Large-scale Data Mining MapReduce and Beyond

Part 3: Applications

Spiros Papadimitriou, IBM Research

Jimeng Sun, IBM Research

Rong Yan, Facebook

Part 3: Applications

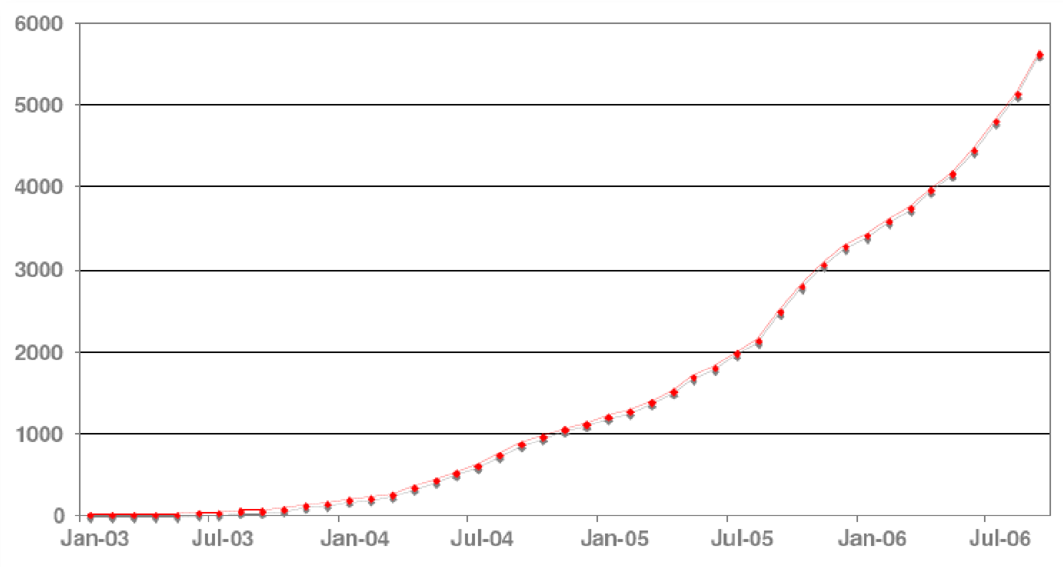
- Introduction
- Applications of MapReduce
  - Text Processing
  - Data Warehousing
  - Machine Learning
- Conclusions

MapReduce Applications in the Real World

http://www.dbms2.com/2008/08/26/known-applications-of-mapreduce/

Organizations	Application of MapReduce
Google	Wide-range applications, grep / sorting, machine learning, clustering, report extraction, graph computation
Yahoo	Data model training, Web map construction, Web log processing using Pig, and much, much more
Amazon	Build product search indices
Facebook	Web log processing via both MapReduce and Hive
PowerSet (Microsoft)	HBase for natural language search
Twitter	Web log processing using Pig
New York Times	Large-scale image conversion
…	…
Others (>74)	Details in http://wiki.apache.org/hadoop/PoweredBy (so far, the longest list of applications for MapReduce)

Growth of MapReduce Applications in Google
[Dean, PACT'06 Keynote]



Growth of MapReduce Programs
in Google Source Tree (2003 – 2006)
(Implemented as C++ library)

Example Use

Distributed grep

Distributed sort

Term-vector per host

Document clustering

Web access log stat

Web link reversal

Inverted index

Statistical translation

Red: discussed in part 2

MapReduce Goes Big: More Examples

- **Google**: >100,000 jobs submitted, 20PB data processed per day
  - Anyone can process tera-bytes of data w/o difficulties
- **Yahoo**: >100,000 CPUs in >25,000 computers running Hadoop
  - Biggest cluster: 4000 nodes (2*4 CPUs with 4*1TB disk)
  - Support research for Ad system and web search
- **Facebook**: 600 nodes with 4800 cores and ~2PB storage
  - Store internal logs and dimension user data

User Experience on MapReduce

Simplicity, Fault-Tolerance and Scalability

Google: "completely rewrote the production indexing system using MapReduce in 2004" [Dean, OSDI' 2004]

- **Simpler** code (Reduce 3800 C++ lines to 700)
- MapReduce handles **failures** and **slow machines**
- Easy to **speedup** indexing by adding more machines

Nutch: "convert major algorithms to MapReduce implementation in 2 weeks" [Cutting, Yahoo!, 2005]

- Before: several **undistributed** scalability bottlenecks, impractical to manage collections **>100M** pages
- After: the system becomes scalable, distributed, easy to operate; it permits **multi-billion** page collections

MapReduce in Academic Papers

http://atbrox.com/2009/10/01/mapreduce-and-hadoop-academic-papers/

- 981 papers cite the first MapReduce paper [Dean & Ghemawat, OSDI'04]
  - Category: **Algorithmic**, cloud overview, infrastructure, future work
  - Company: Internet (Google, Microsoft, Yahoo ..), IT (HP, IBM, Intel) University: CMU, U. Penn, UC. Berkeley, UCF, U. of Missouri, …
- >10 research areas covered by algorithmic papers
  - Indexing & Parsing, Machine Translation
  - Information Extraction, Spam & Malware Detection
  - Ads analysis, Search Query Analysis
  - Image & Video Processing, Networking
  - Simulation, Graphs, Statistics, …
- 3 categories for MapReduce applications
  - Text processing: tokenization and indexing
  - Data warehousing: managing and querying structured data
  - Machine learning: learning and predicting data patterns

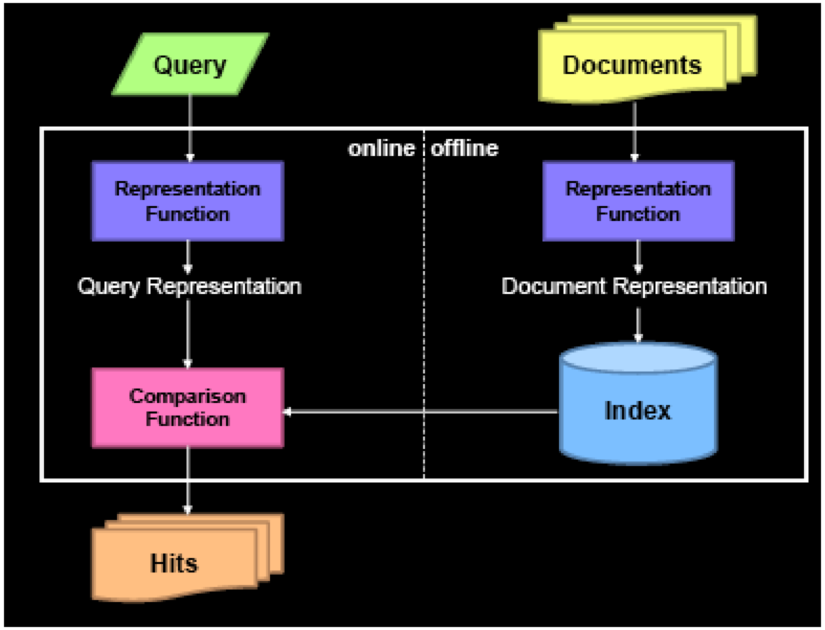
Outline

- Introduction
- Applications
  - **Text indexing and retrieval**
  - Data warehousing
  - Machine learning
- Conclusions

Text Indexing and Retrieval: Overview

[Lin & Dryer, Tutorial at NAACL/HLT 2009]

- Two stages: offline indexing and online retrieval
- Retrieval: sort documents by likelihood of documents
  - Estimate relevance between docs and queries
  - Sort and display documents by relevance
- Standard model: vector space model with TF.IDF weighting
  - Indexing: represent docs and queries as weight vectors



Similarity w. Inner Products

$$sim(q, d_i) = \sum_{t \in V} w_{t,d_i} w_{t,q}$$

TF.IDF indexing

$$w_{i,j} = tf_{i,j} \cdot \log \frac{N}{n_i}$$

MapReduce for Text Retrieval?

- Stage 1: Indexing problem
  - No requirement for real-time processing
  - Scalability and incremental u rtant

Suitable for MapR

Most popular MapReduce application

- Stage 2: Retrieval problem
  - Require sub-second response to que
  - Only few retrieval results are needed

Not ideal for MapReduce

Inverted Index for Text Retrieval

[Lin & Dryer, Tutorial at NAACL/HLT 2009]

	tf 1	2	3	4	*idf*
complicated			5	2	0.301
contaminated	4	1	3		0.125
fallout	5		4	3	0.125
information	6	3	3	2	0.000
interesting		1			0.602
nuclear	3		7		0.301
retrieval		6	1	4	0.125
siberia	2				0.602

Doc 1 ▪ ▪ Doc 4

11

Indexing Construction using MapReduce

More details in Part 1 & 2

- Map over documents on each node to collect statistics
  - Emit term as keys, (docid, tf) as values
  - Emit other meta-data as necessary (e.g., term position)

- Reduce to aggregate doc. statistics across nodes
  - Each value represents a posting for a given key
  - Sort the posting at the end (e.g., based on docid)

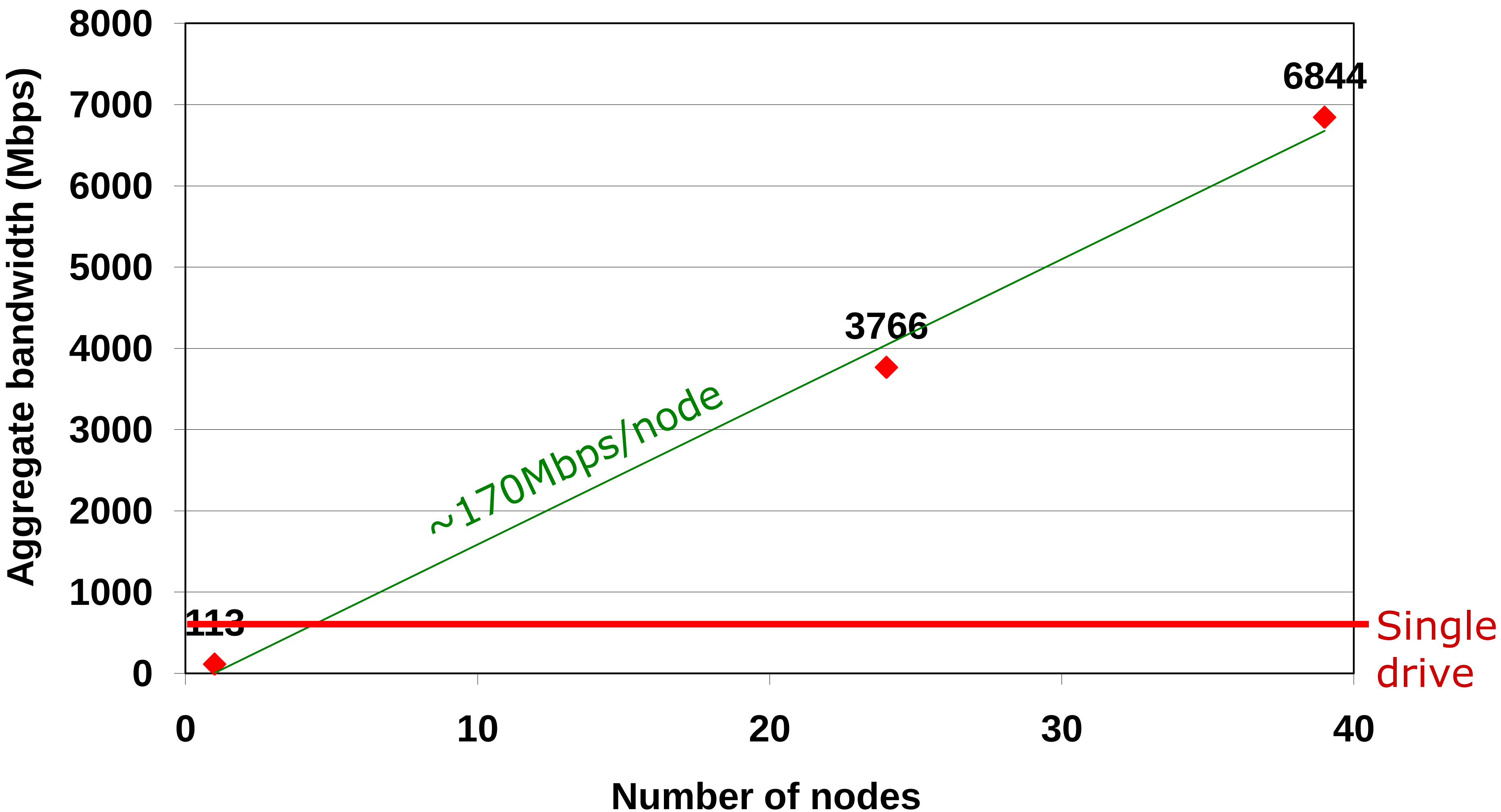
- MapReduce will do all the heavy lifting
  - Typically postings cannot be fit in memory of a single node

Example: Simple Indexing Benchmark

- Node configuration: 1, 24 and 39 nodes
  - 347.5GB raw log indexing input
  - ~30KB total combiner output
  - Dual-CPU, dual-core machines
  - Variety of local drives (ATA-100 to SAS)

- Hadoop configuration
  - 64MB HDFS block size (default)
  - 64-256MB MapReduce chunk size
  - 6 (= # cores + 2) tasks per task-tracker
  - Increased buffer and thread pool sizes

Scalability: Aggregate Bandwidth



Caveat: cluster is running a single job

Nutch: MapReduce-based Web-scale search engine

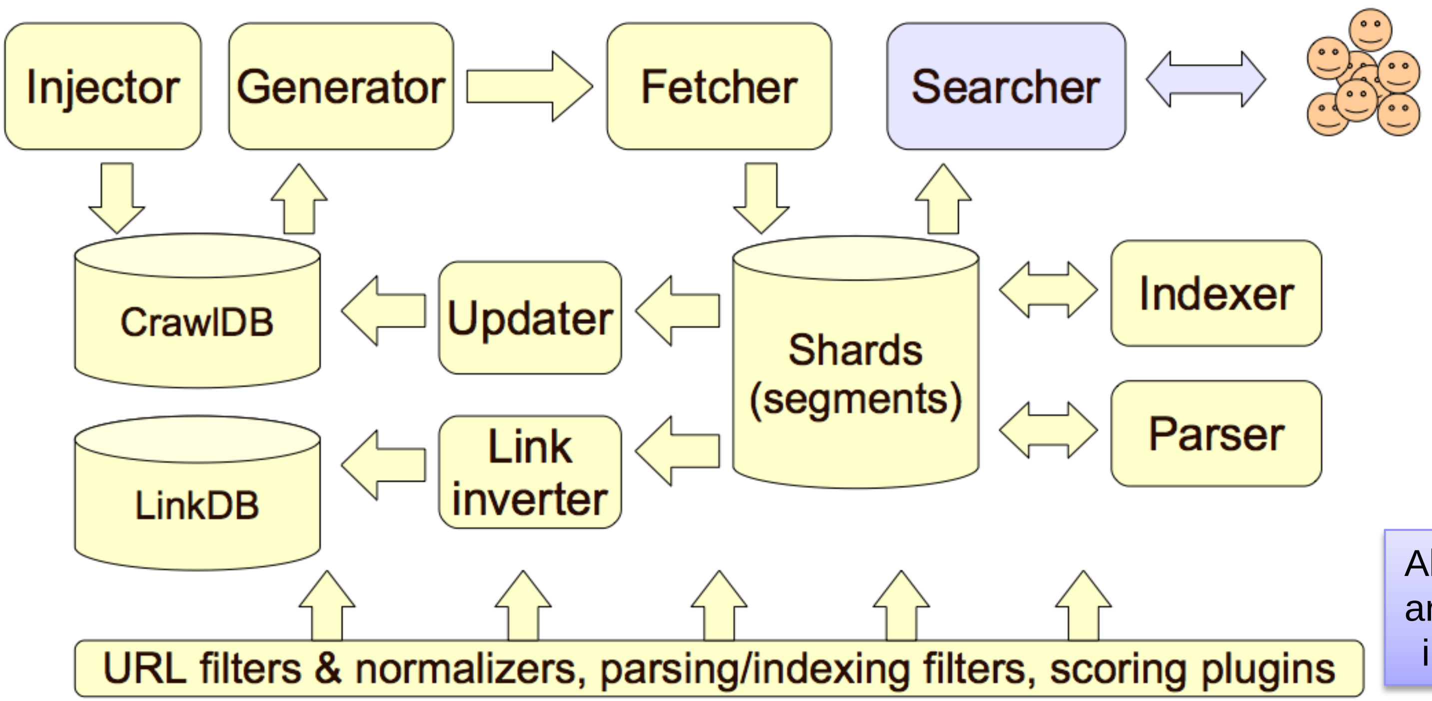
Official site: http://lucene.apache.org/nutch/

- Doug Cutting, the creator of Hadoop, and Mike Cafarella founded in 2003
  - Map-Reduce / DFS → Hadoop
  - Content type detection → Tika
- Many installations in operation
  - >48 sites listed in Nutch wiki
  - Mostly vertical search
- Scalable to the entire web
  - Collections can contain **1M – 200M** documents, webpages on millions of different servers, billions of pages
  - Complete crawl takes weeks
  - State-of-the-art search quality
  - Thousands of searches per second

Nutch Building Blocks: MapReduce Foundation

[Bialecki, ApacheCon 2009]

- **MapReduce**: central to the Nutch algorithms
  - Processing tasks are executed as one or more MapReduce jobs
- Data maintained as Hadoop SequenceFiles
  - Massive updates very efficient, small updates costly



All yellow boxes are implemented in MapReduce

Nutch in Practice

- Convert major algorithms to MapReduce in 2 weeks
- Scale from tens-million pages to multi-billion pages

Doug Cutting, Founder of Hadoop / Nutch

- A scale-out system, e.g., Nutch/Lucene, could achieve a performance level on a cluster of blades that was not achievable on any scale-up computers, e.g., the Power5

Michael et al., IBM Research, IPDPS'07

Part 3: Applications

- Introduction
- Applications of MapReduce
  - Text Processing
  - **Data Warehousing**
  - Machine Learning
- Conclusions

Why use MapReduce for Data Warehouse?

- The amount of data you need to store, manage, and analyze is growing relentlessly
  - Facebook: >1PB raw data managed in database today
- Traditional data warehouses struggle to keep pace with this data explosion, also analytic depth and performance.
  - Difficult to scale to more than PB of data and thousands of nodes
  - Data mining can involve very high-dimensional problems with super-sparse tables, inverted indexes and graphs
- MapReduce: highly parallel data warehousing solution
  - AsterData SQL-MapReduce: up to 1PB on commodity hardware
  - Increases query performance by >9x over SQL-only systems

Status quo: Data Warehouse + MapReduce

Available MapReduce Software for Data Warehouse

- **Open Source:** Hive (http://wiki.apache.org/hadoop/Hive)
- **Commercial:** AsterData (SQL-MR), Greenplum
- **Coming:** Teradata, Netezza, omr.sql (Oracle)

Huge Data Warehouses using MapReduce

- **Facebook:** multiple PBs using Hive in production
- **Hi5:** use Hive for analytics, machine learning, social analysis
- **eBay:** 6.5PB database running on Greenplum
- **Yahoo:** >PB web/network events database using Hadoop
- **MySpace:** multi-hundred terabyte databases running on Greenplum and AsterData *nCluster*

HIVE: A Hadoop Data Warehouse Platform

Offical webpage:http://hadoop.apache.org/hive, cont. from Part I

- Motivations
  - Manage and query structured data using MapReduce
  - Improve programmablitiy of MapReduce
  - Allow to publish data in well known schemas

- Key building principles:
  - MapReduce for execution, HDFS for storage
  - SQL on structured data as a familiar data warehousing tool
  - Extensibility – Types, Functions, Formats, Scripts
  - Scalability, interoperability, and performance

Simplifying Hadoop based on SQL

[Thusoo, Hive ApacheCon 2008]

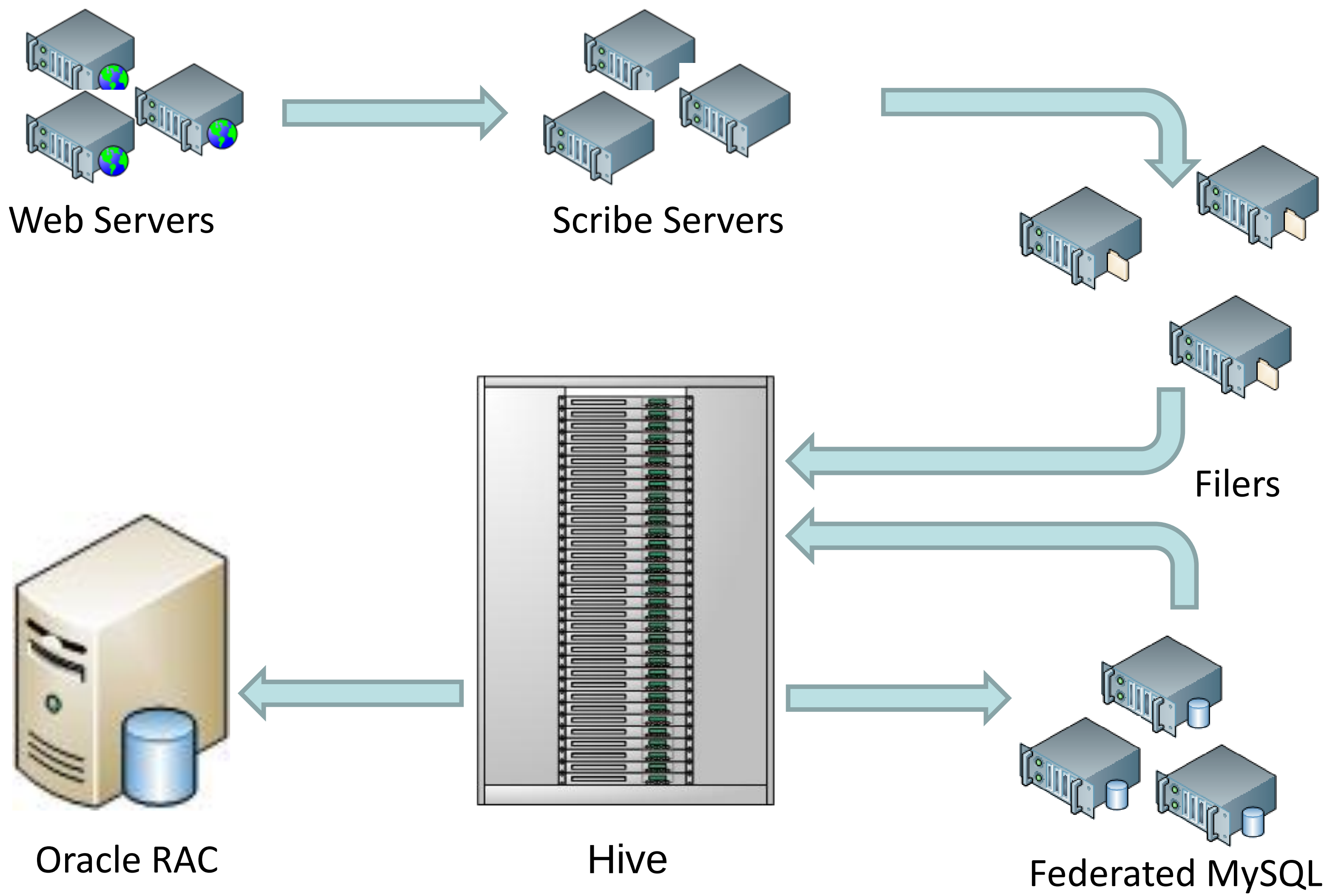
```
hive> select key, count(1) from kv1 where key > 100
   group by key;
```

vs.

```
$ cat > /tmp/reducer.sh
uniq -c | awk '{print $2"\t"$1}‘
$ cat > /tmp/map.sh
awk -F '\001' '{if($1 > 100) print $1}‘
$ bin/hadoop jar contrib/hadoop-0.19.2-dev-
   streaming.jar -input /user/hive/warehouse/kv1 -
   mapper map.sh -file /tmp/reducer.sh -file
   /tmp/map.sh -reducer reducer.sh -output
   /tmp/largekey -numReduceTasks 1
$ bin/hadoop dfs –cat /tmp/largekey/part*
```

Data Warehousing at Facebook Today

[Thusoo, Hive ApacheCon 2008]



Hive/Hadoop Usage @ Facebook

[Jain and Shao, Hadoop Summit' 09]

- Types of Applications:
  - Reporting
    - e.g. Daily/Weekly aggregations of impression/click counts
    - Complex measures of user engagement
  - Ad hoc Analysis
    - e.g. how many group admins broken down by state/country
  - Collecting training data
    - e.g. User engagement as a function of user attributes
  - Spam Detection
    - Anomalous patterns for Site Integrity
    - Application API usage patterns
  - Ad Optimization

Hadoop Usage @ Facebook

[Jain and Shao, Hadoop Summit' 09]

- Data statistics (Jun. 2009) :
  - Total Data: ~1.7PB
  - Cluster Capacity ~2.4PB
  - Net Data added/day: ~15TB
    - 6TB of uncompressed source logs
    - 4TB of uncompressed dimension data reloaded daily
  - Compression Factor ~5x (gzip, more with bzip)

- Usage statistics:
  - 3200 jobs/day with 800K tasks(map-reduce tasks)/day
  - 55TB of compressed data scanned daily
  - 15TB of compressed output data written to hdfs
  - 80 MM compute minutes/day

Thoughts: MapReduce for Database

- The strength of MapReduce is **simplicity** and **scalability**
  - No database system can come close to the performance of MapReduce infrastructure
  - RDBMSs cannot scale to that degree, not as fault-tolerant, ...
- Abstract ideas have been known before
  - “Mapreduce: A Major Step Backwards?”, DeWitt and Stonebraker
  - Implement-able using user-defined aggregates in PostgreSQL
- MapReduce is very good at what it was designed for, but it may not be the one-fits-all solution
  - E.g. joins are tricky to do: MapReduce assumes a single input

Part 3: Applications

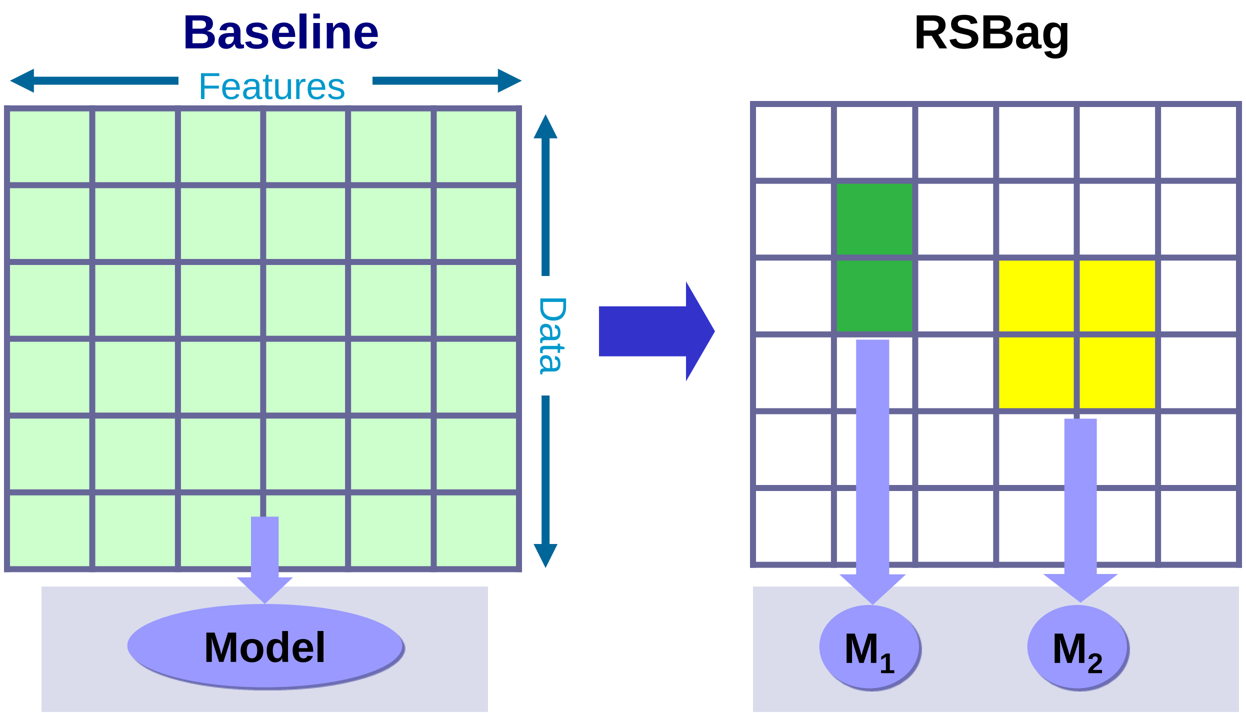
- Introduction
- Applications of MapReduce
  - Text Processing
  - Data Warehousing
  - **Machine Learning**
- Conclusions

MapReduce for Machine Learning

- MapReduce: simple parallel framework for learning
  - More difficult to parallelize machine learning algorithms using many existing parallel languages, e.g., Orca, Occam ABCL, SNOW, MPI and PARLOG
- Key observations: many learning algorithms can be written as summation forms [Chu et al., NIPS 2006]
  - Expressible as a sum over data points
  - Solvable with a small number of iterations
- This fits well with MapReduce algorithms
  - Map: distribute data points to nodes
  - Reduce: aggregate the statistics from each node

Example: Random Subspace Bagging (RSBag)

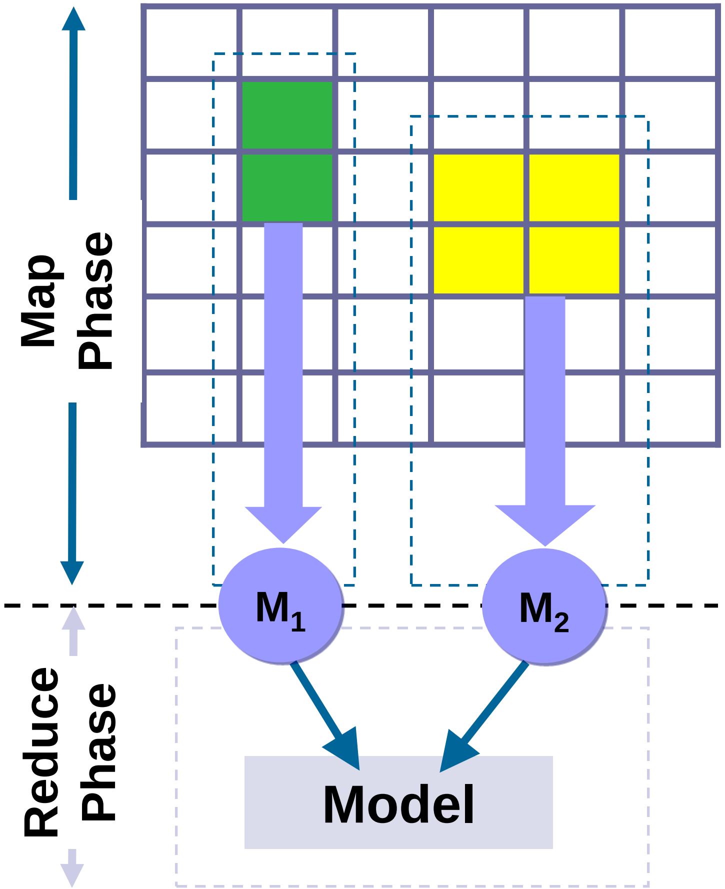
Scaling over data and feature space



- RSBag: reduce redundancy of concept models in data and feature space
  - Select multiple bags of training examples from sampled **data and feature space**
  - Learn a **base model** on each bag of data w. any classifiers, e.g. SVMs
  - Fuse them into a **composite classifier** for each concept
- Advantage: achieve similar performance with theoretical guarantee w. less learning time, recover to Random Forest w. decision trees [Breimann, 01]

MapReduce version of Random Subspace Bagging
[Yan et al., ACM Workshop on LS-MMRM'09]

- Mapping phase
  - Each task learns a SVM model based on sampled data and features
  - These tasks are independent with each other, so they can be fully distributed
- Reducing phase
  - For each concept, combine its SVM models into a composite classifier
- Advantages over other MapReduce solutions on baseline SVMs
  - RSBag is more efficient than baseline
  - RSBag naturally partitions the learning problem into multiple independent tasks, thus existing learning code is re-usable



MapReduce for Other Learning Algorithms

Favored Algorithms	Unfavored Algorithms
Naïve Bayes	Perceptron
k Nearest Neighbor	AdaBoost
kMeans / EM	Support Vector Machine
Random Bagging	Logistic Regression
Gaussian Mixture	Spectral Clustering
Linear Regression	

Bold: discussed

Few Iterations & Long Inner-Loop Cycle

Many Iterations & Short Inner-Loop Cycle

Machine Learning Applications: Examples

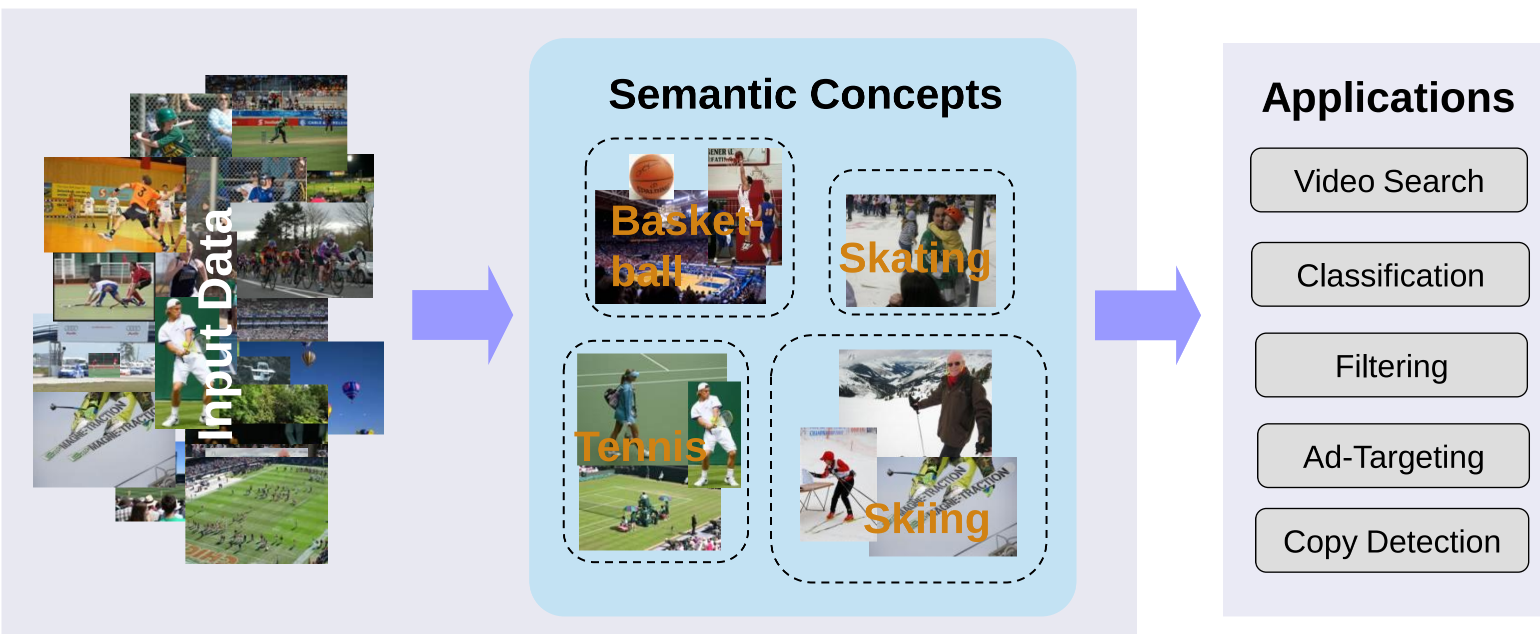
http://atbrox.com/2009/10/01/mapreduce-and-hadoop-academic-papers/

- Multimedia concept detection 1
- Machine translation 2
- Distributed co-clustering 3
- Social network analysis
- DNA sequence alignment
- Image / video clustering
- Spam & Malware Detection
- Advertisement Analysis
- ….

Application I: Multimedia Concept Detection

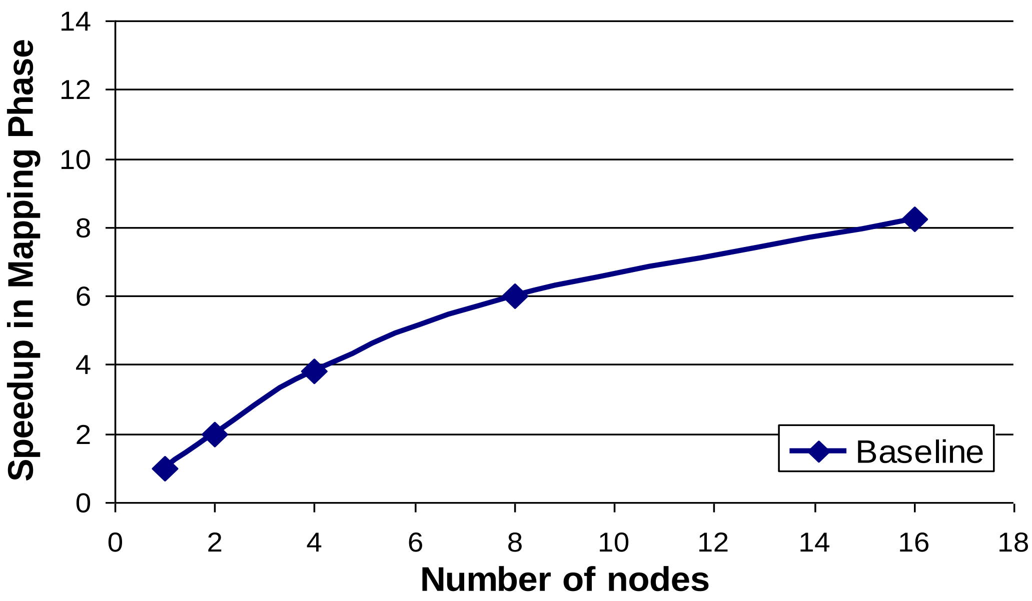
[Yan et al., ACM Workshop on LS-MMRM'09]

- Automatically categorize image / video into a list of semantic concepts using statistical learning methods
  - Foundations for several downstream use cases
- Apply **MapReduce** for multimedia concept detection
  - Learning methods: Random subspace bagging with SVMs



First Results: MapReduce-RSBag Scalability

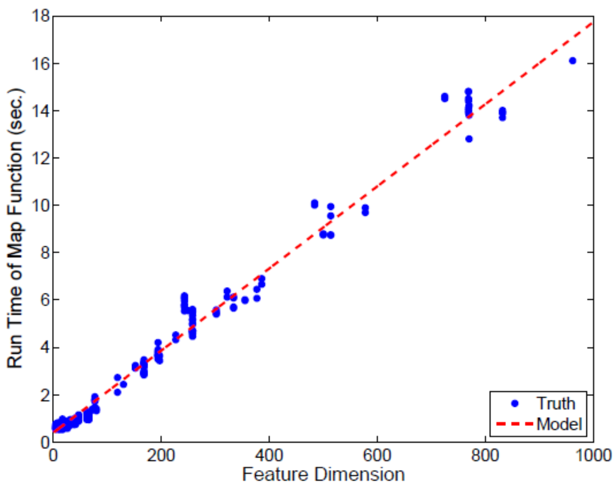
- Results: speedup in mapping phase on 1, 2, 4, 8 and 16 nodes when learning 10 semantic concepts (>100GB features)
- Linear scalability on 1 – 4 nodes, but sub-linear on > 8 nodes
  - Hypothesis: Because of higher communication cost using more nodes? No.
  - Fact: The running time of our tasks varies a lot, but MapReduce assumes each map task takes similar time, Hadoop's task scheduler is too simple.



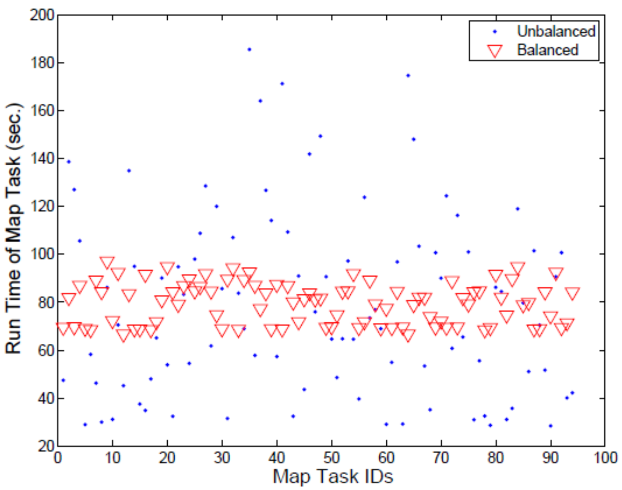
Improve Scheduling Methods for Heterogeneous Tasks

- Goal: develop more effective offline task scheduling algorithms in presence of task heterogeneity
- Task Scheduling Approaches
  - Runtime modeling: predict the running time of task based on historical data

$$T(|X|,|F|) = t_0 + |X|^{\alpha}|F|^{\beta}t_1 + \epsilon$$

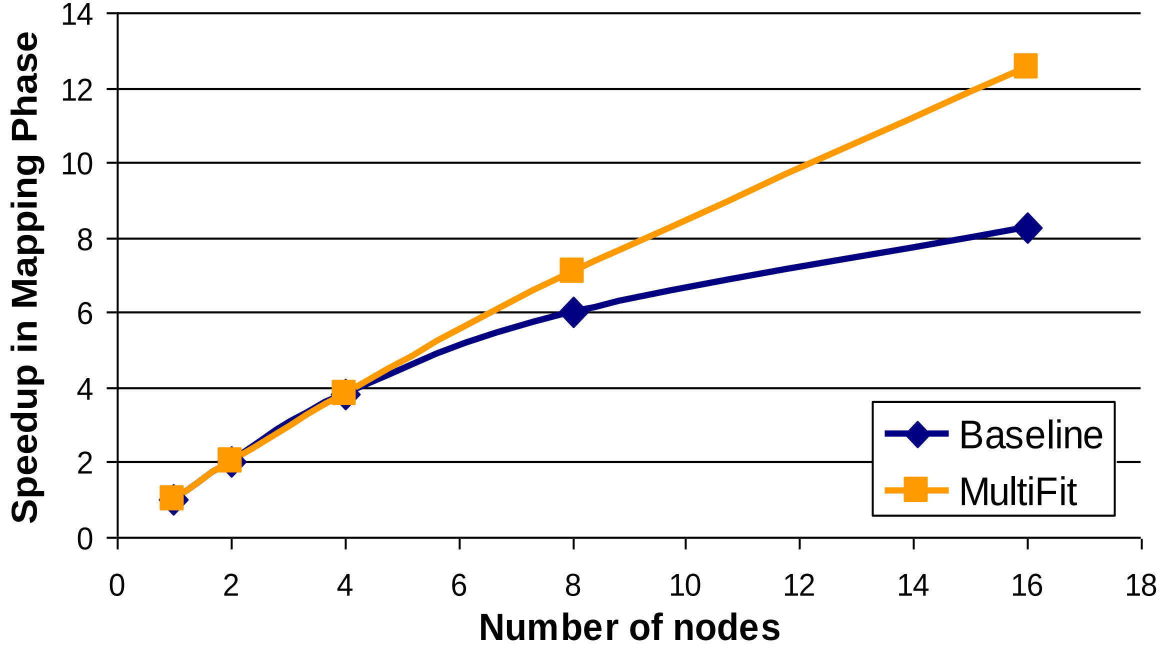


  - Formulate the task scheduling problem as a **multi-processor scheduling** problem
  - Apply the **Multi-Fit** algorithm with First-Fit-Decreasing bin packing to find the shortest time to run all the tasks using a fixed number of nodes
- Results: significantly improve the balance between multiple tasks



Scalability Results w. Improved Task Scheduling

- Results: speedup in mapping phase on 1, 2, 4, 8 and 16 nodes when learning 10 semantic concepts (>100GB)
- Achieve considerably better scalability than Hadoop baseline results

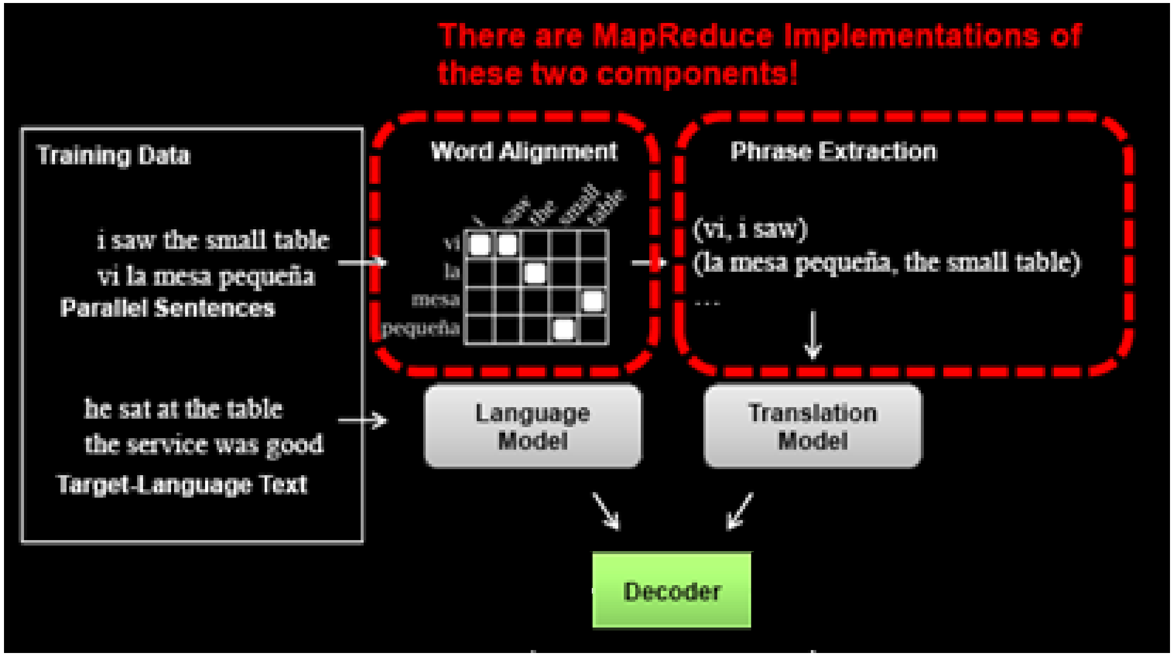


Application 2: Machine Translation

- Formulation: translate foreign f into English e

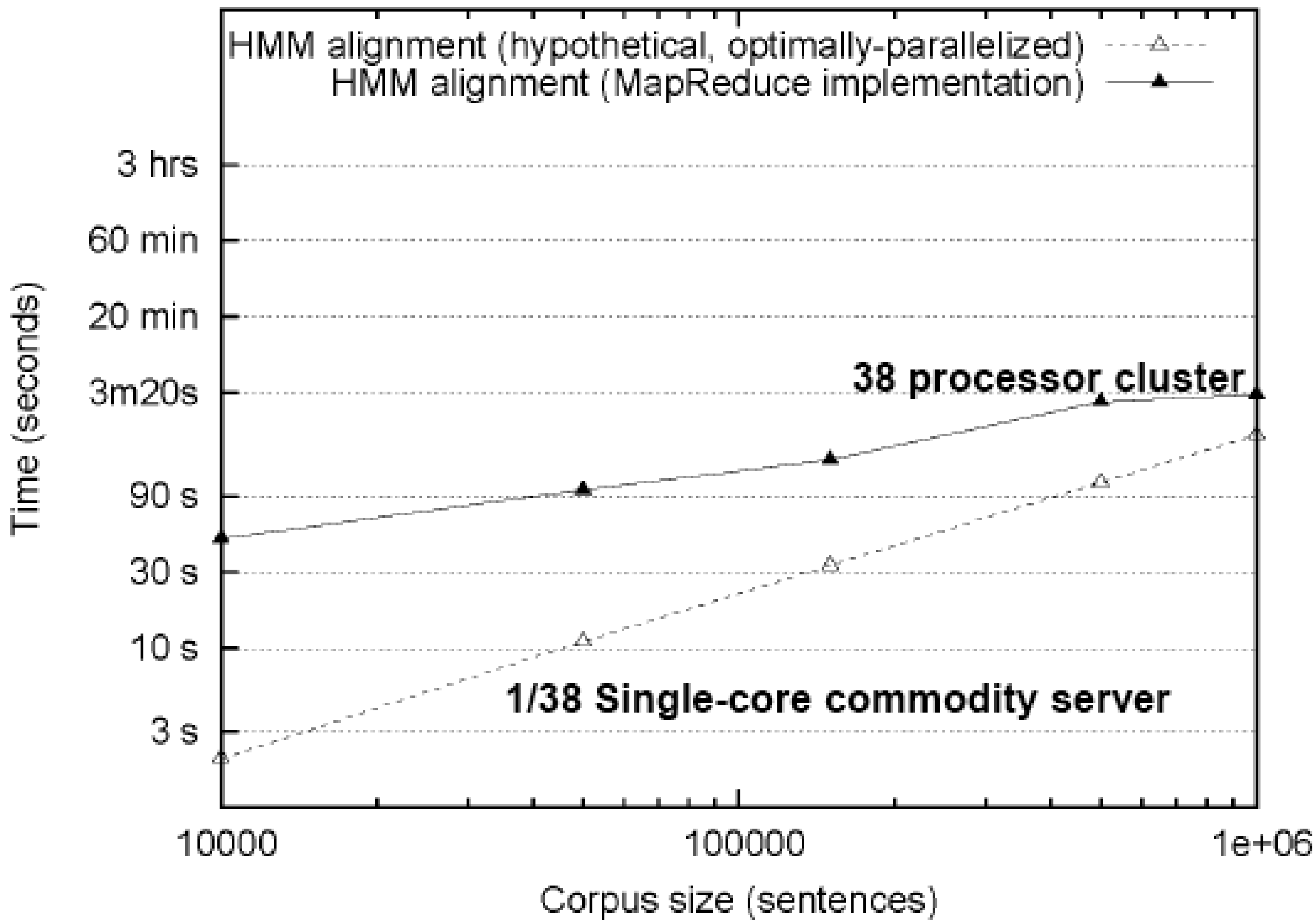
$$\hat{e} = \arg\max_{e} P(f \mid e)P(e)$$

- MT Architecture [Lin & Dryer, Tutorial at NAACL/HLT 2009]
  - Two main components: word alignment & phrase extraction

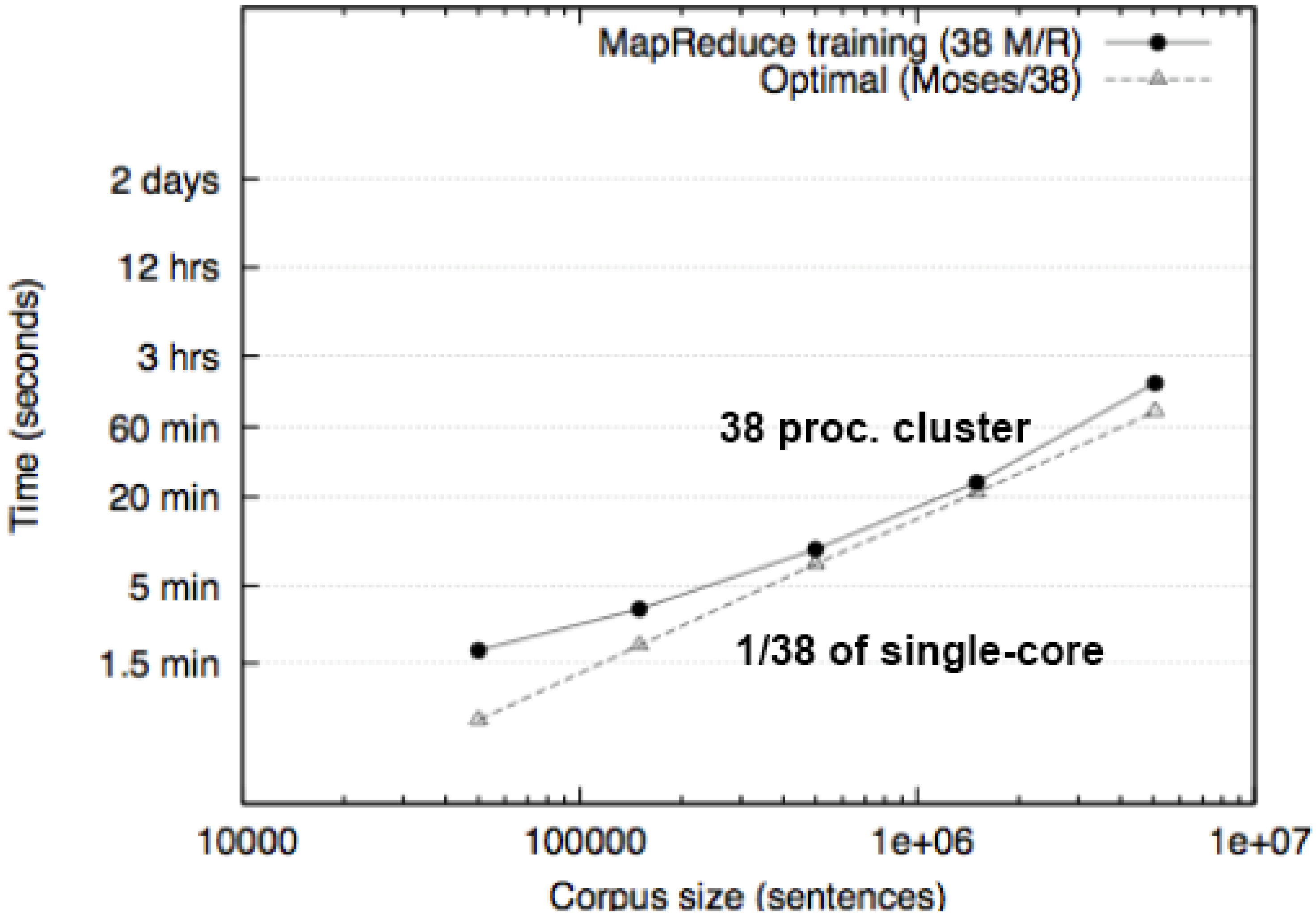


Word Alignment Results

[Lin & Dryer, Tutorial at NAACL/HLT 2009]

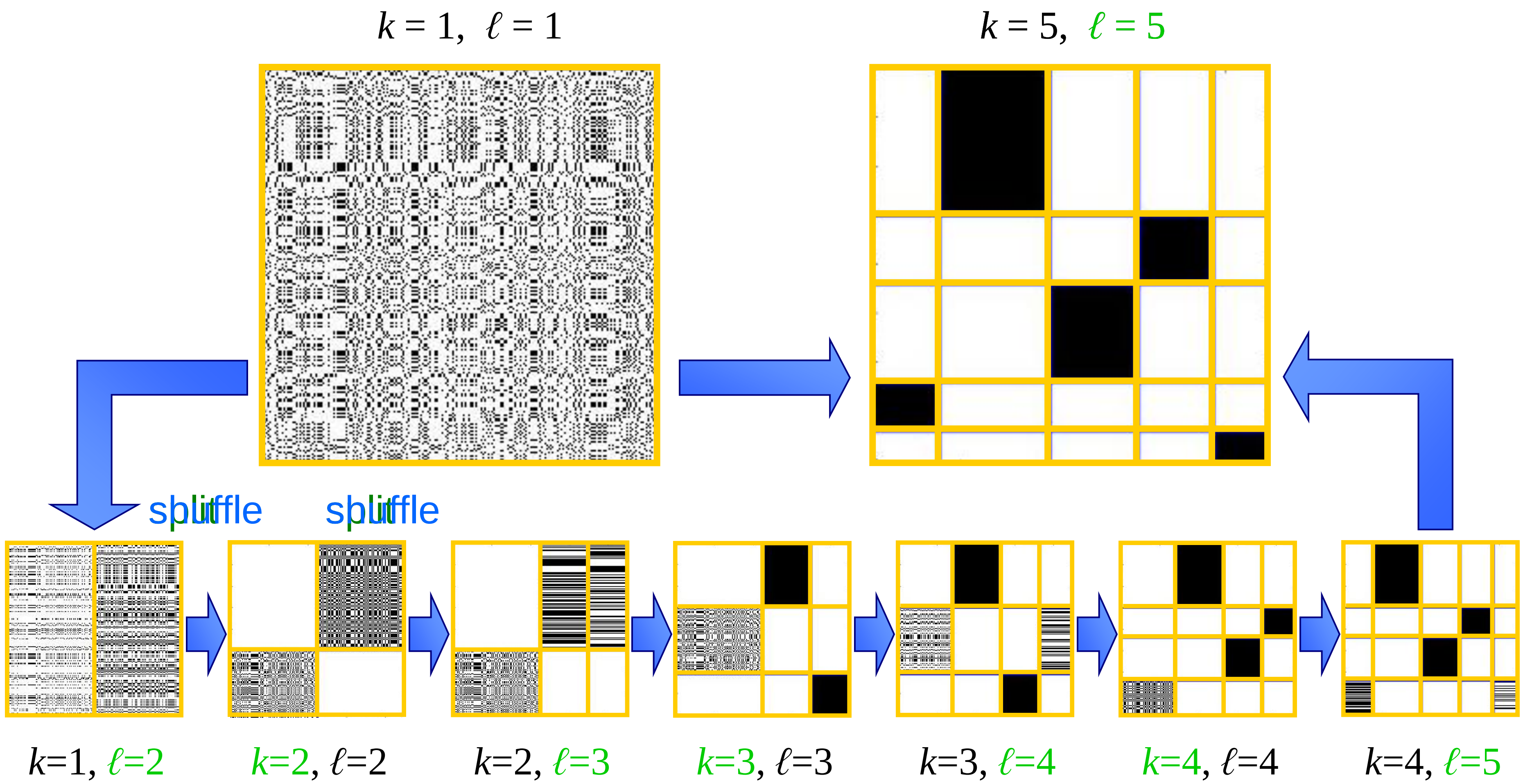


Phrase Table Construction
[Lin & Dryer, Tutorial at NAACL/HLT 2009]



Application 3: Distributed Co-Clustering

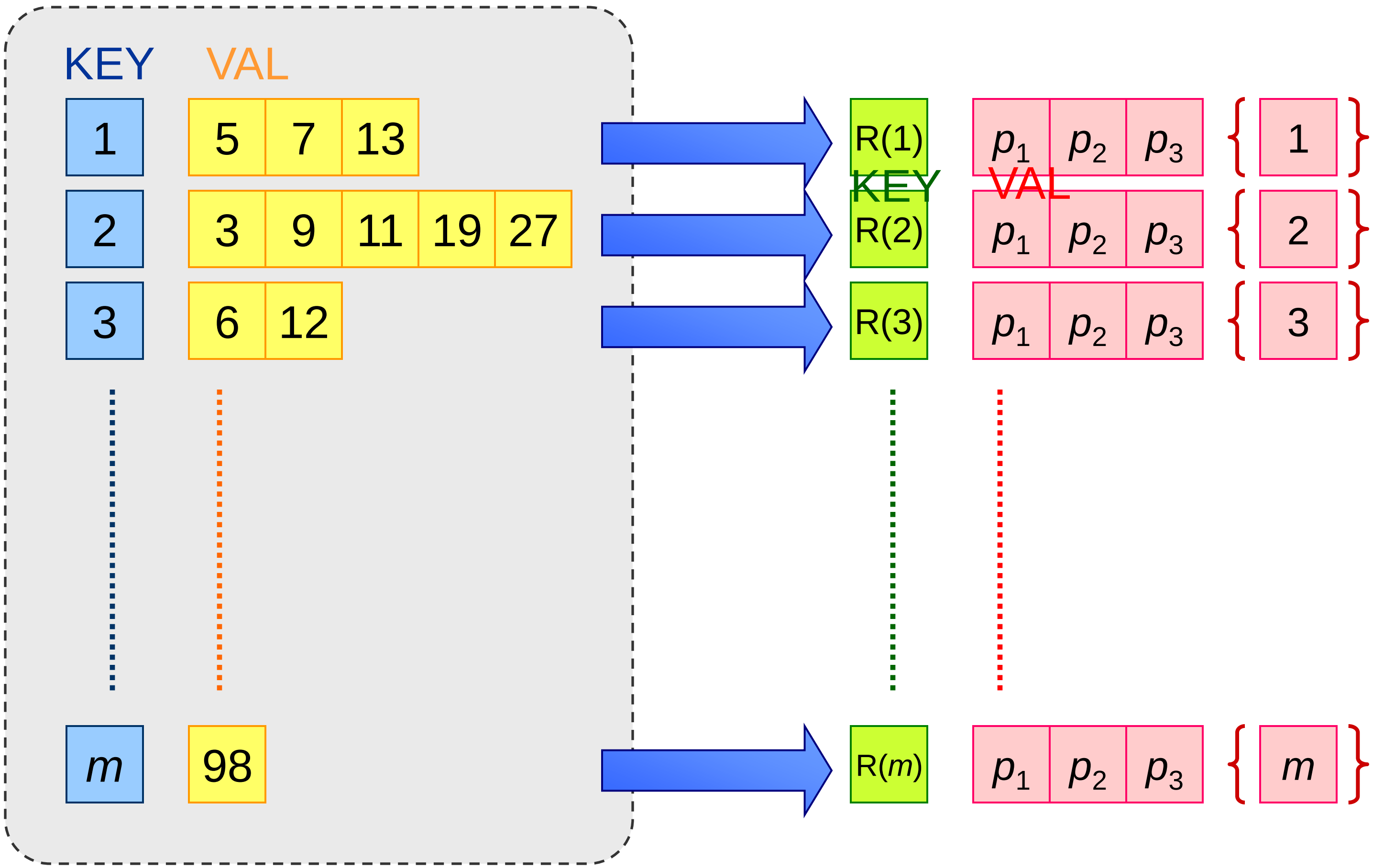
[Papadimitriou & Sun, ICDM'08]



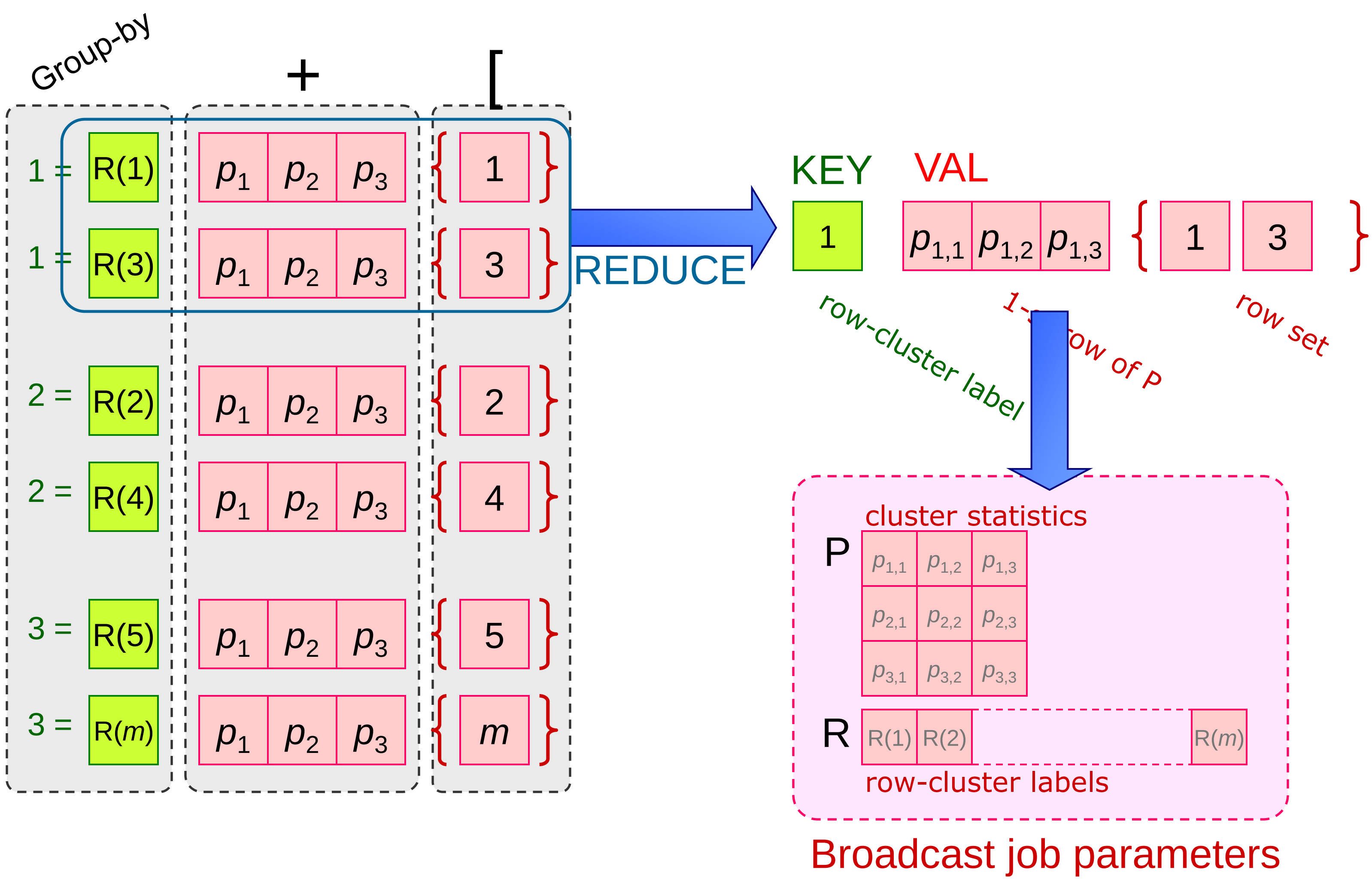
Split:
Increase k or ℓ

Shuffle:
Rearrange rows and cols

(Co-)clustering with MapReduce



(Co-)clustering with MapReduce



Scalability of MapReduce Co-Clustering

Scales up to ~10-15 nodes
But, at the moment, Hadoop implementation is sub-optimal for short jobs…

Machine Learning w. MapReduce: Remarks

- MapReduce is applicable to many scenarios
  - Convertible to MapReduce for summation-form algorithms
  - Suitable for algorithms with less iterations and large computational cost inside the loop
- No universally optimal parallelized methods
  - Tradeoff: Hadoop overhead and parallelization
  - Need algorithm design and parameter tuning for specific tasks
  - Goldilocks argument: it's all about the right-level abstraction
- Useful resources:
  - MR toolboxes: Apache Mahout
  - ICDM'09, Workshop on "Large-scale data mining"
  - ACM MM'09, Workshop on "Large-scale multimedia mining"
  - NIPS'09, Workshop on "Large-scale machine learning"

Practical Experience on MapReduce

[Dean, PACT'06 Keynote]

- **Fine granularity tasks:** map tasks (200K) >> nodes (2K)
  - Minimizes time for fault recovery
  - Can pipeline shuffling with map execution
  - Better dynamic load balancing
- **Fault Tolerance**: handled by re-execution
  - Lost 1600/1800 machines once → finished ok
- **Speculative execution:** spawn tasks when near to end
  - Avoid slow workers which significantly delay completion time
- **Locality optimization**: move the code to "data"
  - Thousands of machines read at local speed
- **Multi-core:** more effective than multi-processors

Conclusions

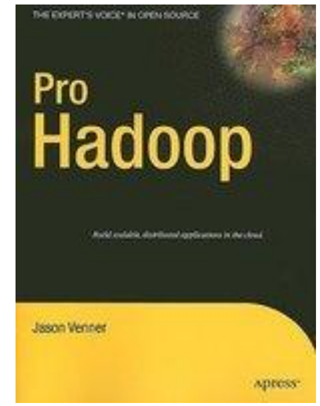
- MapReduce: simplified parallel programming model
  - Build ground-up from scalability, simplicity, fault-tolerance
  - Hadoop: open-source platform on commodity machines
  - Growing collections of components & extensions
- Data Mining Algorithms with MapReduce
  - MapReduce-compatible for summation-form algorithms
  - Need task-specific algorithm design and tuning
- MapReduce has been widely used in a broad range of applications and by many organizations
  - Growing tractions from both academia and industry
  - Three application categories: text processing, data warhousing and machine learning

Future Research Opportunities
MapReduce for Data Mining

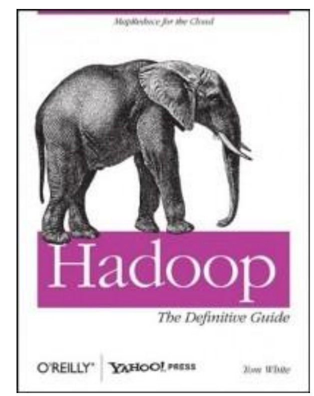
- **Algorithm** perspective
  - Convert known algorithms to their MapReduce version
  - Design descriptive language for MapReduce mining
  - Extend MapReduce primitives for data mining, such as multi-iteration MapReduce with data sharing
- **System** perspective
  - Improve MapReduce scalability for mining algorithms
- **Application** perspective
  - Discover novel applications by learning and processing such an unprecedented scale of data

MapReduce Books

- Pro Hadoop by Jason Venner
  - Hadoop Version: 0.20
  - Publisher: Apress
  - Date of Publishing: June 22, 2009

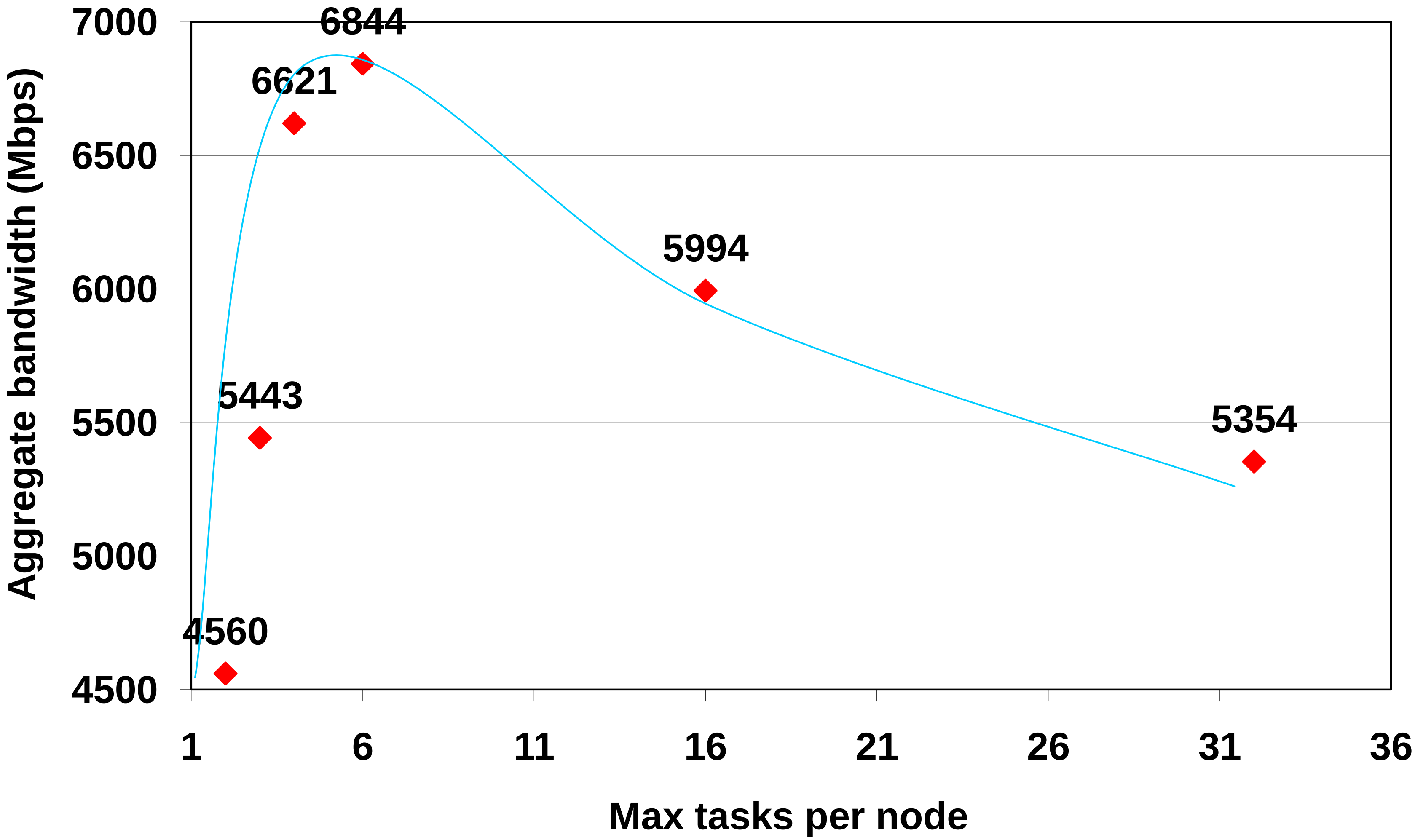


- Hadoop: The Definitive Guide by Tom White
  - Hadoop Version: 0.20
  - Publisher: O'Reilly
  - Date of Publishing: June 19, 2009

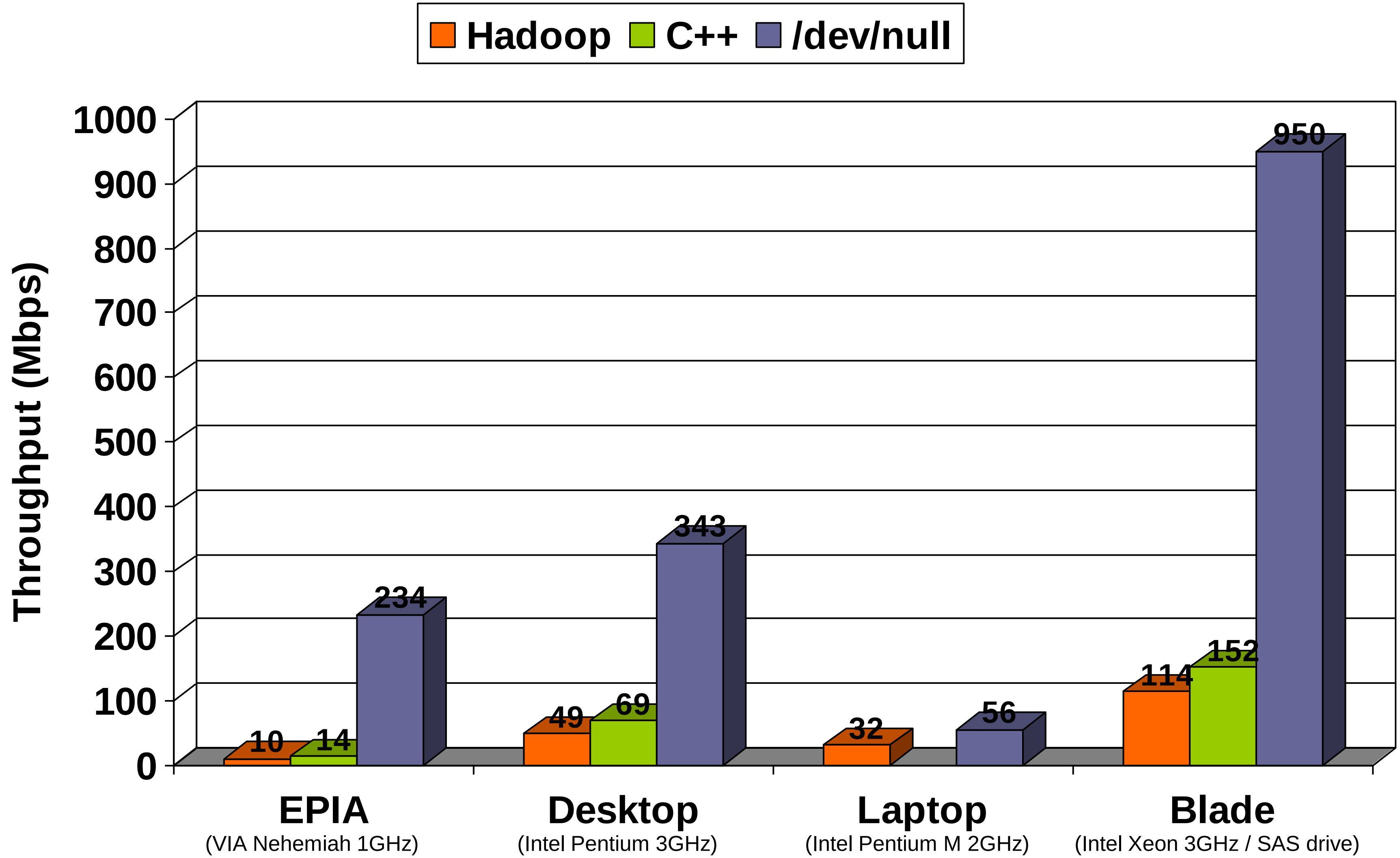


BACKUP

Thread pool size



Single-core performance



Out-of-the-box configuration(s)