SLA-Based Coordinated Superscheduling Scheme for Computational Grids

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

Abstract

The Service Level Agreement (SLA) based grid superscheduling approach promotes coordinated resource sharing. Superscheduling is facilitated between administratively and topologically distributed grid sites via grid schedulers such as Resource brokers and workflow engines. In this work, we present a market-based SLA coordination mechanism, based on a well known *contract net protocol*.

The key advantages of our approach are that it allows: (i) resource owners to have finer degree of control over the resource allocation which is something that is not possible with traditional mechanisms; and (ii) superschedulers to bid for SLA contracts in the contract net, with focus on completing a job within a user specified deadline. In this work, we use simulation to show the effectiveness of our proposed approach.

1 Introduction

The Grid superscheduling [13] problem is defined as: "scheduling jobs across the grid resources such as computational clusters, parallel supercomputers, desktop machines that belong to different administrative domains". Superscheduling in computational grids is facilitated by specialized superschedulers such as Grid Federation Agent [9], NASA-Superscheduler [14]. In this work, we propose a SLA [8] based coordinated superscheduling scheme for federated grid systems. An SLA is the agreement negotiated between a superscheduler (resource consumer) and LRMSes (resource provider) about acceptable job QoS constraints. These QoS constraints may include the job response time and budget spent. Inherently, a SLA is the guarantee given by a resource provider to a remote site job superscheduler for completing the job within the specified deadline, within the agreed budget or satisfying both at the same time. A SLA-based coordinated job superscheduling approach has many advantages: (i) It inhibits superschedulers from submitting unbounded amounts of work to LRMSes; (ii) once a SLA is reached, users' are certain that agreed QoS shall be delivered by the system; (iii) job queuing or processing delay is significantly reduced, thus leading to enhanced QoS, otherwise a penalty model [17] is applied to compensate them ; and (iv) gives LRMSes more autonomy and better control over resource allocation decisions.

Our SLA model incorporates an economic mechanism [3] for job superscheduling and resource allocation. The economic mechanism enables the regulation of supply and demand of resources, offers incentive to the resource owners for leasing, and promotes QoS based resource allocation. In this work, we mainly focus on the decentralized commodity market model [16]. In this model every resource has a price, which is based on the demand, supply and value. An economy driven resource allocation methodology focuses on: (i) optimizing resource provider's payoff function; and (ii) increasing end-user's perceived QoS value. Note that our proposed superscheduling approach is studied as part of a new and emerging grid system which we call as Grid-Federation [9]. General details about this system can be found in Section 2.

Our SLA model considers a collection of computational cluster resources as a contract net [15]. As jobs arrive, the grid superschedulers undertake one-to-one contract negotiation with the LRMSes managing the concerned resource. The SLA contract negotiation message includes: (i) whether a job can be completed within the specified deadline; and (ii) SLA bid expiration time (maximum amount of time a superscheduler is willing to wait before finalizing the SLA). The SLA bid expiration time methodology we apply here is different from that adopted in the Tycoon system [7]. In Tycoon, the SLA bid expiration time at a resource is the same for all the jobs irrespective of their size or deadline. In this case,

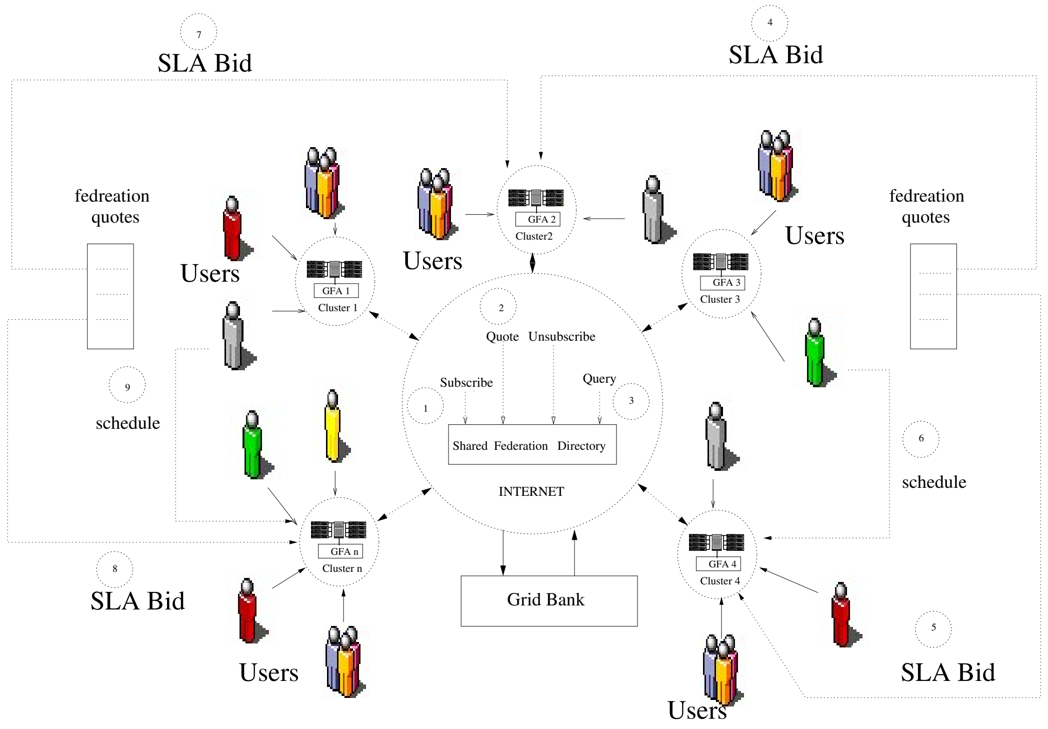


Figure 1. Grid-Federation

the total bid-processing delay is directly controlled by the local resource auctioneer. In our model, the superscheduler bids with a SLA bid expiration time proportional to the job's deadline. The focus is on meeting the job's SLA requirements, particularly the job's deadline.

Our time constrained SLA bid-based contract negotiation approach gives LRMSes finer control over the resource allocation decision as compared to traditional First-Come-First-Serve (FCFS) approach. Existing superscheduling systems including NASA-Superscheduler assumes every LRMS allocates the resources using FCFS scheduling scheme. In this work, we propose a Greedy backfilling LRMS scheduling that focus on maximizing resource owner's payoff function. In this case, a LRMS maintains a queue of SLA bid requests generated by various superschedulers in the system at a time t. Every SLA bid has an associated expiry time. If the concerned LRMS does not reply within that expiry period, then the SLA request is considered to be expired. Greedy backfilling is based on well known Greedy or Knapsack method [6]. The LRMSes periodically iterates through the local SLA bids and finalizes the contract with those that fit the resource owner's payoff function.

The main contribution of this work includes: (i) a SLA bid based superscheduling approach; (ii) a Greedy backfilling cluster scheduling approach for LRMSes that focus on maximizing the resource owners' payoff function; and (iii) allowing resource owners to have a finer degree of control over resource allocation decisions. In this work, we use simulation to evaluate the feasibility of our proposed approach. The paper is organized as follows. In section 2, we present a brief overview of our Grid-Federation superscheduling framework. Section 3.1 presents details about our proposed bid-based SLA contract negotiation model. In section 3.2, we give details about our proposed Greedy backfilling LRMS scheduling approach. In section 4, we present various experiments and discuss our results. We end this paper with concluding remarks and our future vision in Section 5. Note that, in Table. 1 we define the various notations that we use in this paper.

2 Brief overview of Grid-Federation

The Grid-Federation [9] system is defined as a large scale resource sharing system that consists of a cooperative federation of distributed clusters based on policies defined by their owners (shown in Fig.1). Fig.1 shows an abstract model of our Grid-Federation deployed over a shared federation directory. To enable policy based transparent resource sharing between these clusters, we define and model a new RMS system, which we call Grid Federation Agent (GFA). Currently, we assume that the directory information is shared us-

Table 1. Notations

Symbol	Meaning
n	number of Grid Federation Agents (GFAs).
c_i	resource access cost at GFA i.
p_i	number of processors at GFA i.
$J_{i,j,k}$	i-th job from the j-th user of k-th GFA.
$p_{i,j,k}$	number of processor required by $J_{i,j,k}$.
$b_{i,j,k}$	assigned budget to $J_{i,j,k}$.
$d_{i,j,k}$	assigned deadline to $J_{i,j,k}$.
$d^e_{i,j,k}$	effective deadline for $J_{i,j,k}$.
$D(J_{i,j,k}, R_k)$	time function (expected response time for $J_{i,j,k}$ at resource k).
$B(J_{i,j,k}, R_k)$	cost function (expected budget spent for $J_{i,j,k}$ at resource k).
I_k	incentive earned by resource owner k over simulation period.
$\tau(J_{i,j,k})$	returns next SLA bid interval $\Delta tneg_{i,j,k,p}$ for $J_{i,j,k}$.
$tneg_{i,j,k}$	total SLA bid interval/delay for $J_{i,j,k}$.
$Q_{m,t}$	set of jobs that have been assigned but not accepted at GFA m at time t.
$Q^a_{m,t}$	set of jobs that have been accepted at GFA m at time t.
$Q^s_{m,t}$	set of jobs sorted in decreasing order of incentive it provides to the resource owner at GFA m at time t.
n_u	number of users over all clusters ($\sum_{k=1}^{n} n_k$, n_k number of users at GFA k).
n_j	total jobs in the federation ($\sum_{(k,u_i)=1}^{n} n_k, u_i$).
$ts_{i,j,k}$	job submission delay (user to GFA).
$tr_{i,j,k}$	finished job return delay (GFA to user) .
$\Delta tneg_{i,j,k,p}$	total delay for p-th SLA bid for $J_{i,j,k}$.
λ_{SLA_i}	SLA arrival rate at GFA i.
μ_{SLA_i}	SLA satisfaction rate at GFA i.
$l_{i,j,k}$	job length for $J_{i,j,k}$ (in terms of million instructions)
$\alpha_{i,j,k}$	communication overhead for $J_{i,j,k}$

ing some efficient protocol (e.g. a peer-to-peer protocol [12, 5]). In this case, the P2P system provides a decentralized database with efficient updates and range query capabilities. Individual GFAs access the directory information using the interface shown in Fig.1, i.e. subscribe, quote, unsubscribe, and query. The specifics of the interface can be found in [10]. Our approach considers the emerging computational economy metaphor for the Grid-Federation. Some of the commonly used economic models [3] in resource allocation includes the commodity market model, the posted price model, the bargaining model, the tendering/contract-net model, the auction model, the bid-based proportional resource sharing model, the community/coalition model and the monopoly model. Grid-Federation considers decentralized commodity market model for managing job scheduling and resource allocation. In this case, the resource owners: (i) can clearly define what is shared in the Grid-Federation while maintaining a complete autonomy; (ii) can dictate who is given access; and (iii) get incentives for leasing their resources to federation users.

In Fig.1 a user who is local to GFA 3 is submitting a job. If the user's job QoS can't be satisfied locally then GFA 3 queries the federation directory to obtain the quote of the 1-st fastest (if the user is seeking optimize for time (OFT)) or 1-st cheapest cluster (if the user is seeking to optimize for cost (OFC)). In this case, the federation directory returns the quote advertised by GFA 2. Following this, GFA 3 bids for SLA contract (enquiry about QoS guarantee in terms of response time) at GFA 2. If GFA has too much load or the SLA bid does not fit the resource owner payoff function, the bid eventually timeouts. In this case, the SLA bid by GFA 2 times out. As the next superscheduling iteration, GFA 3 queries the federation directory for the 2-nd cheapest/fastest GFA and so on. The process of SLA bids is repeated until GFA 3 finds a GFA that can schedule the job (i.e. accept the SLA bid) (in this example the job is finally scheduled on cluster 4).

3 Models

3.1 SLA model

The SLA model we consider is that of a set of distributed cluster resources each offering a fixed amount of processing power. The resources form part of the federated grid environment and are shared amongst the end-users, each having its own SLA parameters. SLAs are managed and coordinated through an admission control mechanism enforced by GFA at each resource site. Each user in the federation has a job $J_{i,j,k}$. We write $J_{i,j,k}$ to represent the i-th job from the j-th user of the k-th resource. A job consists of the number of processors required, $p_{i,j,k}$, the job length, $l_{i,j,k}$ (in terms of millions of instructions), the communication overhead, $\alpha_{i,j,k}$ and SLA parameters the budget, $b_{i,j,k}$, the deadline or maximum delay, $d_{i,j,k}$. More details about the job model can be found in [9].

3.1.1 SLA bid with expiration time (based on contract net protocol [15])

The collection of GFAs in the federation are referred to as a contract net, and job-migration in the net is facilitated through the SLA contracts. Each GFA can take on two roles either a *manager* or *contractor*. The GFA to which a user submits a job for processing is referred to as the manager GFA. The manager GFA is responsible for superscheduling the job in the net. The GFA which accepts the job from the manager GFA and overlooks its execution is referred to as the contractor GFA. Individual GFAs are not assigned these roles in advance. The role may change dynamically over time as per the user's job requirements. Thus, the GFA alternates between these two roles or adheres to both over the course of superscheduling.

As jobs arrive at a GFA, the GFA adopts the role of a manager. Following this, the manager GFA queries the shared federation directory to obtain the quote for the contractor GFA that matches the user specified SLA parameters. Note that, users can seek optimization for one of the SLA parameters i.e. either response time (OFT) or the budget spent (OFC). Once, the manager obtains the quote for the desired contractor, it undertakes

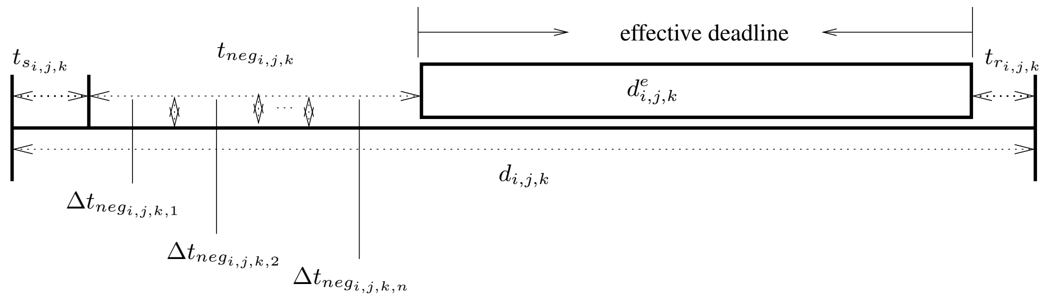


Figure 2. Job superscheduling timeline

one-to-one SLA contract negotiation with the contractor. The SLA contract negotiation message includes: (i) whether the job $J_{i,j,k}$ can be completed within the specified deadline; and (ii) SLA bid expiration time $\Delta t_{negi,j,k,l}$. The contractor GFA has to reply within the bid time $\Delta t_{negi,j,k,l}$, otherwise the manager GFA undertakes SLA contract negotiation with the next available contractor in the net. Algorithm SLA bidding mechanism (refer to Algorithm 1) depicts various events and corresponding superscheduling actions undertaken by a GFA.

Our SLA contract model considers a part of the total job deadline as the SLA contract negotiation time (refer to Eq. 1). The manager GFA bids with a different SLA expiration interval given by Eq. 2. In Fig. 2 we show the job superscheduling timeline. The timeline includes the job submission delay, $t_{s_{i,j,k}}$, total SLA contract negotiation delay, $t_{neg_{i,j,k}}$, expected response time (computed using Eq. 1) and finished job return delay, $t_{r_{i,j,k}}$. The total SLA contract bidding delay available to the manager GFA for superscheduling job $J_{i,j,k}$ is given by:

$$t_{neg_{i,j,k}} = d_{i,j,k} - t_{s_{i,j,k}} - d^e_{i,j,k} - t_{r_{i,j,k}} \qquad (1)$$

The total SLA contract bid negotiation delay $t_{neg_{i,j,k}}$ assumes a finite number of values $\Delta t_{neg_{i,j,k,1}}$, $\Delta t_{neg_{i,j,k,2}}$,...,$\Delta t_{neg_{i,j,k,n}}$ in superscheduling a job $J_{i,j,k}$ (refer to Fig. 2). We define the value of $\Delta t_{neg_{i,j,k,l}}$ by

$$\Delta t_{neg_{i,j,k,l}} = \frac{t_{neg_{i,j,k}} - \sum_{p=1}^{l-1} \Delta t_{neg_{i,j,k,p}}}{2}, l > 0 \qquad (2)$$

Note that, the value for $\Delta t_{neg_{i,j,k,l}}$ can be given by other distributions [2] such as uniform or random. We intend to analyze various distributions for SLA bid interval and study its effect on our proposed superscheduling approach in our future work. For simplicity, in this work we use the distribution given by Eq. 2.

As the superscheduling iteration increases, the manager GFAs give less time to the contractor to decide on the SLA in order to meet the user's job deadline. This approach allows a large number of scheduling iterations to the manager GFA. However, if the user's SLA parameters cannot be satisfied (after iterating up to the greatest r such that GFA could feasibly complete the job), then the job is dropped. To summarizes, a SLA bid for job $J_{i,j,k}$ includes:

- l-th SLA bid expiry interval $t_{neg_{i,j,k,l}}$ (computed using Eq. 2);
- expected response time ($d^e_{i,j,k}$) (computed using Eq. 1).

We consider the function:

$$\tau : J_{i,j,k} \longrightarrow \mathcal{Z}^+ \qquad (3)$$

which returns the next allowed SLA bidding time interval $\Delta t_{neg_{i,j,k,p}}$ for a job $J_{i,j,k}$ using Eq.2.

3.2 Greedy backfilling: (LRMS scheduling model)

Most of the existing LRMSes apply system-centric policies for allocating jobs to resources. Some of the well known system-centric policies include: (i) FCFS; (ii) Conservative backfilling; and (ii) Easy backfilling. Experiments have shown that the job backfilling approach offers significant improvement in performance over the FCFS scheme. However, these system centric approaches allocate resources based on parameters that enhance system utilization or throughput. The LRMS either focuses on minimizing the response time (sum of queue time and actual execution time) or maximizing overall resource utilization of the system, and these are not specifically applied on a per-user basis (user oblivious). Further, the system centric LRMSes treat all resources with the same scale, thus neglecting the resource owner payoff function. In this case, the resource owners

Algorithm 1: SLA bidding mechanism

```
0.1  PROCEDURE: SLA_BIDDING_MECHANISM
0.2  begin
0.3      begin
0.4          SUB-PROCEDURE:
             EVENT_USER_JOB_SUBMIT ($J_{i,j,k}$)
0.5          call SLA_BID ($J_{i,j,k}$).
0.6      end
0.7      begin
0.8          SUB-PROCEDURE: SLA_BID ($J_{i,j,k}$)
0.9          Send SLA bid for job $J_{i,j,k}$ to the next available
             contractor GFA (obtained by querying the shared
             federation directory).
0.10     end
0.11     begin
0.12         SUB-PROCEDURE:
             EVENT_SLA_BID_REPLY ($J_{i,j,k}$)
0.13         if SLA Contract Accepted then
0.14             Send the job $J_{i,j,k}$ to accepting GFA.
0.15         end
0.16         else
0.17             call SLA_BID_TIMEOUT ($J_{i,j,k}$).
0.18         end
0.19     end
0.20     begin
0.21         SUB-PROCEDURE:
             SLA_BID_TIMEOUT($J_{i,j,k}$)
0.22         if $\tau$ ($J_{i,j,k}$) $\geq 0$ then
0.23             call SLA_BID ($J_{i,j,k}$).
0.24         end
0.25         else
0.26             Drop the job $J_{i,j,k}$.
0.27         end
0.28     end
0.29 end
```

do not have any control over resource allocation decisions. While in reality the resource owner would like to dictate how his resources are made available to the outside world and apply a resource allocation policy that suits his payoff function. To summarize, the system-centric approaches do not provide mechanisms for resource owners to dictate resource: (i) sharing; (ii) access and ; (iii) allocation policies.

To address this, we propose a Greedy method based resource allocation heuristic for LRMSes. Our proposed heuristic focuses on maximizing payoff function for the resource owners. The heuristic is based on the well known Greedy method. The Greedy method for solving optimization problems considers greedily maximizing or minimizing the short-term goals and hoping for the best, without regard to the long-term effects. This method has been used to solve *the knapsack problem* [6]. Greedy method considers a set S, consisting of n items. Each item i has a associated positive benefit b_i and a positive weight w_i. Given the knapsack capacity W, the Greedy heuristic focuses on maximizing the total benefit $\sum_{i \in S^a} b_i(x_i/w_i)$ with constraint $\sum_{i \in S^a} x_i \leq W$, such that $S^a \subseteq S$. In this case, x_i being the part of item i

Algorithm 2: Greedy-Backfilling

```
1.1  PROCEDURE: GREEDY_BACKFILLING
1.2  begin
1.3      $r = p_m$
1.4      $c = 0$
1.5      $Q_{m,t} \leftarrow \phi$
1.6      $Q^a_{m,t} \leftarrow \phi$
1.7      $Q^s_{m,t} \leftarrow \phi$
1.8      begin
1.9          SUB-PROCEDURE:Event_SLA_Bid_ARRIVAL($J_{i,j,k}$)
1.10         A SLA request message for the job $J_{i,j,k}$ that arrives at a
             GFA $Q_{m,t} \leftarrow Q_{m,t} \cup \{J_{i,j,k}\}$
1.11         Schedule the SLA bid timeout event after $\tau(J_{i,j,k})$ time
             units
1.12         call STRICT_GREEDY()
1.13     end
1.14     begin
1.15         SUB-PROCEDURE:Event_SLA_Bid_Timeout($J_{i,j,k}$)
1.16         A SLA bid for job $J_{i,j,k}$ that reaches timeout period
1.17         if ($r \geq p_{i,j,k}$ and $d^e_{i,j,k} \geq D(J_{i,j,k}, R_m)$) then
1.18             Call RESERVE($J_{i,j,k}$)
1.19         end
1.20         else
1.21             Reject the SLA bid for job $J_{i,j,k}$
1.22             Reset $Q_{m,t} \leftarrow Q_{m,t} - \{J_{i,j,k}\}$
1.23         end
1.24     end
1.25     begin
1.26         SUB-PROCEDURE:Event_Job_Finish($J_{i,j,k}$)
1.27         A job $J_{i,j,k}$ that finishes at a GFA  Reset $r = r + p_{i,j,k}$
1.28         call STRICT_GREEDY()
1.29     end
1.30     begin
1.31         SUB-PROCEDURE: RESERVE($J_{i,j,k}$)
1.32         Reserve $p_{i,j,k}$ processors for the job $J_{i,j,k}$
1.33         Reset $r = r - p_{i,j,k}$, $Q_{m,t} \leftarrow Q_{m,t} - \{J_{i,j,k}\}$,
             $Q^a_{m,t} \leftarrow Q^a_{m,t} \cup \{J_{i,j,k}\}$
1.34     end
1.35     begin
1.36         SUB-PROCEDURE: STRICT_GREEDY()
1.37         Reset $c = 0$
1.38         Sort SLA bids in $Q_{m,t}$ in decreasing order of incentives
             and store in $Q^s_{m,t}$
1.39         Get next SLA bid for job $J_{i,j,k}$ from the list $Q^s_{m,t}$, c=c+1
1.40         if ($r \geq p_{i,j,k}$ and $d^e_{i,j,k} \geq D(J_{i,j,k}, R_m)$) then
1.41             Call RESERVE($J_{i,j,k}$)
1.42         end
1.43         else
1.44             if $c <$ sizeof($Q^s_{m,t}$) then
1.45                 Iterate through step 1.39
1.46             end
1.47         end
1.48     end
1.49 end
```

selected by the Greedy method.

Fig.3 shows the queue of SLA bids at each site in the federation. Every incoming SLA bid is added to the LRMS request queue, $Q_{m,t}$ and a bid expiration timeout event is scheduled after time interval $\tau(J_{i,j,k})$. Every resource i has a different SLA bid arrival rate, λ_{SLA_i} and SLA bid satisfaction rate, μ_{SLA_i}. The LRMS scheduler iterates through the SLA bid queue in case any of the following events occur: (i) new SLA bid arrives to the site; (ii) job completion; or (iii) SLA bid reaches its expiration time. Procedure Greedy backfilling (refer to Algorithm 2) depicts various events and corresponding scheduling actions undertaken by the LRMS.

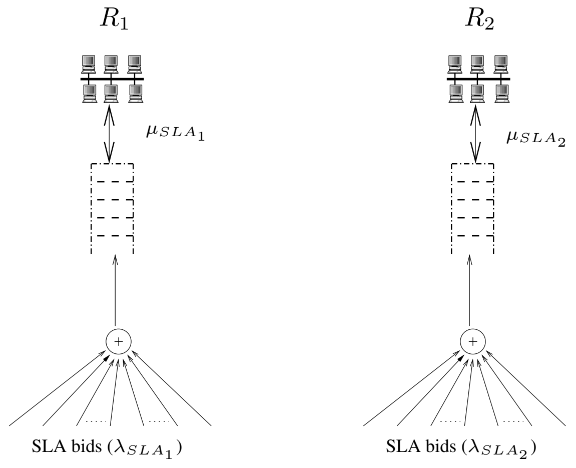


Figure 3. SLA bid queues in Grid-Federation

3.3 Integer linear programming (ILP) formulation of scheduling heuristic

Queue, $Q_{m,t}$, maintains the the set of job SLA bids currently negotiated with the LRMS at GFA m by time t. We consider the SLA bid acceptance variable $x_{i,j,k}$

- Definition of variable:

  $x_{i,j,k} = 1$ if the SLA request for job $J_{i,j,k}$ is accepted;

  $x_{i,j,k} = 0$ otherwise.

  The Greedy-Backfilling heuristic accepts SLA requests constrained to the availability of number of processors requested for job $J_{i,j,k}$ and expected response time $d^e_{i,j,k}$.

- Definition of the constraints:

$$\sum_{\substack{1 \le i \le n_j \\ 1 \le j \le n_u \\ 1 \le k \le n}} p_{i,j,k} \le p_m \qquad (4)$$

  p_m total number of processors available at a LRMS (GFA) m. $p_{i,j,k}$ denotes number of processor requested during the SLA bid for job $J_{i,j,k}$. All the accepted SLA bids for jobs are maintained in the queue $Q^a_{m,t}$.

- Payoff or Objective function: The LRMS scheduler accepts SLA bids for the jobs such that it maximizes the resource owners' payoff function by applying the Greedy backfilling heuristic

$$I_m = max(\sum_{\substack{1 \le i \le n_j \\ 1 \le j \le n_u \\ 1 \le k \le n \\ 1 \le m \le n}} B(J_{i,j,k}, R_m)) \qquad (5)$$

Note that, models for economic parameters i.e. how resource owners determine their price and how jobs are assigned deadline and budget can be found in [11].

4 Experiments and observations

4.1 Workload and resource methodology

We performed trace based simulation to evaluate the effectiveness of our SLA-based superscheduling approach. The workload trace data was obtained from [1]. The trace contains real time workload of various resources/supercomputers including CTC SP2, KTH SP2, LANL CM5, LANL Origin, NASA iPSC, SDSC Par96, SDSC Blue, SDSC SP2 (See Table 2). The simulator was implemented using the GridSim [4] toolkit that allows modeling and simulation of distributed system entities for evaluation of scheduling algorithms. The simulation experiments were conducted by utilizing workload trace data over the total period of four days (in simulation units) at all the resources. We consider federation with computational economy mechanism as the resource sharing environment for our experiments.

4.2 Experiment 1 - Quantifying scheduling parameters related to resource owners and end-users with varying total SLA bid time

We performed the simulations which comprised of end-users seeking OFT for their jobs (i.e. 100% users seek OFT). We vary the total SLA bid from 0% to 50% of total allowed job deadline. In case, no SLA bid delay is allowed (i.e. 0% of total allowed deadline) then the contacted GFA has to immediately make the admission control decision. In this case, we simulate FCFS based strategy for finalizing the SLA. However, in other cases we consider a Greedy backfilling SLA approach. Note that, due to space constraint we could not include all the details in this paper. Hence, the interested readers are adviced to refer to the report [11] for additional experiments and results.

4.3 Results and observations

4.3.1 Federation perspective

In experiment 1, we measure how varying of the total time for SLA bids coupled with Greedy backfill-

Table 2. Workload and Resource Configuration

Index	Resource / Cluster Name	Trace Date	Processors	MIPS (rating)	Jobs	Quote(Price)	NIC to Network Bandwidth (Gb/Sec)
1	CTC SP2	June96-May97	512	850	79,302	4.84	2
2	KTH SP2	Sep96-Aug97	100	900	28,490	5.12	1.6
3	LANL CM5	Oct94-Sep96	1024	700	201,387	3.98	1
4	LANL Origin	Nov99-Apr2000	2048	630	121,989	3.59	1.6
5	NASA iPSC	Oct93-Dec93	128	930	42,264	5.3	4
6	SDSC Par96	Dec95-Dec96	416	710	38,719	4.04	1
7	SDSC Blue	Apr2000-Jan2003	1152	730	250,440	4.16	2
8	SDSC SP2	Apr98-Apr2000	128	920	73,496	5.24	4

ing resource allocation strategy affects the Grid participants across the federation. We quantify how the additional decision making time given to the LRMSes before finalizing the SLA contracts affects the overall system performance in terms of resource owner's and end-user's objective functions. We observed that when the LRMSes across the federation applied FCFS technique for finalizing the SLAs (i.e. no decision making time was given, so the LRMSes have to reply as soon as the SLA request was made), the resource owner's made 4.102×10^9 grid dollars as incentive (refer to Fig.4(a)).

We observed that with an increase in the total SLA bidding time (i.e. as the LRMSes were allowed decision making time before finalizing the SLAs hence they applied Greedy backfilling scheduling on the queue of SLA bids), the resource owners earned more incentive as compared to the FCFS case. When 10% of the total deadline was allowed for SLA bids, the total incentive earned across the federation increased to 4.219×10^9 grid dollars. While, in case 50% of total job deadline was allowed for the SLA bids, the total incentive accounted to 4.558×10^9 grid dollars. Hence, the resource owners across federation exprienced an increase of approximately 10% in their incentive as compared to the FCFS case.

However, we observed that with an increase in the total SLA bid delay, the end-users across the federation experienced degraded QoS. During the FCFS case, the average response time across the federation was 1.183×10^4 sim units (refer to Fig.4(b)). However, in case of 10% SLA bid delay the average response time increased to 1.344×10^4 sim units. Finally, when 50% of the total job deadline was allowed as SLA bid delay the average response time further increased to 1.956×10^4 sim units. Furthermore, in this case the end-users end up spending more budget as compared to the FCFS case (refer to Fig.4(c)).

Hence, we can see that although the proposed approach leads to better optimization of resource owners' payoff function, it has degrading effect on the end-user's QoS satisfaction function across the federation.

5 Conclusion and future work

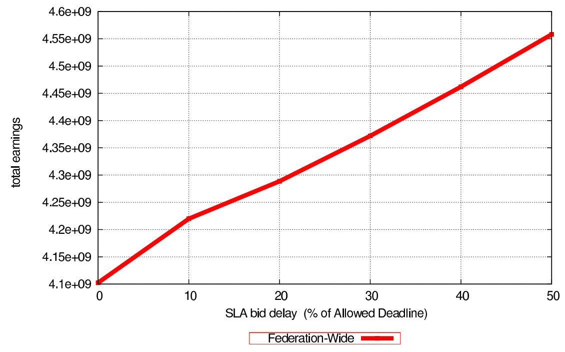
In this paper, we presented an SLA-based superscheduling approach based on the contract net protocol. The proposed approach models a set of resource providers as a contract net while job superschedulers work as managers, responsible for negotiating SLA contracts and job superscheduling in the net. Superschedulers bid for SLA contracts in the net with a focus on completing the job within the user specified deadline. We analyzed how the varying degree of SLA bidding time (i.e. admission control decision making time for LRMSes) affects the resource providers' payoff function. The results show that the proposed approach gives resource owners finer control over resource allocation decisions. However, the results also indicate that the proposed approach has a degrading effect on the user's QoS satisfaction. However, we need to do more research on abstracting the user's QoS requirement. We need to analyze how the deadline type for the user jobs can be abstracted into different types such as into urgent and relaxed deadline. In these cases, jobs with an urgent requirement can be given a preference while finalizing SLA contracts hence providing improved QoS satisfaction to users. In our future work we will study to what extent the user profile can change and how pricing polices for resources leads to varied utility of the system. We also intend to look into simultaneously bidding for SLA contracts at multiple contractors in the net, for a superscheduling iteration l for a job $J_{i,j,k}$. This approach can increase the end-user's QoS satisfaction in terms of response time, as in this case the total waiting time per SLA bid is greatly reduced.

References

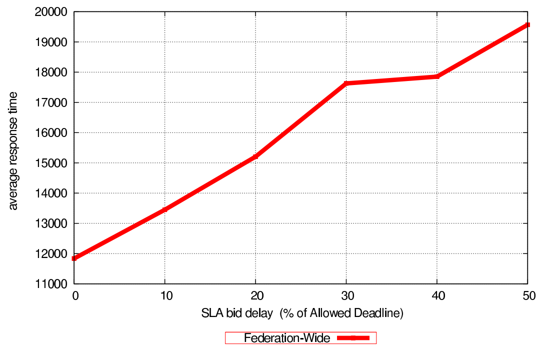
[1] *http://www.cs.huji.ac.il/labs/parallel*.

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

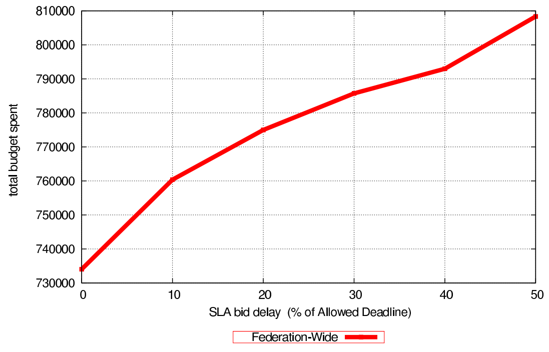
[3] R. Buyya, D. Abramson, J. Giddy, and H. Stockinger. Economic models for resource management and scheduling in grid computing. *Special Issue on Grid computing Environment, The Journal of concurrency and Computation:Practice and Experience (CCPE), Volume 14, Issue 13-15, Wiley Press*, 2002.



(a) total SLA bid delay vs. total federation earning (grid dollars)



(b) total SLA bid delay vs. average response time (sim units)



(c) total SLA bid delay vs. average budget spent (grid dollars)

Figure 4. Federation perspective

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

[5] M. Cai, M. Frank, J. Chen, and P. Szekely. Maan: A Multi-atribute addressable network for grid information services. *Proceedings of the Fourth IEEE/ACM International workshop on Grid Computing;Page(s):184 - 191*, 2003.

[6] G. Gambosi, A. Postiglione, and M. Talamo. Algorithms for the relaxed online bin-packing model. *SIAM J. Comput.*, 30(5):1532–1551, 2001.

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

[8] D. Ouelhadj, J. Garibaldi, J. MacLaren, R. Sakellariou, and K. Krishnakumar. A multi-agent infrastructure and a service level agreement negotiation protocol for robust scheduling in grid computing. In *Proceedings of the European Grid Conference*. Lecture Notes in Computer Science, Springer-Verlag, 2005.

[9] R. Ranjan, R. Buyya, and A. Harwood. A case for cooperative and incentive based coupling of distributed clusters. In *Proceedings of the 7th IEEE International Conference on Cluster Computing (CLUSTER'05), Boston, MA*.

[10] R. Ranjan, A. Harwood, and R. Buyya. Grid-federation: A resource management model for cooperative federation of distributed clusters. *Technical Report, GRIDS-TR-2004-10, Grid Computing and Distributed Systems Laboratory, University of Melbourne, Australia*, 2004.

[11] R. Ranjan, A. Harwood, and R. Buyya. SLA-based coordinated superscheduling scheme and performance for computational grids, http://arxiv.org/abs/cs.dc/0605057, 2006.

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

[13] J. Schopf. Ten actions when superscheduling. In *Global Grid Forum*, 2001.

[14] H. Shan, L. Oliker, and R. Biswas. Job superscheduler architecture and performance in computational grid environments. In *SC '03: Proceedings of the 2003 ACM/IEEE conference on Supercomputing*, page 44, Washington, DC, USA, 2003. IEEE Computer Society.

[15] R. G. Smith. The contract net protocol: high-level communication and control in a distributed problem solver. pages 357–366, 1988.

[16] R. Wolski, J. S. Plank, J. Brevik, and T. Bryan. G-commerce: Market formulations controlling resource allocation on the computational grid. In *IPDPS '01: Proceedings of the 15th International Parallel & Distributed Processing Symposium*, page 46, Washington, DC, USA, 2001. IEEE Computer Society.

[17] C. Yeo and R. Buyya. Service level agreement based allocation of cluster resources: Handling penalty to enhance utility. In *Proceedings of the 7th IEEE International Conference on Cluster Computing (CLUSTER'05), Boston, MA*.