

Name:

email:

ID:

Final Exam
CS555
10 May 2011

You have 2 hours for this exam. The exam has 7 pages. There are 100 possible points. Show all your work for partial credit.

Definitions

Each question is worth 1 point. Answer all questions in this section.

1. Define the following

a) Synchronous disk write

Answer: A disk write where the OS blocks the requesting process until completion.

b) Soft state

Answer: State that is continually refreshed to remain valid

c) Replay attack

Answer: An attempt to cause a system to misbehave by replaying old messages.

d) Caching

Answer: Saving calculated responses to avoid recalculating them

e) V-node

Answer: Generalization of i-nodes in the unix file system, used to easily support multiple file systems

f) Microkernel

Answer: A minimal kernel that supplies only fundamental abstractions and requires minimal code to operate at a high cpu privilege.

g) Aliasing (in a namespace)

Answer: When two names refer to the same object (or when one name points to another in the same namespace).

h) Process migration

Answer: Transferring a running thread (and address space) from one system to another. This is system to system, not between cores on one physical multiprocessor.

i) Real time process scheduler

Answer: A deadline driven scheduler uses to give processes a guaranteed amount of run-time.

j) Confinement problem

Answer: The difficulty of preventing undetected passage of secure information from inside an organization out.

Short Answer

Each question gives its value in the question. Answer all questions in this section. This section is worth 50 points.

Name:

email:

ID:

2. When a monitor lock is released and other threads are waiting for it, conventional operating systems generally assign the lock using either a first-come-first-served (FIFO) queue or a priority queue. Explain how one could use Lottery Scheduling to assign the lock without considering thread priorities. (5 pts). Modify this to reflect thread priorities. (2 pts). Contrast your priority system with a pure priority queue. (3 pts). You should assume that only the lock assignment mechanism changes, specifically a conventional priority queue is used for scheduling the processor(s).

Answer:

Without considering thread priorities, each thread is assigned an equal number of tickets for the lock lottery, and when the lock is released, a lottery is held and the winning thread is assigned the lock. When the lock is returned, each waiting thread has an equal chance to get the lock.

One could take priorities into account by assigning each thread tickets proportional to its priority. When the lock is released, higher priority threads have a higher chance to acquire the lock.

Because the lock acquisition is probabilistic, not deterministic as in a traditional priority queue, a high priority process will generally wait longer than under strict priority queueing. As a result, transient priority inversions will last longer, and strict guarantees are more difficult to make.

3. A colleague describes a new microkernel he or she has found. The only console interface the microkernel provides allows the caller to set the color of individual pixels on the screen one at a time. Your colleague claims that the overall system can hide the latency caused by this interface by creating a server that exports an interface to change larger blocks of pixels at once. Do you agree that this will speed up the system (support your answer)? (3 pts) Do you think this console design is an argument against microkernels (support your answer)? (2 pts)

Answer:

The system won't be sped up. The microkernel provides the access to the hardware device, and each pixel change will require a message/system call to the microkernel whether the user or the server makes it. (The server interface may make user code easier to write, however).

This is not an argument against microkernels. Only supporting a single pixel interface is a bad design choice in a microkernel or a monolithic kernel. While the kernel crossings into a monolithic kernel may be cheaper (and that's a big may) a design that puts that many kernel crossings into a common operation like manipulating the screen is going to impose significant overhead independently of the underlying design philosophy.

Name:

email:

ID:

4. Xen uses paravirtualization to create multiple operating system instances on a single machine. Briefly explain paravirtualization (3 pts). Parts of paravirtualization come from the authors working around limitations in the hardware they possessed, but they argue paravirtualization is useful even when the hardware supports other choices. Why do they believe paravirtualization is useful when the hardware does not require it? (2 pts)

Answer:

Paravirtualization aggressively uses the underlying hardware with minimal modification by vetting each operating system's changes to that hardware. Privileged instructions are replaced with hypervisor calls, and during standard operation the paging hardware and CPU operate normally.

The authors argue that because paravirtualization requires minimal run-time overhead and generally operates the hardware in its most efficient way, it provides extremely good performance. Essentially the interface change allows them to minimize run-time overhead.

5. Briefly explain two techniques that the flask access control system uses to improve its performance. (2 pts each) Why are such performance improvements important in Flask? (1 pt)

Answer:

Flask reduces complex labels to integer security identifiers (SIDs) to reduce the overhead of calls to the object managers and security manager. It also enables caching of access control decisions by object managers to avoid later calls to the security manager.

These improvements are important because the Flask microkernel is a microkernel, and any improvements are an advantage, and because the security decisions are inherently more complex than traditional access control lists or group permissions and carried out on each access. This high cost has to be minimized to keep the system usable.

Name:

email:

ID:

6. Briefly describe each of the RAID disk layouts (one line is enough). (5 pts)

Answer:

- Disk mirroring
- Bit-sliced data with an error correction code on the check disks
- Bit sliced data with error isolation provided by hardware (1 check disk)
- Sector sliced data with error isolation provided by hardware (1 check disk)
- Sector sliced data with error isolation provided by hardware and a distributed check disk (1 check disk)

7. In Needham/Schroeder or Kerberos authentication how does an endpoint know that a message is not a replay? (3 pts) How does BAN logic represent that conclusion? (2 pts)

Answer: A message is not a replay if it contains a nonce that the endpoint has not seen before. In BAN logic this moves the message from the past to the present. The predicate used in BAN logic to express a nonce that has not been seen is that **fresh**(*nonce*) is true.

Name:

email:

ID:

8. Define capability. (1 pt) Consider the three capability systems we discussed this semester: SPIN, Hydra, Amoeba. Briefly describe how they represent capabilities and enforce unforgeability: (6 pts). Give an strength of each implementation (3 pts).

Answer:

A capability is a combination of an object reference and the permissions to perform certain operations on it.

The capabilities we studied:

- Spin: capabilities are represented as strong types in Modula-3. Language constructs and dynamic linking are used to prevent forgery. An advantage is that within their closed environment there is no overhead for capability use - the compiler has checked validity.
 - Amoeba: capabilities are represented by cryptographically signed bit strings. Unforgeability is prevented by checking the keyed hash on each capability. An advantage is easy export of these capabilities to outside systems.
 - Hydra: capabilities are represented by kernel data structures. Because applications cannot manipulate them directly or insert them into the kernel, forgery is impossible. These capabilities are easy to amplify and modify and provide richer semantics than most capability systems.
9. What does a Lamport distributed snapshot capture? (3 pts) How is it related to Jefferson's Virtual Time (Time Warp) distributed system? (2 pts)

Answer:

A Lamport snapshot captures a stable property of the system: a property that once it holds about a system, it remains true of that system. It is used in Virtual Time to capture global virtual time (GVT). The fact that GVT has exceeded some value is stable, and Lamport snaps can capture it.

Name:

email:

ID:

Long answer

This section is worth 40 points. Each question gives its value. Do all questions in this section.

10. We discussed several file systems tuned for different environments. For each environment below, name a file system that is well suited for it (1 pt) and describe a feature of that system that supports your choice. (4 pts).

- a) An networked environment that requires all computers to see the same file system – namespace and contents. There is enough data that multiple servers are necessary. The environment covers a campus or small business area.

Answer: Sprite provides such a namespace directly using their prefix tables and a broadcast protocol to locate file servers by a file's prefix.

- b) An environment dominated by large database files. The files are multiple GB, and are accessed by conventional applications.

Answer: XFS is a good choice here because its internal data structures (B+trees) are designed to handle such large files. Note that the Google file system is not a good choice as it requires application changes.

- c) A single-system environment where performance is dominated by write performance. Applications require a high overall throughput when writing to the file system.

Answer: Many of the file systems we talked about this semester work here. The issue is explaining high write throughput feature. The LFS is designed for high write throughput by collecting small application writes into large device writes and by laying out the file system to avoid costly seeks. Soft-Updates improves write performance by removing as many synchronous constraints on writes as they can, meaning writes can be clustered and carried out asynchronously. Even XFS provides asynchronous logging and DMA to and from user space to improve write performance.

- d) A networked environment where failures are fairly common and file locality is high.

Answer: Locus, Ficus, or Coda all have applicable features here. They are all designed to deal with network partition and failure, and optimistic recovery works well when file locality is high.

Name:

email:

ID:

11. Several systems we discussed this semester gained significantly from altering interfaces. For each of the systems below explain what interface they changed and how (3 pts) and what advantage they gained (2 pts).

a) The Google File System

Answer: The GFS required applications to change their writing interface to appending only and their reading interface to include self-validating records. The advantage is that very large files can be manipulated through that interface efficiently.

b) The Xen virtualization system

Answer: They change the OS/hardware interface. Privileged instructions become hypervisor calls. In exchange performance for guest OSES improves over other systems that maintain strict compatibility.

c) The L3 (Leidtke) microkernel RPC interface

Answer: The RPC interface was changed from send/receive to call and respond and wait for next, requiring servers to use these primitives. The advantage was that in the common case fewer kernel crossings were required for each RPC, at a substantial savings.

d) Emeralds semaphores

Answer: The I/O calls were expanded to include any semaphore that was to be acquired shortly after the I/O call returned. This allowed the kernel to better schedule processes, sometimes delaying a process to avoid a useless context switch. The interface change was mitigated by using static code analysis by the compiler to fill in the expanded interface parameters.