Use of Agent-Based Service Discovery for Resource Management in Metacomputing Environment

Junwei Cao, Darren J. Kerbyson, and Graham R. Nudd

High Performance Systems Laboratory, Department of Computer Science
University of Warwick, U.K.
{junwei,djke,grn}@dcs.warwick.ac.uk

Abstract. A new methodology is presented in this paper for resource management in a metacomputing environment using a hierarchy of homogeneous agents that has the capability of service discovery. The PACE [6] tools are used to provide quantitative data concerning the performance of sophisticated applications running on local high performance resources. At metacomputing level, an agent hierarchy is used to model the resource management system, and the implementation of resource management, scheduling, and allocation can be abstracted to the processes of service advertisement and service discovery. Different optimisation strategies can be used to improve the performance of the agent system.

1 Introduction

The overall aim of the resource management in the metacomputing environment is to efficiently schedule applications that need to utilize the available resources [5]. Such goals within the high performance community will rely on accurate performance prediction capabilities.

Our previous works on PACE [2] can be used to provide quantitative data concerning the performance of sophisticated applications running on local high performance resources. While extremely well-suited for managing a locally distributed multi-computer, the PACE functions do not map well onto wide-area environments, where heterogeneity, multiple administrative domains, and communication irregularities dramatically complicate the job of resource management. There are two key challenges that must be addressed.

Scalability. A given component of the grid will have it's own functions, resources, and environment. These are not necessarily geared to work together in the overall grid. They may be physically located in different organizations and may not be aware of each other.

Adaptability. A grid is a dynamic environment where the location, type, and performance of the components are constantly changing. For example, a component resource may be added to, or removed from, the grid at any time. These resources may not be entirely dedicated to the grid; hence their computational capabilities will vary over time.

R. Sakellariou et al. (Eds.): Euro-Par 2001, LNCS 2150, pp. 882-886, 2001.

In this work, an agent-based service discovery model for resource management in metacomputing environment is introduced to address above challenges. An agent is a local computing resource manager equipped with PACE functions. These homogeneous agents are organised into a hierarchy, which can be used to address the problem of the scalability. An agent is considered to be both a service provider and a service requestor. We use service here to describe the details of a resource within the grid. Resource management, scheduling, and allocation can be abstracted to the processes of service *advertisement* and service *discovery*. Performance issues arise when service is advertised and discovered in the agent system. Different optimisation strategies can be used to improve the system performance.

2 PACE Toolset

Our previous works on PACE toolset provide the base of the implementation of resource management for metacomputing.

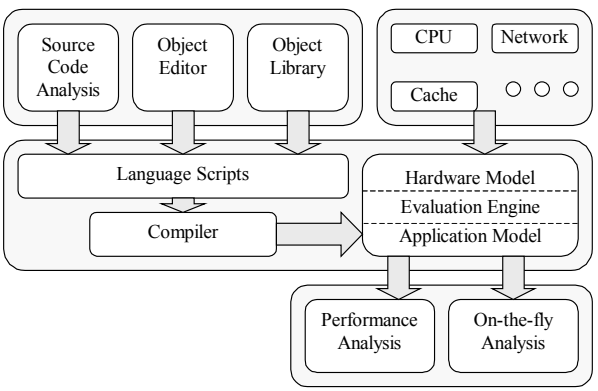

Fig. 1. The PACE Toolset

The main components of the PACE toolset are shown in Fig. 1. A core component of PACE is a performance language, CHIP^{3}S, which describes the performance aspects of an application and its parallelisation. The evaluation engine combines the workload information with component hardware models to produce time estimates, or trace information of the expected application behaviour. An important application of prediction data is that of dynamic multi-processor scheduling, which can be applied for efficient local resource management.

The key features of PACE performance prediction capabilities include: a reasonable prediction accuracy; a rapid evaluation time and easy performance comparison across different computational systems [1]. These enable the prediction data to be used for resource management in a highly dynamic environment.

3 Agent-Based Service Discovery for Resource Management

In this section, a hierarchy of homogenous agents with service discovery capabilities is used for resource management in metacomputing environment. The introduction below includes four parts: agent structure, agent hierarchy, service model and service discovery.

3.1 Agent Structure

There is a single type of the component, the agent, which is used to compose the whole resource management system. Each agent has the same set of functions and acts as a manager of local high performance resources. However, at a meta-level, an agent must also take part in the cooperation with other agents, which differentiate an agent from a current PACE system. A layered agent structure is shown in Fig. 2 and described in detail below.

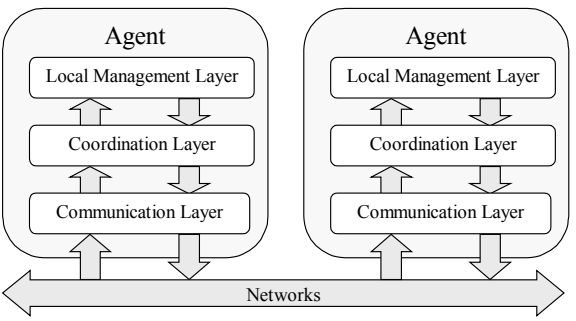

Fig. 2. Layered Agent Structure

Communication Layer – Agents in the system must be able to communicate with each other using common data models and communication protocols.

Coordination Layer – The data an agent receives at the communication layer should be explained and submitted to the coordination layer, which decides how the agent should act on the data according to its own knowledge. The work described in this paper mainly focus on the implementation of this layer, and the service discovery model used to achieve the agent coordination will be described in sections later.

Local Management Layer – This uses existing PACE tools for the management of the locally distributed computing resources. The local manager can also provide information needed by the coordination layer to make decisions.

3.2 Agent Hierarchy

Different agents cannot work together without some base organisation. In this section, a hierarchical model is introduced, which is an extension of our previous work [3].

When a new resource is added into the grid, a new agent is created. The new agent can register with one of existing agents, and hence join the agent hierarchy. An agent can only have one connection to an agent higher in the hierarchy to register with, but

be registered with many lower level agents. An agent in the hierarchy has only the identities of its upper agent and lower agents and can communicate with them at the beginning. However, after joining the system, agent can learn more information about other agents and communicate with more agents gradually. Then the agent can benefit from the other agents' capabilities, cooperate with each other, and manage all of the computing resources in the metacomputing environment. An agent can also leave the hierarchy at any time, which means the resources it manages are not available to the grid any more. In this situation, its lower agents must be informed to register with a new agent to keep the relations with the hierarchy.

The agent-based hierarchical model is used to address the problem of the scalability. The key feature that enables the scalability of this model is the homogeneity of agents. There is no agent with more special functions than any other. The broker does not have any privileges compared to coordinators and agents.

3.3 Service Model

An agent is considered to be both a service provider and a service requestor. Every agent can also act as a router between a request and a service. A service model is used to describe the details of local high performance resources within the grid. An initial implementation of a service model is a simple data structure, including an agent identity, corresponding resource information, and some optimisation options.

The main part of a service model is the resource information. This should include all of the information needed for performance prediction of corresponding resources, from hardware statistics to application states. When a request arrives, it should be able to turn into useful performance information of corresponding resources using the PACE evaluation engine. These predictions can be used for agents to make decision whether the corresponding resources can meet the performance requirements from the request at the coordination layer.

An agent can use several kinds of Agent Capability Tables (ACTs) for maintenance of service models from other agents. The resource performance can vary over time, thus the service offered by the agent will change over time. When this occurs, the resource information in corresponding service model needs also to be updated. The dynamics of this system increases the difficulty of resource management and allocation. The essential issue is how an agent advertises its services and also coordinates with other agents to discover the required services in the most efficient way.

3.4 Service Discovery

Resource management, scheduling, and allocation at the meta-level are abstracted to the processes of service advertisement and service discovery in this work. These are performed at the coordination layer in each agent in the system.

Service Advertisement. The service model of an agent can be advertised in the hierarchy (both up and down).

Service Discovery. When an agent receives a request, it will first check its own knowledge to see if it is already aware of an available service, which can meet the performance requirements of the request. If it is, it will contact the target agent

directly. Otherwise it may contact other (e.g. its upper or lower) agents until the available service is found.

Different strategies can be used to decide when, and how, to advertise or discover a service but with different performances. These include use of cache, using local and global knowledge, limit service lifetime, limit scope. The performance issues are discussed in greater detail in [4].

4 Conclusions

The resource management system in the grid computing environment will be a large-scale distributed software system with high dynamics. In this work, we have developed a homogeneous agent-based hierarchical model to meet the requirements of the scalability. We also abstract the resource management, scheduling and allocation into the processes of the service advertisement and discovery.

Ongoing works include the implementation of an agent communication language and a service description language. Future implementation will also integrate a new version of PACE toolset, which is specially lightened up for remote performance evaluation.

References

1. J. Cao, D. J. Kerbyson, E. Papaefstathiou, and G. R. Nudd, "Modeling of ASCI High Performance Applications Using PACE", in Proc. of 15th Annual UK Performance Engineering Workshop, Bristol, UK, pp. 413-424, 1999.
2. J. Cao, D. J. Kerbyson, E. Papaefstathiou, and G. R. Nudd, "Performance Modeling of Parallel and Distributed Computing Using PACE", in Proc. of 19th IEEE Int. Performance, Computing and Communication Conf., Phoenix, USA, pp. 485-492, 2000.
3. J. Cao, D. J. Kerbyson, and G. R. Nudd, "Dynamic Application Integration Using Agent-Based Operational Administration", in Proc. of 5th Int. Conf. on Practical Application of Intelligent Agents and Multi-Agent Technology, Manchester, UK, pp. 393-396, 2000.
4. J. Cao, D. J. Kerbyson, and G. R. Nudd, "Performance Evaluation of an Agent-Based Resource Management Infrastructure for Grid Computing", in Proc. of 1st IEEE Int. Symp. on Cluster Computing and the Grid (CCGrid'01), Brisbane, Australia, pp. 311-318, 2001.
5. I. Foster, and C. Kesselman, "The Grid: Blueprint for a New Computing Infrastructure", Morgan-Kaufmann, 1998.
6. G. R. Nudd, D. J. Kerbyson, E. Papaefstathiou, S. C. Perry, J. S. Harper, and D. V. Wilcox, "PACE – A Toolset for the Performance Prediction of Parallel and Distributed Systems", Int. J. of High Performance Computing Applications, Special Issues on Performance Modelling – Part I, Sage Science Press, 14(3), pp. 228-251, Fall 2000.