Homework 6

CS425/ECE428 Spring 2021

Due: Thursday, April 29 at 11:59 p.m.

1. Two phase commit and Paxos ..... 12 points

   In a 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 two participants. Each of these has a Paxos replica group with 3 nodes. The leader of each replica group also acts as the proposer and the distinguished learner for the Paxos protocol, while the remaining two nodes are acceptors.

   The communication latency between each pair of nodes *within* each group is exactly $5ms$ and the communication latency between any pair of nodes in two different groups is exactly $10ms$. 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. Both 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,

      (ai) 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? *(3 points)*

      (aii) how many messages are exchanged in the system before each node in the participant group commits the transaction? (Ignore any message that a process may send to itself). *(3 points)*

      [Hint: Think about the message exchanges required by each protocol (2PC and Paxos). Are there messages that can be 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 is/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) (4 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. Two nodes in each participant group continue to be acceptors. With this modification, 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. DHT ..... 14 points

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

```
  451  (1c3),   1195  (4ab),   1460  (5b4),   3854  (f0e),
 4175 (104f),   4400 (1130),   5651 (1613),   6357 (18d5),
 7483 (1d3b),   9105 (2391),  10073 (2759),  10483 (28f3),
10825 (2a49),  11273 (2c09),  11446 (2cb6),  11535 (2d0f),
11693 (2dad),  11925 (2e95),  12324 (3024),  13700 (3584),
15390 (3c1e),  15925 (3e35),  16030 (3e9e),  16569 (40b9),
16737 (4161),  17154 (4302),  19264 (4b40),  19477 (4c15),
19503 (4c2f),  19919 (4dcf),  19954 (4df2),  21065 (5249),
21324 (534c),  22507 (57eb),  22508 (57ec),  22972 (59bc),
23062 (5a16),  23168 (5a80),  23403 (5b6b),  23478 (5bb6),
23933 (5d7d),  24070 (5e06),  25209 (6279),  27513 (6b79),
29012 (7154),  30150 (75c6),  30179 (75e3),  30297 (7659),
31325 (7a5d),  32048 (7d30),  33824 (8420),  34612 (8734),
35080 (8908),  35394 (8a42),  35643 (8b3b),  37013 (9095),
37957 (9445),  38222 (954e),  39475 (9a33),  39558 (9a86),
39683 (9b03),  40571 (9e7b),  40996 (a024),  41553 (a251),
41995 (a40b),  43305 (a929),  43541 (aa15),  44307 (ad13),
44623 (ae4f),  45889 (b341),  46198 (b476),  46655 (b63f),
48044 (bbac),  48736 (be60),  49061 (bfa5),  49720 (c238),
50683 (c5fb),  52347 (cc7b),  52806 (ce46),  52841 (ce69),
53305 (d039),  53819 (d23b),  55639 (d957),  56470 (dc96),
56905 (de49),  57082 (defa),  57250 (dfa2),  57496 (e098),
57585 (e0f1),  58280 (e3a8),  58800 (e5b0),  59048 (e6a8),
60672 (ed00),  61532 (f05c),  61851 (f19b),  62819 (f563),
63086 (f66e),  63966 (f9de),  64459 (fbcb),  64654 (fc8e),
```

For programmatic computations, these numbers have also been made available at:
`https://courses.grainger.illinois.edu/ece428/sp2021/assets/hw/hw6-ids.txt`

(a) (4 points) List the fingers of node 49720 (c238).

(b) (5 points) List the nodes that would be encountered on the lookup of key 15975 (3e67) by node 49720 (c238).

(c) (5 points) A power outage takes out a few specific nodes: the ones whose identifiers are perfect multiples of 3. Assume no stabilization algorithm has had a chance to run, and so the finger tables have not been updated. List the nodes that 49720 (c238) would contact to look up the key 15975 (3e67). When a node in the normal lookup protocol tries to contact a finger entry that is no longer alive, it switches to the next best option in its finger table that is alive.

3. MapReduce.....14 points

(a) (4 points) Use a map-reduce chain to trace people who came in unsafe contact with each other during an epidemic outbreak in a university town. The input to the map-reduce chain is in the following key-value format: (k, v), with $k = personID$, and v is a list of tuple $((x, y), \textit{entry timestamp}, \textit{exit timestamp}, \textit{vaccinated})$, where (x, y) is the GPS coordinate of a location, *entry timestamp* is when the person appeared at that location, *exit timestamp* is when the person left the location, and *vaccinated* is a boolean variable set to *true* if the person was fully vaccinated at that time (for the entire duration between the entry and exit timestamps). If a person visited the same location more than once, the value list for that person would contain multiple entries for that location but with different timestamps and vaccination status. Output of the map-reduce chain must have the following format: (k, v), with $k = personID$, and v is a list of ids of people k "came in unsafe contact with". Assume that person A "comes in unsafe contact with" another person B if both A and B are in the same location at the same time for more than 5 consecutive minutes, and at least one of them is not fully vaccinated during that time.

(b) (4 points) Use a map-reduce chain to compute the dot product of two vectors V_1 and V_2, each having a dimension of N. 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 ($V_n[i]$). Your map-reduce chain must support proper partitioning and load-balancing across workers. In particular, assuming a vector dimension of 5000, and 20 workers, ensure that a single worker is not required to handle more than ≈250 values for any key at any stage.

(c) (6 points) Given a directed graph $G = (V, E)$, use a map-reduce chain to compute the set of vertices that are reachable in *exactly 4 hops* from each vertex. For example, in a graph with vertices $\{a, b, c, d, e\}$ and the following directed edges, $a \to b \to c \to d \to e$, the vertex e is reachable in exactly four hops from vertex a. The input to the map-reduce chain is in the following key-value format: (k, v) where k is a graph vertex and v is a list of its out-neighbors; i.e., for each $x \in v, (k, x)$ is a directed edge in E. The output must be key-value pairs (k, v), where k is a graph vertex and v is a list of vertices that are reachable in exactly four hops from k. The list must be empty if there are no vertices reachable in exactly four hops from k. Vertices maybe repeated in the four-hop path and need not be distinct. It is also possible for a vertex to be exactly four hops away from itself, in which case it can be included in the list. For your assistance, the first map function for an exemplar map-reduce chain has been provided below. You may choose to use the same function, or design your own.

```
function MAP1((k, v)):
    for node in v do
        emit ( ( node , ( "in" , k ) ) )
        emit ( ( k , ( "out" , node ) ) )
    end for
    if v is empty then
        emit ( (k, ("out", _)))
    end if
end function
```