Grid Information Services using Software Agents

H.N. Lim Choi Keung * J. Cao D.P. Spooner
S.A. Jarvis G.R. Nudd

Abstract

Computational grids allow large-scale, pervasive and consistent sharing of geographically dispersed resources. Their inherent nature incorporates issues including the discovery of resources located in different administrative domains, predicting the performance of those resources and monitoring their behaviour. The Monitoring and Discovery Service (MDS), one of the pillars provided by the Globus toolkit, can be used to offer Grid information services to an existing agent-based resource advertisement and discovery system. This paper presents an agent system which implements the GRid Information Protocol (GRIP) and the GRid Registration Protocol (GRRP) of the MDS to discover virtual organisations and monitor their respective resources. The resulting system has the effect of resource brokering, monitoring and performance prediction.

1 Introduction

Grid computing enables wide-spread sharing and coordinated use of geographically dispersed, networked resources[10]. Such sharing may be short- or long-term, and may consist of heterogeneous resources, thereby creating continuously dynamic virtual organisations. Consequently, users would have no knowledge of the resources which are participating in the virtual organisation at any one time. Grid Information Services, for example the Globus Monitoring and Discovery Service (MDS), are designed to discover and monitor the existence and behaviour of resources, computations and services, and other entities which form part of a Grid[9].

Grid computing defines the scalability of a large number of networked resources. One issue that this raises is that of resource contention, which itself will also affect application performance. It is therefore important that the effects of resource contention are carefully managed, for example through resource monitoring and scheduling.

*High Performance Systems Group, Dept. of Computer Science, University of Warwick, Coventry, UK. E-mail : {hlck, junwei, dps, saj, grn}@dcs.warwick.ac.uk

1.1 Grid Resource Management

1.1.1 The A4 Agent System

The Agile Architecture and Autonomous Agent system (A4)[3, 6] addresses the general problem of resource management using an agent-based implementation where agents cooperate to discover available resources. This process is termed *service advertisement and discovery*. Every agent has knowledge about its neighbouring agents which process one another's service advertisement and discovery requests[7].

In A4, a hierarchy of agents is used to provide wide-area resource sharing. The agents are homogeneous and consist of a number of functional layers - see Figure 1.

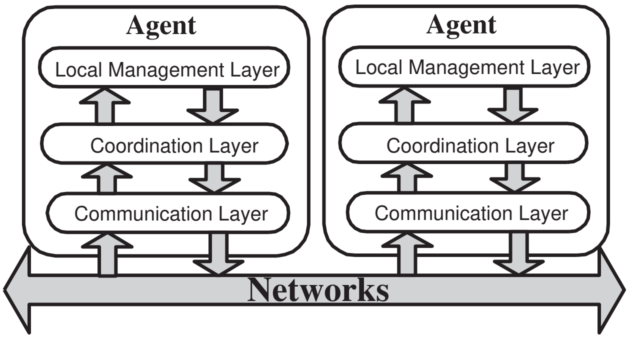

Figure 1: The basic structure of an agent consists of three interacting layers. For more details see [6].

- **Communication Layer** Agents use this layer to communicate with one another using common data models and communication protocols. An Agent Communication Language (ACL) can also be used by agents to exchange knowledge with one another.
- **Coordination Layer** This layer decides how the agent should act on the data at the communication layer according to its own knowledge.
- **Local Management Layer** This layer encapsulates the functions needed for the management of local services. It also provides local service information needed by the coordination layer [5].

This wide-area agent system has been integrated with a local-area grid task scheduler known as Titan.

1.1.2 Titan Local Resource Manager

Titan[16] is a local-area workload management system used to select suitable resources for a particular task, given a varied, dynamic resource pool. The search space for the multi-parameter scheduling problem is large and not fully defined until runtime. Consequently, a *just-in-time* approach to performance prediction is adopted so that runtime variables and resource load can be used to assist task and resource allocation while maintaining prescribed service contracts.

An iterative, heuristic algorithm forms the basis of each local scheduler. This algorithm aims to minimise processor idle time and makespan, which is the expected completion time of the last scheduled job. The algorithm is written in such a way as to allow changes to be absorbed; these include the addition or deletion of tasks, and changes in physical resources including the number of hosts or processors. The approach used by Titan is to generate a set of schedules and to evaluate the schedules to obtain a measure of fitness. It then selects the most appropriate and combines them using operators(crossover and mutation) to formulate a new set of solutions. This process is repeated, resulting in a *fittest* solution. An important aspect of the algorithm is the use of predictive performance data from the PACE toolkit that forms the basis for its scheduling decisions.

1.1.3 Performance Prediction with PACE

The Performance Analysis and Characterisation Environment (PACE) is an important component of the agent-based system because of its performance prediction capabilities.

PACE is a dynamic performance prediction modelling toolset used by high performance distributed applications. Some of the tools that it contains are for model definition, model creation, evaluation and performance analysis. It uses associative objects organised in a layered framework as a basis for representing each of a system's components. Moreover, the dynamic instrumentation technique used by PACE allows the automatic analysis of applications and hence, the production of reliable prediction results. These results are then used to dynamically steer the execution of these applications[1]. PACE also encompasses the performance aspects of application software, its resource use and mapping, and the performance characteristics of hardware systems[2]. The main components of PACE are shown in Figure 2; these are:

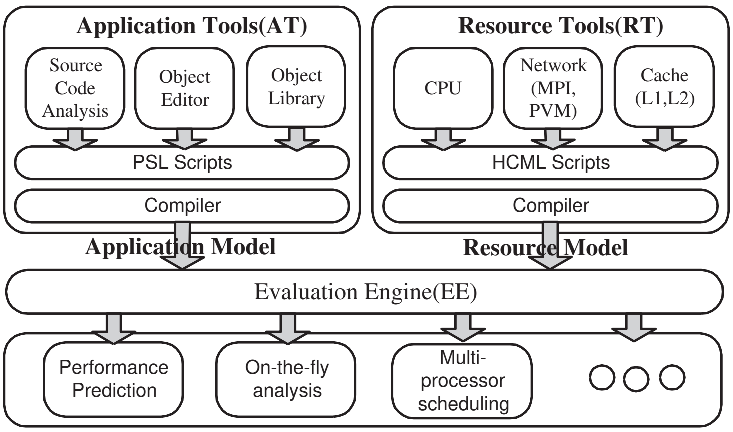

Figure 2: Components of the PACE Toolkit.

- **Application Tools** The performance characteristics of an application and its parallelisation are described using the toolkit's performance specification language(PSL). The Source Code Analyser converts sequential source code components into performance descriptions. These descriptions are edited using the Object Editor and the Object Library holds existing objects.

- **Resource Tools** A PACE hardware modelling and configuration language (HMCL) is used to define the computing environment in terms of its constituent performance model components.

- **Evaluation Engine** This forms the core of the PACE toolkit. It executes completed performance models to produce evaluation results. These include time estimates and trace information relating to the expected application behaviour.

PACE is also used for dynamic multi-processor scheduling and for efficient resource management[4]. Furthermore, it provides realistic performance predictions of expected application execution. The PACE toolset is comprehensive in its approach and is used in many different application areas[12, 13].

PACE is based on a layered characterisation methodology and is an analytical-based approach. It also supports the entire software lifecycle including development, execution and post-mortem performance analysis[11, 14].

1.1.4 Grid Information Management using Software Agents

Although the local scheduler has information about the resources in its local environment, it has no knowledge about resources in other local scheduling environments or in other administrative domains. As agents discover resources or receive service advertisements from neighbouring agents, they have to store this information. While each agent has enough information to propagate a task to its 'best suited' neighbour, there is no 'global' information repository which agents can access on demand. Service advertisement in the agent hierarchy is currently only carried out with nearby, neighbouring agents. Therefore, an agent advertises its service information to its upper or lower agent only. In this case, resource information is propagated to a wider infrastructure only after many iterations of service advertisement and discovery, which can take a long time. There is also the danger of a large amount of resource information duplication. The use of the MDS will overcome these problems.

The movement of service discovery requests from one agent to another will occur in the same way within a virtual organisation or across virtual organisations. This method therefore allows the agent hierarchy to be scalable. In this paper, it will be shown how agents implement the GRIP and GRRP protocols to pull information from resources and push information into aggregate directories. The paper also demonstrates how such agents can implement an automatic referral mechanism to discover other aggregate directories and hence resources in other virtual organisations.

This paper presents a way in which the Grid Information Services can be used in the agent-based resource management system to discover and monitor resources within large-scale Grid systems. In this context, a *virtual organisation(VO)* is defined as being a set of resources which, collectively, are within the same administrative domain. This paper is organised as follows: in Section 2, the Grid Information Services components and protocols which will be used are explained. Section 3 introduces the implementation of Grid Information

Services with software agents. Conclusions and future work are presented in Section 4.

2 Grid Information Services

Grid Information Services are a collection of services carrying out resource discovery and monitoring. Resource characteristics can be classified as static and dynamic. Static characteristics include the number of processors, the host model, machine architecture and the operating system version. Dynamic characteristics include CPU availability, CPU load, amount of memory, available processors and network load.

The Globus Monitoring and Discovery Service uses the LDAP model[15] and represents information in a hierarchical way, resulting in a Directory Information Tree (DIT). The MDS architecture consists of two basic entities:

1. Individual configurable information providers called Grid Resource Information Service (GRIS), which adhere to LDAP and provide information about individual entities. An entity is characterised by a set of 'objects' comprised of typed attribute-value pairs.

2. A higher-level, configurable aggregate directory component called a Grid Index Information Service (GIIS). This component collects, manages and indexes information provided by one or more information providers. The aggregate directory services can implement both generic and specialised views and can also provide searching functions. Thus, resources would register with a GIIS which would then pull information from them when requested by a client or when its cache has expired.

2.1 MDS Protocols

Each information provider implements two basic protocols: the GRid Information Protocol (GRIP) which enquires about the structure and state of a resource or service, and the GRid Resource Registration Protocol (GRRP) which allows a resource to both register with another entity and to notify the latter of its availability. The protocol also specifies how to contact the entity for the purposes of enquiry or control.

GRIS adheres to the GRIP protocol, and GIIS to the GRRP[8].

2.2 GRIS

This information provider framework is implemented as an OpenLDAP server backend which is customisable using plug-ins from specific information sources; this is shown in Figure 3. Considering the Grid environment, each resource under local scheduler management can run a local GRIS. A GRIS can service requests for specific resources, but in this work, a GRIS is configured to register itself with an aggregate directory service (GIIS) via the local scheduler, so that information can be passed onto other agents. The GIIS is explained in more detail in Section 2.3.

Typically, a local resource manager would send out an information request to each GRIS under its logical management on a periodic basis (for example, every minute). The GRIS on each resource would then authenticate and parse the incoming information request, and then dispatch the request to be handled by a local information provider. The local resource manager then merges the results from each of its GRIS resources and pushes them to the aggregate directory service.

The communication between a GRIS and an information provider takes place over a well-defined API. The local resource manager also caches each information provider's results for a specified period of time. Thus, the number of provider invocations is greatly reduced, the response time is improved and deployment capability is maximised. This configurable length of time is the cache's time-to-live (TTL), which is specified for each information provider at configuration.

The information providers return static host information including the number of processors, the CPU model, the operating system version and the architecture type. Dynamic host information is also pulled, including the load average, CPU availability, the number of available processors, storage information and network information. The local resource manager then pushes the information to the agent's GIIS, when requested.

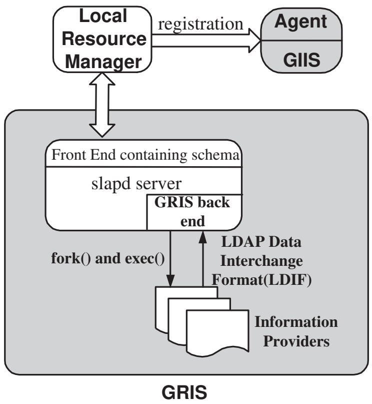

Figure 3: Within the GRIS, the server front end contains an LDAP schema for the providers. The GRIS back end is within the LDAP server and it forks processes and execs provider programs which then return LDIF data to the backend.

2.3 GIIS

The MDS also provides a framework for constructing aggregate directories called Grid Index Information Services (GIIS). GRRP messages are passed from 'child' GRIS to the directory to form a unified information repository. The three major components making up the GIIS framework are: 1. Generic GRRP handling 2. Pluggable index construction 3. Pluggable search handling.

A preliminary analysis of the above systems indicates that the A4 agents and the MDS architecture share two major features: a hierarchy and an information storage capability. Further work demonstrates that the agents' structure can be mapped onto the MDS, resulting in the integration of both architectures. The following describes how this mapping and integration are done.

Figure 4 shows an overview of the hierarchy of agent-based Grid information services components. Each agent interfaces with a GIIS which has information

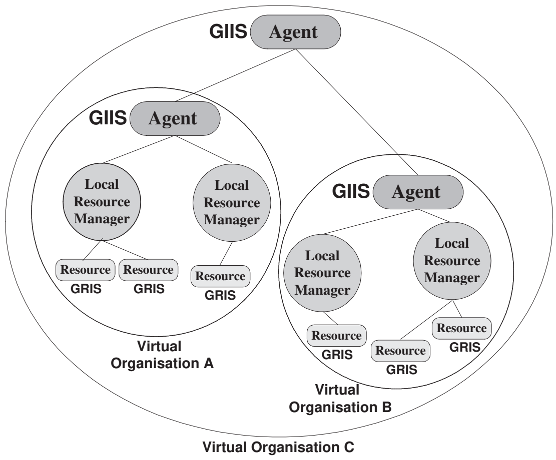

Figure 4: Agent-based Information Services Structure with a Hierarchy of Virtual Organisations.

about all the resources in its administrative domain, including all its local resource managers. The *scope* of information for each agent spans downwards towards all the other agents which have registered their resources with the former agent.

Therefore, homogeneous agents which communicate with one another, are arranged in a tree-like hierarchy. To be able to act as a high-level broker to the underlying metasystem and to carry out performance prediction, those agents need to have accurate resource information. The Globus MDS provides the information required for that purpose. Every agent will be closely integrated with its own GIIS which will contain information about all the resources within the VO.

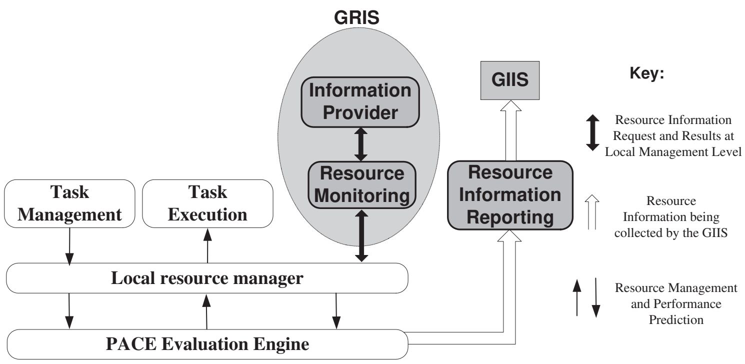

Figure 5: GRIS within a local resource manager.

Figure 5 shows a more detailed diagram of the components of the local resource manager. The latter pulls information from its information providers via its *Resource Monitoring* component and pushes that information to the agent's GIIS via its *Resource Information Reporting* module. The GRIS component is made up of the information providers and the *Resource Monitoring* module.

3 Grid Information Services with Agents

An agent will not only act as a broker to the underlying local resource manager but it will also interface with a Grid Information Service. The agents can therefore, automatically refer to other neighbouring agents in order to access the whole set of their resource information, if their own resources do not meet users' job requirements. The overall system provides an autonomous Grid Information Service which can locate resources across virtual organisations. The way in which the A4 software agents will interface with the MDS is described below.

3.1 Structure of a Grid-enabled Agent

An agent's structure is made up of three layers: the communication, coordination and local management layers. The coordination layer further splits into the following components: the Information Service Module, the PACE Evaluation Engine and the Referrer. The structure of an agent and the interface to its GIIS are shown in Figure 6. The various functions of these components are described.

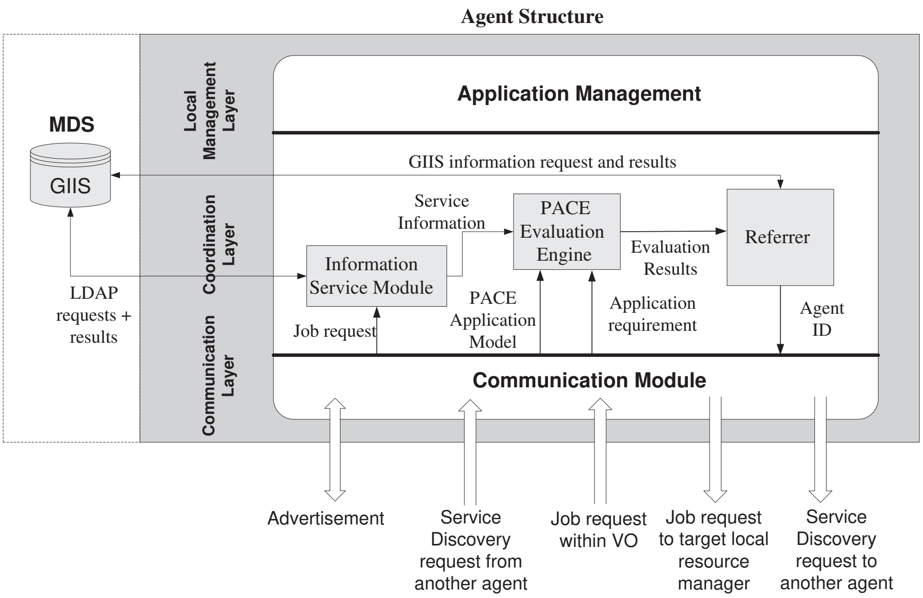

Figure 6: Agent-based GIIS structure. The structure of an agent interfacing with its GIIS is shown. The information flow during resource discovery and performance prediction are indicated by the respective arrows.

One of the functions of the Information Service Module (ISM) is to convert a service discovery request in XML format to an LDAP-based request. This request is serviced by the GIIS which interfaces with an agent. The GIIS contains resource information about all the local resource managers in the corresponding virtual organisation. It also has potential access to information about other virtual organisations' resources since it receives service discovery messages. Therefore, the aggregate directory structure is accessed by the ISM when a new service discovery request arrives. The purpose is to determine whether there is a local resource manager with available resources which can process the job whilst satisfying the user's requirements. In short, querying the GIIS is done via the LDAP request that has been generated beforehand.

A user wishing to run an application on the Grid, submits the actual application and a *job request*. The latter consists of resource requirements, a PACE application performance model and performance metrics which includes deadline requirements. On receipt of the job request, a *service discovery request* is generated, containing resource requirements. The corresponding virtual organisation GIIS is then contacted to discover whether such a resource is available or not. Once a matching resource has been found, the PACE performance model and the deadline requirements are copied over to the discovered local resource manager for performance prediction to be carried out.

When jobs are submitted anywhere within a virtual organisation, the service discovery request is automatically moved to the nearest high-level agent. This is done for the purposes of Grid resource discovery and performance evaluation. In brief, a *request for service discovery* is sent to the agent.

From the LDAP results which are returned from the GIIS, there could be two outcomes. If an adequate resource has been found in the virtual organisation meeting the user's requirements, the job request is passed on to the PACE Evaluation engine. The PACE application model is therefore input to the PACE Evaluation Engine. The latter then performs performance prediction and passes the evaluation results to the referrer. Depending on the evaluation results, the referrer could undertake either of the following. If a resource has been found which can meet the job's predicted execution time and fits onto an existing schedule, it gets scheduled. If this is not possible, the referrer evaluates which agent to contact next for GIIS information; this agent could be an upper or lower agent. Subsequently, the service discovery request is moved to that agent and the above process starts all over again. Likewise, if no adequate resource has been found in the GIIS from the outset, the referrer determines an agent to which the service discovery request is forwarded.

The agent receives both service advertisement and discovery messages via its communication module. It interprets the contents of each message and submits the information to corresponding modules in the coordination layer of the agent. For example, an advertisement message from another agent will be handled by the Information Service module which will consequently add the new information in its own GIIS. Moreover, the communication module is responsible for communicating service advertisement and discovery messages with other agents.

3.2 Information in the Agent-Enhanced GIIS

Cooperation amongst homogeneous agents is defined by the service advertisement and discovery taking place for the purpose of resource management. An agent has access to the following information stored in the GIIS:

1. **Resource information** in the VO.

2. **Agent ID** This is the contact information for an agent. The agent ID is very useful in contacting other agents to search their GIIS. Each agent will initially store its upper agent ID and later when other agents register with it, it will store lower agent IDs.

3. **Service Information** This is where performance-related information about resources is stored. The agent uses this information to evaluate the performance of resources and to ensure that the performance metrics are satisfied. The service information is also used as part of service discovery decisions.

The GIIS will thus contain resource information, as described earlier, as well as performance information.

A resource's GRIS can be directly contacted for information. However, in a Grid environment with a large number of heterogeneous resources, it is more helpful for the local resource manager to contact its agent to look up its GIIS. Each agent will have one or more local resource managers, with each resource manager monitoring one or more resources.

The agent ID is used as follows. The resource needed by a job request might be found in the local virtual organisation, but if no such resource is found, the agent will use one of its agent IDs to contact another agent via its communication layer. More specifically, at the agent's coordination layer, the GIIS is accessed and the ID of the selected agent to contact next is retrieved. The service request is then sent to that agent. It might be appropriate to contact the agents in the hierarchy in a breadth-first search fashion. However, no method of traversing the agent hierarchy is particularly preferred. If a suitable resource and schedule have been found in the GIIS of one of the agents in the hierarchy, the corresponding local resource manager is contacted and the job is forwarded to it. Consequently, at each stage until a schedule on a suitable resource is found, the job remains on the submission host, but the job request moves from one agent to the next.

It is intended that an LDAP data model is used to represent all the information needed to carry out resource discovery and monitoring, and performance prediction. In short, the information is classified as follows:

- All of the resource information(static and dynamic) within the VO;
- Service information pertaining to the performance of resources in the VO (and others);
- ID of other agents which an agent knows about, including its lower and upper agents.

3.3 Information Flow Overview

There are four main types of cross-VO communication between agents:

1. Service advertisement and service discovery messages;
2. Service discovery requests moving from one agent to another;
3. Jobs moving from the portal to the target local resource manager;
4. Job requests moving from one agent to another.

When a new virtual organisation wishes to join the Grid, its agent advertises its service information to other agents. At the same time, other agents are in the process of discovering new agents. Once the new agent has chosen its upper agent, it registers its GIIS with it and the new virtual organisation is now part of the Grid. The performance offered by resources is likely to vary over time, and if a virtual organisation wishes to cease to be part of the Grid, its agent should unregister from its upper agent, thus its resources are no longer available to execute jobs. In this way, the rest of the Grid is unaffected by the removal of a GIIS.

Each resource has an upper local resource manager which continuously monitors its performance. The existing GIIS schema has to be extended from having only hardware information to including the agent ID and service information as well. This is necessary because when a resource has been found, the job request is not sent to that resource directly but to its local resource manager. This method thus allows the resource manager to perform performance prediction before actually submitting the job to the targeted resource.

3.4 Service Advertisement and Discovery

When a virtual organisation wishes to register with another high-level one, the agent uses the GRRP protocol. LDAP queries can be sent from the higher-level GIIS to the lower level one, but there is no existing mechanism for the lower-level GIIS to search higher-level GIIS with which it registered.

This drawback can be avoided by using the agent service advertisement mechanism at the time the virtual organisation is registered. Therefore, the new virtual organisation's agent advertises its service information to other agents. This happens with its upper agent in the first instance. Once the new agent has registered with its upper agent, a record of the latter's ID is stored in the new agent's GIIS.

Service Discovery occurs when other agents' GIIS need to be queried for resource information. The lower or upper agents are thus contacted.

4 Conclusions and Future Work

In this paper, an agent-based Monitoring and Discovery Service has been described, where GIIS from other virtual organisations are automatically contacted for resource information search. This is brought about by using agent service advertisement and discovery. The agents perform such searches automatically, thus creating a referral mechanism which traverses the agent hierarchy in a managed way.

This work is ongoing and the aim of future work is to explore the best agent hierarchy traversal mechanism and the related performance issues when the agents carry out information services searches. The overall performance obtained when writing and reading dynamic information to and from the GIIS respectively, will also be investigated.

Acknowledgements

This work is sponsored in part by grants from the NASA AMES Research Centres (administered by USARDSG, contract no. N68171-01-C-9012) and the EPSRC (contract no. GR/R47424/01).

References

[1] A. M. Alkindi, D. J. Kerbyson, and G. R. Nudd. Dynamic Instrumentation and Performance Prediction of Application Execution. *Proceedings of High Performance Computing and Networking(HPCN2001), Lecture Notes in Computer Science*, Volume 2110, Springer-Verlag, Amsterdam:313–323, June 2001.

[2] A. M. Alkindi, D. J. Kerbyson, E. Papaefstathiou, and G. R. Nudd. Optimisation of Application Execution on Dynamic Systems. *Future Generation Computer Systems*, Volume 17(8):941–949, June 2001. Elsevier.

[3] J. Cao, S. A. Jarvis, S. Saini, D. J. Kerbyson, and G. R. Nudd. ARMS: An Agent-based Resource Management System for Grid Computing. In *Scientific Programming Special Issue on Grid Computing*, 2002.

[4] J. Cao, S. A. Jarvis, D. P. Spooner, J. D. Turner, D. J. Kerbyson, and G. R. Nudd. Performance Prediction Technology for Agent-based Resource Management in Grid Environments. *11th IEEE Heterogeneous Computing Workshop (HCW02)*, 15-19th April 2002. Marriott Marina, Fort Lauderdale, Florida.

[5] J. Cao, D. J. Kerbyson, and G. R. Nudd. High Performance Service Discovery in Large-Scale Multi-Agent and Mobile-Agent Systems. *International Journal of Software Engineering and Knowledge Engineering, Special Issue on Multi-Agent Systems and Mobile Agents, World Scientific Publishing*, 11(5):621–641, 2001.

[6] J. Cao, D. J. Kerbyson, and G. R. Nudd. Use of Agent-based Service Discovery for Resource Management in Metacomputing Environment. *Proceedings of 7th International Euro-Par Conference, Manchester, UK, Lecture Notes in Computer Science, 2150, Springer-Verlag*, pages 882–886, August 2001.

[7] J. Cao, D. P. Spooner, J. D. Turner, S. A. Jarvis, D. J. Kerbyson, S. Saini, and G. R. Nudd. Agent-based Resource Management for Grid Computing. *2nd IEEE International Symposium on Cluster Computing and the Grid*, May 2002. Berlin, Germany.

[8] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman. Grid Information Services for Distributed Resource Sharing. *Proc, 10th IEEE International Symposium on High-Performance Distributed Computing (HPDC-10)*, 2001. IEEE Press.

[9] S. Fitzgerald, I. Foster, C. Kesselman, G. Laszewski, W. Smith, and S. Tuecke. A Directory Service for Configuring High-Performance Distributed Computations.

[10] I. Foster and C. Kesselman. *The GRID: Blueprint for a New Computing Infrastructure.* Morgan Kaufmann Publishers, Inc., 1999.

[11] D. J. Kerbyson, J. S. Harper, A. Craig, and G. R. Nudd. PACE: A Toolset to Investigate and Predict Performance in Parallel Systems. *Presented in European Parallel Tools Meeting, ONERA, Paris*, October 1996.

[12] D. J. Kerbyson, J. S. Harper, E. Papaefstathiou, D. V. Wilcox, and G. R. Nudd. Use of Performance Technology for the Management of Distributed Systems. *Euro-Par 2000, LNCS, Springer-Verlag*, August 2000.

[13] D. J. Kerbyson, E. Papaefstathiou, J. S. Harper, S. C. Perry, and G. R. Nudd. Is Predictive Tracing too Late for HPC Users? *R.J. Allan, A.Simpson and D.A.Nicole (Eds), High Performance Computing, Plenum Press*, 1998.

[14] D. J. Kerbyson, E. Papaefstathiou, and G. R. Nudd. Application Execution Steering Using On-the-fly Performance Prediction. *High Performance Computing and Networking, Amsterdam, Holland, Lecture Notes in Computer Science, Springer-Verlag*, April 1998.

[15] G. Laszewski and I. Foster. Usage of LDAP in Globus.

[16] D. P. Spooner, J. Cao, J. D. Turner, H. N. Lim Choi Keung, S. A. Jarvis, and G. R. Nudd. Localised Workload Management Using Performance Prediction and QoS Contracts. *Eighteenth Annual UK Performance Engineering Workshop (UKPEW' 2002).*