Enabling the Simulation of Service-Oriented Computing and Provisioning Policies for Autonomic Utility Grids

Marcos Dias de Assunção[1,2], Werner Streitberger[3], Torsten Eymann[3], and Rajkumar Buyya[1]

[1] **Gr**id **C**omputing and **D**istributed **S**ystems (GRIDS) Laboratory
[2] NICTA Victoria Research Laboratory
The University of Melbourne, Australia
{marcosd, raj}@csse.unimelb.edu.au

[3] Chair for Information Systems Management
University of Bayreuth, Germany
{streitberger, eymann}@uni-bayreuth.de

Abstract. There are key challenges in utility computing environments such as the provisioning, orchestration and allocation of resources to services. In these environments, providers need to decide how resources are allocated to service applications according to their workloads in order to guarantee the Quality of Service (QoS) required by customers. Autonomic computing inspired mechanisms are appealing to enable self-organising resource allocation and provisioning. However, these mechanisms are difficult to evaluate in practice either because of the lack of a real test bed or the difficulty in replicating experimental results. This work thus describes a service framework for a Grid simulator. This framework allows the modelling and evaluation of the provisioning and negotiation of services and resources. We also discuss experimental results that demonstrate the usefulness of this framework for the simulation of a decentralised and self-organising economic model for service and resource negotiation termed *Catallaxy*.

Keywords: Resource provisioning, Grid computing, utility computing, simulation framework.

1 Introduction

Service-Oriented Architectures (SOAs) underlie several Grid initiatives and reflect the current Grid computing infrastructure, where participants offer and request application services. A SOA defines standard interfaces and protocols that enable the encapsulation of resources of different complexity and value as services that clients access without having knowledge of their internal workings [1].

In current utility computing environments, resource providers host services and provide the tools needed by scientists and companies to expose the core functionalities of their research or business as services that are subsequently

used by clients or collaborators; providers offer their resources generally in a pay-as-you-go manner. Virtualisation technology offers powerful resource management mechanisms for these environments by enabling performance isolation, migration, suspension and resumption of Virtual Machines (VMs). One key issue, however, is the provisioning, orchestration and allocation of resources to services. Providers need to decide how resources are allocated to service applications according to their workloads in order to guarantee the QoS expected by their customers. Autonomic computing [2] inspired mechanisms and policies are appealing to enable self-organising allocation of resources to services, as well as for service provisioning and negotiation [3, 4].

However, it is challenging to design and evaluate practical allocation policies that permit utility computing environments to self-manage and adjust resource allocations according to the provisioning decisions of the offered services. Moreover, it is a challenge to evaluate these policies and negotiation strategies either due to the difficulty of replicating experiments or a lack of a real testbed.

The modelling and evaluation of these mechanisms and related policies can be augmented by the use of simulators. However, current simulation tools focus on issues related to resource modelling and allocation assuming in general a job abstraction. The existing Grid simulation toolkits do not provide the features needed to model and simulate services, their placement on resources, their workloads and provisioning policies let aside the abstraction of containers or VMs. In this work, we present a framework that allows the modelling, simulation and evaluation of mechanisms and policies for service provisioning, negotiation and resource management. This framework supports the simulation of service-oriented applications, and considers service dependencies, for different domains including high-performance, on-demand and utility computing. We demonstrate the usefulness of our framework by modelling and simulating an Application Layer Network (ALN) and an economic model termed *Catallaxy* for service and resource negotiation.

The rest of this paper is organised as follows. Section 2 presents background and related work. Section 3 describes the service framework. In Section 4, we present the design of a decentralised economic bargaining model for ALNs (i.e. the Catallaxy). Section 5 presents the performance evaluation results and finally, Section 6 concludes the paper.

2 Background and Related Work

In order to demonstrate the mechanisms and policies that we want to model and simulate, we consider a utility data centre that hosts service applications, and provides resources on demand to its customers' business applications (see Fig. 1) [5]. The centre is composed of a pool of physical resources that are managed by server virtualisation technology [6]. The services offered to customers run on Application Environments (AEs) within the resource pool, which are isolated from one another. An AE is a set of virtual resources (i.e. containers or VMs). The resource arbitrator allocates resources to each AE according to the resource allocation policies in order to meet the required performance and

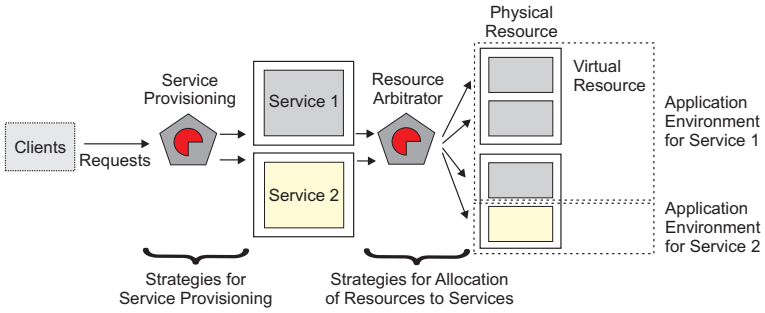


Fig. 1. Abstract view of a utility data centre.

QoS. Customers can utilise services without the knowledge of the internal infrastructure of the resource layer and the resource allocation policies. However, customers and providers negotiate the Quality of Service (QoS) required, and customers want to have guarantees about the service delivery. These guarantees are stated in Service Level Agreements (SLAs). Service provisioning policies define how the service is provided in order to achieve the service levels stated in the SLA. In this case, the provider has to decide on how the service is provisioned.

The services have a workload that can vary. The number of requests to the hosted services and the expected QoS will guide the arbitrator on the resource allocation decisions. The arbitrator decides on the resources required by each service and if new resources have to be allocated to meet peak demands. A decoupling of service and resource layers allows one to model strategies for the placement of services on resources and resource orchestration. One can also evaluate distinct markets or mechanisms for service negotiation and resource allocation. Therefore, a provider has two policies: one that defines how a service is provisioned and another that specifies how resources are allocated.

The scenario above is an example; however, a simulation framework should be flexible enough to enable the modelling and simulation of varying scenarios. For instance, the ALN presented in this work follows a two-layer market model. In one layer, resource providers provide processing and storage resources. Service providers negotiate with resource providers to acquire capacity to host services. The second layer corresponds to the negotiation between service providers for the delivery of composite services. For example, a service provider can negotiate the access to several atomic services in the service market to deliver it as a bundle, or composite service, to its customers. Similar scenarios are considered in other utility computing strategies [7].

2.1 Related Work

Several Grid simulators allow the modelling and simulation of Grid resources and allocation policies; examples include OptorSim [8], SimGrid [9] and MicroGrid [10]. OptorSim is a discrete event simulator that follows the abstraction of data resources. It has been designed to model and evaluate the data transfer strategies for data Grids, and does not provide a service-oriented application model.

MicroGrid enables the emulation of Grid environments. A user can run his Grid application on an emulated environment, while the simulator intercepts the exchanged messages. Although it is possible to simulate service-oriented applications, MicroGrid does not provide a decoupling of the service and resource layers, which would enable the design and evaluation of different mechanisms for each layer.

SimGrid is a trace-based event simulator that provides a set of abstractions and functionalities to build simulators for several application domains. The core features can be used to model and evaluate parallel application scheduling on distributed computing platforms. SimGrid also provides emulation facilities for running distributed and parallel applications in an emulated Grid environment. SimGrid like the the other simulators, uses the abstraction of 'resources'.

GridSim [11] is a Grid simulation toolkit that enables the modelling of application composition, information services, and heterogeneous computational resources of variable performance. GridSim also provides an auction framework for the design and evaluation of auction protocols for Grid systems. With these features, it is possible to model and evaluate the scheduling of jobs on Grid resources and the impact of varying allocation policies. Similar to other simulators, GridSim enables the design and modelling of the resource layer.

In this work, we leverage the existing features of GridSim and provide a service framework that enables the modelling and evaluation of service provisioning policies, resource allocation policies and multiple economic mechanisms for service negotiation and resource management. GridSim, along with the extensions described here, provides means for evaluating autonomic computing systems, utility computing environments and utility Grids.

3 A Service Framework for GridSim

GridSim [11] is a discrete event simulator built on top of SimJava2 simulation package. A simulation in GridSim comprises of GridSim entities that communicate with one another by scheduling simulation events. Applications are modelled as jobs that are executed on Grid resources. A *Gridlet* represents a job and has parameters like job length expressed in Millions of Instructions (MIs), amount of CPUs required, among others. It is possible to model Grid resources of varying configurations, where the processing capability of the resource's CPUs is expressed in Millions of Instructions Per Second (MIPS). GridSim provides default resource allocation policies (e.g. space-shared, time-shared and space-shared supporting advance reservations), but the user can develop his own.

GridSim provides a hierarchical Grid Information Service (GIS) that can comprise of multiple regional GISs. At the start of the simulation, a Grid resource registers itself with a regional GIS. By default the Grid resource registers only its resource ID and indicates whether it supports advance reservation; however, the user can specify additional information to be provided to the GIS.

Based on the utility computing scenario described in Section 2, we design the framework considering two distinct stages: (i) the negotiation for and allocation of the resources to host services, and the negotiation for services and the

required QoS; and (ii) the actual utilisation of the services and resources. The framework provides means for modelling service registries and discovery, service and resource negotiation as well as means for measuring the resource utilisation imposed by the services' workloads. A provider in this scenario has two policies: one that defines how a service is provisioned and one that defines how resources are allocated to a service. The allocations may change according to the service workloads.

Fig. 2 demonstrates the relationship between the main classes of the framework. The class *Provider* is a GridSim entity that implements the basic behaviour of a provider. A provider has characteristics represented by *ProviderCharacteristics*. The class *ProviderCharacteristics* contains a list of *Services* offered by the provider and other attributes like time zone, and the provisioning and acquisition policies utilised. *Service* corresponds to a service offered by the provider and has *ServiceAttributes* and a *ServiceRequirementList*. At the start of the simulation, the provider registers itself and the attributes of her services with a regional Grid Service Registry (GSR). *ServiceAttributes* include information like service cost, name and type. We implement service attributes as a distinct class for the sake of performance and minimisation of simulation events. *ServiceRequirements* correspond to atomic services or specific resources required to deliver the service to clients. For example, a provider may offer a service, but does not allocate resources to it until the service is required.

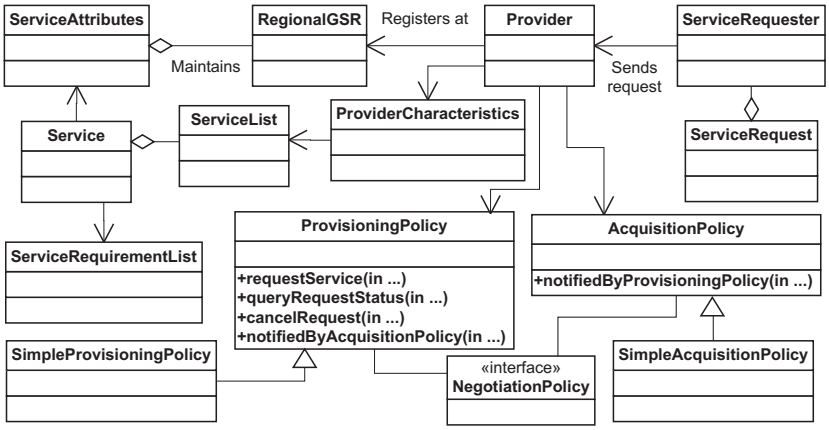


Fig. 2. Main classes of the framework.

The Provider can engage in a market with clients for negotiating its resources. It can also participate in different markets with different mechanisms for negotiating and providing the resources necessary to host the services and satisfy the requests for a service. Both *ProvisioningPolicy* and *AcquisitionPolicy* implement the *NegotiationPolicy* interface. *NegotiationPolicy* defines the methods necessary to handle negotiations for service provisioning or resource allocation based on WS-Agreement. *ProvisioningPolicy* defines how a service is provisioned while the *AcquisitionPolicy* specifies the resource allocation. In other words, the *ProvisioningPolicy* defines how the provider manages the negotiation with clients

for service provisioning and how it handles the resource requests. *AcquisitionPolicy* specifies the provider's behaviour in negotiating with other providers for accessing the required services or resources. These services may be needed for composite services and the resources are required to host service applications. In many instances, provisioning and acquisition policies have to be synchronised or informed about one another decisions, as demonstrated by Grit *et al.* [12]. We provide methods that allow the policies to be synchronised.

Two examples of provisioning and acquisition policies are provided. In *SimpleProvisioningPolicy*, the provider accepts requests while the maximum number of instances for the service is not achieved. *SimpleAcquisitionPolicy* selects the first resource from the provider's resource pool to deal with the workload generated by the service requests. Although the Provider class can be extended, it is not necessary since it is possible to define different behaviours for a provider by extending the *ProvisioningPolicy* and *AcquisitionPolicy* classes to provide the strategies required.

The *ServiceRequester* class is a GridSim entity that queries services at a GSR and makes requests to providers. These queries can be performed by passing a filter to the GSR, which corresponds to specifying the parameters for a query. For example, the service requester can pass an object whose class extends *ServiceFilter* to select all the *ServiceAttributes* with a given service type and name. The GSR uses the filter to select and return a list of *ServiceAttributes* that match the given criteria.

A request for a service accepted by a provider generates a workload. The workload is composed of items that can be either requests for atomic services or *ServiceGridlets* that are sent to the resources allocated to the service. The *ServiceGridlet* class extends *Gridlet* by specifying additional parameters such as memory and storage required to fulfil the request. The values of these parameters for a service request can be estimated through profiling techniques, such as those described by Urgaonkar *et al.* [13], where a service application is examined in isolation and its workload is obtained by analysing the use of resources such as memory, CPU and disk. By following this model, it is possible to analyse the impact of different provisioning and acquisition decisions on resource utilisation.

4 Modelling the Catallaxy Scenario with GridSim

The CATNETS project investigates the use of an economic model, termed Catallaxy, for service negotiation and resource allocation in ALNs, such as Grids and P2P networks. Catallaxy is a decentralised self-organising economic model derived from Hayek's concept of spontaneous order [14]. The Catallaxy is based on the self-interested actions of participants, who try to maximise their own utility under incomplete information and bounded rationality. The goal of Catallaxy is to achieve a state of coordinated actions, through the bartering and communication of participants, to achieve a common goal that no single user has planned. Hayek's Catallaxy is the result of descriptive and qualitative research about economic decision-making of human participants. Its results are taken to construct ALN markets with software participants who reason about economic decisions using artificial intelligence.

The interdependencies between services and resources existing in ALNs are separated by creating two interrelated markets: a resource market for trading of computational and data resources; and a service market for trading services. This separation allows instances of a service to be hosted on different resources [15]. Fig. 3 shows the abstract model adopted by CATNETS. A Complex Service (CS) is a composite service, like a workflow, that requires the execution of other interdependent services, termed Basic Services (BSs). A CS is the entry point for the application layer network. The traded products on the service market, the BSs, are completely standardised and have a single attribute name. The name is a unique identifier whose intended semantics is shared among all complex service providers. Multiple instances of the same BS can co-exist in the ALN. For example, two or more basic service providers are allowed to provide a specific BS. The service market is used by Complex Service Providers (CSPs) to allocate

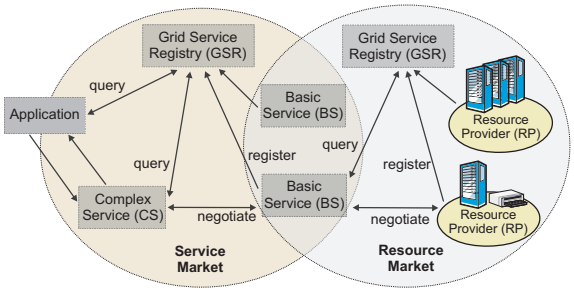


Fig. 3. The Catallaxy market model.

BSs from Basic Service Providers (BSPs). BSPs are registered in a GSR. A CSP queries a GSR to receive a list of required trading partners (BSPs) able to provide the BS required. This list is ranked according to the BS offered price. The best BS offer is selected for the succeeding bargaining process. This discovery process is modelled using GSRs and discovery process offered by the simulation framework.

After a successful negotiation in the service market, BSPs negotiate with Resource Providers (RPs) for the resources necessary to host services and serve the service requests. RPs utilise the existing resource management systems to allocate the necessary resources. RP offer resources in Resource Bundles (RBs). A resource bundle is described by a set of pairs of resource type and quantity. Every BS has an associated resource bundle. The bundle defines the type and quantity of resources needed for provisioning that service. In the CATNETS scenario, the resource bundle required for a BS is predefined for the sake of simplicity. In general, the model allows the use of any BS to resource bundle mapping function. In the resource market, the allocation process follows the service market. First, a BSP queries for RPs which are able to provide the specified resource bundle and ranks the received list of RPs according to the offered price. Second, the bargaining for the resource bundle is carried out. If the resource negotiation ends successfully, the BS is executed on the contracted resources from a RP.

To realise these two markets in GridSim, we implement provisioning and acquisition policies for the three kinds of providers (i.e. CSPs, BSPs and RPs). The

providers differ in terms of the policies used for service and resource provisioning and acquisition. The execution of a market participant's policy for acquiring services or resources (i.e. *AcquisitionPolicy*) is shown in Algorithm. 1 and that of a market participant's policy for service provisioning (i.e. *ProvisioningPolicy*) is depicted in Algorithm 2. The most important part of the implemented

Algorithm 1 Pseudo-code of the execution of an acquisition policy.

```
1: loop
2:    event ← wait for an event
3:    if event = message from provisioning policy then
4:       proposals ← ∅
5:       request_accepted ← the request ∈ event
6:       CFP ← create Call For Proposal (CFP) for request_accepted
7:       send CFP
8:       proposals ← collect the proposals
9:       best ← select best proposal ∈ proposals
10:      start bargaining process
11:      outcome ← result of the bargaining process
12:      if outcome = success then
13:         inform other participants about the success
14:      end if
15:      apply learning algorithm
16:      notify provisioning policy about outcome
17:   end if
18:   if event = learning message then
19:      treat message received
20:      apply learning algorithm
21:   end if
22: end loop
```

Algorithm 2 Pseudo-code of a provisioning policy.

```
1: loop
2:    event ← wait for an event
3:    if event = Call For Proposal then
4:       CFP ← get the Call For Proposal (CFP) ∈ event
5:       proposal ← formulate proposal for CFP
6:       reserve the resources
7:       send proposal
8:    end if
9:    if event = bargaining then
10:      start bargaining process
11:      outcome ← result of bargaining process
12:      if outcome = success then
13:         notify acquisition policy
14:         inform other participants about the success
15:      else
16:         release resources
17:      end if
18:      apply learning algorithm
19:   end if
20:   if event = reject proposal then
21:      release the resources
22:   end if
23:   if event = learning message then
24:      treat message received
25:      apply learning algorithm
26:   end if
27: end loop
```

policies is the utilised bidding strategy. This includes what a provider bids. The bid denotes the provider's valuation and reservation prices, i.e. the maximum price which an agent is willing to pay for the service and the minimum price an

agent has for selling a BS or a RB respectively. The generation of the valuation is influenced by external factors such as the market price and the learning algorithm. For the formal model of the implemented strategy we refer to the work by Reinicke *et al.* [16].

The proposed realisation for the CATNETS markets is the usage of a bilateral negotiation protocol for exchanging bids in a point-to-point communication. Initially, both trading partners define a reservation price that reflects their estimation of the value of the good. For a buyer, this is the maximum price; for a seller it is a minimum price. The start price represents the negotiation starting point. By subsequent concessions, the opponents move closer to a compromise and a possible contract. Each opponent tries to maximise its own utility, which is the difference between the price of purchase and the reservation price. Thus, the buyer's and seller's policies converge to a trade-off point in an iterative way using the exchange of offers and counter-offers and successive concessions.

In the implementation in GridSim, CSs and BSs are modelled as *Services*. The service requirements of a CS define the BSs that are needed to deliver the CS. The service requirements of a BS define a minimum RB required to host the BS. The requirements of a BS j are represented by $BSR_j = (u_j, p_j, y_j, m_j, s_j)$, where u_j is the number of resources required; p_j represents the number of CPUs in each resource; y_j is the speed of the processors in MIPS; m_j is the amount of memory per resource; and s_j represents the storage capacity required.

A RP has a resource pool within which it creates Application Environments (AEs) with the resource configuration required by a BS. A RB corresponds to the resources offered by the RP. A RB i is represented by $RB_i = (u_i, p_i, y_i, m_i, s_i)$, where u_i is the number of resources in the bundle; p_i represents the number of CPUs in each resource; y_i is the speed of the processors in MIPS; m_i is the amount of memory per resource; and s_i represents the storage capacity per resource. A RP registers the RB with the GSR, which is viewed as a service by the BSP. That is, the RP provides a service that enables the BSP to acquire resources.

The negotiation for the resources needed by the BS starts after the negotiation for a BS is complete. The BSP searches for RPs that can provide a RB with the minimum amount of resources required. The BSP then starts the negotiation by sending a Call For Proposal (CFP) to the selected RPs. The BSP bargains with the RP that offers the best proposal. When the bargaining process ends, the RP allocates its resources to host the BS. Although a RP can divide its resource pool in various ways and change the allocations of AEs over time, in the CATNETS implementation, we consider that they are pre-determined and do not change. The strategy used by a RP when it receives a CFP from a BSP during the negotiation of resources for a BS is summarised in Algorithm 3.

5 Performance Evaluation

We present experimental results that demonstrate how GridSim with the extensions discussed in this work can be used to model and evaluate service provisioning and resource allocation policies for service-oriented Grids and autonomic

Algorithm 3 RP's strategy upon the arrival of a Call For Proposal (CFP) j.

```
1: BSR_j ← get required resource bundle from the CFP_j
2: RB_i ← the resource bundle advertised
3: selected_resources ← ∅
4: booking_id ← 0
5: for all resource R_i ∈ RB_i do
6:    if R_i is not allocated then
7:       if p_j ≤ p_i and y_j ≤ y_i and m_j ≤ m_i and s_j ≤ s_i then
8:          selected_resources ← selected_resources ⋃ R_i
9:       end if
10:   end if
11:   if selected_resources = u_j then
12:      booking_id ← book(selected_resources)
13:      break for
14:   end if
15: end for
16: if booking_id = 0 then
17:    proposal ← create_proposal(selected_resources)
18:    send proposal
19: else
20:    reject CFP_j
21: end if
```

utility computing environments. The experiments particularly measure how the Catallaxy model, built on top of the discussed framework, coordinates the use of services and resources. We evaluate the allocation rate by identifying the number of service requests that are satisfied and the overhead imposed by the service and resource negotiations.

5.1 Experimental Scenario

We consider an environment in which RPs provide resource bundles and BSs require a particular resource bundle for a given time slot to host the service and execute the service workload. The experiments have been carried out considering a CS termed Workflow Service (WFS) that requires two BSs, namely Processing Service (PS) and Storage Service (SS). These two BSs, in turn, require a Processing Bundle (PB) and a Storage Bundle (SB) respectively. PB has the following configuration: ($p = 2, y = 1500MIPS, m = 1GB$ and $s = 2GB$), while SB is given by: ($p = 1, y = 1500MIPS, m = 2GB$ and $s = 4GB$).

We perform our experiments with varying numbers of RPs, BSPs and CSPs. The parameters used in the experiments are shown in Table 1. The values for PS Request Length (PSRL) and SS Request Length (SSRL) are given by $WSRL/2$ because we consider that WFS first requires processing and further stores the results of the processing activity. For simulating the workload of PS and SS and obtaining the final time of the service utilisation, we consider a simple approach. For example, the workload generated by an invocation j of PS at RP i is given in MIs by: $p_j * y_j * PSRL_j$ where p_j is the number of processors required by the PS, y_j is the processor speed in MIPS and $PSRL_j$ is PS request length.

Table 1 summarises the experiments performed and the values used for the simulation of the service application in GridSim using the Catallaxy economic model and the presented service framework. The parameters TBWS, WSRL, INSIZE and OUTSIZE use uniform distributions. In addition, we consider that the BSPs are able to provide and negotiate for one BS at a time.

Table 1. Description of the parameters Used in the Experiments.

Parameter Description	Acronym	Exp. 1	Exp. 2	Exp. 3	Exp. 4
Number of Providers of Workflow Services		10	20	50	50
Number of Providers of Processing Basic Services		10	20	50	50
Number of Providers of Storage Basic Services		10	20	50	50
Number of Providers of Processing Resource Bundles		10	20	50	20
Number of Providers of Storage Resource Bundles		10	20	50	20
Number of Service Instances Per WFS Provider	SI	40			
Number of Resource Bundles Per Resource Provider	RU	1			
Number of Requests to Workflow Service	WSR	1000			
Time between arrivals of WFS requests	TBWS	0-120s			
WFS Request Length	WSRL	30-60s			
PS Request Length	PSRL	WSRL/2			
SS Request Length	SSRL	WSRL/2			
Input File Size	INSIZE	30-50KB			
Output File Size	OUTSIZE	100-200KB			

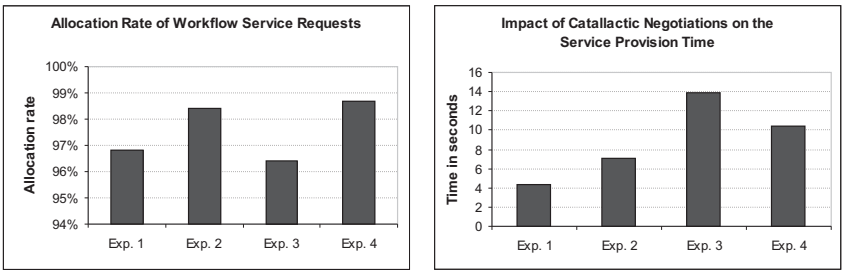
5.2 Experimental Results

Fig. 4(a) shows the allocation rate of workflow service requests in the different experiments. The allocation rate is above 96% in all experiments. However, in Experiment 3 the allocation rate is lower than in Experiment 4, even though more resource providers are available. The reason for such behaviour is that a provider reserves its services or resources when it receives a CFP. Once an announcement is sent by the provider who initiated the negotiation, the providers that have not been selected release their services or resources. As the number of providers increase, more messages are sent, the negotiations take more time and the resources are kept reserved for a longer time. In Experiment 4, we reduce the number of resource providers and the allocation rate increases.

We then evaluate the impact of the negotiations on the service provisioning process. The experiments measure the amount of time spent on negotiation for a BS. Fig. 4(b) shows the time spent in different scenarios. We observe that the time spent is highly dependent on the initial timeout during which the negotiator waits for proposals, which in this case is 30 seconds (15 seconds in negotiation for the BS and 15 seconds in negotiation for the resource). We thus omit this 30 second interval from the results presented in the figure. In the scenarios evaluated, we consider that users and service providers are in different networks connected through a network link with a bandwidth of 1Mbps while service providers and resource providers are connected through another network link with a bandwidth of 1Mbps. Both links present a latency of 50 milliseconds, which we consider to be representative of the latency in many wide area networks. The time required to send proposals and to bargain to achieve the final price is generally smaller than 10 seconds. The initial timeout can be reduced if the initial negotiator knows how many providers have been contacted and how many messages should be received. However, we envision a scenario in which a P2P network is used to broadcast calls for proposals and the negotiator does not know exactly how many providers will receive the proposals and send a reply.

6 Conclusion and Future Work

This paper describes a model for the simulation of service-oriented Grid applications to allow the decoupling of service negotiation and resource management



(a) **(b)**

Fig. 4. (a)Allocation rate of WFS requests. (b)Time taken for a BS negotiation.

into two distinct layers. By decoupling these, it is possible to model and evaluate different strategies for both service provisioning and resource allocation. The model also enables the simulation and evaluation of policies for negotiation of SLAs for service usage and the evaluation of centralised and decentralised economic models. We present experimental results that demonstrate the use of the framework for modelling and evaluation of a decentralised economic bargaining mechanism, the Catallaxy, for service and resource negotiation.

For future work, we would like to evaluate the suitability of the framework for modelling large-scale scenarios and improve the acquisition policies to support advance reservation and co-allocation of Grid resources. In addition, we would like to evaluate the economic models considering dynamic environments with varying failure probabilities for resources. We will consider acquiring data from existing Grid test beds for determining the failure probability of Grid resources and include these in the Grid simulator.

In addition, we would like to incorporate models for what can be called elastic containers or elastic VMs. In these models, the allocation policy of a utility data centre, for instance, may decide to expand the amount of memory, storage and CPU of VMs in an AE according to the service workload. We would like to incorporate these VM models and enable the changes in the configurations of VMs on the fly. These features can enable the evaluation of varying provisioning policies.

Acknowledgments

We thank Chee Shin Yeo, Krishna Nadiminti, James Broberg and Al-Mukaddim Khan Pathan from the University of Melbourne for their assistance in improving the quality of this paper and for sharing their thoughts on the topic. This work is supported by the European Union, DEST and ARC Project grants. Marcos' PhD research is partially supported by National ICT Australia (NICTA).

References

1. Foster, I.: Service-oriented science. Science **308**(5723) (2005) 814–817
2. Kephart, J.O., Chess, D.M.: The vision of autonomic computing. Computer **36**(1) (2003) 41–50

3. Almeida, J., Almeida, V., Ardagna, D., Francalanci, C., Trubian, M.: Resource management in the autonomic service-oriented architecture. In: 3rd IEEE International Conference on Autonomic Computing (ICAC 2006), Dublin, Ireland (2006) 84–92
4. Bennani, M.N., Menascé, D.A.: Resource allocation for autonomic data centers using analytic performance models. In: 2nd IEEE International Conference on Autonomic Computing (ICAC 2005), Seattle, Washington (2005) 229–240
5. Walsh, W.E., Tesauro, G., Kephart, J.O., Das, R.: Utility functions in autonomic systems. In: International Conference on Autonomic Computing (ICAC 2004), New York, NY (2004) 70–77
6. Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., Warfield, A.: Xen and the art of virtualization. In: 19th ACM Symposium on Operating Systems Principles (SOSP '03), New York, NY, USA, ACM Press (2003) 164–177
7. Low, C., Byde, A.: Market-based approaches to utility computing. Technical Report HPL-2006-23, Internet Systems and Storage Laboratory, Hewlett Packard Laboratories Bristol (2006)
8. Bell, W.H., Cameron, D.G., Capozza, L., Millar, A.P., Stockinger, K., Zini, F.: Simulation of dynamic grid replication strategies in optorsim. In: 3rd International Workshop on Grid Computing (GRID 2002), London, UK, Springer-Verlag (2002) 46–57
9. Casanova, H.: Simgrid: A toolkit for the simulation of application scheduling. In: 1st IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2001), Brisbane, Australia (2001) 430–437
10. Song, H., Liu, X., Jakobsen, D., Bhagwan, R., Zhang, X., Taura, K., Chien, A.: The microgrid: a scientific tool for modeling computational grids. In: ACM/IEEE Supercomputing 2000 Conference (SC'00). (2000) 53–53
11. Buyya, R., Murshed, M.: Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. Concurrency and Computation: Practice and Experience (CPE) **14**(13-15) (2002) 11751220
12. Grit, L., Inwin, D., Yumerefendi, A., Chase, J.: Virtual machine hosting for networked clusters: Building the foundations for 'autonomic' orchestration. In: 1st International Workshop on Virtualization Technology in Distributed Computing (VTDC 2006), Tampa, Florida (2006)
13. Urgaonkar, B., Roscoe, P.S.T.: Resource overbooking and application profiling in shared hosting platforms. In: 5th Symposium on Operating Systems Design and Implementation, Boston, Massachusetts (2002) 239–254
14. Hayek, F.A.V.: The Collected Works of F.A. Hayek. University of Chicago Press (1989)
15. Eymann, T., Ardaiz, O., Catalano, M., Chacin, P., Chao, I., Freitag, F., Gallegati, M., Giulioni, G., Joita, L., Navarro, L., Neumann, D.G., Rana, O., Reinicke, M., Schiaffino, R.C., Schnizler, B., Streitberger, W., Veit, D., Zini, F.: Catallaxy-based grid markets. International Journal on Multiagent and Grid Systems, Special Issue on Smart Grid Technologies & Market Models **1**(4) (2005) 297–307
16. Reinicke, M., Streitberger, W., Eymann, T.: Scalability analysis of matchmakers in self-optimizing computing networks. Journal of Autonomic and Trusted Computing (JoATC) (2005)