Service Oriented Computing and Applications - SOCA (2009) 3:109–125
The original publication is available at www.springerlink.com

Midland: A Service-Oriented Cluster Computing Infrastructure

Youcef Derbal
Ted Rogers School of Information Technology Management,
Ryerson University,
Toronto, Canada
Phone: (416) 979-5000x 7918
Email: yderbal@ryerson.ca

Abstract: Midland is a service-oriented software infrastructure that enables the clustering of arbitrarily large collections of computing resources. The resulting clusters may be integrated to form an open, dynamically configurable computational grid system where each cluster defines a self-reliant and independent management domain. Web Services make up the primary integration mechanism both at the cluster and grid levels respectively. This is complemented by a light XML based messaging protocol exclusively used for cluster bound interactions. The paper describes Midland's architecture, and the service-oriented approach taken to develop the associated resource management mechanisms. It also includes an exposition of the model of service capacity which is one of the enablers of the service-centric strategy underlying the cluster management mechanisms. The operational performance of Midland is illustrated experimentally in the context of a Grid test-bed comprising three clusters. The experimental results highlight the performance of the model of service capacity as well as some aspects of Midland operational performance.

Keywords*:* Computational Grids, Service-Oriented Architecture, Service Capacity, Resource Modeling.

Service Oriented Computing and Applications - SOCA (2009) 3:109–125
The original publication is available at www.springerlink.com

1. INTRODUCTION

GRID computing systems are large scale distributed networks of aggregated computing resources integrated through a decentralized management framework to harness a collective computing capability with a target performance and an expandable service capacity and diversity. In general, grid systems are built around a collection of geographically distributed resource clusters, were each cluster constitute a self reliant, independent management domain. This two-tier architecture decouples the grid layer from the cluster layer enabling hence the grid nodes to apply their chosen policies of resource exploitation and service provision.

Today, the list of commercial and research products that provide the software infrastructure for the management of diverse computing and data resources is impressively long [1-7]. Among these systems, the Globus Toolkit [8] and UNICORE [5] provide the infrastructure necessary for the integration of large scale, geographically distributed computational grids. Both systems enable a single sign-on access to the grid system based on the X.509 security standard. While UNICORE is designed to offer an integrated and uniform access to distributed resources, the Globus toolkit allows the development of grid application using Globus services [8, 9] . The active standardization effort led by the Open Grid Forum (OGF) and other forums such as the Organization for the Advancement of Structured Information Standards (OASIS), the World Wide Web Consortium (W3C), the Internet Engineering Task Force (IETF), and the Distributed Management Task Force (DMTF) are providing a considerable impetus to grid computing and its application in scientific research and business [10-16]. However, there are persisting challenges shared by these initiatives which include the key areas of security, performance stability, fault-tolerance, reliability, management and movement of data, assurance of Quality of Service (QoS), and accounting. Furthermore, although the grid computing community has endorsed the Open Grid Service Architecture (OGSA) [17], there is a discord between the current state of focus of many local grid resource management systems and the grid level mechanisms that are expected to emerge as a result of the implementation of OGSA and other standards such as WS-Agreements [18]. Indeed, while a non-negligible body of research has been dedicated to the management mechanisms such as task and workload management, co-scheduling, advanced reservation, and resource brokerage, the focus of many local grid resource management systems, commercial and research oriented, tended to be devoted to task performance management with little or no attention to the provisioning and binding of other resources such as datasets, bandwidth or devises [18]. The recently proposed Service Level Agreement (SLA) based framework of resource management is a first step towards the correction of this trend [18]. It reinforces the widely accepted notion that a service level agreement is an effective mechanism to reconcile the consumer's demand for QoS assurance with the provider's local needs [19].

In an effort to research the issues associated with service-oriented grid architectures, Midland was developed with underlying models and strategies that are inherently service-centric. Web Services are used as the primary integration mechanism both within Midland clusters and at the grid level. A novel model is utilized to provide a uniform quantification of the hosting nodes' service capacity. The model allows a simplified formulation of the core management mechanisms such as service discovery, scheduling, and resource state dissemination. It is also

projected to be instrumental in future implementations of SLA agreement-based resource management which is expected to become central to the development of large scale computational as well as utility grids [18].

The rest of the paper is organized as follows. Section 2 highlights some salient issues encountered in the application of the grid computing paradigm, and the challenges associated with the adoption of service-oriented architectures. Section 3 describes Midland's infrastructure and its underlying management mechanisms and services. Section 4 discusses experimental results relevant to some aspects of Midland's operational performance with particular attention to the model of service capacity. Related works are discussed in Section 5 followed by conclusions given in Section 6.

2. Service-Oriented Grid Computing

Future computational grids are expected to be complex systems that involve heterogeneous computing resources; including raw data from sensors, satellites, and Supervisory Control and Data Acquisition (SCADA) systems, large repositories of data both processed and raw, supercomputers as well as clusters of desktop machines, application services, and rich user interfaces to enable visualization and analysis (see Fig. 1). The integration of these heterogeneous resources requires a convergence of the disciplines of sensing, data management, and computing and application services [20]. In order to appreciate the complexity of the issues associated with grid computing and its applications to real world problems, consider a hypothetical Grid (that we call Water Grid) developed to process large and diverse water related datasets that originate from water treatment plants, municipalities, conservation authorities, public health units, environment ministries, laboratories, and research facilities. Such grid does not only serve as a distributed data management system, but it also provides the necessary computing services to enable researchers to leverage this data to conduct research on all aspects of water quality management and related issues of public health and the environment. The grid nodes are clusters that may provide storage as well as compute capacity and host various application services such as data mining, simulation, visualization, etc. To bring into context the challenges associated with distributed computing on such open system we consider the following use case scenario. A researcher developed a risk model for drinking water quality and decided to simulate the model for a given region of the country using the computing services and the relevant data repositories available throughout the Water Grid. The datasets of interest as well as the computing services necessary for the execution of the risk model may be available from multiple clusters of the grid. In order to satisfy the functional behaviour required by the above use case scenario, the grid management mechanisms have to locate the datasets in question as well as the application services necessary for the execution of the risk model. Subsequently, the data and the risk model binaries are staged to the provider node (provider of the application services) before the simulation is initiated. Upon completion of the simulation task, the results are sent back to the researcher, a copy of which is perhaps catalogued and archived locally as a newly created dataset for future use. The operational behavior required from the grid computational platform to realize this use case scenario is obviously intricate. However, the associated complexity may still be

considered moderate with respect to the future sophistication expected from a grid environment. For instance, the application services needed for the simulation may be hosted by different clusters. This requires co-scheduling and execution flow synchronization across a distributed set of providers that needs to take place in the face of numerous grid environmental challenges; including the ubiquity of faults, the uncertainty on resource state information, the heterogeneity of resources and their intermittent participation, the network latency, and the lack of central control.

Our approach to deal with the heterogeneity and diversity of grid resources as well as the presence of distinct providers and administrative domains is to use the service abstraction in conjunction with a grid-wide uniform quantification model of service capacity to enable an efficient selection and exploitation of resources (see Fig. 2). The proposed capacity model, detailed in Section 3.5, maps the availability indicators associated with a node's computing and data resources to a single service capacity metric that quantifies its aggregated capability to handle the load associated with the provision of a class of hosted services. The uniform use of the service abstraction in the development of Midland management mechanisms such as service discovery, and scheduling paves the way for the future integration of Midland clusters into grid systems that are compliant with the WSRF standards [21]. Furthermore, the use of a single aggregate measure of service capacity enables the development of service discovery strategies that require a low density of data exchange. This leads to lower latency and reduced network congestion in comparison to strategies that rely on queering large centralized or hierarchical resource information repositories of resource attribute-value pairs, whose lack of scalability has been well illustrated in the absence of data caching [22]. Furthermore, the service-centric nature of Midland enables its integration within the service supply-demand framework of resource exploitation which is increasingly gaining prominence in business application of grid computing [14-16, 23]

3. Midland Cluster System

The common thread that runs through Midland is the uniform service view of all resources irrespective of their nature being datasets, documents, compute cycle, application services, storage capacity or network bandwidth. These resources are inevitably bound to physical hosts or devices which could be workstations, storage servers, laptops, interface devises, management devices, or SCADA systems. The Midland system provides a resource management infrastructure through the deployment of an agent on each one of these physical resource hosts and devices. The collection of agents make up a Midland cluster whose operation is coordinated by a single agent designated as the principal. The principal serves as the point of interaction with peer clusters in a computational grid (see Fig. 3). The Midland Grid Server (MGS) is the underlying software implementation of an agent. MGS is a multithreaded pure Java server that can be configured as a simple agent or a principal using a set of XML configuration files. A principal node of a Midland cluster includes an MGS instance configured as a principal, a database server and a Tomcat-Axis web server. In order to provide a reliable operation of the principal node, the MGS is deployed on a host distinct from the one used for the Tomcat-Axis server and the cluster's database. Consequently other cluster agents may

assume the principal role if the current principal succumbs to a fatal failure. An agent node includes an MGS instance configured as a simple agent in addition to a local resource and load information server. Inter-cluster interactions are enabled using Web Services deployed on the Tomcat-Axis server associated with the principal node. For intra-cluster interactions, all data intensive communications are established using Web Services, while commands and light data messages such as status notifications are exchanged using an XML socket based messaging protocol called the Midland Messaging Protocol (MMP) to be described later. The Midland cluster's operation is supported by a set of core management services to be described in more details in Section 4.

3.1 Grid Topology

The grid topology envisaged for the deployment of Midland is a flat network of clusters where identity, state and capacity information is selectively exchanged among neighbors to supports grid wide management strategies such as service discovery and delegation of service request handling (see Fig. 4). In this neighborhood-based topology, there are no restrictions imposed on the geographic proximity or network distances between two neighbors. However, the challenge posed by latency to the adequate operation of tightly coupled grid-enabled applications, such as real time simulations, necessitate the choice of a configuration that makes an effective use of the underlying network infrastructure. In this respect, the chosen grid topology has the properties of a power law network where most nodes have few links and few nodes have numerous links, and as a result it may be more robust against random failure of the agents [24, 25]. Furthermore, since the inter-cluster interactions are always mediated by the principals, the creation of inter-agent overlay networks is avoided, and the problem of excessive traffic illustrated for the case of Gnutella networks may be prevented [26].

In a Midland-enabled grid, each cluster has the necessary infrastructure to schedule as well as handle service requests for which it has a sufficient available capacity. It also has the capability to delegate the handling of service requests to other clusters. While the exchange of state and capacity information is performed regularly among neighbors, establishing a transitory communication link between any two nodes is not restricted. This is in fact needed for data transfer between a node from which a service request originated and a peer to which such request has been delegated for handling.

The grid topology and the configuration of its nodes are defined using the MGS configuration files of the nodes and their agents respectively. The bulk of the information included in the MGS configuration file is described using the UML diagram of Fig. 5 Each cluster is identified by a grid-wide unique identifier (node-id) assigned through an out of bound process.

A map is maintained between the node-id and the agent-ip of the principal so as to insulate the neighboring clusters from any configuration changes taking place within the cluster, such as the assumption of the principal role by a different agent. Each cluster maintains the state information of its neighboring peers and its agents in the cluster-bound database which is accessible to all its member agents. The Statechart of Fig. 6 describes the grid node operational states and the associated

transitions during the node's lifecycle delimited by the join and leave point in time. The agent lifecycle is described using an identical set of states except that the target domain of membership is the cluster instead of the grid. The join process, initiated by the joining node, consists of an exchange of identity information used to update the configurations of the future neighbors. The identity information includes the node-id and a Web Service endpoint used for the inter-node interaction during the join and leave process. The information used by the joining node to initiate the process is obtained out of bound.

The agent join process is essentially the same as that of a node join. The only difference is the assignment of the node-id which in this case is made by the principal. The leave process for a node or an agent consists in no more than setting the states of the leaving node or agent to "disabled". However, the implications on the work in progress are critical to the overall performance and robustness of the computing platform. The section on fault-tolerance discusses some of the related issues.

3.2 Fault Tolerant Architecture

In order to assert its role and enable the selection of a successor in case of failure, the principal sends a periodic heartbeat signal to all agents in the cluster. The process of selecting a new principal (changeover) is initiated as soon as a configured number of agents assert the silence of the heartbeat signal. The new principal is selected from a priority list predefined based on the computing capacity of the agent hosting machines. Since all the agents have access to the cluster state, the configuration information, and the priority list, the changeover process consists essentially of an update to the map between the cluster *node-id* and the new principal's *agent-ip*.

As to the node and agent leave processes, Midland does not currently have any support for work recovery following an unpredictable departing of a node from the grid or an agent from the cluster. Instead, if the leave process is initiated during the "Active" state, the task or service request handling is rescheduled and restarted. The issue of work recovery after failure and unpredictable departure of nodes and agents is actively researched in the broader context of fault-tolerant grid computing [27]. Upon convergence of these research efforts, the resulting strategies and models will be integrated in Midland.

3.3 Midland Messaging Protocol (MMP)

The MMP protocol relies on a synchronized socket based exchange of the objects *MMPRequest,* and *MMPResponse* illustrated in Fig.7. These objects identify the sender along with a username and a password for simple authentication. The *MMPRequest* is designed to include all the necessary information to reconstruct a call to a Java method on the receiving end. The structure of the request object includes a command parameter that is mapped to an action to be taken by the recipient component (or service), as well as the necessary values of the action's arguments when it applies. The arguments may be primitive Java types or complex user defined Java objects that may themselves include other user defined Java objects or collections as their property types. In conjunction with the MPP protocol,

a scheme is developed to serialize Java objects into XML documents. This XML serialization supports the Java HashMap, ArrayList, and arrays. The DataUnit object represents the XML body which includes the set of argument values associated with the action in question. Each one of the types or class properties used within the DataUnit hierarchy is an element (with possible children elements) of the XML serialized document. The serialization approach uses the categorization of the Java objects into primitive and complex types. Complex types include user defined objects and collections. Each property of the object hierarchy being serialized is associated with an object identifier (OID). The basic serialization method consists in constructing a directional reference tree of OIDs, where each OID references a basic KyObject element that describes the specification of the object being serialized. The depth and span of the tree depend on the depth of the object's hierarchy and is always terminated by nodes that represent the properties with primitive types.

3.4 Midland Computational Model

The computational model addresses the specification format and structure of a service request and the task execution process that supports its handling. The model restricts the execution of tasks of the same service request to the same cluster to simplify the management of the execution process and to avoid the latency that would otherwise be incurred from the inter-cluster coordination that would be needed if dependent tasks for the same service request are executed on different clusters. Future works will analyze the tradeoffs associated with the potential relaxation of this restriction.

A typical use of a Midland cluster within a grid starts with the submission of user service request $sr = \{A, S, \Phi, Q\}$ defined as a collection of tasks, each requiring the availability of one or more grid services, to be executed in accordance with a specified workflow. The user required quality of service Q may be specified as a list of performance metrics such as the "maximum wait time before scheduling" or the "time limit before execution start". Φ is the required task execution workflow, $A = \{a_0, a_1, .., a_n\}$ is the set of sr tasks, and $S = \{s_0, s_1, .., s_m\}$ is the set of required grid services. The sr handling process starts with an attempt to schedule the tasks on the local cluster in compliance with the required quality of service. If the available capacity of the required services is sufficient, the sr tasks are queued for local execution, otherwise the execution of the tasks is delegated to a peer cluster using various approaches as will be elaborated in section 4.2.

The management of the task execution workflow is coordinated by the submission cluster, i.e. the cluster where the sr originated. The sr handling lifecycle is subdivided into n consecutive phases where each phase corresponds to the concurrent execution of a set of component tasks of the sr (see Fig. 8). The collection of tasks to be executed in phase i is dependent on the tasks executed in phase $i-1$. The ordering of tasks according to their dependence is inferred directly from the user specification of the submitted sr. Different approaches to the coordination of the sr execution were explored. However, after extensive experimentation, it became apparent, at least in the case of this developed infrastructure, that the coordination of the sr execution through synchronized

interactions was not achievable with an acceptable rate of failure. Some of the fault sources such as runtime exceptions where satisfactorily addressed. However, other failures such as frequent communication timeouts are often caused by congestion and network faults and as such they are difficult to correct especially for the public network used for inter-cluster communication. As a result, an asynchronous process of workflow coordination is used to achieve a more reliable *sr* execution. Coordination messages such as task state change notifications are persistently logged to persistent data stores both at the execution and submission clusters, while the initiation of the successive phases is performed through a periodic polling of the persisted task states.

The message-based asynchronous coordination used in Midland provides improved robustness against faults compared to a synchronized approach. However, this is achieved at the cost of higher *sr* execution overhead. This may be acceptable for data mining applications which often include long running tasks. On the other hand, the support of real time simulations with low tolerance for high latency may require different workflow coordination methods to be investigated.

3.5 Model of Service Capacity

As stated above, a service request may require for its handling the availability of a set of grid services. Such availability requires more than the assertion that a required grid service is indeed deployed on the target hosting environment. In particular, the hosting environment has to possess sufficient resources for the instantiation of the service and the subsequent invocations of its operations. Since the resources of the hosting environment are limited in their levels of availability, the maximum number of service requests that may be concurrently handled is finite. Consequently at any given time a grid service that might be deployed on a grid node may not be available for lack of sufficient resources necessary for its instantiation and the execution of its operations.

Traditionally, computing capacity has been defined in terms of a maximum number of slots assigned to computing hosts based on their CPU and RAM availabilities. While this may be appropriate for pure compute-cycle jobs, it is hardly adequate for complex services that rely on a spectrum of resources which may include CPU, Memory, storage capacity, datasets, network bandwidth, application software, and others services. This list of base resources may be extended to include support infrastructures such as databases, application servers, messaging systems, virtual machines such as the Java Virtual Machine, and runtime environments such as the .NET framework. In the face of a large set of resources that may be required for the handling of a service request, one possible approach to discover and select an appropriate service provider would be to query a potentially large directory of resource attribute-value pairs. Such multi-dimensional search entails a significant complexity of the discovery and selection mechanisms. Furthermore, the resource information systems that have been so far devised to support detailed enumerations of resource attribute-value pairs are not scalable without data caching [22]. Given these concerns, an alternate approach in the form of a service capacity model is developed to provide a quantification of service availability based on the aggregate behavior of the resources of the hosting environment. Let $R = \{r_0, r_1, ..., r_{n-1}\}$ be the set of cluster resources that supports the

operation of hosted services denoted by the set $S = \{s_0, s_1, ..., s_{m-1}\}$. Then we define the aggregate service capacity $C(t)$ of the cluster as follows:

$$C(t) = \sum_{i=0}^{n-1} \alpha_i . \frac{l_i(t)}{l_i^{(\max)}} \quad (1.1)$$

$$\sum_{i=0}^{n-1} \alpha_i = 1 \quad (1.2)$$

$l_i(t) \geq 0$ is the level at the discrete time $t \in \mathbb{N}$ of $r_i \in R$, and $l_i^{(\max)} > 0$ is its all time maximum. Since for some resources such as CPU, the availability is expressed as a percentage of a maximum level, the entity of relevance for the practical implementation of the model is $\frac{l_i(t)}{l_i^{(\max)}}$. Hence we may state for example that the RAM and thread availability is 20% and 50% respectively. The weighting factors $\alpha_i > 0$ specify the relative importance of the various resources in defining the aggregate service capacity of the hosting environment. Resources that are more critical to the operation of the service category, such as runtime memory, are assigned higher weight in the definition of the service capacity. The actual values of these weights are determined through experimental runs for a select service of the category being deployed on the hosting environment in question. The parameters are tuned to achieve zero capacity for a high *sr* load, a medium capacity for a medium *sr* load and a high capacity for a low *sr* load. The mentioned levels of load and capacity are manually estimated by inspecting the response time of the hosting environment in reaction to user interaction. For ease of implementation, the capacity $C(t)$ is scaled up by a factor of a hundred and rounded up to the next integer.

Let $L(t)$ be the *sr* load defined as the number of concurrently handled service requests at time t. Then the average share of $C(t)$ needed to handle a single service request, that we call *servslot*, is estimated as follows:

$$\omega(t) = \omega(t-1) + \lambda \left(\frac{1}{M} \sum_{i=1}^{M} \frac{C(t-i)}{1 + L(t-i)} - \omega(t-1) \right) \quad (2)$$

Where $\omega(0)$ is set to the estimated average share of $C(t)$ consumed by the least demanding member of S. This determination is made through profiling of resource consumption for a randomly distributed load of the service in question. The above estimation scheme provides a numerically stable computation of the moving average value of a servslot. $0 < \lambda < 1$ is a forgetting factor which may be dynamically set to a high value for an *sr* distribution with high makespan, and a low value for an *sr* distribution with low makespan so as to emphasize their respective low and fast dynamics of resource consumption. The width M of the moving average window is also experimentally tuned to remove excessive fluctuation from the estimated value and enable the capture of the stable trend of

$\omega(t)$. Given the above definition of a *servslot*, the estimated number of service requests that can be handled by the hosting environment at time t is given by:

$$N(t) = \frac{C(t)}{\omega(t)} \quad (3)$$

The implementation of the above model of service capacity is enabled through a periodic monitoring of the levels of resource availability. Aspects of this implementation are described in the subsequent sections. In future works, the Common Information Model (CIM) [28] , which is likely to be adopted by the OGF, will be considered for the representation of the computing resources and their availability levels.

4. Cluster Management Services

An overview of Midland management mechanisms is given in Fig. 9. These mechanisms include the following core services:

- Service Request Submission
- Scheduling
- Service Capacity Information Dissemination
- Resource State and Load Monitoring
- Task Management

There are two common threads in the approach to the development of Midland core services. First, Web Service Interfaces (WSI) are uniformly used for the interactions between peer clusters and partly used for the cluster's integration. Second, persistent storage using relational databases is used for all entities vital to the reliable operation of the cluster; including service request specifications, resource and load state information, and service capacity information. As a result, the core services can be deployed on separate hosts which may provide simplifications for failure recovery processes in addition to the potential increase in resilience to faults.

Service discovery is a core service planned to be integrated with Midland. This service is not currently used because of the reliance on the capacity information disseminated among neighbors. However, different service discovery strategies have been explored whereby the distributed network of service registries maintained by the clusters is leveraged to yield a scalable performance [29].

4.1 Service Request Submission

Service request submission to a Midland cluster requires the pre-packaging of the submitted *sr* using a utility developed for this purpose (see Fig. 10). A service request is defined as a collection of tasks (also called jobs) to be executed in compliance with a specified flow. The packaged *sr* submission file includes a specification of the task requirements such as the quality of service and the required grid services or executables. The packaged *sr* file may include the relevant data and

executable binaries or a URL reference to the repository where they are stored along with the necessary access credentials.

The current Midland design recognizes four classes of quality of service; namely: bronze, silver, gold and platinum. The bronze and platinum service classes correspond to the least and most stringent requirements respectively. Although the model of quality of service is flexible and open to extension, the current implementation of these QoS classes is limited to the consideration of the "maximum time before scheduling", the "maximum time before service handling starts", and "the number of restart after failure".

Service requests are submitted through a simple web interface where the *sr* file is uploaded to the submission cluster. The *sr* specification is then parsed and the user receives an *sr* identifier (SRID) used to monitor and track the handling of the submitted *sr*. The data and the binaries are then stored on a secondary data repository accessible through a set of web services. All operations of scheduling and delegation associated with the service requests are performed using a subset of the *sr* specification. This is to avoid the unnecessary movement of binaries and input data before the actual scheduling on the target execution environment. Whence a host environment is selected for the handling of the *sr*, the input data and binaries are then appropriately staged.

4.2 Scheduling

Different scheduling strategies have been explored for a Midland-based grid, among them we can cite two strategies that fit in the multi-step scheduling approach adopted for Midland and which are addressed in [30, 31]. We will refer to the first strategy as Probabilistic Confidence Scheduling (PCS) [31] while the second strategy will be refer to as Entropic Scheduling (ES) [30]. For this chosen multi-step approach, the scheduling problem is seen as a set of two sub-problems, namely: (1) a local scheduling sub-problem; and (2) a delegation sub-problem. For both mentioned scheduling schemes, the future availability of local service capacity is predicted using a Markov chain model. The difference between the two scheduling algorithms lies in the approach to the delegation decision. The PCS algorithm uses a confidence model that estimates, based on past inter-cluster interactions, the likelihood that a delegation of *sr* handling to a given cluster would lead to a successful execution of its tasks [31]. The second algorithm focuses on the uncertainty of the dissemination of capacity information as a source of ineffective scheduling decisions. The Entropy associated with the disseminated capacity information is hence used as a measure to guide the delegation decision [30]. The confidence model used in the PCS strategy essentially yields an index $\lambda_{xy}(t)$ that captures the confidence that cluster x has vis-à-vis a cluster y at time t. In its simplest form the confidence index is computed as the ratio of the number of *sr* delegations from cluster x to cluster y which lead to successful scheduling over the number of *sr* delegations received by x from y within some adjustable window of time. In addition to the confidence model, the PCS approach uses a Markov chain model of service availability to estimate the probability $P_x^{(sr)}(t_0 + TTS)$ that cluster x will have sufficient service capacity to meet the requirement of the *sr* in question within a future window of time TTS (Time-To-Schedule) starting from the

submission time t_0. The TTS is a user specified QoS parameter which quantifies the maximum acceptable time delay before scheduling of the submitted sr. An *sr* arriving through submission or delegation at cluster x would be delegated to a peer cluster only if $P_x^{(sr)}(t_0 + TTS) \leq p_0$, where p_0 is a threshold set to a value less than 0.25. In other words a delegation is chosen instead of local queuing only when it is estimated with a high degree of certainty that cluster x will not have a sufficient service capacity to handle the *sr* in the window of time $[t_0 \quad t_0 + TTS]$. If an *sr* is to be delegated the target cluster would be the neighboring cluster z of x associated with the highest confidence index among all neighbors $Ne(x)$ of x. In particular, the cluster z is chosen so that $\lambda_{xz}(t) = \max_{z \in Ne(x)} \lambda_{xz}(t)$. The process of delegation is repeatedly applied following the above strategy until the *sr* is scheduled or the process is aborted through a direct intervention from the user or a user specified QoS timeout is reached. For the ES strategy, the state of available capacity for hosted services is disseminated among neighboring clusters. In this case the decision to delegate an *sr* instead of locally queuing it at the current cluster x is taken when it is asserted that it is unlikely that the local cluster would have sufficient capacity to handle the *sr* within $[t_0 \quad t_0 + TTS]$ and that the chosen target cluster $z \in Ne(x)$ for delegation is associated with the lowest value of the defined Entropy function among the neighbors $Ne(x)$ of x [30].

The multi-step, decentralized scheduling model is adopted in Midland because of its scalability and inherent accommodation of the distinct administrative domains present in a grid. Furthermore, the model allows the inclusion of incentives for the consumption of closer services before seeking the service availability of distant providers at a higher cost of network bandwidth and failure rate. In the current implementation of Midland, the scheduler components use a Web Service Interface to interact with the SR Database and the service registry (Fig. 9). As a result different implementations of the scheduling service may be used with little effect on the rest of the infrastructure.

4.3 Service Capacity Information Dissemination

The function of the service capacity dissemination service is to relay to immediate neighbors the capacity and load information associated with its hosted services. The actual entities being exchanged include the service name, the capacity $C(t)$ as defined by relation (1), the available number of *servslots* $N(t)$ given by equation (3), and the *sr* load $L(t)$. This information exchange is performed with a frequency comparable to that of the agent's resource monitoring services. The approach to the dissemination of service capacity information is devised to enable the realization of an adequate scalability and performance of the dependent management mechanisms such as scheduling and service discovery. In particular, consideration was given to the following critical issues:

- Scalability of the dissemination strategy
- Limited network bandwidth
- Overhead storage of service capacity information

- Network and processing latency
- Uncertainty associated with disseminated information

The scalability issue is partly addressed through the choice of a grid topology that limits the flow of capacity information to pathways established between neighbors as mentioned in previous sections. Furthermore, the limited amount of data being exchanged requires a modest storage and processing overhead in addition to the consequently low bandwidth consumption and overall network and processing latency. Concerning the uncertainty on the disseminated service capacity information, Midland includes a simple version of a stochastic estimation model that has been explored to enable potential consumer clusters to assess the uncertainty on the service capacity in light of its stochastically approximated dynamics [32]. In this model the dynamics of service capacity is viewed as spanning a finite set of discrete states corresponding to defined levels of availability. The inter-cluster dissemination of information about the available service capacity enables each cluster to maintain a stochastic prediction model about the capacity of peer neighboring clusters. In particular, the model enables the estimation of the probability $P_i^{(x,s)}(n)$ that a service s from cluster x is in a state i of availability at the discrete time n. One of the simplest forms of the model is based on the approximation $\hat{P}_i^{(x,s)}(n) = \frac{M_i^{(s)}(n, n-k)}{k+1}$, where $M_i^{(s)}(n, n-k)$ is the number of instances the service s has been observed, based on the disseminated capacity information, to be in state i within the window $[n-k \quad n]$ of discrete time. The above estimated probability, which accounts for the past capacity information, enables a consumer cluster to quantify the level of confidence that it ought to have in the accuracy of the capacity information received at any given point in time.

4.4 Resource State and Load Monitoring

This service involves the agents deployed on the computing hosts and the principal that coordinates the operation of the cluster. The monitoring of resource state and load at the various agent-managed resource hosts is performed using the Resource Load Information Service (RLIS). Currently, Midland supports an implementation of RLIS for Windows XP in the form of a Windows Service. Windows Management Instrumentation libraries are used to enable the collection of resource state and load attributes as well as running tasks' status [33]. The periodically collected information is communicated to the cluster management infrastructure using a Web Service Interface. The integration of the Network Weather Service (NWS) is considered for the future support of Unix and Linux platforms [34].

4.5 Task Management

The task management is performed by a dedicated Execution Manager (XEM) deployed as part of the MGS agent. Since MGS is a pure Java server application, the support of Windows XP, as the most popular desktop OS, presents some challenges. One of these challenges is the fact that all Java executables are automatically identified by the OS with a "java" image name. Hence, no distinction

can be made based on the image name for the various Java tasks initiated by the XEM. Furthermore, the Java runtime does not return the OS generated process ID that identifies the task being executed. The identification of the task by its process ID is critical to its management. Our implementation consists in renaming the "java.exe" with a name that includes the Midland generated task ID, then use the information supplied by RLIS to map the task ID to the actual OS assigned process ID by identifying the presence of the task ID in the image name. This operation requires a clean up of the renamed "java.exe" executable as soon as the associated task completes or is terminated. As evident from the above, the RLIS acts also as a task status monitor. The support of other operating systems such as Linux and UNIX is planned for future works.

5. Experimental Results

The objectives of the experimental work are to illustrate the operation of Midland and to quantitatively characterize the operational performance of a Midland managed grid. Given the size of Midland's infrastructure and the diversity of the associated services, a full experimental analysis and characterization of its operational performance requires a dedicated article. As a result two aspects are selected for treatment in this section; namely: (1) the model of service capacity; and (2) the throughput performance of a Midland-enabled grid.

The operational performance of Midland is illustrated using a grid of three clusters configured as immediate neighbors and connected using a 100 Mbps local area network. Each cluster comprises of a collection of desktop machines equipped with a Pentium 4(2.79 GHZ) processor, 0.5 GBytes of RAM and running Windows XP. Each cluster includes an IBM xSeries 345 Server with dual Intel Xeon processor (3.2GHz/533MHz front-side bus) and 1GB of RAM. Currently, Midland uses the Apache Tomcat 5.5.20 with Axis2-1.1 and MySQL 4.1.12-nt which are hosted by the IBM xSeries server nodes. A set of sorting services based on the bubble, inversion and selection algorithms are deployed on all three clusters. The resource set considered for the experiments includes the CPU, RAM, Virtual Memory, and running threads. The inter-arrival time of the service requests submitted to the cluster follows an exponential distribution, while the actual sorting service requested is randomly chosen from one of the three possibilities. The size of data being sorted is randomly generated using a Gaussian distribution where μ and σ designate the mean and the standard deviation respectively. The standard deviation provides a mechanism for the generation of an arbitrarily wide or narrow distribution of the service request's makespan. For the reported experiments the levels of the considered resources (CPU, RAM, VRAM) are expressed as percentage of the all time maximum values.

In order to illustrate the ability of the proposed model to adequately capture the capacity of the hosting environment, a Gaussian distribution of service requests with σ=30 is used for all three clusters. Figs 11-13 illustrate the service load (number of concurrently handled service requests), the service capacity, and the actual distribution of the service request's makespan (running time) for one cluster. Similar results were obtained for the other clusters and are hence omitted for the sake of conciseness. The correlation between the service load and actual service capacity, whereby a high load is associated with low capacity and a low load is

accompanied with a high capacity, suggests that the model does indeed reflect the capacity of the hosting environment. Furthermore, the resources' dynamics of availability do not exhibit any pattern of resource starvation for a relatively long running experiment (Figs 14-15). This absence of resource starvation indicates a healthy regulation of service load that is partly the result of an adequate estimation of the available capacity by the model. Indeed, given the random service request distribution used for this experiment (Fig. 13), a non satisfactory estimation of the capacity would have lead to an overload irrespective of the performance of the scheduling mechanism used for the regulation of resource consumption and load.

The second set of experiments provides a characterization of a Midland-enabled grid with respect to throughput for different values of the standard deviation σ of the service requests' distribution. For these experiments, clusters may delegate the handling of their service requests to neighbors. A delegation is performed if the local service capacity is below a threshold of 5%. The cluster with the highest available service capacity is selected to handle the service request provided that such capacity is higher than 5%. Otherwise the service request is re-queued locally. Figs 16-19 show the grid-wide average throughputs and corresponding makespan distributions for σ=20, and σ=40 respectively. The different values of standard deviations associated with the size of the sorted data resulted in a widening of the service request distribution as well as an increase in the occurrence of instances of service requests that have higher makespan as shown in Figs 16, and 18. In the face of these visibly random distributions with different degrees of deviations, the grid throughput converged towards stable levels after the fluctuation of a relatively short transitory period (see Figs 17, and 19). The steady state levels of throughput can be correlated to the makespan distributions whereby a lower throughput results from longer running tasks of the service requests. The overall stability and consistent throughput performance for random distributions of different deviants of service demand are good features of Midland. This is especially important since throughput stability has been used as a criteria in comparative analysis of batch scheduling systems [35].

Since this paper is primarily intended to describe some proposed models and architectural mechanisms for the design of service-oriented grid management infrastructures, more experimental reporting and analysis will be carried out in alternate venues to covers other aspects of Midland operational performance.

6. Related Works

Midland is developed around service-oriented management mechanisms that are reliant on a model of uniform service capacity quantification instead of the traditional attribute-value based resource information models. As a result, existing grid middleware systems such as Globus [8], UNICORE [5], Condor [1], AppleS [36], and Legion [4], are related to the research effort reported in this paper but only in general terms of their function and in some cases with respect to elements of architecture and strategies of management. For example, UNICORE uses standard protocols such as SSL over public networks, and socket-to-socket communication over trusted intranets. This is similar to Midland where inter-cluster interactions use the Simple Object Access Protocol (SOAP) [37], while cluster-bound connectivity is established using MMP over sockets. The service-oriented approach to the

quantification of resource availability is at the root of the distinction between Midland scheduling strategies and other approaches that are based on matching jobs to resource hosts with the use of resource description languages such as Globus RSL[38] and Condor ClassAds [1]. In particular, Midland's use of an aggregated service capacity measure leverages the service abstraction to encapsulate resource heterogeneity and job requirement diversity enabling hence the avoidance of the multi-dimensional search that is required for scheduling based on resource-attribute matching of jobs to machines found in batch computing systems such as LSF[39], PBS[35] and Condor[40]. Compared to these widely used batch scheduling systems, Midland is conceived with a service-centric philosophy to facilitate the integration of future virtualized distributed computing environments expected to provision services to be consumed based on the required functional behavior and QoS and where it will be no longer needed to specify some required configuration of the computing platform with specific parameters such as operating systems and processor architectures.

To the best of our knowledge the work reported in [23, 41] is the only research effort that espoused the above notion of service capacity quantification as a means to effectively manage resource exploitation within a grid. In [41], a supply-demand control system is used to regulate the consumption of services provided by a federation of data centers organized as a Utility Grid. The node capacity is quantified using two metrics called *server share* and *service share* respectively. The *server share* is defined as the maximum load that can be handled by a server environment for a given class of services, and the *service share* is the portion of the *server share* required by a given service. Given the use of a benchmark application for the definition of the *server share*, it is not clear how the model may be extended to be applied beyond a single administrative domain. In contrast, Midland uses a dynamically assessed capacity unity (*servslot*) to avoid the need for benchmarking service load which is inevitably difficult to apply in an open grid of distinct providers and administrative domains.

There are other works reported in the literature with a focus that may be deemed related to that of the strategies and models underlying Midland's infrastructure [41-43]. The super-peer model [42] has been originally proposed to organize resource consuming clients and a server within a super-peer cluster connected to other super-peers so as to achieve a scalable mechanism of resource discovery. This model has been adopted in the development of a distributed grid information system to avoid the drawbacks of centralized and hierarchical discovery mechanisms [44]. The neighborhood-based grid topology enabled by Midland is similar to a super-peer network in that both topologies are chosen to reduce the effect of latency and the bottleneck of centralized resource information systems. However, this comparison may not be extended beyond the similarity of the topological models since the aim of a super-peer cluster is to enable its client members to find resources quickly, whereas a Midland cluster is geared towards the effective exploitation of the resources managed by its member agents.

In [43], a distributed publish-subscribe messaging system implemented through a network of event brokers is put forth to enable asynchronous peer interactions of a proposed peer-to-peer grid. The broker network may be organized in an arbitrarily deep hierarchy with a single broker at each layer serving as a gateway. While the scope of this related work is limited to the development of the messaging infrastructure, it does along with the work on super-peer networks endorse the approach taken in Midland with respect to the choice of a grid topology that is

conducive to the minimization of the effect of latency.

7. Conclusion

The paper describes Midland, a service oriented management infrastructure for resource clusters. This represents an effort towards the development of a grid management infrastructure where the diversity of the computing resources being exploited is handled through a uniform service view so as to facilitate the future implementation of needed mechanisms such as dynamic negotiations of SLAs in the face of varying demands. Furthermore, Midland grid management mechanisms and services are developed along a service-centric formulation in order to pave the way for future compliance with community standards such as the WSRF. The experimental results show that the proposed model of service capacity provides a satisfactory performance for a realistic grid test-bed driven by a random service request distribution. The average throughput of the Midland –enabled grid is shown to be equally satisfactory for two different spreads of the service request distributions.

References

[1] D. Thain, T. Tannenbaum, and M. Livny, Distributed computing in practice: The Condor experience, Concurrency Computation Practice and Experience, 17 (2005), 323-356.

[2] S. Zhou, X. Zheng, J. Wang, and P. Delisle, Utopia: A load sharing facility for large, heterogeneous distributed computer systems, Software - Practice and Experience, 23 (1993), 224-234.

[3] G. Holt, Time-critical scheduling on a well utilised HPC system at ECMWF using loadleveler with resource reservation, in: Lecture Notes in Computer Science, 2005, pp. 102-124.

[4] A. S. Grimshaw and A. Natrajan, Legion: Lessons learned building a grid operating system, Proceedings of the IEEE, 93 (2005), 589-603.

[5] M. Romberg, The UNICORE grid infrastructure, Scientific Programming, 10 (2002), 149-157.

[6] D. Chen, A. Demichev, D. Foster, V. Kalyaev, A. Kryukov, M. Lamanna, V. Pose, R. Rocha, and C. Wang, OGSA Globus Toolkits evaluation activity at CERN, in: Nuclear Instruments and Methods in Physics Research, Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 2004, pp. 80-84.

[7] R. L. Henderson, Job scheduling under the Portable Batch System, Lecture Notes in Computer Science: Springer Verlag, Heidelberg, D-69121, Germany, 1995), pp. 279-294.

[8] I. Foster, Globus Toolkit Version 4: Software for Service-Oriented Systems., Journal of Computer Science and Technology, 21 (2006), 513-520.

[9] S. C. Farantos, S. Stamatiadis, N. Nellari, and D. Maric, *Grid Enabling Technology*, ENACTS 2002.

[10] K. Miura, Overview of Japanese National Research Grid Initiative (NAREGI) project, Fujitsu Scientific and Technical Journal, 40 (2004), 196-204.

[11] D. Bernholdt, et al., The Earth System Grid: Supporting the next generation of climate modeling research, Proceedings of the IEEE, 93 (2005), 485-495.

[12] P. Avery and I. Foster, The GriPhyN Project: Toward Petascale Virtual Data Grids, 2001.

[13] T. Boden, The grid enterprise - Structuring the agile business of the future, BT Technology Journal, 22 (2004), 107-117.

[14] C. H. Crawford, G. P. Bate, L. Cherbakov, K. L. Holley, and C. Tsocanos, Toward an on demand service-oriented architecture, IBM Systems Journal, 44 (2005), 81-107.

[15] J. Joseph, M. Ernest, and C. Fellenstein, Evolution of grid computing architecture and grid adoption models, IBM Systems Journal, 43 (2004), 624-645.

[16] A. Leff, J. T. Rayfield, and D. M. Dias, Service-level agreements and commercial grids, IEEE Internet Computing, 7 (2003), 44-50.

[17] GGF, *Open Grid Services Architecture (OGSA)*, Global Grid Forum 2004.

[18] K. Czajkowski, I. Foster, and C. Kesselman, Agreement-based resource management, Proceedings of the IEEE, 93 (2005), 631-643.

[19] L.-O. Burchard, M. Hovestadt, O. Kao, A. Keller, and B. Linnert, The virtual resource manager: An architecture for SLA-aware resource management, in: 2004 IEEE International Symposium on Cluster Computing and the Grid, CCGrid 2004, 2004, pp. 126-133.

[20] G. Fox, Integrating computing and information on grids, Computing in Science and Engineering, 5 (2003), 94- 96.

[21] OASIS, Web Services Resource Framework (WSRF) - http://docs.oasis-open.org/wsrf/wsrf-primer-1.2-primer-cd-01.pdf, (2005),

[22] X. Zhang and J. M. Schopf, Performance analysis of the Globus toolkit monitoring and discovery service, MDS2, in: IEEE International Performance, Computing and Communications Conference, Chicago, IL USA, 2004, pp. 843-849.

[23] S. Graupner, V. Kotov, A. Andrzejak, and H. Trinks, *Control Architecture for Service Grids in a Federation of Utility Data Centers*, HP Labs 2002.

[24] M. Faloutsos, P. Faloutsos, and C. Faloutsos, On power-law relationships of the Internet topology, Computer Communication Review, 29 (1999), 251-262.

[25] A. Barabási and R. Albert, Emergence of Scaling in Random Networks, Science, 286 (1999), 509-512.

[26] M. Ripeanu, A. Iamnitchi, and I. Foster, Mapping the Gnutella network, IEEE Internet Computing, 6 (2002), 50-57.

[27] Y. Derbal, A new fault-tolerance framework for grid computing, Multiagent and Grid Systems, 2 (2006), 115 - 133.

[28] Desktop Management Task Force, *Common Information Model (CIM)*, *http://www.dmtf.org/spec/cims.html*, 1999.

[29] Y. Derbal, Confidence-based grid service discovery, International Journal of Web and Grid Services, 4 (2008), 189-210.

[30] Y. Derbal, Entropic grid scheduling, Journal of Grid Computing, 4 (2006), 373-394.

[31] Y. Derbal, A probabilistic scheduling heuristic for computational grids, Multiagent and Grid Systems, 2 (2006), 45-59.

[32] Y. Derbal, Probabilistic resource state estimation in networked environments: the case of computational grids, in: Proceedings of the Third Annual Conference on Communication Networks and Services Research (CNSR 2005), Halifax, Nova Scotia, 2005, pp. 189-194.

[33] Windows Management Instrumentation (WMI). http://www.microsoft.com/whdc/system/pnppwr/wmi/default.mspx,

[34] Rich Wolski, Neil T. Spring, and J. Hayes, The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing, Journal of Future Generation Computing Systems, 15 (1999), 757--768.

[35] T. El-Ghazawi, K. Gaj, N. Alexandridis, F. Vroman, N. Nguyen, J. R. Radzikowski, P. Samipagdi, and S. A. Suboh, A performance study of job management systems, Concurrency Computation Practice and Experience, 16 (2004), 1229-1246.

[36] F. Berman, R. Wolski, H. Casanova, W. Cirne, H. Dail, M. Faerman, S. Figueira, J. Hayes, G. Obertelli, J. Schopf, G. Shao, S. Smallen, N. Spring, A. Su, and D. Zagorodnov, Adaptive computing on the grid using AppLeS, IEEE Transactions on Parallel and Distributed Systems, 14 (2003), 369-382.

[37] W3C, SOAP Version 1.2 Part 0: Primer - http://www.w3.org/TR/soap12-part0/, (2002),

[38] K. Czajkowski, Foster, I., Karonis, N., Kesselman, C., Martin, and S. S., W., and Tuecke, S. 1998., A resource management architecture for metacomputing systems., in: Lecture Notes in Comuter Science, vol 1459, pp. 62-82.

[39] S. Zhou, LSF: load sharing in large-scale heterogeneous distributed systems, in, 1992., pp.

[40] M. J. Litzkow, M. Livny, and M. W. Mutka, Condor - a hunter of idle workstations, in: Proceedings of the International Conference on Distributed Computing Systems, San Jose, CA, USA, 1988, pp. 104-111.

[41] S. Graupner, V. Kotov, A. Andrzejak, and H. Trinks, Service-centric globally distributed computing, IEEE Internet Computing, 7 (2003), 36 - 43.

[42] B. Beverly Yang and H. Garcia-Molina, Designing a super-peer network, in: 19th International Conference on Data Engineering, Bangalore, India, 2003, pp. 49-60.

[43] G. Fox, S. Pallickara, and X. Rao, Towards enabling peer-to-peer Grids, Concurrency Computation Practice and Experience, 17 (2005), 1109-1131.

[44] C. Mastroianni, D. Talia, and O. Verta, A super-peer model for resource discovery services in large-scale Grids, Future Generation Computer Systems, 21 (2005), 1235-1248.

Service Oriented Computing and Applications - SOCA (2009) 3:109–125
The original publication is available at www.springerlink.com

Figure Captions

Fig. 1: The grid is a complex large scale open system with two distinct management domains; namely: the local management domain associated with the resource owner, and the grid level integration management domain.

Fig.2: The grid can be viewed as a service supply-demand system regulated by an SLA-based resource exploitation mechanism that reconciles the consumer's demand for the assurance of a QoS with the provider's local needs for resource availability and performance.

Fig. 3: Infrastructure of a Midland-based cluster.

Fig. 4: Grid Neighborhood Topology.

Fig. 5: UML schema of a Midland cluster configuration file.

Fig. 6: Midland Node operation's lifecycle.

Fig. 7: Overview of the request and response objects of the Midland Messaging Protocol.

Fig. 8: Task dependence and service request's execution phases.

Fig. 9: Overview of the components and services involved in the implementation of the resource management mechanisms and the *sr* handling process in a Midland computing cluster.

Fig. 10: The SrBuilder utility provides an easy to use interface to build the Service Request specification.

Fig. 11: Service Load (number of concurrently handled service requests) for σ =30.

Fig. 12: Available (residual) Service Capacity σ =30.

Fig. 13: Distribution of service request Makespan σ =30.

Fig. 14: Resource Load for σ =30.

Fig. 15: Thread Load for σ =30.

Fig. 16: Distribution of service request Makespan for σ =20.

Fig. 17: Throughput (number of completed service request per minute) for σ =20.

Fig. 18: Distribution of service request Makespan for σ =40.

Fig. 19: Throughput (number of completed service request per minute) for σ =40.

Service Oriented Computing and Applications - SOCA (2009) 3:109–125
The original publication is available at www.springerlink.com

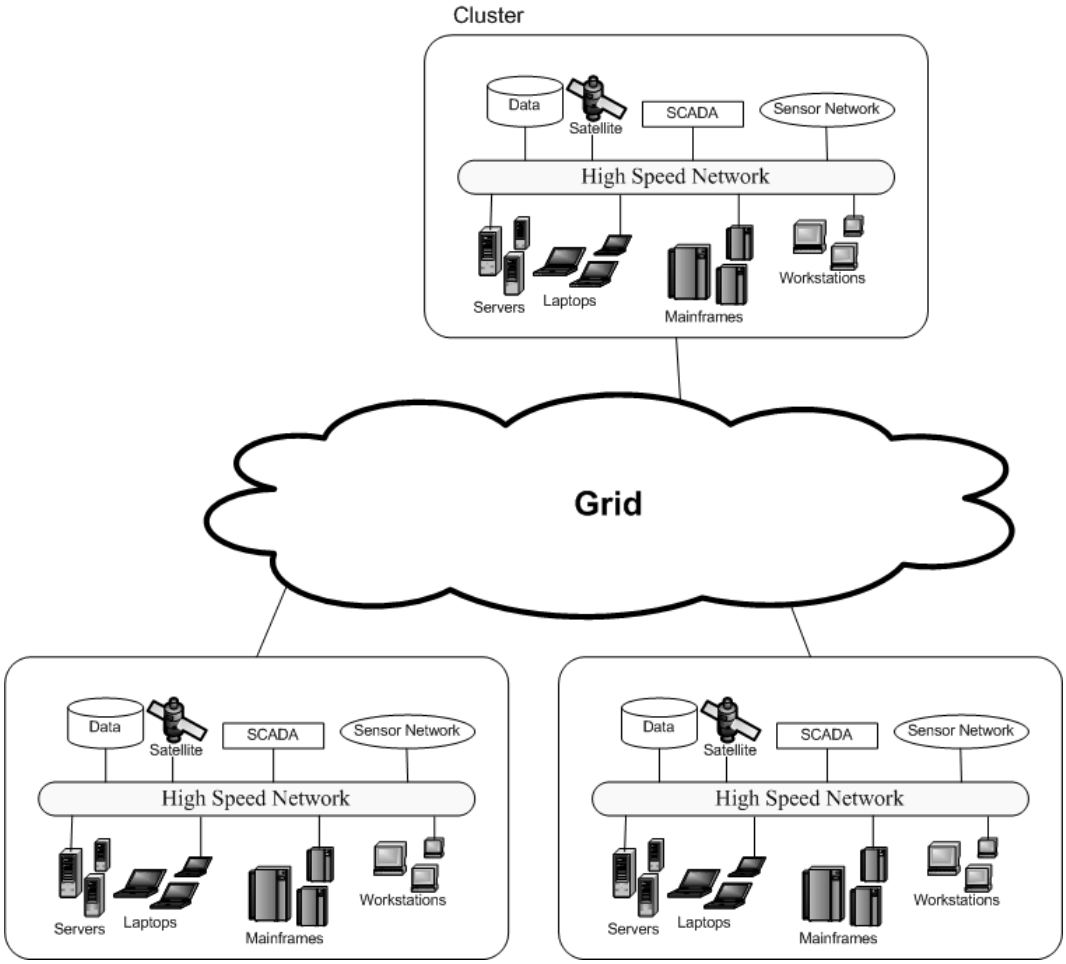

Figure 1

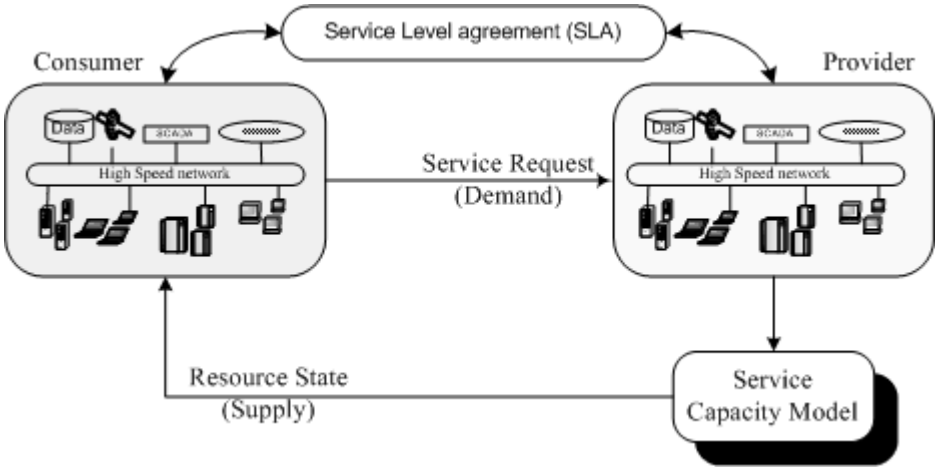

Figure 2

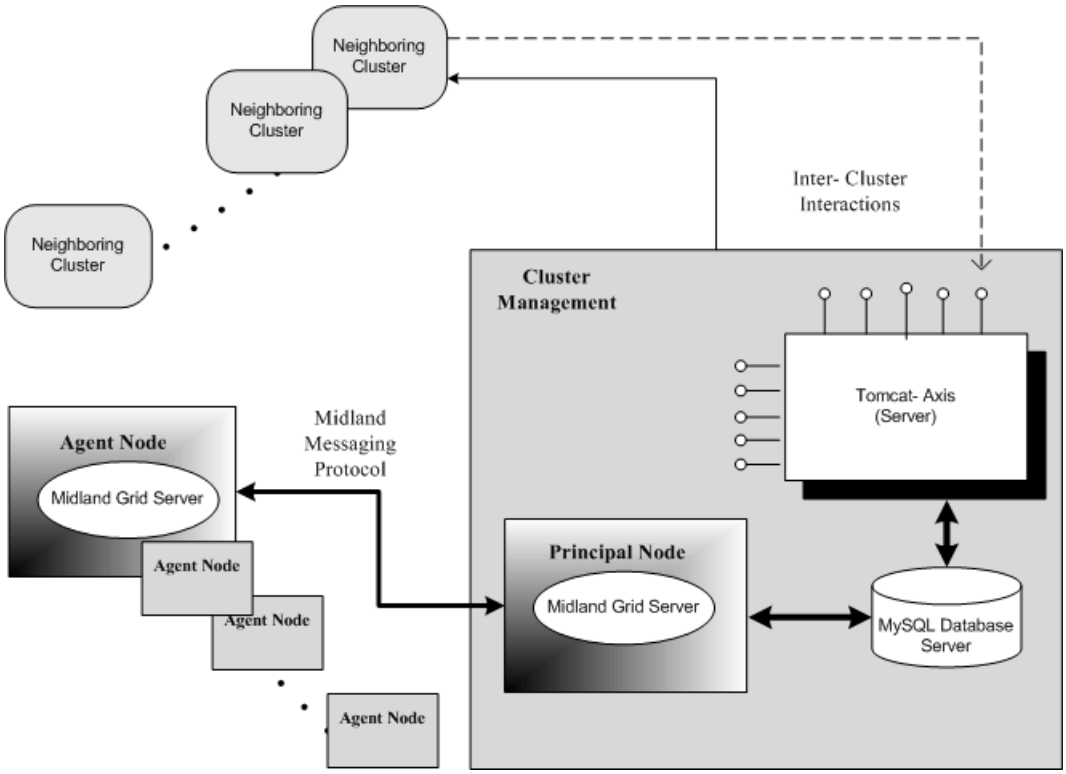

Figure 3

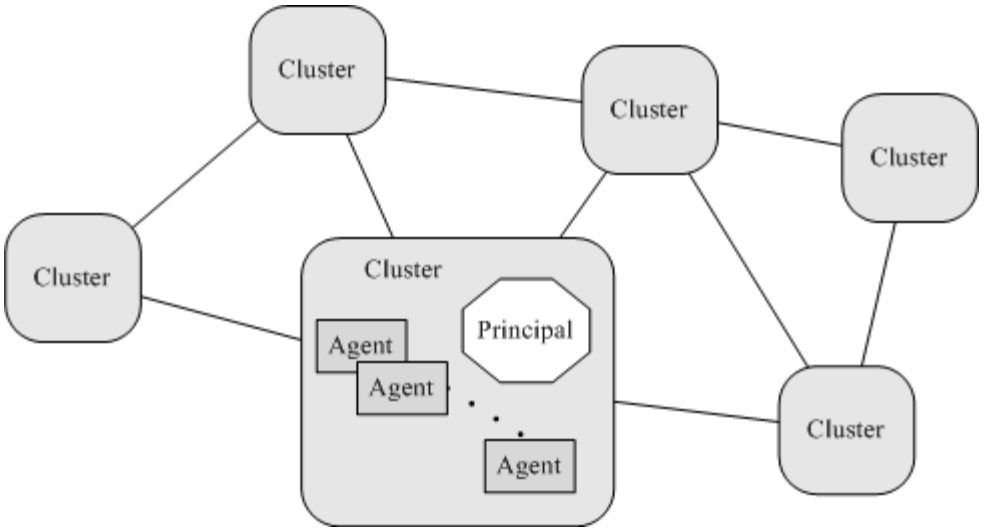

Figure 4

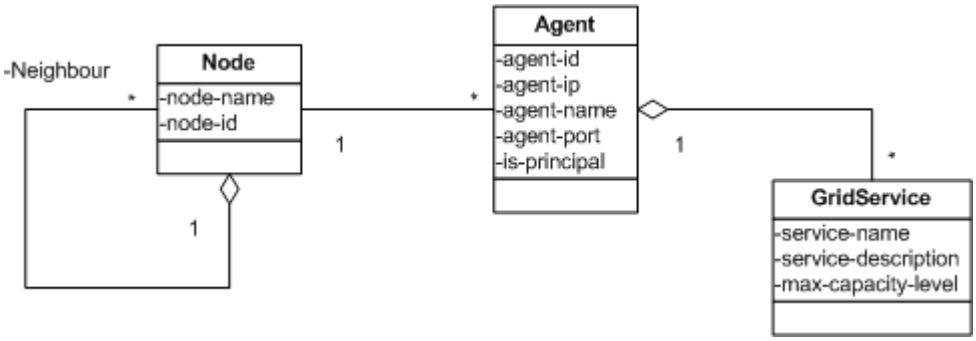

Figure 5

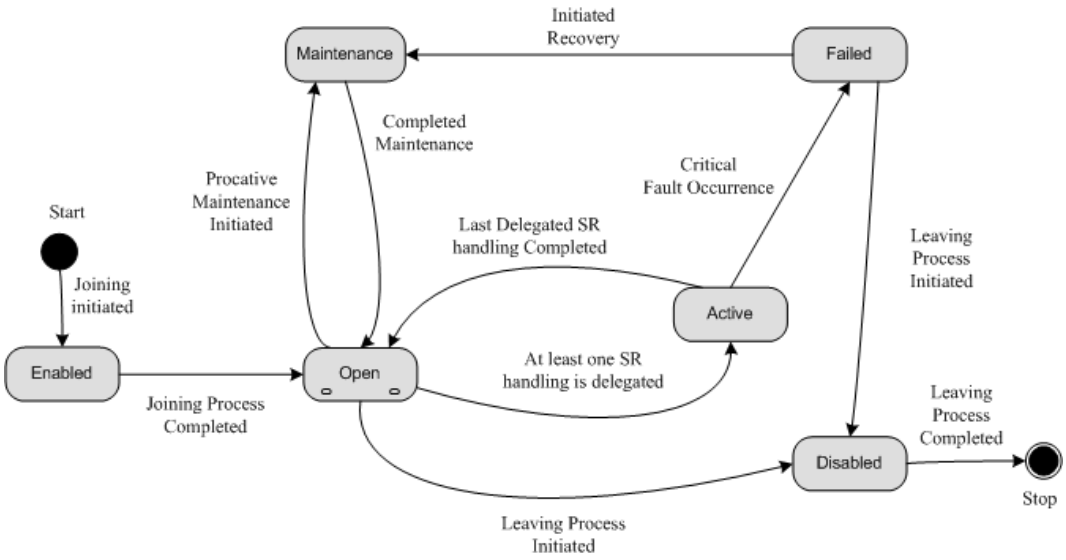

Figure 6

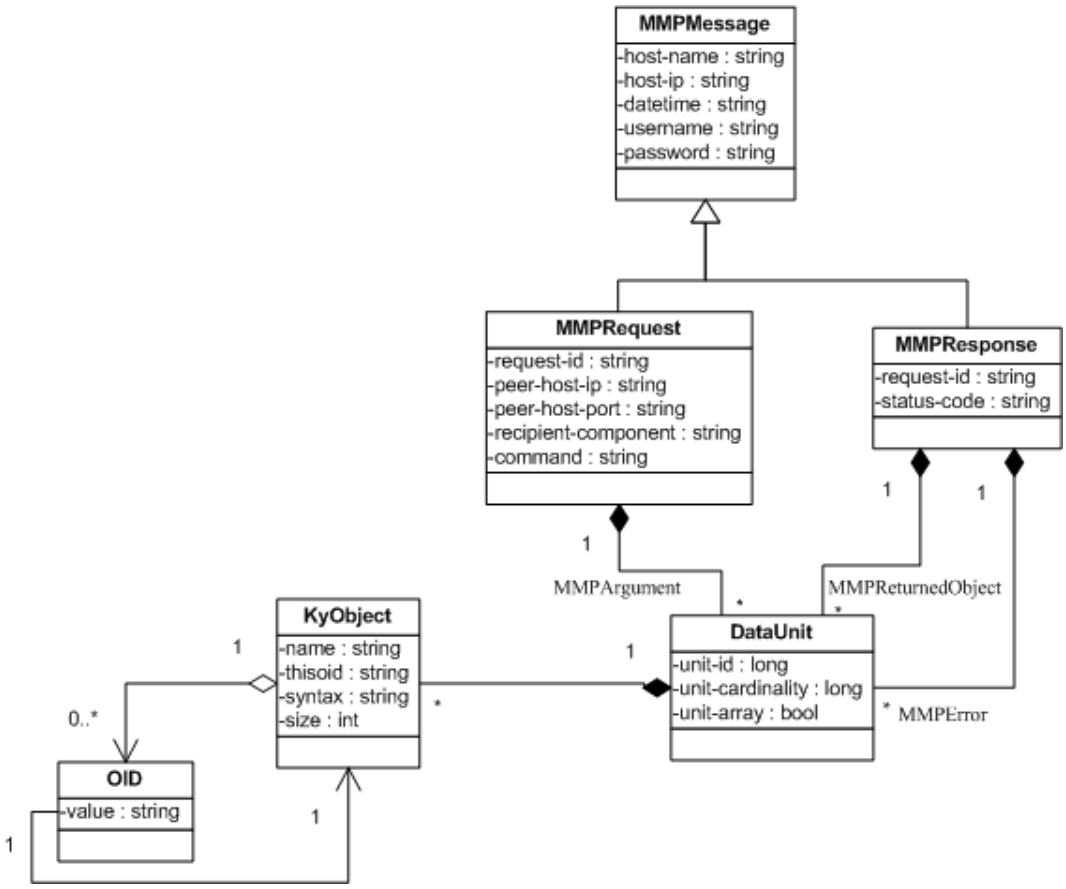

Figure 7

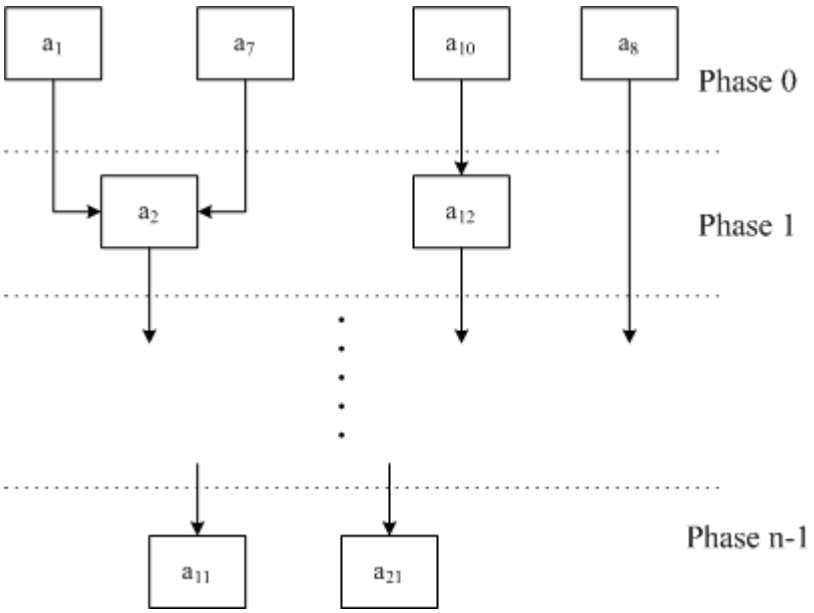

Figure 8

Service Oriented Computing and Applications - SOCA (2009) 3:109–125
The original publication is available at www.springerlink.com

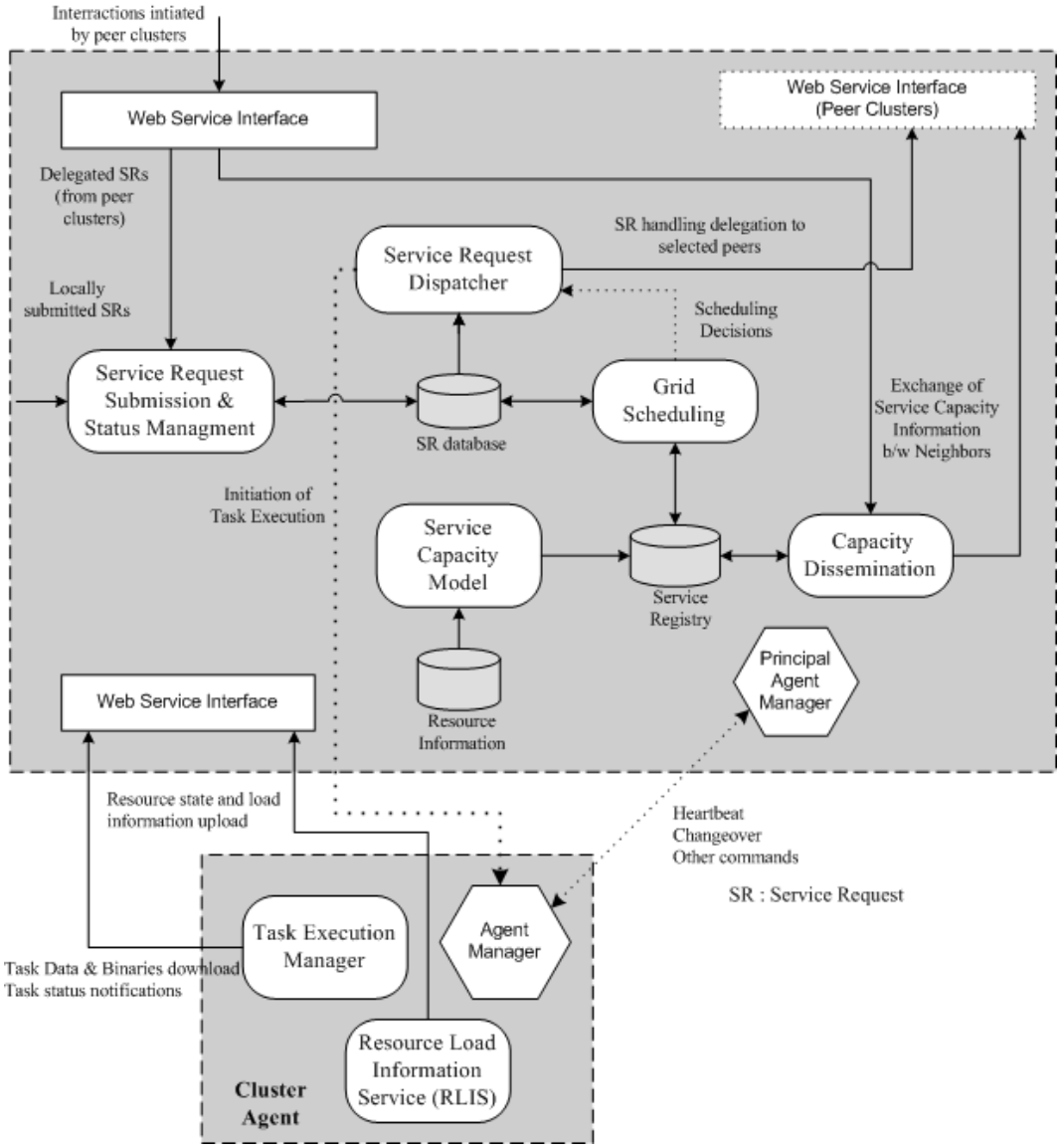

Figure 9

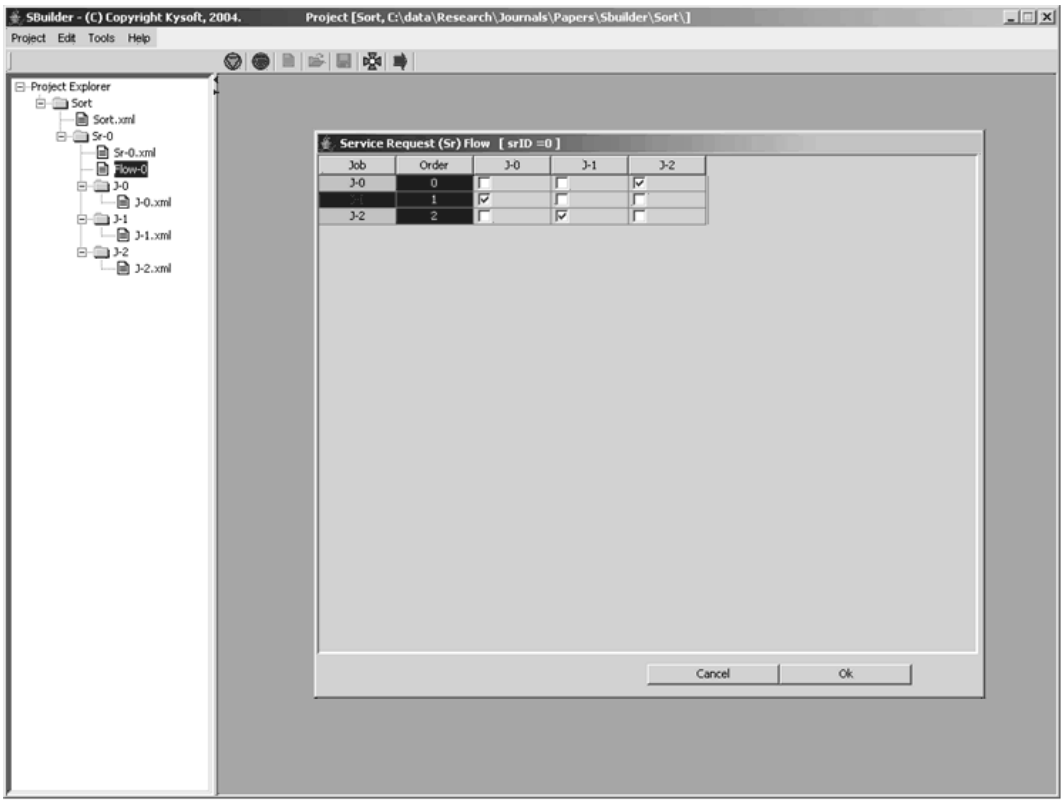

Figure 10

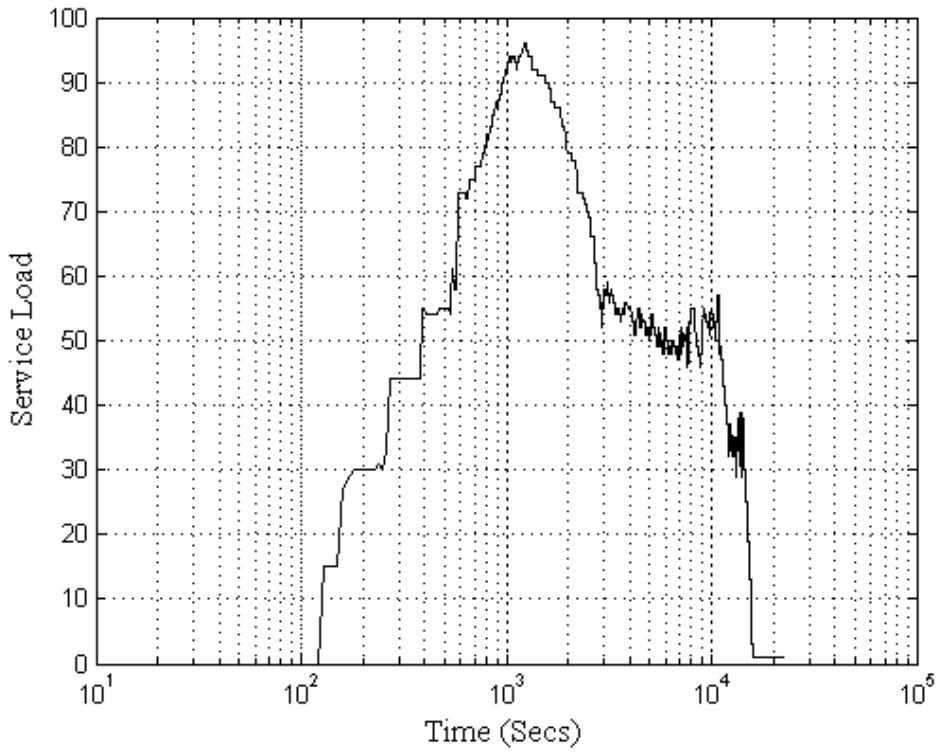

Figure 11

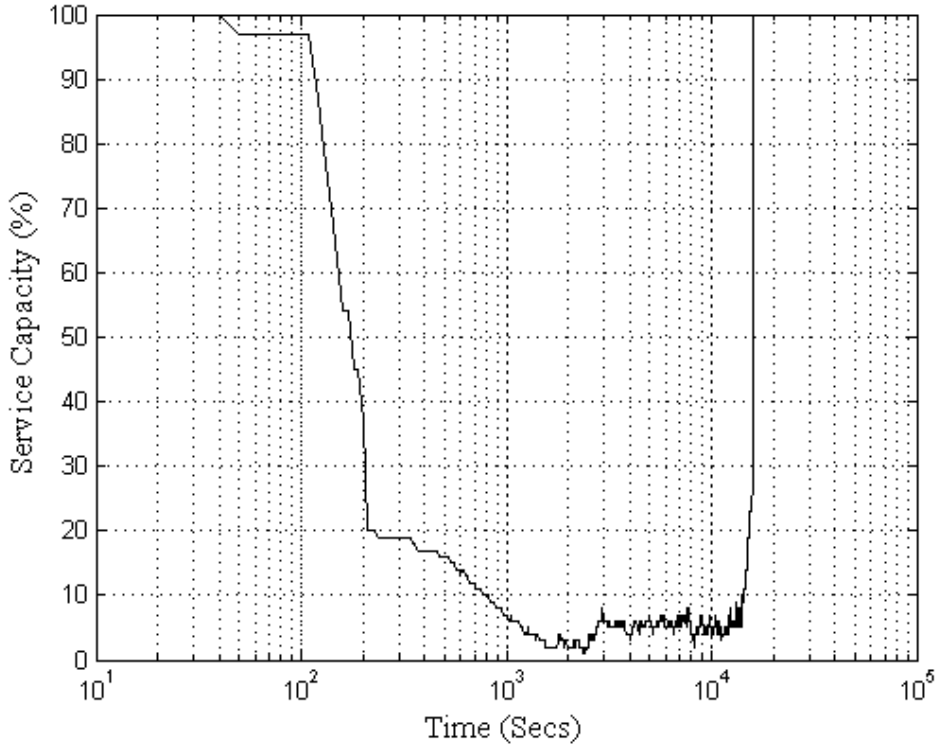

Figure 12

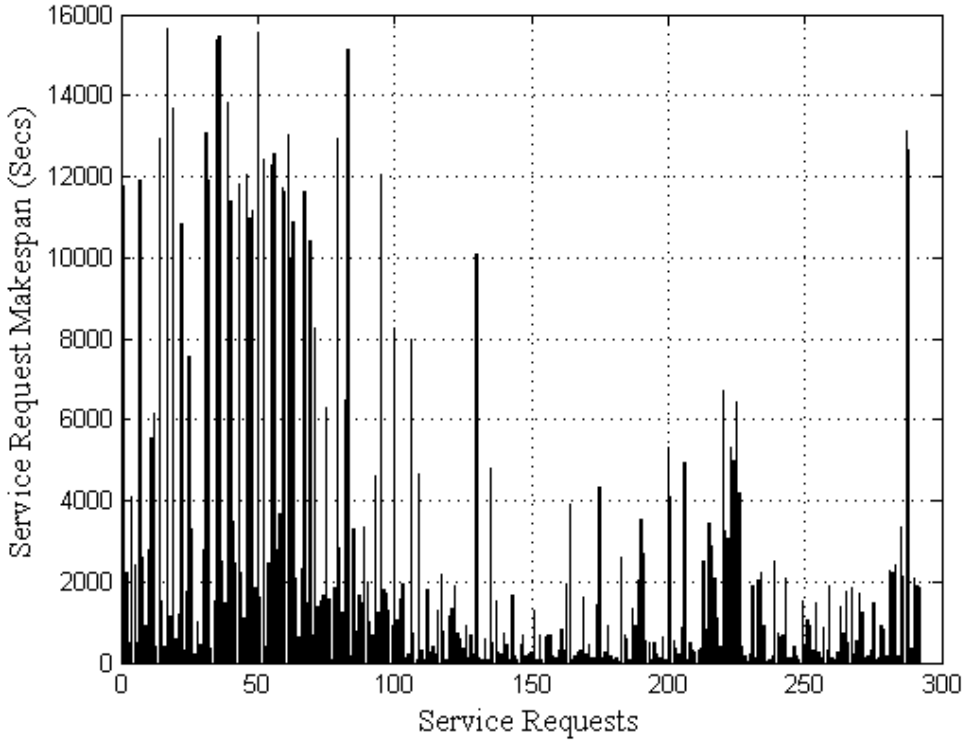

Figure 13

Service Oriented Computing and Applications - SOCA (2009) 3:109–125
The original publication is available at www.springerlink.com

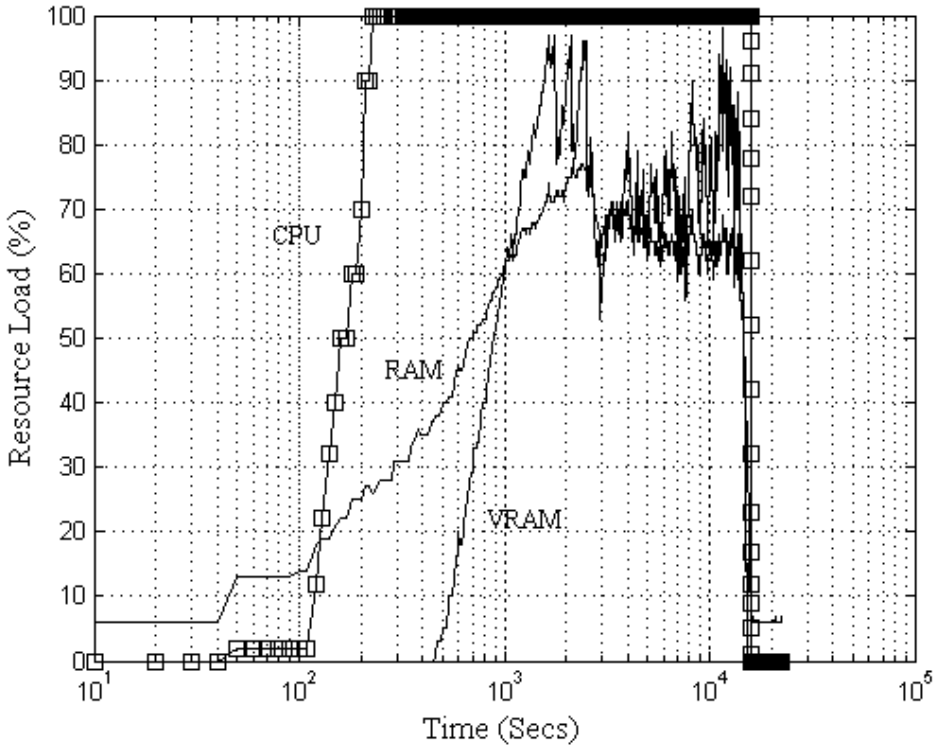

Figure 14

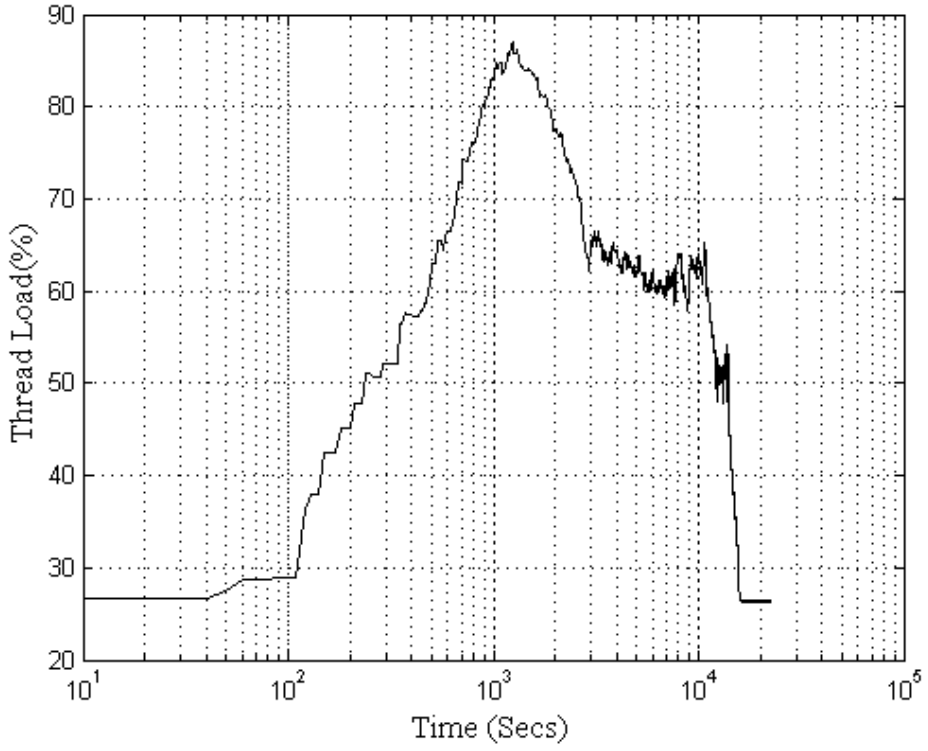

Figure 15

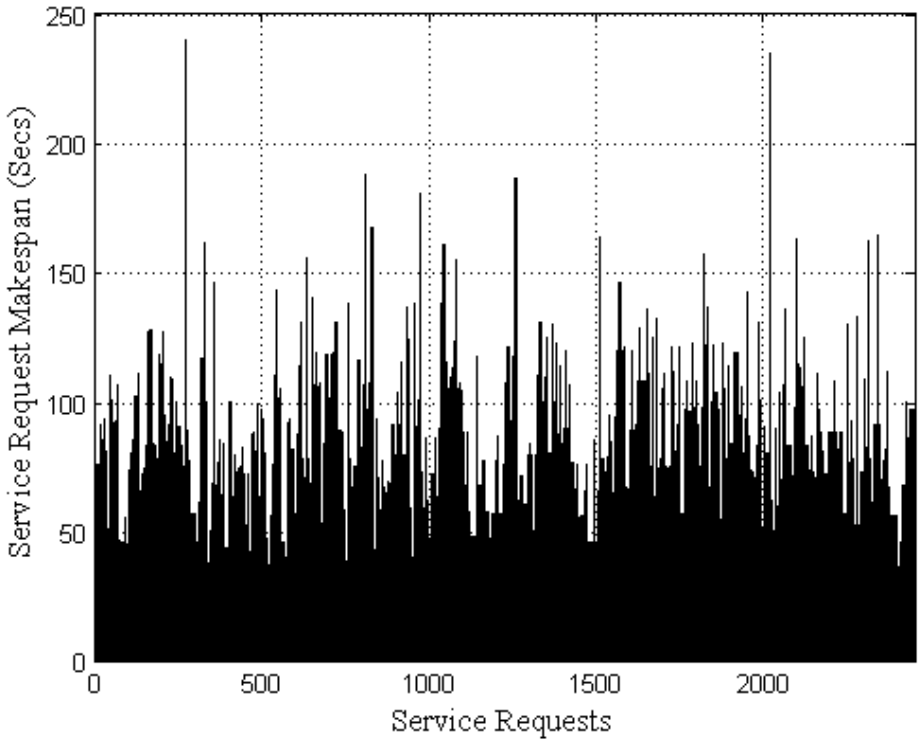

Figure 16

Service Oriented Computing and Applications - SOCA (2009) 3:109–125
The original publication is available at www.springerlink.com

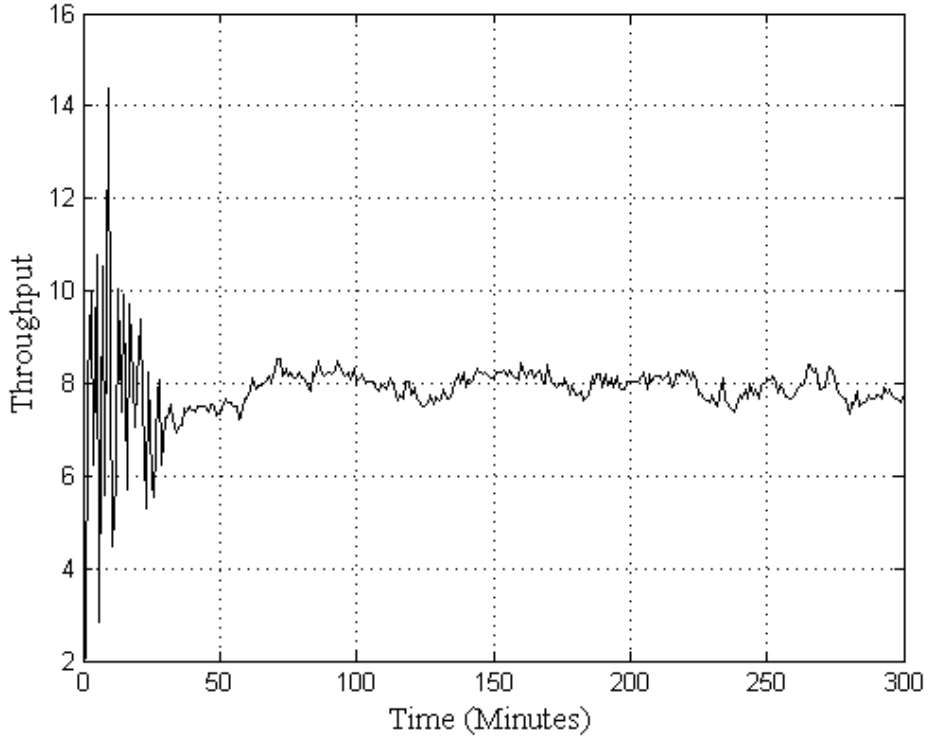

Figure 17

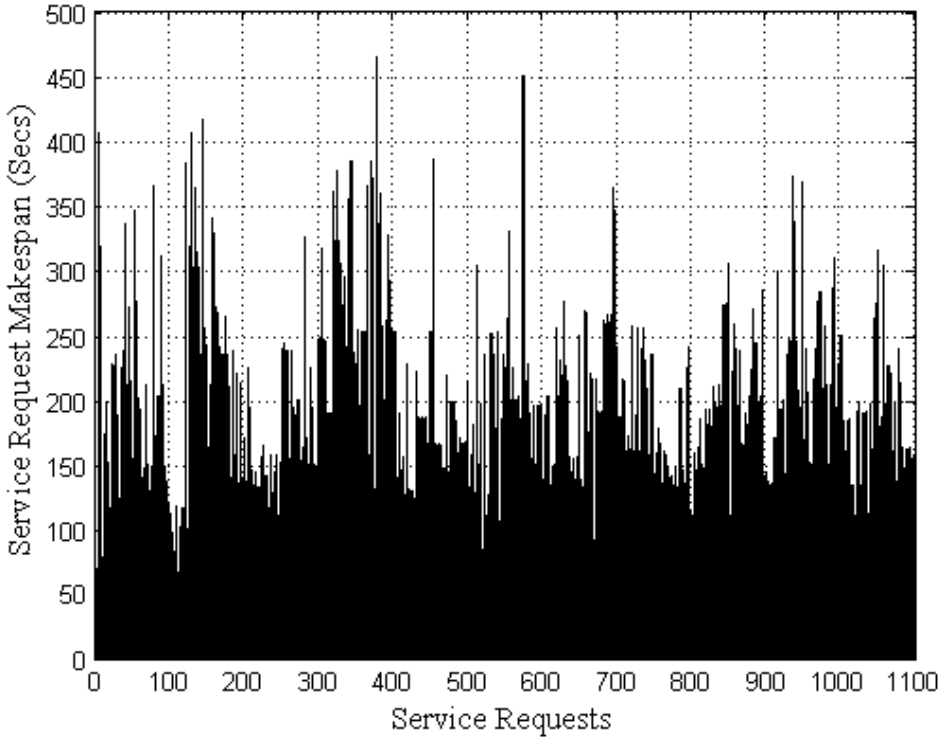

Figure 18

Service Oriented Computing and Applications - SOCA (2009) 3:109–125
The original publication is available at www.springerlink.com

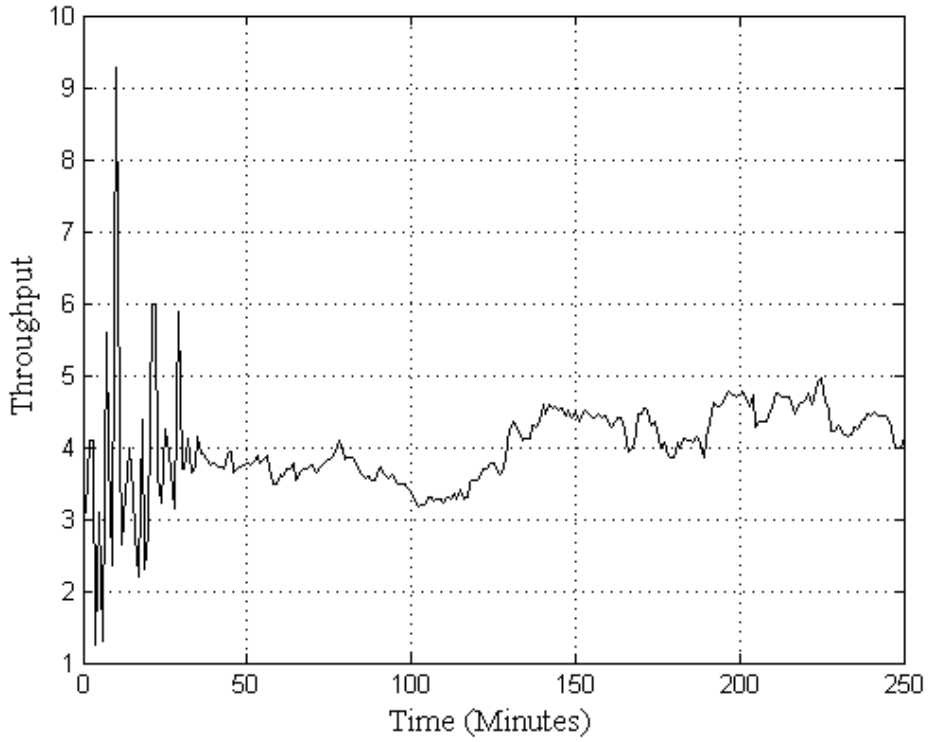

Figure 19