

Homework Set 2

Due: Wednesday, 11/12/2008, 5pm

Please work in groups of 3 students

Instructions: Submit to the box outside Gates 310 by the due date above. Show your work, state your assumptions, and justify your answers to receive full credit; we prefer concise, correct explanations. Check the Bulletin Board for corrections and updates.

Homework Policies: Collaboration on homework assignments is encouraged subject to the Honor Code and the following guidelines:

No more than three (3) people may collaborate on a homework solution.

- A group should submit a single solution.
- Any assistance received should be acknowledged in writing on the assignment.
- All students should work on all problems.

SCPD students will receive graded homeworks back through SCPD. There will be no extensions or exceptions to the submission deadline. Solutions to the HW will be posted on the class webpage a few hours after the submission deadline.

Problem 1 (20 points)—Disks and DMA

Assume the following about a machine. All cycles referenced in this problem are processor cycles. Words are 32B.

Loads take 2 cycles if they hit in cache, stores take 2 cycles if they hit in cache, and all other instructions take an average of one cycle. For our workload, 15% of all instructions are loads and 5% are stores.

The machine has one level of cache, with a choice of two policies. For both policies, each cache block is 4 words, and the whole block is fetched on any miss (read or write).

Write-through. A cache miss takes 23 cycles. A write buffer hides some of the latency of write through, so that we stall for an average of 6 cycles on a write-through (whether it's a hit or a miss). On a write-through, we just write a single word to memory rather than the entire cache block.

Write-back. A cache miss takes 23 cycles for a clean block and 36 cycles for a dirty block. On a miss, the block to be replaced is dirty 30% of the time.

The processor/caches, memory, and disk are all linked by a single bus that can transfer a maximum of 1 word per clock cycle. Assume that we don't have bus contention until the bus is 80% utilized, at which point we suffer severe performance penalties.

Part A.

Assume that the cache miss rate is 5%. On average, what percentage of the bus is used for processor-memory traffic for each cache write policy?

Part B.

Continue to assume a 5% cache miss rate. If we don't want the bus utilization to go over 80%, how much bus bandwidth (in B/cycle) is available for I/O for each cache write policy?

Part C.

Assume that DMA I/O can take place simultaneously with CPU cache hits. Also assume that the OS can guarantee that there will be no stale-data problem in the cache due to I/O.

Start with the same assumptions as in Part A, and continue to assume that we cap the bus utilization at 80%. Assume that a 1 KB disk sector takes 1000 clock cycles to initiate a read, 100,000 clock cycles to find the data on disk, and then transfers the data to memory via DMA at the bus rate of 1 word/cycle. With a single disk, how many (complete) disk reads can occur per million instructions executed for each write policy? How would this change if the cache miss rate were 3% instead of 5%?

Part D.

Start with the same assumptions as in Part C. Now you can have arbitrarily many disks. What is the maximum number of sector reads that can occur per million instructions executed, assuming ideal scheduling?

Problem 2 (20 points) –Virtual Memory and Page Tables

You are designing a system that uses virtual memory. Assume that you have a 36-bit, byte-addressable virtual address space and a 32-bit, byte-addressable physical address space. The page size is 4 KB.

Part A.

Page table entries (PTEs) contain the following metadata:

- A valid bit
- A dirty bit
- Support for 2 protection modes
- A use bit that is periodically cleared

What is the minimum size of a PTE? What is the size (in both bits/bytes and pages) of a process's page table, assuming a single-level, non-inverted page table?

Part B.

Assume that the size of a PTE is 32 bits.

Dismayed by the per-process page table size you computed in Part A, you decide to build a hierarchical page table to reduce the number of pages that need to be contiguously mapped.

Your hierarchical page table will break the virtual page number into several fields, which we'll number 1 through N. Field 1 will index into the base page table, which will point to a page in the Level 2 page table. Field 2 will index into this page of the secondary page table, which will yield the address of a page in the 3rd page table...until finally the Nth page table will yield the address of the PTE. Assume that entries in all of these page tables are also 32 bits.

The goals of your hierarchical design are twofold, and are sometimes in conflict with each other:

- Minimize the number of levels of the page table (in order to minimize the number of memory references in a translation).
- Minimize the number of pages that need to be mapped to memory at any given time, assuming that the average case is closer to the best case than the worst case. In particular, the Level 1 page table should be a very small fraction of physical memory (less than 1%).

Design a hierarchical page table that meets these goals, and answer the following questions about your design:

- How many levels of page table are there?
- How many pages are in each level?
- How is the virtual page number divided up (which bits correspond to which page tables)?
- Why did you choose this design?

Problem 3 (15 points) —Virtualization

Part (1)

Repeat the following steps three (3) times; choose a different problem and solution each time.

- a) State and describe an important problem that virtual machines attempt to solve.
- b) Give a specific example of how virtualization helps.
- c) Say whether the problem has been solved and what complications have been introduced.

Part (2)

In 1974, Popek and Goldberg introduced a theorem that stated the necessary conditions for a computer architecture to support virtualization. Virtualization is possible if there are at least two execution modes (user and kernel) and that all sensitive instructions (those that change or are affected by HW configuration) are privileged (that is, if executed in user mode they trap in kernel mode). Their definition of classical virtualization includes the notion of equivalent execution or fidelity, whereby a program running in a virtual environment runs “identically to running natively, barring differences in resource availability and timing.”

Stating clearly any relevant assumptions you are making about the virtual machine, give pseudocode for a program whose timing behavior would be different when run under virtualization versus directly on the hardware.

Do you think it is possible to code a virtual machine such that a program like the one above does not exist? That is, could any (useful) virtual machine provide exact fidelity? Explain your reasoning.

Part (3)

A group of computer scientists and systems architects are gathered in a room listening to a presentation on a new OS security project, based on the idea of intercepting and sanitizing all sensitive and privileged operations. One of the high-paid external consultants shakes her head and admonishes, “Everyone who has ever built an application around the idea of intercepting system calls has lived to regret it.”

How would you defend the consultant, if you were so inclined? Despite your compelling argument, why has VMware (and others) met with such success?

Problem 4 (15 points) — TLBs

A processor’s TLB takes 2.2 ns to translate an address. The tag array of the cache takes 2.5 ns to access, the hit/miss logic takes 1.0 ns, the data array has an access time of 3.4 ns, and it takes 0.5 ns to return data to the processor if a hit occurs. Assuming the TLB hits, what is the cache hit latency if the cache is:

- Physically indexed, physically tagged?
- Virtually indexed, physically tagged?
- Virtually indexed, virtually tagged?

Some systems handle TLB misses in software, as an exception, while others handle it in hardware.

What are the tradeoffs between these two methods?

Will TLB miss handling in software always be slower? Explain.

Are there page table structures that would be difficult to handle in software but easy in hardware? Vice versa?

Why are TLB miss rates for floating point programs higher than those of integer programs?

Explain concisely a couple of techniques that would allow you to reduce the size of your TLB (by number of entries) without increasing the TLB miss rate.

Problem 5 (15 points)—Parallel Applications

In general, parallel applications are lucky to approach near-linear speedup from parallelization. In other words, the best-case scenario from going from 1 to 4 processors is usually a 4 x speedup. For some applications and architectures, however, we can actually achieve a speedup that is better than linear.

Part (a)

Consider a depth-first search of a tree. If you are unfamiliar with this algorithm, this animation may help:

http://www.rci.rutgers.edu/~cfs/472_html/AI_SEARCH/SearchAnimations.html

Assume that traversing each edge of the tree (i.e. going from one node to the next) takes one unit of time, and that time is solely a function of the edges traversed.

For some instances of a tree and a value to search for, parallelizing this algorithm can actually lead to a superlinear speedup.

Describe what inputs can cause this to happen.

Provide one example of a tree and a value to search for that leads to this result. Compute both the sequential time and the superlinear speedup from parallelization.

Part (b)

Describe two more situations in which parallelization can achieve superlinear speedup.

Problem 6 (15 points) — Cluster Design

Section 6.7 of the textbook describes the design of the Internet Archive cluster, and the example on pages 395-396 describes how to calculate the IOPS of an 80 TB rack. Read the section, and answer the following questions.

Part A.

Currently, the disks are the bottleneck in this system. Therefore, we decide to replace them with 15,000 rpm Ultra320 SCSI disks. The good news is that each of these new disks has an average seek time of just 3.5 ms, and can transfer data from the disk at a rate of 135MB/s. Also, the latency of the SCSI controller and the link are now negligible. The bad news is that the disks cost \$1000 each and consume 5 more Watts per disk than the old disks. Also, the capacity of each disk is just 300 GB, so we have to buy 7 of them to get the same total capacity as the 4 PATA disks.

- Calculate the new IOPS and cost/IOPS for a rack for both types of accesses described at the top of page 395.
- Estimate the power consumption of the new configuration (power details are at the bottom of p. 393). Does the new configuration violate the power constraint?

Part B.

A generous benefactor donates the new disks described in Part A to the Internet Archive project. The benefactor now offers some additional equipment:

- 10Gb Ethernet switches, each of which costs \$8000 and 7 additional Watts.
- High-end dual-core mobile processors, each of which adds \$200 to the cost of the original configuration. Due to improvements in architecture and compiler technology, the OS now uses just 40,000 instructions for a disk I/O, and the network protocol stacks need just 85,000 instructions per data block. Each processor can be working on its own I/O simultaneously, doubling throughput.

Do you accept both donations, one, or neither? You are trying to maximize IOPS while fitting in the power budget. If two configurations have equal performance, select the one with lower power consumption.

What is the cost per IOPS for the new configuration for the 16KB accesses?