Implementation of Scalable Servers - Design Work & Testing

David Tran - A00801942 | Cole Rees - A00741578

COMP 6D

COMP 8005 - Assignment 2

British Columbia Institute of Technology

Aman Abdulla

Monday, February 24 2014

Table of Contents

Server Characteristics

Criteria 1: a server should be receiving something from the client
Criteria 2: the server should “service” the client in some way
Criteria 3: the server should do some amount of “heavy work”
Criteria 4: all three implementations of the server should be similar in functionality
Criteria 5: server shall maintain a list of all connected clients, number of requests and amount of data transferred to each client

1. Send a string back to the client. Same string that was received.
2. Instead of just echoing back the integer, why not echo back the time it took to process it?
3. Code first in multithreads, then branch off into epoll and select and build those in tandem.

Client Characteristics

Criteria 1: the client should be no more than an echo client (send and receive)
Criteria 2: the client should send variable lengths of text strings
Criteria 3: the client should be able to send multiple times defined by the user
Criteria 4: clients shall keep track of the number of requests made to the server, amount of data sent to the server and the amount of time it takes the server to respond

1. Client should send a user-varied string, but it should be the same.
2. Sending multiple times can be factors of loops in shell script executing our program or requests made by a single client.

Design

Server Pseudo-Code

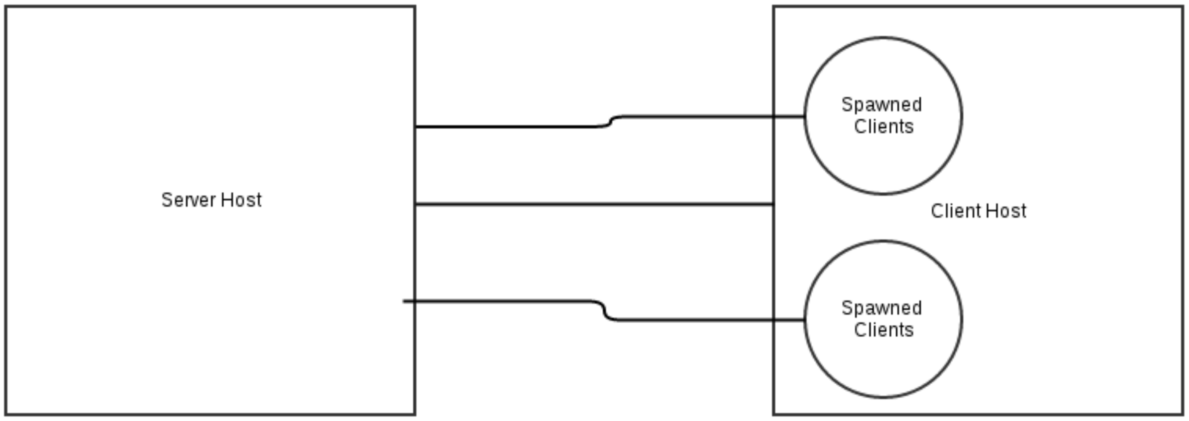
```
// create socket
// bind socket
// while loop starts
    // listen for connections and...
    // wait until a client connects
    // when a connection is received
        // add client to list
        // create a thread
            // do the prime number decomp.
            // once done, send the results back to the client
            // end the session
// {end of while loop}
// close socket
// work for the thread…
    // while the client is still connected and sending stuff
        // receive the number
        // increment the request of this client
        // get timer 1.
        // run the decomp.
        // get timer 2 when decomp is done
        // timer2 - timer1 = time to process number
        // echo back the factors + time to process
    // remove this client from the list
    // exit
```

Client Pseudo-Code

```
// have a set of integers to give to the server
// create the socket
// connect to the client
// while connected…
    // for( i=0; i < the set of integers; i++)
        // increment the amount of requests made to the server
        // (data should be the same)
        // get timer1
        // send number[index of i]
        // receive stuff from server
        // get timer2
        // timer2 - timer1 = time for server to respond
        // echo to console: request#, number, response, time to
           respond
    // end loop
    // close the connection
// echo to console: total requests
```

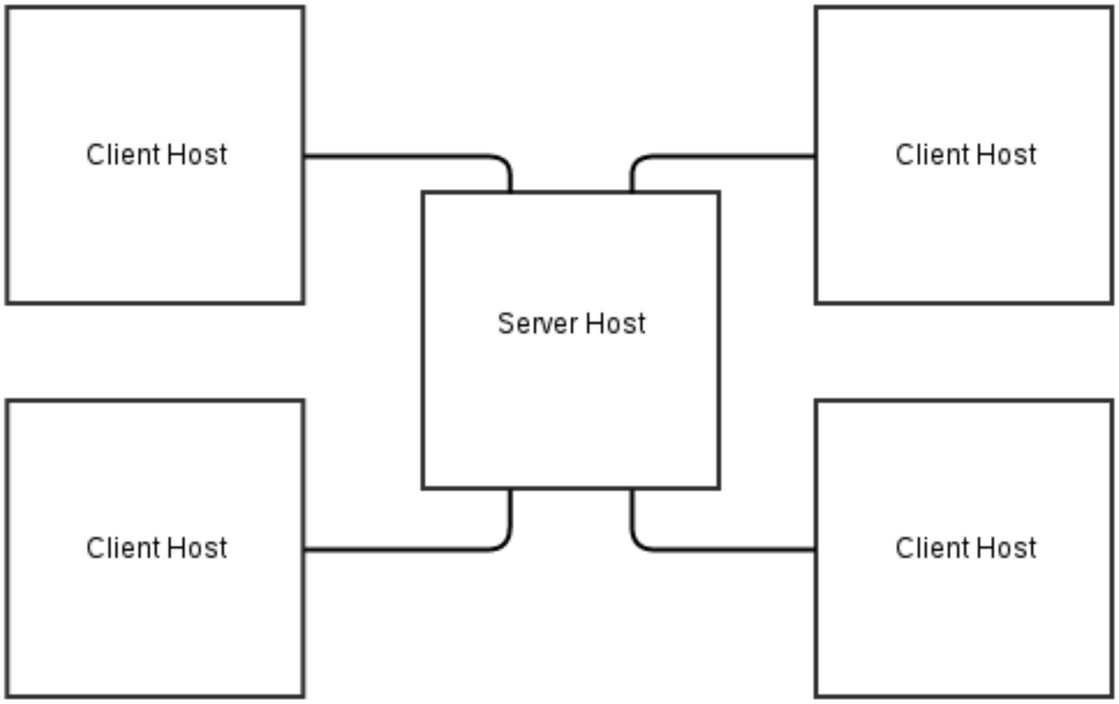
Diagrams

A one to one representation:



One Server host to service multiple clients from one Client Host machine.

A one to many representation:



One server is being accessed by 4 Client Hosts. With respect to the previous image, these Hosts will also spawn Clients within themselves, thus providing multiple client accesses.