作業系統 Report

作業一 R10945061 林宇恆

1. Motivation:

根據作業要求敘述，需要各自執行 matmult 以及 sort 兩個測試檔，且需要修改 sort.c 來達到相同的結果，而以上兩個檔案都需要大量記憶體來運行，因此根據作業要求concurrently，也就需要上次作業的multithread來協助達到共行。根據作業要求使用pageTable，FrameTable以及SwapTable來解決有限記憶體的問題，分別代表virtual，physical跟swap，因此如下圖所示，在machine.h中定義以下表格來記錄使用過程。

```
TranslationEntry *pageTable;
unsigned int pageTableSize;
bool ReadMem(int addr, int size, int* value);
bool UsedPhyPage[NumPhysPages];
bool UsedVirtual[NumPhysPages];
int machine_id;
int PhyPage[NumPhysPages];
TranslationEntry *main[NumPhysPages];
```

發想為透過pageTable將logical memory映射到physical memory，映射後pageTable透過1bit的標記判斷page有沒有效，若有效則代表該page還在main memory中執行，無效則代表page在輔助記憶體中（如硬碟,SSD）。

根據作業提示在以下file中執行。

- For the disk usage details, see:
  - /filesys/synchdisk.*
- For the swap space initialization:
  - /userprog/userkernel.*
- For the table maintaining, see:
  - /machine/machine.*
- For the loading of pages:
  - /userprog/addrspace.*

2. Implementation:

在userkernel.h中建立一個新的SynchDisk(SwapDisk)，且初始化來模擬輔助記憶

體。

```
    Machine *machine;
    FileSystem *fileSystem;
    SynchDisk *SwapDisk;
#ifdef FILESYS
    SynchDisk *synchDisk;
#endif // FILESYS
```

```
void
UserProgKernel::Initialize()
{
    ThreadedKernel::Initialize();   // init multithreading

    machine = new Machine(debugUserProg);
    fileSystem = new FileSystem();
    SwapDisk = new SynchDisk("New SwapDisk");
#ifdef FILESYS
    synchDisk = new SynchDisk("New SynchDisk");
#endif // FILESYS
}
void UserProgKernel::Initialize(SchedulerType type)
{
    ThreadedKernel::Initialize(type);   // init multithreading
    machine = new Machine(debugUserProg);
    fileSystem = new FileSystem();
    SwapDisk = new SynchDisk("New SwapDisk");// Swap disk for virtual memory
#ifdef FILESYS
    synchDisk = new SynchDisk("New SynchDisk");
#endif // FILESYS
}
```

在addrspace的load函數中新增，用for迴圈來尋找可用空間，j則用來檢查第幾個frame已被使用，若使用則加一，若j小於NumPhysPages，也就是還有空frame時，就可以將Page放入main memory，第二種情況是main memory滿了，則透過下面的while迴圈檢查virtual memory，最後將其放入輔助記憶體，達到context-switch。

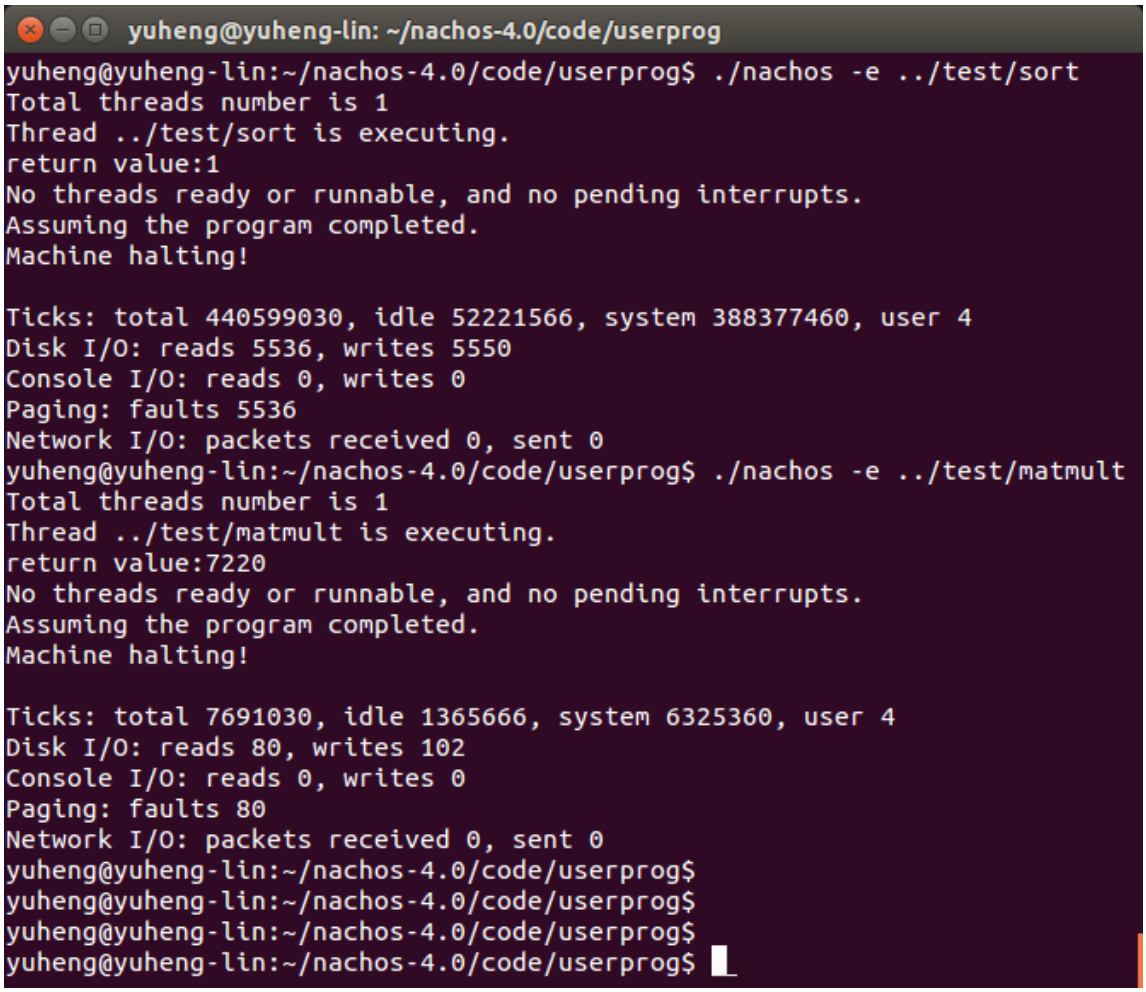
```
if (noffH.code.size > 0) {

for(unsigned int j=0,i=0;i < numPages ;i++){
        j=0;
        while(kernel->machine->UsedPhyPage[j]!=FALSE&&j<NumPhysPages){j++;}
        // if main memory is enough, put the page to main memory
        if(j<NumPhysPages){
            kernel->machine->UsedPhyPage[j]=TRUE;
            kernel->machine->PhyPage[j]=id;
            kernel->machine->main[j]=&pageTable[i];
            pageTable[i].physicalPage = j;
            pageTable[i].valid = TRUE;
            pageTable[i].use = FALSE;
            pageTable[i].dirty = FALSE;
            pageTable[i].readOnly = FALSE;
            pageTable[i].page_id =id;
            pageTable[i].counter++; // counter for saving memory
            executable->ReadAt(&(kernel->machine->mainMemory[j*PageSize]),PageSize, noffH.code.inFileAddr+(i*PageSize));
        }
        // if main memory is not enough,put the page to virtual memory
        else{
            char *buffer;
            buffer = new char[PageSize];
            tmp=0;
            while(kernel->machine->UsedVirtual[tmp]!=FALSE){tmp++;}
            kernel->machine->UsedVirtual[tmp]=true;
            pageTable[i].virtualPage=tmp;
            pageTable[i].valid = FALSE;
            pageTable[i].use = FALSE;
            pageTable[i].dirty = FALSE;
            pageTable[i].readOnly = FALSE;
            pageTable[i].page_id =id;
            executable->ReadAt(buffer,PageSize, noffH.code.inFileAddr+(i*PageSize));
            kernel->SwapDisk->WriteSector(tmp,buffer); // write in virtual memory (SwapDisk)

        }
    }
}
```

演算法的部分使用了Random replacement algorithm，他是隨機選擇一項且在必要時丟棄，因此他不需要保留有關訪問歷史的任何資訊。第一個if條件為當main memory沒滿的情況下載入page到main memory中，buffer則用來暫時儲存page，第二格條件則為當main memory滿了的情況，dead則為被選到的人因為之後就會丟棄因此必死，32為32位元。

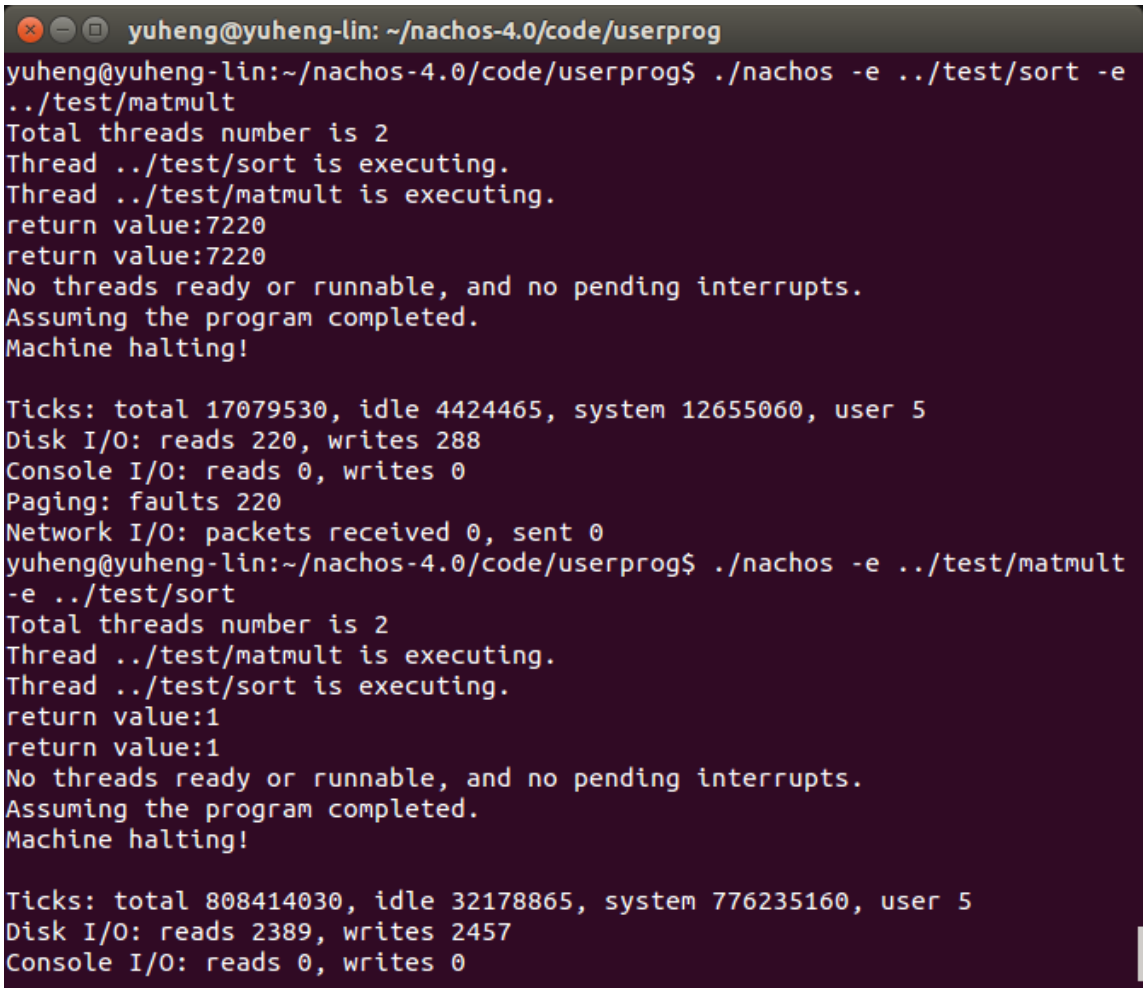
```
else if (!pageTable[vpn].valid) {
    kernel->stats->numPageFaults++; // page fault
    j=0;
    while(kernel->machine->UsedPhyPage[j]!=FALSE&&j<NumPhysPages){j++;}
    if(j<NumPhysPages){
        char *buffer; //save page
        buffer = new char[PageSize];
        kernel->machine->UsedPhyPage[j]=TRUE;
        kernel->machine->PhyPage[j]=pageTable[vpn].page_id;
        kernel->machine->main[j]=&pageTable[vpn];
        pageTable[vpn].physicalPage = j;
        pageTable[vpn].valid = TRUE;
    pageTable[vpn].counter++;

        kernel->SwapDisk->ReadSector(pageTable[vpn].virtualPage, buffer);
        bcopy(buffer,&mainMemory[j*PageSize],PageSize);
    }
    else{
        char *buffer1;
        char *buffer2;
        buffer1 = new char[PageSize];
        buffer2 = new char[PageSize];
        //Random
        dead = (rand()%32);

        bcopy(&mainMemory[dead*PageSize],buffer1,PageSize);
        kernel->SwapDisk->ReadSector(pageTable[vpn].virtualPage, buffer2);
        bcopy(buffer2,&mainMemory[dead*PageSize],PageSize);
        kernel->SwapDisk->WriteSector(pageTable[vpn].virtualPage,buffer1);

        main[dead]->virtualPage=pageTable[vpn].virtualPage;
        main[dead]->valid=FALSE;

        pageTable[vpn].valid = TRUE;
        pageTable[vpn].physicalPage = dead;
        kernel->machine->PhyPage[dead]=pageTable[vpn].page_id;
        main[dead]=&pageTable[vpn];
    }
}
```

3. Result:

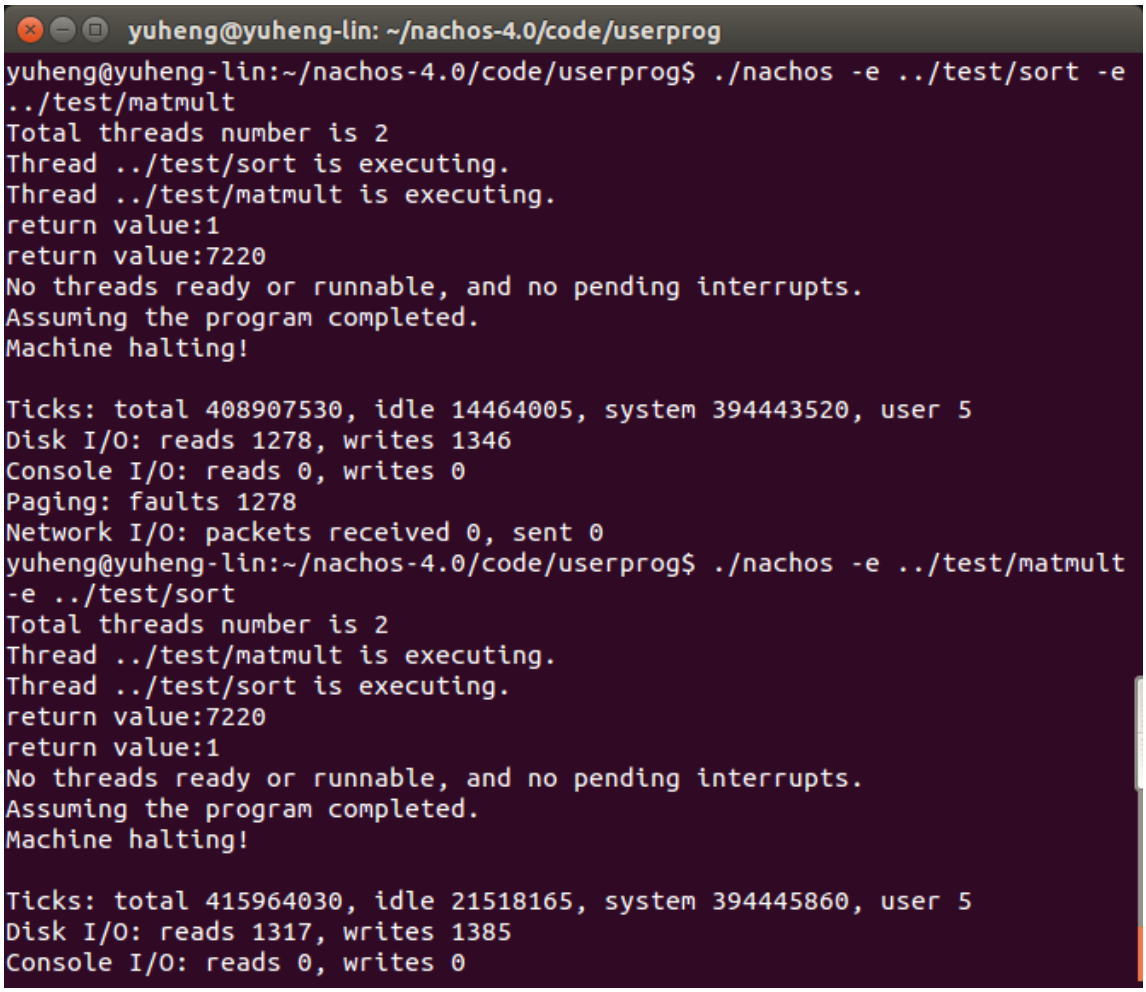
分開執行的結果如下，各自都可以達到準確的答案。



但一開始沒有使用multithread導致執行結果如下圖，共同執行的話結果會跟第二個測試檔相同。



最後加入了multithread後就能夠得到正確答案了。


yuheng@yuheng-lin: ~/nachos-4.0/code/userprog
yuheng@yuheng-lin:~/nachos-4.0/code/userprog$ ./nachos -e ../test/sort -e ../test/matmult
Total threads number is 2
Thread ../test/sort is executing.
Thread ../test/matmult is executing.
return value:1
return value:7220
No threads ready or runnable, and no pending interrupts.
Assuming the program completed.
Machine halting!
Ticks: total 408907530, idle 14464005, system 394443520, user 5
Disk I/O: reads 1278, writes 1346
Console I/O: reads 0, writes 0
Paging: faults 1278
Network I/O: packets received 0, sent 0
yuheng@yuheng-lin:~/nachos-4.0/code/userprog$ ./nachos -e ../test/matmult -e ../test/sort
Total threads number is 2
Thread ../test/matmult is executing.
Thread ../test/sort is executing.
return value:7220
return value:1
No threads ready or runnable, and no pending interrupts.
Assuming the program completed.
Machine halting!
Ticks: total 415964030, idle 21518165, system 394445860, user 5
Disk I/O: reads 1317, writes 1385
Console I/O: reads 0, writes 0