**Computer Systems Technology**

British Columbia Institute of Technology

COMP 8005 - Assignment2- Report

Albert Huang&

Aiyan Ma

Mar 5, 2018

# Table of Contents

[Table of Contents 1](#_Toc1667016842)

[1. Summary 3](#_Toc874767404)

[2. Introduction 4](#_Toc2139831603)

[3. Background 5](#_Toc266633914)

[4. Discussion 6](#_Toc1577232796)

[5. Conclusion 9](#_Toc568634831)

# Summary

The purpose of this work is to design and compare the 3 models of doing high performance servers:

1. Multi-thread ( traditional way)
2. Poll ( Level ttrigger)
3. EPoll ( Edge trigger)

To archive it, Three servers are designed and compared by using java nio library.

For EPoll and Poll server, message cache, state machine with java nio are utilized with the intention to enhance the scalability and performance ( see details in the design work). For the Multi thread server, one thread per connection model is used to accept and handle the client request. The client, named SuperClient, also designed to connect and send packets in such a way that connections and packets are generated as much as possible. For these 3 models, the scalability(connection) and the performance both are analyzed.

The python version servers are also introduced in this test. To do so, Python 3.x was used in a Linux Fedora OS on lab computers.

To analyze and compare these three server mode, a set of performance tests were developed to help show any differences between them. Testing programs were setup on lab’s computers in attempts to test and analyze performance, notably in regards to time consuming, CPU usage etc.

My hypothesis was that the select mode server (consider python version select mode was limited by file descriptor number[1024] so I would use level triggered epoll instead of select mode and do the comparison with edge triggered epoll) would be outclassed by the my epoll mode server and the multi-thread server would be outclassed by both of the select mode and epoll mode server. But our results proved more complicated. Time consuming were comparable between the different models, but performance of my edge triggered epoll was unable to defeat the level triggered mode. The level triggered mode performance in most cases is even way better than edge triggered epoll.

My discoveries were interesting. Not all servers slowed considerably as the connection number went up. My multi-threaded version show its vulnerable when the connection come to 49xx and its respond time get extremely long the average number can arrive to above 600ms. While both level triggered and edge triggered server never get crushed even the connection count comes to 110,000. Overall, the two kinds of epoll server performed good, the only bottleneck we find is the traffic, if the traffic get a number both of them would not able to work stable as usual.

# Introduction

The purpose of this assignment was to analyze and compare the performance between the multi-threaded, select mode and epoll mode server implementations. The tests were developed using Python3 and were run on the Linux Fedora OS using lab computers and Ubuntu OS using my laptop.

The goal of this assignment is to have a better understanding of the different types of server perform on same platforms compared to each other.

# Background

Because of the computer technology evolution, the Multi thread is the traditional way in java j2SE1.1 nearly 20 years ago, then java nio launched in J2SE1.4 and become popular in Java5. From these facts, the hypothesis is, the later technology, as they declare, the more high performance can be archived.

Similar case happen for Python to introduce the multi-thread, Poll and Epoll technology, although maybe not so explicitly.

There fore, from scalablility and performance view, we can anticipate that

**EPoll>Poll>Multi-Thread.**

That’s our hypothesis.

We want to verify to confirm or get other findings thought our design and test.

About python version, programs were written for all three servers types as well as a common client using multi-processing and multi-threading.

Our client both java and python version take several arguments including server ip, port, working tread number, packet number. The python version client would generate a number of processes and each process create user defined number of working threads. The processes number is equal to the number of CPUs on the host. Then each thread connects to the server and send a number of data. After that an echo of that data would be received in response. When everything is done, an conclusion about how long time this task using will be print out to the terminal.

The Multi-Threaded server creates a set of worker threads to manage each connection request. Data is echoed back and forth to client until no more data incoming. When the work threads finish the job, then the main stream will print the number of connections it deal with this time.

The select mode server is very straight forward, since python3 supply a good library named selectors, and the default trigger for epoll is level trigger. So I just need to listens for connections on a socket and if there are events happened make it direct to callback functions to deal with.

The epoll mode server is alomst the same as select mode, since in python3 the only way of using edge trigger is to override the register method which is in the selectors lib. Add the flag ET accompany with read or write event flag.

# Discussion

My hypothesis was that the multi-threaded server would be outclassed by the rest of servers and the select mode server would be outclassed by the epoll mode server.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| server type | # conn | # host | # clients/host | avg res | 2k | 4k | 4.8k |
| multi-Thread | 4k | 1 | 4800 | 652.639 | 0.859 | 165.499 | 347.516 |
|  | 10k | 2 | 4800 | N/A | N/A | N/A | N/A |
|  | 20k | 5 | 4800 | N/A | N/A | N/A | N/A |
|  | 40k | 10 | 4800 | N/A | N/A | N/A | N/A |
|  | 100k | 24 | 4800 | N/A | N/A | N/A | N/A |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| level trigger | 10k | 2 | 4800 | 0.101 | 0.2 | 0.178 | 0.162 |
|  | 20k | 5 | 4800 | 0.089 | 0.209 | 0.139 | 0.129 |
|  | 40k | 10 | 4800 | 0.087 | 0.177 | 0.156 | 0.155 |
|  | 100k | 24 | 4800 | 0.07 | 0.227 | 0.194 | 0.159 |
|  |  |  |  |  |  |  |  |
| edge trigger | 10k | 2 | 4800 | 0.157 | 0.281 | 0.206 | 0.187 |
|  | 20k | 5 | 4800 | 0.105 | 0.222 | 0.186 | 0.182 |
|  | 40k | 10 | 4800 | 0.108 | 0.297 | 0.254 | 0.257 |
|  | 100k | 24 | 4800 | 0.103 | 0.291 | 0.225 | 0.213 |

To test my hypothesis, a formal series of tests were run with in the lab platform. The time consuming and CPU usage were averaged and compared. I also collect data of each server approach for proving.

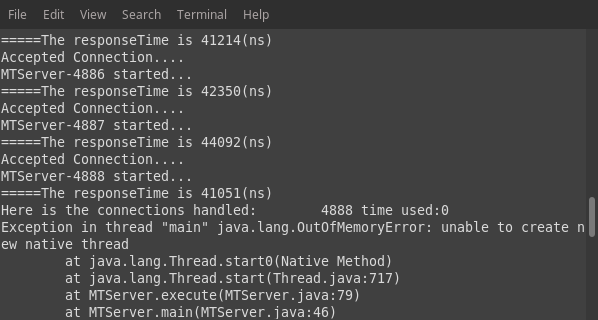
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| server type | # host | avg res | 10k | 20k | 40k | 80k | 110k |
|  |  |  |  |  |  |  |  |
| level trigger | 2 | 0.101 | 0.101 |  |  |  |  |
|  | 5 | 0.089 | 0.107 | 0.084 |  |  |  |
|  | 10 | 0.087 | 0.111 | 0.103 | 0.086 |  |  |
|  | 24 | 0.07 | 0.115 | 0.085 | 0.081 | 0.071 | 0.058 |
|  |  |  |  |  |  |  |  |
| edge trigger | 2 | 0.157 | 0.164 |  |  |  |  |
|  | 5 | 0.105 | 0.125 | 0.106 |  |  |  |
|  | 10 | 0.108 | 0.184 | 0.141 | 0.114 |  |  |
|  | 24 | 0.103 | 0.175 | 0.142 | 0.127 | 0.101 | 0.098 |

(2k, 4k, 4.8k means there are 2000, 4000, 4800 connections)

**The results show that:**

1. Similar tests produced similar results (level triggered server is better than edge triggered one).
2. The multi-thread server failed around 5000 connections.
3. The graph shows both level triggered or edge triggered server will get better average response times when the connections go up.

According to the screen shot below, the multi-thread version server would down around the connection come to 4888, data from java version multi-thread; while python version is very similar.



So what make level triggered mode perform better than edge triggered one? This question trigger me to go further test by using perf. And I get data like below shows:

(ET means Edge Triggered server ; LT means Level Triggered server)

We can find from the bar chart above, that when the connection count the same, the edge triggered server always spend more task-clock than level triggered version. While we think we find the reason: the number of instructions. The edge triggered server always got double size of instructions by comparing with the level triggered one. Even though the data of ipc (instruction per second) indicate that the speed of edge triggered is litter faster than level triggered version, but consider the huge gap of number of instructions it won’t help too much. The context-switches also win level trigger server a bit since the edge trigger try to avoid much context-switches, but the same reason, instruction number, it won’t help a lot. And in reality experiment, we found that level trigger can guarantee all the event would happen because epoll\_wait() would always remind to finish the work it fail to done last time; while edge triggered one would always lose event sometimes even connections, this is may because it epoll\_wait() would only inform once. The mystery thing to me is why the instruction number can have a so big gap.

# Conclusion

From the data collected, it can be seen that Multi-thread server is the weakest one in these three.