Systems Ph.D. Qualifying Exam – Spring 2007 (March 15, 2007)

NOTE: PLEASE ATTEMPT 6 OUT OF THE 8 QUESTIONS GIVEN BELOW. (NOTE THAT 3 AND 4 SHOULD BE ATTEMPTED TOGETHER, WHILE 6 AND 7 MAY BE ATTEMPTED AS INDIVIDUAL QUESTIONS)

Q1. OPERATING SYSTEMS: MEMORY MANAGEMENT

Consider a modern shared-memory multiprocessor built using Multicore processors.

1) What are the sources of "unscalibility" of the OS mechanisms for the efficient support of multithreaded and multiprogrammed workloads on such a machine?

There have been several proposals in the literature (such as Tornado, Spring, Solaris MC, and SGI Iris) for efficient construction of OS for multiprocessors using object technology, component reuse, etc.

2) Discuss your design approach for memory management, synchronization, and scheduling using such technologies that avoids the pitfalls of unscalability. Be as specific as you can be in proposing mechanisms as well as data structures and pseudo-code in answering this question.

Q2. OPERATING SYSTEMS: DESIGN ISSUES

We see a recurring theme in the design of system mechanisms, namely, "greediness does not pay". Explain how this is borne out in the design of

1) synchronization algorithms

2) multiprocessor scheduling

3) network protocols such as Ethernet

Q3. CPU SCHEDULING

A new, important domain in server systems is that of online power management. Interestingly, solutions to this topic are being considered in the context of new virtualization technologies (e.g., consider Amazon's Compute Cloud announcement).

This question asks you to develop a new scheduling algorithm that uses power consumption as its principal metric, while also meeting applications performance demands. Specifically, using the idea of affinity scheduling as a basis, construct a 'power

affinity' scheduling algorithm for multicore platforms, making the following assumptions:

- schedule Virtual Machines such that their performance requirements are met (in terms of the `speed of the virtual CPU allocated to them);
- assuming that the most power gains are derived from IDLING a processor (i.e., not running anything on it), use an affinity-scheduling idea to
- consolidate VMs onto a single processor; and extend the notion to dealing with multicore processors and with multiprocessor VMs (i.e., VMs that span multiple processors or cores).

1) Describe the additional information about each VM and about the platform on which the VM is running, which you require for your energy saving CPU scheduling algorithm.

2) Describe your new scheduling algorithm (and the assumptions you are making about the hypervisor's VM scheduler that underlies it).

3) Discuss `where' you would implement this algorithm (not in a VM, of course).

4) Talk about the complexity of your scheduling algorithm.

5) Discuss your approach's limitations (i.e., the assumptions you had to make), then use that discussion to talk about your algorithm's extension to dealing with multiprocessor VMs and multicore systems.

Q4. MIDDLEWARE

Middleware is an increasingly prevalent medium for large-scale program development. The purpose of this question is to explore new middleware for reliable distributed applications. The proposal is to build middleware that offers enhanced reliability guarantees to applications. Building on the controversy between CATOCS software and simpler approaches, you are proposing to resolve this controversy by adding CATOCS-like methods to the implementation of Websphere. However, in contrast to entirely hiding CATOCS under the covers of the middleware'ss message exchanges for remote service invocation, you propose to open up the CATOCS implementation to deal with some of the criticisms of the `pure CATOCS approach.``Opening up involves using interfaces similar to those used by the `message factory implementation of RMI. If you don't know what that entails, for purposes of this question, you can assume it's similar to the notions of subcontracts or policies associated with remote method invocations.

In your answer to this question, define and deal with two of the following four criticisms of CATOCS:

- unrecognized causality (can't say for sure)
- lack of serialization ability (can't say together)

- incidental orderings (can't say whole story), also termed unexpressed semantic ordering constraints
- efficiency - CATOCS vs. state-level techniques

Explain your improved CATOCS facility and its APIs, then explain two of the four above.

Q5. EMBEDDED SYSTEMS

Traditional embedded systems are independent devices. With the evolution of ubiquitous environments, embedded devices have become interconnected through wireless networks. There are some complications with the construction of networked embedded systems. Discuss the following research challenges and one example of techniques useful in building networked embedded systems.

1) Reliability of mobile ad hot networks, where each node may move or fail in semi-independent ways.
2) Power management in devices, both individually and in concert, to prolong the functionality of the network.
3) Management of a large number of small devices, for example, software updates and tracking and replacement of batteries.

Q6. REAL-TIME SCHEDULING

1) Define the following classes of real-time schedulers:
   (a) static schedulers,
   (b) dynamic priority scheduler such as earliest deadline first (EDF) and minimum laxity first (MLF),
   (c) fixed priority dynamic scheduler such as rate monotonic.

2) Compare each pair of scheduler classes, (a) with (b), (b) with (c), and (a) with (c), with respect to their execution overhead, the worst case achievable utilization, and ease of implementation.

Q7. DISTRIBUTED SYSTEMS

This question explores the ins and outs of remote procedure calls, and how they are used to build distributed services.

1) One key component of any RPC system is stub generation. Describe the process of stub generation, and how it eases the process of distributed application development. What types of problems are avoided through the automation of this process?
2) Naming is an important ingredient in any RPC system. First, describe where the issue of naming arises (i.e., discuss when clients and servers would utilize a name service).

Then, describe how a particular RPC system utilizes a naming service (the RPC system as described by Birrel and Nelson, Sun RPC, or Java RMI are all suitable). For this last part of the question, a rough outline of the important steps and interactions is all that is needed.

3) One key question that arises in building a distributed service is how to partition work across the client and the server. First, describe why is partitioning important. Then, present and discuss a general set of guidelines which one could use to partition a service across a client and server.
4) Finally, describe general guidelines that one could use to partition a service over a P2P overlay network.

Q8. TIME WRAP

Consider Jefferson's Time Warp algorithm.

1) Show that the algorithm is guaranteed to make forward progress. Specifically, state precisely what "forward progress" means, and show that Time Warp is guaranteed to make forward progress given your definition. State any assumptions that are necessary to prove this property.
2) Is the Time Warp algorithm guaranteed to terminate for a simulation of finite duration when executed sequentially? Assume the precision on time stamps is infinitely large.