641 Computer Performance Analysis

HW3 – Disk Queue Simulation

*Disk as a load-dependent server*

10/09/2015

Table of Content: Page

**I. About the project ……………………………………………………………………………… 2**

**II. Program description …………………………………………………………………………. 2**

**III. Error in calculation …………………………………………………………………………… 3**

**IV. Program output ……………………………………………………………………………….. 4**

**V. Data …………………………………………………………………………………………………. 5**

**VI. Conclusion ……………………………………………………………………………………….. 5**

**I. About:**

In this program we simulated disk operations as a load-dependent server. We took a particular disk, Quantum Atlas III and used its parameters in the simulation.

The “disk manager” we used in this simulation is a SSJF (Shortest Seek-time Job First). By utilizing the job priority based on shorted distance the head has to travel, we minimize the time the CPU waits for something to do.

Although, in order to get a benefit from this type of optimization, the server queue must be longer than one. If there a maximum of one item in the queue at any time, there is no optimization. With this simulation we prove that the performance of a disk - throughput – increases as the server queue length increases. This can be shown by a simple equation of Tseek:

Tseek = Tmax \* sqrt(X/Xmax).

After we simulate disk operations for a large N, we can average Tseek and find the average seek time.

*Problem with SSJF: Starvation*

Even though this simple way of optimizing disk operations is useful, it can lead to job starvation. A job that is too far away from the current location of the disk head, might never be performed or executed, if there will be a constant stream of jobs that are closer to the disk read head.

**II. Program description:**

**In order to keep all calculations from causing random error, I keep all values and parameters as doubles**

1. define the parameters of Quantum Atlas III as global variables, so all functions have access to them

2. Then I create a random generator function that returns a random cylinder 1 – 8057

- this is done with the win32 rand() function, I generate a URN (uniform random number) and multiply that real number by the max number of cylinders

3. **double sim(int Qsize){}**

- this the simulation function.

Create an array of size maxQueueSize and fill the array with random cylinders – up to a total amount Qsize speficied

Set the position of the head, also using the random cylinder generator

Find the cylinder with the shortest distance to the current location, x = distance

“process” that cylinder and replace it with another random cylinder.

Find Tseek with the current x value -> add up all Tseek

**Return** average Tseek

4. From main, run sim() up to a Qsize of 25

**III. Error in calculation.**

I had to make a correction for a very infrequent error that occurs when the cylinder picked out of the job queue was the same cylinder as the current location of the disk head.

When this happened, the distance disk head will travel is 0: x = 0

When x = 0

Tseek = t + c(x – 1)^r

Tseek = 1.5455 + .3197(-1)^0.3868

This means that a negative number (-1) is raised to a power

This generates a NAN (not a number) error.

To correct this, if( x == 0) x = 1;

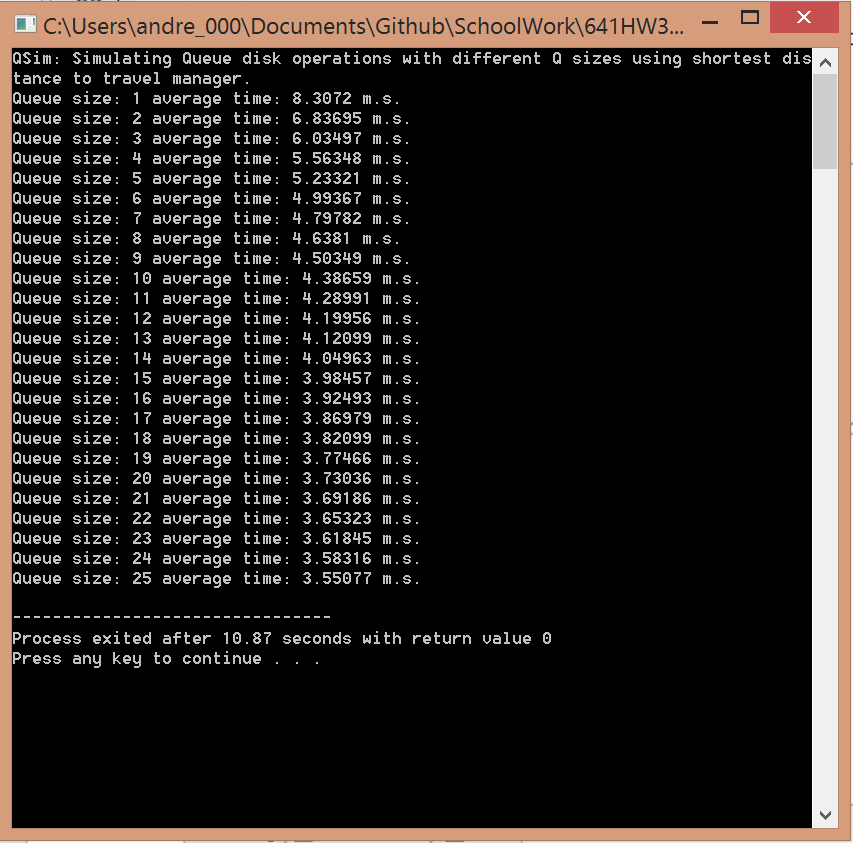
Because this happens only on 1/11000000 of simulations, the error is very small.

Tseek is offset by .3197/11kk ms, or **- 2.906e-8 milliseconds**

**IV. Program output**

This program was created on a Surface Pro 3 running Windows 8.1 with Dev C++

This is the output of the program.



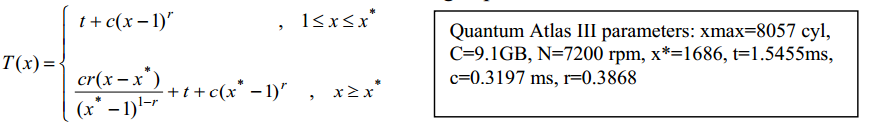
**V. Data**

This is the data from a disk load-dependent server simulation for a max Queue size of 25 and 1000000 simulations each Queue size.

Having 1000000 simulations allows me to report data up to 3 decimal places – by Monte Carlos principle

|  |  |
| --- | --- |
| Queue Size | Average  Tseek |
| 1 | 8.307 |
| 2 | 6.837 |
| 3 | 6.035 |
| 4 | 5.563 |
| 5 | 5.233 |
| 6 | 4.994 |
| 7 | 4.798 |
| 8 | 4.638 |
| 9 | 4.503 |
| 10 | 4.387 |
| 11 | 4.29 |
| 12 | 4.199 |
| 13 | 4.121 |
| 14 | 4.05 |
| 15 | 3.985 |
| 16 | 3.925 |
| 17 | 3.87 |
| 18 | 3.821 |
| 19 | 3.775 |
| 20 | 3.73 |
| 21 | 3.692 |
| 22 | 3.653 |
| 23 | 3.618 |
| 24 | 3.583 |
| 25 | 3.55 |

The formulas and parameters used to obtain this data:



**VI. Conclusion:**

Overall, this project was informative and a good introduction to simulations programming. With proper set and up formulas, simulations can show a very accurate representation of actual performance. Simulations can also help you create rough estimations of formulas. Although, in order for a simulation to show accurate data, you must be sure that the formulas it uses depict the behavior with a small margin of error.

The data obtained shows that using a SSJF disk scheduler does actually give a boost in performance.