COMP 551 – Applied Machine Learning
Lecture 23: Parallelization methods for large-scale machine learning

Instructor: Joelle Pineau (*jpineau@cs.mcgill.ca)*

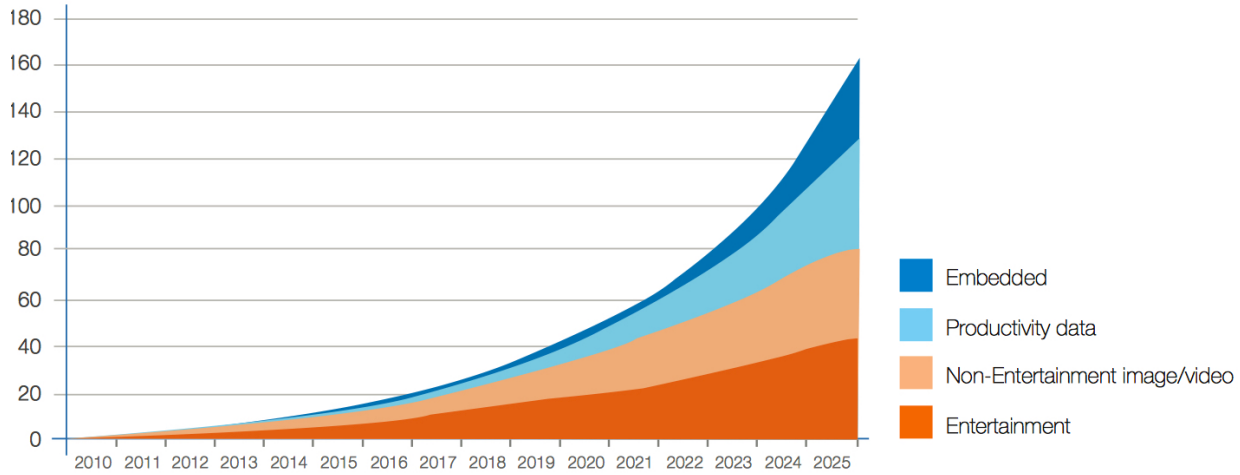
Class web page: *www.cs.mcgill.ca/~jpineau/comp551*

Unless otherwise noted, all material posted for this course are copyright of the instructor, and cannot be reused or reposted without the instructor's written permission.

Big Data by the numbers

Data creation by type:

(in zetabytes)



- An estimated 200PB of data stored by Facebook (2013).
- An estimated 2,000PB of capacity on the NSA cluster (2013).
- An estimated 455PB of data stored by Yahoo! (2014).
- An estimated 500PB of data stored by Youtube (2015).
- An estimated 10-15 exabytes of data stored by Google (2016).

Big data is large # of features, or large # of datapoints, or both.

Big Data

Major issues for machine learning? Usually we love more data!

Big Data

Major issues for machine learning? Usually we love more data!

- Hard to **do** something useful with all that data!
  - 1 computer reads 30-35 MB/sec from disk. ~4 months to read the web. ~1000 hard drives to store the web.
- Copying large amounts of data over a network takes time.
- Need distributed data storage & access.
- Need parallelized algorithms.
- Can't use algorithms that are superlinear in the number of examples.
- Large number of features causes overfitting.
- Often need to deal with unbalanced datasets.
- Need to represent high-dimensional data.

An era of supercomputers

Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	National Supercomputing Center in Wuxi China	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway NRCPC	10,649,600	93,014.6	125,435.9	15,371
2	National Super Computer Center in Guangzhou China	**Tianhe-2 (MilkyWay-2)** - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P NUDT	3,120,000	33,862.7	54,902.4	17,808
3	DOE/SC/Oak Ridge National Laboratory United States	**Titan** - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.	560,640	17,590.0	27,112.5	8,209
4	DOE/NNSA/LLNL United States	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM	1,572,864	17,173.2	20,132.7	7,890
5	DOE/SC/LBNL/NERSC United States	**Cori** - Cray XC40, Intel Xeon Phi 7250 68C 1.4GHz, Aries interconnect Cray Inc.	622,336	14,014.7	27,880.7	3,939
6	Joint Center for Advanced High Performance Computing Japan	**Oakforest-PACS** - PRIMERGY CX1640 M1, Intel Xeon Phi 7250 68C 1.4GHz, Intel Omni-Path Fujitsu	556,104	13,554.6	24,913.5	2,719

http://top500.org/lists/2016/11/

An era of supercomputers

Name / Insitution	Cores	Summary
GP3 Compute Canada	null	**Capacity cluster** 20,000+ CPU cores, 400+ GPU devices, >128GB/node, and most nodes with local storage; bigmem nodes
GP2 Compute Canada	null	**Capacity cluster** 20,000+ CPU cores, 400+ GPU devices, >128GB/node, and most nodes with local storage; bigmem nodes
Mammouth-Parallel II Université de Sherbrooke	30984	**Capability cluster** 30984 cores, 24 cores/node, 32 G/node & FAT nodes, opteron 2.1 GHz
GPC University of Toronto	30912	**Capability cluster** 30,240 cores; 8 cores/node; 16 GB/node; x86; IB
Guillimin McGill University	19456	**Capability cluster** 20176 cores, 12 or 16 cores/node, QDR IB
orcinus University of British Columbia	9616	**Capability cluster** 9616 cores total: 3088 cores, 8 cores/node, 16 GB/node, x86, IB: Plus 6528 cores, 12 cores/node, 24GB/node, x86, IB

https://computecanada.ca/

Hadoop

- Open-source software framework for datacentric computing:

  Distributed File System + MapReduce + Advanced components.

  MapReduce = The system that assigns jobs to nodes in the cluster.

- Allows distributed processing of large datasets across clusters.
- High degree of fault tolerance:
  - When nodes go down, jobs are re-distributed.
  - Data is replicated multiple times, in “small” chunks.
- Substantial uptake by industry.

Hadoop Distributed File System (HDFS)

- All data is stored multiple times:
  - Robust to failure. Don't need to backup a Hadoop cluster.
  - Multiple access points for any one piece of data.

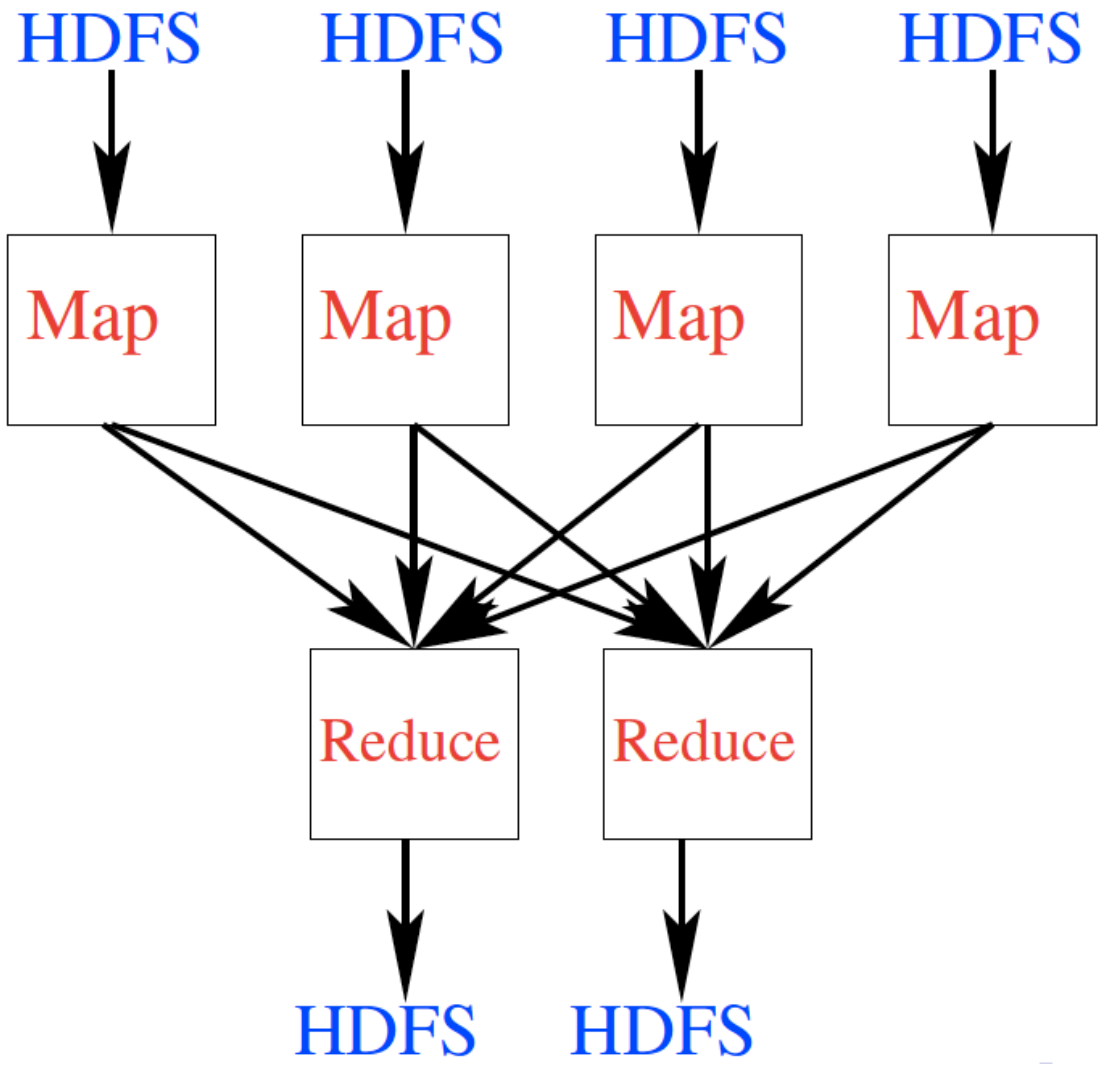
- Files are stored in 64MB chunks ("shards").

Hadoop Distributed File System (HDFS)

- All data is stored multiple times:
  - Robust to failure. Don't need to backup a Hadoop cluster.
  - Multiple access points for any one piece of data.

- Files are stored in 64MB chunks ("shards").
  - Sequential reads are fast. 100MB/s disk requires 0.64s to read a chunk but only 0.01s to start reading it.
  - Not made for random reads.
  - Not efficient for accessing small files.
  - No support for file modification.

MapReduce



MapReduce

- Programming framework for applying parallel algorithms to large datasets, across a cluster of computers. Developed by Google.

- *Map*: Partition the data across nodes, perform computation (independently) on each partition.

- *Reduce*: Merge values from nodes to produce global output.

Example

- Task: Count word occurrences across many documents

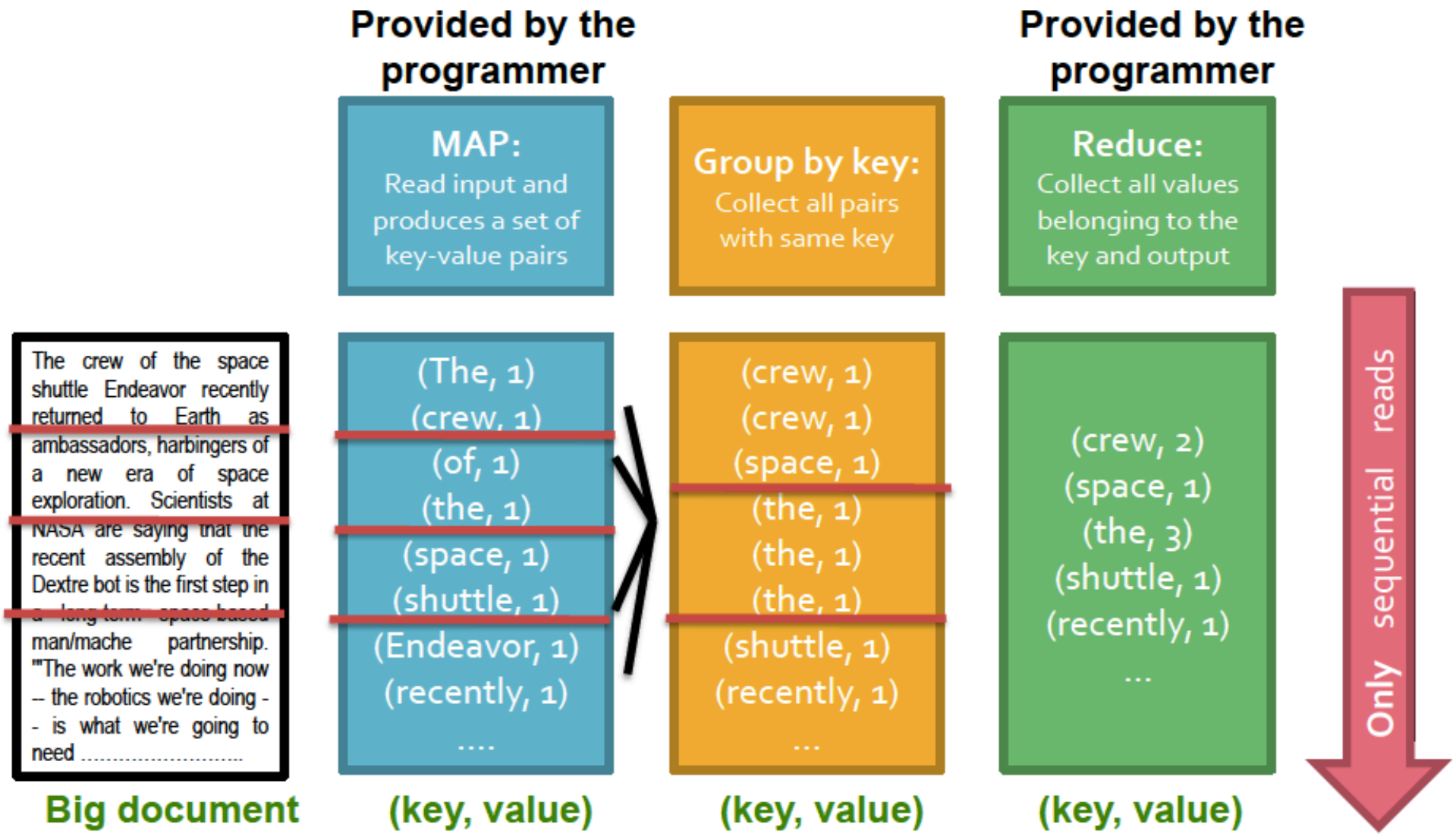
Example

- Task: Count word occurrences across many documents

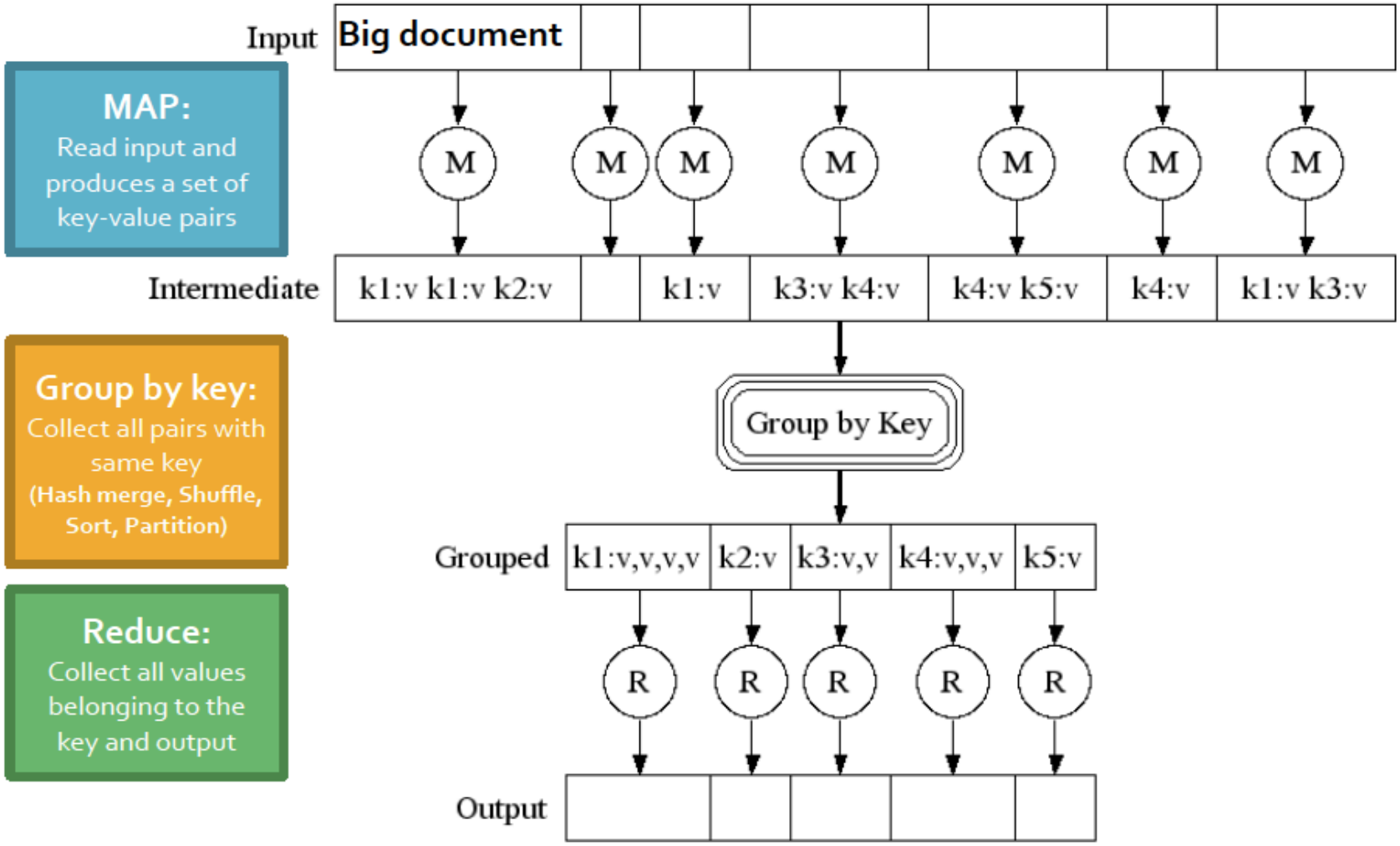
```
map(String input_key, String input_value):
  // input_key: document name
  // input_value: document contents
  for each word w in input_value:
    EmitIntermediate(w, "1");


reduce(String output_key, Iterator intermediate_values):
  // output_key: a word
  // output_values: a list of counts
  int result = 0;
  for each v in intermediate_values:
    result += ParseInt(v);
  Emit(AsString(result));
```

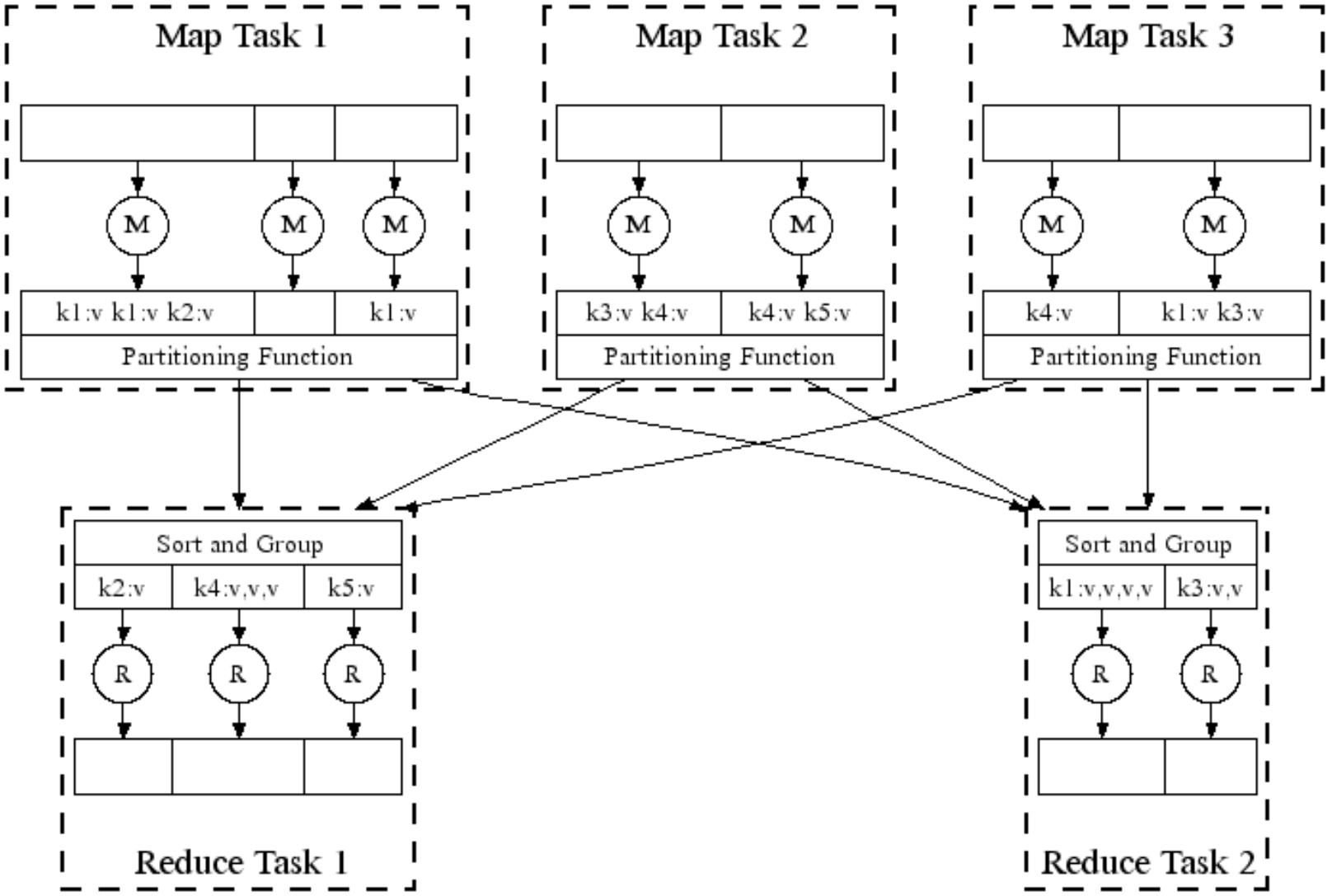
Word count example



Execution

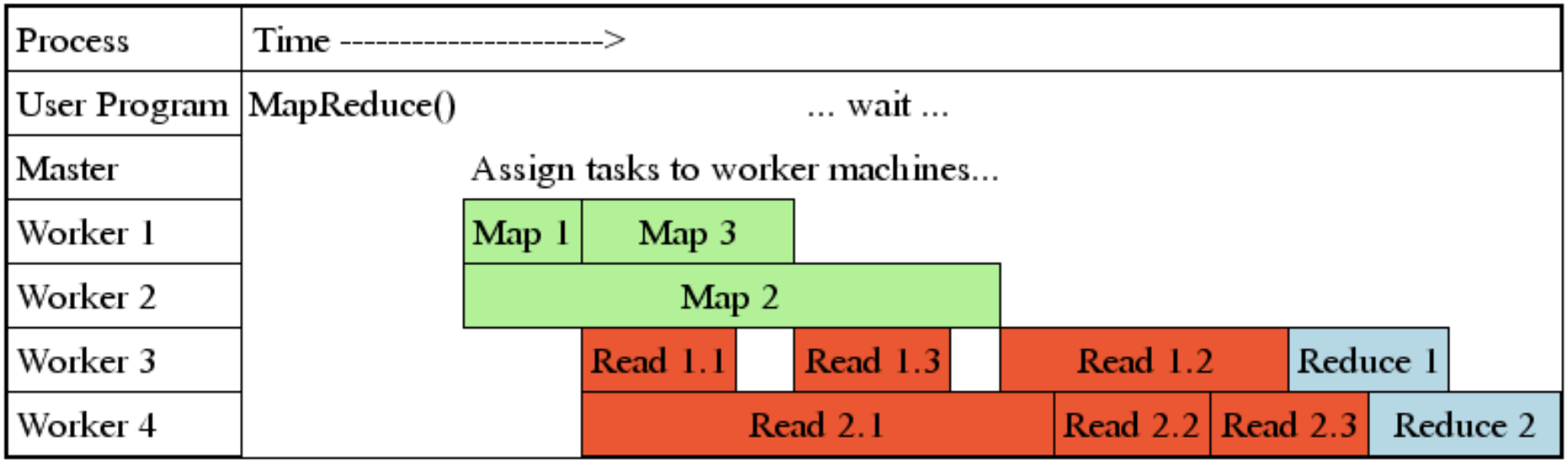

MAP:
Read input and produces a set of key-value pairs
Group by key:
Collect all pairs with same key
(Hash merge, Shuffle, Sort, Partition)
Reduce:
Collect all values belonging to the key and output
Input
Big document
M
Intermediate
k1:v k1:v k2:v
k1:v
k3:v k4:v
k4:v k5:v
k4:v
k1:v k3:v
Group by Key
Grouped
k1:v,v,v,v
k2:v
k3:v,v
k4:v,v,v
k5:v
R
Output

Parallel Execution



Task scheduling

- Typically many more tasks than machines.
  - E.g. 200,000 map / 5000 reduce tasks w/ 2000 machines.



Fault tolerance

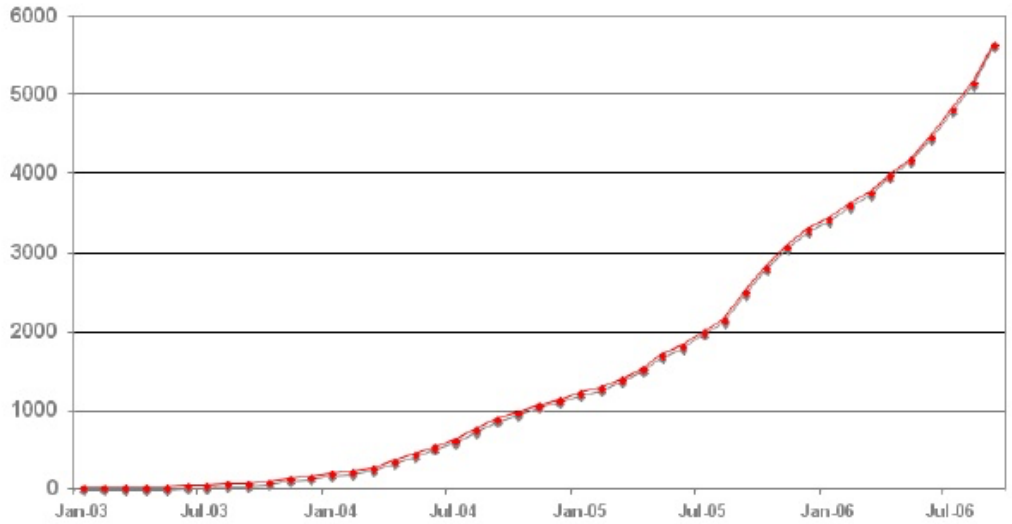
- Handled via re-execution.
- Worker failure:
  - Detect via periodic checking.
  - Re-execute completed and in-progress *map* tasks.
  - Re-execute in-progress *reduce* tasks.

Fault tolerance

- Handled via re-execution.
- Worker failure:
  - Detect via periodic checking.
  - Re-execute completed and in-progress *map* tasks.
  - Re-execute in-progress *reduce* tasks.
- Master failure:
  - Could handle; highly unlikely.
- Slow workers significantly lengthen completion time:
  - Near end of job, spawn backup copies of in-progress tasks.

Growing applicability

MapReduce Programs In Google Source Tree



Example uses:

distributed grep	distributed sort	web link-graph reversal
term-vector per host	web access log stats	inverted index construction
document clustering	machine learning	statistical machine translation
...	...	...

Properties

- Advantages:
  - Automatic parallelization and distribution.
  - Fault-tolerance.
  - I/O scheduling
  - Status and monitoring

- Limitations:
  - Cannot share data across jobs.

MapReduce for machine learning

- Many ML algorithms can be re-written in "summation form".

- Consider linear regression:

  $Err(w) = \sum_{i=1:n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2 \quad \rightarrow \quad \hat{\mathbf{w}} = (X^T X)^{-1} X^T Y$

  Let $A = (X^T X)$, $B = (X^T Y)$ $\quad \rightarrow \quad A = \sum_{i=1:n} (x_i x_i^T)$, $B = \sum_{i=1:n} (x_i y_i)$

  Now the computation of A and B can be split between different nodes.

MapReduce for machine learning

- Many ML algorithms can be re-written in "summation form".

- Consider linear regression:

  $Err(w) = \sum_{i=1:n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2 \quad \rightarrow \quad \hat{\mathbf{w}} = (X^T X)^{-1} X^T Y$

  Let $A = (X^T X)$, $B = (X^T Y)$ $\quad \rightarrow \quad A = \sum_{i=1:n} (x_i x_i^T)$, $B = \sum_{i=1:n} (x_i y_i)$

  Now the computation of A and B can be split between different nodes.

- Other algorithms that are amenable to this form:
  - Naïve Bayes, LDA/QDA, K-means, logistic regression, neural networks, mixture of Gaussians (EM), SVMs, PCA.

Parallel linear learning

- Given 2.1 TB of data, how can you learn a good linear predictor? How long does it take?
  - 17B examples, 16M parameters, 1K computation nodes.

- Stochastic gradient descent will take a long time going through all this data!

Parallel linear learning

- Given 2.1 TB of data, how can you learn a good linear predictor? How long does it take?
  - 17B examples, 16M parameters, 1K computation nodes.

- Stochastic gradient descent will take a long time going through all this data!

- Answer: Can be done using parallelization in 70 minutes = 500M features/second.
  - Faster than the I/O bandwidth of a single machine.

Parallel learning for parameter search

- One of the most common use of parallelization in ML is for hyper-parameter optimization.

- *Map*: Create separate jobs, exploring different subsets of parameters. Train separate models, calculate performance on validation set.

- *Reduce*: Select subset with the best validation set performance.

Parallel learning of decision trees

- Usually, all data examples are used to select each subtree split.
  - If data is not partitioned properly, may produce poor trees.

- When parallelizing, each processor can handle either:
  - Subset of the features: Easier to efficiently parallelize tree.
  - Subset of the training data: More complex to properly parallelize tree.

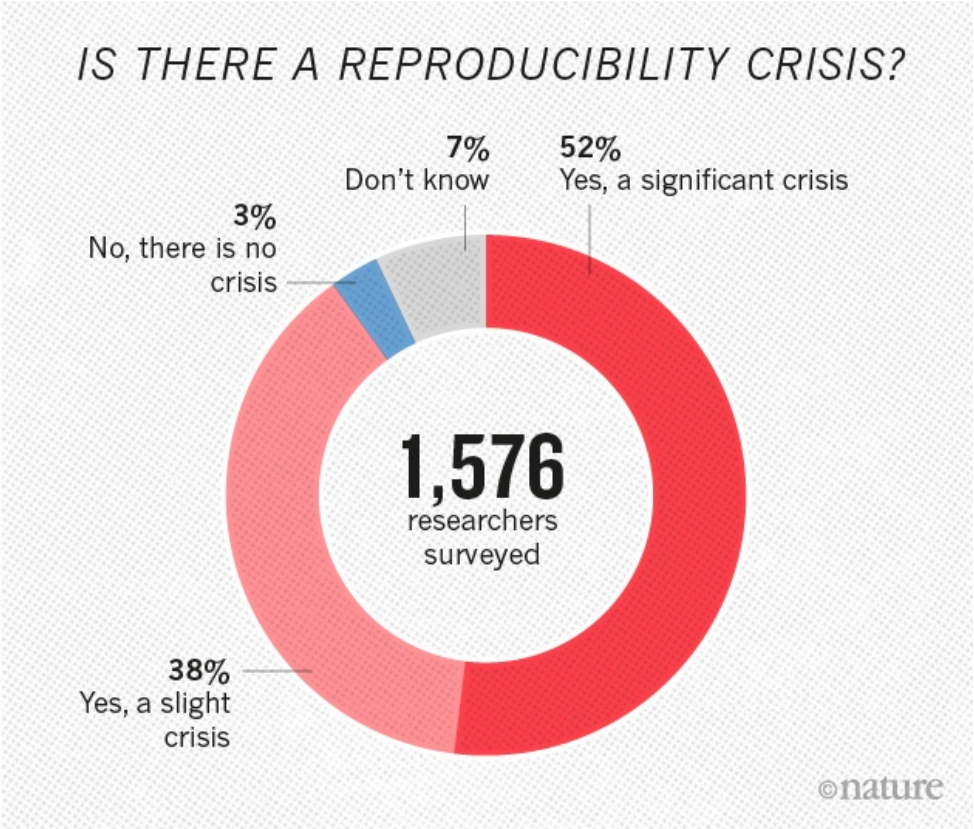
- Easier to parallelize learning of **Random Forests** (true of bagging in general).

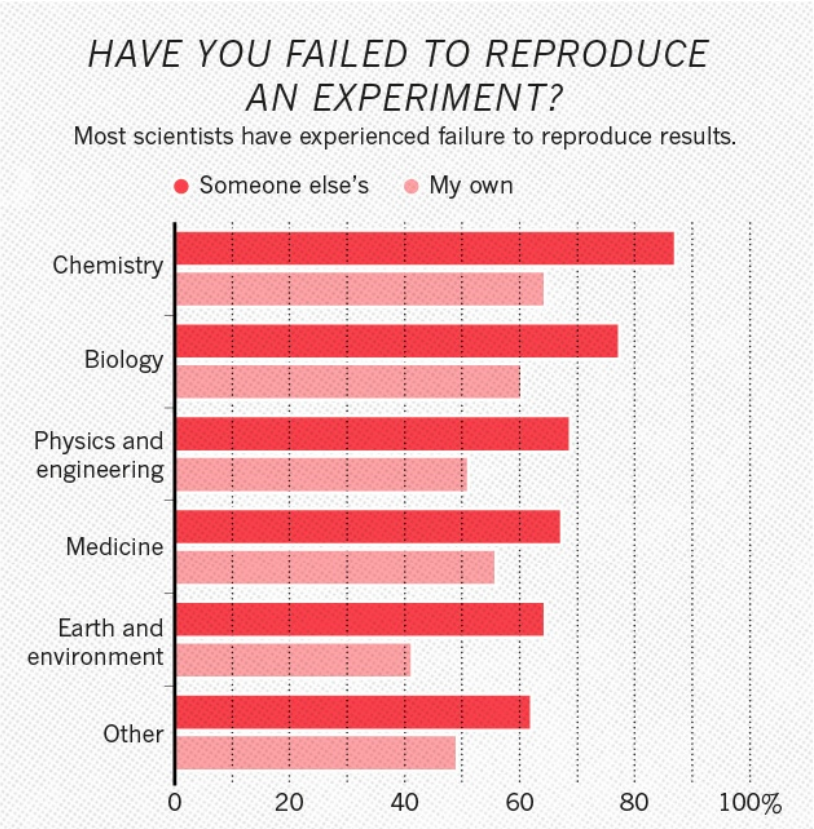
Change of topic: **Reproducibility**

"Reproducibility refers to the **ability of a researcher to duplicate the results of a prior study** using the same materials as were used by the original investigator. That is, a second researcher might use the same raw data to build the same analysis files and implement the same statistical analysis in an attempt to yield the same results. Reproducibility is a minimum necessary condition for a finding to be believable and informative."

K. Bollen, J. T. Cacioppo, R. Kaplan, J. Krosnick, J. L. Olds, *Social, Behavioral, and Economic Sciences Perspectives on Robust and Reliable Science,* National Science Foundation, 2015.

Reproducibility crisis in science





https://www.nature.com/news/1-500-scientists-lift-the-lid-on-reproducibility-1.19970

A criteria for reproducible research

"An article about computational science in a scientific publication is not the scholarship itself, it is merely the advertising of the scholarship. The actual scholarship is the complete software development environment and the complete set of instructions which generated the figures."

Buckheit and Donoho, *WaveLab and reproducible research*. Technical report, Stanford University,1995

Reproducibility in machine learning

From:

Olorisade, Brereton, Andras. *Reproducibility in Machine Learning-Based Studies: An Example of Text Mining*. ICML workshop on Reproducibility in ML. 2017. *https://openreview.net/pdf?id=By4l2PbQ-*

- Factors that affect reproducibility in ML:
  - Access to target dataset (exact copy).
  - Partitioning information (details of train/validate/test splits).
  - Implementation or executable files of the proposed methods.
  - Names and version numbers of modules and packages used.
  - Randomization control (random seed used).

Any others you can think of?

Reproducibility metrics

- Availability of datasets, partitioning information.
- Availability of code, names and version numbers of dependencies.
- Availability of random seed and all hyperparameters.
- Alignment between the paper and the code.
  - Code generates the exact pictures, tables, results in the paper?
- Clarity of code & paper (ease of understanding, language).
- Details of computing infrastructure used.
- Computation requirements (time, memory, number/type of machines).
- Reimplementation effort (time, expertise).
- Number and complexity of interactions with the authors.

Final notes

- **Project #3**
  - Peer reviews due on Thursday (I think – check CMT).
- **Project #4:**
  - Don't forget to sign up for the challenge!
    - https://docs.google.com/forms/d/1GAZnZWYW2suf6Z9polBlTQvTvMJIjkMy7CNyMapNKuY/edit?ts =59d53577
  - Pick a presentation slot; so far 21 teams signed up.
    - https://docs.google.com/spreadsheets/d/1G_wGgR7leHvfr2TSri_IrMVZwXZGXgtx-nlik-4GSZo/edit#gid=0
  - Submit your slides for the presentation:
    - https://drive.google.com/drive/folders/15AtV4cjE2Zlj5KgzG4vDm8QLkcN720Mp?usp=sharing
  - Final submission Dec.15 on CMT (report&code) and OpenReview (review).
- **Course evaluations** now available on Minerva. Please fill out!

Today's lecture

- Significant material (pictures, text, equations) for the material on parallelization was taken from:
  - *http://www.stanford.edu/class/cs246*
  - *http://cilvr.cs.nyu.edu/doku.php?id=courses:bigdata:slides:start*
  - *http://research.google.com/archive/mapreduce-osdi04-slides/index-auto-0002.html*
  - *http://machinelearning.wustl.edu/mlpapers/paper_files/NIPS2006_725.pdf*