

An Efficient Decentralized Grid Service Advertisement Approach Using Multi-Agent System

Aliaa A.A. Youssif

Department of Computer Science, University of Helwan, Helwan, Egypt

Tel: 20-1-0199-2964 E-mail: aliaay@helwan.edu.eg

Atef Zaki Ghalwash

Department of Computer Science, University of Helwan, Helwan, Egypt

Tel: 20-1-2055-3955 E-mail: ghalwash@cabinet.gov.eg

Mohammed Ezz El Dien (Corresponding author)

HP Enterprise Services, Best Shore Application Services

Commercial Professionals Syndicate Building, 29 Emtedad Ramses Street, Cairo-Egypt, 11471

Tel: 20-1-1860-0224 E-mail: mohamed.ezz011@gmail.com; mohammed.abdelkader@hp.com

Abstract

Grid resource management mainly schedules applications over the available resources. Such scheduling over a large scale distributed environment raises performance issues. Most of the current implementations paid a little consideration of performance issues that are raised at service discovery and advertisement processes.

In this work an agent based resource management system, ARMS, is enhanced with an adaptable decentralized service advertisement strategy to reduce the cost of the advertisement process within ARMS system. A Performance Monitoring and Advisory for ARMS component is implemented to compare the developed strategy with those previously defined in literature. Simulated test results recorded a noticeably better performance for the new strategy since it is more adaptable to continuous changes in both workload and structure of ARMS system.

Keywords: Grid computing, Resource management, performance optimization, Decentralized service advertisement, Agent-based resource management

1. Introduction

A computational grid (Foster & Kesselman, 1999) can be seen as a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities. As a result, grid computing is increasingly viewed as the next phase of distributed computing.

The overall aim of resource management is to efficiently schedule applications that need to utilize the available resources in the meta-computing environment. The management of resources in the grid environment becomes complex as they are geographically distributed, heterogeneous in nature and owned by different individuals/organizations each having their own resource management policies and different access and cost models.

There are different models designed to tackle the problem of resource management, architectural, abstract owner and computational market model (Buyya, Chapin, & DiNucci, 2000). Architectural model is followed in most contemporary systems, while abstract owner is futuristic one that follows order and delivery approach. Economy offers incentives for resource owners as they contribute their resources to grid.

ARMS system (Cao, 2001) follows architectural model designed to provide an agent based solution for resource management component of service layer of grid computing. It builds a tree hierarchy of homogenous agents where each agent is responsible for a grid resource within the hierarchy that can be scaled without limitation.

Several considerations were followed in ARMS (Foster, 2002). First, ARMS design does not state any central point of control, as each agent in the hierarchy plays different roles at the same time.

An agent can be:

- Scheduler as it contains internal logic for scheduling.

- Information service node through ACTs (Agent Capability Table) which act as a database contained within each agent.
- Deployment agent as it can assign and start tasks on local resources.
- Admission control agents as each agent can determine if additional requests will lead to congestion or missing user requirements for a specific request.
- Monitoring agent as it can detect when a task started, finished or failed on a local specific resource (Buyya et al., 2000).

Secondly, every agent has a partial knowledge of nearby agents in the hierarchy, namely its parents and children, stored in its internal database, ACTs (Agent Capability Tables). User requests sent to grid are disclosed to agents unless they contribute in the process of service discovery to satisfy the request. Every agent is only responsible for its local resources, so there is no central point for controlling global grid components.

Finally, ARMS considered user requirements and preferences while scheduling his request as it receives user-cost model that determines execution deadline, minimum memory requirements,...etc. ARMS system has performance overhead resulted from agents interaction at the time of discovery and advertisement process. Some strategies for performance optimization, modelling and simulation, were addressed in (Cao, Kerbyson, & Nudd, 2001).

In this work, a new decentralized service advertisement strategy UST (Updatable Service Lifetime) is developed to adapt to workload changes and dynamic structure of the system. A Performance Monitoring and Advisory, PMA, component is developed to compare UST with other strategies. Recorded results showed that UST strategy has the lowest cost of advertisement process and hence PMA recommended ARMS system to configure each agent with UST while the system is running.

The paper is organized as follow: Section 2 reviews some of Grid projects implementations and their monitoring components. In section 3, UST: a new decentralized service advertisement strategy is presented. In section 4, False Discovery & Cost Index, new performance evaluation metrics to measure performance are presented. In section 5 a case study is explained. Finally, in section 6, the conclusion of the paper is presented.

2. Grid Resource Management

Resource management is a complex task involving, fault tolerance along with scheduling in which resources are allocated and assigned (Kandagatla, 2003).

In Grid environment, many projects addressed resource management as a central component without considering performance monitoring and advising.

2.1 Globus

Is a toolkit developed by Globus Alliance (Ripeanu, Bowman, Chase, Foster, & Milenkovic, 2004). It uses a collection of technologies (recently, it began to adopt web-services based technology) that provides middleware of basic services (like security, resource management, and information services) for grid computing to create VO (Virtual Organization).

Resource Management System of Globus addresses five problems (site autonomy, policy extensibility, heterogeneous substrate, co-allocation and online control) (Czjkowski, et al., 1998). It uses RSL (resource specification language) in communication between components of the system. Resource brokers transform RSL specifications into concrete specification through serialization process. Those transformed specifications are passed to co-allocator (DURCO) which is responsible for coordination, allocation, and management of resources at multiple sites.

Resource Brokers discover resources through querying MDS (Meta Directory Service) which is LDAP based network directory.

Latest Globus versions provide aggregator services to collect recent state information and allow users to query and access the collected information, (Liming, 2008).

Indexes can register to each other in a hierarchical fashion in order to aggregate data at several levels. Indexes are “self-cleaning” and will be removed from the index if it is not refreshed before it expires.

A drawback of Globus is that it depends on push protocol to disseminate resource information or uses a static value (GT 4.0, 2009) to specify when information should be refreshed. In both cases performance issues, which is not monitored by the system, may result from un-necessary updates/communication within the system. Moreover, update mechanisms are not adapted automatically to the system changes as they depend on manual

intervention.

2.2 Legion

Is a grid, object-based, operating system (Grimshaw, Humphrey, & Natraian, 2004)(Chapin, Karpyich, Grimshaw, & Katramtos, 1999). In addition to operating system normal functions, Legion provides numerous other features, such as complexity management, wide-area access, heterogeneity management, multi-language support and legacy application support, which are required in the context of a grid system.

Scheduler in Legion has a hierarchical structure framework; it acts as mediator to find a match between the placement of requests and processors. When a job request is submitted, appropriate scheduler for the job is selected from the framework but co-allocation of resources is not supported.

A draw back of Legion is that it uses periodic pull or push mechanism to update a collection of objects that hold information about resources. The cost of updating such collection from connections with resources to acquire the new status may cause performance overhead which is not monitored or handled in this system.

2.3 Condor

Is another specialized workload management system for compute-intensive jobs. Condor provides set of features, scheduling, co-allocation, pre-emption, fault tolerance and recovery, flocking and remote system calls which are explained in the following:

Condor has a centralized scheduling model with a central manager dedicated to scheduling. Each condor workstation submits the jobs in its local queue to the central scheduler which finds suitable resources for the job execution. The information about suitable available resources to run the job (execution machine information) is returned to the job submission machine. A shadow process is forked on the submission machine for each job. The submission machine becomes responsible for contacting and staging the job on the execution machine and monitoring its progress.

Co-allocation of network and CPU, in Condor, enable users to efficiently use network resources. For example, a customer may request a CPU with at least 8 Mbps peak network capacity with a preference for a smaller number of network hops to the home file system. In pre-emption of running jobs, as if the execution machine decides to withdraw the resources, Condor can preempt the job and reschedule it on another machine, thus providing resource owner autonomy.

Fault tolerance and recovery (Basney, & Livny, 2000) are done transparently where the current state of a job is stored and can be reconstructed if the job is migrated to another machine. Condor has other features such as flocking; that allows jobs to run on machines owned by different organizations, remote system calls, and resource dissemination (that provides an extremely flexible and expressive framework for matching resource requests (jobs) with resource offers (machines)) (Wright, Miller, & Linvny, 2002).

Periodic push updates model, to disseminate resource status in Condor, is not widely accepted since it may cause intrusiveness to the system, in addition to its moderate adaptability to workload changes in the system.

2.4 ARMS

An Agent-Based resource management system, is a hierarchy of homogenous agents that cooperate with each other to provide service advertisements and discovery in order to schedule an application for a grid resource (Cao, 2001).

An agent within ARMS is a layered component (e.g., communication, local management and coordination layers) in which layers cooperate with each other to perform service advertisement and discovery, and application execution.

Information about resources in ARMS is stored within each agent in ACT (Agent Capability Table) database managed by ACT Manager. ACT database contains a variety of records, namely, T_ACT holds local resource information, L_ACT holds service information of lower agents in the hierarchy, G_ACT hold service information of higher agents in the hierarchy, and C_ACT hold cached service information of other agents.

ARMS addresses two key challenges rose from Grid resource management, namely, scalability and adaptability, as the entire system hierarchy is built of identical software agents with the same functionality. These agents are recognized as a powerful high level abstraction for modelling complex software systems.

Another advantage of ARMS system is its Performance Monitoring and Advisory component, PMA, which is a special agent that relies on modelling and simulation to monitor and improve ARMS system performance by advising agents to use efficient strategies.

However ARMS still requires more research to find more adaptable and efficient strategies to enhance its performance. The main challenge in this work is developing a new strategy to lessen the cost of the advertisement process, which is characterized by its adaptability to the system changes and efficiency, as described in detail through the next section.

3. UST (Updatable Service Lifetime) a Decentralized Service Advertisement Strategy

Services offered by agents in ARMS can change overtime which require an advertisement messages to be sent to the interested neighbour agents in order to be aware of such change (ex. an agent can be busy or idle depends on whether the agent is booked or freed from a schedule event)

The process of service advertisement is the main challenge that causes performance overhead resulting from higher cost of advertisement process which requires connections between agents to communicate such change.

ARMS monitors performance overheads by two performance optimization strategies, namely, periodical and event driven strategies. Periodical strategy is a resource dissemination strategy that updates resource information within each agent periodically which makes it not adaptable to workload changes in the system. On the other side, event-driven strategy blindly updates resource information within each agent at the time of service discovery to guarantee that the scheduling decisions are based on the latest resource information. In such case, the strategy causes un-necessary resource dissemination messages.

However both strategies suffer from un-necessary update flaw. Consider, for example, an agent A received a request at time t , for which A can schedule between t_1 & t_2 , where $t_1 > t$. If A was configured with periodical strategy then its ACT (database of neighbored agents) will be updated on a periodical basis away from the dynamicity of the system. On the other hand, if event driven strategy was applied then A will update its ACT, at time t , upon receiving the request causing un-necessary updates in case that A failed to schedule the request due to lack of resources at its ACT.

Moreover, both strategies suffer from performance issues as a result of un-necessary advertisement messages to neighbor agents. The presented strategy UST (Updateable Service Lifetime) adapts both workload and tree structure changes in the system. The strategy produces a self-expiry value UST, attached to each agent. When that value expires, the agent starts updating other nearby agents with its latest status. A new UST value is re-calculated based on three factors: the frequency of changes in the agent's status (SRV_DYN), the relation between the number of agent's children to the overall number of agents in the system (CHD_FCT), and the list of active children which frequently contacted their parent (ACT_CHD_FCT).

3.1 (SRV_DYN) Service Change Dynamicity

In ARMS, the service changes at the start or at the end of scheduling. When a service changes for an agent, an advertisement message is broadcasted to the nearby agents in order to update their local database. Based on historical events occurred to an agent, the expected time for a new change to the agent is calculated using the following forecasting formula:

$$SRV_DYN = a + bx \quad (1)$$

$$\text{Where } a = \bar{y} - b\bar{x} \quad \text{and} \quad b = \frac{\sum (x - \bar{x})(y - \bar{y})}{\sum (x - \bar{x})^2}$$

where \bar{x} is a count number for the previous changes to the agent, and y is the time at which a change occurred, while \bar{x} and \bar{y} are the average values for both x and y .

3.2 (CHD_FCT) Children of Agent

Within ARMS system, agents in the hierarchy have different number of children. So, an agent i , that has a large number of children (CN $_i$), relatively sends more advertisement messages than its counterpart that has a smaller number of children. Hence, the factor (CHD_FCT) will relatively control the next time for an agent to update its children as shown by the following formula:

$$CHD_FCT = \frac{CN_i}{V} \quad (2)$$

Where, CN $_i$ is the number of children for agent i , and V is the average number of children within ARMS tree.

Finally, the new value for UST is calculated using equations (1) and (2):

$$UST = SRV_DYN(1 + CHD_FCT) \quad (3)$$

3.3 (ACT_CHD_FCT) Active Children

In the UST strategy, a parent agent updates only the ACTs of its active children with which it was frequently contacted during the period that preceded the update time as shown in Fig. 1. This feature effectively reduces the number of advertisement messages between an agent and its children. In contrary, in both periodic and event driven strategies an agent updates all its children causing a noticeable increase in the number of advertisement messages.

Fig.1 Agent A contacts its children to update GACTs database.

4. Performance Evaluation Metrics

ARMS monitored and addressed the performance. In addition, (Cao, Kerbyson, & Nudd, 2001) evaluation metrics such as discovery speed (DS) and efficiency (E) are used by the monitoring component (PMA) to select a strategy that enhances performance.

4.1(DS) Discovery Speed

It is defined as the number of connections needed to find a suitable agent for a request,

$$DS = \frac{R}{D} \quad (4)$$

Where R is the total number of requests sent to ARMS system, and D is the total number of connections made between agents at time of service discovery to schedule requests.

4.2(E) Efficiency

It is the total number of connections between agents in the system whether at time of discovery or advertisement processes,

$$E = \frac{R}{(D + A)} \quad (5)$$

Where R is total number of requests sent to ARMS system, and D is the total number of connections made between agents at time of service discovery to schedule requests, and A is the total number of connections made between agents at time of service advertisements.

4.3 False Discovery (FD)

It is a novel defined metric that measures whether a certain strategy causes a database of an agent holds a false or a recent (correct) status of surrounding agents,

$$FD = \frac{Fs}{R} \quad (6)$$

Where Fs is the total number of times an agent detected a false status of other agents in its ACT database, and R is the total number of requests sent to the system.

4.4 Cost Index

It is another novel defined metric that measures how much of connections required to successfully schedule user requests sent to ARMS whether at time of service discovery or service advertisement.

$$C = \frac{(A + D)}{Rs} \quad (7)$$

Where Rs is the total number of successfully scheduled requests, D is the total number of connections made between agents at time of service discovery to schedule requests, and A is the total number of connections made between agents at time of service advertisements.

As was discussed above, the novel UST strategy reduced un-necessary advertisement messages, which greatly lessen the cost of advertisement process which enhanced overall system performance.

Moreover, UST is more adaptable to a variety of changes in the ARMS system such as, workload changes per agent, changes in structure of ARMS tree, and changes in communication between agents.

5. Case Study

PMA (Performance monitoring and advisory) is a special agent that runs in the background of ARMS system to monitor system performance and generate a model for the current status of the system. Based on the generated

model along with the simulation results, PMA selects the best strategy with which ARMS system will be configured.

In this study, the PMA component of ARMS system contains a core and a graphical user interface (GUI) engines which are built based on java and threading models. The core engine manages applications, users, and agents/resources components within the system. It also manages the creation of performance model for the simulation. On the other hand, the GUI engine enables a user or administrator to configure a model, and view the simulated results.

In this paper, a case study is considered to compare the new (UST) strategy with other strategies. The PMA is used to configure a performance model characterized by heavy load of user requests. Three different experiments Exp.1, Exp.2 and Exp.3, configured with different performance strategies, were conducted and the results were recorded, as will be shown through the rest of the following section:

Table.1 shows the performance model settings in which each experiment runs 3000 steps, using 101 agents/resources, 19 applications and 10 users.

Heavy load of user requests are achieved when requests, sent by the users to the system, are every 2 to 3 steps on average (e.g., Min/Max AFR = 2/3). This enables every agent in the system to receive about $(3000/2)/101 = 15$ requests on average during the experiment. Consequently, an agent receives a request every 200 steps. Heavy load in the simulation is achieved by setting the values of application deadline (DE) larger than 200 and application lifetime between 100 to 200 steps.

Table 1 Parameter Settings for the Model

The model has 101 homogenous hierarchical agents, as shown in Fig. 2, each of which represents a high performance computing resource. The Root agent of the hierarchy registers ten other sub_root agents, namely A1, A2, ..., and A10 each of which registers ten other agents.

Fig.2 ARMS tree hierarchy used in the simulation

The model is tested using different combinations of performance strategies, see Table 2. These strategies are used to update the database of every agent within ARMS.

Different combinations of performance optimization strategies for the model

Table.2 Different combinations of performance optimization strategies for the model

Experiments Exp.1, Exp.2, and Exp.3 use and maintain the global database within each agent. In such case, the system suffers from high traffic due to the broadcasted messages from a parent agent to its children. Exp.1 applies the event driven strategy, while Exp.2 considers the new UST strategy. In Exp.3 the UST strategy along with the active children list factor (ACT_CHD_FCT) are applied.

Fig.3 records the average values of Rs, C and FD after running every experiment for 10 times.

Fig.3 Performance Measurement

It can be noticed that success rate (Rs) which is the ratio of successfully scheduled requests to the total number of requests sent to ARMS, is almost the same ratio for all strategies which means that satisfying a user request did not affected much by switching strategies.

The cost index value in event driven strategy was the highest due to the overhead of advertisement messages broadcasted to every child at every time an agent receives a request. Meanwhile, the efficiency got lower in the UST strategy because each agent updated its children depending on the UST value that relatively increased the next update period. In Exp.3 the cost index value recorded the lowest because the update process using the UST strategy was more restricted to the list of active children that needed the least advertisement connections. In contrary, the false discovery increased because the UST strategy does not guarantee optimum coherency of agent's ACT.

The cost index (C), which is an index for the cost of connections required to satisfy a user request, recorded a noticeable decrease with the new strategies. This decrease helps create a negligible effect on the success rate, due to the increased false discovery rate in the proposed strategies, which reserves ARMS performance un-affected

6. Conclusion and Future Work

In this paper, a novel UST strategy was presented and implemented in the ARMS system. Performance monitoring and advisory "PMA" component of an agent based grid resource management system "ARMS" was also implemented to evaluate the proposed decentralized advertisement strategy, updatable service lifetime

"UST". The new UST strategy greatly decreased the total number of advertisement messages between agents causing a noticeable increase in system performance.

Although coherency of agent's database was affected due to the reduction in advertisement messages, test results showed that cost index metric was better with the new strategy.

For future work, ARMS system needs to consider new emerging challenges in grid computing like co-allocation of resources and fault tolerant, interactive and batched applications (Lenica, Ogel, Peshanski, & Briot, 2006). Instead of the first come first served, scheduling algorithms are to be considered, using genetic algorithm (Cao, Spooner, Jarvis, & Nudd, 2005). Moreover, applying strategies on the agent level instead of the system level is a fruitful topic highly recommended for future work

References

- Basney, J., & Livny, M. (2000). Managing Network Resources in Condor. *Proceedings of the Ninth IEEE Symposium on High Performance Distributed Computing*, 298-299.
- Buyya, R., Chapin, S., & DiNucci, D. (2000). Architectural Models for Resource Management in the Grid. *Proceedings of the First IEEE/ACM International Workshop on Grid Computing*, 18-35.
- Cao, J. (2001). Agent-based resource management for grid computing, Ph.D. Dissertation, University of Warwick.
- Cao, J., Kerbyson, D. J., & Nudd, G. R. (2001). Performance Evaluation of an Agent-Based Resource Management Infrastructure for Grid Computing, *1st IEEE/ACM International Symposium on Cluster Computing and the Grid*, Australia, 311-318.
- Cao, J., Spooner, D.P., Jarvis, S.A., & Nudd G.R. (2005). Grid Load Balancing Using Intelligent Agents. *Future Generation Computer Systems*, 21 (1), 135-149.
- Chapin, S., Karpovich, J.F., Grimshaw, A. ,& Katramatos, D. (1999). The Legion resource management system. *Proceedings of the 5th Workshop on Job Scheduling Strategies for Parallel Processing*, 162-178.
- Czajkowski, K., Foster, I., Karonis, N., Kesselman, C., Martin, S., Smith, W., & Tuecke, S. (1998). A resource management architecture for metacomputing systems, *In The 4th Workshop on Job Scheduling Strategies for Parallel Processing*. 62–82.
- D. Wright, K. Miller, & M. Livny (Eds.) (2002). Condor - A Distributed Job Scheduler. In T. Tannenbaum, *Beowulf Cluster Computing with Linux*. The MIT Press.
- Foster, I. (2002). What is the Grid? A Three Point Checklist, *Grid Today*, 1 (6).
- Grimshaw, A. A., Humphrey, M. A., & Natrajan, A. (2004). A philosophical and technical comparison of Legion and Globus *IBM Journal of Research and Development* , 48 (2) , 233-254.
- GT 4.0 WS MDS Aggregator Framework: Configuring the Subscription Aggregator Source. (2009). [Online] Available: http://www-unix.globus.org/toolkit/docs/4.0/info/aggregator/Subscription_Aggregator_Source.html.
- I. Foster, & C. Kesselman (Eds.). (1999). Computational Grids. *The Grid: Blueprint for a New Computing Infrastructure*. :Morgan-Kaufman.
- Kandagatla, C. (2003). Survey and Taxonomy of Grid Resource Management Systems, University of Texas, Austin. [Online] Available: <http://www.cs.utexas.edu/users/browne/cs395f2003/projects/KandagatlaReport.pdf>.
- Lenica, A., Ogel, F., Peshanski, F., Briot, J-P. (2006). Agent-based grid resource management, *In The 2006 International Conference on Computational Science International Workshop on Grid Computing Security and Resource Management*.
- Liming, L. (2008). Globus Primer: An Introduction to Globus Software ,*Open Source Grid and Cluster Conference*.
- Ripeanu, M., Bowman, M., Chase, J.S, Foster, I., & Milenkovic, M. (2004). Globus and Planet Lab resource management solutions compared. *Proceedings of the 13th IEEE International Symposium on High performance Distributed Computing*, 246-255.

Table 1. Parameter settings for the model

Parameters	Settings
Exp.Lifetime	3000
Min/Max AFR	2/3
DE	201
Min/Max AE	100/200
No.Agents	101
No.Resources	101
No.Users	10
No.Applications	19

This table contain parameter settings used in experiments which is described within text above.

Table 2. Different combinations of performance optimization strategies for the model

Combinations	Performance Strategies		
	Cash	Local	Global
Exp.1	N.A	N.A	Event Driven
Exp.2	N.A	N.A	UST
Exp.3	N.A	N.A	UST+AC

This table contain experiments performance optimization strategies applied which is described within text above.

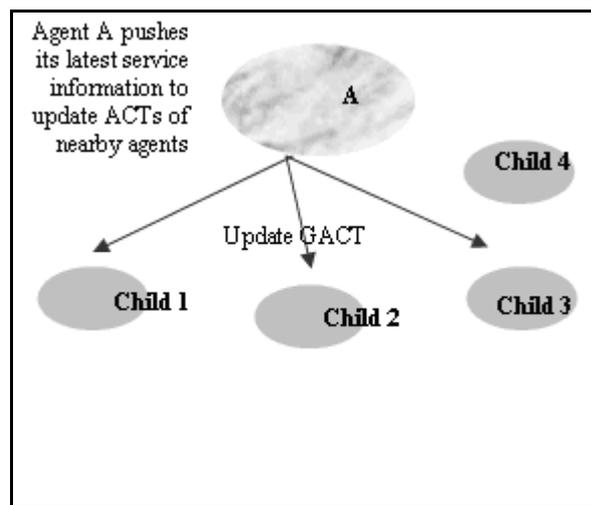


Figure 1. Agent A contacts its children to update their database

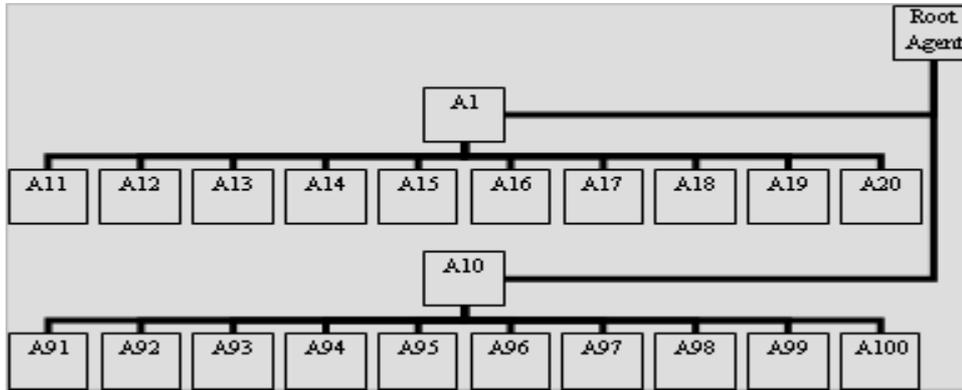


Figure 2. ARMS tree hierarchy used in the simulation

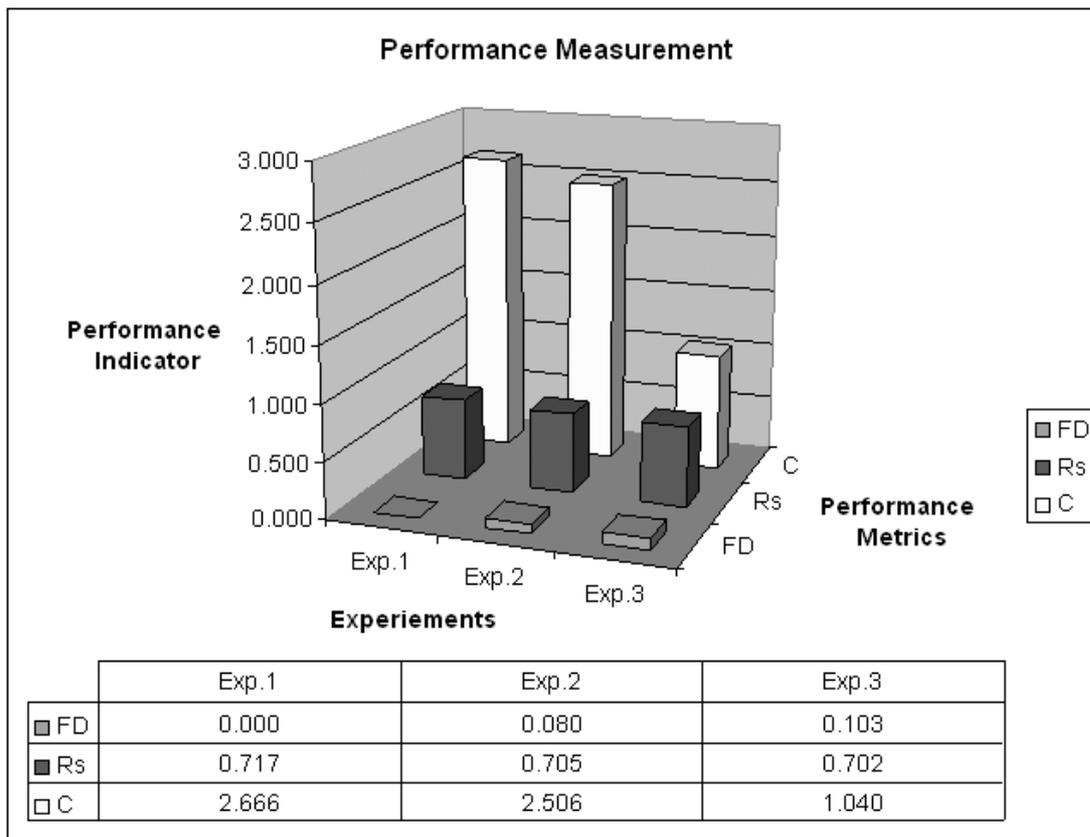


Figure 3. Performance Measurement