Reducing Latency Gaps in Client Server Environment with Overclocking Technology

Win Win Aung
University of Computer Studies, Mandalay
The Republic of the Union of Myanmar
nng.winwinaung@gmail.com

Abstract

In client server communication, the client is completely dependent on the server for computation logic; while, the application logic is performing for the client. The latency measurement is important and this paper focuses on minimizing the amount of delay between client and server. A single machine's operating system communication combined with a popular speed up method is used. The safest technique to access the smooth performance is the explanation and reintroduction of the overclocking method. The overclocking method can be used efficiently for the system under some special considerations which are beyond the manufactures warranty for rating in unusual processors. In order to reduce the amount of latency time in workload transferring, the distributed environment based overclocking method is used. In this hardware test bed platform, the clear results of a well known intensive benchmark simulation performance tests are presented. The latency reduction of single processor server and client computations are performed interchangeably. According to this, the operational latency on the benchmark test improved significantly as shown in figures.

***Keywords**: Overclocking, latency, workload transferring, benchmark running*

1. Introduction

The requesting server processes to provide a service uses popular distributed model of client server computing. Because different computers are connected over a computer network, they can run efficiently. A client application process or program sends messages to the server from the network and then the server looking up in the database record or file on the server's hard disk. In fact, the client can only perform the functions like display, keyboard typing, less storage of local disks and other unimportant peripherals. However, the server accepts and performs those processes from the client through the network.

Based on those requests, the server processed the database queries and reading files. Most of the server processes are run on the powerful PCs, workstations or on mainframe computers. As the extension of object-based programming model with the structure of large pieces of software into smaller pieces in well defined interfaces, client server model is introduced. The server components are called by the client component which then creates better maintainability and extensibility. All hosts of operating system brands and hardware from multiple vendors are the significant environment of client/server computing. When the number of clients increases, the client server centralized solutions become vital. Server processes can thus run in parallel, each process serving its own set of clients. In the parallel server processes running, each of the own set of clients are served their processes.

Client/server models have the security problem in distributed environment rather than centralized one. The failure of one server severely effects on all of the clients which are rely on them and so lead to rendering problem .The failure of the server makes less progress to the clients connected them.

From the perspective of network fails, the servers are suffered from unreachable, performance and reliability. Long response time becomes from high network traffic of one client to all of the other clients. So, when one or two servers with one or two dozens of clients environments is used, the popular architecture to use is two-tier. But three-tier architecture is more scalable than two-tier architecture. Decoupling clients and servers' intermediary processed is used in three-tier architecture (D. Hemmendinger, A. Ralston, E. D. Reilly, 1997, [1]). There are logically three sources of delay in client server system, a sum of sub delays comprising user think time at a terminal and further ones in the host computer system of the client, two directions of request and reply of the network and finally the server. In typically, various kinds of network communication and complex nodes are linked in clientserver system. They consist of many clients and a variety of server levels. When the local server fulfilled the client's request and then in the transaction process, one server and the network connection is processed. The second level or higher level of the servers until the request can be satisfied by forwarding one network after another. In simplest example, such systems are quite complex topology with the sharing of applications and databases. In this place, the response time is completely dependent of the server level which transaction has to be sent and distribute the response time (A.J. Field, P.G. Harrison and J. Parry, 1993, [2]).

The last twenty years have seen a steady increase in the performance of computer systems at cost efficiency unmatched by any other industry. But the increasing capacity and density of components in the systems has resulted in a steady growth in power supply and cooling requirements. Although manufacturing technology and circuit design considerations both limit processor clock speed, increasingly the first-order limitation for the processor family on a specific platform is power (R.Junan, R.Karthick, R.Freeman, H.Heather, G.Soraya and K.Tom, 2005, [3]). In the paper of the authors (A.Yongwoo and B.Riccardo 2007, [10]), they used overclocking to encourage the device to operate at greater speed with zero cost. To achieve the ratings faster than the manufacturer and so make speeding up the computers' components is overclocking method. For the new computer users, overclocking may be a thriller event. The smart idea of making speed up the processor's ratings becomes more popular in nowadays. But when overclocking method is required to implement successfully, the beginners must have sufficient knowledge about the SPEC of their processor (Beginners Guide, [5]). Moreover, overclocking mainly depends on the hardware ingredients and types of processors, whether it is locked or unlocked processors.

The harmonic balance of overall computing performance of the PC is obtained by implementing on overclocking method and the extra speed results is also shown (H.Ahmed and A.Azween, 2009, [4]). The precise term of overclocking originated from squeezing the other additional performance with no cost. In all resource intensive environments such as game, computing benchmarking programs and obsolete deskop computing, this method is flexible in anyway. Since almost every modern processor and memory module is overclockable to at least a slight degree, there are few reasons not to attempt it. Whether or not the processor is overclockable relies on several factors of the modern processors. It refers to operating a component such as the GPU, CPU, chipset or memory at a higher frequency than originally intended by the manufacturer. Moreover, it makes use of a certain safety margin (often called 'headroom') present in practically every chip. Because the chips are designed to operate smoothly in a wide variety of surroundings and conditions, this often allows overclockers to squeeze a little extra performance out of a chip using its safety margin (A.Cumbur and H.Sumi [9], 2000).

The composition of sections are overclocking and their occurrences (Section 2), the role of client server in latency improvement (Section3), the proposed system implementation (Section4) and the last (Section 5) is conclusion.

2. Overclocking related Functions

Overclocking has been introduced many years ago with the smarter performance results. The following researchers have published successful presentations based on overclocing method. Most of them used these methods on single processor. In fact, the method of overclocking is only the upgrading of the performance of the processor rather than the limitation by the manufacturer (J.Bok.Gyu and K.Sung.Min, 2012, [8]). The process of upgrading the performance of the processor is the useful method for resource intensive workers which mainly focused on graphics and memory. But the latter focusing is dangerous for all of the enthusiasts. Overclocking the CPU is a beloved activity of PC enthusiasts since the dawn of the computer age. The process or "art" of pushing a processor's clock speed is what the manufacturer suggested (Maekinen.S, 2006, [6]). When the time came to reach to speed up the required amount of processor, overclocking became essential with respect to less vulnerability and cost effective. And so, the overclocking is simple to show that processor can function at much higher clock frequencies, giving you a boost on everything from resource intensive tasks to everyday processing including games (G.Brian and T.Josep, 2007, [7]).

2.1. Drawbacks of Overclocking

- Heavily depends on cooling solutions to protect the chip from overheating and significantly lessen lifetime reliability of the system.
- Must be monitored not to exceed the system's temperature about 60 degrees Celsius.
- Stability is also important and must be use programs to make sense of instability conditions.
- Must clean dust from CPU heatsink before overclock.
- Another important thing to regard is that it is also difficult in separating problems between CPU/RAM and PSU (Power Supply Unit).
- Achieving thermal balance and lifetime reliability also plays an essential role.

2.2. Advantages of Overclocking

- Other upgrades are not required to turn existing dead end machines which are absolutely obsolete.
- More suitable for developing countries where a few budgets is gained for new machines.
- The labor costs of setting up the upgrade are almost zero cost in order to gain the efficient performance.
- The energy consumption of overclocking server is insignificant amount compared with traditional client-server computing and so it leads to green for future.

2.3. Major Contributions

Overclocking method is mostly used in standalone PC environment and the proposed system is used in single machine client server.Operational latency problems are solved by overclocking the server.

- A procedure for client overclocking via the server under the safe condition is also introduced.

2.4. The Major Stimulus Factors

- Our PC no longer grunts to run software comfortably due to lack of performance and efficiency.
- Necessary to exercise a known safe clock speed.
- Fun, hobby or competitive reasons.
- Load sharing and load balancing methods for client server architectures are complex for system users rather than overclocking.

2.5. The Objectives

- To achieve flexible computation for all kinds of software platform.
- To protect the system temperature under proper circumstance.
- To emerge intuitive users for future.
- To provide the simplest solution for users this is rare at other techniques.
- To reduce energy consumption for each PC by using centralized system which then share its resources to many clients .

2.6. Useful Equations of Client Overclocking

The following equations are used to measure the energy consumption of CPU and after that we need to measure the temperature of the processor to prevent from overheating which leads to totally damaged.

CPU OCed Watts = DW x (SO/S) x (VO/V) ^ 2 …(1)

Where, the overclocked watts can be obtained from multiplying the default watts (DW) of the processor and the overclocked seconds (SO) which is divided by the original value(S). Finally the overclocked voltage (VO) appeared from the remainder of the initial voltage (V) which is then multiplied by 2.

Proc Temp = (C/W Value x OCed W) + Case Temp …(2)

Then calculate the temperature of the processor, where (C/W) is cycles per watts.

3. Role of Cli/Ser in Latency Improvement

3.1 A Procedure Description

Start

(1) Connect the client with the server.

(2) Select an application to run on client to test the performance.

(3) If the size of the program is greater than the threshold (T) value then capture the SPEC of the server and overclock it.

Else run the task normally.

(4) Keep the temperature to exist between 45 and 65 degree Celsius while overclocking and then keep on the following steps.

Else adjust the temperature to reach the safe condition.

(5)When the app successfully finished, stop overclocking and show running time.

Else capture of the SPEC of the server and overclock again.

(6) Run a memory stress test program to test the stability.

(7) Run a processor stress test program to test the stability.

(8) Show the stability results to validate the process is successfully finished.

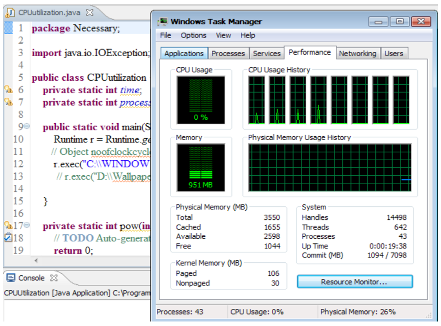
Stop.

The above procedure is used in order to speed up the communication latency time based on the application running. When the app is run slower than the threshold millisecond value, then the client is overclocked by the server. Finally, the app is run successfully.

4. The Proposed System Implementation

4.1. Capturing the Tasks Running

First of all, the applications from server are absorbed by thin client on normal client server system. Then the client performs the tasks and the results are replied to server. In this case, when the client has some problems with running the application and as a result, it leads to consume more energy as well as the thermal. Due to the latency time is increased and need to speed up the client computing environment, finally, thermal reduction is required to detect the processor is functioning well. Before these steps, screen updates are required from server to gain the essential performance. The increasing number of clients may lead to more workload to server.



CPUutilization.java

```
package Necessary;

import java.io.IOException;

public class CPUutilization
    private static int time;
    private static int process

    public static void main(S
        Runtime r = Runtime.ge
       // Object noofclockcycl
        r.exec("C:\\WINDOW
         // r.exec("D:\\Wallpape

    }

    private static int pow(in
        // TODO Auto-generat
        return 0;
```

Console

CPUUtilization [Java Application] C:\Program

Windows Task Manager

Physical Memory (MB)
Total 3550
Cached 1655
Available 2598
Free 1044

Kernel Memory (MB)
Paged 106
Nonpaged 30

System
Handles 14498
Threads 642
Processes 43
Up Time 0:00:19:38
Commit (MB) 1094 / 7098

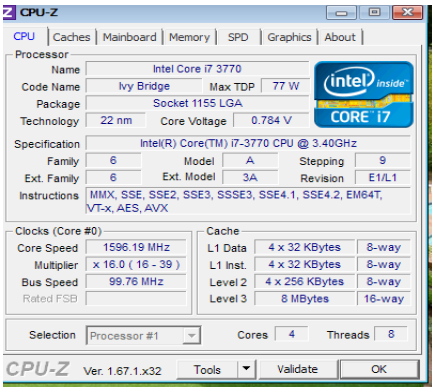
Processes: 43 | CPU Usage: 0% | Physical Memory: 26%

Figure 1. Capturing the Tasks with Java

In order to capture the task of the computer that is currently running, a java program is used to trace about the workloads, in Figure1.

4.2. Using the CPUID CPU-Z Tool

This tool is valuable to detect the components of the CPU and its' related specifications. Moreover, by using this tool, the detailed information of the processor is also captured in Figure 2.



Processor
Name: Intel Core i7 3770
Code Name: Ivy Bridge | Max TDP: 77 W
Package: Socket 1155 LGA
Technology: 22 nm | Core Voltage: 0.784 V
Specification: Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz
Family: 6 | Model: A | Stepping: 9
Ext. Family: 6 | Ext. Model: 3A | Revision: E1/L1
Instructions: MMX, SSE, SSE2, SSE3, SSSE3, SSE4.1, SSE4.2, EM64T, VT-x, AES, AVX

Clocks (Core #0)
Core Speed: 1596.19 MHz
Multiplier: x 16.0 (16 - 39)
Bus Speed: 99.76 MHz
Rated FSB

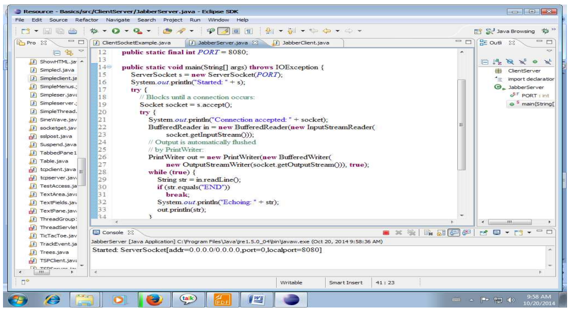
Cache
L1 Data: 4 x 32 KBytes | 8-way
L1 Inst.: 4 x 32 KBytes | 8-way
Level 2: 4 x 256 KBytes | 8-way
Level 3: 8 MBytes | 16-way

Selection: Processor #1 | Cores: 4 | Threads: 8

CPU-Z Ver. 1.67.1.x32

Figure 2. Detecting the SPEC of the Processor

4.3. Making the Connection between Client Server



Resource - Basics/src/ClientServer/JabberServer.java - Eclipse SDK

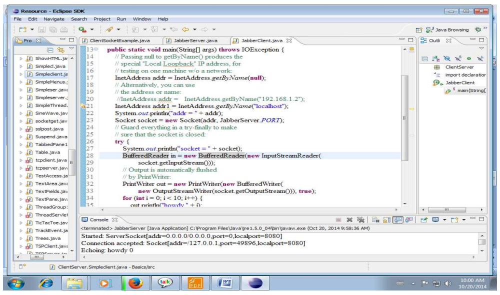
```
    public static final int PORT = 8080;

    public static void main(String[] args) throws IOException {
        ServerSocket s = new ServerSocket(PORT);
        System.out.println("Started: " + s);
        try {
            // Blocks until a connection occurs:
            Socket socket = s.accept();
            try {
                System.out.println("Connection accepted: " + socket);
                BufferedReader in = new BufferedReader(new InputStreamReader(
                        socket.getInputStream()));
                // Output is automatically flushed
                // by PrintWriter:
                PrintWriter out = new PrintWriter(new BufferedWriter(
                        new OutputStreamWriter(socket.getOutputStream())), true);
                while (true) {
                    String str = in.readLine();
                    if (str.equals("END"))
                        break;
                    System.out.println("Echoing: " + str);
                    out.println(str);
                }
```

Console

```
JabberServer [Java Application] C:\Program Files\Java\jre1.5.0_04\bin\javaw.exe (Oct 20, 2014 9:58:36 AM)
Started: ServerSocket[addr=0.0.0.0/0.0.0.0,port=0,localport=8080]
```

Figure 3. Server Side Configuration

Figure 3 is the simple java program to connect the dependent client from server side. This system is successfully implemented on the same server as client/server manner.



Resource - Eclipse SDK

```
    public static void main(String[] args) throws IOException {
        // Passing null to getByName() produces the
        // special "Local Loopback" IP address, for
        // testing on one machine w/o a network:
        InetAddress addr = InetAddress.getByName(null);
        // Alternatively, you can use
        // the address or name:
        //InetAddress addr =   InetAddress.getByName("192.168.1.2");
        InetAddress addr1 = InetAddress.getByName("localhost");
        System.out.println("addr = " + addr);
        Socket socket = new Socket(addr, JabberServer.PORT);
        // Guard everything in a try-finally to make
        // sure that the socket is closed:
        try {
            System.out.println("socket = " + socket);
            BufferedReader in = new BufferedReader(new InputStreamReader(
                    socket.getInputStream()));
            // Output is automatically flushed
            // by PrintWriter:
            PrintWriter out = new PrintWriter(new BufferedWriter(
                    new OutputStreamWriter(socket.getOutputStream())), true);
            for (int i = 0; i < 10; i++) {
                out.println("howdy " + i);
```

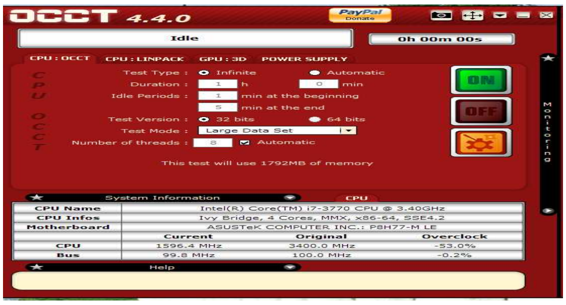
Console

```
<terminated> JabberServer [Java Application] C:\Program Files\Java\jre1.5.0_04\bin\javaw.exe (Oct 20, 2014 9:58:36 AM)
Started: ServerSocket[addr=0.0.0.0/0.0.0.0,port=0,localport=8080]
Connection accepted: Socket[addr=/127.0.0.1,port=49896,localport=8080]
Echoing: howdy 0
```

Figure 4. Client Side Configuration

By using another client side invocation program in Figure 4, we can detect the port numbers and the running tasks on it. The detailed information of the client is captured and the show the running conditions specifically.

4.4 The Phase of Overclocking (OCing) Server with OCCT Tool



Idle — 0h 00m 00s

Test Type : ● Infinite ○ Automatic
Duration : 1 h 0 min
Idle Periods : 1 min at the beginning
5 min at the end
Test Version : ● 32 bits ○ 64 bits
Test Mode : Large Data Set
Number of threads : 8 ☑ Automatic

This test will use 1792MB of memory

System Information — CPU

CPU Name	Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz		
CPU Infos	Ivy Bridge, 4 Cores, MMX, x86-64, SSE4.2		
Motherboard	ASUSTeK COMPUTER INC.: P8H77-M LE		
	Current	Original	Overclock
CPU	1596.4 MHz	3400.0 MHz	-53.0%
Bus	99.8 MHz	100.0 MHz	-0.2%

Help

Figure 5. Overclocking tool running

This Figure 5 is the most popular tool to speed up the processor with precise, concise and timely data monitoring. This tool shows the detailed SPEC of the processor including core information and overclocked conditions.

4.5. Temperature Conditions with OCing

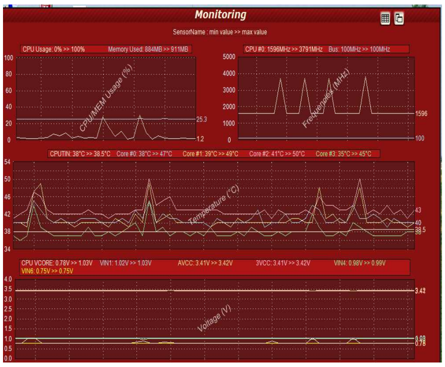
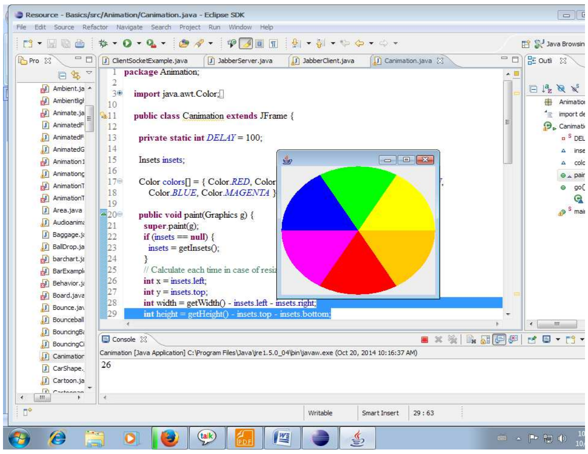


Figure 6. Description of Temp Situations

The temperature about the processor is also presented in order to adjust and avoid worst case conditions in Figure 6. The voltage, temperatures, frequencies and at least CPU usage is presented in this tool.

4.6. Simple Animation Program Running on Client

In this Figure 7, the animation with intensive resource on client machine is run by sending server push action. This animation shows how much power consumption as well as taking time in milliseconds.

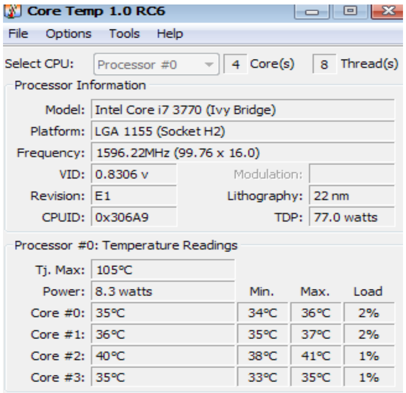

Resource - Basics/src/Animation/Canimation.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
ClientSocketExample.java
JabberServer.java
JabberClient.java
Canimation.java
Console
Writable
Smart Insert
29 : 63

```
package Animation;

import java.awt.Color;

public class Canimation extends JFrame {

  private static int DELAY = 100;

  Insets insets;

  Color colors[] = { Color.RED, Color
     Color.BLUE, Color.MAGENTA }

  public void paint(Graphics g) {
    super.paint(g);
    if (insets == null) {
      insets = getInsets();
    }
    // Calculate each time in case of resiz
    int x = insets.left;
    int y = insets.top;
    int width = getWidth() - insets.left - insets.right;
    int height = getHeight() - insets.top - insets.bottom;
```

```
Canimation [Java Application] C:\Program Files\Java\jre1.5.0_04\bin\javaw.exe (Oct 20, 2014 10:16:37 AM)
26
```

Figure 7. Simple Intensive Animation Program

4.7. The Results of Core Temp



Select CPU: Processor #0 | 4 Core(s) | 8 Thread(s)

Processor Information

Model:	Intel Core i7 3770 (Ivy Bridge)		
Platform:	LGA 1155 (Socket H2)		
Frequency:	1596.22MHz (99.76 x 16.0)		
VID:	0.8306 v	Modulation:	
Revision:	E1	Lithography:	22 nm
CPUID:	0x306A9	TDP:	77.0 watts

Processor #0: Temperature Readings

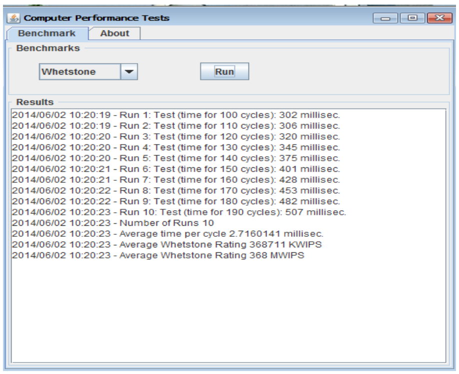
		Min.	Max.	Load
Tj. Max:	105°C			
Power:	8.3 watts			
Core #0:	35°C	34°C	36°C	2%
Core #1:	36°C	35°C	37°C	2%
Core #2:	40°C	38°C	41°C	1%
Core #3:	35°C	33°C	35°C	1%

Figure 8. After Running Results of CoreTemp

When we run the intensive programs on client side, what else we need is to run the Core Temp in order to achieve the Watts usage and the temperature of the respective cores in Figure 8. Without using relevant temperature measurement, the system will lead to completely down to the operating system and will end all of the processings.

4.8. Benchmark Test for Latency before overclocking

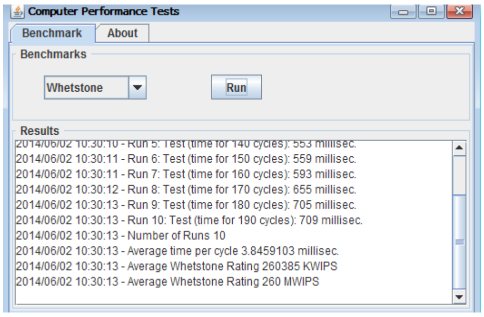
To achieve the conditions of the latency before and after overclocking, benchmark program is also used with 2.7160141 ms for latency in Figure 9. This is the result of the performance of the processor before running overclocking.



```
2014/06/02 10:20:19 - Run 1: Test (time for 100 cycles): 302 millisec.
2014/06/02 10:20:19 - Run 2: Test (time for 110 cycles): 306 millisec.
2014/06/02 10:20:20 - Run 3: Test (time for 120 cycles): 320 millisec.
2014/06/02 10:20:20 - Run 4: Test (time for 130 cycles): 345 millisec.
2014/06/02 10:20:20 - Run 5: Test (time for 140 cycles): 375 millisec.
2014/06/02 10:20:21 - Run 6: Test (time for 150 cycles): 401 millisec.
2014/06/02 10:20:21 - Run 7: Test (time for 160 cycles): 428 millisec.
2014/06/02 10:20:22 - Run 8: Test (time for 170 cycles): 453 millisec.
2014/06/02 10:20:22 - Run 9: Test (time for 180 cycles): 482 millisec.
2014/06/02 10:20:23 - Run 10: Test (time for 190 cycles): 507 millisec.
2014/06/02 10:20:23 - Number of Runs 10
2014/06/02 10:20:23 - Average time per cycle 2.7160141 millisec.
2014/06/02 10:20:23 - Average Whetstone Rating 368711 KWIPS
2014/06/02 10:20:23 - Average Whetstone Rating 368 MWIPS
```

Figure 9. Before Running OCCT

4.9. Benchmark Test after overclocking



```
2014/06/02 10:30:10 - Run 5: Test (time for 140 cycles): 553 millisec.
2014/06/02 10:30:11 - Run 6: Test (time for 150 cycles): 559 millisec.
2014/06/02 10:30:11 - Run 7: Test (time for 160 cycles): 593 millisec.
2014/06/02 10:30:12 - Run 8: Test (time for 170 cycles): 655 millisec.
2014/06/02 10:30:13 - Run 9: Test (time for 180 cycles): 705 millisec.
2014/06/02 10:30:13 - Run 10: Test (time for 190 cycles): 709 millisec.
2014/06/02 10:30:13 - Number of Runs 10
2014/06/02 10:30:13 - Average time per cycle 3.8459103 millisec.
2014/06/02 10:30:13 - Average Whetstone Rating 260385 KWIPS
2014/06/02 10:30:13 - Average Whetstone Rating 260 MWIPS
```

Figure 10. Results after Running OCCT

These clear results showed in Figure 10 with the significant improvement of the latency 3.8459103 ms faster after overclocking.

4.10. The Flow of the Proposed Client Server System

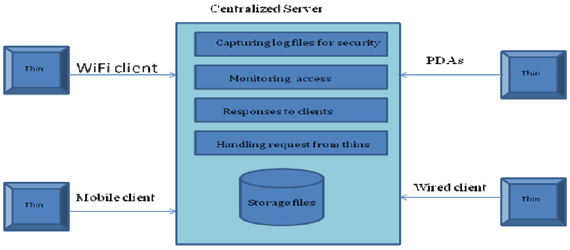


Figure 11. The Proposed Simplest Centralized Structure with Different Client Types

The above Figure 11 shows multiple clients with centralized server. Only the server is responsible to monitor the conditions of the clients with stateful behavior. In this figure, all kinds of clients are presented including wire connected clients, wireless clients, mobile clients and Personal Digital Assistants (PDAs). In order to cover the whole system, these clients are named thins.

5. Conclusion and Future Work

The developed education environment considers how to retain the obsolete PCs, how to reduce the amount of energy consumption and how to recycle existing equipments as they have sufficient technical supports, so thin clients can be used as further extension. So, the system vividly points out the efficient memory usage of obsolete clients with the help of server overclocking method and even it can greatly save the green environment. The most significant fact is that clients' running time latency in milliseconds of the benchmarks has been improved as shown in figure. In order to improve long lasting of the client's lifetime, this method could give the best strategy for the entire client server computing environments. The resource intensive applications such as 3D models rendering over the network, oil exploration, weather forecasting and storm movement direction with the use of some image processing methods may be further extension of this work.

References

[1] D. Hemmendinger, A. Ralston, E. D. Reilly, December 5, 1997.

[2] A.J. Field, P.G. Harrison and J. Parry, 1993.

[3] R.Junan, R.Karthick, R.Freeman, H.Heather, G.Soraya and K.Tom, 2005.

[4] H.Ahmed and A.Azween, IJCSNS International Journal of Computer Science and Network Security, VOL.9.

[5] Beginners Guides: Overclocking the CPU,Motherboard and Memory.

[6] Maekinen.S , CPU & GPU Overclocking Guide,2006.

[7] G.Brian and T.Josep, supported in part by the National Science Foundation under grant CCR-0325603; DARPA under grant NBCH30390004; DOE under grant B347886; SRC GRC under grant 2007-HJ-1592.

[8] J.Bok.Gyu and K.Sung.Min, International Journal of Smart Home ,Vol. 6, No. 4, October, 2012.

[9] A.Cumbur and H.Sumi, Proceedings of the Third IEEE Workshop on Mobile Computing Systems and Applications (WMCSAí00), 2000.

A.Yongwoo and B.Riccardo, Euromicro Conference on Real- Time Systems, IEEE, 2007.