Study of Mobile Agents as a Tool for Load Balancing in Distributed Environment

Amit Mishra
Computer Science and Engineering
Jodhpur Institute of Engineering & Technology (JIET)
Jodhpur, India
mishranamit2211@gmail.com

Abstract— Distributed system may be considered as a collection of resources shared by different users. Distributed systems provide methods for sharing and aggregation of resources. Resources may be storage systems, computer machines, hardware devices and other specialized devices. Load balancing is a technique which can be used to improve the performance of distributed system by allowing load migration for the purpose of efficient resource utilization. In this paper, use of mobile agent is proposed as a tool for load balancing

Keywords- mobile agent; load balancing; distributed systems; mobile agent cloning

I. INTRODUCTION

Mobile agents are movable codes which can migrate from one node to another node according to need and requirement [1]. It consists of 3 main parts named-Code, State and Attributes. Mobile agents can reduce the network traffic; provide higher scalability, robustness and flexibility. Mobile agents are created at the home machine and can be distributed to other nodes or host machines for execution. It uses all the resources of host machine for purpose of execution.

A mobile agent developer also has an option to move the code to the data rather than moving data to the code. It also provides the feature of security because we are just moving the code not the actual data. Implementation of mobile agent can be done with many tools like Aglets from IBM, Voyager from ObjectSpace, and Concordia from Mitsubishi etc.

Load Balancing is very important in distributed environment for the efficient use of resources, which are valuable to both system and applications. Distributed load balancing techniques can be classified as: Sender initiated, receiver initiated and symmetrically initiated. [2].

a) **Sender Initiated Schemes:** In this scheme, highly loaded processors dispatch load to lightly loaded processors. We maintain a load counter on each node and increment this load counter whenever node gets new task for completion. When this load counter reaches to a predefined value, the node becomes eligible to transfer the load. Now this node can select the destination node using any one of the following techniques: Random, Probe Based and Shortest.

b) **Receiver Initiated Scheme:** lightly loaded processors generate requests for jobs from the highly loaded processors. The load balancing action is initiated if number of jobs falls below the predefined value. Now this node becomes eligible to work as a receiver.

c) **Symmetrically-initiated schemes:** It is a combination of previously defined schemes in which both sender and receiver can initiate the load balancing process.

So now we have all the basic details regarding the mobile agent and load balancing.

II. MOBILE AGENTS AND LOAD BALANCING

As described in the previous section, Mobile agents are software gypsies who can migrate from one node to another and are platform independent [3]. It executes on the host machine for a specific purpose. Migration of process was a traditional solution of load balancing under the supervision of centralized controller. But it was having all the drawbacks of centralized system like controller failure, repetitive election process in case of controller failure etc. So multi agent system can be used as a solution to this problem and it provides decentralized procedure for load distribution. Each complex application is divided into several independent parts, and each of which is assigned to one or more mobile agents. Now this mobile agent is responsible for the execution of that part. This particular mobile agent searches the network for a host machine which offers most suitable resources for the execution.

Agent can move to the other node during execution if that node is offering more computational resources and lightly loaded, in order to perform load balancing [4].

III. PROPOSED SOLUTION

Here we are proposing mobile agents as a tool for load balancing. Use of mobile agent for load balancing can be described as follows:

a. The home machine partitions the task into some independent sub tasks.
b. It generates mobile agents according to number of subtasks.
c. Configures and assigns each sub task to one or mobile agents.
d. Now mobile agents are free to move in the network for the host selection with appropriate resources.
e. When it gets the suitable host (here we called it machine A), it sends back information to the home machine for tracking purpose.
f. After a predefined time period this current mobile agent makes a clone or copy of itself which includes its state, values of attributes etc. and sends them to its neighbors for searching more suitable working environment. This predefined time period should be calculated smartly so that only single node performs this action at a time. It is important because if many mobile agents perform this action simultaneously than it will create huge network traffic. It will degrade the overall performance of the system.
g. If any mobile agent clone finds the more suitable host machine (called B) it sends this information to the machine that generated this mobile agent clone.
h. Now mobile agent transfers from this Host machine A to this new host machine B. It also informs to the Home machine about this change so that host machine can track the current location of mobile agent. It also sends a message to other clones except machine B for deletion.
i. It follows this process until it does not complete its task and when it completes its task it frees all the resources and get back to the home machine.
j. Now Home machine collects results from each mobile agent and combine them to generate the final output.

So in this way we can achieve Load Balancing in a distributed environment. The whole idea can be shown in the following figure:

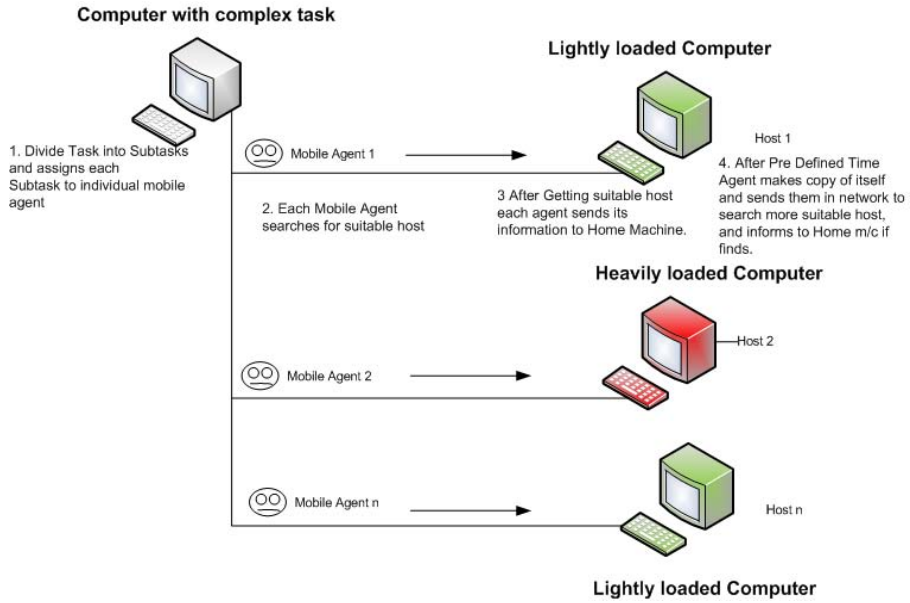

Figure 1. Process of Load Balancing

IV. EXPERIMENTAL SETUP

We implemented the above idea using aglet. It is a mobile agent system provided by IBM [5]. The language used is JAVA, because it provides features like safe execution, platform independence, threading etc. We can divide the whole idea in the following major subparts:-

1) Mobile Agent Module- it contains all the implementation part related to mobile agent. We configured the aglet environment using Tahiti Server, and written the code for intelligent mobile agent.
2) Task Partition & Reassembly Module- This module divides the task into subtasks and it can merge sub-results to make final result. It is a very important module because if the task is not divided properly then

it may affect the whole system. Here for experimental purpose, we are using a task, in which we can easily identify the boundaries for the subtasks.

Now the whole system can be implemented in LAN having speed of 10 Mbps.

V. CONCLUSION AND FUTURE WORK

It can be concluded that well-placed agents in the network reinforce the optimal use of the resources throughout the system. The whole idea depends upon the movement of mobile agent and selection of suitable host. The communication cost and generated network traffic can be further improved for the whole system, which are still a subject of considerable research. This approach is useful in the cases where complex computing is required. The approach can be improved to reduce network traffic.

REFERENCES

[1] G. Hyacinth S. Nwana, "Software Agents: An Overview " Intelligent systems Research, Advanced Applications & Technology Department, Ipswich, Suffolk U.K. Cambridge University Press 1996

[2] Daniel Grosu, "Load Balancing in Distributed Systems: A Game Theoretic Approach

[3] Amit Mishra, Anamika Choudhary "Mobile Agent:Security Issues and Solution" International Journal of Computer Technology and Electronics Engineering (IJCTEE) Volume 2, Issue 6, December 2012

[4] G. Cabri, L. Leonardi, F. Zambonelli, "Weak and Strong Mobility in Mobile Agent Applications".

[5] Maxwell, Y.Aridov and D.Lang, "Agent design patterns: element of agent application design". In proceedings of Autonomous Agents 1998, pp108-115. ACM, 1998.