CS 423 MP3 Report from Group 30

Abdullah Motan (amotan2)
Yayi Ning (yning4)
Zhiyu Zhu(zzhu24)
Minli Yang (myang46)

Basic Ideas:

1. We use local and virtual nodes to transfer and distribute the workload in order to achieve higher performance. With multiple different discipline of workload distribution, we try to find out the best way to balance the load of the local and virtual nodes.

2. The working process is that : We first save all the data we need to calculate on in the heap. Then we set up 512 jobs to finish all the calculations. With sockets, we send the second half of all the jobs ID and data to the virtual nodes. After both nodes finishes the jobs assigned to them, the virtual nodes pass back all the data that are done with the calculation.

Functions in Local:

1. int read_from_server(): After connect to the port of local machine, we then do three steps to create the TCP. Socket: create an endpoint for networking communication. Bind: associates an abstract socket with an actual network interface and port. Listen: specifies the queue size for the number of incoming, unhandled connections that have not yet been assigned a network descriptor. After the handshaking TCP is build, we read from the buffer to get all the data in the finished job from the virtual machine. This step is done with a loop of multiple reads. After all buffer are read, we close the descriptors.

2. int send_from_client: This function is to send all the jobs number and the status of last half of the job queue to assign workload to the virtual machine. In this function, after we connect to the port successfully, we do socket: this creates a socket object in the kernel with which one can communicate with the outside world/network. This returns a fd so you can use it like a normal file descriptor. Then connect: pass it the sockfd, then the address you want to go to and its length and you will be off connecting. After the handshaking TCP is build, we send all the job numbers of the unfinished jobs with their status to indicate the jobs that need to be done in the virtual machine with write function.

3. int Adaptor(): The adapter balance the workload distribution between the local machine and the virtual machine according to the size of jobs.

4. void job_initial(): We initialize all the jobs as a queue. Each of them are assigned with an unique job number and job status initialized as 0. In addition, all the data are initialized as 1.111111 as required in the document.

5. void* sending_function(void* job_id): When the left jobs are still more than the limitation of adaptor, we keep pull the job queue to get a new job unfinished. Then we send this job with all its information to the virtual machine.

6. void* receiving_function(void* ptr): In this function, we call read_from_server() to get all the information of data that already finished with calculation from the virtual machine.

7. void myWorkingFunction(int job_id): In this function, we assign each job with the data that it will perform the calculation on. Then this specific job will do the calculation that adding 1.111111 to the data 6000 time to finish the calculation.

8. int main (int argc, char* argv): The main function first initialize all the variables that we will be using in passing data and doing the calculation. The data array A is intialized with all 1.111111 in the heap. Then we initialize the queue of all the jobs. After all the initialization are done, we create another two thread: one for sending data to virtual machine and one for receiving the data finished with calculation. The main thread will be doing all the jobs assigned to the local machine assigned by the adaptor. After the thread is done, we call the pthread_join on it.

Functions in Virtual:

1. int read_from_server(): After connect to the port of local machine, we then do three steps to create the TCP. Socket: create an endpoint for networking communication. Bind: associates an abstract socket with an actual network interface and port. Listen: specifies the queue size for the number of incoming, unhandled connections that have not yet been assigned a network descriptor. After the handshaking TCP is build, we read from the buffer to get all the data in the finished job from the virtual machine. This step is done with a loop of multiple reads. After all buffer are read, we close the descriptors.

2. int send_from_remote(job* finished_job): This function is to send all the jobs number and the status of last half of the job queue to assign workload to the virtual machine. In this function, after we connect to the port successfully, we do socket: this creates a socket object in the kernel with which one can communicate with the outside world/network. This returns a fd so you can use it like a normal file descriptor. Then connect: pass it the sockfd, then the address you want to go to and its length and you will be off connecting. After the handshaking TCP is build, we send all the job numbers of the unfinished jobs with their status to indicate the jobs that need to be done in the virtual machine with write function.

3. void remoteWorkingFunction(int job_id): In this function, we assign each job with the data that it will perform the calculation on. Then this specific job will do the calculation that adding 1.111111 to the data 6000 time to finish the calculation.

9. int main (int argc, char* argv): The main function first initialize all the variables that we will be using in passing data and doing the calculation. The data array A is intialized with all 1.111111 in the heap. Then we initialize the queue of all the jobs. After all the initialization are done, we create another two thread: one for sending data to virtual machine and one for receiving the data finished with calculation. The main thread will be doing all the jobs assigned to the local machine assigned by the adaptor. After the thread is done, we call the pthread_join on it.