# OPERATING SYSTEMS PROJECT REPORT

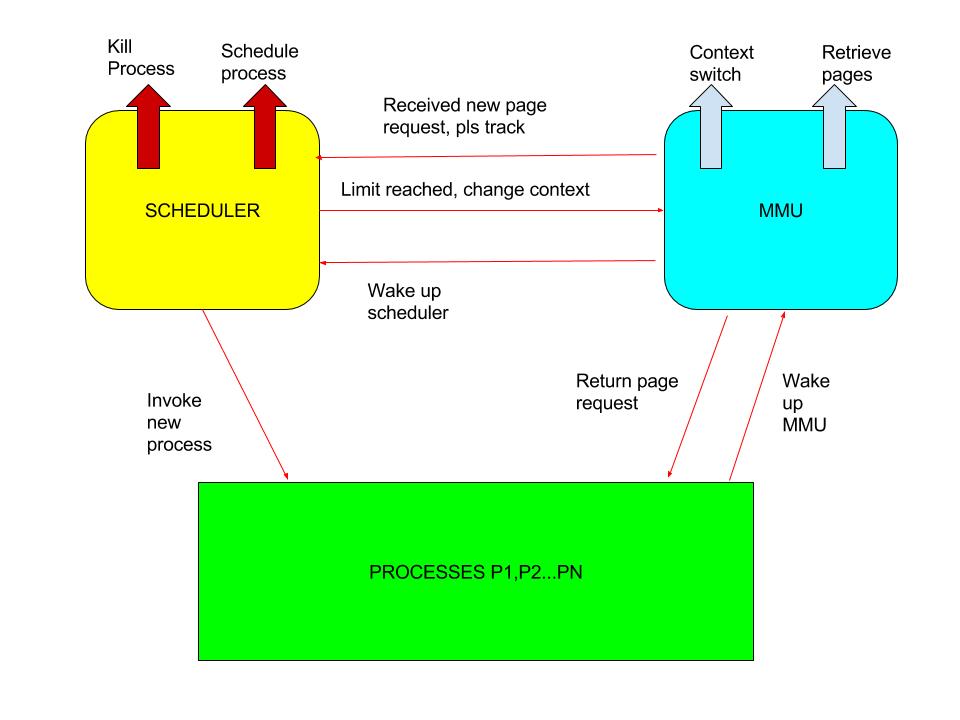
By

Chandramouli S Sastry (1PI13CS046)

Darshan SJ (1PI13CS051)

Debarati Das (1PI13CS052)

**System Design**



**Python Code**

from multiprocessing import \*

from threading import Thread

import random,signal

import os

num\_requests=0

l=Lock()

page\_fault=0

time=0

tlb\_miss=0

end\_for\_app,end\_for\_mmu=Pipe()

end\_for\_mmuS,end\_for\_schedulerM=Pipe()

sem1=Semaphore(0)#for waking mmu from app

sem2=Semaphore(0)#for waking scheduler from mmu

end=Semaphore(0)#for indicating current process is over...

switch=Semaphore(0)#for invoking mmu switch

switch\_over=Semaphore(0)

proc\_name=Queue()

class MMU(Process):

app\_pipe=end\_for\_mmu

sem\_scheduler=sem2

sem\_app=sem1

scheduler\_pipe=end\_for\_mmuS

def \_\_init\_\_(self,t,p,taccess,phit,pmiss,n):

self.tlb={}#page\_no:num

self.memory={}#page\_no:num

self.t=t

self.p=p

self.taccess=taccess

self.phit=phit

self.current\_process=0

self.pmiss=pmiss

self.finished=0

Process.\_\_init\_\_(self)

self.n=n

def run(self):

global num\_requests,page\_fault,time,tlb\_miss

t=Thread(target=self.mmu\_switch)

t.start()

while True:

MMU.sem\_app.acquire()

page\_num=MMU.app\_pipe.recv()

num\_requests+=1

if page\_num==-1:

num\_requests-=1

self.finished+=1

if self.finished==self.n:

print 'Page Fault Rate',page\_fault/float(num\_requests)

print 'Effective Memory Access Time',time/float(num\_requests)

print 'Number of TLB Misses',tlb\_miss

os.kill(os.getppid(),signal.SIGTERM)

os.kill(os.getpid(),signal.SIGTERM)

end.release()

continue

elif page\_num in self.tlb:

# print 'Request by ',proc\_name.get()+'. Page num',page\_num,'Time:',str(self.taccess+self.phit)

for i in self.tlb:

self.tlb[i]+=1

self.tlb[page\_num]=0

time+=self.taccess+self.phit

elif page\_num in self.memory:

# print 'TLB MISS'

# print 'Request by ',proc\_name.get()+'. Page num',page\_num,'Time:',str(self.taccess+2\*self.phit)

tlb\_miss+=1

max\_num=-1

max\_page=0

for i in self.memory:

self.memory[i]+=1

self.memory[page\_num]=0

if len(self.tlb)==self.t:

for i in self.tlb:

if self.tlb[i]>max\_num:#?

max\_num=self.tlb[i]

max\_page=i

del self.tlb[max\_page]

for i in self.tlb:

self.tlb[i]+=1

self.tlb[page\_num]=0

time+=self.taccess+2\*self.phit

else:

# print 'PAGE FAULT'

page\_fault+=1

tlb\_miss+=1

max\_num,max\_page=-1,0

if len(self.memory)==self.p:

for i in self.memory:

if self.memory[i]>max\_num:

max\_num=self.memory[i]

max\_page=i

del self.memory[max\_page]

#self.memory[self.current\_process][page\_num]=0#its below..

for i in self.memory:

self.memory[i]+=1

self.memory[page\_num]=0

max\_num=-1

max\_page=0

if len(self.tlb)==self.t:

for i in self.tlb:

if self.tlb[i]>max\_num:

max\_num=self.tlb[i]

max\_page=i

del self.tlb[max\_page]

for i in self.tlb:

self.tlb[i]+=1

self.tlb[page\_num]=0

# print 'Request by ',proc\_name.get()+'. Page num',page\_num,'Time:',str(self.taccess+self.phit+self.pmiss)

time+=self.taccess+self.phit+self.pmiss

MMU.sem\_scheduler.release()

def mmu\_switch(self):

while True:

switch.acquire()

self.count=0

self.tlb={}

switch\_over.release()

class App(Process):

mmu\_pipe=end\_for\_app

sem\_mmu=sem1

def \_\_init\_\_(self,name,max\_req,total\_page,sem):

Process.\_\_init\_\_(self)

self.name=name

self.max\_req=max\_req

self.total\_page=total\_page

self.sem=sem

def run(self):

count=1

while count<=self.max\_req:

self.sem.acquire()

page\_num=random.randint(1,self.total\_page)

proc\_name.put(self.name)

App.mmu\_pipe.send((self.name,page\_num))

App.sem\_mmu.release()

count+=1

self.sem.acquire()

App.mmu\_pipe.send(-1)

App.sem\_mmu.release()

class Scheduler\_details:

pass

class Scheduler(Process):

def \_\_init\_\_(self,C):

Process.\_\_init\_\_(self)

self.process\_list=[]

self.semaphore\_list=[]

self.current\_process=0

self.count=0

def remove\_current\_process(self):

while True:

end.acquire()

del self.process\_list[self.current\_process]

del self.semaphore\_list[self.current\_process]

if len(self.process\_list)==0:

break

switch.release()

switch\_over.acquire()

self.count=0

self.current\_process=(self.current\_process)%len(self.process\_list)

self.semaphore\_list[self.current\_process].release()

def schedule(self):

self.current\_process=0

self.count=0

self.semaphore\_list[self.current\_process].release()

while True:

sem2.acquire()

self.count+=1

if self.count==self.C:

switch.release()

switch\_over.acquire()

self.current\_process=(self.current\_process+1)%len(self.process\_list)

self.count=0

self.semaphore\_list[self.current\_process].release()

def run(self):

with open('proc\_config.txt') as f:

t=f.readlines()

for i in t[1:]:

i=i.split(',')

self.semaphore\_list.append(Semaphore(0))

self.process\_list.append(App(i[0],int(i[2].strip()),int(i[1].strip()),self.semaphore\_list[-1]))

with open('scheduler\_config.txt') as f:

self.C=int(f.readlines()[-1])

with open('mmu\_config.txt') as f:

t=f.readlines()[-1]#P,Phit,Pmiss,T,Taccess..t,p,taccess,phit,pmiss,C

k=map(int,t.split(','))

mmu=MMU(k[3],k[0],k[-1],k[1],k[2],len(self.process\_list))

mmu.start()

for proc in self.process\_list:

proc.start()

t=Thread(target=Scheduler.remove\_current\_process,args=(self,))

t.start()

self.schedule()

if \_\_name\_\_=='\_\_main\_\_':

Scheduler(None).start()

**Analysis of Various Input and Output**

|  |  |  |  |
| --- | --- | --- | --- |
| Category | Input | Observation | Inference |
| **SCHEDULER** | App,V,N  ps,3,10000  ls,5,100000  P,Phit,Pmiss,T,Taccess  8,200,800000,4,20  C  10 | Page Fault Rate 7.27272727273e-05  Effective Memory Access Time 368.723636364  Number of TLB Misses 49806 | In these set of results, we observe that Page Fault Rate is same in all cases. This is because the total number of pages required by both the processes is same as the memory size. We observe that the number of TLB misses decrease as the Slice is increased and saturates. |
| App,V,N  ps,3,10000  ls,5,100000  P,Phit,Pmiss,T,Taccess  8,200,800000,4,20  C  1000 | Page Fault Rate 7.27272727273e-05  Effective Memory Access Time 315.285454545  Number of TLB Misses 20415 |
| App,V,N  ps,3,10000  ls,5,100000  P,Phit,Pmiss,T,Taccess  8,200,800000,4,20  C  10000 | Page Fault Rate 7.27272727273e-05  Effective Memory Access Time 314.625454545  Number of TLB Misses 20052 |
| App,V,N  ps,3,10000  ls,5,100000  P,Phit,Pmiss,T,Taccess  8,200,800000,4,20  C  100000 | Page Fault Rate 7.27272727273e-05  Effective Memory Access Time 314.641818182  Number of TLB Misses 20061 |
| **SCHEDULER-With Page Size of ls bigger.** | App,V,N  ps,3,10000  ls,6,100000  P,Phit,Pmiss,T,Taccess  8,200,800000,4,20  C  1000 | Page Fault Rate 0.000418181818182  Effective Memory Access Time 615.601818182  Number of TLB Misses 33627 | Here, we increased the page size of ls from 5 to 6. We observe that the Number of TLB misses decreases as we increase time slice. The result here is clearer than the previous set of processes. Here, we can observe that the Page Fault Rate goes on converging with increase in Scheduler slice. Once, scheduler slice becomes as large as the longest process, page fault rate will become constant. |
| App,V,N  ps,3,10000  ls,5,100000  P,Phit,Pmiss,T,Taccess  8,200,800000,4,20  C  10000 | Page Fault Rate 8.18181818182e-05  Effective Memory Access Time 345.589090909  Number of TLB Misses 33083 |
| App,V,N  ps,3,10000  ls,6,100000  P,Phit,Pmiss,T,Taccess  8,200,800000,4,20  C  100000 | Page Fault Rate 7.27272727273e-05  Effective Memory Access Time 314.641818182  Number of TLB Misses 20061 |
| **PROCESS** | App,V,N  ps,3,10000  ls,5,100000  as,10,1000  bs,10,1000  P,Phit,Pmiss,T,Taccess  8,200,800000,4,20  C  10000 | Page Fault Rate 0.00375892857143  Effective Memory Access Time 3264.65  Number of TLB Misses 21425 | Here, we test the results by varying the process-config file. In case 1, we added more processes and in case-2, we made process ls larger. We observe that the performance is more affected when 1 process becomes bigger. We have maintained the total number of pages in case 1 and case 2 to be same to bring the contrast. |
| App,V,N  ps,3,10000  ls,25,100000  P,Phit,Pmiss,T,Taccess  8,200,800000,4,20  C  10000 | Page Fault Rate 0.616827272727  Effective Memory Access Time 493711.194545  Number of TLB Misses 84008 |
| **MEMORY** | App,V,N  ps,3,10000  ls,5,100000  as,10,1000  bs,10,1000  P,Phit,Pmiss,T,Taccess  16,200,800000,8,20  C  10000 | Page Fault Rate 0.000294642857143  Effective Memory Access Time 456.432142857  Number of TLB Misses 435 | In this case, processes ps,as and bs will complete in 1 slice itself. We have doubled the Memory size and the tlb size. We find that the performance has improved (wrt the first row in Process Section). |
| App,V,N  ps,3,10000  ls,5,100000  as,10,1000  bs,10,1000  P,Phit,Pmiss,T,Taccess  16,200,800000,4,20  C  10000 | Page Fault Rate 0.000294642857143  Effective Memory Access Time 493.735714286  Number of TLB Misses 21325 | Here, we have halved the TLB size wrt the previous configuration. As expected, the number of TLB misses and the effective access time increase. |
| App,V,N  ps,3,10000  ls,5,100000  as,60,1000  bs,50,1000  P,Phit,Pmiss,T,Taccess  16,200,800000,4,20  C  10000 | Page Fault Rate 0.0130803571429  Effective Memory Access Time 10720.9285714  Number of TLB Misses 21985 | In this case, we have added 2 new processes with large number of pages. As expected, there is a drastic increase in access time and page fault rate. However, number of TLB misses is more or less similar to previous result. |
| App,V,N  ps,3,10000  ls,5,100000  as,60,1000  bs,50,1000  P,Phit,Pmiss,T,Taccess  30,200,800000,4,20  C-10000 | Page Fault Rate 0.00873214285714  Effective Memory Access Time 7243.47678571  Number of TLB Misses 22125 | Here, we increase the memory size and observe the results. As expected, the system improves as we go on improving the memory size. |
| App,V,N  ps,3,10000  ls,5,100000  as,60,1000  bs,50,1000  P,Phit,Pmiss,T,Taccess  45,200,800000,4,20  C-10000 | Page Fault Rate 0.00385714285714  Effective Memory Access Time 3344.24107143  Number of TLB Misses 22007 |

**Conclusions**

These are the following conclusions that we can make from the above table:

1. If we increase scheduler time slice, the memory access time improves. However, what is not observed is the degree of concurrency that is directly influenced by the scheduler time slice.
2. If a process is bigger, then performance will deteriorate, given that the other configurations remain same. However, even if the number of processes increase, the impact isn’t too big.
3. One thing that we can observe across all the inputs and outputs is that if memory and tlb are bigger, then the performance of the system improves.