COMP 8005 Assignment 01

Measuring the performance and efficiency of using multiple processes and threads.

Thilina Ratnayake – A00802338

1/26/2015

An experiment to observe the difference in performance between multi-threading versus multi-processing.

Table Of Contents:

Objective:

Use multiple processes and threads on the Linux operating system and measure the differences of performance (if any) between each mechanism.

Experiment Design

1. The experiment will measure the performance of each subject (processes or threads) based on time taken to finish a task.

2. The language that will be used is C, and will be written for and tested on the Linux OS.

3. The experiment involves both Mathematical Computations and file I/O.

    a. Mathematical computations: A string ("HASHSTRINGX" where X will be the iteration number) will be hashed using the SHA1 hashing algorithm a specified number of times.

    b. File I/O: The plaintext and hashed text will be printed onto text files.

4. Both subjects will be tested on hashing 1000,10000,100000,1000000, and 10,000,000 times.

Controls:

1. The experiment will be run 3 times for each subject, and completion times will be averaged.

2. The experiment will be run on the Fedora 21 distro.

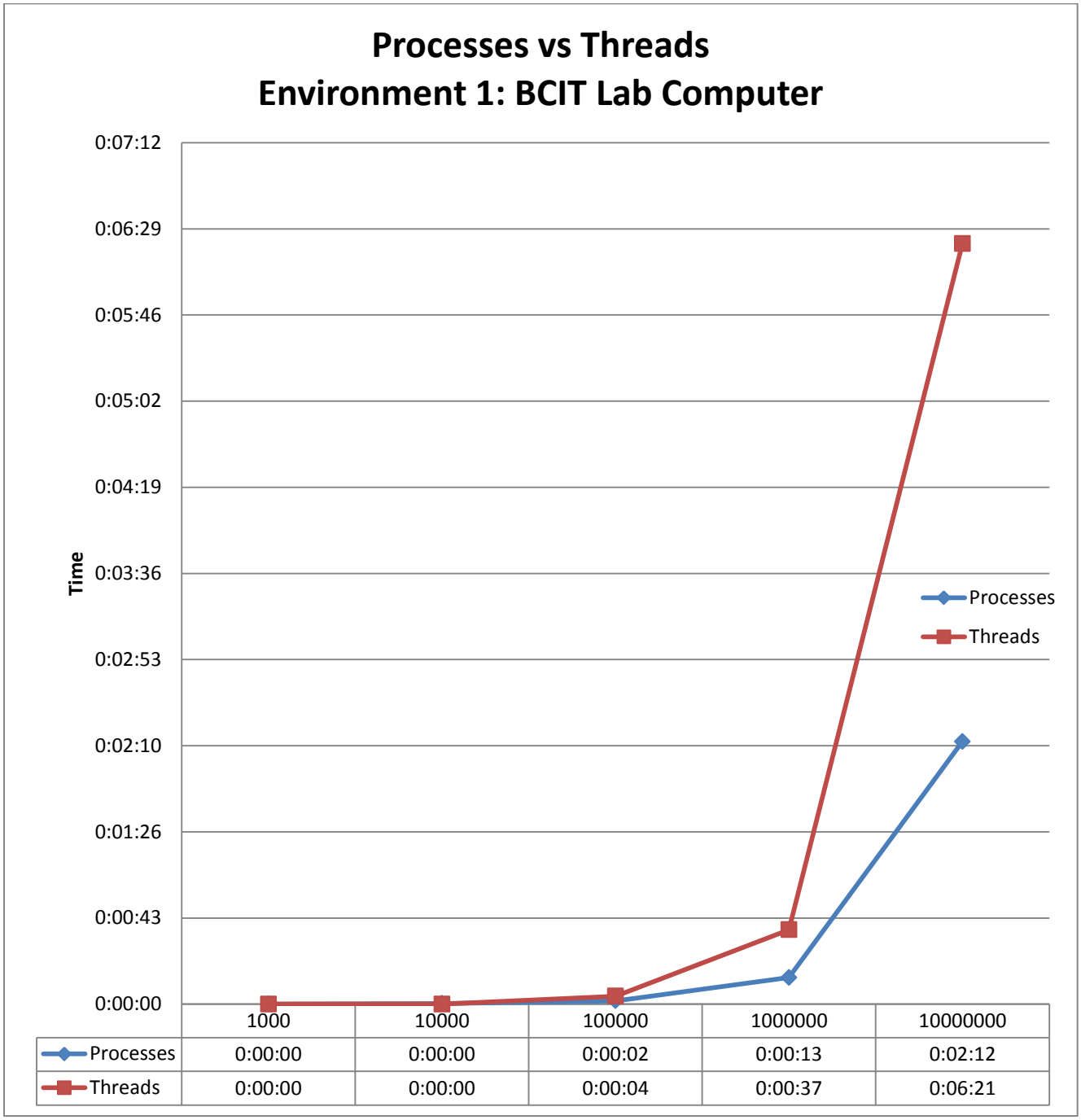
3. The experiment will be run on two sets of hardware/environments.

    Specifications:

Environment 1 - BCIT Lab	**Environment 2 -**Personal PC with Linux VM
• Processor Cores: 4 • Processor: Intel Core i5-2400k CPU @ 3.10GHz -64bit • RAM:8GB	• Processor Cores: 4 • Processor: Intel Core i5-4670K CPU @ 3.40 GHz -64bit • Processor Speed: • RAM:1GB

Results:

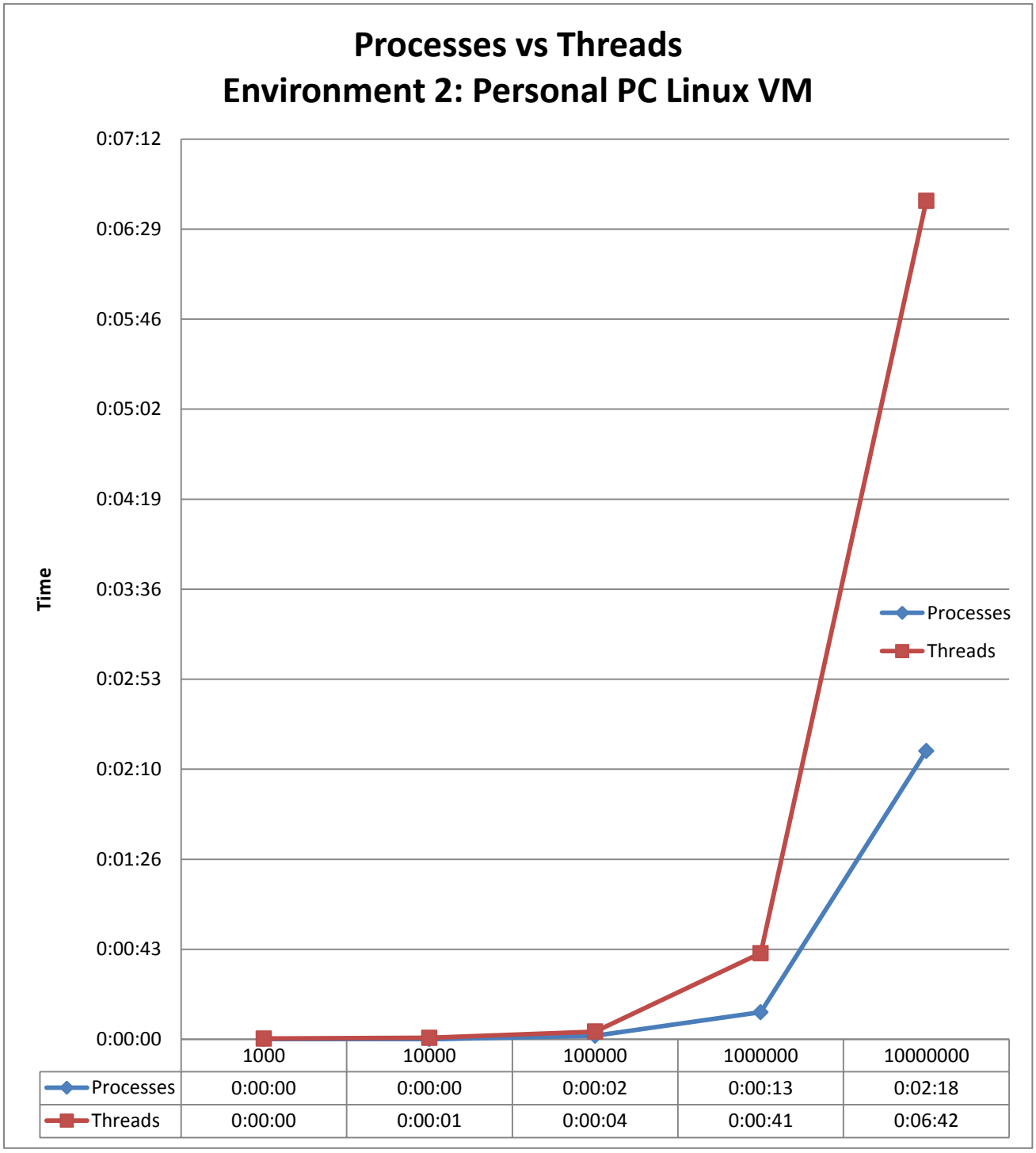
BCIT Lab:



	1000	10000	100000	1000000	10000000
Processes	0:00:00	0:00:00	0:00:02	0:00:13	0:02:12
Threads	0:00:00	0:00:00	0:00:04	0:00:37	0:06:21

Fig 1A: Chart of performance times over amount of iterations hashed in Environment 1(BCIT Lab)

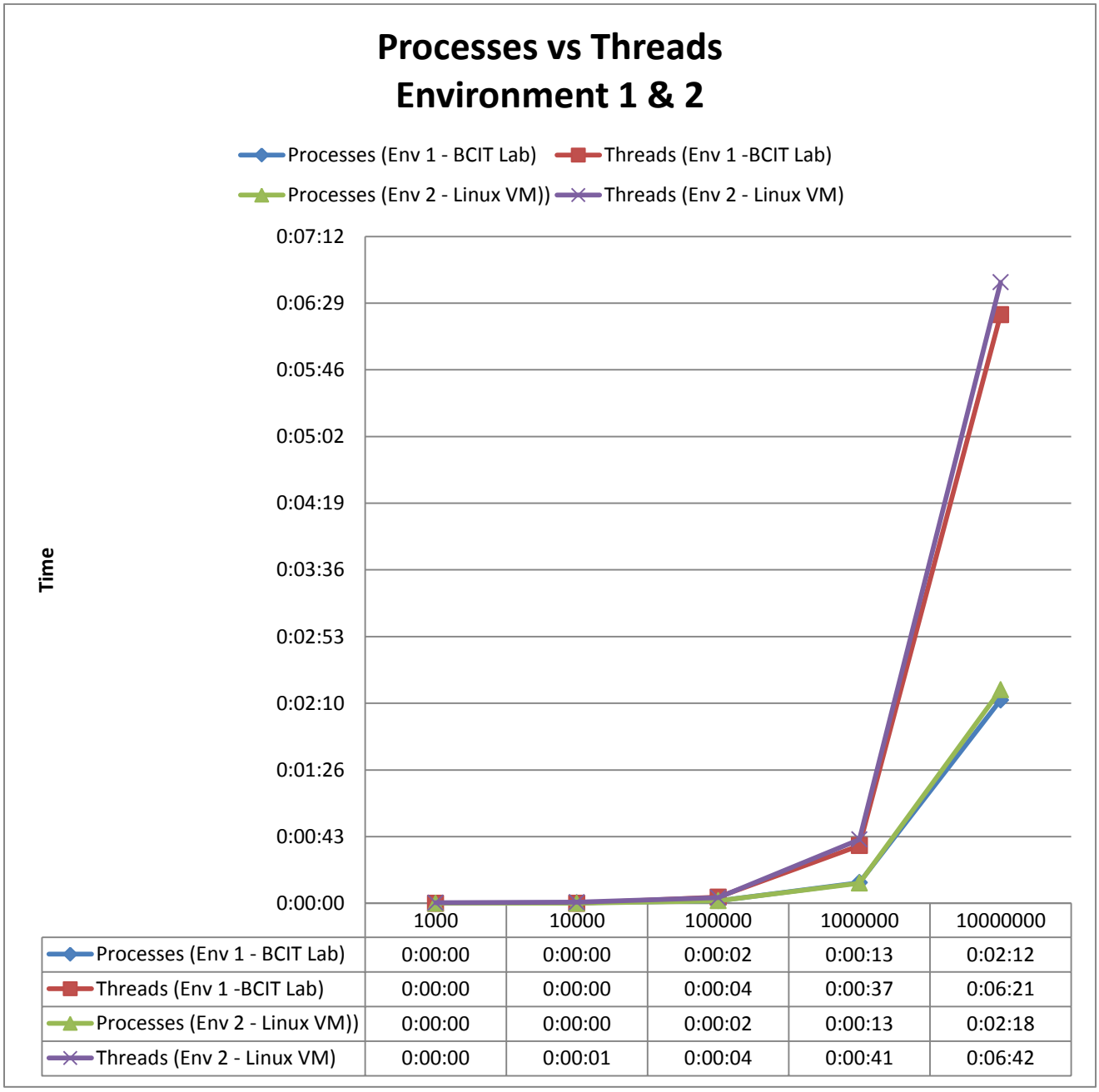
Linux VM:



	1000	10000	100000	1000000	10000000
Processes	0:00:00	0:00:00	0:00:02	0:00:13	0:02:18
Threads	0:00:00	0:00:01	0:00:04	0:00:41	0:06:42

Fig 1C: Chart of performance times over amount of iterations hashed in Environment 2 (Linux VM)

Final Comparison



	1000	10000	100000	1000000	10000000
Processes (Env 1 - BCIT Lab)	0:00:00	0:00:00	0:00:02	0:00:13	0:02:12
Threads (Env 1 -BCIT Lab)	0:00:00	0:00:00	0:00:04	0:00:37	0:06:21
Processes (Env 2 - Linux VM))	0:00:00	0:00:00	0:00:02	0:00:13	0:02:18
Threads (Env 2 - Linux VM)	0:00:00	0:00:01	0:00:04	0:00:41	0:06:42

Fig 1E: Chart comparing results from both environments.

Observations & Remarks:

1. The performance between the two mechanisms are the same when dealing with a task of requiring lower processing requirements, as demonstrated by their performance from iterations 1000 to 100,000.

2. As compared to my colleague's experiments which utilized prime number decomposition as it's task instead of hashing, their results indicated that the difference in performance between the two mechanisms was marginal through-out the experiment, whereas in this experiment there is a drastic change in performance.

3. In trying to understand why multi-threading took so much time, the sched_yield() call and mutex_locks were taken out as they do not utilize any of the same resources, thus avoiding any race conditions. This change in the program only reduced ~30 seconds off the total time.

Conclusion

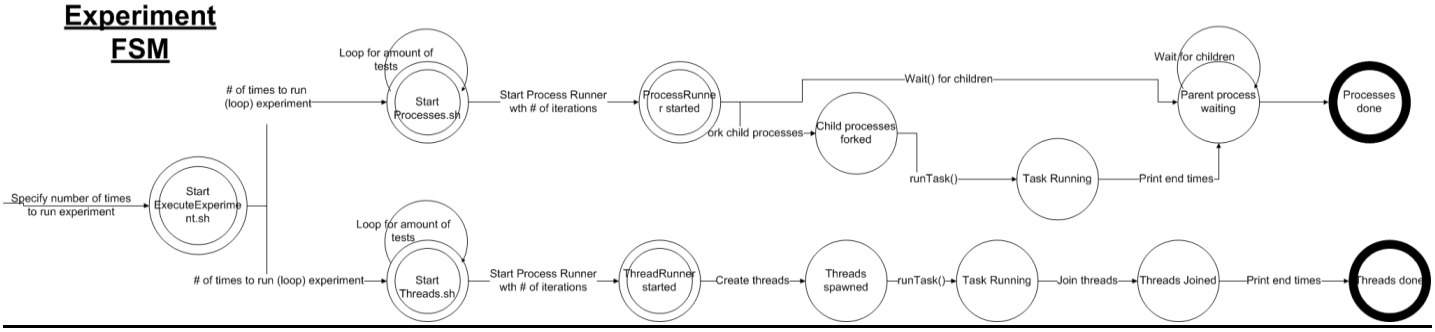
The conclusion that can be reached from this experiment is that the efficiency of utilizing a multi-threading mechanism decreases with the rise of processing requirements for a specific task. When compared to a multi-processing mechanism, which is approximately 291% more efficient than multi-threading at a task of 10,000,000 iterations.

DESIGN

Finite State Diagrams

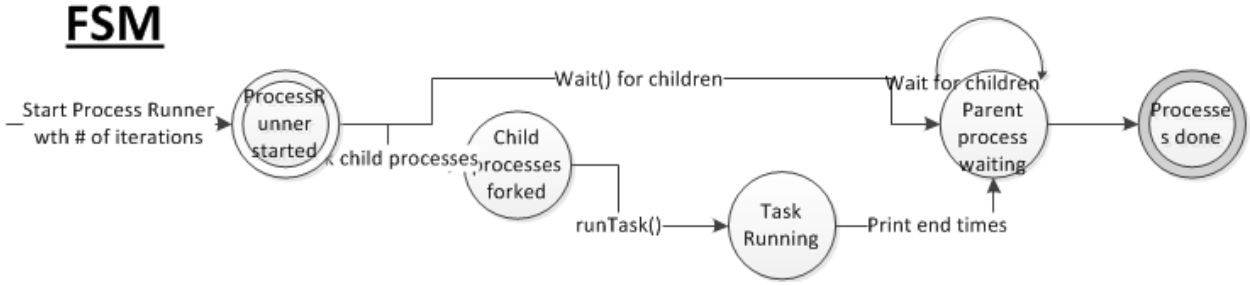
(These diagrams can be found in /FSMs for closer inspection)

Main Experiment FSM

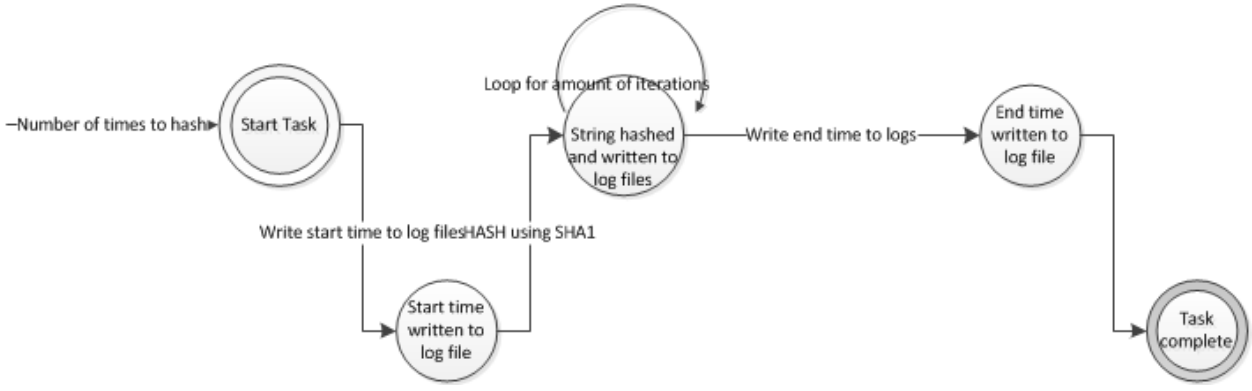


Threads FSM

Processes FSM



Task FSM



Program Components:

Component	Functions	Inputs	Outputs
ExecuteExperiment.sh	Script that runs Processes.sh and Threads.sh one after the other.	Number of times to run tests	--
Processes.sh	Script that runs ProcessRunner.c in a loop with desired amount of hash iterations.	Number of times to run tests (ProcessRunner)	./logfiles/DATETIME/DATETIME.csv
Threads.sh	Script that runs ThreadRunner.c in a loop with desired amount of hash iterations.	Number of times to run tests(ThreadRunner)	./logfiles/DATETIME/DATETIME.csv
ProcessRunner.c	Spawns 5 child processes and executes a task that hashes a string for the specified amount of iterations and outputs to logfiles.	Number of iterations.	./ProcessFiles/ProcessTaskOutputFileX.txt (where X = Process number) *Note:* When run with the script, all output files will be deleted as part of tear down procedure to save disk space.
ThreadRunner.c	Spawns 5 child threads and executes a task that hashes a string for the specified amount of iterations and	Number of iteartions.	./ThreadFiles/ThreadTaskOutputFileThreadX.txt (where X = Thread Number) *Note:* When run with the script, all output files will be deleted as part of tear down procedure to save disk space.

Program Structure:

```
bash: $: command not found...
[root@localhost 8005-Assignment-1]# tree
.
├── ExecuteExperiment.sh
├── Processes
│   ├── logfiles
│   │   ├── 2015_01_23__22_29_59
│   │   │   ├── Processes2015_01_23__22_29_59.csv
│   │   │   └── Processes2015_01_23__22_32_52.csv
│   │   └── 2015_01_24__10_05_38
│   │       ├── Processes2015_01_24__10_05_38.csv
│   │       └── Processes2015_01_24__10_08_32.csv
│   ├── Makefile
│   ├── Processes
│   ├── Processes.sh
│   ├── ProcessFiles
│   ├── ProcessLogFile.txt
│   ├── ProcessRunner
│   └── ProcessRunner.c
├── PsuedoCode:\ Processes.txt
├── PsuedoCode:\ Tasks.txt
└── Threads
    ├── logfiles
    │   ├── 2015_01_23__22_35_46
    │   │   ├── Threads2015_01_23__22_35_46.csv
    │   │   └── Threads2015_01_23__22_43_23.csv
    │   └── 2015_01_24__10_11_28
    │       ├── Threads2015_01_24__10_11_28.csv
    │       └── Threads2015_01_24__10_19_21.csv
    ├── Makefile
    ├── ThreadFiles
    ├── ThreadLogFile.txt
    ├── ThreadRunner
    ├── ThreadRunner.c
    └── Threads.sh
```

Psuedo-Code

Processes:

```
PsuedoCode: Processes

Main(number of processes, number of iterations){
    - Write to main log file the starting time.

    for loop (i to number of threads){
        //fork processes.

        pid = fork();
    }

    //handle the different type of processes

    switch(pid){
        if Unsuccessful forks:
        error

        if its a child process
        case 0:
        run the task(number of iterations);

        if its a parent
        case(anything else):

            loop through number of processes
            {
                wait for each process
                //this next statement only executes if the child is done.
                subtract -1 number of processes.
            }

        //All children finished.
        print ending time-stamp to log file.

        exit.
    }

}

run the task(number of iterations){

    print to main log file that task has started
    print to individual log file that task has started

    (loop from 0 - number of iterations){

        plaintext = Concatenate "HASHSTRING"+IterationNumber";
        Hashtext = Hash(plaintext);
        write Hashtext to individual logfile.

    }

    print to individual log file that task has finished.
    print to main log file that task has finished.
}
```

Threads:

```
PsuedoCode: Tasks

//threadObject structure
threadObject= {Name of the thread: "Thread1", Number of iterations: number of times to hash}

Main(number of processes, number of iterations){

    - Write to main log file the starting time.

    - Create the thread objects that contain name of thread, and number of iterations.

    - Create the threads, arguments to do runTask and supply the thread object.

    - join each thread to make sure that the program waits till they finish.

    - Write to main log file timestamp once all threads completed.

}

run the task(threadObject){

    - do a sched yield to make sure that we yield to other processes once this one finishes.

    - lock resources to execute thread.


    print to main log file that task has started
    print to individual log file that task has started

    (loop from 0 - number of iterations){

        plaintext = Concatenate "HASHSTRING"+IterationNumber";
        Hashtext = Hash(plaintext);
        write Hashtext to individual logfile.

    }

    print to individual log file that task has finished.
    print to main log file that task has finished.

    - unlock resources

}

    |
```

Program Requirements:

1. openssl-devel-1:1.0.1j-1.fc21.x86_64

Execution Instructions

To execute the experiment as it was done for gathering data.

1. Install required libraries
    a. *Yum install openssl-devel*
    b. *Yum install openssl-libs-1.0.1j-1.fc21.x86_64 zlib-1.2.8-7.fc21.x86_64*
2. Compile source code.
    a. *CD ./Processes*
    b. *Make ProcessRunner*
    c. *CD ../Threads*
    d. *Make ThreadRunner*
3. Within main directory, execute
    a. *SH ExecuteExperiment.sh 4*

To only run the Thread or ProcessRunner

1. Install required libraries
    a. *Yum install openssl-devel*
    b. *Yum install openssl-libs-1.0.1j-1.fc21.x86_64 zlib-1.2.8-7.fc21.x86_64*
2. Compile with following flags
    a. –lcrypto -pthread

Testing

#	Name	Resources	Expected	Actual	Fig
1	Hash specified amount of times	ProcessRunner.c, ThreadRunner.c	The end of the log file to contain the last hash number which should be the same as the iteration number	Same as expected	1.1
2	Generate unique hashes.	ThreadOutputFileX.txt, ProcessOutputFIleX.txt	The hashes should all be different from each other.	As expected	4.1
3	Spawn 5 Threads/Processes	ThreadRunner.c,ProcessRunner.c	1 parent thread and 5 child threads.	Same as expected.	2.1A,2.1B
4	Wait for child processes / other threads.	ProcessRunner.c, ThreadRunner.c	Program will not finish (print out end time) until tasks and processes have all finished writing to file.	Same as expected.	3.1A,3.1B

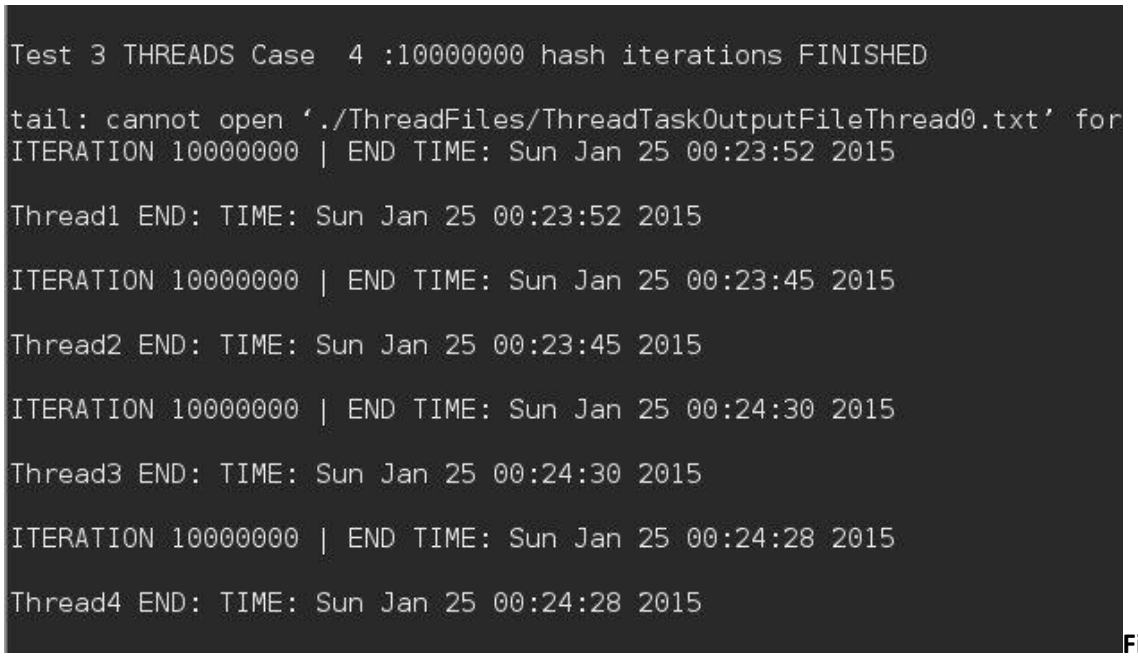


Fig 1.1

Hash specified amount of times. (“Iteration 10000000 | END TIME …)” is the output of a “tail –n4” command reading from a log file. This proves that 1000000 iterations were written.

```
KiB Swap:   839676 total,   387708 free,   451968 used.   245408 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
50536 root      20   0   14684    300      0 D  66.1  0.0   0:06.01 ProcessRunner
50532 root      20   0   14684    300      0 D  65.8  0.0   0:06.01 ProcessRunner
50535 root      20   0   14684    300      0 R  65.8  0.0   0:05.90 ProcessRunner
50533 root      20   0   14684    300      0 D  58.1  0.0   0:05.75 ProcessRunner
50534 root      20   0   14684    300      0 D  56.5  0.0   0:05.97 ProcessRunner
```

Fix 2.1A 5 child processes running

```
root      50290     2 50290  0    1 00:16 ?        00:00:00 [kworker/0:0]
root      50372 50064 50372  0    6 00:17 pts/3    00:00:00 ./ThreadRunner 10000000
root      50372 50064 50374 24    6 00:17 pts/3    00:00:49 ./ThreadRunner 10000000
root      50372 50064 50375 25    6 00:17 pts/3    00:00:50 ./ThreadRunner 10000000
root      50372 50064 50376 22    6 00:17 pts/3    00:00:46 ./ThreadRunner 10000000
root      50372 50064 50377 23    6 00:17 pts/3    00:00:46 ./ThreadRunner 10000000
root      50372 50064 50378 23    6 00:17 pts/3    00:00:47 ./ThreadRunner 10000000
root      50379 48644 50379  0    1 00:20 pts/2    00:00:00 ps -eLf
```

Fig 2.1B 5 child threads running.

```
Test 3 THREADS Case  4 :10000000 hash iterations
rm: cannot remove ‘./ThreadFiles/*.txt’: No such file or directory
** THREADS PROGRAM START** TIME: Sun Jan 25 00:17:26 2015


** Thread1 START** TIME Sun Jan 25 00:17:26 2015


** Thread5 START** TIME Sun Jan 25 00:17:26 2015


** Thread2 START** TIME Sun Jan 25 00:17:26 2015


** Thread3 START** TIME Sun Jan 25 00:17:26 2015


** Thread4 START** TIME Sun Jan 25 00:17:26 2015


** Thread2 END** TIME Sun Jan 25 00:23:45 2015


** Thread1 END** TIME Sun Jan 25 00:23:52 2015


** Thread5 END** TIME Sun Jan 25 00:24:27 2015


** Thread4 END** TIME Sun Jan 25 00:24:28 2015


** Thread3 END** TIME Sun Jan 25 00:24:30 2015


Threads Complete
** THREADS PROGRAM END** TIME: Sun Jan 25 00:24:30 2015
```

Fig 3.1A Waiting for child threads to finish before exiting.

```
Test 1 PROCESSES Case  3 :1000000 hash iterations
rm: cannot remove ‘./ProcessFiles/*.txt’: No such file or directory
** PROCESSES PROGRAM WITH NUMBER OF ITERATIONS: 1000000 START TIME: Sun Jan 25 00:35:31 2015

** PROCESS: 50516 START** TIME Sun Jan 25 00:35:31 2015
** PROCESS: 50517 START** TIME Sun Jan 25 00:35:31 2015


** PROCESS: 50518 START** TIME Sun Jan 25 00:35:31 2015


** PROCESS: 50519 START** TIME Sun Jan 25 00:35:31 2015




** PROCESS: 50520 START** TIME Sun Jan 25 00:35:31 2015


** PROCESS: 50517 END** TIME Sun Jan 25 00:35:43 2015


** PROCESS: 50520 END** TIME Sun Jan 25 00:35:44 2015


** PROCESS: 50519 END** TIME Sun Jan 25 00:35:44 2015


** PROCESS: 50518 END** TIME Sun Jan 25 00:35:44 2015


** PROCESS: 50516 END** TIME Sun Jan 25 00:35:44 2015


** PROGRAM END TIME, Sun Jan 25 00:35:44 2015
```

Fig 3.1B Waiting for child processes to finish before exiting.

```
PLAINTEXT: HASHSTRING5789245
HASH:
ffffffd2337810ffffff6ffffffdffffffa833ffffffa6ffffffe9ffffffc41d56ffffff8effffff9e31ffffff99ffffffcff
ffffa23a
ITERATION 5789245 | END TIME: Sun Jan 25 01:20:28 2015

ITERATION 5789246 | TIME: Sun Jan 25 01:20:28 2015

PLAINTEXT: HASHSTRING5789246
HASH:
37ffffffda68ffffffff4ffffffaafffffffb9684577ffffffc04dffffffeaffffffdfffffff3834cffffffaffffffb6
ITERATION 5789246 | END TIME: Sun Jan 25 01:20:28 2015

ITERATION 5789247 | TIME: Sun Jan 25 01:20:28 2015

PLAINTEXT: HASHSTRING5789247
HASH:
e4ffffff8ac766afffffff343ffffffb73246ffffffd4ffffffd0227effffffee76ffffffe3e5d
ITERATION 5789247 | END TIME: Sun Jan 25 01:20:28 2015

ITERATION 5789248 | TIME: Sun Jan 25 01:20:28 2015

PLAINTEXT: HASHSTRING5789248
HASH:
ffffffbf4352ffffffbb5473ffffffbd04bffffffe64a7bffffffa556468ffffffde45ffffffaeffffffe8
ITERATION 5789248 | END TIME: Sun Jan 25 01:20:28 2015

ITERATION 5789249 | TIME: Sun Jan 25 01:20:28 2015

PLAINTEXT: HASHSTRING5789249
HASH:
355bffffffe6ffffff8cffffffaf465c1ffffffff2ffffffe0ffffffa22e28ffffffc2fffffff040ffffffdcffffffc1ffffffb8
1c
ITERATION 5789249 | END TIME: Sun Jan 25 01:20:28 2015

ITERATION 5789250 | TIME: Sun Jan 25 01:20:28 2015

PLAINTEXT: HASHSTRING5789250
HASH:
ffffff8f38ffffff9effffffd8ffffffc2ffffffb7ffffffcdffffffb56f6effffffe9763959ffffffaf3dffffffc3161dffffff
90
ITERATION 5789250 | END TIME: Sun Jan 25 01:20:28 2015

ITERATION 5789251 | TIME: Sun Jan 25 01:20:28 2015

PLAINTEXT: HASHSTRING5789251
HASH:
fffffffee31ffffffff0ffffffa932ffffffd4ffffffd4ffffffd61f231bffffff9d35ffffffaaffffffba561ffffffff9effffffb6
ffffffd5
ITERATION 5789251 | END TIME: Sun Jan 25 01:20:28 2015
```

Fig 4.1 Generate unique hashes

Annex A: Experiment Data

Environment 1 (BCIT Lab) Data

	Iterations				
Processes	1000	10000	100000	1000000	10000000
Average Time	0:00:00	0:00:00	0:00:02	0:00:13	0:02:12
Test 1	0:00:00	0:00:01	0:00:01	0:00:14	0:02:11
Test 2	0:00:00	0:00:00	0:00:02	0:00:13	0:02:10
Test 3	0:00:00	0:00:00	0:00:02	0:00:13	0:02:14

	Iterations				
Threads	1000	10000	100000	1000000	10000000
Average Time	0:00:00	0:00:00	0:00:04	0:00:37	0:06:21
Test 1	0:00:00	0:00:00	0:00:04	0:00:38	0:06:26
Test 2	0:00:00	0:00:00	0:00:04	0:00:37	0:06:21
Test 3	0:00:00	0:00:00	0:00:04	0:00:37	0:06:17

Graph Data

Graph Data	Iterations				
Average time over 3 tests	1000	10000	100000	1000000	10000000
Processes	0:00:00	0:00:00	0:00:02	0:00:13	0:02:12
Threads	0:00:00	0:00:00	0:00:04	0:00:37	0:06:21

Fig 1B: Data of performance times over amount of iterations hashed in Environment 1 (BCIT Lab)

Environment 1 (Linux VM) Data

	Iterations				
Processes	1000	10000	100000	1000000	10000000
Average Time	0:00:00	0:00:00	0:00:02	0:00:13	0:02:18
Test 1	0:00:00	0:00:00	0:00:02	0:00:13	0:02:20
Test 2	0:00:00	0:00:00	0:00:01	0:00:13	0:02:17
Test 3	0:00:00	0:00:00	0:00:02	0:00:13	0:02:18

	Iterations				
Threads	1000	10000	100000	1000000	10000000
Average Time	0:00:00	0:00:01	0:00:04	0:00:41	0:06:42
Test 1	0:00:01	0:00:00	0:00:04	0:00:41	0:06:40
Test 2	0:00:00	0:00:01	0:00:04	0:00:44	0:06:57
Test 3	0:00:00	0:00:01	0:00:03	0:00:39	0:06:30

Graph Data

Graph Data	Iterations				
Average time over 3 tests	1000	10000	100000	1000000	10000000
Processes	0:00:00	0:00:00	0:00:02	0:00:13	0:02:18
Threads	0:00:00	0:00:01	0:00:04	0:00:41	0:06:42

Fig 1D: Data of performance times over amount of iterations hashed in Environment 2 (Linux VM)