Runtime Experiment
Project 3
COP 4634 Systems and Networks 1
Fall 2017

Jeffrey Murphy
Timothy L. J. Stewart
20 November 2017

Background:

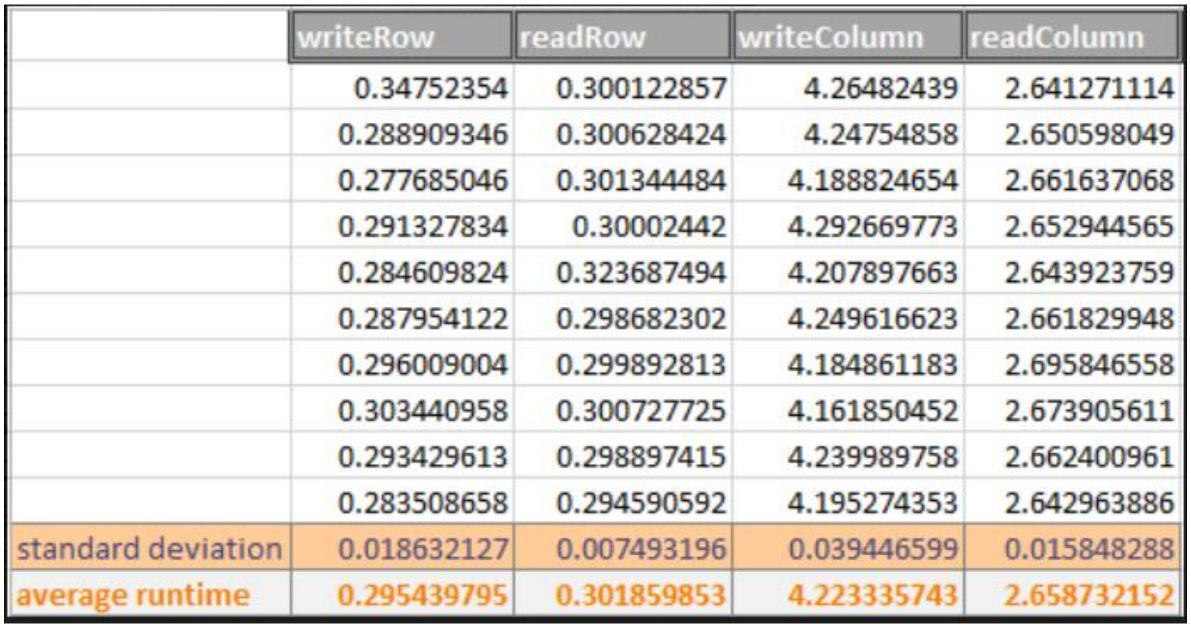
This assignment studied the use and misuse of large arrays and the effects on virtual memory from a practical and theoretical perspective. This report describes the program that was used to compare the times required to read and write elements of large arrays in both row-major and column-major order. A two-dimensional array was created using 20480 rows, each 4096 bytes long. The size of the rows were too large for the stack or heap thus were stored globally, and each row was exactly the size of a page on the system.

Discussion:

Four different measurements were taken for these operations:

1. Write Row
2. Write Column
3. Read Row
4. Read Column

Each of the operations were performed 10 times and the average values were computed for each operation. Figure 1 shows the results for the Write Row, Write Column, Read Row, and Read Column respectively. Optimization was turned off using the -O0 option when compiling the program.



	writeRow	readRow	writeColumn	readColumn
	0.34752354	0.300122857	4.26482439	2.641271114
	0.288909346	0.300628424	4.24754858	2.650598049
	0.277685046	0.301344484	4.188824654	2.661637068
	0.291327834	0.30002442	4.292669773	2.652944565
	0.284609824	0.323687494	4.207897663	2.643923759
	0.287954122	0.298682302	4.249616623	2.661829948
	0.296009004	0.299892813	4.184861183	2.695846558
	0.303440958	0.300727725	4.161850452	2.673905611
	0.293429613	0.298897415	4.239989758	2.662400961
	0.283508658	0.294590592	4.195274353	2.642963886
standard deviation	0.018632127	0.007493196	0.039446599	0.015848288
average runtime	0.295439795	0.301859853	4.223335743	2.658732152

Figure 1. Table of operation times when run on SSH server

This experiment was conducted in one program. The values were calculated with a start time just prior to the operation and stop time just after to conserve time. The specs for the system are shown in Figure 3. The expectations for the experiment include the following:

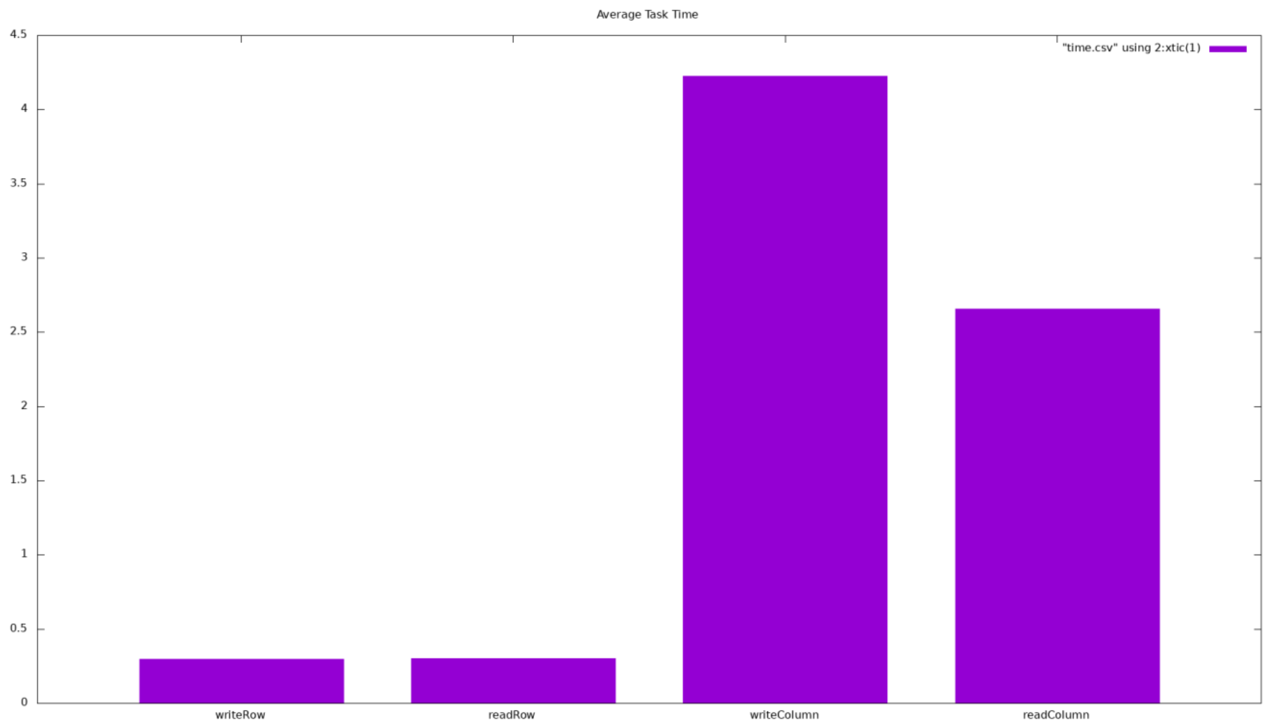
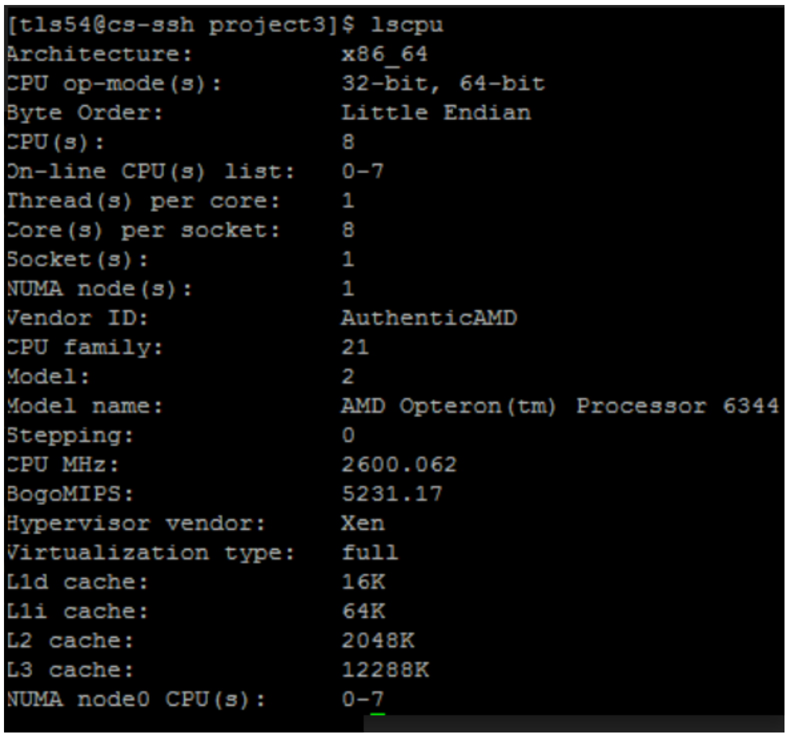


Figure 2. Bar Chart of Four different Average Runtime Results



```
[tls54@cs-ssh project3]$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                8
On-line CPU(s) list:   0-7
Thread(s) per core:    1
Core(s) per socket:    8
Socket(s):             1
NUMA node(s):          1
Vendor ID:             AuthenticAMD
CPU family:            21
Model:                 2
Model name:            AMD Opteron(tm) Processor 6344
Stepping:              0
CPU MHz:               2600.062
BogoMIPS:              5231.17
Hypervisor vendor:     Xen
Virtualization type:   full
L1d cache:             16K
L1i cache:             64K
L2 cache:              2048K
L3 cache:              12288K
NUMA node0 CPU(s):     0-7
```

Figure 3. Specifications of the System Used to Conduct Experiment

Conclusion:

The average runtime results were as expected, however, the actual number of page faults had no correlation with runtimes. The write by column was the slowest, then in increasing speed were read by column, write by row, and fastest was read by row. No additional compilation options were used to optimize the experiment. The table for the runtimes was shown in Figure 1. The bar charts for the experiment are shown in Figure 2.

The problems encountered during this experiment include:

1. Similar number of page faults for all the operations. Furthermore, the same number of page faults resulted in with different times. We believed the number of page faults would correlate with the average run times. There seemed to be no correlation with time and page faults.
2. The size of the man page on optimization seemed a bit daunting since there were so many ways to attempt optimization. –O0 was supposed to be the lowest level of optimization, which is also the default. This still lead to NOP codes.