A Resource Look up Strategy for Distributed Computing

F. AGOSTARO, A. GENCO, S. SORCE
DINFO - Dipartimento di Ingegneria Informatica
Università degli Studi di Palermo
Viale delle Scienze, edificio 6 – 90128 Palermo
ITALY

Abstract: - In this paper we present an implementation of a mobile agent-based tool for resource discovery within a distributed computing virtual community. Such a tool enables community members to suitably arrange their own distributed virtual machine, using resources available within the community. Mobile Agents are used to detect network addresses and computers that will be part of a virtual machine, according to a resource discovery strategy which allows members to the use of remote hardware and software resources.

Keywords: - Mobile agents, distributed computing, resource discovery.

1 Introduction
The grid approach can enable us to configure a distributed machine by the use of computing resources remotely located on a network. This topic is widely discussed in literature, and many related problems have been solved by various software packages, like PVM [1], Linda [2] or P4 [3]. There are several running projects which aim at arranging a framework for managing high performance computation oriented services on distributed systems. The goal of a distributed machine configuration is achieved by means of coordinated and controlled resource sharing among the members of a dynamic multi-institutional virtual community. [4]
Members agree on which resources will be shared and which members of the community will be enabled to access resources, thus defining a set of sharing rules and permissions. Some tools are available which provide basic facilities for network management, authentication, communication, and data access. [5] Apart the software tool employed to set up and use the virtual machine, one of the most relevant problem members have to deal with is grid resources lookup. Members with computational needs and those with available resources, even if somehow connected, do not know each other’s availability or needs.
As discussed in [6], we propose a resource discovery system in a virtual community Mobile Agent based [7]. A Distributed Computing Virtual Community (hereafter DCVC) is set up to allow its members to share computing resources.

According to the discovery strategy described in section 3, mobile agents are used to search among a set of addresses, being instructed to single out the ones of those sites fitting the member requirements in terms of hardware and software resources.
Users who want to become members of the Community, need to install an agent-based software on their machines to let their resources to be shared and enable them to look up and use community resources. To this end, Community members publish their availability or needs and interact according to a peer to peer resource provision protocol.

2 DCVC components
The main DCVC components are (fig.1):
• the client Mobile Agent: this is the community key-component and acts for members who need computing resources. It pursues a resource discovery mission, according to the strategy described in the next section;
• the server application. This runs on the member machines whose resources are available. It receives Mobile Agents incoming from other sites;
• a distributed database to store and manage the site addresses of available resources. This is implemented by means of a Friend User Table stored in each member machine. Such a distributed solution extends our previous model, in which we used a centralized database and a different resource discovery strategy. [8]
3 Resource Discovery strategy

Once a Community starts, the steps which lead to members cooperation for distributed virtual machine setup are:

1) A user starts acting as a Community Member (CM) by launching a copy of the AgentServer and waiting for client agents. This way users become active members of the Community. Each member has a Friend User Table, in which a set of couples <resource, address> are listed. Notice that when a member registers within a Community, the Friend User Table is empty.

2) When a member shows his need for computing resources, an agent is generated and instructed to start the resource discovery process.

- First the Agent searches for the requested resources among the addresses stored in its Friend User Table. If the search succeeds, the agent migrates to the site where resource are located and there acts as a broker to start cooperation.

Now the agent steps into the network to test the current availability of the collected servers. This is necessary because one or more servers which have notified their availability could be not available anymore, or even be disconnected. For each IP address the agent tests the server matching to client requirements. If test succeeds, the agent books the server for a time interval large enough to end the test round. If test fails, it proceeds its round with the next server. Once the test round ended, if the bookings placed by the agent allow the configuration required by the client, they will be confirmed, and the agent will go back to the client with the IP addresses list. This are used to configure the distributed virtual machine. Some other details are reported which can be used to plan a good load balancing strategy. If the bookings do not fit client requirements, member will be notified of a request fault.

- If the requested resources are not found in its Friend User Table, the agent migrates to a Default Friend User (DFU), which is selected when a member registers within a Community. If the requested resources are found there, the agent migrates to the sites where resource are located and acts as described above. At its coming back, the agent address list is used to setup distributed virtual machine and to update the client’s Friend User Table too.

If the resources are not found within the Default Friend User, the agent continue its discovery process by recursive invocation of Default Friend Users, until reaching the community bounds, where a community manager can be invoked to manage the client.
query. This, which probably requires a manual insertion of new addresses, aims at community enhancement, thus filling the resource gap and providing the last querying member with the address of the requested resource. This entities collaboration strategy is shown in fig. 2.

Notice that an Agent can take some addresses from its Friend User Table, some others from the Default Friend User and some others by querying the community manager. At the end of its search, the agent returns an address list of available and suitable machines, or a message telling the unavailability of what requested. In the first case, if the address list has been obtained by recursive invocation of Default Friend User, the list is used to update the client’s Friend User Table.

Furthermore we want to point out that an agent is the key component of the system, because it takes care of the server availability test and of the decision whether to include them into the distributed virtual machine or not, according to member requirements.

4 Software agent-based tool
As a feasible implementation of our discovery strategy model [6], here we present a software tool, based on Mobile Agents technology, along with a graphical user interface to setup servers and instruct agents. The tool aims at finding IP addresses of machines whose resources are shared by their owners.

Users who want to become members of the community have to install the agent-based software to allow their machines to act as client or server, according to their needs or availability (fig. 3).

Members who want to act as server select the “Supply” button. The “Server setup” window is then opened allowing to insert all technical data of the local machine (fig. 4).

When all specification are inserted, by clicking on “OK” the application send the information to the database and the Agent Server is started waiting for client Agents.

Users who want to act as client, select the “Request” button on the community setup window (fig. 3). The “Agent setup” window is then opened to insert all specifications to instruct the client agent (fig. 5).

Once all data is inserted, by clicking on the “OK” button the mobile agent is launched to make its search among the addresses listed in the database. The agent “owner” waits for agent coming back. At the end of its search, the agent return the address list of available and suitable machines, or a message telling the unavailability of suitable machines.
4.1 Tool operating test

In our operating test, we used the tool to arrange a distributed machine made up of two hosts, each complying with requirements shown in fig. 5. The community is composed following machines:
1. Pentium III 500 MHz, 256 Mb RAM, 60 Gb Hard Disk, static IP address, running a copy of the Agent Server on Windows XP Pro O.S.
2. Pentium II 333 MHz, 256 Mb RAM, 20 Gb Hard Disk, dynamic IP address, running a copy of the Agent Server on Windows 98 SE O.S. and acting as DFU of previous machine.
3. Pentium III 1000 MHz, 256 Mb RAM, 40 Gb Hard disk, static IP address, running a copy of the Agent Server on Windows 2000 Pro O.S. and acting as DFU of previous machine.
4. Pentium III 1000 MHz, 512 Mb RAM, 20 Gb Hard Disk, dynamic IP address, running the client software on Windows XP Pro O.S.

At the end of the test, the tool generates a log in which the agent operation and migration are shown (fig. 6).

5 Conclusions

In this work we presented a mobile agent-based tool for hardware and software resource discovery in a distributed computing virtual community. The proposed strategy turned out to be efficient, fault tolerant, and capable of dynamically adapting to environment changes.

Future works on these topics will include security and authentication features and enhanced booked server management.

6 References

