CS425 MP4 Report

Taipeng Liu(taipeng2) & Xiang Li(xiangl14)

**1. Design**

**Algorithm:** Our “MapleJuice” system is a MapReduce-like parallel computing framework, which is built on top of reliable Distributed File System and full-membership failure-detection module. We implement RM-AM model, a cluster will have one server working as the global resource manager (RM), in charge of Maple & Juice task scheduling and tracking. Specifically, we pick the same server as namenode in SDFS. All datanodes will act as application manager (AM), finishing tasks and submitting results to SDFS for next phase. We use RPC to execute each task from RM to AMs.

In Maple phase, we use Go channel to schedule tasks. In Juice phase, We realize hash & range partition based on the available working nodes’ number, which is user-selectable. Also, user can decide whether to delete the input files.

**Inside MapleJuice**: In Maple phase, each AM will be assigned a fair amount of work (file number) to run user-specific executable, which outputs a series of key-value pairs. We’ll cache the output on local disk without put them into SDFS, which will be a huge amount of overhead. In Juice phase, each AM fetches intermediate files from local disks and append to one single result file in SDFS.

**How our MapleJuice works**: First, client/user should put their executables into SDFS, along with the dataset. Then client will RPC call RM to perform Maple function for the input data. After maple finishes, intermediate results will be stored separately on the cluster. In Juice phase, client will RPC call RM to shuffle tasks based on key and assign tasks. After all tasks are finished and returned, the client will be notified.

RM is critical since it assigns and monitors each task’s process, also records each working nodes’ task list. Also, it should handle the failure situation and reassign failed tasks. Failure could happen anytime. First, we must make sure the result remains intact. Hence, if a node fails during writing the result, we should get rid of this round and restart juice phase. Besides, namenode will reassign failed tasks to idle nodes. Our protocol can tolerate namenode’s failure too.

**2. Measurement**