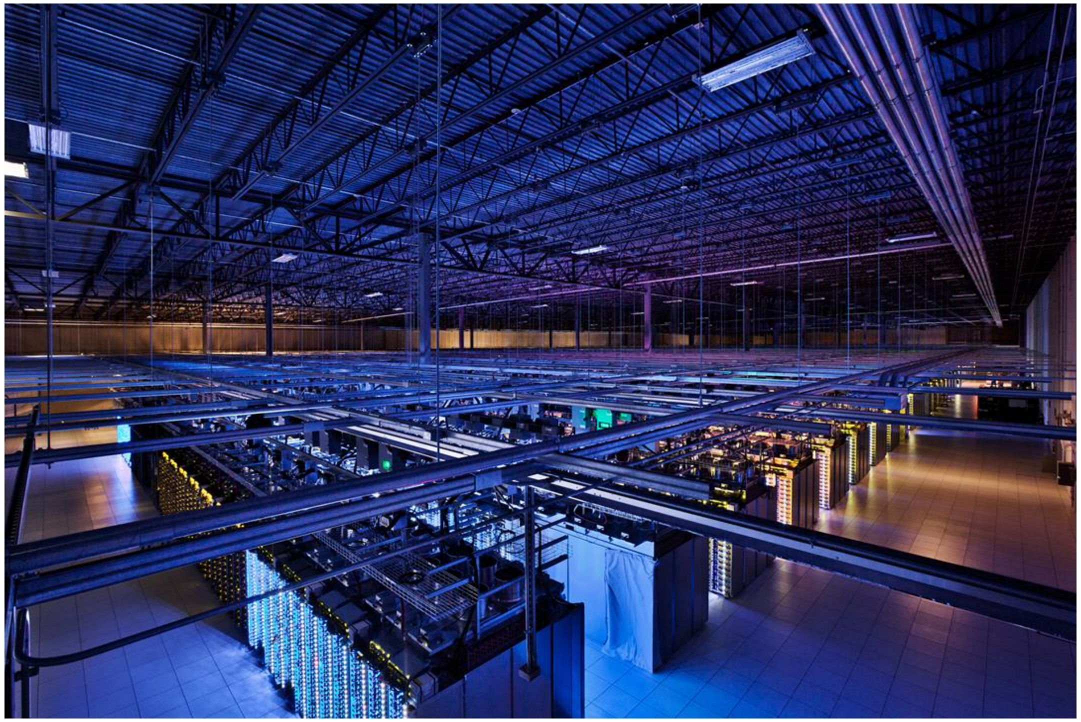
Big Data Infrastructures & Technologies

Hadoop Streaming Revisit



www.cwi.nl/~boncz/bads

ENRON Mapper

```python
#!/usr/bin/env python

import sys

for line in sys.stdin:
    line = line.strip().split('\t')
    if len(line) != 3:
        continue
    date, sender, recipient = line
    date = date[:10]
    if (date <= "2001-11-05" or date >= "2001-11-08"):
        continue
    if "enron" in recipient or not "enron" in sender:
        continue
    print sender + '\t' + recipient
```

ENRON Mapper Output (Excerpt)

acomnes@enron.com edward.snowden@cia.gov

blake.walker@enron.com alex.berenson@nyt.com

ENRON Reducer

```
#!/usr/bin/env python

import sys

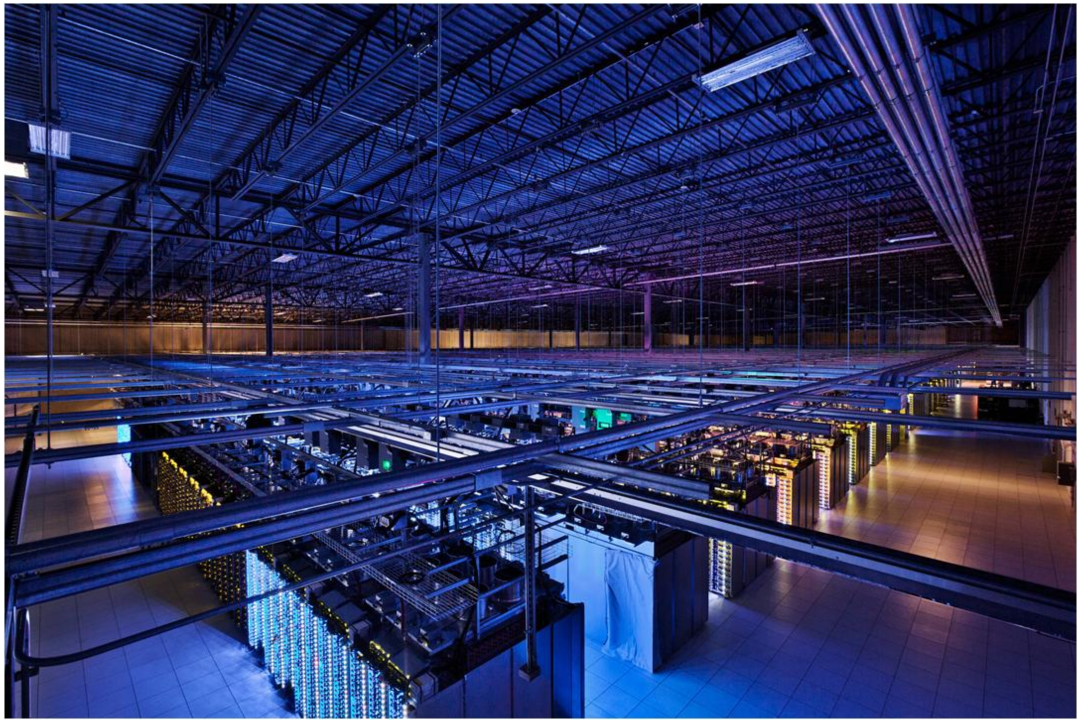
current_sender = ''
current_count = 0
for line in sys.stdin:
    line = line.strip().split('\t')
    if len(line) != 2:
        continue
    sender, recipient = line
    if (sender != current_sender):
        if (current_count > 1):
            print current_sender + '\t' + str(current_count)
        current_sender = sender
        current_count = 0
    current_count += 1
if (current_count > 1):
    print current_sender + '\t' + str(current_count)
```

Optimization?

ENRON Final Results (Excerpt)

acomnes@enron.com	32
anne.jolibois@enron.com	18
billy.lemmons@enron.com	11
brad.guilmino@enron.com	11

The Spark Framework



credits:
Matei Zaharia & Xiangrui Meng

www.cwi.nl/~boncz/bads

What is Spark?

- Fast and expressive cluster computing system interoperable with Apache Hadoop
- Improves efficiency through:
  - In-memory computing primitives
  - General computation graphs

→ Up to 100 × faster (2-10 × on disk)

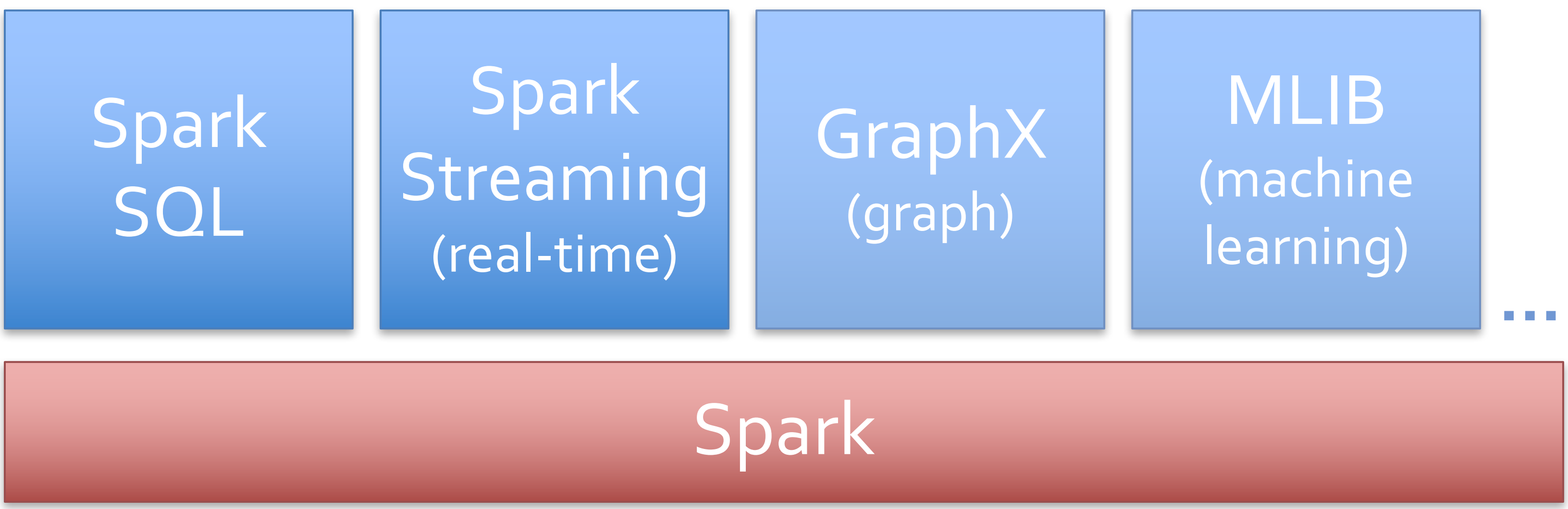
- Improves usability through:
  - Rich APIs in Scala, Java, Python
  - Interactive shell

→ Often 5 × less code

credits:
Matei Zaharia & Xiangrui Meng

The Spark Stack

- Spark is the basis of a wide set of projects in the Berkeley Data Analytics Stack (BDAS)



More details: amplab.berkeley.edu

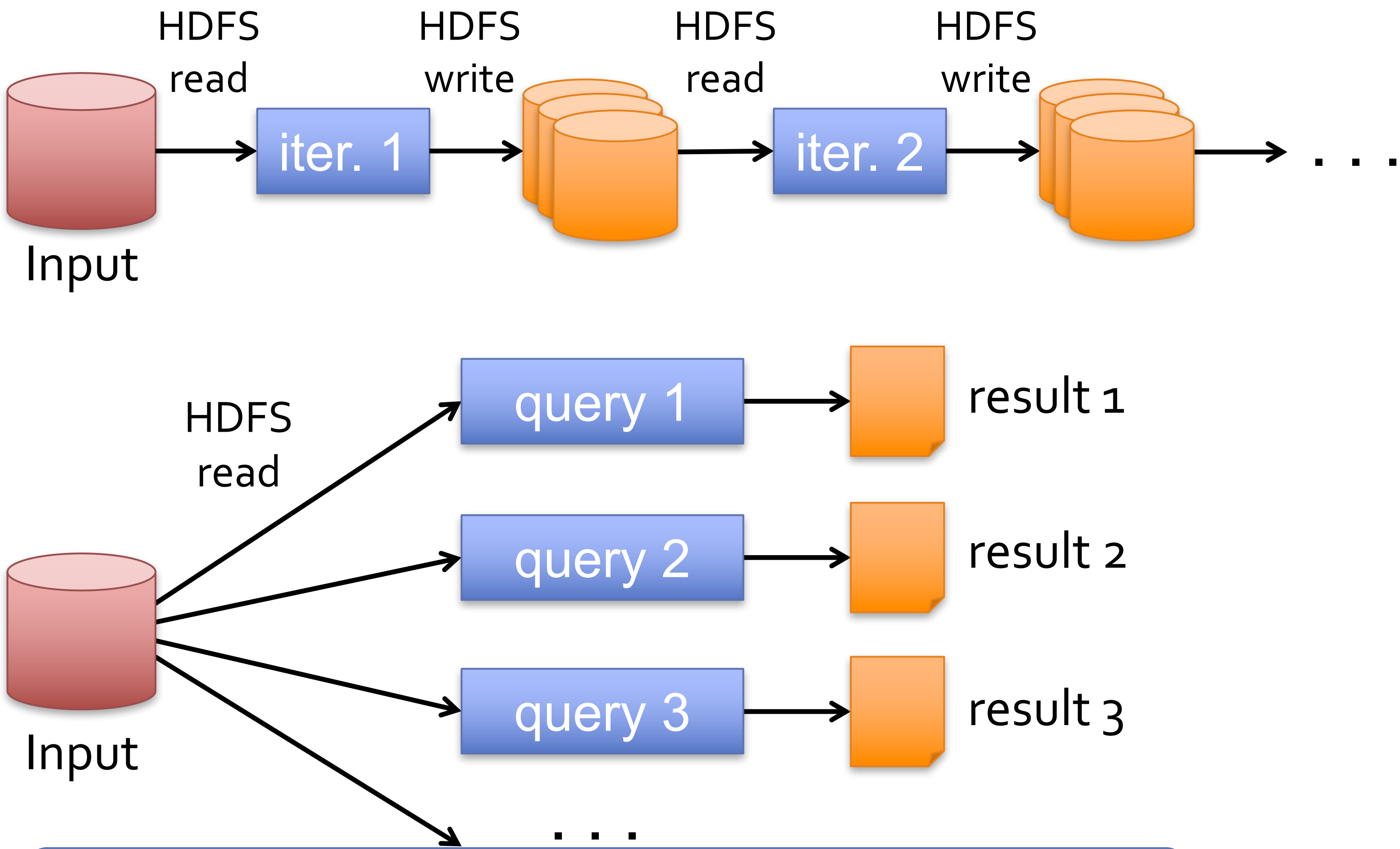
credits:
Matei Zaharia & Xiangrui Meng

Why a New Programming Model?

- MapReduce greatly simplified big data analysis
- But as soon as it got popular, users wanted more:
  - More **complex**, multi-pass analytics (e.g. ML, graph)
  - More **interactive** ad-hoc queries
  - More **real-time** stream processing
- All 3 need faster **data sharing** across parallel jobs

credits:
Matei Zaharia & Xiangrui Meng

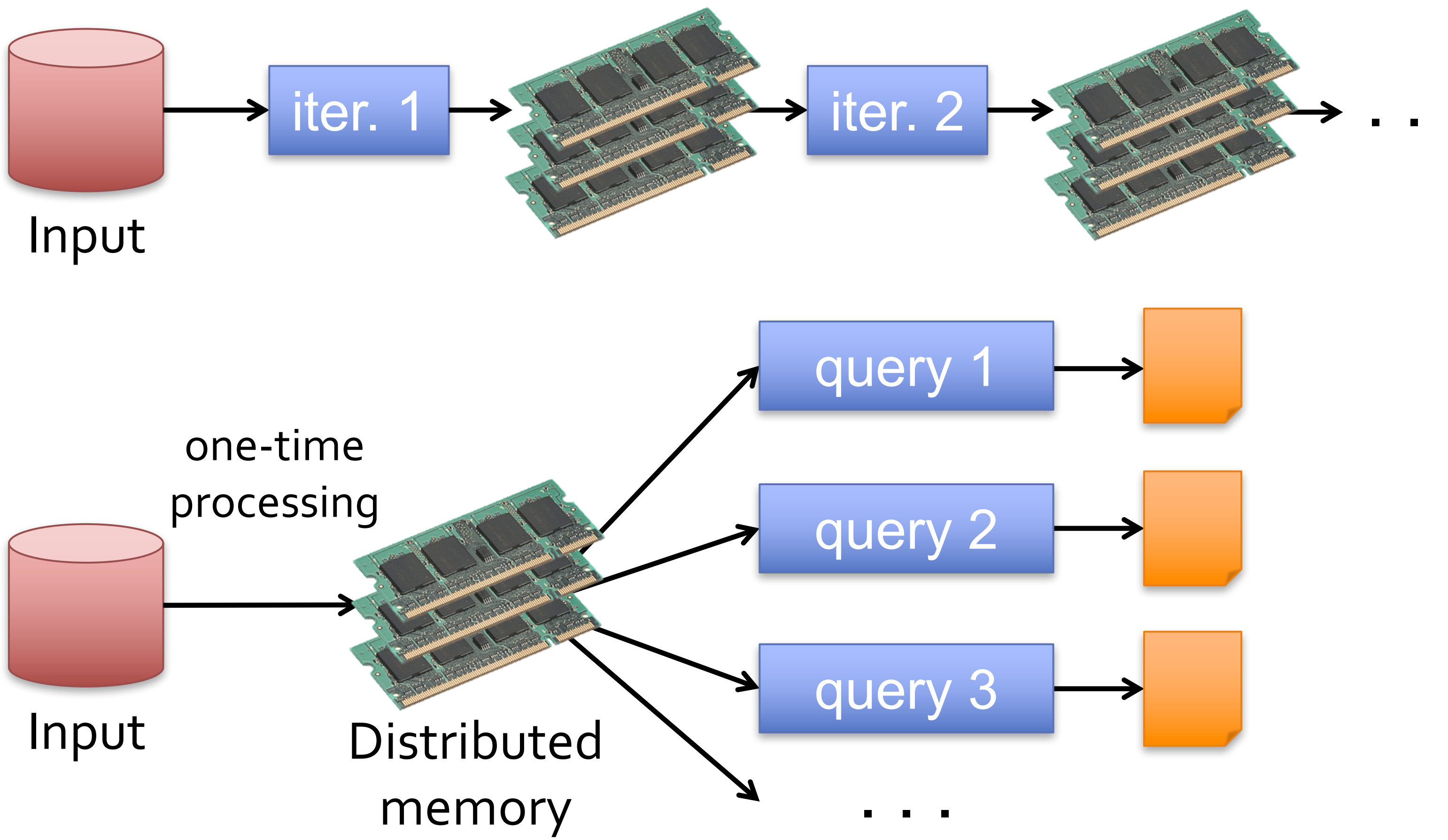
Data Sharing in MapReduce



Slow due to replication, serialization, and disk IO

credits:
Matei Zaharia & Xiangrui Meng

Data Sharing in Spark



~10 × faster than network and disk

credits:
Matei Zaharia & Xiangrui Meng

Spark Programming Model

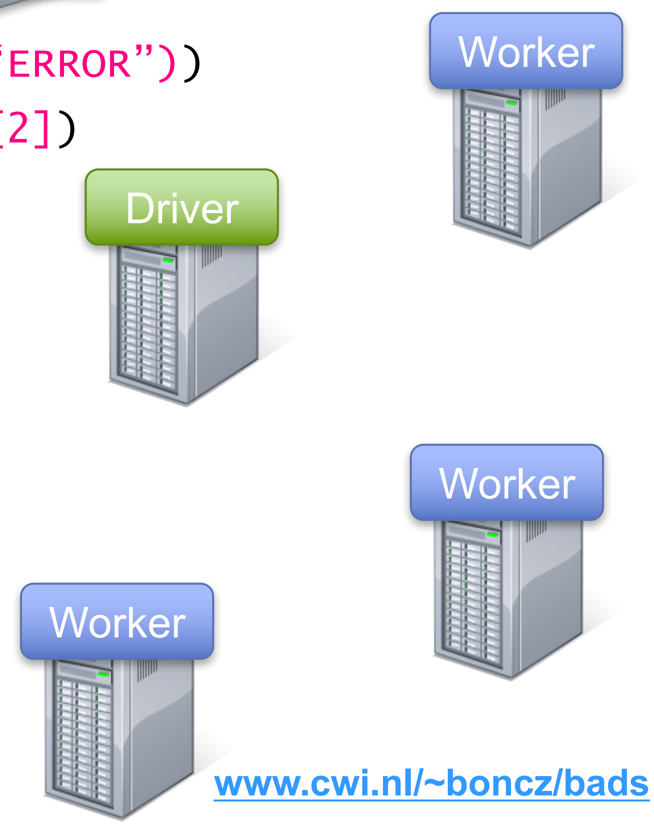
- Key idea: *resilient distributed datasets (RDDs)*
  - Distributed collections of objects that can be cached in memory across the cluster
  - Manipulated through parallel operators
  - Automatically *recomputed* on failure
- Programming interface
  - Functional APIs in Scala, Java, Python
  - Interactive use from Scala shell

Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

Transformed RDD

```
lines = spark.textFile("hdfs://...")
errors = lines.filter(lambda x: x.startswith("ERROR"))
messages = errors.map(lambda x: x.split('\t')[2])
messages.cache()
```



credits:
Matei Zaharia & Xiangrui Meng

Lambda Functions

```
errors = lines.filter(lambda x: x.startswith("ERROR"))
messages = errors.map(lambda x: x.split('\t')[2])
```

Lambda function ← functional programming!

= implicit function definition

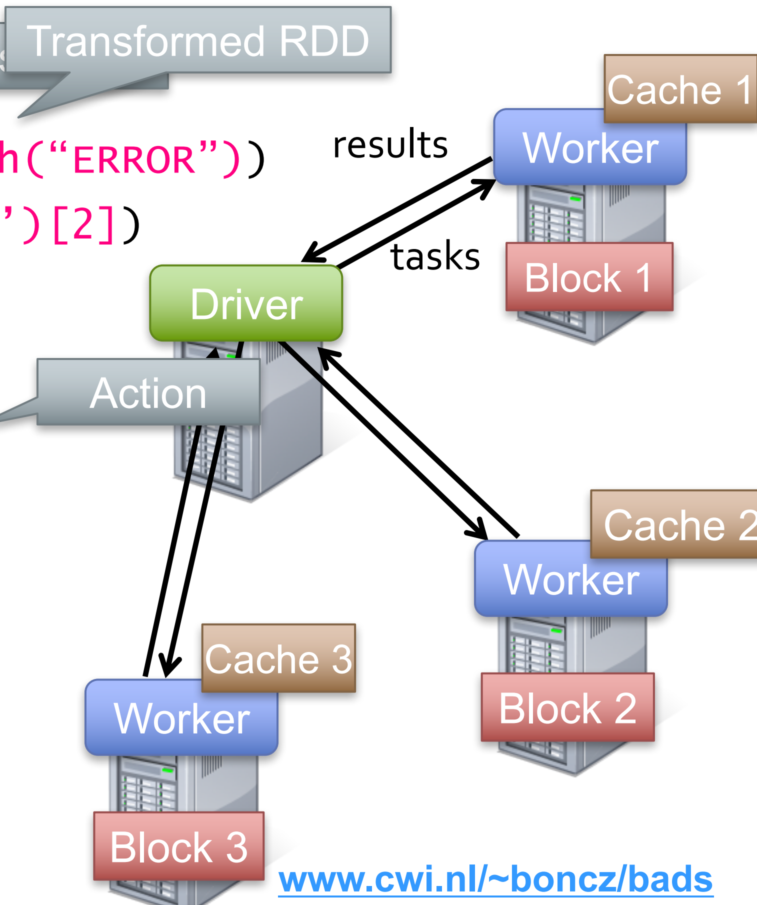
```
bool detect_error(string x) {
    return x.startswith("ERROR");
}
```

Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

```
lines = spark.textFile("hdfs://...")
errors = lines.filter(lambda x: x.startswith("ERROR"))
messages = errors.map(lambda x: x.split('\t')[2])
messages.cache()

messages.filter(lambda x: "foo" in x).count
messages.filter(lambda x: "bar" in x).count
. . .
```



Result: scaled to 1 TB data in 5-7 sec (vs 170 sec for on-disk data)

Result: full-text search of Wikipedia in <1 sec (vs 20 sec for on-disk data)

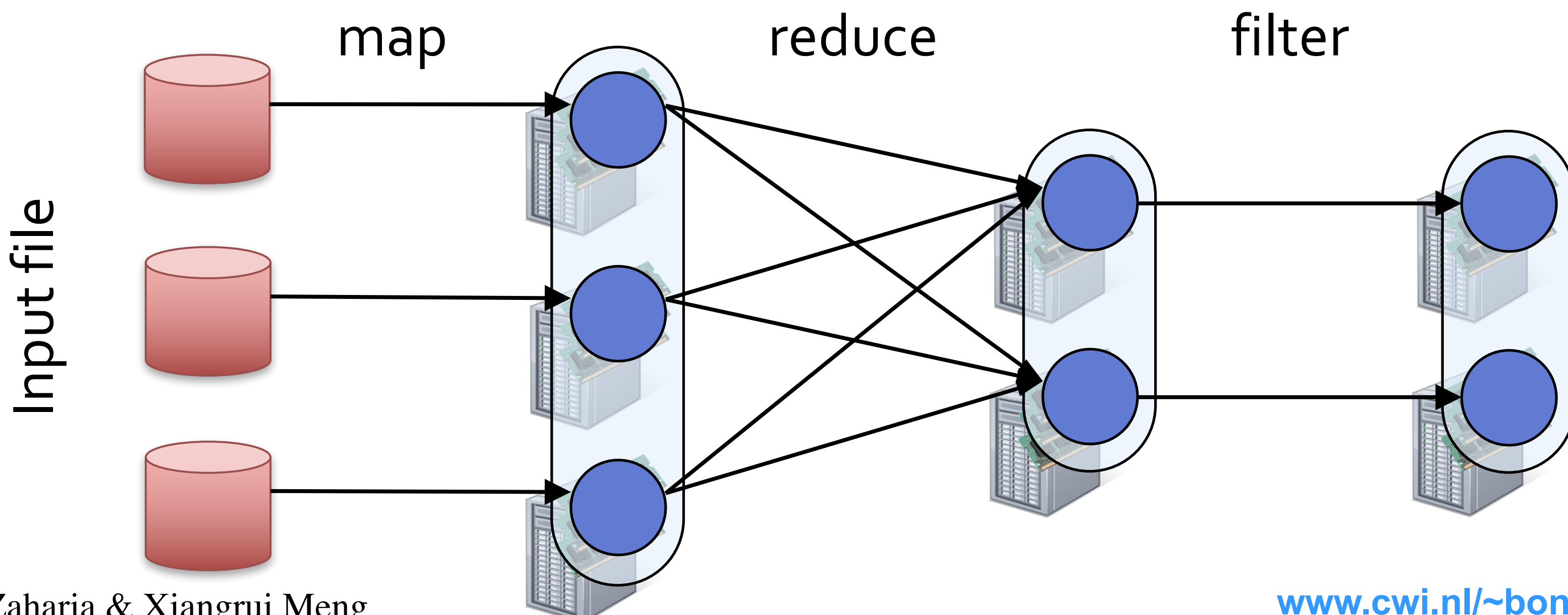
credi
Mate

www.cwi.nl/~boncz/bads

Fault Tolerance

RDDs track *lineage* info to rebuild lost data

- ```
  file.map(lambda rec: (rec.type, 1))
      .reduceByKey(lambda x, y: x + y)
      .filter(lambda (type, count): count > 10)
  ```

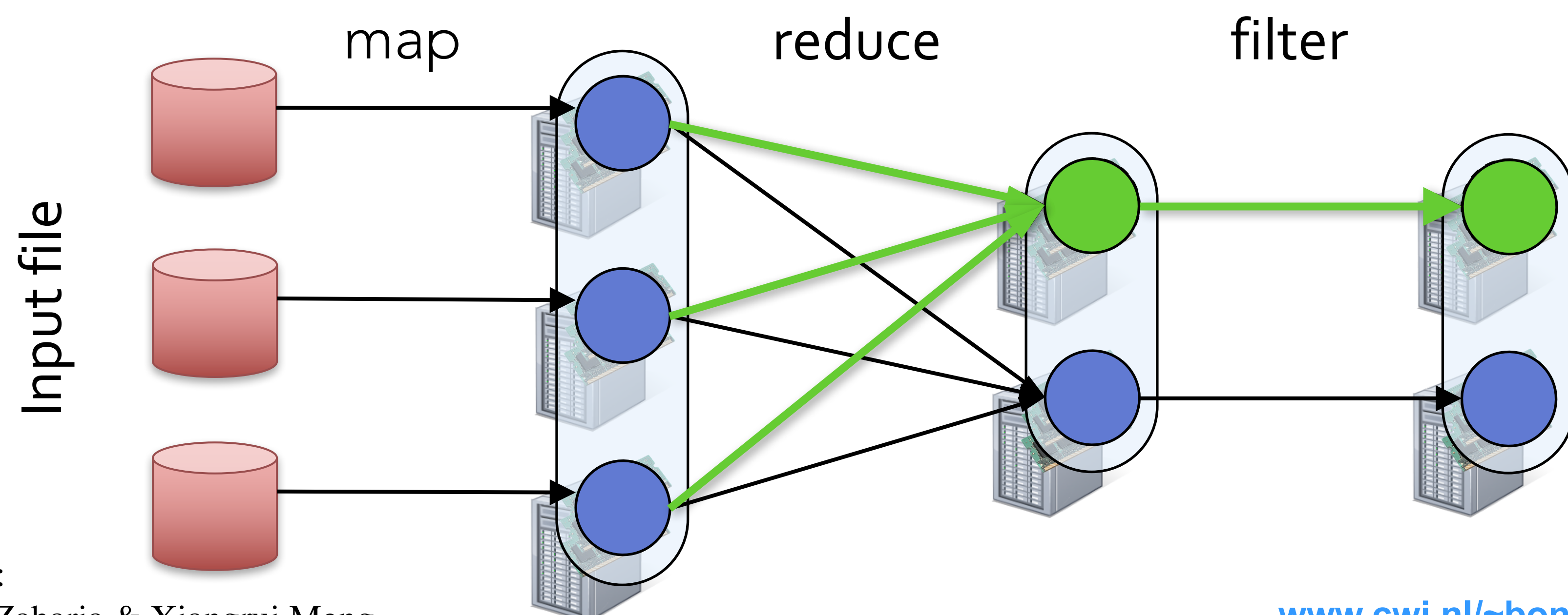


credits:
Matei Zaharia & Xiangrui Meng
  ```

Fault Tolerance

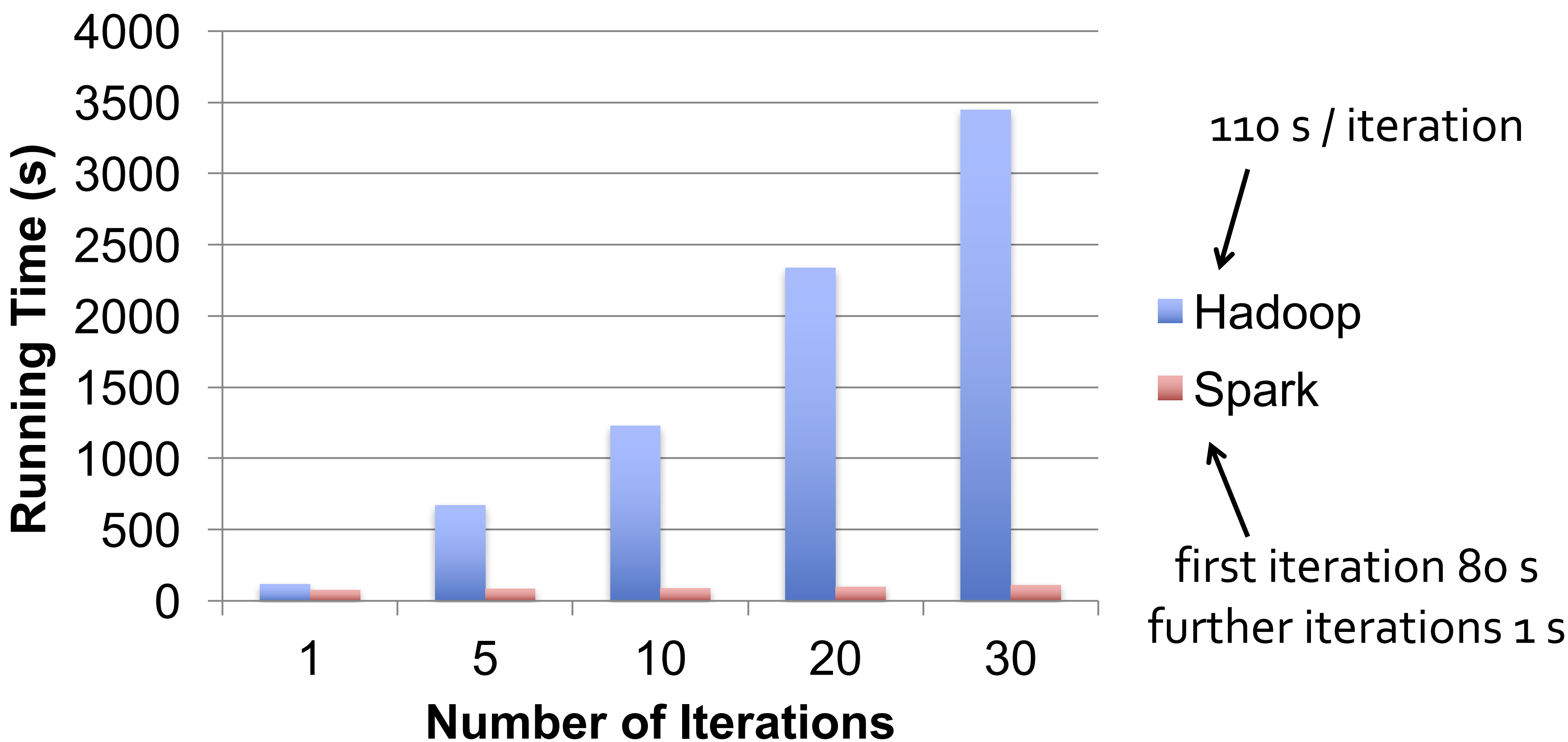
RDDs track *lineage* info to rebuild lost data

```
• file.map(lambda rec: (rec.type, 1))
      .reduceByKey(lambda x, y: x + y)
      .filter(lambda (type, count): count > 10)
```



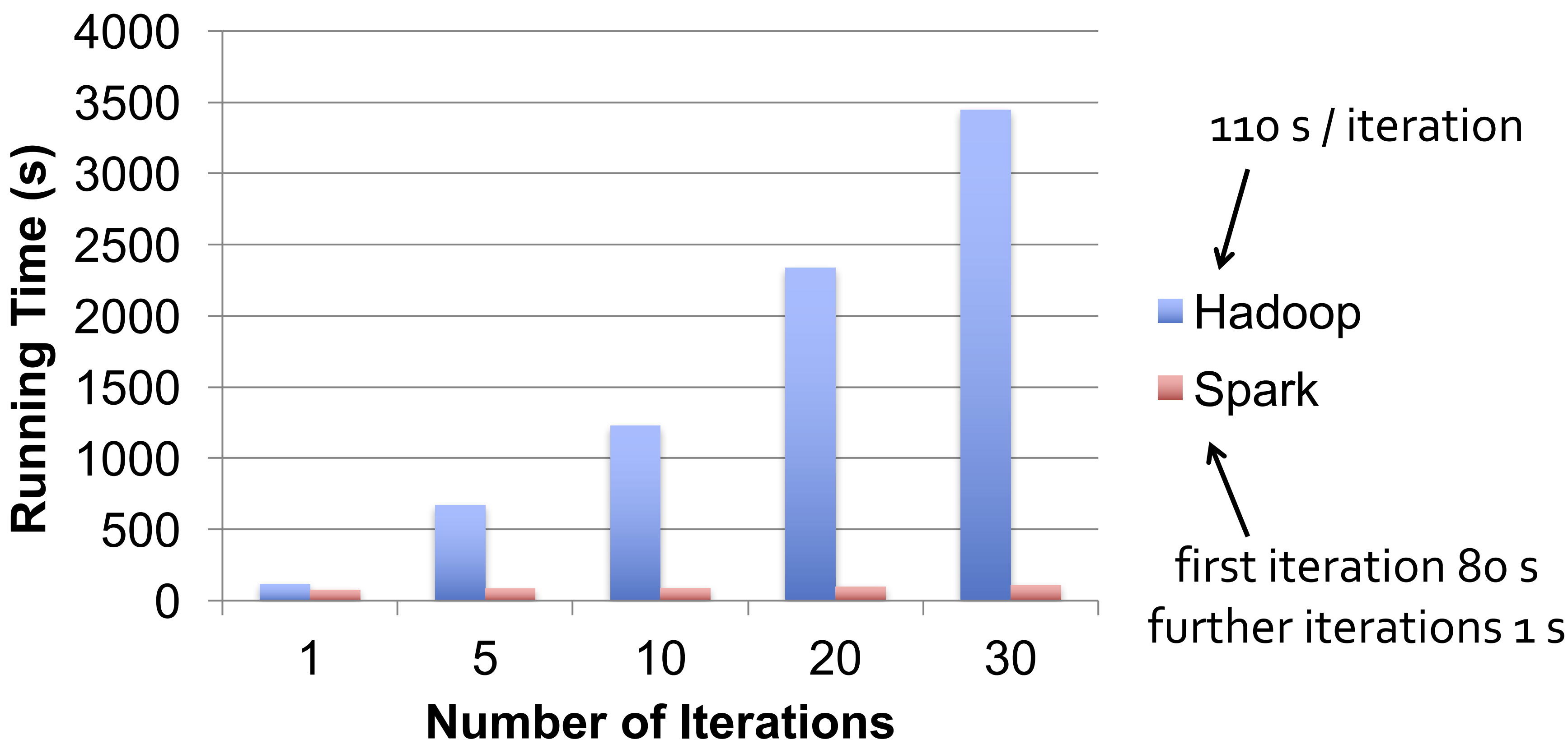
credits:
Matei Zaharia & Xiangrui Meng

Example: Logistic Regression



credits:
Matei Zaharia & Xiangrui Meng

Example: Logistic Regression



credits:
Matei Zaharia & Xiangrui Meng

Spark in Scala and Java

```
// Scala:

val lines = sc.textFile(...)
lines.filter(x => x.contains("ERROR")).count()


// Java:

JavaRDD<String> lines = sc.textFile(...);
lines.filter(new Function<String, Boolean>() {
  Boolean call(String s) {
    return s.contains("error");
  }
}).count();
```

credits:
Matei Zaharia & Xiangrui Meng

Supported Operators

- map
- filter
- groupBy
- sort
- union
- join
- leftOuterJoin
- rightOuterJoin

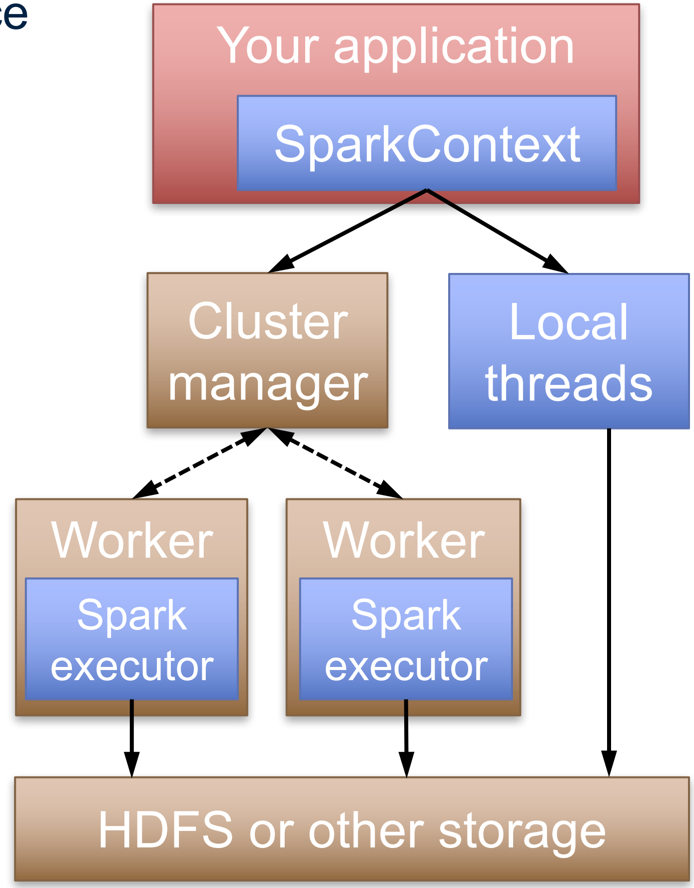
- reduce
- count
- fold
- reduceByKey
- groupByKey
- cogroup
- cross
- zip

sample
take
first
partitionBy
mapWith
pipe
save
...

credits:
Matei Zaharia & Xiangrui Meng

Software Components

- Spark client is library in user program (1 instance per app)
- Runs tasks locally or on cluster
  – Mesos, YARN, standalone mode
- Accesses storage systems via Hadoop InputFormat API
  – Can use HBase, HDFS, S3, …



credits:
Matei Zaharia & Xiangrui Meng

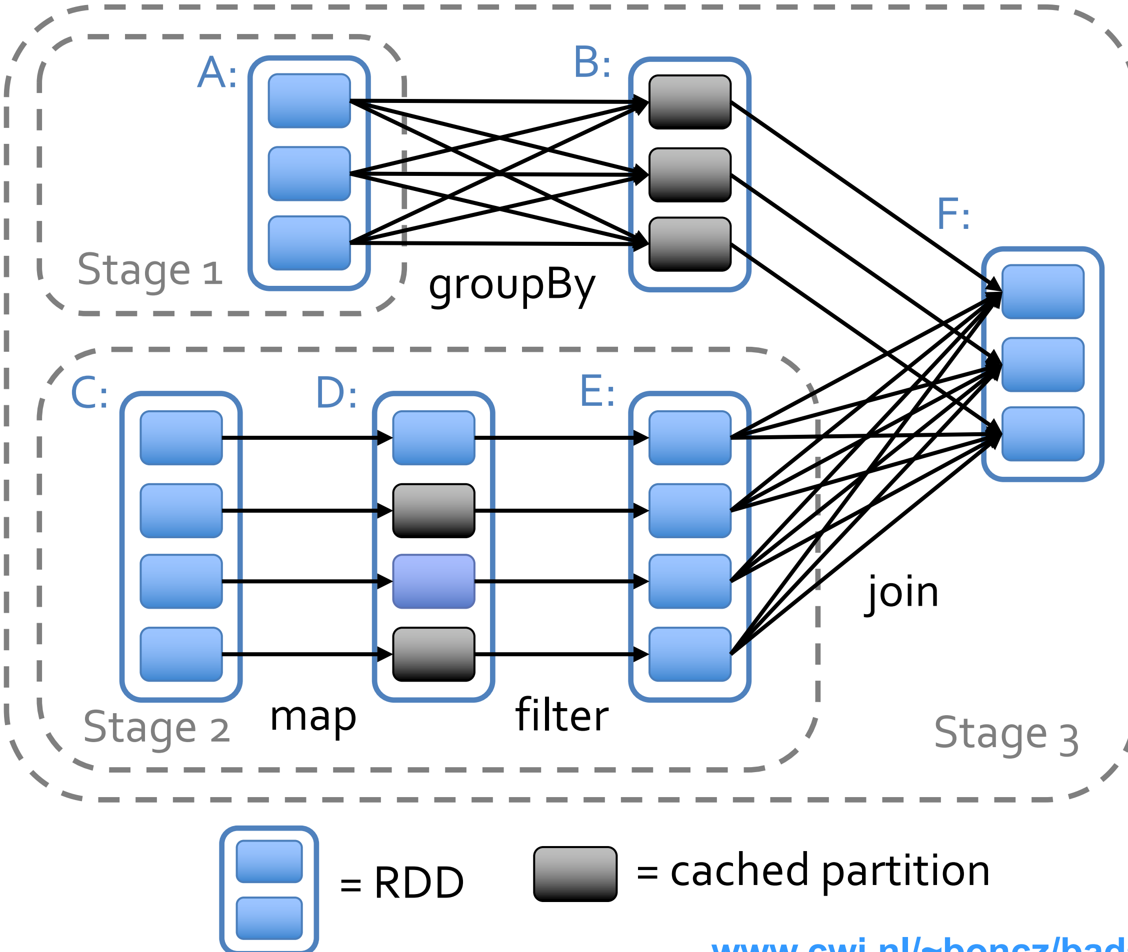
Task Scheduler

General task graphs

Automatically pipelines functions

Data locality aware

Partitioning aware to avoid shuffles



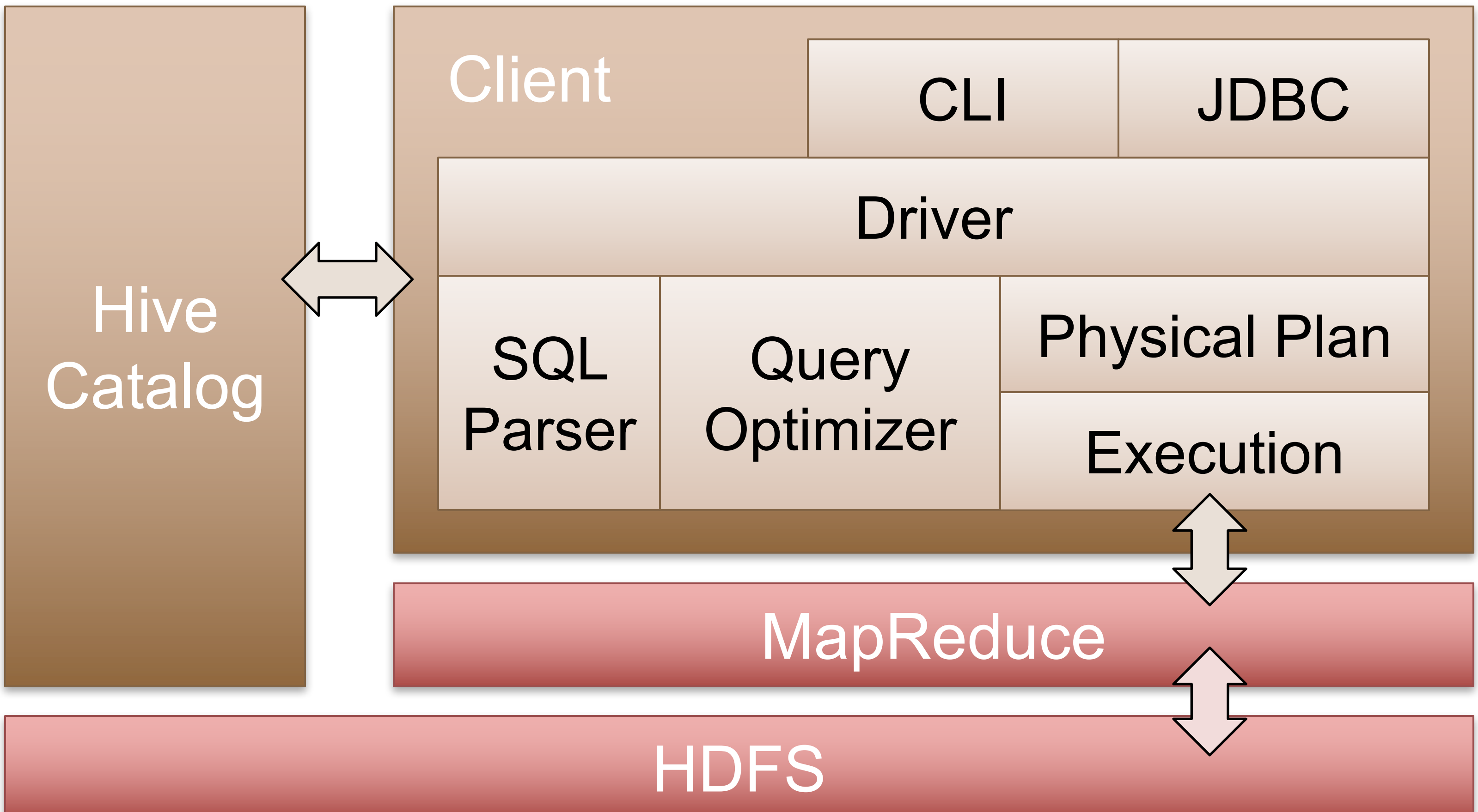
credits:
Matei Zaharia & Xiangrui Meng

Spark SQL

- Columnar SQL analytics engine for Spark
  - Support both SQL and complex analytics
  - Columnar storage, JIT-compiled execution, Java/Scala/Python UDFs
  - Catalyst query optimizer (also for DataFrame scripts)

credits:
Matei Zaharia & Xiangrui Meng

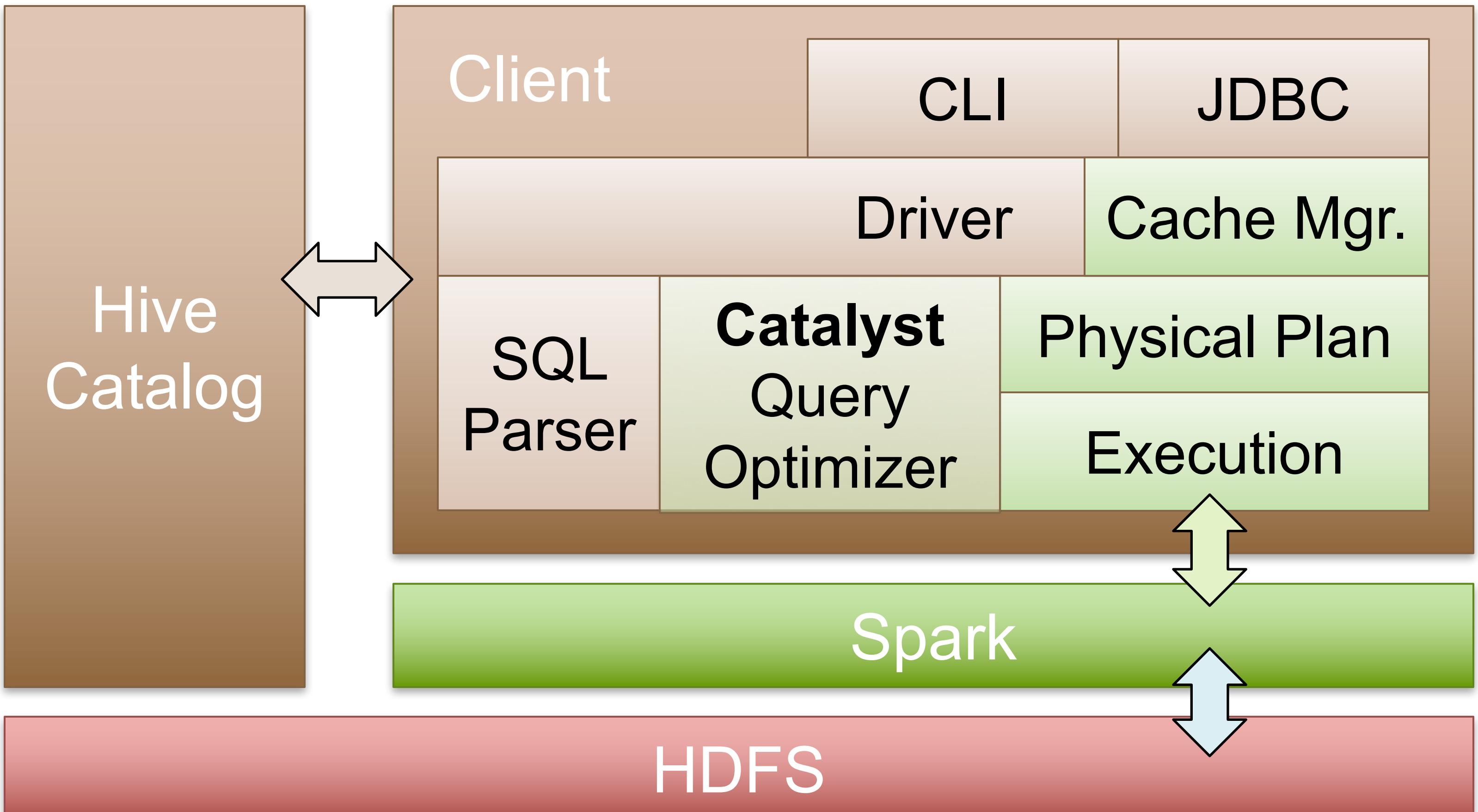
Hive Architecture



credits:
Matei Zaharia & Xiangrui Meng

CWI
Centrum Wiskunde & Informatica

Spark SQL Architecture



credits:
Matei Zaharia & Xiangrui Meng

[Engle et al, SIGMOD 2012]

www.cwi.nl/~boncz/bads

From RDD to DataFrame

```
ctx = new HiveContext()
users = ctx.table("users")
young = users.where(users("age") < 21)
println(young.count())
```

- A distributed collection of rows with the same schema (RDDs suffer from type erasure)
- Can be constructed from external data sources or RDDs into essentially an RDD of Row objects (SchemaRDDs as of Spark < 1.3)
- Supports relational operators (e.g. *where*, *groupby*) as well as Spark operations.
- Evaluated lazily → non-materialized *logical* plan

credits:
Matei Zaharia & Reynold Xin

DataFrame: Data Model

- Nested data model
- Supports both primitive SQL types (boolean, integer, double, decimal, string, data, timestamp) and complex types (structs, arrays, maps, and unions); also user defined types.
- First class support for complex data types

DataFrame Operations

- Relational operations (select, where, join, groupBy) via a DSL
- Operators take *expression* objects
- Operators build up an abstract syntax tree (AST), which is then optimized by *Catalyst*.

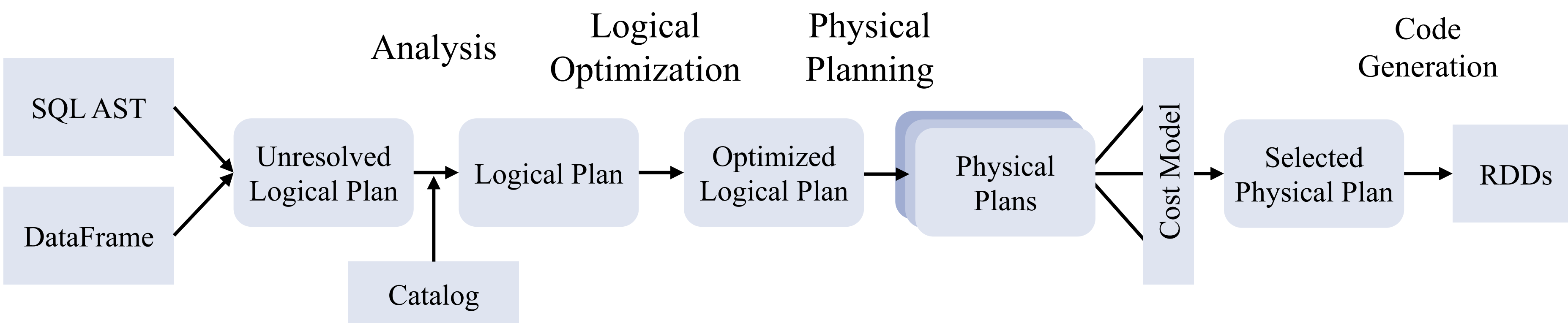
```
employees
  .join(dept, employees("deptId") === dept("id"))
  .where(employees("gender") === "female")
  .groupBy(dept("id"), dept("name"))
  .agg(count("name"))
```

- Alternatively, register as temp SQL table and perform traditional SQL query strings

```
users.where(users("age") < 21)
     .registerTempTable("young")
ctx.sql("SELECT count(*), avg(age) FROM young")
```

credits:
Matei Zaharia & Reynold Xin

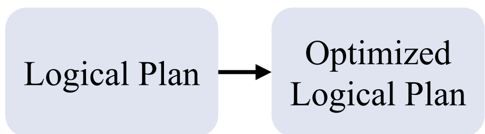
Catalyst: Plan Optimization & Execution



credits:
Matei Zaharia & Reynold Xin

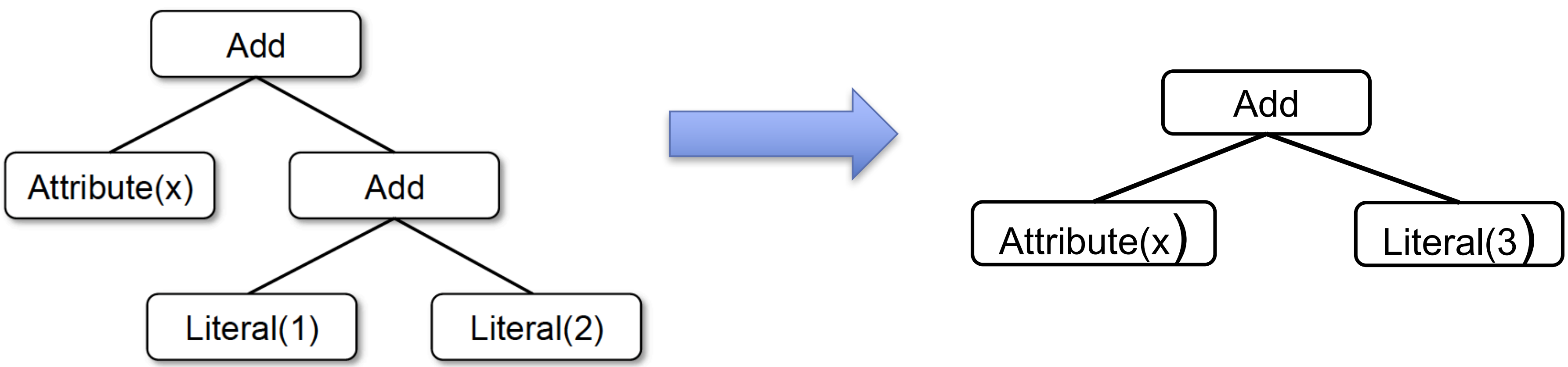
Catalyst Optimization Rules

Logical Optimization



- Applies standard rule-based optimization (constant folding, predicate-pushdown, projection pruning, null propagation, boolean expression simplification, etc)

```
tree.transform {
  case Add(Literal(c1), Literal(c2)) => Literal(c1+c2)
}
```



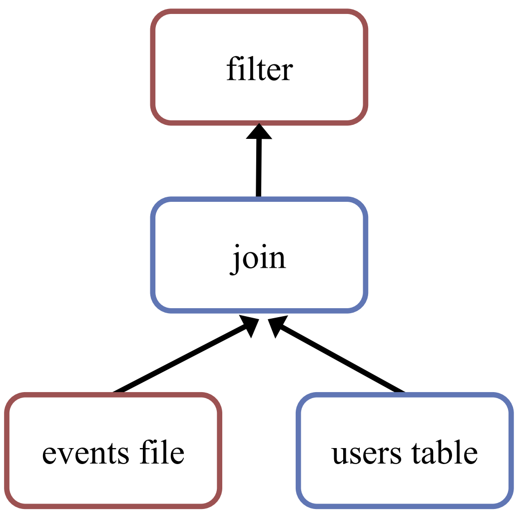
x + (1 + 2)

x + 3

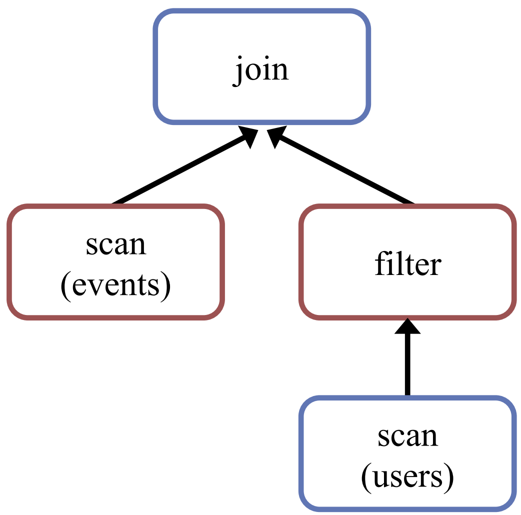
credits:
Matei Zaharia & Reynold Xin

```python
def add_demographics(events):
   u = sqlCtx.table("users")                     # Load partitioned Hive table
   events \
     .join(u, events.user_id == u.user_id) \     # Join on user_id
     .withColumn("city", zipToCity(df.zip))      # Run udf to add city column
events = add_demographics(sqlCtx.load("/data/events", "parquet"))
training_data = events.where(events.city == "Melbourne").select(events.timestamp).collect()
```

Logical Plan

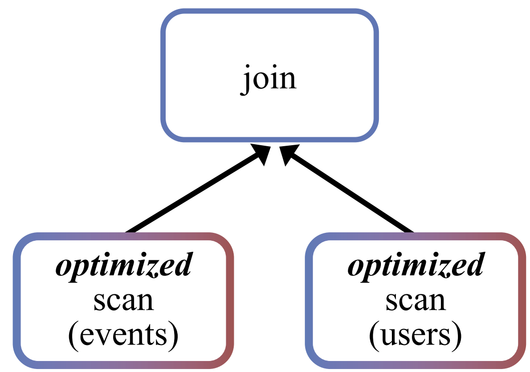


Physical Plan



Physical Plan

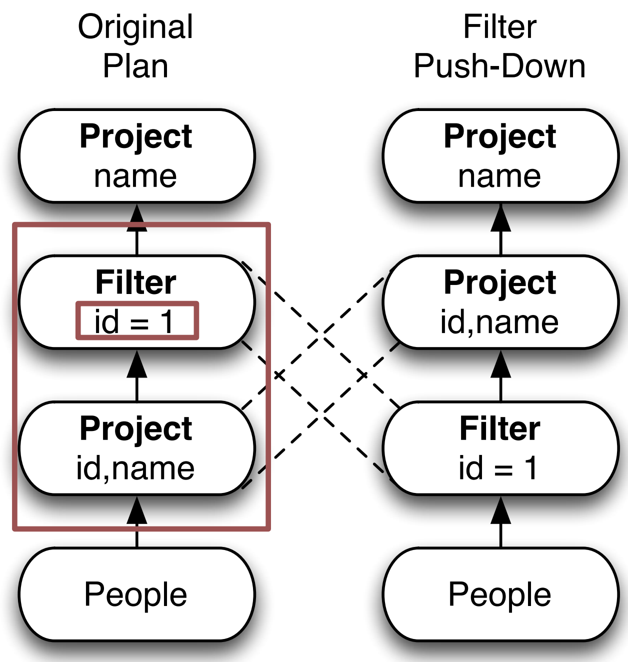
with Predicate Pushdown
and Column Pruning



credits:
Matei Zaharia & Reynold Xin

An Example Catalyst Transformation

1. Find filters on top of projections.
2. Check that the filter can be evaluated without the result of the project.
3. If so, switch the operators.



credits:
Matei Zaharia & Reynold Xin

Other Spark Stack Projects

We will revisit **Spark SQL** in the **SQL on Big Data** lecture

- **Structured Streaming:** stateful, fault-tolerant stream
  - ```
    sc.twitterStream(...)
       .flatMap(_.getText.split(“ ”))
       .map(word => (word, 1))
       .reduceByWindow(“5s”, _ + _)
    ```
  - we will revisit **structured streaming** in the Data Streaming lecture

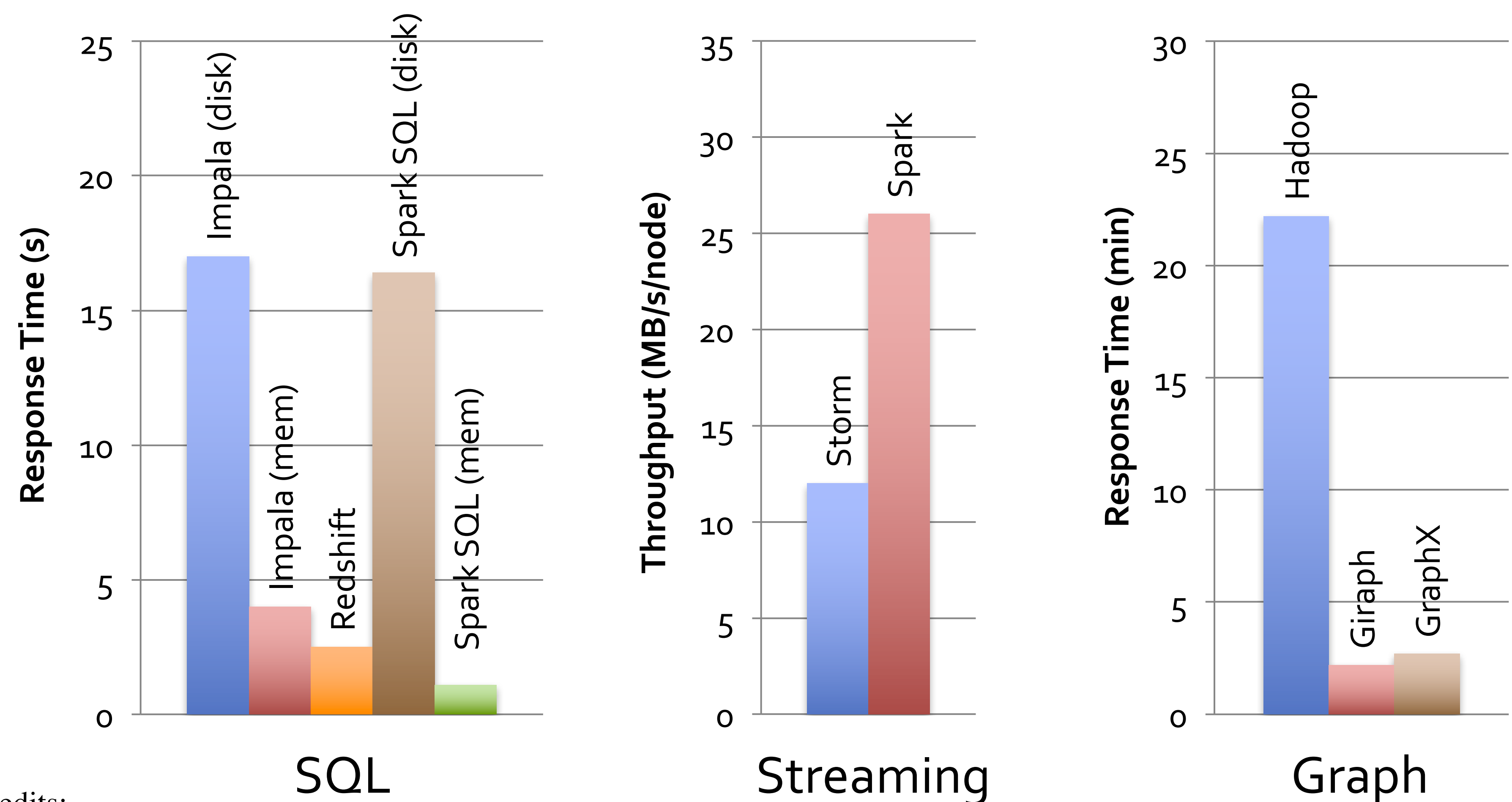
this lecture, still:

- **GraphX & GraphFrames:** graph-processing framework
- **MLlib:** Library of high-quality machine learning algorithms

credits:
Matei Zaharia & Xiangrui Meng
    ```

Centrum Wiskunde & Informatica

Performance



credits:
Matei Zaharia & Xiangrui Meng

www.cwi.nl/~boncz/bads

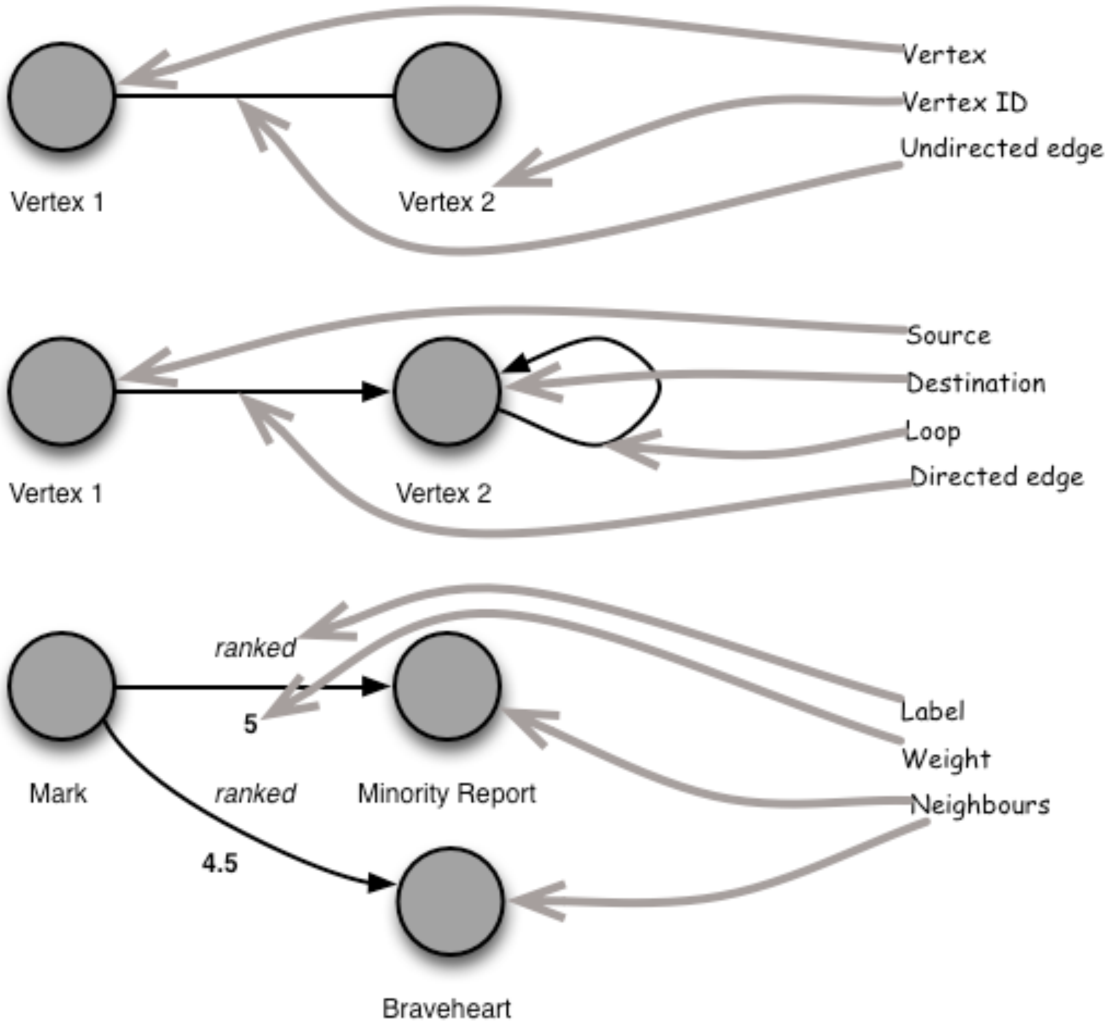
Centrum Wiskunde & Informatica

THINK AS A VERTEX: GRAPH PROGRAMMING

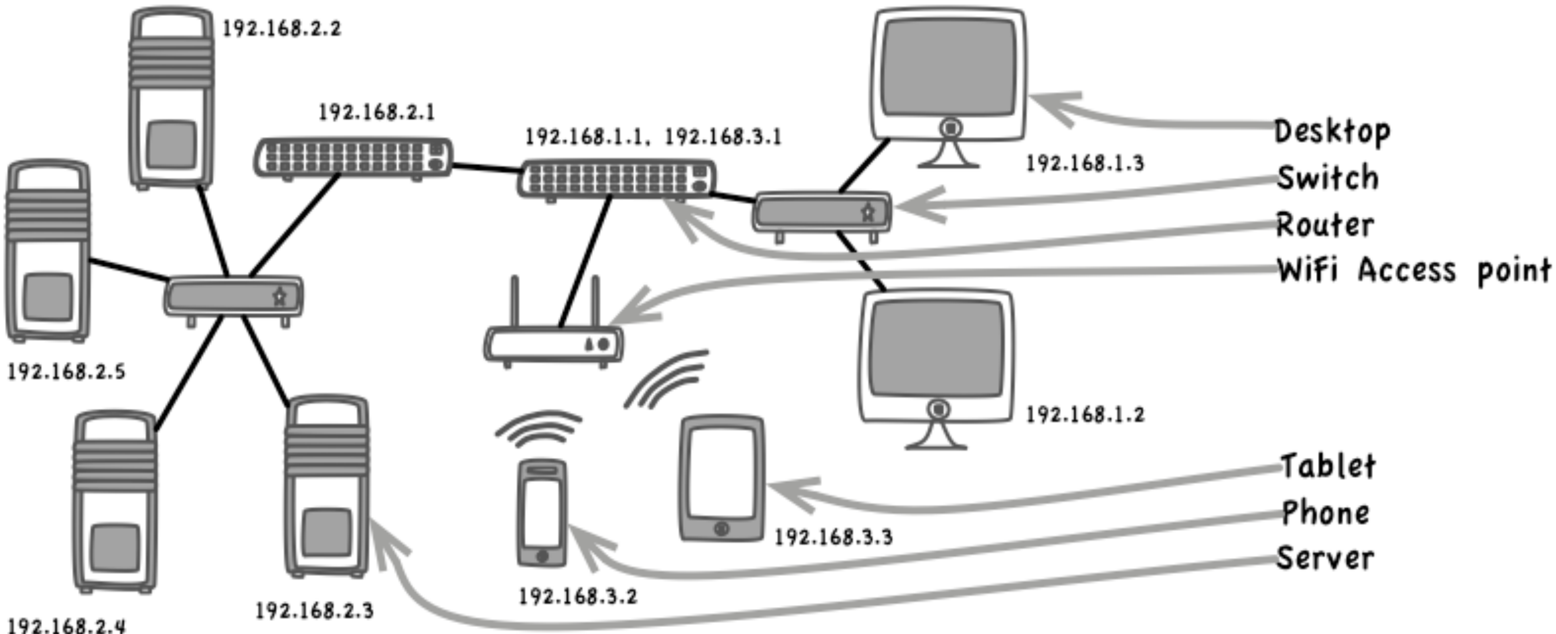
PREGEL, GIRAPH, GRAPHX

Centrum Wiskunde & Informatica

Graphs are Simple

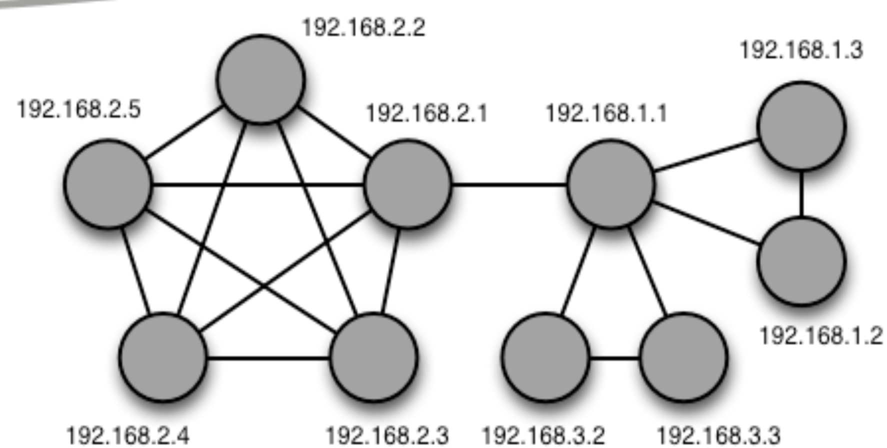


A Computer Network

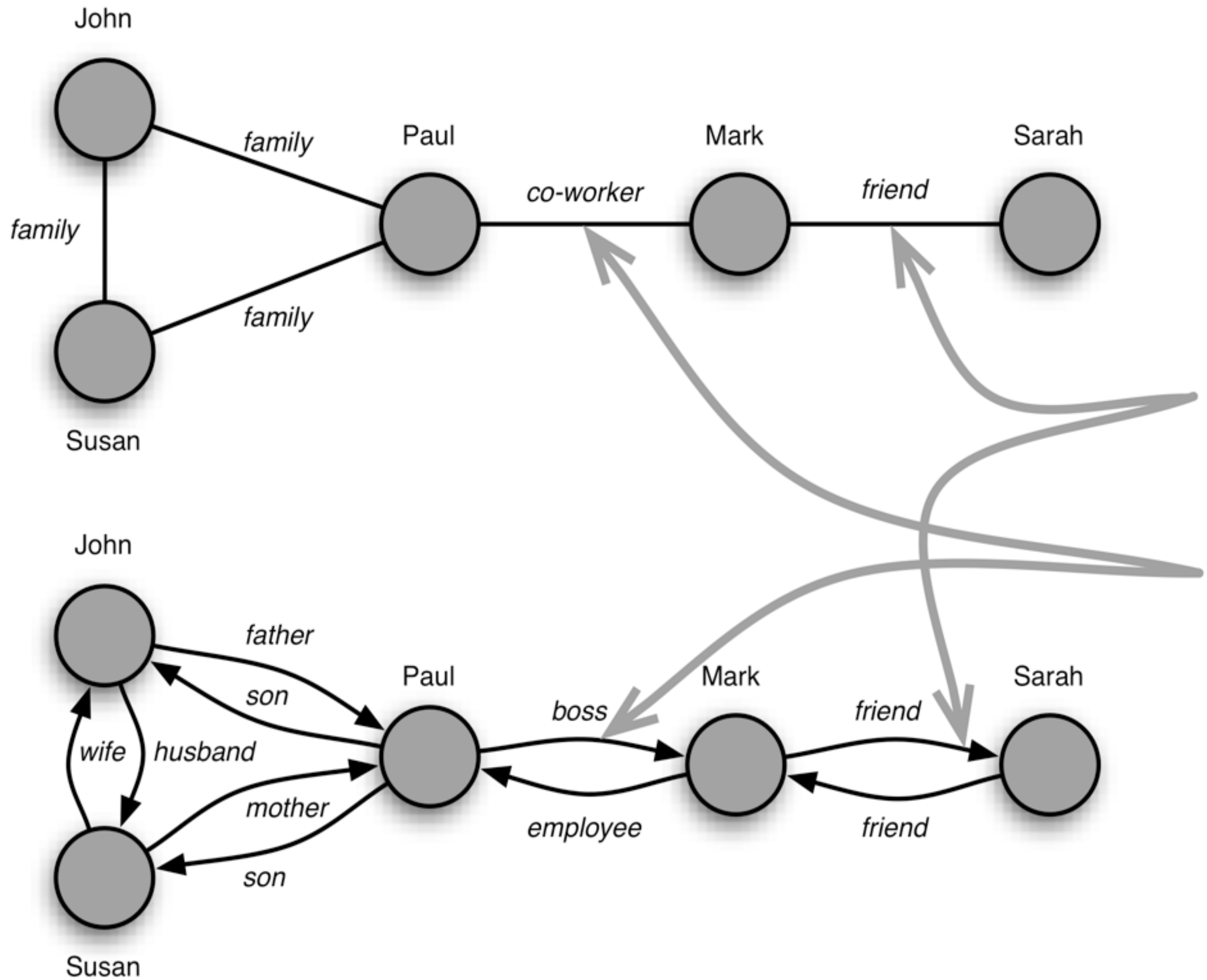


Note

1. There are three networks: servers, desktops and mobile.
2. They are connected through two routers/firewalls.
3. We ignored the switches and the access point in the graph.
4. Router 192.168.1.1 has two interfaces but we used one as vertex ID.



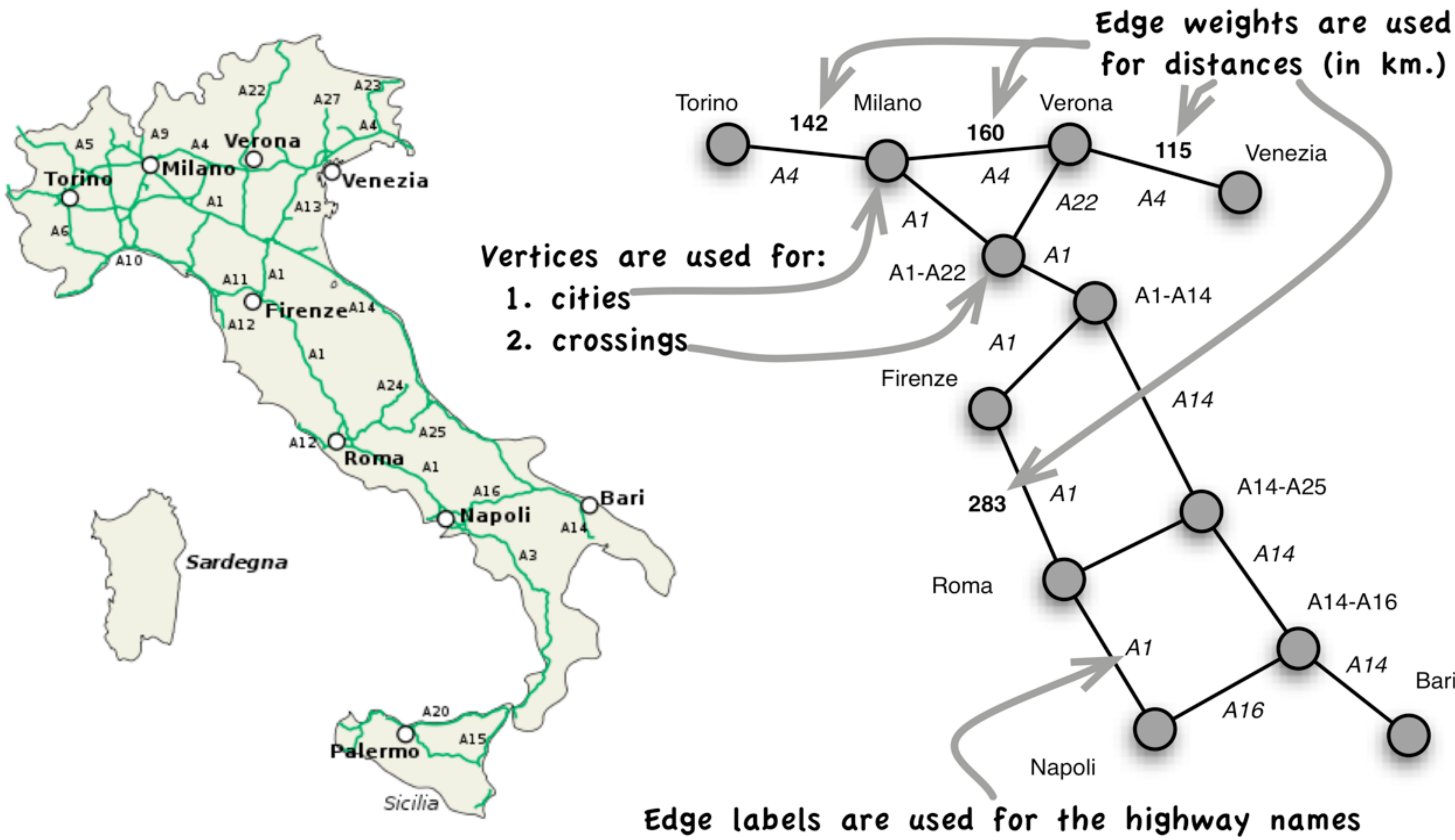
A Social Network



Note

1. A symmetric relationship is substituted by two directed edges.
2. A relationship does not have to be substituted by two edges, but e.g. by a more specific one.

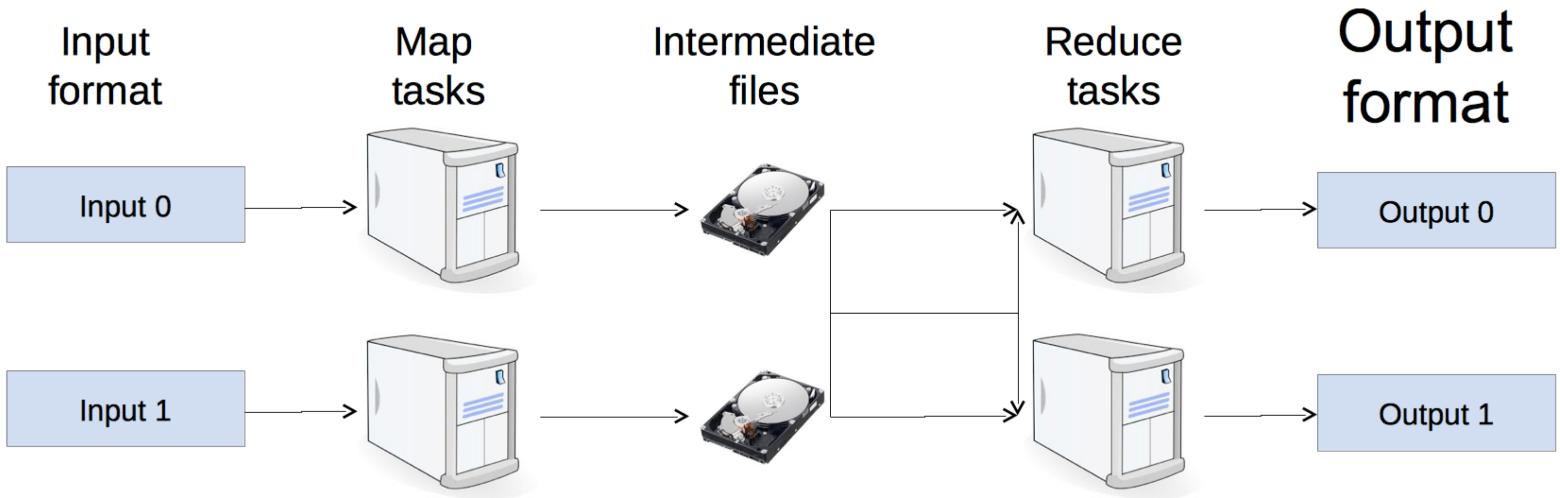
Maps are Graphs as well



PageRank in MapReduce

- Record: < v_i, pr, [v_j, ..., v_k] >
- Mapper: emits < v_j, pr / #neighbours >
- Reducer: sums the partial values

MapReduce DataFlow



- Each job is executed N times
- Job bootstrap
- Mappers send PR values and structure
- Extensive IO at input, shuffle & sort, output

Pregel: computational model

- Based on Bulk Synchronous Parallel (BSP)
  - Computational units encoded in a directed graph
  - Computation proceeds in a series of supersteps
  - Message passing architecture
- Each vertex, at each superstep:
  - Receives messages directed at it from previous superstep
  - Executes a user-defined function (modifying state)
  - Emits messages to other vertices (for the next superstep)
- Termination:
  - A vertex can choose to deactivate itself
  - Is “woken up” if new messages received
  - Computation halts when all vertices are inactive

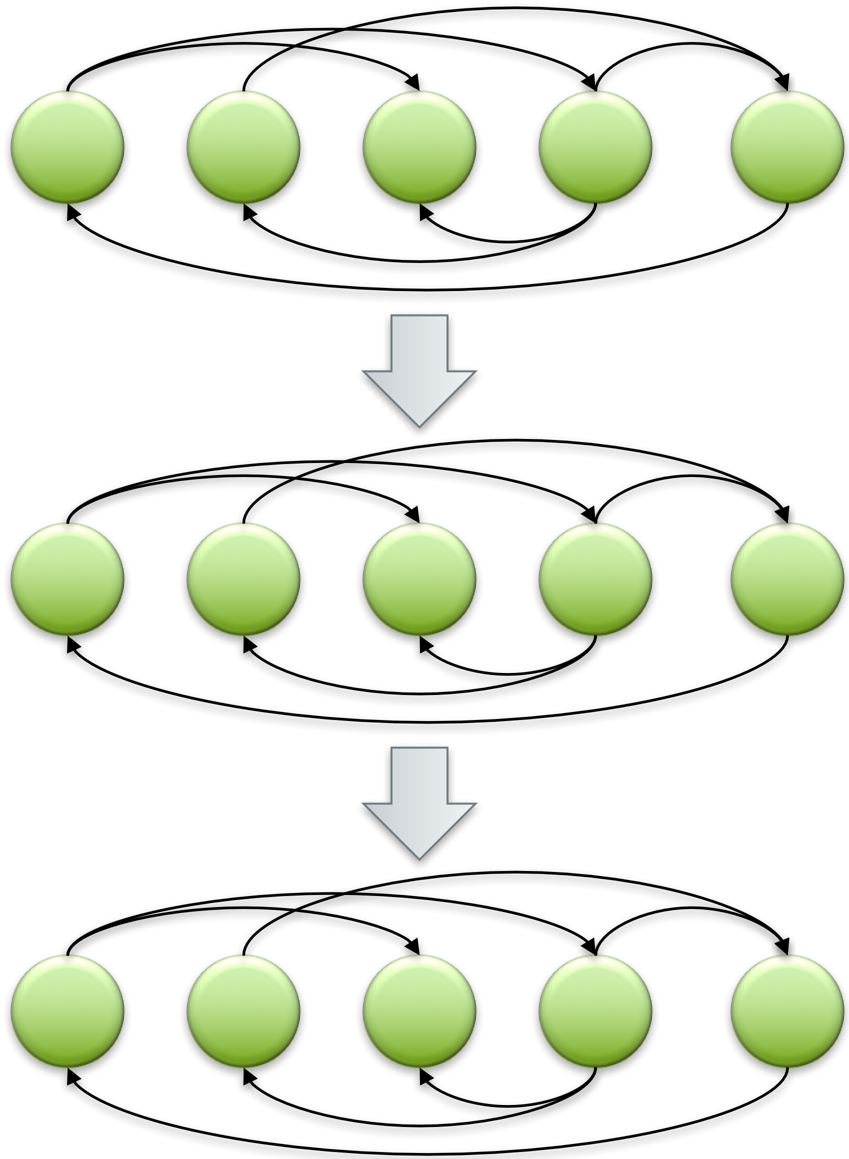
Source: Malewicz et al. (2010) Pregel: A System for Large-Scale Graph Processing. SIGMOD.

Pregel

superstep *t*

superstep *t+1*

superstep *t+2*



Source: Malewicz et al. (2010) Pregel: A System for Large-Scale Graph Processing. SIGMOD.

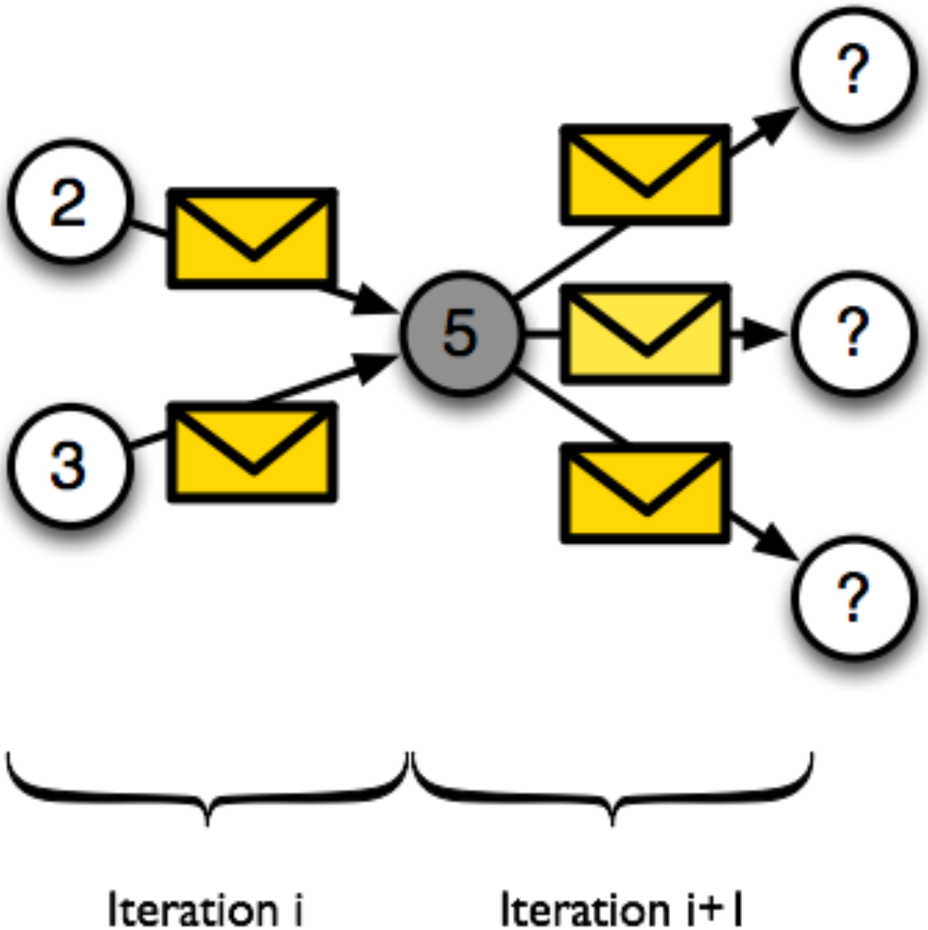
Pregel: implementation

- Master-Slave architecture
  - Vertices are hash partitioned (by default) and assigned to workers
  - Everything happens in memory
- Processing cycle
  - Master tells all workers to advance a single superstep
  - Worker delivers messages from previous superstep, executing vertex computation
  - Messages sent asynchronously (in batches)
  - Worker notifies master of number of active vertices
- Fault tolerance
  - Checkpointing
  - Heartbeat/revert

Source: Malewicz et al. (2010) Pregel: A System for Large-Scale Graph Processing. SIGMOD.

Centrum Wiskunde & Informatica

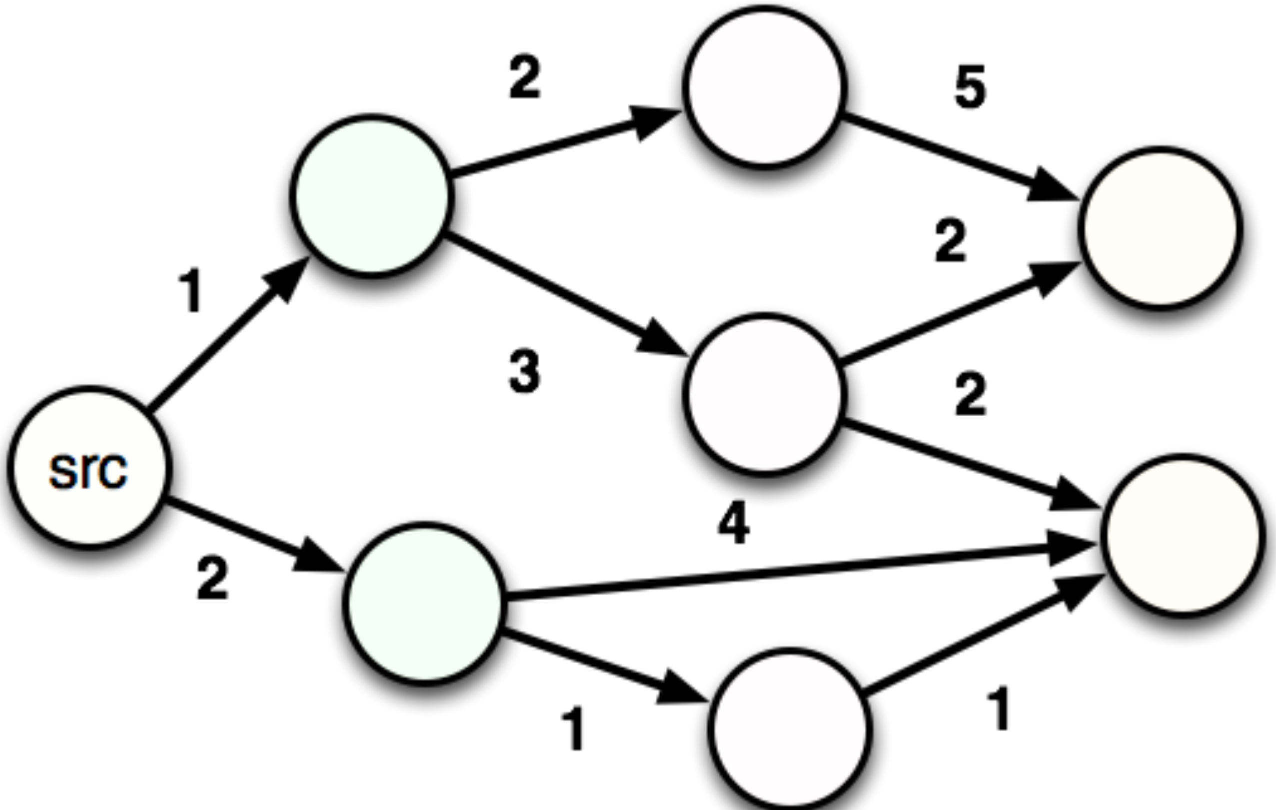
Vertex-centric API



www.cwi.nl/~boncz/bads

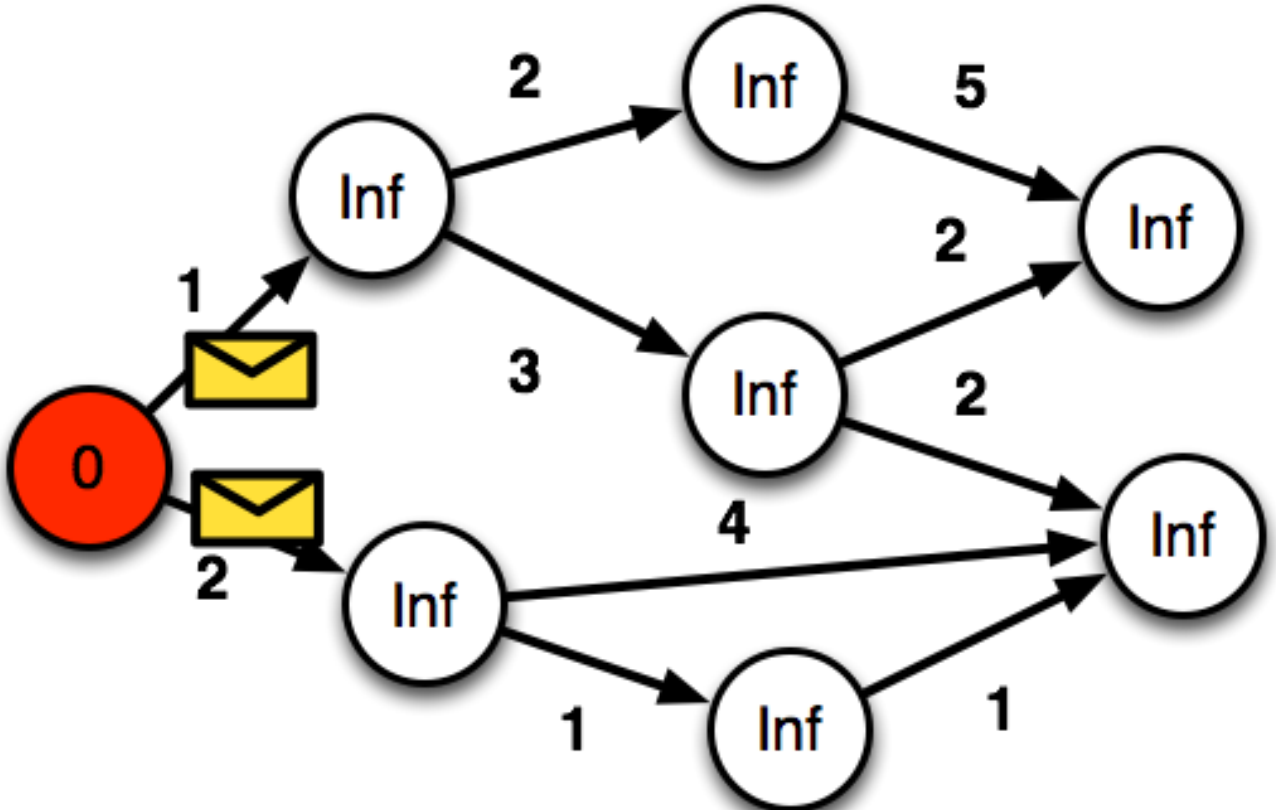
Centrum Wiskunde & Informatica

Shortest Paths

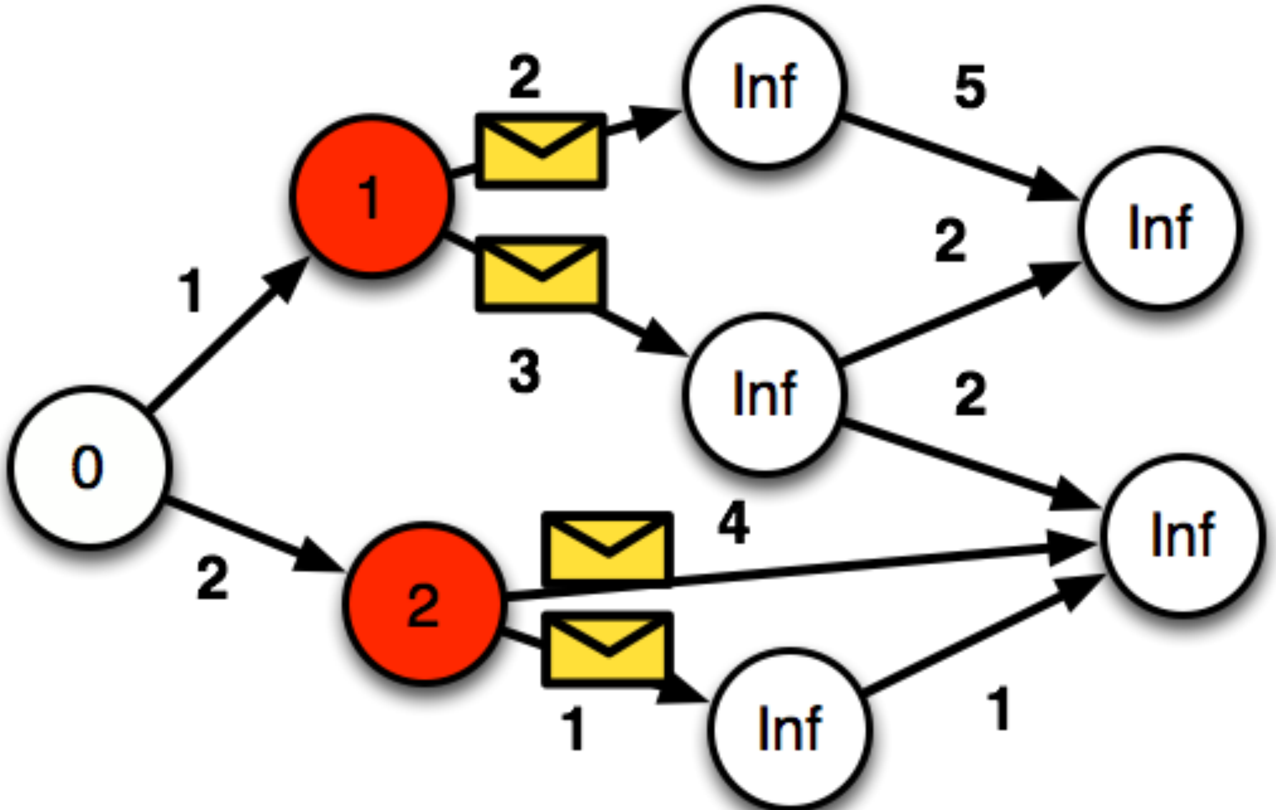


www.cwi.nl/~boncz/bads

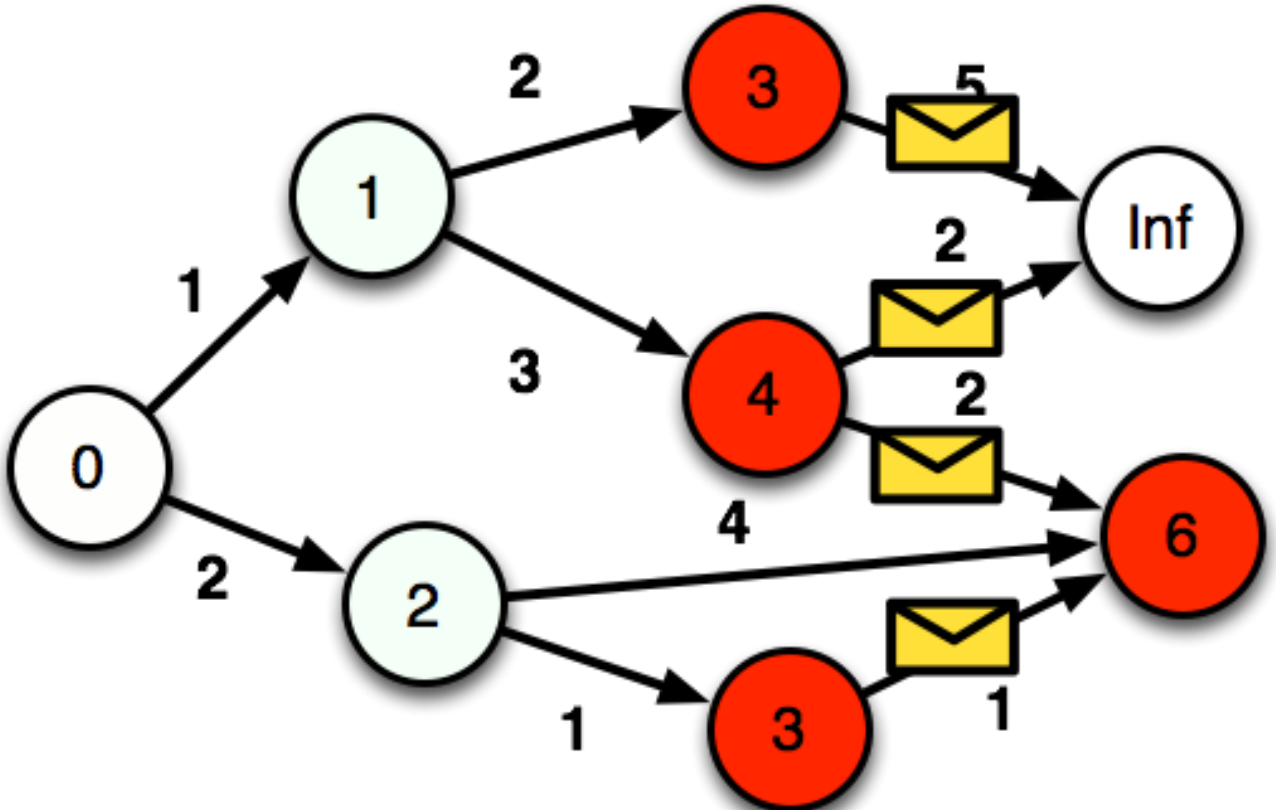
Shortest Paths



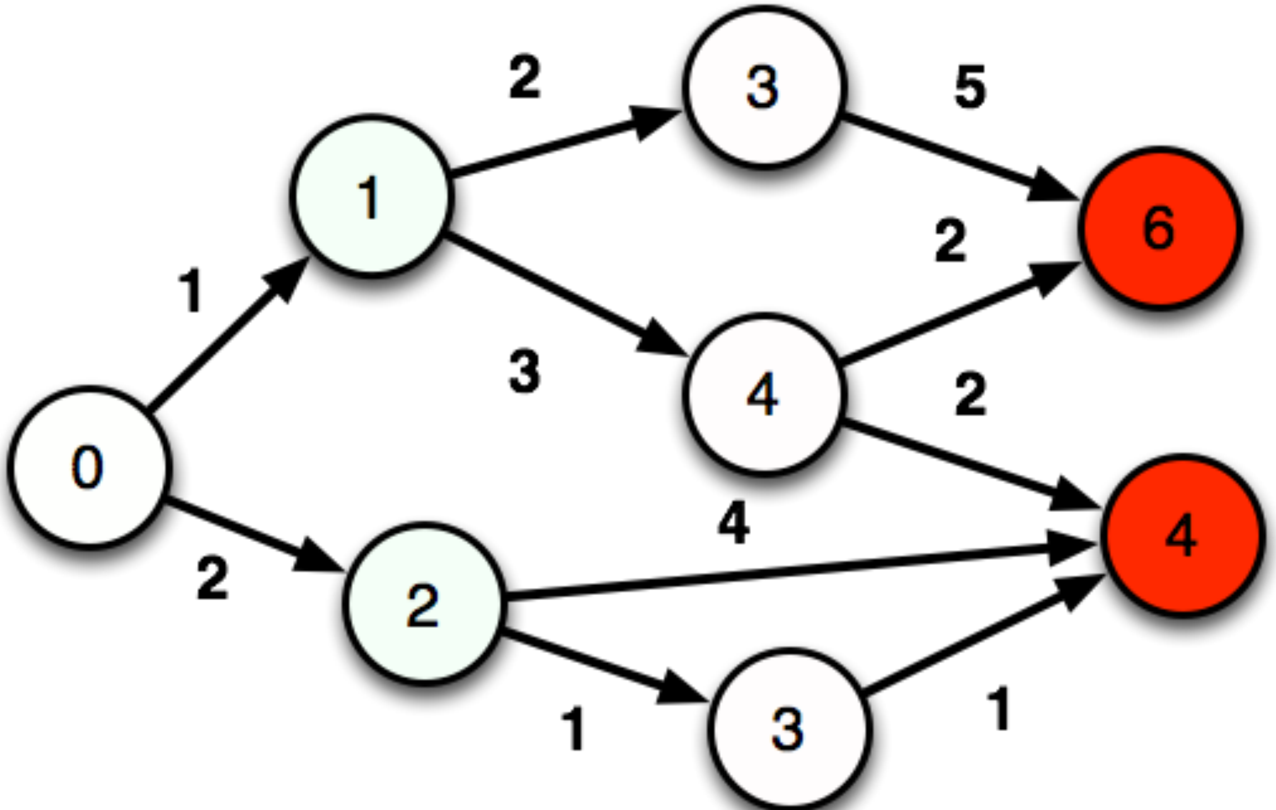
Shortest Paths



Shortest Paths



Shortest Paths



Shortest Paths

```
def compute(vertex, messages):
    minValue = Inf    # float('Inf')
    for m in messages:
        minValue = min(minValue, m)
    if minValue < vertex.getValue():
        vertex.setValue(minValue)
        for edge in vertex.getEdges():
            message = minValue + edge.getValue()
            sendMessage(edge.getTargetId(), message)
    vertex.voteToHalt()
```

Graph Programming Frameworks

- Google Pregel
  - Non open-source, probably not used much anymore
- Apache Giraph
  - Developed and used by Facebook
- Apache Flink
  - Gelly API
- Apache Spark
  - GraphX API

  +

  - DataFrames API

Source: Malewicz et al. (2010) Pregel: A System for Large-Scale Graph Processing. SIGMOD.

SPARK MLLIB

credits:
Matei Zaharia & Xiangrui Meng

What is MLLIB?

MLlib is a Spark subproject providing machine learning primitives:

- initial contribution from AMPLab, UC Berkeley
- shipped with Spark since version 0.8

credits:
Matei Zaharia & Xiangrui Meng

What is MLLIB?

Algorithms:

- **classification**: logistic regression, linear support vector machine (SVM), naive Bayes
- **regression**: generalized linear regression (GLM)
- **collaborative filtering**: alternating least squares (ALS)
- **clustering**: k-means
- **decomposition**: singular value decomposition (SVD), principal component analysis (PCA)

credits:
Matei Zaharia & Xiangrui Meng

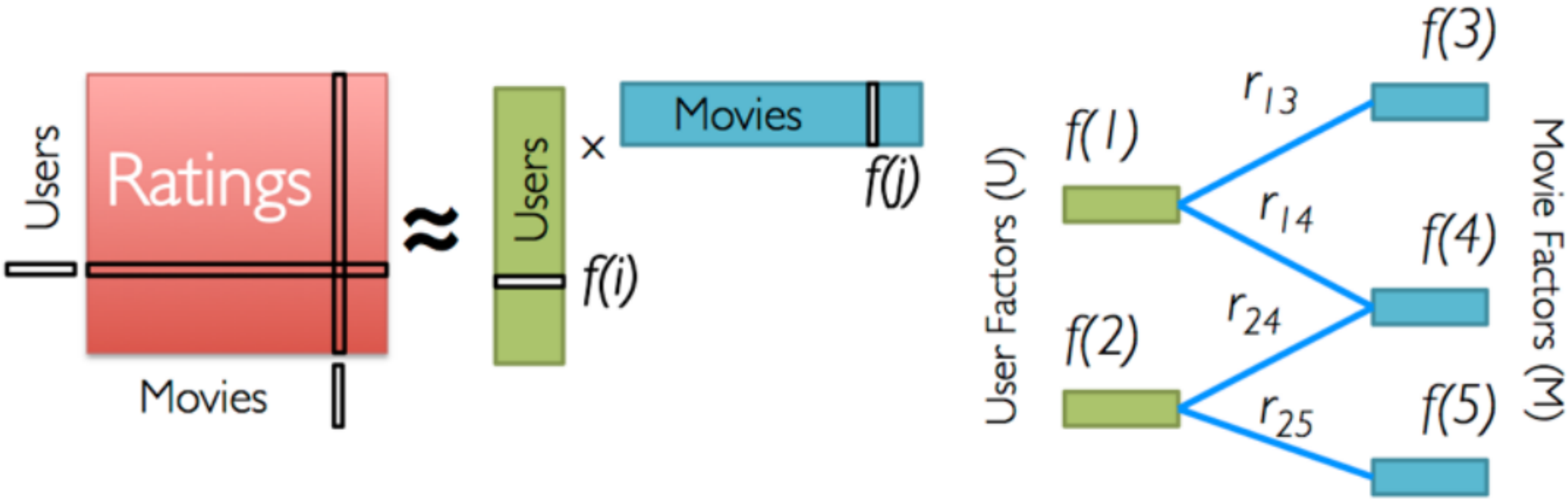
Collaborative Filtering

★	★★★★	?
★	★★★	★★
★★★★	?	★
★	?	★★
?	★★★	★★
★★★★	★★	?

- Recover a rating matrix from a subset of its entries.

Centrum Wiskunde & Informatica

Alternating Least Squares (ALS)



Iterate:

$$f[i] = \arg\min_{w \in \mathbb{R}^d} \sum_{j \in \mathrm{Nbrs}(i)} \left(r_{ij} - w^T f[j]\right)^2 + \lambda ||w||_2^2$$

credits:
Matei Zaharia & Xiangrui Meng

Collaborative Filtering in Spark MLLIB

```
trainset =
  sc.textFile("s3n://bads-music-dataset/train_*.gz")
    .map(lambda l: l.split('\t'))
    .map(lambda l: Rating(int(l[0]), int(l[1]), int(l[2])))

model = ALS.train(trainset, rank=10, iterations=10) # train

testset =                                    # load testing set
  sc.textFile("s3n://bads-music-dataset/test_*.gz")
    .map(lambda l: l.split('\t'))
    .map(lambda l: Rating(int(l[0]), int(l[1]), int(l[2])))

# apply model to testing set (only first two cols) to predict
predictions =
   model.predictAll(testset.map(lambda p: (p[0], p[1])))
        .map(lambda r: ((r[0], r[1]), r[2]))
```

credits:
Matei Zaharia & Xiangrui Meng

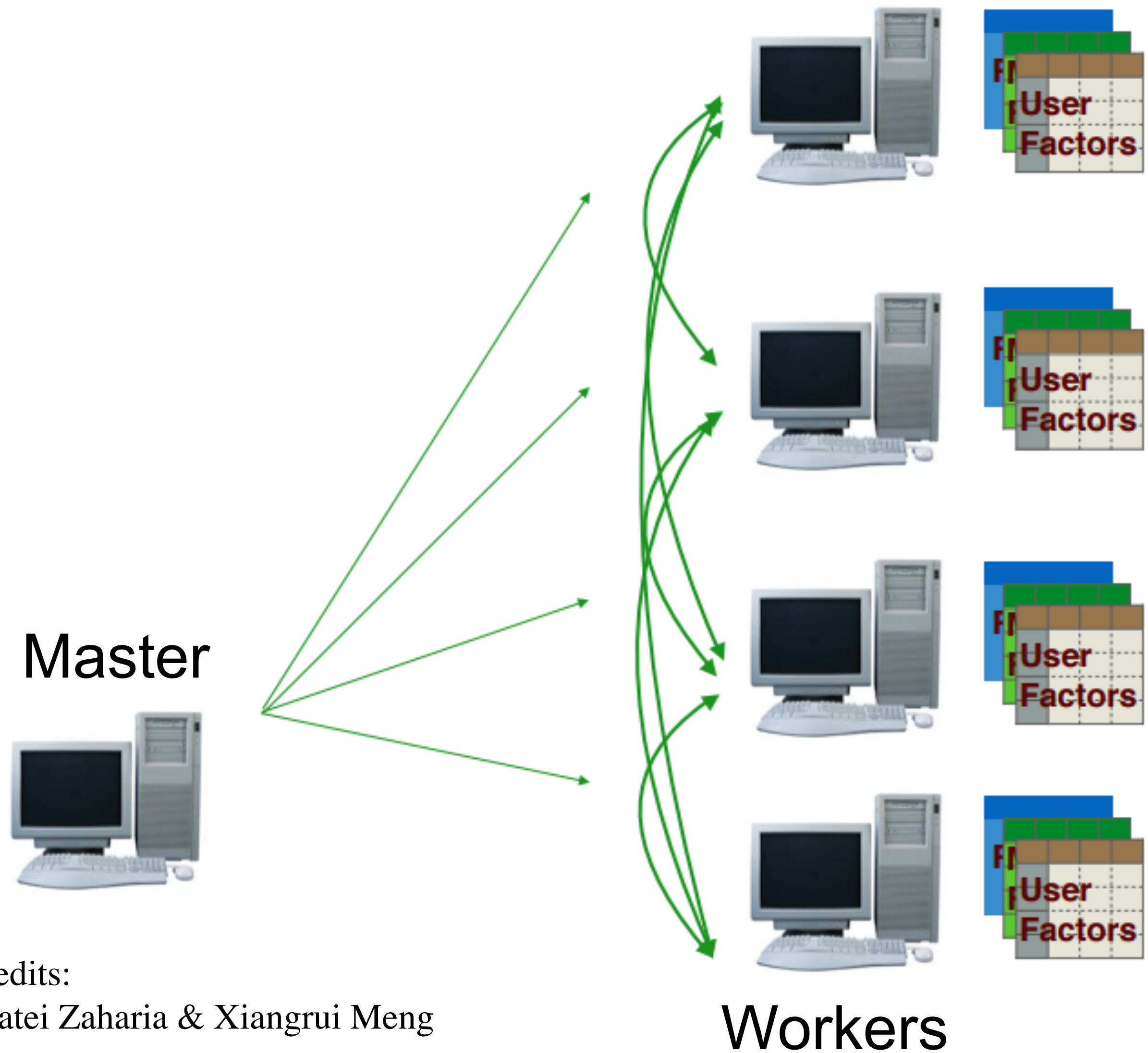
Spark MLLIB – ALS Performance

System	Wall-clock /me (seconds)
Matlab	15443
Mahout	4206
GraphLab	291
MLlib	481

- Dataset: Netflix data
- Cluster: 9 machines.
- MLlib is an order of magnitude faster than Mahout.
- MLlib is within factor of 2 of GraphLab.

credits:
Matei Zaharia & Xiangrui Meng

Spark Implementation of ALS



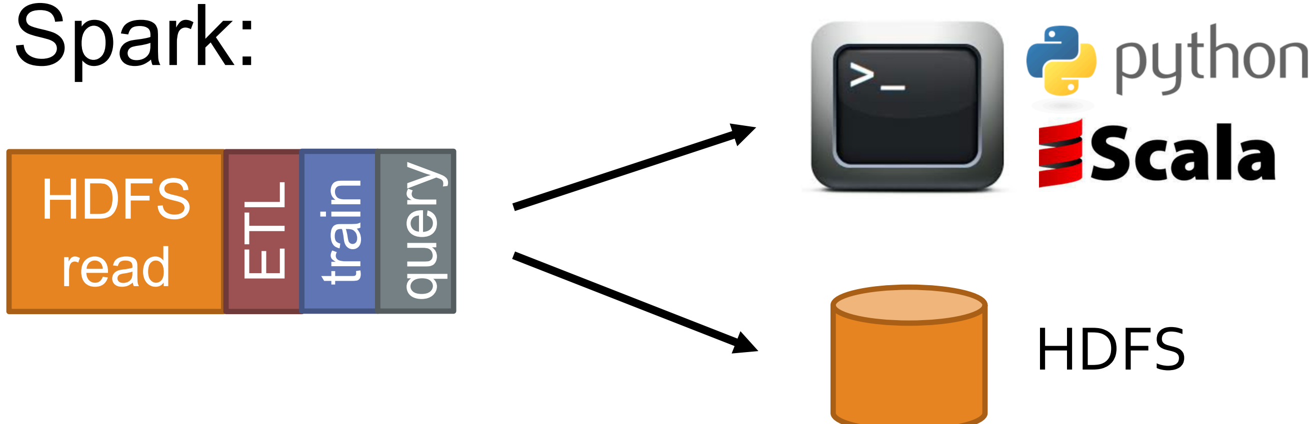
- Workers load data
- Models are instantiated at workers.
- At each iteration, models are shared via join between workers.
- Good scalability.
- Works on large datasets

credits:
Matei Zaharia & Xiangrui Meng

What it Means for Users

- Separate frameworks:

Spark:



credits:
Matei Zaharia & Xiangrui Meng

CWI
Centrum Wiskunde & Informatica

Summary

- The Spark Framework
  - Generalize Map(),Reduce() to a much larger set of operations
    - Join, filter, group-by, …➔ closer to database queries
  - High(er) performance (than MapReduce)
    - In-memory caching, catalyst query optimizer, JIT compilation, ..
    - RDDs ➔ DataFrames
- Spark GraphX: Graph Analytics (similar to Pregel/Giraph/Gelly)
  - Graph algorithms are often iterative (multi-job) ➔ a pain in MapReduce
  - Vertex-centric programming model:
    - Who to send messages to (halt if none)
    - How to compute new vertex state from messages
- Spark MLlib: scalable Machine learning
  - classification, regression, ALS, k-means, decomposition
  - Parallel DataFrame operations: allows analyzing data > RAM

CWI

Centrum Wiskunde & Informatica

Conclusion

- Big data analytics is evolving to include:
  - More **complex** analytics (e.g. machine learning)
  - More **interactive** ad-hoc queries
  - More **real-time** stream processing
- Spark is a fast platform that *unifies* these apps
- More info: spark-project.org

credits:
Matei Zaharia & Xiangrui Meng