Kennesaw State University

Department of Computer Science

CS3530 Operating Systems

Assignment 3 : Batch system with multiple computers

Michael Hug

hmichae4@students.kennesaw.edu

24 September 2013

Initial Problem Statement

A simulation of a set of computers was proposed. The set of computers would use different selection strategies to schedule jobs to be completed.

The archive batch-130909-early-terminating-submissions.zip contains the batch4A package which is the basis for this exercise.

Modify Computer to provide a method that returns the number of jobs waiting in its queue.

Modify SimulationMain to create and start an array of computers. The number of computers in the array should be a constant defined with the other constants. Modify SimulationMain to display the size of each queue in the computer once the simulation is done. Modify UserPool to accept the array of computers, rather than a single computer.

You will need to modify the way UserPool creates a job. When UserPool creates a Job, it will need to pick out a computer for that Job. You will compare different strategies fo choosing a computer for each job:

1. *Pick the first computer always.*
2. *Pick a computer with an empty batch queue. If no computer has an empty queue, use the first computer*
3. *Pick the computer with the shortest queue*

The first strategy is, of course, just like having one computer. The other computers are never used. The other two strategies do make use of the other computers, but the goal here is to determine how much of a difference they make. Start by implementing the first strategy. That requires almost no change in the code for UserPool.

For this assignment, once you have assured yourself that the number of computers can be changed and the program still works as it should, set the number of computers to 2. Set the service time mean to twice the inter-arrival time. Set the cutoff for job submission so that the batch queues are all empty when the simulation finishes. Don't set the cutoff too low or you won't be able to collect much data. Run the simulation fifteen times and collect the total wait time and the number of jobs for each run. Record the raw data and determine the overall average wait time per job. This will be the baseline for an overloaded system.

Now, implement each strategy in turn and run multiple simulations and determine the overall average wait time per job.

Create a document to summarize what you find. Report the average wait time for each strategy. Report the mean service time and the mean inter-arrival time you used for all these simulations. Summarize with a paragraph comparing the three strategies, reporting which one seems to work best (or which ones are tied for best).

Summary and Purpose of the Assignment Activity

This activity is meant to show the job wait time difference of 3 basic types of schedulers. First, a baseline is established by using only one computer. Next the number of computers is increased. Different types of selection strategies are used to determine which computer gets assigned the next job. The effectiveness of each strategy is timed and calculated by averaging the wait time for jobs in each selection strategy. Finally the different strategy average wait times are all considered.

Constants Used

Constants were used in the simulations to create an environment where a selection strategy for new jobs would become important. These constants could be further altered to create different simulation environments. The following constants were used in all the simulations including the baseline simulation to ensure constancy. The constants that were important to this assignment were as follows:

```
SIMULATION_LENGTH = 10000

SERVICE_TIME_MEAN = 10

INTER_ARRIVAL_TIME_MEAN = 5
```

The mean service time constant is set at ten and represents the average length of time that a job will take to execute. The mean inter arrival time is set at five and represents the average length of time between job creation. Setting the mean service time to greater the mean inter arrival time insures that jobs will arrive faster than they are processed. When jobs arrive faster than they can be processed, new jobs must be placed in a queue and wait to be executed. This project uses a service time mean double the mean inter arrival time, ensuring that queues are utilized. When the simulation length becomes too large, the PsimJ2 library can hang.

Data Collected

The data collected is the average wait time per job after fifteen simulations. Every job was completed in each simulation, a simulation that did not complete each job would skew the wait times because wait times for unprocessed jobs would not be quantified. Each of the simulations ran fifteen times. Each simulation's total wait was summed. Each simulation's total

jobs completed was summed. The total wait time and the total jobs completed were divided to give an average wait time per job per simulation strategy. The raw data is stored in accompanying files. The average wait times per simulation strategy were as follows :

Computer Selection Strategy	Mean Wait Time
Unmodified	978.88890034134986934832528 31
Strategy 1 : Pick the first computer always.	960.72298506592334919442357 46
Strategy 2 : Pick a computer with an empty batch queue. If no computer has an empty queue, use the first computer	334.42186941838807476153647 85
Strategy 3 : Pick the computer with the shortest queue	186.23252816817837634408602 15

Comments and Conclusion

Assignment 3 examined different computer selection strategies for job submission. The strategy that produced the least wait time per job was “Strategy 3 : Pick the computer with the shortest queue”. Strategy 3 gave the best results within the parameters of assignment 3 using the PsimJ2 library. The unmodified strategy and strategy 1 are within a standard error of 10 and can be interpreted as having similar results. Strategy 2 had a performance gain over strategy 1, but was still multiples less than strategy 3's performance.

Libraries and Script Files

The simulation was be carried out by a program using the PsimJ2 library. This is a process based simulation framework developed by Dr. Garrido at Kennesaw State University. Available at http://science.kennesaw.edu/~jgarrido/psimj.html .

A python script was also used to automate simulations and simulation calculations. The source code for the python script is as follows :

```
#!/usr/bin/python2
'''
@author: Michael Hug hmichae4@students.kennesaw.edu
Created for Dr Setzer's Fall 2013 3530 Operating Systems Class
Assignment_3
23 September 2013
'''
import sys
import os
import csv
import decimal

def RMshifteighttraces():
    os.system("rm ./traces/*")

def simulationLoop(name):
    str = "java -jar ./runables/"
    str += name
    str += ".jar"
    for _ in range(1,16):
        os.system(str)

def concatenateCSV(name):
    str = "cat ./traces/T* | sed 's/\"//g' > ./traces/"
    str +=name
    str +=".csv"
    os.system(str)
    os.system("rm ./traces/T*")

def CSVparse(name):
    str = "./traces/"
    str += name
    str += ".csv"
    filename = csv.reader(open(str,"rb"))
    numJobs = 0
    waitTime = 0
    for row in filename:
        numJobs += decimal.Decimal(row[1])
        waitTime += decimal.Decimal(row[3])
        if int(row[4])!=0:
            print "Error, re-run script. All jobs did not finish."
            exit(1)
    with open("./traces/"+name+".txt", "w") as text_file:
        text_file.write("{}".format(waitTime/numJobs))

RMshifteighttraces()
for name in
("unmodified","computerarrayStrategy1","computerarrayStrategy2","computerarra
yStrategy3"):
    simulationLoop(name)
    concatenateCSV(name)
    CSVparse(name)
```