Optimized Job = 2 s

Non- optimized Job = 10s

int waitTimeSlaveA

int waitTimeSlaveB

Every single time we have a request for job a or b, we will compare the theoretical time with the wait time of both slaves:

Ex.

Int waitTimeSlaveA;

Int waitTimeSlaveB ;

JobType = A;

if (jobType = A)

if (waitTimeSlave A + 2 < waitTimeSlaveB + 10)

// put job on slave A

waitTimeSlaveA = waitTimeSlaveA + 2;

Else

// put job on slave B

waitTimeSlaveB = waitTimeSlaveB + 10;

Else if (jobType = B)

If (waitTimeSlaveB + 2 < waitTimeSlaveA + 10)

// put job on slave B

waitTimeSlaveB = waitTimeSlave B + 2;

Else

// put job on slave a

waitTimeSlaveA = waitTimeSlaveA + 10;

Meanwhile a different thread will actually be performing the jobs based on there job type

For example, Slave A: Job A – Takes 2 sec to do job – “sleep” 1 sec, do job 1 sec

Slave A: Job B – Takes 10 sec to do job – “sleep” 9 sec, do job 1 sec

Slave B: Job A – Takes 10 sec to do job – “sleep” 9 sec, do job 1 sec

Slave B: Job B – takes 2 sec to do job – “sleep” 1 sec, do job 1 sec

Two options for decrementing waitTime variables based on waiting jobs:

1. Decrement when finished sleeping and job either 10s or 2s
2. Decrement waittime variables by 1 for every sec of sleeping and actually doing job

One thing to note, is that these two threads must be mutually exclusive, since they are both involved in changing the same 2 variables, so we must implement peterson’s algorithm to makes sure they don’t run on the same time, and overwrite the variables.