CENTENNIAL COLLEGE

School of Engineering Technology and Applied Science

Performance Provisioning in MapReduce

Reza Dibaj

Contents:

- History of Databases
- NoSQL
- Hadoop
- Structured vs Unstructured
- Relational Tables vs Key-Value
- MapReduce
  - Introduction
  - Architecture
  - Basic Concepts
  - DBMS vs MapReduce
  - Advantages and Pitfalls

History

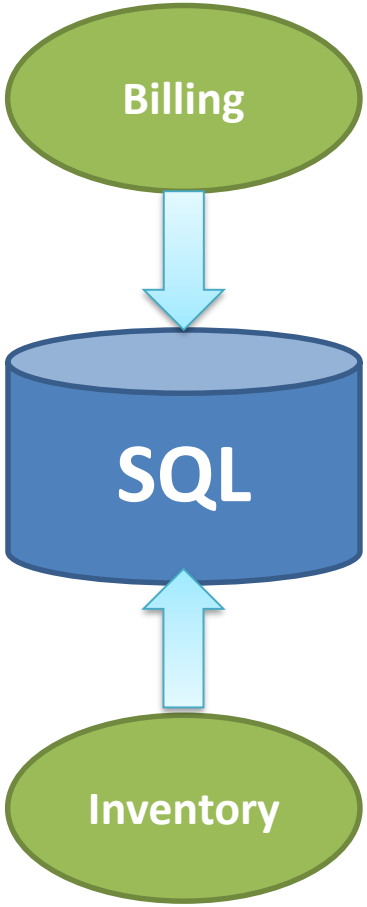
1980s

SQL, a dominant database environment

Data Persistency

Integration and Reporting for many Organizations

Impedance mismatch



History

1990s

Rise of Object Databases

- In memory structure → Save data directly from memory to disk

Still we have Relational Dominance

History

2000s

Amazon
Google } **High Traffic**

limitation scaling up, we will hit a ceiling on RAM/CPU we can use on larger and larger boxes. so we need to resort to soln 2

1st Solution: Scale up the hardware
Buy Bigger Boxes (BBB)

2nd Solution: Scale out the hardware
Buy Lots of Little Boxes "comodity hardware"

Problem: SQL was designed to work on a large box, and not on a large amount of small boxes.

History

2000s **NoSQL Movement**

Some organizations started to work on something rather than traditional SQL environments

Google → BigTable (Column Family) sth similar to excel

Amazon → Dynamo (Key-Value)

regulations to write data for RDBMS
regulations to READ data for NoSQL

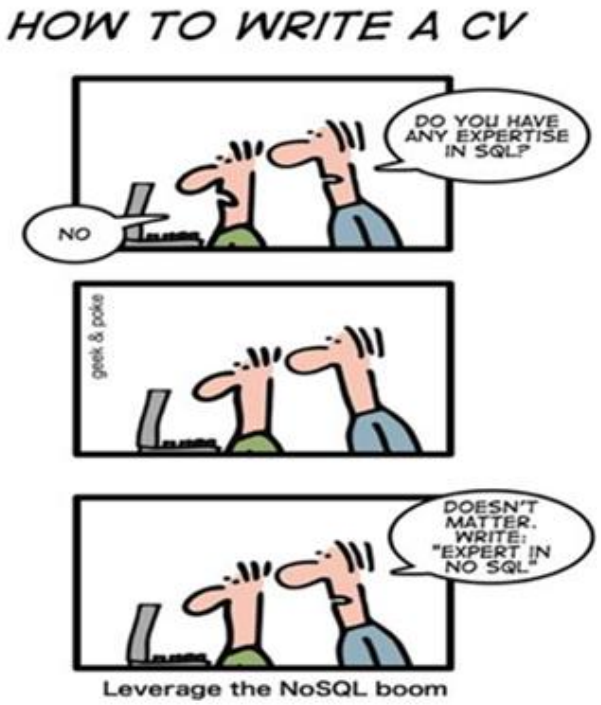
NoSQL

Whenever we talk about Big Data we talk about unstructured data.

Some NoSQL Characteristics

- Non-relational (Unstructured)
- Cluster-friendly
- Open-source

referring to the commodity hardware limitation we discussed prev



NoSQL

NoSQL Data Models

- Key-value
- Document data model
- Column-family

} Aggregate-Oriented, Not ACID

- Graph — Graphs are ACID (Atomic, Consistent, Isolated, Durable)

All NoSQL Data Models are Schema-less

Aggregate-Oriented DBs are naturally fit-in storing data on large clusters.

Graphs are good for handling relationship between things. Even relational Databases work with a high degree of complexity with relationships, while graphs are very good at this feature. There is a kind of query language specifically designed for that.

NoSQL

NoSQL

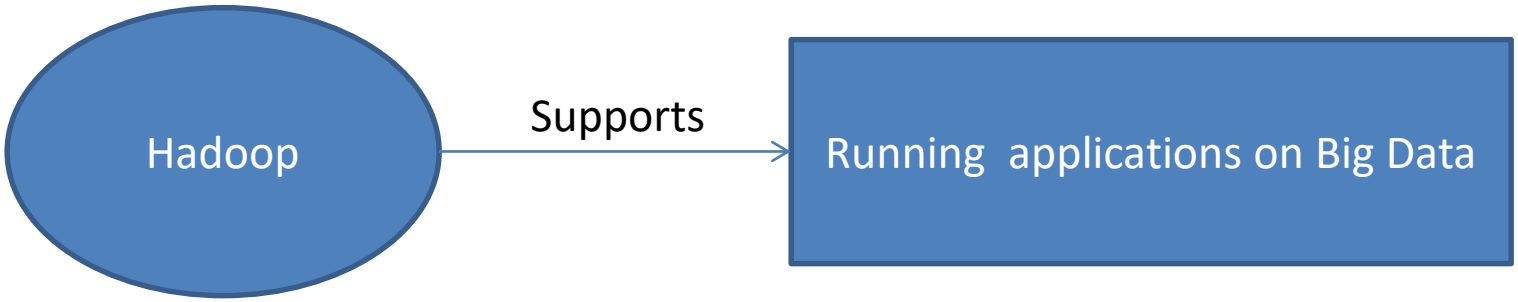
- Easier development
- Large-scale data (Big Data)

Is NoSQL the future of Databases?

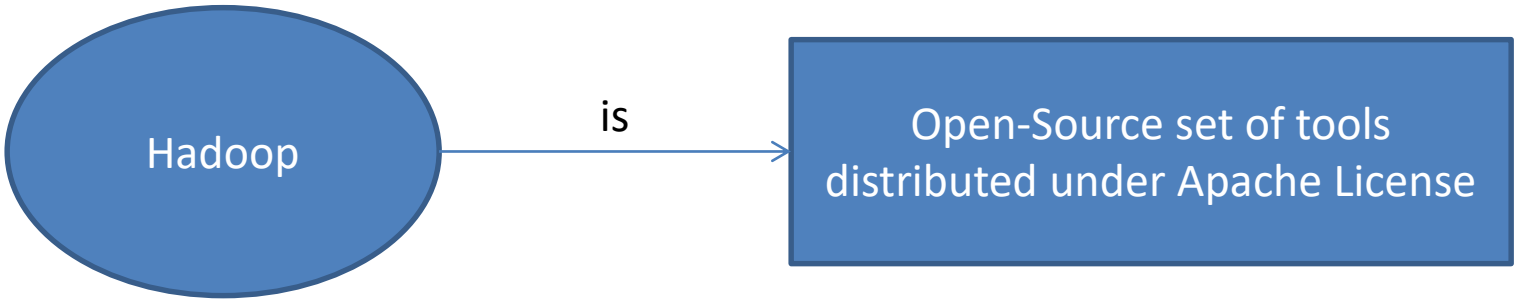
We will have different models of Databases, and we should chose the appropriate one for our specific problem.

will not replace RDBMS, they have their own uses

Hadoop Objectives



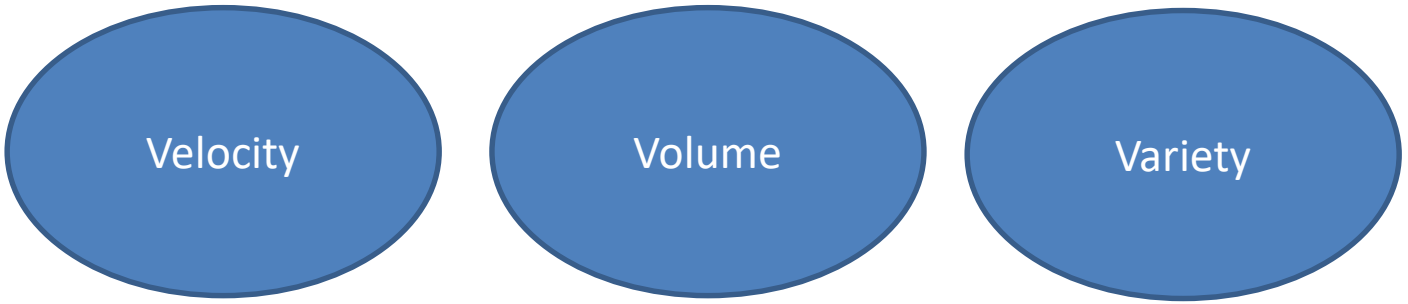
Open-Source Environment



d3js.org
https://beta.observablehq.com/@mbostock/d3-chord-dependency-diagram

Big Data Challenge Points

The 7's of Big Data: https://impact.com/marketing-intelligence/7-vs-big-data/



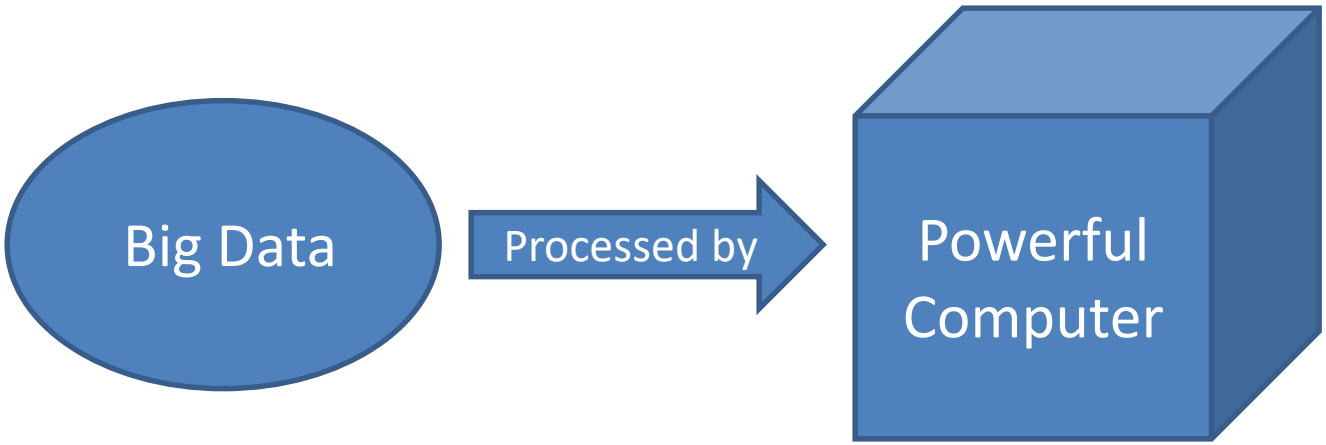
variability, veracity, visualization, value

visualization: tool to bring information to knowledge

variability:

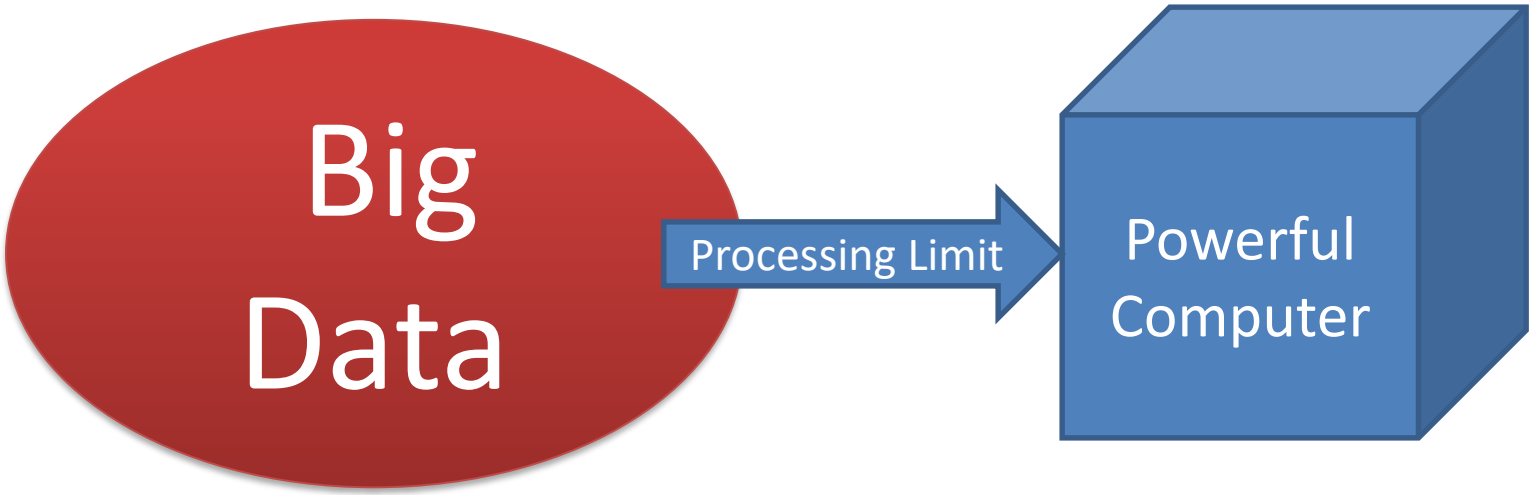
Traditional Approach

Enterprise Approach



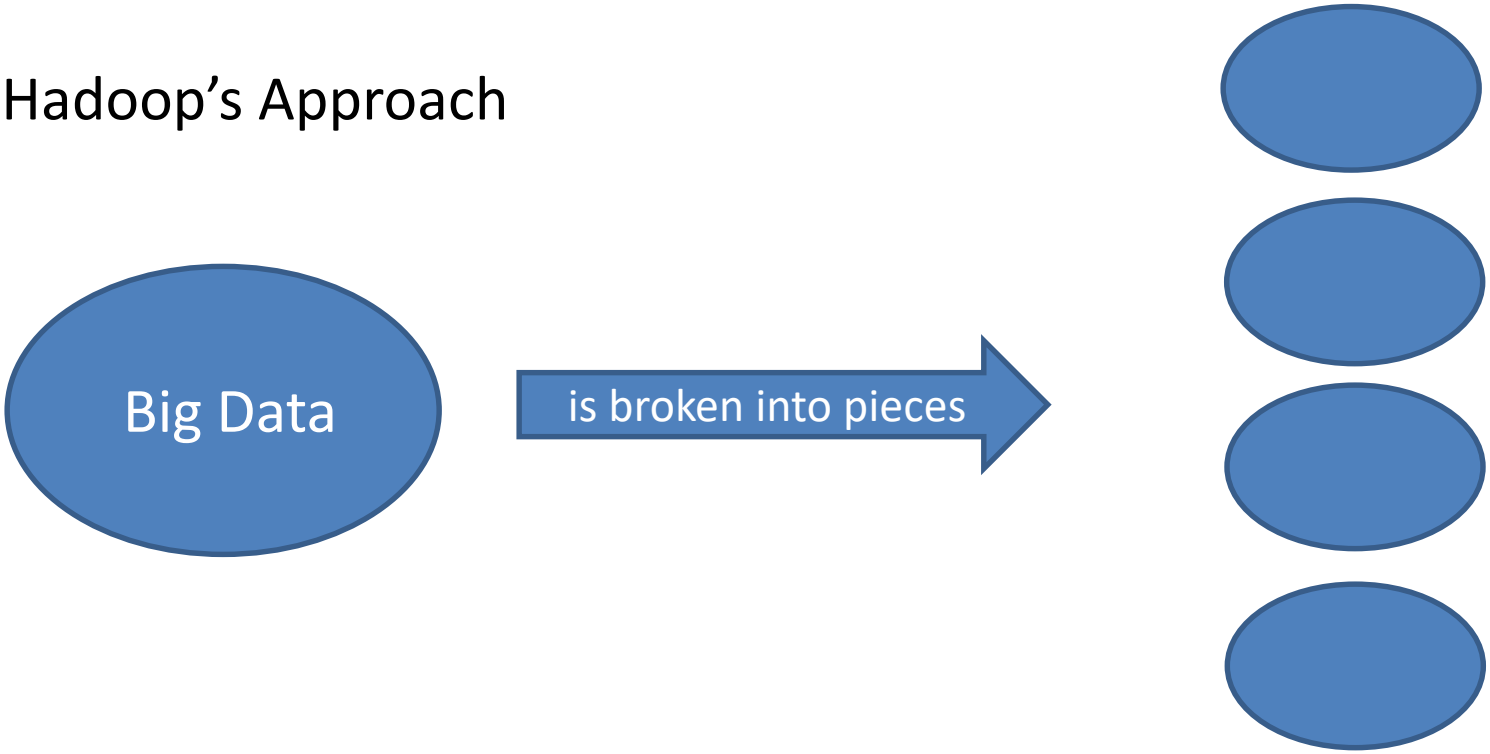
Traditional Approach

Enterprise Approach



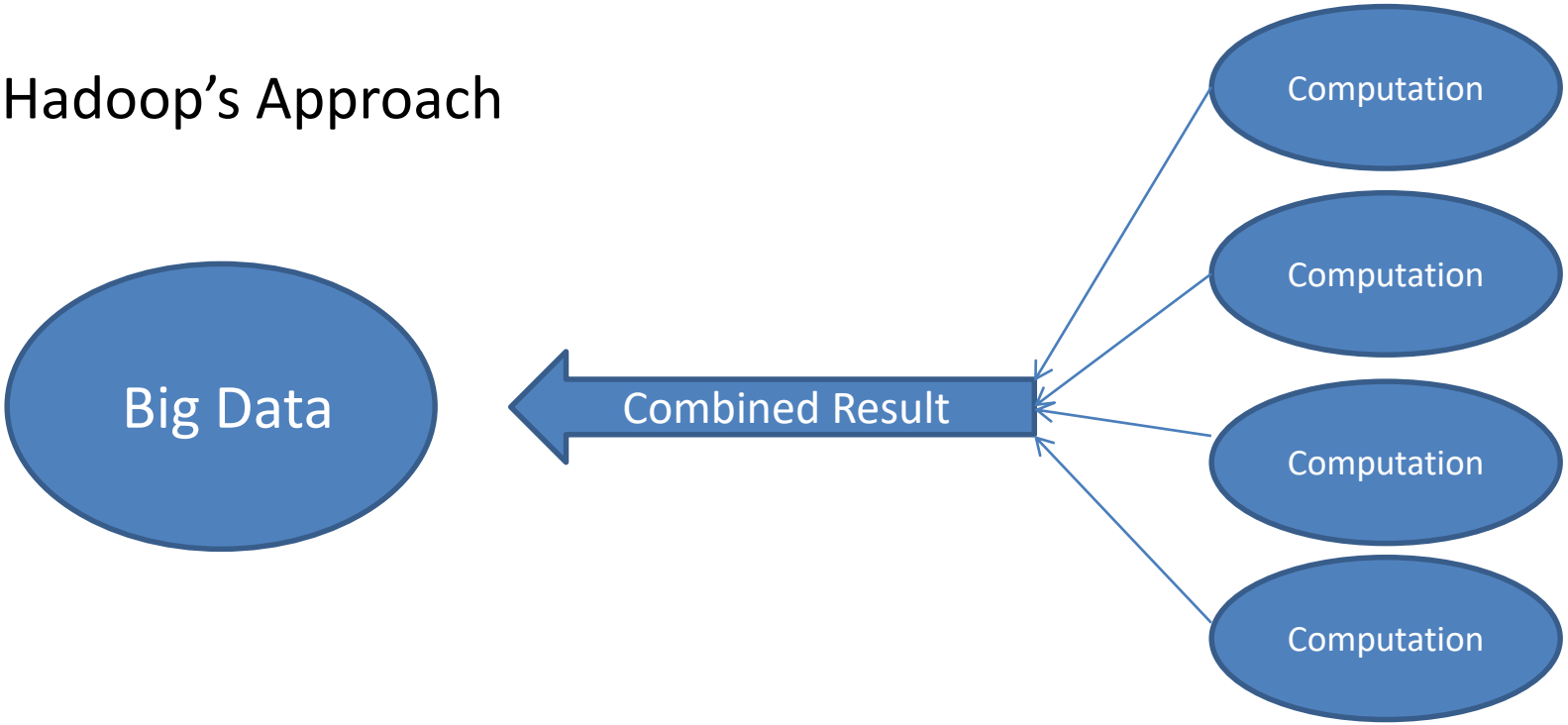
Breaking the Data

Hadoop's Approach

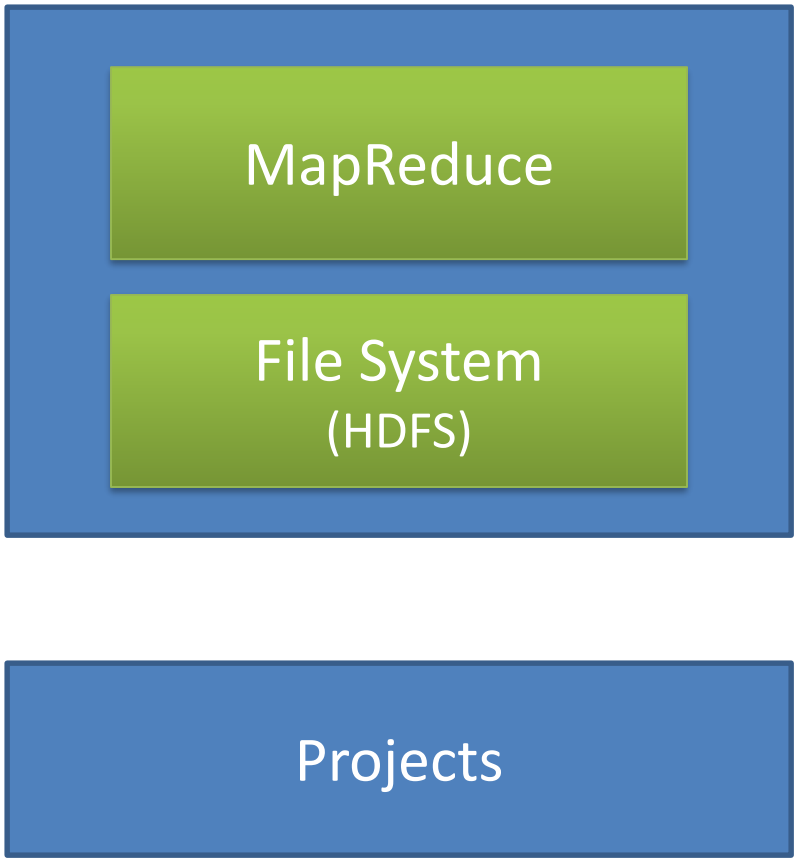


Move Computation to the Data

Hadoop's Approach

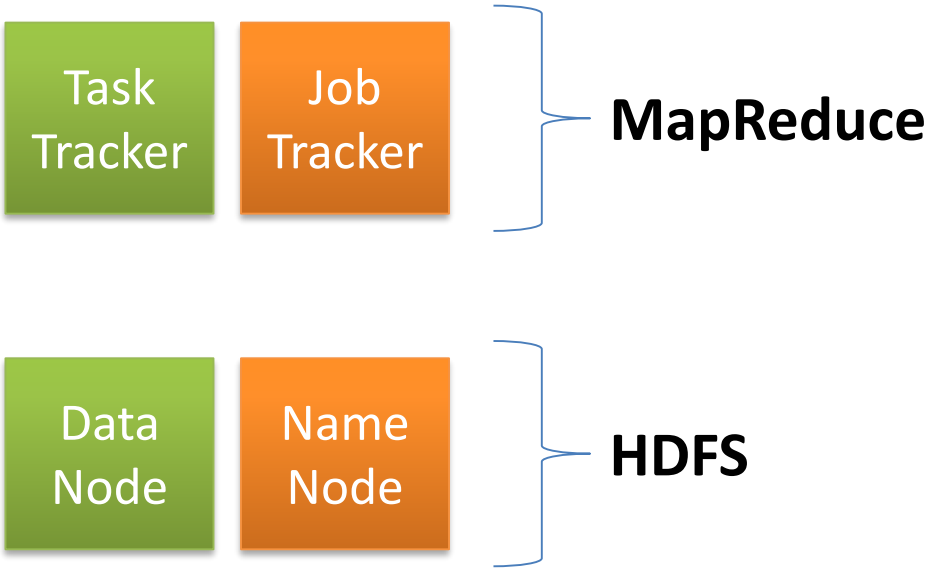


Hadoop's Architecture



MapReduce and HDFS

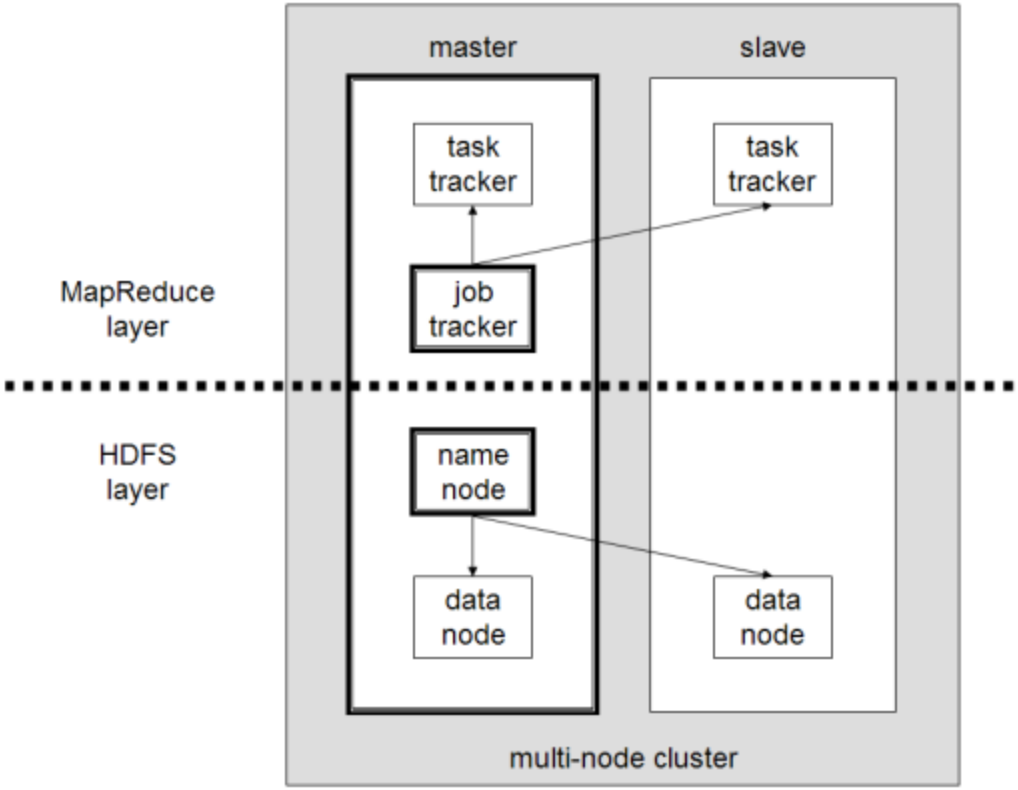
job split into smaller tasks



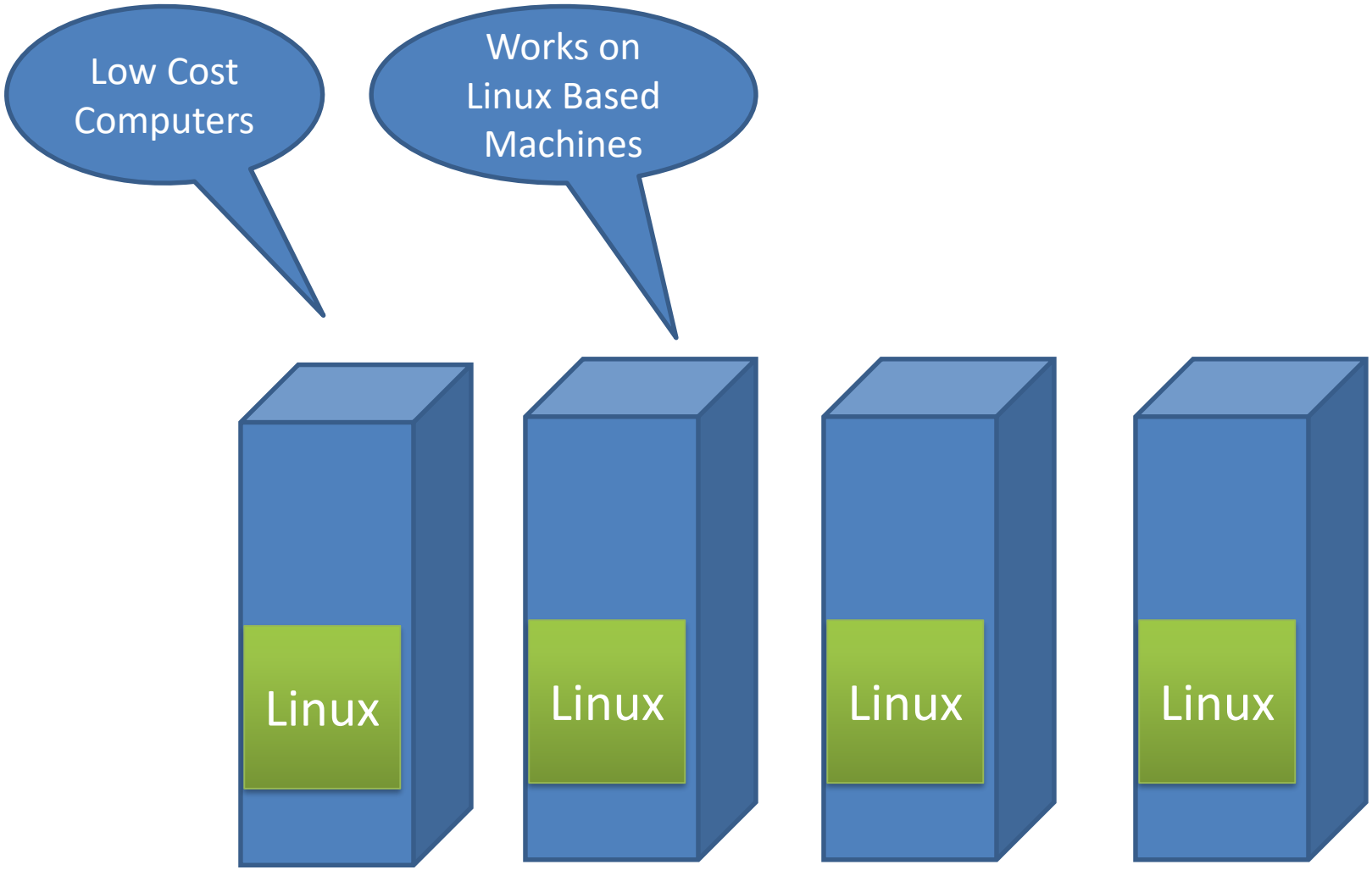
Any application contacts the Master Node.
Then the task will go into the queue.

Hadoop's Architecture

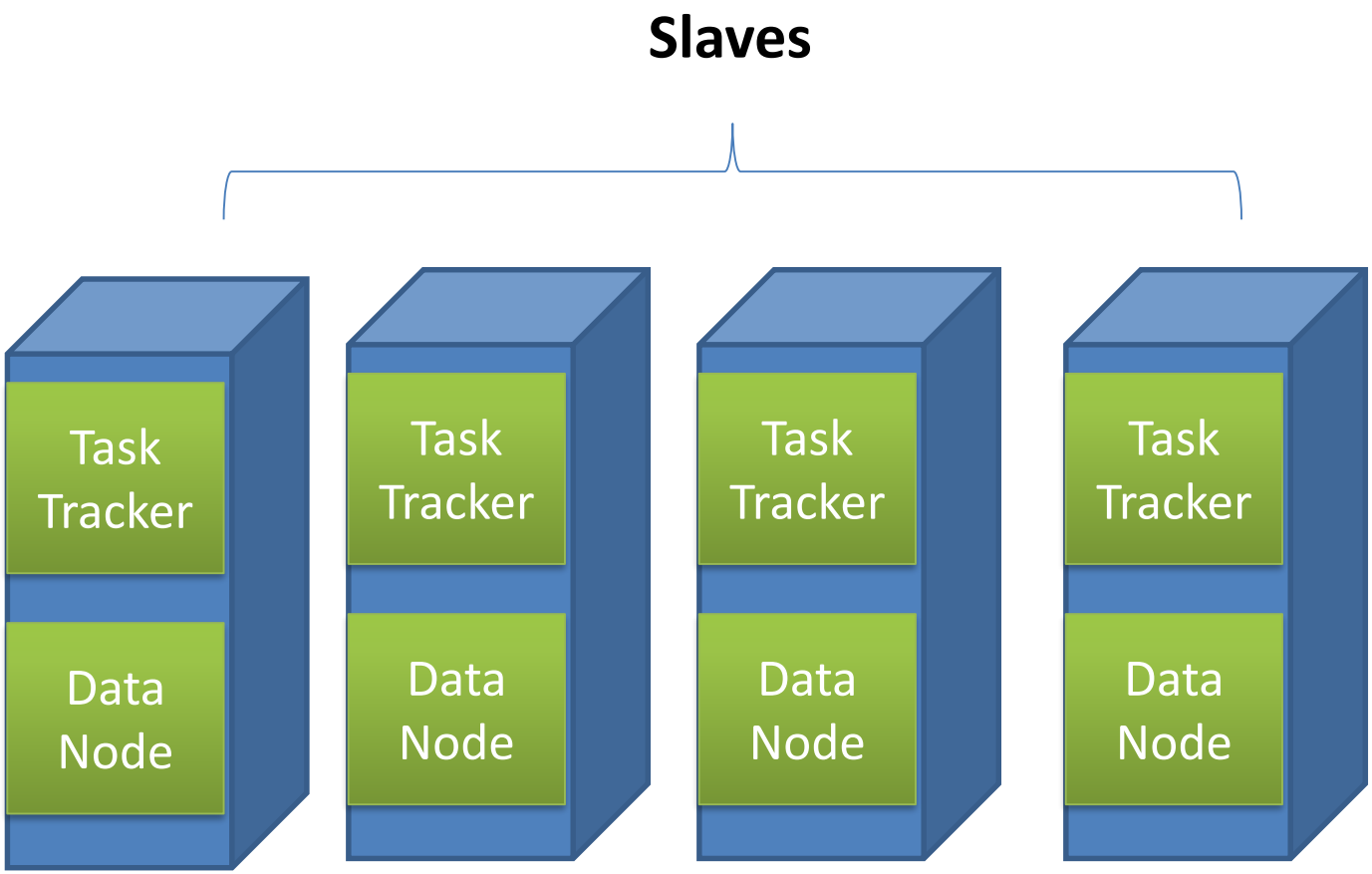
master splits job into tasks for slaves; it can also take on role of slave and do work too in its free time after delegating the work and before collecting results from slaves



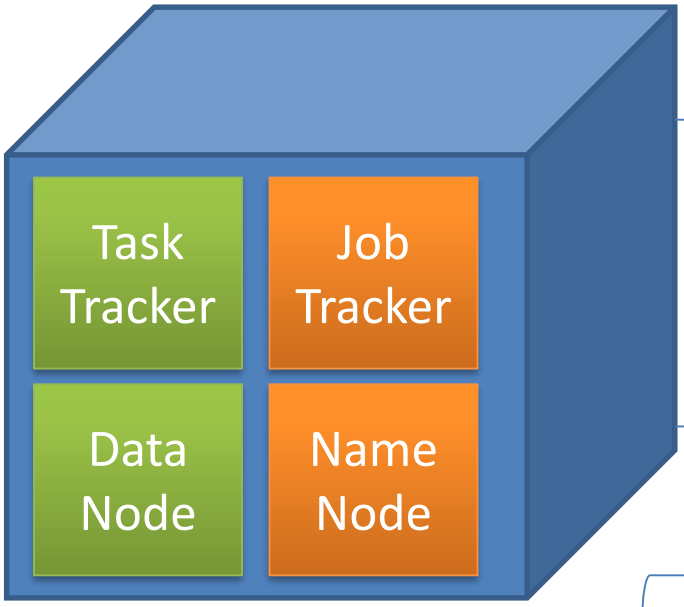
Distributed Model



Task Trackers and Data Nodes



Master



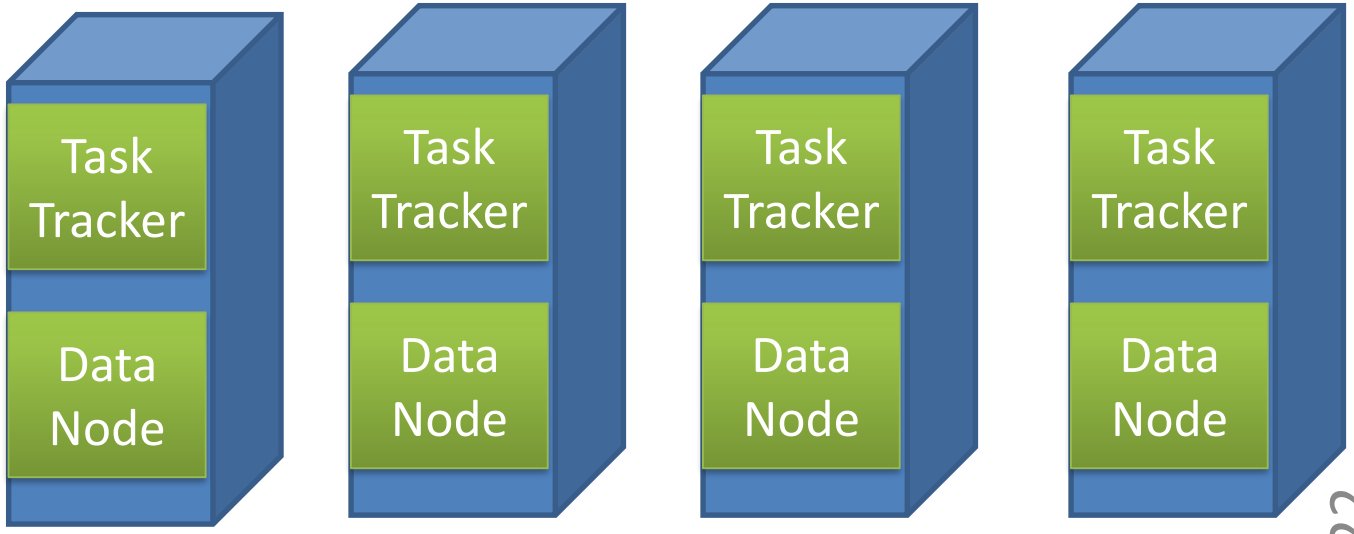
Master

splitting job into tasks and the I/O related to this process is the most time consuming task in this master -> slave process

this I/O is 200x slower than when compared to the I/O in a central model

however, after the I/O bottleneck the performance goes through the roof :)

Slaves



MapReduce and HDFS

Job Tracker

Break higher bigger task to smaller pieces and to send each smaller piece of computation to the task trackers. It also can combine the results and send it back to the application.

Name Node

It is responsible to keep the index that which data is resides on which data node. When application contacts a Name Node, it tells the application that in order to get your particular data; you should go to which computer.

An Introduction to MapReduce

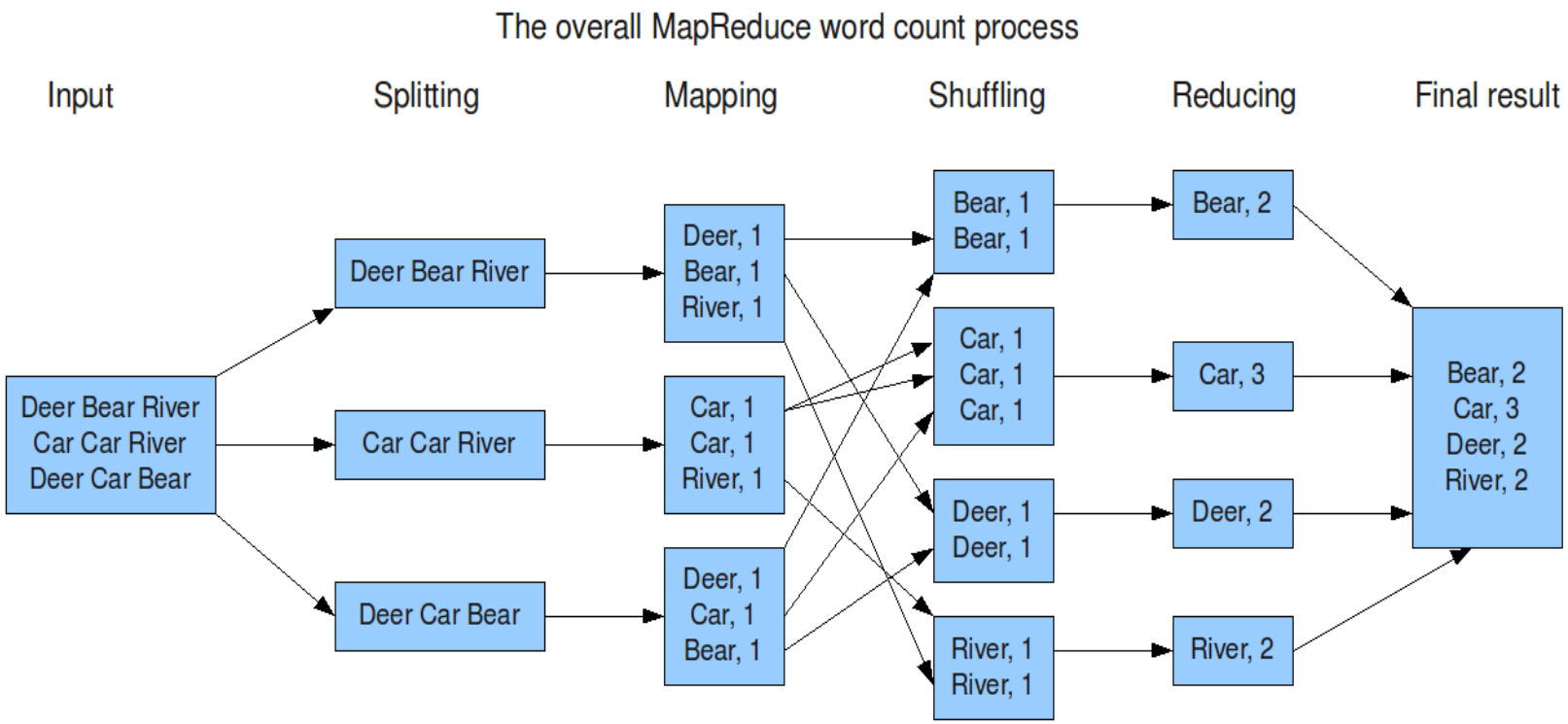
MapReduce

- Simple
- Scalable
- Fault-tolerant

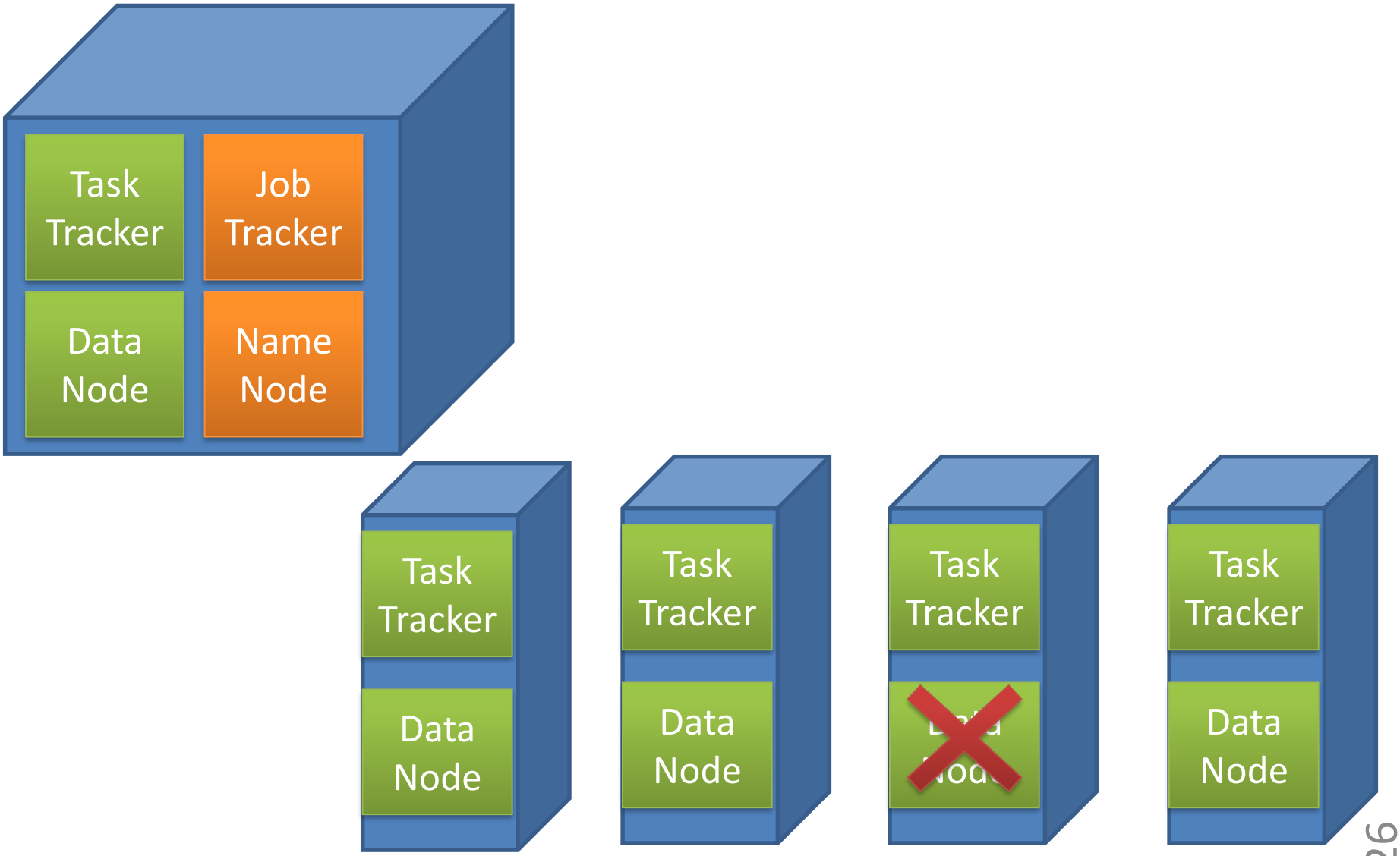
- limitations on its performance and efficiency

this is why we will not replace RDBMS:
1. performance
2. security

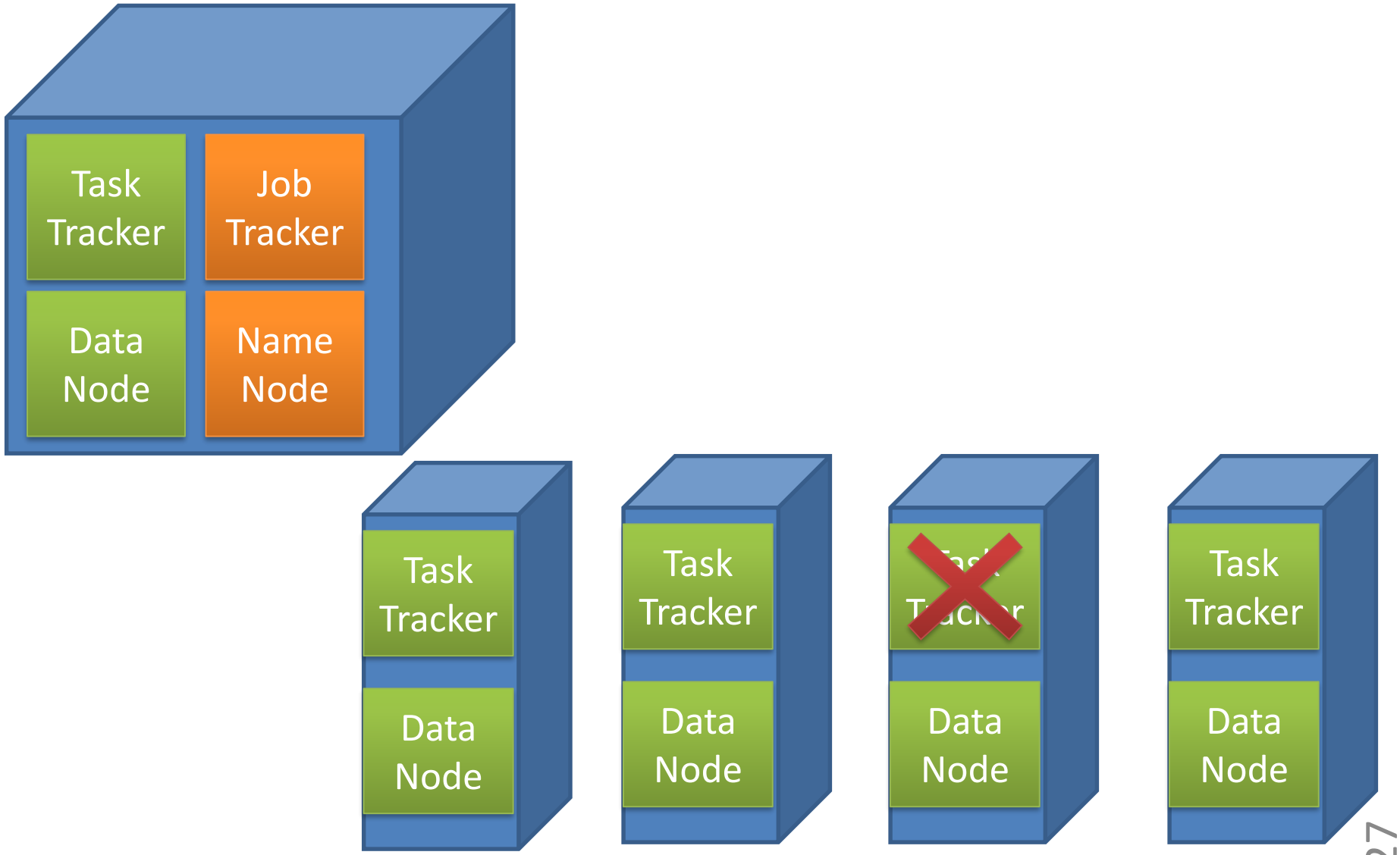
MapReduce Architecture



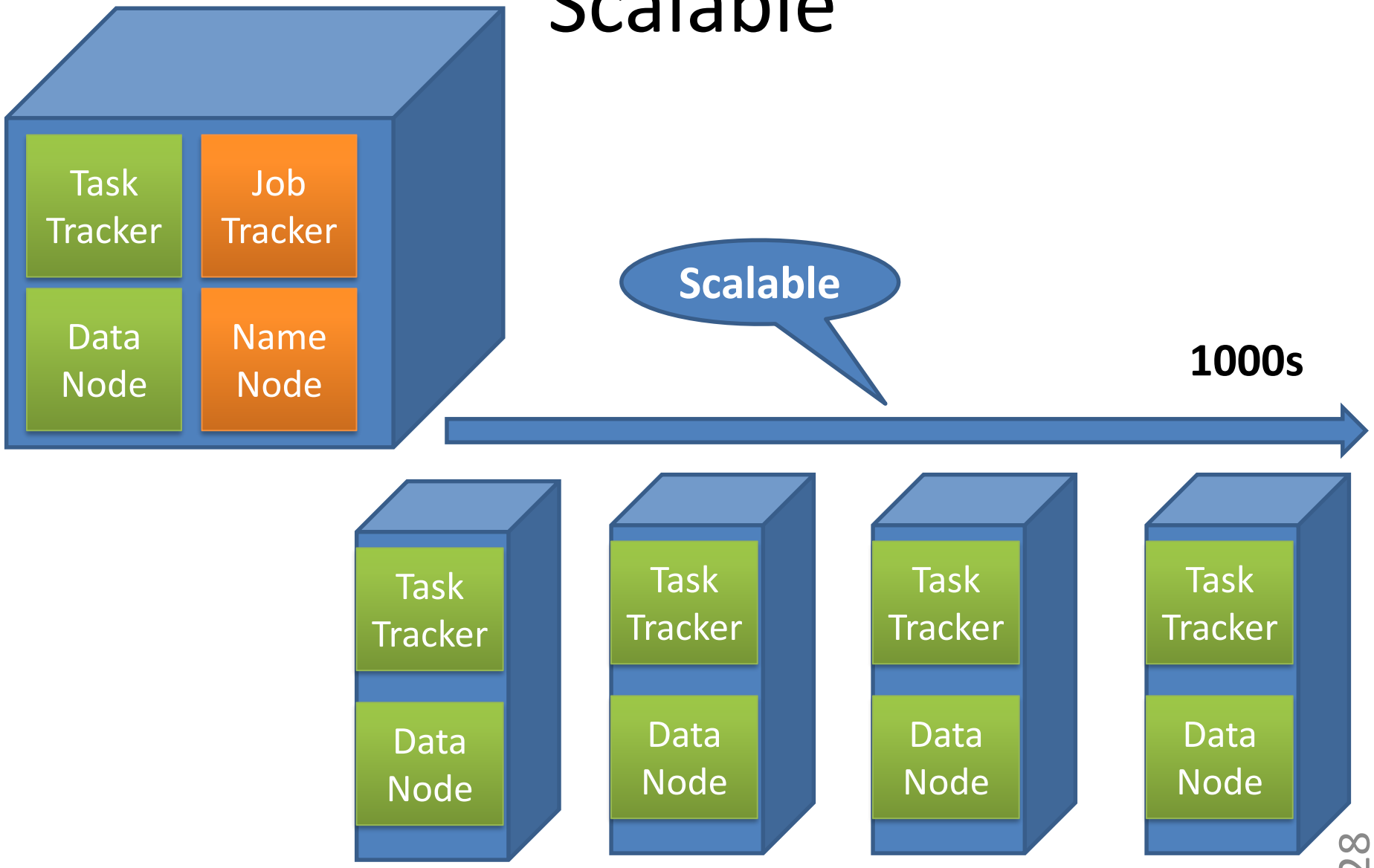
Fault Tolerance for Data



Fault Tolerance for Processing



Scalable


Task Tracker
Job Tracker
Data Node
Name Node
Scalable
1000s
Task Tracker
Data Node
Task Tracker
Data Node
Task Tracker
Data Node
Task Tracker
Data Node

Easy Programming

Programmers

Do not have to worry about

- Where the program is located
- How to manage the failures
- How to break computations into smaller pieces
- How to program for scaling

Easy Programming

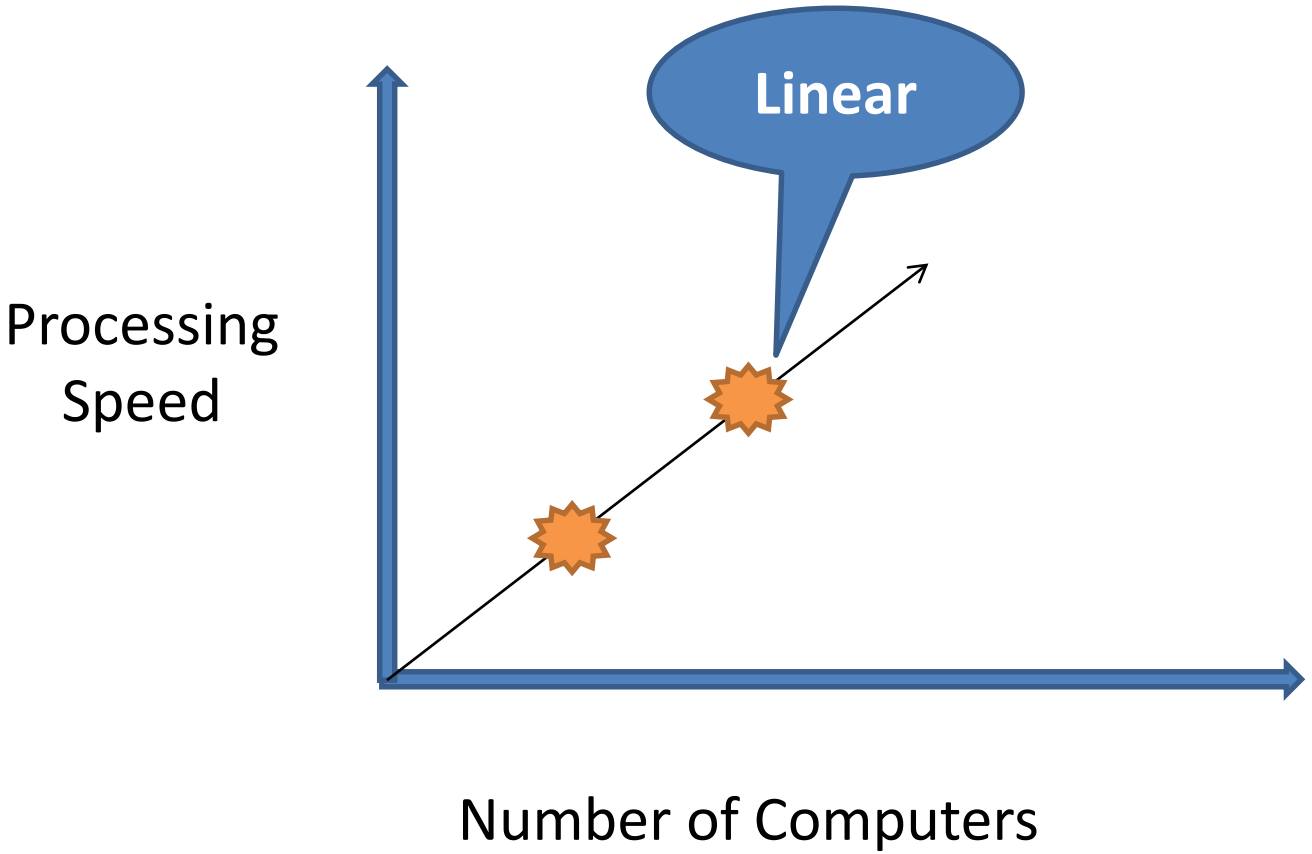
Programmers

Could focus on

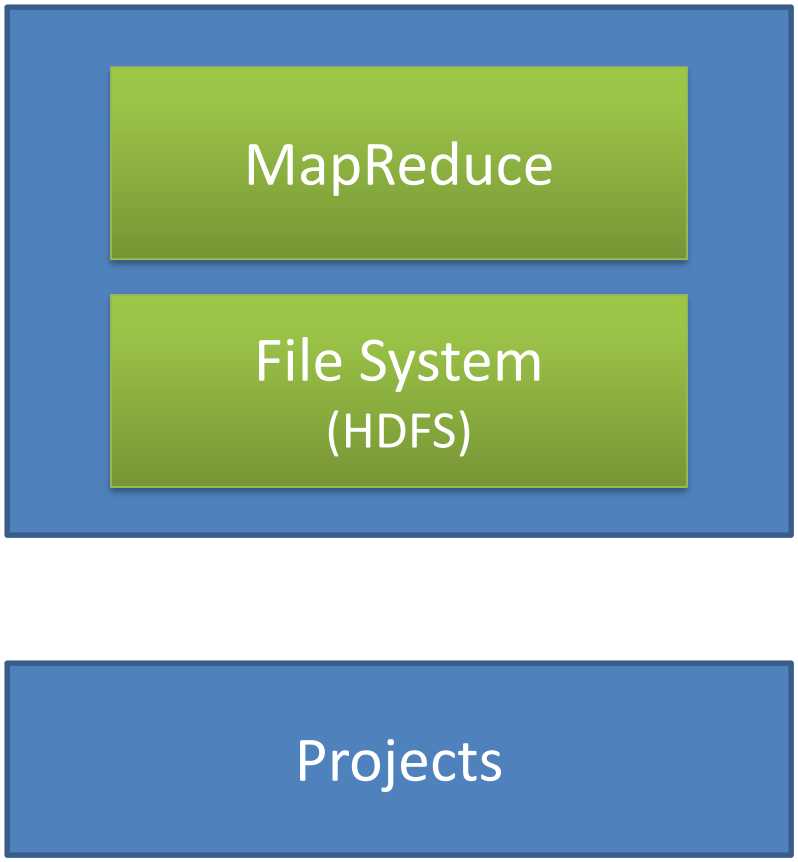
Writing Scale-free Programs

apache hive
apache mahout

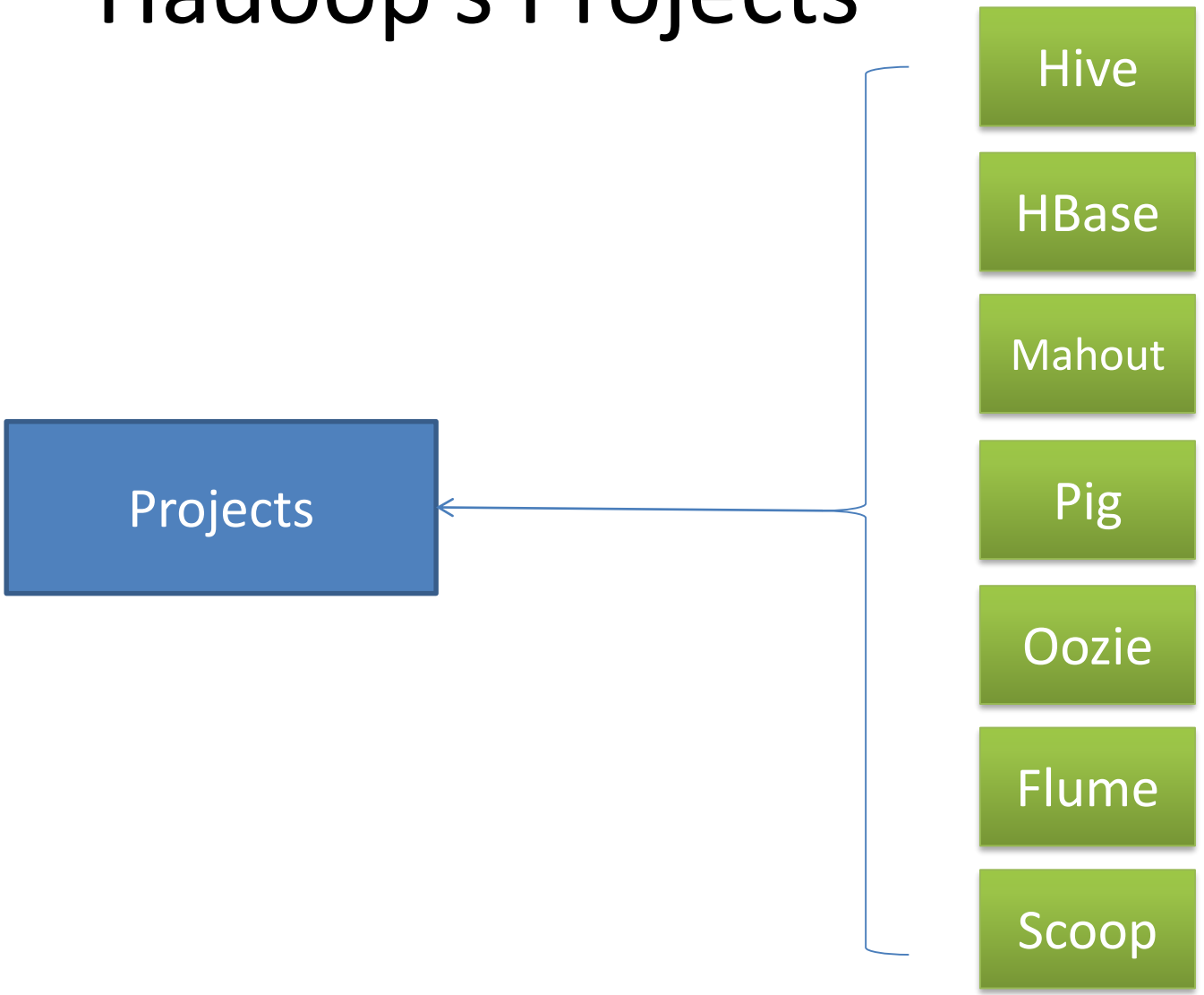
Scalability Cost



Hadoop's Projects



Hadoop's Projects



Little about Hadoop's History

2004-2005

Hadoop was created in 2005 by Doug Cutting and Michael Cafarella who both worked for Yahoo!

2006

In 2006 the project was donated to Apache.

Hadoop's Users

Administrators

- Installation
- monitoring/managing the system
- tuning the system

Users

- Design Applications
- Import/Export Data
- Work with tools

Hadoop's Usage Areas

- Social Media
- Retail
- Financial Services
- Searching Tools
- Governments
- Intelligence

Hadoop's User Companies

- Yahoo
- Facebook
- Amazon
- eBay
- American Airline
- The New York Times
- Federal Reserve Board
- Chevron
- IBM

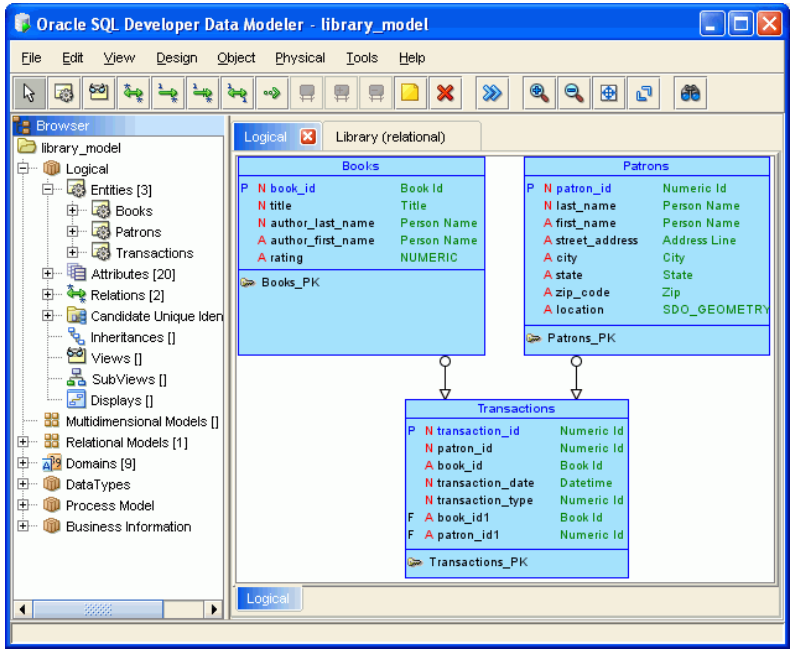
Outlook of Hadoop

- By 2015 more than 50% of Enterprise Data was processed by Hadoop

Structured vs Unstructured

SQL

is by design targeted at structured data.

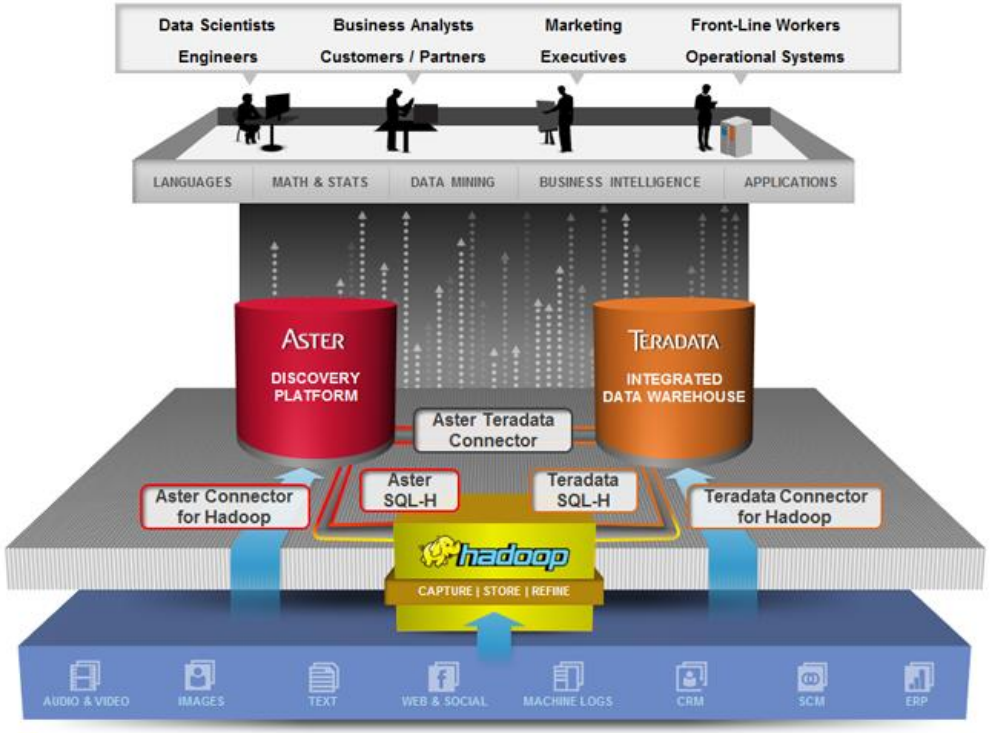


Hadoop

many of its initial applications deal with unstructured data such as text.

Structured vs Unstructured

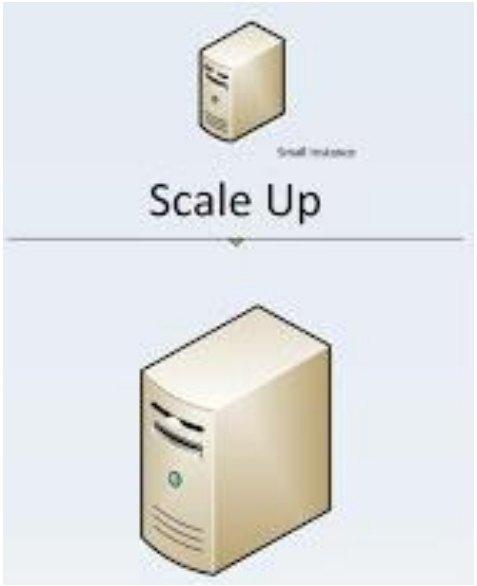
SQL and Hadoop can be complementary, as SQL is a query language which can be implemented on top of Hadoop as the execution engine.



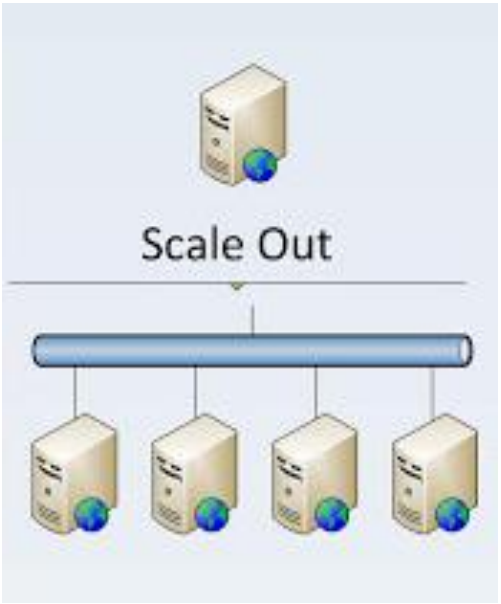
can have hive, pig, etc. on top of hadoop

Structured vs Unstructured

SQL design is more friendly to **scaling up**.

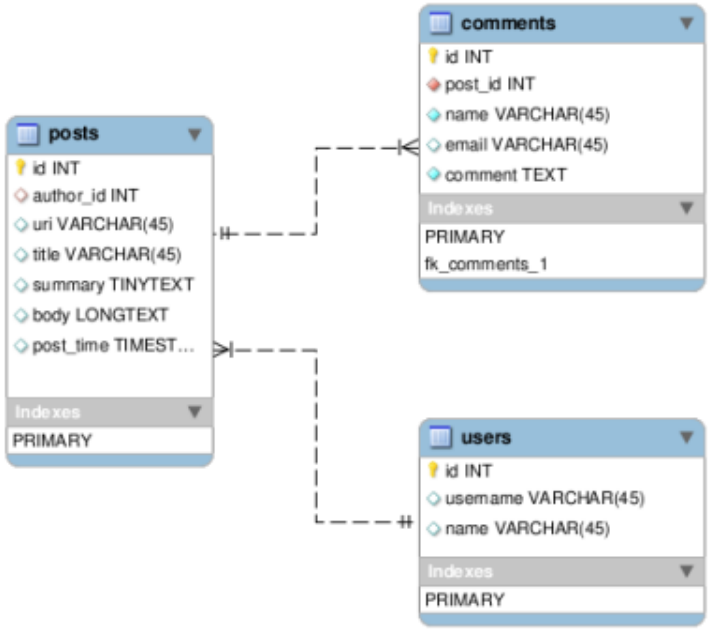


Hadoop is designed to be a **scale-out** architecture operating on a cluster of commodity PC machines.

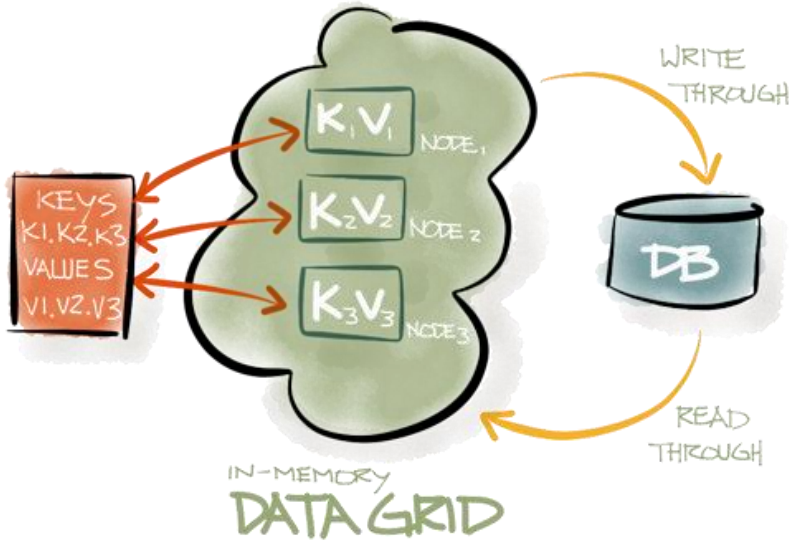


Relational Tables vs Key-Value

In **SQL** data resides in tables having relational structure defined by a schema.



Hadoop uses key/value pairs as its basic data unit, which is flexible enough to work with the less-structured data types.



Declarative Query vs Functional Programming

Under **SQL** you have query statements.

```
SELECT Book.title AS Title, COUNT(*) AS Authors
FROM Book
JOIN Book_author
ON Book.isbn = Book_author.isbn
GROUP BY Book.title;
```

Under Hadoop you have scripts and codes. MapReduce allows you to process data in a more general fashion than SQL queries.

```
function map(String name, String document):
// name: document name
// document: document contents
for each word w in document:
        emit (w, 1)

function reduce(String word, Iterator partialCounts):
// word: a word
// partialCounts: a list of aggregated partial counts
sum = 0
for each pc in partialCounts:
        sum += ParseInt(pc)
emit (word, sum)
```

DBMS vs MapReduce

DBMS

- Generates a query plan tree for execution

MapReduce

- A plan for executions in MapReduce is determined entirely at runtime

Pros and Cons

Debate

- DBMSs have adopted “one size fits all” strategy and are not suited for solving extremely large scale data processing tasks.
- MapReduce is referred to as a new way of processing big data in data-center computing.

Slow

- Hadoop is 2∼50 times slower than parallel DBMS except in the case of data loading.

Poor Efficiency

- Hadoop system is scalable, but achieves very low efficiency per node, less than 5MB/s processing rates, repeating a mistake that previous studies on high performance systems often made by “focusing on scalability but missing efficiency”
- Loosing efficiency for the fault tolerance: MapReduce increases the fault tolerance of long-time analysis by frequent checkpoints of completed tasks and data replication.

MapReduce Advantages and Pitfalls

- Simple and easy to use
- Flexible
- Independent of the storage
- Fault-tolerance
- High scalability

- No high-level language
- No schema and no index
- A single fixed dataflow
- Low efficiency
- Very young

indexes are for making processing faster

hadoop has sth like indexes diff from rdbms

References

[1]Lee, K. H., Lee, Y. J., Choi, H., Chung, Y. D., & Moon, B. (2012). Parallel data processing with MapReduce: a survey. *AcM sIGMoD Record*, *40*(4), 11-20. doi: 10.1145/2094114.2094118

[2]Pavlo, A., Paulson, E., Rasin, A., Abadi, D. J., DeWitt, D. J., Madden, S., & Stonebraker, M. (2009, June). A comparison of approaches to large-scale data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data* (pp. 165-178). ACM. (3) The performance of MapReduce: an in-depth study. doi: 10.1145/1559845.1559865

[3]Jiang, D., Ooi, B. C., Shi, L., & Wu, S. (2010). The performance of MapReduce: an in-depth study. *Proceedings of the VLDB Endowment*, *3*(1-2), 472-483. doi: 10.14778/1920841.1920903

[4]Eltabakh, M. Y., Özcan, F., Sismanis, Y., Haas, P. J., Pirahesh, H., & Vondrak, J. (2013, March). Eagle-eyed elephant: split-oriented indexing in hadoop. In*Proceedings of the 16th International Conference on Extending Database Technology* (pp. 89-100). ACM. doi: 10.1145/2452376.2452388

[5]Chen, G., Vo, H. T., Wu, S., Ooi, B. C., & Özsu, M. T. (2011). A Framework for Supporting DBMS-like Indexes in the Cloud. *Proceedings of the VLDB Endowment*, *4*(11), 702-713. doi: 10.14778/2556549.2556550

Thank you