<u>CS425 Fall 2024 – Homework 1</u>

<u>(a.k.a. "Olympics Odyssey!")</u>

Out: Aug 27, 2024. Due: Sep 19, 2024 (2 pm US Central)

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

Instructions:

1. **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!
2. 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).
3. **All students (On-campus and Online/Coursera)** – Please submit PDF only! Please submit on Gradescope. [https://www.gradescope.com/]
4. 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!**
5. 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.**
6. Each problem has the same grade value as the others (10 points each).
7. 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.
8. You can discuss lecture concepts and the questions on Piazza and with your friends, but you cannot discuss solutions or ideas on Piazza.

Prologue: In the 2024 Olympics, various teams from the US relied on distributed computing and cloud computing systems to improve their team's performances and chances of medaling. This question

This homework uses fictitious stories and characters from the ongoing presidential campaigns to frame the homework problems, and to keep the questions fun and relevant (and also informative!). The choice of candidates or parties in questions, or their frequency across questions, is purely arbitrary, and there is no political message 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 candidate, or make comments about any candidate. They are not at supporting or endorsing, nor at criticizing or disparaging, any candidate, campaign, people in campaigns, political parties or affiliations, or voters or citizens or persons living in the US. All interns are fictional, as are their goals and actions.

Problems:

1. Since they became very popular and successful, 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 nodes (processes), evenly spread out across $\sqrt{N}$ subnets (each subnet containing $\sqrt{N}$ nodes), all joined by 1 router. The subnets are numbered S0, S1, S2, … S($\sqrt{N}$-1). All these $\sqrt{N}$ subnets are connected together via 1 router. You can assume all nodes have a full membership list, and there are no failures (messages or processes). The topology-aware gossip works as follows. Consider a process Pj choosing gossip targets. The process' gossip targets depend on the subnet Si that it lies in. During a gossip round, the process Pj selects either *b* "inside-subnet Si gossip targets" with probability (1-1/$\sqrt{N}$), OR *b* "outside-subnet Si gossip targets" with probability 1/$\sqrt{N}$. The only "restriction" is that after process Pj is infected, for the next O(log($\sqrt{N}$)) rounds Pj picks only inside-subnet targets (no outside-subnet targets) – thereafter in a gossip round at Pj, either all its targets are inside-subnet or all are outside-subnet. Inside-subnet gossip targets from Pj (in Si) are selected uniformly at random from among the processes of Si. Outside-gossip targets from Pj (in Si) are *only* picked from the processes in the "next" subnet S((i+1)mod$\sqrt{N}$), and they are picked uniformly at randomly from the processes lying in that "next" subnet. The gossiping of a message does not stop (i.e., it is gossiped forever based on the above protocol). Does this topology-aware gossip protocol satisfy both the requirements of: (i) O(log(N)) average dissemination time for a gossip (with one sender from any subnet), and (ii) an O(1) messages/time unit load on the router at any time during the gossip spread? Justify your answers.
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 US women's

gymnastics team, fresh off their All-round team Gold, along with the US men's basketball team which also won Gold, 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 a GPU instance and what are two key differences between a AWS GPU instance and a "general purpose" AWS EC2 instance?
d. What is the key difference between a "regular" AWS instance and a "burstable" AWS instance?
e. What is the difference between a GPU and a "TPU" (among cloud offerings)?
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: Docker, Container, 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. The failing US men's 4X100m Olympic medley swimming relay team, which lost its Gold Medal 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: each process i selects a set of target processes (once selected, these targets don't change) and asks these processes to send <u>to</u> it (i.e., to process i) heartbeats directly. Targets are selected as follows. All processes are organized in a virtual ring. Targets of a process i include three subsets: its k predecessors, its k successors, and k further processes chosen randomly (uniformly at random) among all non-predecessor/successor/i processes. Once the list of targets is selected, it is not changed (including the

randomly selected members). *k* is a fixed number much smaller than *N*, and known to all. Heartbeats are not relayed (so this is not gossip, but more like ring failure detection, except there is no ring), and process *i* times out if it doesn't receive heartbeats. A process is detected as failed if any of its heartbeat receivers do not receive expected heartbeats within a pre-specified timeout.

   a. When is completeness violated? That is, find the value *M* so that if there are (*M-1*) simultaneous failures, all of them will be detected, but if there are *M* simultaneous failures then not all of them may be detected.
   b. Is the algorithm 100% accurate?
   c. If the period is fixed (say 1 s) at all processes, what is the load on each process in terms of heartbeats that it needs to send? Calculate the worst case, best case, and average load.

5. The Olympics athletes from all teams are so large in number (over 10K athletes in Paris 2024!) that they would like to build a new system to connect athletes' phones with each other for dates, I mean, for conversations. This new membership protocol (in a system with N total nodes) is a partial membership protocol where each node knows a subset of nodes. The membership graph is created as follows: for every pair of nodes (A,B) there is a bidirectional link A-B with probability p (flip a coin with heads probability p: if heads then A-B link exists, otherwise there is no A-B link). Since Olympic athletes are selected carefully and they stay for the Olympics, for this question you can assume no node failures, and no nodes join or leave. The goal is to select the smallest p (as a function of N) so that the entire graph of N nodes is connected (i.e., every node is reachable via some path from every other node). Find the asymptotic value of p as a function of N for this to be true. You can do this either mathematically or via simulation. We just need the Big-O notation for p. (One way to go about solving this problem: You may want to write a small program that simulates these graphs, for various values of N. Find the threshold p that gives connectivity for each N. Then plot these threshold p values vs. N, and notice the trend.)
6. Did you know that the Olympics in 2028 will be Los Angeles?! Oh yeah! But LA is large, so leveraging the membership protocol of the previous question, the LA Olympics 2028 organizing team wants to use it for sending gossip multicast messages. You can assume the membership graph is connected (i.e., p is high enough). In a given membership graph, each node has *k* neighbors selected at random (much smaller than *N* the total number of processes in the system. Don't worry about how the membership protocol runs. While this way of selecting neighbors is different from the previous question, you can rest assured that the same limits for number of neighbors you have calculated in the previous question also hold for this new setup, to assure graph connectivity! One of the

wonders of Math!). For this question you can assume processes don't fail or join. Each gossip multicast message is gossiped to *m* randomly selected neighbors (from the membership list), where $m < k$, and $m = O(\log(N))$, with the latter needed to ensure spread of gossip. Note that *m* is the total set of gossip targets (over multiple protocol periods) picked by a given process, not a per-round fanout. The organizers argue that due to random selection, the overall "behavior" of this protocol (in terms of dissemination time of gossips, etc.) is the same as in the case where all processes might have had full membership lists (everyone known to everyone in the group), and each gossip (partial membership list) was sent to *m* neighbors (in total, over all gossip periods). Are they right or is their reasoning as poor as LA's horrible traffic? If yes, then give a mathematical proof. If no, show why.

7. *(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!)* In the entire Olympic village, to do contact tracing for Covid and other communicable diseases, the LA 2028 Olympic organizing team wants to write a MapReduce program. You are given two datasets: D1 is a large dataset contains (in each line of input) triples (Unique_Person_Name, location, start_time, end_time) indicating that Unique_Person_Name was in location from start_time to end_time (times include dates, etc.—you don't need to worry about that). You can assume there are fixed set of locations (very large in number). Dataset D2 contains (in each line of input) a Unique_Person_Names of persons who tested positive for Covid. Your goal is to find the list of individuals (unique names, non-duplicated) from D1 who were in the same place and same time as at least one individual in D2 (note that D2's names will also appear in D1). Write a (chained) MapReduce program for this. You can assume times can be compared (< > =). Feel free to assume appropriate library functions (e.g., test if two time intervals overlap), and you don't need to use fancy datastructures. As usual, all data is sharded in HDFS, and you can chain MapReduces (but be judicious), you should have some parallelism, etc. Each line is read as the value and the key is empty (in the first Map stage). To keep it simple, we recommend starting with two MapReduce programs, one that reads from D1 and one that reads from D2. Later MapReduces can read from output of previous MapReduces, as well as D1 and/or D2. Note that intermediate data from a Map is not available for subsequent stages! (The datasets are kept private within CDC, so you can ignore privacy concerns for this question!) Correctness is important, efficiency is secondary (but you must have some parallelism). Write either pseudocode, or clear unambiguous descriptions. You can chain Mapreduces if you want (but

only if you must, and even then, only the least number). Be sure to output each pair at most once (e.g., in sorted order). You don't need to write code – pseudocode is fine as long as it is understandable. Hint: Think about the "key" in Map output. As a rule, all Mapreduce programs must avoid duplicates in the final output. That is, the same output element must not be repeated multiple times in the output. (Following advice valid for all MR questions in this HW.) In the initial parts of your chain, use one Mapreduce job to read a given dataset from HDFS/GFS file storage (one MR job cannot read two differently formatted datasets). You should read a dataset from storage only via one MR job, then store it as KV pairs in HDFS/GFS for later MR jobs to use them (if needed).

8. Inspired by the "semi-naked blue guy" from the Paris 2024 Olympics opening ceremony, LA 2028 Opening Ceremony is thinking of picking a lucky (unlucky?) social media celebrity to be the center of controversy. They start with an input file containing information from an asymmetrical social network (e.g., Twitter/X/Insta/TikTok) about which users "follow" which other users. If user a follows b, the entry line in the input dataset is (a, b), and this line appears exactly once. Each line of the input dataset contains only one such (a,b) pair. Write a MapReduce program (Map and Reduce separately) that outputs the list of all users U who satisfy the following three conditions simultaneously: U has at least 10 million followers, and U herself/himself follows at most 100 users, and U follows at least one user V who in turn has at least 10 million followers. You can chain Mapreduces if you want (but only if you must, and even then, only the least number). Be sure to output each pair at most once (e.g., in sorted order). You don't need to write code – pseudocode is fine as long as it is understandable. Hint: Think about the "key" in Map output. As a rule, all Mapreduce programs must avoid duplicates in the final output. That is, the same output element must not be repeated multiple times in the output.
9. In LA 2028 Olympics, Cricket will be a new sport! The LA 2028 organizing committee along with ICC (International Cricket Council) wants to use social media to spread awareness of cricket and how similar yet different it is from baseball. Unlike the previous question this question uses a *symmetrical* social network, with one link contained one per line. An entry (a,b) implies a and b both follow each other. Note that a and b may be in any order of ids, i.e., it is not guaranteed that a is lexicographically smaller than b, or vice versa. There is also no repetition: if (a,b) appears, there is no repetition of either (a,b) or (b,a) elsewhere in the dataset. Write a MapReduce program that outputs all pairs (a,b) such that a and b are connected to each other, both a and b are connected to @Olympics, but exactly one of a or b (not both, not neither) is connected to @ICC. Neither @Olympics nor @ICC should be output among the answers (i.e., neither

a nor b in your outputs should be ICC or Olympics). You can chain Mapreduces if you want (but only if you must, and even then, only the least number). Be sure to output each pair at most once (e.g., in sorted order). You don't need to write code – pseudocode is fine as long as it is understandable. Hint: Think about the "key" in Map output. As a rule, all Mapreduce programs must avoid duplicates in the final output. That is, the same output element must not be repeated multiple times in the output.

10. Fireworks (at Olympics or elsewhere) require cloud computing! Someone tells you that MapReduce job for the Paris 2024 closing ceremony fireworks show uses 160 Maps, 80 Reduces, and runs across a cluster of 20 machines. Each machine has 4 containers (i.e., can run at most 4 tasks simultaneously). There is only one MapReduce job running in the entire cluster. Each Map task takes exactly 10 seconds to execute (this includes time to fetch its input), and generates exactly 100 MB of output data. The total bandwidth of the network is 10 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 all the Map tasks finishing their computation, and the shuffle traffic starting to transfer (remember from lecture where the Map output is stored!). The Reduce task execution time is 40 seconds (includes time to write output but does not include shuffle traffic time to fetch its input). There is a barrier between all the Reduce tasks finishing their computation, and the Reduce outputs being written to HDFS with 3-way replication. Calculate the total time for the MapReduce job to finish.

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