

QoS-GRAF: A Framework for QoS based Grid Resource Allocation with Failure provisioning

Gargi B Dasgupta, Koustuv Dasgupta, Amit Purohit and Balaji Viswanathan

IBM India Research Lab

Email: {gdasgupt,kdasgupta,ampurohi,bviswana}@in.ibm.com

I. INTRODUCTION

For a long time grid computing was viewed as the key technology for harnessing the compute power of high performance virtual supercomputers connected across a WAN. However, with the emergence of the utility grid computing paradigm, it has become increasingly desirable for organizations worldwide to improve efficiency by outsourcing their business processes, and for providers to supply grid capacity with the primary goals of charging for service usage and high utilization of shareable computing. IBM's Deep Computing Center on Demand (DCCoD) and the Sun Grid are examples of on-demand grids providing large and small companies access to their supercomputing power. In this utility model, a successful and competitive provider must be able to maintain customer goodwill by satisfying the demand for high-quality service, and at the same time constantly look towards improving its efficiency via business process management technologies. At the heart of these technologies lies the problem of business revenue maximization. Differentiated pricing schemes offered by customers, associated with different levels of QoS expectations, empowers the provider with the capability of business revenue maximization by way of quality of service differentiation. We outline some of the key design issues involved in providing business profit maximization for grid service providers.

A. SLA-based Service Differentiation

Service Level Agreements(SLA) are emerging as the standard concept based on which the grid utility computing model is shaping up [1]. SLA captures the high-level service requirements, commonly termed as Service Level Objectives (SLOs), as well as the business value associated with satisfying these requirements. Multi level SLAs, with each level having its own SLOs and business value are also popular. In essence, the SLOs along with pricing serve as an incentive for providers to perform service differentiation and revenue maximization. In order to satisfy customer requirements with guarantees, the SLOs need to be translated into corresponding resource requirements [2], that can be monitored, managed, and provisioned. These resource requirements can then be satisfied using provisioning techniques proposed in [3] that deliver end-to-end QoS guarantees using *advance reservation*.

B. Demand-Revenue

It is possible to determine a *demand-revenue* function that captures the relationship between the resource demand for a certain level of SLA and the price paid to a provider for meeting it. We consider demand-revenue functions that have non-decreasing revenue with increase in demand. From the customer's point of view, it is intuitive to adopt a step-wise demand-revenue function [4]. We assume that a job has a demand-revenue function associated with a resource type. Each step in this function represents a particular SLA level. Each SLA level represents revenue earned if demand at that level is met. A job could be dependent on multiple resource types. We call it dependencies of the job. Dependencies can have differing importance for a particular job, hence it is reasonable to assume that, dependencies can have different associated demand-revenue functions.

C. Failure Provisioning

QoS service guarantees need to be met even in case of resource failures/downtime, which are not uncommon in the grid composed of heterogeneous resources and services. Since SLAs often have high penalty clauses, it is imperative for grid service providers to additionally provision resources for failures. Failure resource provisioning should be done apriori to avoid any SLA violations (and as a result revenue loss) due to delays in resource discovery/provisioning or unavailability. Advance failure provisioning guarantees efficient switch to backup resource in case of a failure. SLA pricing policy based provisioning helps jobs to be backed up at their optimal SLA level to minimize the overall revenue loss. Given the primary resource allocation for a job, the central issue in apriori failure provisioning is to determine the amount of backup resource for that job. The amount of resources dedicated for backup reservation represents the percentage of overbooking in the system. One way to reduce overbooking in a system is to share backup resources among jobs that can be guaranteed never to fail simultaneously.

D. Tying the knot: Service Differentiation with Failure Provisioning

We address the problem of business profit maximization by considering two important principles: SLA based service differentiation and failure provisioning. SLA based business profit maximization has been studied for web servers [4]. [5] proposes a business process management framework that

enables optimizing the business performance of executing workflows by associating a business-value function with each process. However, in the grid environment a job typically has multiple dependencies and aforementioned approaches fail to address this case. [6] proposes a co-selection solution, where business values of jobs are interpreted from their priorities. However, in general, a job could have different QoS requirements (with different price offerings) for each of its dependencies. Therefore, assigning static priorities to jobs over-simplifies the problem. In addition, none of these techniques address the issue of provisioning for failures.

II. QoS-GRAF FRAMEWORK

We propose a framework for QoS based Grid Resource Allocation with Failure provisioning (QoS-GRAF), that advocates optimizing business metrics based on SLA driven pricing policies. Our work is most relevant to service oriented grid computing and failure provisioning. Within the framework, we focus on designing revenue maximization algorithms – where given a batch of jobs, their SLA specifications for multiple dependencies and the set of available resources in the grid, our objective is to (a) assign a *primary* resource for each job such that total revenue earned is maximized, and (b) assign a *backup* resource for each job such that the total revenue loss is minimized in the case of a failure. To the best of our knowledge, this is the first approach that marries the idea of SLA based business performance optimization in utility computing grids with the notion of failure provisioning.

Input to the framework is a set of grid jobs with their SLA descriptions. An SLA Manager maps their application level requirements to the corresponding resource level requirements. It also finds out the demand-revenue step functions for all dependencies of a job. Using job start/end times and other meta-information, a batch scheduler computes a temporal schedule of jobs and forms batches of jobs of similar duration of activity. For each batch of jobs, the QoS-GRAF controller invokes the primary and backup allocators to compute the set of resource allocations that will be reserved in advance. Once a failure happens the set of concerned jobs are switched to their backups. Failed jobs are either resumed or restarted depending upon the type of checkpointing support available.

A. QoS-GRAF Algorithms

In this section, we address the central problem of provisioning primary and backup resources for a batch of jobs. We consider a grid with J heterogeneous resources. The input to the system is a batch of K jobs, where each job has *atmost* D resource dependencies. Each resource can satisfy exactly one dependency. The demand-revenue function of a job, for each dependency, is a step-wise function with I distinct steps. Each step represents a different SLA level that can be used to service the job. Further, we assume that the available resource capacity is sufficient to guarantee the minimum requirement for each dependency of a job. Given this input we solve the two problems of (1) finding a primary allocation for all jobs that maximizes earned revenue (2) given

a set of primary allocations, finding the backup allocations that minimizes revenue loss. We design two linear relaxation based heuristics, Maximum Revenue Primary Allocation (MRPA) and Minimum Loss Backup Allocation (MLBA). We assume a single failure model for provisioning of backup resources.

$$\max_{\delta=1} \sum_{i=1}^D \sum_{j=1}^J \sum_{k=1}^K U_{\delta}^{i,j,k} R_{\delta}(i,k), \quad (1)$$

Objective function for MRPA is as shown in eqn.1, where $R_{\delta}(i,k)$ denotes the revenue earned by job k if demand for dependency δ is satisfied at level i . For MRPA we solve the revenue maximization problem subject to following constraints. The *capacity constraint* ensures that resource capacities are not exceeded. The *minimum, maximum requirement constraint* ensures that the minimum demand is met for each dependency and that the allocation does not exceed the maximum demand for any dependency. The *feasibility assignment constraint* makes sure that each job is assigned to exactly one resource and at exactly one priority level. Finally, the *integrality constraint* prevents jobs from being split across resources and priority levels. To solve MLBA the *integrality, feasible assignment* and *minimum requirement* constraints are the same as before. The *maximum requirement* is changed not to allow a backup to exceed its corresponding primary allocation. The *disjoint primary constraint* ensures that primary and backup resources of a job are disjoint. The *backup sharing constraint* ensures that backups of disjoint primaries can share the backup capacity. Finally, the *capacity constraint* ensures that the total shared backup capacities are within the available limits. Both are mixed integer programming problem, which can shown to be NP-hard by reduction from the multi-bin knapsack problem. We describe a heuristic approach to solve the problem. We relax the *integrality constraint* to obtain the fractional optimal solutions. This solution may be infeasible due to split jobs across SLA levels/resources. We use a sort-and-round technique based on the real values of the indicator variables. If a job can be fitted on the “indicated” resource at the “indicated” level, then assignment is done. If not, the job is initially allocated to a lower SLA level and marked for later handling. Once the initial assignment for all jobs is complete, an iterative routine tries to update allocations by boosting the SLA level of all marked jobs. This is done by either boosting its level at the already assigned resource, or assigning it to a different resource, depending upon the available resource capacity. Both heuristics employ similar techniques but the backup heuristic additionally utilizes sharing information.

III. PERFORMANCE EVALUATION

In this section, we present a set of representative experimental results to illustrate the effectiveness of our approach for (i) maximizing earned revenue, and (ii) minimizing revenue loss due to failure. We conducted a large number of experiments under a wide range of settings in a simulated grid environment. We show results for UNIT_BIG, which is found to be the most powerful heuristic. UNIT_BIG satisfies the base resource

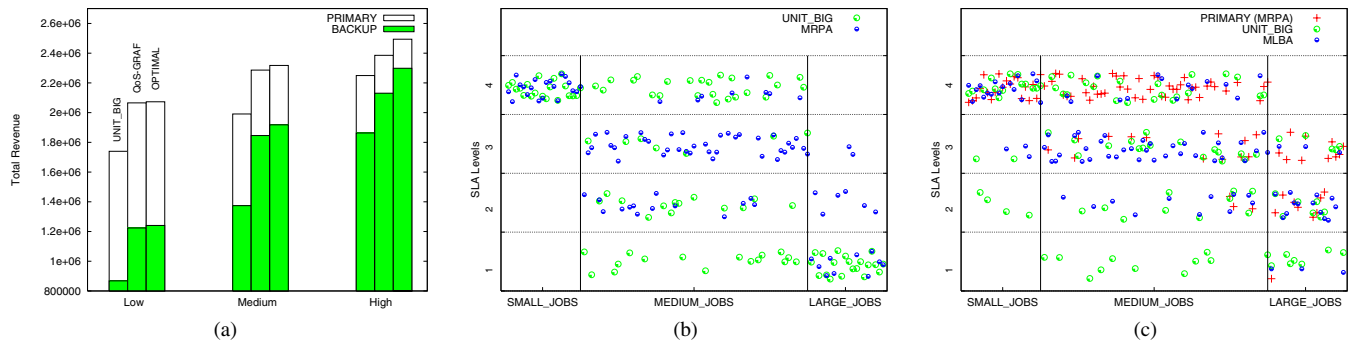


Fig. 1. (a) Primary revenue and revenue after one failure in different resource availability conditions (b) Type of job with its respective SLA level for primary allocation (c) Type of job with its respective SLA level for backup allocation

requirement of all jobs and remaining resources are distributed to the jobs starting with the highest priority job. Priority is determined by revenue earned per unit demand (revenue loss per unit demand in case of backup) of the jobs. Among multiple resources that satisfy the job requirement, one with the largest available capacity is chosen. The input to the algorithms is a batch of 100 jobs and the set of resources J that the jobs will be competing for. Each job has three resource dependencies. Each dependency is associated with a four step demand-revenue function. The total available capacity in the system is the sum of available primary and backup capacities, where the backup capacity is some fraction of the primary.

We classify jobs as small, medium and large based on their demand. For all runs the mix of jobs is 20%,60%,20% respectively. We define the low, medium and high resource availability conditions depending on the amount of primary resources available in the system. In Fig.1a, we compare the total revenue earned by each algorithm with the optimal (fractional) solution at various resource availability conditions. Backup bar is the revenue after the first failure. The difference between the backup and primary revenue gives the total revenue loss. We observe that all algorithms earn more revenue at medium and high availability conditions than in lower availability condition, where there is high contention for resources. MRPA outperforms UNIT_BIG by about 20% and always achieves performance within 5% of the optimal at the low, medium and high availability conditions. We show the near-optimality of MLBA as the revenue loss is always within 5% of the optimal in medium and high availability regions. In the low availability region, where resource contention is high, it can produce solutions that are within 2% of the optimal loss. In comparison, UNIT_BIG can be twice as bad as the optimal solution in medium and high availability regions.

For deeper understanding, we present the scatter plot in Fig.1b showing the SLA levels for the small, medium and large jobs for only one dependency. The results show UNIT_BIG earns most of its revenue by running more medium jobs at the highest priority. However, because of this, it is forced to push all large jobs at the lowest priority. In comparison, MRPA runs the medium and large jobs well distributed between SLA levels

2 and 3, with its maximum revenue coming from medium jobs at level 3. Because MRPA is able to adjust priorities of all classes of jobs, it does not penalize any particular class while earning steady revenue across all classes. Fig.1c shows that MLBA allocates backup resources at the same (or closer) levels as the corresponding primary, thus resulting in minimum potential revenue loss.

IV. CONCLUSION

In this paper, we propose QoS-GRAF, a framework for providing revenue maximization in a utility computing grid where jobs have multiple resource dependencies and differentiated QoS pricing. To solve the revenue maximization problem we propose Linear relaxation based algorithms, MRPA and MLBA, that achieve performance within 1–5% of the optimal solution and significantly outperform alternative approaches. Both show better revenue earnings across small, medium and large jobs, with efficient resource utilization. As a part of ongoing work, we are developing backup algorithms for multiple failures. We are trying to incorporate scheduling algorithms to produce maximum profitable schedule considering job deadlines.

REFERENCES

- [1] GRAAP-WG, “Grid resource allocation agreement protocol working group in the global grid forum.” <http://www.fz-juelich.de/zam/RD/coop/ggf/graap/>.
- [2] A. Keller, G. Kar, H. Ludwig, A. Dan, and J. Hellerstein, “Managing dynamic services: A contract based approach to a conceptual architecture,” in *Proceedings of NOMS’02*, 2002.
- [3] I. Foster, C. Kesselman, C. Lee, R. Lindell, K. Nahrstedt, and A. Roy, “A distributed resource management architecture that supports advance reservations and co-allocation,” in *Proceedings of IWQoS’99*, 1999. [Online]. Available: citeseer.ist.psu.edu/foster99distributed.html
- [4] L. Zhang and D. Ardagna, “Sla based profit optimization in autonomic computing systems,” in *Proceedings of ICSOC’04*, 2004.
- [5] M. J. Buco, R. N. Chang, L. Z. Luan, E. So, C. Tang, and C. Ward, “Pem: A framework enabling continual optimization of workflow process executions based upon business value metrics.” in *Proceedings of IEEE SCC’05*, 2005.
- [6] V. Naik, C. Liu, L. Yang, and J. Wagner, “On-line resource matching in a heterogeneous grid environment,” in *IEEE International Symposium on Cluster Computing and the Grid*, 2005.