Systems Area Qualifier Written Exam

(2009 Fall)

You are required to choose a minimum of 6 out of the 9 questions in this written exam. Good luck!

**Question 1. Large Scale Distributed System Design**

Camera deployment is becoming ubiquitous. London has a city-wide deployment of on the order of 10’s of thousands of cameras. New York and Chicago are planning a large-scale city wide deployment of cameras. More generally speaking sensors of various kinds are pervasive in our environment and will become more so in the future. Such large-scale networked sensor systems have wide-ranging applications including surveillance, emergency response, traffic monitoring, and so on. This question pertains to the design challenges in such large-scale distributed systems.

a) Enumerate at least three significant system challenges you perceive in building such large-scale distributed systems. You may use one driver application as a vehicle to discuss such challenges and make the discussion more concrete.

b) For each challenge, discuss (with qualitative and/or quantitative reasoning) at what level of the system software stack (application level, middleware level, OS level, etc.) do you envision the need for innovation to address the challenge.

c) Identify solutions from existing literature that may be appropriate for addressing each challenge.

d) Identify the shortcomings (if any) of existing solutions to these challenges and propose ideas for addressing them better.

**Question 2. Transactions and OS**

Quicksilver operating system proposed using transaction as a fundamental building block in constructing the functionalities expected in a distributed operating system. There have been similar attempts in the past such as the Light-weight Reliable Virtual Memory (LRVM) that propose using transaction as a building block of operating system services.

a) Distinguish between transaction as proposed and used in such operating system level services versus full-blown traditional database transactions.

b) Justify with examples of operating system services why such limited transactional semantics are appropriate in such systems.

c) Commercial operating systems such as Vista and Linux do not use transaction as a building block. Discuss why.

d) Discuss how transactions may help in building robust applications and system services.
**Question 3. System Virtualization**

This question concerns virtualization technologies for HPC systems and their applications. The basic issue you should address is the following. Many HPC codes desire full and complete access to hardware, so that they can derive best performance from it. Virtualization, on the other hand, attempts to take hardware and share it fairly across multiple applications. This requires technology to implement fair sharing and performance isolation support, both of which can reduce the absolute performance seen by application codes.

Assume multicore hardware, high end network connections, large memory, accelerators but no storage per HPC system node. First, present at least two examples where with current virtualization technology, the absolute performance seen by an HPC code can be reduced if hardware is virtualized, even if only a single VM is given all hardware resources by the Virtual Machine Monitor (VMM). Second, for a HPC system with 8 core multicore nodes, where two VMs each get 4 cores and `their own' accelerator (so, assume two accelerators per node), identify at least two performance issues that arise from one application competing with another. Third, there is the well-known issue of OS noise in HPC (i.e., an operating system' actions or functions randomly delay an application program, which at scale, can affect parallel program performance -- consider barriers). Provide at least one example of potential hypervisor noise and an approach to solving it.

**Question 4. Replication and Consistency in Distributed Systems**

A well-known issue in distributed systems is that of consistency, where replicated data items are difficult to keep consistent when providing the illusion of shared data. There are many consistency management methods for distributed shared memory. 1) Explain at least two such methods, not including the traditional full consistency used in shared memory multiprocessors and use that explanation to argue why consistency management is expensive and/or difficult to scale. 2) An idea pursued in some work is to consider application-level information for consistency management, such as 'context'. Come up with at least one example of the use of context to simplify consistency management or make it cheaper. Argue a bit more formally how much your context-based method can help (e.g., under what conditions, etc.). 3) To what extent can consistency management deal with real-time application requirements and if such requirements exist, what can be done to address them?

**Question 5. Fault Tolerance in Distributed Systems**

There are various well-known issues with failure handling in distributed systems. First, explain the different types of failures found in such systems. Next, for byzantine failures, many systems try to address them by using replication and then enhancing underlying communication infrastructures with additional libraries. Explain one specific approach
used to deal with byzantine failures and the basic structure of the libraries used to implement that approach. Then (and for the main part of this question) come up with issues and approaches for the use of such libraries in mobile systems, subject to energy limitations and to disconnection. Identify at least two technical issues (one for each topic - energy and disconnection), then outline solution approaches that might work to deal with these issues.

**Question 6. Resource Management**

(Part 1) Consider a system consisting of m resources of the same type, being shared by n processors. Resources can be requested and released by processors only one at a time. Show that the system is dead-lock free if the following two conditions hold:
(a) The maximum need of each process is between 1 and m resources, and
(b) The sum of all maximum need is less than m+n.

(Part 2) In modern multiprocessor systems that have multiple CPUs. Each CPU usually contains a local cache.
(a) Explain the “write-through” cache update method that is needed and its advantages and disadvantages.
(b) Propose an alternative method that is different from “write-through” during the cache updating. Explain the advantages and disadvantages of your “non-write-through” method by comparing it with “write-through”.

**Question 7. Distributed Hash Tables (DHTs)**

(1) Structured peer-to-peer (p2p) networks implement distributed hash table abstraction. Explain what this statement means. Reflect on the role of consistent hashing in enabling p2p systems.

(2) Explain how the full-membership model employed in Amazon’s Dynamo system can help improve the performance of available DHT implementations. You can use any of the widely used DHT’s (Chord, CAN, Pastry) as an example or discuss using generic abstractions.

(3) The usefulness of DHT’s is heavily dependent on uniform distribution of load to the participants. Under what conditions will a DHT become unbalanced? What do you see as some of the major drawbacks of an unbalanced DHT? Discuss ideas for maintaining a well-balanced DHT.

**Instructions/Hints:** State all your assumptions, and provide all references. Original ideas will be given more credit. Please limit your answers to a maximum of one page per part. The Amazon’s Dynamo system is documented in SOSP 07, titled “Dynamo: Amazon's Highly Available Key-value Store”, by Guiseppe DeCandia, Deniz Hastorun, Madan
Question 8. New Real-Time Applications

There are many new real-time applications that use real-time sensor data. One example is real-time traffic monitoring, for example, used by drivers to get around congestions.

(1) [25%] Describe a plausible design, including data flow from sensors to server and to clients, for the Georgia-Navigator system (www.georgia-navigator.com). It is known that they use a vision-based speed monitoring system that captures images of cars from fixed black-and-white TV monitors. The images are transmitted to a centralized server through optical fiber networks. [No more than one page.]

(2) [25%] Describe a plausible design, including data flow from sensors to server and to clients, for a smartphone-based system that collects location information from GPS-equipped smartphones instead of dedicated sensors such as Georgia-Navigator. You may assume a mixed network (e.g., combining cell phone network and optical backbone) and a centralized server. [No more than one page.]

(3) Compare the two systems (specialized sensors in (1) and general purpose sensors (2)) along the following dimensions:

- [20%] Maximum coverage and routine availability of traffic data
- [20%] Reliability of results (e.g., timeliness and accuracy of traffic data displayed)
- [10%] Security and privacy of data collected and its usage

Question 9. Application of Specialization

Specialization is a widely used technique to improve system software in various dimensions. For example, specialization can be used to improve system performance, by using a shorter version of the software for the most common case. Another example is specialization for embedded systems, where a session-specific TCP protocol stack may be many times smaller (in memory footprint) than the generic version. A third example applies specialization to improve software security through artificial diversity.

(1) [30%] Briefly explain the purpose and use of the ASLR (Address Space Layout Randomization) in operating systems (e.g., Linux and Windows Vista). Outline the differences between code generated with and without ASLR enabled. [No more than one page.]

(2) [30%] Choose an example of exploit (e.g., buffer overflow), and explain how it can be prevented by random code generated by ASLR-style artificial diversity. [No more than one page.]

(3) [30%] Discuss and compare the similarities and differences in the application of specialization for (a) reducing code execution time, (b) code footprint in memory, and (c) code diversity such as ASLR. [No more than one page.]
(4) [10%] In addition to execution time, memory footprint, and security, discuss any other dimensions in the improvement of system software through applications of specialization.