CS 6210 Fall 2023 Test 3

(120 min Canvas Quiz)

Max Points: 90

Internet Scale Computing [42 points]

Giant-Scale Services [23 points]

1. You are building your own new search service, and the underlying corpus is 100 TB in size. Say you have machines with SSDs of size 10TB each. You want to ensure that the server is able to handle 100 simultaneous queries. Assume that for a successful response, the entire corpus needs to be scanned (i.e., each query needs a full harvest).

(a) [3 points] How many total machines will you need and how will the data be partitioned and replicated among the machines to meet the service level expectation?

Answer:

- 100 TB partitioned on 10 machines
- 10 machines for each query; for 100 simultaneous queries 100 replicas of each partition.
- Totally 1000 machines

(b) [4 points] If two machines fail, how many queries can the service handle simultaneously now?

[2 points] CASE 1: If the two machines are storing the same copy of data, then 98 simultaneous queries.

[2 points] CASE 2: If the two machines are not storing the same copy, then 99 simultaneous queries.

2. You are a system administrator running rolling upgrades on 2 differently structured applications. Application A runs on 50 servers with data partitioned evenly across the 50 servers for answering queries. Application B uses 100 servers with data fully replicated on all the servers.
The rolling upgrade applies the upgrade for 10% of the servers in each wave for each application, respectively. Assume the time to answer a query is directly proportional to the data harvested.

(a) [2 points] During the rolling upgrade, what will be the loss in application A's yield and harvest?

Answer:

During each rolling upgrade window, 5 servers out of the total 50 will be down. There will be a 10% reduction in harvest (+1), but yield will remain the same (+1)

(b) [2 points] During the rolling upgrade, what will be the loss in application B's yield and harvest?

Answer:

During each rolling upgrade window, 10 servers out of the total 100 will be down. There will be no reduction in harvest (+1), but yield will reduce by 10%.

3. [4 points] A data center uses full replication of the data corpus on all its servers. The load manager implements a simple layer-3 (i.e., network level) scheduler that directs an incoming client request to a specific server in a strictly round-robin fashion. State two disadvantages to this load management strategy

Answer:

- May not achieve good balance since all client requests may not take the same amount of time to complete.
- Since there is a one-to-one relationship between a given client and given server there is no way to hide server failures from the client.

4. [4 points] How would you design a load balancer that overcomes the disadvantages you identified in the network-level load balancer?

Answer:

- To hide server failures, implement the load balancer to work at a higher level in the network protocol stack: e.g., layer-4 (transport level), and implement failover to a different server upon server failure
- For better load balance, take the current server load into account in the load balancer while assigning new work

5. [2 points] (Answer True/False with Justification) Graceful degradation when there is an overload at a datacenter always means reducing the harvest while keeping the yield the same.

Answer:

- False (0 with no justification; +1 with justification even if flawed)
- Some services (e.g., mail) may need full harvest (+1)

6. [2 points] (Answer True/False with Justification) Partitioning the data corpus among the servers at a datacenter is done to increase fault tolerance.

Answer:

- False (0 with no justification; +1 with justification even if flawed)
- It is done to increase parallelism in query processing (+1)

Map Reduce [7 points]

7. [2 points] With M mappers and R reducers, how many intermediate files are created by each mapper, and how many intermediate files are created in total?

[+1] R files per mapper
[+1] M*R total

8. The Master node decides to simultaneously spawn more number of mappers for each input shard and more number of reducers for each distinct output to be produced by the map-reduce computation.

(a) [2 points] Is there a benefit to this strategy?

Answer:
Ensures that node failures do not severely impact the response time due to the redundant computations spawned simultaneously.

(b) [1 point] How does the Master ensure correctness when there are redundant mappers for each input shard?
Answer:
[+1] Ignore late arrivals for mapper

(c) [2 points] How does the Master ensure correctness when there are redundant reducers for each distinct output?

Answer:
[+1] Upon completion of a reducer, the master will make the result visible to the application via "rename" command to make the local file of the reducer globally visible
[+1] if this is a straggler reducer (for an already completed reduce task) then the result will be ignored by the master

CDN Coral [12 points]

(12 points)

9. Alice at **Node-id 0**, wants to store a new video of her dance on Coral CDN. She computes the hash of the video to be **8**. Coral has 16 nodes with Node-id's 0 to 15. Assume all nodes are directly reachable from any node.
Here is the state of the nodes for **key=8** (Node-id's ranging from 0 to 15):

- Nodes 3, 5, 7, 8: Full or loadedNodes 0, 1, 2, 4, 6, 9, 10, 11, 12, 13, 14, 15: Neither full nor loaded

(a) (2 point) what is the command issued by Alice?

Answer:
Put(8, 0)

(b) (4 points) List the nodes visited by Alice's command with reasoning

Answer: (+1 for each hop)

- 1st hop: Node 12 (1/2 distance to the desired destination)
- 2nd hop: Node 10 (1/4 distance to the desired destination)
- 3rd hop: Node 9 (1/8 distance to the desired destination)
- 4th hop: Node 8 (desired destination)

(c) (3 points) Where will Alice's put command end up storing her <key, value> pair? Why?

Answer:

- Node 9 (+1)
- Node 8 cannot accept the put command since it is full/loaded (+1)
- So on the return path the put will place the <key, value> pair in Node 9 since it is neither full nor loaded (+1)

(d) (3 points) Assume that when Alice's put command reaches the final destination per Coral's key-based routing protocol, ALL the nodes except Node-id 0 get full/loaded for key=8. In this case, where will Alice's put command end up storing her <key, value> pair? Why?

Answer:

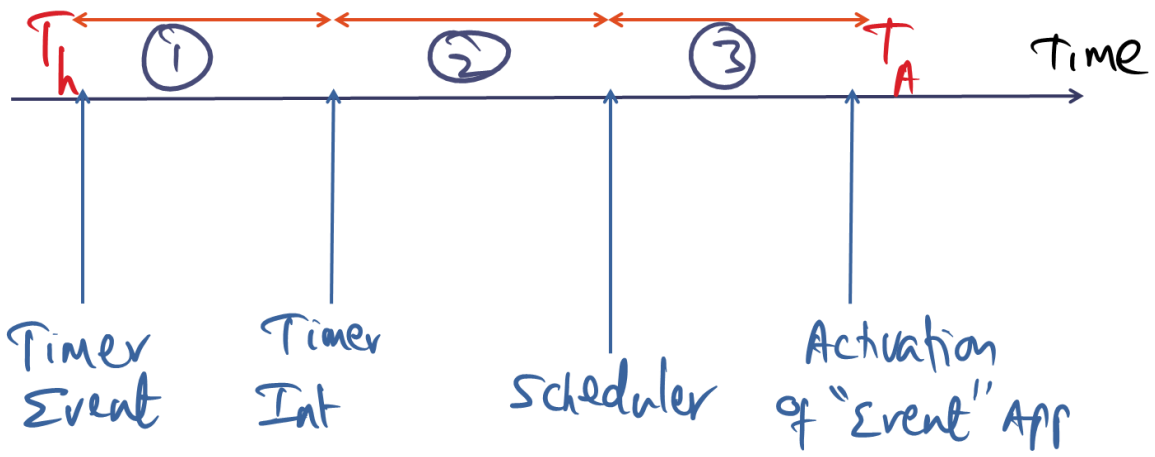
- Node 0 (+1)

On the return path, the put will not be able to place the <key, value> pair in any of the nodes it visited on the forward path (namely, 9, 10, 12). (+2)

[8 points]

Real-time and Multimedia [20 points]

TS-Linux [14 points]



1. The above picture shows the latencies involved between a timer event happening and the activation of the event (i.e., processing of the event) by the application for which this event is intended.

(a) [2 points] In the above picture, succinctly explain the source of the latency marked by "2".

Answer:

- "2" represents the time taken by the kernel to complete processing the interrupt. There are two components to this latency:
  - a) the time elapsed before the interrupt handler is run which could be due to the interrupts being turned off by the OS (e.g., if it is inside a critical section in the kernel) (+2)
  - (Bonus) the actual time taken by the interrupt handler code in the kernel (+0.5)

(b) [2 points] With the objective of supporting real-time applications in a general-purpose OS like Linux, how can one decrease the latency marked by "2"?

Answer:

- Allow the kernel to be pre-empted to run the interrupt handler code immediately (+2)

(c) [2 points] In the above picture, succinctly explain the source of the latency marked by "3".

Answer:

- The scheduler has to make a decision to run the "next" runnable process on the processor. This chosen process need not be the process for which the interrupt happened if there are some higher priority processes to be run. (+2)

(d) [2 points] What is the suggested solution in TS-Linux to guarantee that the process for which the interrupt happened gets to run in a timely manner?

Answer:

- Admission control. (+1)
- At launch time, the scheduler guarantees a share of “x%” of CPU time every “T” units of wall-clock time. (+1)

2. [2 points] Clio, a developer, is testing a new audio processing application on TS-Linux that requires precise timing for signal processing tasks. Clio has set up a one-shot timer to trigger every 500 microseconds for real-time audio processing. For the application to work without audio glitches, he needs to ensure that his application code runs in a timely manner subsequent to the timer interrupt going off. Clio notices occasional audio glitches. Why could this be happening?

Answer:

- Clio may have forgotten to do admission control when he launched his app to guarantee timely execution of his application code.
- The system may be experiencing heavy kernel activity at times resulting in kernel pre-emption latency despite Clio doing admission control at app launch.

(+2 if either point mentioned)

3. Lyra, a TS-Linux system administrator, observes that a time-sensitive application is experiencing occasional timing glitches despite being coded correctly using one-shot timers, doing admission control, and running at the highest priority level for scheduling.

(a) [2 points] Why could this be happening?

Answer:

- Due to priority inversion. (+1)
- The app could be making blocking calls to a low-priority server (e.g., a window manager) which gets pre-empted by some other higher priority task. (+1)

(b) [2 points] What could be done to ensure that such glitches do not happen?

Answer:

- Modify the scheduler to inherit the priority of the calling application on blocking calls to system services. (+2)
- Enhance the kernel's scheduling flexibility by introducing additional preemption points to minimize the duration of non-preemptable critical sections, thus reducing the likelihood of timing glitches in high-priority processes. (+2)

PTS [6 points]

4. In an application using PTS, assume channel ch1 has items with timestamps 25, 50, 75, 100, 125

(a) (2 points)
Consider the following code sequence by a thread T1 of the application:

<item, ts> = Get (ch1, "now"); // returns the latest item from channel ch1
Digest = Process (item); // code to process the item just gotten
Put (ch2, Digest, ts); // put digest with timestamp ts

What is the timestamp associated with the above put operation?

Answer:

- Get(ch1, "now") would return 125 as the timestamp since it corresponds to the latest item in the channel and Put() will use the same timestamp 125, to maintain the temporal ordering as items move through various channels.

(b)
Immediately following the above execution, another thread T2 of the application executes the following code sequence. Assume that no more items have been added to ch1.

<item, ts> = Get (ch1, "now"); // returns the latest item from channel ch1
Digest = Process (item); // code to process the item just gotten
Put (ch2, Digest, ts); // put digest with timestamp ts

(i) (2 points) (Answer True/False with justification – No points without justification)

T2's put operation will overwrite T1's earlier put operation in the channel ch2.

Answer:
- False. (0 with no justification; +1 with justification even if flawed)
- Items in PTS channels are immutable (+1)

(ii) (2 points) (Answer True/False with justification – No points without justification)

T2's put operation will result in PTS runtime returning an error since an item with that timestamp is already present in ch2.

Answer:

- False. (0 with no justification; +1 with justification even if flawed)
- It is fine to have multiple items with the same timestamp in a PTS channel. So, PTS will simply place the new item (with the same timestamp 125) in the channel as well (+1)

Failures and Recovery [17 points]

LRVM [4 points]

1. [2 points] During crash recovery, the redo log is applied to the data segments to bring the server to a consistent state.

Why does LRVM choose to apply the log to the affected data segments starting from the tail of the log? (Succinct bullet please)

Answer:

- To avoid redundant work since the redo log may contain updates to the same blocks of the data segments from consecutive transactions.

2. [2 points] How does the "no restore" mode in begin-transaction help in improving the performance of a server written on top of LRVM?

Answer:

- No need to create an in-memory UNDO record for the set-range calls within this transaction.

RioVista [7 points]

3. (3 points) The server using RioVista directly modifies the external data segments mapped into its virtual memory within transactions. Suppose there is a power failure in the middle of the transaction. What are the actions taken by RioVista to bring the server to a consistent state before restarting the server when the power is restored?

Answer:

- UNDO log corresponding to the set-range calls within the transaction is created at the beginning of the transaction.
- UNDO log is in the Rio file cache which is a portion of DRAM that is battery backed and hence survives the power failure.
- Once power is restored, the UNDO log is applied to the external data segments to bring the server to a consistent state.

4. (4 points) The question pertains to a server that uses RioVista for its persistent state management. Upon crash recovery, is the server restarted from the beginning of the transaction where the crash occurred? Explain with succinct bullets to get credit.

Answer:

- RVM is not checkpoint/restart facility. It only records changes to persistent state that occurred during the execution of the transactions contained in the server (not the entire volatile memory state). Thus, there is no way for restarting the server precisely at the beginning of the transaction at which the crash occurred. (+4)
- The only thing that matters for the server is the consistency of the external data segments and NOT the volatile state of the server at the point of the crash. (+1)
- Therefore, the server is always started anew so that it can do all the initialization (mapping external data segments to the virtual memory) before getting ready for business. (+1)

QuickSilver [6 points]

5. [6 points] You have implemented all the system services (window manager, file system, socket library, etc.) on top of QuickSilver OS fully utilizing the recovery management mechanism that is built into the OS. Take one service, let's say window manager. With succinct bullets, list what you must do as the developer of the window manager service to make sure that if your client crashes all the resources allocated on the client machine as a result of interaction with your service are recovered.

Answer:

- You will supply handlers that the TMs on the client machine and the machine on which your service is running can invoke every time an API call to your service is made by the client. (+2)
- The TMs via the handler will record whatever resource is allocated to the client as a result of the API call (memory, display real estate, etc.) in the log file on the affected machines. (+2)
- If the client crashes, the shadow Transaction tree created on behalf of the client interaction with your service will use the logs created by the TMs on the affected machines (client machine and the machine where your service is running) to clean up all the breadcrumbs. (+2)

Security [11 points]

Security Principles, AFS [11 points]

1. You are in the midst of a passionate discussion about the security guarantees of the Andrew File System with a colleague. Your colleague makes the following 4 claims about AFS. Explain comprehensively yet succinctly whether you agree or disagree with your colleague and provide the relevant justification.

(a) [2 points] Claim 1: There is no way for the server to validate the authenticity of the client during RPC session establishment

Ans - The claim is inaccurate. During the rpc establishment phase, the server sends a Random number E[Yr, HKC] to the client and expects E[Yr+1, HKC] as the response. Validating that the response indeed contains Yr+1 helps the server validate the client.

(b) [3 points] Claim 2: Nothing in the AFS design prevents replay attacks.

Ans - The claim is inaccurate.

When client contacts server, server sends a random challenge encrypted with client's key, which is only known to that client. If replay attacker contacts server, it will receive this encrypted challenge but will be unable to decrypt it and thus is unable to provide correct challenge response.

Within an RPC session, the server is anyway looking for duplicate packets (with the same sequence number) since there could be retransmissions due to packet loss; a replayed packet by an attacker will be treated as a duplicate packet and will not affect the security guarantees

(c) [3 points] Claim 3: AFS has controls to prevent over exposure of the user's login credentials (username and password).

Ans - The claim is accurate.

The user's password is only used ONLY ONCE for each login session of the user. Subsequently for the duration of the login session the ephemeral identity (secrettoken) is used as the client-identity (not the username). Similarly, for the duration of the login session the ephemeral handshake key provided by VICE used as the key for encryption for establishing each new RPC session.

(d) [3 points] Claim 4: AFS does not pass Saltzer's litmus test for Denial-of-Service Attacks.

Ans – Claim is true

- A malicious attacker could grab a packet on the untrusted network link from a legitimate user. He can then replay the packet continuously causing the server to do useless work thus preventing legitimate users from getting useful work done.
- A legitimate but malicious user could indirectly thwart others from getting useful work done by flooding the server with requests. This form of "Denial of Service" attack is a by-product of the fact that AFS has no containment of resource usage for each user.