

Nikhila Byreddy
SMU ID - 47293718

1) Assignment Requirements :

Install VMWare Fusion.

Install 32 bit Ubuntu as a guest Operating System.

Later install another virtual machine —> 64 bit Ubuntu as a guest Operating System.

Consider the code smuprogram1a which is used in Lab2.

2) Experiment Design :

(a) Machine Configuration :



MacBook Pro

▼ Hardware

ATA

Audio

Bluetooth

Camera

Card Reader

Diagnostics

Disc Burning

Ethernet Cards

Fibre Channel

FireWire

Graphics/Displays

Hardware RAID

Memory

NVMeExpress

PCI

Parallel SCSI

Power

Printers

SAS

SATA/SATA Express

SPI

Storage

Thunderbolt

USB

iBridge

▼ Network

Firewall

Locations

Volumes

Hardware Overview:

Model Name:

MacBook Pro

Model Identifier:

MacBookPro14,1

Processor Name:

Intel Core i5

Processor Speed:

2.3 GHz

Number of Processors:

1

Total Number of Cores:

2

L2 Cache (per Core):

256 KB

L3 Cache:

4 MB

Memory:

16 GB

Boot ROM Version:

MBP141.0167.B00

SMC Version (system):

2.43f6

Serial Number (system):

C02VC0HHHV2J

Hardware UUID:

C79F7E60-152D-5D36-91D4-FA0675AED171

Nikhila's MacBook Pro > Hardware



ubuntu 16.04 LTS

Device name

ubuntu

Memory 997.9 MiB

Processor Intel® Core™ i5-7360U CPU @ 2.30GHz

Graphics Gallium 0.4 on SVGA3D; build: RELEASE; LLVM;

OS type 32-bit

Disk 19.9 GB

2B) I have used smuprogram1a

It is shown below :

```

/* John M. Medellin SMU Lyle CSE 7346 Fall 2017, Program 1 a computational exhaustion */

/* Program name is program1a.c compiled using gcc smuprogram1a.c -o smuprogram1a.out */
/* Define Includes */
/* -----*/

#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <stdarg.h>
#include <time.h>
#include <signal.h>
#include <limits.h>
#include <pthread.h>
#include <sys/types.h>
#include <stdbool.h>
#include <unistd.h>

/* Define counter...number of modulo multiples to find before the program finishes
random number, modulus operator to determine if it's a valid number
*/

    long long int count = 0;
    int maxcount = 3;
    long long int randnum = 1;
    int powerof = 1;
    long long int modoper = 1;
    long long int numtries = 0;
    long long int totnumtries = 0;

int main (void)
{
    clock_t begin = clock();
    /* Accept input from the key board to drive the number of modular compliant numbers
    the modulo operator to be used and the elevation of rand to a power of 10

    THE CODE DEFAULTS TO THE FOLLOWING:

    100 MULTIPLES OF THE ROUND NUMBER TO FIND
    2^34TH MODULO OPERATOR
    2^10TH IS USED TO MULTIPLY TIMES ROUND NUMBER TO COMPARE TOT THE 2^34TH MODULO

    THE CODE CAN ALSO BE MADE AS INPUT FROM THE KEYBOARD BY MAKING THE PRINTF/SCAN LINES ACTIVE

```

```

//      printf("Enter the number of modulars to find (default is 3) \n");
//      scanf ("%i",&maxcount);
//      maxcount = 100;

//      printf("Enter the modulo operator in power of 2 (10 = 1024) \n");
//      scanf ("%lli",&modoper);
//      modoper = 34;

//      modoper = ldexp(1,modoper);
//      printf("%lli \n",modoper);
//      printf("Enter the power of 2 that we are to elevate the random number \n");
//      scanf ("%i",&powerof);
//      powerof = 10;

//      printf("Rand^pwr \t Modulo \t [rand^pwr]/Modulo \t #Iter \n");
//      printf("----- \n");

// BEGIN LOOP TO FIND THE NUMBER OF OCCURRENCES WHERE THE RANDOM IS A FUNCTION OF THE MODULO
{
    while (count < maxcount)
    {
        /* Fetch a random number, elevate it to the power, increase the number of tries counter
        ----- */

        randnum = rand() * ldexp(1,powerof);
        numtries ++;

        /* If it is modulo that we want, report it with the number of tries, increase counter
        and zero out the number of tries so we can report the variable again
        ----- */

        if (randnum % modoper == 0){

            printf("%lli \t %lli \t %lli \t %lli \n",randnum,modoper,(randnum/modoper),numtries);
            count ++;
            totnumtries = totnumtries + numtries;
            numtries = 0;}

    }

// ADD UP THE NUMBER OF TRIES IT TOOK TO GET TO THE NUMBER OF OCCURRENCES AND PRINT IT
printf("Total number of tries %lli \n",totnumtries);

// SYSTEM COMMAND TO ISSUE THE STATISTICS OF THE RUN
system("ps aux");

```

```

system("ps aux");|

// PRINT THE PROCESS ID SO IT CAN BE USED TO DETERMINE THE USAGE STATISTICS

printf("process id %d \n",getpid());
clock_t end = clock();
double time_spent = (double)(end - begin) / CLOCKS_PER_SEC;
printf("Time taken:%lf", (double)(end-begin)/CLOCKS_PER_SEC);
}

```

2C) I have performed the Optimizations HV-01, HV-02, HV-05

3

(a) Baseline results :

For 32 bit Ubuntu

To find the time taken to execute the smuprogram1a you may need to include begin and end statements as shown below

```

int main (void)
{
clock_t begin = clock();

```

```

printf("process id %d \n",getpid());
clock_t end = clock();
double time_spent = (double)(end - begin) / CLOCKS_PER_SEC;
printf("Time taken:%lf", (double)(end-begin)/CLOCKS_PER_SEC);

```

```

process id 5595
Time taken:76.505915

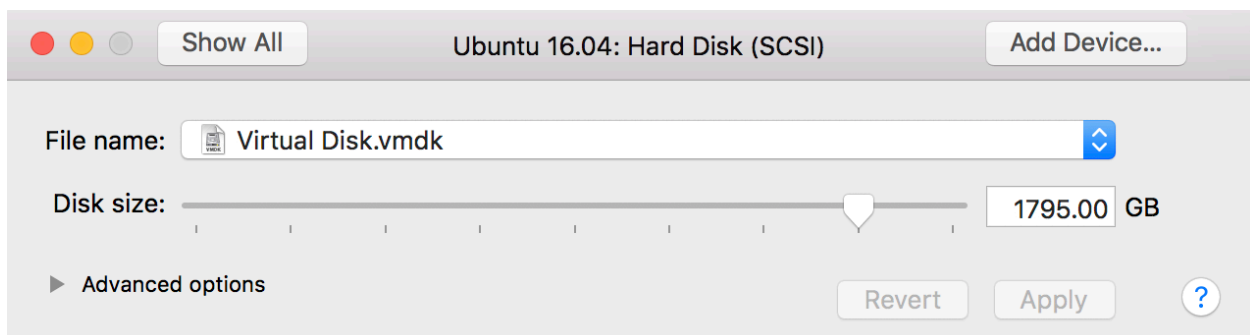
```

3b) HV-01

Firstly increase the hard disk memory

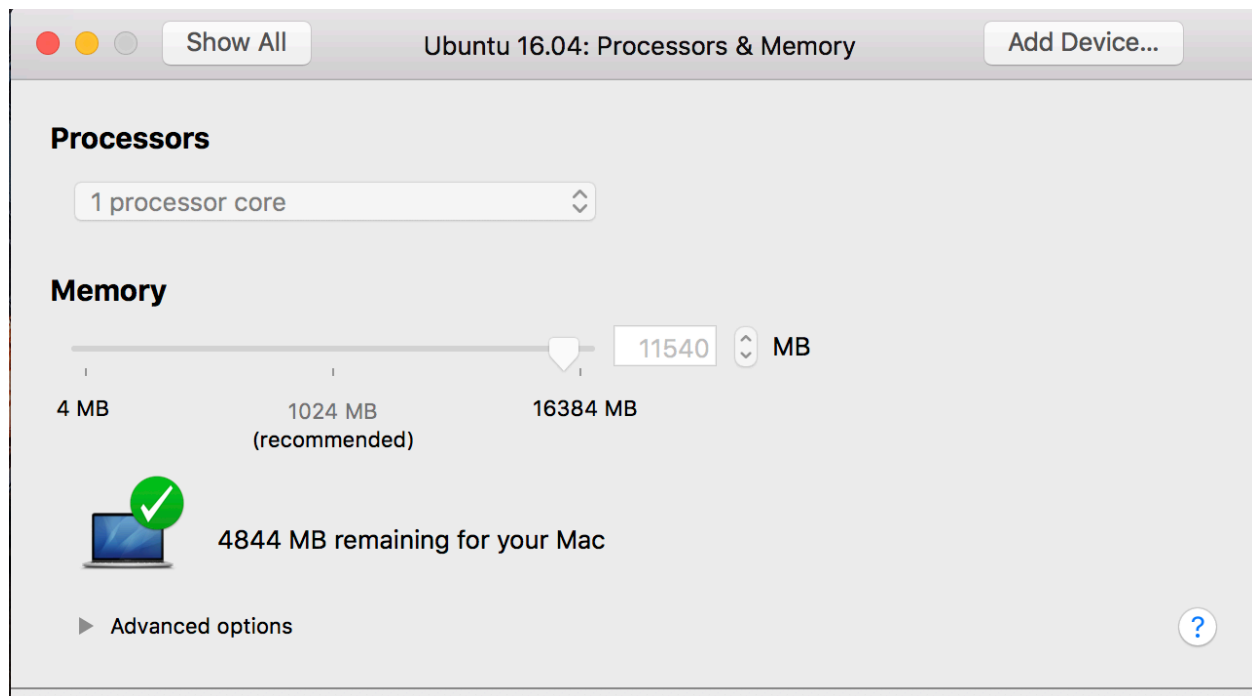
This can be done by

Go to Virtual Machine —> Settings —> Hard Disk



Also increase the memory. This can be done by

Go to Settings —> Processors & Memory and then increase the Memory



Now when you execute the program you get

```
nikhila+ 2174 0.0 0.0 7756 3168 pts/
process id 2170
Time taken:69.876693nikhilabyreddy@ubuntu:
```

HV-02

For 32 bit

Time taken for Smuprogram1a is

```
nikhila+ 2174 0.0 0.0 7756 3168 pts/
process id 2170
Time taken:69.876693nikhilabyreddy@ubuntu:
```

For 64 bit time taken is


```

nikhila+ 2408 0.0 0.3 44432 3272 pts/
process id 2384
Time taken:43.843480nikhilabyreddy@ubuntu

```

HV-05

When I tried running two concurrent 64 bit virtual machines simultaneously time taken is

Time taken is

Terminal		Terminal	
nikhilabyreddy@ubuntu: ~/Downloads		nikhilabyreddy@ubuntu: ~/Desktop/sf	
nikhila+ 1816 0.0 0.8 410676 8216 ?	SL 13:	nikhila+ 2113 0.0 0.6 266592 6532 ?	
nikhila+ 1822 0.0 0.4 264600 4520 ?	SL 13:	nikhila+ 2123 0.0 0.4 264600 4780 ?	
nikhila+ 1833 0.0 0.7 278788 7532 ?	SL 13:	root 2134 0.1 3.2 635332 32716 ?	
nikhila+ 1853 0.2 4.5 814132 45168 ?	SL 13:	nikhila+ 2148 0.0 4.8 814132 48596 ?	
nikhila+ 1855 0.0 1.3 704360 13024 ?	SL 13:	nikhila+ 2151 0.0 1.6 704360 16560 ?	
root 1861 0.7 3.3 635212 33644 ?	SL 13:	nikhila+ 2164 0.0 1.6 788040 16436 ?	
nikhila+ 1869 0.0 1.4 788040 14456 ?	SL 13:	nikhila+ 2177 0.0 0.8 370716 8212 ?	
nikhila+ 1891 0.0 0.7 370716 7684 ?	SL 13:	nikhila+ 2208 0.0 0.5 193048 5364 ?	
nikhila+ 1934 0.0 1.8 498072 18232 ?	SL 13:	nikhila+ 2249 0.1 3.7 669168 37172 ?	
nikhila+ 1941 0.0 0.0 4508 708 ?	S 13:	nikhila+ 2255 0.0 0.5 29492 4984 pts/6	
nikhila+ 1945 0.0 0.9 358096 8996 ?	SL 13:	nikhila+ 2266 0.0 2.0 594856 20464 ?	
nikhila+ 1953 0.0 1.5 321468 15220 ?	SL 13:	nikhila+ 2274 0.0 1.0 150304 10196 ?	
nikhila+ 1965 0.0 1.7 654176 16944 ?	SL 13:	nikhila+ 2278 0.0 1.6 497952 16408 ?	
nikhila+ 1977 0.2 2.1 656816 21660 ?	SL 13:	nikhila+ 2284 0.0 0.0 4508 852 ?	
nikhila+ 1978 0.0 1.3 580352 13316 ?	SL 13:	nikhila+ 2288 0.0 0.8 357764 8484 ?	
nikhila+ 2023 0.4 3.2 669008 32516 ?	SL 13:	nikhila+ 2296 0.0 1.5 321464 15300 ?	
nikhila+ 2030 0.0 0.4 29448 4884 pts/17	Ss 13:	nikhila+ 2346 0.0 3.1 576228 30884 ?	
nikhila+ 2054 4.9 0.0 4356 784 pts/17	T 13:	root 2393 0.0 0.0 0 0 ?	
nikhila+ 2076 0.1 2.5 607108 25552 ?	SL 13:	nikhila+ 2397 0.0 0.8 530296 8672 ?	
nikhila+ 2082 94.0 0.0 4356 656 pts/17	S+ 13:	nikhila+ 2430 98.1 0.0 4356 632 pts/6	
nikhila+ 2108 0.0 0.0 4508 708 pts/17	S+ 13:	nikhila+ 2439 0.0 0.0 4508 708 pts/6	
nikhila+ 2109 0.0 0.3 44432 3220 pts/17	R+ 13:	nikhila+ 2440 0.0 0.3 44432 3344 pts/6	
process id 2082		process id 2430	
Time taken:49.834448nikhilabyreddy@ubuntu:~/Downloads\$		Time taken:50.062183nikhilabyreddy@ubuntu:~/	

Time taken when I run
Virtual machine 1 alone is

```

nikhila+ 2418 0.0 0.3 44432 3272 pts/
process id 2423
Time taken:43.890791nikhilabyreddy@ubuntu

```

And the time taken when you run Virtual machine 2 alone is

```

process id 2122
Time taken:46.756996r

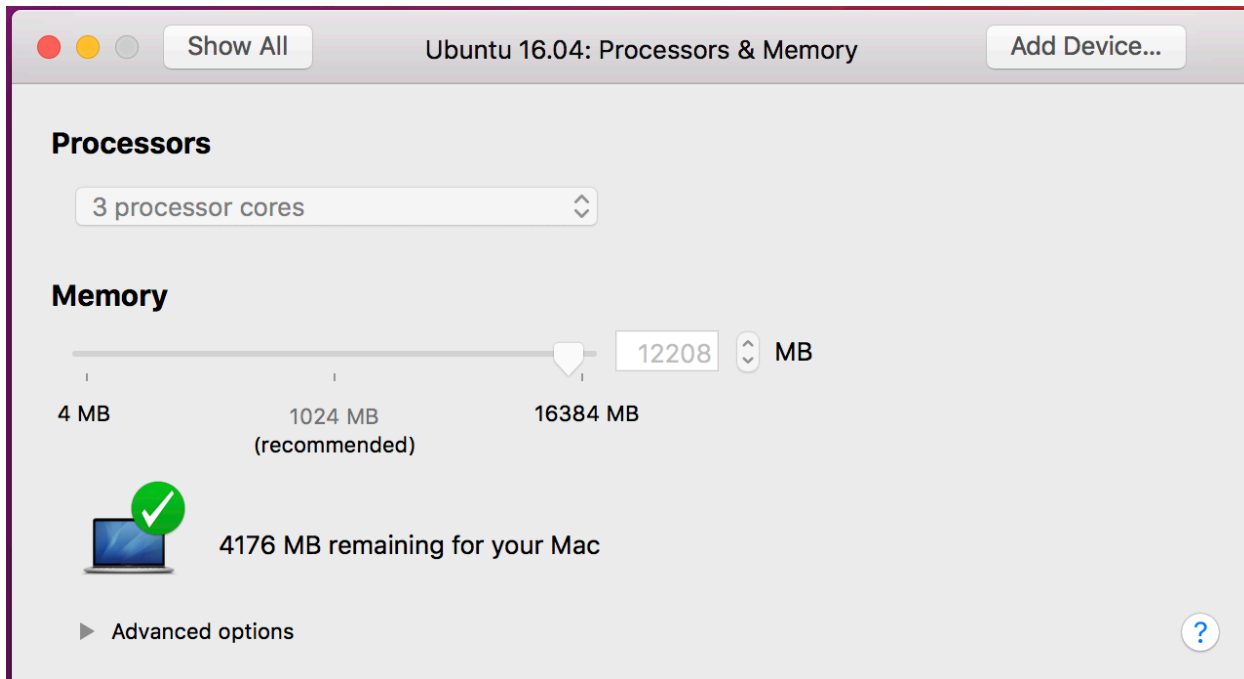
```

My Own :

Increase the number of core processors.

This can be done by:

Go to settings —> processors & memory



```
process id 2082
Time taken:79.620822nikhilabyreddy@ubuntu:~/
```

(4A) For Optimization 1

When I ran the code without increasing the memory the time taken was 76.50ms.

Later when I increased the memory space and hard disk space time taken is 69.87ms.

For Optimization 2

when I ran the code for 32 bit Ubuntu Virtual machine the time taken is 76.50ms before increasing the memory and harddisk space and 69.87ms after increasing the space.

when I ran the code for 64 bit Ubuntu Virtual machine the time taken is 43.84ms.

For Optimization 3

When I tried running two 64 bit Ubuntu virtual machines simultaneously time taken for virtual machine 1 is 49.83ms and for virtual machine 2 is 50.06ms.

When I tried running virtual machine 1 alone time taken is 43.89ms and virtual machine2 alone is 50.06ms.

For Optimization 4(MY OWN)

when I tried increasing the number of processor cores from 1 to 3 time taken for the execution of program increased thereby reducing the performance.

(4b) When you increase the memory size as in optimization1, the time taken is reduced. This is because as the memory size increases speed of execution of processes increases thereby decreasing the time of execution.

Similarly when you run 64 bit machine, it has more registers compared to 32 bit registers thereby reducing the time for execution in 64 bit machine.

Similarly when you run two 64 bit machines simultaneously performance degrades much worse than linearly with increased concurrency.