Intelligent Mobile Agents for Heterogeneous Devices in Cloud Computing.

Dr.K.P.Kaliyamurthie[1], A.Antonidoss[2]

1.Professor and Head, Dept. of IT, Bharath University, Chennai-600 073.
kpkaliyamurthie@gmail.com
2.Asst. Prof.(SG), Dept. of CSE,Tagore Engineering College,Chennai,Tamilnadu,India
aro.antoni@gmail.com

Abstract- Cloud computing enables highly scalable services to be easily consumed over the Internet on an as-needed basis. A software agent offers a new computing paradigm in which a program, can suspend its execution on a host computer and can transfer to another agent enabled computer on network so that it could run there. Such mobile agents are used when there is limited capacity in computers. But at times, the systems could not be able to run even the mobile agents. This paper proposes an approach to execute mobile agents on any sort of systems even with limited capacity of Cloud. A platform that supports various mobile agents depending upon the capacity of the system will be developed. The mobile clients are differentiated into two types depending upon their capacity and the corresponding mobile agent is chosen for the traversal.

Keywords – Cloud, Mobile agents, Mobile clients..

1 Introduction

CLOUD computing presents a new way to supplement the current consumption and delivery model for IT services based on the Internet, by providing for dynamically scalable and often virtualized resources as a service over the Internet. To date, there are a number of notable commercial and individual cloud computing services, including Amazon, Google, Microsoft, Yahoo, and Salesforce. Details of the services provided are abstracted from the users who no longer need to be experts of technology infrastructure. Moreover, users may not know the machines which actually process and host their data. While enjoying the convenience brought by this new technology, users also start worrying about losing control of their own data. The data processed on clouds are often outsourced, leading to a number of issues related to accountability, including the handling of personally identifiable information. Such fears are becoming a significant barrier to the wide adoption of cloud services [30].

Mobile agents traverse between various mobile clients for the execution of certain applications. These mobile agents are used in order to overcome the limited capability problem of mobile clients.

Even with mobile agents, a new problem arose. The mobile clients were unable to run even the mobile agents.
The mobile agents usually carry code to execute the application and the data to be executed. This load is minimized for certain mobile clients so that they may execute mobile agents easily.

When the configuration of client is suitable enough to send mobile agent containing both code and data to server, then the same will be sent to server. In the specialized search of a large free-text database, it may be more efficient to move the program to the database server rather than move large amounts of data to the client program. The result will be made available at the client. When the client configuration is minimal, so that mobile agent containing both code and data cannot be sent to the
Server, then only the state of the client will be sent to server. The code will be stored already in the server.

Platform designed will obtain the configuration of client associated. According to the configuration the mobile agents will be sent, either with state alone or with code also.

2 Existing works

Accessing a mobile agent is possible only with a mobile client having sophisticated and powerful resources. Sending a mobile agent to another computer to do the data collection and computation can save battery power. This also means that heavy computations that will take long time on a mobile device can be executed at a more powerful computer with more memory, faster CPU and without any power limitation. In addition, problems with slow and error-prone network connections can be reduced by dispatching the agent to another machine, and get the agent with the result back when finished. Most mobile devices will consume a lot of battery power when transferring data over a network a long period of time. Mobile agents can reduce these problems.

Any application, to be run using mobile agents is done as follows: The data and required code of the application will be put together as mobile agents and will be sent to the server. It will be executed at the server and the results will be available at the client.

The problem faced here was, executing mobile agents at the client became difficult because of very limited capacity.

3 Motivations

According to the survey in [8], Aglets, Voyager and Grasshopper were among the four best mobile agent platforms; their ranking was: 1) Grasshopper, 2) Jumping Beans, 3) Aglets and 4) Voyager. Tryllian, JADE, Tracy and SPRINGS is also available. The last version of Tryllian has been released recently as open source, JADE is a very popular platform for the development of agent-

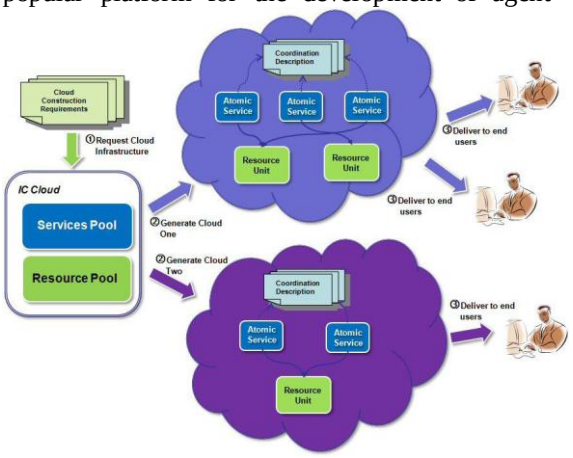

Figure 1 Structure of Cloud

Based systems, the development of Tracy has led to the publication of the most recent book on mobile agents (since some years ago), and SPRINGS has been developed very recently and offers a promising scalability.

Cloud Computing

Acknowledging that *"one model fits all"* is unlikely to be appropriate for the development of cloud computing.

 Cloud computing is an aggregation of many existing computing technologies. Different service providers and vendors have their own understandings about cloud computing so are their designs and system implementations. It seems impossible or improper to debate which one is the best design and it's difficult for early cloud computing developers and/or adopters to try out all the possibilities.

The second argument is that cloud computing platforms are still fast evolving. New features and functions will emerge while new requirements are proposing. It's unrealistic to assume that a pre-designed system will fit all the future requirements.

 The last but not the least, cloud computing system is only emerged gradually from the existing computing system as technology evolves. Therefore, it's impossible to clearly define the boundary between these two. In the others words, the new and cloud based systems will incorporate many features, components and data of the old systems and it is hard to pre-define rigidly what the new system's shape would be beforehand.

All these observations suggest that we require a framework where technical design options, system construction approaches can be systematically studied. The need for such a systematical study of cloud system becomes even more important when the dynamical adaptive resource provision becomes desirable in a cloud system. That is, to be scalable and sustainable, a cloud computing system needs a form of meta-scalability which enables the system to evolve along the complex environment in which it operates. A system is said to be meta-level scalable when it has the ability to adapt to any change in an evolving environment. In the case of cloud computing, a cloud platform is said to be meta-level scalable when it is designed in such a way that it can be reconfigured by its users or by the system itself dynamically to satisfy specific needs, emerging activities, means of coordination and rules. This means that cloud platform designer should adopt a different approach. They should separately focus on the development of *atomic services* of cloud computing systems and the *composition mechanisms* of them.

4 Proposed System

In this paper, the proposal is made basically to eliminate the limitations with minimal resource mobile agents. Till now the mobile clients with minimal resources cannot access mobile agents and produce results successfully. To overcome this a platform is been designed so that it works in two different ways.

4.1 Differentiating mobile clients The platform proposed for design will get the configuration of the client. Using it, the mobile clients will be differentiated into thick and thin mobile clients.

4.1.1 Thick mobile client When the client is said to have enough resources, it is called as thick mobile client. When such a client is in use, both the code and data is combined in mobile agent and sent for execution. The result will be then sent back to the mobile client. Thus the power of mobile device is saved by making the execution done at some other powerful system. The mobile agent used here is called Thick Mobile Agent.

4.1.2 Thin mobile client: When the configuration of the mobile client is not enough for a particular application then it is called as Thin mobile client. When such a client is in use, only the data called as state will be sent to the server by using mobile agents. The code for the application will be stored already on the server. Thus, even if the mobile client does not support the required resources, the application could be executed. This method avoids the excessive resource usage for dispatching the

code to the server. Tailoring for specific devices can also be done to improve the usability of the agent clients based on the screen size, graphical capacities and data entry capabilities of the devices

5 Architecture

5.1 Thick client

For the thick client the database containing the code (in a database) and the data to be executed will be available.

```
CREATE TABLE `t_mobile_agents_info` (
  `f_deviceName` varchar(30) default NULL,
  `f_deviceOS` varchar(30) default NULL,
  `f_cpuName` varchar(30) default NULL,
  `f_cpuType` varchar(30) default NULL,
  `f_cpuSpeed` varchar(30) default NULL,
  `f_totalMemory` varchar(30) default NULL,
  `f_availableMemory` varchar(30) default NULL,
  `f_secondaryMemory` varchar(30) default NULL,
  `f_javaedition1` varchar(30) default NULL,
  `f_javaversion1` varchar(30) default NULL,
  `f_javaedition2` varchar(30) default NULL,
  `f_javaversion2` varchar(30) default NULL,
  `f_javaedition3` varchar(30) default NULL,
  `f_javaversion3` varchar(30) default NULL,
  `f_downloadType` varchar(30) default NULL,
  `f_downloadAddress` varchar(30) default NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Fig:0 Describes the Mobile Agent Database

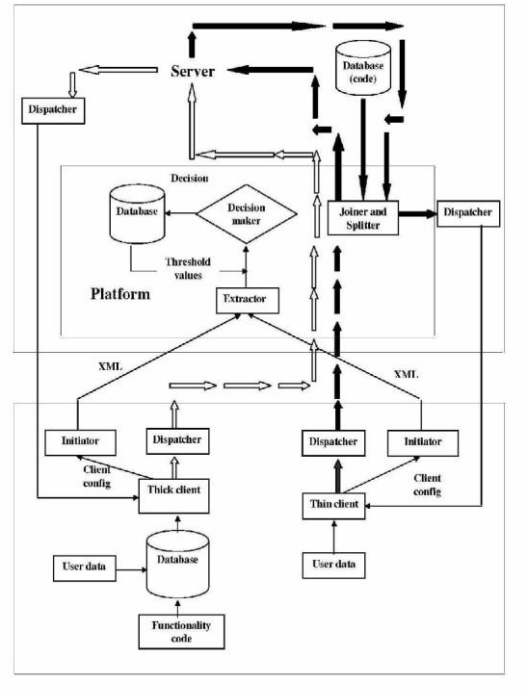

Fig: 1 Architecture

5.2. Thin client

For the thin client, database will not be available. The code will be present only in the server. Hence the data alone should be sent.

The thin clients also follow the same traversal path, except that the code to execute the application will be present in the server. In case of thin clients, the dispatcher has some more work of combining and splitting the code and data.

5.3 Platform

The platform will contain the decision maker. The threshold values for decision making will be present in the database already. This will be specified depending upon the efficiency of the clients.

The decision will be made based on the capabilities like memory, processor, java support etc. Extractor is used to extract the client capabilities obtained from the initiator.

A central part of the architecture is the agent system repository where agent information and agents can be stored along with client information such as client device capabilities. We have defined a thin agent client that only deal with the agent state, where the code of the agent will be stored on the agent server. The thin agent client contains code for transmitting and receiving agent data and a simple GUI for manipulating this data. For devices with sufficient memory and CPU, mobile agents will run locally on the device. Above the repository the general agent server services are located that controls the essentials agent services such as registration of agents, locating agents, management of agent lifetimes (initiate agents, clone agents, kill agents) etc. The rest of the architecture is split into two main parts; one for thick agent clients, and one for thin agent clients. The thick agent client is able to execute mobile agents locally, while the thin agent client only is able to manipulate the agent data (state).

There is only one part that distinguishes the architecture of the thick and thin agent client; the Agent joiner/splitter. The *Agent joiner/splitter* makes it possible to split the data and code of the mobile agents before dispatching the agent to the thin client. This makes it possible to access mobile agents on less capable devices. The *Agent initiator* is responsible for initiating the agent clients first time, or reconfiguring the agent client based on changes in the mobile agent software or hardware/software changes in the client. Although there are two Agent initiators drawn in the figure 2, only one such service is running (same for thick and thin clients). The last component of the architecture is the *Mobile Agent Dispatcher* which makes it possible dispatch agents between clients and server. For thin clients the Agent joiner/splitter is used to split the agent before dispatching to a

client, and to assemble the agent when received from the client.

5.4 The Agent Initiator

When a client connects to the agent server for the first time, it must state the client device capabilities in terms of CPU speed, execution memory, storage, and Java Virtual Machine edition and version. Although a client device has enough memory and fast enough CPU, there could e.g. be no Java virtual machine available that support serialization for the device. This is the reason that Java Virtual machine is also a part of the client capabilities. The capabilities are described in XML. Figure 6.1 shows an example of a description for a client running on a Palm Tungsten T PDA. The client

```
01: <ClientCapabilities>
02: <Device>
03: <Name>Palm Tungsten T</Name>
04: <OS>PalmOS 5.0</OS>
05: </Device>
06: <CPU>
07: <Name>OMAP1510</Name>
08: <Type>32bits</Type>
09: <Speed>175MHz</Speed>
10: </CPU>
11: <Memory>
12: <RAM>16MB</RAM>
13: <Available>5MB</Available>
14: <Secondary>64MB</Secondary>
15: </Memory>
16: <Java>
17: <VirtualMachine>
18: <Edition>MicroEdition</Edition>
19: <Version>1.0</Version>
20: </VirtualMachine>
21: <VirtualMachine>
22: <Edition>SuperWaba</Edition>
23: <Version>1.3</Version>
24: </VirtualMachine>
25: </Java>
26: <Download>
27: <Type>Email</Type>
28: <AddInfo>james@bond.007</AddInfo>
29: </Download>
30: </ClientCapabilities>
```

When a mobile agent client is initialized the first time, it follows the following process:

5.4.1. Install agent initiator: The agent initiator is a simple Java program that must be installed on the mobile device before installing the agent client. This program is used for extracting device capabilities on the client. The devices capabilities that cannot automatically be detected must be entered manually by the user. In addition, the agent initiator contains software for communicating with an agent server.

5.4.2. Configure the agent initiator: The user should look through the device capabilities detected, and add missing, or change faulty information.

5.4.3. Initiate the agent initiator: The agent initiator sends the device capability information to the agent server.

5.4.4. Install agent client: Based on the device capabilities, a suited agent client will be sent to the device and then installed. The agent client can be sent directly using the agent initiator, or indirectly using transport channels such as email or HTTP-download.

If any characteristics of the client are changed (e.g. a new virtual machine for Java has been released), a new initializing process is initiated. Also if there are major changes of the agent system itself, the agent server may initiate a client update process (similar to the process shown above). The reason for this is that the new version of the mobile agent system could be more CPU and/or memory efficient than the previous one, making it possible to run the full mobile agent system on clients that previously were not powerful enough.

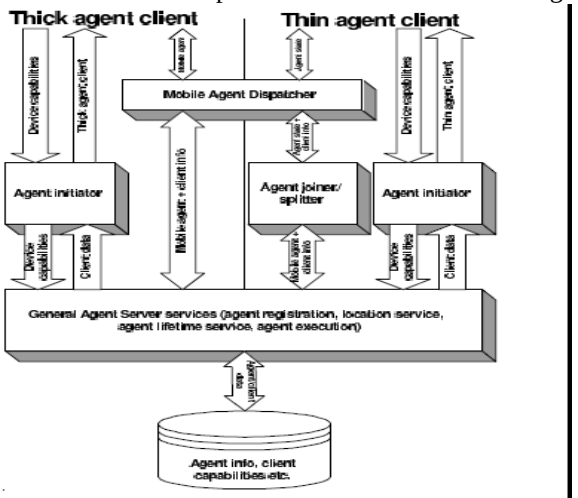

Figure 2. A logical view of the mobile agent architecture

5.5 The Agent Joiner/Splitter

For clients that cannot run the mobile agents Locally (because of client capabilities), we have designed an Agent joiner/splitter for allowing these clients to access the agents anyway. This is done by separating the state (or data) of the agent with the code of the agent. This means that the thin agent client only receives the data part of the mobile agent while the code part resides on agent server. In addition, the client on the mobile device has a graphical user interface (GUI) making it possible to instruct the agents. When the user decides to send an agent from the mobile device, the state of the agent is transmitted to an agent server. This means that both the code and the state of the agent are

joined on the agent server. Figure 4 illustrates the process view of a migration of an agent (client application) from the mobile device to the agent server. When an agent migrates to another agent

server or thick agent client, both the code and the state are transmitted (as for normal mobile agent systems).

6 Issues to be concentrated

6.1 Agent Separation:

Since the mobile devices still have minimal resources to run mobile agents, it was never an option to run an entire agent on a mobile device. To overcome this problem, the agent application was separated in two parts (code and data), with only the data residing on the mobile device along with the user interface classes.

6.2 Writing to the Memory:

Since an agent can be configured to operate for a long period of time, it was necessary to save the agent ids in the non-volatile memory of the mobile device. This makes it possible to exit the application, and recall the same agent the next time the application is started. J2ME's Record Management System (RMS) was used to store the agent id.

7 Related Works

7.1 JADE

(Java Agent DEvelopment Framework) is a software Framework fully implemented in Java language. It simplifies the implementation of multi-agent systems through a middle-ware that complies with the FIPA specifications and through a set of graphical tools that supports the debugging and deployment phases. The agent platform can be distributed across machines (which not even need to share the same OS) and the configuration can be controlled via a remote GUI. The configuration can be even changed at run-time by moving agents from one machine to another one, as and when required. Although, it is possible to run JADE on all devices that support J2SE, there are currently only few mobile devices that are able to that. This means that JADE only partially solves the problem of running mobile agents on mobile devices, since only the most powerful ones are capable

7.2 Mobile Information Agents

The aim of project MIA (Mobile Information Agents) is to develop an intelligent information system, which puts *information of local relevance* from the World Wide Web into the hands of a mobile user. All information needed for such a system is available on the World Wide Web somewhere. The MIA-project implements a system that collects this information, and gives it into the hand of the user. The MIA system also needs to know where the user is. This can be done by using a GPS-receiver or the mobile phone can be tracked in the mobile network.

7.3 Personalized Information Retrieval Service

The design of a mobile-agent system that provides a mobile user with a personalized information retrieval service and we describe the implementation of the infrastructure for such a system. This "Personal Agent System" gathers information from the Internet and uses context-aware mechanisms to manage the information according to a mobile user's needs and preferences. The user's schedule and location are the context indicators in this system. These indicators are critical in ensuring that users obtain only the information they want, receive information in a form that is most useful for viewing on their mobile device, and is notified of new information in a minimally intrusive manner. The system incorporates a rule-based learning system to enhance the personalization achieved by the system.

8 Conclusions

Mobile devices even with very limited capability will be able to execute any sort of application very easily. All the mobile devices may not have enough capability to run certain large applications, wherein this mobile agent platform will be much helpful. Not all of these limitations such as the screen size, network bandwidth, battery capacity, and input devices, have much influence on a mobile agent application. There are other more noticeable constraints that limit an agent application. However, these constraints are possible to overcome by this platform.

9 References

[1] Comparison and Performance Evaluation of Mobile Agent Platforms Trillo, R.; Ilarri, S.; Mena, E.; Autonomic and Autonomous Systems, 2007. ICAS07. Third International Conference on 19-25 June 2007 Page(s):41 – 41 Digital Object Identifier 10.1109/CONIELECOMP.2007.66

[2] On the Performance of Distributed Search by Mobile Agents. A. Mawlood-Yunis, A. Nayak, D. Nussbaum and N. Santoro. Proc. 1st International Workshop on Mobility Aware Technologies & Applications, 2004.

[3] Mobile-Agent versus Client/Server Performance: Scalability in an Information- Retrieval. R. S. Gray, D. Kotz, R. A. Peterson, J. Barton, D. Chacn, P. Gerken, M. Hofmann, J. Bradshaw, M. Breedy, R. Jeffers, and N. Suri. Task. Proc. 5th International Conference on Mobile Agents, pp. 229-243 , 2002.

[4] A mobile agent platform for active telecommunication. C. B¨aumer and T. Magedanz. Grasshopper. In Third International Workshop Intelligent Agents for Telecommunication Applications (IATA'99), Stockholm, Sweden, pages 19–32, August 1999.

[5] Trusted Java Virtual Machine IBM, http://www.almaden.ibm. com/cs/projects/jvm/, 2012..

[6] A FIPA Platform for Handheld and Mobile Devices. Bergenti, Federico, Poggi, and Agostino. http://leap.crm-paris.com/public/docs/ATAL2001 .pdf,2001.

[7] Mobile Information Agents for the WWW. MIA Research

Group. MIA - http://www.unikoblenz. de/Obthomas/MIA TML/, 2002.

[8] A survey and evaluation of agent platforms. K. Ludwig, A. Josef, W. E. Edgar, S. Wolfgang, and G. Franz. In *Second International Workshop on Infrastructure for Agents, Multi-Agent Systems, and Scalable Multi-agent Systems, at the 5th International Conference on Autonomous Agents, Montreal, Canada*. ACM Press, May 2001.

[9] Design and implementation of a hybrid intelligent and mobile agent platform Su-Zhi Zhang; Chun-Lin Li; Zheng-Ding Lu Machine Learning and Cybernetics, 2003 International Conference on Volume 4, Issue , 2-5 Nov. 2003 Page(s): 2025 - 2030 Vol.4 Digital Object Identifier 10.1109/ICMLC.2003.1259836

[10] TILAB. JADE (Java Agent Development Framework). http://sharon.cselt.it/projects/jade,2002.