

# Systems Comprehensive Exam, Spring 2004

Tuesday, January 13, 2004

## 1 Instructions

This is a closed-book, closed-notes exam with a total of 75 points. You may not use any external source for answering these questions. Please direct any questions about this exam to Professor Bridges, who may be reached either in person in his office in 345G Farris, by phone at 277-3032, or by email at `bridges@cs.unm.edu`. From 10:00am to 1:00pm, Professor Bridges will be available only by email or cell phone at 363-8798. Turn your exam in to Professor Bridges or the front office by 5:00 PM MST on Tuesday, January 13, 2004. Exams *will not* be accepted after this time except by prior arrangement with Professor Bridges.

Type or write your answers to the stated number of questions in each of the following three sections. Make any *reasonable* assumptions necessary to answer the question, but be sure to state any assumptions that you make.

## 2 Short Answer - Answer 3 of 4 (30 points)

*Briefly* answer 3 of the following 4 questions. Your answer should be no longer than *one* paragraph.

1. What is the alias problem in virtually indexed and tagged caches?
2. In a paged virtual memory system, what are the tradeoffs of choosing larger or smaller page sizes?
3. Shortest-job-first is a scheduling discipline that always runs the job with the least amount of compute time remaining. What are the advantages and disadvantages of this scheduling discipline?
4. Compare and contrast the access guarantees provided by token-ring networks and ethernet networks.

## 3 Medium Answer - Answer 2 of 3 (20 points)

Provided detailed answers to two of the following three questions. Be sure to state any assumptions you make and to fully justify your answers.

1. Consider an operating system's network stack that processes headers on network packets prior to transmitting them to the network or delivering them to the application (e.g., protocol stacks in UNIX or the *x*-kernel.) Such network stacks can process packets either in their entirety, with each packet traversing the entire set of layers and being delivered to the application before the next packet is processed, or in batches with each layer processing several packets at a time. What tradeoff is involved in this choice? Be sure to consider the cache performance of each approach.
2. Describe and illustrate the page table manipulations for sharing of data between two machines in a distributed shared memory system and the state changes necessary when each machine first reads from a shared page and later writes to a shared page.
3. (a) What is the difference between normal data and metadata in a file system?  
(b) In what ways do these differences impact file system design?

## 4 Design - Answer 1 of 2 (25 points)

2

Provide a *detailed* answer to one of the following two questions. Be sure to state any assumptions you make and to fully justify your answer.

### 4.1 Power Management

Modern laptops frequently spin down (stop rotation) their hard disks when they have been idle in an effort to conserve battery power. Because spinning up (start rotating again) these hard disks is a potentially expensive operation, the operating system may want to delay writes to disk. Consider two different approaches to delaying having to spin up the hard disk in such a system:

1. A method that is transparent to the application
2. A method that exposes the state of the hard disk to the application, allowing the application to participate in the decision of when to spin up the hard disk

For both of these options, describe a potential implementation, how it would work, and its potential advantages and shortcomings. Be sure to address which applications are likely to benefit most from each of these approaches.

### 4.2 Recovery in Message-passing Systems

#### 4.2.1 Background

Large-scale, long-running scientific computations<sup>1</sup> generally use some form of recovery to tolerate crashes, as failures happen relatively often on large-scale machines. *Checkpointing* is by far the most common form of recovery in these systems; in this approach, a copy of the state of the entire system (a checkpoint) is periodically written to global stable storage. In the case of a crash, computation can be restarted from the most recent checkpoint. Unfortunately, checkpointing is generally costly due to the amount of data that must be saved to shared storage at each checkpoint. As a result, a variety of techniques for reducing the cost and frequency for checkpointing have been proposed. In this question, you are asked to evaluate the tradeoffs of one such potential optimization.

#### 4.2.2 System Optimization

Assume we have a distributed system where each node is normally diskless but could optionally have non-volatile local storage installed in every node if desirable. We propose optimizing checkpointing performance by logging received messages to this optional local storage; after a failure (a failure entails loss of all *volatile* state), each machine would recover from the most recent global checkpoint and then use the local log of received messages to *roll forward* from the checkpoint without having to wait for messages from other machines. (This optimization is normally referred to as *message logging*.)

Evaluate this proposed optimization, making sure to address the following issues:

1. What restrictions on application behavior are necessary with this approach? Are these restrictions fundamental, or could they be worked around with modifications to the proposed optimization scheme?
2. As it involves changes to both the hardware configuration and the system software runtime, the proposed optimization affects application performance in a variety of ways. Describe in detail the potential costs and benefits of this optimization in terms of its effect on expected total application runtime.
3. Given your answers to the previous questions, under what circumstances is this optimization worth doing? For a given hardware system and application, what measurements would you suggest taking to determine whether or not to perform this optimization?

---

<sup>1</sup>Applications spanning hundreds or thousands of nodes and running for days or weeks