MapReduce-I

Knowledge objectives

1. Enumerate several use cases of MapReduce
2. Describe what the MapReduce environment is
3. Explain 6 benefits of using MapReduce
4. Explain four problems of MapReduce

Understanding Objectives

1. Describe the different steps of a MapReduce execution

Application Objectives

1. Identify the usefulness of MapReduce in a given use case

Typical uses

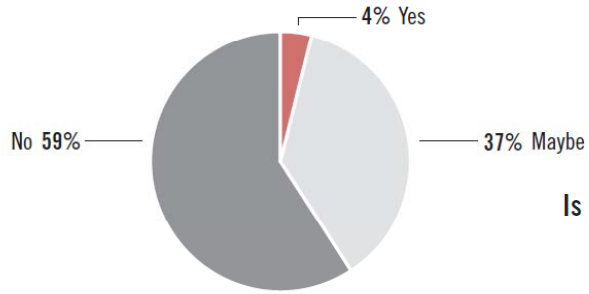
- Find which source pages link to a target page
- Count the number of accesses to each Web page
- Count the number of accesses to each domain
- Create an index structure that maps search terms to document IDs
- Retrieve introductory paragraph of all Web pages so that “x”
- Find all pairs of users accessing the same URL
- Find the average age of users accessing a given URL
- Find all friends of a given user
- Find all friends of friends of a given user
- Find all women friends of men friends of a given user
- Grouping different manifestations of the same real world object

Not so typical uses

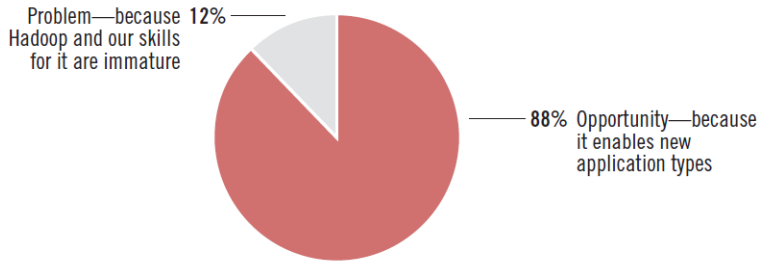
- Mobile Commerce
- Electricity
- Agricultural Planning
- Fuel Conservation
- National Intelligence
- Drug Development and Personalization
- Financial Service Security

Opinions

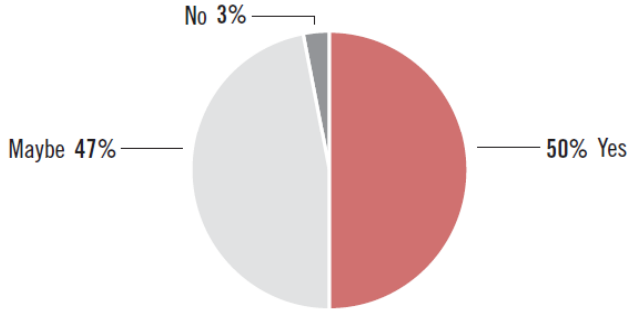
Can the Hadoop Distributed File System (HDFS) replace your enterprise data warehouse (EDW)?



Is Hadoop a problem or an opportunity?



Can HDFS augment your enterprise data warehouse (EDW) or other data infrastructure?



TDWI Best Practices Report, 2013

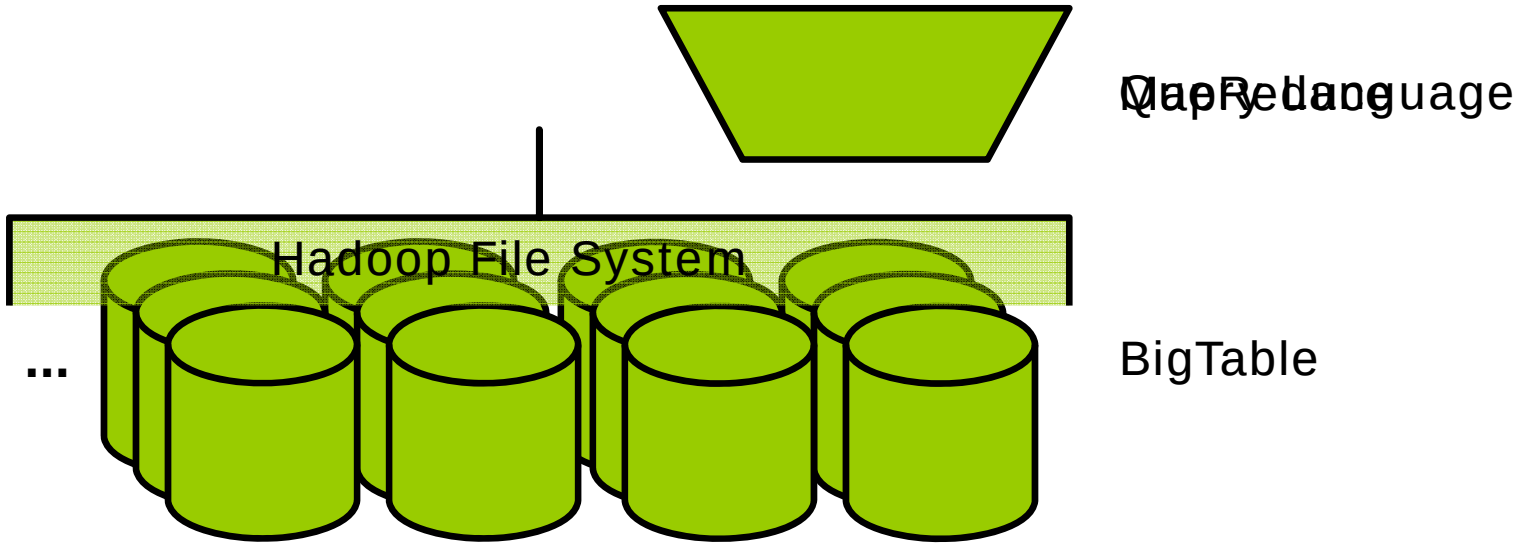
Busting 10 Myths about Hadoop

- Hadoop consists in multiple products
- Hadoop is open source but available from vendors, too
- Hadoop is an ecosystem, not a single product
- HDFS is a file system, not a DBMS
- Hive resembles SQL but is not standard SQL
- Hadoop and MapReduce are related but don't require each other
- MapReduce provides control for analytics, not analytics per se
- Hadoop is about data diversity, not just data volume
- Hadoop complements a DW; it's rarely a replacement
- Hadoop enables many types of analytics, not just Web analytics

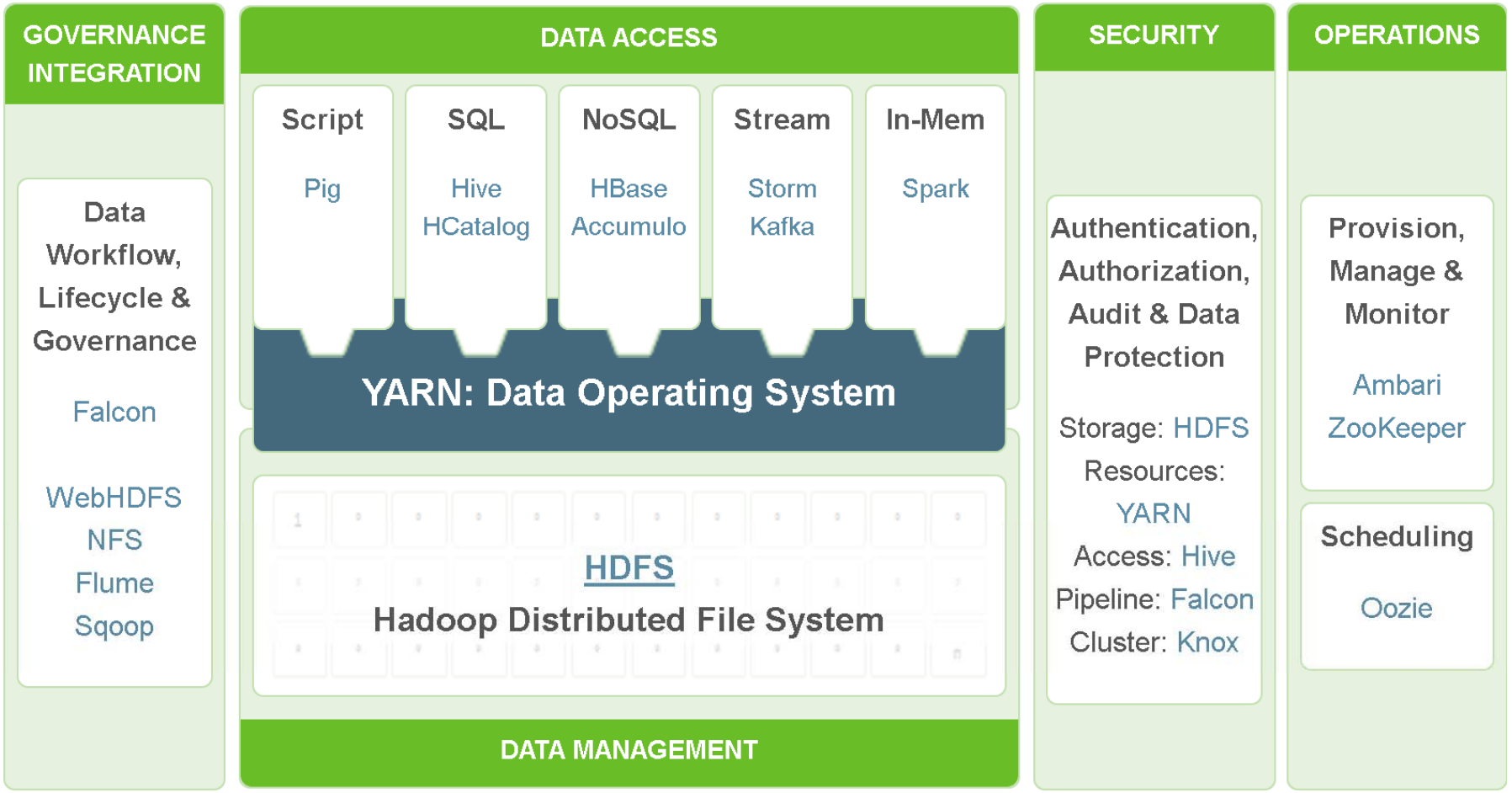
Philip Russom

Google ecosystem

- High-performance is mainly achieved by means of parallelism
  - Divide-and-conquer principle
- MapReduce
  - It is a query language that provides parallelism in a transparent manner

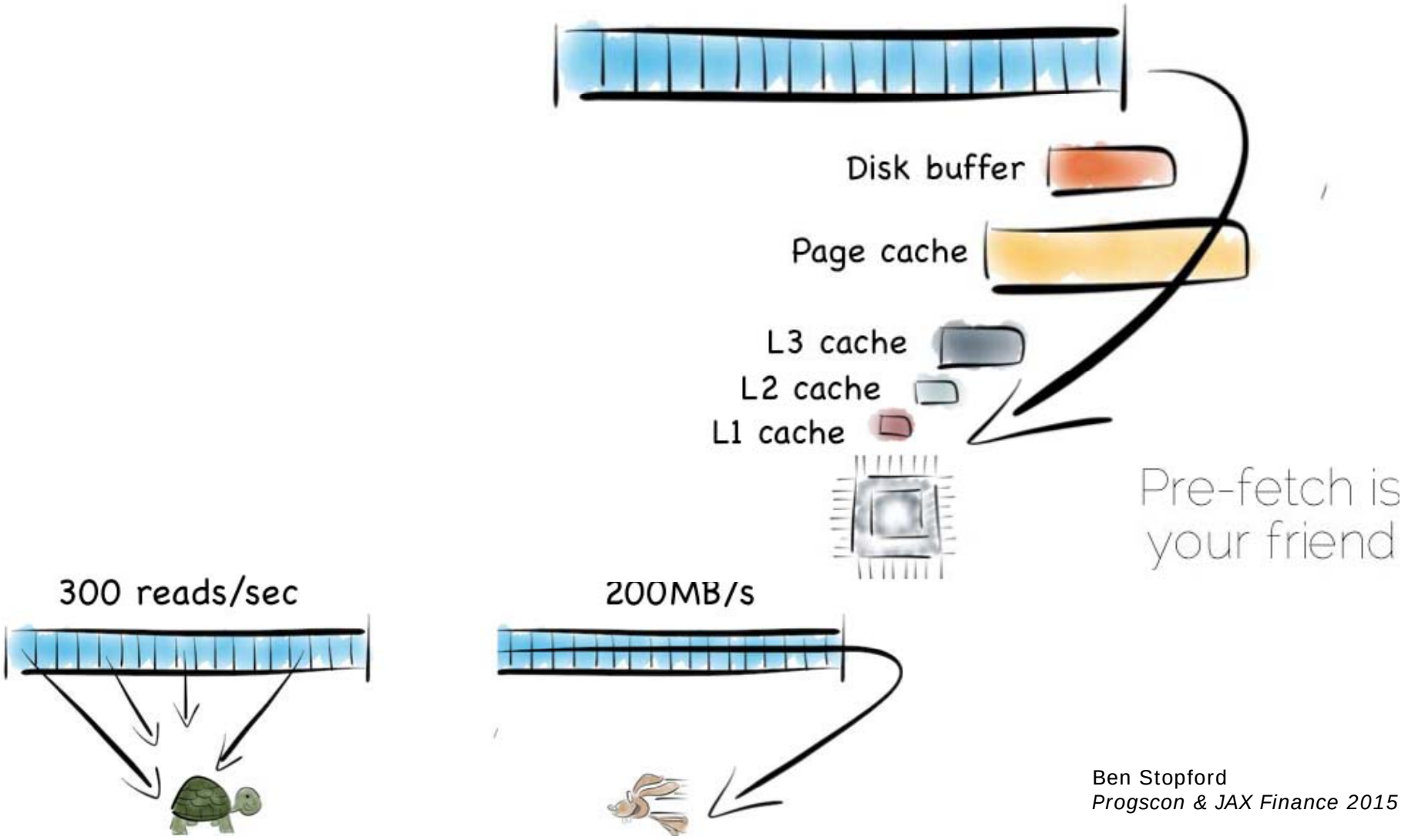


Hadoop ecosystem



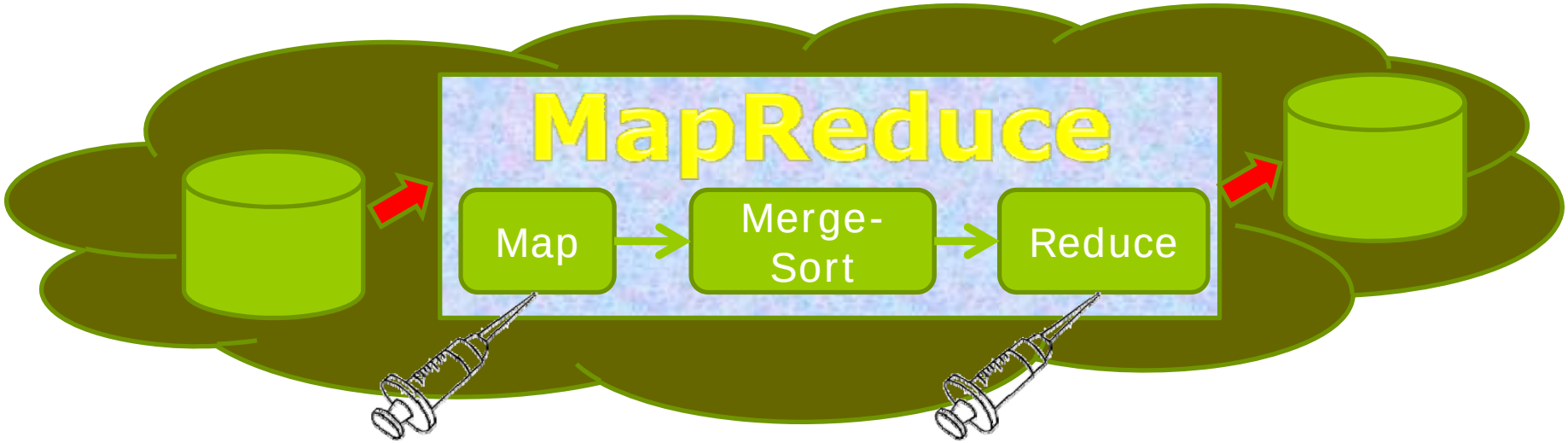
By HortonWorks

Sequential access



Ben Stopford
Progscon & JAX Finance 2015

MapReduce Basics



- Simple model to express relatively sophisticated distributed programs
  - Processes pairs [key, value]
  - Signature:

$$\text{map}(\text{key } k, \text{value } v) \mapsto [(ik_1, iv_1), \ldots, (ik_{m(k,v)}, iv_{m(k,v)})]$$

$$\text{reduce}(\text{key } ik, \text{vset } ivs) \mapsto [ov_1, .., ov_{r(ik,ivs)}]$$

WordCount Execution Example

The Project Gutenberg EBook of The Outline of Science, Vol. 1 (of 4), by J. Arthur Thomson

This eBook is for the use of anyone anywhere at no cost and with almost no restrictions whatsoever. You may copy it, give it away or re-use it under the terms of the Project Gutenberg License included with this eBook or online at www.gutenberg.org

Title: The Outline of Science, Vol. 1 (of 4)
A Plain Story Simply Told

Author: J. Arthur Thomson

Release Date: January 22, 2007 [EBook #20417]

Language: English

Character set encoding: ASCII

*** START OF THIS PROJECT GUTENBERG EBOOK OUTLINE OF SCIENCE ***

Produced by Brian Janes, Leonard Johnson and the Online Distributed Proofreading Team at http://www.pgdp.net

[Illustration: THE GREAT SCARLET SOLAR PROMINENCES, WHICH ARE SUCH A NOTABLE FEATURE OF THE SOLAR PHENOMENA, ARE IMMENSE OUTBURSTS OF FLAMING HYDROGEN RISING SOMETIMES TO A HEIGHT OF 500,000 MILES]

THE
OUTLINE OF SCIENCE

A PLAIN STORY SIMPLY TOLD

EDITED BY
J. ARTHUR THOMSON
REGIUS PROFESSOR OF NATURAL HISTORY IN THE UNIVERSITY OF ABERDEEN

WITH OVER 800 ILLUSTRATIONS
OF WHICH ABOUT 40 ARE IN COLOUR

<#line, text>

The Project Gutenberg EBook of The Outline of Science, Vol. 1 (of 4), by

Map

The	1
Project	1
Gutemberg	1
Ebook	1
of	1
The	1
Outline	1
Of	1
Science,	1
Vol.	1
1	1
(of	1
4),	1
by	1

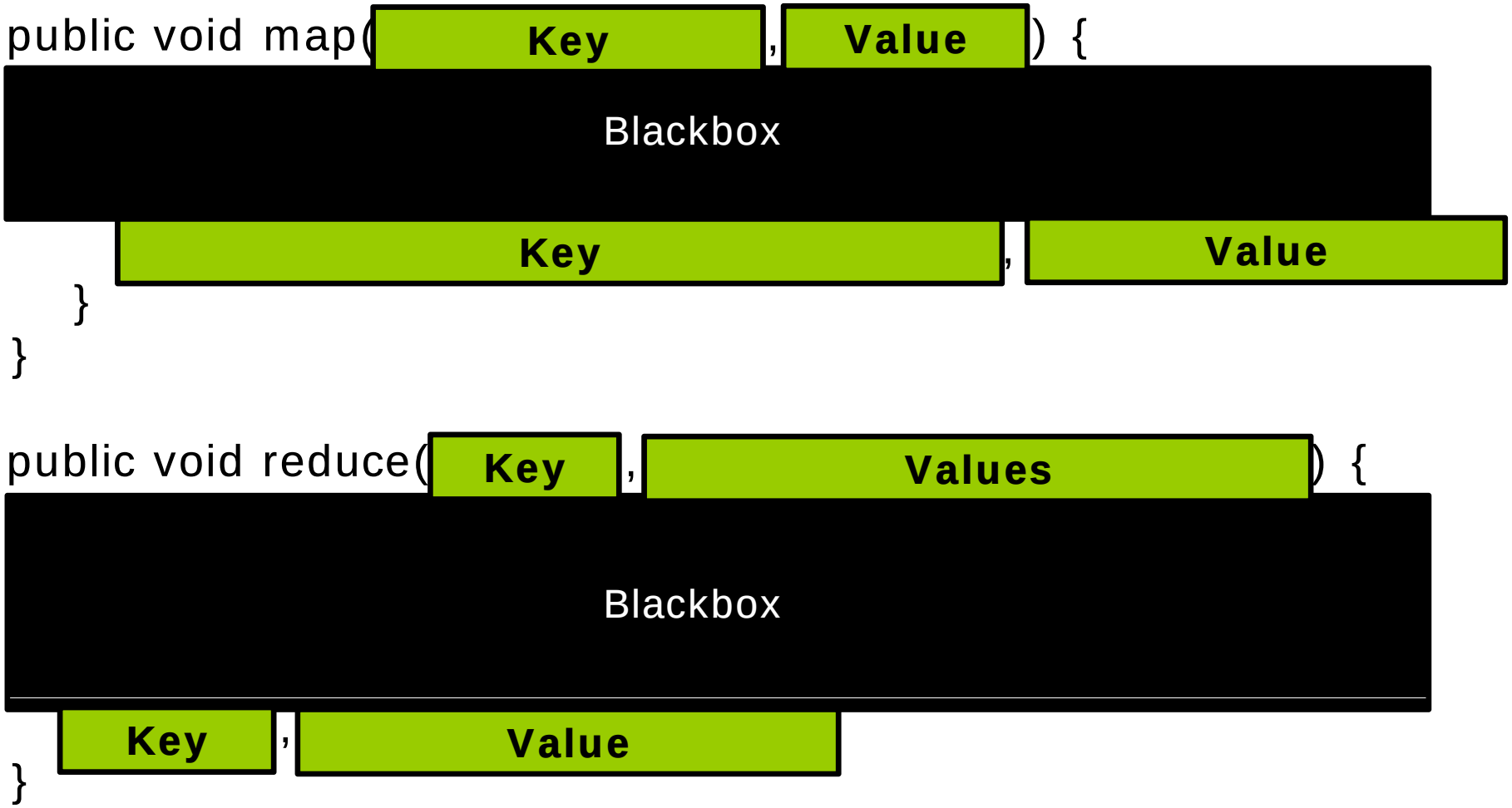
Merge-Short

["The",[1,1,1,1,...]]

Reduce

The	57631

WordCount Code Example



Benefits

- Programming (functional) model simple yet expressive
  - Without joins
- Able to process structured or unstructured
- Elastically scalable
  - Transparently distributes data
    - Exploits data locality
  - Hides parallelization
  - Balances workload
  - Provides fine grained fault tolerance

Problems

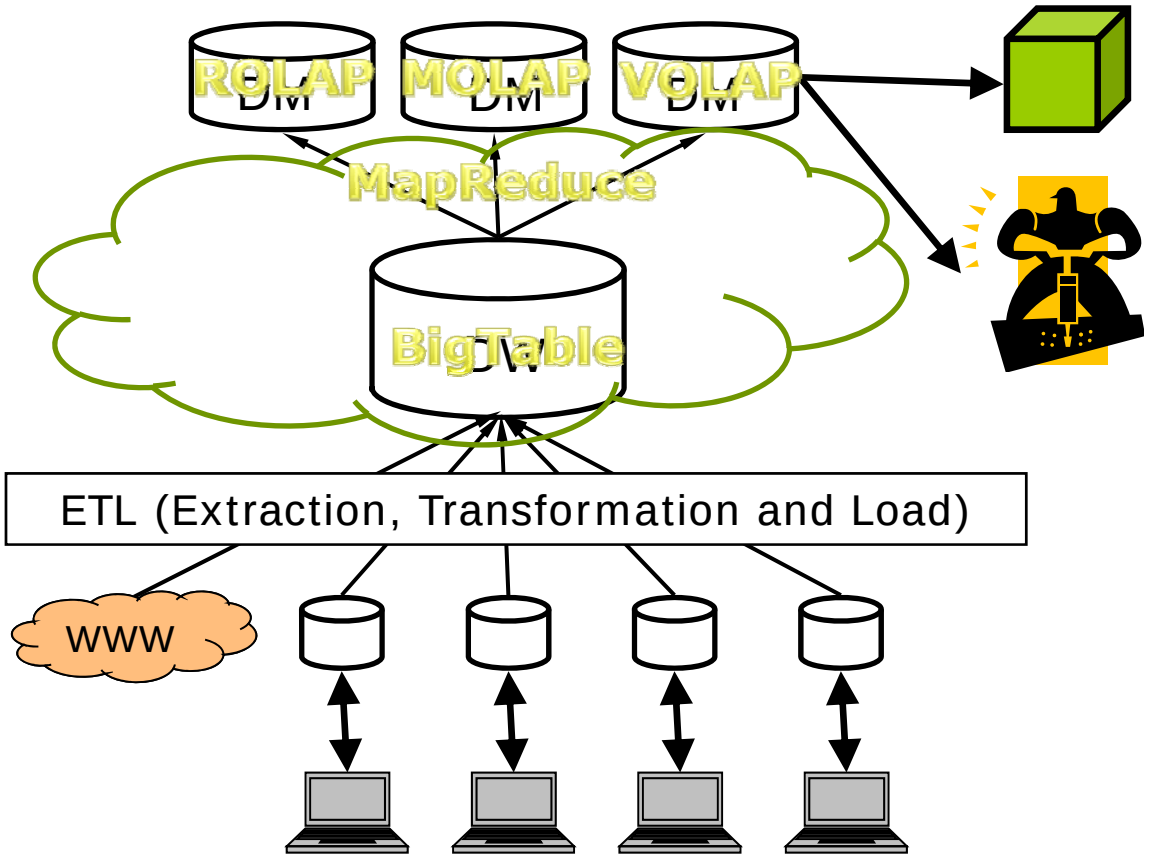
- Writes intermediate results to disk
  - Reduce tasks pull intermediate data
- Defines the execution plan on the fly
  - Schedules one block at a time
- Does not provide transactions
- Does not benefit from compression

Market tools

"... But to really unlock the power of Hadoop, you must be able to efficiently extract data stored across multiple (often tens or hundreds) of nodes with a user-friendly ETL (extract, transform and load) tool that will then allow you to move your Hadoop data into a relational data mart or warehouse where you can use BI tools for analysis. "

Ian Fyfe
Pentaho

Reference Architecture

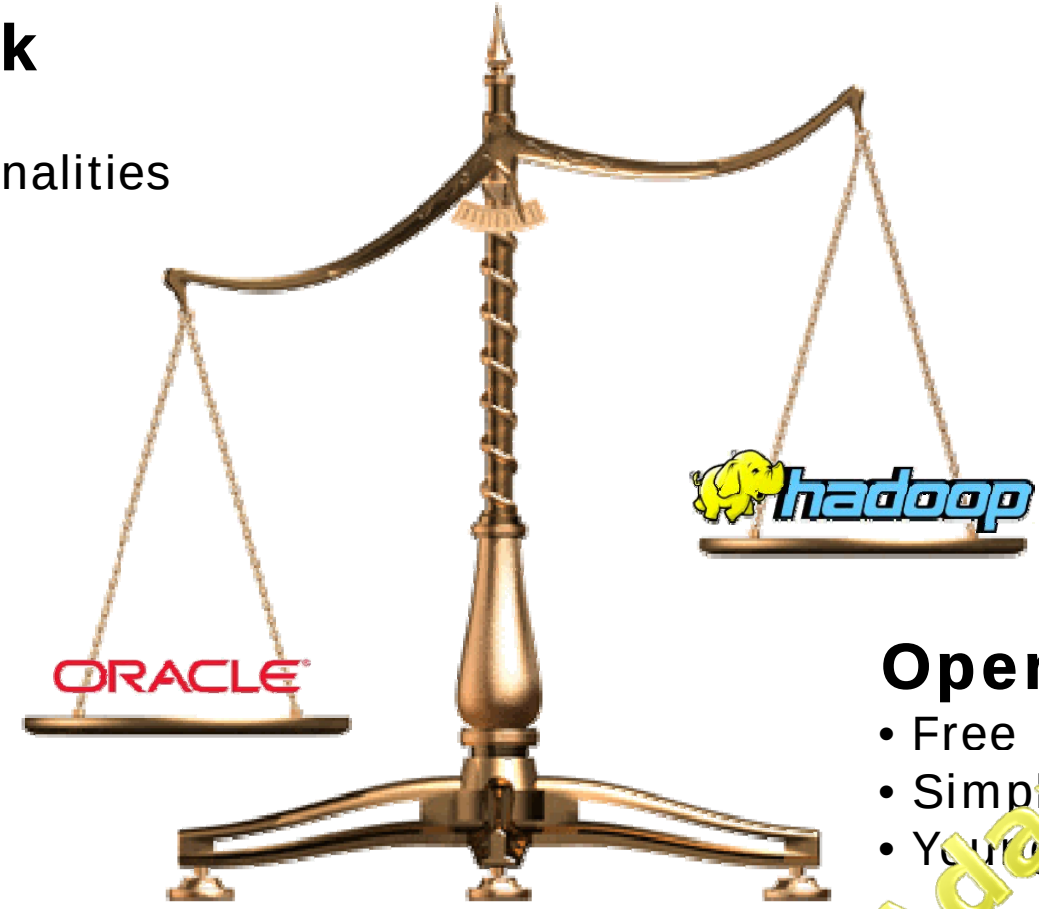


Big elephant or little elephant?

Trademark

- Expensive
- Many functionalities
- Mature

Huge

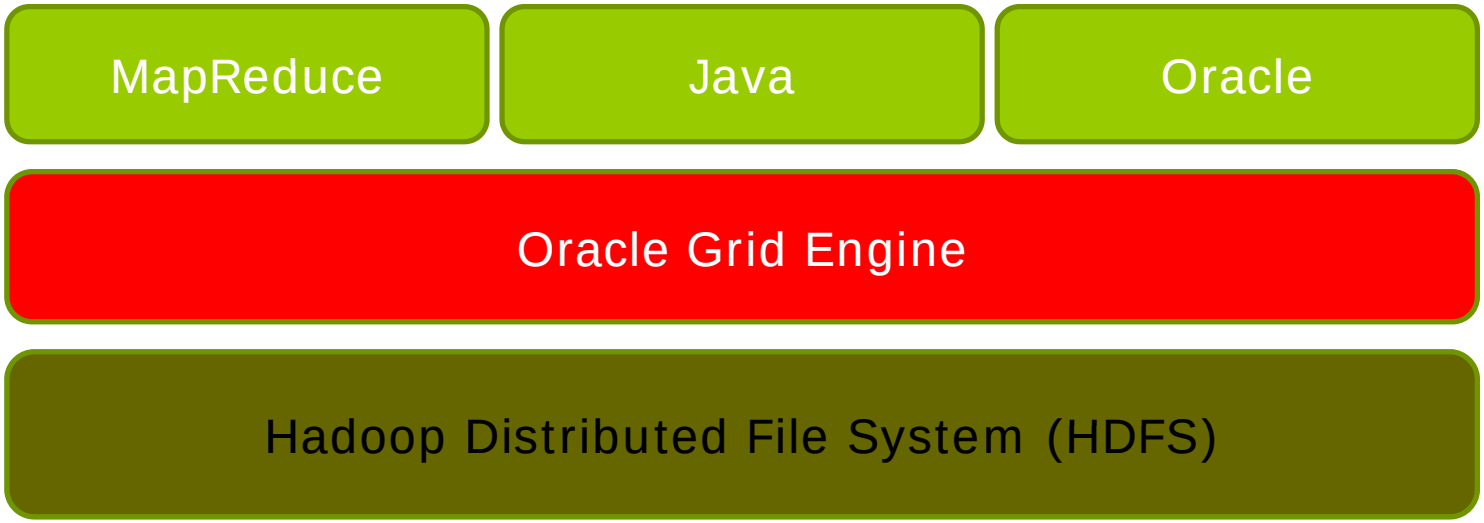


Open source

- Free
- Simple functionalities
- Young

Adaptative

Friends or foes

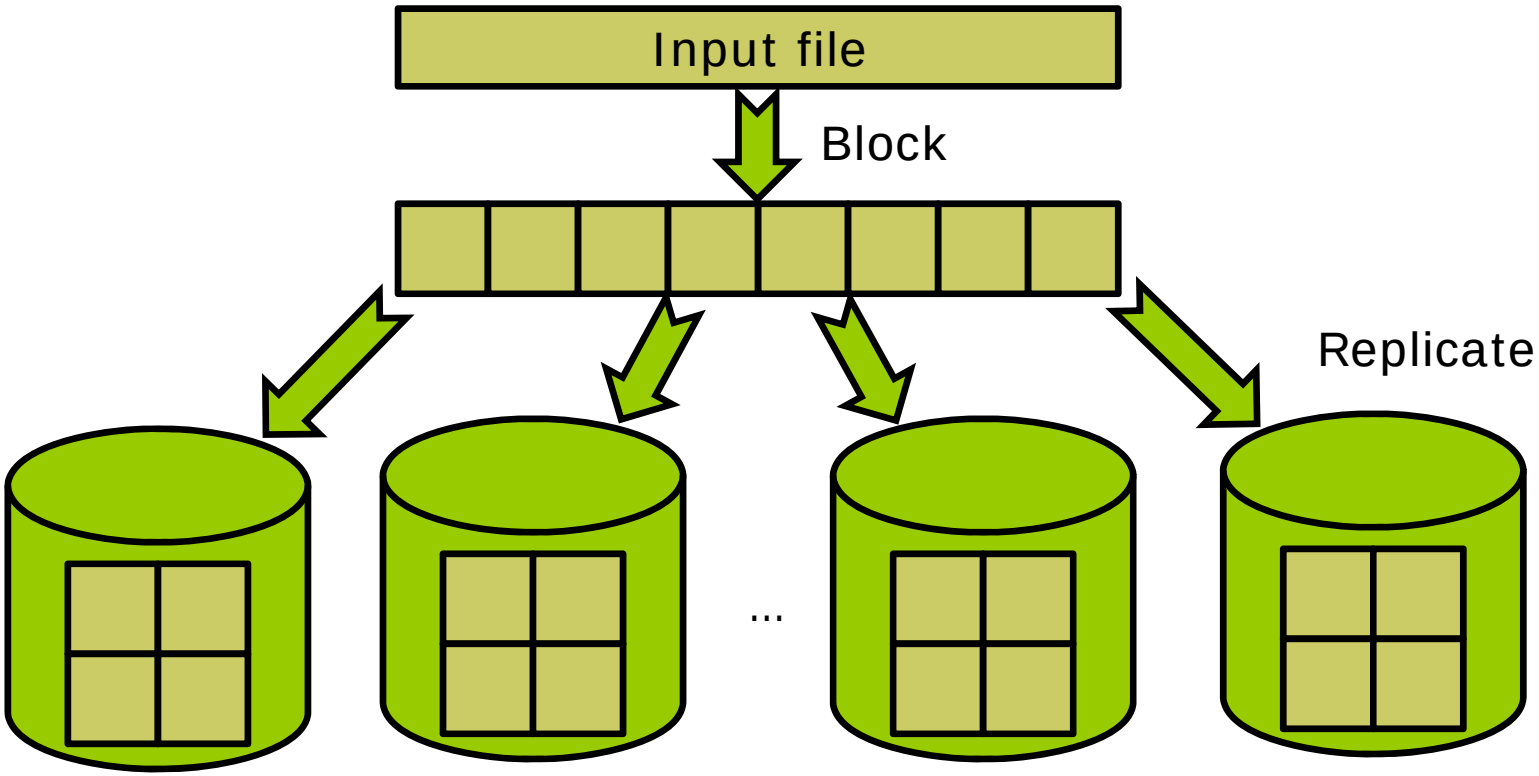


Activity

- *Objective: Understand the usefulness of MapReduce*
- *Tasks:*
  1. *(6') Read two use cases*
  2. *(12') Explain the use cases to the others*
  3. *(5')* Find the main characteristics needed to use Hadoop
  4. *Hand in a prioritized list of characteristics*
- *Roles for the team-mates during task 2:*
  a) *Explains his/her material*
  b) *Asks for clarification of blur concepts*
  c) *Mediates and **controls time***

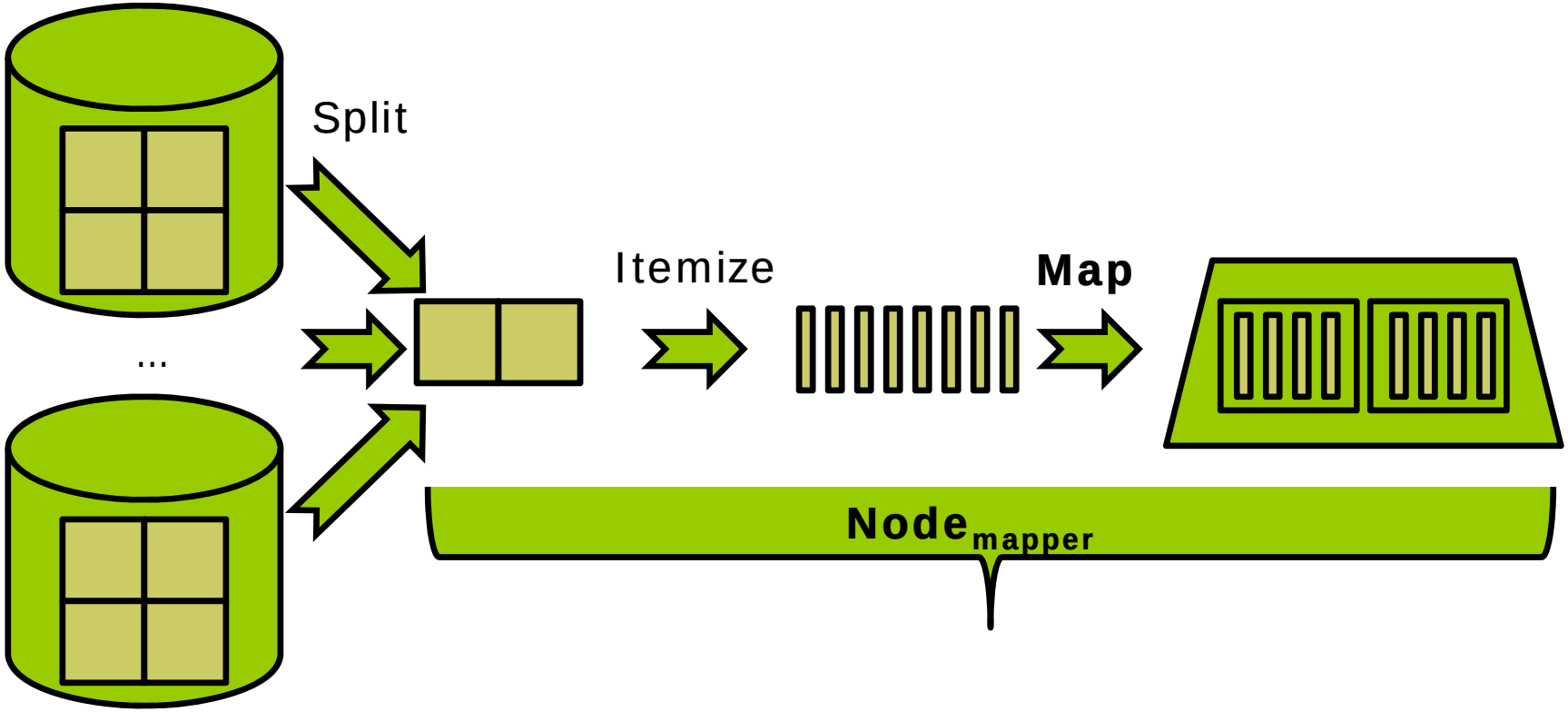
Algorithm: Data Load

1. The input data is partitioned into blocks
   - It can be done by using HDFS or any other storage (e.g., HadoopDB, MongoDB, Cassandra, CouchDB, etc.)
2. Replicate them in different nodes



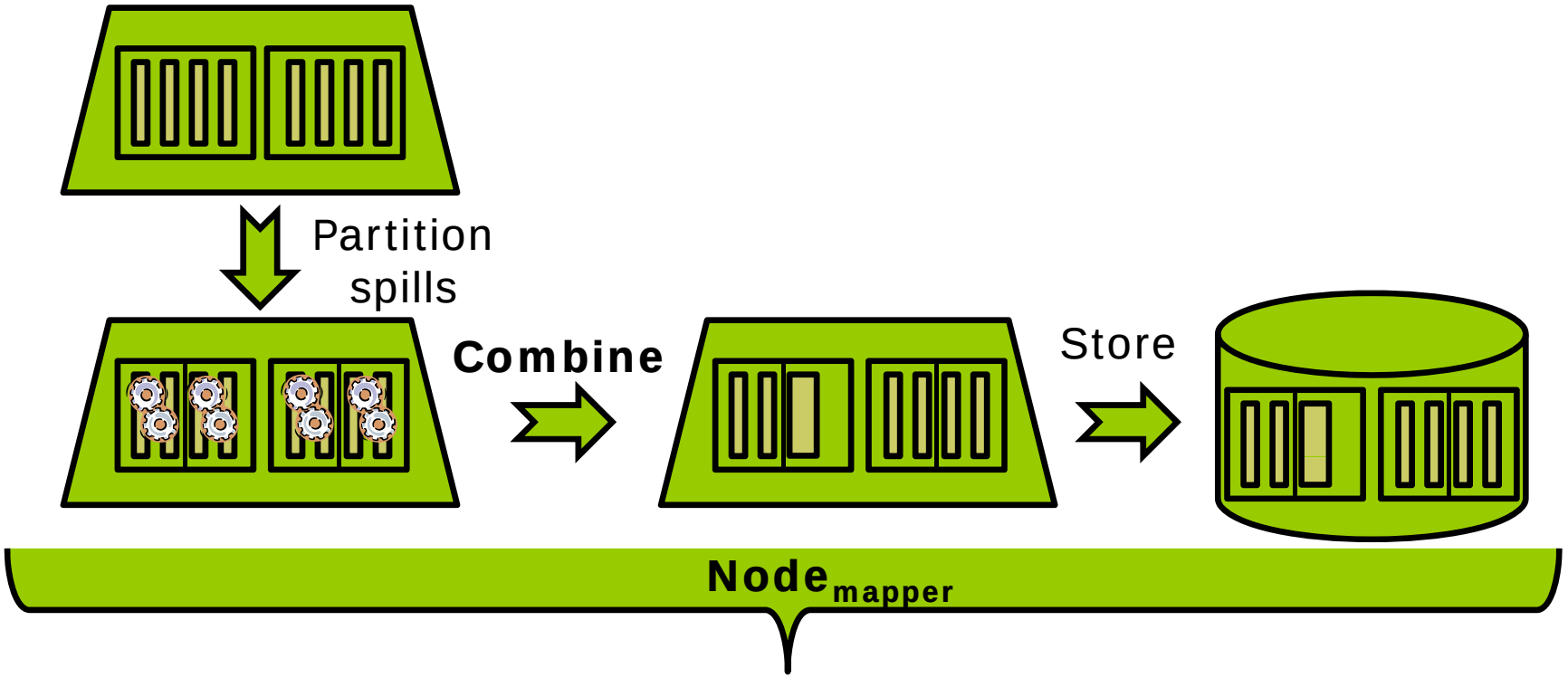
Algorithm: Map Phase (I)

3. Each map subplan reads a subset of blocks (i.e., split)
4. Divides it into records
5. Executes the map for each record and leaves them in memory divided into spills



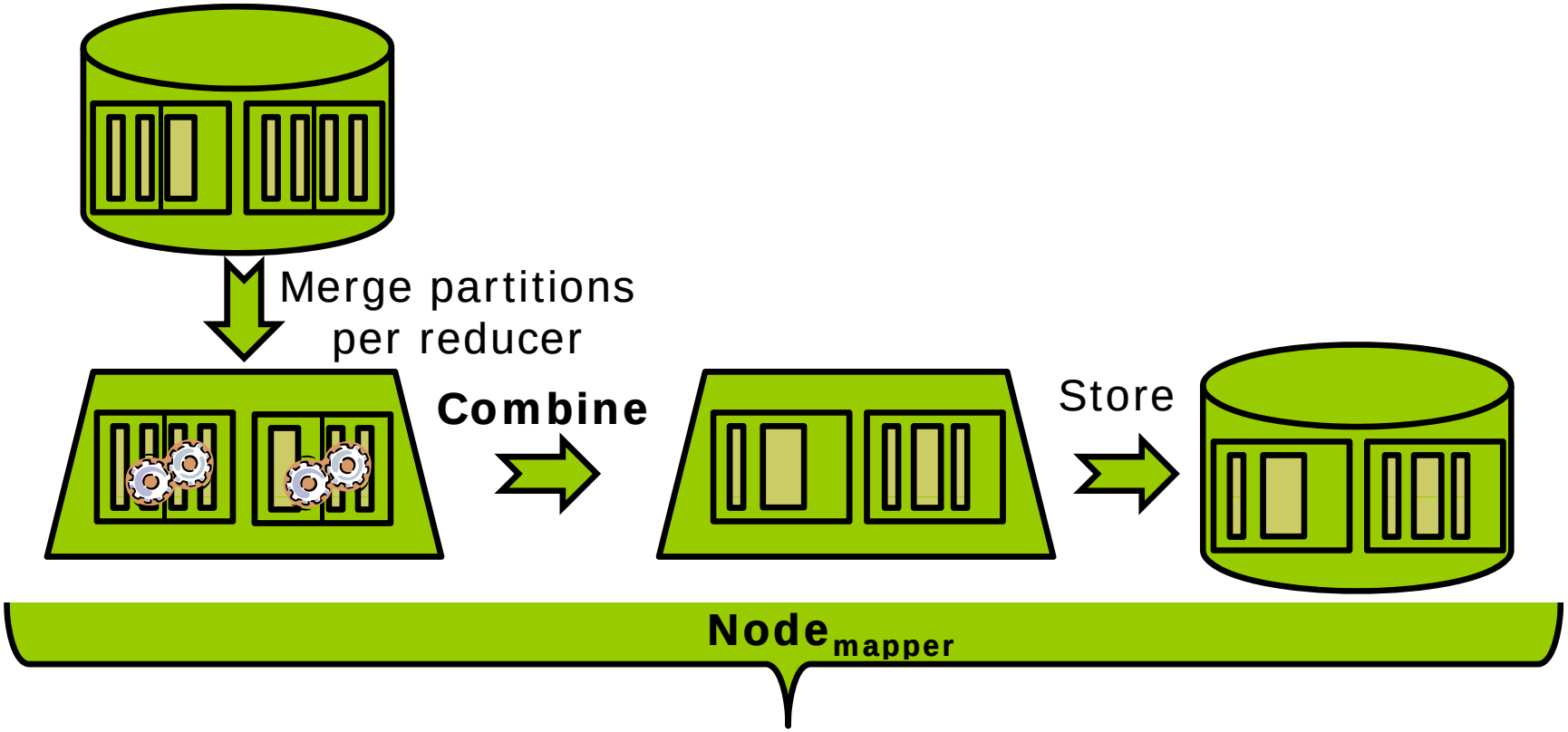
Algorithm: Map Phase (II)

6. Each spill is then partitioned per reducers
   - Using a hash function *f* over the new key
7. Each partition is sorted independently
   - If a combine is defined, it is executed locally after sorting
8. Store the spills into disk (massive writing)



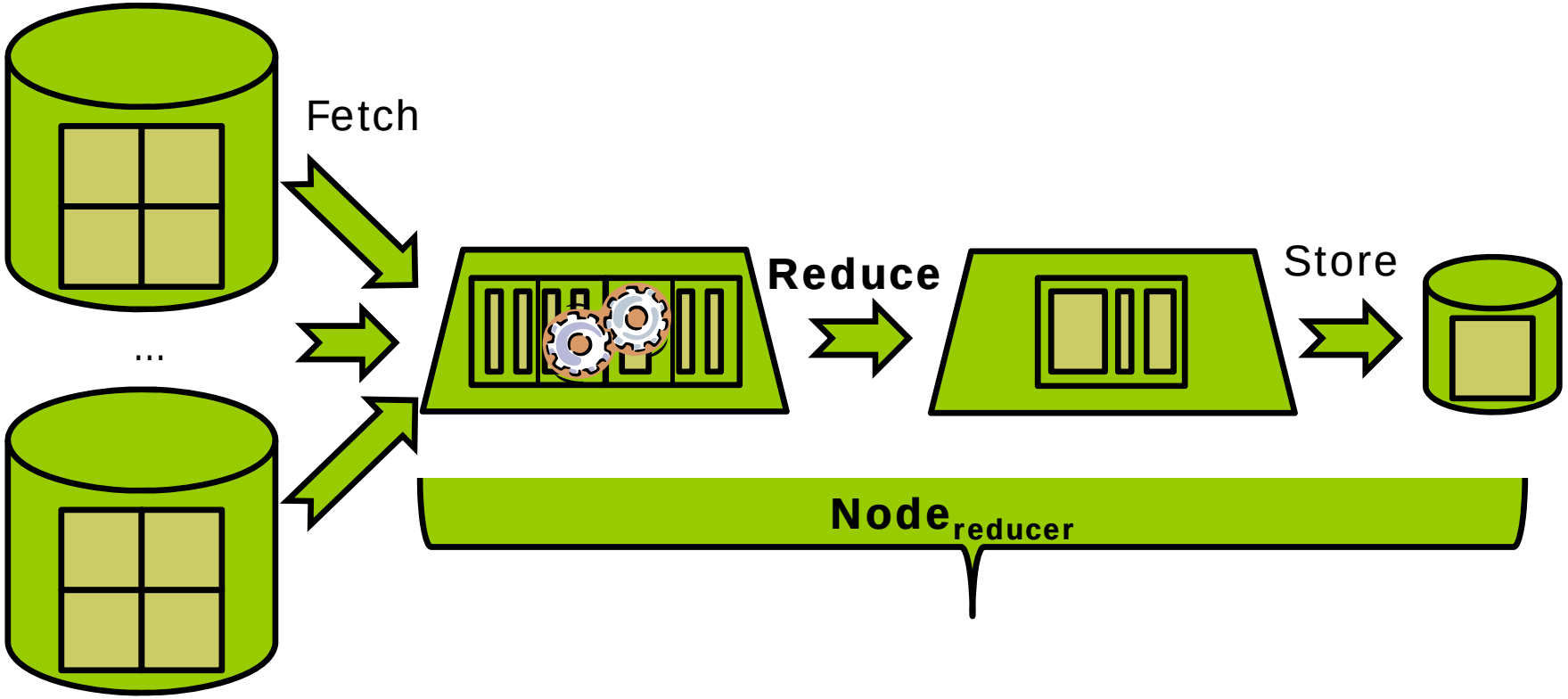
Algorithm: Map phase (III)

9. Spill partitions are merged
   - Each merge is sorted independently
10. Store the result into disk



Algorithm: Shuffle and Reduce

11. Reducers fetch data from mappers (massive data transfer)
12. Mappers output is merged and sorted
13. Reduce function is executed per key
14. Store the result into disk



Local-Global Aggregation: Combine

- Combine is executed locally
  - Assumes uniform random distribution of input
  - Reduces the number of tuples sent to reducers
- Only possible when the reducer function is:
  - Commutative
  - Associative
- Only makes sense if |I|/|O| >> #CPU

Summary

- MapReduce usefulness
- MapReduce benefits
- MapReduce problems
- MapReduce algorithm

Bibliography

- J. Dean et al. *MapReduce: Simplified Data Processing on Large Clusters*. OSDI'04
- P. Sadagale and M. Fowler. *NoSQL distilled*. Addison-Wesley, 2013
- S. Abiteboul et al. *Web data management*. Cambridge University Press, 2011
- M. Stonebraker et al. *MapReduce and parallel DBMSs: friends or foes?* Communication of ACM 53(1), 2010

Resources

- http://hadoop.apache.org
- http://www.cloudera.com
- http://flink.incubator.apache.org
  - Former http://stratosphere.eu
- https://spark.apache.org