CS425 Fall 2025 – Homework 1

(a.k.a. "Sports Smorgasbord!")

Out: Aug 26, 2025. Due: Sep 18, 2025 (2 pm US Central)

Topics: Clouds, Mapreduce, Gossip, Failure detectors, Grids (Lectures 1-6)

Instructions:

1. Please double-check that the top of this page mentions the correct semester you are taking the course in (otherwise, you are looking at an old version and we will not accept your submitted solutions).
2. **Attempt any 8 out of the 10 problems** in this homework (regardless of how many credits you're taking the course for). If you attempt more, we will grade only the first 8 solutions that appear in your homework (and ignore the rest). Choose wisely!
3. Please hand in **solutions that are typed** (you may use your favorite word processor. We will not accept handwritten solutions. Figures (e.g., timeline questions) and equations (if any) may be drawn by hand (and scanned).
4. **All students (On-campus and Online/Coursera)** – Please submit PDF only! Please submit on Gradescope. [https://www.gradescope.com/]
5. Please **start each problem on a fresh page**, and **type your name at the top of each page**. **And on Gradescope please tag each page with the problem number!**
6. Homeworks will be **due at time and date noted above. No extensions. For DRES students only:** once the solutions are posted (typically a few hours after the HW is due), subsequent submissions will get a zero**. All non-DRES students must submit by the deadline time+date.**
7. Each problem has the same grade value as the others (10 points each).
8. Unless otherwise specified in the question, the only resources you can avail of in your HWs are the provided course materials (slides, textbooks, etc.), and communication with instructor/TA via discussion forum and e-mail.
9. You can discuss lecture concepts and the questions on Piazza and with your friends, but you cannot discuss solutions or ideas on Piazza.

Prologue: As we gear up for various sports events in 2025, various teams rely on distributed computing and cloud computing systems to improve their team's performances and chances of medaling. This homework uses fictitious stories and characters from sports to keep the questions fun and relevant (and also informative!). The choice of teams or players in questions, or their frequency across questions, is purely arbitrary, and there is no message implied in the question framing. Any resemblance to persons, places, or events, living or dead, past, present or future, is purely coincidental. These stories and this homework are not intended to side with or against any team or sportsperson, or make comments about any of them. They are not at supporting or endorsing, nor at criticizing or disparaging, any team or sportsperson. All interns are fictional, as are their goals and actions.

Problems:

1. To continue their winning ways in international competitions, the US Relay teams (swimming and athletics, and in women's, men's, and mixed relays) wants to build a system for gossiping tips on how to pass the baton efficiently and reliably. They want to analyze the "topology-aware gossip" protocol you've seen in lecture. However, instead of the lecture slide example of 2 subnets joined by 1 router, here we have a total of N processes (nodes), where $N=2^k-1$, organized in a perfectly-balanced binary tree with k levels of nodes (k is large, and therefore so is N). Each node has 1 parent and 2 children. All nodes have a full membership list, and there are no failures (messages or processes). The root node's two immediate subtrees are called the "left subtree" and the "right subtree" (each of those two subtrees containing $N=2^{k-1}-1$ nodes). Each node in the tree knows whether it belongs to the left or right subtree. The topology-aware push gossip works as follows. A process Pj (not the root of the tree) picks a gossip target within its subtree (left subtree or right subtree) with probability (1-2/N), and it picks the root of the tree as gossip target with probability (2/N). The root picks gossip targets uniformly at random across the tree. All nodes use the same gossip period, each node picks one gossip target per round, and they are synchronized. Only push gossip is used. Calculate the following:
    a. O() average dissemination time for a gossip (with one sender from any subnet; note the root is not the sender), as a function of N, and
    b. Maximum (across all nodes) of the average load on the node.

   Please (always) justify all your answers, and (always) show your working. (This is true for all questions on all Homeworks and all exam questions except multiple-choice/fill in the blank questions).

2. (You can use other websites only for this question, but you should not cut and paste text. Please write answers in your own words!) The Dallas Wings WNBA

and LA Lakers Pistons NBA teams who just can't seem to win anything nowadays in spite of multi-million dollar salaries, want to together AWS to do analytics of their performances. But neither team knows anything about cloud computing! Can you help them? (Please limit your answer for each part to less than 50 words. Be concise!)

a. What is the key difference between AWS Lambda and AWS EC2?
b. What are the two key differences between AWS Lambda and AWS spot instances (think: pricing and how long instances last)?
c. What is the key difference between a "regular" AWS instance (EC2) and a "burstable" AWS instance?
d. What is the difference between a GPU and a "TPU" (among cloud offerings)?
e. What is the difference between AWS spot instances and Google Cloud preemptible instances?
f. Give one example application (class) where you would prefer AWS EC2, and one where you would prefer AWS Lambda. Justify your choices briefly.

3. (You can use other websites only for this question, but you should not cut and paste text. Please write answers in your own words!) The US fencing team wants to contain their next opponents and send them to the cloud, so naturally they start looking at containers. They spend their free time researching the topic, and they come across the following terms: Container, Docker, Virtual Machine (VM), Kubernetes (K8s), K8s Pod, K8s cluster. Help them please! Do the following:
   a. Define each of these 6 terms concisely (1 sentence per term).
   b. Among the four possibilities of docker, container, VM, and K8s, give one scenario where one would use each of them over the three possibilities. Think of applications! (Keep your answer to this part to < 100 words).
   c. Give two key differences between a K8s node, K8s pod, and K8s cluster.

4. Drone fireworks are becoming an eco-friendly alternative. They require cloud computing to plan the drone paths! Someone tells you that MapReduce job for the newest drone-based fireworks for the upcoming LA 2028 Olympics closing ceremony show will use a total 240 Map tasks, 180 Reduce tasks, and runs across a cluster of 40 machines. Under YARN, each machine has 4 containers. Since fireworks are time critical, this is the only MapReduce job running in the entire

cluster. Each Map task takes exactly 15 seconds to execute (this includes time to fetch its input, but not shuffle time), and generates exactly 20 MB of output data. The total bandwidth of the network is 2 Gbps. The partitioning function is such that each Reduce task fetches an equal amount of data from each Map task. There is a barrier between when *all* the Map tasks finish their computation, and the start of the shuffle traffic transfer from *any* of the Map tasks (remember from lecture where the Map output is stored!). The Reduce task execution time is 35 seconds (includes time to write output to HDFS including 3-way replication but does not include shuffle traffic time to fetch its input). Calculate the total time for the MapReduce job to finish.

5. The failing US men's 4X100m Olympic medley swimming relay team, which lost its Gold Medal in the Paris 2024 Olympics after a 64 year unbeaten streak is very upset, and they are trying to improve their performance. They want to avoid future failures by building a failure detector for an asynchronous system of *N* processes (*N* very large). Their ring-based failure detection protocol works as follows: Assume there are N=M*K*R processes in the system (M, K, R, are positive integers, each > 2). Arrange these N processes in a MxKxR 3-dimensional matrix (tesseract), with M processes in each column, and K processes in each row, and R processes in the 3rd dimension (aisles). All processes maintain a full membership list, however pinging is partial. Each process P_{ijk} (in *i*-th row and *j*-th column and *k*-th aisle) ~~periodically (every T time units)~~ marks a subset of its membership list as its *Monitoring Set.* The monitoring set of a given process, once selected, does not change. The monitoring set of P_{ijk} contains: i) all the processes in in its own column P_{*jk} , ii) all the other processes in its own row P_{i*k} , and ii) all the processes in in its own aisle P_{ij*} . At this point, there are two options available to you: <u>Option 1</u> – Each process sends heartbeats to its monitoring set members. <u>Option 2</u> – Each process periodically pings all its monitoring set members; pings are responded to by acks, just like in the SWIM protocol (but there are no indirect pings or indirect acks or suspicions.). Failure detection timeouts work as usual: Option 1 has the heartbeat receiver timeout waiting for a heartbeat, while Option 2 has the pinging process (pinger) time out. The suspected process is immediately marked as failed. This is run in an asynchronous distributed system.
   a. How many failures does Option 1 take to violate completeness? That is, find the value *L* so that if there are (*L-1*) simultaneous failures, all of them

will be detected, but if there are *L* simultaneous failures then not all of them may be detected.

   b. Answer the same above question for Option 2.
   c. The Jamaican relay team (Oh Yeah, the US has competition!) claims this algorithm satisfies accuracy for *S* simultaneous failures or fewer, *for both Option 1 and Option 2*. Find the value of *S* (as a function of K, M, R, N, etc.).

6. To train new sportspersons the US sports agencies have been wondering whether it is faster to "push" gossip about new coaching opportunities or have the young sportspersons "pull" gossip about these new coaching opportunities. You are given a network of M groups (M very large) of "nodes". You can assume each group has K nodes (K very large), so there is a total of N=M*K+1 nodes, with the extra node being the US sports agency (the root). The root is the sender of all multicast gossips. However, the membership graph is not a full membership graph. Instead, each node (apart from the root) knows only all the nodes in its own group, AND knows the root node, but knows no other nodes in any other group. The root node knows ONE node from each of the M groups (i.e., the root has M neighbors, one from each group). Each of the N processes executes Push gossip (or respectively Pull gossip) exactly as described in class. Each node (including root) picks exactly ONE target per gossip round. The rounds are synchronized. Analyze in O() notation (as a function of K, M):
   a. Expected Latency of pure Push gossip, and
   b. Expected Latency of pure Pull gossip.
   c. Which of those two is faster? Why (give a brief intuitive reason)?
   d. What is one downside of Pull gossip in the above setting?

7. The US has a cricket team! (Yes, they actually participated in the recent world cup and even beat a world-class team). Since their main player is an engineer, the US cricket team designed a failure detector but to "make it faster" to find injuries among players. They have experimented with various failure detectors they learnt in class, but they have made a few tweaks. For each of these changes (in isolation), say what is the one biggest advantage and the one biggest disadvantage of the change (and why). Keep each answer to under 50 words (give brief justifications).
   a. They use Gossip-style failure detection, but they set $T_{cleanup} = 0$.
   b. They use SWIM-style failure detection, but they removed the Suspicion feature.

c. They use SWIM-style failure detection, but they removed the round robin pinging + random permutation, and instead just randomly select each ping target.

8. *(Note: For all Mapreduce questions please only use the class definition/template for Mapreduce jobs, i.e., Map/Reduce tasks/functions, and not other sources from the Web, since there are many different variants on the Web!)* Instead of using judges to judge events like figure skating and gymnastics, the Olympics are considering moving to an audience-based voting system to score the players. An intern suggests using Condorcet voting. This intern thinks MapReduce could be useful for "instant runoff voting" in primaries. (Fun fact: several states, including Alaska, recently started using instant runoff voting in their elections!) Here's how instant runoff voting works. Consider an election with exactly three candidates on the ballot (i.e., competing sportspeople) – A, B, C. (While we use an example with 3 candidates, Condorcet voting generalizes to an arbitrary number of candidates). Every voter ranks the three candidates as first preference, second preference, and last preference. Between any two candidates X and Y, if a majority of voters ranked X above Y, then X *dominates* Y (and vice versa)—note that this only takes into account X and Y's relative rankings in the preference order, not where the third candidate appears. A *Condorcet* winner is a candidate that dominates (gets more votes than) all other candidates (pair wise) on the ballot. By definition an election can have at most one Condorcet winner (either one winner or no winner). You are given a dataset of votes from *N* voters (*N* is odd and large, and so dataset is sharded), where each vote V has three fields V.1, V.2, V.3, respectively for the first, second, and third preference votes of that voter. Each line of input is one such voter's vote V (input to initial Map function). Write a MapReduce program that outputs either the unique single Condorcet winner among the three candidates A, B, or C, *or* if there is no single Condorcet winner, then it outputs the list of candidate (or candidates if tied) with the highest Condorcet count (those that dominate the most number of candidates). For background -- in MapReduce, one writes a program for Map that processes one input line at a time and outputs zero or more (key, value) pairs; and one writes a program for Reduce that processes an input of (key, all values for key). The iteration over input lines is done automatically by the MapReduce framework. You can assume this data is already sharded in HDFS and can be loaded from there. Each line is one vote V and is read as the value and the key is empty (in the first by Map stage). Note that intermediate data from a Map is not available for subsequent stages! Correctness is important, efficiency is secondary (but you must have some parallelism). Write either pseudocode, or clear unambiguous

descriptions. Requirements are similar to homework questions on Mapreduce. you should have at least some parallelism. Ensure that your output does not contain duplicates. You can set your key and value to arbitrary objects. You cannot retain data at any of the machines from a task (Map or Reduce) for use in a later task. Chaining MapReduces is allowed, as long as you don't over-use chaining (where parallelization could have helped instead). Each MapReduce in a chain can read the dataset, but there is no other persistent memory across the chain. Pseudocode should be clear and unambiguous, and it can be coarse-grained, e.g., you can say "get top k from this list by field x", "sort this list by key x", etc.). Be clear and concise and don't miss any steps.

9. You are an intern at the MLB. The MLB gives you a dataset D containing information from an asymmetrical social network. Each line in D is a pair (a,b) which means user a follows user b. Write a MapReduce program (Map and Reduce separately) that outputs the list of all users X who satisfy the following FOUR conditions simultaneously: i) user X has at least 10 million followers, and ii) X follows fewer than 10 other users, iii) all the users that X follows, also follow X back, and iv) X does NOT follow the unique user u, where u is the user who has the highest number of followers of any user in the entire dataset D (you are told there is only one such unique user u, though you are not told who u is). Hint: This question is an "extended" version of one of the exercise problems in the lecture slides. You should have at least some parallelism. Ensure that your output does not contain duplicates. You can set your key and value to arbitrary objects. You cannot retain data at any of the machines from a task (Map or Reduce) for use in a later task. Chaining MapReduces is allowed, as long as you don't over-use chaining (where parallelization could have helped instead). Each MapReduce in a chain can read the dataset, but there is no other persistent memory across the chain. Pseudocode should be clear and unambiguous, and it can be coarse-grained, e.g., you can say "get top k from this list by field x", "sort this list by key x", etc.). Be clear and concise and don't miss any steps.

10. The US Tennis Association would like to analyze why their players are giants on social media but don't seem to win anywhere as many majors as Martina Navratilova, Chris Evert, or Pete Sampras did. They have noticed particularly that on Mondays and Fridays the current US tennis players seem to play the worst. They want to use Mapreduce to analyze this. You are given three datasets. D1 – Post Log: log of a social network log that captures information about all the posts on a social network over several weeks. Each line contains tuples (a, p, wd:hh:mm:ss) where a is a user id, p is the post id, and wd:hh:mm:ss is the

timestamp of the post p made by the user a at time wd:hh:mm:ss (weekday:hour:minute:second). Note that wd (weekday) only takes only 7 values: Sun, Mon, Tue, Wed, Thu, Fri, Sat. If a user has multiple posts, they will appear as separate entries. The entries are not sorted in any order. You can assume post id's (p) are unique globally (each appears exactly once in the log). A second dataset D2 is a "Like Log": log of social network that captures likes on the social network. Each line contains a pair (a, p, wd:hh:mm:ss) indicating user a liked post with post id p at time wd:hh:mm:ss. There are no duplicate pairs containing the same (a,p,*) pair (i.e., each user likes a post at most once). Finally dataset D3 is a (large) list of users (a) who are current US tennis players. Write a MapReduce program to find the *top K posters (user ids from D3) who get the most likes from posts they post on Monday or Friday*. Your output should be a file with K lines in it, each line containing a user id. Outputs should not contain duplicates. Please ensure that a Map stage reads data from only one input dataset (i.e., if a Map reads directly from D2, don't use it to also read from D1. And vice-versa.) – this is good practice consistent with good MR practices. You should have at least some parallelism. Ensure that your output does not contain duplicates. You can set your key and value to arbitrary objects. You cannot retain data at any of the machines from a task (Map or Reduce) for use in a later task. Chaining MapReduces is allowed, as long as you don't over-use chaining (where parallelization could have helped instead). Each MapReduce in a chain can read the dataset, but there is no other persistent memory across the chain. Pseudocode should be clear and unambiguous, and it can be coarse-grained, e.g., you can say "get top k from this list by field x", "sort this list by key x", etc.). Be clear and concise and don't miss any steps.

====== END OF HOMEWORK 1 =====