Journal of Computer Science and Technology. 22(04) : 505-514, 2007.
The original publication is available at www.springerlink.com

A Model of Grid Service Capacity

Youcef Derbal
School of Information Technology Management
Ryerson University
350 Victoria Street, Toronto, ON, M5B 2K3, Canada
E-mail: yderbal@ryerson.ca

Abstract: Computational grids (CGs) are large scale networks of geographically distributed aggregates of resource clusters that may be contributed by distinct organizations for the provision of computing services such as model simulation, compute cycle and data mining. Traditionally, the decision-making strategies underlying the grid management mechanisms rely on the physical view of the grid resource model. This entails the need for complex multi-dimensional search strategies and a considerable level of resource state information exchange between the grid management domains. In this paper we argue that with the adoption of service oriented grid architectures, a logical service-oriented view of the resource model provides a more appropriate level of abstraction to express the grid capacity to handle incoming service requests. In this respect, we propose a quantification model of the aggregated service capacity of the hosting environment that is updated based on the monitored state of the various environmental resources required by the hosted services. A comparative experimental validation of the model shows its performance towards enabling an adequate exploitation of provisioned services.

Keywords: Computational Grids, Service Provision, Resource Model, Service Capacity.

1 Introduction

Computational Grids (CGs) are large scale networks of geographically distributed aggregates of resource clusters often contributed by distinct organizations for the provision of computing services such as scientific simulations, data mining and parameter sweep applications. The providers may enter into agreement to share the exploitation of these resources or organize them along an economy model to provide commercially viable business services. For this paradigm, the resource model underlying the various decision-making strategies is pivotal to the overall effectiveness of the grid resource exploitation [1]. Traditionally, the grid resource model is constructed as a dictionary of uniquely identified computing hosts with attributes such as CPU slots, RAM and Disk space, etc. Hence, it captures what may be labeled as the physical view of grid resources. Architectural variants of this physical view of grid resources have been utilized in the grid management strategies; including resource state dissemination [2-5]; scheduling [6-8]; and resource discovery [9-14]. The reliance of the grid management mechanisms on a large dictionary of resource attribute-value pairs entails the need for complex multi-dimensional search strategies and a considerable level of resource state information exchange between the distinct grid administrative domains. In addition, the queering of a large resource attribute-based model and the dissemination of its state across distinct administrative domains induces a higher processing and communication latency as well as an increase of network congestion. Furthermore, with the expanding application realm of grid computing to various scientific and business domains, it is becoming clear that grid resources are not limited to clusters of desktop machines, workstations and supercomputers but rather consist of any set of resources that enable the provision of computing services. These include specialized instruments such telescopes or digital signal processors, raw data from sensors, large repository of datasets both processed and raw, application services, and rich user interfaces to enable visualization and analysis [15]. The diversity and heterogeneity of grid resources provides a strong motivation for the use of the service abstraction to synthesize and integrate the management

Journal of Computer Science and Technology. 22(04) : 505-514, 2007.
The original publication is available at www.springerlink.com

mechanisms in the open service-oriented grid architecture. Given this context, the paper addresses the development of a model of the aggregate service capacity that quantifies the potential of a hosting environment to handle incoming service requests. The model is updated using the monitored state of the various resources required by the hosted services. The considered resources are not limited to computing devices and datasets but also include other infrastructure components such as application and database servers. For the later resources, issues such as the maximum number of open database connections, the maximum number of spawn threads, and the number of instances of a Virtual Machine that can be spawned by an application server are concrete indicators of the ability of the grid environment to handle requests directed to its hosted services. In this respect, the traditional notion of CPU slot may not constitute and adequate measure of a hosting environment' capacity to handle incoming service requests. Hence, we introduce, in conjunction with the service-oriented resource model, a unit of such aggregate service capacity that we call *servslot*. A *servslot* quantifies the level of resource needs necessary to handle a single service request. The share of resources attached to a *servslot* is adaptively computed based on the current service request load and its associated resource consumption. The introduced notion of service capacity results in two immediate benefits. First, a more effective resource exploitation is achieved because the dynamic quantification of a *servslot* enables the adaptation to varying resource needs across distinct service categories and across requests within the same service category. Second, simplifications of the management mechanisms such as scheduling are expected since the use of service-oriented quantification of hosting capacity is coherent with the view of the grid as a supply-demand system with service-centric exploitation processes [16,17].

The rest of the paper is organized as follows: Section 2 provides an overview of the grid architecture under consideration. The proposed model of service capacity is detailed in Section 3. Sections 4 and 5 describe the approaches used for the grid–wide integration of the resource model as well as its management at the cluster level. The Experimental validation of the model is detailed in Section 6 and the paper is concluded in Section 7 along with a discussion about future works.

2 Grid Architecture

The framework under consideration views the grid as a dynamic federation of resource clusters managed by distinct organizations for the provision of computing services. Each cluster constitutes a distinct management domain which includes a set of agents that manage the provision of the hosted services (see Fig. 1). One agent, called the Principal, is designated to coordinate inter-cluster operations such as resource state dissemination, and delegation of service requests' handling to peer clusters. The resulting grid has the properties of a power law network such as robustness against random failures of the agents [18,19]. The inter-cluster communication is mediated by the Principals, as a result overlay networks spanning multiple clusters is avoided, and the problem of excessive traffic illustrated for the case of Gnutella networks may be prevented [20].

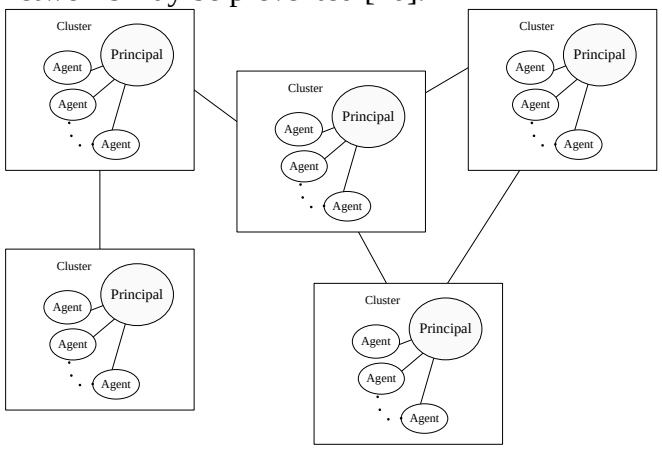

Fig. 1. The grid as a dynamic federation of resource clusters

The Grid nodes are assumed to have the ability to delegate the handling of service requests to other clusters in function of some inter-cluster Service Level Agreements (SLAs) [21]. In addition to the assumed federation-based organization of the grid, the considered architectural framework utilizes the notion of neighborhood and neighbors. In this respect, two nodes are said to be neighbors if they

Journal of Computer Science and Technology. 22(04) : 505-514, 2007.
The original publication is available at www.springerlink.com

maintain a regular communication link for the exchange of information such as resource state. The set of neighbors to a given node defines its neighborhood. Node membership in the grid is assumed to be dynamic. It is established through the exchange of identity information among neighboring nodes, which should ideally be geographically proximate. The identity information includes the node IP address, assigned communication port numbers, as well as any other credential or configuration parameters necessary for the establishment of communication links. The clusters that make up the gird may join or leave the federation at any time, with an effect limited to the configuration of neighboring clusters. However, since service request handling may be delegated beyond a neighborhood, extra management mechanisms are needed to minimize loss of work that may already be in progress within a departing cluster. In case where inter-cluster agreements are dynamically negotiated or established out of bound, it is assumed that they would include elements which deal with changes to a cluster membership in the grid and the mentioned issue of loss of work.

3 Model of Service Capacity

Consider a User Service Request (USR) submitted to a grid cluster. Let us assume that such request requires for its handing the availability of a single grid service. Such availability would necessarily go beyond the assertion that the required grid service is indeed deployed. In particular, the service hosting environment has to possess sufficient resource availability for the instantiation of the grid service in question, the subsequent invocation of its operations, and the maintenance of its state. The required resources may include CPU slots, RAM, other service components, disk space, swap space, memory cache as well as any required licenses of application software that the service instance may need for its successful operation. Let $R=\{r_0, r_1, ..., r_{n-1}\}$ be the set of cluster resources required for the provision of a set $S=\{s_0, s_1, ..., s_{m-1}\}$ of deployed services. The competing resource needs of these services may be illustrated in Fig. 2 indicating their inevitable coupling. While it may be possible to allocate, on a service by service basis, the needed levels of resources, the overall resource exploitation would be less than effective. For example, let us assume that the services in S are exposed using Web Services deployed on an application server and rely for their operation on the access to a database server. In addition to the nominal resource needs of the database and application servers such as RAM and disk, the handling of inbound service requests will have their own extra resource needs. These are expected to fluctuate from one request to the next depending on the nature of the business function realized by the service in question.

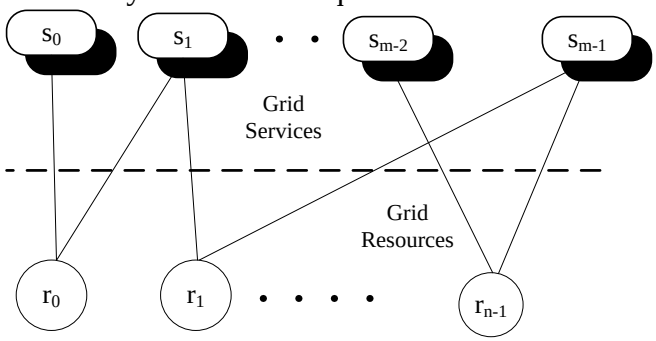

Fig. 2. Deployed services rely on common resources

Furthermore, the level of resource usage varies during the lifecycle of a single service request handling. Consequently, there is a need for a model of service capacity that captures the time-varying aggregate behavior of the resources within the hosting environment. In practice, it is plausible to assume that the deployed grid services belong to m distinct categories, each essentially reliant on a common set of resources such that $S=\bigcup_{k=0}^{m-1} S^{(k)}$, and $\bigcap_{k=0}^{m-1} S^{(k)}=\varnothing$. Let the time-varying service capacity of the hosting environment for the category k of deployed services be defined as follows:

$$C_k(t)=\sum_{i=0}^{n-1}\alpha_{ik}\frac{l_i(t)}{l_i^{(\max)}} \qquad (1)$$

Where $\sum_{i=0}^{n-1}\alpha_{ik}=1$, $l_i(t)\geq 0$ is the availability level of resource $r_i\in R$ at the discrete time $t\in\mathbb{N}$. $l_i^{(\max)}>0$ is its maximum level of availability, and $\mathbb{N}$ is the set

Journal of Computer Science and Technology. 22(04) : 505-514, 2007.
The original publication is available at www.springerlink.com

of natural numbers. The weighting factor $\alpha_{ik} \geq 0$ defines the relative importance of the various resources in shaping the aggregate service capacity of the hosting environment. Resources that are more critical to the operation of the service category will have a higher weight in the definition of the service capacity. While the above model of service capacity reflects the availability levels of the physical resource, there are other considerations that would require its refinement. In particular, let us consider the interaction between the decision-making mechanisms (such as task scheduling) and the estimation of the resource availability. This closed loop interaction is embodied by the time-varying dynamic feedback system of Fig. 3, where f_s and f_o are the scheduling and monitoring frequencies respectively. In compliance with Shannon's Sampling Theorem [22], f_o would have to be at least twice the scheduling frequency f_s in order to achieve an accurate observation of the dynamics of resource availability resulting from the allocations of the scheduling decisions.

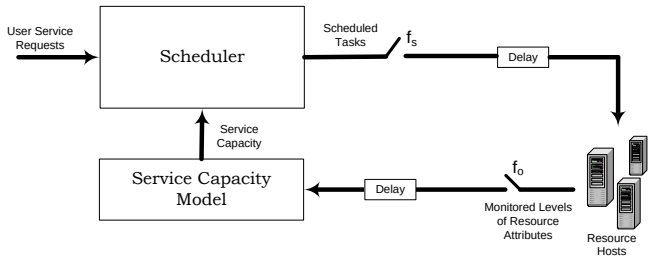

Fig. 3. Grid decision-making processes operate in the context of a closed loop feedback system.

In keeping with the practice in digital control systems, we may choose the above frequencies such that $f_o \geq 2.5 f_s$. The validity of this choice relies on the assumption that the changes in resource availability are the direct result of the scheduling decisions, which is always true at the cluster level of a grid system. Even with the satisfaction of the above condition, with an added margin to account for network delays, the simplicity of the capacity model given by (1) has an undesirable drawback. Indeed, the model may some times indicate a non-zero capacity even when the hosting environment is not able to instantiate the service or simply fail to do it within an expected delay period as specified by some required QoS (Quality of Service). Assuming the compliance with the Sampling Theorem, this anomaly can be corrected through a refinement that takes into consideration an estimate of the maximum service load that a hosting environment may handle for a given service category. In this respect, let $I_k^{(\max)}$ be the estimated maximum number of service requests that can ever be handled concurrently for a benchmark service of the category. The capacity model given in (1) is then refined as follows:

$$C_k(t) = \left(\sum_{i=0}^{n-1} \alpha_{ik} \frac{l_i(t)}{l_i^{(\max)}} \right) . \gamma_k(t) \qquad (2)$$

Where $\gamma_k(t) = 1 - \frac{I_k(t)}{I_k^{(\max)}}$, $\sum_{i=0}^{n-1} \alpha_{ik} = 1$, and $I_k(t)$ is the number of concurrently handled service requests of category k. The factor $\gamma_k(t)$, that we call the instantaneous service utilization factor, ensures that as more service instances are concurrently running, the resulting resource utilization is directly and proportionally translated into a diminished availability of residual service capacity. Since all service categories rely in part on a common subset of physical resources such as CPU and RAM, the above version of the capacity model has to be normalized so as to ensure that the utilization of common resources by any one of the service categories is reflected on the residual service capacity for all service categories. Hence, we further constrain the model as follows:

$$C_k(t) = \lambda_k \left(\sum_{i=0}^{n-1} \alpha_{ik} . \psi_i(t) \right) . \gamma_k(t) \qquad (3)$$

λ_k is an inter-category normalization factor, and

$$\psi_i(t) = \frac{l_i(t)}{l_i^{(\max)}}, \gamma_k(t) = 1 - \frac{I_k(t)}{I_k^{(\max)}}, \sum_{i=0}^{n-1} \alpha_{ik} = 1, \sum_{k=0}^{m-1} \lambda_k = 1.$$

λ_k can be set to $\frac{1}{m}$ for all categories implementing therefore a fair share policy of common resources. Other policies of sharing can also be implemented based on known demand patterns of hosted service categories. For example, the service categories with

Journal of Computer Science and Technology. 22(04) : 505-514, 2007.
The original publication is available at www.springerlink.com

higher demands would have higher values of λ_k compared to those associated with patterns of lower demand levels. For ease of implementation $C_k(t)$ is scaled up with a factor of 100 and rounded to the closest integer. We can then interpret the introduced capacity measure as the percentage or share of the total environment's service capacity that would be available under zero load (when no service requests is being handled). Consequently, a *servslot* is defined as the portion of $C_k(t)$ necessary for the handling of a single service request, and is set initially to $\frac{C_k(t)}{I_k^{(\max)}}$. However, this value is dynamically updated thereafter using the actual load (instantaneous number of concurrently handled service requests) instead of $I_k^{(\max)}$.

In order to support the grid wide resource management mechanisms such as scheduling and service discovery, the service capacity is maintained in a cluster-bound registry whose content is selectively disseminated to peer clusters. The resulting network of service registries make up a non-disjoint distributed repository updated through regular exchange of service capacity information among neighbors (see Fig. 4). It has been shown that such non-uniform, selective dissemination strategy of resource state information is scalable [23]. In addition, given the low data density being exchanged, both the network bandwidth usage and the registries' storage requirements are expected to be relatively low.

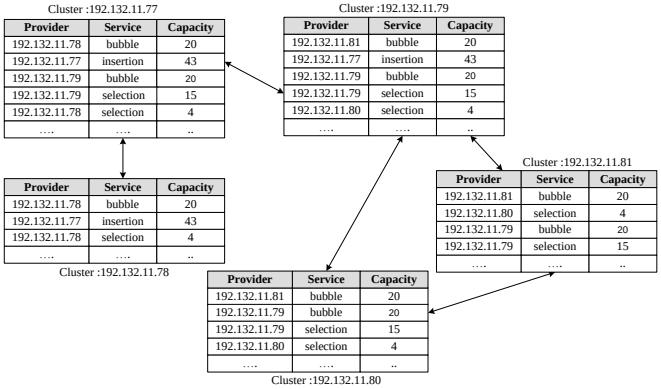

Fig. 4 A distributed repository of service capacity

The proposed capacity model is concerned with the quantification of a node capacity to handle incoming service requests. However, the model may also accommodate composed services that span multiple clusters or nodes. In this respect, the capacity published by the node that hosts the service endpoint of a composed service may be expressed as follows:

$$C_k(t) = \min_{j=0,..,n_s-1} C_k^{(j)}(t) \qquad (4)$$

Where n_S is the number of component services, and $C_k^{(j)}(t),\ j=0,...,n_S$ are their respective capacities. The distributed network of service registries provides the capacity information associated with the component services and enable hence the implementation of the above model. The considered architectural framework assumes the composed services to be reliant on component services hosted by neighboring clusters. However, such constraint may be removed by adding a discovery process to query the capacities of component services hosted by non-neighboring clusters.

4 Dynamic Estimation of Service Capacity

Since the capacity unit (*servslot*) is tied to a service or a class of services as assessed by the provider cluster, an auxiliary mechanism is necessary to enable peer clusters to interpret the disseminated capacity information with respect to the needs of their service requests. In this respect, let the residual (unused) capacity per running service instance be defined at time t as follows:

$$\delta_k(t) = \frac{C_k(t)}{N_k(t)} \qquad (5)$$

Where $N_k(t)$ is the number of running service instances of category k at time t, $C_k(t)$ is the total available capacity as disseminated by the provider. A small value of δ_k indicates a low relative margin of available capacity. Such margin is needed to address the variable resource needs during the lifecycles of requests' handling. The provider may assert that a new service request may not be handled successfully unless δ_k is greater than a threshold value $\Delta_k > 0$. Δ_k may be estimated through monitoring of the trends of resource consumption by the hosted services. Provided that C_k, Δ_k, and

Journal of Computer Science and Technology. 22(04) : 505-514, 2007.
The original publication is available at www.springerlink.com

$N_k(t)$ are disseminated, potential service consumers may estimate the ability of the provider to handle their requests prior to any request handling delegation. In particular, the future residual capacity per service request may be defined for a supplementary load of m service request as follows:

$$\delta_k^{(m)}(t) = \frac{C_k(t)}{N_k(t)+m} \qquad (6)$$

A satisfaction of the relationship $\delta_k^{(m)} \geq \Delta_k$ would indicate a high likelihood that the hosting environment will have a sufficient capacity to concurrently handle m extra service requests in addition to its current load. This estimation scheme has an inherent advantage in that it doesn't require an expensive and non-scalable resource-attribute-based search of a potentially large resource directory such as the Globus MDS2 [24]. The selective non-uniform dissemination of available service capacity among neighboring peers improves the scalability of those decision making mechanisms that rely on the capacity information such as scheduling, and service discovery. However, the uncertainty on the capacity information and the frequency of dissemination would necessarily affect the performance of the resource exploitation strategies. In other works, the uncertainty on the resource state has been addressed through a supplementary confidence model [25]. The model provides a potential consumer with a measure of confidence in the ability of the service provider to handle future service requests. The integration of the proposed model of service capacity, the dissemination strategy and the handling of the associated uncertainty is an important issue that requires further treatments in future works.

5 Service Management

The implementation of the proposed model requires a service provision framework that incorporates the monitoring and management of resource availability. As deployed grid services claim and use the shared resources of their home cluster, their available capacity has to be updated accordingly. In other words, the actual availability of the physical resources (CPU, RAM, Swap, Disk space, database connections, threads, etc.) has to be regularly monitored to provide a timely state feedback to the model of service capacity. Furthermore, given the presumably random distribution of the capacity needs of the service requests, Δ_k needs to be updated at regular time intervals so as to accurately reflect the critical point at which the hosting environment would become unable to accept additional load for a specific service. For this reported work, a moving average over a discrete window of time M is used to estimate Δ_k, that is:

$$\Delta_k = \frac{1}{M+1}\sum_{i=n-M}^{n}\frac{C_k(n)}{N_k(n)} \qquad (7)$$

The use of the above model provides an averaged historical profile of capacity consumption per service request. Hence, it is deemed adequate to estimate Δ_k which is essentially the minimum capacity necessary for a future service request to be successfully handled.

The proposed model of service capacity has been integrated with Midland, a grid middleware developed by the author (See Fig. 5). A cluster bound relational database is used to implement the service registry which stores the service capacity information. The resource state information is relayed by the host resource managers to the Service Capacity Manager, which is responsible for the maintenance of the capacity information as well as its dissemination to neighboring peers.

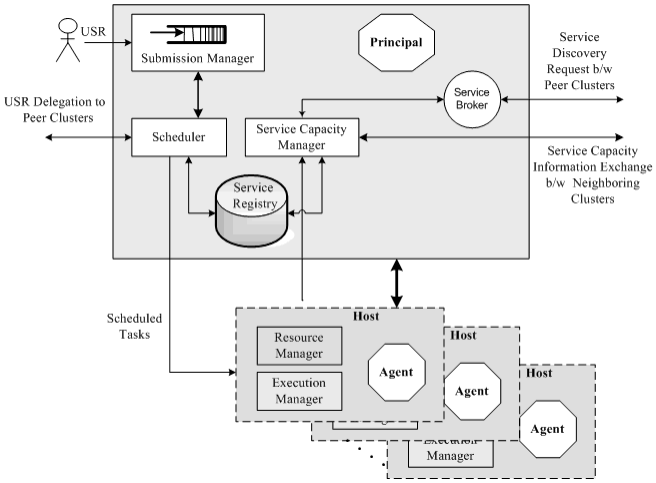

Fig. 5. Overview of Midland architecture

Journal of Computer Science and Technology. 22(04) : 505-514, 2007.
The original publication is available at www.springerlink.com

6 Related Works

To the best of our knowledge, the proposed approach of service capacity modeling has not been explored with the exception of one related research work reported in [17]. In this work two metrics called *server share* and *service share* are introduced to quantify the capacity of a server environment to handle service requests. The *server share* is defined for a given class of services as the maximum load the server environment is able to sustain. The *service share* is the percentage of the *server share* required by a given service. The maximum server load is established in relation to a benchmark application for the class of services under consideration. This approach of service capacity quantification has been applied to services provided by a data centre managed within the framework of a single administrative domain of a utility grid. The capacity quantification model developed in [17] provides a significant contribution towards the development of a service oriented exploitation of grid resources. However, the reliance on a benchmark application for the definition of the capacity unit may limit the extension of the approach to grids with distinct management domains unless a significant standardization effort is undertaken regarding benchmark service loads. Benchmark applications would have to be selected as part of a community standard and thereafter continuously updated to account for new classes of services. This may not be practically feasible given the strong linkage between resource usage profiles of deployed services and the nature and configuration of the software and hardware infrastructure of the hosting environment which may vary considerably from one provider to the next. This is in contrast to the proposed approach where potential service consumer nodes rely on a dynamic estimation of the *servslot* unit using the disseminated capacity information. As a result, any need for prior grid-wide normalization or benchmarking of service resource needs is avoided. This maintains the independence of the exploitation mechanisms applied within the distinct administrative domains of the grid, and enables the grid-wide implementation of the proposed model of service capacity.

7 Experimental Results

A grid test-bed is setup using three Midland managed clusters connected through a 100 mps local area network. Each cluster includes a collection of desktop machines (2.7 GHZ CPU and 0.5 GB of RAM), and an IBM xSeries server which hosts the cluster's principal. A set of sorting services based on the bubble, insertion and selection algorithms are deployed on the grid test-bed. The prototype implementation of the proposed capacity model considers a resource set that includes the CPU, RAM, virtual memory, running processes, and running threads. Although the ability of the system to spawn new threads and processes depends on the first three types of resources, the inclusion of the later elements enable a more efficient control of the resources' usage by the deployed services. For the reported experiments, the inter-arrival time of service requests follows a Poisson distribution with a randomly selected rate. The requested services are randomly chosen among the three deployed sorting services. The size of the data being sorted is generated using a Gaussian process so as to simulate a random distribution of the resource needs of the service requests.

The objectives of the experimental work is threefold; namely: (1) to illustrate the ability of the model to provide an adequate quantification of the hosting environment's capacity to handle inbound service requests; (2) to compare the proposed model, labeled as the Dynamic Service Capacity (DSC) model, against the traditional approach of static slot capacity allocation referred to here as the Static Slot Capacity (SSC); and (3) to illustrate the validity of the proposed strategy of capacity estimation by potential service consumers.

For the first set of experiments, Figs 6-9 show a direct correlation between the load and the residual capacity, whereby a load increase resulted in the decrease of the capacity and vise-versa. This correlation is asserted for a significant spread of the service request makespan distribution as illustrated in Fig. 10.

Journal of Computer Science and Technology. 22(04) : 505-514, 2007.
The original publication is available at www.springerlink.com

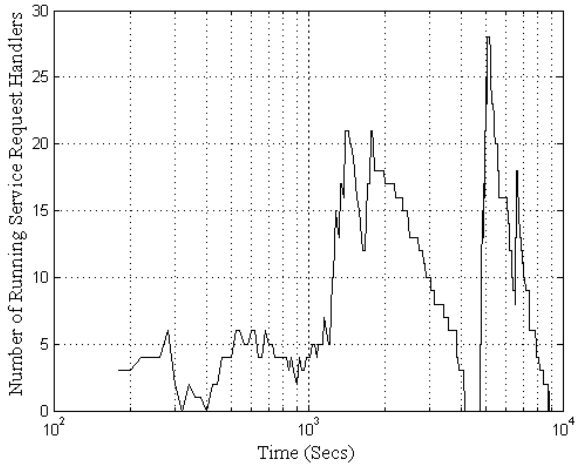

Fig. 6. Service load

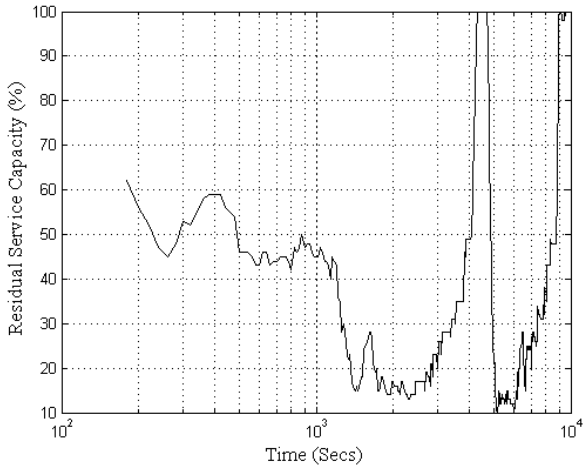

Fig. 7. Residual capacity as a percentage of the zero load capacity.

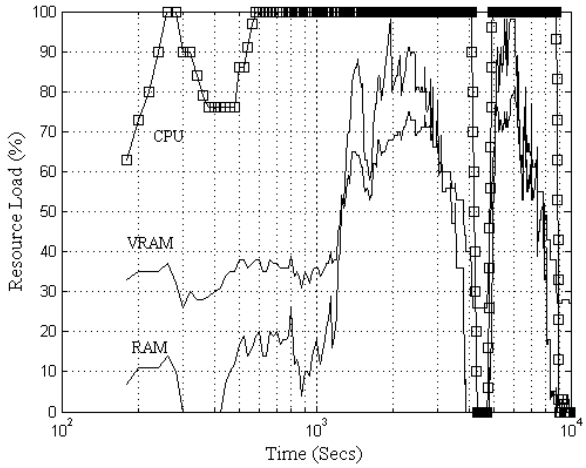

Fig. 8. Resource load as a percentage of the zero load levels

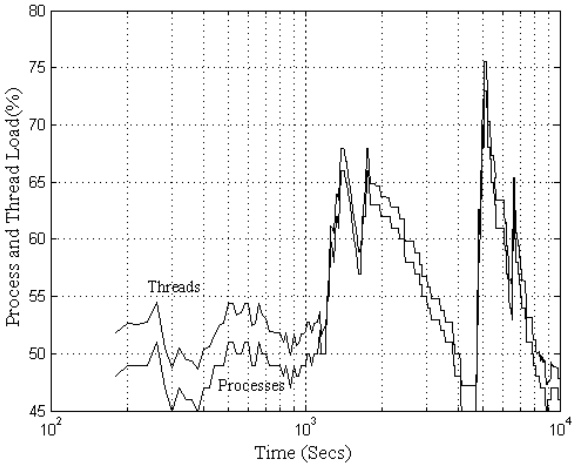

Fig. 9. Process and Thread loads as a percentage of the maximum allowed numbers of processes and threads.

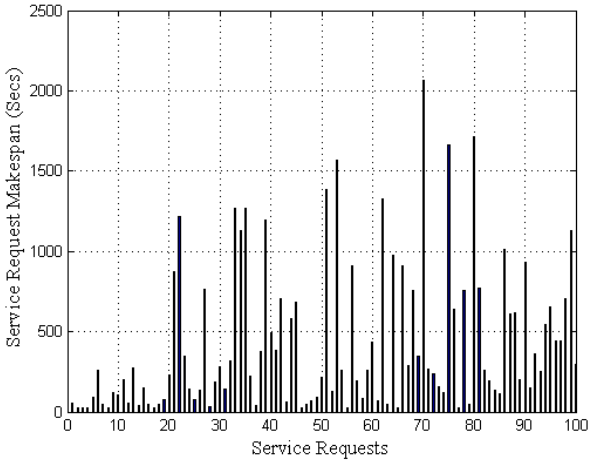

Fig. 10. Service request makespan distribution

This observed load-capacity correlation in addition to the absence of any pattern of resource starvation may suggest that the model reflects the real capacity of the hosting environment (see Figs 8-9). In order to further assert the validity of the above argument, the following conjecture is formulated: if the proposed model is adequate in its representation of the environment's service capacity, then it would be possible for a scheduler in closed loop with the proposed model to achieve a steady state exploitation dynamics where the load is maintained at its maximum level with a low residual service capacity to spare. Another experiment that was run to test this conjecture show that (Figs 11-13): (1) indeed the resource exploitation converged, in a stable fashion, to a steady state; (2) the steady state corresponds to a maximum exploitation where the

Journal of Computer Science and Technology. 22(04) : 505-514, 2007.
The original publication is available at www.springerlink.com

number of concurrently handled service requests was maintained with little or no residual capacity to spare. It needs to be noted that while the service capacity available at zero load is normalized to 100, it does not mean that the number of service requests that can be handled concurrently is 100. This is because the *servslot*, as defined in section 3, is equal to the average share of the total initial capacity that is consumed per running service instance. As a result its value changes in time in response to the nature of the load. For example, a load of 10 requests and an observed zero residual capacity at a given time instant means that, for that specific instant a servslot is worth 10% of the zero load capacity.

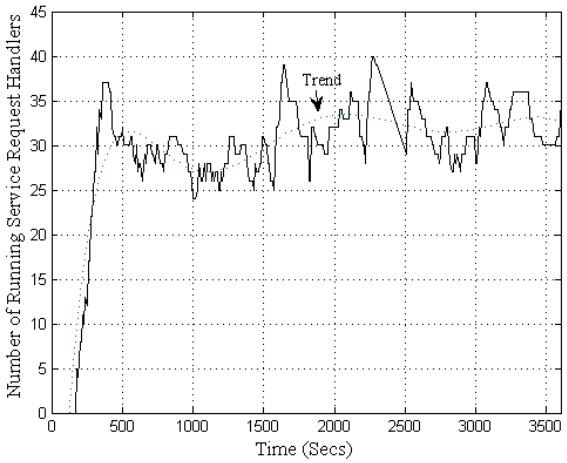

Fig. 11. Service load

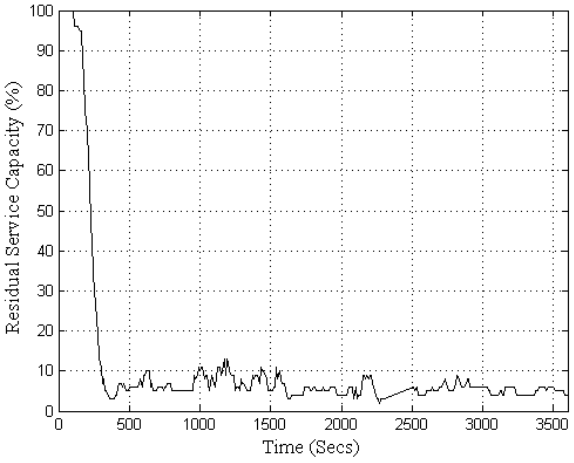

Fig. 12. Residual Service Capacity

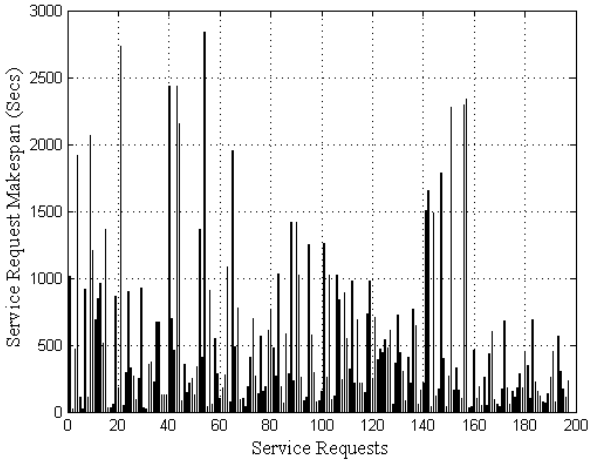

Fig. 13. Service request makespan distribution

In a second set of experiments a comparison is conducted against the SSC model because of its equally advantageous simplicity. The size of the sorted data is chosen to be Gaussian distributed to create a realistic distribution of the capacity needs of the service requests. Figures 14 and 15 illustrate the performance of the SSC and DSC models for a standard deviation $\sigma = 30$ and different maximum slot limits $a_{\max}$ for the SSC model. The observed fluctuating low residual capacity for the SSC model increases the likelihood of sudden resource depletion and potential runtime fault occurrences. For some experimental runs, the virtual memory was so depleted that the Windows Services used to collect the resource and task states crashed.

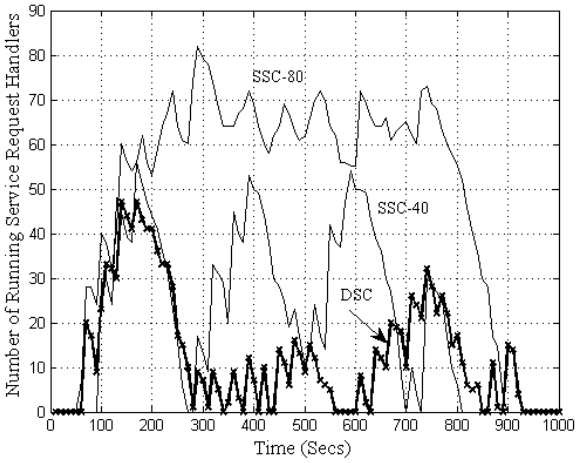

Fig. 14. Service load for $\sigma = 30$ and $a_{\max}$ =40, 80

Journal of Computer Science and Technology. 22(04) : 505-514, 2007.
The original publication is available at www.springerlink.com

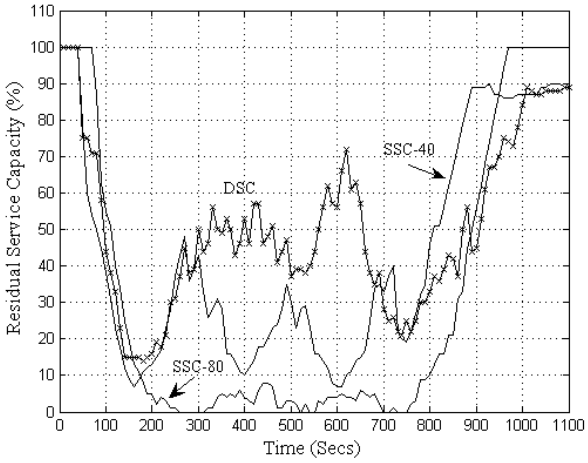

Fig. 15. Residual capacity for σ =30 and a_{max} =40, 80

In contrast, the DSC model exhibited a stable performance that was expected given the resource state feedback. More experimental runs were conducted for a wider distribution of the capacity needs of the submitted service requests. The dynamics of resource consumption shown in Figs 16-19 is visibly more oscillatory for the SSC model and did cause a critical depletion of resources and a consequent crash of the monitoring infrastructure which had to be restarted (Figs 18-19).

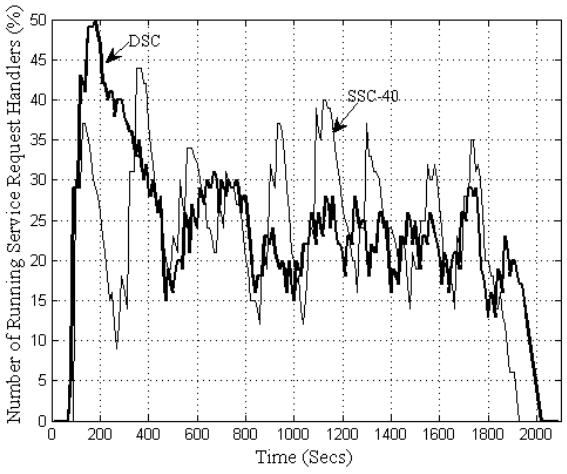

Fig. 16. Service load σ =60 and a_{max} =40

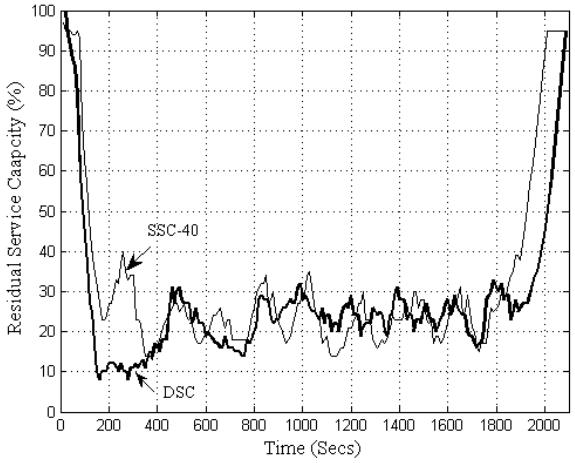

Fig. 17. Residual capacity for σ =60 and a_{max} = 40

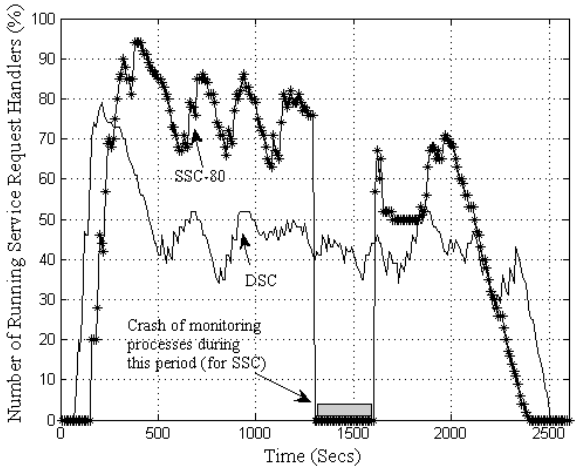

Fig. 18. Service load for σ =60 and a_{max} = 80

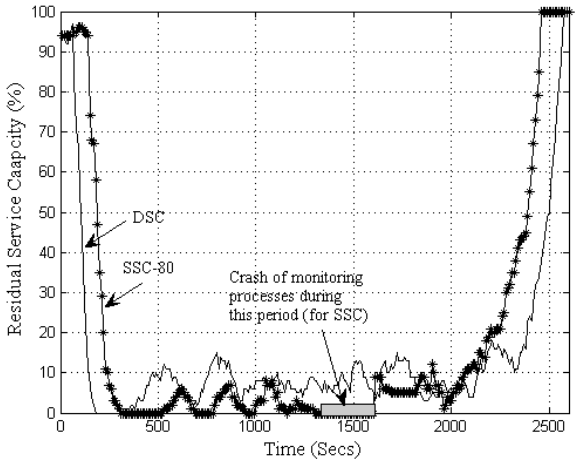

Fig. 19. Residual capacity for σ =60 and a_{max} = 80

The last set of experiments involved three clusters configured as immediate neighbors. Submitted requests at each cluster are automatically forwarded

Journal of Computer Science and Technology. 22(04) : 505-514, 2007.
The original publication is available at www.springerlink.com

to neighboring clusters without any optimized scheduling so as to analyze the dynamic estimation strategy given in section 4. Fig. 20 depicts the ratio of service denials $\eta_D = \frac{n_D}{n_T}$ versus $\delta_k^{(m)}$. n_T is the total number of requests submitted across all three clusters, and n_D is the number of service requests that were denied handling because of insufficient availability of service capacity. The key result of Fig. 20 is the observed inverse proportionality between η_D and $\delta_k^{(m)}$ which suggests that the use of $\delta_k^{(m)}$ is an adequate predictor of the available capacity of peer clusters. However, given the latencies associated with the dissemination of capacity information between clusters, it may appropriate to consider in future works the use of past values of $\delta_k^{(m)}$ as part of a more generalized prediction model.

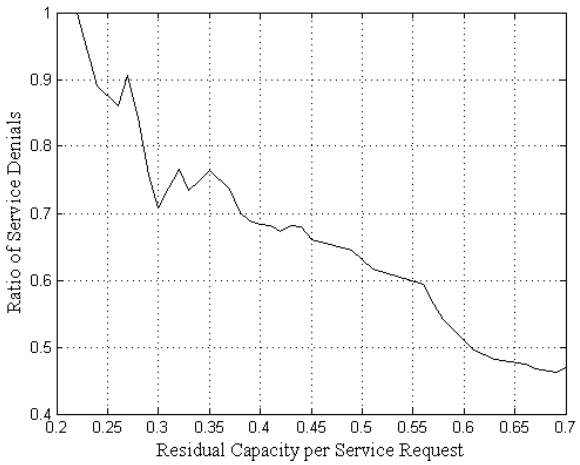

Fig. 20. Ratio of service denials as a function of $\delta_k^{(m)}$

7 Conclusion and Future Works

The paper put forth the foundations for a model of grid service capacity to overcome some of the complexities and challenges associated with the use of traditional attribute-value based resource models. The new approach includes the quantification of the aggregated service capacity of a hosting environment using a dynamic measure called *servslot*. The experimental results show the extent to which the proposed model provides an adequate measure of capacity to support the grid management mechanisms such as scheduling. Compared to the traditional model of static slot capacity, the proposed model is shown to enable more robust resource exploitation in the face of varying resource state. However, there remain numerous issues that may be addressed in future works. In particular, the use of a non-linear model of service capacity using Neural Network, trained through profiling of resource consumption, may offer a generalized solution to the modeling of grid service capacity. Second, the estimation of the ability of peer clusters to handle forwarded services requests may benefit from a strategy based on a time-series model of residual capacity. Finally, more investigations are needed to study the effect of latency on the leverage of the disseminated capacity information provided by the proposed model.

References

[1] Foster, I., and Kesselman, C. The grid: blueprint for a new computing infrastructure, Elsevier Science, 2004.

[2] Iyengar, V., Tilak, S., Lewis, M. J., and Abu-Ghazaleh, N. B. Non-Uniform Information Dissemination for Dynamic Grid Resource Discovery. In Proc. 3rd IEEE International Symposium on Network Computing and Applications (NCA04), Cambridge, MA, USA, 2004, pp. 97-106.

[3] Krauter, K., Buyya, R., and Maheswaran, M. A taxonomy and survey of grid resource management systems for distributed computing. *Software - Practice and Experience*, 2002, 32(2)**:** 135-164.

[4] Wu, X.-C., Li, H., and Ju, J.-B. A prototype of dynamically disseminating and discovering resource information for resource managements in computational grid. In Proc. 3rd International Conference on Machine Learning and Cybernetics, Shanghai. China, 2004, pp. 2893-2898.

[5] Maheswaran, M. Data dissemination approaches for performance discovery in grid computing systems. In Proc. 15th International Parallel and Distributed Processing Symposium (IPDPS '01), Nice, France, 2001, pp. 910 - 923.

Journal of Computer Science and Technology. 22(04) : 505-514, 2007.
The original publication is available at www.springerlink.com

[6] Casavant, T. L., and Kuhl, J. G. A Taxonomy of Scheduling in General-Purpose Distributed Computing Systems. *IEEE Transactions on Software Engineering*, 1988, 14(2)**:** 141-155.
[7] He, X., Sun, X., and Von Laszewski, G. QoS guided Min-Min heuristic for grid task scheduling. *Journal of Computer Science and Technology*, 2003, 18(4)**:** 442-451.
[8] Spooner, D. P., Jarvis, S. A., Cao, J., Saini, S., and Nudd, G. R. Local grid scheduling techniques using performance prediction. *IEE Proceedings: Computers and Digital Techniques*, 2003, 150(2)**:** 87-96.
[9] Bukhari, U., and Abbas, F. A comparative study of naming, resolution and discovery schemes for networked environments. In Proc. 2nd Annual Conference on Communication Networks and Services Research, Suzhou, China, 2004, pp. 265 - 272.
[10] Dimakopoulos, V. V., and Pitoura, E. A peer-to-peer approach to resource discovery in multi-agent systems. In *Lecture Notes in Artificial Intelligence 2782*, Springer-Verlag, 2003, pp. 62-77.
[11] Huang, Z., Gu, L., Du, B., and He, C. Grid resource specification language based on XML and its usage in resource registry meta-service. In Proc. 2004 IEEE International Conference on Services Computing, Shanghai, China, 2004, pp. 467 - 470.
[12] Zhu, Y., and Zhang, J.-L. Distributed storage based on intelligent agent. In Proc. 3rd International Conference on Machine Learning and Cybernetics, Shanghai, China, 2004, pp. 297-301.
[13] Bradley, A., Curran, K., and Parr, G. Discovering resources in computational GRID environments. *Journal of Supercomputing*, 2006, 35(1)**:** 27-49.
[14] Huang, Y., and Bhatti, S. N. Decentralized resilient grid resource management overlay networks. In Proc. 2004 IEEE International Conference on Services Computing, SCC 2004, 2004, pp. 372 - 379.
[15] Fox, G. Integrating computing and information on grids. *Computing in Science and Engineering*, 2003, 5(4)**:** 94- 96.
[16] Czajkowski, K., Foster, I., and Kesselman, C. Agreement-based resource management. *Proceedings of the IEEE*, 2005, 93(3)**:** 631-643.
[17] Graupner, S., Kotov, V., Andrzejak, A., and Trinks, H. Service-centric globally distributed computing. *IEEE Internet Computing*, 2003, 7(4)**:** 36 - 43.
[18] Faloutsos, M., Faloutsos, P., and Faloutsos, C. On power-law relationships of the Internet topology. *Computer Communication Review*, 1999, 29(4)**:** 251-262.
[19] Barabási, A., and Albert, R. Emergence of Scaling in Random Networks. *Science*, 1999, 286(5489)**:** 509-512.
[20] Ripeanu, M., Iamnitchi, A., and Foster, I. Mapping the Gnutella network. *IEEE Internet Computing*, 2002, 6(1)**:** 50-57.
[21] Al-Ali, R., Hafid, A., Rana, O., and Walker, D. An approach for quality of service adaptation in service-oriented Grids. *Concurrency Computation Practice and Experience*, 2004, 16(5)**:** 401-412.
[22] Shannon, C. E. A mathematical theory of communication. *Bell Systems Technical Journal*, 1948, 27(1)**:** 623-656.
[23] Derbal, Y. Entropic grid scheduling. *Journal of Grid Computing*, 2006, 4(4)**:** 373-394.
[24] Zhang, X., and Schopf, J. M. Performance analysis of the Globus toolkit monitoring and discovery service, MDS2. In Proc. IEEE International Performance, Computing and Communications Conference, Chicago, IL USA, 2004, pp. 843-849.
[25] Derbal, Y. A probabilistic scheduling heuristic for computational grids. *Multiagent and Grid Systems*, 2006, 2(1)**:** 45-59.