COS 597D:
Principles of
Database and Information Systems

Distributed computing on large data sets

1

Goals

- Do one computation faster - latency
- Do more computations in given time - throughput
- Tolerate failure of 1+ machines

2

Distributing query evaluation

- Data distribution?
- Computation distribution?

- Goals
  - Keep all machines busy
  - Be able to replace badly-behaved machines seamlessly!

3

MapReduce framework

- programming model
- implementation for large clusters

- Google introduced for index building and PageRank circa 2003
  - "for processing and generating large data sets"
- The Apache Hadoop project developed open-source software

4

MapReduce Programming Model

- input set: {(input key_i, $value_i$)| $0 \le i \le$ input size}
  - user chooses type value – e.g. whole document
- output set: {(output key_i, $value_i$)| $0 \le i \le$ output size}

- Map (written by user):

(input key, value) →
{(intermed. key_j, $value_j$)| $0 \le j \le$ Map result size}

- system groups all Map output pairs for input set by intermediate key (shuffle phase)
  - gathers by intermediate key value
  - supply to Reduce by iterator

- Reduce (written by user) process intermediate values:

(intermed. key, list of values) → (output key, value)

5

MapReduce for building inverted index

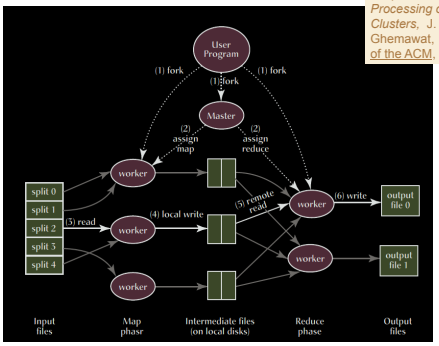
- Input pair: (docID, contents of doc)
- Map: produce {(term, docID)} for each term appearing in docID
- Input to Reduce: list of all (term, docID) pairs for one term
- Output of Reduce: (term, sorted list of docIDs containing that term)
  - postings list!

keys

6

Diagram of computation distribution

MapReduce: Simplified Data Processing on Large Clusters, J. Dean; S. Ghemawat, Comm. of the ACM, (2008)



MapReduce parallelism

- Map phase and shuffle phase may overlap
- Shuffle phase and reduce phase may overlap
- Map phase must finish before reduce phase starts
  - reduce depends on all values associated with a given key

8

MapReduce Fault Tolerance

- Master fails => restart whole computation
- Worker node fails
  - Master detects failure
  - must redo all Map tasks assigned to worker
    - output of completed Map tasks on failed worker's disk
  - for failed Map worker, Master
    - reschedules each Map task
    - notifies reducer workers of change in input location
  - for failed Reduce worker, Master
    - reschedules each Reduce task
  - rescheduling occurs as live workers become available

10

Communication Cost

Rajaraman, Leskovec and Ullman

- Model: algorithm is acyclic network of tasks
  - cascaded MapReduce tasks
- Communication cost task = size of input
  - bytes versus tuples
- Commun. cost alg. = sum of cost of all tasks
- Captures
  - Network cost
  - Disk cost
- Why not output cost?
  - input to another algorithm
  - output "rarely large"

11

Remarks

- Google built on large collections of inexpensive "commodity PCs"
  - always some not functioning
- Solve fault-tolerance problem in software
  - redundancy & flexibility NOT special-purpose hardware
- Keep machines relative generalists
  - machine becomes free ⇒
    assign to any one of set of tasks

13