Project 3 3-parts

Parts 1 & 2 - demand paged V.M.

Part3 - Networking - Client / Server (Distributed Locks/CVs

An approach to parts 1&2

Compile & run from VM

CUntile part 3 is ready

move to network dir

vm directory Makefile

Nachos will use the TLB =>PageFaultException

Not the page table

#1 anywhere you see

machine->pageTable = pageTable;

comment it out

if (which == SyscallException) {

} else if (which == PageFaultException) {

//part 1 & 2 code

} else {

}

Step 1 - Populate the TLB from the page table

Project 2: Infinite memory (check)

still preload (check)

On a PageFaultException

Use pagetable to populate TLB

1->What is the VPN containing the needed V.A.?

The needed address is in register 39 - Bad VAddrReg

Divided by PageSize

currentThread->space->pageTable

Copy the pagetable data into TLB

2->Where to copy to in TLB?

machine->TLB

array of TranslationEntry

size 4

method: treat it like a “circular Queue”

create a variable - int curentTLB = 0;

write to that index

currentTLB = (currentTLB++)% TLBSize

disable interrupts

//populate TLB

restore interrupts

On a context switch

Invalidate the TLB

TLBLock ->Acquire();

for (int i=0;i<TLBSize;i++) {

machine->TLB[i].valid = false;

}

AddSpace ->SaveState (interupt already off for AddSpace)

->RestoreState

Thread->SaveUserState

->RestoreUserState

-------------------------Result: all p2 runs

Step 2 - Implement IPT

Can be an array

Have NumPhysPages entries

Still maintain project 2 assumptions

Add IPT population code to where page table is set

where ever you do a memmapFind

|-> index to IPT

AddrSpace constructor

Fork syscall

Exit

on a clear - set valid to false in IPT

On a PFE, use IPT

Issue: On a PFE, we have VPN, IPT indexed by PPN

Do a sequential search of 3 values

valid bit is true

match VPN

match AddrSpace\*

If all 3 match, copy that index location to TLB

Two Choices (Because you can’t change TranslationEntry)

Option 1-> Create a class that inherits from TranslationEntry

add AddrSpace\* / Process ID

Option 2->Copy/paste T.E to a new class

-----------------------------------Result: p2 runs

Step 3 - Don’t preload anything

still lots of memory

can now get an IPT miss

Solution: move the preloading code & IPT population codefrom step 2

into P.F.E. handling code

AddrSpace Constructor

For (i = 0 to numPages) {

|

//comment out int ppn = bitMap->Find();

|

pageTable[i].valid = false;

|

//commen out executable->ReadAt( );

}

similar change in Fork

On a PFE - Step2

----------------------------------------------------------

int ppn = -1;

for (i = 0 to NumPhyPages) {

if ( 3 values check) {

ppn = i;

break;

}

}

-----------------------------------------from step 2

-----------------------------------------------------------------

if (ppn == -1) {

handleIPTMiss(VPN);

}

------------------------------------------step 3

step 2 //populate the TLB from IPT

Handling an IPT miss

Allocate 1 memory page

copy from executable (if needed)

Update the pagetable

valid bit = true

physicalPage

populate IPT from pagetable

Issue 1: we do not have header info!

Issue 2: delete executable;

1-> Must keep executable open until process is done

move executable variable into AddrSpace

comment out delete executable

2-> noffH.code.inFileAdr + (i of PageSize)

Solution: Add data to page table

Another new class

-byteoffset

-in executable or not, or swap file

----------

code

-----------

I.data

------------

u.data

int inExec = divRoundUp( code + initdata);

for ( ) {

set not in exec

if (i < inExec) {

set byteOffset

set in executable

}

}

------------------------------result: project 2 still runs

Step 4 - Demand paged Virtual Memory

Set NumPhysPages to 32

memory will fillup

On a P.F.E. on an IPT miss, your Find will return -1

You select a page to replace

two policies

random

FIFO -> must use a queue

anytime on a Find(not -1)

append ppn to queue

on Find of -1, take front entry

@end of IPTMiss - add to Queue

command line -PRAND -PFIFO

Selected page may be dirty

must be sved to swapfile

only 1 swap file for O.S.

system.h/ .cc

open in Initialize (like executable)

from step 3, on IPT miss

int ppn = bitMap->Find();

if (ppn == -1) { //memory is full

ppn = handleMemoryFull( );

}

----------------------------

|-> rest of step 3

two userprogs :

matmult -> value to exit 7220 //less means dirty not handled, more means bigger proble

sort-> value to exit 1023

(print register 4 for exit)

make nonphypages bigger to see if it gets the correct value

Final Test

Fork 2 matmult/sort

Exec 2 matmult/sort

How to handle swap file

cache of dirty, evicted memory pages

create a swap file bitmap - 5000 or bigger

check for -1

outside of nachos

create swapfile & open

Dirty Bits

Nachos only sets TLB dirty bits

any time you change TLB, propagate dirty bit to IPT if the valid bit is true

in step 1

disable interrupts

//add if block

if (machine->TLB[currentTLB].valid) {

//copy dirty bit to IPT

machine->TLB[currentTLB]=> \_\_\_\_\_ = ipt[ppn]. \_\_\_\_\_\_;

currentTLB = \_\_\_\_\_\_\_\_\_\_\_\_\_\_

restore interrupts

for network

nachos -m 0 -o 1 //assign machine id 0

nachos -m 1 -o 0 // the other machine

postoffice