Intro to Big Data Computing

ESCOLA D'ENGINYERIA. UAB

Remo Suppi
Remo.Suppi@uab.cat

Big Data Buzz

"Why big data is a big deal"

InfoWorld – 9/1/11

"The challenge– and opportunity– of big data"

McKinsey Quarterly—5/11

"Ten reasons why Big Data will change the travel industry"

Tnooz -8/15/11

"Keeping Afloat in a Sea of 'Big Data"

ITBusinessEdge – 9/6/11

"Getting a Handle on Big Data with Hadoop"

Businessweek-9/7/11

"The promise of Big Data"

Intelligent Utility-8/28/11

Big Data Context

Man on the moon with 32KB (1969); my laptop had 8GB RAM (2015)

Google collects 270PB data in a month (2007). Today processes over 40,000 search queries every second on average, which translates to over 3.5 billion searches per day (1.2 trillion searches per year worldwide). Google currently processes over 20,000 petabytes of data per day through an average of 100,000 MapReduce jobs spread across its massive computing clusters. The average MapReduce job ran across approximately, crunching approximately 11,000 machine years in a single month.

2010 census data is expected to be a huge gold mine of information

Data mining huge amounts of data collected in a wide range of domains from astronomy to healthcare has become essential for planning and performance.

We are in a knowledge economy. Data is an important asset to any organization

Discovery of knowledge; Enabling discovery; annotation of data

We are looking at newer programming models, and

Supporting algorithms and data structures.

NSF refers to it as “data-intensive computing” and industry calls it “big-data” and “cloud computing”

Big Data Context

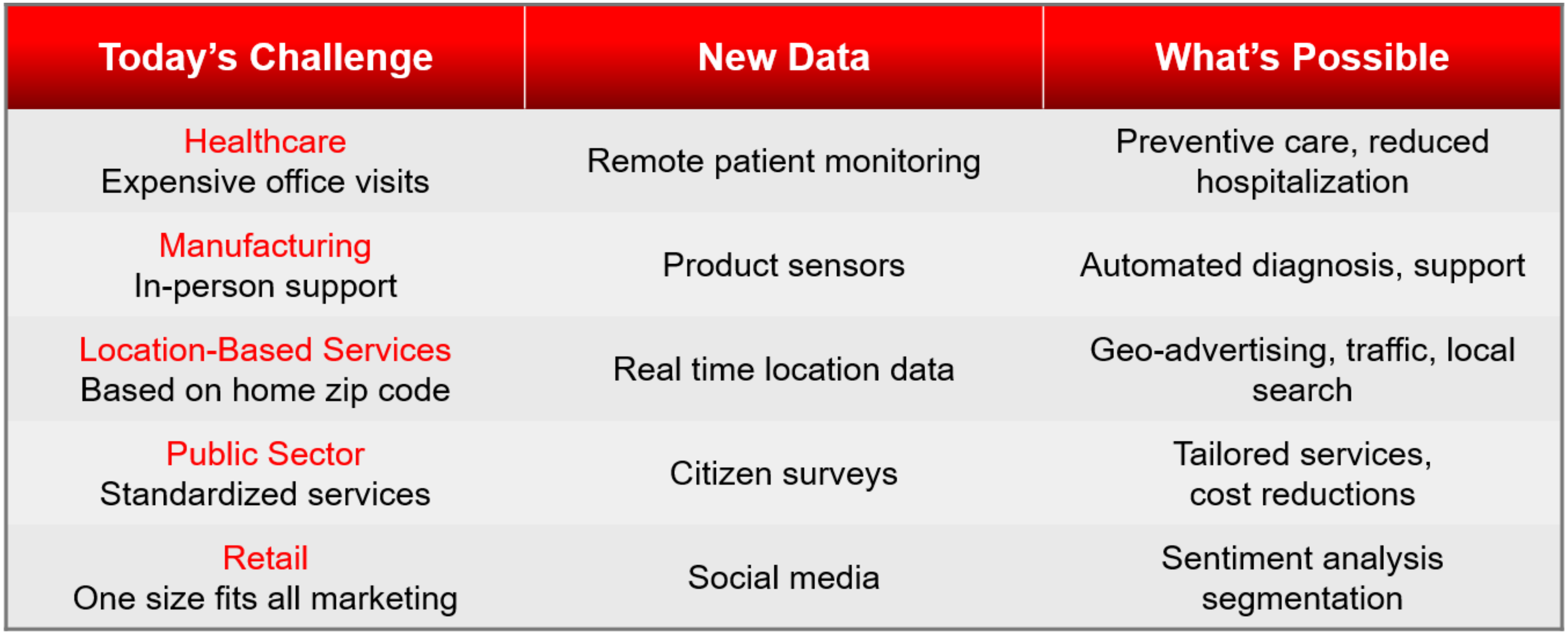
“The big-data computing” has been transformed into an essential advancement that has a potential to impact significantly the CS.

New tools and procedures have been necessary to facilitate their treatment:

- a programming model called MapReduce for processing “big-data”
- A supporting file system called Hadoop Distributed File System (HDFS)

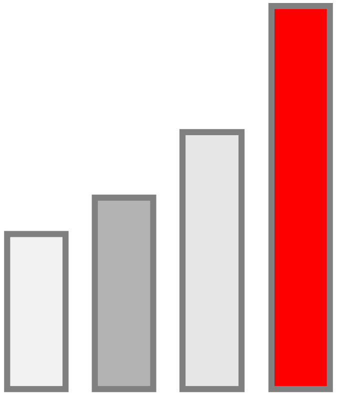
Nowadays it is necessary to have skills and abilities in Big Data environments.

Big Data Use Cases

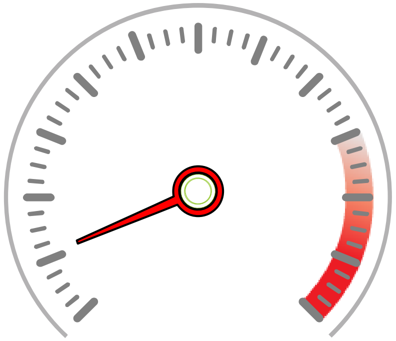


Today's Challenge	New Data	What's Possible
Healthcare Expensive office visits	Remote patient monitoring	Preventive care, reduced hospitalization
Manufacturing In-person support	Product sensors	Automated diagnosis, support
Location-Based Services Based on home zip code	Real time location data	Geo-advertising, traffic, local search
Public Sector Standardized services	Citizen surveys	Tailored services, cost reductions
Retail One size fits all marketing	Social media	Sentiment analysis segmentation

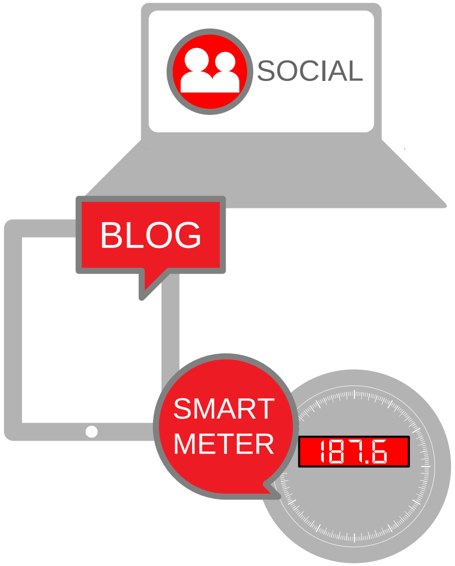
What Makes it Big Data?



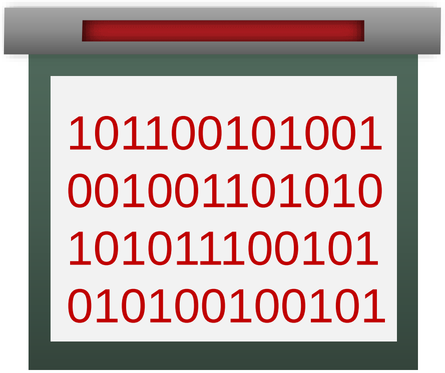
VOLUME



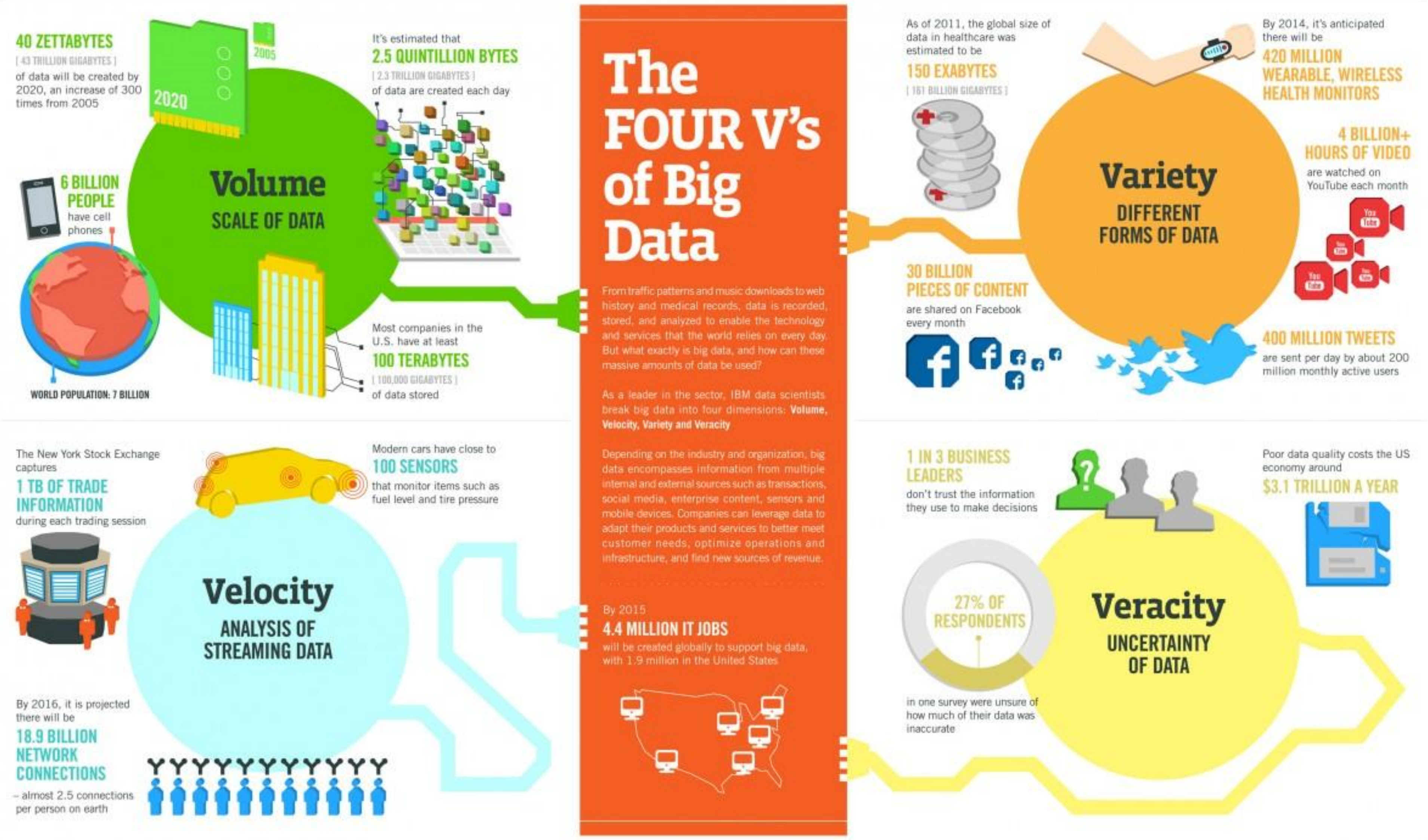
VELOCITY



VARIETY



VERACITY



The FOUR V's of Big Data

From traffic patterns and music downloads to web history and medical records, data is recorded, stored, and analyzed to enable the technology and services that the world relies on every day. But what exactly is big data, and how can these massive amounts of data be used?

As a leader in the sector, IBM data scientists break big data into four dimensions: **Volume, Velocity, Variety and Veracity**

Depending on the industry and organization, big data encompasses information from multiple internal and external sources such as transactions, social media, enterprise content, sensors and mobile devices. Companies can leverage data to adapt their products and services to better meet customer needs, optimize operations and infrastructure, and find new sources of revenue.

By 2015

4.4 MILLION IT JOBS

will be created globally to support big data, with 1.9 million in the United States

Volume

SCALE OF DATA

40 ZETTABYTES

[43 TRILLION GIGABYTES]

of data will be created by 2020, an increase of 300 times from 2005

6 BILLION PEOPLE

have cell phones

WORLD POPULATION: 7 BILLION

It's estimated that

2.5 QUINTILLION BYTES

[2.3 TRILLION GIGABYTES]

of data are created each day

Most companies in the U.S. have at least

100 TERABYTES

[100,000 GIGABYTES]

of data stored

Velocity

ANALYSIS OF STREAMING DATA

The New York Stock Exchange captures

1 TB OF TRADE INFORMATION

during each trading session

Modern cars have close to

100 SENSORS

that monitor items such as fuel level and tire pressure

By 2016, it is projected there will be

18.9 BILLION NETWORK CONNECTIONS

– almost 2.5 connections per person on earth

Variety

DIFFERENT FORMS OF DATA

As of 2011, the global size of data in healthcare was estimated to be

150 EXABYTES

[161 BILLION GIGABYTES]

30 BILLION PIECES OF CONTENT

are shared on Facebook every month

By 2014, it's anticipated there will be

420 MILLION WEARABLE, WIRELESS HEALTH MONITORS

4 BILLION+ HOURS OF VIDEO

are watched on YouTube each month

400 MILLION TWEETS

are sent per day by about 200 million monthly active users

Veracity

UNCERTAINTY OF DATA

1 IN 3 BUSINESS LEADERS

don't trust the information they use to make decisions

27% OF RESPONDENTS

in one survey were unsure of how much of their data was inaccurate

Poor data quality costs the US economy around

$3.1 TRILLION A YEAR

Sources: McKinsey Global Institute, Twitter, Cisco, Gartner, EMC, SAS, IBM, MEPTEC, QAS

IBM.

So What is Big Data?

Big Data refers to datasets that grow so large that it is difficult to capture, store, manage, share, analyse and visualize with the typical database software tools.

How much is Big?
It is not a single number but a set of parameters

Where Do We See Big Data?

Datawarehouses

OLTP
(OnLine Transaction Processing)

Social Networks

Scientific Devices

Useful?

Big Data:

Decisions based on all your data

Video and Images

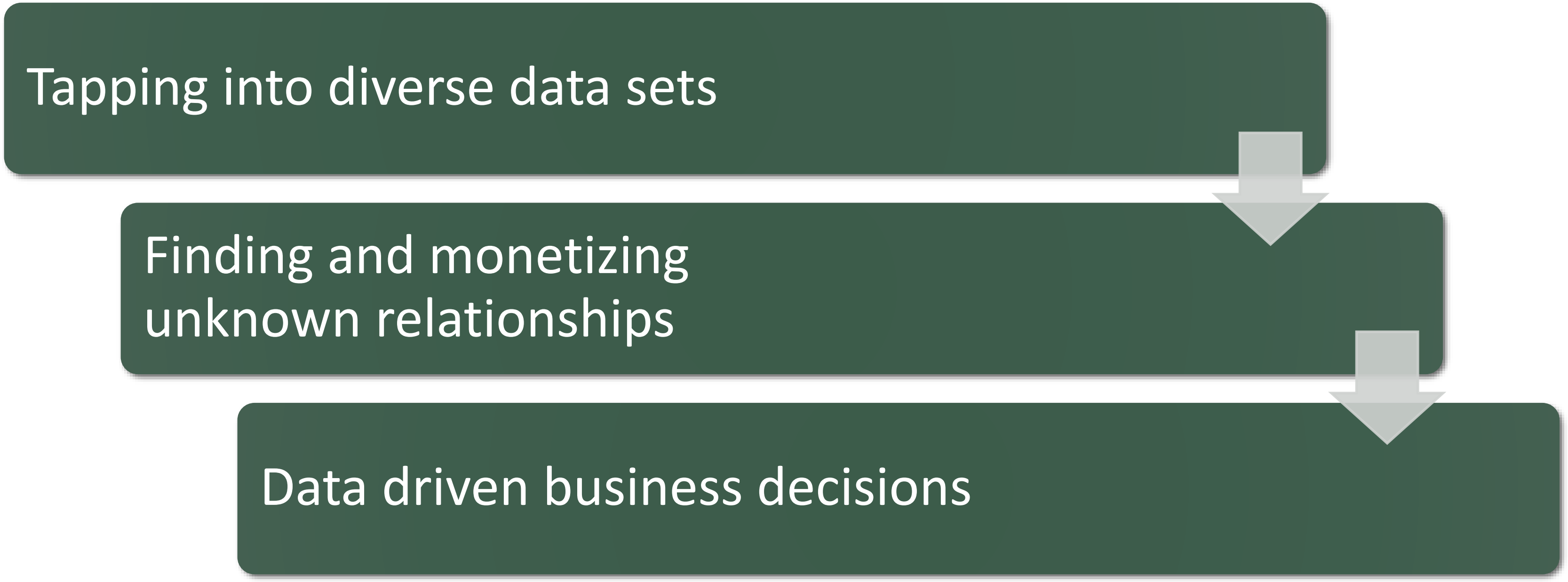
Documents

Social Data

Machine-Generated Data

US HEALTH CARE	MANUFACTURING	GLOBAL PERSONAL LOCATION DATA	EUROPE PUBLIC SECTOR ADMIN	US RETAIL
Increase industry value per year by	Decrease dev., assembly costs by	Increase service provider revenue by	Increase industry value per year by	Increase net margin by
$300 B	**–50%**	**$100 B**	**€250 B**	**60+%**

Big Data Is About…

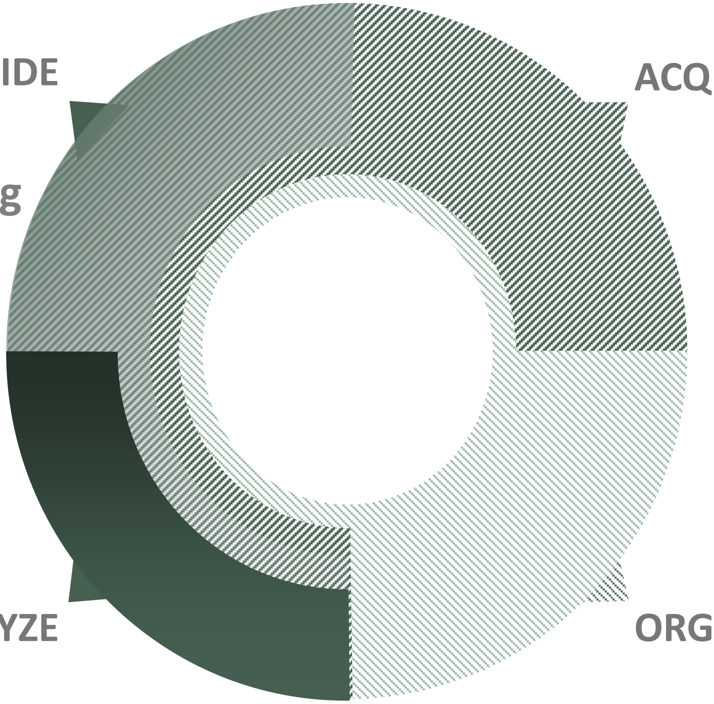


Tapping into diverse data sets

Finding and monetizing unknown relationships

Data driven business decisions

Big Data in Action



DECIDE

Decide based on real-time big data

ACQUIRE

Acquire all available data

ORGANIZE

Organize and **distill** big data using massive parallelism

ANALYZE

Analyze all your data, at once

Make Better Decisions Using Big Data

Big Data Today

Opportunity

- Answer formerly unanswerable questions
- Formulate new questions and become much more agile
- Make evidence based decisions
- Democratize your data
- Visualize invisible knowledge

Threat

- Big data is here – now
- Data breaches
- Intrusion of privacy
- Unfair use of Data

Where?

- Operational efficiency and productivity
- Fraud detection and prevention
- Close tax gaps
- Value for money for citizens
- Prevent crime waves
- Customize actions based on population segments
- Public utilities to reduce consumption
- Produce safety from farm to fork

MapReduce

MapReduce is a programming model Google has used successfully is processing its "big-data" sets (~ 20,000 Pbytes per day)

Users specify the computation in terms of a map and a reduce function,

Underlying runtime system automatically parallelizes the computation across large-scale clusters of machines, and

Underlying system also handles machine failures, efficient communications, and performance issues.

Consider a large data collection:
{web, weed, green, sun, moon, land, part, web, green,…}
Problem: Count the occurrences of the different words in the collection.

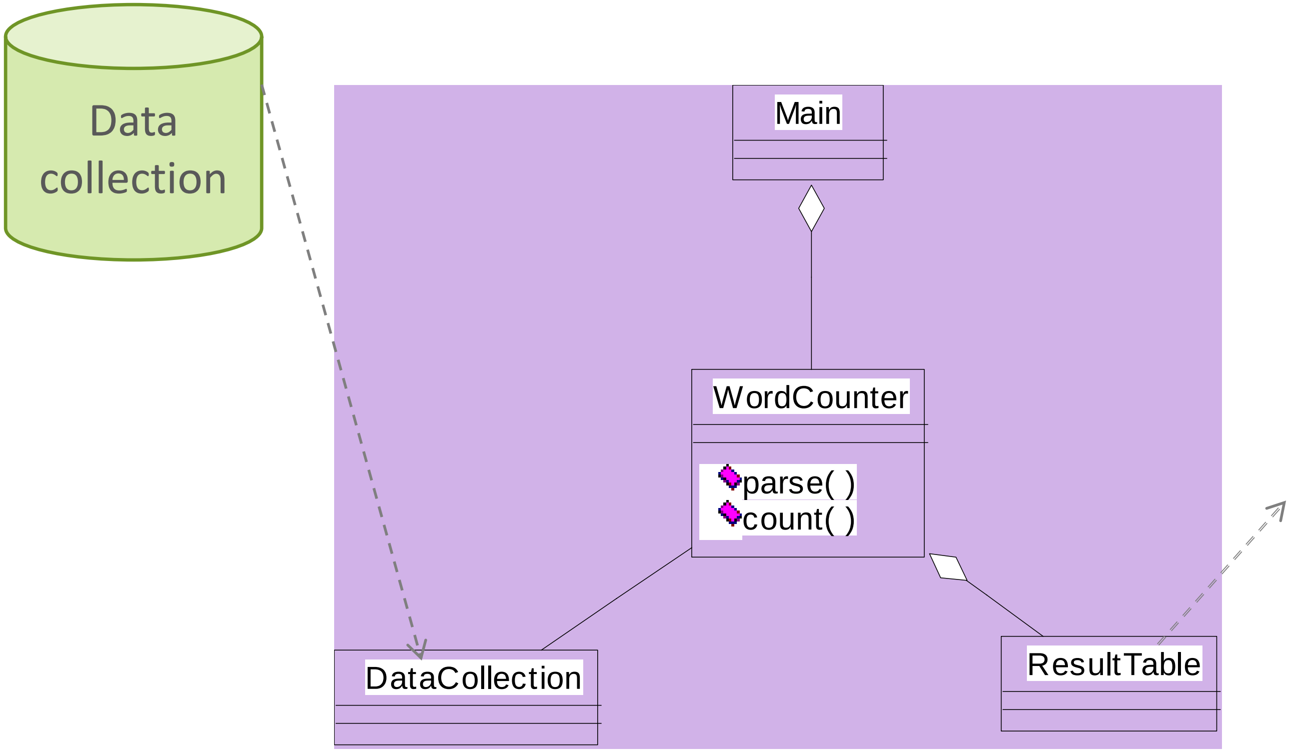
Lets design a solution for this problem;

- We will start from scratch
- We will add and relax constraints
- We will do incremental design, improving the solution for performance and scalability

Reference: Dean, J. and Ghemawat, S. 2008. MapReduce: simplified data processing on large clusters. Communication of ACM 51, 1 (Jan. 2008), 107-113.

Word Counter and Result Table

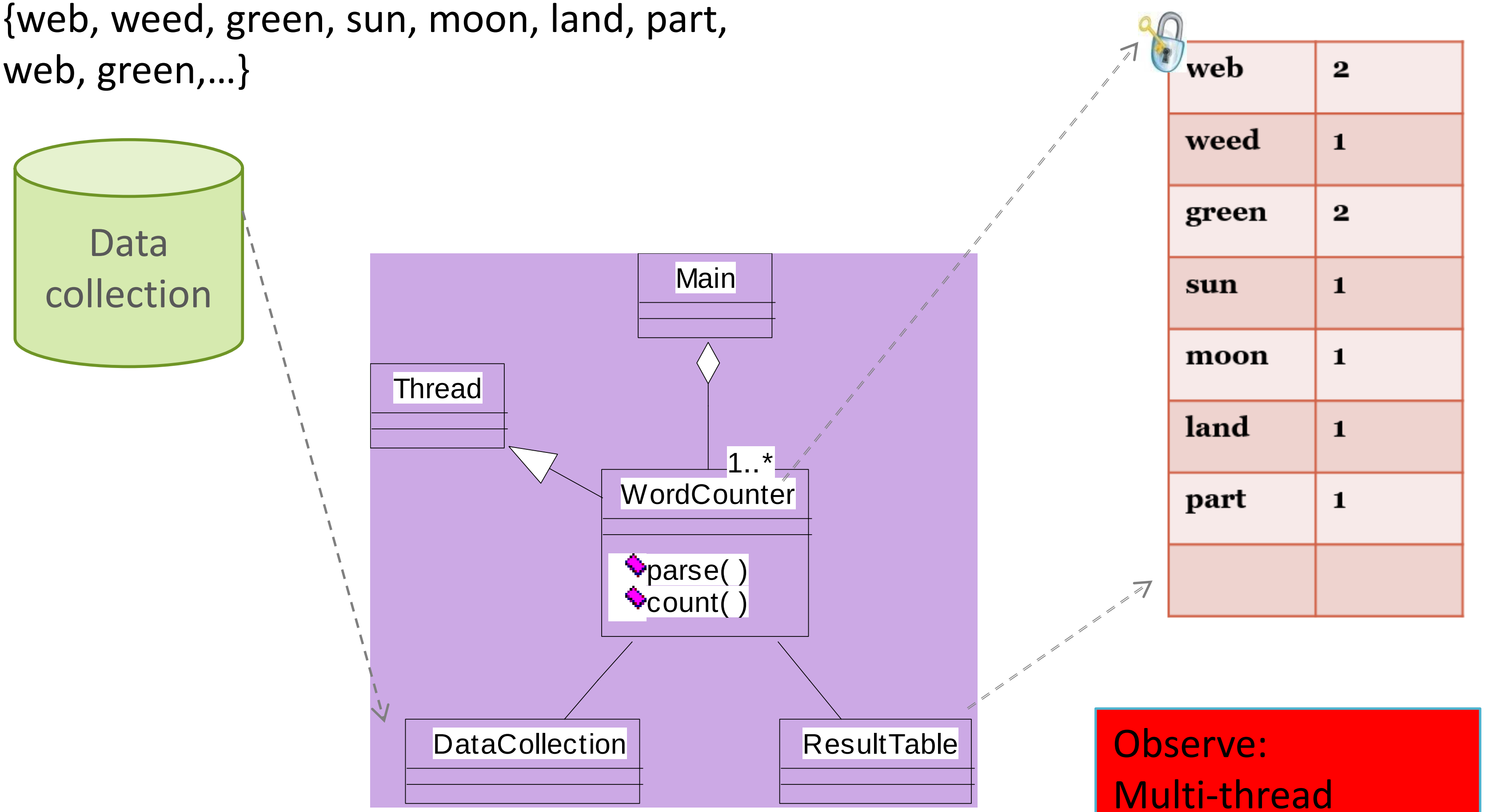
{web, weed, green, sun, moon, land, part, web, green,…}



web	2
weed	1
green	2
sun	1
moon	1
land	1
part	1

Performance? Multiple Instances Word Counter

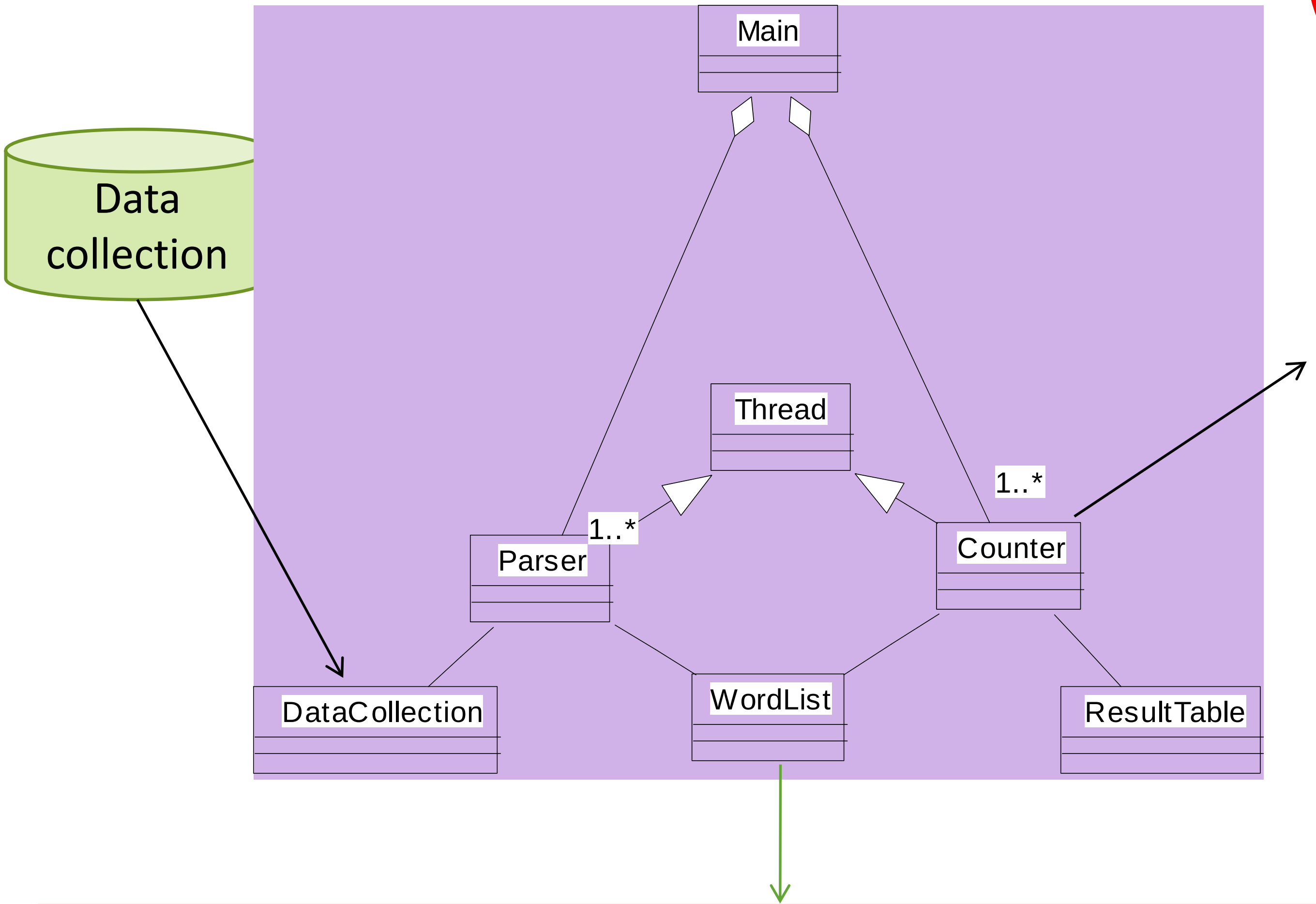
{web, weed, green, sun, moon, land, part, web, green,…}



web	2
weed	1
green	2
sun	1
moon	1
land	1
part	1

Observe:
Multi-thread
Lock on shared data

Improve Word Counter for Performance



N

o No need for lock

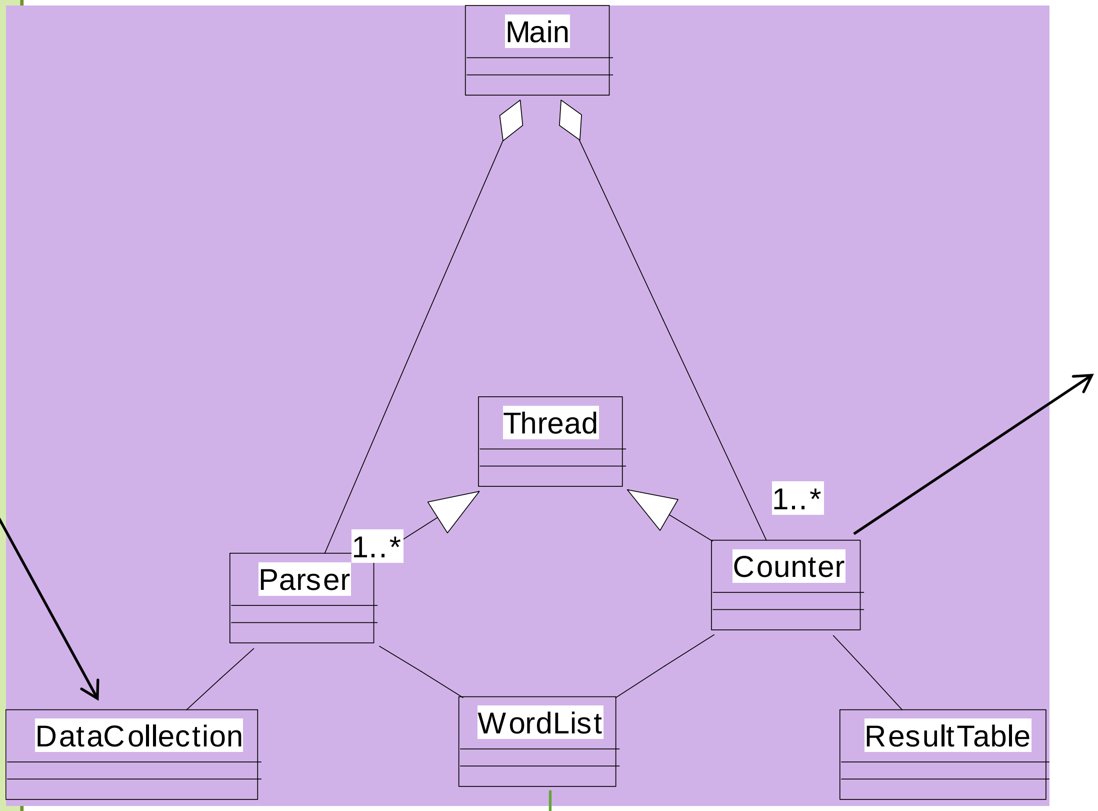
web	2
weed	1
green	2
sun	1
moon	1
land	1
part	1

Separate counters

KEY	web	weed	green	sun	moon	land	part	web	green	…….
VALUE										

Peta-scale Data?

Data collection



N
o
No need for lock

web	2
weed	1
green	2
sun	1
moon	1
land	1
part	1

KEY	web	weed	green	sun	moon	land	part	web	green	…….
VALUE										

Addressing the Scale Issue

Single machine cannot serve all the data: a distributed special (file) system is necessary.

Large number of commodity hardware disks: say, 1000 disks 1TB each

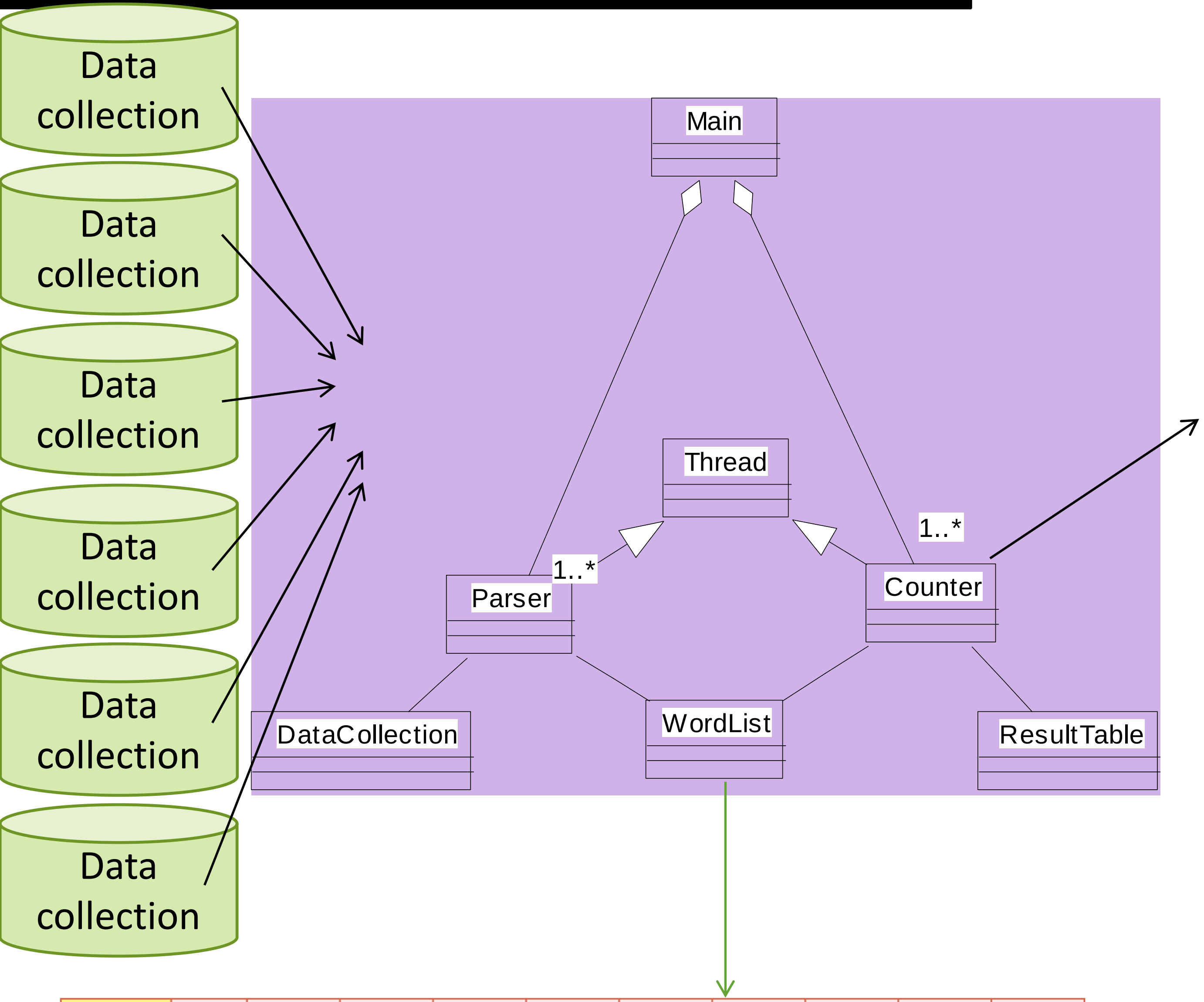
> Issue: With Mean time between failures (MTBF) or failure rate of 1/1000, then at least 1 of the above 1000 disks would be down at a given time.
>
> Thus failure is norm and not an exception.
>
> File system has to be fault-tolerant: replication, checksum
>
> Data transfer bandwidth is critical (location of data)

Critical aspects: fault tolerance + replication + load balancing, monitoring

Exploit parallelism afforded by splitting parsing and counting

Provision and locate computing at data locations

Peta Scale Data is Commonly Distributed



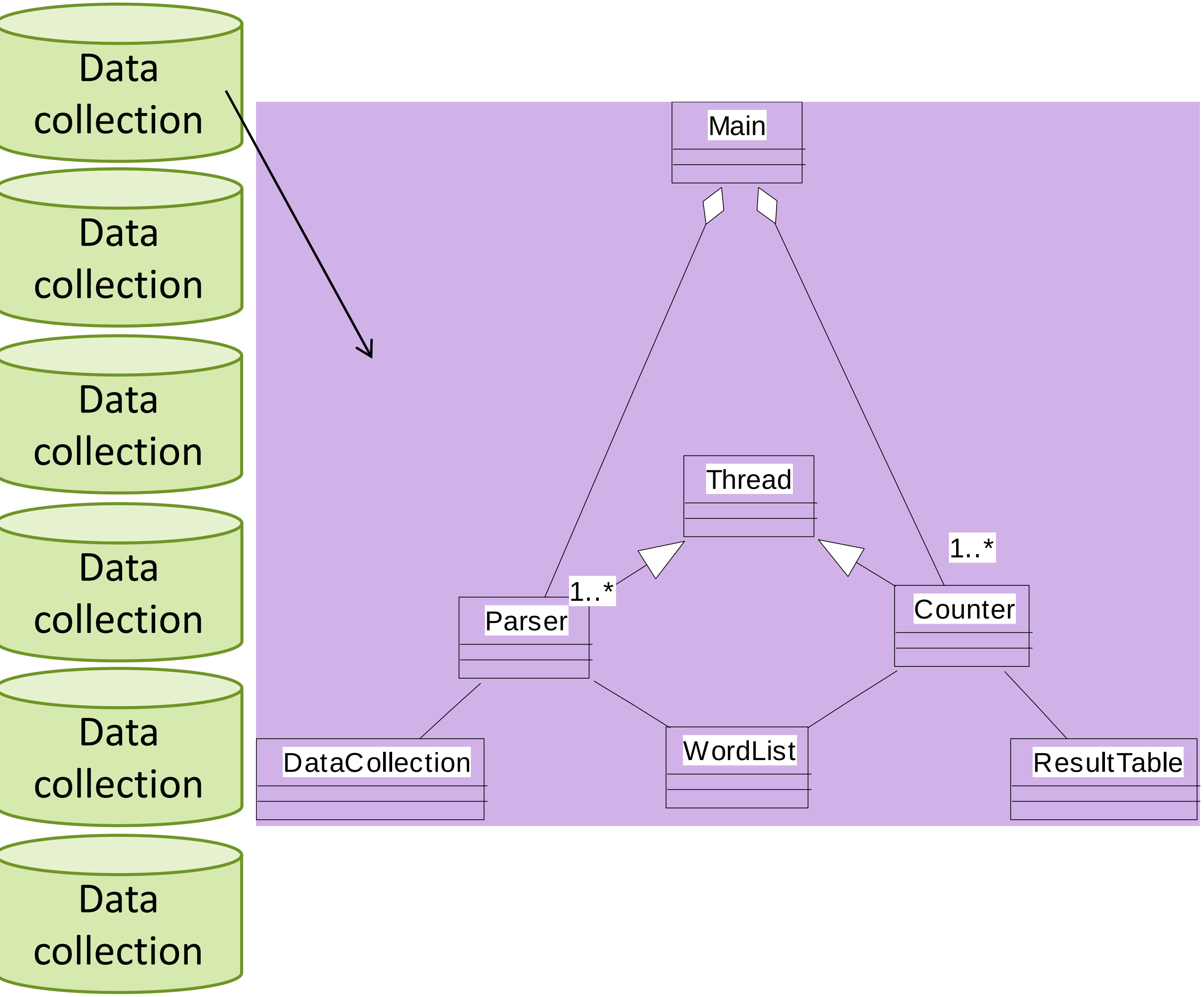
web	2
weed	1
green	2
sun	1
moon	1
land	1
part	1

Issue: managing the large scale data

Write Once Read Many (WORM) data

KEY	web	weed	green	sun	moon	land	part	web	green	…….
VALUE										

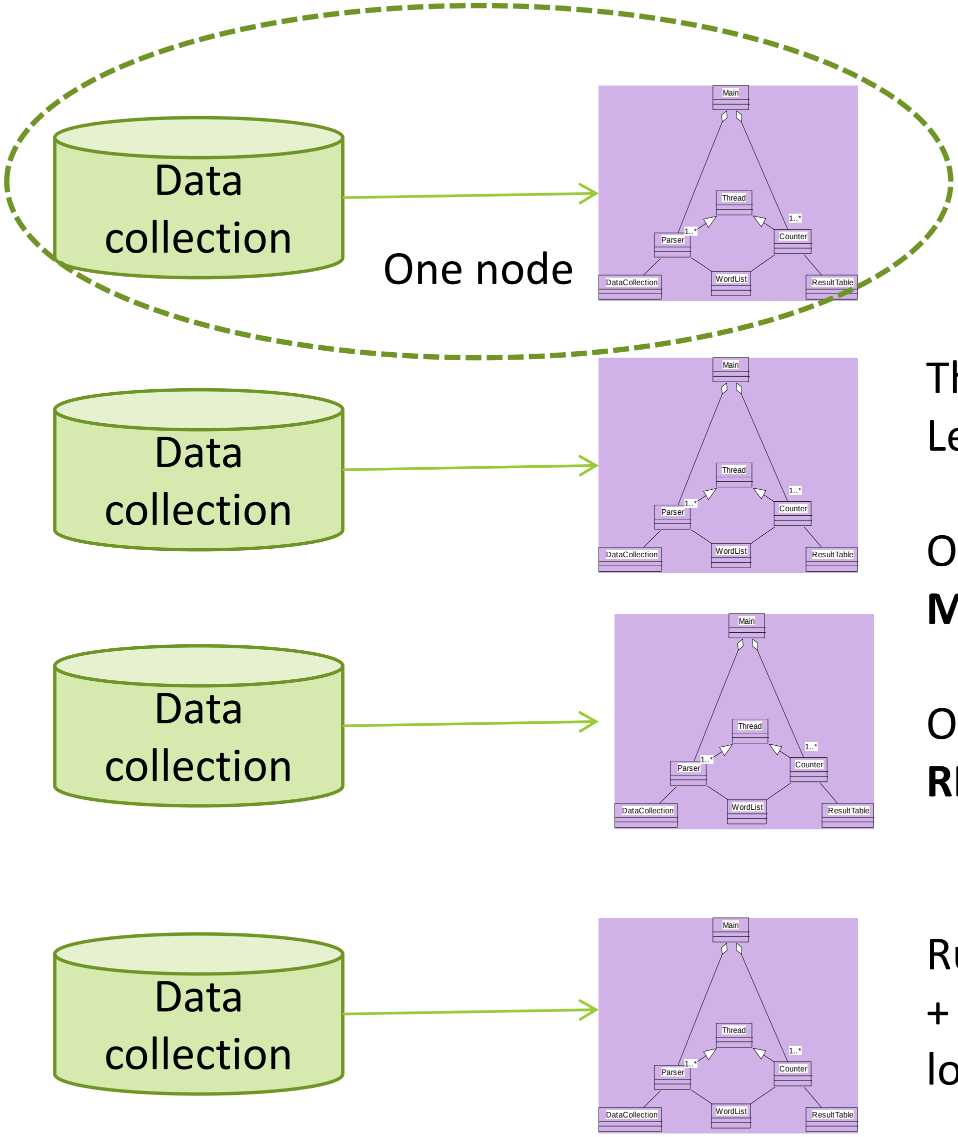
Worm Data is Amenable to Parallelism



1. Data with WORM characteristics: yields to parallel processing;

2. Data without dependencies: yields to out of order processing

Divide and Conquer: Provision Computing at Data Location



For our example,
#1: Schedule parallel parse tasks
#2: Schedule parallel count tasks

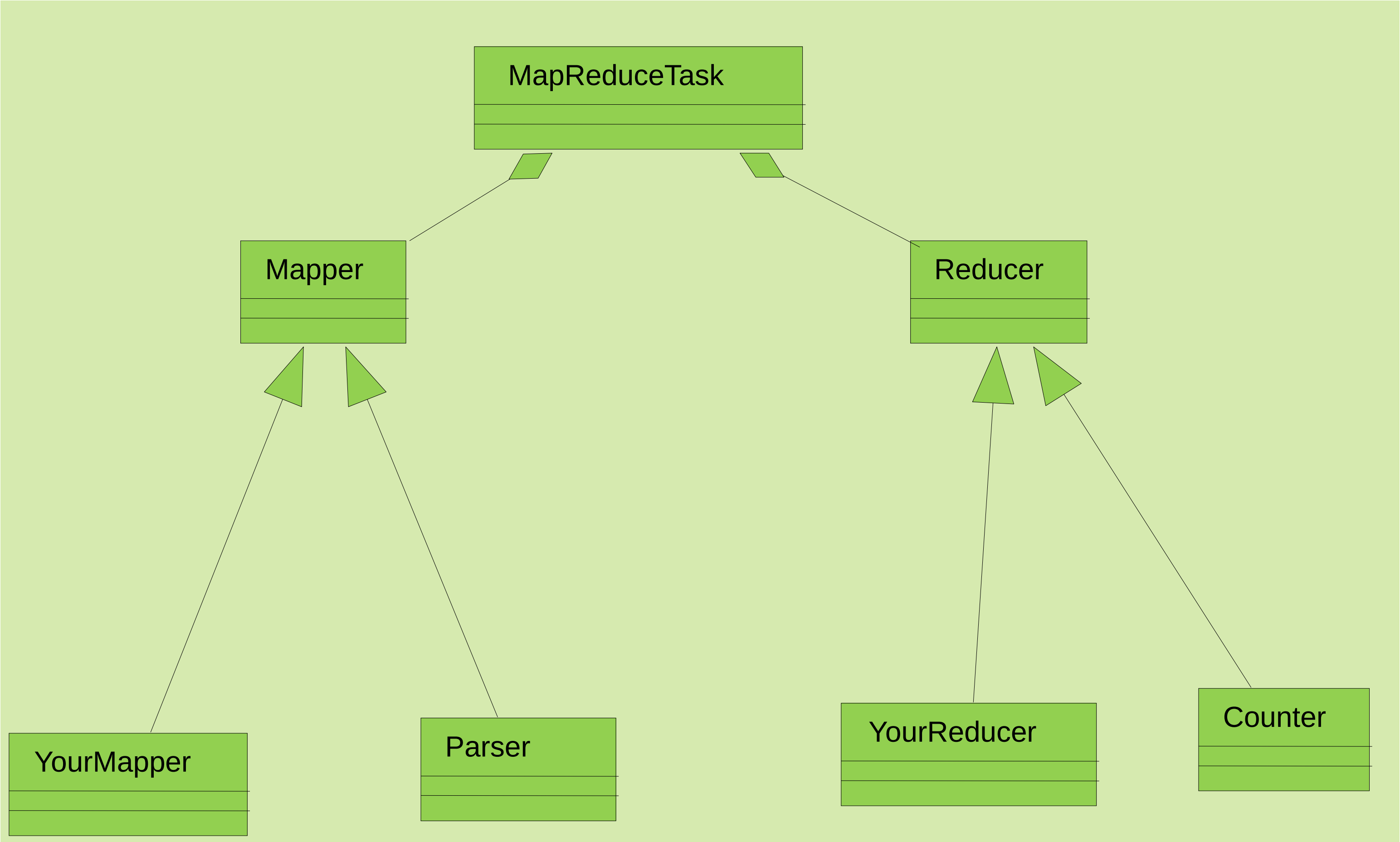
This is a particular solution;
Lets generalize it:

Our parse is a mapping operation:
MAP: input → <key, value> pairs

Our count is a reduce operation:
REDUCE: <key, value> pairs reduced

Runtime adds distribution + fault tolerance + replication + monitoring +
load balancing to your base application!

Mapper and Reducer



MapReduce is simplified processing for larger data sets: MapReduce Version of WordCount Source code

Mapper and Reducer

```
class Mapper
    method Map(docid a; doc d)
        for all term t in doc d do
            emit(term t; count 1)

class Reducer
    method Reduce(term t; counts [c1; c2; : : :])
        sum =  0
        for all count c in counts [c1; c2; : : :] do
            sum = sum + c
        emit(term t; count sum)
```

This is a cat
Cat sits on a roof

<this 1> <is 1> <a <1,1,>> <cat <1,1>> <sits 1> <on 1> <roof 1>

Cat kicks the can
It rolls on the roof and falls on the next roof

<cat 1> <kicks 1> <the <1,1>> <can 1> <it 1> <roll 1> <on <1,1>> <roof <1,1>> <and 1> <falls 1> <next 1>

Combine the counts of all the same words:

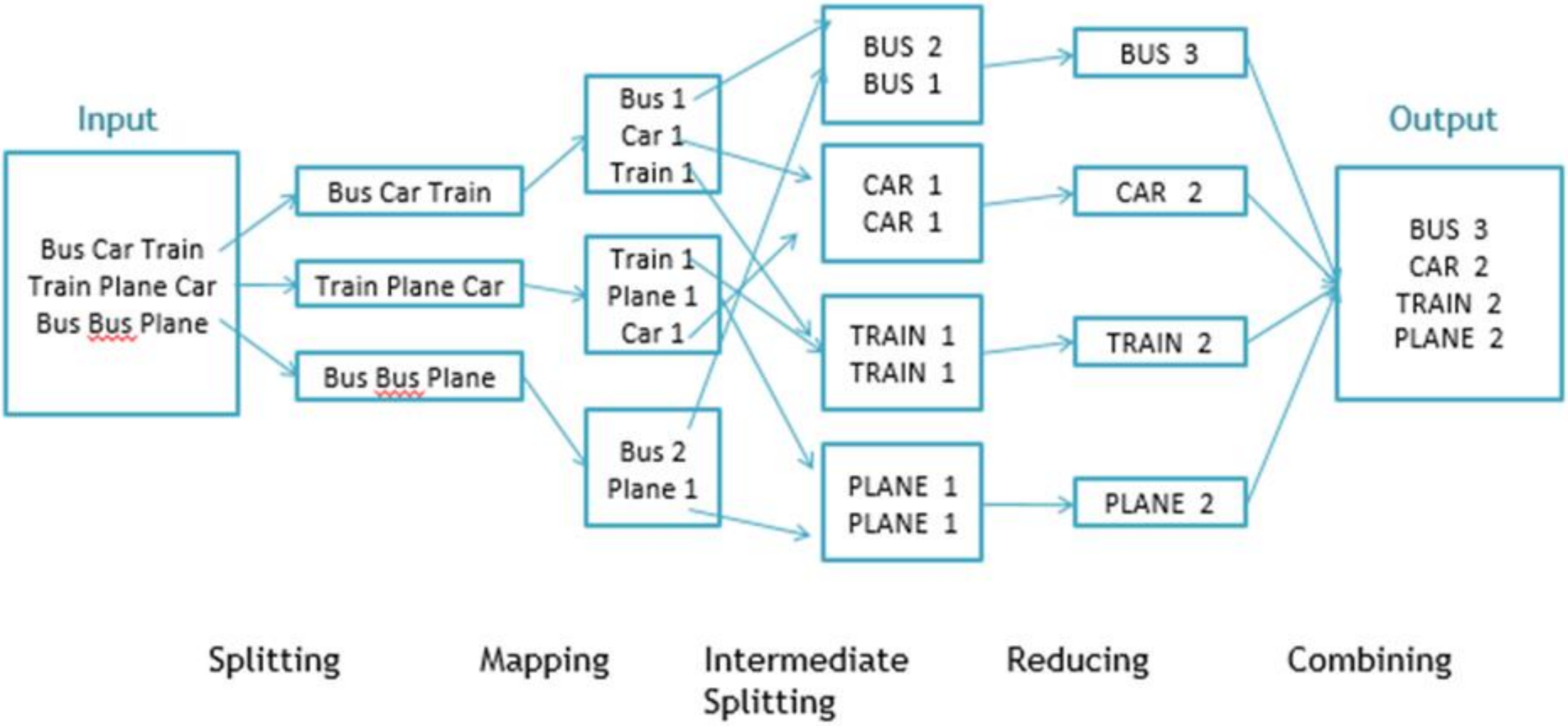
<cat <1,1,1>>
<roof <1,1>>
<can <1>>

Reduce (sum in this case) the counts:

<cat 3>
<can 1>
<roof 2>

Workflow

MAP: Input data → <key, value> pair



REDUCE: <key, value> pair → <result>

Code WordCount

```
import java.io.IOException;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount {

 public static class TokenizerMapper extends Mapper<Object, Text, Text, IntWritable> {

  private final static IntWritable one = new IntWritable(1);
  private Text word = new Text();

  public void map (Object key, Text value, Context context ) throws IOException, InterruptedException {
   StringTokenizer itr = new StringTokenizer(value.toString());
   while (itr.hasMoreTokens()) {
    word.set(itr.nextToken());
    context.write(word, one);
    }
  }  // end map
 }  // end TokenizerMapper
```

Code WordCount

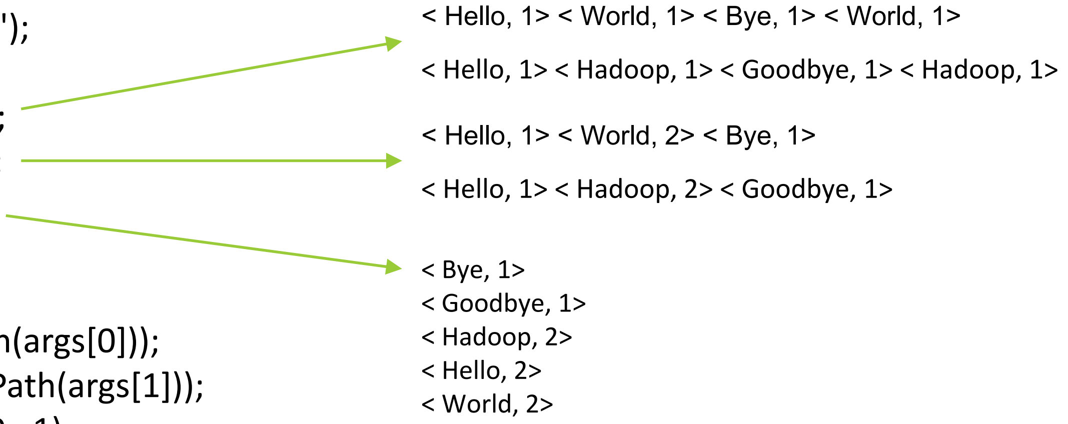
```
  public static class IntSumReducer extends Reducer<Text,IntWritable,Text,IntWritable> {

    private IntWritable result = new IntWritable();

    public void reduce(Text key, Iterable<IntWritable> values, Context context ) throws IOException, InterruptedException {
      int sum = 0;
      for (IntWritable val : values) {
        sum += val.get();
      }
      result.set(sum);
      context.write(key, result);
    }
  }

  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

File01: Hello World Bye World
File02: Hello Hadoop Goodbye Hadoop



< Hello, 1> < World, 1> < Bye, 1> < World, 1>

< Hello, 1> < Hadoop, 1> < Goodbye, 1> < Hadoop, 1>

< Hello, 1> < World, 2> < Bye, 1>

< Hello, 1> < Hadoop, 2> < Goodbye, 1>

< Bye, 1>
< Goodbye, 1>
< Hadoop, 2>
< Hello, 2>
< World, 2>

MR: Conclusions

Very large scale data: peta, exa bytes

Write once and read many data: allows for parallelism without mutexes

Map and Reduce are the main operations: simple code

There are other supporting operations such as combine and partition.

All the map should be completed before reduce operation starts. All Mappers work in parallel. Barriers enforce all mappers completion before Reducers start.

Map and reduce operations are typically performed by the same physical processor.

Number of map tasks and reduce tasks are configurable.

Operations are provisioned near the data.

Commodity hardware and storage.

Runtime takes care of splitting and moving data for operations.

Special distributed file system: Hadoop Distributed File System and Hadoop Runtime.

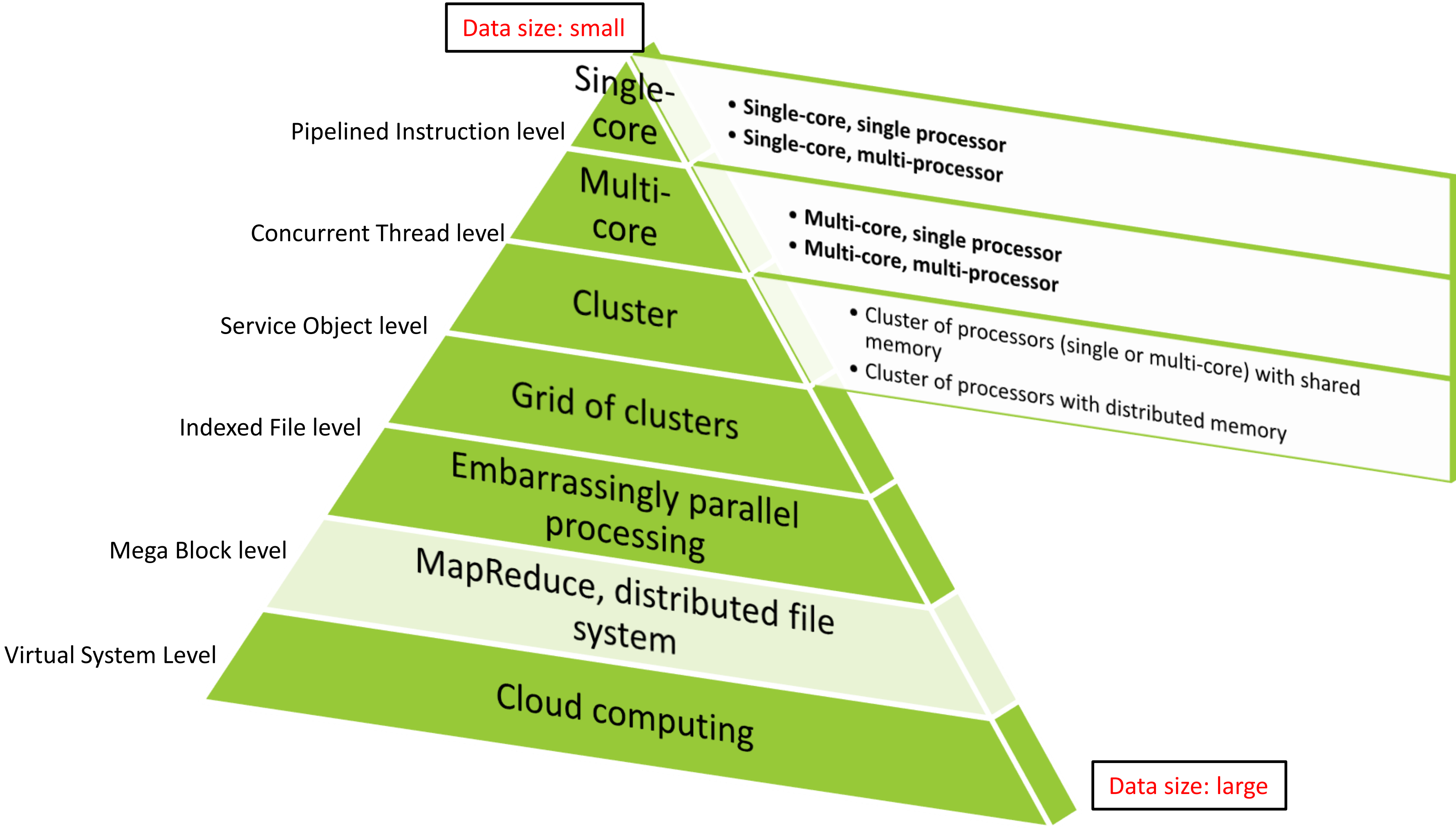
MR: Classes of problems “mapreducable”

Benchmark for comparing: Jim Gray’s challenge on data-intensive computing. Ex: “Sort”

Google uses it (we think) for wordcount, adwords, pagerank, indexing data. Simple algorithms such as grep, text-indexing, reverse indexing

Bayesian classification: data mining domain, Facebook uses it for various operations: demographics

Financial services use it for analytics, Astronomy: Gaussian analysis for locating extra-terrestrial objects, Expected to play a critical role in semantic web and web3.0



Data size: small

Pipelined Instruction level

Single-core

- Single-core, single processor
- Single-core, multi-processor

Concurrent Thread level

Multi-core

- Multi-core, single processor
- Multi-core, multi-processor

Service Object level

Cluster

- Cluster of processors (single or multi-core) with shared memory
- Cluster of processors with distributed memory

Indexed File level

Grid of clusters

Embarrassingly parallel processing

Mega Block level

MapReduce, distributed file system

Virtual System Level

Cloud computing

Data size: large

What is Hadoop

At Google MapReduce operation are run on a special file system called Google File System (GFS) that is highly optimized for this purpose. **GFS is not open source.**

Doug Cutting and Yahoo! reverse engineered the GFS and called it Hadoop Distributed File System (HDFS).

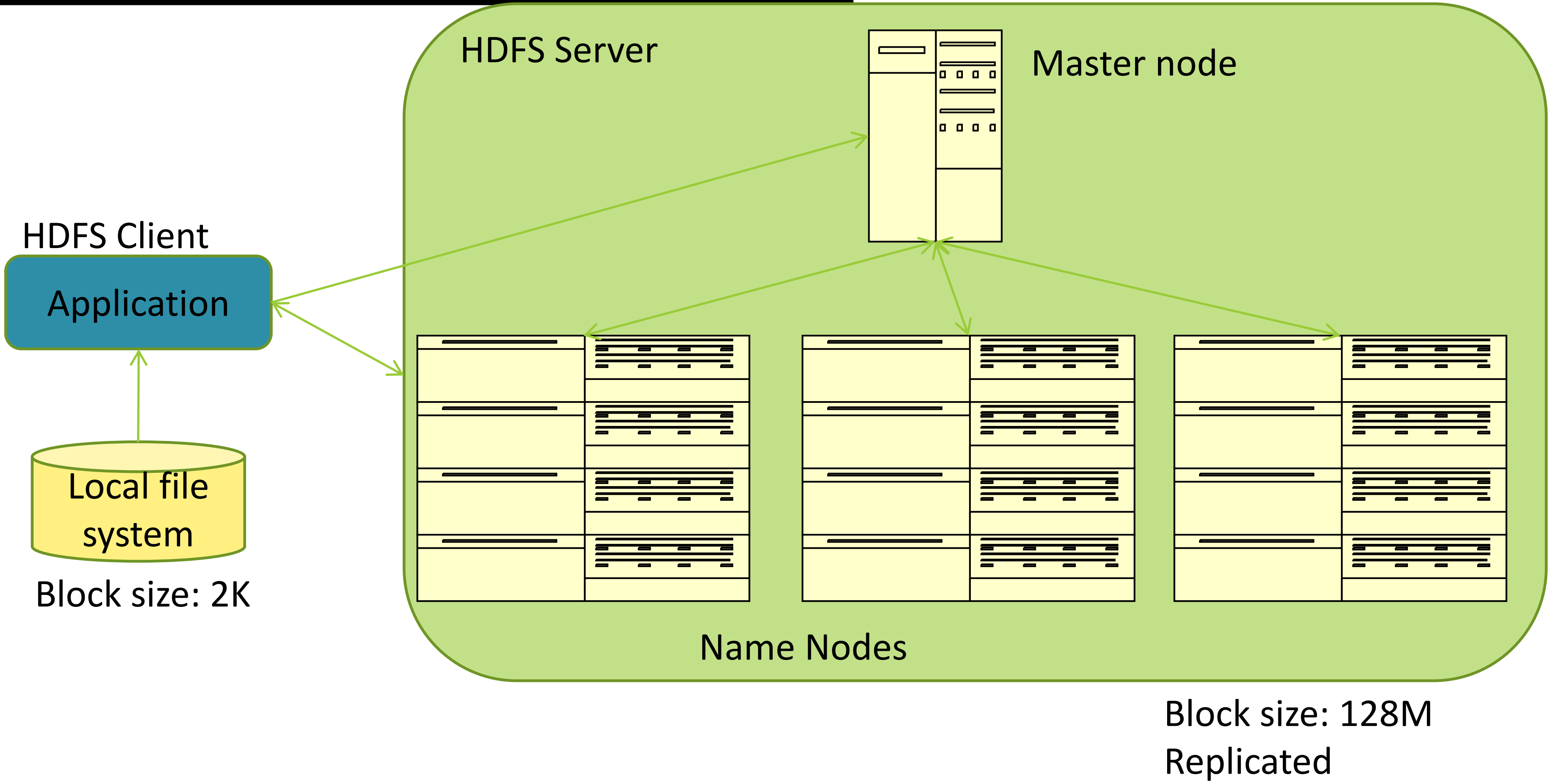
The software framework that supports HDFS, MapReduce and other related entities is called the project Hadoop or simply Hadoop.

This is open source and distributed by Apache.

HDFS:

Highly fault-tolerant
High throughput
Suitable for applications with large data sets
Streaming access to file system data
Can be built out of commodity hardware

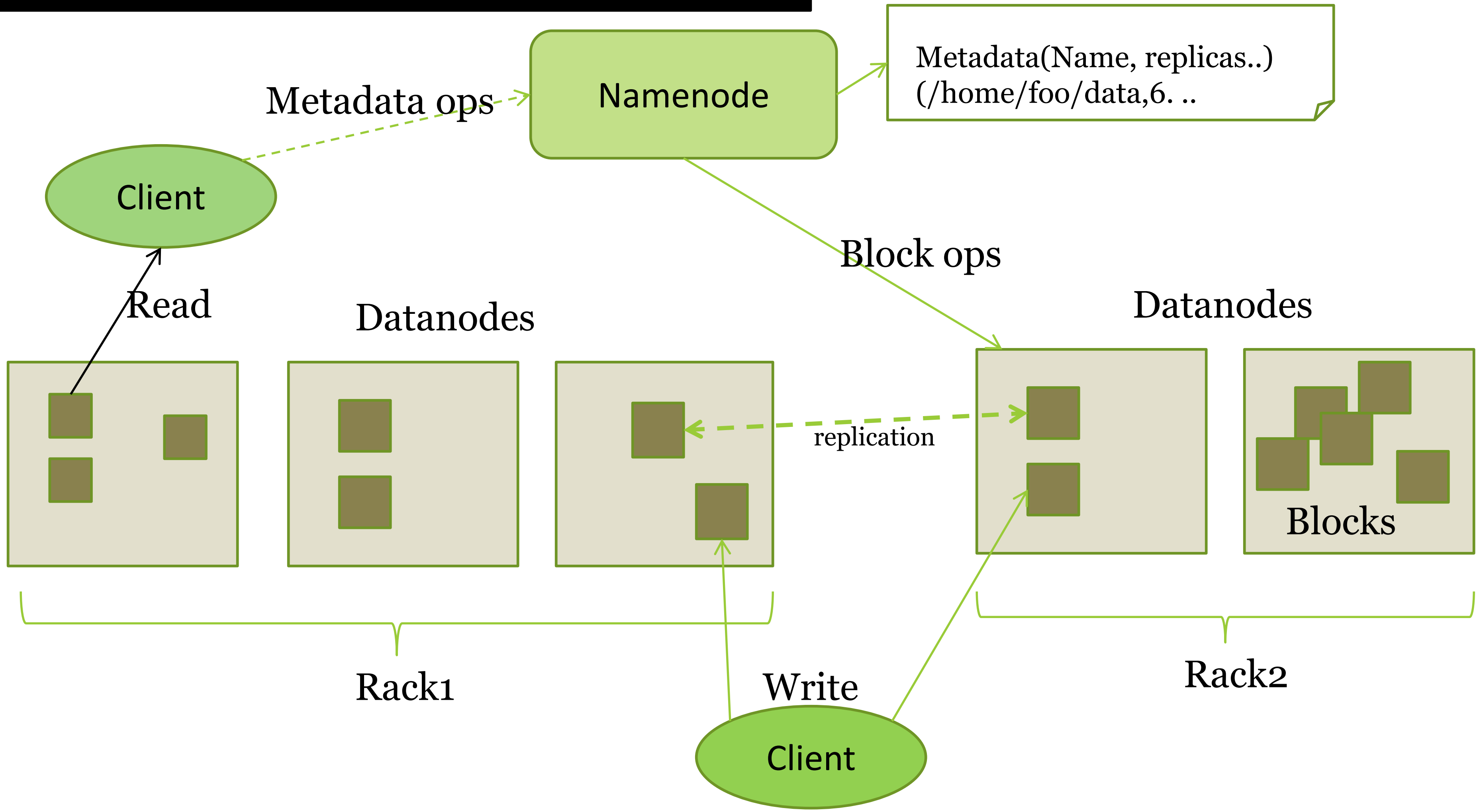
HDFS: Hadoop Distributed File System



Divide user data into blocks and replicate those blocks across the local disks of nodes in the cluster

A master/slave architecture in which the master maintains the file namespace (metadata, directory structure, file to block mapping, location of blocks, and access permissions) and the slaves manage the actual data blocks : **namenode and datanodes**

HDFS: Hadoop Distributed File System



- HDFS cluster consists of a single **Namenode**, a master server that manages the file system namespace and regulates access to files by clients.
- There are a number of **DataNodes** usually one per node in a cluster.
- The DataNodes manage storage attached to the nodes that they run on.
- HDFS exposes a file system namespace and allows user data to be stored in files.
- A file is split into one or more blocks and set of blocks are stored in DataNodes.
- DataNodes: serves read, write requests, performs block creation, deletion, and replication upon instruction from Namenode.

Hadoop Project

Hadoop Common: The common utilities that support the other Hadoop modules.

Hadoop Distributed File System (HDFS™): A distributed file system that provides high-throughput access to application data.

Hadoop YARN: A framework for job scheduling and cluster resource management.

Hadoop MapReduce: A YARN-based system for parallel processing of large data sets.

Other Hadoop-related projects at Apache include:

Ambari™: A web-based tool for provisioning, managing, and monitoring Apache Hadoop clusters which includes support for Hadoop HDFS, Hadoop MapReduce, Hive, HCatalog, HBase, ZooKeeper, Oozie, Pig and Sqoop. Ambari also provides a dashboard for viewing cluster health such as heatmaps and ability to view MapReduce, Pig and Hive applications visually with features to diagnose their performance characteristics in a user-friendly manner.

Avro™: A data serialization system.

Cassandra™: A scalable multi-master database with no single points of failure.

Chukwa™: A data collection system for managing large distributed systems.

HBase™: A scalable, distributed database that supports structured data storage for large tables.

Hadoop Project

Hive™: A data warehouse infrastructure that provides data summarization and ad hoc querying.

Mahout™: A Scalable machine learning and data mining library.

Pig™: A high-level data-flow language and execution framework for parallel computation.

Spark™: A fast and general compute engine for Hadoop data. Spark provides a simple and expressive programming model that supports a wide range of applications, including ETL, machine learning, stream processing, and graph computation.

Tez™: A generalized data-flow programming framework, built on Hadoop YARN, which provides a powerful and flexible engine to execute an arbitrary DAG of tasks to process data for both batch and interactive use-cases. Tez is being adopted by Hive™, Pig™ and other frameworks in the Hadoop ecosystem, and also by other commercial software (e.g. ETL tools), to replace Hadoop™ MapReduce as the underlying execution engine.

ZooKeeper™: A high-performance coordination service for distributed applications.

- Google GFS; Open Source equivalent is Hadoop Distributed File System
- Google’s BigTable: a sparse, distributed, persistent multidimensional sorted map; Hbase is the open-source equivalent

References:

1. Advanced Data Management. Jiaheng Lu. Department of Computer Science. Renmin University of China. www.jiahenglu.net
2. Cloud Computing for Networked Libraries and Information Centers. V. Caintic. Learning and Information Center. University of Mindanao. 2010. virginiacaintic@yahoo.com
3. The Challenges in ICT: Debunking the Hype. K. Jeffery. Science and Technology Facilities Council. Harwell Oxford. Rutherford Appleton Laboratory. UK. keith.jeffery@stfc.ac.uk
4. Big Data Analytics. Lecture Series . Kalapriya Kannan. IBM Research Labs. July, 2013
5. Infrastructure and Implementation Topics. T. Sridhar. The Internet Protocol Journal, Volume 12, No.4. http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_12-4/124_cloud2.html
6. Cloud Computing: Concepts, Technologies and Business Implications. B. Ramamurthy & K. Madurai bina@buffalo.edu & kumar.madurai@ctg.com
7. A Mainframe Guy Is Still Thinking About Cloud Computing. Glenn Anderson, © IBM Training. 2011.
8. Business in the cloud. M. Hugos, D. Hulitzky. Wiley. ISBN 978 0 470 61623 9
9. Distributed Computing. Chapter 13. Sunita Mahajan, Seema Shah. Oxford press.
10. Cloud Application Architectures. G. Reese. O'Reilly. ISBN: 978-0-596-15636-7
11. Cloud Computing Bible. B. Sosinsky. Wiley. ISBN: 978-0-470-90356-8
12. Cloud Security and Privacy. T. Mather, S. Kumaraswamy, and S. Latif. O'Reilly. ISBN: 978-0-596-80276-9
13. Cloud Computing. Theory and Practice. D. Marinescu. Elselvier. ISBN: 978-0-12404-627-6
14. Above the Clouds: A Berkeley View of Cloud Computing. M. Armbrust et al. University of California at Berkeley. http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf
15. An Introduction to Cloud Computing with OpenNebula. Daniel Molina Aranda. dmolina@opennebula.org. http://www.slideshare.net/opennebula/tecni-28445050
16. OpenNebula 4.2: 3-hour Hands-on Tutorial. http://opennebula.org/documentation/tutorials/.