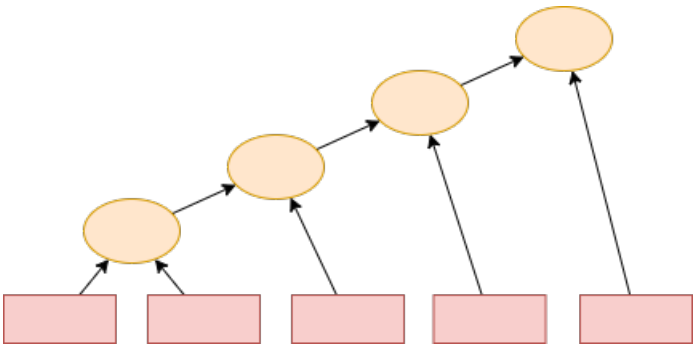
# Exam 2018

June 19, 2023

**Exercise 0.1.** Consider a left-deep process tree corresponding to a query, where each internal node is a join, and every leaf a data source (e.g., relational table). Knowing that the tree contains 9 nodes (including leaves), the system has infinite parallelism capacity in pipelining mode (no other kind of parallelism is available), which is the occupancy if the overall cost of the query is 4 seconds? Explicit any assumption you need to make.

The tree is the following:



Then, there are 4 operators, so the occupancy is

$$O = \frac{T}{N} = \frac{4s}{4op} = 1s/op.$$

There are no stalls since we assumed an infinite parallelism capacity for pipelining.

**Exercise 0.2.** Consider an HDFS cluster with 100 data nodes, without replication. If I upload a file with 10 chunks and 10 blocks each, answer the following questions and briefly justify your answer:

1. Which is the maximum number of machines containing data?

   10, since the unit of distribution is the chunk and there is no replication.

2. Which is the probability of the maximum number of machines actually contain data?

   The first chunk can land on any machine, the second one can land only on 99, the third one in 98,... so

$$P = \frac{100}{100} \times \frac{99}{100} \times \frac{98}{100} \times \frac{97}{100} \times \frac{96}{100} \times \frac{95}{100} \times \frac{94}{100} \times \frac{92}{100} \times \frac{91}{100} = 0.6282.$$

**Exercise 0.3.** Briefly explain how you would implement an intersection (i.e., T⋂S) with MapReduce. Clearly explicit which will be the key and which the value. Since it is a binary operation, assume the existence of a function "input: key -> [T|S]", as well as an operation "⊕" that concatenates attributes as needed.

$$T \cap S = \begin{cases} map\,(k,v) \mapsto & \\ [v, T \oplus k] & if\ \ (k,v) \in T \\ [v, S \oplus k] & if\ \ (k,v) \in S \\ reduce\,(k, ivs) \mapsto & \\ ivs[0,1] & if\ \ [T,S] \subset ivs[0,:] \\ \emptyset & else \end{cases}$$

**Exercise 0.4.** Let's suppose we have a log file recording the events coming from different machines. Thus, for each event we have the following information: (logID, traceID, eventID, duration)

The logID corresponds to the IP of the machine; the traceID identifies the transaction inside the machine (i.e., two traceIDs can coincide in different machines); the eventID identifies the kind of action performed by the machine; finally, the duration is the number of milliseconds taken to implement the action. Assuming that we cannot keep the pace of processing all log entries, and we decide to randomly sample them, briefly explain how would you implement the sample to bound the error of the following query: "Return the average sum of the duration per transaction".

If we sample based on the logID, we would miss entire machines, so the result could be biased if we drop slow machines, for example.

If we sample based on eventID, we would miss entirekind of actions, and we may bias the result as well.

If we sample based on (logID,eventID), we face a similar problem, because a machine could be specially slow for some actions.

If we sample based on traceID, then we would be sampling all the machines using the same dropping structure, which may bias the result, specially if we run the same workloads in different machines.

If we sample based on (logID,traceID), our sampling would in principle be independent inside a machine, since the traceID is non repeating inside a machine. Also, it would be independent across machines. Moreover, it does no harm the eventID randomness. I would go for this one.