Group members:
Harshit Garg (2018A7PS0218P)
Shreyans Jain (2018A7PS0253P)
Pranav Gupta (2018A7PS0190P)

DESIGN DOCUMENT

P2: Preforking Server System with fork()ing

Problem Statement - Create a basic preforking server system using fork(). It has the following features:

1. Number of idle children isn't more than a threshold
2. Number of idle children isn't less than a threshold
3. The production of new children has upper limit of 32 children/s
4. Child gets flushed once it reaches a threshold value of connections
5. Child responds to HTTP requests
6. Updates by parent on change in process pool
7. Updates on SIGINT signal
8. Zombie process prevented

Server-client-parent Communication Model

The parent initially creates a minimum number of spare servers expected, which is taken in as a command line argument. The production of new children is limited to a value of 32 children per second which is achieved by running the for loop a fixed number of times at a time, a value which gets big by a factor of 2 uptil 32. The socket initialized by the parent binds to the address INADDR_ANY and port specified in the PORT variable only once. And then the socet variable is shared by the children processes, who use it to accept new connections and work on the new socket. The parent maintains a linked list of the processes created by storing the pid of the child processes. This is done so that it knows which processes to kill when the maximum number of spare process limit is exceeded. Everytime the child opens a new connection with the client (to whom it appears as the server), it sends a message to the parent process through a UNIX socket (with socket file stored in /tmp as child_{pid}), to inform that a new active connection has been established. When the child is done with the communications, it closes the socket and sends a message back to the parent to confirm that the active connection has been closed. Also a check is present to see if the child has made enough connections for the process to be flushed. Signal handlers have been installed to ensure that updates are shown on the terminal screen when SIGINT (Ctrl C) signal is sent. Updates are also shown when the parent changes the child process pool by deleting a process or adding a new child process. The parent process prevents zombie processes from happening because when the child exists, it sends a SIGCHLD to the parent, which handles it separately by calling wait and reducing the count of running child processes.

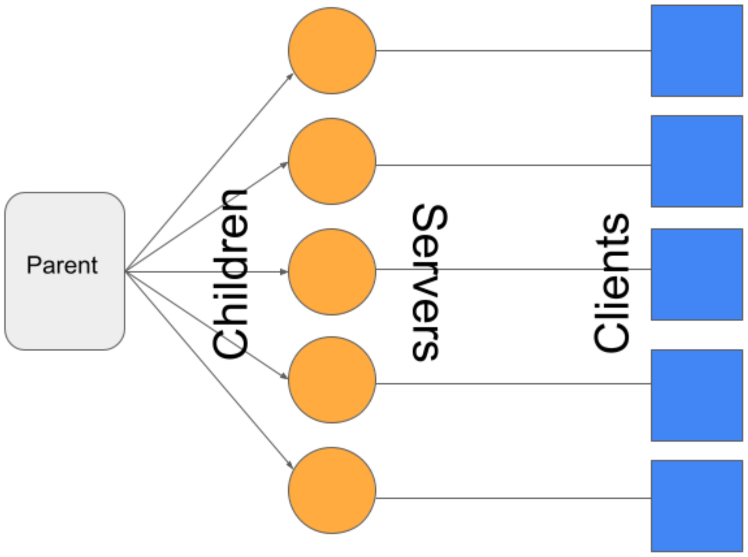

Fig 1. *Graphical representation of the communication model*

A sample output has been provided in `output.txt`.

Usage

The code can be compiled using the Makefile:

```
make rerun

# In another terminal window, run
```

```
ab -n 60 -c 3 http://127.0.0.1:12345/
```