CMU SCS

15-826: Multimedia Databases and Data Mining

Extra: intro to hadoop

C. Faloutsos

CMU SCS

Resources

- Software
  - http://hadoop.apache.org/
- Map/reduce paper [Dean & Ghemawat]
  - http://research.google.com/archive/mapreduce.html
- Tutorial: see part1, foils #9-20 from
  - videolectures.net/site/normal_dl/tag=75554/kdd2010_papadimitriou_sun_yan_lsdm_01.pdf
  - videolectures.net/kdd2010_papadimitriou_sun_yan_lsdm/

CMU SCS

Motivation: Scalability

- Google: > 450,000 processors in clusters of ~2000 processors each [Barroso, Dean, Hölzle, *"Web Search for a Planet: The Google Cluster Architecture"* IEEE Micro 2003]
- Yahoo: 5Pb of data [Fayyad, KDD'07]
- Problem: **machine failures**, on a daily basis
- How to parallelize data mining tasks, then?
- A: map/reduce – hadoop (open-source clone) http://hadoop.apache.org/

CMU SCS

details

2' intro to hadoop

- master-slave architecture; n-way replication (default n=3)
- 'group by' of SQL (in parallel, fault-tolerant way)
- e.g, find histogram of word frequency
  - compute local histograms
  - then merge into global histogram

select course-id, count(*)
from ENROLLMENT
group by course-id

CMU SCS

details

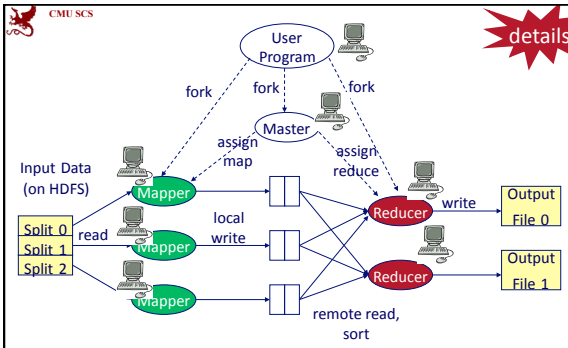
2' intro to hadoop

- master-slave architecture; n-way replication (default n=3)
- 'group by' of SQL (in parallel, fault-tolerant way)
- e.g, find histogram of word frequency
  - compute local histograms
  - then merge into global histogram

select course-id, count(*) reduce
from ENROLLMENT
group by course-id map

CMU SCS

details



By default: 3-way replication;
Late/dead machines: ignored, **transparently** (!)

CMU SCS

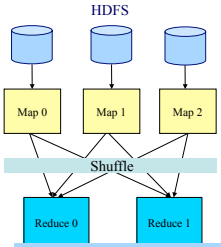
More details:

- (thanks to U Kang for the animations)

CMU SCS

Background: MapReduce

- MapReduce/Hadoop Framework



HDFS: fault tolerant, scalable, distributed storage system

Mapper: read data from HDFS, output (k,v) pair

Output sorted by the key

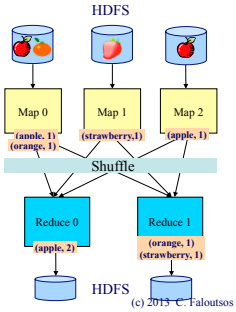
Reducer: read output from mappers, output a new (k,v) pair

Programmers need to provide only map() and reduce() functions

CMU SCS

Background: MapReduce

- Example: histogram of fruit names




```
map( fruit ) {
output(fruit, 1);
}

reduce( fruit, v[1..n] ) {
for(i=1; i <=n; i++)
sum = sum + v[i];
output(fruit, sum);
}
```

CMU SCS

Conclusions

- Convenient mechanism to process Tera- and Peta-bytes of data, on >hundreds of machines
- User provides `map()` and `reduce()` functions
- Hadoop handles the rest (eg., machine failures etc)