并行处理在大数据分析中所面对的挑战

张晓东

美国俄亥俄州立大学

合作单位

美国俄亥俄州立大学计算机科学与工程系

Facebook 数据管理系统团队

美国埃默里大学医学院 （Emory University）

中科院计算所

Broad Challenges of Big Data Analytics

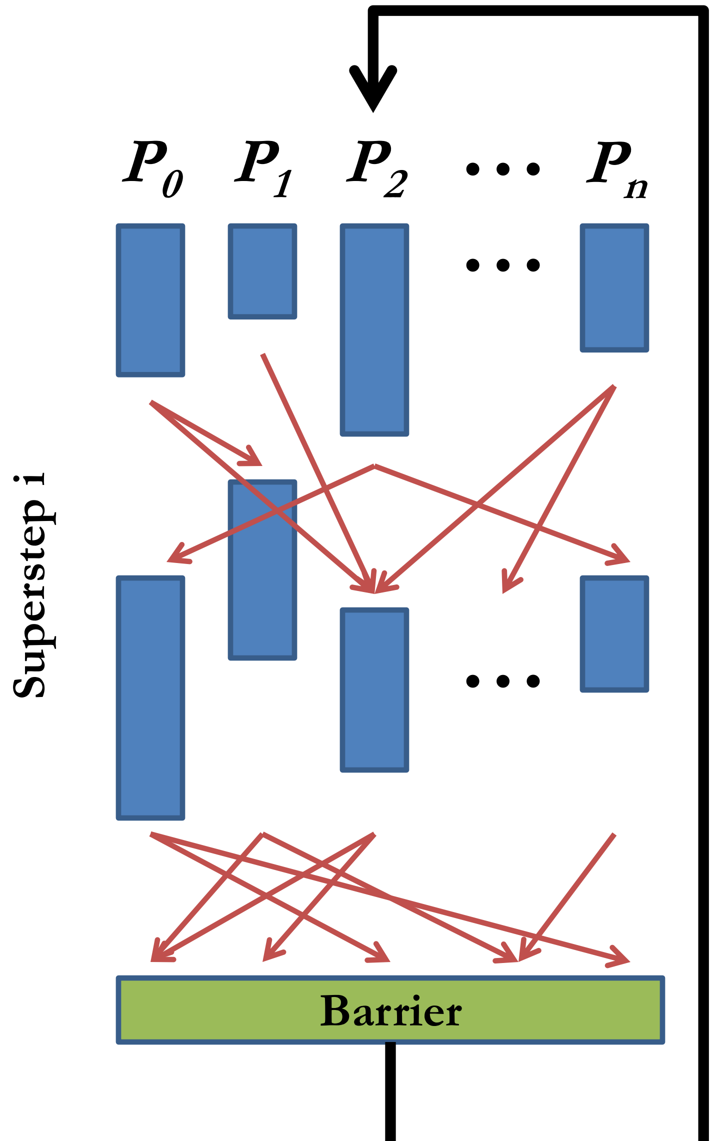
- **Existing DB technology is not prepared for the huge volume**
  - Needs new system infrastructure for structured and non-structured data
  - Must be hardware independent, HP, HT, scalable, and fault-tolerant
- **Truly multidisciplinary collaborations across all the disciplines**
  - Although data volumes in all fields are high, the **formats** of data and **natures of analytics** are very different.
- **Big data analytics must be cost effective to the society**
  - Conventional database model is not affordable
  - Low cost clusters and open source software are foundations
- **Open many new research challenges**
  - In algorithms, systems design/implementations, and in many applications

Computing Paradigm Shift for Big Data Analytics

- **Conventional parallel processing model is "scale-up" based**
  - **BSP model**, CACM, 1990: optimizations in both hardware and software
  - **Hardware**: low ratio of comp/comm, fast locks, large cache and memory
  - **Software:** overlapping comp/comm, exploiting locality, co-scheduling …

- **Big data processing model is "scale-out" based**
  - **DOT model**, SOCC'11: hardware independent software design
  - **Scalability:** maintain a **sustained throughput growth** by continuously adding low cost computing and storage nodes in distributed systems
  - **Constraints** in computing patterns: communication- and data-sharing-free

BSP is a Scale-Up Model for HPC



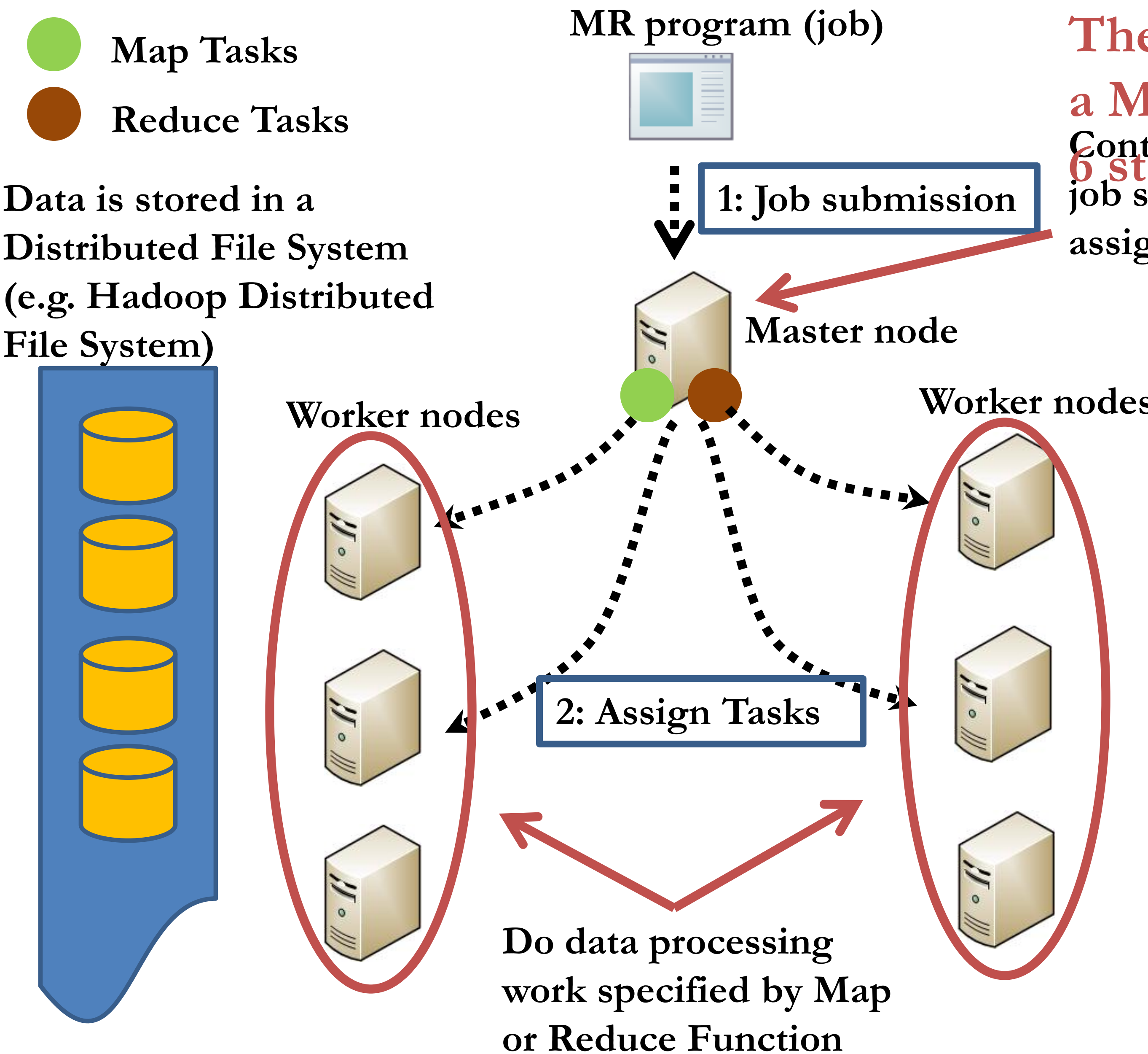
- **A parallel architecture model**
  - **Key parameters:** p, node speed, message speed, synch latency
  - **Low ratio of computing/message** is key
- **A programming model**
  - **Reducing message/synch latency** is key:
    - Overlapping computing and communication
    - Exploiting locality
    - Load balance to minimize synch latency
- **A cost model**
  - Combining both hardware/software parameters, we can predict **execution time**
- **BSP does not support**
  - **Data-intensive applications, big data**
  - **hardware independent performance**
  - **sustained scalability and high throughput**

Major Hurdles of Parallel Processing in Big Data Analytics

- **Scale-out = sustained throughput growth as # nodes grows**
  - MR processed **1 PB** data in 6h 2m on **4000 nodes** in 11/2008
  - MR processed **10 PB** in 6 h 27 m on **8000 nodes** 9/2011
  - Big data with **simple analytics**
- **Data processing must be highly parallel with several constraints**
  - No specific communication hardware support
  - Lacking runtime and global coordination in clusters
  - Lacking software tools to optimize and standardize analytics programs
  - Data sets are hardly or even not movable after they are stored
- **MapReduce (Hadoop) is the basic big data processing engine**
  - High scalability (minimum dependency)
  - High fault tolerance (simple and independent operations in each node)
  - We must overcome hurdles to process **big data with complex analytics**

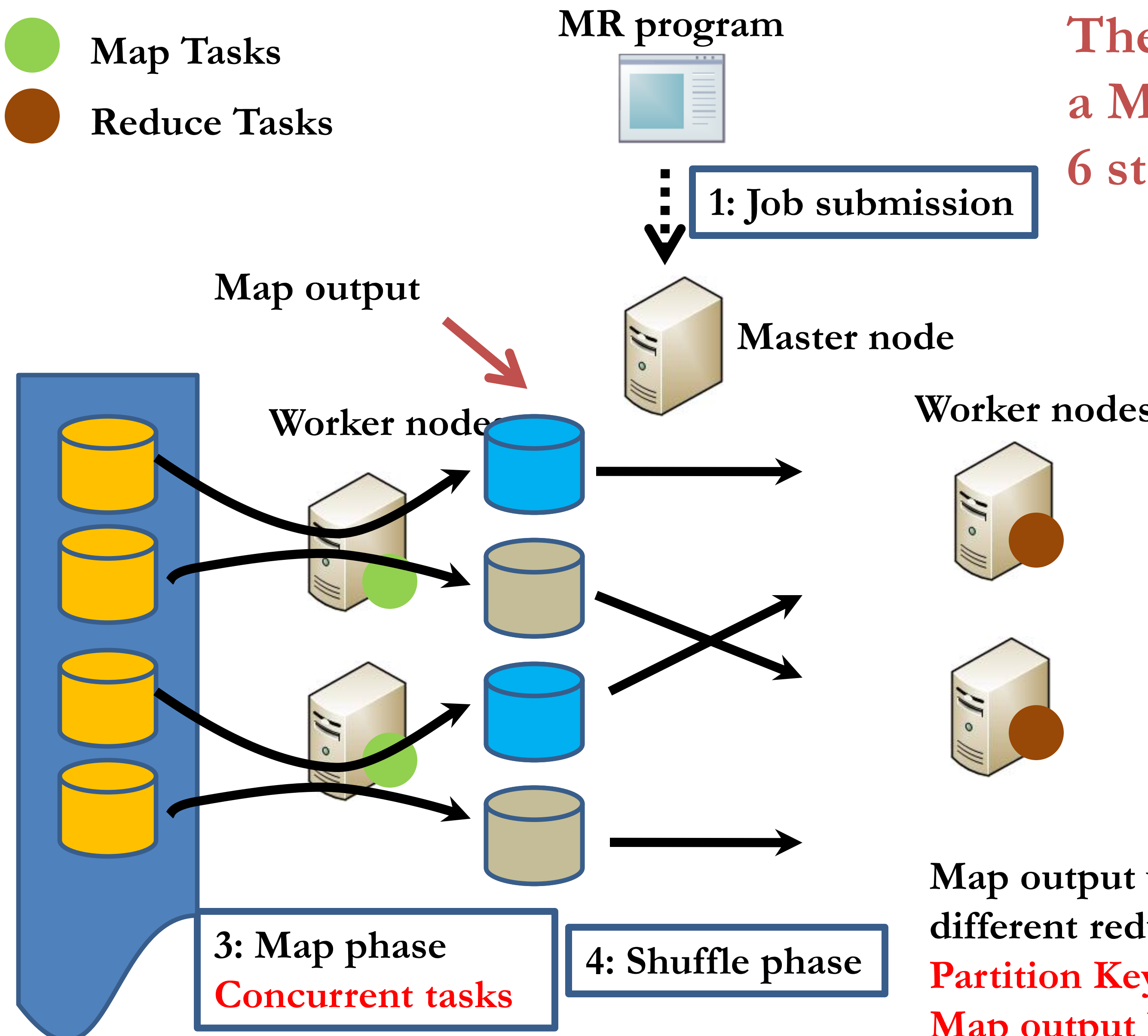
MR(Hadoop) Job Execution Patterns

The execution of a MR job involves 6 steps



MR(Hadoop) Job Execution Patterns

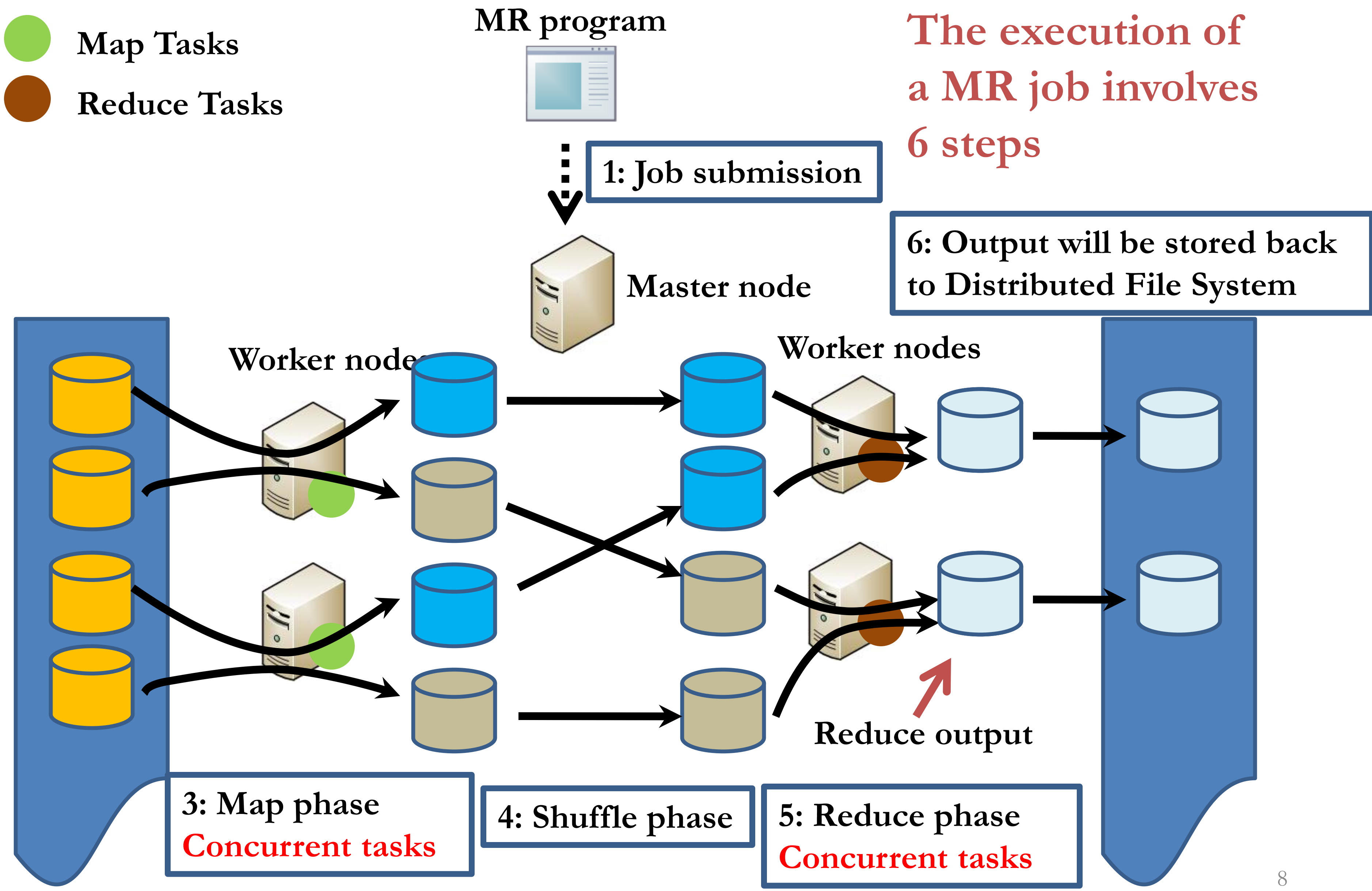
The execution of a MR job involves 6 steps



Map output will be shuffled to different reduce tasks based on **Partition Keys (PKs) (usually Map output keys)**

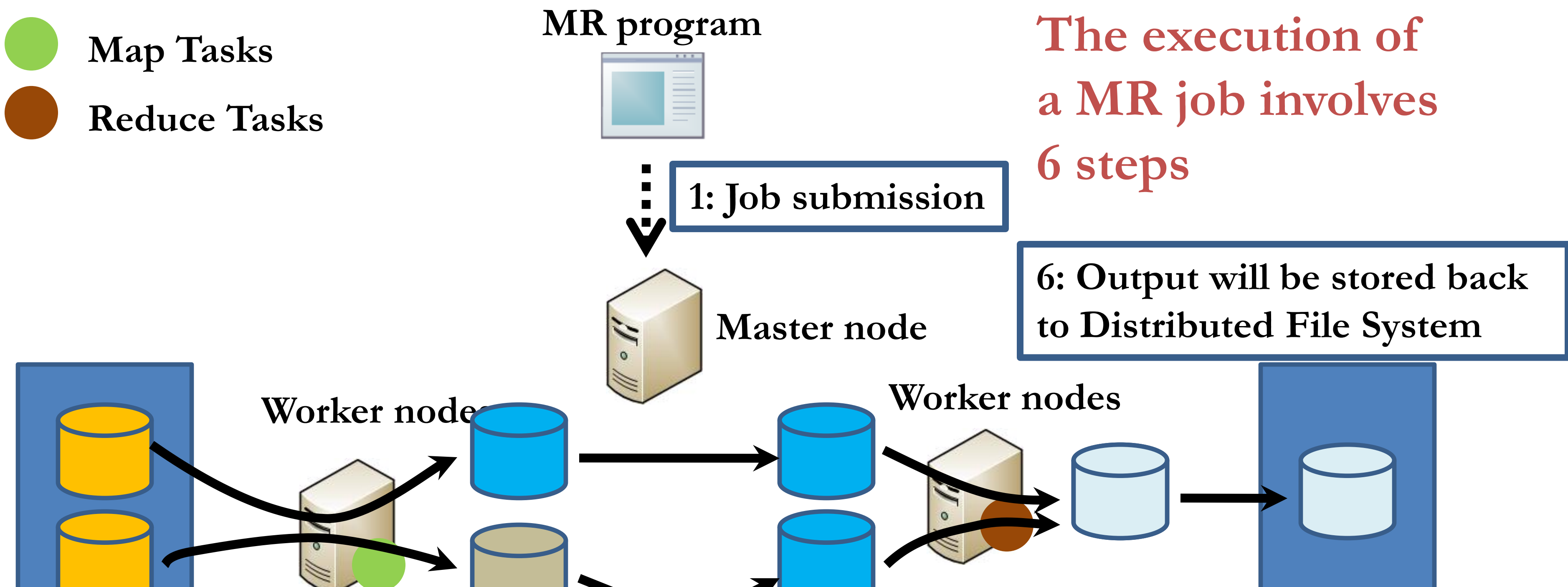
MR(Hadoop) Job Execution Patterns

The execution of a MR job involves 6 steps



MR(Hadoop) Job Execution Patterns

The execution of a MR job involves 6 steps



MapReduce model is synchronous, net/disk demanding:
1: Input data scan in the Map phase => local or remote I/Os
2: Store intermediate results of Map output => local I/Os
3: Transfer data across in the Shuffle phase => network costs
4: Store final results of this MR job => local I/Os + network costs (replicate data)

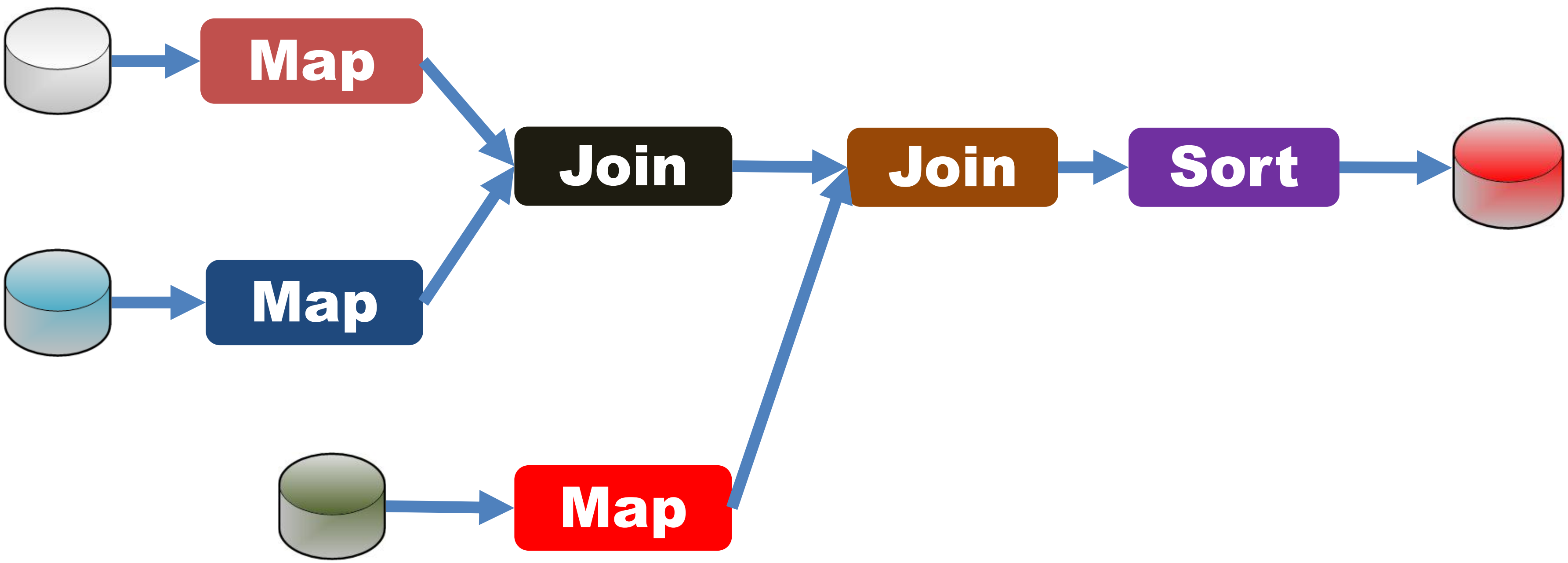
Sources of Bottlenecks

- **Storage latency:** unnecessary data transfers between local hard disks and nodes.
- **Network latency:** unnecessary data communication among nodes
- **Recovery time for fault tolerance:** the latency to resume processing after node(s) are crashed
- **Processing engine modification:** the application scope of a customized and application dependent Hadoop is very limited
- **"One size fits all" methodology:** enhancing functionalities under one software framework
- **Processing engine structure unawareness:** software tools that are not designed in a target way may not be efficient

Outline

- **SideWalk: a messaging facility for big data analytics**
  - A communication facility for critically necessary message exchange
  - A light-weight message sharing facility without affecting scalability
  - A user communication vehicle with restrictions
- **YSmart: a highly efficient query-to-MapReduce translator**
  - Correlations-aware is the key
  - Fundamental Rules in the translation process
  - A part of production systems and a MapReduce teaching tool
- **Data Placement: related optimization and analysis**
  - Formal and problem definitions
  - Placement analysis under a unified evaluation framework
  - Why RCFile is the most balanced structure and widely used?
- **Conclusion**

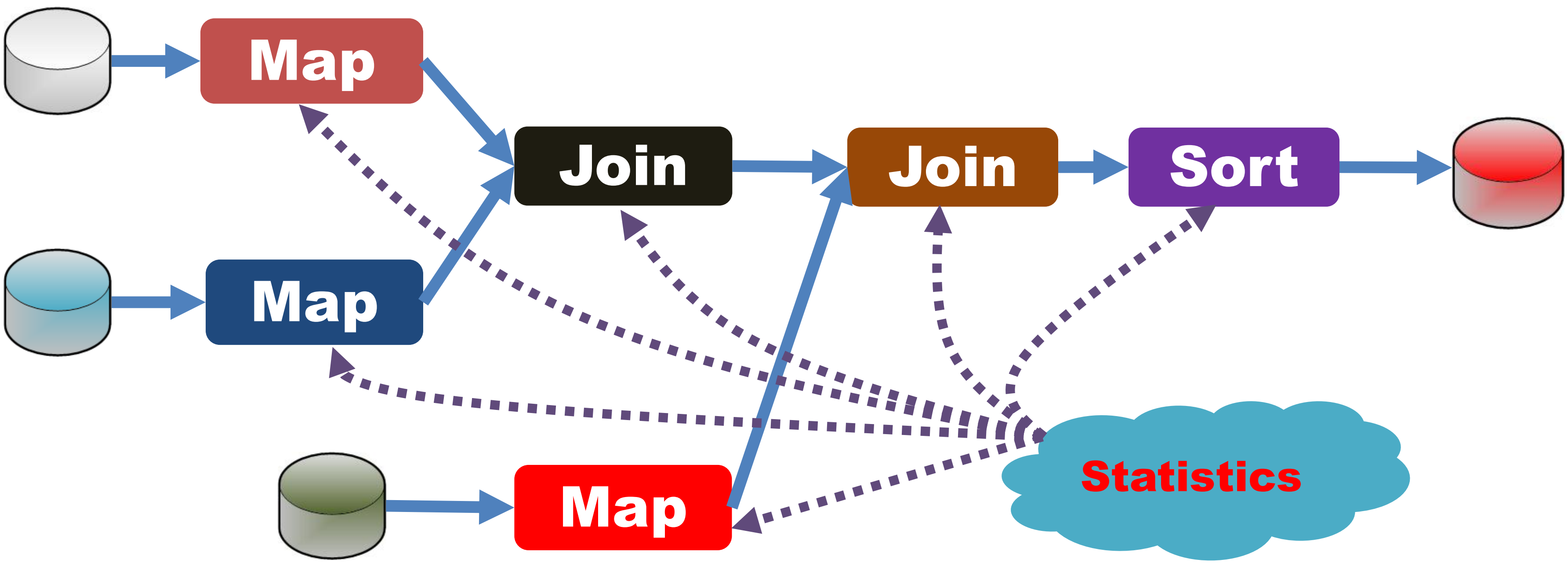
A Typical Dataflow in MapReduce



Communications for data transfers?

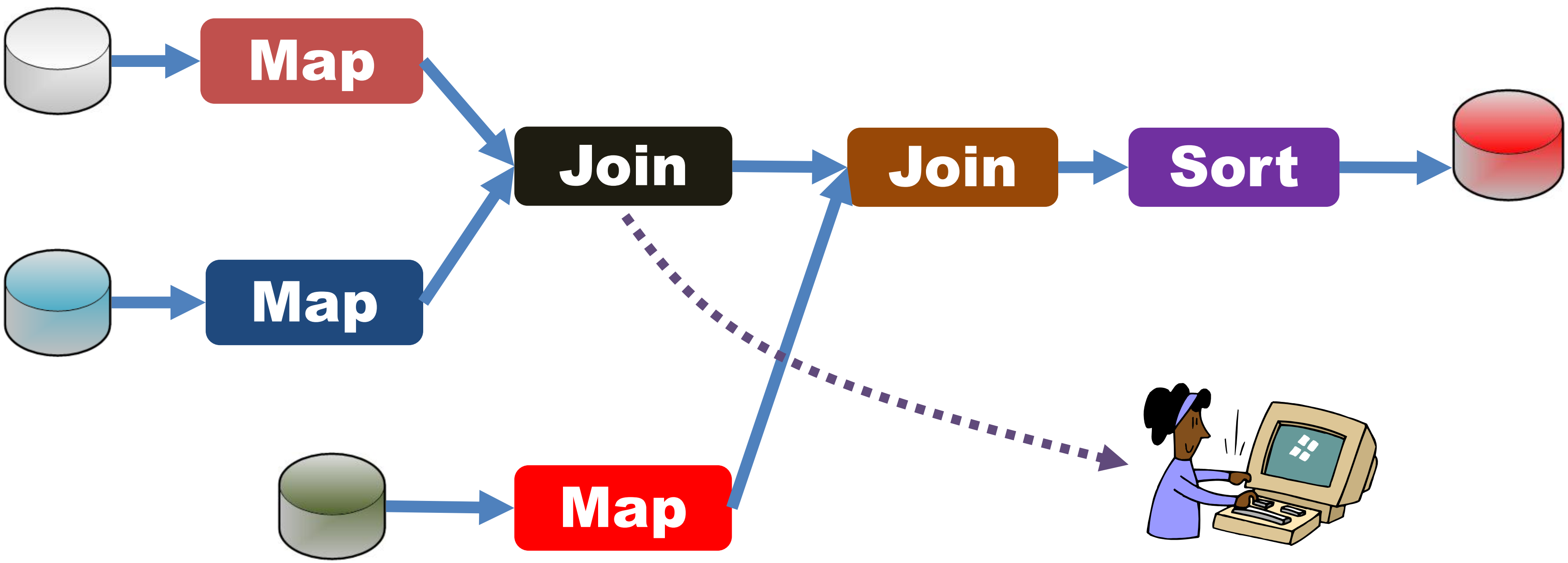
Communications are strictly defined by the framework

Demands beyond the Dataflow



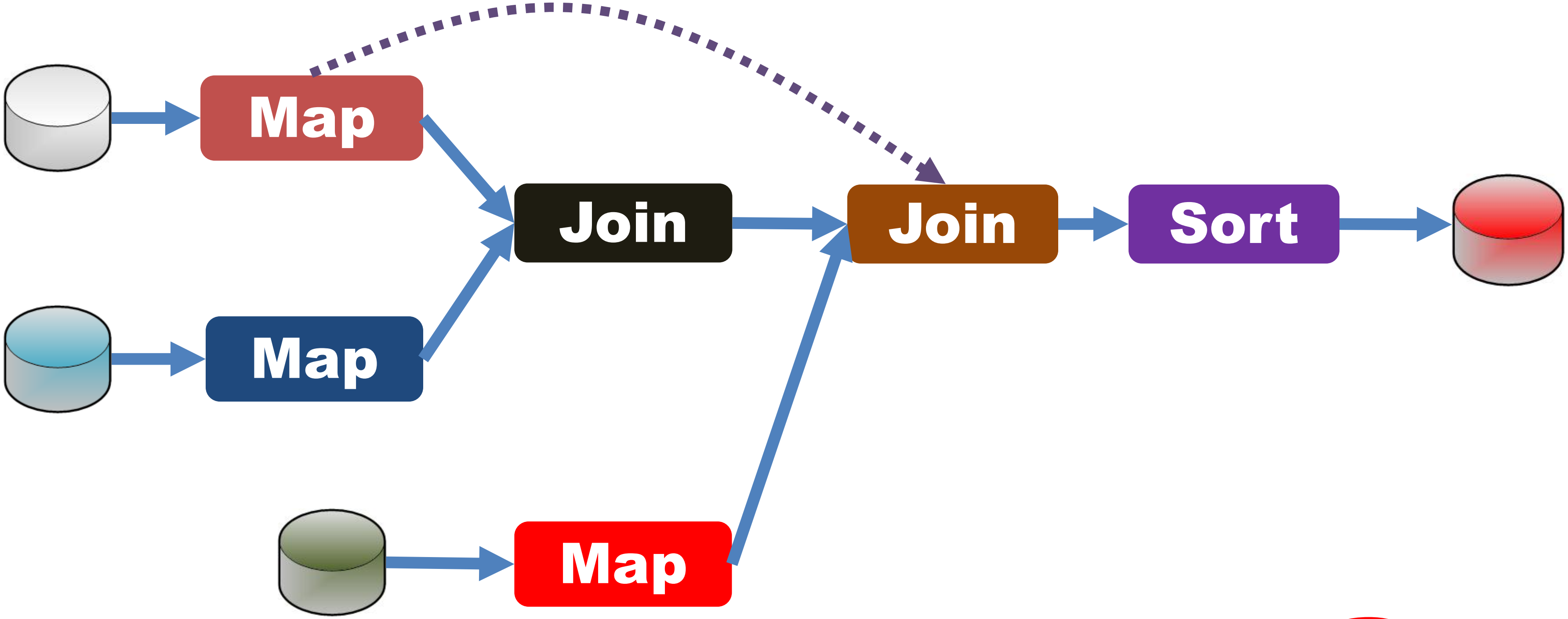
We want to optimize this execution with statistics collected and summarized from previous executions

Demands beyond Dataflow



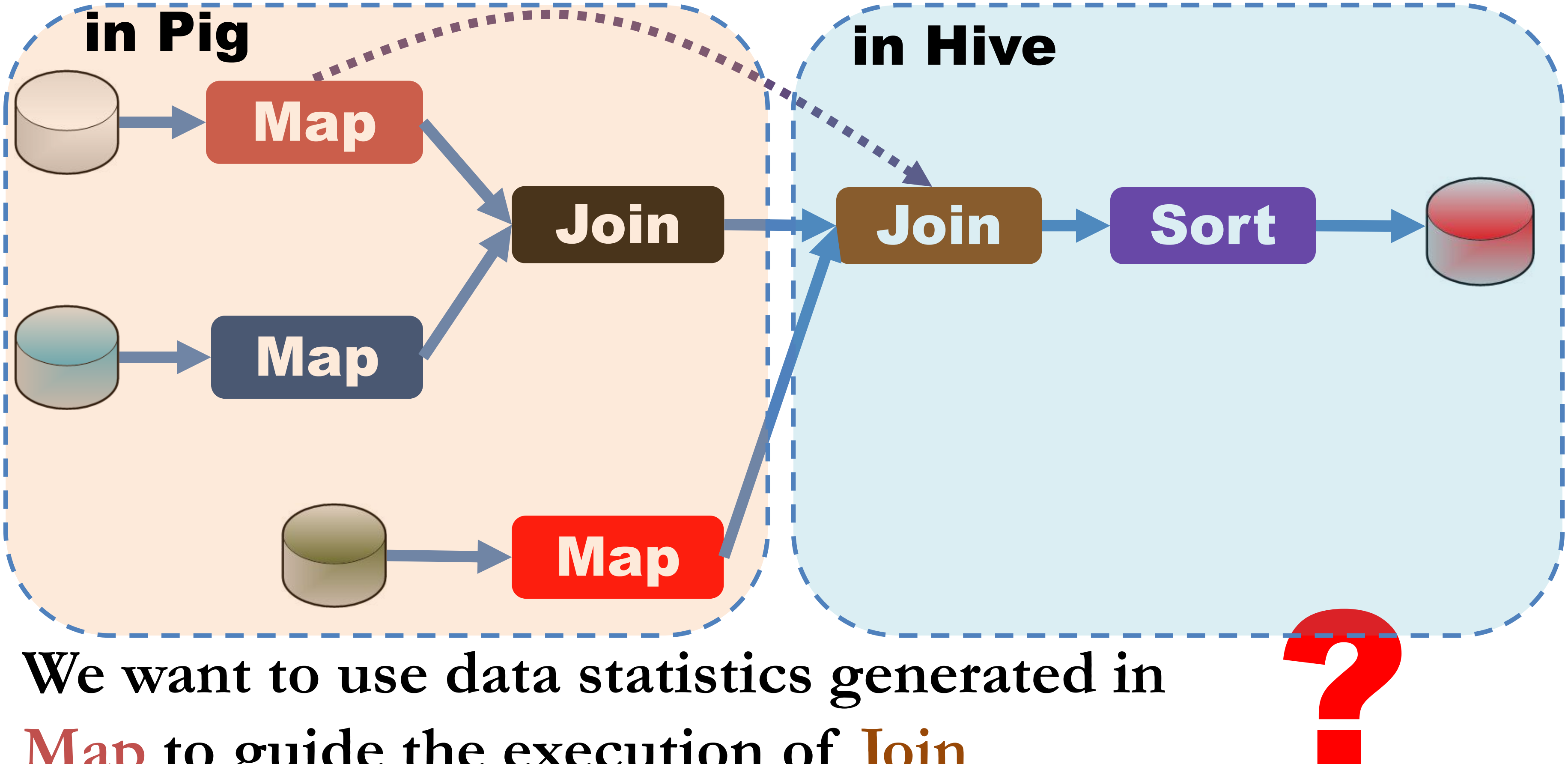
We want to gather ad hoc information from the inside of dataflow to know if the execution runs normally

Demands beyond Dataflow



We want to use data statistics generated in Map to guide the execution of Join

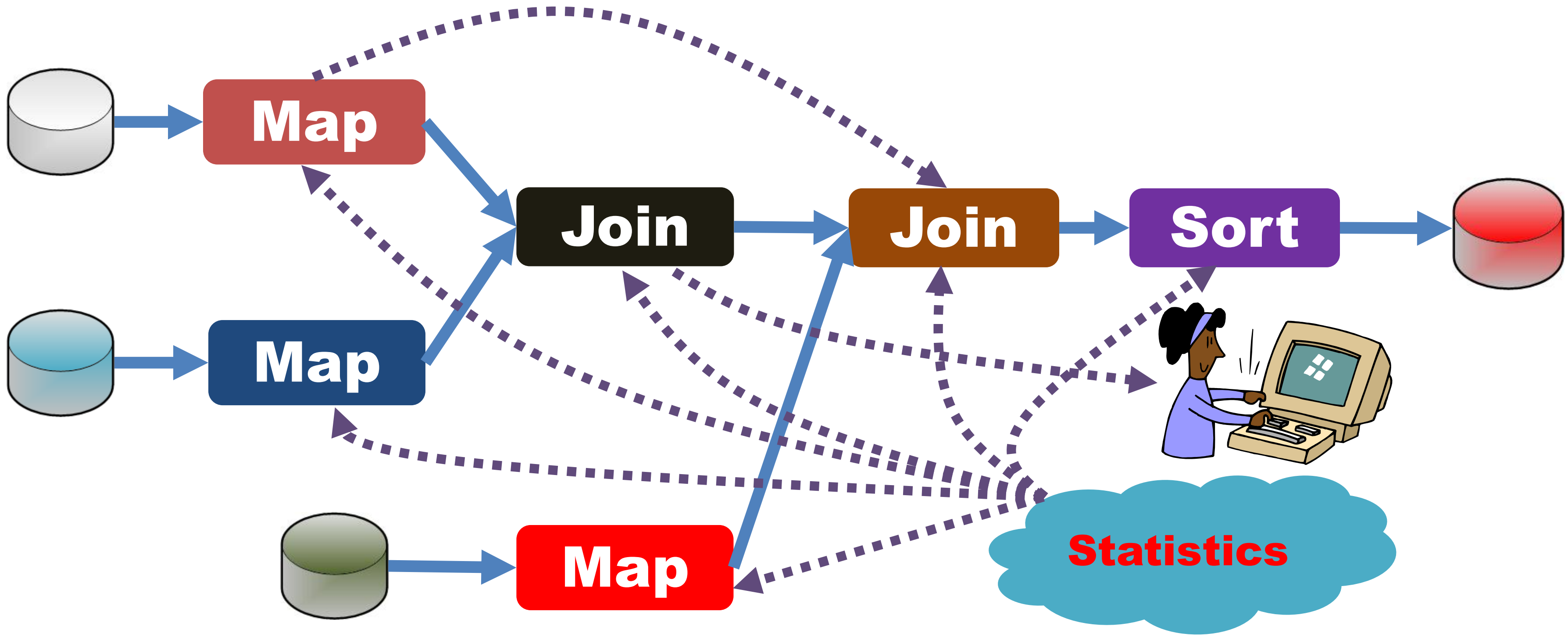
Demands beyond Dataflow



We want to use data statistics generated in Map to guide the execution of Join

If Map and Join are executed in different applications or systems?

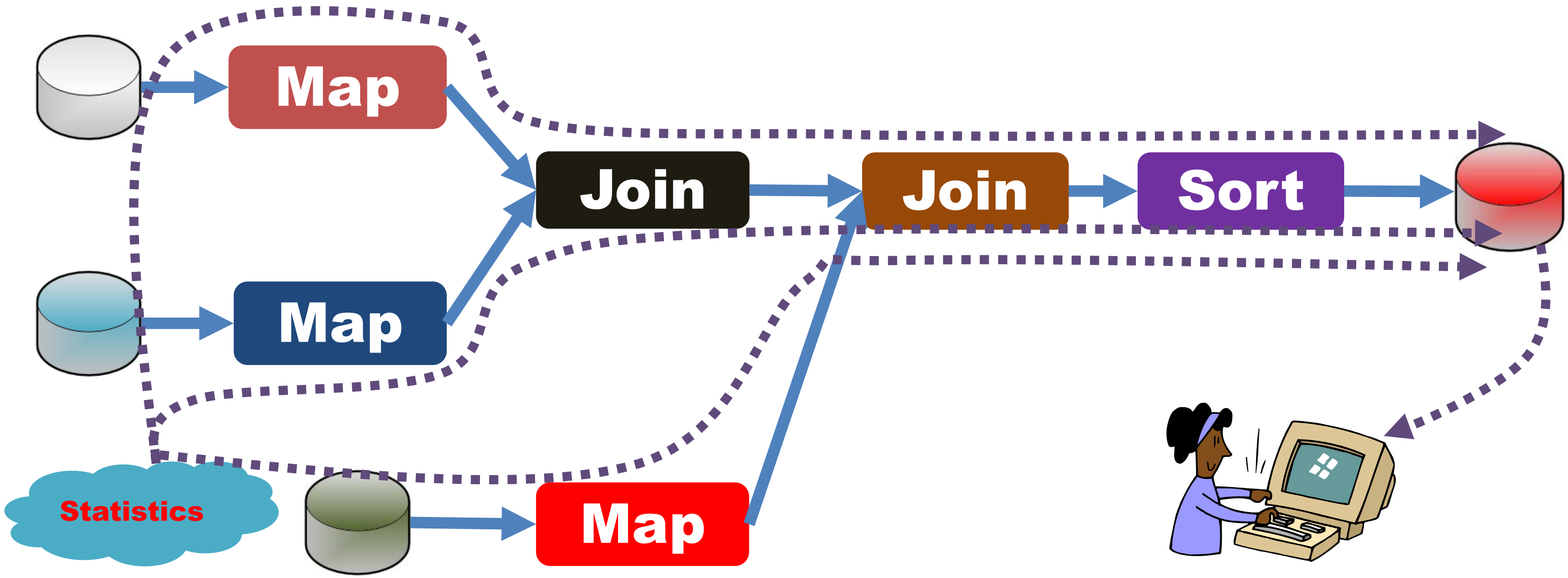
Demands beyond Dataflow



- Those communications are *out-of-band* of communications for data transfers
- Those communications are carrying light-weight information
- Existing systems, such as Hadoop, are not designed to support out-of-band communications

A Straight Forward Way

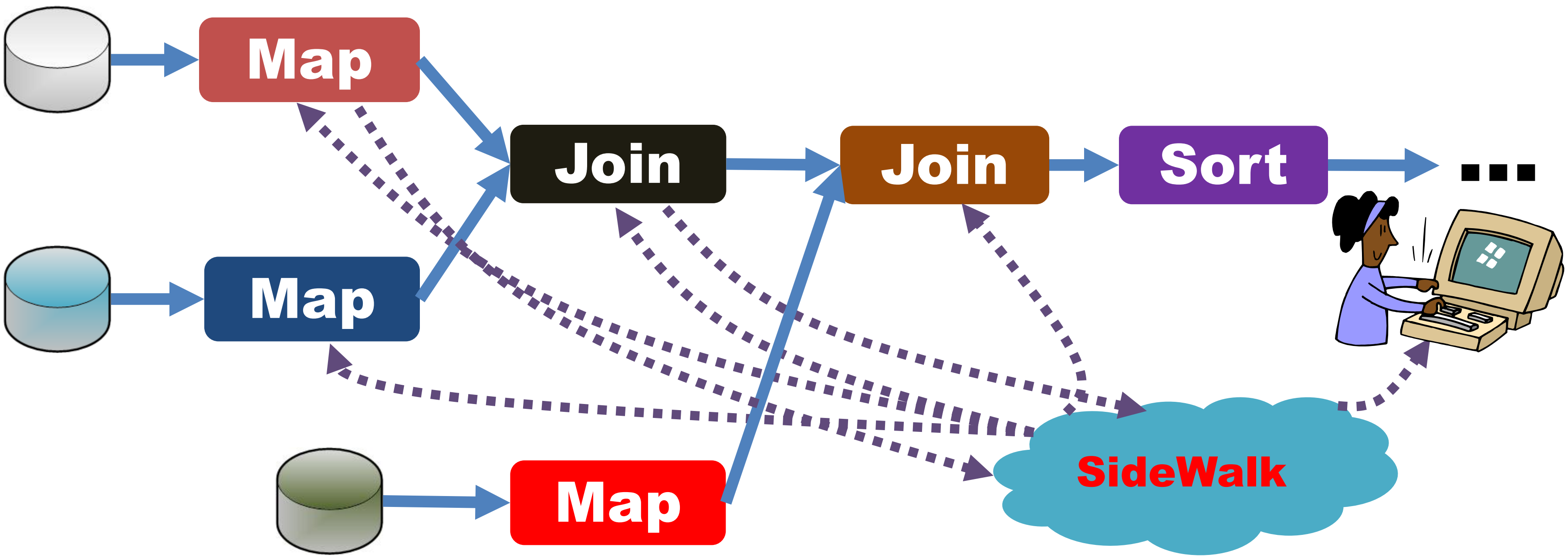
Fit out-of-band communications into defined data transfers



- High overhead
- Low programming productivity
- Hard to make this method general-purpose

Where is SideWalk in the Big Data Ecosystem?

- An General-purpose Out-of-band Communication Facility in Large-scale Data Processing



Principles and Functions of SideWalk

- **Out-of-band communications are carried by Auxiliary Datum**
- **Auxiliary Datum is generated by a transformation function**
- **Users only focus on defining transformation functions**
  - Built-in functions can be deployed with SideWalk
- **A small set of standardized APIs are designed to make SideWalk support different applications and systems**
- **Abusing the capability of SideWalk is not allowed**
  - SideWalk does not handle regular data processing tasks
  - SideWalk is not a relay for arbitrary communications

Options of Making Out-of-Bound Communications

Out-of-band communications require special treatment

Goals	Ad hoc solution (User-based)	Specific software (App-based)	SideWalk
Programming productivity	Low	High	High
Applicability	Limited (Not general-purpose)		Maximal (General-purpose)
Side-effects	Potentially high		Minimal

Outline

- SideWalk: a messaging facility for big data analytics
  - A communication facility for critically necessary message exchange
  - A light-weight message sharing facility without affecting scalability
  - A user vehicle for communications with restrictions
- **YSmart: a highly efficient query-to-MapReduce translator**
  - Correlations-aware is the key
  - Fundamental Rules in the translation process
  - A part of production systems and MapReduce teaching tool
- Data Placement: related optimization and analysis
  - Formal and problem definitions
  - Placement analysis under a unified evaluation framework
  - Why RCFile is the most balanced structure and widely used?
- Conclusion

MR programming is not that “simple”!

```
package tpch;
import java.io.IOException;
import java.util.ArrayList;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
```

```
public static class Reduce extends Reducer<IntWritable,Text,IntWritable,Text> {
        private Text result = new Text();

        public void reduce(IntWritable key, Iterable<Text> values,
                                        Context context
                        ) throws IOException, InterruptedException {

                                double sumQuantity = 0.0;
                                IntWritable newKey = new IntWritable();
```

This complex code is for a simple MR job

```
import org.apache.hadoop.util.GenericOptionsParser;
import org.apache.hadoop.util.Tool;
```

```
                                        }
                                        else
```

Low Productivity!

```
                if (inputFile.compareTo("lineitem.tbl") == 0){
                        isLineitem = true;
```

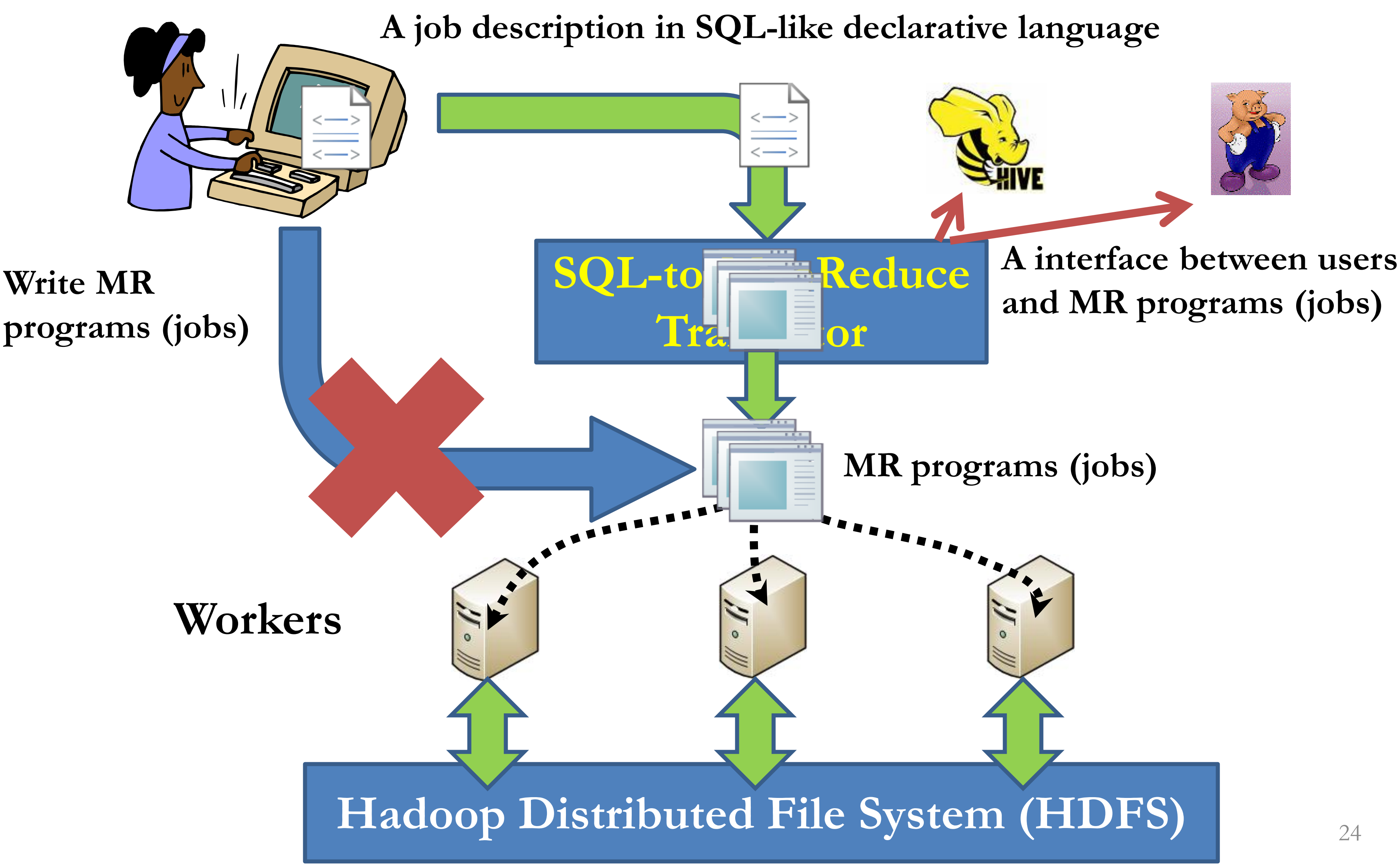
```
                                        context.write(newKey, result);
                                }
                        }
```

We all want to simply write:

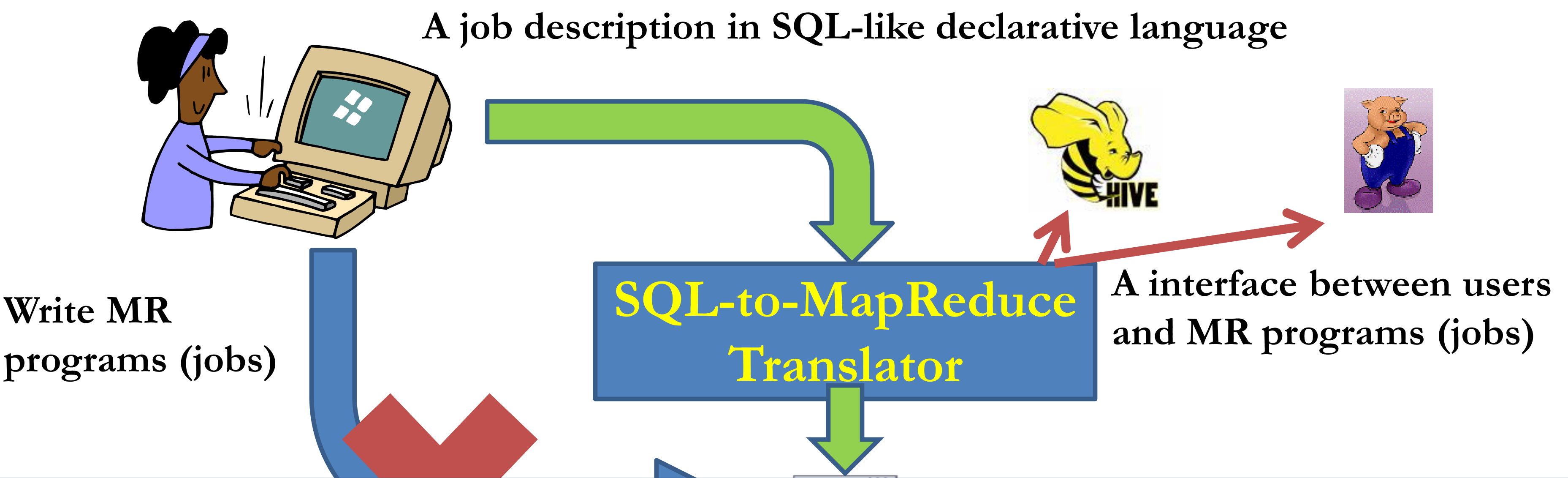
“SELECT * FROM Book WHERE price > 100.00”?

```
                int res = ToolRunner.run(new Configuration(), new Q18Job1(), args);
                System.exit(res);
        }
}
```

High Quality MapReduce in Automation



High Quality MapReduce in Automation



A data warehousing system (Facebook)

A high-level programming environment (Yahoo!)

Improve productivity from hand-coding MapReduce programs

- 95%+ Hadoop jobs in Facebook are generated by **Hive** *
- 75%+ Hadoop jobs in Yahoo! are invoked by **Pig** * *

* http://www.borthakur.com/ftp/hadoopworld.pdf

** http://hadooplondon.eventbrite.com/

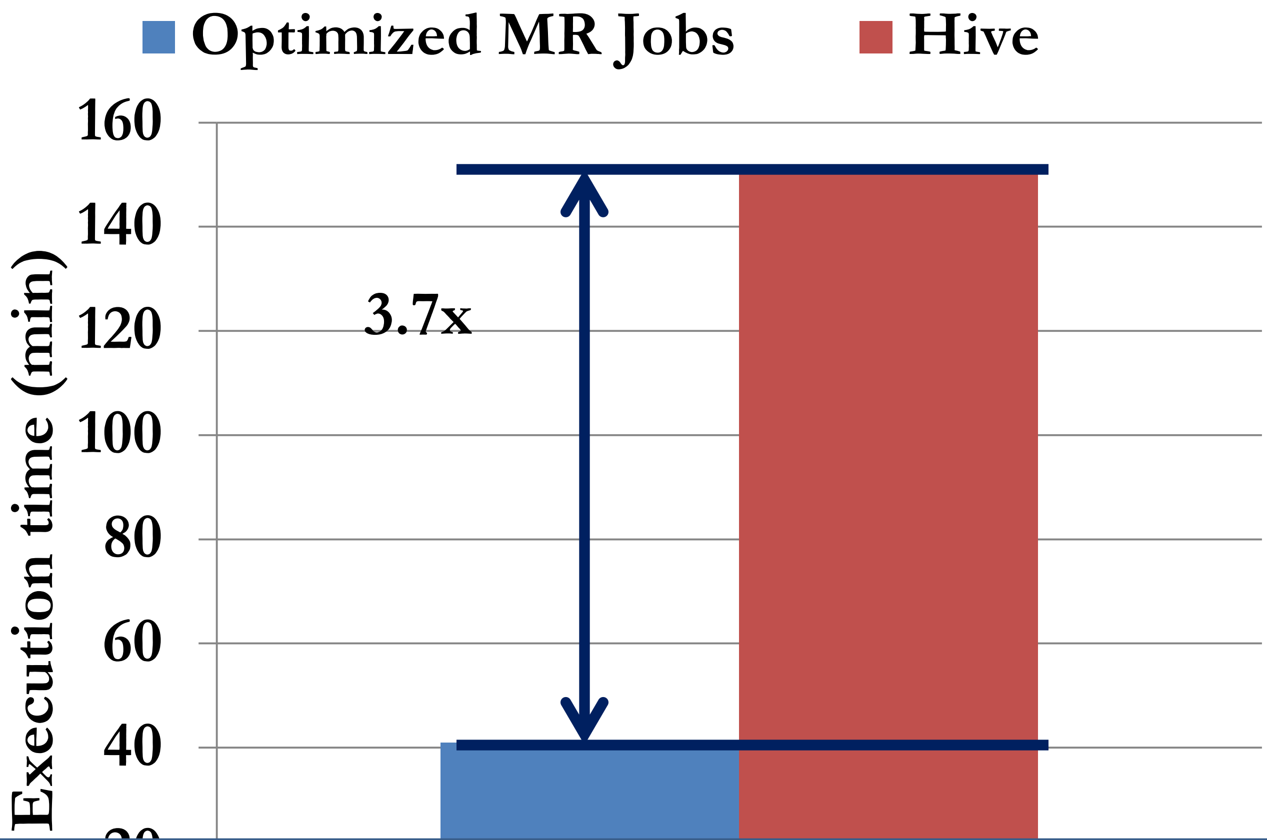
Translating SQL-like Queries to MapReduce Jobs: Existing Approach

- **"Sentence by sentence" translation**
  - [C. Olston et al. SIGMOD 2008], [A. Gates et al., VLDB 2009] and [A. Thusoo et al., ICDE2010]
  - Implementation: Hive and Pig
- **Three steps**
  - **Identify major sentences with operations that shuffle the data**
    - Such as: Join, Group by and Order by
  - **For every operation in the major sentence that shuffles the data, a corresponding MR job is generated**
    - e.g. a join op. => a join MR job
  - **Add other operations, such as selection and projection, into corresponding MR jobs**

Existing SQL-to-MapReduce translators give unacceptable performance.

An Example: TPC-H Q21

- ❑ One of the most complex and time-consuming queries in the TPC-H benchmark for data warehousing performance
- ❑ Optimized MR Jobs vs. Hive in a Facebook production cluster

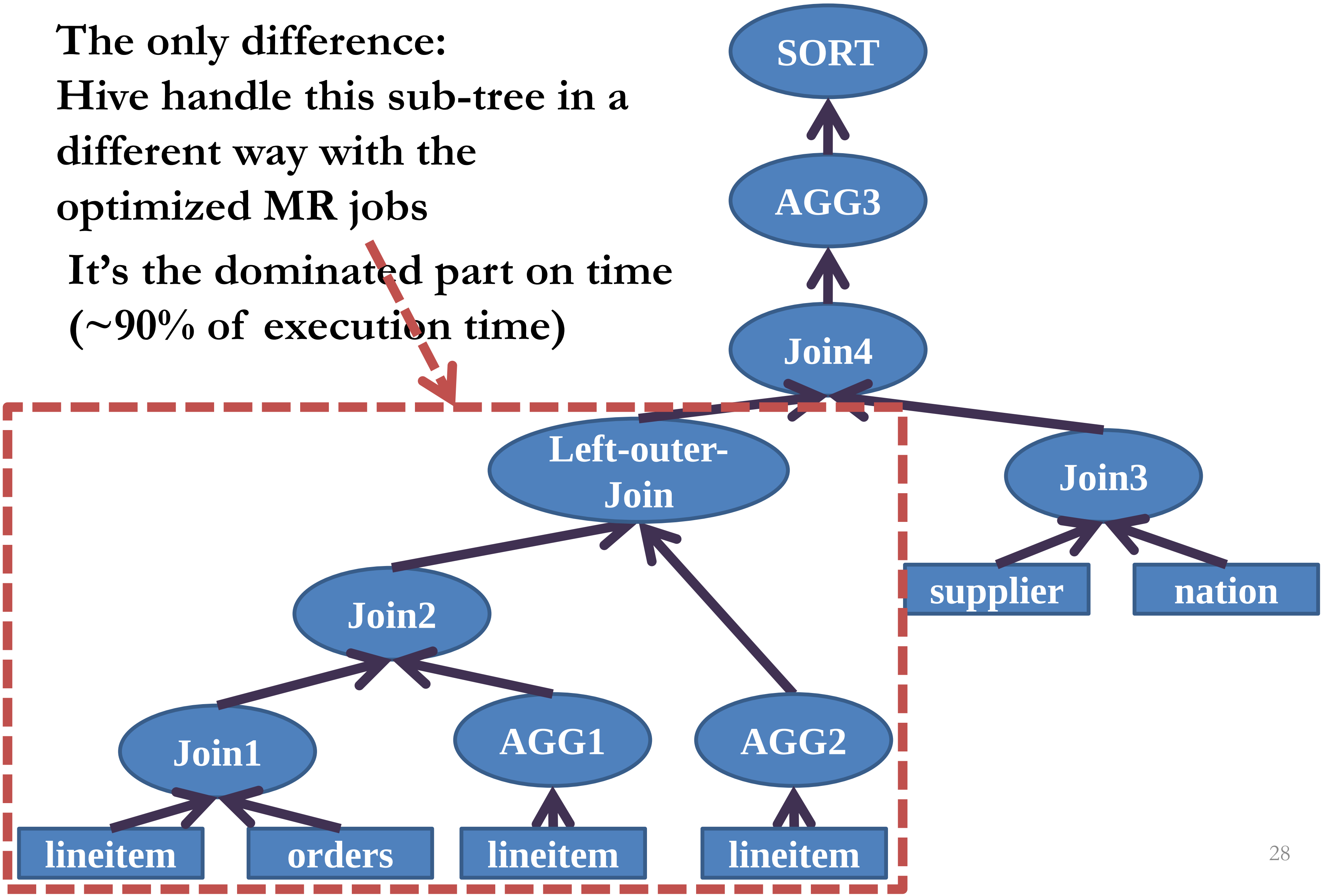


What's wrong?

The Execution Plan of TPC-H Q21

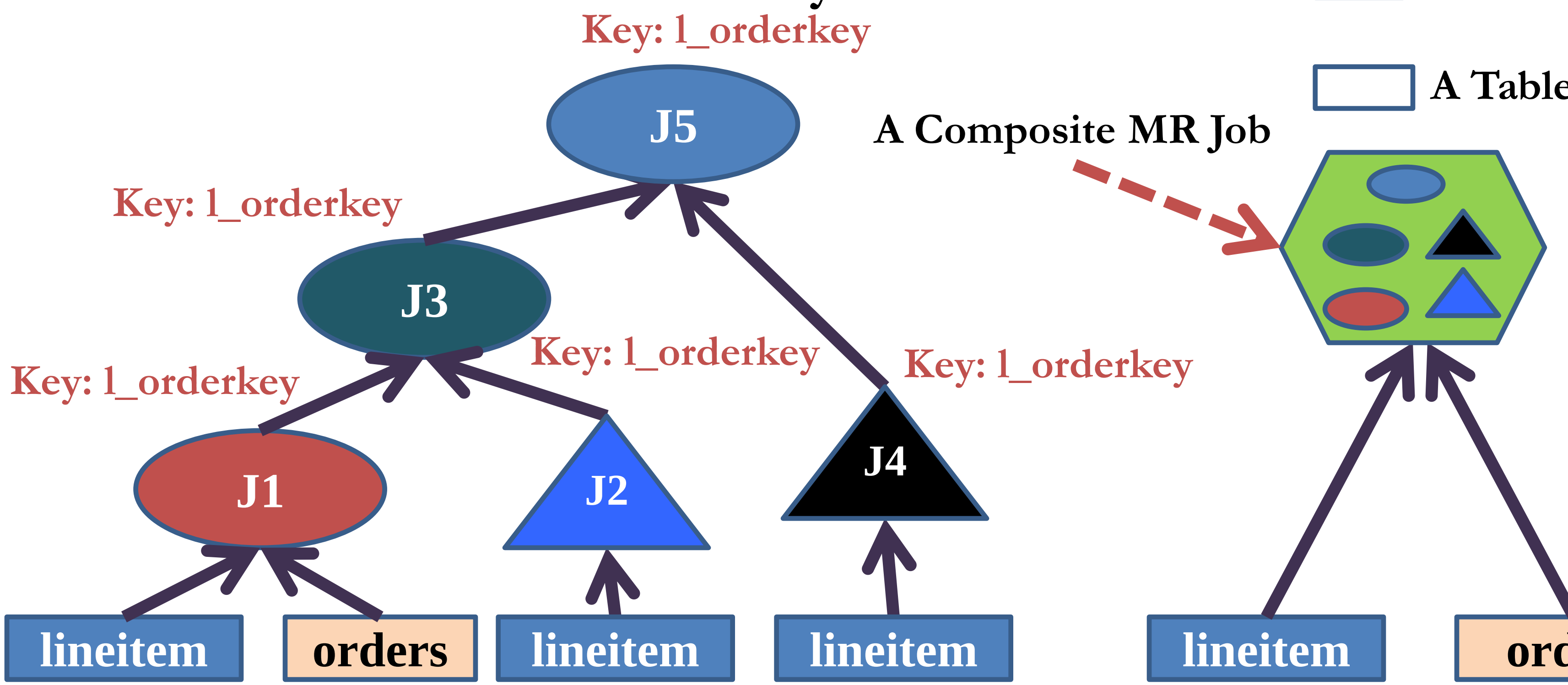
The only difference:
Hive handle this sub-tree in a different way with the optimized MR jobs

It's the dominated part on time (~90% of execution time)



However, inter-job correlations exist.

Let’s look at the Partition Key



J1 to J5 all use the same partition key ‘l_orderkey’

What’s wrong with existing SQL-to-MR translators?

Existing translators are correlation-unaware

1. Ignore common data input
2. Ignore common data transition

Approaches of Big Data Analytics in MR: The landscape

Performance

Hand-coding MR jobs

Correlation-aware SQL-to-MR translator

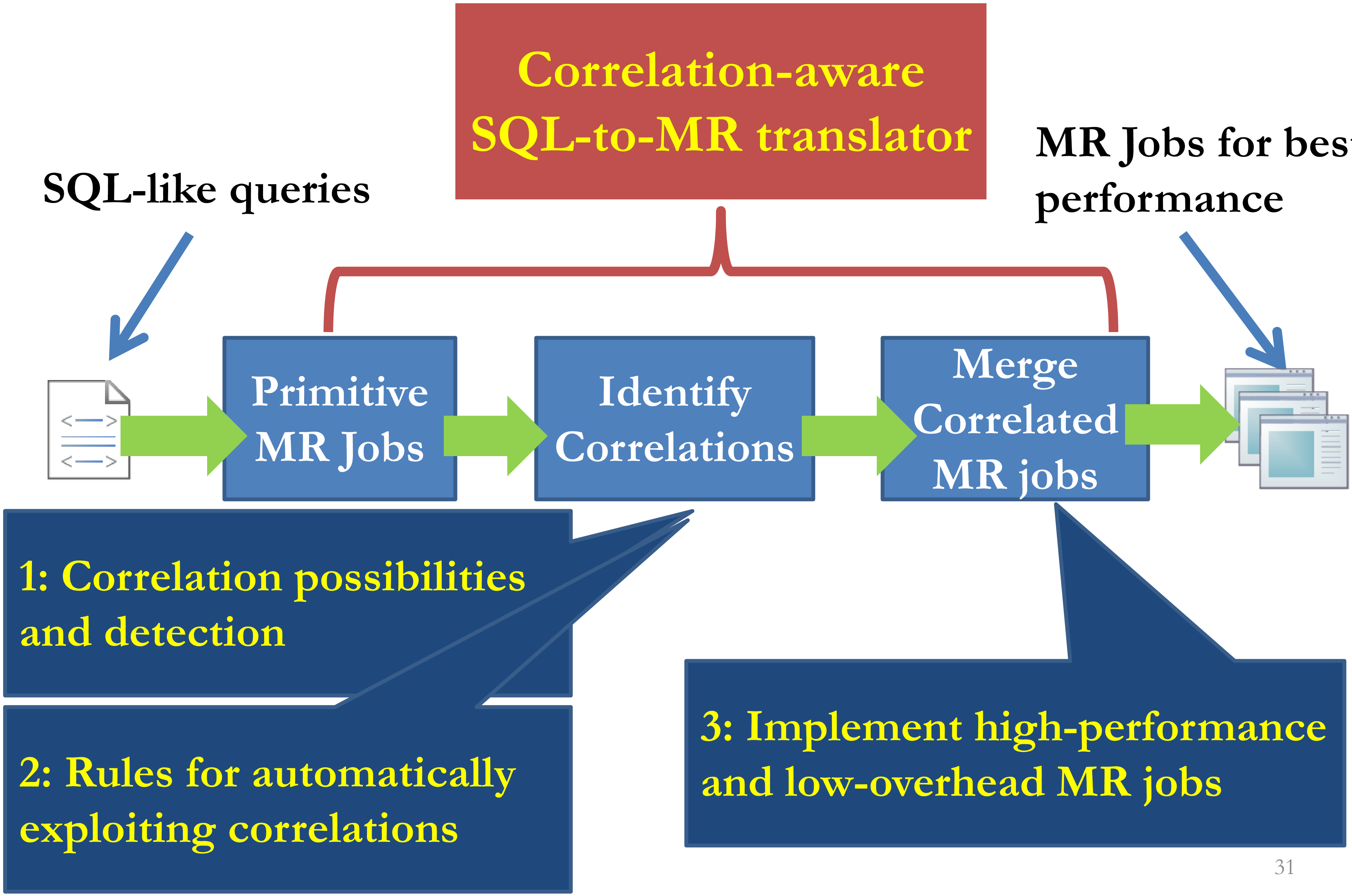
Pro:
high performance MR programs
Con:
1: lots of coding even for a simple job
2: Redundant coding is inevitable
3: Hard to debug
[J. Tan et al., ICDCS 2010]

Pro:
Easy programming, high productivity
Con:
Poor performance on complex queries
(complex queries are usual in daily operations)

Existing SQL-to-MR Translators

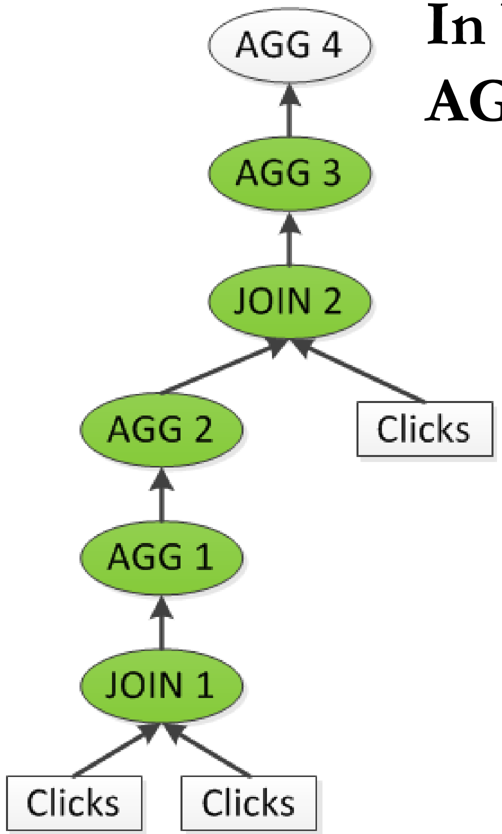
Productivity

Our Approaches and Critical Challenges

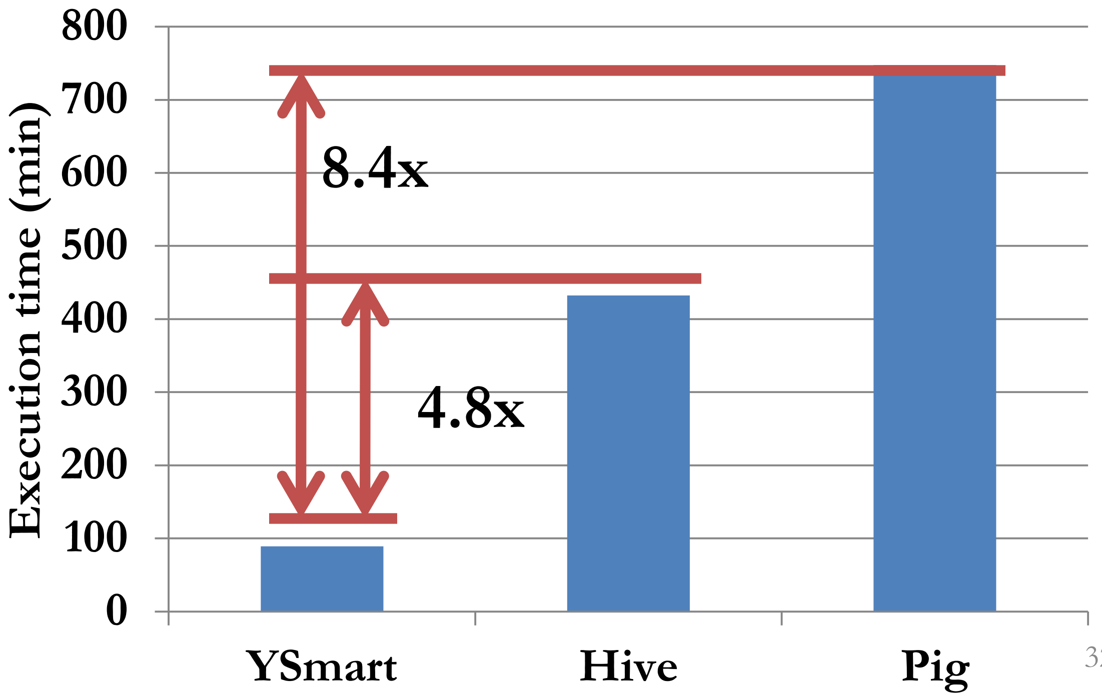


Click-stream Analysis

A typical query in production clickstream analysis: *"what is the average number of pages a user visits between a page in category '**X**' and a page in category '**Y**'?"*



In YSmart JOIN1, AGG1, AGG2, JOIN2 and AGG3 are executed in a single MR job



YSmart is Open Source Software Being Used World Wide
http://ysmart.cse.ohio-state.edu

THE OHIO STATE UNIVERSITY

YSmart

An SQL-to-MapReduce Translator

- Overview
- News
- Download
- Online Version
- Get Started
- Performance
- Publications
- Team

Overview

YSmart is a correlation aware SQL-to-MapReduce translator, which is built on top of the widely used Hadoop platform. Given an SQL query and table schemas, YSmart can automatically translate the query into a series of Hadoop MapReduce programs written in Java. Compared to other SQL-to-MapReduce translators, YSmart has the following advantages:

- **High Performance**. The MapReduce programs generated by YSmart have a very good performance. YSmart can automatically detect and utilize intra-query correlations when translating a query. This correlation-aware ability significantly reduces redundant computation, unnecessary disk IO operations and network overhead. See the Performance page to learn the performance benefits of YSmart.
- **High Extensibility**. YSmart is easy to modify and extend. It is designed with the goal of extensibility. The major part of YSmart is implemented in Python which makes the codes much easier to understand. Due to its modularity and script nature, users can easily modify the current functionalities or add new functionalities to YSmart.
- **High Flexibility**. YSmart can run in two different modes: translation-mode and execution-mode. In the translation-mode, YSmart only translates the query into Java codes while in the execution-mode YSmart will also compile and execute the generated codes. Because of this flexibility, users can easily read, modify and customize the generated codes.

YSmart is an independent open source project with the Apache 2.0 license. It is implemented as a teaching and learning tool for executing queries on top of Hadoop. Currently YSmart only supports a subset features of select queries in SQL. It is still under continuous development. If you have any question or suggestion, please email the authors at yuanyu@cse.ohio-state.edu.

YSmart has been patched in Hive data warehousing systems and will be merged into Hive soon.

News

- Jan. 1, 2012 YSmart Release 12.01 available

Download

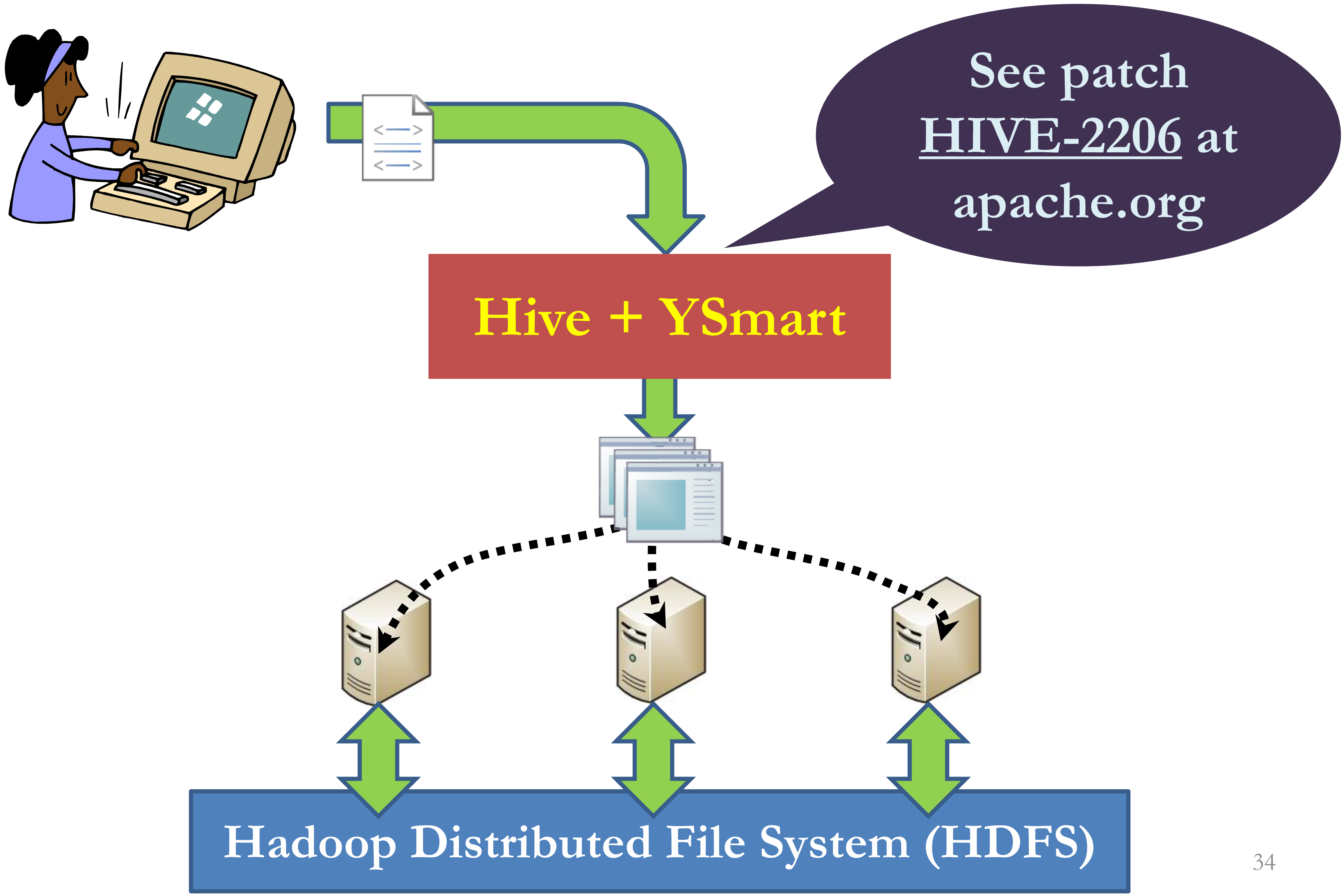
- Source code (Linux 32bit or 64bit): ysmart-12.01.tar.gz

Get Started

To start using YSmart, you can try the YSmart Online Version or download and install YSmart on your own computer.

YSmart is very easy to install and configure. You can install the YSmart if you have a Linux system with GCC, Python and Java 1.6 installed.

YSmart in the Hadoop Ecosystem
(in the final stage of merging into Hive)



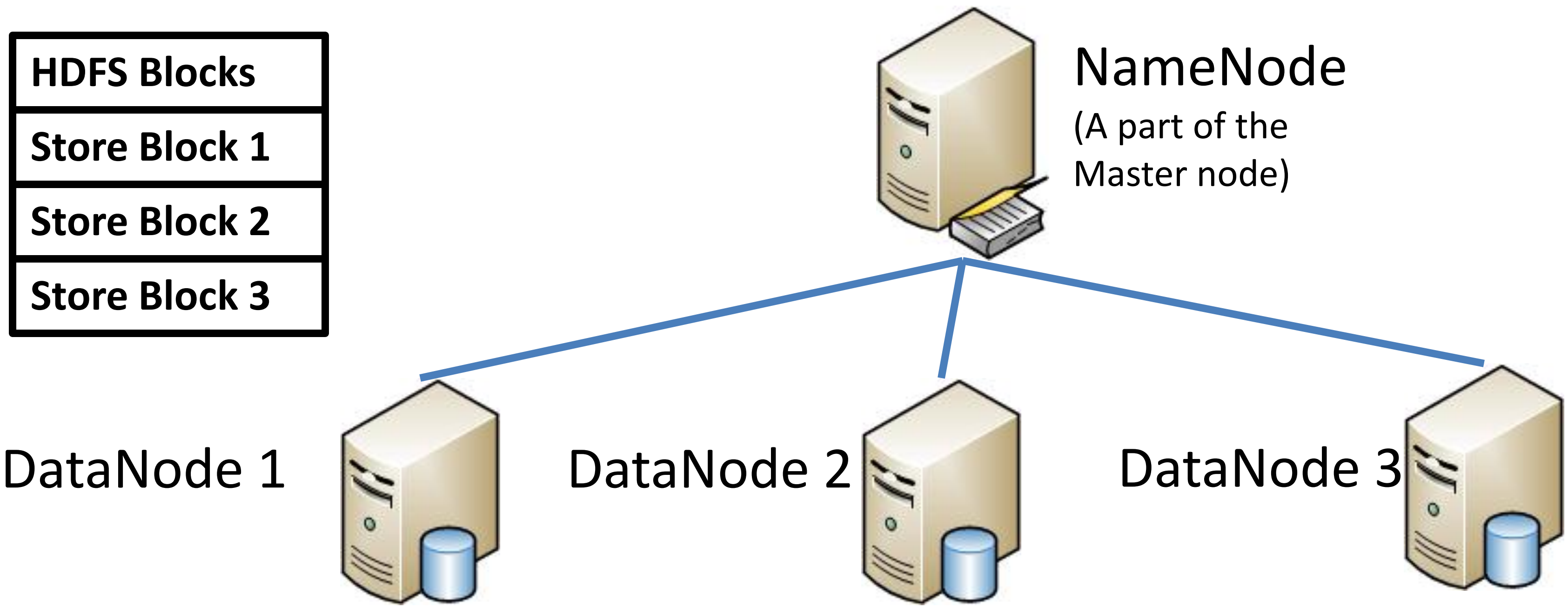
Summary of YSmary

- YSmart is a correlation-aware SQL-to-MapReduce translator
- Its efficient software structure is MapReduce based
- YSmart can outperform Hive by 4.8x, and Pig by 8.4x
- YSmart is in the final stage to be integrated into Hive
- The independent version of YSmart was released in January 2012
    - http://ysmart.cse.ohio-state.edu/

Outline

- SideWalk: a messaging facility for big data analytics
  - A communication facility for critically necessary message exchange
  - A light-weight message sharing facility without affecting scalability
  - A user vehicle for communications with restrictions
- YSmart: a highly efficient query-to-MapReduce translator
  - Correlations-aware is the key
  - Fundamental Rules in the translation process
  - A part of production systems and MapReduce teaching tool
- **Data Placement: related optimization and analysis**
  - Formal and problem definitions
  - Placement analysis under a unified evaluation framework
  - Why RCFile is the most balanced structure and widely used?
- Conclusion

Initial Stores of Big Data in Distributed Environment



- HDFS (Hadoop Distributed File System) blocks are distributed
- Users have a limited ability to specify customized data placement policy
  - e.g. to specify which blocks should be co-located
- Minimizing I/O costs in local disks and intra network communication

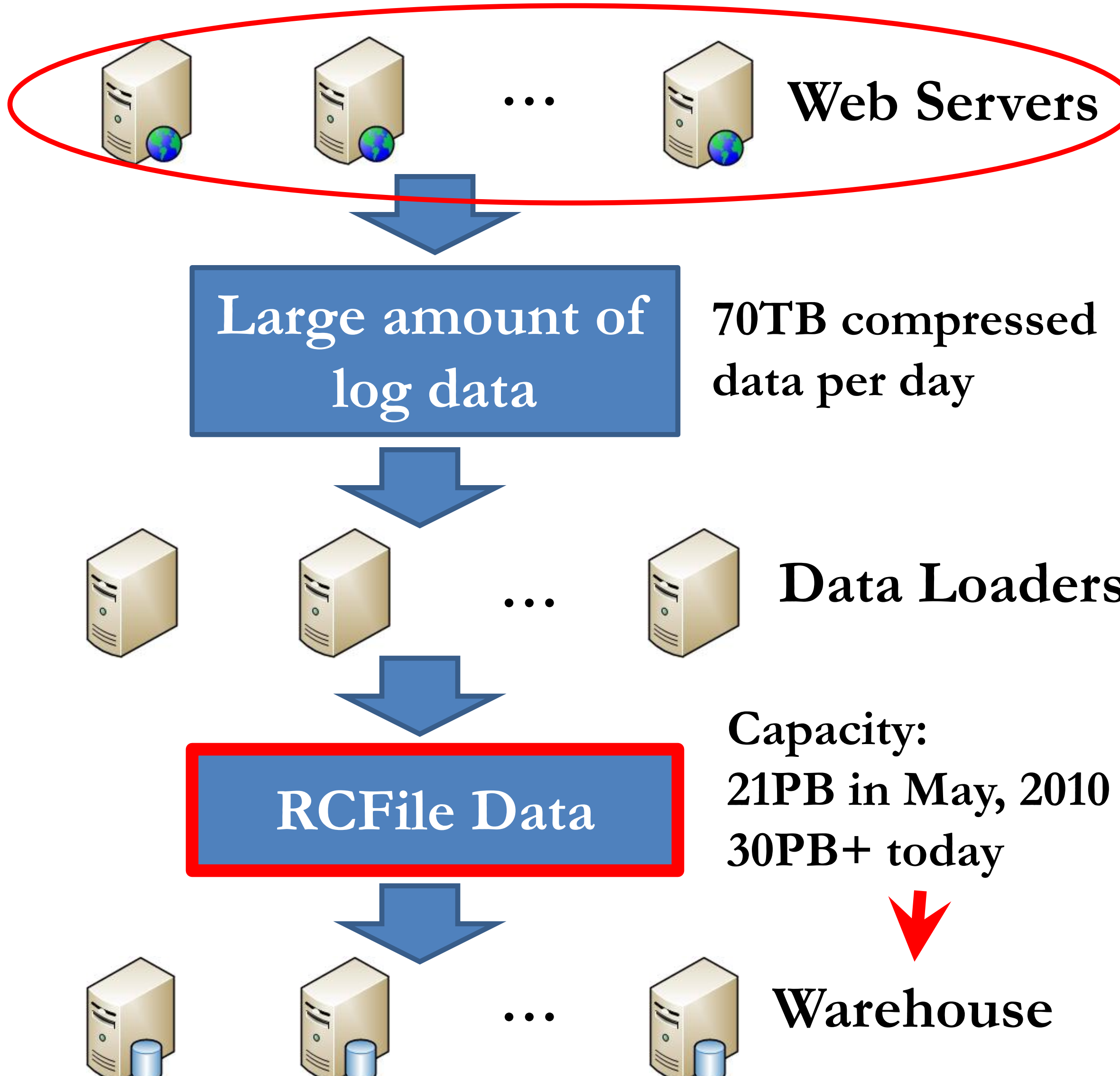
Four Requirements of Data Placement

- **Data loading (L)**
  - the overhead of writing data to distributed files system and local disks
- **Query processing (P)**
  - local storage bandwidths of query processing
  - the amount of network transfers
- **Storage space utilization (S)**
  - Data compression ratio
  - The convenience of applying efficient compression algorithms
- **Adaptive to dynamic workload patterns (W)**
  - Additional overhead on certain queries

➢ **Objective: to design and implement a data placement structure meeting these requirements in MapReduce-based data warehouses**

RCFile in Facebook

The interface to 1 billion+ users



Picture source: Visualizing Friendships, http://www.facebook.com/notes/facebook-engineering/visualizing-friendships/469716398919

Other Impact of RCFile

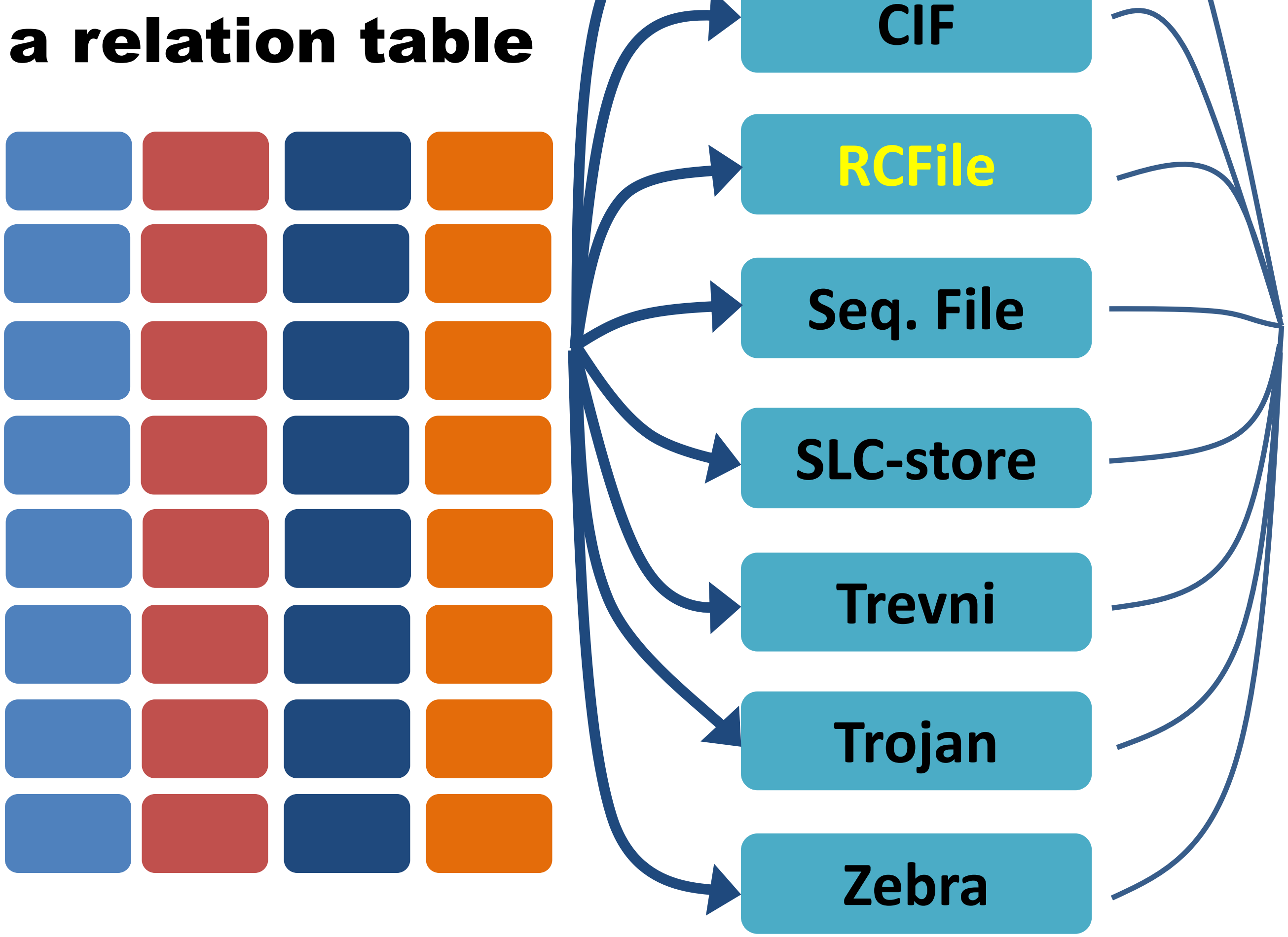
- RCFile is the default data placement structure in Facebook's production data warehouse cluster, the largest Hadoop cluster in the world.
- RCFile has been adopted in Apache Hive (since v0.4) supporting many major organizations: Taobao, Netflix and others
- RCFile has been adopted in Apache Pig (since v0.7) supporting many major organizations: Twitter, Yahoo!, Linkedin, AOL and others
- RCFile is a standard data storage structure in Hadoop software environment supported by HCatelog project.
- RCFile is a part of Elephant Bird Library developed by Twitter

A Unified Framework to Evaluate Data Placement Structures

- Several R&D projects on data placements after RCFile

  - CFile [SIGMOD 2011]
  - CIF [VLDB 2011]
  - SLC-Store [Cluster 2012]
  - Trevni in Apache Avro [open source, 2012]
  - Trojan data layouts [SOCC 2011]

Placement Structures

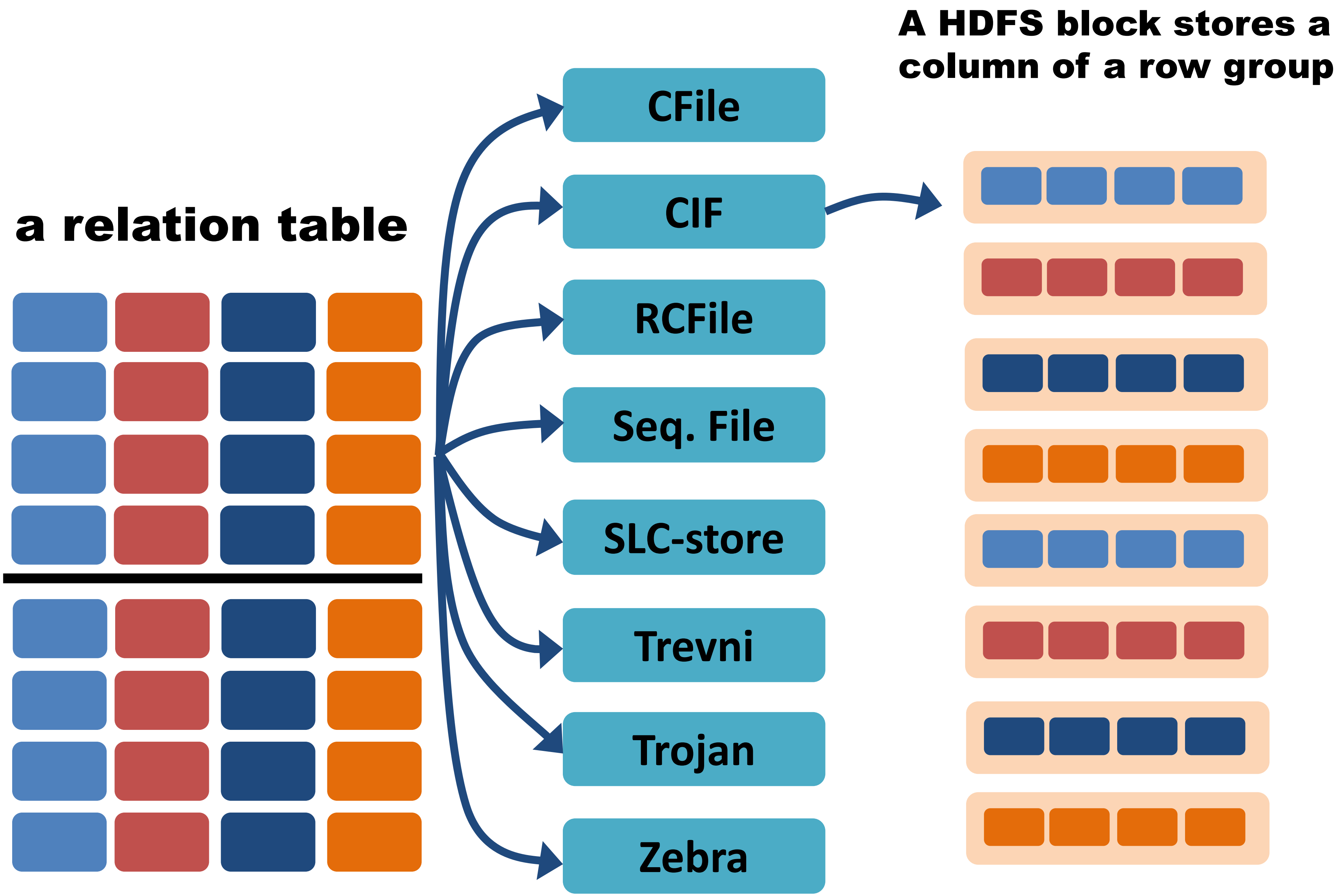


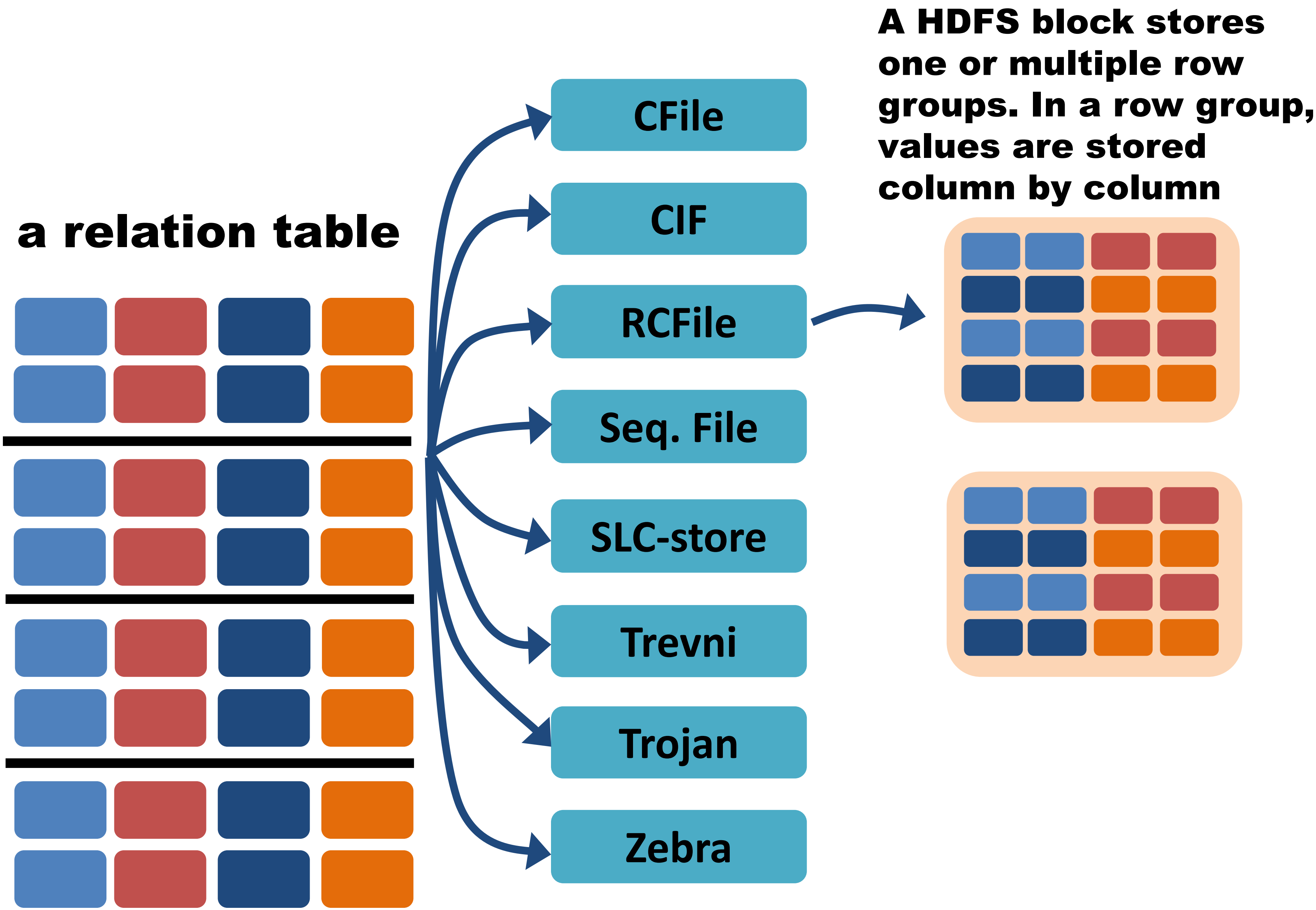
Basic Operations:

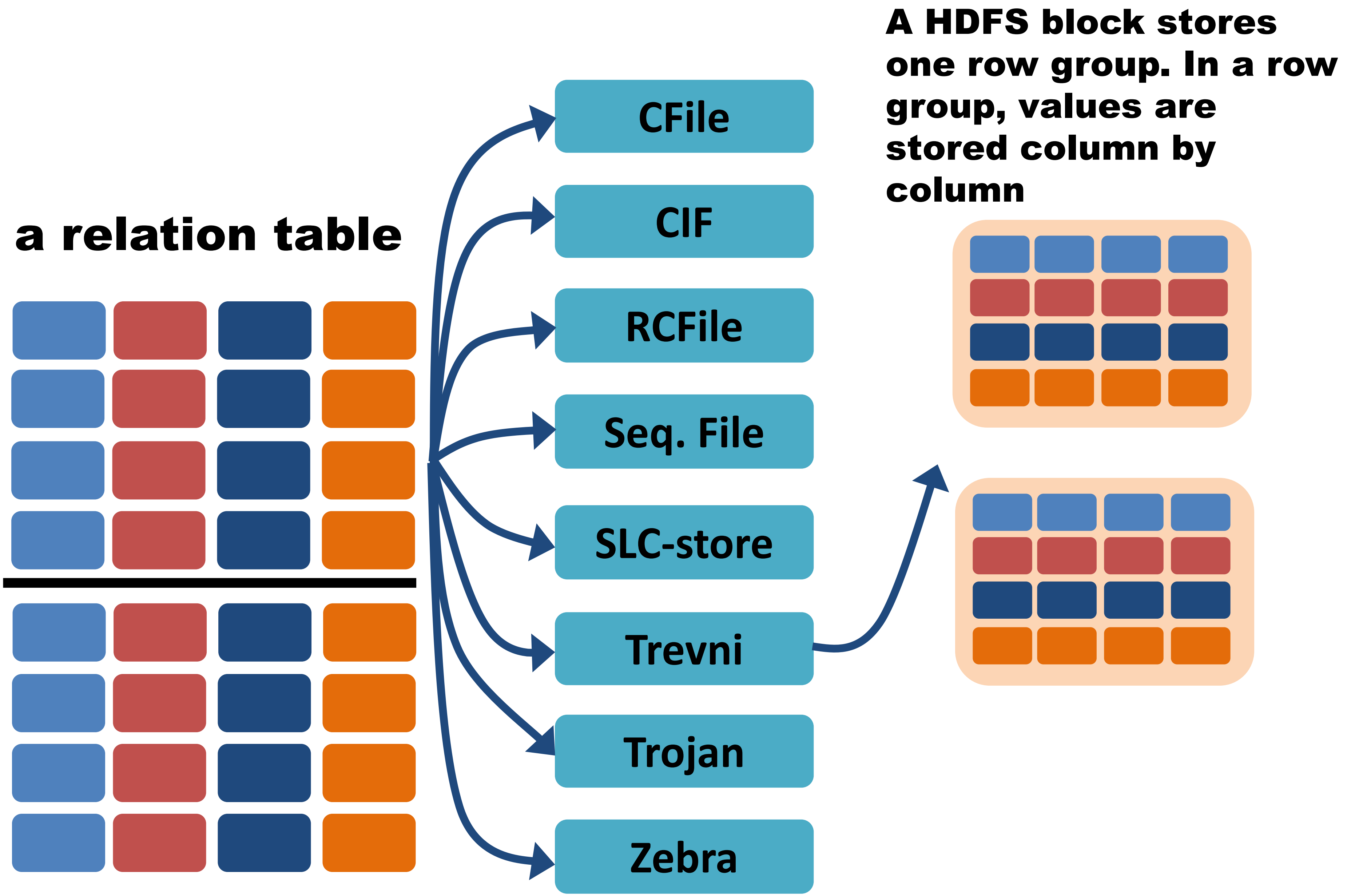
(1) **Row-column partitioning**

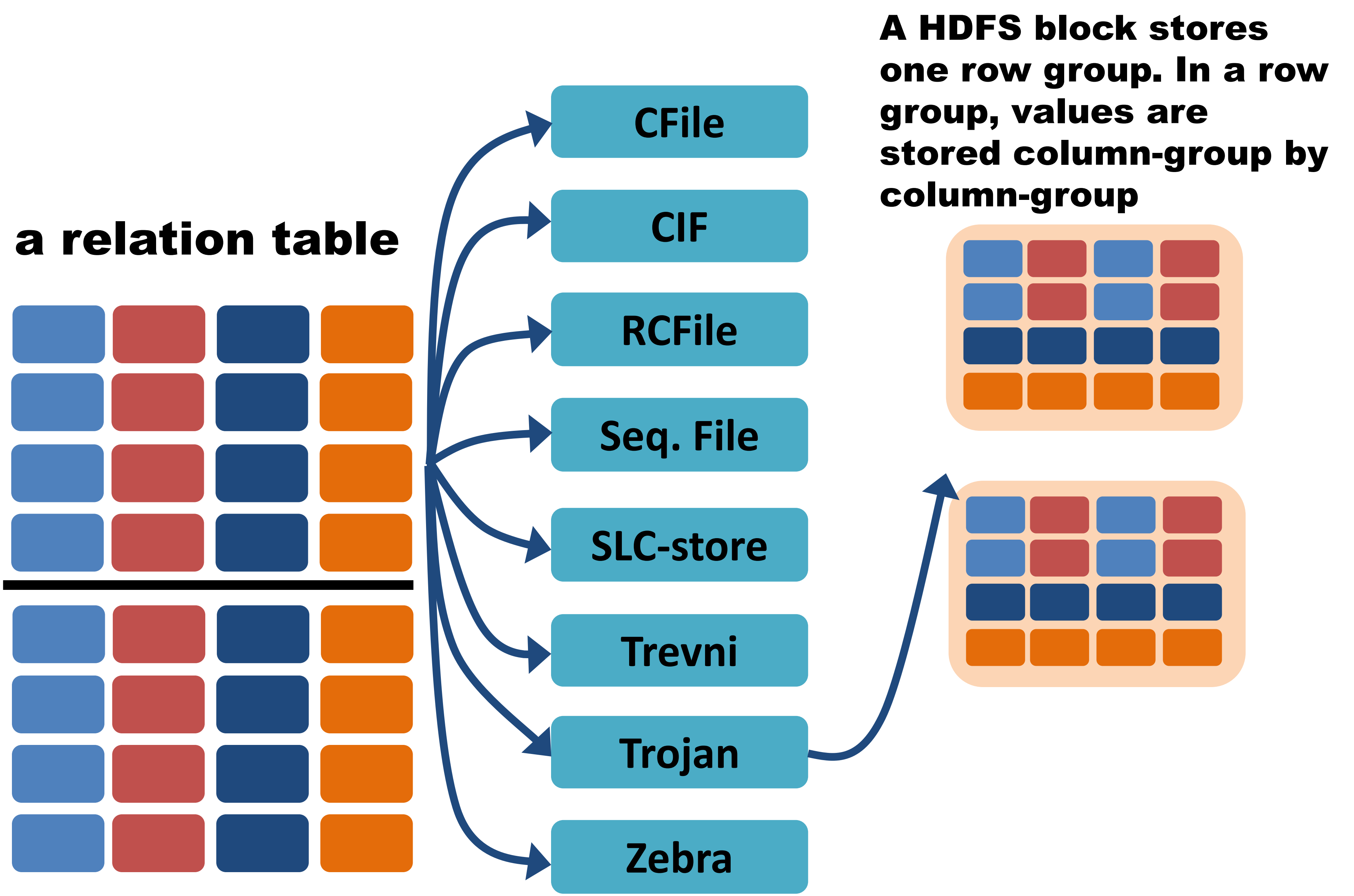
(2) **Placing partitions to file blocks**

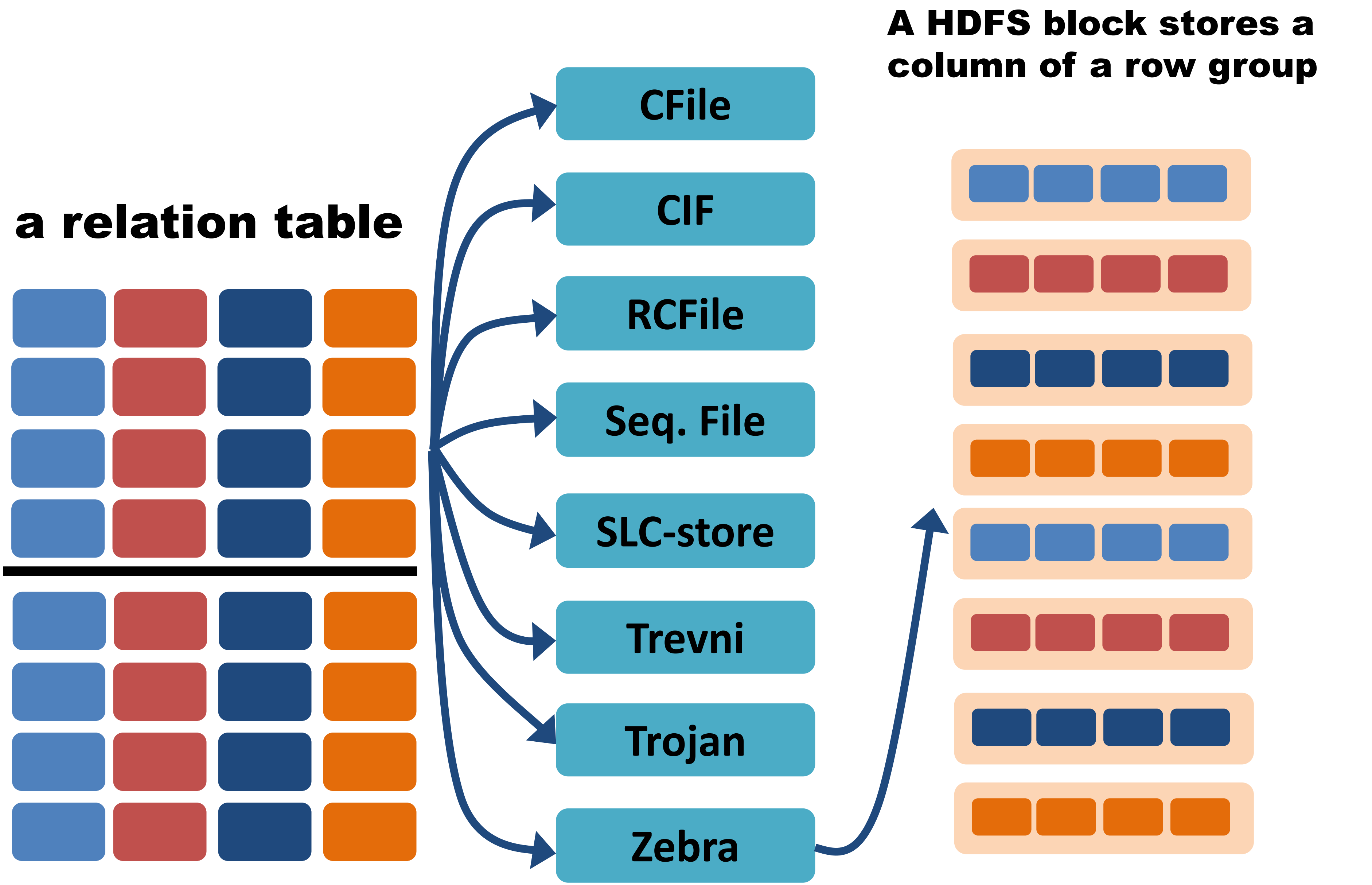
A unified representation for all schemes


A HDFS block stores a
column of a row group
a relation table
CFile
CIF
RCFile
Seq. File
SLC-store
Trevni
Trojan
Zebra

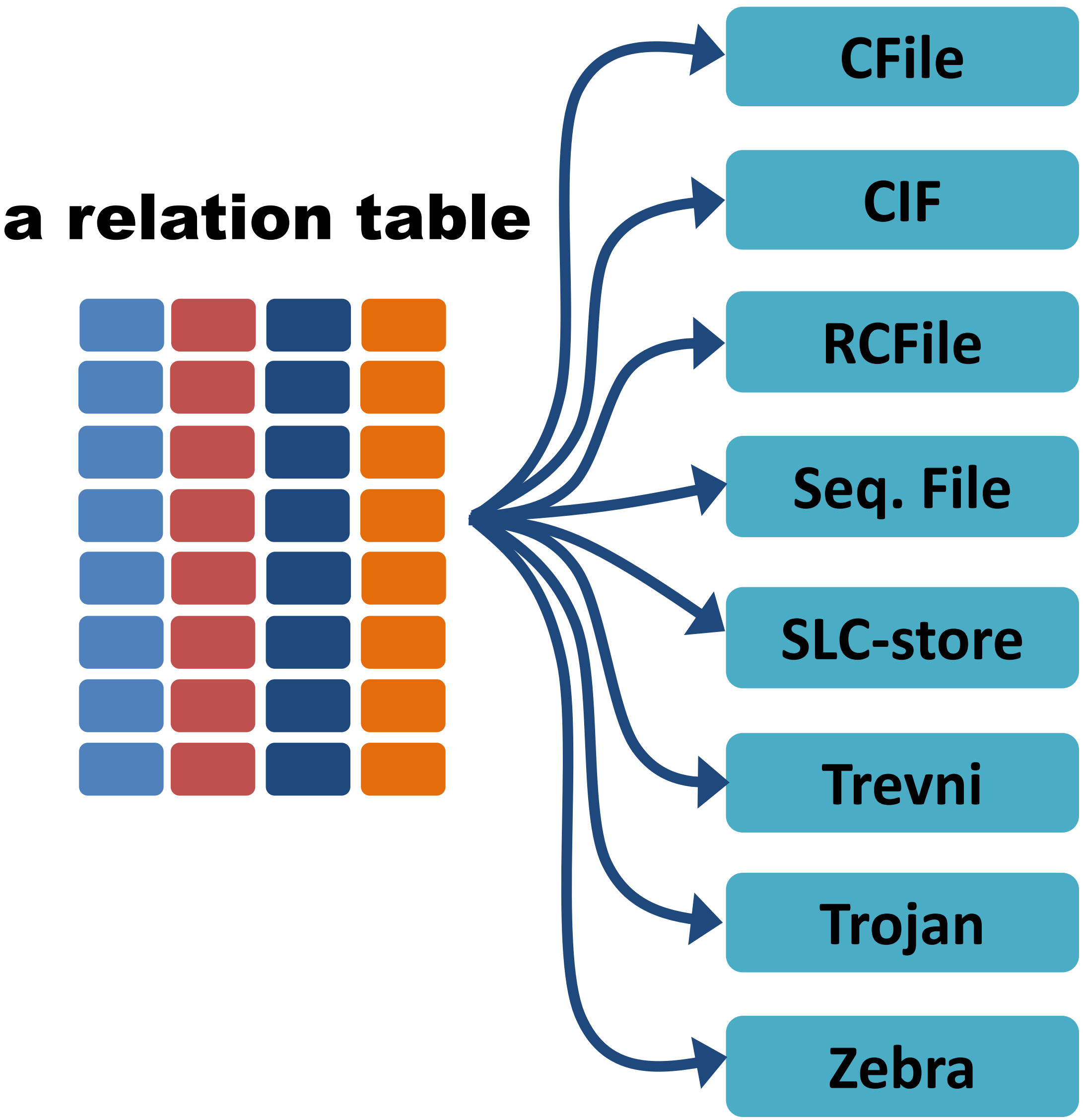

A HDFS block stores one or multiple row groups. In a row group, values are stored column by column
a relation table
CFile
CIF
RCFile
Seq. File
SLC-store
Trevni
Trojan
Zebra


A HDFS block stores one row group. In a row group, values are stored column by column
a relation table
CFile
CIF
RCFile
Seq. File
SLC-store
Trevni
Trojan
Zebra


A HDFS block stores one row group. In a row group, values are stored column-group by column-group
a relation table
CFile
CIF
RCFile
Seq. File
SLC-store
Trevni
Trojan
Zebra


A HDFS block stores a column of a row group
a relation table
CFile
CIF
RCFile
Seq. File
SLC-store
Trevni
Trojan
Zebra

Evaluations of Table Placement Schemes in Different Formats

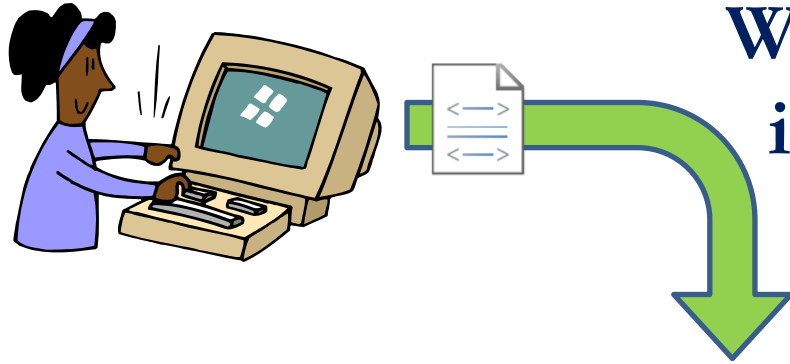


Comprehensive analyses

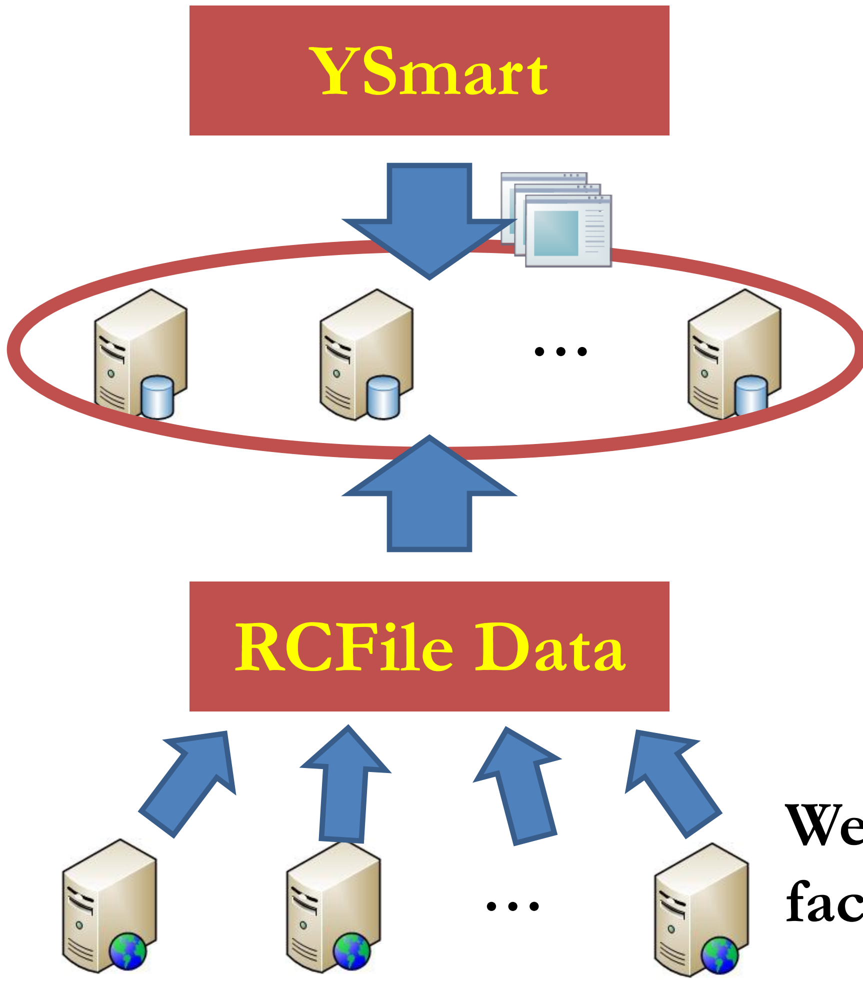
- **Performance**
  - **Execution time**
- **Fault-tolerance**
  - **Recovery time**
- **Implications of design choices**

Final conclusion:
Rcfile plays the most balanced role

Where are YSmart and RCFile in the Big Data Ecosystem?



Translate SQL-like queries to MapReduce jobs



A Hadoop-powered Data Warehousing System (Hive)

Web servers for 1 billion facebook users

Conclusion

- **The original MapReduce model serving as an big data processing engine can only provide simple analytics**
- **Several hurtles blocking highly parallel processing are addressed by our R&D efforts:**
  - **Unnecessary network and disk latencies** (SideWalk, Ysmart, RCFile)
  - **Fault tolerance overhead** (RCFile)
  - **Processing engine modification** (SideWalk, Ysmart, RCFile)
  - **"One size fits all" methodology** (SideWalk)
  - **Processing engine structure unawareness** (Ysmart)
- **RCFile and YSmart are in the critical path of big data ecosystem.**
- **SideWalk provides a communication facility for critically necessary messages in big data ecosystems**

Thank You!