A Framework for Utilizing Ideal Workstations in a network for Data Processing

Muthamil Selvam .T
Dept. of Computer Engineering
Lecturer (SG)
Seshasayee Institute of Technology,Trichy
muthamilselvam@gmail.com

ABSTRACT

When the client (browser) request for the web page from the server, the server loads both the webpage and the agent to the requested client. The agent act as a middleware between the workstation (clients) and web-server. Whenever the CPU utilization is less than 30%, the server loads the job to the client and processes it .When the CPU utilization increases or when the client closes the web page, the currently processed job in the background is stopped and returned back to the server so that in no way the workstation fails or slowed down. When the user closes the web page , the agent also gets unloaded from the workstation .

Keywords : workstation , processing , agent , middleware , job processing , web server

1.0 INTRODUCTION

Clients (user agents) on the World Wide Web vary greatly in computing power from low-end WAP devices to high-end desktops. It should be possible to build applications, which transparently adapt to the varying user agents. In this thesis, we suggest a mechanism where an application can be written such that the application can be executed on the client side directed by the server. Applications of this type will allow loaded servers to transfer part of the load to the clients to exploit the computing power available at client side.

Since its inception, in last decade, the WWW has witnessed an unprecedented growth. It is the most popular application ever used over the Internet with millions of users using it each day. WWW is being used for variety of applications varying from information sharing, to e-commerce services. It has become a wide source of online information.

Users can very easily get the information spreading all over the world, just at the click of a mouse button. With the explosive increase in the number of users, load on the web servers providing services to users has also increased exponentially. Servers of organizations, which are providing commercial services like e-commerce, financial transactions, and intranet applications are generally overloaded with users' requests. For these organizations, businesses can lose millions of dollars when mission-critical applications are not available. So, it is important that these applications are available at all times and provide good response time to users.

2.0 PROBLEM DEFINITION

In the present web-model, the application programmer mostly writes the programs to be executed fully on the server. They do not take the computing power available on the client side into account. Client side technologies such as Applets and Java scripts can be used to transfer some load from server to client, but the scope is limited to the user interface and small computations only. In the present model, the communication between the client and the server is limited to the program level with CGI programs and applets performing it by request/reply. This makes writing true distributed application, a difficult task for the programmer. There is no fine-grained distribution of load between server and client.

2.1. PROPOSED SYSTEM

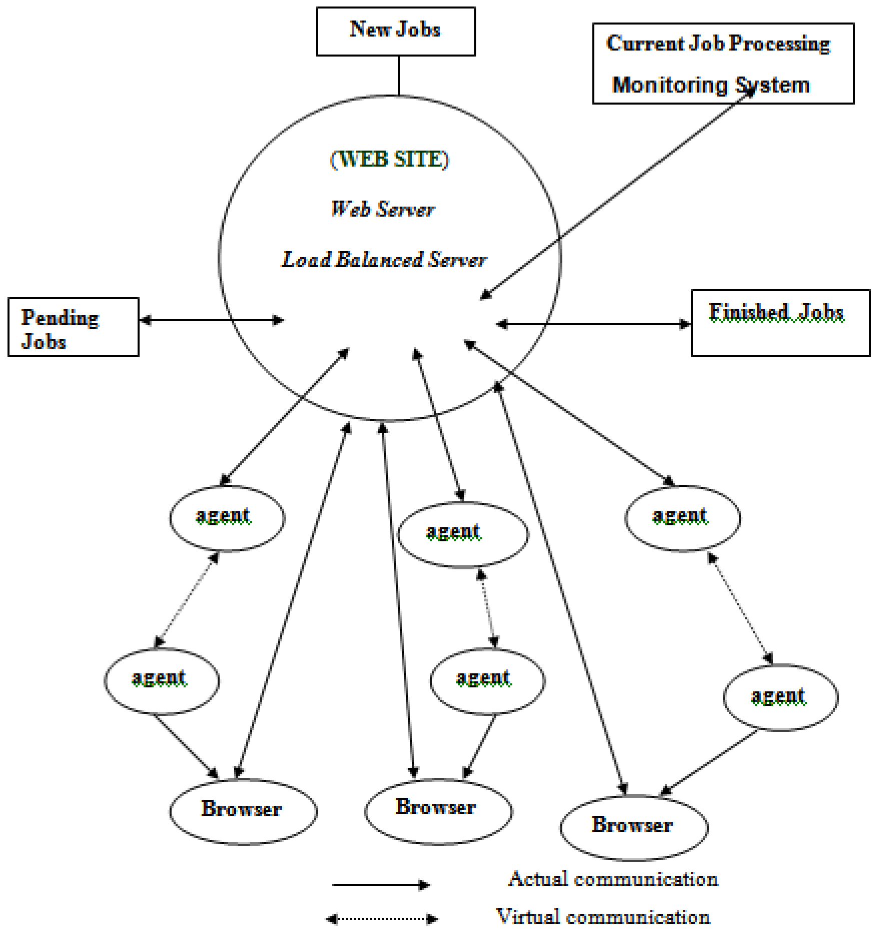
In the present model, all the processing is done at server side only. Computers connected to Internet is performing only downloading the HTML documents & executing simple java/VB scripts in such a way that utilization of the client are very less .Our proposal is to utilize the idle workstations in the internet to process our jobs without affecting their work .

For processing huge amount of tasks, we need dedicated machines which leads to more money & man power. In order to avoid such problems, we have created an automatic load sharing system which loads the jobs from the server to the client where the jobs are processed in the background .While, the man power is reduced and is more economical.

2.2. METHODOLOGY

The logic applied here is that, server contains the webpage .The end-user who accesses this webpage will be loaded with the job from the server, only when the CPU utilization is less than 30% and it is IDLE(i.e., the user is not currently loaded with job) . The information about the status of the user is provided by the agent, which is being attached with the webpage. Thus the jobs are processed with out investment.

2.3. ARCHITECTURE DIAGRAM



3.0 IMPLEMENTATION

The design part has the following modules:

1. Design of Web site .
2. Job Submission for processing & Queue management .
3. Job Distribution to the agent.
4. Job Processing .
   - Encryption.
   - Decryption.
5. Job split and combine .
6. Design and Implementation of master agent (Server side) .
7. Design and Implementation of agents(Client side).

3.1. DESIGN OF WEB-SITE

The second function of the agent is to check whether the job We have designed the website for viewing the online books related to software's. When the client(browser) request our webpage, server loads the html contents and the agent with the webpage to the end user. The agent is an applet program which is resided in the webpage.

The Agent program contains two main functions:

1) It sends the current CPU utilization of the client(browser) to the server.

2) Processes the job if the CPU utilization is less than 30% & terminates the job if the CPU utilization increases.It then sends the finished job or the incomplete job to the server.

The first function of the agent is to send the current utilization of CPU to the server. While reading the books, the CPU utilization of the client will be less. In order to utilize the idle CPU of the client, server sends the job to the client if the CPU utilization of the client is less than 30%.

has to be encrypted or decrypted and now it does the required operation on the job . When the operation has been completed, agent sends the finished job to the server. While processing the job, if the client's (some other application)CPU utilization has increases or if the client closes the webpage, the incomplete job and the percentage of completion is being sent to the server. Once the client closes the webpage, the agent is unloaded from the client(browser).

3.2. JOB SUBMISSION FOR PROCESSING & QUEUE MANAGEMENT

The submission of the job to the master agent is done manually and the processing of the job is done automatically by the server.

It maintains 4 separate lists in a log ,namely:

- New job
- Pending job
- Finished job
- Status list

- The new job list acts as a FIFO queue. It contains a list of jobs for processing with the file name prefixed with the job ID(1.2.2) , size of the file and the type of operation to be done (E-Encryption/D-Decryption).

- When the incomplete job returns to the server, the pending job list contains the half processed job .The remaining unprocessed job is added along with the new job.
- The completely processed job from the client is added to the finished job list.
- The status list shows the client's IP address and their CPU utilization .

3.3. JOB DISTRIBUTION TO AGENTS

The master agent keeps track of the status list & finds if any client's CPU utilization is less than 30 % & if the CPU is idle .If so, the job from the new job list is being sent to the clients for processing .To make the processing to be done automatically, we use a thread. The first job from the new job list is taken The jobs to be processed is distributed to the agents, only if their CPU utilization is less than 30 %.

The CPU utilization is kept track by the Agent.When the CPU utilization is less than 30%, the jobs are being distributed automatically to the agents for processing (Encryption and Decryption).All these distribution takes place across the WWW .

3.4. JOB SPLIT & COMBINE

When the job is not fully processed, the unprocessed job is split into two, which is the pending job(finished job) & new job(unprocessed job) and the new job is sent again to the agent for processing. When the unprocessed job is fully processed, it's combined with the pending job & placed in the " Finished Job " list.

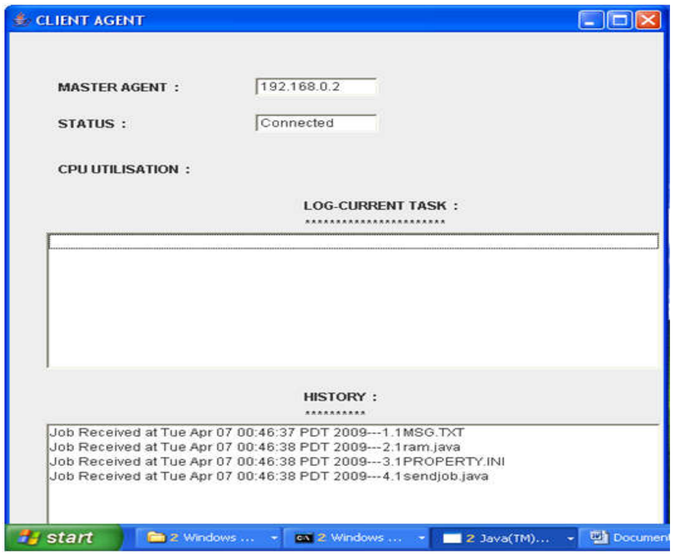
We use 4 list boxes:

One for submitting the job, other for processing half of the job(pending job) & the third list box for combining the jobs & placing the fully processed job .

The last list box is used to keep track of the CPU utilization, agent's IP address & the status of the job whether it is in idle condition or in processing condition .

4.0 OUTPUT

Client Agent



MASTER AGENT : 192.168.0.2

STATUS : Connected

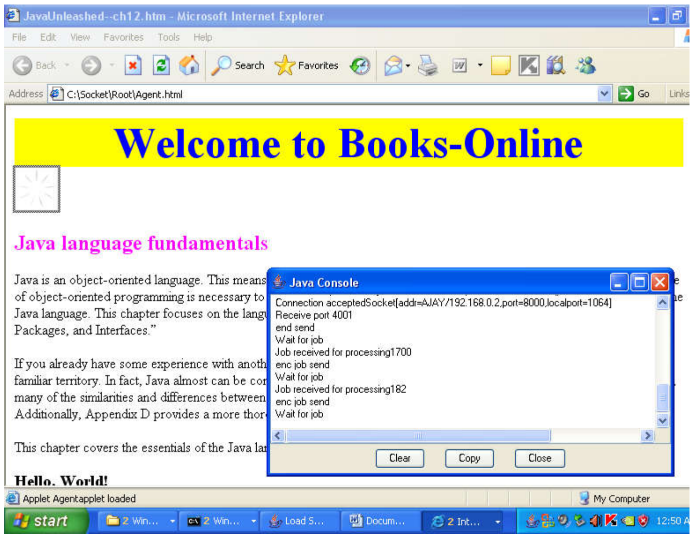
CPU UTILISATION :

LOG-CURRENT TASK :

HISTORY :

```
Job Received at Tue Apr 07 00:46:37 PDT 2009---1.1MSG.TXT
Job Received at Tue Apr 07 00:46:38 PDT 2009---2.1ram.java
Job Received at Tue Apr 07 00:46:38 PDT 2009---3.1PROPERTY.INI
Job Received at Tue Apr 07 00:46:38 PDT 2009---4.1sendjob.java
```

Website


JavaUnleashed--ch12.htm - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\Socket\Root\Agent.html
Go Links

Welcome to Books-Online

Java language fundamentals

Java is an object-oriented language. This means
of object-oriented programming is necessary to
Java language. This chapter focuses on the langu
Packages, and Interfaces.”

If you already have some experience with anoth
familiar territory. In fact, Java almost can be cor
many of the similarities and differences between
Additionally, Appendix D provides a more thor

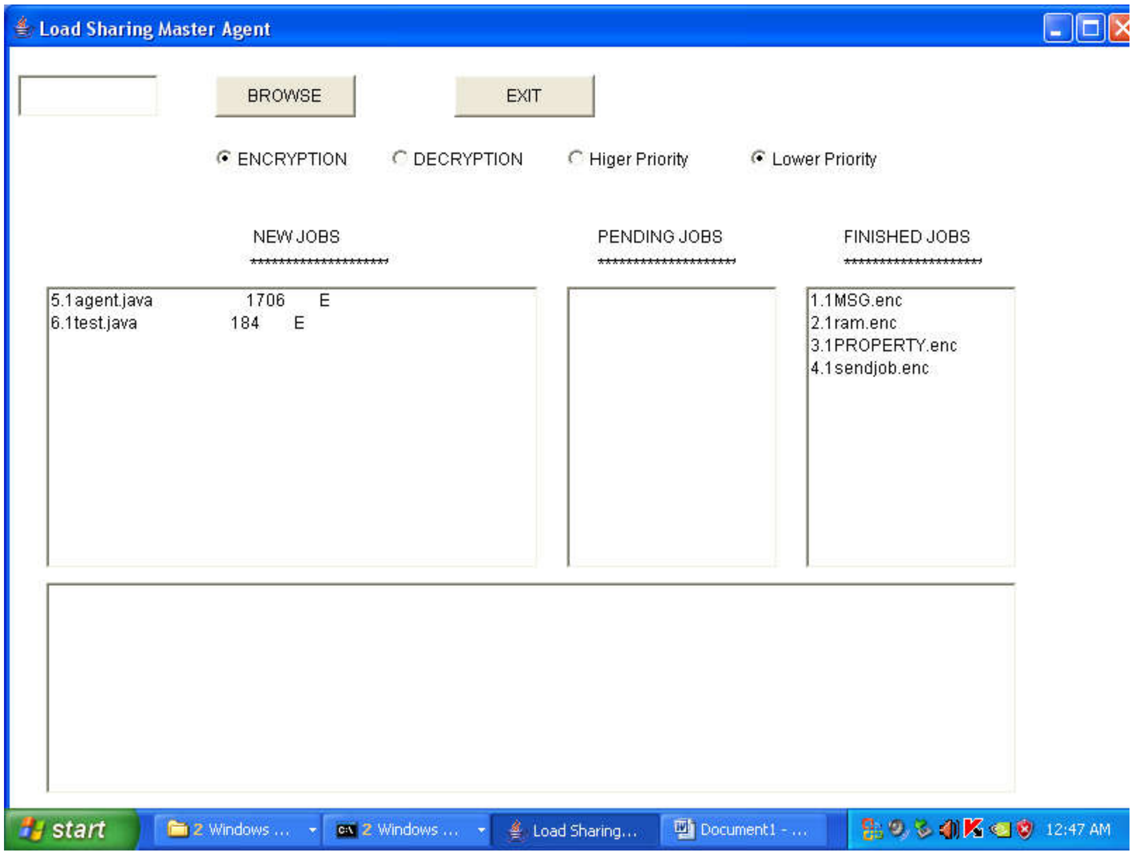
This chapter covers the essentials of the Java lar

Hello, World!

Java Console

```
Connection acceptedSocket[addr=AJAY/192.168.0.2,port=8000,localport=1064]
Receive port 4001
end send
Wait for job
Job received for processing1700
enc job send
Wait for job
Job received for processing182
enc job send
Wait for job
```

Master Agent



[✓ ENCRYPTION] DECRYPTION Higer Priority [✓ Lower Priority]

NEW JOBS			PENDING JOBS	FINISHED JOBS
5.1agent.java	1706	E		1.1MSG.enc
6.1test.java	184	E		2.1ram.enc
				3.1PROPERTY.enc
				4.1sendjob.enc

5.0 CONCLUSION

The proposed system provides a transparent platform for creating load-balanced web applications. It makes response time of a request as least of time required when calculation is done completely on the client and time required calculation is completely done on the server. Hence it gives the better of the two approaches on varying load conditions. This system can be used to reduce load from the highly loaded web server and to provide the good response time to the clients. This is very useful for the applications which requires large amount of computations as well as server resources like databases. This type of application cannot easily be programmed using applets only. Our system will make it easier to program such applications as it will transparently allow to access the database on the server as well as it will automatically migrate the computation code to the client.

6.0 Reference

6.1.Book

[1] Patrick Naughton, Herbert Schildt – “The Complete Reference- Java2” – Tata Mc Graw Hill–3rd Edition

6.2. Journal Article

[1] Balakrishnan, H., Kaashoek, M.F., Karger, D., Morris, R., & Stoica, I., (February 2003), Looking Up Data in P2P Systems, Communications of the ACM, Volume 46 , Issue 2, ACM Press, USA

[2] Ripeanu, M., (July, 2001), Peer-to-Peer Architecture Case Study: Gnutella Network, Technical Report TR-2001-26, University of Chicago, USA.

[3] M.Colajanni, P.S. Yu, and D.M. Dias, “Analysis of Task Assignment Policies in Scalable Distributed Web-Server Systems,” IEEE Trans. Parallel and Distributed Systems, Vol. 9, No. 6, June 1998, pp. 585–600

[4] Ali M Alakeel, “A Guide To Dynamic Load Balancing In Distributed Computer Systems”, International Journal of Computer Science and Network Security, Vol. 10 No. 6, June 2010.