DYNAMIC COALITION IN AGENT AWARE ADHOC VIRTUAL P2P INTERCONNECT GRID COMPUTING SYSTEM – A^{3P}VIGRID

Avinash Shankar, Chattrakul Sombattheera, Aneesh Krishna, Aditya Ghose, Philip Ogunbona
Decision Systems Lab [DSL]
School of IT and Computer Science,Univerity of Wollongong, NSW-2522, Australia

Keywords: Dynamic Coalition, Grid, P2P, Agents, A^{3p}viGRID.

Abstract: Artificial Intelligence and High performance Grid computing systems are two different fields of technologies that have been much researched upon and are as old as the development of personal computers and its related technologies. Technologies such as Agent based systems and the Semantic Grid have evolved with the use of Artificially Intelligent techniques such as the Turing system of intelligence measurement. Similarly Peer to Peer computing and Supercomputing Grids have evolved from distributed and middleware clustering systems such as Condor. In this paper a new architectural schematic is proposed where technologies such as Agents, P2P computing and Adhoc systems are incorporated in a Grid Computing framework for the optimal Job processing and delivery to the end user transparently. Applying Dynamic Coalition techniques in Agent based Grid computing systems has been a meagrely researched Area. The proposed system A^{3p}viGrid tends to deploy a service oriented schematic that enables users to search for, negotiate using agents, do remote Job Processing and use resources without the need for a resource discovery model in place that is commonly used in current day high performance Grid systems.

1 INTRODUCTION

Applying Agent based Concepts to Grid Computing brings a new era of Computing systems that learn and act according to the needs of the end user. Here we give a brief introduction to the various techniques and technologies that will be used in designing the A^{3p}viGRID System.

"Grid computing can be defined as a decentralized, pervasive, transparent and simplistic schematic which is governed by no central body that uses different types of heterogeneous computing platforms / technologies and their resources over a decentralized interconnected network such as the Internet."(Source:http://www.huihoo.com/grid/grid_ computing_info_centre.htm, Last visited 5/01/2005)

The driving force behind Grid and High performance computing research seems to be a desire to harness and share idle computing resources across organizations world over. Grid technology has gotten a lot of attention from both academic and commercial environments. The academic community is seeking better ways to tackle High performance problems, and the commercial industries interest's lies in more e-client usage of commodity hardware to reduce operational costs (or) replace expensive specialized computers. Good examples of High-performance Cluster or Grid Computing systems would be the TeraGrid project (Source: www.teragrid.org), UK E-Science project (Source:www.escience-grid.org.uk), etc, that performs in the measure of Teraflops in terms of processing speeds and Terabytes of storage space. Such vast resources should be made available and be used to the fullest extent possible. The A^{3p}viGrid System tries to provide such a Framework for effective utilization of such systems to bring service oriented Grid computing to the desktop user irrespective of the location and topology. Peer-to-Peer computing is the concept of sharing resources in ways of a give and take policy. P2P computing is a subset of High performance Grids or Cluster computing. The technology was an offspring of many different technologies and has become famous after the success of High performance computing. A peer is a single entity (or) node that shares its resources with other peers (or) nodes directly (or) indirectly connected to it. Early peer-to-peer systems were primarily made to enable users to share, often illegal, files easily and in public. Early systems either had scalability problems or were not pure peer-to-peer systems. Now there are more serious research and product development efforts on peer-to-peer technologies. Sun's JXTA (Source: www.jxta.org) effort is one important arena today. In the open source community peer-to-peer systems also seems to have matured and issues on scalability have been addressed in some systems, e.g. (Condor / Condor - G, www.cs.wisc.edu/condor/ and Bearshare, Kazza, Gnutella, www.zeropaid.com). Peer-to-peer systems also have a basic goal of utilizing distributed resources and providing services to the participating nodes. Today services have primarily been file sharing systems, but Sun's JXTA framework shows peer-to-peer computing is not limited to just that. Especially Sun Microsystems JXTA and P2P-JXTA (Source: www.openp2p.com) show that peer-to-peer computing actually has many similarities with Grid and agent based systems.

2 COALITIONS IN AGENT-BASED GRIDS

A coalition, in the context of agent-based systems, is usually defined to be a group of agents that come together to solve a common task or achieve a common objective. Coalition has its roots from Game theory where players {agents} form groups and plot a strategy for winning a Game. In general with respect to Agent based Systems and Game Theory, coalition formation occurs on the fly where agents tend to form groups to achieve a common goal such as Job processing or maximizing their utility value. Here with respect to Grid computing we en-route and define two new concepts called Static coalition and dynamic coalition in agents based grid systems. Two categories of coalitions are of interest here: static and dynamic.

Static coalitions are typically formed on the basis of more persistent common goals and tasks, and are less likely to change from problem to problem.

Dynamic coalitions, on the other hand, are groupings that are formed to address the needs of a specific task or common objective. Once these tasks are completed, or the common objectives met, dynamic coalitions tend to disband, and re-form in different ways. Here in A^{3p}viGRID we use dynamic coalition formation techniques for effective job processing and aggregation of resources available. Our fundamental premise is that coalition mechanisms add value in the context of agent-based grids, for the following reasons:

• Coalitions of peers can reduce the communication overhead involved in executing complex tasks and services which require the use of multiple peers.

• Coalitions of peers can better enable better matching between the requirements of tasks/services and the infrastructure that is made available to execute these. For instance, an appropriate coalition formation mechanism can put together a collection of peers with similar platforms and QoS characteristics that are best suited for a given task.

• Coalition formation mechanisms can be used to optimize complex trade-offs between the objectives of maximizing the utility of the service requester(s) and the service providers. For example a service requestor could be maximizing its payoff for the given task by being an intermediary service provider that outsources its job to third party agents thus maximizing his individual utility value.

• Coalition formation mechanisms can economically increase system throughput as a whole. After some negotiation among agents, tasks will be allocated to appropriate coalitions who can execute them with minimal costs and time. Thus agents seem to be better off. A good example of this would be the formation of coalition among agent in a local Linux cluster where the maximum payoff is achievable with minimal communication costs.

3 THE A^{3p}viGRID ARCHITECTURE

The A^{3p}viGRID Architecture is primarily focused on providing a Peer to Peer based Adhoc Multi Agent Environment that enables users to remotely join the A^{3p}viGRID system to search for new serial/parallel programs and submit jobs for job processing. A very good example would be P2P file-sharing systems such as Bear share, Kazza, Gnutella etc that use directory services to register the location of peers along with the information of the list of files used for sharing.

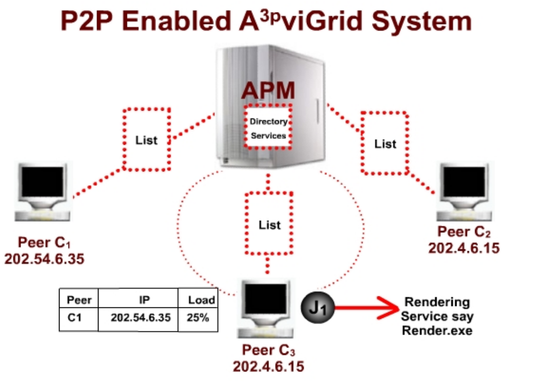


Figure 1: Peer to Peer A^{3p}viGrid System

3.1 Agent Based Peer Manager [APM]

The Agent based Peer manager is an intelligent agent that handles negotiating and registering of services based on directory services. Although it poses a threat as a centralized scheme, it's primarily used for discovery and communications between agents and their respective peers based on a light weight directory services model such as LDAP (Source: Open LDAP directory service protocol - Source: www.openldap.org). The primary role of the APM is to register services of agents based on commonality and self-interests factors and also help in discovery and formation of Coalitions in agents based on the commonality of their services rendered. The APM also tends to act as a broker or middleware for all agents associated with it. An economic brokering system could be incorporated based on the usage of the APM which renders as a service to self-interested agents and agent based systems in its locality.

3.2 Electing a RSD Leader

Let there be a set of agents {A, B...E} located in nearby locations. We assume that these agents form a Regional service domain [RSD] based on a set of attributes that are adhered to by the agents and the service requestor. That is all the agents available in a common region are grouped by a leader called the Regional leader. So each region can have a number of RSD's based on the how close agents are to each other. Once the RSD is formed each agent votes for a leader for representing the agents in a particular region. Here the agents say {A, B, C, D, E} will bid on the Job J1 along with A the elected leader by forming new coalitions. The least loaded Coalition having an optimal turn around time will get the job based on the negotiated payoff value. R after negotiations decides upon selecting the best coalition for the job and offloads the job for job processing. The primary use of the RSD leader is to represent a set of closely-knit agents to minimize the communication costs between the agents and the APM.

4 COALITION FORMATION METHODOLOGY

This architecture allows agents to dynamically form coalitions in order to bid for, executing tasks. Firstly the job description will be broadcast to all agents of the RSD. Agents exchange information with respect to their present ability to perform the tasks. The information is composed of a set of attributes that explains the current state of the agents. There is a utility function that assigns a utility to the coalition. The utility function takes the attributes of all the coalition members and computes a utility that indicates which coalitions are suitable for the tasks. Agents then try to form coalitions based on this indicative utility. Agents decide to join a coalition if they are offered a payoff, which is not less than a payoff threshold computed individually.

4.1 Ranking by Indicative Utility

The information is a set of attributes or properties that affect the way jobs are to be processed by agents and their respective peers. Attributes in agent-based grids can be the load of machine, turn around time of a process, latency, QoS factor, bandwidth requirement, distance, etc. Each agent maintains an attribute table, which contains all other agents attributes it collects after the exchanging of information. Based on the table shown in Figure 4 we can assume that each job is associated with an attributed value with respect to the properties associated by that attribute. Based on this we can compute the utility value for each agent by. The figure 4 indicates that Agents A, B and C have A1 …. A3 attributes which are needed for computing the utility value of each potential coalition based on the attributes of the job. Each agent can then compute all the possible coalitions it may form. These coalitions are ones that each agent would like to propose to other agents. The agent uses each coalition in this list as a proposal for forming potential coalitions. It then computes the utility value for each coalition. The utility value indicates how well a coalition can perform a task. The utility can be computed by applying a LESS or a MORE value with respect to the properties of the attribute. For example if we were computing the Load of individual agents then its useful to have a less loaded system than that of the jobs attribute value. So a coalition formation will lead to an average of loads to compute the average value of the potential coalition and a LESS is specified for computing the validity of the potential coalition formation between say agents A and B. The agent then ranks coalitions based on their utility values in a ranking table called RTable in descending order. In the case that the multiple coalitions yield the same utility value, then smaller coalitions are preferred. Based on the ranking table RTable the best coalition formation strategy is deployed. The price for executing the job successfully will be equally distributed among the coalition members as their individual payoffs.

4.2 Coalition Formation Protocol and Decision Making

Step 1: R is a remote node that needs to offload its job say J1. R sends a Job Description message say <ID, Job desc, T, A1, A2...An> to the APM for the job J1. A set of attributes such as Latency, Time of Completion, Payoff, etc is defined as A1, A2, A3 …An.

Step 2: The APM in turn sends a message to the Regional service Domain [RSD] leader which in turn advertises the job J1 to its local agents for dynamic coalition formation.

Step 3: Let us take an agent A that is interested in computing the job J1.

Step 4: To compute potentially good coalitions, agent A sends messages to all other agents in the RSD querying for the attributes as shown in Fig 4.

Step 5: Each agent uses an internal table to compute the indicative utility and ranks the coalitions in descending order in a ranking table RTable. Now agents have the capability to decide which agents are suitable for coalition formation.

Step 6: Each agent now computes individual payoff for each coalition member in all coalitions by distributing the price of the job equally.

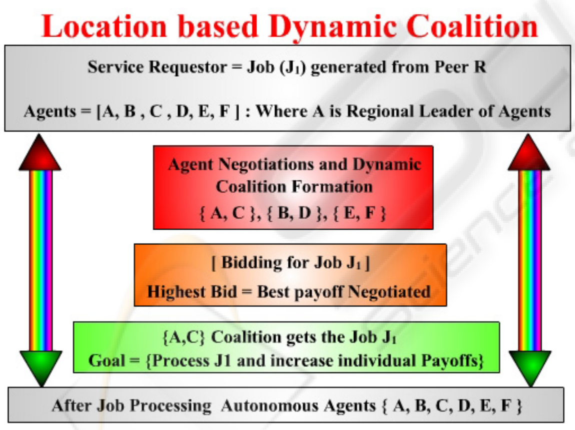

Location based Dynamic Coalition
Service Requestor = Job (J_1) generated from Peer R
Agents = [A, B , C , D, E, F] : Where A is Regional Leader of Agents
Agent Negotiations and Dynamic Coalition Formation
{ A, C }, { B, D }, { E, F }
[Bidding for Job J_1]
Highest Bid = Best payoff Negotiated
{A,C} Coalition gets the Job J_1
Goal = {Process J1 and increase individual Payoffs}
After Job Processing Autonomous Agents { A, B, C, D, E, F }

Figure 2: Location Based Dynamic Coalition

Step 7: Each agent selects all the coalition members from its top proposal based on its ranking table RTable and then sends out a message <ID, coalition, payoff vector> (ID is identity of the requestor while payoff vector specifies the individual payoff for each coalition member) to all agents in that proposal.

Step 8: Each receiving agent compares its individual payoffs, Xi, in the proposed payoff vectors and compares them to the payoff Xi* in its top individual proposal. If

Xi >= Xi*

then agents reply with ACK, or else with a NO-ACK signal.

Step 9: If the Head of the coalition receives ACK signal from all the coalition members, it declares that a coalition is formed and the task is bided for by the new coalition that is formed.

Step 10: And if the requestor receives a NO-ACK, it decreases its payoff Ui* in the top payoff vector by a certain value, i.e. ∅ then increases the payoff Uj* for all other coalition member by ∅ / (|S|-1), where |S| is the size of the coalition. If the new Ui* is less than the equivalent payoff in the second payoff vector then delete the top proposal Repeat step 7.

Step 11: The newly formed Coalitions then bid for the job J1 by negotiating with the originator R to obtain the job J1. The negotiation can involve comparison of attributes such as payoff value, goals of agents, etc, to negotiate for J1.

Step 12: R then compares and contrasts the attributes of the coalitions and then offloads to the best negotiated coalition formation available for its job namely J1.

Step 13: As soon the job J1 is allocated to a specific coalition, all the other coalitions are dispersed to form autonomous agents once more.

Step 14: The selected coalition then performs the Job J1 and on completion sends a finished FACK status to the originator R of job J1; it then receives its payoff for the job and disperses the coalition to join the host of autonomous agents in the RSD to become autonomous once more.

Step 15: Continue back to Step 1 for originator of new jobs and coalitions respectively.

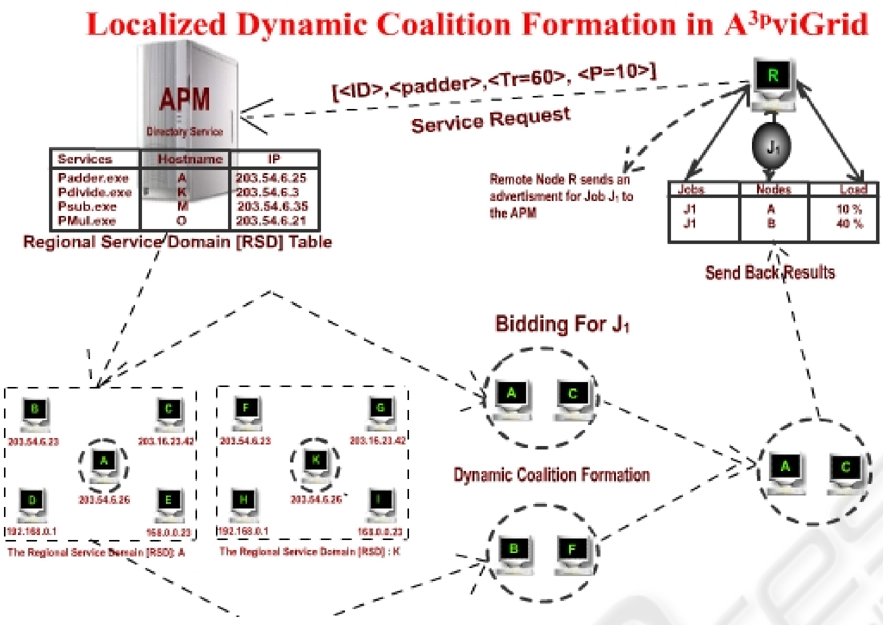
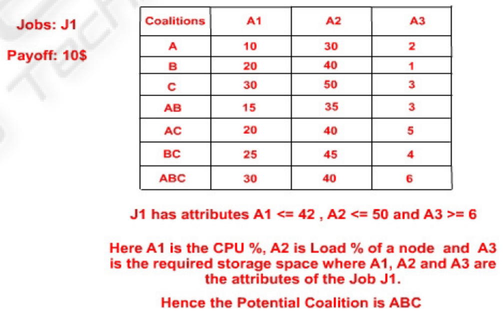

Localized Dynamic Coalition Formation in A3pviGrid
APM
Directory Service
[<ID>,<padder>,<Tr=60>, <P=10>]
Service Request
Services | Hostname | IP
Padder.exe | A | 203.54.6.25
Pdivide.exe | K | 203.54.6.3
Psub.exe | M | 203.54.6.35
PMul.exe | O | 203.54.6.21
Regional Service Domain [RSD] Table
R
J1
Remote Node R sends an advertisment for Job J1 to the APM
Jobs | Nodes | Load
J1 | A | 10 %
J1 | B | 40 %
Send Back Results
Bidding For J1
Dynamic Coalition Formation
The Regional Service Domain [RSD]: A
The Regional Service Domain [RSD] : K

Figure 3: Applying Dynamic Coalition in A3pviGrid Schematic

5 EXAMPLE OF APPLYING DYNAMIC COALITION TO THE A^{3P}viGrid SYSTEM.

As shown in Fig 3, there is a remote node R with IP address 203.54.6.35 and has a Job processing requirement called J1. The APM stores information about the locations of the regional agents and there services. An example of an attribute can be the turnaround time, Latency, Payoff, etc. When node R wants to execute the job J1, it authenticates itself with the APM by sending its IP address say 203.54.6.35; sends an advertisement about a remote Job J1 that has a requirement for a service called padder.exe to the APM. Based on the location of R and the services requested, the APM passes the request to the respective regional service domains [RSD] leader A. Attributes are properties that are adhered to by the agents based on self-interests. So each agent is specifically interested in job processing based on a set of special value attributes that plays an important role in the way jobs are processed. Attributes can be anything from resource requirements, latency to trust issues with respect to agents and the originator of the job. As shown in Figure 4 we can see that A1…..A3 are the attributes values of CPU %, Load % and Storage. Now each agent based on the properties of its environment will agree or are bound to satisfy certain attribute values. As shown in the figure, let A, B and C be three agents having different attribute values. Whenever an agent satisfies an attribute value, the agent puts its calculated value to compute utility value of the potential coalition based on the available attributes of its immediate environment. Attributes based on the job description tend to form the basis for agents to form coalitions. Our example denotes attribute values to be properties which affect the way jobs are being processed by agents and their respective peers. Attributes with respect to agent based Grids can be the Load of a machine {A1}, turnaround time of a process {A2}, Latency {A3}, QOS factors {A4},


Jobs: J1

Payoff: 10$

Coalitions	A1	A2	A3
A	10	30	2
B	20	40	1
C	30	50	3
AB	15	35	3
AC	20	40	5
BC	25	45	4
ABC	30	40	6

J1 has attributes A1 <= 42 , A2 <= 50 and A3 >= 6

Here A1 is the CPU %, A2 is Load % of a node and A3 is the required storage space where A1, A2 and A3 are the attributes of the Job J1.

Hence the Potential Coalition is ABC

Figure 4: Computing Potential Coalitions

Bandwidth requirement {A5}, distance {A6}, etc. As shown in Figure 4 let job J1 that originated from remote node R have the following three attributes that needs to be satisfied in order to do job processing. Now we can see that A1…..A3 are the attributes having CPU %, Load % and Storage computed approximately for processing job J1. The attributes become a requirement here for job processing. Each agent say A, B, C compute their individual values and the appropriate potential coalitions by comparing which potential coalitions formations tries to satisfy the attributes of the Job J1 to the fullest. As we can A, B and C first check to see if they satisfy the requirements of the Job J1 independently before proposals are sent for coalition formation. Then a set of possible coalitions proposals are formulated and computed based on the attributes available. Let us take one of the attributes say Load % to examine how the computation works. For every value computed we see if a LESS or MORE value is required by that attribute. That is Job J1 here needs the load of a system to be less than or equal to 50% then we compute every possible coalitions and do an average of the loads for all the coalitions to see if they satisfy the load value say <= 50%. As shown all the possible coalitions are calculated and compared to see which of the coalitions are <= 50% loaded. Similarly we need storage space of 6 megabytes, and that becomes a MORE as we need to have a system which can offer 6 or more number of megabytes for storage. For attributes such as load or CPU we have to calculate the average by dividing the sum with the number of agents forming the coalition. After calculating the LESS and MORE values of all the possible coalitions the agent needs to decide which coalition formation satisfies all or most of the requirements of the Job J1. As we can see the coalition formation of ABC seems to match all the criteria's of the Job J1 and the next best coalition that satisfies most of the requirements is AC that is ranked in the agents table as a potential coalition in case ABC cannot be formed. Each coalition then selects a leader to represent the coalition to optimize on communications with the RSD and the originator R. The leaders associated with every coalition formed starts to contact R and bids for the Job J1. Hence we have computed the utility value for potential coalitions and the best coalition available to the agent is used for bidding. The list of potential coalitions after accepting the proposals will form coalition to bid for the Job J1 and after negotiations with respect to payoffs the agents will perform the Job J1. Coalitions are dispersed once the selected coalition is allocated for job processing and when the Job is finished the Agents break the Coalition formation and become autonomous as they join the RSD again. Two or more Coalitions can also be maintained by the remote peer R to achieve fault tolerance with respect to job processing. Thus dynamic coalition in agents tries to provide a good solution to Agent based service oriented Grid computing systems where best possible coalitions are computed to provide optimal job processing among agents.

6 LIMITATIONS

The primary limitations posed by dynamic coalition in agents is that coalition formation might take more time than expected due to the negotiations among the agents. The other problem associated with agents is the possibility of not bidding for a Job and doing Job processing at all due to their autonomous properties such as different goals or dissimilarities observed in agents.

7 CONCLUSION

Applying Dynamic Coalition methodologies to Multi Agent based High performance Grid computing systems has lead to a new perspective to the usage of intelligent agents in Grid computing systems. The A^{3p}viGrid tries to provide solutions to minimizing the usage of Resource discovery models and process analyzers by utilizing directory services such as the APM and the effective usage of dynamic coalition schemes. Local coalitions are dynamically formed by intelligent agents having a commonality of goals based on a service oriented schematic.

REFERENCES

Andrew S. Tanenbaum and Robbert Van Renesse; ACM Comput. Surv. 17, 4 (Dec. 1985), Distributed Operating Systems, Pages 419 - 470.

Avinash Shankar, Daniel Saffioti, Ian Piper, Ashwin Shankar, June 21st – 24th 2004, "Service Oriented Web Based Meta Resource Sharing Platform - The CBWeB

Portal ". The 2005 International Conference on Parallel and Distributed Processing Techniques and Applications, CSREA Press, Pages 744-749

Justin R.D. Dyson, Nathan E. Griffiths, Hélène N. Lim Choi Keung, Stephen A. Jarvis, Graham R. Nudd, Trust in Agents for Grid Computing.

Condor Cycle Stealing Technology – Condor / Condor - G, www.cs.wisc.edu/condor/, Last accessed 6/3/2004

File Sharing applications - Bearshare, Kazza, Gnutella, www.zeropaid.com, Last visited 5/01/2005

Grid Computing Definition - Source: http://www.huihoo.com/grid/grid_computing_info_centre.htm, Last accessed: 5/01/2005

Open LDAP directory service protocol - Source: www.openldap.org, Last accessed: 24/6/2004

P2PJXTA Framework - www.openp2p.com Last visited 5/01/2005

Sun Microsystems JXTA Framework - www.jxta.org Last accessed: 5/01/2005

The TeraGrid Project - www.teragrid.org

UK's E-Science Project - Source: www.escience-grid.org.uk, Last accessed: 5/01/2005