Homework 5

CS425/ECE428 Spring 2024

Due: Wednesday, April 24 at 11:59 p.m.

1. Two phase commit and Paxos .... 14 points

   In Spanner and similar systems, a combination of two-phase commit (2PC) and Paxos protocols are used. Both the coordinator and participants in 2PC are implemented as *replica groups*, using Paxos to achieve consensus in the group. Each replica group has a leader, so during 2PC, the leader of the coordinator group communicates with the leaders of the participant groups.

   During the execution of 2PC in such a system, there are three points at which a consensus must be achieved within the nodes in a replica group for a transaction to be committed: (i) at each participant group to prepare for a commit, (ii) at the coordinator to decide on a commit after receiving a vote from each participant, and (iii) at each participant again to log the final commit.

   Suppose that there is one coordinator and three participants. Each of these has a Paxos replica group with six nodes. The leader of each replica group also acts as the proposer and the distinguished learner for the Paxos protocol, while the remaining five nodes are acceptors (the leader sends its prepare and accept messages to all five acceptors). The leaders of the participant and the coordinator replica groups send appropriate messages for 2PC to one another once consensus has been achieved (a decision has been reached) in their respective replica groups. Assume for simplicity that the coordinator replica group only coordinates the transaction and does not participate in processing the transaction (so the coordinator leader need not send prepare and commit messages to itself during 2PC).

   The communication latency between each pair of nodes *within* each group is exactly $10ms$ and the communication latency between any pair of nodes in two different groups is exactly $30ms$. The processing latency at each node is negligible.

   Answer the following questions assuming that there are no failures or lost messages. Further assume that the leader of each replica group has already been elected / pre-configured. All participant groups are willing to commit the transaction, and all nodes within each replica group are completely in sync with one-another.

   (a) (6 points) With this combined 2PC / Paxos protocol,

      (i) what is the minimum amount of time it would take for each node in the participant group to commit a transaction after the leader of the coordinator group receives the "commit" command from the client? *(2 points)*

      (ii) how many messages are exchanged in the system before all nodes in the participant groups commit the transaction? (Ignore any message that a process may send to itself). *(4 points)*

      [Hint: Think about the message exchanges required by each protocol (2PC and Paxos). Are there messages that can be safely sent in parallel to reduce the commit latency?]

   (b) (2 points) What is the earliest point at which the coordinator group's leader can safely tell the client that the transaction will be successfully committed? Calculate the latency until this point (from the time since the leader of the coordinator group receives the "commit" command from the client).

   (c) (6 points) Suppose we re-configure the system such that the leader of the coordinator group also acts as the leader (proposer and distinguished learner) for the participant Paxos groups. Five nodes in each participant group continue to be acceptors. The original leader within each participant replica group simply acts as a learner (and is no longer the leader/proposer/distinguished learner). With this modification:

      (i) what is the minimum time it takes for each node in the participant group to commit a transaction after the leader of the coordinator group receives the "commit" command from the client? (2 points)

      (ii) how many messages are exchanged in the system before all nodes in the participant groups commit the transaction? (Ignore any message that a process may send to itself). *(4 points)*

2. DHT .... 16 points

Consider a Chord DHT with a 16-bit address space and the following 100 nodes (hexadecimal values in parentheses).

834 (342),	1847 (737),	2180 (884),	4562 (11d2),
4883 (1313),	5579 (15cb),	6016 (1780),	6134 (17f6),
6351 (18cf),	7576 (1d98),	9379 (24a3),	9916 (26bc),
10023 (2727),	10111 (277f),	10336 (2860),	10967 (2ad7),
11053 (2b2d),	11101 (2b5d),	11967 (2ebf),	12721 (31b1),
12972 (32ac),	12982 (32b6),	14007 (36b7),	14305 (37e1),
16121 (3ef9),	16641 (4101),	17460 (4434),	17949 (461d),
18572 (488c),	18622 (48be),	19963 (4dfb),	20012 (4e2c),
20368 (4f90),	20721 (50f1),	21251 (5303),	21422 (53ae),
22213 (56c5),	24052 (5df4),	25092 (6204),	28927 (70ff),
29112 (71b8),	30656 (77c0),	31428 (7ac4),	32083 (7d53),
32199 (7dc7),	32403 (7e93),	32753 (7ff1),	33876 (8454),
35527 (8ac7),	36849 (8ff1),	37774 (938e),	38193 (9531),
39091 (98b3),	39606 (9ab6),	40067 (9c83),	41627 (a29b),
42532 (a624),	42784 (a720),	43304 (a928),	43590 (aa46),
43935 (ab9f),	43968 (abc0),	44644 (ae64),	44673 (ae81),
44686 (ae8e),	45039 (afef),	46261 (b4b5),	46306 (b4e2),
46685 (b65d),	47254 (b896),	47478 (b976),	48441 (bd39),
48680 (be28),	48694 (be36),	49660 (c1fc),	49844 (c2b4),
50197 (c415),	51425 (c8e1),	52368 (cc90),	52848 (ce70),
53684 (d1b4),	55038 (d6fe),	55264 (d7e0),	55393 (d861),
55556 (d904),	56344 (dc18),	56740 (dda4),	58569 (e4c9),
58641 (e511),	60436 (ec14),	60597 (ecb5),	60599 (ecb7),
62458 (f3fa),	62795 (f54b),	62930 (f5d2),	64744 (fce8),
65011 (fdf3),	65217 (fec1),	65424 (ff90),	65454 (ffae),

For programmatic computations, these numbers have also been made available at:
`https://courses.grainger.illinois.edu/ece428/sp2024/assets/hw/hw5-ids.txt`

(a) (6 points) List the fingers of node 24052.

(b) (6 points) List the nodes that would be encountered on the lookup of the following keys by node 24052:

  (i) 36918

  (ii) 19834

(c) (4 points) A power outage takes out a few specific nodes: the ones whose identifiers are even numbers. Assume that each node maintains only one successor, and no stabilization algorithm has had a chance to run, so the finger tables have not been updated. When a node in the normal lookup protocol tries to contact a finger entry that is no longer alive (i.e. its attempt to connect with that node fails), it switches to the next best option in its finger table that is alive. Under these conditions, will a lookup of the key 51039 by node 16641 be successful? If yes, list the nodes that 16641 would contact. If not, explain why.

3. MapReduce . . . . 6 points

   (a) (6 points) Given three vectors V_1, V_2, and V_3 each having a dimension of N. Use a map-reduce chain to compute the dot product of $(V_1 + V_2)$ and V_3. The input to the map-reduce chain is in the following key-value format: (k, v), with $k = (i, n)$, where $i \in [1, N]$ is the index of the vector V_n, and v is the corresponding value $(V_n[i])$. The output of your map-reduce chain must of the form (-, final result). Assume there are 100 nodes (or servers) in your cluster. Your map-reduce chain must support proper load-balancing across these nodes. In particular, assuming a vector dimension of 5000, ensure that a single node is not required to handle more than $\approx$150 values at any stage. You can assume that, if allowed by your map-reduce semantics, the underlying framework perfectly load-balances how different keys are sent to different nodes. Also explain what aspect of your map-reduce chain allows it to satisfy this load-balancing requirement.

4. Dominant Resource Fair Scheduling . . . . 4 points

   (a) (4 points) Consider two cloud jobs, Job 1 and Job 2. Each task of Job 1 requires 5 units of CPU and 10MB RAM. Each task of Job 2 requires 2 units of CPU and 30MB RAM. You need to schedule these jobs on a system with 60 units of CPU and 300MB RAM using the dominant resource fairness criteria. How many tasks for each job will you schedule to maximize resource utilization while achieving dominant resource fairness (to the best extent possible)?

   Note that you cannot schedule fractional tasks. The unfairness introduced due to rounding the number of tasks to integer values can be ignored.