Applying Mobile Agent in Parallel Computing for Solving Numerical Equations

Nay Lin Soe, Ei The` Phyu
Computer University (Taung-Ngu)
naylinsoe2007@gmail.com, eithephyu@gmail.com

Abstract

Mobile Agent technology has the ability to travel from host to host in different or same network. Mobile Agents can migrate on themselves, their program and their state across the network and can execute the process at remote site. It motivates force in reducing network traffic and operates asynchronously and autonomously of the process. For these reason, Mobile Agent has been an effective choice for parallel and distributed computing. This paper presents the usage of Mobile Agent in parallel computing for solving the numerical problem of equations such as Gauss Jordan Iteration. Master-worker design of mobile agent will be used in solving the system of equations in parallel. The master agent will send a number of worker agents to different processors, calculate in parallel and will return the partial result to the host site after solving them. After all calculations, the master agent will produce the final result.

Keywords: Mobile agent, master-worker design, parallel and distributed computing

1. Introduction

Rapid development of computing technology increasingly relies on the network. Mobile agent technology is a new networking technology that deals with both form of logical and physical mobility.

Distributed computing is more general and universal than parallel computing. The distinction is subtle but important. Parallelism is restricted form of distributed computing. Parallel computing is distributed computing, where the entire system is devoted to solving a single problem in the shortest time possible. Thus, parallel computers have optimized performance. Distributed computing is more general and encompasses other forms of optimization. Parallel computation involves dividing a problem into parts in which separate processors perform the computational of the parts.

Mobile Agent Technology offers a new computing paradigm in which a program, in the form of an intelligent software agent can migrate themselves, their program and their state across the network and execute the process at remote site. Mobile Agents are an effective choice for many application for several reasons [6], including improvement in latency and bandwidth to network disconnection [3]. The driving force motivating the use of mobile agents in parallel and distributed computing is twofold. First, mobile agents provide an efficient, flexible and asynchronous method for searching for information or services in rapidly evolving networks: mobile agents are launched into the unstructured network and roam around to gather information or make other computations. Second, mobile agents support intermittent connectivity, slow networks, and lightweight devices. This second property makes the use of mobile agents very attractive [3]. The purpose of this study is to provide a comprehensive parallel computation of mobile agent for calculation of some numerical problems.

The proposed Mobile Agent Technology (MAT) is suitable computing platform for parallel computation models such as embarrassingly and pipelined models. The pipelined model has to send the data back and forth between two agents. This communication cost increases the execution timings. The embarrassingly model needs no communication between agents. Therefore, there was no problem as the data size increases or the number of worker increases. The mobile agent performs the best in this embarrassingly parallel computing model. Therefore, this paper emphasizes the embarrassingly model in parallel and distributed computing through the use of parallelisms.

2. Related Works

There has been an increasing amount of research activities to exploit mobile agents to support distributed computing. The ASDK framework is a lightweight mobile agent technology from IBM's Tokyo Research Laboratory. With aglets, it's straightforward to develop standalone 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.

Several researches have been used mobile agent in developing parallel applications on the Web using Java mobile agents and Java threads. [1, 9, 10] Mobile agents are being used already in a variety of Internet-based distributed computing applications: web database [8] cooperative environment [8, 9], and information gathering systems [6], electronic commerce systems [1] and so on.

3. Background Theory

As shown in figure 1, a mobile agent is a program which represents a user in computer network and is capable of migrating autonomously from one host to another to perform the computation on behalf of the user.

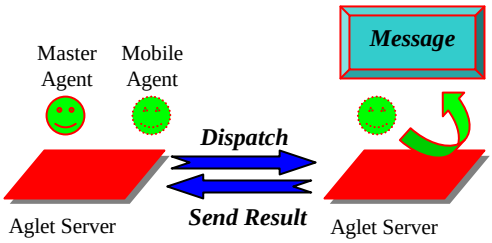

Figure 1. Mobile Agent Technology

They can roam the network either on a predetermined path or one that the agents themselves determine based on dynamically gathered information. Having accomplished their goals, the agent may return to their "home site" in order to report their result to the user. [2]

3.1 Parallel Computation Models

Parallel computation [5] consists of dividing a problem into parts in which separate processors perform the computation of the parts through use of parallelism.

3.1.1Embarrassingly Parallel Computation Model

In embarrassingly parallel computation, each process requires different or the same data and produces results from its input without any need for results from other processes. A computation can be divided into a number of completely independent parts, each of which can be executed by a separate processor. A truly embarrassingly parallel computation suggests no communication between the separate processes [5]. This situation will give the maximum possible speedup if all the available processors can be assigned processes for the total duration of the computation. The only constructs required here are simply to distribute the data and to start the processes. In this paper, we intend to use the embarrassingly parallel computation model in the form of master-slaves pattern. The embarrassingly model for parallel computing is shown in figure 2.

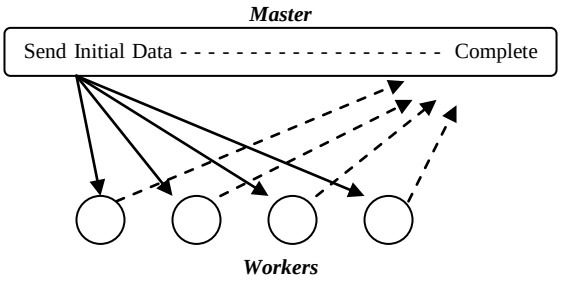

Figure 2. Embarrassingly Parallel Computation Model

3.1.2 Pipelined Computation Model

In pipeline computation, the problem is divided into a series of tasks that have to be completed one after the other as shown in figure 3. In fact, this is the basis of sequential programming [5]. Each task will be executed by a separate process or processor, as shown in Figure 3.

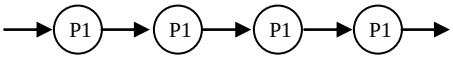

Figure 3. Pipelined Computation Model

Each stage will contribute to the overall problem and pass on information that is needed for subsequent stages. This parallelism can be viewed as a form of functional decomposition. The problem is divided into separate functions that must be performed and the functions are performed in succession.

Given that the problem can be divided into a series of sequential tasks, the pipelined approach can provide increased speed under the following three types of computations:

- if more than one instance of the complete problem is to be executed
- if a series of data items must be processed, each requiring multiple operations
- if information to start the next process can be passed forward before the process has completed all its internal operations.

There are many problems that can be pipelined: adding numbers, sorting numbers, prime number generation and linear equations.

3.1.3 Mobile Agent in Parallel Computing

Mobile agents are dispatched to remote site transparently from the user based on the parallel computation model. For parallel computing, master agent starts and dynamically creates and, initiates identical slave or worker agents. And then the master agent dispatches a variable number of mobile (worker) agents to several workstations to work in parallel. This system intends to use master-worker

model of parallel computing. The responsibilities of master-worker agents are [5]:

- A master agent creates a number of worker agents.
- The workers initiate their tasks and data.
- The worker moves to a remote host and perform task.
- The worker sends the partial results of the task to the master.
- The worker disposes of itself.

4. Proposed System

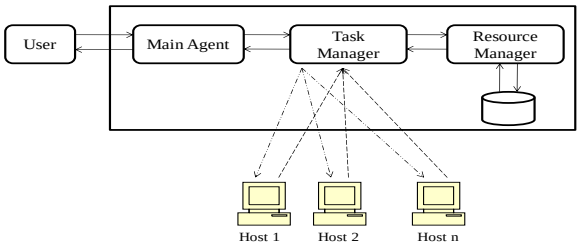

Figure 4. Overview of the System

The overview of the proposed system is shown in figure 4. When the user inputs the equations into the system, the main agent accepts this system of equations. And it sends these equations to task manager agent. When the task manager agent accepts the system of equations, this agent checks the number of resources required for solving these equations and requests the resource manager agent for the resource lists. Then the task manager agent creates a number of worker mobile agents and assigns task to them. On creating of worker mobile agent, this agent accepts the data and task from task manager agent and goes to resource (host) for calculation. After calculation, it returns the partial result to the task manager agent. The task manager agent collects this partial result to produce the final complete result.

Agents used in this system are:

- Register mobile agent
- Main agent
- Resource Manager agent
- Task Manager (Master) agent
- Worker mobile agents

4.1 Register Mobile Agent

The register mobile agent is used to register a computing resource (workplace or the address of the resource) to the server. The algorithm for this agent is as shown in figure 5.

```
class RegisterAgent extends Aglet{
   void onCreation(){
      host = getHomeAddress();
      Message msg = new Message("Register");
      msg.setArg("Host",host);
      this.dispatch( /*Server URL */);
   }
   void onArrival(){
      call_ResourceManager(msg);
   }
}
```

Figure 5. Algorithm for Register Mobile Agent

4.2 Main Agent

The main task of this agent is to interact with the user and to coordinate the task manager agent. It accepts the input equations from the user. This paper intends to use Gauss Jordan Iteration method as a case study. These equations are sent to Task manager for parallel calculation and the final result will be displayed to the user by this agent. The algorithm for this agent is as shown in figure 6.

```
class MainAgent extends Aglet{
   void onCreation(){
      showFormDesign();
      getInput();
      create_TaskManagerAgent(user Input);
   }
   void handleMessage( Message msg ) {
      if (msg == "show"){
         showResult();
      } else if (msg == "InvalidEqt"){
         showInvalidEqt();
      }
   }
}
```

Figure 6. Algorithm for Main Agent

4.3 Resource Manager Agent

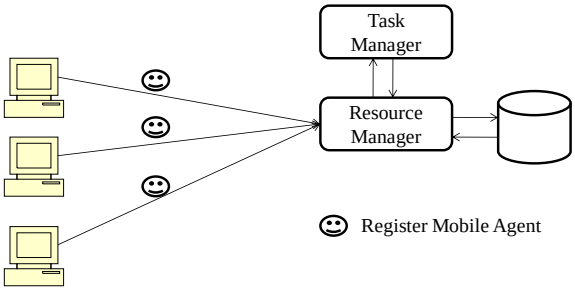

Figure 7. Resource Manager Agent

This agent accepts the register request from the workstations and stores the workstation list in the database. This list is provided to the Task manager for parallel computation. (see Figure 7) The algorithm for this agent is shown in figure 8.

```
class ResourceManager extends Aglet{
   handleMessage( Message msg ) {
      if (msg == "Register"){
            insertDB(msg.getArg("Host"));
      } else if (msg == "getResource"){
            resource[]=getResource (n);
            return resource;
      }
   }
}
```

Figure 8. Algorithm for Resource Manager Agent

4.4 Task Manager Agent

This agent gets the system of equations from the main agent and accepts the required resources from the resource manager agent. And then it creates and sends a number of worker agents to the resources. The results from all worker agents are collected and send the final result to the main agent to display to the user. The algorithm for this agent is shown in figure 9.

```
class TaskManager extends Aglet{
   void onCreation(UserInput from MainAgent){
      getUserInputEquation();
       rm=create_ResourceManagerAgent();
      resource[] = rm.setMsg("getResource", n);
      GaussJordan();
   }
}

void GaussJordan(){
   A[][] = get A matrix for the equation Ax=b;
   for j=1 to n do
      for i=1 to n do in parallel //create worker agents
         if i!= j then  send (A,j,i,n,resource[i]);
      end for

      wait for allReply();
   end for

   for i=1 to n do // finding final result value
      x[i] = a[i][n+1] / a[i][i];
   end for
}
```

Figure 9. Algorithm for Task Manager Agent

4.5 Worker Mobile Agent

This agent accepts the task from task manager agent and will go to the resource to be computed. After the calculation their partial result is returned to the Task Manger agent. The algorithm for this agent is shown in figure 10.

The sequence diagram of this proposed system is also shown in figure 11.

```
class WorkerGJ extends Aglet{
   void onCreation(){
      get A,j,i,n;
      go to resource[i];
   }
   void onArrival(){
      for k = j to n+1 do
         a[i][k] = a[i][k] – (a[i][j] / a[j][j]) a[j][k];
      end for;
      send A;
   }
}
```

Figure 10. Algorithm for Resource Manager Agent

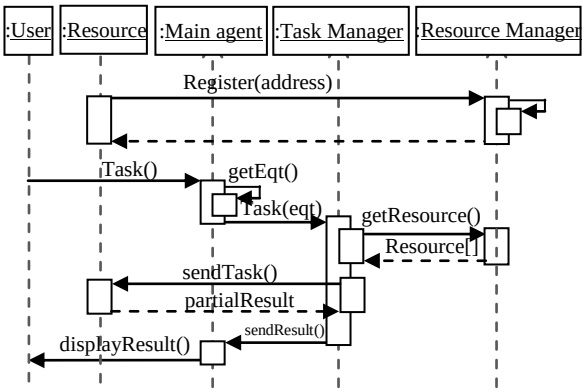

Figure 11. The Sequence Diagram of the System

5. Experimental Result

In this study, we test the system of equations such as:

$X_1 - X_2 + 3X_3 + 2X_4 = 15$

$-X_1 + 5X_2 - 5X_3 - 2X_4 = -35$

$3X_1 - 5X_2 + 19X_3 + 3X_4 = 94$

$2X_1 - 2X_2 + 3X_3 + 21X_4 = 1$

After accepting the resource list from resource manager agent, the task manager agent firstly creates three mobile worker agents and sends them to respective server for calculation in parallel. After collecting the partial results from the mobile workers, the task manager agent calculates the final result and then sends to Main agent to display to the user. The execution time with three worker agents is 1400 ms. And then the task manager agent creates four mobile worker agents for the same equations. We also mark the execution time. Finally, task manager agent creates five mobile worker agents and tests the same equations. Final output result is the same and execution time is reduced. So, the large amount of agents is used, the execution time will reduce.

In this example, the final results are $X_1 = 2$, $X_2 = -3$, $X_3 = 4$, $X_4 = -1$. The time comparison of this experimental results is shown in figure 12.

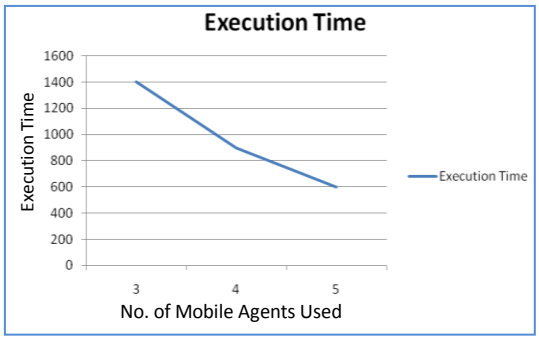

Figure 12. Performance Comparison of Experimental Results

6. Conclusion

The mobile agent is an emerging infrastructure for high performance computing, and parallel and distributed computing is also the efficient computing. In this paper, the methodology, tools and applications of agent-based computing for parallel models are presented. By using this system, numerical system of equations can be solved by using Gauss Jordan Iteration method. With the help of mobile agents, these problems can be solved in parallel on different processors.

The mobile agent successfully dispatched to the destination hosts performed the parallel computing as expected when tested from MAT. As an agent was able to meet the client at a given acceptor site and it could also do the job assigned to it at the nominated host autonomously. Moreover, if the large amount of mobile agents is used for the computing, their execution time can be reduced. The more the large amount of mobile agents used, the smaller the execution time.

6. References

[1] B.Wilkinson and M. Allen. "Parallel Programming." Prentice hall, Upper Saddle River, NJ, 1999.

[2] C. G. Harrison, D. M. Chess, and Kershenbaum, A. "Mobile Agents: Are they a good idea?", Technical report, IBM T. J. Watson Research Center, March 1995.

[3] D. B. Lange and M. Oshima, "Seven Good Reasons for Mobile Agents", Communications of the ACM, Vol.42, No.3, 1999.

[4] Hnin Aye Thant, "Efficient Load Balancing Method for Cluster Based Parallel Applications Using Mobile Agents", in Proceedings of 3rd International Conference on Computer Applications, March 9-10-2005, Myanmar,

[5] Khin Marlar Tun and Thinn Thu Naing, "Parallel and Distributed Computing Models for Mobile Agent", in Proceedings of 2nd International Conference on Computer Application 04, January 8, 2004, Myanmar.

[6] 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.

[7] S. G. Aki, "The Design and Analysis of Parallel Algorithms", Kingston, Ontario, 1989.

[8] S. Papastavrou, G. Samaras and E. Pitoura. "Mobile Agents for WWW Distribted Database Access", proceedings of the 15th International Conference on Data Engineering, 1999.

[9] T. Samaras, M. D. Dikaiakos, C. Spyrou, A. Liverdos, "Mobile Agent Platforms for Web Databases: A Qualitative and Quantitive Assessment", proceedings of first International Symposium on Agent Systems and Applications and third International Symposium on Mobile Agents (ASA/MA 99), October 1999.

[10] W. F. Wong , L. F. Lau, A. L. Ananda, G. Tan. "GUCHA: An Internet-Based Parallel Computing System Using Java", proceeding of 4th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP), December 2000.