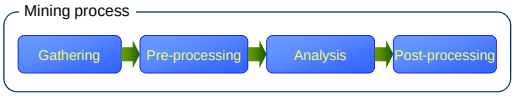
DisCo

A Case Study on Distributed Mining

Spiros Papadimitriou
Jimeng Sun

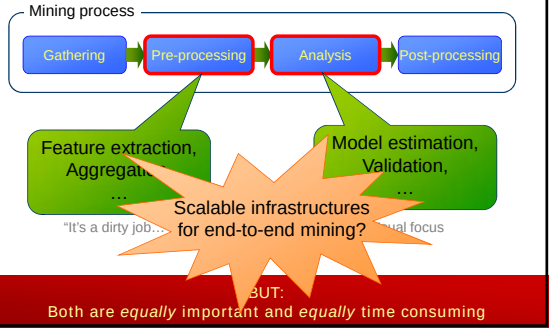
IBM T.J. Watson Research Center

Data processing and mining

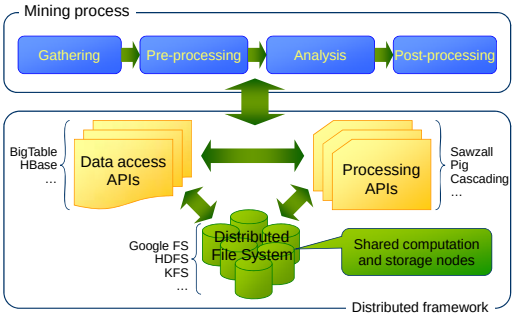


2

Data processing and mining



Data processing and mining



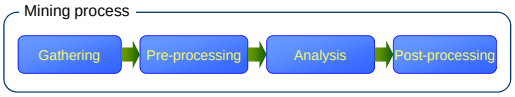
4

Overview

- Background — MapReduce
- Feature extraction — Aggregates & edgelists
- Mining — Co-clustering
- Conclusion

5

Overview



- Background: MapReduce
- Feature extraction
- Mining
- Conclusion

6

Example – Programming model

employees.txt

# LAST	FIRST	SALARY
Smith	John	$90,000
Brown	David	$70,000
Johnson	George	$95,000
Yates	John	$80,000
Miller	Bill	$65,000
Moore	Jack	$85,000
Taylor	Fred	$75,000
Smith	David	$80,000
Harris	John	$90,000
...	...	...
...	...	...

mapper

```
def getName (line):
   return  line.split('\t')[1]
```

reducer

```
def addCounts (hist,  name):
   hist[name] = \
   hist.get(name,default=0) + 1
   return hist

input = open('employees.txt', 'r')
intermediate = map(getName, input)
result = reduce(addCounts, \
                intermediate, {})
```

Q: "What is the frequency of each first name?"

Three (fixed) "streams": key-value iterators

Comparison

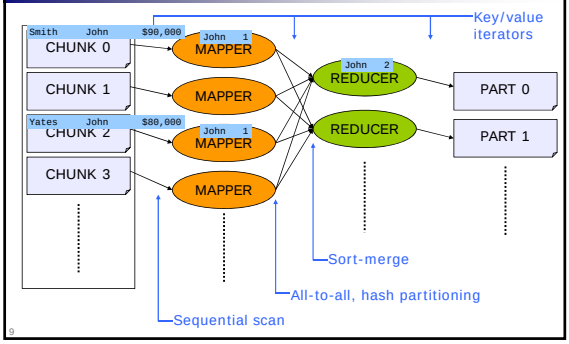
Quick-n-dirty script	vs	MapReduce (Hadoop)
~5 lines of (non-boilerplate) code		
Single machine, local drive		Up to *thousands* of machines and drives

What is hidden to achieve this:

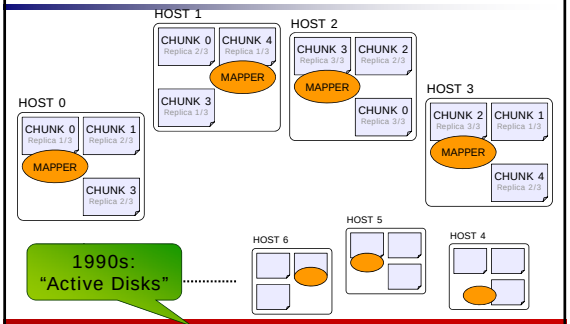
- Data partitioning, placement and replication
- Computation placement (and replication)
- Number of nodes (mappers / reducers)

As a user, you don't *need* to know what I'm about to show you next...

Execution model: Flow

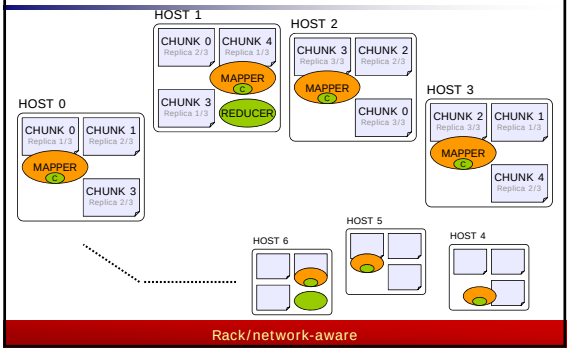


Execution model: Placement



Computation *co-located* with data (as much as possible)

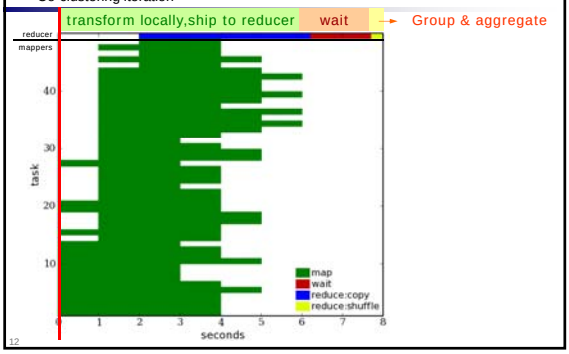
Execution model: Placement



Rack/network-aware

Execution timeline

Co-clustering iteration

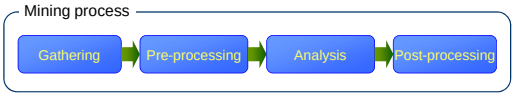


Hadoop

- Open-source implementation (ASF/Yahoo!)
- Main Hadoop components:
  - HDFS: Clone of Google FS
  - MapReduce
  - HTable: Clone of BigTable
  - Hadoop On Demand (HOD)

13

Overview



- Background
- Feature extraction
- Mining
- Conclusion

14

Histogram benchmark

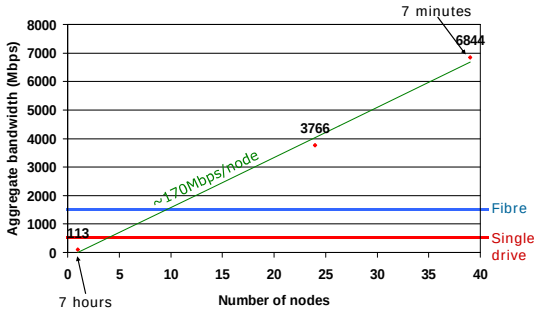
- 347.5GB raw data input (text)
- ~30KB total reducer output

- Up to 39 quad-core nodes
- …see paper for setup details

15

Scalability

Simple benchmark



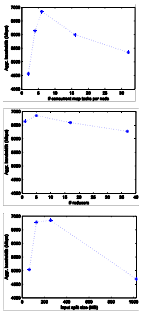
16

Caveat: cluster is running a single job

Parameter tuning study

- Process pool size
- Number of reducers
- Input file split size

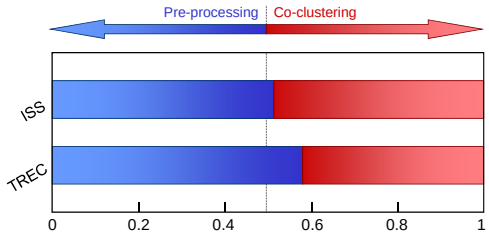
- …see paper



17

Pre-processing vs. co-clustering

Edgelist extraction, 39 nodes



… both are *equally* important and *equally* time consuming

Overview

Mining process

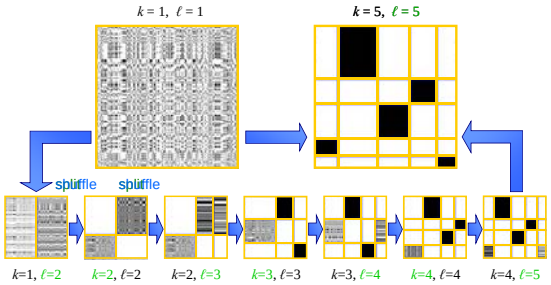
Gathering → Pre-processing → Analysis → Post-processing

- Background
- Feature extraction
- Mining: (co-)clustering
- Conclusion

19

Co-clustering

Summary

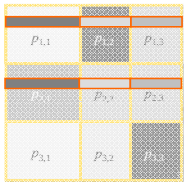


Split: Increase k or ℓ

Shuffle: Rearrange rows and cols

Search for solution

Shuffles

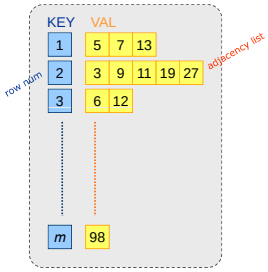


Similarity (KL-divergence) of row fragments to blocks of a row group

Assign to second row-group

21

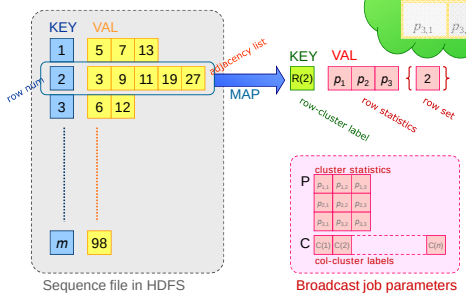
(Co-)clustering with MapReduce



Sequence file in HDFS

22

(Co-)clustering with MapR

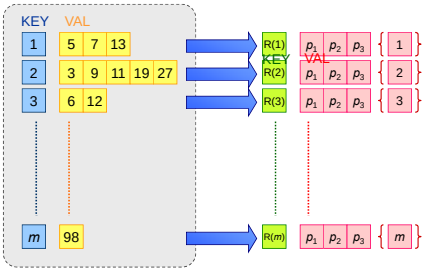


Sequence file in HDFS

Broadcast job parameters

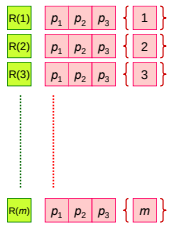
23

(Co-)clustering with MapReduce



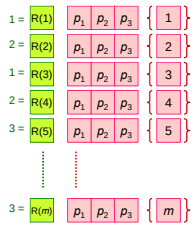
24

(Co-)clustering with MapReduce



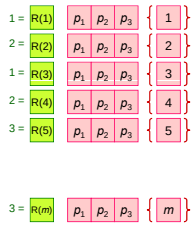
25

(Co-)clustering with MapReduce



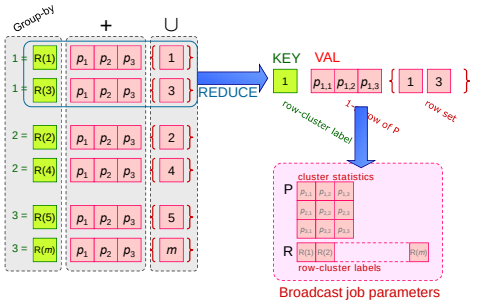
26

(Co-)clustering with MapReduce



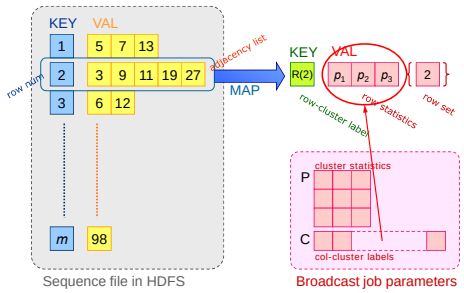
27

(Co-)clustering with MapReduce



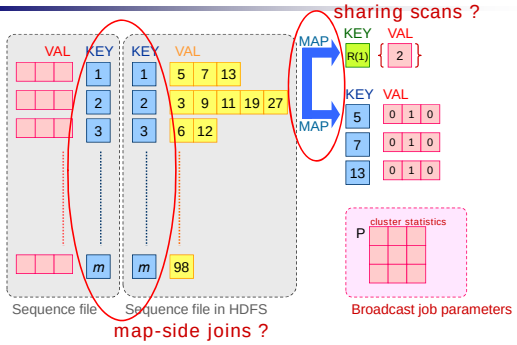
28

(Co-)clustering with MapReduce



Scales w.r.t. number of *edges* – what about nodes ?

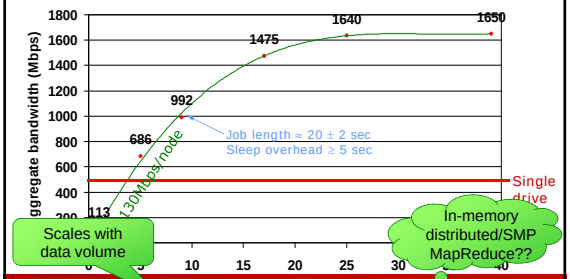
Co-clustering with MapReduce



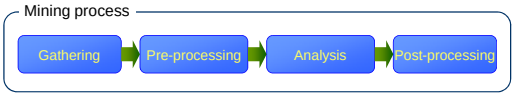
30

(Co-)clustering
Scalability

Raw data: ~100GB (knee beyond 39 nodes)
Binary edgelist: ~4GB (knee ~15 nodes)



Overview



- Background
- Feature extraction
- Mining
- Conclusion

32

Other work
Hadoop/MapReduce "ecosystem"

- Lower-level (storage/virtualization):
  - Kosmos FS (KFS), Amazon S3, Parascale VSN, Ceph, Lustre/Gluster, PanFS
  - Amazon EC2
- Higher-level (data access):
  - BigTable, HBase, Hypertable
- Higher-level (computation):
  - PIG, Cascading, Sawzall, Dryad/DryadLINQ

33

Other work
Distributed/Parallel Mining

- Sector
- FREERIDE-G
- MapReduce for ML on multi-core [NIPS06]

34

Conclusions and lessons
MapReduce

- Simple programming model
- Part of a growing, open "ecosystem" of data processing tools
- Scalable and fault-tolerant
  - Scalability ≠ performance
- Ideal for (pre-)processing large volumes of data

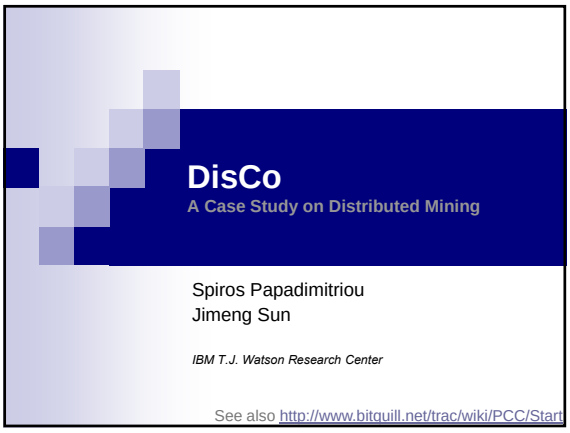
'[…] MapReduce is the first instruction of the "data center computer" […]'
– David Patterson,
"*The Data Center Is The Computer*",
CACM, Jan. 2008

35

Conclusions and lessons
Distributed mining process

- Data pre-processing *and* mining
  - End-to-end view
  - Both equally important and time-consuming
- How big the data *really* is depends on the task at hand (cf. [NIPS06])

36


DisCo
A Case Study on Distributed Mining
Spiros Papadimitriou
Jimeng Sun
IBM T.J. Watson Research Center
See also http://www.bitquill.net/trac/wiki/PCC/Start