MapReduce

Programming model

- Map
  - <doc-name, doc-content> => list<word, count>
- Reduce
  - <word, list<count>> => list<count>
- Partition (optional)
  - default: hash(key) mod R

Workflow

- client
- master
- input files (GFS)
- mappers
- intermediate files (local)
- reducers
- output (GFS)

Failure

- master failure: abort
- mapper failure: re-execute
  - completed: output stored locally
- reducer failure:
  - re-execute (in progress)
  - do nothing (completed, stored in GFS)

Optimizations

- combiner
  - local reducer
- backup task
  - stragglers (slow machines)
- scheduling locality
  - choose mappers close to input

Example questions: T or F

- Mappers
- Reducers
- Failure
- Optimization

Mappers 1/6

MapReduce executes different map tasks in parallel.

Mappers 1/6

MapReduce executes different map tasks in parallel.

T.

Mappers 2/6

To achieve locality, map workers always execute on the same machine as the input data that they consume.

Mappers 2/6

To achieve locality, map workers always execute on the same machine as the input data that they consume.

F. The master will try that though.

Mappers 3/6

MapReduce always schedules two instances of every task (corresponding to the GFS replicas of the input data) to guard against worker failure and stragglers.

Mappers 3/6

MapReduce always schedules two instances of every task (corresponding to the GFS replicas of the input data) to guard against worker failure and stragglers.

F.

Mappers 4/6

MapReduce guarantees that each map task is executed only once to preserve functional behavior.

Mappers 4/6

MapReduce guarantees that each map task is executed only once to preserve functional behavior.

F.

Mappers 5/6

Each map task is automatically distributed so its output is read only by a single reduce task.

Mappers 5/6

Each map task is automatically distributed so its output is read only by a single reduce task.

F. The output of a mapper is read by all.

Mappers 6/6

Intermediate data passed between the map workers and reduce workers is stored in the GFS.

Mappers 6/6

Intermediate data passed between the map workers and reduce workers is stored in the GFS.

F. Local disk on mappers.

Reducers 1/4

MapReduce may execute the same reduce computation at the same time on different machines and use the first results that are returned.

Reducers 1/4

MapReduce may execute the same reduce computation at the same time on different machines and use the first results that are returned.

T. For stragglers.

Reducers 2/4

File renaming is used to ensure that only a single execution of a reduce task is represented in the final output.

Reducers 2/4

File renaming is used to ensure that only a single execution of a reduce task is represented in the final output.

T. Write to a temporary file and do renaming.

Reducers 3/4

MapReduce places computations on machines that have the data that the computation will access locally available.

Reducers 3/4

MapReduce places computations on machines that have the data that the computation will access locally available.

F. Each reduce needs data from all map processes.

Reducers 4/4

MapReduce writes the output of each reduce computation to disks on different machines.

Reducers 4/4

MapReduce writes the output of each reduce computation to disks on different machines.

T. The output is written to GFS, which writes to multiple machines.

Failure 1/3

No single machine failure will prevent a MapReduce computation from successfully completing.

Failure 1/3

No single machine failure will prevent a MapReduce computation from successfully completing.

F. The master "wugui".

Failure 2/3

A programmer writes a map operator that has a bug that causes it to fail non-deterministically. During execution, several map tasks fail. This MapReduce job will still execute to completion.

Failure 2/3

A programmer writes a map operator that has a bug that causes it to fail non-deterministically. During execution, several map tasks fail. This MapReduce job will still execute to completion.

T.

Failure 3/3

The master incorrectly concludes that a reduce task has failed, even though it is still running (e. g., temporary network failure). The master will start another reduce task, and both tasks could complete execution of the same set of reduce operations.

Failure 3/3

The master incorrectly concludes that a reduce task has failed, even though it is still running (e. g., temporary network failure). The master will start another reduce task, and both tasks could complete execution of the same set of reduce operations.

T. GFS ensures only one write succeeds.

Performance 1/3

If there are M map tasks, using more than M workers in the map phase may still improve performance beyond that achieved with M workers.

Performance 1/3

If there are M map tasks, using more than M workers in the map phase may still improve performance beyond that achieved with M workers.

T. Stragglers/failed nodes.

Performance 2/3

If the performance of the system is not limited by disk/network throughput, then doubling the number of nodes in the system will probably improve performance if the number of map and reduce tasks is greater than the number of nodes.

Performance 2/3

If the performance of the system is not limited by disk/network throughput, then doubling the number of nodes in the system will probably improve performance if the number of map and reduce tasks is greater than the number of nodes.

T. Some tasks are waiting to be processed.

Performance 3/3

If some tasks on some workers take much longer than others to complete, and these stragglers are the bottleneck in the system, then allocating additional nodes to these straggler tasks will definitely improve performance.

Performance 3/3

If some tasks on some workers take much longer than others to complete, and these stragglers are the bottleneck in the system, then allocating additional nodes to these straggler tasks will definitely improve performance.

F. Complex computation, not because they are running on slow machines.

Questions?

Good luck!