Code and Screenshots

PA1\_CPU\_First

#include <stdio.h>

#include <pthread.h>

#include <time.h>

#include <sys/time.h>

#include <math.h>

float threadFunc\_int(long int noItr)

{

int i,val1=1,val2=2,val3=4;

struct timeval start\_time;

struct timeval end\_time;

gettimeofday(&start\_time,NULL);

for(i=0;i<noItr;i++)

{

val1=val2\*val3;

val3=val1+val2;

val2=val3-val1;

val1=val2/val3;

}

gettimeofday(&end\_time,NULL);

printf("%d %d",end\_time.tv\_sec,start\_time.tv\_sec);

float timeTaken=(float)(end\_time.tv\_sec-start\_time.tv\_sec);

return timeTaken;

}

float threadFunc\_float(long int noItr)

{

int i;

float val1=1.2,val2=2.4,val3=4.6;

struct timeval start\_time;

struct timeval end\_time;

gettimeofday(&start\_time,NULL);

for(i=0;i<noItr;i++)

{

val1=val2\*val3;

val3=val1+val2;

val2=val3-val1;

val1=val2/val3;

}

gettimeofday(&end\_time,NULL);

float timeTaken=(float)(end\_time.tv\_sec-start\_time.tv\_sec);

return timeTaken;

}

int main(void)

{

int i=0;

int noThreads;

struct timeval begin\_time;

struct timeval last\_time;

int op;

float timeTaken=0;

float avgTime=0.0;

printf("Enter Operation to Perform 1 int 2 float");

fflush(stdout);

scanf("%d",&op);

printf("Enter threads");

fflush(stdout);

scanf("%d",&noThreads);

pthread\_t pth[noThreads];

for(i=0;i<noThreads;i++)

{

if(op==1)

pthread\_create(&pth[i],NULL,threadFunc\_int,100000000);

else

pthread\_create(&pth[i],NULL,threadFunc\_float,100000000);

}

for(i=0;i<noThreads;i++)

{

pthread\_join(pth[i],&timeTaken);

avgTime=avgTime+timeTaken;

}

float ops=(float)(100000000\*4\*noThreads)/avgTime;

if(op==1)

{

printf(" Iops in Hertz is %f",(double)ops);

printf(" Iops in Giga Hertz is %f",ops/1000000000);

}

else

{

printf(" Flops in Hertz is %lf",ops);

printf(" Flops in Giga Hertz is %f",ops/1000000000);

}

return 0;

}

PA1\_CPU\_Second.c

#include <stdio.h>

#include <pthread.h>

#include <time.h>

#include <sys/time.h>

#include <fcntl.h>

#include <sys/stat.h>

pthread\_mutex\_t lock;

int threadFunc\_int(int s)

{

int i=0,a=1,b=2,c=4;

struct timeval t1;

struct timeval t2;

gettimeofday(&t1,NULL);

for(i=0;i<s;i++)

{

a=b\*c;

c=a+b;

b=c-a;

a=(int)b/c;

gettimeofday(&t2,NULL);

if(t2.tv\_sec-t1.tv\_sec==1)

{

//printf("i is %d",i);

return i;

}

}

}

int threadFunc\_float(int s)

{

int i;

float a=1.2,b=2.3,c=4.5;

struct timeval t1;

struct timeval t2;

int timer=0;

while(timer==0)

{

gettimeofday(&t1,NULL);

for(i=0;i<s;i++)

{

a=b\*i;

c=a+b;

b=c-a;

a=(float)b/c;

gettimeofday(&t2,NULL);

if(t2.tv\_sec-t1.tv\_sec==1)

{

timer=1;

//printf("%d",i);

return i;

}

}

}

}

int main(void)

{

int i=0,j=0;

int noThreads;

int op;

/\* if (pthread\_mutex\_init(&lock, NULL) != 0)

{

printf("\n mutex init failed\n");

return 1;

}

\*/

printf("Enter Operation 1 int 2 float");

fflush(stdout);

scanf("%d",&op);

printf("Enter threads");

fflush(stdout);

scanf("%d",&noThreads);

pthread\_t pth[noThreads];

char \*fileName="values.txt";

int fileDesc = open(fileName,O\_CREAT|O\_RDWR,S\_IRWXU);

for(j=0;j<600;j++)

{

int f=0;

float avg=0;

for(i=0;i<noThreads;i++)

{

if(op==1)

{

pthread\_create(&pth[i],NULL,threadFunc\_int,100000000);

}

else

{

pthread\_create(&pth[i],NULL,threadFunc\_float,100000000);

}

}

for(i=0;i<noThreads;i++)

{

pthread\_join(pth[i],&f);

avg=(float)avg+(f\*4);

//printf("avg is %f",avg);

}

//printf("\n avg is %d",(int)avg/noThreads);

float iops=(float)(avg/noThreads);

printf("\n flops is %f ",iops);

printf("\n flops in Giga Hertz is %f",iops/1000000000);

char \*c = (char \*)malloc(sizeof(char)\*10);

sprintf (c,"%f",iops/1000000000);

strcat(c,"\r\n");

write(fileDesc,c,strlen(c));

}

return 0;

}

Pa1\_Memory.c

**#include <stdio.h>**

**#include <pthread.h>**

**#include <time.h>**

**#include <sys/time.h>**

**#include <fcntl.h>**

**#include <sys/stat.h>**

**#include <malloc.h>**

**pthread\_mutex\_t lock;**

**struct timeval start\_time,end\_time;**

**float seq\_thread\_func(long int size)**

**{**

**// allocate a memory of blocksize 20 mb ,**

**//read blocks of data using sequentital access from one memory area to other using memcpy**

**//and return the time taken for read and write**

**int k=0;**

**pthread\_mutex\_lock(&lock);**

**long int blockSize=(long int)(20\*1000000)/size;**

**char \*mem1;**

**char \*mem2;**

**mem1=malloc(blockSize\*size);**

**mem2=malloc(blockSize\*size);**

**strncat(mem2,"hello",blockSize\*size);**

**gettimeofday(&start\_time,NULL);**

**for(k=0;k<(long int)(blockSize);k++)**

**{**

**memcpy(mem1+k,mem2+k,size);**

**// copy required bytes from one memory to other sequentially**

**}**

**gettimeofday(&end\_time,NULL);**

**double data1=(double)start\_time.tv\_sec+((double)start\_time.tv\_usec/1000000);**

**double data2=(double)end\_time.tv\_sec+((double)end\_time.tv\_usec/1000000);**

**float dataTime=data2-data1;//calculate the time taken for read and write in sec**

**free(mem1);**

**free(mem2);**

**printf("%f",dataTime);**

**pthread\_mutex\_unlock(&lock);**

**return dataTime;**

**}**

**float random\_thread\_func(long int size)**

**{**

**// allocate a memory of blocksize 20 mb ,**

**//read blocks of data using random access from one memory area to other using memcpy**

**//and return the time taken for read and write**

**int random\_pos=0,k;**

**pthread\_mutex\_lock(&lock);**

**long int blockSize=(1000000\*20)/size;**

**char \*mem1;**

**char \*mem2;**

**mem1=malloc(blockSize\*size);//allocate memory**

**mem2=malloc(blockSize\*size);//allocate memory**

**strncat(mem2,"hello",blockSize\*size);**

**gettimeofday(&start\_time,NULL);**

**for(k=0;k<(long int)blockSize;k++)**

**{**

**random\_pos = rand()%(blockSize);**

**memcpy(mem1+random\_pos,mem2+random\_pos,size);**

**// copy required bytes from one memory to other in random\_pos position**

**}**

**gettimeofday(&end\_time,NULL);**

**double data1=(double)start\_time.tv\_sec+((double)start\_time.tv\_usec/1000000);**

**double data2=(double)end\_time.tv\_sec+((double)end\_time.tv\_usec/1000000);**

**float dataTime=data2-data1;//calculate the time taken for read and write in sec**

**free(mem1);**

**free(mem2);**

**printf("%f",dataTime);**

**pthread\_mutex\_unlock(&lock);**

**return dataTime;**

**}**

**void main()**

**{**

**float throughput,latency;**

**int access;**

**int nothreads;**

**int operation;**

**long int size;**

**int i;**

**float timeTaken=0,data=0;**

**printf("\n Block size:(Select 1 for 1B=1,2 for 1KB=1024,3 for 1MB=1048576):");**

**fflush(stdout);**

**scanf("%d",&size);**

**printf("Access Method : 1-Sequential 2-Random");**

**fflush(stdout);**

**scanf("%d",&access);**

**printf("Enter number of threads 1,2,4 :");**

**fflush(stdout);**

**scanf("%d",&nothreads);**

**pthread\_t pth[nothreads];**

**if(size==2)**

**{**

**// if 2 is selected assign 1024 bytes to read and write in memory**

**size=(long int)1024;**

**}**

**if(size==3)**

**{**

**// if 3 is selected assign 1024\*2014 bytes to read and write in memory**

**size=(long int)(1024\*1024);**

**}**

**if (pthread\_mutex\_init(&lock, NULL) != 0)**

**{**

**printf("\n mutex init failed\n");**

**return 1;**

**}**

**// create given no of threads and perform operations (read/write)**

**for(i=0;i<nothreads;i++)**

**{**

**if(access==1)**

**//this block is for doing memcopy sequentially**

**pthread\_create(&pth[i],NULL,seq\_thread\_func,(long int)size);**

**else**

**//this block is for doing memcopy to random memory**

**pthread\_create(&pth[i],NULL,random\_thread\_func,(long int)size);**

**}**

**// joining the thread with other thread**

**for(i=0;i<nothreads;i++)**

**{**

**pthread\_join(pth[i],&data);**

**timeTaken=timeTaken+data; // add the time taken for all threads**

**}**

**printf("%f",timeTaken);**

**timeTaken=timeTaken/nothreads;**

**latency=(float)(timeTaken\*size\*1000)/(2\*1000000\*20);**

**printf("Latency in milliseconds : %f \n",latency);**

**throughput = (float)(1000000\*20\*2)/(timeTaken);//(no of loops\*size\*2/time taken)**

**printf("Throughput in MB/sec : %f\n",(throughput)/(1024\*1024));**

**int filesc;**

**char \*fs="values.txt";**

**filesc = open(fs,O\_CREAT|O\_RDWR,S\_IRWXU);**

**char \*c = (char \*)malloc(sizeof(char)\*10000);**

**char \*s = (char \*)malloc(sizeof(char)\*10);**

**s[0]='\0';**

**c[0] = '\0';**

**strcat(c,"\r\n");**

**strcat(c,"size");**

**strcat(c," ");**

**strcat(c,"threads");**

**strcat(c," ");**

**strcat(c,"access");**

**strcat(c," ");**

**strcat(c,"throughput");**

**strcat(c," ");**

**strcat(c,"latency");**

**strcat(c,"\r\n");**

**sprintf (s,"%d",(int)size);**

**strcat(c,s);**

**strcat(c," ");**

**s[0]='\0';**

**sprintf (s,"%d",(int)nothreads);**

**strcat(c,s);**

**strcat(c," ");**

**sprintf(s,"%d",(int)access);**

**strcat(c,s);**

**strcat(c," ");**

**sprintf(s,"%f",(float)throughput/(1024\*1024));**

**strcat(c,s);**

**strcat(c," ");**

**s[0]='\0';**

**sprintf(s,"%f",(float)latency);**

**strcat(c,s);**

**strcat(c,"\r\n");**

**printf("\n c is \n %s",c);**

**write(filesc,c,strlen(c));**

**free(c);**

**free(s);**

**}**

Pa1\_Disk.c#include <stdio.h>

#include <pthread.h>

#include <time.h>

#include <sys/time.h>

#include <fcntl.h>

#include <sys/stat.h>

int fileDesc ;

pthread\_mutex\_t lock;

struct timeval start\_time,end\_time;

float write\_thread\_seq(long int size)

{

// open the file , write blocks of data using sequential access and return the time taken for write

pthread\_mutex\_lock(&lock);

long int blockSize=(long int)((1000000\*20)/size);

char \*fileName="trial.txt";

fileDesc = open(fileName,O\_RDWR,S\_IRWXU);

int itr;

char \*writeData;

writeData=(char \*)malloc(size);

gettimeofday(&start\_time,NULL);

for(itr=0;itr<blockSize;itr++)

{

int currentPageSize =write(fileDesc,writeData,size);//write sequentially from starting position of file

}

gettimeofday(&end\_time,NULL);

double data1=(double)start\_time.tv\_sec+((double)start\_time.tv\_usec/1000000);

double data2=(double)end\_time.tv\_sec+((double)end\_time.tv\_usec/1000000);

float dataTime=(double)data2-data1;//calculate the time taken for write in sec

pthread\_mutex\_unlock(&lock);

free(writeData);

return dataTime;

}

float read\_thread\_seq(long int size)

{

// open the file , read blocks of data using sequential access and return the time taken for read

long int blockSize=(long int)(1000000\*20)/size;

pthread\_mutex\_lock(&lock);

char \*fileName="trial.txt";

fileDesc = open(fileName,O\_RDWR,S\_IRWXU);

int itr;

char \*readData;

readData=(char \*)malloc(size);

gettimeofday(&start\_time,NULL);

for(itr=0;itr<blockSize;itr++)

{

int currentPageSize =read(fileDesc,readData,size);//read sequentially from starting position of file

}

gettimeofday(&end\_time,NULL);

double data1=(double)start\_time.tv\_sec+((double)start\_time.tv\_usec/1000000);

double data2=(double)end\_time.tv\_sec+((double)end\_time.tv\_usec/1000000);

float dataTime=data2-data1;//calculate the time taken for read in sec

pthread\_mutex\_unlock(&lock);

free(readData);

return dataTime;

}

float write\_thread\_random(long int size)

{

// open the file , write blocks of data using random access and return the time taken for write

int itr;

off\_t random\_pos;

long int blockSize=(long int)(1000000\*20)/size;

char \*fileName="trial.txt";

char \*writeData;

fileDesc = open(fileName,O\_RDWR,S\_IRWXU);

pthread\_mutex\_lock(&lock);

writeData=(char \*)malloc(size);

gettimeofday(&start\_time,NULL);

for(itr=0;itr<blockSize;itr++)

{

random\_pos = rand()%(int)blockSize;//assign random block to write

int currentPageSize =pwrite(fileDesc,writeData,size,random\_pos);// writes from random block position

}

gettimeofday(&end\_time,NULL);

double data1=(double)start\_time.tv\_sec+((double)start\_time.tv\_usec/1000000);

double data2=(double)end\_time.tv\_sec+((double)end\_time.tv\_usec/1000000);

float dataTime=data2-data1;//calculate the time taken for write in sec

free(writeData);

pthread\_mutex\_unlock(&lock);

return dataTime;

}

float read\_thread\_random(long int size)

{

// open the file , read blocks of data using random access and return the time taken for read

int itr;

off\_t random\_pos;

long int blockSize=(long int)(1000000\*20)/size;

char \*fileName="trial.txt";

char \*readData;

fileDesc = open(fileName,O\_RDWR,S\_IRWXU);

pthread\_mutex\_lock(&lock);

readData=(char \*)malloc(size);

gettimeofday(&start\_time,NULL);

for(itr=0;itr<blockSize;itr++)

{

random\_pos = rand()%(int)blockSize;//assign random block to read

int currentPageSize =pread(fileDesc,readData,size,random\_pos);// read from random block position

}

gettimeofday(&end\_time,NULL);

double data1=(double)start\_time.tv\_sec+((double)start\_time.tv\_usec/1000000);

double data2=(double)end\_time.tv\_sec+((double)end\_time.tv\_usec/1000000);

float dataTime=data2-data1; //calculate the time taken for read in sec

free(readData);

pthread\_mutex\_unlock(&lock);

return dataTime;

}

void main()

{

float throughput,latency;

int access;

int nothreads;

int operation;

long int size;

int i;

float timeTaken=0,dataTime;

printf("Enter Operation to perform 1-Read 2-Write");

fflush(stdout);

scanf("%d",&operation);

printf("\n Block size:(Select 1 for 1B=1,2 for 1KB=1024,3 for 1MB=1048576):");

fflush(stdout);

scanf("%d",&size);

printf("Access Method : 1-Sequential 2-Random");

fflush(stdout);

scanf("%d",&access);

printf("Enter number of threads 1,2,4 :");

fflush(stdout);

scanf("%d",&nothreads);

pthread\_t pth[nothreads];

if(size==2)

{

// if 2 is selected assign 1024 bytes to read or write from disk

size=(long int)(1024);

}

if(size==3)

{ // if 3 is selected assign 1024\*2014 bytes to read or write from disk

size=(long int)(1024\*1024);

}

if (pthread\_mutex\_init(&lock, NULL) != 0)

{

printf("\n mutex init failed\n");

return 1;

}

char \*fileName="trial.txt";

char \*f="hs";

fileDesc = open(fileName,O\_CREAT|O\_RDWR,S\_IRWXU);

for(i=0;i<=(1024\*1024\*20);i++)

{

int currentPageSize =write(fileDesc,f,2);

}

// create given no of threads and perform operations (read/write)

for(i=0;i<nothreads;i++)

{

if(access==1 && operation==1)

{

//this block is for read operation sequential access

pthread\_create(&pth[i],NULL,read\_thread\_seq,(long int)size);

//creates a thread and calls read\_thread function

}

if(access==2 && operation==1 )

{ //this block is for read operation random access

pthread\_create(&pth[i],NULL,read\_thread\_random,(long int)size);

//creates a thread and calls read\_thread\_random function

}

if(access==1 && operation==2)

{ //this block is for write operation using sequential access

pthread\_create(&pth[i],NULL,write\_thread\_seq,(long int)size);

//creates a thread and calls write\_thread\_seq function

}

if(access==2 && operation==2)

{ //this block is for write operation using random access

pthread\_create(&pth[i],NULL,write\_thread\_random,(long int)size);

//creates a thread and calls write\_thread\_random function

}

}

// joining the thread with other thread

for(i=0;i<nothreads;i++)

{

pthread\_join(pth[i],&dataTime);

timeTaken=(float)timeTaken+dataTime; // add the time taken for all threads

}

timeTaken=(timeTaken)/nothreads;

throughput=(float)(1000000\*20)/(timeTaken); //(no of loops\*size\*2/time taken)

latency=(float)(timeTaken\*1000\*size)/(1000000\*20);

if(access==1 && operation==1)

{

printf("Throughput for sequential access read is %f ",(float)(throughput)/(1024\*1024));

printf("Latency for sequential access read is %f",latency);

}

if(access==2 && operation==1)

{

printf("Throughput for random access read is %f",(float)(throughput)/(1024\*1024));

printf("Latency for random access read is %f",latency);

} if(access==1 && operation==2)

{

printf("Throughput for sequential access write is %f",throughput/(1024\*1024));

printf("Latency for sequential access write is %f",latency);

}

if(access==2 && operation==2)

{

printf("Throughput for random access write is %f",throughput/(1024\*1024));

printf("Latency for random access write is %f",latency);

}

//store result in file

int filesc;

char \*fs="values.txt";

filesc = open(fs,O\_CREAT|O\_RDWR,S\_IRWXU);

char \*c = (char \*)malloc(sizeof(char)\*10000);

char \*s = (char \*)malloc(sizeof(char)\*100);

c[0] = '\0';

strcat(c,"\r\n");

strcat(c,"size");

strcat(c," ");

strcat(c,"operation");

strcat(c," ");

strcat(c,"threads");

strcat(c," ");

strcat(c,"access");

strcat(c," ");

strcat(c,"throughput");

strcat(c," ");

strcat(c,"latency");

strcat(c,"\r\n");

sprintf (s,"%d",(int)size);

strcat(c,s);

strcat(c," ");

sprintf (s,"%d",(int)operation);

strcat(c,s);

strcat(c," ");

sprintf (s,"%d",(int)nothreads);

strcat(c,s);

strcat(c," ");

sprintf (s,"%d",(int)access);

strcat(c,s);

strcat(c," ");

sprintf (s,"%f",(float)throughput/(1024\*1024));

strcat(c,s);

strcat(c," ");

sprintf (s,"%f",(float)latency);

strcat(c,s);

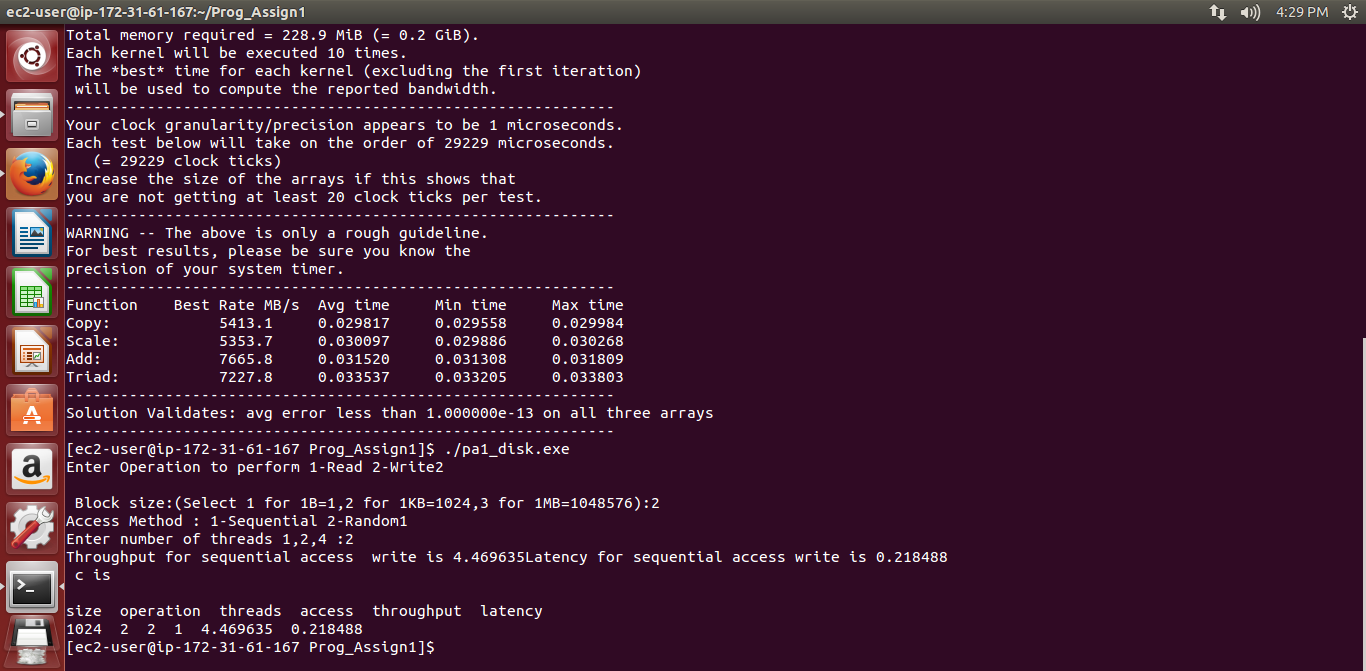
strcat(c,"\r\n");

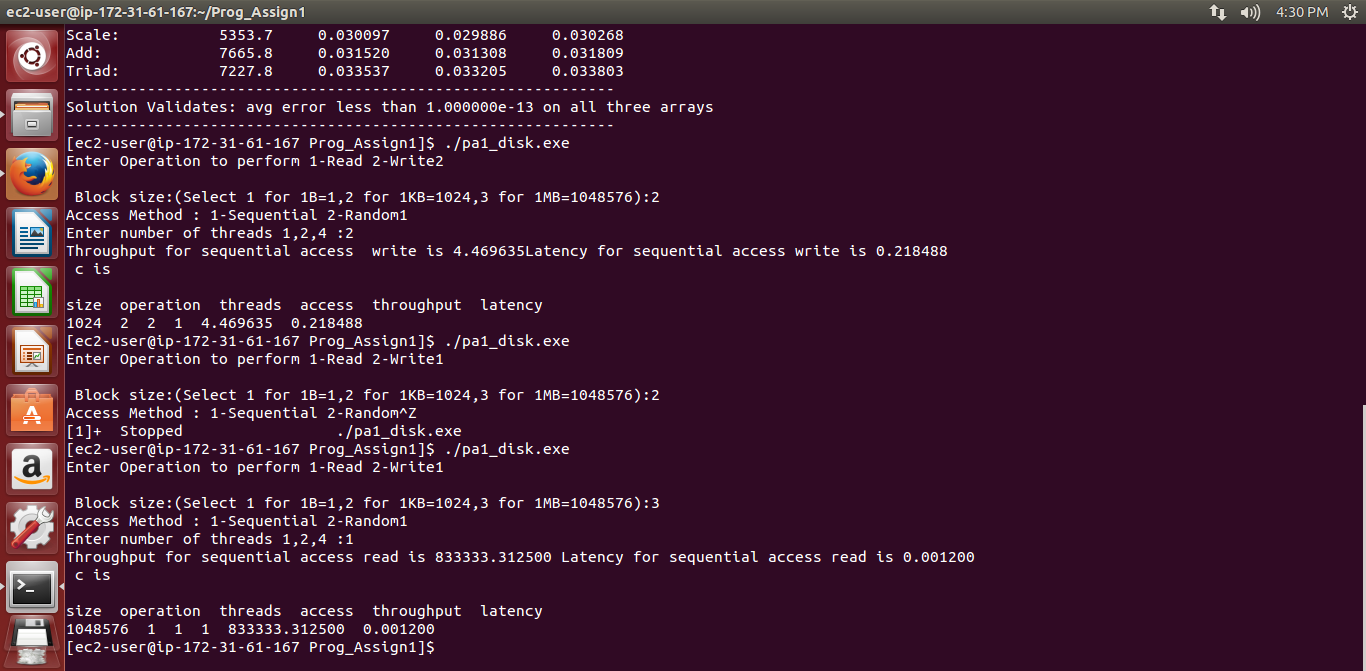
printf("\n c is \n %s",c);

write(filesc,c,strlen(c));

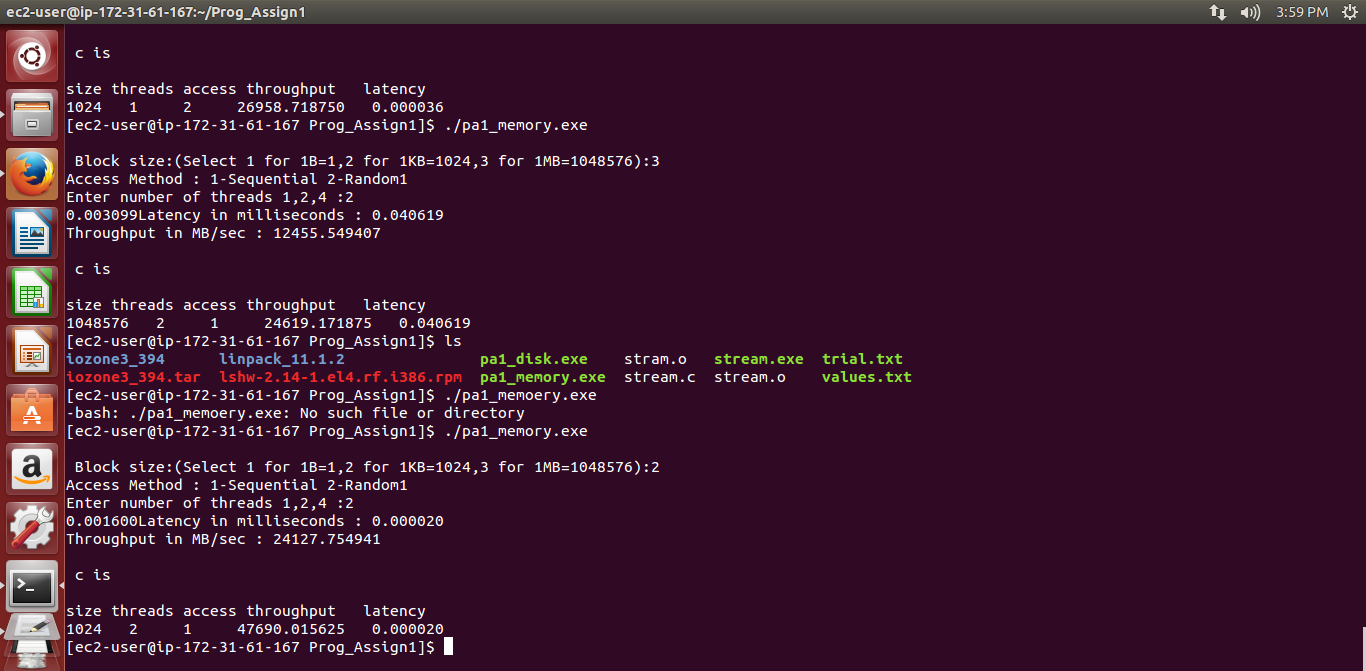
}

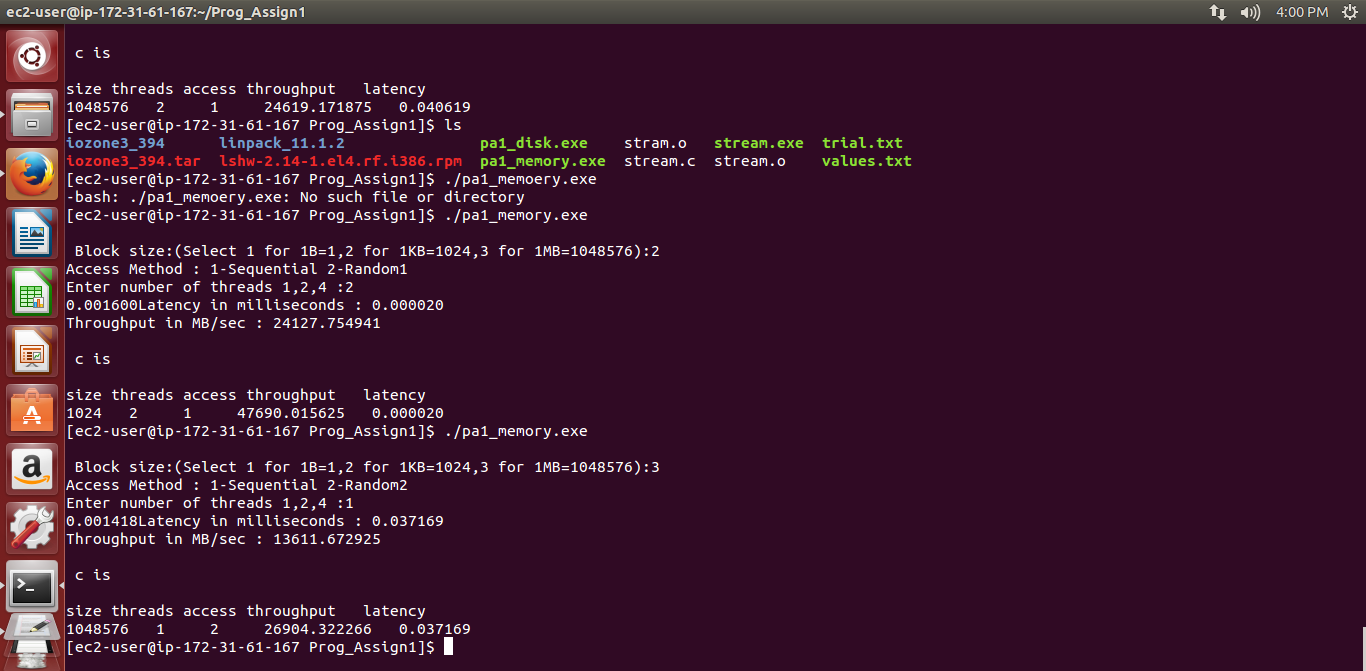
Disk Benchmark Screenshots





Memory Screenshot





Cpu Screenshot

