CS425 Fall 2026 – Homework 1

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

Out: Aug 27, 2026. Due: Sep 20, 2026 (Sunday, 11:59 pm US Central)

Topics: Clouds, MapReduce, Gossip, Failure detectors, Grids (Lectures 1-5)

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. If you attempt more, we will grade only the first 8 solutions that appear in your homework. Each problem has the same grade value (10 points each).
3. Your **solutions must be typed and not handwritten**. Figures (e.g., timelines) and equations (if any) may be hand-drawn (and scanned).
4. Submit PDF only. **Start each problem on a fresh page**.
5. Please submit on Gradescope. [https://www.gradescope.com/] Self-signup using the code on the website.
6. **Very important: On Gradescope, please tag each page with the problem number!**
7. Homework is **due on the date and time 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**.**
8. Always justify all your answers, and show your working.
9. Unless otherwise specified in the question, the only resources you can avail of in your HW are the provided course materials (slides, textbooks, etc.) and communication with instructors/TA via the discussion forum.
10. You can discuss lecture concepts and the questions on Piazza and with your friends, but do not discuss solutions or ideas on Piazza.

Prologue*:* As we gear up for various sports events in 2026, various teams rely on distributed and cloud computing systems to improve their performance and chance of medaling. The choice and frequency of occurrence of team or player names is arbitrary. Any resemblance to persons, places, or events, living or dead, past, present, or future, is coincidental. These stories are not intended to take sides with or against any team or sportsperson, and do not support, endorse, criticize, or disparage any team or sportsperson.

Problems:

1. To continue their winning ways in international competitions, the US Relay teams plan to build a system for gossiping tips for efficient passing of the baton. They analyze the "topology-aware gossip" protocol from the lecture. We have two subnets, each with N/2 processes (nodes), that are connected via one root process (aka root node); so a total of N+1 processes/nodes. All nodes have a full membership list, and there are no failures (messages or nodes). Let's call the subnets A and B. Each node knows whether it is the root, or if it belongs to A or B. The topology-aware push gossip works as follows. Any node that is not the root picks a gossip target within its own subnet with probability (1 - 2/N), and it picks the root node as the gossip target with probability (2/N). The root node picks a gossip target randomly from the N nodes with uniform probability. All nodes use the same gossip period; each node picks one gossip target per round, and the rounds are synchronized. Only push gossip is used. Calculate the following:
    a. O( ) average dissemination time for a gossip with one sender from any subtree (note the root is not the sender), as a function of N.
    b. Maximum (across all nodes) of the average load on the node.

2. (You may use the web to research this question, but you should not cut and paste text. Use your own words, but fewer than 50 words per part.) The WNBA Dallas Wings and the NBA LA Lakers teams can't seem to win nowadays despite multi-million dollar player salaries. They want to use AWS to analyze their games. But neither team knows much about cloud computing! Can you help them?
    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 may use the web to research this question, but you should not cut and paste text. Use 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!
    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 the fencing team would use each of them over the other three possibilities. Think of applications. (Keep your answer 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 the MapReduce job for the drone-based fireworks for the upcoming LA 2028 Olympics closing ceremony show will use a total of 300 Map tasks, 200 Reduce tasks, and run across a cluster of 20 machines. Under YARN, each machine has 5 containers. Since fireworks are time-critical, this is the only MapReduce job running in the entire cluster. Each Map task takes exactly 20 seconds to execute (this includes time to fetch its input, but not shuffle time), and generates exactly 25 MB of output data. The total bandwidth of the network is 2.5 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 3-way replicated HDFS, 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, wants to improve its performance. It wants to build a failure detector for an asynchronous system of N processes (N very large). Their failure detection protocol works as follows: Assume there are N=M*K*R processes in the system (M, K, R, are integers, each > 2). Arrange these N processes in an M×K×R

3-dimensional matrix (tesseract), with M processes in each column, 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 uses a subset of its membership list as its *monitoring set.* The monitoring set of P_{ijk} (in the *i*-th row and *j*-th column and *k*-th aisle) contains: (i) all other processes in its own row P_{*jk} , (ii) all other processes in its own column P_{i*k} , and (iii) all other processes in its own aisle P_{ij*} . We have two options: Option 1 – Each process sends heartbeats to its monitoring set members. Option 2 – Each process 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 uses a heartbeat receiver timeout waiting for a heartbeat, while in Option 2 the pinging process times out. A suspected process is immediately marked as failed. This is run as an asynchronous distributed system.

   a. How many failures does it take to violate completeness in Option 1? 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 above question for Option 2.
   c. The Jamaican relay team (Oh yeah, that's the main 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, and N).

6. To train new sportspersons, US sports agencies have been wondering whether it is faster to "push" gossip about new coaching opportunities or to have the young sportspersons "pull" gossip about them. Someone argues, "Push gossip is slow. It is better to do pull gossip, and even with fixed fanout, it converges in O(log(log(N)) time!" Are they right? If yes, give a proof (informal proof ok). If they are wrong, give a proof (informal proof). (Note: Push gossip and pull gossip mentioned here are the same protocols discussed in lecture)

7. The US has a cricket 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 the one biggest advantage and the one biggest disadvantage of the change is (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 selected each ping target.

You must read these instructions before you solve problems 8–10

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!

You should have at least some parallelism. You can set your key and value to arbitrary objects.

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 cannot retain data on any of the machines from a task (Map or Reduce) for use in a later task. Each MapReduce in a chain can read the dataset, but there is no other persistent memory across the chain.

Chaining MapReduces is allowed, as long as you don't over-use chaining (where parallelization could have helped instead).

Pseudocode should be 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.

8. You are an intern at the NBA and WNBA. Their presidents give 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, and 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). Ensure that your output does not contain duplicates.
Hint: This question is an "extended" version of one of the exercise problems in the lecture slides.

9. The US Tennis Association would like to analyze why their players are giants on social media but don't seem to win anywhere near as many majors as Martina Navratilova, Chris Evert, or Pete Sampras did. They want to use MapReduce to analyze this.

   You are given three datasets.

   - D1 – Post Log: log of all the posts on a social network over several weeks. Each line contains tuples (a, p, ts), where a is a user ID, p is the post ID, and ts is the timestamp of the post p made by the user. If a user has multiple posts, they will appear as separate entries. The entries are not sorted in any order. You can assume post IDs (p) are unique globally (each appears exactly once in the log).
   - D2 – "Like Log": log of a social network that captures likes on the social network. Each line contains a pair (a, p, ts) indicating user a liked post with post ID p at time ts. There are no duplicate pairs containing the same (a,p,*) pair (i.e., each user likes a post at most once).
   - D3 – a (large) list of user IDs of 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*. Your output should be a file with K lines in it, each line containing a user ID. Outputs should not contain duplicates.

10. *(You may use the web to read more about BPE, but not the MapReduce-related part of the question.)* Byte Pair Encoding (BPE) is a technique used to construct a vocabulary of tokens from a large text corpus. Suppose you have been tasked with writing a MapReduce program to train a BPE tokenizer using the following simplified version:

    Assume that each line of input is one sentence in the corpus (you do not need to merge pairs across sentence boundaries). The input corpus is stored in HDFS. The initial vocabulary is the individual characters found in the text. You can assume each input sentence has already been split into a sequence of initial tokens based on the initial vocabulary. For example, the word "low" already appears as: l o w in the input text.

    BPE training then proceeds for K iterations. In each iteration:

    1. Count the frequency of every adjacent token pair in the corpus.

       For example, in the token sequence:

       l o w _ l o w

       the adjacent pairs are (l,o), (o,w), (w,_), (_,l), (l,o), (o,w).

    2. Find the globally most frequent adjacent token pair. If there is a tie, break ties lexicographically.

3. Create a new token in the vocabulary by merging that pair (*otherwise referred to as creating a merge rule*). For example, if (l,o) is the most frequent pair, you will create a new token "lo" and add it to the vocabulary.
4. Finally, rewrite the corpus by replacing every occurrence of that most frequent adjacent pair with the newly created token. In the above example, the sequence
l o w _ l o w
becomes
lo w _ lo w

After K iterations, the trainer will output (i) trained vocabulary: the initial vocabulary together with all new tokens created during the K iterations (ii) the final merged corpus.

Write a chained MapReduce program to train this BPE tokenizer.

a. Describe the Map and Reduce functions for the MapReduce job(s) required to implement one BPE iteration. Assume a tokenized corpus at the beginning of an iteration; explain how you count adjacent token-pair frequencies, find the globally most frequent pair to create the new token, and rewrite the corpus with the newly created token to produce the input corpus for the next iteration.
b. Give an intuition of how you would run K BPE iterations. (In practice, the number of training iterations in practice is determined by the target vocabulary size. For example, to reach GPT -3's target vocabulary size, one would have to run tens of thousands of iterations.)
c. Given your solutions to (a) and (b), and what you know about MapReduce's execution model, argue whether MapReduce is a good overall framework for BPE training. *Hints:* (i) Does BPE's iterative nature fit well with MapReduce? (ii) Within a single BPE iteration, think about which parts can exploit MapReduce's parallelism well, and which parts limit parallelism. Please use only the MapReduce instructions above.

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