Paper Submitted to World Wide Web (WWWJ26-01)

Revised Version ***(Accepted)***

Title: Study of Building Internet Marketplaces on the Basis of Mobile Agents for Parallel Processing

Authors:

1

Dr. Yan Wang
Research Fellow
Department of Computer Science
National University of Singapore
3 Science Drive 2
Republic of Singapore, 117543
Email: ywang@comp.nus.edu.sg
Telephone Number: (65) 874-8440 (office)
Fax: (65) 777-9096

2

Dr. Kian-Lee Tan **(Contact Author)**
Associate Professor
Department of Computer Science
National University of Singapore
3 Science Drive 2
Republic of Singapore, 117543
Email: tankl@comp.nus.edu.sg
Telephone Number: (65) 874-2862 (office)
Fax: (65) 779-4580

3

Jian Ren
Master Student
Department of Computer Science
National University of Singapore
3 Science Drive 2
Republic of Singapore, 117543
Email: renjian@comp.nus.edu.sg
Fax: (65) 777-9096

Study of Building Internet Marketplaces on the Basis of Mobile Agents for Parallel Processing

Yan Wang, Kian-Lee Tan Jian Ren

Department of Computer Science
National University of Singapore
3 Science Drive 2, 117543
Republic of Singapore
{ywang, tankl, renjian}@comp.nus.edu.sg

Abstract: In this paper, we propose a framework of Internet marketplaces on the basis of mobile agents. It not only simulates real commercial activities by consumers, agents and merchants, but also provides an environment for parallel processing. The latter is particularly important as more shops (sites) can be searched in real time to provide consumers with better choices. Meanwhile, if the number of mobile agents is very large and the dispatch is processed in a serial way, it can become a bottleneck that impacts the efficiency as a whole. In this paper, we also present and discuss several hierarchical dispatch models where the dispatch of multiple mobile agents can be processed in parallel over different hosts. We study these models analytically and empirically. The conducted experiments show that, in comparison with several serial mobile agent models, parallel mobile agent models can improve the performance significantly. In addition, in the best case for the parallel dispatch model, the time complexity for dispatching n mobile agents is $O(\log_2 n)$.

Keywords: Mobile agent, e-commerce, Internet marketplace, parallel dispatch

1 Introduction

The advances of web technologies such as the Internet, HTML, Java and XML have greatly pushed the development of Electronic Commerce (EC). Today, many online shops (e-shops) publish their product catalogues on the Internet, offering a wide variety of goods. More importantly, consumers are turning to the Internet for such information and purchase their goods online.

However, the wide variety of choices to the consumers has also introduced the problem of information overloading. Moreover, there are so many e-shops and goods for the consumers that it has become too time-consuming, if not impossible, to find the best (cheapest) deal. Hence, further research and development are necessary to gain experiences to provide consumers with a more convenient and user-friendly environment. One promising direction is to exploit mobile agents for e-commerce [11].

Mobile agents are mobile, flexible, autonomous, dynamic and efficient [4]. When encapsulated with a task, a mobile agent can be dispatched to a remote host by the original host. After executing and accomplishing its tasks at the remote host, it can send the results back by returning to the original host or sending them through a message. The mobile agent approach is also suitable for deploying parallel processes over distributed sites on the Internet. The tasks can be decomposed and encapsulated to multiple mobile agents. Every mobile agent can run independently to accomplish its task. And all the mobile agents can run in parallel on distributed hosts so that the whole tasks can be completed in a short time.

Based on these features, as pointed out by Rodrigo [9], future e-commerce models will enhance current models by using mobile agents. On one hand, in our real life, people can turn to a few agents or agencies for buying something such as an air ticket, or renting a house. They can choose a satisfactory one from multiple provided plans. On the other hand, the mobile agent scenario offers us more flexibility and agent-oriented modeling capability to apply the consumer/agent/merchant model of real commercial activities to the building of electronic marketplaces. In addition, it can also provide an environment for parallel processing over distributed sites to achieve greater efficiency [7].

In this paper, we first propose a framework for Internet marketplaces that exploits mobile agent technology extensively. It not only supports activities of consumers and merchants, it also facilitates parallel computation. The latter is especially important as more sites/shops can be searched in a shorter time to provide consumers with better choices in their decision-making. The mobile agent based framework can inherit and extend the conventional client/server architecture of the HTML and applet technologies, which are widely adopted by existing e-shops and logically it is transparent to end users. In addition, we propose several hierarchical models suitable for the parallel dispatch of mobile agents in large scale.

Based on the proposed framework, we have implemented a prototype system and conducted 2 sets of experiments. In the first experiment, we explore several serial mobile agent models and one parallel mobile agent model that can be adopted in such an environment. In the second experiment, several dispatch models are compared. All programs are implemented as 100% Java codes using the Aglets system [15]. The system was set up on top of 17 PCs connected in a LAN. The results of our experiments show that the parallel mobile agent models outperform other models by a wide margin and the experimental results of dispatch models match theoretical analysis.

The rest of the paper is organized as follows. Section 2 reviews some related work. In Section 3, we present a framework of Internet marketplaces. Section 4 presents the characteristics of three stages for a dispatched mobile agent from dispatching to sending back results. Section 5 describes several hierarchical dispatch models and their performances are theoretically analyzed in section 6. Section 7 presents our experimental study and reports our results, and finally, we conclude in Section 8.

2 Related Work

There has been an increasing amount of research activities to exploit mobile agents to support electronic markets or enterprises.

In [11], Sohn proposed an architecture for electronic market by applying the mobile agent technology. In his work, the market consists of conductors and members. The conductor manages the market and members participate in electronic commerce activities. Members are providers, shops and consumers. The conductor provides the framework of the market and manages the setup of members, product ontology and member information. Sohn's work gives us a fundamental description for setting up an electronic market with mobile agents and it introduces some internal activities that these agents should do. But his work addresses only an individual market without any focus on an electronic market community on distributed sites.

Chrysanthis's work views the establishment of a virtual enterprise (VE) as a problem of dynamically expanding and integrating workflows in decentralized, autonomous and interacting workflow management systems [1]. In this framework, mobile agents are employed for

advertising, negotiating and exchanging information as well as its management. Chrysanthis's contribution is the idea of using workflows to support multi-organizational processes to form a VE and [1] gives a brief description on how to utilize the mobile agent technology.

Lange briefly introduced a mobile agent based marketplace architecture in [5] and showed that the IBM Aglet Software Developing Kit (ASDK) system [4, 15] is suitable to build an electronic marketplace and the meeting pattern and communication mechanism of Aglets, which are mobile Java objects, can be adopted to meet the requirements for representing the behaviors of mobile agents. In his framework, the consumer agent visits marketplaces one by one for a request and performs negotiation activities. There is, regrettably, no global control mechanism. A similar work is also introduced in [2].

The above-mentioned works benefit much from the deployment of mobile agents, such as good mobility, high autonomy as well as the role simulation and role specification that present the realistic simulation to the real commercial activities. But they simply put mobile agents in a serial working pattern and their global control mechanisms are not clear.

The work of Silva, Papastavrou, Panayiotou and Wang all showed the advantages of applying the mobile agent approach to parallel processing over distributed databases or data sources [7, 8, 10, and 13]. A mobile agent can decompose its tasks to multiple sub-mobile agents and dispatch them to distributed sites simultaneously in order to let them work in parallel. Hence, the mobile agent technology is naturally suitable for deploying parallel and distributed computation. The performance is comparable to, and in some sense outperforms the current approach via expensive network and slow network, such as the wireless network or dial-up network.

The performance issue is another important consideration for adopting the mobile agent approach when building electronic marketplaces. Particularly, consumers need to know the updated/revised prices of goods. Those cache strategies adopted by web search engines are not suitable. In such an environment, obviously serial migration will not provide satisfactory performance and novel dispatch models for large-scale mobile agents are desirable. In addition, the mobile agent approach is suitable for supporting mobile clients since it does not require permanent network connections [6]. Furthermore, the Java based mobile agents inherit the computer-independent feature from Java programs and hence provide the platform-independent integration of heterogeneous databases and data sources [12]. Therefore, building electronic marketplaces on the basis of mobile agents with a uniform framework for the market community is expected to be beneficial to consumers to provide them with the best-buy trades more efficiently over lots of electronic shops.

3 The Infrastructure for Internet Marketplaces

3.1 Overview

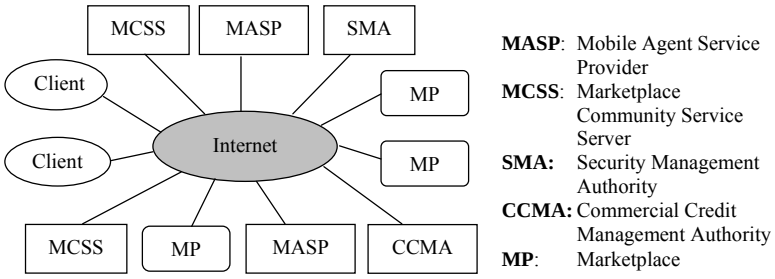

Figure 1 The Overview of the Marketplaces

In our proposed framework, there exists a set of marketplaces (see Figure 1). They are connected to the Internet. The Mobile Agent Service Provider (MASP) is an execution environment for mobile agents. A consumer-agent can be created at MASP at the client's request. Such a consumer-agent can reside in the MASP, act as a master agent and dispatch its worker agents to related marketplaces (MP) to fulfill the tasks. Meanwhile, a set of MASPs should be set up and distributed globally. They are similar to today's ISPs (Internet Service Provider).

Additionally, in the proposed architecture, there is a set of Marketplace Community Service Servers (MCSS). These MCSSs maintain the information of MPs and e-shops in the MPs. The information can be provided to relevant mobile agents when their clients query about certain products.

3.2 System Components

In this section, we shall briefly describe the components of the system.

3.2.1 MCSS (Marketplace Community Service Server)

In the proposed architecture, there is a set of Marketplace Community Service Servers (MCSS). A MCSS is responsible for maintaining the information of MPs. The information should include the domain names, IP addresses of MPs and e-shops, goods catalogue, and the identifications of the MPM (MP Manager) and shop-agents running at each MP. These information of related MPs and e-shops can be provided when a client tells the MCSS what kind of goods it needs. The role and mechanism of MCSS are similar to the DNS (Domain Name Service) server, which offers the conversion between domain name and IP address. Similar to the DNS server, when only a few MPs are set up, one MCSS is sufficient. When more MPs are set up, a set of MCSSs should be distributed in different zones.

3.2.2 MASP (Mobile Agent Service Provider)

MASP is a provider of the service enabling mobile agents as the response to clients' requests. It is a server provided to registered clients where a consumer-agent is created at the request from the client for searching the information of one or more specified goods. With the client's searching criteria, the consumer-agent will dispatch in parallel a pool of mobile agents to relevant e-shops, which will return the queried results. The whole process is introduced in section 3.4.

3.2.3 SMA (Security Management Authority)

SMA is responsible for generating certificates for all MPs, e-shops and MASPs, and managing them. In addition, SMA is responsible for taking security investigations and making security assessments on those authorized hosts according to attack reports. Here a host donates the MP-Server or E-shop server where mobile agents can be dispatched.

3.2.4 CCMA (Commercial Credit Management Authority)

CCMA is the authority making commercial credit assessment and management over all e-shops. When merchant cheating occurs, a client can report it to CCMA. After investigation, the commercial credit of the e-shop will be downgraded. On the other hand, successful transactions will help to upgrade the commercial credit.

3.2.5 Client

A client should be a registered user of any MASP before utilizing the facility of the MASP based on mobile agents. A registered user can

(1) search for specified goods through the service based on mobile agents from the MASP till making the payment.
(2) appeal to the CCMA for any merchant-cheating that may occur during the purchase and may hardly be detected before payment. If the cheating is true, the merchant's commercial credit will be downgraded. This will result in fewer consumer-agents being dispatched there.

3.2.6 MP components

A MP is the Internet marketplace consisting of a set of e-shops that run simultaneously on different servers. Within each MP, it has the facility to accept registrations of e-shops, maintain a directory of them, and authenticate foreign mobile agents.

The components of a MP are as follows:

1 **MPSM** (MP Security Manager): MPSM is similar to the firewall of an intranet. It is in charge of:
(1) maintaining the security of the whole MP, such as authenticating incoming foreign mobile agents and monitoring the communication out of the MP from a mobile agent or an e-shop, and
(2) broadcasting the certificates of e-shops to other relevant sites, such as MASPs, and
(3) registering the MP to MCSS.

2 **MPM** (MP Manager): A MPM is responsible for the management of the MP, such as accepting the registration of an e-shop in the MP and the application of withdrawal, and maintaining the directory of e-shops in the MPDS directory server (MP Directory Server). The MPM is also responsible for accepting the registration of a MASP so that mobile agents can be dispatched from it.

3 **MP-Server**: It is a server where MPSM and MPM run. It is also an execution environment for incoming mobile agents. An agent dispatched by a master consumer-agent for visiting e-shops in the MP will first arrive here. Only after having passed through the security check by MPSM, then can it enter any e-shops in the MP.

4 **Shop-Agent**: A Shop-Agent is an agent running at the shop server that is responsible for
(1) maintaining the shop information and goods information stored in the shop database.
(2) periodically sending updated information to MPM for modifying the goods-catalogue of the e-shop maintained in the MPDS (MP Directory Server).
(3) communicating with incoming consumer-agents providing the goods information they required.
(4) monitoring the execution of foreign consumer-agents and protecting the local resources of the e-shop
(5) registering the e-shop to the MPM and through it registering to the MCSS when the e-shop is set up
(6) applying the certificate of the e-shop from SMA and sending it to the MPM

5 **MPDS** (MP Directory Server): Its responsibility is to store the registration information and the goods catalogue information of all e-shops in the MP. Only the MPM can update and maintain them.

6 **E-shop Server**: An e-shop server is the place where the e-shop is set up within the domain of a MP and the shop-agent runs. It is also the execution environment of incoming mobile agents.

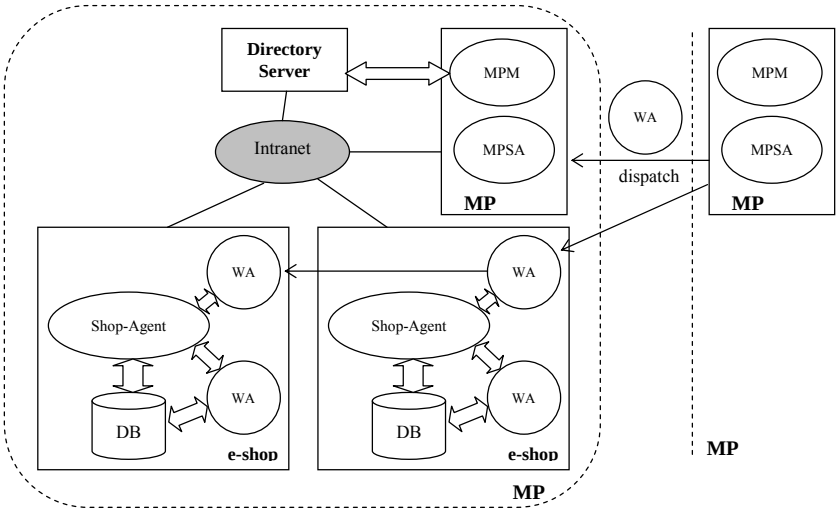

Figure 2 MP components

3.3 Procedures of Setting up a MP and an e-Shop

When setting up a MP, the MPM should register the MP to MCSS by sending the following:

(1) MP's name, domain name and IP address
(2) MP's certificate including its public key obtained from SMA
(3) id of the MPM
(4) IP addresses, certificates, directories and goods catalogue of all e-shops in the MP
(5) identifications of corresponding shop-agents
(6) current time

When an e-shop is set up in a MP, the shop-agent should register to the MPM by sending the following:

(1) e-shop's name and IP address
(2) e-shop's certificate including its public key
(3) e-shop's goods catalogue
(4) identification of the shop-agent
(5) current time

Information (1) and (2) are put in the MPDS (MP Directory Server) by MPM. If the goods catalogue of an e-shop is changed, the shop-agent will notify the MPM and MPM will report these changes to MCSS. The MPM will also report to MCSS when any e-shop withdraws or the whole MP withdraws.

3.4 Process Workflow

Based on our framework, the process enabling buying and selling can be described as follows:

(1) Input request

For a client, he/she chooses a MASP where he/she has registered as a user to input the information of a good such as the name, model, type, some selection criteria for the goods such as the warrantee service and delivery/shipment service, and potential merchants, such as the security rank and commercial credit.

(2) First phase evaluation and searching e-shops

With the request of a client, a consumer-agent is created at the server of the MASP, who will act as a master agent and dispatch a pool of mobile agents, which are termed as worker agents (WA) and Primary Worker Agents (PWA), to qualified e-shops after carrying out the first phase of the two-phase evaluation, which makes an evaluation on security rank, commercial credit of all e-shops that sell the same kind of specified goods. These attributes are obtained from MCSS, SMA and CCMA. After evaluation, a pool of WAs are dispatched to qualified e-shops [17].

(3) Second phase evaluation

After all mobile agents return the results, the second phase evaluation is performed on both goods' information and e-shops' security rank and commercial credit. The sorted results are presented to the client by the master agent.

(4) Negotiation

With the client's selection, a few e-shops will be selected for negotiation by dispatching a negotiation-agent. Some negotiation models have been proposed, such as [3]. In this paper, we will not address this issue.

(5) Book and Payment

With the success result of negotiation, one e-shop will be selected to book the goods and make an online secure payment.

(6) Purchase Feedback by Consumers

3.5 2-Phase Evaluation

In our proposed framework, we can easily deploy mobile agents for parallel processing. When the master consumer-agent is created and running, it can get a list of e-shops from the MCSS that offer the goods that its client needs to buy. For the phase of searching and negotiating the consumer-agent can act as a master agent and dispatch multiple worker agents to these e-shops for querying goods' information, such as the stock status and the price. Each work agent is responsible for visiting one e-shop. Once it fulfils its task, it sends the results to the master agent.

If there are a huge number of e-shops (i.e., in thousands) providing the needed goods, to reduce the network load, we conduct a *two-phase evaluation process* at the side of master consumer-agent before and after dispatching worker agents. This process uses the principle of utility theory and fuzzy-set rules. The evaluation function of goods x is given over a set of domain specific attributes x_i as follows:

$$U(x) = \sum_i w_i * V_i(F_i(x_i))$$

where F_i is the grading function that calculates the firing level for attributes x_i. This grading function is attributes dependent, i.e., for price, the category of levels can be very good, good, moderate, poor and very poor. V_i is the score function that maps the attribute grading levels into interval [1, 10] and w_i is the weight of attribute x_i. We assume the weights are normalized, i.e. $\sum w_i = 1$.

Before dispatching worker agents, the first phase evaluation is done only over two attributes. They are commercial credit and security rank. The commercial credit of an e-shop is set by CCMA based on the number of its previous successful transactions. The security raking is done by SMA according to the security history of an e-shop. The client should input the weight for each attribute and the selection criteria such as the number of e-shops to be searched or the lower limit for evaluation value. Those attributes for the first phase evaluation are stored in the MCSS with the goods types for all e-shops. After the first phase evaluation, worker agents will be dispatched to those qualified e-shops for searching in parallel.

On the second phase, when all worker agents send back addition details of the goods, such as the price, stock status, warranty service and its expense, and delivery/shipment service and relevant expense, the evaluation process will be taken again over all attributes and the sorted results will be presented to the end user. With his/her selection, the master consumer-agent can send new worker agents to a small set of visited e-shops to negotiate for lower price and/or more convenient services. According to the results, the end user will choose one e-shop for booking and payment.

4 Three Stages for a Mobile Agent

When a mobile agent is dispatched to a remote host to accomplish a specified task, the whole process can typically be decomposed as follows:

(1) Dispatching

In this stage, the dispatched mobile agent should first be created by the master agent, which is mobile or stationary. When it is created, some arguments are encapsulated into the mobile agent, including the task, the address information of the master agent for sending back results. The code for accomplishing the task should also be included in the dispatched mobile agent. After the mobile agent has been created, the master agent will dispatch it to the remote host through a socket-based connection.

Generally, the time for this stage depends on the bandwidth, traffic state of the network, the size of the mobile agents and the dispatch algorithm. The dispatch process is mainly a network-intensive job.

(2) Accomplishing Tasks

If the dispatched mobile agent has been successfully dispatched to the remote host, it should begin to execute and access local data so as to accomplish its task. Due to the characteristics of

the task, the mobile agent can communicate with local stationary agent or access local data, such as XML documents or database, directly.

Since the mobile agent approach is well suitable for deploying parallel processing over distributed data resources, a mobile agent can be assigned a simple task so that it has small size and can visit only one remote host to accomplish its task. A mobile agent can also be assigned a set of tasks that should be accomplished by visiting a set of remote hosts. If these tasks are semantically dependent and should only be finished in a specific order, dispatching one mobile agent is essential and good enough that it can migrate in an itinerary pattern. Otherwise, if these tasks are semantically independent and the number of remote hosts that should be visited is large, these tasks should be distributed to multiple mobile agents so that each mobile agent has only one relatively simple task that it will not take a long time to accomplish it. Thus, the master agent can get all the final results in a short time since these dispatched agents can execute in parallel over different processors. In this case, taking e-commerce as an example, the end user can easily get a large set of quotations for his/her desired products in a very short time.

(3) Sending Results Back

When a dispatch mobile agent has accomplished its task, it should send back the results. It can either dispatch itself to the origin host carrying its results or send the results back through a message. Generally speaking, the latter way can be faster since the former way should send back both the results and the code of the mobile agent. This way is necessary when the network is partially connected or the connection is dynamically changed, where the autonomous migration of a mobile agent can help to choose different route for returning.

As introduced above, the execution time for a mobile agent at the remote server side depends on many factors. They include the nature of the task, such as how much data the mobile agent should access, the complexity of the task, such as whether it is a simple query or a conjunctive query, and the current state of the remote server.

For the period of sending back results, when a large number of mobile agents are dispatched, it is difficult to restrict a model for data collection since most mobile agents have different tasks and the individual execution time of each mobile agent is dependent on many factors as discussed above. In the e-commerce applications, the size of result data sets is generally small in most cases. Therefore, when a large number of data sets of small size are sent back in different time periods, it is unlikely to result in network congestion. The whole execution time is therefore dependent on the mobile agent who is the last one to complete its task.

However, when a large number of mobile agents are dispatched for the same kind of task, such as searching for the price of specified goods, the dispatch process can cause a bottleneck easily at the dispatching side. In this case, the serial dispatch process is obviously not a good choice. If the dispatching process can be divided into several segments so that some of them can be moved to different hosts and they can be processed in parallel, the total dispatch time can be reduced. Based on this idea, we propose several hierarchical dispatch models that can greatly improve the performance.

5 Parallel Dispatch Models

In a hierarchical model, the Master Agent is only responsible for dispatching Primary Worker Agent (PWA) and the dispatch work is partially moved to PWAs. Each PWA is responsible for dispatching a cluster of PWAs or Worker Agents (WA). A WA performs the assigned task at the

remote server. In comparison to the model in which the Master Agent should dispatch all the mobile agents, the Master Agent in the hierarchical model has greatly reduced its load.

1 Master Agent (MA): A MA is a mobile agent who is responsible for offering the interface to end users for inputting query tasks, decomposing these tasks, dispatching mobile agents for accomplishing the tasks, and collecting results.

2 Primary Worker Agent (PWA): A PWA is created and dispatched by a MA or a PWA. When it has been dispatched to a remote server, its main task is to dispatch Worker Agents to remote hosts, and distribute the tasks from the MA to these Worker Agents. In an optimized model, a PWA can also carry its own data access task besides the dispatching tasks and can begin to accomplish the task after it has finished the dispatch tasks.

3 Worker Agent (WA): A WA is a mobile agent who is created by a MA or a PWA and dispatched to a remote server for accomplishing the task specified by its master agent. After having accomplished its task, the WA should report to its master agent, a PWA or a MA, for sending the results back.

To illustrate each hierarchical model clearly, we introduce the notion of Dispatch Tree (D-Tree).

A *D-Tree* is a tree in which the root vertex is the Master Agent, each leaf is a WA and each non-leaf middle vertex is a PWA if it exists. Every edge is a directed edge denoting the dispatching process that the parent vertex dispatches the child vertex. To be consistent to the hierarchical models discussed in this paper, we restrict the height of a D-Tree to be no less than 1 and it should have at least 2 leaves.

To simplify, the analysis and discussion are taken in this paper with the assumption that the time for dispatching a PWA or a WA in each model is the same.

5.1 Serial Dispatch Model H1

Before discussing any hierarchical dispatch models, we first introduce the simplest model to dispatch multiple mobile agents, which is termed as H1 model.

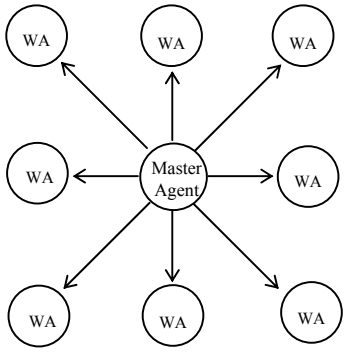

Figure 3 Serial Dispatch of Model H1

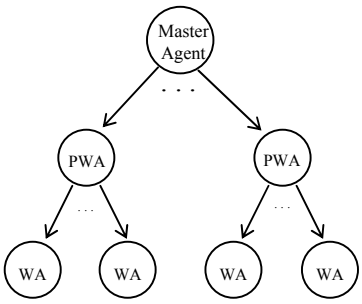

Figure 4 Model H2

H1 Model: In this model, the Master Agent dispatches a cluster of mobile agents one by one. It is in fact a serial dispatching model. In the dispatch tree corresponding to this model, the height

of the tree is 1. This model is the simplest one and obviously the slowest. Its time complexity is *O(n)* where *n* is the number of dispatched WAs.

5.2 Hierarchical Dispatch Models

5.2.1 Model H2

In model H2, the height of the D-Tree is fixed to 2. That means, as shown in Figure 4, the Master Agent dispatches a group of PWAs and each PWA dispatches a cluster of WAs, which try to accomplish their tasks. Here we suppose that the number of mobile agents in each cluster is the same. The Master Agent can divide all the WAs into several groups and distribute them, together with their corresponding tasks and the addresses of the destination hosts, to PWAs. Model H2 is easy to operate when programming.

5.2.2 Model Tm

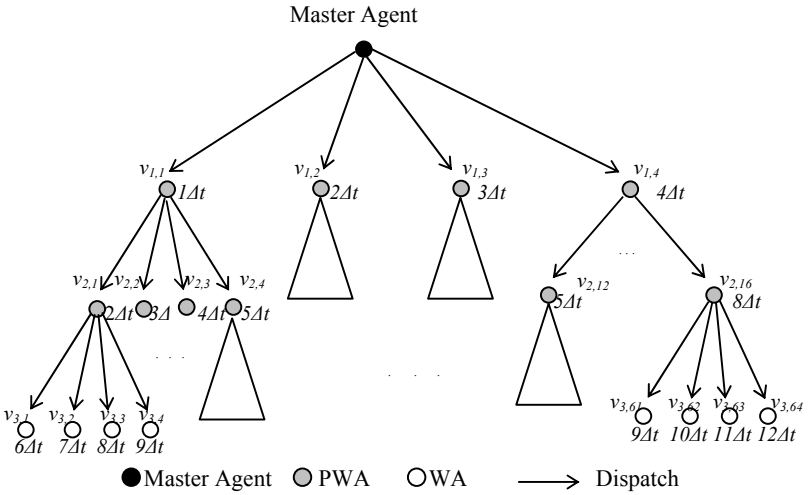

Figure 5 4-branch Dispatch Tree of Model Tm with 64 Mobile Agents

Both model H1 and H2 have fixed height of a dispatch tree. In model Tm, the D-Tree is formed as a m-branch tree where the number is the dispatch tree is fixed. That means that the MA should dispatch *m* PWAs and each PWA also should dispatch *m* PWAs if it has more than *m* WAs in its cluster and all the WAs that should finally be dispatched are equally distributed to these new PWAs. The process goes on if a new PWA has more than m members in its cluster. It will dispatch m new PWAs and distribute these members to them. When a newly dispatched PWA has just m members in its cluster, it will dispatch m WAs directly.

Figure 5 presents the D-Tree of model Tm with 64 WAs where *m=4*. In this tree, the MA should dispatch 4 PWAs only (i.e., vertex $v_{1,1}$, $v_{1,2}$, $v_{1,3}$, $v_{1,4}$). Since there are 64 WAs to be dispatched finally, they are first distributed to the groups hold by 4 PWAs. Each PWA is responsible for dispatching 16 WAs altogether. Taking $v_{1,1}$ as an example, it should dispatch four PWAs (i.e., vertex $v_{2,1}$, $v_{2,1}$, $v_{2,3}$, $v_{2,4}$) first and each PWA dispatches 4 WAs so that 16 WAs are dispatched finally in this group. Suppose the time for dispatching a PWA or a WA is Δt, the total dispatch time by model Tm is $12\Delta t$, which is greatly shorter than $64\Delta t$ of model H1.

5.2.3 Model Tm+

In model Tm, the task of a PWA is to dispatch WAs only. However, if there are *n* WAs to be dispatched to *n* remote servers, *m (m<n)* PWAs should be dispatched to *m* remote servers first for dispatching *n* WAs. Therefore, there will be *(m+n)* mobile agents to be dispatched altogether. Though the time segments for dispatching are overlapped since the dispatch work is processed in the hierarchical way, the total dispatching time can be decreased if the amount of dispatched mobile agents can be reduced.

In model Tm, *m* PWAs should be dispatched to remote hosts first before they begin to dispatch other PWAs or WAs. If each PWA can carry a task that should be assigned to a WA in model Tm, as well as the corresponding code for carrying out the task after it has finished to dispatch all the WAs in its cluster, there will be *m* PWAs and *(n-m)* WAs totally that should be dispatched altogether. For any PWA, for example, if it has 4 WAs to dispatch in the case of Tm model, it will dispatch 3 WAs in the updated case by Tm+ model because the PWA will fulfill the task of the original 4th WA. Certainly the PWA should be dispatched to the remote host where the "4th WA" should go. In this way, each PWA vertex in the D-Tree can save the dispatching time, as accumulated, the total dispatch time for dispatching all the WAs can be reduced.

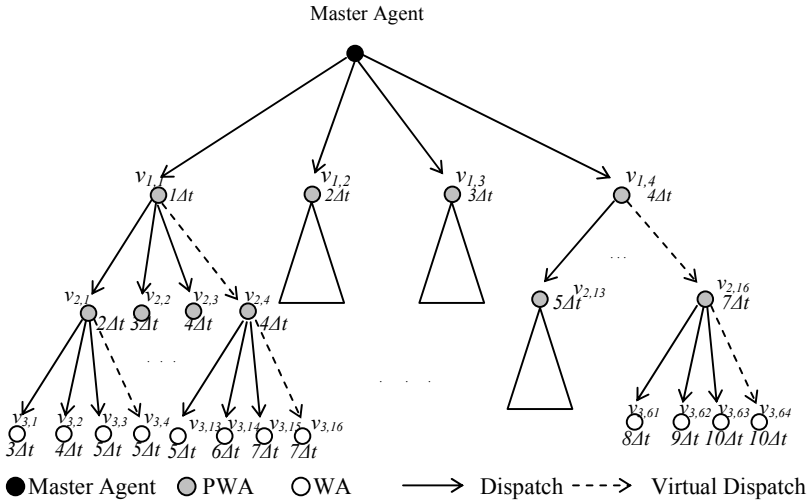

Figure 6 4-branch Dispatch Tree of Model Tm$^+$ with 64 Mobile Agents

Figure 6 shows the D-Tree of model Tm+ with 64 WAs. In this tree, the Master Agent dispatched 4 PWAs first. They are $v_{1,1}$, $v_{1,2}$, $v_{1,3}$,$v_{1,4}$. This process is the same as model Tm. PWA $v_{1,1}$, for example, only dispatches 3 other PWAs (i.e., $v_{2,1}$, $v_{2,2}$, $v_{2,3}$). Each PWA should dispatch 4 WAs. $v_{1,1}$ virtually dispatched $v_{2,4}$. That means the 4th dispatching task is still carried by $v_{1,1}$. $v_{2,4}$ first dispatched 3 WAs to different remote hosts and then begin to carry out the task of its own. In Figure 6, when $v_{2,4}$ has dispatched $v_{3,13}$, $v_{3,14}$ and $v_{3,15}$, $v_{3,16}$ is virtually dispatched since $v_{3,16}$, $v_{2,4}$ and $v_{1,1}$ are the same vertex and no real dispatch action should be taken. So, if Δt is the time for dispatching a PWA or a WA, $7\Delta t$ is the time when $v_{3,15}$ is dispatched. It is also the time $v_{3,16}$ is virtually dispatched and the time for $v_{3,16}$ to start its data access task. Therefore, as shown in Figure 6, there are 64 WAs are finally dispatched and the time for dispatching all WAs is $10\ \Delta t$, which is shorter than $12\Delta t$ from model Tm.

5.2.4 Model H2+

Similar to model Tm+, if each PWA in model H2 can carry a task for local data access, the total dispatch time can also be reduced. Compared to model H2, this model is termed as H2+.

5.3 Processing Any Number of Mobile Agents

In the above discussions (section 5.1 and section 5.2), we have restricted the number of mobile agents such that the depth of each WA vertex (leaf vertex) in the dispatch tree is the same. But these models can be easily extended to any number of mobile agents. The variation of the number of WAs will cause the changes of some leaves vertexes. For example, suppose we have 64 WAs and adopt model Tm (e.g., $m=4$). The dispatch tree is shown in Figure 5. If the whole number is 63 now, the PWA $v_{2,16}$ will dispatch $v_{3,61}$, $v_{3,62}$ and $v_{3,63}$. $v_{3,64}$ will not exist. If the number is 60, $v_{2,16}$ will become a WA (leaf vertex). Moreover, when the number is 32, the height of the dispatch tree will become 2. For simplicity, we shall focus on the cases where all leaf vertexes in a dispatch tree have the same depth.

6 Theoretical Analysis

With some theoretical analysis, the dispatch time and the time complexity for each model can be obtained as follows.

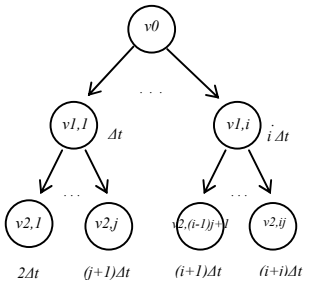

Figure 7 Dispatch Time for Model H2

Theorem 1: There are $n=m^2$ (m is an integer) WAs to be dispatched by model H2. Suppose there are i PWAs and each PWA dispatches j WAs. When $i=j=n^{\frac{1}{2}}$, the time to dispatch all WAs is the minimum.

Proof. Suppose the time for dispatching a mobile agent is Δt.

According to Figure 7, if the master agent (Vertex v_0) has i PWAs to dispatch, it requires $i\cdot\Delta t$ time to dispatch all of them. Accordingly, vertex v_{1j} is the last PWA to be dispatched. Since it has j WAs to dispatch, it requires $j\cdot\Delta t$ time to finish it. Therefore, when the last WA $v_{2,ij}$ is dispatched, the total time for dispatching all WAs is

$$T=(i+j)\cdot\Delta t, \quad where\ i\cdot j=n$$

$$T=(i+\frac{n}{i})\cdot\Delta t$$

$$Let\ F(x)=x+\frac{n}{x}$$

$$\Theta\ F'(x)=1-\frac{n}{x^2}$$

$$\therefore When\ F'(x)=0, \quad x=n^{\frac{1}{2}}$$

It is easy to show that when $x=n^{\frac{1}{2}}$, $F(x)$ gets its minimal value.

Therefore, when $i=j=n^{\frac{1}{2}}$

T obtains its minimal value, namely,

$$T_{min}=2\cdot n^{\frac{1}{2}}\cdot \Delta t$$ ف

Similarly, the dispatch time for model H2+ is $T=(i+j-1)\cdot\Delta t$, *where* $i\cdot j=n$.

Therefore, we can get the following conclusions.

Theorem 2: There are $n=m^2$ (m is an integer) WAs to be dispatched by H2+ model. Suppose there are i PWAs and each PWA dispatches j WAs. When $i=j=n^{\frac{1}{2}}$, the minimal dispatch time for n WAs is

$$T_{min}=(2\cdot n^{\frac{1}{2}}-1)\cdot \Delta t$$

Corollary: The time complexity of H2 and H2+ model are $O(n^{\frac{1}{2}})$.

Theorem 3: If n $(n\geq 2)$ WAs should be dispatched by model Tm, m is the branch number of the D-Tree $(m\geq 2)$, and $h=\log_m n$ $(h\geq 1)$ is an integer and the height of the D-Tree, Δt is the time for dispatching a PWA or a WA, then the total dispatch time for n WAs is

$$T=h\cdot m\cdot \Delta t.$$

Proof. The proof is by deduction of h.

(1) For $h=1$, the hypothesis is true since the MA should use $m\cdot\Delta t$ to dispatch m WAs.
(2) Assume the hypothesis is true for all values less than h $(h\geq 1)$.
(3) Let D-T be a m-branch dispatch tree of height h, see Figure 8.

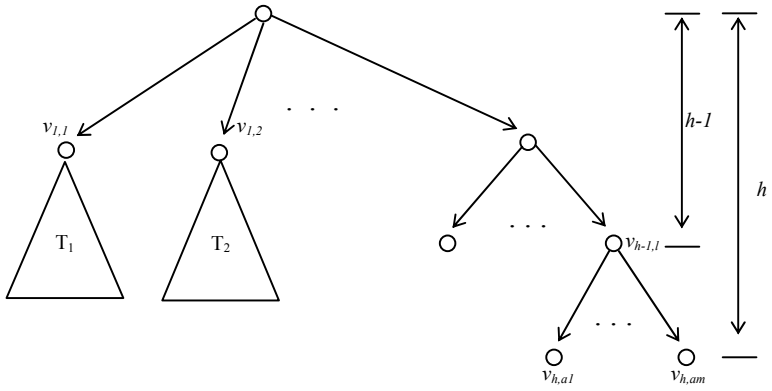

Figure 8 Dispatch Tree for Theorem 3

Suppose $v_{h-1,l}$ is the last vertex to be dispatched at the same layer of the D-Tree where the height is $(h-1)$. Then the total time from the beginning to the time when $v_{h-1,l}$ is dispatched is $(h-1)\cdot m\cdot\Delta$t.

$v_{h-1,l}$ is a PWA and it should dispatch m WAs. These WAs are $v_{h,a1}$, ,..., $v_{h,am}$, respectively. Since it takes $m\cdot\Delta t$ to dispatch them, the time for dispatching all n WAs should be

$$T=(h-1)\cdot m\cdot \Delta t + m\cdot \Delta t = h\cdot m\cdot \Delta t$$ ف

Similarly, we can calculate the dispatching time for model Tm+.

Theorem 4: If n $(n\geq 2)$ WAs should be dispatched by model Tm+ and m $(m\geq 2)$ is the branch number of the D-Tree, and $h=\log_m n$ is an integer and the height of the D-Tree, Δt is the time for dispatching one mobile agent, then the total dispatch time is

$$T=(h\cdot m-h+1)\Delta t$$

With Theorems 3 and 4, we can obtain the following corollaries.

Corollary 1: With the same WA number and the same branch number, model Tm+ can save $(h-1)\cdot \Delta t$ in comparison with model Tm.

Corollary 2: Model Tm and Tm+ have the same time complexity of $O(\log_m n)$.

Corollary 3: When all n WAs are dispatched in a binary dispatch tree by model Tm+, the dispatch time is the minimum as

$$T_{min}=(\log_2 n+1)\Delta t$$

7 Experiment Results

On the basis of the proposed architecture, we have partially implemented a prototype system, which is set up in a LAN consisting of PCs running Window NT, JDK 1.1.6, Aglets 1.0.3 [15] and the SAX APIs (the Simple API for XML) from Sun Microsystems [14].

In order to prove the availability of the proposed architecture and observe the performance improvements by parallel processing, we conducted experiments using a cluster of PCs running the prototype system. All PCs have the same configuration of Pentium 200MHz CPU and 64MB RAM. They are connected to the LAN through 10Mbits/sec Ethernet cards. One of them is dedicated as the MASP while the others are e-shops. All mobile agent programs were implemented using the Aglet system and the Tahiti server for Aglets was set up in each site.

7.1 Experiment One

In this experiment we compared the efficiency of one parallel mobile agent model with some serial models.

7.1.1 Models

We implemented five mobile agent models of execution in the experiments. Three are fully serial mobile agent models, one is a virtually parallel model (VP Agent model) and another is parallel mobile agent model. All models except the VP Agent model adopt the master/worker pattern. Each of them has one master consumer-agent. It dispatches one or multiple worker agents and collects the results from them. These serial mobile agent models resemble the typical operations of fully serial searching activities on the web by consumers. In the VP Agent model,

the consumer-agent executes locally to access all remote data through HTTP connections. The parallel model used multiple worker agents for parallel data access (see Table 1).

Table 1 Five Models in the Experiments

Model	Name	Execution Pattern
1	Itinerary Agent	Serial
2	Shuttle Agent	Serial
3	Serial Agent	Serial
4	VP Agent	Virtual Parallel
5	SParallel Agent	Serial Dispatch (H1), Parallel Data Access

In the first model, an itinerary agent is created by the master consumer-agent with the addresses of a list of e-shops. It migrates and visits all e-shops in its list one by one. When migrating to the next e-shop, it carries the data accumulated from all previous e-shops including the current one. After it has visited all the e-shops, it returns to the MASP and carries back the whole data. This model corresponds to the approach proposed by [4].

In the Shuttle Agent model, there are also one master consumer-agent and only one worker agent. The worker agent is dispatched to an e-shop in the list maintained by the master consumer-agent. When it has read the data, it returns to the MASP carrying the results. After sending the data to the master consumer-agent, it migrates to the next e-shop and repeats the process until it has visited all e-shops and returns all the data.

In the third model, similar to the second one, the master consumer-agent dispatches a worker agent to an e-shop. When the worker agent has obtained the data, it sends it back by a message and then disposes. Having gotten the result, the master consumer-agent will dispatch a new worker agent to the next e-shop until all e-shops are visited.

In the VP Agent model, the master consumer-agent is created at the client end and it starts multiple threads. Each thread locally reads data located in a remote HTTP server. The thread stops when it has finished reading the data and thus gets final results. This model resembles in a more efficient way with virtually parallel searching activities. Obviously, in this model, no worker agent is dispatched.

The fifth model benefits from the fully parallel data access. In this model, the master consumer-agent dispatches multiple worker consumer-agents one by one to all e-shops that should be visited. These worker agents can access local data in parallel and send their results to the master consumer-agent in succession but the dispatch process is fully serial.

For all models, the execution time is calculated for the period from the creation of the master consumer-agent till the moment when all results are collected. In each model with dispatched mobile agent the dispatched worker agent is designed to read data from the XML files of an e-shop directly and locally. In the implementations of all models the master consumer-agent is used to count the execution time with the uniform clock at the MASP side.

The common feature of the implementations of all models with dispatched mobile agents is that the complexity for parsing XML documents is hidden at the e-shop side where work agents, which are dispatched to e-shops from the MASP, finish the parsing tasks. For the VP Agent model, the parsing task is done at the side where the master agent resides. Furthermore, in the implementations of all models, they share the same Java codes for querying data from XML files. And certainly, they are compared with the same environment since they all run on Tahiti servers.

7.1.2 Performance Analysis

The results presented in Figure 9 to Figure 14 are the average obtained from 4 independent executions. We reboot all Tahiti servers for every execution to avoid the impact of the class cache on the execution time of repeated experiments. The variance across different executions of the same model is not significant.

For each model, the dispatched worker agent visiting an e-shop obtains 1Kbytes or 100Kbytes data from the local XML file. The length of the XML file is set to 1 Mbytes, 10 Mbytes and 100 Mbytes respectively. The read data can finally be obtained only after the whole XML file is parsed. The number of e-shops is set to 2, 4, 8 and 16. These variations aim to observe the performance differences among experiments and illustrate how these variations impact the performance.

Figure 9 presents the results from the case of reading 1Kbytes from the 1Mbytes XML file of each e-shop. It shows that the SParallel Agent model is better than serial agent models in most cases shown in Figures 9 to 16. The results in Figure 9 also show that the performances of serial agent models are very close while the parallel processing can gain improvements more significantly.

Similar conclusions can be drawn from the results showed in Figures 10 and 11 where the file size is set as 10Mbytes and 100Mbytes respectively. The difference is that the performance of the SParallel Agent model is greatly improved and it can outperform each serial model.

With respect to the VP Agent model, it can still outperform the three serial mobile agent models in Figure 9. Nevertheless, it becomes as bad as them with the increase of the size of read files (see Figures 10 and 11). It shows that when the data set size is small (e.g. 1Mbytes in the experiments), the VP Agent model can outperform other serial mobile agent models because in this case, these mobile agent models would incur more overheads from creating and dispatching worker agents while the time spent on reading a small file is short. However, with the increase of the file size as well as the number of visited e-shops, other serial mobile agent models improve since they can benefit from the worker agent's reading local files and in comparison the VP Agent model should read remote files and it evidently gets more overheads from it (see Figure 10 and 11).

When the size of data set obtained from each e-shop is set to 100Kbytes, some new changes can be observed. These results are presented in Figures 12, 13 and 14. The most significant change is that the Itinerary Agent model becomes extremely bad compared to the other serial models when the result data sets are obtained from 1MBytes XML files and 10Mbytes XML files as presented in Figures 12 and 13. This shows with the increase of the size of data set it carries, the time for a mobile agent to migrate from one site to another increases significantly. This observation is clearly shown when the experiments are being done.

However, when the result data is obtained from the 100Mbytes XML file, as shown in Figure 14, the difference between the Itinerary Agent model and other serial models is not so significant since most time should be spent on the data accessing and hence the delay from migrations for the Itinerary Agent model is not a critical factor any more.

In comparison with other serial models, the VP Agent model is good when accessing the 1Mbytes files (see Figure 12). However, when the size of file is set to 10Mbytes and 100Mbytes,

its performance starts to degrade significantly. As shown in Figure 14, it becomes the worst serial agent model. This shows that when the data size is large, the network delay can significantly impact the performance. This situation becomes worse with the increase of the number of e-shops.

With respect to the SParallel Agent model, it is good in most cases since it can benefit from the parallel execution of multiple dispatched agents. However, if the size of accessed file is 1Mbytes when the time for accessing the file is very short, as shown in Figures 9 and 12, it is not the best all the time because in this case the whole dispatch time becomes the main overhead and the serial dispatch process impacts the whole performance. Its disadvantage can be covered up when the accessed file is large so that the dispatch process is not the main overhead. As a whole, the results show that SParallel mobile agent model can sufficiently improve the performance when the accessed data sets are large and they are distributed over a large number of sites. Since all computers have the same configuration, the saving of execution time is totally obtained from the parallel execution on the basis of mobile agents. With respect to the dispatch process, the experiment in Section 7.2 shows that binary dispatch model can further improve it with a good margin.

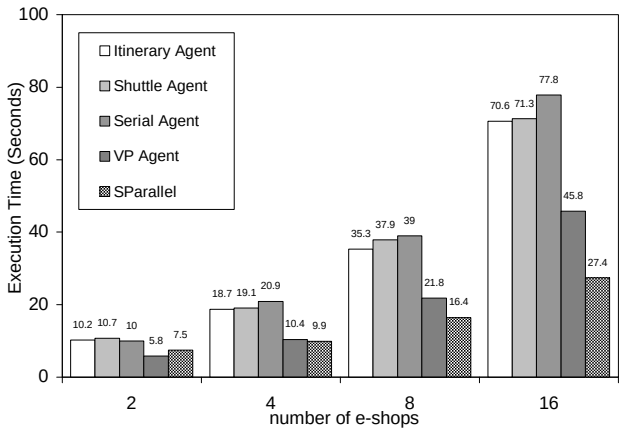

Figure 9 Results for Reading 1Kbytes Data from the 1Mbytes XML File of Every MP

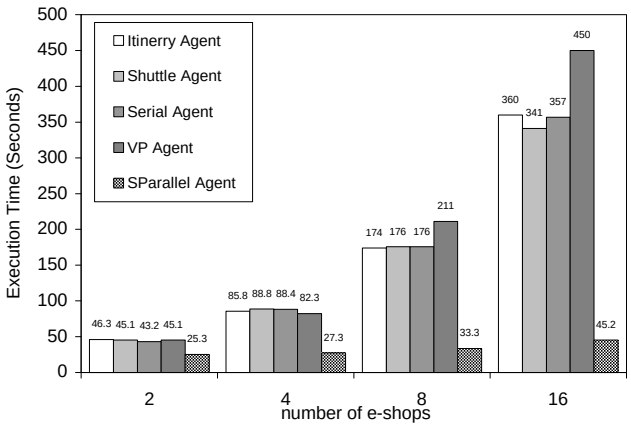

Figure 10 Results for Reading 1 Kbytes Data from the 10Mbytes XML File of Every MP

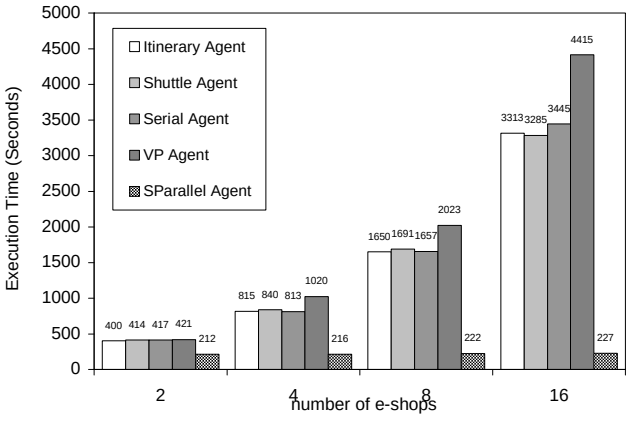

Figure 11 Results For Reading 1Kbytes Data from the 100Mbytes XML File of Every MP

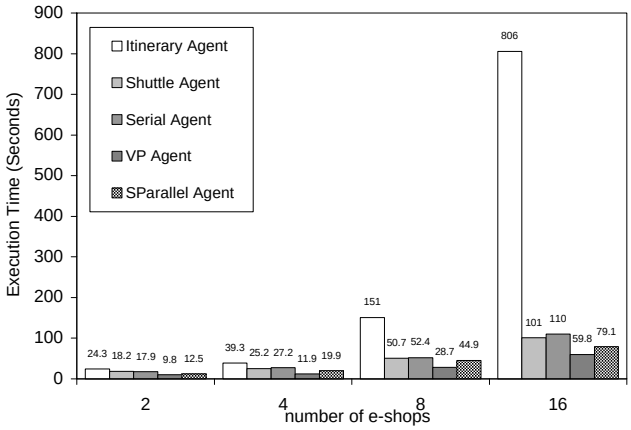

Figure 12 Results for Reading 100 Kbytes Data from the 1Mbytes XML File of Every MP

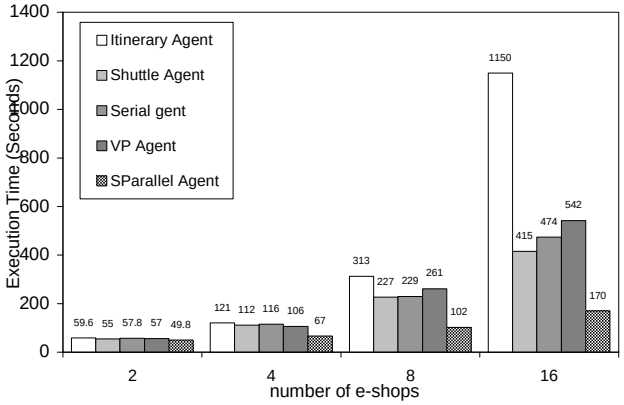

Figure 13 Results for Reading 100 Kbytes Data from the 10Mbytes XML File of Every MP

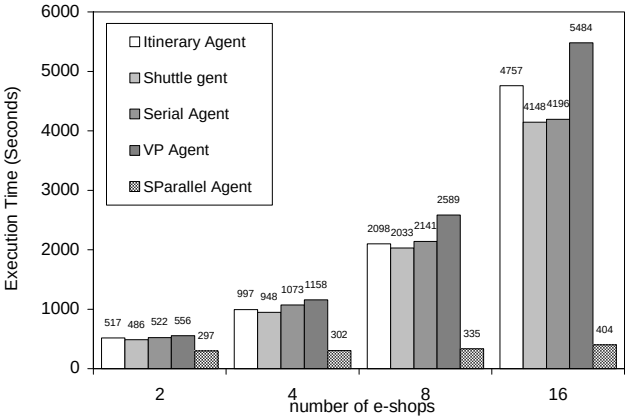

Figure 14 Results for Reading 100 Kbytes Data from the 100Mbytes XML File of Every MP

Due to the condition limitations, it is difficult to execute all the experiments on the Internet. However, in the Internet environment, the dispatch time of a mobile agent will become longer and it is more dependent on the current state of network traffic. For the execution of local data accessing, it is dependent on the performance of the e-shop server and the number of agents that are running there at the same time. No matter in what condition it is, serial agent models will be more dependent on the network traffic and e-shop servers. In contrast, the parallel model will continue to benefit from the parallel executions of worker agents and thus it remains superior.

7.2 Experiment Two

In this set of experiments, we study the performance of different dispatch models. We compare models H1, H2+ and Tm+. In fact, H2+ model can be taken as a special case of model Tm+ when $m=n^{\frac{1}{2}}$ and therefore $h=2$. To compare the performance variations, we set the size of PWAs and WAs to different values when they are created. That means that Δt is set to be different in different experiments and it is logically fixed in each one. What should be clear is that in real applications, the size of a mobile agent may hardly be 10Mbyte or 100Mbytes. However, in the LAN environment, the network delay is evidently short and it can be increased when the increase of the size of the mobile agent. So by setting the huge size of a mobile agent, we aim to simulate the network delay in a real Internet environment with a smaller size of the mobile agent. Simultaneously, we set the number of dispatched mobile agents to 4, 16 and 64 in order to show the performance variations.

The results of the experiments confirmed the conclusions drawn by the theoretical analysis.

When there are only 4 WAs, as shown in Figure 15, the serial dispatch model H1 and parallel dispatch model H2+ didn't show any significant difference.

However, when *n=16 or 64*, as shown in Figures 16 and 17, the performance differences are significant. However, the differences between the two parallel dispatch models, H2+ and Tm+, are smaller. But with the increase of the number and size of WAs, the differences widen.

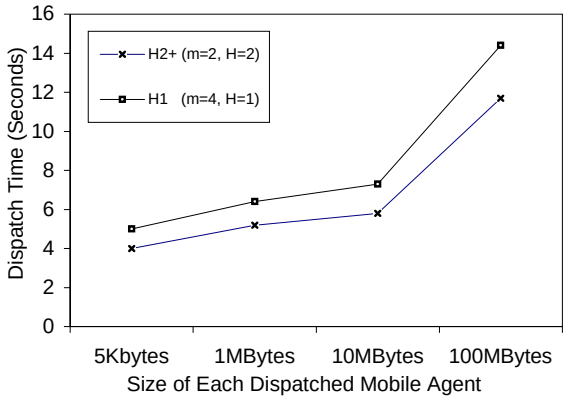

Figure 15 Results for Dispatching 4 WAs

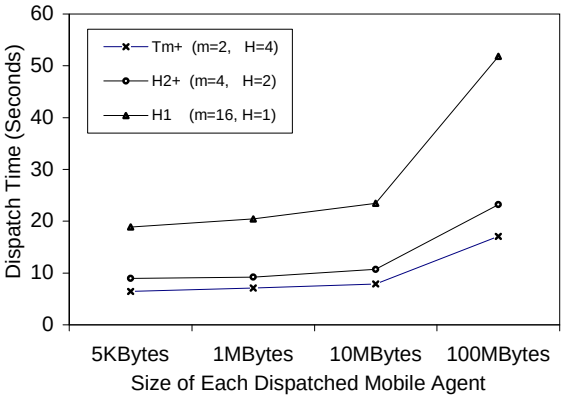

Figure 16 Results for Dispatching 16 WAs

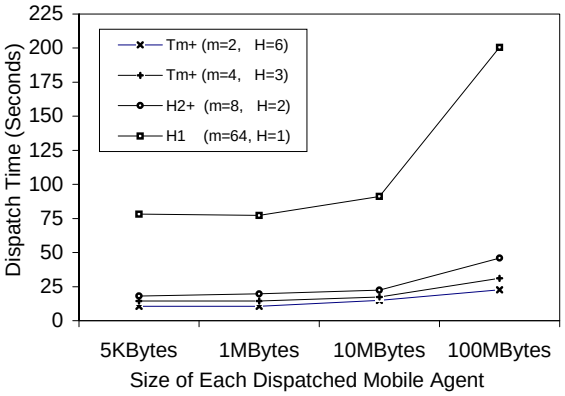

Figure 17 Results for Dispatching 64 WAs

For model Tm+, with the same number and size of WAs, the performance is improved with the decrease of *m* and it can obtain the minimal dispatch time when *m=2*. As shown in Figure 17, the saving percentage of Tm+ model with *m=2* varies from 13.9% to 27.7% when being compared with Tm+ model with *m=4*. In comparison with H2+ model, the saving percentage is from 33.5% to 51.1%. In the Internet, we can estimate that the average dispatch time will be longer than in a LAN but the performance differences will be similar.

8 Conclusions

This paper presents a framework of Internet marketplaces built on the basis of mobile agent technology to support parallel processing over distributed sites. By using the mobile agent model, it provides the possibility to realistically simulate consumers' commercial activities. The Java based mobile agent technology enables the features of light-weighted, portable and platform-independent agents. The architecture is suitable for both mobile and stationary users.

The evident benefit is that the proposed architecture can easily support both serial processing and parallel processing by mobile agents. The searching and negotiating models can be designed and implemented in the control strategy of the master consumer-agent, which uses a pool of worker agents to fulfill the consumer's input tasks in parallel. Some serial processing and the migration of mobile agents offer more flexibility and make the model closer to people's real activities. However, exploiting parallel processing provides the most important benefit and it can help to set up a system with higher efficiency and provide better supports for the consumers' best-buy strategy. The conducted experiments showed the significant performance improvements gained by parallel processing while providing 'fresh' information on goods. Especially, these improvements are gained with the platform-independent feature and the scenario hides the complexity at the e-shop side for parsing XML data, which is practical for current client/server models. In addition, the framework is generic and it is transparent to end-users.

Meanwhile, when the parallel processing environments are widely built up, the number of dispatched searching mobile agents should be large. With the improvements of network and hardware, the response time for an individual mobile agent can become short. In comparison, the dispatch process may cause the bottleneck and the performance of dispatch models becomes a critical factor. In such an environment, model Tm+ provides a good solution with the time complexity of $O(\log_m n)$. And the binary dispatch of model Tm+ can provide the shortest dispatch time both theoretically and practically. Some security mechanisms based on binary dispatch have been proposed in [18] so that both efficiency and security for parallel dispatch can be ensured.

At last, the architecture is based on the Internet and the setting up process is similar to that of the Internet. Obviously, XML is not a replacement of HTML. The XML is used to describe and carry data [16]. The HTML is used to display the data. Therefore, within our proposed architecture, each shop can still set up web pages for individual user to browse and access in the traditional way.

What we should address is that mobile agent approach may not be a replacement of client/server model. But it extends it to be more flexible enabling agent-oriented modeling and programming, and can give interesting solutions for some suitable applications such as e-commerce.

Both mobile agent and XML technologies are very new in comparison with other web technologies. When we implemented this system using the Aglets system, it is not too complicated for the communication between mobile agents with the message mechanism of the Aglets system. Nevertheless, as there exist many Java based mobile agent systems and each mobile agent should run automatically, in the long run, standards and protocols should address some issues, such as the standard communication interface between any mobile or stationary agents, and the description of goods catalogue by XML. With the support of these standards and

protocols, the mobile agent and XML technologies can bring the e-commerce into a more efficient stage.

Acknowledgements

This project is partially supported by a research grant R-252-000-015-112/303 co-funded by MOE and NSTB.

References

1. P. Chrysanthis, T. Znati, S. Banerjee, and S. K. Chang, "Establishing Virtual Enterprises by Means of Mobile Agents", Proceeding of Ninth International Workshop on Research Issues on Data Engineering: Information Technology for Virtual Enterprises (RIDE-VE '99), Sydney, Australia, March 23-24, 1999, pp. 116-123
2. P. Dasgupta, N. Narasimhan, L. E. Moser, P. M. Melliar-Smith, "MAgNET: Mobile Agents for Networked Electronic Trading", IEEE Transactions on Knowledge and Data Engineering, Volume: 11 Issue: 4 , July-Aug. 1999, pp. 509 -525
3. R. H. Guttman, and P. Maes, "Agent-mediated Integrative Negotiation for Retail Electronic Commerce", Proceedings of Workshop on Agent Mediated Electronic Trading, Minneapolis, Minnesota, USA, May 1998.
4. D. Lange, and M. Oshima, "Programming and Deploying Java Mobile Agents with Aglets", Addison-Wesley Press, Massachusetts, USA, 1998
5. D. Lange, and M. Oshima, "Mobile Agents with Java: The Aglet API", appears in "Mobility: Process, Computers, and Agents" (edited by D. Milojicic, F. Douglis, and R. Wheeler), Addison-Wesley Press, Reading, Massachusetts, USA, 1999, pp. 495-512
6. K. C. K. Law, and Y. Wang, "An Agent based Approach for Distributed Database Access in a Mobile Environment", Electronic Proceedings of the 2nd International Conference on Information, Communications & Signal Processing (ICICS'99), Singapore, December 7-10, 1999
7. C. Panayionou, G. Samaras, E. Pitoura, and P. Evripidou, "Parallel Computing Using Java Mobile Agents", Proceedings of 25th EUROMICRO Conference, Milan, Italy, September 8-10, 1999, Volume 2, pp. 430-437
8. S. Papastavrou, G. Samaras and E. Pitoura, "Mobile Agents for WWW Distributed Database Access", Proceeding of 15th International Conference on Data Engineering (ICDE'99), Sydney, Australia, March 23-26, 1999, pp. 228 -237
9. T. D. Rodrigo, and A. Stanski, "The Evolving Future of Agent-based Electronic Commerce", in Electronic Commerce: Opportunity and Challenges (Edited by S.M. Rahman and M. S. Raisinghani), Idea Group Publishing, Hershey, USA, 2000, pp. 337-351
10. L. M. Silva, V. Batista, P. Martins, and G. Soares, "Using Mobile Agents for Parallel Processing", Proceedings of International Symposium on Distributed Objects and Applications (DOA'99), Edinburgh, Scotland, September 1999, pp. 34-42
11. S. Sohn, and K. J. Yoo, "An Architecture of Electronic Market Applying Mobile Agent technology", Proceedings of 3rd IEEE Symposium on Computers and Communications (ISCC '98), Athens, Greece, June 30-July 2, 1998, pp. 359-364
12. A. D. Stefano, L. L. Bello and C. Santoro, "A Distributed Heterogeneous Database System based on Mobile Agents", Proceeding of 7th IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, Stanford, California, USA, June 17-19, 1998, pp.223-228
13. Y. Wang, K. C. K. Law and K. L. Tan, "A Mobile Agent based Protocol for Distributed Databases Access", in Proc. of 2000 IEEE International Conference on Systems, Man,

and Cybernetics (SMC'2000), 8-11 October 2000, Nashville, Tennessee, USA, pp. 2028-2033
14. URL: http://developer.java.sun.com/developer/products/xml/docs/api/overview-summary.html
15. URL: http://www.trl.ibm.co.jp/aglets/
16. URL: http://www.w3schools.com/site/site_intro.asp
17. J. Ren, Y. Wang, X. Pang and K.L. Tan, The 2-Phase Evaluation Model for Agent-mediated Internet Marketplaces, Proc. of 2nd International Conference on Web Information Systems Engineering (WISE'2001), Kyoto, Japan, Dec. 8-11, 2001, pp. 73-82
18. Y. Wang and K. L. Tan, A Secure Model for the Parallel Dispatch of Mobile Agents, Proc. of 3rd International Conference on Information and Communications Security (ICICS'2001), Springer-Verlag, LNCS Vol. 2229, Xi'an, China, 13-16 November, 2001, pp. 386-397