CS PhD Qualification (Systems Area)
Nov. 2, 2006

Student No.

There are 9 questions. You are required to answer six of them. Good luck!

Question 1. Shared Memory Multiprocessors
This question is based on the material in the paper by Mellor-Crummey, J. M. and Scott, M., "Algorithms for Scalable Synchronization on Shared-Memory Multiprocessors ", ACM Transactions on Computer Systems, Feb. 1991.

(a) The paper claims that the MCS barrier algorithm may be the best for large-scale cache coherent multiprocessors. Argue why this may not be necessarily true. Be as quantitative as possible to illustrate your point.

(b) MCS paper shows results that MCS barrier does better than tournament on a bus-based shared memory machine like the Sequent Symmetry. Explain why this is so. Be as quantitative as possible to explain your answer.

Question 2. Network File Systems
This question is based on the material in the paper by Anderson, T. et al., " Serverless Network File System ", ACM Transaction on Computer Systems, February 1996.

(a) xFS builds on three prior technologies: RAID, Zebra LFS, and multiprocessor cache consistency. Discuss the limitations in each of these technologies with respect to realizing a scalable serverless NFS.

(b) In a centralized file system, the server performs the functions of managing the data blocks, metadata for the files, server-side file cache, and consistency of datablocks of files cached by multiple clients. Discuss how these functions are carried out in xFS.

(c) This paper was written in 1996. To what extent do you think the premise of the paper and the resultant design relevant today?

Question 3. Parallel discrete event simulation
Several algorithms have been proposed in the parallel discrete event simulation literature to relax message ordering in order to improve performance. One approach is to use time intervals rather than precise time stamps on events, to indicate the event could happen any time within the specified interval.

(a) Describe precisely the partial ordering that would apply using only time intervals to order events.

(b) Modify the Time Warp algorithm to work with time intervals, and specify the local control algorithm that would be used.

(c) Define global virtual time using time intervals, and give an algorithm for computing its value.

Question 4. Replication and Fault-Tolerance in Distributed Systems

There are many distributed state sharing schemes ranging from replicated file systems to distributed shared memory systems. In these systems, the focus is on performance of the system for read and write operations. Consider the following problem that would arise when we are concerned about when and how often certain state is accessed by various nodes of the distributed system. Assume that in addition to reading and writing common state, the users are also allowed to query when and on what nodes certain objects were read or written. The returned information may be per node count or more detailed information including timestamps. This information may be used to detect anomalous access to objects. We want any node to be able to access this information even if some other nodes fail. You are to develop an algorithm for making such information available to certain users under the following conditions:

(a) Node failures are not considered and but consistent results (correct number of reads and writes) must be provided.

(b) Nodes may experience crash failures and results returned may be somewhat stale or may only reflect operations that executed at non-faulty nodes.

(c) Nodes can experience Byzantine failures. In this case, returned results must be correct with respect to all reads and writes that happen at non-faulty nodes.

You may have to limit the number of failures in cases (2) and (3). If you do, specify the maximum number of failures that can be tolerated. If a solution cannot be developed in the presence of certain kind of failures, you need to explain why that is the case.

Question 5. Distributed File Systems

Network file systems such as the Andrew File System (AFS) and the World Wide Web (WWW) both provide users access to remote files, but the two systems have very different user interfaces. In network file systems, the user only needs to mount the remote file system onto the local machine. Then he or she can access remote files just as if they were local. In WWW, a "browser" sends an "address" (a URL) of the file to a Web Server and displays the result to the user.

(a). List the major differences between the two systems' services. In particular, point out their differences with respect to

- granularity of file accesses;
- semantics of the caching of remote files on local disks; and
- handling concurrent reads and writes of a file.

(b) Since the two systems provide different services to the user, naturally, their implementations are different. (i) Briefly outline the main features of the AFS file system implementation. (ii) Suggest an algorithm for implementing the WWW. You do *not* have to describe how a browser implements the display of the Web documents, but you should suggest a framework for maintaining a cache of recently-accessed documents to decrease network traffic and for making sure the documents in the cache are up-to-date. If you happen to know the WWW actually does it, you can describe the algorithm. Otherwise, you can make up a plausible algorithm of your own.

(c) If AFS were universally available, how would that simplify the task of implementing the WWW?

Question 6. OS and System Configurability

Another relatively recent set of research addresses operating system kernels/structuring with respect to system configurability.

(a) Why is this a new topic, given that people in software engineering have been worrying about extensible software, etc. for quite some time now? In other words, what different assumptions are being made in newer work like SPIN, etc., compared to what is being offered in existing commercial operating systems?

(b) What specific research questions would you argue need to be addressed with such new work on configurable operating systems? List at least 3 topics and argue `why' for each. Be as specific as possible, including referring to particular operating system components. Do not list c) below.

(c) One common theme of such recent work is 'protection'. Describe the protection problem being addressed by systems like SPIN. Discuss what approach, if any, is used by rival systems like Exokernel. Show how these protection problems differ, if any, from the protection problems addressed in the early 80's (e.g., in HYDRA).

First, discuss the practical impact of OS kernel configurability results during the last ten years. Make the discussion concrete by outlining or comparing the support for configuration in a production use kernel such as VxWorks, Embedded Linux, and Windows Embedded family. Second, two of the major technical challenges for building these configuration tools are: (1) a careful componentization of the kernel, combined with (2) the dependency management among the components. Outline the techniques used in

componentization and dependency management that made these configuration tools feasible and give a representative citation to each of these two broad areas.

Question 7. Real-time scheduling
A dynamic priority scheduler such as earliest deadline first (EDF) and minimum laxity first (MLF) is considered optimal since it is able to maintain feasible schedules up to 100% of CPU utilization. In contrast, a fixed priority dynamic scheduler such as rate monotonic has worst case achievable utilization limits (0.693 in a theoretical analysis and about 0.9 in an average case analysis) that are less than 100%.

An example schedule that can be scheduled by EDF or MLF (with all jobs completing before deadline), but cannot be scheduled by rate monotonic (with some jobs missing their deadlines) is the following: job-1 with period of 50ms and CPU requirement of 25ms, and job-2 with period of 75ms and CPU requirement of 30ms.

It has been shown that the above problem of rate monotonic disappears (worst case utilization becomes 1) when all pairs of periods are in harmonic relations. It also has been shown that appropriately extending the deadlines of jobs for rate monotonic achieves the same goal.

Explain intuitively the reasons why these special conditions make rate monotonic optimal by eliminating the inefficiency described in the above example.

Question 8. Autonomic Systems
Performance monitoring and understanding are key elements of ongoing work ins self-managing (or autonomic) systems. A key issue here is the information being monitored and thus, the conclusions one can draw about application and system behaviors from such information. Assuming a typical three-tier application (backend storage services, mid-tier application services, front end web server):

(a) Identify application behaviors that cannot be captured or understood with purely middleware-level monitoring, i.e., with monitoring methods built into the middleware (e.g., J2EE middleware) that is used by such applications.

(b) Identify behaviors that cannot be understood with purely system-level methods (i.e., kernel-level monitoring), but could be understood by middleware-level techniques.

(c) Identify behaviors that cannot be understood with either a. or b., but instead, require application-level instrumentation and/or information about application semantics.

(d) Briefly (!) sketch the different elements of a future, sufficiently rich monitoring infrastructure able to capture a.-c.

Question 9. Virtualization

Virtualization coupled with multi-core machines is leading to new opportunities and challenges in systems research. One opportunity is that there will likely be substantial additional compute cycles that can be used to enrich the way in which operating systems and/or applications function. Specific cases in point include communications enriched with online methods for XML interpretation, real-time encryption of off-chip or off-machine content, and others. Drawing on some of the many examples you have seen in your readings, e.g., capability-based architectures for protecting data access, collective communications, ...), in this question, you are asked to:

(a) Describe or come up with some innovative system-level (i.e., operating system or device-level) functionality that would benefit some class of applications (which you identify/select).

(b) Describe at least two alternative software realizations of this functionality on a future multi-core, virtualized computing platform. Please describe 'where' your software would operate (in hypervisor, with devices, in OS kernel, ...).

(c) Discuss the performance/cost tradeoffs for these realizations, including a discussion defending the viability of your ideas.