

# CS 450 Computer Networks

## Programming Assignment 2 Report

### Performance Monitoring and Evaluation of a Concurrent File Server

Waseem Ahmad  
ECE Department, University of Illinois at Chicago  
wahmad@ece.uic.edu

## 1 Introduction

The report discusses performance measurement, evaluation and problems encountered to a concurrent file server while serving multiple clients at one time based upon a simulated environment. It also proposes ways to cope with these problems. The experiment is based upon a following scenario. Let us assume that there is a popular internet content distribution server and attracts thousand of clients each requesting for content that has huge size. A few questions are to be answered here. How many clients at a time will the server be able to serve without really undermining the QoS requirements of each individual client? How can the server improve total throughput of the network resources yet not affecting individual client requirements? These two questions are inter-related and this report is aimed at answering these questions.

I implement a concurrent file server mimicking huge content by using virtual files. Multiple clients are simulated by spawning multiple copies of the client programme each requesting the server to be served with a virtual file. A virtual file is best suited for simulating huge content as it has infinite size and is also easy to handle as it does not require OS file descriptors. Then there is a monitor to which clients have to send reports of total throughput experienced in the last second over a UDP socket. The clients are distinguished by Monitor based upon their IDs, which are assigned to them manually as a command line argument while executing them. The source code of Server (server.c), Client (client.c) and Monitor (monitor.c) is given at the end of report along with Makefile.

The simulated environment consisted of following settings

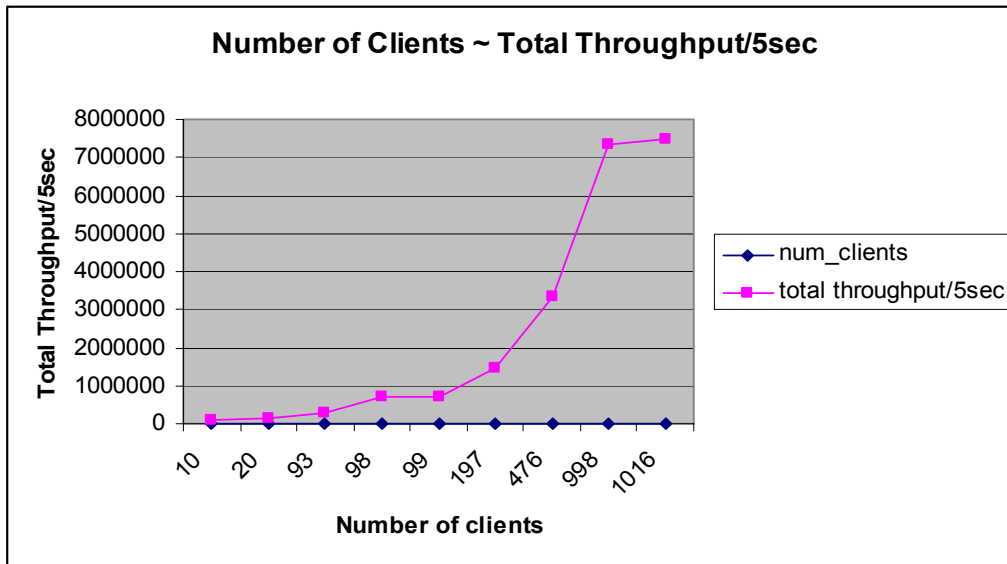
1. Server running on safire.ece.uic.edu. Safire [1] is a Beowulf cluster having 22 Intel XEON 2.4 GHz processors.
2. Clients running on multimedia.ece.uic.edu.
3. Monitor running on mia.ece.uic.edu.

The following observations were made

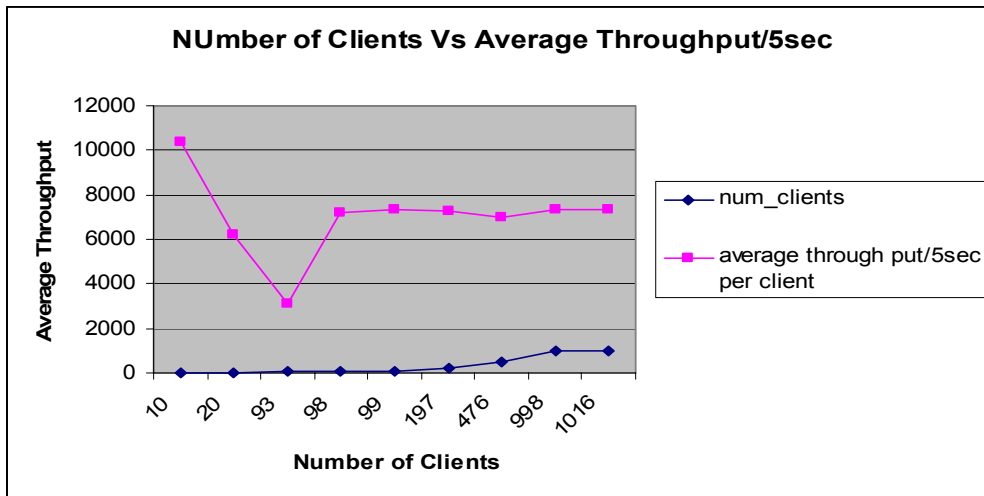
1. The server always stopped accepting more clients when the number of clients became more than 1022. At that time, server is not allowed to terminate but in fact is kept alive to serve the already connected clients. The number of clients being served strictly depends upon the server resources. Because each new client results in creation of a new process (using fork () system call) while the OS itself has imposed restriction on the number of user processes. One possible solution to this problem is to remove overhead associated with the fork system call by making use of select command from UNIX socket API which enumerates active client sockets individually and selects one of them to be served at one time. Different scheduling techniques can be applied to serve all clients appropriately.  
It is worth mentioning here that I did experiments by varying queue size of pending requests. Even increasing it to a large value (1000) did not result in more clients being accepted at a time.
2. The total throughput increases up to a certain limit on increasing the number of clients. Just when it starts deteriorating, server stops accepting more connections. See

graph1. This behavior is acceptable as it ensures good Quality of Service (QoS) to the already connected clients. We identify in the next section what System resource is most important in handling large number of clients and still maintaining reasonable QoS to each of them.

- The individual throughput remains more or less consistent but again at the time it started deteriorating, the server stopped accepting more connections (see Graph2).



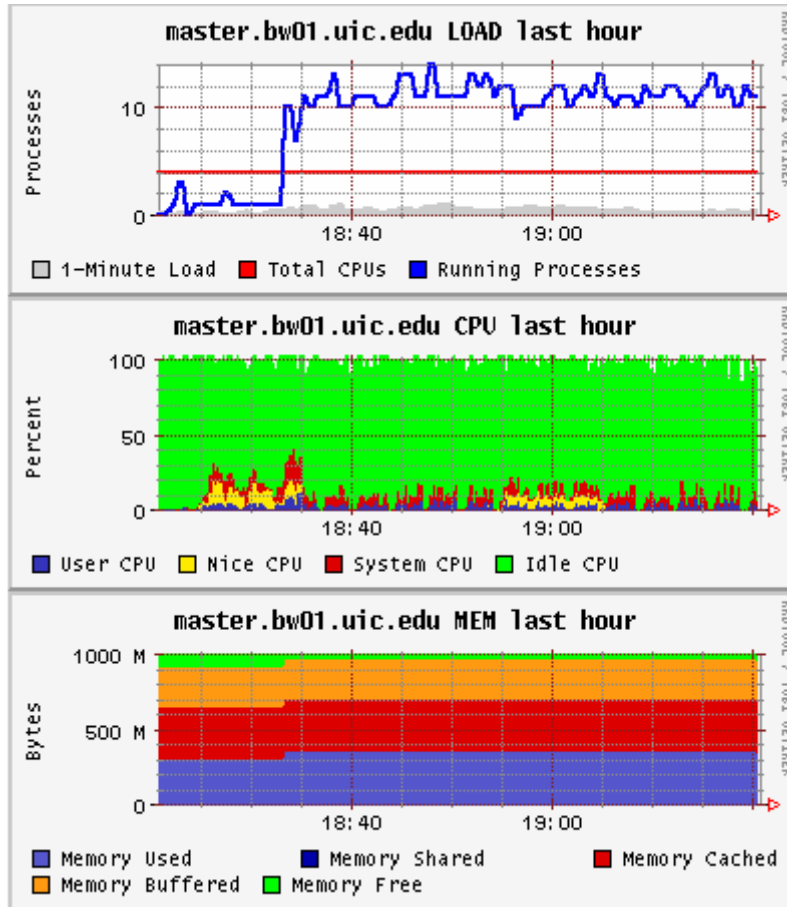
Graph1. Graph of Number of Clients against total throughput over a 5 second period. Note that at the when server stops accepting connections i-e when number of clients reaches 1022, the total throughput is saturated.



Graph2. Graph between Number of clients and average throughput per client over a 5 second period.

## 2 Interesting Observations

Ganglia cluster status report [2] for Safire during the time simulations were being run



Note that master.bw01.uic.edu is the head node of the cluster with 4 Intel XEON processors running at 2.4 GHz and total RAM of 1GB. It has two interfaces external interface is named safire.ece.uic.edu and the internal is \*.bw01.uic.edu.

The first graph (the top most one) shows the number of processes active at one time during execution of the simulation, middle one shows the CPU usage and the third at the bottom shows the memory usage. CPU usage is pretty less than anticipated because of high memory access rate. The memory usage graph shows that its memory which becomes the bottleneck in an effort to server more and more clients. Thus increasing memory buffers should affect performance positively. This emphasis on memory increase should not undermine the effectiveness of Network resources. This simulation was carried in a uniform High Bandwidth environment where server and clients are on the same physical network (UIC LAN) and not on public internet. I estimate that on public internet there will be lots of network bandwidth bottlenecks and the server hardware memory will become the secondary issue.

### 3 Links

1. Safire. Multimedia Lab Beowulf Cluster. <http://safire.ece.uic.edu>
2. Ganglia Cluster Status Report for Safire. <http://safire.ece.uic.edu/ganglia/>