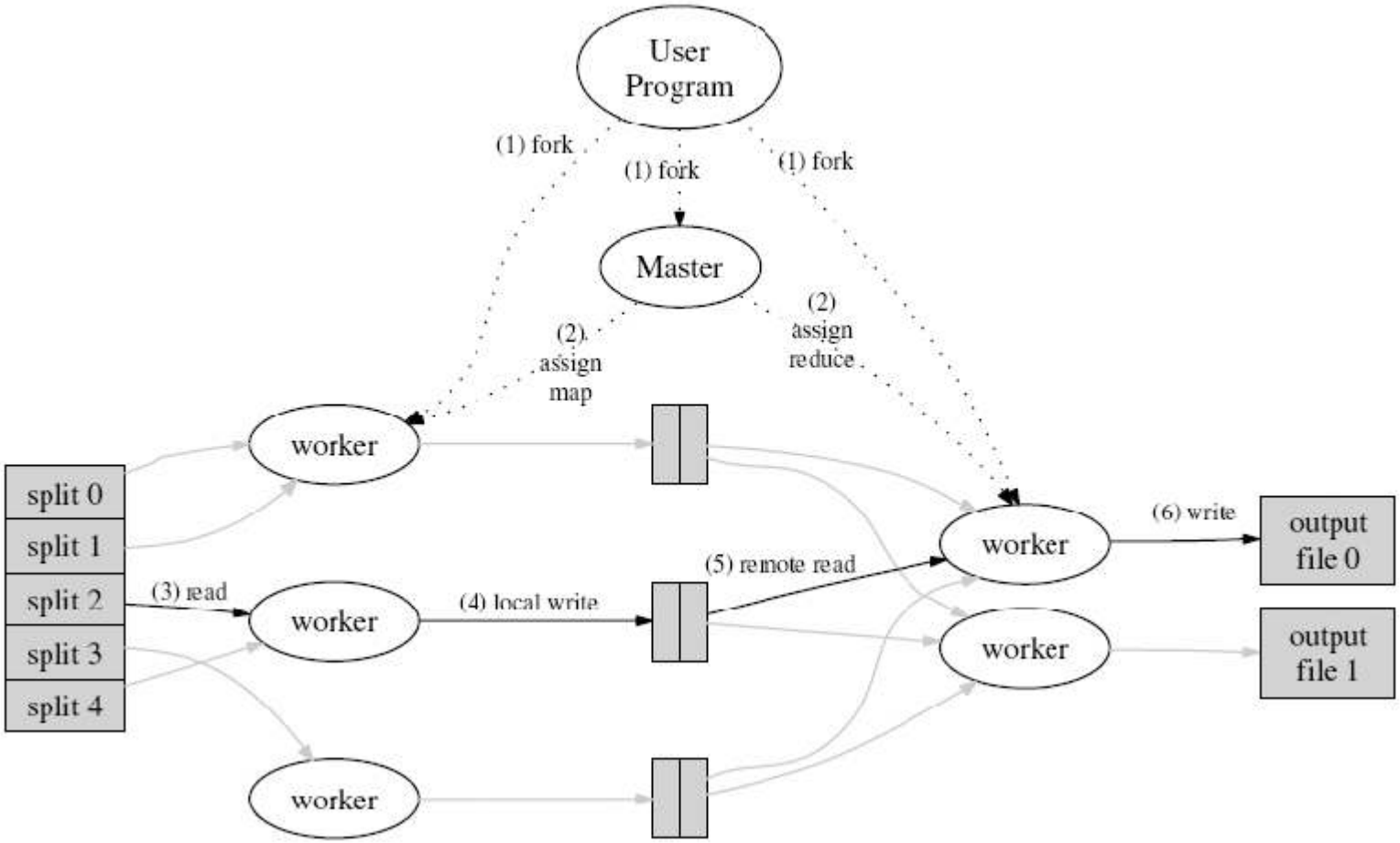
Analysis of Hadoop Online

Xin Zhan

CS 848

Recap of MapReduce



Fault Tolerance

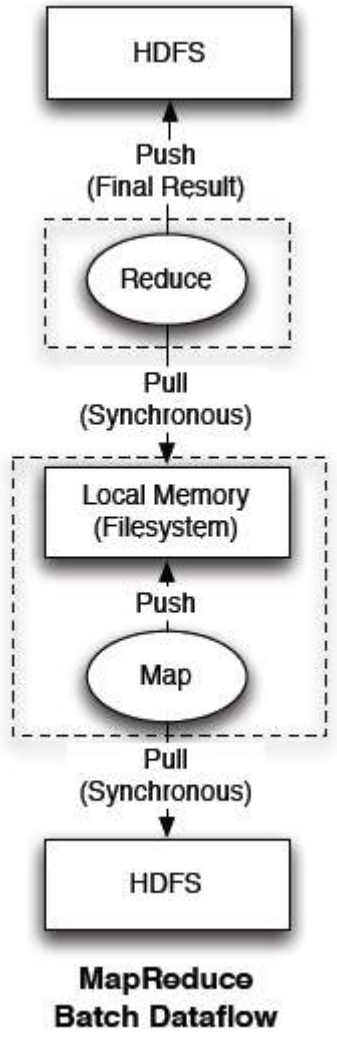
- Both map and reduce write output to disk.
- Materialization simplifies fault tolerance.
- If any task fails, JobTracker simply schedule a new task.

Life Beyond Batch Processing

- Online aggregation
- Continuous analysis of data streams

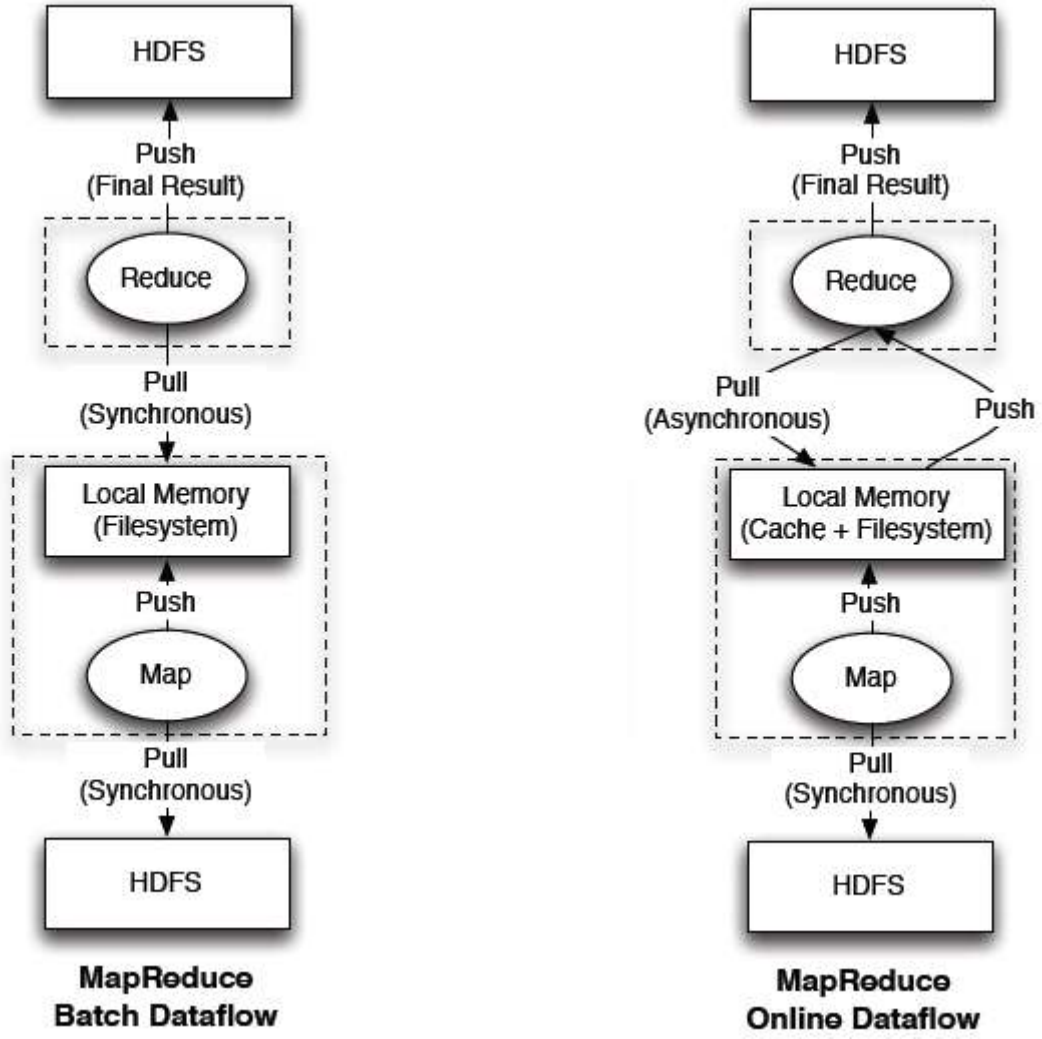
- Blocking operators → Pipelined operators

MapReduce → MapReduce Online



MapReduce Batch Dataflow

MapReduce → MapReduce Online



Tyson Condie et al., MapReduce Online, UC Berkeley, 2009

Pipelined MapReduce

- Reduce task opens a socket with map task
- Map task writes out in-memory buffer to spill fills
- Spill files are sent to pipelines, accepted by Reducers
- Once map tasks complete, reducer merge all spill files and apply user-defined function
- Reduce task can write its final output to another map task.

Fault Tolerance – Map Failures

- Reducer treats the output of pipelined map task as "tentative" until map task has committed
- Envision checkpoint mechanism

Fault Tolerance – Reduce Failure

- Mapper write complete output file to disk before committing, allows map's output to be reproduced
- Reducer is NOT blocked waiting for complete output of the map task to be written to disk
  - merge spill files generated by the same uncommitted mapper
  - won't combine them with the output of other map tasks

Online Aggregation

- Data records are incrementally sent to reducers
- Reducer can apply reduce functions to data received so far, produce snapshots
- “Progress score”

Continuous Queries

- Assume continuous reduce function depends on a suffix of history of map stream.
- Map output is already delivered to appropriate reduce tasks, shortly after it is generated.
- User-defined functions need to be invoked periodically at reducer.

Performance Studies of Hadoop Online

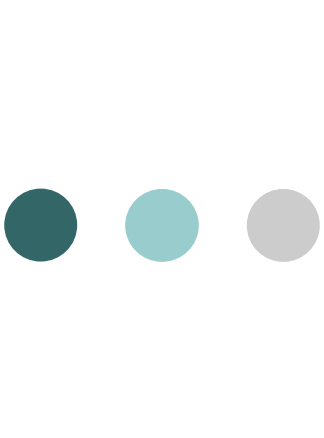
- Best case scenario studied in the paper
  - Fewer than 1 Map task assigned per node
  - Map task does not filter any data
  - No combiner function
- Interesting to see how much performance improves in less ideal settings
- Understand where the performance improvement comes from

Goal of My Experiments

- Measure performance improvements when running more Map tasks per node
- Isolate sources of improvements
  - Reducer job starts early
  - Cache vs. disk seeks/reads

Create my own image on Amazon EC2

- Modify environment setting bin/hadoop-ec2-env.sh
  - Security keys, S3 bucket, Hadoop version (hop_0.1), Base image, Arch
- Modify script image/create-hadoop-image-remote
  - Download and install Hadoop Online Prototype
- Run bin/hahoop-ec2 create-image
- Launch cluster / Terminate cluster



Thank You