# Systems Ph.D. Qualifying Exam

Spring 2011 (March 22, 2011)

**NOTE: PLEASE ATTEMPT 6 OUT OF THE 8 QUESTIONS GIVEN BELOW.**

**Question 1 (Multicore)**

There are now multiple outstanding proposals and prototype systems to re-structure hypervisors or operating systems for multicore platforms. This question asks you to explore some of that design space.

a) Explain some of the competing proposals (at least 2).

b) State in your own words why/why not existing solutions like those already present in Linux and developed for existing shared memory multiprocessors do/do not apply to multicore platforms. Be explicit here, by focusing on two key resources: CPU and memory.

c) From your arguments in b), design your own favored solution for a mythical 1000 core future multicore chip: briefly describe the chip and its main components, then explain and motivate your design. Again focus on CPU and memory resources.

**Question 2 (V**irtualization Technologies)

There is increased use of virtualization technologies in both datacenter and personal systems. Recent work has even suggested their use for mobile devices. A particular challenge for these technologies, however, has been I/O virtualization, and without good solutions in that space, it will be difficult to routinely virtualize I/O-rich mobile platforms. This question asks you to consider specific issues in I/O virtualization, then devise solutions and explain them.

a) Describe I/O virtualization issues for 2 devices, disks and network, specifically addressing performance as well as fairness: what does it take to efficiently virtualize these devices? when/how can fairness be an issue?

b) Describe hardware or software (or both) solutions to the performance and fairness issues you identify in a). However, be sure to focus on mobile devices and their I/O rather than on server systems.

c) Discuss your solutions in terms of past work in server virtualization...where can you/can you not make the same assumptions as those made in server systems?

**Question 3 (**Cloud Computing)

At a recent cloud computing meeting, arguments were made to 'bring back virtual sychrony' in order to make it easier to program and prove correct cloud applications. One recent attempt, made in the PNUTS infrastructure developed at Yahoo, is called 'timeline consistency' and is defined as follows: all replicas of a given record apply all updates to the record in the same order, but this notion is provided only on a per record basis, where applications have to cooperate with the PNUTS infrastructure to properly mark records as timeline-consistent or not.

a) Discuss a use case for timeline consistency (hint: consider multiple data centers).

b) Is timeline consistency stronger/weaker than the virtual synchrony defined in papers like those written by Birman (Horus, ...)?

c) Describe at least one possible implementation of timeline consistency, then also discuss the failure model implications of this implementation.

**Question 4 (Distributed Shared Memory Systems)**

Since the mid-80's, there have been several proposals to mimic "shared memory" in a distributed system via software, even though the nodes do not physically share memory.

(a) (15%) Explain the philosophical reasons behind such proposals.

(b) (25%) Enumerate the correctness and performance problems in implementing such proposals. Your answer should be general and not specific to any particular system.

(c) (60%) Discuss the range of options available to you as a system designer in implementing such a software system. Discuss the pros and cons of each of these options from the point of view of performance. Discuss the pros and cons of each of these options from the point of view of interactions with other software subsystems (e.g., virtual memory, network protocol stack, etc.) as well as hardware. Your answer should be complete with concrete examples of systems that have adopted such an option.

**Question 5** (Sensor Networks and Distributed Systems)

Sensor networks may be considered a special case of a distributed system.

(a) (40%) Explain the similarities and differences of a sensor network to traditional distributed system. Be specific in identifying problems you may encounter in a sensor network that can be couched as a traditional distributed systems problem. Similarly, identify problems you may encounter in a sensor network that does not have an obvious distributed systems counterpart.

(b) (60%) Consider a large-scale multimodal sensor network comprising of cameras and other sensing modalities. The sensor network has access to backend computational infrastructure for doing compute-intensive analyses (e.g., computer vision). The intent is to use such a sensor network for airport surveillance to identify and warn of emerging threats in real-time. Discuss the system issues to be addressed for such a large-scale sensor network including programming abstractions, sensor discovery, efficient resource management, failures, etc. Your answer should be complete in identifying unique problems and solution approaches thereof. You can assume that domain-specific software components (e.g., computer vision) are available to you as black boxes.

**Question 6** (Distributed Systems and Web Search)

Your grandmother accesses Google and types in a query and gets a set of responses.

(a) (25%) Enumerate all the software components that come together behind the scene to help your grandmother's query. Your answer should include not just what happens instantaneously in real-time when the query is typed but what may have been done ahead of time to serve such queries.

(b) (40%) Discuss the roles played by various software subsystems (e.g., GFS, Bigtable, Map-reduce, etc.) to answer your grandmother's query.

(c) (35%) Discuss how scalability of these software systems is addressed to ensure that Google can handle a whole community of grandmothers who may be typing in similar queries simultaneously.

**Question 7 (Embedded Systems)**

Traditional embedded systems are typically isolated devices that provide specific functionality. In contrast, modern embedded systems usually have some kind of network access capability and become part of a larger distributed system. An example of such networked embedded systems is smart phones, which have hardware capabilities comparable to "normal" computers. Some smart phones run specialized OS kernels (e.g., Symbian) while others run specialized versions of "normal" OS's (e.g., Windows Embedded and Android based on Linux).

(a) [15%] Choose a concrete instance of a specialized OS kernel and an instance of a specialized "normal" OS. Compare their kernel call APIs (functionality) in terms of similarities and differences. Use an illustrative comparison by choosing one or two major OS components (e.g., network protocols, file systems, memory management) for a concrete comparison of one or two kernel calls.

(b) [40%] If you compared two OS components in item (1) above, choose one component for this item. Compare the implementation of the OS component of (a) the specialized OS kernel (e.g., Symbian), (b) the specialized version of "normal" OS (e.g., Android OS), and (c) a normal version of the "normal" OS (e.g., a normal release of Linux). The comparison should be concrete (more detailed than

Wikipedia explanations), and illustrative (not exhaustive) using concrete examples to explain similarities and differences (no more than 3 pages).

(c) [25%] For the 3 OS's chosen in item (2) for discussion, explain briefly how battery management is enabled or supported in each OS as an example of their support for embedded systems. Use smart phones as illustrative environments for the specialized OS and specialized version of "normal" OS, and laptop as an illustrative environment for the "normal" OS.

(d) [20%] For 3 OS's chosen in item (2) for discussion, explain first how program specialization techniques can facilitate the portability while maintaining performance when porting applications between the normal version of normal OS (e.g., Linux) and a specialized version (e.g., Android). Then explain the potential difficulties of porting applications using the API of a specialized OS kernel to another OS.

**Question 8 (Autonomic System Management)**

With the continued growth of data centers and cloud computing environments, there is a growing need for autonomic management of large parallel and distributed systems, both at the system level and at the application level.

(a) [25%] Consider highly parallelizable applications such as web search and data analytics, which may use MapReduce to run on hadoop. (a) Explain how such applications may be decomposed and run on a large number of virtualized nodes. (b) Explain what kind of system-level facilities is needed for achieving service level agreements (SLA) on performance (e.g., monitoring) and availability (e.g., recovery) in the Map phase. (c) Choose a current OS kernel (e.g., RedHat Linux) and explain whether/how it supports the above facilities (part 1.b) or is lacking in such support and how an application programmer may achieve SLA at application level.

(b) [25%] Performance scalability: Consider more complex applications such as N-tier applications used in e-commerce (e.g., RUBiS and RUBBoS benchmarks), typically including web servers, application servers, and database servers. When workloads grow, discuss the ease (or difficulties) of scalability of each tier: (a) web servers, (b) application servers, and (c) database servers. [Hint: comparing with MapReduce may be helpful.] Explain the support (or lack of support) of current OS kernels (e.g., Linux) for facilitating dynamic system configuration adaptation due to application scalability requirements. If you are arguing against OS-level adaptation, you should give an answer (what facilities are needed and at which system level) to support your arguments for application-level adaptation.

(c) [25%] Availability: Explain the support (or lack of support) of current OS kernels (e.g., Linux) for facilitating dynamic system configuration adaptation due to application recovery requirements such as business continuity. (a) What system-level facilities would enable seamless recovery for "easily" scalable components such as web server and application server? (b) How about a database server? (c) How about a MapReduce-based application?

(d) [15%] Performance prediction: Explain the difficulties of applying classic queuing theory (e.g., M/M/1) in describing N-tier applications. [Hint: consider the assumptions made in mean value analysis.] Explain one alternative to classic queuing theory that may be able to model N-tier systems.

(e) [10%] Security: Explain the reasons a typical CIO would hesitate to run applications in a public cloud that require proprietary information. What methods would you propose to safeguard the company proprietary information contained in a public cloud? Would the same methods work when safeguarding private information for individuals (e.g., Microsoft HealthVault service)?

--- END ---