CSE 121: Operating Systems - Architecture and Implementation
Homework 2
Winter 2007

Due: Tuesday, February 20th, in class

1 Rio File System Cache

a) Soft Update's performance degrades markedly when the file cache fills up. In particular, its meta-data updates cannot always be performed asynchronously. Given an infinite-sized file cache (i.e., all files could fit in memory), where meta-data updates will always be guaranteed to performed asynchronously, would there be any motivation for using a scheme like the Rio File Cache? Explain why or why not.

b) While Rio is easy to explain, demonstrating its worth turned out to be quite a challenge. The authors conducted a substantial evaluation consisting of many thousands of bug injections in order to create realistic crashes. The key issue was which of these crashes lead to disk corruption. What was the difference between direct and indirect corruption? Why were both checksums and memtest needed? In other words, why not use just one method to catch both types of corruption?

2 Global Memory Management

a) What is the purpose of the *MinAge* parameter in *GMS*? What could happen if it was calculated incorrectly?

b) Microprocessors have reference bits to help Operating Systems approximate LRU. Can this information provide enough information to help decide a global LRU? If yes, explain the how (i.e. how the *initiator node* uses this information) or why it cannot.

3 Application-controlled File Caching and Prefetching

a) The Application-controlled File Caching and Prefetching paper describes four necessary conditions for optimal prefetching. Show, by constructing an example of file access stream, that an algorithm is not optimal if it doesn t satisfy rule 3. (State your assumptions clearly about cache size and the amount of time it takes to read a block)

b) This question will run through an example of the LRU-SP policy high- lighted in the Caching, Prefetching and Disk Scheduling paper. For this question assume that the size of cache is 5 blocks and there are 3 processes running—A, B and C. For readability, a read request is given as a process/block pair. For example, the read request $A12$ means that process A is trying to read block 12. Finally, assume a basic priority scheme where each process gives a

higher priority to a higher numbered block. For example, if the least recently accessed block is $A15$, and A has another, more recently accessed block in cache, $A10$, A will choose to have $A10$ evicted instead of $A15$.

Consider the following string of read requests (order is from left to right i.e., $B25$ is first request): $B25, C21, C05, B05, B13, C07, B05, A21, C05, A13, C11, C31$

Assuming an initially empty cache, for EACH read request, give the the contents of cache as well as any placeholders (and specify the cache blocks each placeholder points to). Start showing cache contents after the first 5 requests (since the first 5 trivially fill up cache) that occurs as a result of servicing that request. For example, the following table gives the cache contents after the 5th read request:

Request	Cache4	Cache3	Cache2	Cache1	Cache0
B13	B25	C21	C05	B05	B13

4 Lottery Scheduling

a) This question will run through a basic example of a lottery scheduler. Assume we start with two processes—A and B—and a base currency of 1000 tickets. 500 Tickets are assigned to process A, and 500 tickets are assigned to process B. A and B both have their own currency of 100 tickets each (which is backed by 500 tickets of base currency for both processes). Assume A has two threads associated with it: $T1$ and $T2$. $T1$ is assigned 25 tickets of A's currency and $T2$ is assigned 75 tickets of A's currency. Process B also has two threads associated with it: $T3$ and $T4$, where $T3$ has 20 of B's tickets and $T4$ is assigned 80 tickets of B's currency. Assume that we are starting a new time quantum τ_i and at the beginning of this time quantum thread $T1$ is current blocked awaiting I/O.

   (a) For time quantum τ_i, for each thread $T_j \in T1, T2, T3, T4$, what is the probability that T_j will be chosen to run?

   (b) Assume that for time quantum τ_i, $T3$ was chosen to run, but $T3$ only ran for half of the quantum. For time quantum τ_{i+1}, $T1$ has become unblocked; for each thread $T_j \in T1, T2, T3, T4$ what is the probability that T_j will be chosen to run?

b) Describe why the lottery scheduler is a well-suited scheduler for the Monte-Carlo algorithm. For contrast, how would one go about mimicking this behavior using a traditional priority-based scheduler?