CS 677 Spring 2019: Midterm

Solutions

May 1, 2019

1. **Short Answer Questions (20 points)**

   (a) (3pts) Will an application that is CPU-intensive in nature face scalability limitations if it is implemented using centralized data? Explain your answer in brief.

   Ans. No. In a CPU-intensive application, CPU cycles are the bottleneck. The rate of I/O is low, so most processing time is not dedicated to I/O. There's therefore a small I/O process queue which can be handled by the same server.

   (b) (4pts) What is the processor pool model in distributed systems? Would you consider a cluster of servers with a batch scheduler an example of the processor pool model?

   Ans. In a processor pool model, end users use diskless terminals with low processing power while a pool of backend processors handle processing of tasks sent by the terminals. Diskless terminals can also be referred to as lightweight workstations. A cluster of servers with a batch scheduler can be considered an example of a processor pool model because they have the same logical arrangement -: the scheduler sends tasks to the pool of available servers for processing.

   (c) (4pts) Consider a multi-tier web application with three tiers. Using a picture, explain whether the application employs a layered approach for designing distributed systems.

   Ans. Consider the illustration shown in Figure 1.

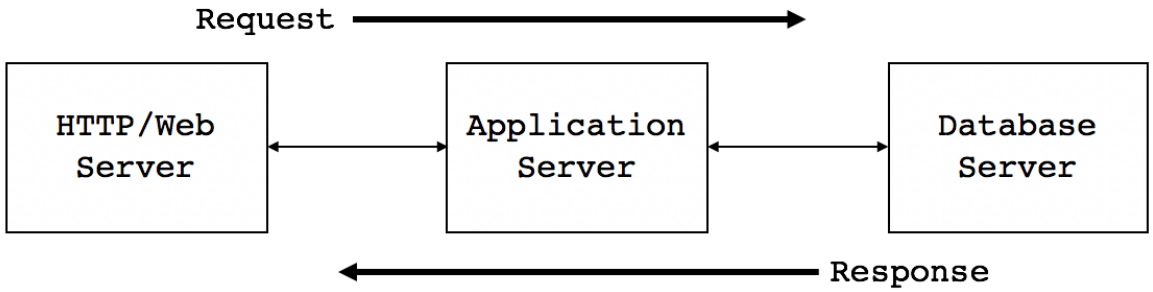

Figure 1: Sample Layered Architecture

A multi-tier web application with three tiers employs a layered approach because each layer only communicates with the layer above and below it in the architecture.

(d) (5pts) Consider the CAN peer-to-peer system that we studied in class. Assuming the system shown in the figure below, explain what happens when peer node with coordinates (0.7, 0.2) leaves the system.

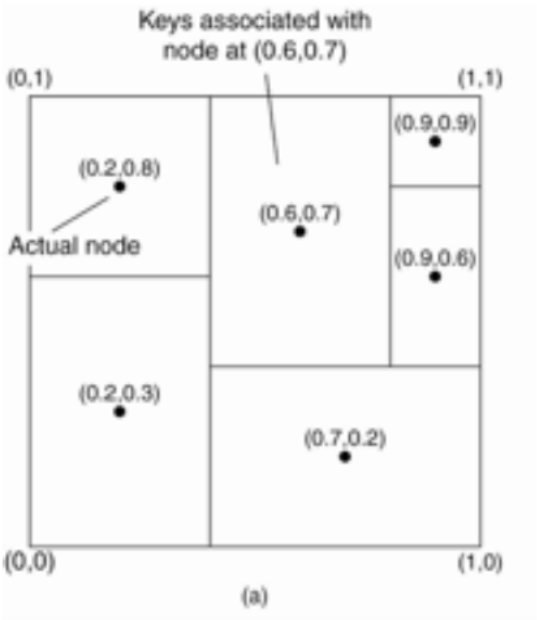

Figure 2: CAN System

Ans. If node (0.7, 0.2) leaves the system, the remaining nodes in the system will have to take over the values for which node (0.7, 0.2) was responsible. Therefore, the rectangle node (0.7, 0.2) will split vertically, joining nodes (0.6, 0.7) and (0.9, 0.6).

(e) (4pts) What is a super peer? Explain why super peers are useful in certain peer-to-peer systems.

Ans. A super-peer is a node in a P2P system that acts as a coordinator node through which other peers communicate with other peers in the network. They reduce messages exchanged in the network because messages only go to them, as opposed to being flooded to the whole network. Regular peers therefore see much less traffic in the presence of super-peers.

2. **Processes, Threads and Scheduling (22pts)**

(a) (6pts) Consider a multiprocessor CPU scheduler with a centralized run queue. Explain any performance bottlenecks that nay be encountered by the system as the number of cores in the system increases. What is an advantage of using a centralized run queue?

Ans. Lock contention - Since each processor must access the centralized queue in a mutually exclusive way, many processors trying to access will degrade performance due to lock contention.

An advantage of using a centralized queue always having a balanced load across multiple CPUs.

(b) (4pts) Explain why asynchronous event-based servers are more efficient than multi-threaded servers. Give a disadvantage of event-based servers.

Ans. Multithreaded servers incur the overhead cost of context switching between threads, which is not the case in event based servers. Event based servers use a single thread that runs in an event loop and does not have this overhead.

A disadvantage of event-based servers is they are unable to take advantage of parallelism in multicore servers. One can also say that they are much harder to program than multithreaded servers.

(c) (6pts) Consider the symmetric policy for distributed scheduling that uses a combination of sender- initiated and receiver-initiated approaches. Argue why the sender-initiated part of the policy is more effective at moderately low loads and why the receiver-initiated part of the policy is more effective at moderately high loads.

Ans. At moderately low loads, there are a lot of idle machines. A sender-initiated policy for scheduling can quickly find an idle server to offload a task to by uniform lookup, making this part of the policy more efficient in this condition.

At moderately high loads, many of the machines are busy. A receiver-initiated policy can quickly find machines to pick a task from by uniform lookup, making this part of the policy more efficient in scheduling in this condition.

(d) (6pts) Explain whether the mapping of an application?s threads to kernel-level schedulable entities follows a 1-1, many-1, or many-many mapping in (i) user-level threads, (ii) kernel-level threads, and (iii) light-weight processes.

Ans.

i. User-level-threads: Many-1. The OS is unaware of the presence of user-level threads, and can schedule them to only one schedulable entity.

ii. Kernel-level-threads: 1-1. The OS is aware of the presence of threads in this case and schedules them itself.

iii. Light-weight processes: Can be 1-1, many-1 or many-many. The mapping of LWPs to schedulable entities depends on the implementation, and can vary logically from a user-level to kernel-level implementation depending on the mapping chosen.

3. **Virtualization and Migration (18pts)**

(a) (8pts) Explain in brief how a hypervisor that employs type 2 virtualization is implemented. Next, explain how a hypervisor that employed paravirtualization is implemented. Compare the two meth- ods in terms of their run-time performance efficiency.

Ans.

Type 2 Virtualization In T2 virtualization, the hypervisor runs on a host OS. If sensitive instructions are encountered, they are dynamically replaced by procedures that

emulate the sensitive instructions through binary translation. This type of virtualization works on an unmodified guest OS.

Paravirtualization In paravirtualization, the guest OS kernel is modified to replace all sensitive instructions with hypercalls. Sensitive instructions are replaced with hypercalls statically. The guest OS acts like a user program making system calls, and the hypervisor executes the privileged operation invoked by the hypercall.

Performance Comparison Since paravirtualization statically replaces all system calls at compile time, it is more efficient that T2 virtualization because it does not incur the overhead of dynamic binary translation as is the case with T2 virtualization.

(b) (5pts)Draw a picture that illustrates why OS level virtualization is more lightweight when compared to hardware virtualization. Explain your reasons in brief.

Ans.

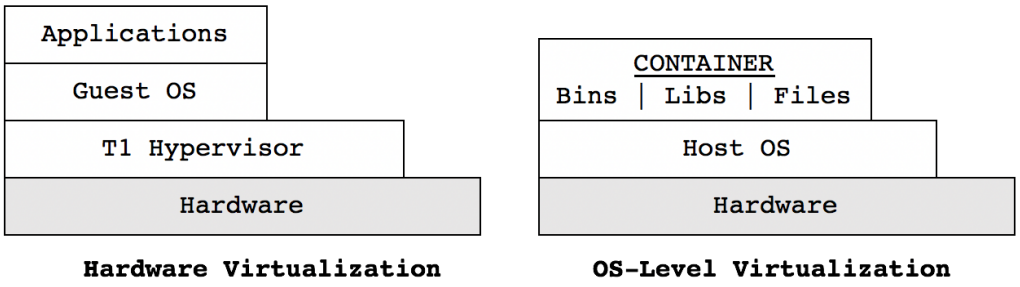

Figure 3: Hardware and OS-Level Virtualization

OS-Level virtualization is more lightweight because it does not require booting of the guest OS, but rather, just isolation of the libraries, files etc for the container. Containers run on the host OS. Containers are also faster to provision compared to hardware virtualization.

(c) (5pts) What strategy can be employed to handle unattached and fixed resources when migrating a process? Assume that process to resource binding is by *identifier*.

Ans.

Possible strategies for handling unattached resources-:

i. Move the resource along with migrating process
ii. Create a global reference to the resource

For fixed resources-:

i. Create a global reference to the resource

4. **Distributed Communication (18pts)**

(a) (6pts) Explain how a traditional RPC system handles (i) passing pointers as parameters, and (ii) differences in little endian and big endian architectures at the callee and caller machines.

Ans.

i. Traditional RPC systems do not support passing pointers as parameters.

ii. When there are differences in data representation (e.g., endianness) between caller and callee machines, a traditional RPC will use a standard data representation convention (e.g., XDR).

(b) (6pts) Explain why the receiver process need not be executing when the sender process sends a message using the persistent communication method. List the key methods used to send and receive messages in a message queuing system and explain each in brief.

Ans. In persistent communication, a message is stored until the (next) receiver is ready to receive (e.g., e-mail). Therefore, the receiver process does not need to be executing when the sender process sends a message using the persistent communication method.

The key methods used to send and receive messages in a message queueing system are:

i. **Put**: Append a message to a specified queue.

ii. **Get**: Block until the specified queue is nonempty, and remove the first message

(c) (6pts) How can buffering at the client-side streaming media player enables the client to avoid play-back glitches caused by jitter in the network reception of packets from the server? Jitter is defined to be the variation in the inter-packet arrival times at the receiver.

Ans. For example, in video streaming a frame is played back after the corresponding packet has been received by the client. In case of jitter, packet may fail to arrive in time. For example, a frame has been played but the packets for the next frame haven't arrived. If no buffering is used, then the client must pause the playback until the packets arrive.

With client-side buffering, the media streaming playback will not start until a certain buffer size has been reached. After the playback has started, even if some packets fail to arrive in time, chances are that the client media player still have content in the buffer to continue playback. Thus with client-side buffering the client will experience much less playback glitches once the playback has started.

5. **Naming and Synchronization (23pts)**

(a) (6pts) Consider a distributed naming system that can be implemented using recursive or iterative name resolution methods. In the scenario where the client and the name servers are on different continents, which method yields better performance from the client latency standpoint? Which method is better from the perspective of load on the name servers?

Ans. From the client latency standpoint, the recursive name resolution method yields better performance. This is because when client and name servers are on different continents, communication between the client and a name server has a much higher RTT than communication between two name servers. In recursive name resolution there is only one request/response between the client and the root name server while other requests/responses are between different name servers. In comparison, in iterative name resolution the client need to request name servers at each level of the hierarchy.

From the perspective of load on name servers, the iterative name resolution method is better. This is because in iterative name resolution, clients can use caching to reduce the load at higher level name servers.

(b) (5pts) Explain using an example why it is harmful to set a clock backwards during clock synchronization.

Ans. Setting clock backward may cause program that depends on timestamps to work incorrectly. An example is the `make` build tool which compares the timestamps of target and source files to determine whether the target needs to be recompiled. If the target is built at time `t1` and then the clock is set backward, changes to source files may cause the source files to have timestamps that are earlier than `t1`. Now if we try to build again, `make` will not recompile the target because it thinks the source files have not been changed since last build, which leads to stale build results.

(c) (6pts) Consider the global positioning system algorithm discussed in class. In a scenario where the clock on the GPS receiver is perfectly synchronized with the GPS satellite, it is possible to simplify the location computation algorithm. If so, explain what simplification is possible. If not, explain why not.

Ans. Yes, it's possible to simplify the location computation algorithm.

In normal case, each GPS satellite gives us an equation

$$c(t + d_r - t_1) = \sqrt{(x - x_1)^2 + (y - y_1)^2 + (z - z_1)^2}$$

which contains 4 unknowns (x, y, z, and d_r). Therefore we need at least 4 satellites (4 equations) to compute 4 unknowns.

If the clock on the GPS receiver is perfectly synchronized with the GPS satellite, then we know that $d_r = 0$. Now the equation becomes

$$c(t - t_1) = \sqrt{(x - x_1)^2 + (y - y_1)^2 + (z - z_1)^2}$$

with only 3 unknowns (x, y, and z). So in this case we only need 3 satellites.

(d) (6pts) What is a vector clock? Construct an example to show that two independent events in the system have incomparable vector clock values.

Ans. A vector clock is an algorithm for generating a partial ordering of events in a distributed system. Each process in the system maintains a vector of clock values which represents the number of events that is known to have occurred at each process in the system. For example, for the vector V_i on process i:

- $V_i[i]$ represents the number of events that have occurred at process i.
- $V_i[j]$ represents the number of events that process i knows have occurred at process j.

Vector clocks help reason about causality: if $V[A] < V[B]$ then A causally precedes B. Vector clocks are updated using the following rules:

- Each time a process experiences an internal event, it increments its own logical clock in the vector by one.
- Each time a process sends a message, it piggybacks its own vector with the message.
- Each time a process receives a message, it increments its own logical clock in the vector by one and updates each element in its vector by taking the maximum of the value in its own vector clock and the value in the vector in the received message for every element.

In the example above, event a and event d are two independent events that have incomparable vector clock values ($\langle 1, 0\rangle$ and $\langle 0, 2\rangle$).