Adaptive Agent-based resource management for GRID

Tariq Abdullah Koen Bertels Stamatis Vassiliadis

Computer Engineering Laboratory, Delft University of Technology,
Mekelweg 4, 2628 CD Delft, The Netherlands

tariq@ce.et.tudelft.nl koen@ce.et.tudelft.nl stamatis@ce.et.tudelft.nl

Keywords: computational grid, agents, adaptive multiagent systems, agent-based resource management

Abstract

A grid system is required to integrate heterogeneous resources with varying quality and quantity. In this article a comprehensive overview of Grid, Grid types and a high level overview of basic components in a Grid environment is presented. A taxonomy of attributes for a Grid RMS, an overview of Grid RMS and their implementations in different Grid projects is presented. Agents are able to behave autonomously, intelligently, learn from environment, and adaptability. Basic concepts of agents are presented. From the literature survey, it is obvious that efforts have been made to use different characteristics of agents for resource management in grid. None of these focuses on an adaptive agent-based approach for grid resource management system.

1 Introduction

Grid computing is used for different areas such as cloud detection, high energy and nuclear physics, gravity wave searches, 3-D time dependent brain scans etc. All the above mentioned example applications and many others application like above require heavy and dedicated computational resources for which grid provide an answer.

Apart from the above specialized domain, individual organizations increasingly execute computing/data intensive applications like OLAP, data mining, data warehousing, stock market simulations, medical instrumentation etc. These applications also required dedicated high power computational resources. Various studies report utilizations of around 30% LAN/WAN computational resources in academic and commercial environments [1]. These idle resources can be used to complete the tasks, but these resources are not available for any specific time period. In addition, the available resources probably come in various flavors and will be characterised by different operating system (versions), processor families etc. It is clear that this heterogeneity also needs to be managed in a certain way. It is required to develop a framework that can utilize these resources on ’when and where available’ basis. Furthermore, this usage needs to be transparent from application/users. To achieve this, we need to develop a grid infrastructure which will be flexible, scalable, adaptive in nature and will enable dynamic composition of loosely coupled resource in a large-scale distributed system.

One way of implementing these grid capabilities is to use multi-agent system. Agents have the ability to adapt themselves according to prevailing circumstances in an [grid] environment. Agents also provide services that are robust and scalable. Because of these characteristics agents are the natural choice for achieving our goal. In this article we present a framework named agent-based adaptive resource management system for grid infrastructure.

This article is organized as follows: in section 2 an overview of grid computing, grid types, basic components of a grid application and an overview of grid RMS is presented. After this, a survey of RMS in different grid toolkits is presented in this section. Basic concept software agents, adaptive multiagent systems and agent-based service discovery for resource management is provided in section 3. In section 4 some problems and their solution is proposed. In section 5 conclusions and future research challenges are discussed.

2 Grid Computing: An Overview

Grid computing is becoming a mainstream technology for large-scale resource sharing and distributed system integration. There are many definitions for Grid Computing, most commonly referred definition is given by Ian Foster [1], ”Resource sharing and coordinated problem solving in dynamic multi-institutional virtual organizations”. In a recent article by Foster [2], this definition is modified to a three point checklist. These three points are:

- Coordinating resources that are not subject to centralized control

- Using standard, open, general-purpose protocols, interfaces
- To deliver nontrivial qualities of service

All these three criterion are met in major large scale Grid deployment projects. Examples include GriPhyN, PPDG, EU DataGrid, iVDGL, DataTAG, Information Power Grid, Distributed ASCI supercomputers (DAS-2), the DOE Science Grid, DISCOM Grid, TeraGrid. All these systems use open, general-purpose protocols (of Globus Toolkit) to share and coordinate resources from multiple locations. All these projects are rendering qualities of service in different directions like security, reliability and performance.

2.1 Grid Types

Grids can be categorized into three broader categories that focus on the functional aspect:

- **Data grid:** Data Grid provides an infrastructure to store and access data across multiple organizations. Data can be stored in the digital libraries or in the data warehouses. Possible use of data can be for synthesizes purposes. A data grid provides location transparent access of data.
- **Service grid:** A service grid provides some functionalities to the existing application services. These services can be message queuing, routing, security etc. This category is further subdivided into on-demand computing, collaborative computing, multimedia computing.
- **Computational grid:** Computational Grid as defined by Foster [1] as, ”a hardware and software infrastructure that provides dependable, consistent, pervasive and inexpensive access to high end computational capabilities”. This can be further subdivided into distributed supercomputing, high throughput.

Different grid categories and some examples are summarized in Table 1. Grid users can be divided into different categories like grid developers, tool developers, application developers, system administrators and end-users.

2.2 Grid Components

A Grid computing environment generally requires several components. A high level overview [3] of some basic functions / services of the components is given below:

- **User or application interface:** A grid interface provides access services to users or to grid applications. Users can launch applications through this interface. Users are able to utilize resources and services provided by Grid. This interface can be as simple as a web page or it may be as complex as anything.
- **Security:** A major requirement for successful implementation of a grid is the adequate management of security. This component provides the following basic services: secure sign-on, authentication and access management. Access management performs the following tasks: confirms authority, coordinates access rights and privileges, exchanges credentials, exchanges certificates and asserts trust relationships.
- **Resource management:** Resources in grid can be computing cycles, storage, caching capability, network bandwidth, databases and application software etc. Mechanism is required to manage available resources to the users and to the applications. The main functions of resource management include resource discovery, ability to locate a needed resource, advertisement and registration of resources, management of resource attributes and characteristics (conguration, availability, cost, usage policy and constraints), and state management of a resource.
- **Job management:** Mechanisms are required to manage user and application-initiated jobs in a grid environment. In general following activities are required to manage a job: initiate a job, schedule a job, monitor and control the job, monitor and control resource assignment to the job, assemble the results and distribute the results.
  Last but not the least is the accounting system. An accounting system mostly includes billing mechanisms. It becomes more important when grid must purchase the resources.
  In the remainder of the paper, we focus exclusively on RMS-issues. We first provide an overview of existing approaches and .then focus on issues that conform to our requirements.

3 Grid Resource Management System

A Grid RMS is a central and complex part of any Grid project/implementation. A typical Grid RMS performs the following jobs: allocate a resource, authenticate a resource, authorize a resource, assurance of a resource, accounting and auditing of a resource. RMS becomes complex due to following issues: site autonomy, resource heterogeneity, difference in resource usage, scheduling policies, security mechanism, requirement for interfacing and interoperation with local resource management systems,

Category	Examples	Characteristics
High throughput	Chip design,Parameter studies, Cryptographic problems	Harnessing many otherwise idle resources to increase aggregate throughput
On demand	Medical instrumentation,Network-enabled solvers,Cloud detection	Remote resources integrated with local computation, often for bounded amount of time
Data intensive	Sky survey,Physics data, Data assimilation	Synthesis of new information from many or large data sources
Collaborative	Collaborative design,Data exploration, Education	Support communication or collaborative work between multiple participants
Distributed Supercomputing	DIS,Stellar Dynamics, Ab-initio chemistry	very large problems needing lots of CPU, memory,etc

Table 1: Grid application categories, examples & characteristics

need for the negotiation between resource users and resource providers [4]. Different taxonomies for Grid RMS can be defined based on the following criterion: Grid type, resource namespace, resource dissemination protocol, resource discovery, scheduling model, state estimation, scheduling policy. These criterion are defined [5] and are refined by [6, 7]. We are introducing some more attributes to introduce our proposed work. These new attributes are robustness, scalability and adaptability.

Survey of RMS in different Grid Toolkits

A good survey of Grid RMS taxonomies of different grid projects and their RMS can be found in [5]. A recent and up-to-date survey of grid resource management systems is provided in [6]. In this paper, a survey of grid RMS in existing Grid toolkits is presented in Table–2, which is an extended version of [5]

- **2K–A distributed operating system:** 2K [8] is a Distributed Operating System developed for development and deployment of distributed service applications. It works in heterogeneous environments and provide a flexible and adaptable architecture to provide a variety of service in a heterogeneous environment. 2K is based on CORBA technology. 2K RMS system uses an object model with a graph-based resource namespace and provides a soft QoS. Agents are used for resource discovery. Mobile agents are used for resource dissemination in an ”on demand” fashion. State estimation and rescheduling policy is not present in 2K but is dependent upon underlying native OS. 2K uses a centralized controller for resources. Scheduling policy is fixed.

- **SmartFrog:** SmartFrog [9] is a framework for service configuration, description and lifecycle management. Its RMS adopts a component-based architecture with a declarative language for describing service conguration and provisioning.

- **AppLeS–A network enabled scheduler:** AppLeS project [10] focuses on utilizing individual scheduling agents for every application on a computational grid. AppLes uses RMS services of other grid toolkits like Globus, Legion and NetSolve to complete application running on it. AppLeS resource schedulers does not offer any QoS support. It contains a predictive heuristic state estimation model, online rescheduling and xed application oriented scheduling policy. Apples is being used for large scale application like satellite radar image visualization, gene sequence comparison.

- **Bond–Java distributed agents:** Bond [11] is based on agents [12] that communicate with each other using KQML. Bond support on-demand service grid and has a flat machine organization. Agents in Bond are structured into finite state machines. Bond's RMS has two level decentralized scheduler and is based on computational market economy. Resource model is object based, has a hard QoS and contains graph-based namespace. Resource discovery is agent-based and dissemination is through periodic push. Rescheduling policy is online and scheduling policy is fixed and application oriented.

- **ASKALON:** Askalon project [13] provides a tool set for service-based performance-oriented development of grid applications.

- **Condor–Cycle stealing technology for high throughput computing:** Condor [14, 15] is a high-throughput computing environment capable of managing large collection of heterogeneous operating environments. Condor is famous for utilizing idle computer resources. Condor has a layered Grid architecture and supports sequential and parallel applications. Condor stores resource/job information in Condor pools and it can have many Condor pools. Each pool follows a at RMS organization. Condor resource requests are normally described in the Condor classified advertisement language. It has

System	Grid Type	Resource Management	Scheduling	Adaptivity	Scalability
2K	Hierarchical On-Demand	agent-based discovery, centralized	hierarchical network resource scheduler, decentralized scheduler for other resources	yes	no
AppLes	High-throughput	Resource model provided by Globus, Legion or NetSolve	Hierarchical scheduler fi xed application oriented policy	no	no
Bond	on-Demand , flat	Extensible object model agent-based discovery	decentralized scheduler, online rescheduling	no	no
gridBus	Computational	different components agent-based discovery	based on computational economy online rescheduling	no	no
Condor-G	Computational Flat	Agent-based, centralized queries	centralized scheduler	yes	no
Darwin	Multimedia Hierarchical	Fixed schema model	Hierarchical scheduler online rescheduling	no	no
European DataGrid	Data	Extensible schema model distributed queries	Hierarchical scheduler, extensible scheduling policy	no	no
Globus	various Hierarchical cell	extensible schema model distributed queries discovery,	decentralized scheduler infrastructure, scheduling provided by external schedulers(AppLes, Nimrod/G	no	no
Javelin	computational Hierarchical	fi xed object model distributed queries discovery	decentralized scheduler, fi xed application oriented policy	no	no
GOPI	Multimedia flat	Extensible object model	decentralized scheduler, ad hoc extensible policy	no	no
Legion	Computational Hierarchical	Extensible object model distributed queries	hierarchical scheduler	no	no
MOL	computational hierarchical cell	extensible, schema model distributed queries discovery	decentralized scheduler, extensible ad hoc scheduling policies	no	no
NetSolve	Computational Hierarchical	agent-based resource allocation	decentralized scheduler, fi xed application oriented policy	no	no
Nimrod/G	High-throughput hierarchical cell	extensible schema model distributed queries discovery	hierarchical decnetralized scheduler, fi xed application oriented policy	no	no
Ninf	Computational Hierarchical	fi xed schema model centralized queries discovery	decentralized scheduler	no	no
PUNCH	Computational hierarchical	extensible schema model distributed queries discovery	hierarchical decentralized scheduler, fi xed application oriented policy	no	no
Globus-Radia[6]	Hierarchical cells	query based, adaptive and scalable middleware	application oriented policy	yes	no

Table 2: Grid RMS

a flat organization. It uses an extensible schema with a hybrid namespace. There is no QoS support. The information store is a network directory. Resource discovery is centralized queries. Resource dissemination is periodic push.

- **GridBus:** It [16] is an open-source middleware toolkit for computational grids. It is based on notion of 'utility computing'. It utilizes various economic models for efficient management and use of resources.

- **Darwin–Resource management for network services:** Darwin [17] provides a virtual network to distributed applications. Applications in Darwin environment,submit resource requirements in form of a graph and accompanying QoS specifications. Xena is resource broker in Darwin. Xena performs global resource allocation. Local resource managers provide low-level resource allocation and coordinate their activity. Darwin uses a hierarchical fair service curve scheduling (H-FSC) algorithm for higher level resource allocation.
  The RMS in Darwin is hierarchical, resource model is a fixed schema with hard QoS support and the resource namespace is graph-based. No separate resource information store, resource discovery protocol or resource dissemination protocol. Scheduling is hierarchical with non-predictive state estimation. Rescheduling is event driven and implemented by the control delegates. The scheduling policy is xed and system oriented.

- **NAREGI:** It [18] aims to research and develop high-performance, scalable grid middleware for the Japanese national scientific computational infrastructure. The implementation of the NAREGI framework is divided into six R and D groups.

- **European DataGrid–Global physics data storage and analysis:** Its focus is on the development of middleware services in order to enable distributed analysis of data. Its core middleware system is based on Globus toolkit with hooks for data Grids [19]. Data is of several peta-bytes and is distributed in a hierarchical manner on multiple sites worldwide. Global namespaces are required to handle the creation of data and access to distributed and replicated data items. Special workload distribution facilities balance the analysis jobs.
  DataGrid project has a multi-tier hierarchical RMS organization. For example, tier-0 is CERN, which stores almost all relevant data, several tier-1 regional centers (in Italy, France, U.K., USA, Japan) support smaller amounts of data. It has an extensible schema-based resource model with a hierarchical namespace organization. RMS has no QoS support. Resource information store is expected to be based on an LDAP network directory. Resource dissemination is batched and periodically pushed to other parts of the Grid. Resource discovery in the Data Grid is decentralized and query based. Its scheduler uses a hierarchical organization with an extensible scheduling policy.

- **GrADS:** This [20] framework is designed to solve numerical applications over the grid. Its RMS has prediction ability about job migration based on marginal differences between actual and predicted execution times.

- **Globus–A toolkit for Grid computing:** Globus toolkit [21] has components to implement basic Grid services like security, resource discovery, resource management, data management, resource reservation and communication. Its RMS has a layered architecture and higher level layers can be developed by using lower-level core services [22]. Globus offers Grid information services via Metacomputing Directory Services (MDS) [23]. MDS has two components: Grid Index Information Service (GIIS) and Grid Resource Information Service (GRIS). GRIS provides resources discovery services. The resource information providers use a push protocol to update GRIS periodically. GIIS provides a global view of the Grid resources. Globus has a push resource dissemination strategy. Resource discovery is performed by querying MDS. Soft QoS support is via resource reservation [23]. Application level schedulers like Nimrod/G, AppLeS and Condor/G can extend predened Globus scheduling policies. Scheduler has a decentralized organization and an ad hoc extensible scheduling policy.

- **GridLab:** GridLab [24] framework is based on a Grid Application Toolkit (GAT). GAT implements a set of high-level APIs and by using these APIs clients and applications can call underlying grid services.

- **Javelin–Java parallel computing:** It [25] is a computational Grid for high-throughput computing. It consists of clients, hosts and brokers. Supports piecework, branch and bound computational models [5]. It has a hierarchical RMS organization. Its resource model is with fixed objects and a graph based namespace organization. Resources are simply the hosts that are attached to a broker. In Javelin, Information is managed by JavelinBNS system. Decentralized query based approach is used for resource discovery. A decentralized approach in scheduling. It has a fixed application oriented scheduling policy.

- **GOPI–Generic Object Platform Infrastructure:** It is based on CORBA with RM-ODP extension (provides extensible architecture for adaptive multimedia applications) [26, 27]. Gives an API and core services that are extended using network-based application specific protocols. In it RMS, resource namespace is based on the RM-ODP computational model and is specified using CORBA IDL. Adaptive QoS support is provided. It has a flat organization. Extensible object-oriented resource model. It uses a graph-based resource namespace. There is no resource information directory, no resource discovery protocol and no resource dissemination protocol. Scheduler organization is decentralized. It has an ad hoc extensible scheduling policy. State estimation and rescheduling are determined by the application specification.

- **Legion–A grid operating system:** Legion [28] is an object-based metasystem that provides the software infrastructure for a Grid. Classes or meta-classes are performing resource information management and information store organization. IN Legion, RMS architecture is hierarchical. Pull resource state estimation strategy is used. Its schedular is decentralized. Extensible scheduling policies are dependent upon underlying system.

- **UNICORE:** It [29] is a vertically integrated grid computing environment. It adopts a 3-tier architecture and an easy to use GUI for creating Abstract Job Object (AJO). AJOs are incorporated in target system specification UNICORE Network Job Supervisor. NJS also manages submitted jobs and performs user authorization.

- **MOL–Metacomputing Online:** The MOL [30] system follows a toolkit approach with the MOL-kernel as its central component. MOL-kernel provides services for resource management, fault management, access provision and supports dynamic communication. MOL-kernel follows a three-tier architecture consisting of resource abstraction, management and access layers containing resource module (RM), center management modules (CMMs) and access module (AM) respectively along with customizable and predened event handlers. MOL follows a service grid model with hierarchical cell-based machine organization. It has a schema-based resource model. It contains hierarchical name space organization. An object-based resource state information. Schedulers of AM perform resource discovery. Scheduling is performed by querying shared objects.

- **NetSolve–A network-enabled computational kernel:** NetSolve [31] is a client-agent-server paradigm based network-enabled application server. Communication between Netsolve clients, agents and servers is performed using TCP/IP sockets. It is a computational grid. Its RMS has a hierarchical machine organization. RMS uses push protocol for resource dissemination. Agent-based resource discovery and scheduling. It has a decentralized scheduler and a fixed scheduling policy.

- **Nimrod/G–Resource broker and economy grid:** Nimrod/G [32, 33] is a Grid resource broker for managing and steering task farming applications on computational Grids. It provides easy to use interfaces for formulation of parameter studies. It uses a single window to manage and control experiments, resource discovery, resource trading and scheduling. Nimrod/G engine can create user-defined scheduling policies. For example, ActiveSheets are used to execute Microsoft Excel computations/cells on the Grid [34]. Its RMS has a hierarchical machine organization. It uses a computational market model for resource management [5, 32, 33]. Uses the services of other toolkits, such as Globus and Legion, for resource discovery and dissemination. State estimation is done through heuristics using historical pricing information. The scheduling policy is xed-application oriented.

4 Software Agents

Software agents grow out of what we called Distributed Artificial Intelligence (DAI). There is no single definition that is acceptable . However our working definition of a software agents is, a piece of software that has a number of properties that are useful for our purpose. Examples are autonomy, social ability, ongoing execution, intelligence, agent awareness, mobility, anthropomorphism, reactivity, proactiveness and adaptiveness.

4.1 Adaptive Multiagent Systems

The adaptive multiagent systems are those systems in which behavior of components is depending upon the behavior of neighboring components. Adaptive multiagent systems are based on the concept of collaborative emergence. These systems are normally plugged into a dynamic environment and a never ending process of adaption is started [35].

4.2 Agent-based Service Discovery for Resource Management

Agents are used in grid and for grid resource management systems. In [36], a general agent-based architecture is proposed for computational grids. An agent framework is defined for service and resource discovery. This framework is based on service advertiser agent, a matchmaker agent and a service request agent. The DARPA agent markup language (DAML) [36] is used for service specification. It appeares that authors have only used agents as wrappers to existing grid components. In [37] negotiation agents based on economic models are used for grid resource management. Adaptivity of proposed system is left an open issue based on the research on multiagent systems by authors. In [38], authors presented adaptive negotiation strategies for agent-based load balancing and grid computing. In this way authors attempted to address issues of scalability and adaptability for grid. This work is based on assumption of neglecting some crucial factors like volume of data to be transferred, network bandwidth, traffic and security etc. In [39], a mobile agent based middleware is proposed that uses those participating machines that are idle at any time. This middleware is composed of a layered architecture. These layers are: collection of client nodes and servers, mobile agents, user wrapper programs and user programs. In [40, 41], agents are organized into a hierarchy. Agents cooperate with one another to discover available grid resources using decentralized resource advertisement and discovery technique. Implementation details of resource management, scheduling and allocation are abstracted from the process of service advertisement and service discovery. In ARMSim [42], a modeling and simulation environment is presented to investigate the Agent-based Resource Management System (ARMS) performance. In [43] agents are viewed as a representative of grid resource. Each agent maintains different Agent Capability Tables for different purposes. PACE [44] evolution engine is integrated into each agent. PACE is also used to provide support in service discovery process. Main focus of this work is on performance prediction for agent-based resource management in grid environments.

4.3 Open issues and proposed solution:

After review of existing literature, following issues are identified for future work. Future grids will require to enable dynamic composition of loosely coupled resources in large-scale distributed systems. These resources include processors, software components, memory and disk storage, high-speed data transfer capabilities, and databases. The success of a grid system implementation requires sufficient strategies for reliable management of large numbers of distributed resources in uncertain and volatile conditions. In a grid implementation it is possible that an application requires more resources which are not available in its environment. In this case, the executing application can be completed by adding more resources from some neighbouring grid. In the converse case, a grid environment may have some excess resources which it want to sell. In either case, some nodes of grid have to communicate with external environment of the grid, in order to negotiate for price/exchange of required/excess resources and presently there is no well defined mechanism to accomplish this task in existing grid toolkits.

A resource manager is responsible for proper distribution of resources in a grid to all participating nodes. Problems arise when a resource manager fails and some other node is required to be declared as a new resource manager. Issues of interest are: how a node will be promoted as a resource manager? How other nodes are informed about failure? How similar future situations cane be avoided?

When an application from a specific application category (like data intensive, computation intensive etc) is executed on a grid infrastructure repeatedly then in every execution cycle, some older challenges and some new challenges are faced. Grid will be required to learn its previous mistakes and don't repeat. At present, there are no mechanisms that can achieve this goal.

In an executing grid, the main problem occurs when an executing node/resource broker fails. The work load of those nodes need to be moved on some other node/resource broker. This requires determining less overloaded nodes and then moving application from failed node/resource broker to the less loaded node. This issue needs to be targeted.

The resource manager has to manage a variety of resources like processors, software components, memory, disk storage, high-speed data transfer capabilities and databases etc. Resource manager has to distribute all available resources in such a way that all resources are properly utilized. It is desirable that grids should have ability to learn from heuristic data and afterward decide optimized and efficient resource utilization strategy by themselves.

- **Communication, cooperation & Scalability:** It is possible that a grid environment, call it Grid "A", needs more resources and some nodes in A (known as communicating nodes) communicate with communicating nodes if grid "B" and ask for required resources. In this scenario, grid may expand its infrastructure and will become large. In a vice versa case, a grid environment may have abundant resources, in this case communicating nodes of Grid A will contact with communicating nodes of Grid B and will offer excess resources. By doing this grid is, size of grid will reduce. Here agents can play their role by utilizing their communication and co-operation characteristics to make a grid scalable.

  Furthermore, In a dynamic environment it may happen sometimes that there are more consumers than providers, in order to scale up with the changed scenario, it is important and necessary to inject required resources to the grid environment, it can be done by inducting agents in grid resource management part and by utilizing their reactivity, autonomy and cooperation capabilities.

- **Adaptivity & Intelligence:** Presently all grid implementations have a fixed number of nodes but in future it is expected that during the course of execution of a grid application, number of participating nodes in grid environment will be changing. According to this changing number of nodes, it will be required to have some option to increase/decrease number of assistant resource manager during the course of execution of a grid-based application. This problem requires some intelligence in resource manager to carry out the decision about dynamically changing number of resource managers. Here we can think to introduce agents, because of their adaptive and intelligence nature.

  In addition to above scenario, infrastructure of a grid environment is designed according to type of grid. Sometimes it may happen that a in data grid there is a heavy requirement of computational resources for a short period of time and vice versa. In this scenario adaptive nature of agents can be used in resource management and agents can do this temporary change by providing required resources to participating nodes in that grid environment.

Another scenario is possible that a partial failure exists in grid environment due to overloaded resource manger. Administrator/ user does not possess sufficient knowledge to tackle that situation and because of it some more partial/complete failures can happen in grid environment. Agents can be useful in this situation if they possess heuristic data and can learn how to handle this situation, so learning capability of agents can be utilized in this scenario.

Furthermore, resources of a grid may include processors, software components, memory and disk storage, high-speed data transfer capabilities and databases. Demand grows for grid computing over increasingly larger scales, the success of grid system implementation requires strategies for reliable management of large numbers of distributed resources in the face of uncertain and volatile conditions. By utilizing adaptive characteristics of agents, this problem can be tackled.

5 Conclusions:

From early days of grid evolution, grid community has focused on ”brawn” (infrastructure, tools, reliable & secure resource sharing etc) and agent community has focused on brain (autonomous, flexible problem-solvers, able to perform specific role, able to interact intelligently with environment etc). By the passage of time, agents require robust infrastructure and grid applications require autonomous, flexible behaviour. Efforts have been made to converge both the communities like. In this paper an overview of existing grid toolkits and grid resource management systems is provided. It is found that grids are being used successfully for very large scale applications with dedicated computing resources. It is proposed that by using idle resources in an organization wide LAN/WAN, grid applications for an organization can be deployed. To achieve this goal some future research issues are identified

References

[1] Ian Foster and Carl Kesselman. *”The Grid: Blueprint for a new computing infrastructure”*. Morgan Kaufmann, 1999.

[2] Ian Foster. What is the grid? a three point checklist. *GridToday*, 2002.

[3] IBM Corporation. *”Enabling applications for Grid Computing with Globus”*, june 2003. Order Number: SG24-6936-00.

[4] Hwa Min Lee, Sung Ho Chin, Jong Hyuk Lee, Dae Won Lee, Kwang Sik Chung, Soon Young Jung, and Heon Chang Yu. ”a resource manager for optimal resource selection and fault tolerance service in grids”. In *IEEE International Symposium on Cluster Computing and the Grid, 2004. CCGrid 2004.*, 2004.

[5] K. Krauter, R. Buyya, and M. Maheswaran. ”a taxonomy and survey of grid resource management systems for distributed computing”. In *Software -Practice and Experience 32(2)*, 2002.

[6] Jyotishman Pathak, Jem Treadwell, Raj Kumar, Philip Vitale, and Fernando Fraticelli. ”a framework for dynamic resource management on the grid”. *HP Laboratories Palo Alto, HPL-2005-153*, August 2005.

[7] Chaitanya Kandagatla. ”survey and taxoonomy of grid resource management systems–technical report”. *University of Texas, Austin*.

[8] Kon F, Campbell R, Mickunas M, and Nahrstedt K. ”2k: A distributed operating system for dynamic heterogeneous environments”. In *9th IEEE International Symposium on High Performance Distributed Computing (HPDC 00)*, 2000.

[9] P. Goldsack, J. Guijarro, A. Lain, G. Mecheneau, P. Murray, and P. Toft. ” smartfrog: Conguration and automatic ignition of distributed applications.”. In *In HP OpenView University Association Workshop*, 2003.

[10] Berman F and Wolski. ”. the apples project: A status report”. In *8th NEC Research Symposium*, 1997.

[11] Boloni L and Marinescu DC. ”an object-oriented framework for building collaborative network agents.”. In *Intelligent Systems and Interfaces, Kandel A, Hoffmann K, Mlynek D, Teodorescu NH (eds.). Kluwer)*, 2000.

[12] Jun K, Boloni L, Palacz K, and Marinescu DC. ”agent-based resource discovery”. In *9th IEEE Heterogeneous Computing Workshop (HCW 00)*, 1999.

[13] T. Fahringer, A. Jugravu, S. Pllana, R. Prodan, C. S. Junior, and H.-L. Truong. ”askalon: A tool set for cluster and grid computing.”. In *Proceedings of Concurrency and Computation: Practice and Experience*, 2005.

[14] Litzkow M.J., Livny M., and Mutka M.W. ”condor: A hunter of idle workstations”. In *8th International Conference of Distributed Computing Systems*, 1998.

[15] Basney J and Livny M. ”deploying a high throughput computing cluster”. In *High Performance Cluster Computing: architectures and systems*, 1999.

[16] Gridbus,http://www.gridbus.org/.

[17] Chandra P, Fisher A, Kosak C, Ng TSE, Steenkiste P, Takahashi E, and Zhang H. Darwin. ”darwin:customizable resource management for value-added network services”. In *6th IEEE International Conference on Network Protocols*, 1998.

[18] S. Matsuoka, S. Shimojo, M. Aoyagi, S. Sekiguchi, H. Usami, and K. Miura. ”japanese computational grid research project: Naregi”. In *Proceedings of the IEEE*, 2005.

[19] Hoscheck W, Jaen-Martinez J, Samar A, Stockinger H, and Stockinger K. ”data management in an international data grid project.”. In *1st IEEE/ACM International Workshop on Grid Computing (Grid 2000)*, 2000.

[20] S. Vadhiyar and J. Dongarra. ”self adaptivity in grid computing. ”. In *Proceedings of Concurrency And Computation: Practice And Experience*, 2005.

[21] Globus home page,http://www.globus.org/.

[22] Czajkowski K, Foster I, Kesselman C, Karonis N, Martin S, Smith W, and Tuecke S. ”a resource management architecture for metacomputing systems.”. In *Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing*, 1998.

[23] Fitzgerald S, Foster I, Kesselman C, von Laszewski G, Smith W, and Tuecke S. ” a directory service for conguring high-performance distributed computations.”. In *Proceedings of the 6th International Symposium on High Performance Distributed Computing (HPDC ’97)*, 1997.

[24] G. Allen, K. Davis, and et al. ”enabling applications on the grid: A gridlab overview.”. In *Intl. Journal of High Performance Computing Applications*, 2003.

[25] Neary M, Phipps A, Richman S, and Cappello P. ”javalin 2.0: Java-based parallel computing on the internet.”. In *Proceedings of the European Parallel Computing Conference (Euro-Par 2000)*, 2000.

[26] Coulson G. ”a configurable multimedia middleware platform”. In *Proceedings of IEEE Multimedia 1999*, 1999.

[27] Coulson G and Clarke M. ”a distributed object platform infrastructure for multimedia applications.”. In *Computer Communications 1998*, 1998.

[28] Chapin S, Karpovich J, and Grimshaw A. ” the legion resource management system.”. In *Proceedings of the 5th Workshop on Job Scheduling Strategie s for Parallel Processing*, 1999.

[29] J. Almond and D. Snelling. ”unicore: Uniform access to supercomputing as an element of electronic commerce.”. In *Proceedings of Future Generation Computer Systems*, 1999.

[30] Gehring J and Streit A. ” robust resource management for metacomputers.”. In *Proceedings 9th IEEE International Symposium on High Performance Distributed Computing*, 2000.

[31] Casanova H and Dongarra J. ” netsolve:a network-enabled server for solving computational science problems.”. In *International Journal of Supercomputer Applications and High Performance Computing 1997*, 1997.

[32] Buyya R, Giddy J, and Abramson D. ” an evaluation of economy-based resource trading and scheduling on computational power grids for parameter sweep applications.”. In *Proceedings of the 2nd International Workshop on Active Middleware Services (AMS 00)*, 2000.

[33] Buyya R, Abramson D, and Giddy J. ” nimrod/g: An architecture for a resource management and scheduling system in a global computational grid.”. In *Proceedings of the International Conference on High Performance Computing in AsiaPacic Region (HPC Asia 2000)*, 2000.

[34] Kotler L, Abramson D, Roe P, and Mather D. ”activesheets: Super-computing with spreadsheets.”. In *Proceedings of High Performance Computing Symposium (HPC01), Advanced Simulation Technologies Conference*, 2001.

[35] Davy Capera, Jean-Pierre George, Marie-Pierre Gleizes, and Pierre Glize. ”the amas theory for complex problem solving based on self-organizing co-operative agents”. In *Proceedings of 12th International Workshop on Enabling Technologies for Collaborative Enterprises (WETICE’03)*, 2003.

[36] O.F. Rana and L. Moreau. ” issues in building agent-based computational grids”. In *UK Multi-Agent Systems Workshop, Oxford*, 2000.

[37] Huahlory Tianfield. ”towards agent based grid resource management”. In *International Conference on Information Technology: Coding and Computing, ITCC 2005*, 2005.

[38] W. Shen, Y. Li, H.H. Ghenniwa, and C. Wang. ”adaptive negotiation for agent-based grid computing.”. 2002.

[39] Fukuda M., Tanaka Y., Suzuki N., Bic L.F., and Kobayashi S. ”a mobile-agent-based pc grid”. In *In proceedings of Autonomic Computing Workshop, 2003*, 2003.

[40] junwei cao, daniel P. Spooner, James D. Turner, Stephen A Jarvis, Darren J. Kerbyson, Subhash Saini, and Graham R. Nudd. ”agent-based resource management for grid computing”. In *2nd IEEE/ACM international symposium on cluster computing and the grid–CCGRID’02*, 2002.

[41] Junwei cao, Darren J. Kerbyson, and Graham R. Nudd. ”use of agent-based service discovery for resource management in metacomputing environments”. In *7th Euro-Par conference, Manchester, UK, Lecture notes in computer science 2150*, 2001.

[42] junwei cao. ”armsim: a modeling and simulation environment for agent-based grid computing”. In *in Transactions of The Society for Modeling and Simulation International, Special Issue on Modeling and Simulation Applications in Cluster and Grid Computing*, 2004.

[43] Junwei Cao, Jarvis S.A., Spooner D.P., Turner J.D.and Kerbyson D.J., and Nudd G.R. ”performance prediction technology for agent-based resource management in grid environments”. In *In Proceedings of International Parallel and Distributed Processing Symposium.,, IPDPS 2002*, 2002.

[44] G. R. Nudd, D. J. Kerbyson, and D. V. Wilcox. ”pace– a toolset for performance prediction of parallel and distributed systems.”. In *International Journal High performance Computing Applications, Special issue on Performance, Modeling*, 2000.