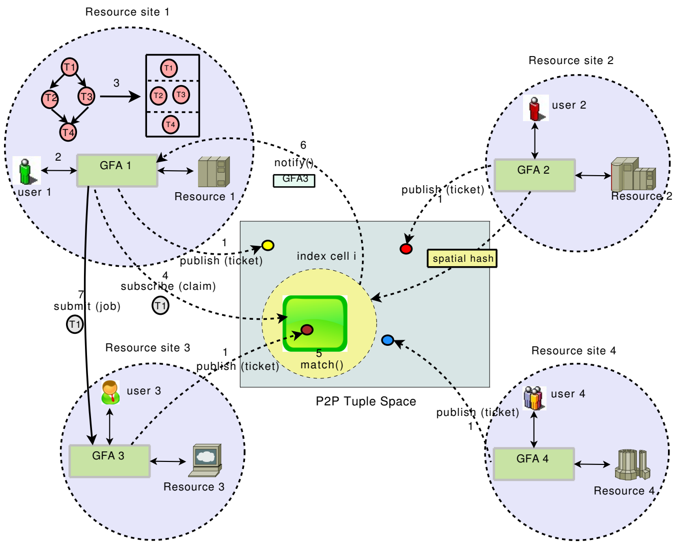


Fig. 2. Proposed workflow scheduling approach.

III. System Models

A. Grid Model

The proposed workflow scheduling algorithm utilizes the *Grid-Federation* [12] model in regards to resource organization and Grid networking. Grid-Federation aggregates distributed resource brokering and allocation services as part of a cooperative resource sharing environment. The Grid-Federation, $G_F = \{R_1, R_2, \ldots, R_n\}$, consists of a number of sites, n, with each site contributing its resource to the federation. Every site in the federation has its own resource description R_i which contains the definition of the resource that it is willing to contribute. R_i, can include information about the CPU architecture, number of processors, memory size, secondary storage size, operating system type, etc. In this work, $R_i = (p_i, x_i, \mu_i, \phi_i)$, which includes the number of processors p_i, processor architecture x_i, their speed μ_i, and installed operating system type ϕ_i.

Resource brokering, indexing and allocation in Grid-Federation are facilitated by a Resource Management System (RMS) known as Grid Federation Agent (GFA). Fig. 2 shows an example Grid-Federation resource sharing model consisting of Internet-wide distributed parallel resources. Every contributing site maintains its own GFA service. GFA service is composed of 3 software entities: Grid Resource Manager (GRM), Local Resource Management System (LRMS) and Distributed Information Manager (DIM) or Grid Peer.

The GRM component of 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 locally submitted jobs (workflows) in the federation. Further, it also manages the execution of remote jobs (workflows) in conjunction with local resource management system. The LRMS software module can be realized using systems such as PBS [1], SGE [6]. Additionally, LRMS performs the other activities for facilitating federation wide job submission and migration process such as answering the GRM queries related to local job queue length, expected response time, and current resource utilization status.

The Grid peer module in conjunction with publish/subscribe indexing service performs tasks related to decentralized resource lookups and updates. A Grid Peer service generates two basic types of objects with respect to coordinated resource brokering: (i) a claim, is an object sent by a broker service to the P2P space for locating the resources that match the user's application requirements and (ii) a ticket, is an update object sent by a Grid site mentioning 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, if the query specifies a range of values for the attributes, then it is referred to as a *d-dimensional Window Query* (DWQ) or a *d-dimensional Range Query* (DRQ) [14]. Thus the Grid peer component of a GFA service is responsible for ticket publication, claim subscription, and overlay management processes.

B. Application Model

In this work, we consider the e-Science workflow applications as the case study for the proposed scheduling approach. Here, a workflow application is modeled as a Directed Acyclic Graph (DAG) where, the tasks in the workflow are represented as nodes in the graph and the dependencies among the tasks are represented as the directed arcs among the nodes. Let $V_{i,j,k}$ be the finite set of tasks $\{T_1, T_2, \ldots, T_x, \ldots, T_y, T_m\}$ for the i-th submitted workflow from the j-th user of k-th workflow broker (GFA) and $E_{i,j,k}$ be the set of dependencies of the form $\{T_{x_{i,j,k}}, T_{y_{i,j,k}}\}$ where, $T_{x_{i,j,k}}$ is the parent task of $T_{y_{i,j,k}}$. Thus, the i-th submitted workflow from the j-th user of k-th workflow broker in the system can be represented as

$$W_{i,j,k} = \{V_{i,j,k}, E_{i,j,k}\}$$

In a workflow, we call a task that does not have any parent task, an entry task and a task that does not have any child task, an exit task. We also assume that a child task can not be executed until all of its parent tasks are completed. At any time of scheduling, the task that has all of its parent tasks finished, is called a ready task.

C. Coordination Space Model

In this section, we first describe the communication, coordination and indexing models that are utilized to facilitate the P2P coordination space. Then, we present the composition of objects, access primitives that form the basis for coordinating application schedules among distributed GFAs/brokers.

1) 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 the Grid-Federation system. These coordination objects include Resource Claim and Resource Ticket.

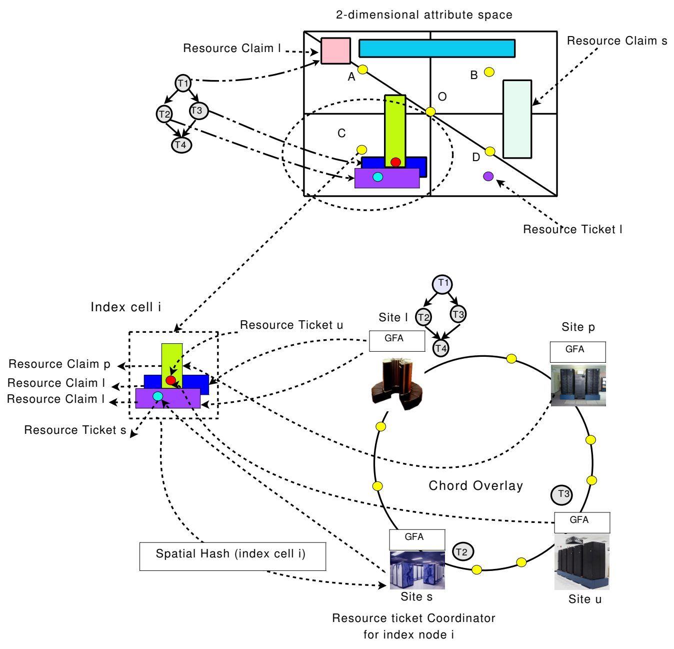


Fig. 3. Resource allocation and application scheduling coordination across Grid sites.

Every GFA in the federation posts its resource ticket through the local Coordination service. A resource ticket object u_i or update query consists of a resource description R_i, for a resource i.

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

A resource claim or lookup object encapsulates the resource configuration requirements of a task in the workflow submitted by the users. In this work, we focus on the workflows for which the requirements are confined to a computational Grid or PlanetLab resources. Users submit their workflow application's resource requirements to the local GFA (refer to Fig. 2). The corresponding GFA service is responsible for searching the suitable resources in the federated system. A GFA aggregates the characteristics of a task including number of processors, processor architecture, and installed operating system with constraint on maximum speed, and resource utilization into a resource claim object, $r_{i,j,k}$.

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

The resource ticket and claim objects are spatially hashed to an index cell i in the d-dimensional coordination space. Similarly, the coordination services of the resource sites in the Grid network hash themselves into this coordination space using the overlay hashing function (SHA-1 in case of Chord and Pastry). In Fig. 3, resource claim objects issued by site p and l are mapped to the index cell i and these claim objects are currently hashed to the site s. Thus 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 small dark circles 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) will be allowed to claim the ticket issued by site u. This load-distribution decision is based on the fact that it should not lead to over-provisioning of resources at site u.

Once a resource ticket matches with one or more resource claims, a coordination service sends *notification* messages to the resource claimers such that it does not lead to the overloading of the concerned resource ticket issuer. Thus, this mechanism prevents the workflow brokers from overloading the same resource. As a result, the problem of conflicting schedules is overcome.

2) D-dimensional Coordination Object Mapping and Routing: 1-dimensional hashing, provided by current implementation of DHTs are 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 support mapping and lookups for complex objects. Management of these objects whose extents lie in the d-dimensional space requires the embedding of 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 space. Providing the background and details on this topic is beyond the scope of this paper; here we only give a high level view. The spatial index that we consider in this work, assigns regions of space to the Grid peers in the Grid-Federation system. 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 this region, as well as storing the objects that are associated with the region. Fig. 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, resulted from this process, remain constant throughout the life of the d-dimensional attribute space and serve as the entry points for subsequent mapping of claim and ticket objects. The number of index cells produced at the minimum division level,f_{min} is always equal to $(f_{min})^{dim}$, where dim is the dimensionality of the Cartesian space. These index cells are called base index cells and they are computed when the Grid peers bootstrap to the coordination network. Finer details on recursive subdivision technique can be found in [16]. Every Grid peer in the network has the basic information about the Cartesian space coordinate values, dimensions and minimum division level.

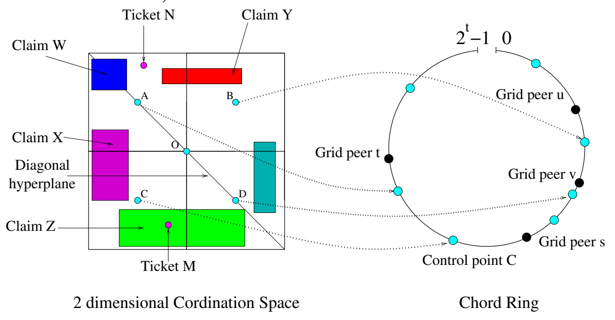


Fig. 4. Spatial resource claims $\{W,X,Y,Z\}$, cell control points $\{A,B,C,D`\}$, point resource tickets $\{M,N\}$ and some of the spacial hashings (dotted lines) to the Chord, i.e., the d-dimensional coordinate values of a cell's control point is used as the DHT key and hashed on to the Chord ring. For this figure, f_{min} = 2, dim=2.

Every cell at the f_{min} level is uniquely identified by its centroid, termed as *control point*. Fig. 4 depicts four control points A, B, C and D. A DHT hashing method such as the Chord method is utilized to hash these control points. So the responsibility for managing an index cell is associated with a Grid peer in the system. In Fig. 4, control point C is hashed to the Grid peer t, which is responsible for managing all claim and ticket objects that are stored with that control point (Claim X, Z and Ticket M).

For mapping claim objects, the process of mapping index cells to the Grid peers depends on whether it is a DPQ or DRQ. For a DPQ type query, the mapping is simple since every point is mapped to only one cell in the Cartesian space. For a DRQ type query, mapping is not always singular because a range lookup can cross more than one cell. To avoid mapping a DRQ to all the cells that it crosses (which can create many unnecessary duplicates), a mapping strategy based on diagonal hyperplane [7] of the Cartesian space is utilized. This mapping involves feeding a DRQ candidate index cell as an input into a mapping function, F_{map}. This function returns the IDs of index cells to which given DRQ should be mapped. 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 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, the mapping process of ticket also involves the identification of the cell in the Cartesian space. A ticket is always associated with a region [7] and all cells that fall fully or partially within that region will be selected to receive the corresponding ticket. The calculation of the region is based upon the diagonal hyperplane of the Cartesian space.

IV. Proposed Algorithms

A. Scheduling Algorithm

In this section, we provide detailed descriptions of the scheduling algorithm that is undertaken by a GFA in the Grid-Federation system following the arrival of a job or workflow:

1. When a workflow application $W_{i,j,k}$ arrives at a GFA, the GFA compiles resource ticket objects for the entry or ready tasks in the workflow. It then posts resource ticket objects for the ready tasks to the P2P tuple space. In Fig. 2 GFA1 is posting a resource claim for the task T1 on behalf of its local user, User1.

2. When a GFA receives a notification of match for a task from the P2P coordination space, it contacts the ticket issuer GFA for possible task submission. After notifying the claimer GFA, the coordination service unsubscribes the resource claim for that task from the tuple space. In Fig. 2, the match event for the submitted task T1 occurs in the tuple space and claimer GFA (GFA1) is notified that it can submit the task to the ticket issuer GFA (GFA3).

3. Once the ticket issuer GFA agrees to grant access to its local resources, then the claimer GFA transfers the locally submitted task to that GFA. Further, the claimer GFA unsubscribes the claim object from the tuple space to remove duplicates.

4. However, if the ticket issuer GFA fails to grant access due to local resource sharing policy, then the claimer GFA reposts the resource claim for that task to the tuple space for future notifications.

B. Coordination Algorithm

The details of the decentralized resource provisioning algorithm that is undertaken by the coordination services across the P2P tuple space is presented in this section.

1. When a resource claim object arrives at a coordination service for future consideration, the coordination service queues it in the existing claim list.

2. When a resource ticket object arrives at a coordination service, the coordination service computes the list of resource claims, which overlap with the submitted resource ticket object in the d-dimensional space. This list is referred to as the match list. The overlap signifies that the task associated with the given claim object can be executed on the ticket issuer's resource subject to its availability.

3. From the match list, the resource claimers are selected one by one based on the First-Come-First-Serve (FCFS) allocation strategy. The coordination service notifies the resource claimers about the resource ticket match until the ticket issuer is not over-provisioned. The coordination procedure can utilize the dynamic resource parameters such as the number of available processors, queue length etc. as the over-provision indicator. These over-provision indicators are encapsulated with the resource ticket object by the GFAs.

4. The GFAs can post the resource ticket object to the tuple space either periodically or whenever the resource condition changes such as a task completion event happens.

V. Performance Evaluation

A. Simulation Model

Our simulation model considers an interconnected network of n Grid peers, where 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. 5). The network messages, delivered through the incoming link (effectively coming from other Grid peers in the overlay) are processed as soon as they arrive. We denote the rates for claim and ticket object by λ_c^{in} and λ_t^{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 claims and tickets that map to the control area currently owned by the Grid peer, and the notification messages arriving from the network. The local index service sends message to its Chord service at a rate, λ_a^{in}. The index service sends notification messages (claim-ticket match message) to its broker service at a rate, μ_r.

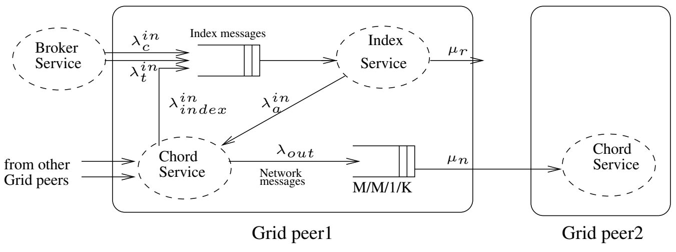


Fig. 5. Network message queueing model at a Grid peer i.

Further, the Chord routing service receives messages from local publish/subscribe index service. These messages are processed as soon as they arrive at the Chord routing service. After processing, Chord routing service queues the messages in the local outgoing queue at a rate, λ_{out}. We denote the message processing rate of outgoing queue by μ_n. 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. The distributions for the delays (including queuing and processing) encountered in an outgoing queue are given by the M/M/1/K queue steady state probabilities.

B. Simulation Setup

Our simulation infrastructure is created by combining two discrete event simulators namely *GridSim* [2], and *PlanetSim* [5]. GridSim offers a concrete base framework for simulation of different kinds of heterogeneous resources, services and application types. PlanetSim is an event-based overlay network simulator that can simulate both unstructured and structured overlays.

1) Workload Configuration: We implement a workflow generator that creates various formats of weighted pseudo-application workflows. The following input parameters are used to create a workflow.

- N, the total number of tasks in the workflow.
- Shape parameter, α: α represents the ratio of the total number of tasks to the width (i.e. maximum number of tasks in a level). Hence, width: $W = \lceil \frac{N}{\alpha} \rceil$.

In this study, we consider fork-join workflow and an example of such workflow is WIEN2K [9], which is a quantum chemistry application developed at Vienna University of Technology. In this kind of workflow, forks of tasks are created and then joined, such that there can be only one entry task and one exit task. But the number of tasks at each level depends on total number of tasks and the width of that level, W. Number of levels in a fork-join workflow is computed as $\lfloor \frac{N}{W+1} \rfloor$. We vary the number of tasks in a workflow over the interval $[20, 100]$ and the size of each task is randomly generated form a uniform distribution between 50000 MI (Million Instruction) to 500000 MI. Further, we assume that workflows are computation intensive. Thus, the data dependency among the tasks in the workflow is negligible.

2) Network Configuration: The experiments run a Chord overlay with 32 bit configuration i.e. number of bits utilized to generate node and key ids. The GFA/broker network size n is fixed to 100. Further, network queue message processing rate, μ_n, is fixed at 4000 messages per second and message queue size, K, is fixed at 10^4.

3) Resource Claim and Ticket Injection Rate: The GFAs inject the ticket objects based on the exponential inter-arrival time distribution. The injection rate, $\frac{1}{\lambda_t^{in}}$, for the resource tickets is distributed over the interval $[100, 300]$ in step of 100 secs. Note that, the inter-arrival delay between injecting the ticket objects is modeled to be the same for all the GFAs in the system. At the beginning of the simulation, the resource claims for the entry tasks of all the workflows in the system are injected. Subsequently, when these tasks finish, then the resource claims for the successive tasks in the workflow are posted. This process is repeated until all the tasks in the workflow are successfully completed. Spatial extent of both resource claims and ticket objects lie in a 4-dimensional attribute space. These attribute dimensions include the number of processors, p_i, their speed, m_i, their architecture, x_i, and operating system type, ϕ_i. The distribution for these resource dimensions is generated by utilizing the configuration of resources that are deployed in the various Grids including NorduGrid, AuverGrid, Grid5000, NaregiGrid, and SHARCNET[1].

4) Spatial Index Configuration: In this simulation, the f_{min} of logical d-dimensional spatial index is set to 4. 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. With dim = 4, total of 256 index cells were produced at the f_{min} level. Hence in a network that consisted of 100 GFAs, on an average the responsibility of managing 2.5 index cells were assigned to each GFA.

5) Resource Load Indicator: 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 stops sending the notifications as the number of processors available with a ticket issuer approaches *zero*.

[1]http://gwa.ewi.tudelft.nl/

C. Results and Observations

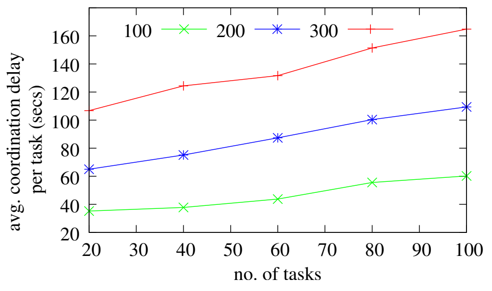
In our simulation, we vary the resource ticket (update) inter-arrival delay over the interval [100, 300] in steps of 100 seconds and the size of the workflow from 20 to 100 tasks. The graphs in Fig. 6 and Fig. 7 show the performance of the proposed scheduling algorithm in terms of scheduling and coordination perspective, respectively.

1) Scheduling perspective: As a measurement of scheduling performance, we use the following metrics namely, average makespan, average coordination delay, average response time, and average number of negotiations. The metric coordination delay sums up the latencies for: (i) resource claim to reach the index cell, (ii) waiting time till a resource ticket matches with the claim, and (iii) notification delay from coordination service to the relevant GFA. CPU time for a task is defined as the time, a task takes to actually execute on a processor. Response time for a task is the delay between the submission time and the arrival time of execution output. Effectively, the response time includes the latencies for coordination and the CPU time. Makespan is measured as the response time of a whole workflow, which equals to the difference between the submission time of the entry task in the workflow and the output arrival time of the exit task in that workflow. Note that, these measurements (except makespan) are collected by averaging the values obtained for each task in the system. The measurement of makespan is taken by averaging over all the workflows in the system.

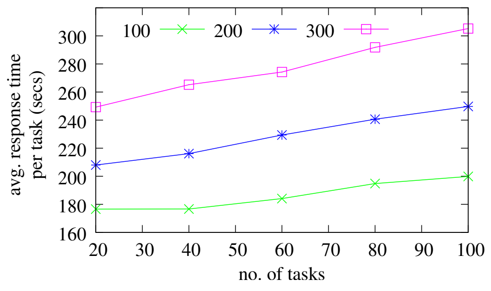
Fig. 6(a) presents the results of average coordination delay for a task with respect to the increase of the number of tasks in a workflow for different inter-arrival delays of resource information update (ticket posting frequency). The results show that at higher inter-arrival delay of tickets, the tasks in a workflow experience increased coordination delay. This happens due to the reason that in this case, the resource claim objects of the corresponding tasks have to wait for longer period of time before they are hit by ticket objects. As the task processing time (CPU time) is not affected by the ticket posting frequency, the average response time for a task shows (refer to Fig. 6(b)) the similar trend as coordination delay with the changes of resource information update delay. However, when the number of tasks in the workflow increases, the resource claim to ticket ratio in the system also increases. This leads to increased coordination delay for each task due to longer waiting period. Therefore, the response time of a task in the workflow is also increased, while the size of workflow increases.

The average makespan of the workflows also shows (see Fig. 6(c)) similar growth over number of tasks and ticket inter-arrival delay as reflected in coordination delay or response time. Thus in our proposed scheduling environment, if the resources update their availability information frequently, then the workflows submitted by the users will be completed early.

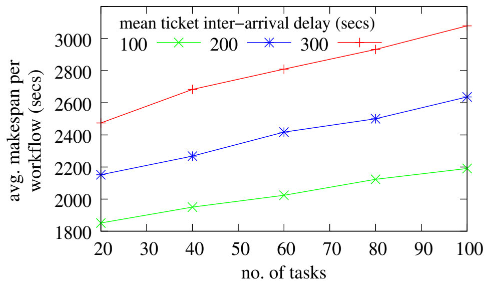
The proposed scheduling approach is also highly successful in reducing the number of negotiations undertaken for the successful submission of a task (see Fig. 6(d)). In a centralized



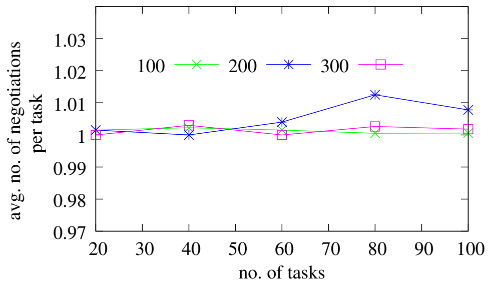
(a) Number of tasks vs. Coordination delay



(b) Number of tasks vs. Response time



(c) Number of tasks vs. Makespan



(d) Number of tasks vs. Number of negotiations

Fig. 6. Effect of workflow size and resource information update interval on different performance metrics (scheduling perspective).

scheduling technique, it requires few negotiation iterations to successfully submit a task, while in this case, the average number of negotiations per task is about one. Moreover, the GFAs/brokers in the system receive on an average, around 1 coordination notification per task. This suggests that the negotiation and notification complexity involved with the scheduling technique is $\Theta(1)$.

2) Coordination perspective: Here, we analyze the performance overhead of the DHT-based coordination space in regards to facilitating coordinated scheduling among the distributed GFAs. For that, we measure the following metrics:(i) number of routing hops, undertaken per task to map claim and ticket objects to index cells (ii) total number of tickets and claims, produced in the system and, (iii) total number of messages, generated for the successfully mapping the coordination objects and receiving notifications.

Fig. 7(a) shows the number of routing hops, undertaken at different ticket injection rates across the GFAs in the system for different sizes of workflow. Suffices *"t"* and *"c"* are used as the label in Fig. 7(a) to represent the values for ticket and claim object, respectively. From the figure, it is evident that the number of routing hops is not changed significantly with the increase of the ticket injection rate or the size of the workflow. Here, the average number of routing hops for mapping ticket or claim objects is around 3.6. This shows that the routing hops for mapping claim/ticket object is as expected in a Chord based routing space, i.e., bounded by the function $O(\log\ n)$. As $\log_2\ (100) = 6.64$, it can be easily shown that $c_1\ \times$ 3.6 $= 6.64$, where c_1 is a constant.

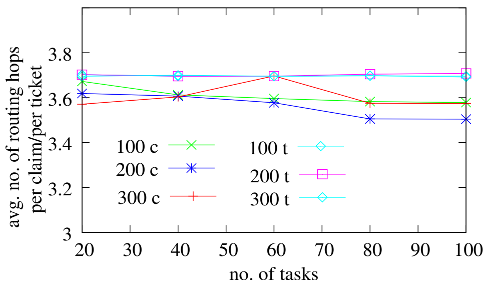
However, the number of claims, generated during the simulation, remains same with the variation of ticket inter-arrival delay (refer to Fig. 7(b)). But it shows a linear growth over the increase in number of tasks in the workflow.

In Fig. 7(c) and 7(d), we show the message overhead, involved with the inter-arrival delay of ticket objects. Fig. 7(c) depicts the total number of ticket objects, posted by all GFAs in the system with respect to increasing workflow size and ticket inter-arrival delay. In Fig. 7(d), we can see that as ticket inter-arrival delay and size of the workflow increase, the number of messages generated during simulation period increases. For instance, when ticket inter-arrival delay is 100 seconds and each workflow consists of 80 tasks, 14000 tickets as well as 410000 messages are generated in the system. Thus if the GFAs publish tickets at relatively faster rate, the message overhead of the system increases substantially. But in this case, average coordination delays per task also decreases moderately (see Fig. 6(a)). Therefore, the ticket inter-arrival delay should be chosen in such a way that a balance between coordination delay and message overhead can exist in the system. In addition, from Fig. 7(d), it is evident that our system is scalable since the total number of messages is increased linearly with respect to number of tasks in the workflow.

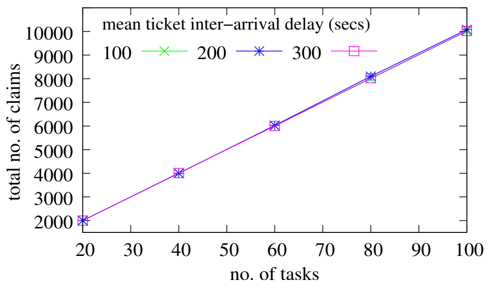
VI. Conclusion and Future Work

In this paper, we have presented a decentralized and cooperative scheduling technique for workflow applications on global Grids. Using simulation, we have measured the performance of our scheduling approach in respect to both scheduling and coordination perspective. The results show that our approach is scalable in terms of scheduling message complexity and makespan. As we leverage the DHT-based coordination space for our scheduling, it can also avoid the limitation of single point of failure regarding resource information service in centralized scheduling techniques.

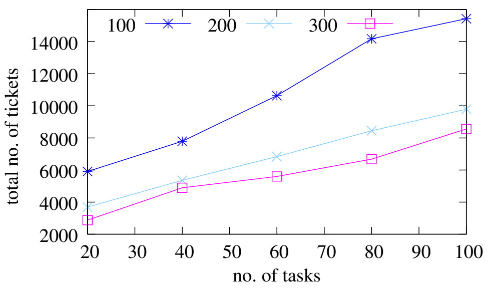
In future, we intend to investigate the incorporation of different optimizations into our proposed algorithm such as spatial hashing of all the tasks in a workflow within a single claim object. This hashing is based on the assumption that the



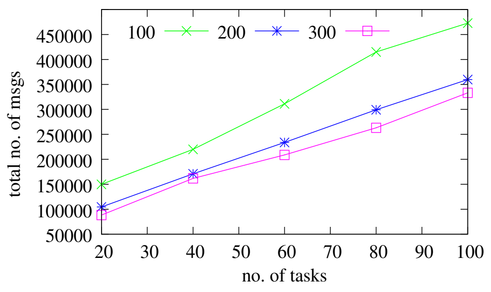
(a) Number of tasks vs. Number of hops



(b) Number of tasks vs. Number of claims



(c) Number of tasks vs. Number of tickets



(d) Number of tasks vs. Number of messages

Fig. 7. Effect of workflow size and resource information update interval on different performance metrics (coordination perspective).

tasks have homogeneous requirements in regards to resource configuration. This approach would be significantly helpful in curbing the number of messages, produced as a result of mapping individual claim objects for different tasks in a workflow. Further, the matchmaking of claim objects to ticket objects is currently based on greedy FCFS approach, which generally leads to suboptimal resource utilization. We want to investigate the matchmaking of claims and tickets based on Back-filling scheduling techniques.

ACKNOWLEDGEMENTS

This work is partially supported by Australian Research Council (ARC) Discovery Project grant.

REFERENCES

[1] B. Bode, D. Halstead, R. Kendall, and D. Jackson. PBS: The portable batch scheduler and the maui scheduler on linux clusters. *Proceedings of the 4th Linux Showcase and Conference, Atlanta, USA*, 2000.

[2] R. Buyya and M. Murshed. Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. *Concurrency and Computation: Practice and Experience, Volume 14, Issue 13-15, pages 1175-1220, Wiley Press*, 2002.

[3] S. Fitzgerald, I. Foster, C. Kesselman, G. von Laszewski, W. Smith, and S. Tuecke. A directory service for configuring high-performance distributed computations. In *Proceedings of 6th IEEE Symposium on High Performance Distributed Computing, Portland, USA*.

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

[5] P. Garca, C. Pairot, R. Mondjar, J. Pujol, H. Tejedor, and R. Rallo. Planetsim: A new overlay network simulation framework. In *Proceedings of Software Engineering and Middleware, Linz, Austria*, 2004.

[6] W. Gentzsch. Sun grid engine: Towards creating a compute power grid. In *Proceedings of 1st IEEE International Symposium on Cluster Computing and the Grid, Brisbane, Australia*, 2001.

[7] A. Gupta, O. D. Sahin, D. Agrawal, and A. El Abbadi. Meghdoot: content-based publish/subscribe over p2p networks. In *Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware, Toronto, Canada*, 2004.

[8] Z. Li and M. Parashar. Comet: A scalable coordination space for decentralized distributed environments. In *Proceedings of the Second International Workshop on Hot Topics in Peer-to-Peer Systems, Sun Diego, USA*, 2005.

[9] P. Blaha nad K. Schwarz, G.K.H. Madsen, D. Kvasnicka, and J. Luitz. Wien2k - an augmented plane wave plus local orbitals program for calculating crystal properties. Technical report, Vienna University of Technology, Austria, 2001.

[10] M. Rahman, S. Venugopal, and R. Buyya. A dynamic critical path algorithm for scheduling scientific workflow applications on global grids. In *Proceedings of the 3rd IEEE International Conference on e-Science and Grid Computing, Bangalore, India*, 2007.

[11] R. Ranjan, A. Harwood, and R. Buyya. Peer-to-peer tuple space: a novel protocol for coordinated resource provisioning. Technical Report GRIDS-TR-2007-14, Grids Laboratory, CSSE Department, The University of Melbourne, Australia, 2007.

[12] R. Ranjan, A. Harwood, and R. Buyya. A case for cooperative and incentive based coupling of distributed clusters. *Future Generation Computer Systems, In Press, Accepted Manuscript, Elsevier Science, The Netherlands*, Available online 15 June 2007.

[13] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware'01: Proc. of IFIP/ACM International Conference on Distributed Systems Platforms*, pages 329–359. SpringerLink, Heidelberg, Germany, 2001.

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

[15] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of ACM SIGCOMM Conference on Applications, technologies, architectures, and protocols for computer communications, San Diego, USA*, 2001.

[16] E. Tanin, A. Harwood, and H. Samet. Using a distributed quadtree index in peer-to-peer networks. *VLDB Journal, Volume 16, Issue 2, pages 165–178*, 2007.

[17] J. Yao, C. K. Tham, and K. Y. Ng. Decentralized dynamic workflow scheduling for grid computing using reinforcement learning. In *Proc. of 14th IEEE International Conference on Networks, Singapore*, 2006.

[18] J. Yu and R. Buyya. A novel architecture for realizing grid workflow using tuple spaces. In *Proceedings of the 5th IEEE/ACM Workshop on Grid Computing, Pittsburgh, USA*, 2004.

[19] X. Zhang, J. L. Freschl, and J. M. Schopf. A performance study of monitoring and information services for distributed systems. In *Proceedings of 12th International Symposium on High Performance Distributed Computing, Seattle, USA*, 2003.