Effectiveness and Suitability of Mobile Agents for Distributed Computing Applications

(Case studies on distributed sorting & searching using IBM – Aglets Workbench)

S. R. Mangalwede {1} , K.K.Tangod{2} ,U.P.Kulkarni{3} ,Prof A R Yardi{4}
{1} shrinivas_mangalwede@yahoo.com {2} kktbgm@yahoo.com
Department of CSE/ISE,Gogte Institute of Techonology, Belgaum-590008, India.
{3}upkulkarni@yahoo.com; Department of CSE, SDM Engineering college, Dharwad-580002, India.
{4}prof_ard@yahoo.com; Professor, E&C Department, Walchand Engineering College, Sangli, India.

Abstract

Autonomous agents are mobile applications that are launched on data warehouses to perform specific queries or to search for patterns in data. With mobile agent technology, a developer is not bound by more traditional distributed computing models, for example, two-tier client-server models, three-tier middleware-oriented models, and so on. The Mobile Agent technology facilitates flexible code distribution, i.e., factoring functionality and communications using the most appropriate strategies for the task at hand. Moreover, this functionality can move to the most appropriate network host, as needed, on demand. In this paper, we investigate the suitability of mobile agent technology for distributed computing applications. Our discussion and observations are based on two practical case studies i.e. distributed sorting and searching using IBM-ASDK (Aglet Software Development Kit) framework for employing mobile agents.

Keywords: Mobile Agent, Distributed Computing, Data Mining, Sequential Itinerary, Parallel Execution.

1. Introduction

Let *t* be a complex computation to be performed and assuming the task is inherently distributed/parallel in nature. Assuming there are *n* mutually distrustful participants available that have access to only a limited amount of resources, the task *t* can be broken down into sub tasks { t_1, t_2, t_3,..., t_n } between these n participants which will then perform the task independently from each other.

Alternatively the same task can be broken down into { t_1, t_2, t_3,..., t_m } where *m* is an integer constant (m>=n). We can then assign each of these tasks to *n* participants in a round robin approach (for m>n).

However, we can consolidate the results obtained by these participants to obtain the solution for the originally stated complex computation.

The division and assignment of these sub tasks can be done based on the existing processing load on individual participant.

In this paper, we investigate the suitability of the use of mobile agent technology and its performance behavior for distributed computing applications. For case study we have taken two problems viz., distributed sorting and distributed searching.

The paper is organized as follows: In section 2 we discuss briefly about the mobile agent technology. In Section 3 we focus on the motivation behind this work and its relevance with regard to the background work carried out by the authors [1] and others [2][3]. In Section 4, we introduce the problem definitions and the test bed conditions on which the problems were solved. In section 5, we focus on the observations of the results we obtained. More precisely, we discuss the readings we obtained using mobile agents and compare them with the results obtained through more traditional approaches. Finally in Section 6, we summarize and conclude the main outcomes of this work.

2. Mobile Agent Technology and its relevance to Distributed computing

A mobile agent can be defined as [4]:

An agent is a software object that is situated within an execution environment; possesses the following mandatory properties:

- *Reactive;* senses changes in the environment and acts in accordance with those changes;
- *Autonomous;* has control over its own actions;
- *Goal-driven;* is pro-active;

- *Temporally continuous:* executes continuously;

and may possess one or more of the following orthogonal properties:

- *Communicative;* can communicate with other agents;
- *Mobile ;*can travel from one host to another;
- *Learning;* adapts in accordance with previous experience;
- *Believable;* appears believable to the end-user.

In today's network-oriented computing environments it's common to find applications that distribute their workload over multiple hosts, for example, network-aware PCs.
In the past, distributed application components were often implemented in traditional languages such as C or C++ as executable files or shared object-code libraries. Implementing application components as machine-dependent binary modules is problematic because it forces developers and system administrators to deal with a whole range of issues:

- Portability of application components
- Interoperability of components developed with different programming languages
- Data communication/exchange across multiple machine architectures
- Rogue modifications to binary modules that compromise security

More recently, Java has become tremendously popular as a development language for distributed application components for several reasons:

- Highly modular, dynamic, class-oriented compilation units
- Portability of compiled code (class files)
- On-demand loading of functionality
- JDBC-related portability across database vendors
- Fine-grained and very configurable security control
- Built-in support for low-level network programming
- URL-related classes and interfaces
- Extensible URL support for custom protocols

Mobile agent technologies support common solutions to implementing cooperating, distributed application components. Just as a graphical toolkit is useful for developing multiple graphical applications, a mobile agent framework facilitates the development of multiple distributed applications, as well as the ongoing enhancement of existing distributed applications.
The ASDK framework is a lightweight mobile agent technology from IBM's Tokyo Research Laboratory. With aglets, it's straightforward to develop stand-alone distributed applications that are independent of large-scale application server frameworks. That is, application components can be truly distributed, and not dependent on a centralized application server that provides a host of distributed middleware services.

3. Motivations and Background Work

"Cumbersome computations can often be broken into discrete units for distribution among a pool of servers or processors. Each of these discrete units can be assigned to an agent, which is then dispatched to an "agent farm," where the work is actually performed. Upon completion, each agent can return home and the results can be aggregated and summarized."
-- Stefan Marti, MIT Media Lab. in the article "Applications for Aglets"

Given that mobile agents can move from node to node and can spawn subagents, one potential use of mobile agent technology is as a way to administer a parallel processing job. If a computation requires so much CPU time as to require breaking up across multiple processors, an infrastructure of mobile agent hosts could be an easy way to get the processes out there.

Distributed computing became a field of intense study as the result of computer hardware miniaturization and advances in networking technologies. Distributed computing aims to unify multiple networked machines to let them share information or other resources, and encompasses multimedia systems, client-server systems, parallel computing, Web programming, mobile agents, and so on.

Given the ever-increasing amount of information available on the Internet and other networks, the activity of collecting information from a network often amounts to searching through vast amounts of data for a few relevant pieces of information

Passivity, timeliness, and untrusted collaborators are all features of a system that provides distributed searching services. The agents must run without human intervention, notify their owners immediately upon finding items of interest, and execute in a domain of distrust -- the Internet. And fortunately this

does appear to be a system that fits well within the mobile agent model.
The experimental results [1] to validate the suitability of mobile agents for applications that involve different issues like:

- Network Latency and Round Trip Time
- Transfer of huge data
- Tasks that involve complex computations to be performed on huge data sets
- Tasks that are inherently parallel and distributed in nature.

The observations [1] are as under:

- As the size of the aglet increases, the network latency and round trip time also increases. However, after some size as shown in the charts above, the latency and round trip time varies minimally even if the size of the aglet is increased.
- Execution environment fails to dispatch the aglet when the size of the aglet reached 6291455 Bytes. The reason for that being the constraints of the Java Virtual Machine on the size of the object.
- Aglets are not suitable for huge data transfers due to the fact that when an aglet is dispatched, cloned, or deactivated, it is marshaled into a byte array, then unmarshalled from it later.
- Amount of time required for computation is reduced considerably when the job is dispatched to carry upon the remote data rather than bringing the data and performing the computations.

4. Problem Definitions and Test Bed Conditions

In This paper we have done two experiments to investigate the performance behavior of aglets for applications that involve distributed computing. The problem definitions are as follows:

Case I:
In this experiment we investigate the suitability of mobile agents for distributed sorting as compared to traditional sorting on a single machine. We use quicksort algorithm on client machines running Tahiti server. At the originating host, we use mergesort algorithm to aggregate the sub sets to obtain a sorted file. We also compare the performance of distributed sorting using aglets with the traditional approach to sorting. As a test we have taken up the task of sorting large amount of data ranging between 1MB-8MB and the time taken for this task is recorded. It is compared with the time for sorting the same data on a single machine.
The processing of testing is done in the following phases:

Phase A:

In this phase the distribution of the load is done in a round robin order i.e. a fixed amount of data is sent to the clients equally with remaining data load distributed in a round robin approach.

Step A1:

In this step the data is divided between 2 clients. Two readings are taken for each set of data.

Step A2:

In this step the data is divided amongst 5 clients. Two readings are taken for each set of data.

Phase B:

In this phase the entire data load is equally divided amongst all clients.

Step B1:

In this step the data is divided between 2 clients. Two readings are taken for each set of data.

Step B2:

In this step the data is divided amongst 5 clients. Two readings are taken for each set of data.

Case II
The purpose of this study is to investigate the effect of using mobile agents in searching for a key element and observe its performance behavior as compared to the traditional searching approach. The application we have chosen is a simple but complete one. It activates most mechanism under the hood of aglet. It also illustrates the suitability of aglet-based distributed computing. As a test we have taken up the task of searching for a key element in a large amount of data about 10MB and the time taken for this task is recorded. We use the Parallel Execution of Aglets Model and compare that with Sequential Itinerary Model.

Phase A:

This phase is based on the parallel execution of Aglets. In this phase the key to be searched is distributed to all the registered client machines. The data, which is present on all the clients, is searched

for the key element. Five recordings are made for each search and the time taken for each of the above cases is noted.

Phase B:

This phase is based on the Sequential Itinerary Model. In this phase the key to be searched is sent to the first registered client machine. On the client machine, the key element is searched in the data present locally. If key element is found on the client, the child aglet reports success; otherwise it moves on to the next registered client in a predetermined format and performs the search. Thus it implements a sequential itinerary model. Five recordings are made for each search and the time taken for each of the above cases is noted.

The following are the test bed conditions used to test the application:

Network specifications: Ethernet based LAN with 10/100 Mbps Intranet and 1Mbps Leased Line Internet connectivity.
Hardware specifications:
Processor: Pentium IV @1.4GHz Memory: 128MB of RAM
Operating System: Windows 2000
[Version 5.00.2195]
Software specifications:
Java runtime environment [Version JDK1.3]
IBM-ASDK2.0.1 with Tahiti Server Environment

5. Experimental Observations

Experimental Observations for verifying the suitability of mobile agents for distributed computing – Case I (Distributed Sorting)

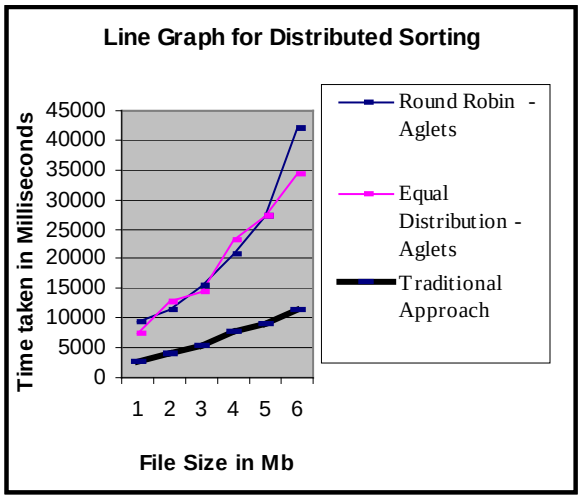
The experiment was conducted for file size varying from 1Mb to 6Mb with first setup containing two (02) client machines.
The same experiment was conducted for file size varying from 1Mb to 6Mb with setup containing five (05) client machines.
Two trials are performed for each case.

In order to measure the suitability of mobile agents for this experiment i.e., distributed sorting, we also wrote a stand-alone Java program that implements quicksort algorithm. The program was executed under different file size inputs ranging from 1Mb to 6Mb. The recordings for this are as follows:

File size (Mb)	Quick Sort on single machine (Traditional Approach)		
	Time Taken in milliseconds		
	Trial 1	Trial 2	Average
1	2537	2671	**2604**
2	3957	4097	**4027**
3	5237	5320	**5279**
4	7540	7695	**7618**
5	9005	9115	**9060**
6	11595	11453	**11524**

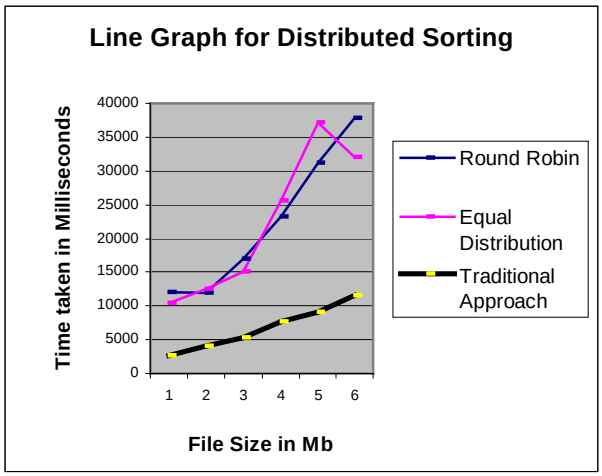
Using Mobile Agents to perform distributed sorting.
Number of Client Machines: 02

File Size	Round Robin	Equal Distribution
1	9446	7511
2	11521	12617
3	15375	14331
4	20702	23016
5	27052	27263
6	41899	34282



Using Mobile Agents to perform distributed sorting.
Number of Clients Machines: 05.
Time in milliseconds.

File Size	Round Robin	Equal Distribution
1	12050	10407
2	11948	12453
3	16953	15006
4	23227	25641
5	31178	37180
6	37808	32047



The observations of the readings for all the phases when analyzed show the following results:

1. Implementation of regular sorting using a traditional approach i.e., sorting performed on a stand-alone machine performs better compared to distributing the task of sorting among many hosts using mobile agents. As we ca see from the results, traditional approach is faster than distributing the task using mobile agents.
2. As the amount of data increases the performance of the mobile agents improves, but only to a certain level. When the amount of data, (subset) the aglet is carrying to a destination host for sorting, exceeds a certain limit, the aglets collapse thus not completing the task in hand. This is in continuance with the results we obtained from experiment number 1.
3. Network latency plays a very crucial role in the distributed application. The process of distributing the data is affected by the network load as also the object serialization mechanism of Java.

Experimental Observations for verifying the suitability of mobile agents for distributed computing – Case II (Distributed Searching)

Phase 1: Distributed Searching based on the Parallel Execution of the Aglets Model.
Phase 2: Searching based on the Sequential Itinerary Model.
Note: M-# in the following table → If the key element is present on the machine #.

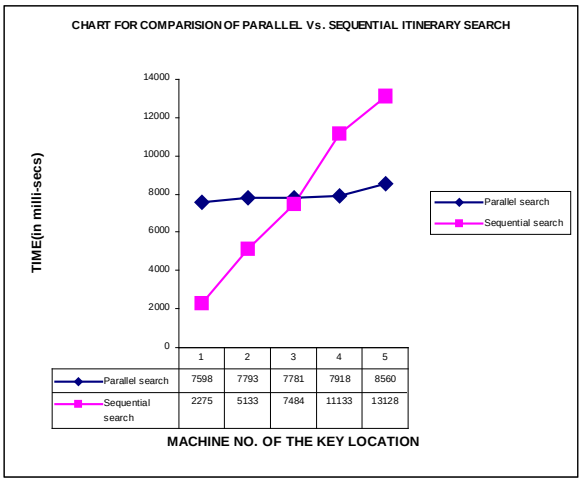
Experimental Results for Distributed Searching (Parallel Execution of Aglets):

	M-1	M-2	M-3	M-4	M-5
Trial1-msec	7450	7690	7797	7690	8250
Trial2-msec	7745	7895	7765	8145	8870
Average (msec)	7598	7793	7781	7918	8560

Experimental Results for Searching (Sequential Itinerary Model):

The following chart shows the comparison of the performance behavior of distributed searching using parallel execution of aglets with the searching using sequential itinerary model.

	M-1	M-2	M-3	M-4	M-5
Trial1-msec	1775	4785	7031	10750	12950
Trial2-msec	2775	5480	7937	11516	13305
Average (msec)	2275	5133	7484	11133	13128



The observations of the readings for all the phases when analyzed show the following results:

- The use of mobile agent technology for searching of the key element on a large distributed data is effective.
- The searching for the key element using parallel execution of aglets on all the registered client machines shows a uniform behavior for all the search cases.
- The searching for the key sequentially on the registered client machines i.e., sequential itinerary model takes lesser time for searching if the search key is found early in the sequence of registered client machines, where as it takes more time for machines that come later in the sequence, thus showing an inefficient behavior.

6. Conclusion

The conclusions that can be derived from the our study are as follows:

1. The use of mobile agents for distributing the task in this case (sorting large amount of data) is not efficient compared to a centralized traditional application, as mobile agents are better when they move to a remote node with only the job rather than the data.
2. Distribution of the task using mobile agents is not suitable for time-critical applications, but can be used optimally to distribute task to idle systems on a network thus increasing the throughput of the systems.
3. Handling large amount of data at once is a bottleneck on both centralized as well as distributed applications since it is constrained by the Java Runtime Environment.
4. From the experimental observations, we can also conclude that the mobile agent technology is a viable and suitable approach for applications that involve distributed computing, if we send the computation to the locality of data, the mobile agent technology is an efficient and suitable approach for distributed computing applications.
5. From the applications that we have chosen and from the results that we obtained it is clear that parallel execution of aglets for searching shows a uniform performance in execution time.
6. The Autonomous Agents for distributed computing and distributed data mining applications shows a viable and more efficient approach as compared to traditional client/server based approaches.

Bibliography

[1] Prof. A R Yardi, U.P.Kulkarni, and S.R.Mangalwede. *A Mobile Agent Technology for Internet Applications.* International Conference on Computers, Controls, and Communication, INCON–CCC 2004, Chennai, India.Pg.474-479.

[2] Chong Xu, Dongbin Tao. *Building Distributed Application with Aglet*.* Project is listed in the IBM Tokyo Research Lab's aglet homepage.

[3] Danny B. Lange. *Mobile Objects and* Mobile Agents: The Future of Distributed *Computing.* 12th European Conference on Object-Oriented Programming Brussels, Belgium, July 20 - 24, 1998

[4] Danny Lange and Mitsuru Oshima. *Programming and deploying Java mobile agents with Aglets*. Addison-Wesley, 1998.