CpSc-418 **Midterm** Oct. 22, 2002

HINT: Brief answers are good. Show your work for calculations. Sentence form answers should be short and to the point.

1. (3 points) What is your name?
   ANS: Mark Greenstreet

2. (20 points) **Synopsis:** *My Cache or Yours...*

   (a) (5 points) What problem does the paper address?
   ANS: The loss of performance that occurs when disk array and client caches hold the same blocks, reducing the effectiveness of the cache in the disk array.

   (b) (5 points) What are the key observations that lead to the solution presented in the paper? **ANS:**

   - Redundancy occurs because both caches load the same blocks on a miss and use similar replacement strategies (in a traditional configuration).
   - The DRAM for disk caches is expensive, especially for large caches, this motivates using it effectively.
   - Networks provide enough bandwidth that clients can send data back to the disk array when a block is evicted, even if it's clean.

   (c) (5 points) What is the solution presented in the paper?
   ANS: The authors add a *DEMOTE* operation to the disk array. When the client evicts a block, it sends the data back to the disk array. They explore various replacement strategies that the disk array can use for its cache.

   (d) (5 points) How is the solution validated? **ANS:**

   - First, they analyse the performance (math formulas) for simple distributions of block accesses.
   - Second, they simulated their design with synthetic workloads (uniform, Zipf, etc.).
   - Third, they simulated their design with traces of real workloads, both for single-client and multi-client applications.

3. (20 points) **Compare and Contrast:**
   Compare and contrast the DEMOTE policy described in the "My Cache or Yours..." paper with cache enhancements proposed by Norman Jouppi in "Improving Direct-Mapped Cache Performance by the Addition of Small Fully-Associateve Cache and Prefetch Buffers."

(a) (10 points) What are two similarities in the approaches proposed? What common factors motivate these similarities? **ANS:**

1: The victim cache and the cache in the disk array both cache items recently evicted from a higher level cache (L1 or client).
In both cases, this is motivated by trying to make better use of cache resources and trying to cache items likely to be accessed in the near future.

2: The victim cache and the cache in the disk array are both fully associative.
In both cases, this is motivated by trying to minimize conflict misses.

3: Both schemes promote exclusivity over inclusivity.
are both fully associative.
In both cases, this is to reduce the number of cache misses. In both cases, other mechanisms are used to address coherency and consistency issues.

4: In both schemes, the lower level cache (victim or the cache in the disk array) is not much larger than the cache that it backs up (L1 or the client's cache). In traditional cache hierarchies, lower levels of the hierarchy have much greater capacities than higher levels.
In both cases, the choices include resource costs. For the disk array cache, the cost is the price of DRAM. For the victim cache, the cost is silicon area and the access time penalty for large caches..

(b) (10 points) What are two differences in the approaches proposed? What factors motivate to these differences? **ANS:**

1: The victim cache is intended to reduce the cost of conflict misses. The *DEMOTE* policy is intended to reduce the cost of capacity misses.
This is because disk arrays are fully associative already and conflict misses aren't an issue.

2: The victim cache is implemented in hardware whereas the *DEMOTE* policy is implemented in software.
This is because an L1 cache miss should only take a few cycles to satisfy – dedicated hardware is required for miss processing, executing a large number of instructions would be too slow. For a disk cache, disk accesses are *very* expensive, and the overhead of software cache management is offset by the lower miss rate achieved by better caching strategies.

3: Jouppi's paper described prefetch buffers whereas the Wang and Wilkes paper says nothing about prefetching.
This is because disk arrays already prefetch. Prefetching nothing new, and it's not the topic of their research.

4: The DEMOTE operation can be aborted whereas evicted lines are always moved into the victim cache.
This is because the SAN bandwidth is a valuable resource and unnecessary block moves should be avoided. When handling an L1 cache miss, the access time for the L2 cache

or main memory is large enough that there is plenty of time to move the victim line to the victim cache before the new data arrives.

4. (30 points) **Slow Down**
The RAID paper included a factor S that reflected the property that when N disks do one operation each in parallel the operation completes when the *last* disk finishes. Accordingly, the average time for all N to finish is greater than the average time for one disk to complete. The factor S is this ratio. This question explores S.

   (a) (10 points) Assume that the time for a disk to complete a read or write is uniformally distributed between 0.2 ms and 10.2 ms. Assume that if N disks perform N reads or writes in parallel, their times to perform these operations are independent random variables.
   Give S as a function of N. What is S for $N = 2$? What is S for $N = 10$?

   ANS: The distribution function for one disk to complete an operation is

$$F_1(a) = \begin{cases} 0, & \text{if } a < u \\ \frac{a-u}{v-u}, & \text{if } u \le a < v \\ 1, & \text{if } v \le a \end{cases}$$

   and the distribution for the last of N disks to complete an operation is

$$F_N(a) = {F_1}^N(a)$$

   The expected time at which the last of N disks completes its operation is

$$\begin{aligned} E_N &= \int_{u=0}^{\infty} 1 - F_N(a)\,da \\ &= \left(\int_0^u 1\,da\right) + \left(\int_u^v 1 - \left(\frac{a-u}{v-u}\right)^N da\right) + \left(\int_v^\infty 0\,da\right) \\ &= u + (v-u) - (v-u)^{-N}\int_0^{v-u} a^N da \\ &= v - \frac{v-u}{N+1} \\ &= \frac{u+Nv}{N+1} \end{aligned}$$

   From the paper, S is the time it takes to complete an operation on N disks divided by the time it takes for one disk:

$$\begin{aligned} S_N &= E_N/E_1 \\ &= \frac{(u+Nv)/(N+1)}{(u+v)/2} \\ &= \frac{2}{N+1}\frac{u+Nv}{u+v} \end{aligned}$$

For this problem, $u = 0.2\text{ms}$ and $v = 10.2\text{ms}$. This yields:

$$\begin{aligned} S_2 &= 1.32 \\ S_{10} &= 1.79 \end{aligned}$$

Note that S is between 0 and 1 as stated in the RAID paper.

(b) (10 points) Now add a cache to the disk controller (in the drive). Assume that 90% of all reads and writes are handled by the cache. When an operation hits in the disk's cache, it takes time 0.2 ms (the transfer time). When an operation misses, it takes time that is uniformally distributed between 0.2 and 10.2 ms. As above, assume that if N disks perform N reads or writes in parallel, their times to perform these operations are independent random variables.

Give S as a function of N. What is S for $N = 2$? What is S FOR $N = 10$?

ANS: Note that the time for an access with a cache hit is $a = 0.2\text{ms}$. Let $p = 0.9$ be the probability of a cache hit, and $q = 1 - p = 0.1$ be the probability of a cache miss. Let $F'_N(a)$ be distribution function for an access to a disk with a cache:

$$\begin{aligned} F'_1(a) &= \begin{cases} 0, & \text{if } a < u \\ p + qF_1(a) & \text{if } a \geq u \end{cases} \\ F'_N(a) &= (F'_1(a))^N \end{aligned}$$

Let E'_N be the time expected time until the last of N operations completes with caching:

$$\begin{aligned} E'_N &= \int_0^{infty} 1 - F'_N(a)\, da \\ &= u + \int_u^v 1 - \left(p + q\frac{a-u}{v-u}\right)^N da \\ &= u + (v-u) - \int_0^{v-u} \left(p + q\frac{a}{v-u}\right)^N da \\ &= v - \frac{1}{N+1}\frac{v-u}{q}\left(p + \frac{qa}{v-u}\right)^{N+1}\Bigg|_{a=0}^{a=v-u} \\ &= v - \frac{1}{N+1}\frac{v-u}{q}\left((p+q)^{N+1} - p^{N+1}\right) \\ &= v - \frac{v-u}{N+1}\frac{1-p^{N+1}}{1-p} \end{aligned}$$

From which,

$$\begin{aligned} E'_1 &= 0.70\text{ms} \\ E'_2 &= 1.17\text{ms} \\ E'_{10} &= 3.96\text{ms} \end{aligned}$$

By definition, $S'_N = E'_N / E_1$, which yields:

$$\begin{aligned} S'_1 &= 1.67 \\ S'_{10} &= 5.66 \end{aligned}$$

Note that with a disk cache, the slow down caused by waiting for the last disk to finish can be *much* larger than without a disk cache.

(c) (10 points) Assume for a "typical" workload, 30% of the time is spent waiting for disk reads and writes of which half is for reads and half is for writes.

What speed-ups are achieved by adding a disk-cache for a storage array with no redundancy ("just-a-bunch-of-disks" – too bad if one fails), RAID 1, and and RAID 5?

ANS:

No redundancy: The overall speed-up can be calculated using Amdahl's law:

$$\begin{aligned} \textit{speed-up}_{\text{no-RAID}} &= \frac{1}{(0.3E'_1/E_1) + 0.7} \\ &= 1.35 \end{aligned}$$

RAID 1: Reads can be done in parallel and take the minimum of the times for the two disks. Writes take the maximum of the times. The expected time for the minimum of two uniformly distributed random variables is the lower bound plus $1/3$ the width of the interval of the distribution. The expected time for the minimum of two uniformly distributed random variables is the lower bound plus $2/3$ the width of the interval of the distribution. Half the requests are reads and half are writes. Thus, the expected time for a disk operation is just the same as for the no-redundancy case (the fast read and slow write cancel). The speed-up is 1.35. This cancellation turned out to be a nice trick that I hadn't expected before I solved the problem. I'll also accept an answer that computes the slow-down for writes and ignores the speed-up for reads.

RAID 5: Reads take the same time as for no-redundancy, but writes take the time computed above. Using the E_{10} number, I get

$$\begin{aligned} \textit{speed-up}_{\text{RAID-5}} &= \frac{1}{(0.15E'_1/E_1) + 0.15(E'_{10}/E_1) + 0.7} \\ &= 1.20 \end{aligned}$$

5. (27 points) **Pot Pourri**

For each of the terms below, write one or two sentences describing what problem it addresses (3 points), one or two sentences describing how it solves it (3 points), and name the paper where it is described (3 points). To name the paper, you can just give the first few words of the title, enough to make your reference unambiguous.

(a) virtual channel

ANS:

Problem addressed: Virtual channels solve the problem of deadlock in network routing.

How it's solved: Multiple virtual channels are associated with individual physical channels. This creates multiple, acyclic virtual networks running on the physical network with cycles.

Traffic is routed and scheduled within these acyclic networks. As each virtual network is deadlock free, so is the combined network.

Which paper: “The Alpha 21364 Network Architecture.”.

(b) iSLIP

ANS:

Problem addressed: How to map input ports to output ports quickly in a way that moves a large number of packets in each scheduling round.

How it’s solved: A “dance floor” algorithm. The following steps are repeated until no more matches are made: (1) Each input offers traffic to every output for which it has waiting packets. (2) Each output selects one input from which it has traffic offered and sends a confirmation. (3) Each input selects one output from which it received a confirmation. (4) Unpaired inputs and outputs continue in the next iteration.

Which paper: “Fast Switched Backplane for a Gigabit Switched Router”. The iSLIP algorithm is also mentioned in “Efficient Randomized Algorithms for Input-Queued Switch Scheduling”. I’ll accept either one for full credit, but I expect that most people will give the first.

(c) ghost caching

ANS:

Problem addressed: How to choose between several different file caching policies.

How it’s solved: The cache metadata is maintained for several different policies. Because the metadata is much smaller than a block, this requires little extra memory. Based on the statistics for the various policies, a policy that works well with the recent workload is used to decide what blocks to keep in the cache and which to remove.

Which paper: “My cache or yours? Making storage more exclusive.” “The Alpha 21364 Network Architecture.”.

6. (5 points) **Extra Credit**

What UBC faculty member is cited twice in the “My Cache or Yours” paper? Given the professor’s name, and the reference numbers for the citations.

ANS: Mike Feeley. References 13 and 42.