# Distributed computing: index building and use

### Goals

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

## **Distributing computations**

Ideas?

- ⇒ Finding results for a query?
- Building index?

# **Distributed Query Evaluation**

2

- Assign different queries to different machines
- Break up lexicon: assign different index terms to different machines?
   – good/bad consequences?
- Break up postings lists: Assign different documents to different machines?
   – good/bad consequences?
- Goals
  - Keep all machines busy
  - Be able to replace badly-behaved machines seamlessly!

# Google query evaluation circa 2002

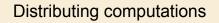
#### Parallelize computation

- distribute documents randomly to pieces of index
  - Pool of machines for each choose one
  - Why random?
- · Load balancing and reliability
  - Scheduler machines
    - assign tasks to pools of machines

monitor performance

#### Google Query Evaluation: Details circa 2002

- Enter query -> DNS-based directed to one of geographically distributed clusters
  - Load balance & fault tolerance
    Round-trip time
- w/in cluster, query directed to 1 Google Web Server (GWS)
- Load balance & fault tolerance
- GWS distributes query to pools of machines
   Load sharing
- Query directed to 1 machine w/in each pool
   Load balance & fault tolerance



#### Ideas?

- ✓ Finding results for a query?
- $\Rightarrow$  Building index?

### **Distributed Index Building**

- Can easily assign different documents to different machines
- Efficient?
- Goals
  - Keep all machines busy
  - Be able to replace badly-behaved machines seamlessly!

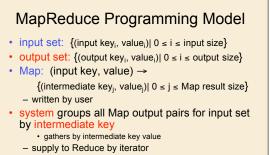


- MapReduce
  - programming model
  - implementation for large clusters
- "for processing and generating large data sets"

#### · Example applications

- \* inverted index
- graph structure of Web docs.

statistics on queries in given time period



 Reduce: (intermediate key, list of values) → (intermediate key, {result values})
 written by user to process intermediate values

# 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 11

# Diagram of computation distribution

### See Figure 1 in

#### MapReduce:

Simplified Data Processing on Large Clusters J. Dean and S. Ghemawat,

Comm. of the ACM,vol. 51, no. 1 (2008), pp. 107-113.

12

# 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