(Big) Data Processing

Marek J. Druzdzel

University of Pittsburgh
School of Information Sciences
and Intelligent Systems Program
marek@sis.pitt.edu
http://www.pitt.edu/~druzdzel

Outline

- Data Warehousing
- Hadoop/MapReduce
- Pig
- Hive

Reminder: Fall 2015 Term Project

“Data Analytics” term project Fall 2015

Predict survival on the Titanic

https://www.kaggle.com/c/titanic

In this competition you are asked to predict what sorts of people were likely to survive in Titanic disaster. In particular, you have to apply the tools of machine learning to predict which passengers survived the tragedy.

“Data Analytics” term project Fall 2015

The data:

survived	pclass	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked
0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171	7.25		S
1	1	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38	1	0	PC 17599	71.2833	C85	C
1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/ O2. 310128	7.925		S
1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	1E+05	53.1	C123	S
0	3	Allen, Mr. William Henry	male	35	0	0	4E+05	8.05		S
0	3	Moran, Mr. James	male		0	0	3E+05	8.4583		Q
0	1	McCarthy, Mr. Timothy J	male	54	0	0	17463	51.8625	E46	S
0	3	Palsson, Master. Gosta Leonard	male	2	3	1	3E+05	21.075		S
1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27	0	2	3E+05	11.1333		S
1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14	1	0	2E+05	30.0708		C
1	3	Sandstrom, Miss. Marguerite Rut	female	4	1	1	PP 9549	16.7	G6	S
1	1	Bonnell, Miss. Elizabeth	female	58	0	0	1E+05	26.55	C103	S
0	3	Saundercock, Mr. William Henry	male	20	0	0	A/5. 2151	8.05		S
0	3	Andersson, Mr. Anders Johan	male	39	1	5	3E+05	31.275		S
0	3	Vestrom, Miss. Hulda Amanda Adolfina	female	14	0	0	4E+05	7.8542		S
1	2	Hewlett, Mrs. (Mary D Kingcome)	female	55	0	0	2E+05	16		S
0	3	Rice, Master. Eugene	male	2	4	1	4E+05	29.125		Q
1	2	Williams, Mr. Charles Eugene	male		0	0	2E+05	13		S
0	3	Vander Planke, Mrs. Julius (Emelia Maria Vandemoortele)	female	31	1	0	3E+05	18		S
1	3	Masselmani, Mrs. Fatima	female		0	0	2649	7.225		C
0	2	Fynney, Mr. Joseph J	male	35	0	0	2E+05	26		S
1	2	Beesley, Mr. Lawrence	male	34	0	0	2E+05	13	D56	S
1	3	McGowan, Miss. Anna "Annie"	female	15	0	0	3E+05	8.0292		Q
1	1	Sloper, Mr. William Thompson	male	28	0	0	1E+05	35.5	A6	S
0	3	Palsson, Miss. Torborg Danira	female	8	3	1	3E+05	21.075		S
1	3	Asplund, Mrs. Carl Oscar (Selma Augusta Emilia Johansson)	female	38	1	5	3E+05	31.3875		S
0	3	Emir, Mr. Farred Chehab	male		0	0	2631	7.225		C

The task: Predict survival

- Not really “Big Data”
- Not really “Big $$$” ☹ for the best team
- Nice end date ☺

- Fortunately, we have a whole infrastructure for the competition ☺

“Data Analytics” term project Fall 2015

Your task:

Be the best team by Friday, 4 December 2015

Your first step (do it ASAP):

Form teams and sign up at
https://www.kaggle.com/c/titanic

Introduction

Processing

"*If you aren't taking advantage of big data, then you don't have big data, you have just a pile of data*,"

-- Jay Parikh, VP of infrastructure at Facebook

Data Warehouses

A producer wants to know ...

Which are our lowest/highest margin customers ?

What is the most effective distribution channel?

Who are my customers and what products are they buying?

What product promo-tions have the biggest impact on revenue?

Which customers are most likely to go to the competition?

What impact will new products/services have on revenue and margins?

There is plenty of data, and yet …

"I can't find the data I need"

- data is scattered over the network
- many versions, subtle differences

"I can't get the data I need"

- need an expert to get the data

"I can't understand the data I found"

- available data poorly documented

"I can't use the data I found"

- results are unexpected
- data needs to be transformed from one form to other

The need for business intelligence

- Maintain competitive edge
  - Market / customer knowledge
  - Fast, easy access to information
- Improve business efficiency
  - Reduce costs
  - Streamline processes

Data analysis and data warehousing

Data warehousing provides an enterprise with a memory

Data analysis provides the enterprise with intelligence

We want to know ...

- Given a database of 100,000 names, which persons are the least likely to default on their credit cards?
- Which types of transactions are likely to be fraudulent given the demographics and transactional history of a particular customer?
- If I raise the price of my product by $1, what is the effect on my ROI (Return on Investment)?
- If I offer only 2,500 airline miles as an incentive to purchase rather than 5,000, how many lost responses will result?
- If I emphasize ease-of-use of the product as opposed to its technical capabilities, what will be the net effect on my revenues?
- Which of my customers are likely to be the most loyal?

Definitions of a data warehouse

"A subject-oriented, integrated, time-variant and non-volatile collection of data in support of management's decision making process"

[W.H. Inmon]

"A copy of transaction data, specifically structured for query and analysis"

[Ralph Kimball]

"A single, complete and consistent store of data obtained from a variety of different sources made available to end users in a what they can understand and use in a business context."

[Barry Devlin]

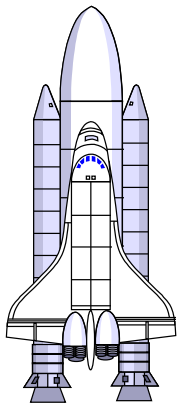
Data warehouse

- For organizational learning to take place, data from many sources must be gathered together and organized in a consistent and useful way – hence, "data warehousing"
- The data warehouse is a collection of data that is pulled together primarily from operational business systems and is structured and tuned for easy access and use by information consumers and analysts, especially for the purpose of decision making.
- The goal of data warehousing is to integrate enterprise wide corporate data into a single repository from which users can easily run queries.
- Data warehouse is an organization's (enterprise's) memory.

Explorers, farmers and tourists

Farmers: Harvest information from known access paths

Tourists: Browse information harvested by farmers



Explorers: Seek out the unknown and previously unsuspected rewards hiding in the detailed data

Data warehouse

Enterprise
"Database"

Customers

Orders

Vendors

etc., …

etc., …

Transactions

Copied,
organized
summarized

Data
Warehouse

Data Mining

Data Miners:
- "Farmers" – they know
- "Explorers" – unpredictable

Data warehouse

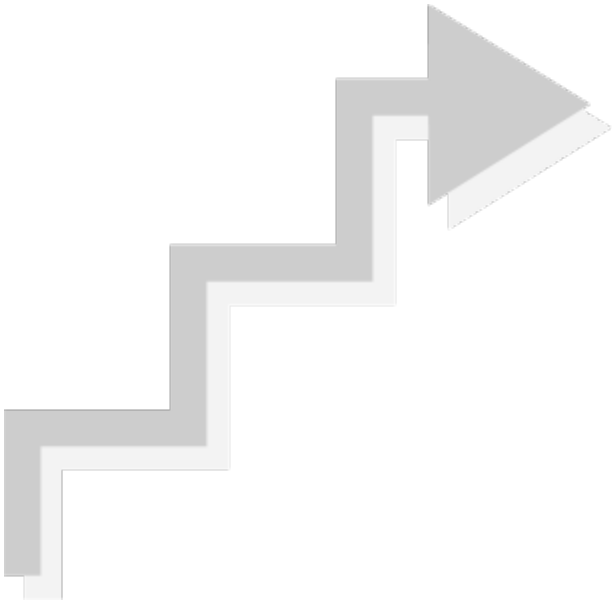
- A data warehouse is a copy of transaction data specifically structured for querying, analysis and reporting – hence, data mining.
- Note that the data warehouse contains a copy of the transactions which are not updated or changed later by the transaction system.
- Also note that this data is specially structured, and may have been transformed when it was copied into the data warehouse.

Expectations from a data warehouse

- Data should be integrated across the enterprise
- Summary data has a real value to the organization
- Historical data holds the key to understanding data over time
- What-if capabilities are required

What is data warehousing?

Information



Data

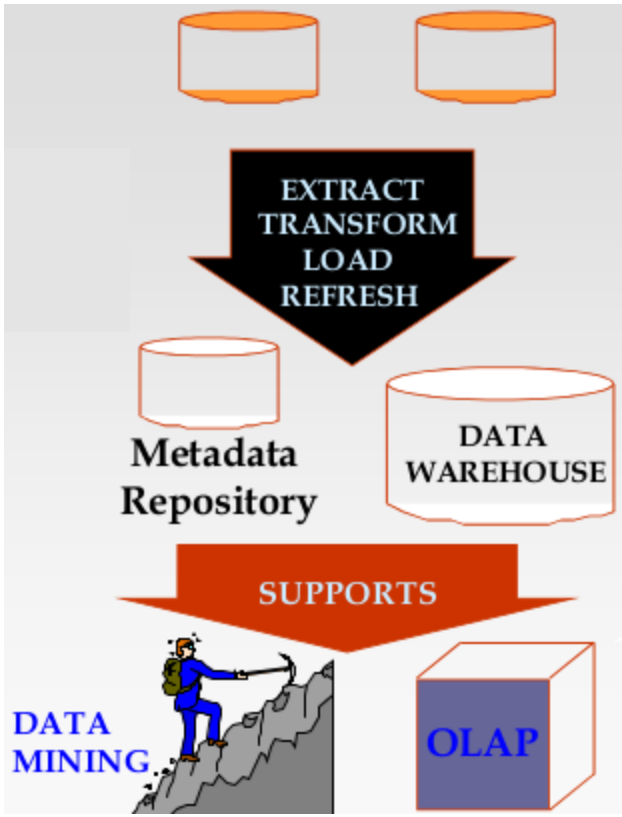
A process of transforming data into information and making it available to users in a timely enough manner to make a difference

[Forrester Research, April 1996]

Data Warehousing

- Intergraded data spanning over long time periods, often augmented with summary information
- Several gigabytes to terabytes common
- Interactive response times expected for complex queries; ad-hoc updates uncommon

external data sources



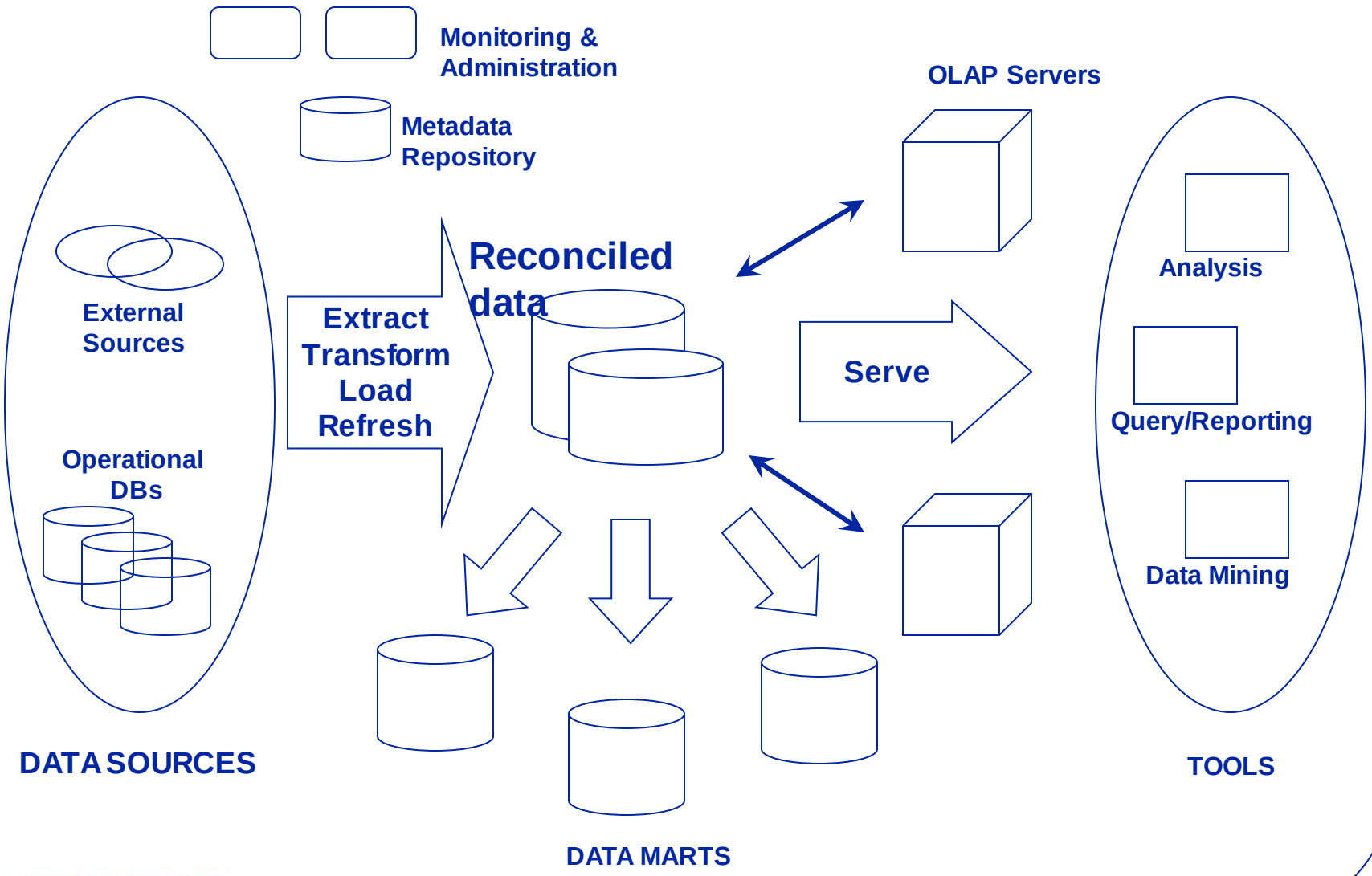
INFSCI 2711 slides of Prof. Zadorozhny class

Warehousing issues

- **Semantic Integration**: When getting data from multiple sources, must eliminate mismatches, e.g., different units (temperature, weight, currency).
- **Heterogeneous Sources**: Must access data from a variety of source formats and repositories
- **Load, Refresh, Purge**: Must load data, periodically refresh it, and purge too-old data
- **Metadata Management**: Must keep track of sources, loading time, and other information for all data in the warehouse

Data Warehouse Architecture

Data warehouse architecture



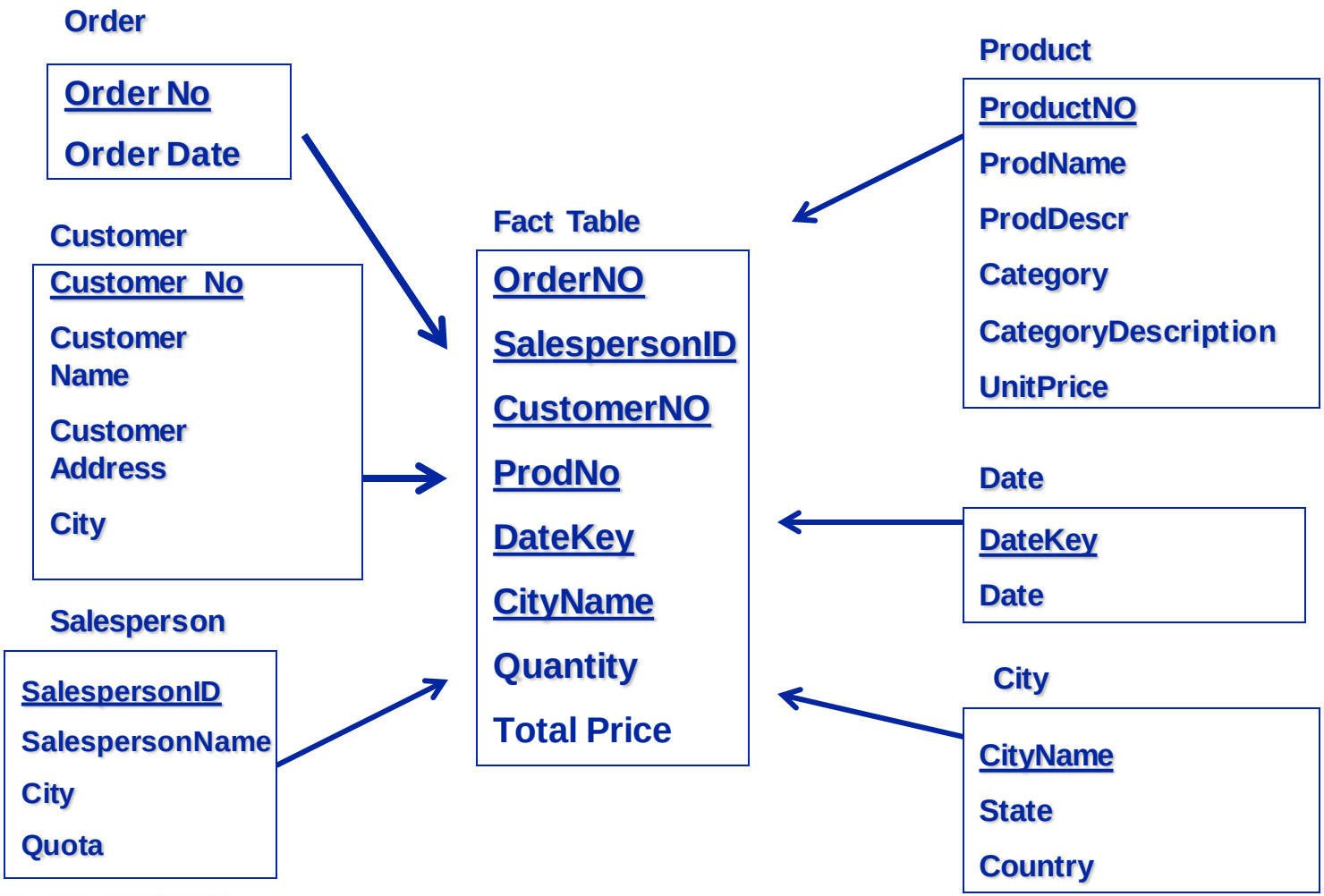
Data warehouse architecture

- Data warehouse server
  - almost always a relational DBMS, rarely flat files
- OLAP servers
  - to support and operate on multi-dimensional data structures
- Clients
  - Query and reporting tools
  - Analysis tools
  - Data mining tools

Data warehouse schema

- “Star” schema
- “Fact constellation” schema
- “Snowflake” schema

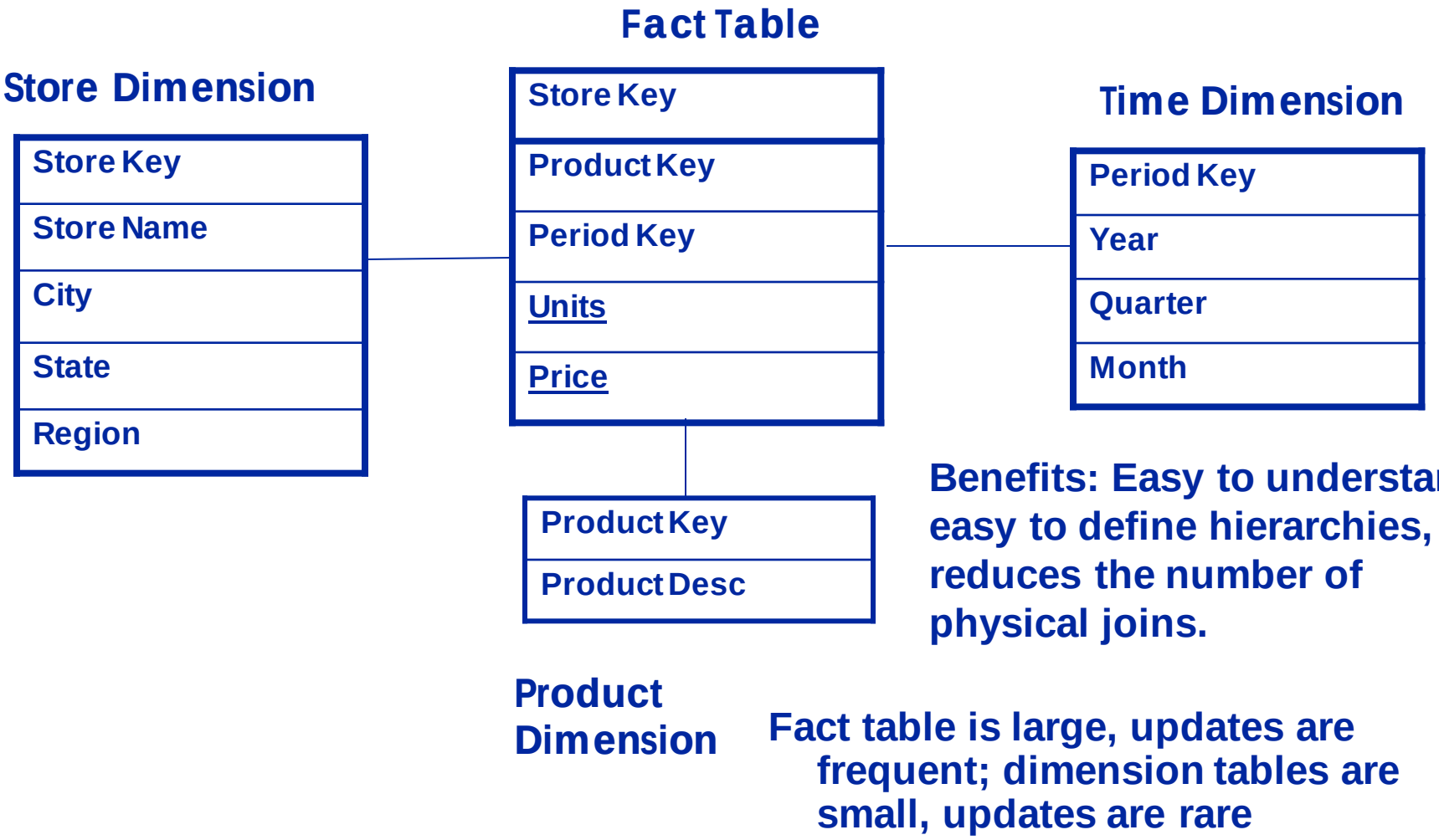
Example of a “star” schema



"Star" schema

- A single, typically large, fact table and one table for each dimension
- Every fact points to one tuple in each of the dimensions and has additional attributes
- Does not capture hierarchies directly
- Generated keys are used for performance and maintenance reasons

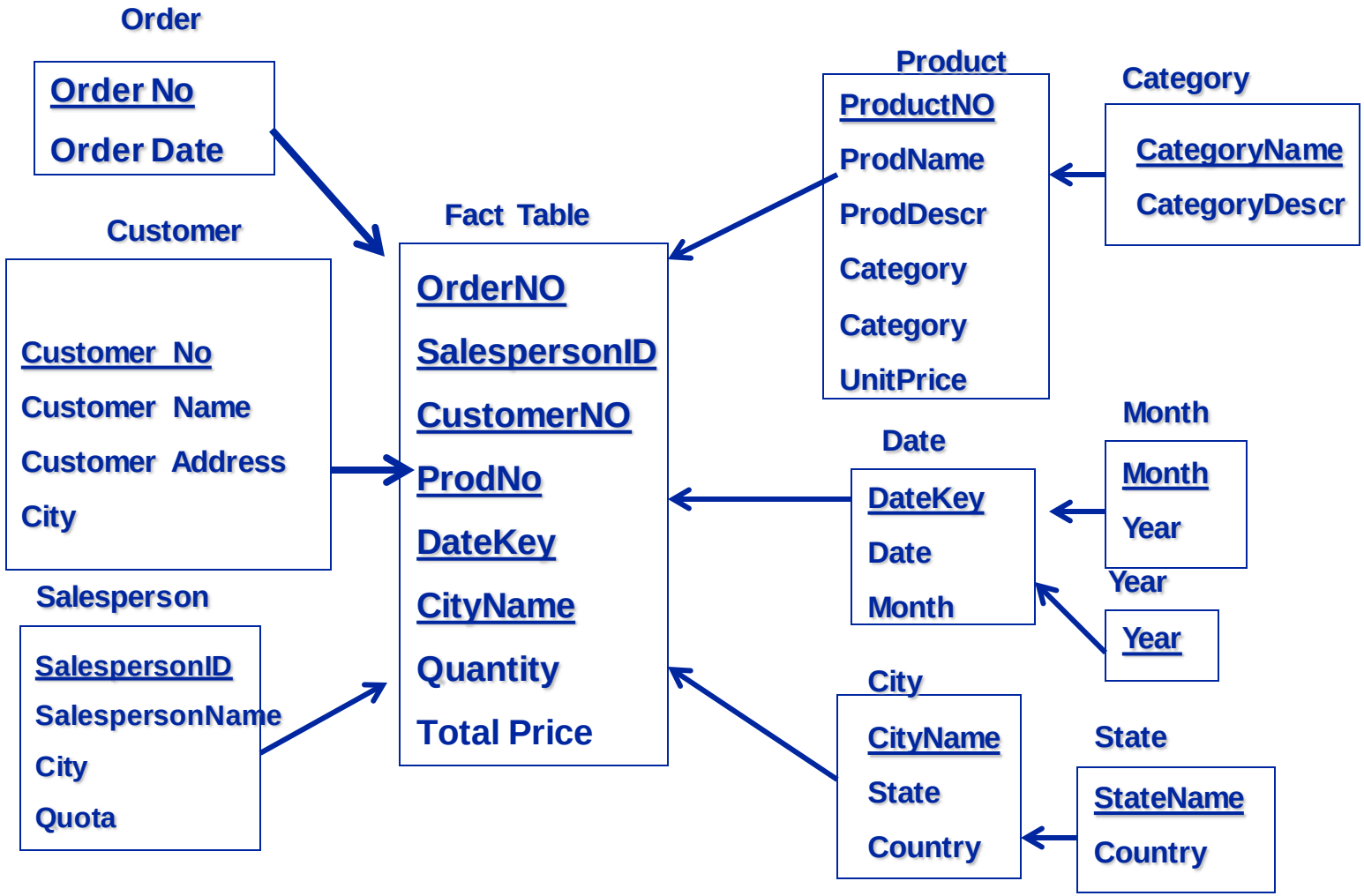
"Star" schema (cont.)



Benefits: Easy to understand, easy to define hierarchies, reduces the number of physical joins.

Fact table is large, updates are frequent; dimension tables are small, updates are rare

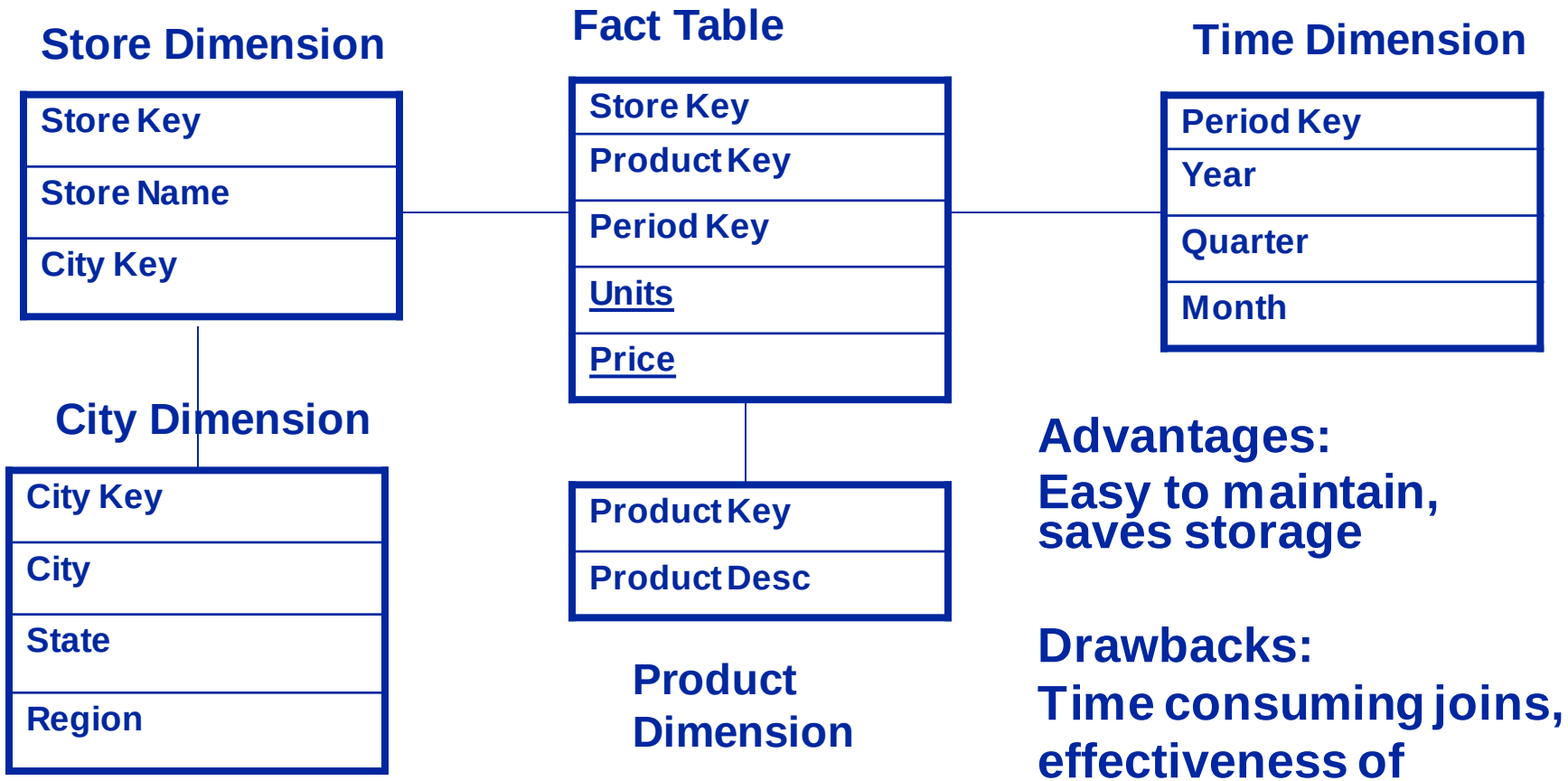
Example of a “snowflake” schema



"Snow flake" schema

- A single, large, and central fact table and one or more tables for each dimension.
- Dimension tables are normalized, i.e., split dimension table data into additional tables
- Represent dimensional hierarchy directly by normalizing the dimension tables

"Snow flake" schema (cont.)



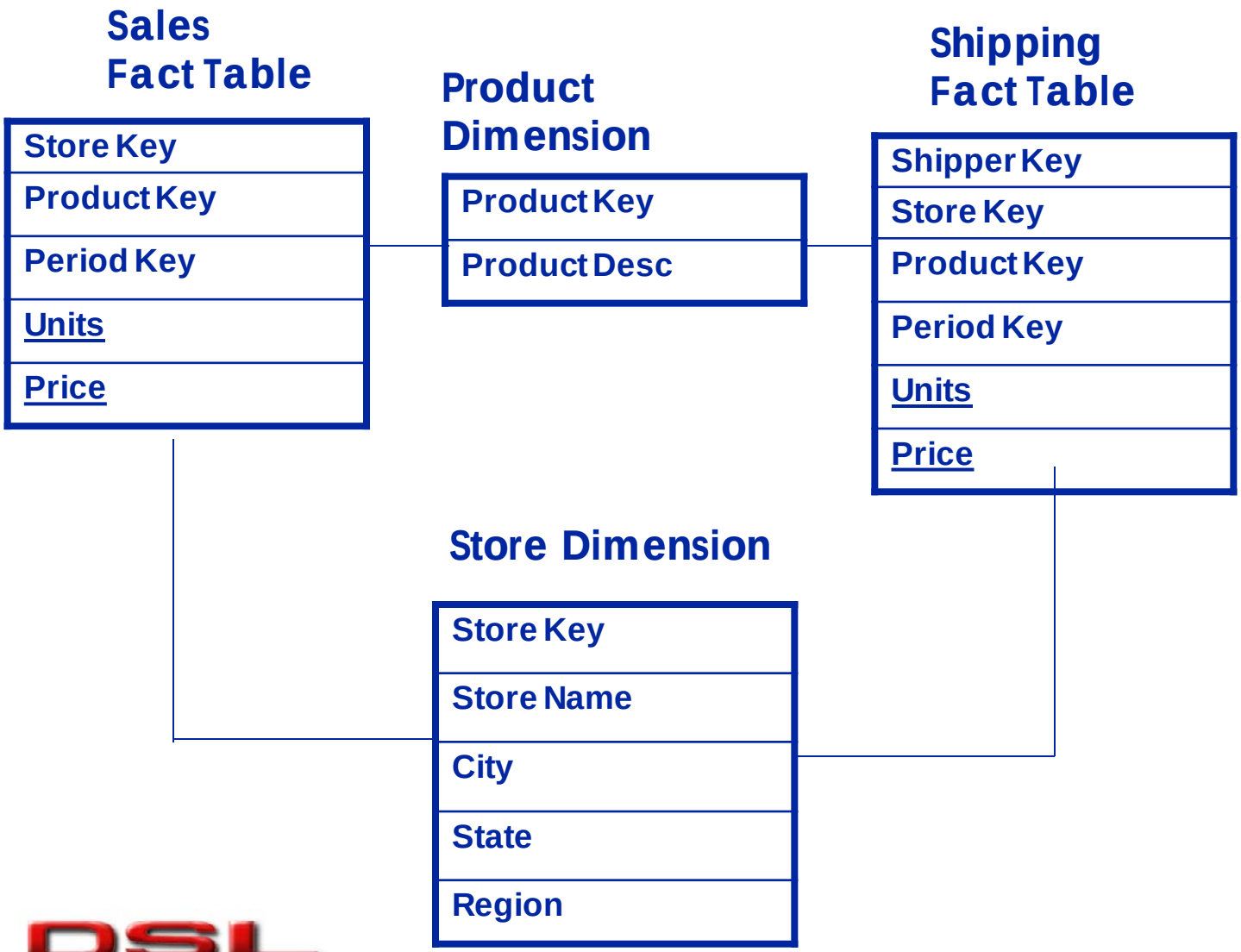
Advantages:
Easy to maintain, saves storage

Drawbacks:
Time consuming joins, effectiveness of browsing suffers, report generation may be slow

“Fact constellation” schema

- Multiple fact tables share dimension tables.
- This schema is viewed as collection of stars hence called “galaxy schema” or “fact constellation.”
- Sophisticated application may require such schema.
- Example: Projected expense and the actual expense may share dimensional tables.

"Fact constellation" schema (cont.)



Virtual warehouse

Created by providing a database view on the operational databases.

- Materialize some summary views for efficient query processing
- Easier to build ☺
- May put too much load on the operational DB servers ☹

OLAP

OLAP and Data Warehousing

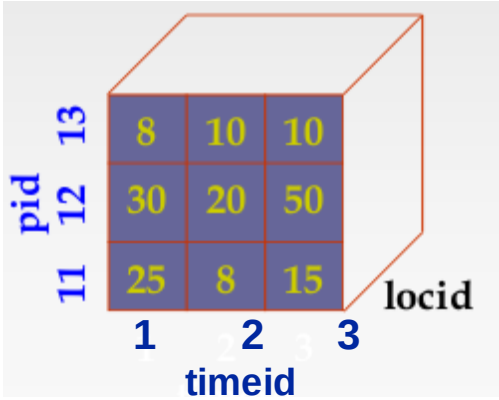
- Data Warehousing: Consolidate data from many sources into one large repository
  - ETL: Extract, Transform (semantic integration), and Load
- OLAP: Online Analytic Processing
  - Complex SQL queries and views
  - Queries based on spreadsheet-style and "multidimensional" view of data
  - Interactive and "online" queries

OLAP: Multidimensional data model

Collection of numeric measures which depend on a set of dimensions

e.g., measure Sales, dimensions Product (pid), Location (locid), and Time (timeid).

Slice locid = 1 is shown:



pid	timeid	locid	sales
11	1	1	25
11	2	1	8
11	3	1	15
12	1	1	30
12	2	1	20
12	3	1	50
13	1	1	8
13	2	1	10
13	3	1	10
11	1	2	35

Sales

OLAP queries

- Influenced by both SQL and spreadsheets
- A common operation is to aggregate (roll-up) a measure over one or more dimensions
  - Find total sales
  - Find total sales for each city or for each state
  - Find top five products ranked by total sales

Operations in multidimensional data model

- **Aggregation** (*roll-up*)
  - dimension reduction: e.g., total sales by city
  - summarization over aggregate hierarchy: e.g., total sales by city and year -> total sales by region and by year
- **Selection** (*slice*) defines a sub-cube
  - e.g., sales where city = Palo Alto and date = 1/15/96
- **Navigation** to detailed data (*drill-down*)
  - e.g., (sales - expense) by city, top 3% of cities by average income
- **Visualization** operations (e.g., *pivot*)

OLAP Queries

- **Drill-down**: The inverse of roll-up
  - e.g., given total sales by state, can drill-down to get total sales by city
  - e.g., can also drill-down on different dimension to get total sales by product for each state
- **Pivoting**: Aggregation on selected dimensions
  - e.g., pivoting on Location and Time

  yields the following cross-tabulation:

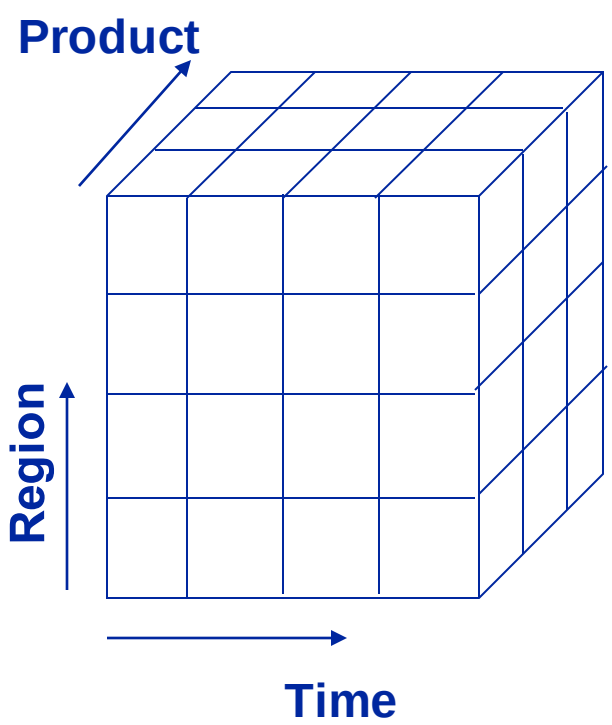
	WI	CA	Total
1995	63	81	144
1996	38	107	145
1997	75	35	110
Total	176	223	339

OLAP Cube

pid	timeid	locid	sales
11	1	1	25
11	2	1	8
11	3	1	15
12	1	1	30
12	2	1	20
12	3	1	50
13	1	1	8
13	2	1	10
13	3	1	10
11	1	2	35

OLAP operations

“drill up” (also “roll up”)



Category (e.g., electrical appliance)

Subcategory (e.g., kitchen)

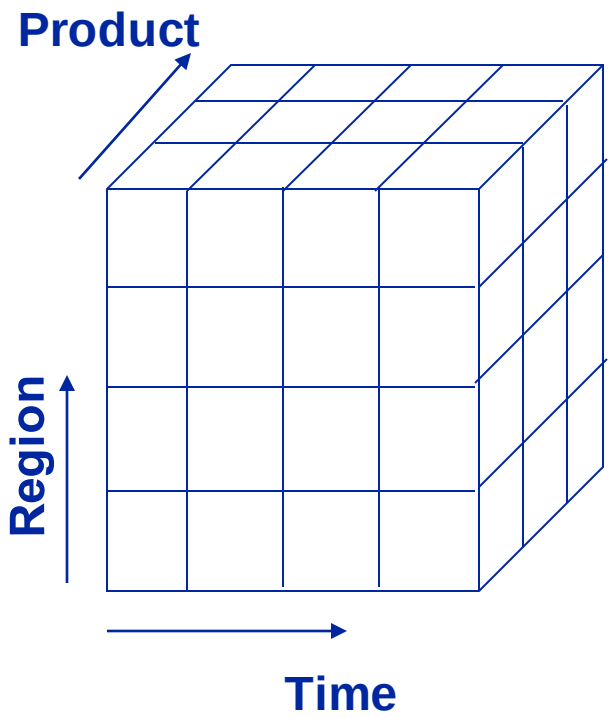
Product (e.g., toaster)

Aggregating at different levels of a dimension hierarchy

- **e.g., given total sales per product, we can drill up to get sales per category**
- **e.g., given total sales by city, we can drill up to get sales by state**

OLAP operations

"drill down" (also "roll down")



Category (e.g., electrical appliances)

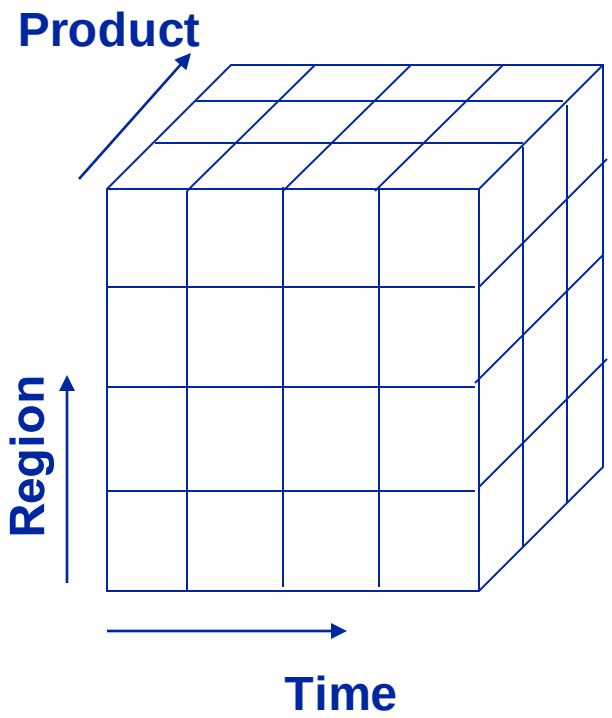
↳ Subcategory (e.g., kitchen)

↳ Product (e.g., toaster)

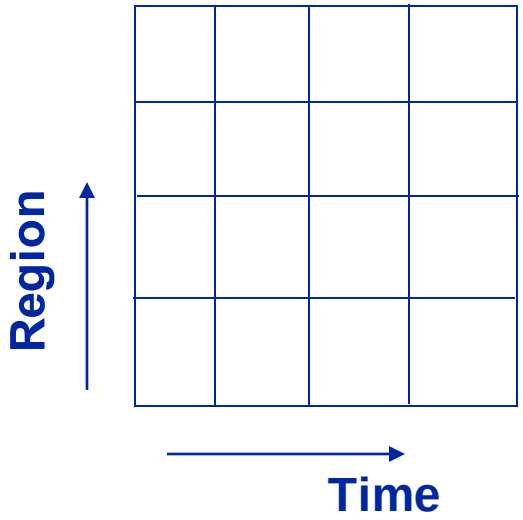
- The inverse of roll-up
  - e.g., given total sales by category, can drill-down to get total sales by product
  - e.g., can also drill-down on different dimension to get total sales by product for each state

OLAP operations

"slice and dice"



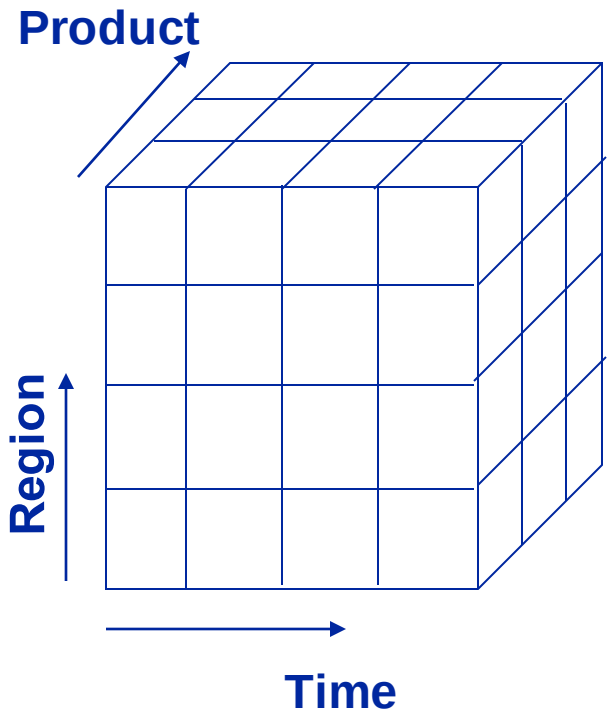
Product=Toaster

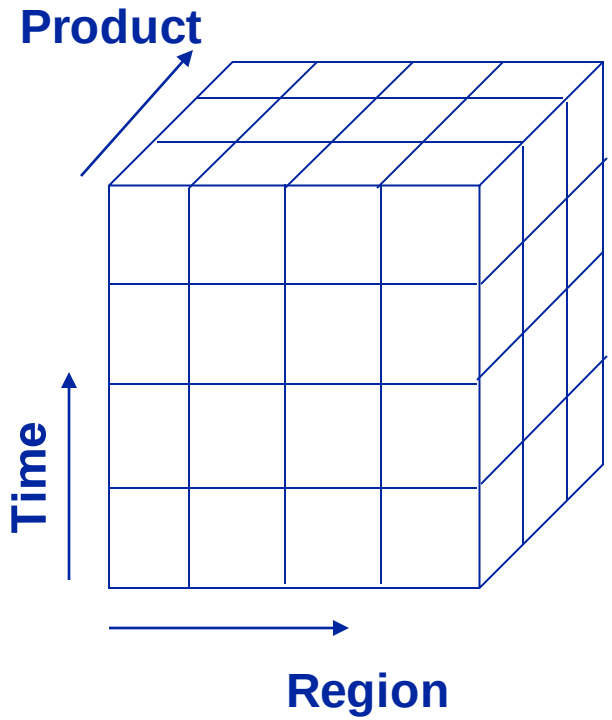


- a.k.a. "selection"
- defines a sub-cube
  - e.g., sales of toasters
  - e.g., sales where region = PA and date = 1/15/96

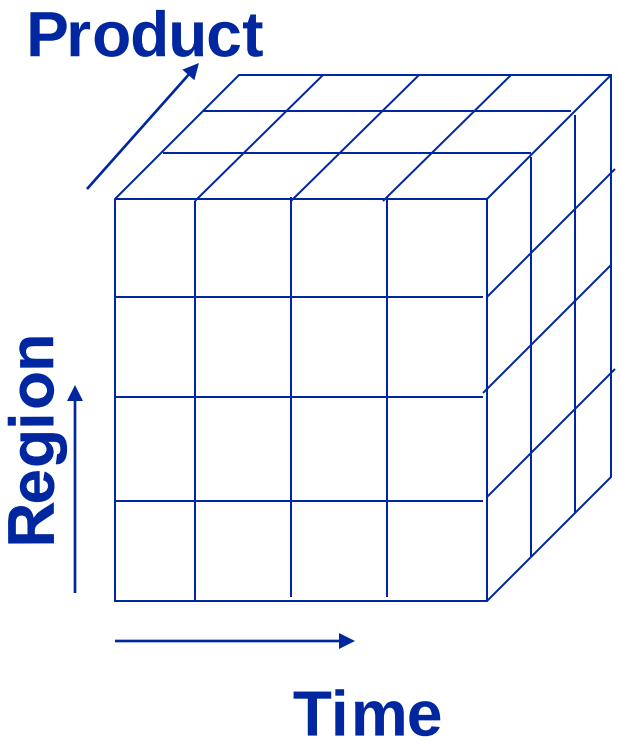
OLAP operations

“pivot” (rotation), a visual operation

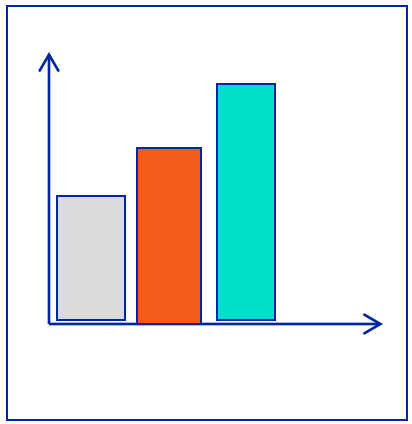




Presentation



Reporting Tool



Report

Hadoop and MapReduce

What's in the name?

"*The name my kid gave a stuffed yellow elephant. Short, relatively easy to spell and pronounce, meaningless, and not used elsewhere: those are my naming criteria. Kids are good at generating such. Googol is a kid's term*."

-- Hadoop project's creator, Doug Cutting

Tom White, Hadoop: The Definitive Guide, 3rd Edition, 2012

Motivation

- 1990:
  - One drive – 1,370 MB with transfer speed of 4.4 MB/s
  - Read full drive in around 5 minutes.
- Today:
  - One drive – 1Tb with transfer speed of 100 MB/s (access speed has not kept up with disk capacity)
  - Read full drive in 2.5 hours (writing is even slower!)
- What if
  - We had 100 drives, each holding 1/100 of the data
  - We could read the data in less than 2 minutes

Tom White, Hadoop: The Definitive Guide, 3rd Edition, 2012

Hadoop

The Hadoop project includes:

- **Hadoop Common**: The common utilities that support the other Hadoop modules (A set of components and interfaces for distributed file systems and general I/O, e.g., serialization, Java RPC, persistent data structures).
- **Hadoop Distributed File System (HDFS™)**: A distributed file system that provides high-throughput access to application data located on large clusters of commodity machines.
- **Hadoop YARN**: A framework for job scheduling and cluster resource management.
- **Hadoop MapReduce**: A YARN-based system for parallel processing of large data sets. A distributed data processing model and execution environment that runs on large clusters of commodity machines.

http://hadoop.apache.org/

Hadoop

- **Avro**: A serialization system for efficient, cross-language RPC, and persistent data storage.
- **Pig**: A data flow language and execution environment for exploring very large datasets. Pig runs on HDFS and MapReduce clusters.
- **Hive**: A distributed data warehouse. Hive manages data stored in HDFS and provides a query language based on SQL (and which is translated by the runtime engine to MapReduce jobs) for querying the data.
- **Cassandra™**: A scalable multi-master database with no single points of failure.
- **Chukwa**: A data collection system for managing large distributed systems.
- **Hbase**: A distributed, column-oriented database. HBase uses HDFS for its underlying storage and supports both batch-style computations using MapReduce and point queries (random reads).
- **ZooKeeper**: A distributed, highly available coordination service. ZooKeeper provides primitives such as distributed locks that can be used for building distributed applications.
- **Sqoop**: A tool for efficiently moving data between relational databases and HDFS.
- **Mahout**: A Scalable machine learning and data mining library.

http://hadoop.apache.org/;
Tom White, Hadoop: The Definitive Guide, 3rd Edition, 2012

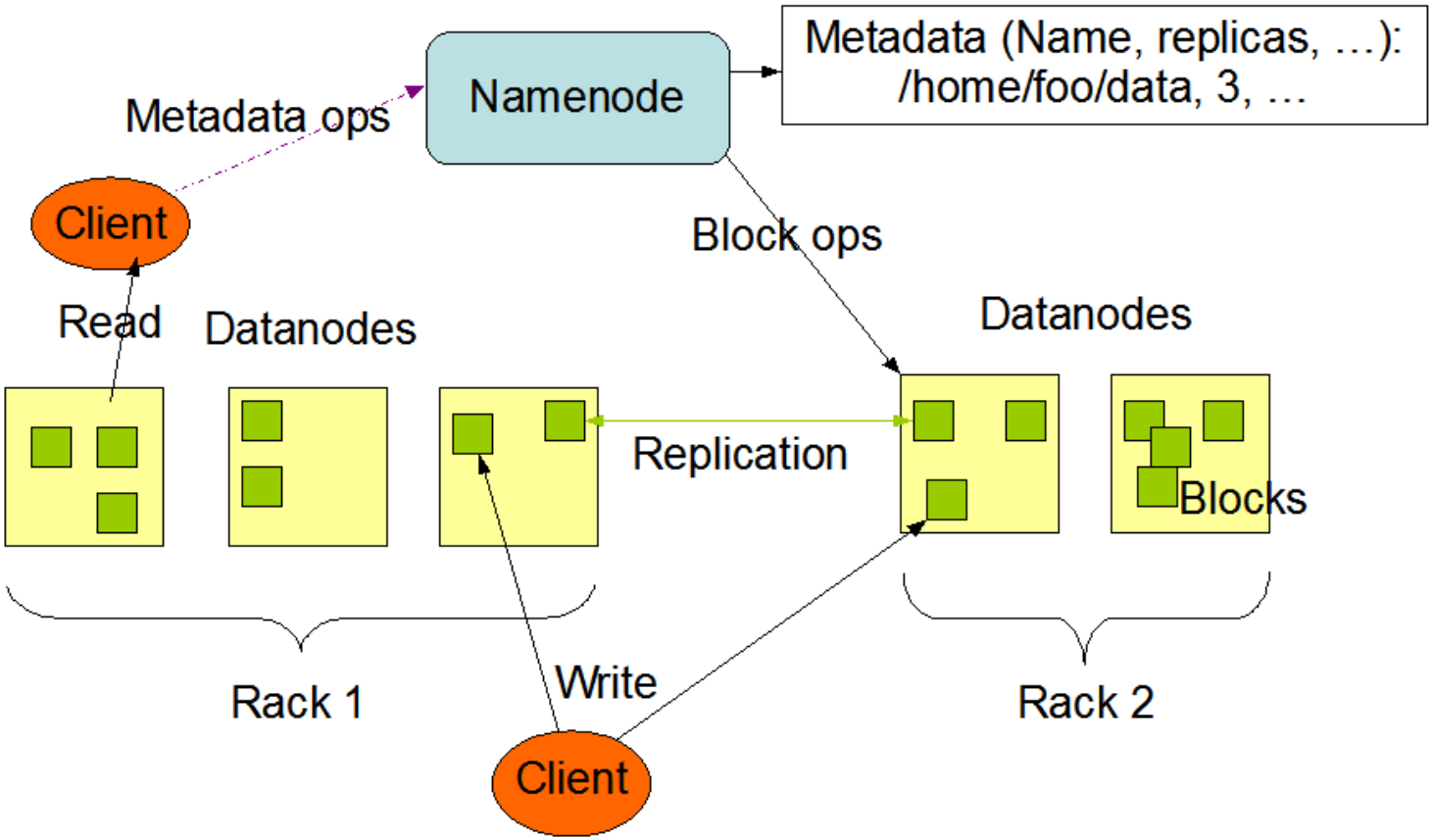
Assumptions and goals

- Hardware failure (with many computers, it is norm rather than exception: Detection of faults and quick recovery is a core architectural goal of HDFS)
- Streaming data access (designed for batch processing of data, emphasis on high throughput)
- Large data sets (gigabytes to terabytes in size)
- Simple coherency model for files ("write-once-read-many:" a file once created should not be changed; quite typical in data analytics)
- "Moving computation is cheaper than moving data" (computation requested by an application is much more efficient when executed near the data that it operates on; minimizes communication and congestion)
- Portability across heterogeneous hardware and software platforms

http://hadoop.apache.org/docs/hdfs/r0.22.0/hdfs_design.html

Architecture

HDFS Architecture



http://hadoop.apache.org/docs/hdfs/r0.22.0/hdfs_design.html

NameNode and DataNode

- Single NameNode
  - manages the file system namespace
  - regulates access to files by clients
  - opening, closing, and renaming files and directories
  - determines the mapping of blocks to DataNodes
- Many DataNodes
  - serving read and write requests from the file system's clients
  - block creation, deletion, and replication upon instruction from the NameNode

http://hadoop.apache.org/docs/hdfs/r0.22.0/hdfs_design.html

Commands

- hadoop fs -<command> <arguments>
- help,
- cat, chmod, cp, get
- ls, mkdir, mv, put
- rm, rmr
- copyFromLocal, copyToLocal

http://hadoop.apache.org/docs/hdfs/r0.22.0/hdfs_design.html

Hadoop and MapReduce

- Automatic parallel execution, fault tolerance, load balancing
- Run-timer takes care of failing nodes, data partitioning, result merging
- Runs on huge cluster of commodity machines
- Primitive operations: split the data, process them separately, combine the result

More than ten thousand distinct programs have been implemented using MapReduce at Google

Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: Simplified data processing on large clusters. *CACM* 51(1), January 2008

Hadoop and MapReduce

Programming model:

- Input – key/value pairs
- Output – key/value pairs
- Two functions:
  - » Map: (K1, V1) → list(K2, V2)
    - MapReduce library: groups intermediate results and sends to reduce
  - » Reduce: (K2, list(V2)) → list(K3, V3)

MapReduce example: Maximum temperature calculation

Example 2-1. Format of a National Climate Data Center record

```
0057
332130   # USAF weather station identifier
99999    # WBAN weather station identifier
19500101 # observation date
0300     # observation time
4
+51317   # latitude (degrees x 1000)
+028783  # longitude (degrees x 1000)
FM-12
+0171    # elevation (meters)
99999
V020
320      # wind direction (degrees)
1        # quality code
N
0072
1
00450    # sky ceiling height (meters)
1        # quality code
C
N
010000   # visibility distance (meters)
1        # quality code
N
9
-0128    # air temperature (degrees Celsius x 10)
1        # quality code
-0139    # dew point temperature (degrees Celsius x 10)
1        # quality code
10268    # atmospheric pressure (hectopascals x 10)
1        # quality code
```

Data from the National Climatic Data Center (NCDC, http://www.ncdc.noaa.gov/).

Stored using a line-oriented ASCII format.

MapReduce example: Maximum temperature calculation

Raw data


```
0067011990999991950051507004...9999999N9+00001+99999999999...
0043011990999991950051512004...9999999N9+00221+99999999999...
0043011990999991950051518004...9999999N9-00111+99999999999...
0043012650999991949032412004...0500001N9+01111+99999999999...
0043012650999991949032418004...0500001N9+00781+99999999999...
```

Pre-processed data (as presented to the map function)

```
(0, 0067011990999991950051507004...9999999N9+00001+99999999999...)
(106, 0043011990999991950051512004...9999999N9+00221+99999999999...)
(212, 0043011990999991950051518004...9999999N9-00111+99999999999...)
(318, 0043012650999991949032412004...0500001N9+01111+99999999999...)
(424, 0043012650999991949032418004...0500001N9+00781+99999999999...)
```

The output of the map function

```
(1950, 0)
(1950, 22)
(1950, -11)
(1949, 111)
(1949, 78)
```

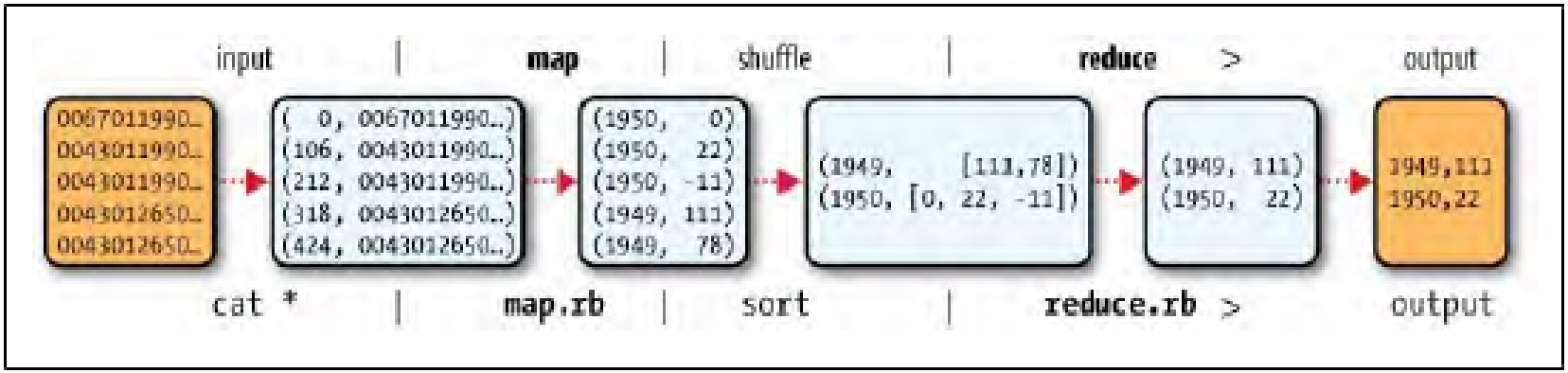
Input to reduce function (after processing by the MapReduce framework)

```
(1949, [111, 78])
(1950, [0, 22, -11])
```

Output of the reduce function

```
(1949, 111)
(1950, 22)
```

MapReduce example: Maximum temperature calculation: Logical data flow



MapReduce example: Maximum temperature calculation

Example 2-3. Mapper for maximum temperature example

```
import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;

public class MaxTemperatureMapper extends MapReduceBase
  implements Mapper<LongWritable, Text, Text, IntWritable> {

  private static final int MISSING = 9999;

  public void map(LongWritable key, Text value,
      OutputCollector<Text, IntWritable> output, Reporter reporter)
      throws IOException {

    String line = value.toString();
    String year = line.substring(15, 19);
    int airTemperature;
    if (line.charAt(87) == '+') { // parseInt doesn't like leading plus signs
      airTemperature = Integer.parseInt(line.substring(88, 92));
    } else {
      airTemperature = Integer.parseInt(line.substring(87, 92));
    }
    String quality = line.substring(92, 93);
    if (airTemperature != MISSING && quality.matches("[01459]")) {
      output.collect(new Text(year), new IntWritable(airTemperature));
    }
  }
}
```

MapReduce example: Maximum temperature calculation

Example 2-4. Reducer for maximum temperature example

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

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;

public class MaxTemperatureReducer extends MapReduceBase
  implements Reducer<Text, IntWritable, Text, IntWritable> {

  public void reduce(Text key, Iterator<IntWritable> values,
      OutputCollector<Text, IntWritable> output, Reporter reporter)
      throws IOException {

    int maxValue = Integer.MIN_VALUE;
    while (values.hasNext()) {
      maxValue = Math.max(maxValue, values.next().get());
    }
    output.collect(key, new IntWritable(maxValue));
  }
}
```

The output types of the map function have to match the input types of the reduce function!

MapReduce example: Maximum temperature calculation

Example 2-5. Application to find the maximum temperature in the weather dataset

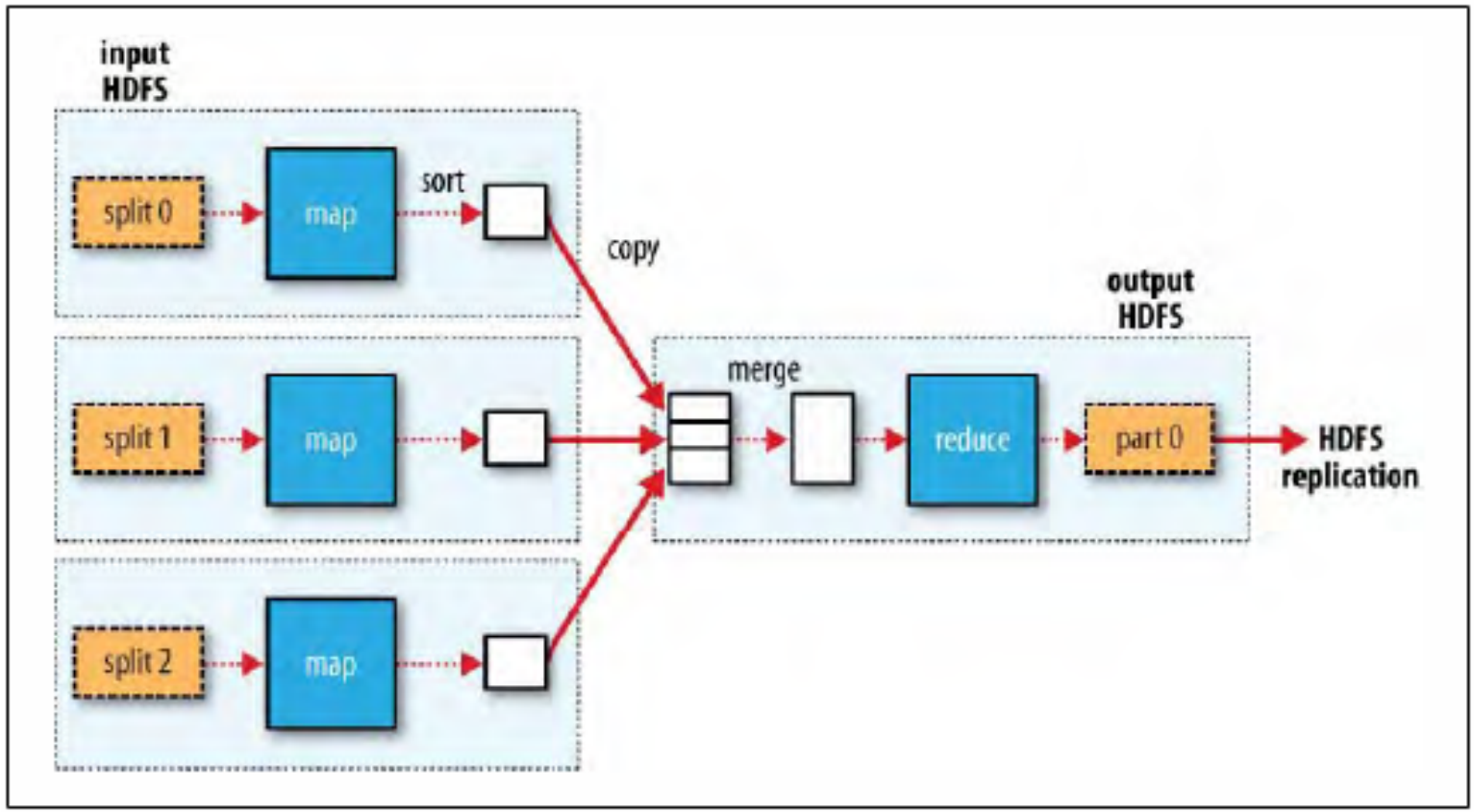
```
import java.io.IOException;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;

public class MaxTemperature {

  public static void main(String[] args) throws IOException {
    if (args.length != 2) {
      System.err.println("Usage: MaxTemperature <input path> <output path>");
      System.exit(-1);
    }

    JobConf conf = new JobConf(MaxTemperature.class);
    conf.setJobName("Max temperature");

    FileInputFormat.addInputPath(conf, new Path(args[0]));
    FileOutputFormat.setOutputPath(conf, new Path(args[1]));

    conf.setMapperClass(MaxTemperatureMapper.class);
    conf.setReducerClass(MaxTemperatureReducer.class);

    conf.setOutputKeyClass(Text.class);
    conf.setOutputValueClass(IntWritable.class);

    JobClient.runJob(conf);
  }
}
```

- This is the actual code that does the MapReduce job.
- Links between the stages are created by means of methods such as addInputPath() in FileInputFormat and setOutputPath() method in FileOutputFormat.

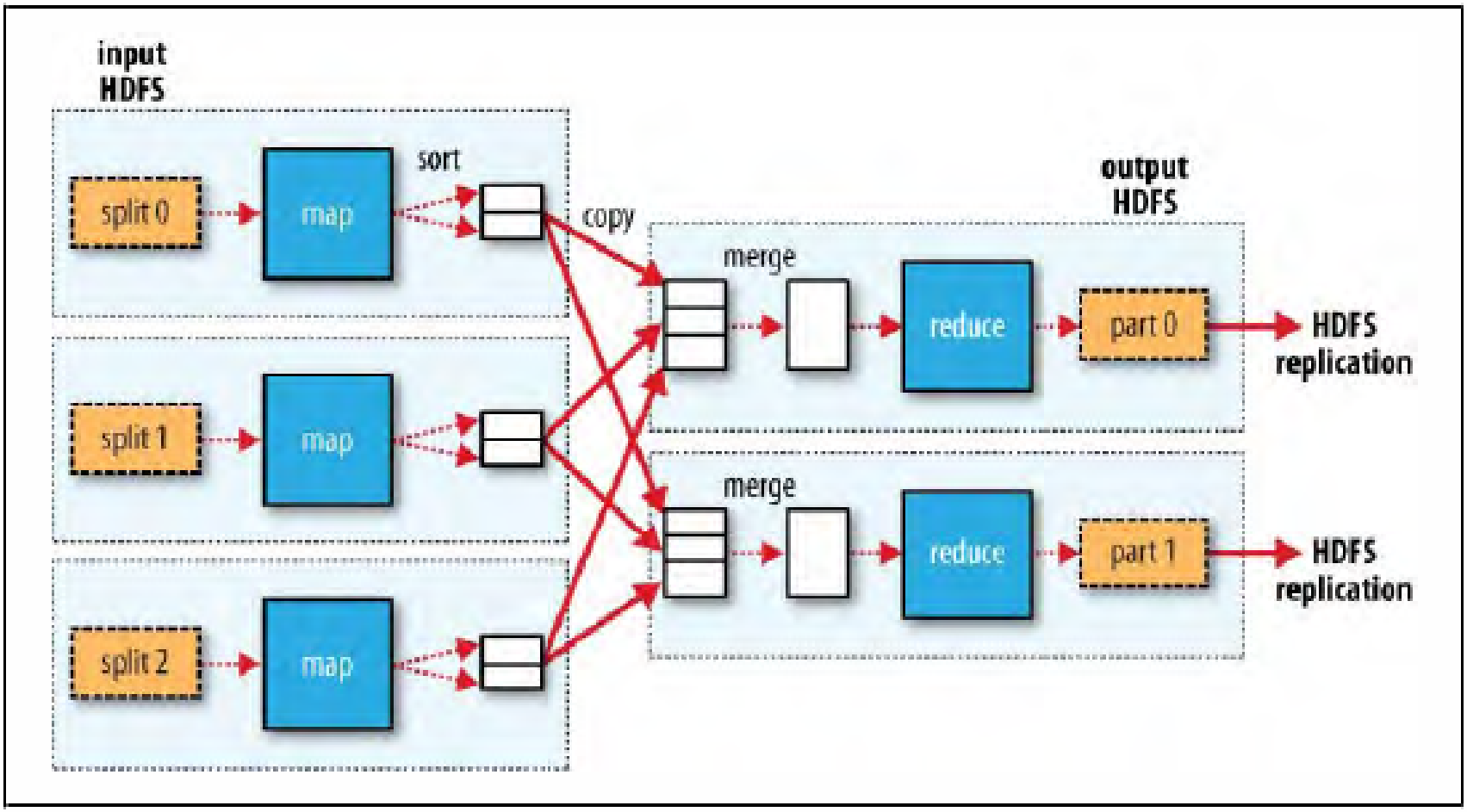
MapReduce example: Maximum temperature calculation


```
% export HADOOP_CLASSPATH=build/classes
% hadoop MaxTemperature input/ncdc/sample.txt output
09/04/07 12:34:35 INFO jvm.JvmMetrics: Initializing JVM Metrics with processName=Job
Tracker, sessionId=
09/04/07 12:34:35 WARN mapred.JobClient: Use GenericOptionsParser for parsing the
arguments. Applications should implement
09/04/07 12:34:35 WARN mapred.JobClient:
be found. See JobConf(Class) or JobConf#
09/04/07 12:34:35 INFO mapred.FileInputF
09/04/07 12:34:35 INFO mapred.JobClient:
09/04/07 12:34:35 INFO mapred.FileInputF
09/04/07 12:34:35 INFO mapred.MapTask: n
09/04/07 12:34:35 INFO mapred.MapTask: i
09/04/07 12:34:35 INFO mapred.MapTask: d
09/04/07 12:34:35 INFO mapred.MapTask: r
09/04/07 12:34:35 INFO mapred.MapTask: S
09/04/07 12:34:36 INFO mapred.MapTask: F
09/04/07 12:34:36 INFO mapred.TaskRunner
done. And is in the process of commiting
09/04/07 12:34:36 INFO mapred.LocalJobRu
cdc/sample.txt:0+529
09/04/07 12:34:36 INFO mapred.TaskRunner
09/04/07 12:34:36 INFO mapred.LocalJobRunner:
09/04/07 12:34:36 INFO mapred.Merger: Merging 1 sorted segments
09/04/07 12:34:36 INFO mapred.Merger: Down to the last merge-pass, with 1 segments
left of total size: 57 bytes
09/04/07 12:34:36 INFO mapred.LocalJobRunner:
09/04/07 12:34:36 INFO mapred.TaskRunner: Task:attempt_local_0001_r_000000_0 is done.
And is in the process of commiting
09/04/07 12:34:36 INFO mapred.LocalJobRunner:
09/04/07 12:34:36 INFO mapred.TaskRunner: Task attempt_local_0001_r_000000_0 is
allowed to commit now
09/04/07 12:34:36 INFO mapred.FileOutputCommitter: Saved output of task
'attempt_local_0001_r_000000_0' to file:/Users/tom/workspace/htdg/output
09/04/07 12:34:36 INFO mapred.LocalJobRunner: reduce > reduce
09/04/07 12:34:36 INFO mapred.TaskRunner: Task 'attempt_local_0001_r_000000_0' done.
09/04/07 12:34:36 INFO mapred.JobClient:  map 100% reduce 100%
09/04/07 12:34:36 INFO mapred.JobClient: Job complete: job_local_0001
09/04/07 12:34:36 INFO mapred.JobClient: Counters: 13
09/04/07 12:34:36 INFO mapred.JobClient:   FileSystemCounters
09/04/07 12:34:36 INFO mapred.JobClient:     FILE_BYTES_READ=27571
09/04/07 12:34:36 INFO mapred.JobClient:     FILE_BYTES_WRITTEN=53907
09/04/07 12:34:36 INFO mapred.JobClient:   Map-Reduce Framework
09/04/07 12:34:36 INFO mapred.JobClient:     Reduce input groups=2
09/04/07 12:34:36 INFO mapred.JobClient:     Combine output records=0
09/04/07 12:34:36 INFO mapred.JobClient:     Map input records=5
09/04/07 12:34:36 INFO mapred.JobClient:     Reduce shuffle bytes=0
09/04/07 12:34:36 INFO mapred.JobClient:     Reduce output records=2
09/04/07 12:34:36 INFO mapred.JobClient:     Spilled Records=10
09/04/07 12:34:36 INFO mapred.JobClient:     Map output bytes=45
09/04/07 12:34:36 INFO mapred.JobClient:     Map input bytes=529
09/04/07 12:34:36 INFO mapred.JobClient:     Combine input records=0
09/04/07 12:34:36 INFO mapred.JobClient:     Map output records=5
09/04/07 12:34:36 INFO mapred.JobClient:     Reduce input records=5
```

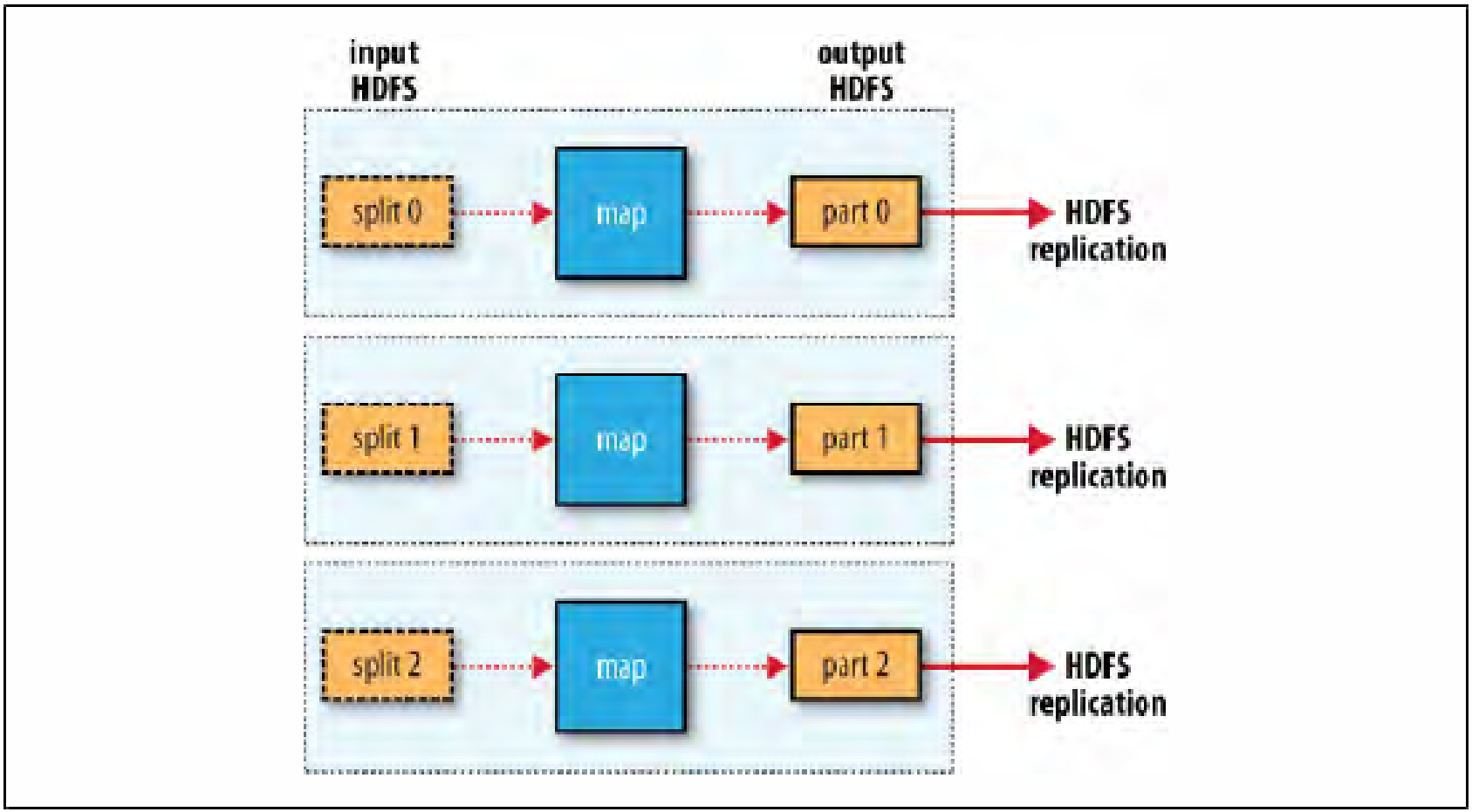
MapReduce data flow (single reduce task)



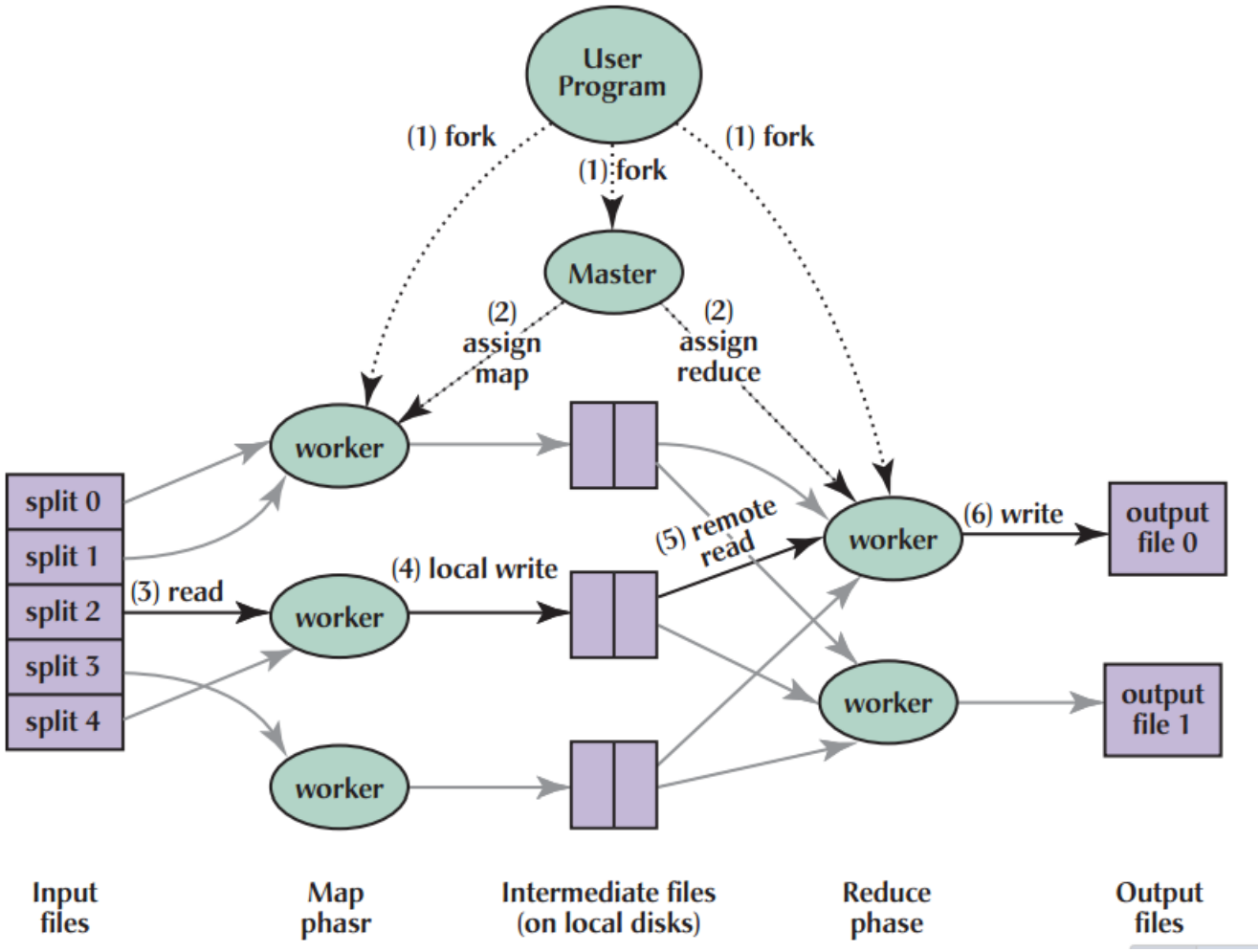
MapReduce data flow (multiple reduce tasks)



MapReduce data flow (no reduce tasks)



Hadoop and MapReduce



Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: Simplified data processing on large clusters. *CACM* 51(1), January 2008

Fault tolerance

- Master pings workers, and reassigns the chunk of work of a failed worker to another worker and notifies other workers of re-execution
- Master periodically writes checkpoints. If master fails – MapReduce operation fails and client may re-execute it again, starting from the last checkpoint

Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: Simplified data processing on large clusters. *CACM* 51(1), January 2008

Pig

Pig

- Created at Yahoo!
- Higher abstraction layer
- Richer data structures
- Set of data transformations
- Extensible
- Designed for batch processing

Tom White, Hadoop: The Definitive Guide, 3rd Edition, 2012

Pig

http://en.wikipedia.org/wiki/Pig_Latin

- Pig is made up of two pieces:
  - The language used to express data flows, called Pig Latin.
  - The execution environment to run Pig Latin programs. There are currently two environments: local execution in a single JVM and distributed execution on a Hadoop cluster.
- A Pig Latin program is made up of a series of operations, or transformations, that are applied to the input data to produce output.

Tom White, Hadoop: The Definitive Guide, 3rd Edition, 2012

Running pig programs

Three ways, work in local and MapReduce mode

- Script
  - » Pig can run a script file that contains Pig commands
- Grunt
  - » Grunt is an interactive shell for running Pig commands
- Embedded
  - » You can run Pig programs from Java using the PigServer class

Tom White, Hadoop: The Definitive Guide, 3rd Edition, 2012

Pig: Example

- Calculate maximum recorded temperature by year:

```
records = LOAD 'input/ncdc/micro-tab/sample.txt'
AS (year:chararray, temperature:int);
filtered_records = FILTER records BY temperature != 9999;
grouped_records = GROUP filtered_records BY year;
max_temp = FOREACH grouped_records GENERATE group,
MAX(filtered_records.temperature);
DUMP max_temp;
```

- Possible result:

```
(1949, 111)
(1950, 22)
```

Tom White, Hadoop: The Definitive Guide, 3rd Edition, 2012

Hive

Hive

- Created at Facebook
- Data Warehouse on the top of Hadoop
  - Map-Reduce for execution
  - HDFS for storage
- HiveQL -SQL like query language
  - Heavily influenced by MySQL
- Storage: flat files (no indexes)

http://hadoop.apache.org/docs/hdfs/r0.22.0/hdfs_design.html

Query language

- DDL
  - {create/alter/drop} {table/view/partition}
  - create table as select
- DML
  - Insert overwrite
- QL
  - Sub-queries in from clause
  - Equi-joins (including Outer joins)
  - Multi-table Insert
  - Sampling
  - Lateral Views
- Interfaces
  - JDBC/ODBC/Thrift

QL


```
SELECT [ALL | DISTINCT]
select_expr, select_expr, ...
FROM table_reference
[WHERE where_condition]
[GROUP BY col_list]
[SORT BY col_list] ]
[LIMIT number]
```

Example

Calculate maximum recorded temperature by year:

- CREATE TABLE records (year STRING, temperature INT)
  ROW FORMAT DELIMITED
  FIELDS TERMINATED BY '\t';
- LOAD DATA LOCAL INPATH 'input/ncdc/micro-tab/sample.txt'
  OVERWRITE INTO TABLE records;
- hive> SELECT year, MAX(temperature)
  > FROM records
  > WHERE temperature != 9999
  > GROUP BY year;

1949 111
1950 22

http://hadoop.apache.org/docs/hdfs/r0.22.0/hdfs_design.html

Example

- The SQL query on the previous slide is nothing special: just a SELECT statement with a GROUP BY clause for grouping rows into years, which uses the MAX() aggregate function to find the maximum temperature for each year group.
- The remarkable thing is that Hive transforms this query into a MapReduce job, which it executes on our behalf, then prints the results to the console.
- There are some nuances such as the SQL constructs that Hive supports and the format of the data that we can query—and we shall explore some of these in this chapter—but it is the ability to execute SQL queries against raw data that gives Hive its power.

http://hadoop.apache.org/docs/hdfs/r0.22.0/hdfs_design.html

Suggested readings (processing)

http://www.dwinfocenter.org/

http://www.vogella.com/articles/ApacheHadoop/article.html

http://hadoop.apache.org/docs/r0.20.2/mapred_tutorial.html

Pig:

http://pig.apache.org/docs/r0.9.1/start.html

Hive:

http://hive.apache.org/

Hadoop lectures (Tom White):

http://www.youtube.com/watch?v=Aq0x2z69syM

http://www.youtube.com/watch?v=2SpTvWiXBcA

LinkedIn lecture Jakob Homan

http://www.youtube.com/watch?v=SS27F-hYWfU