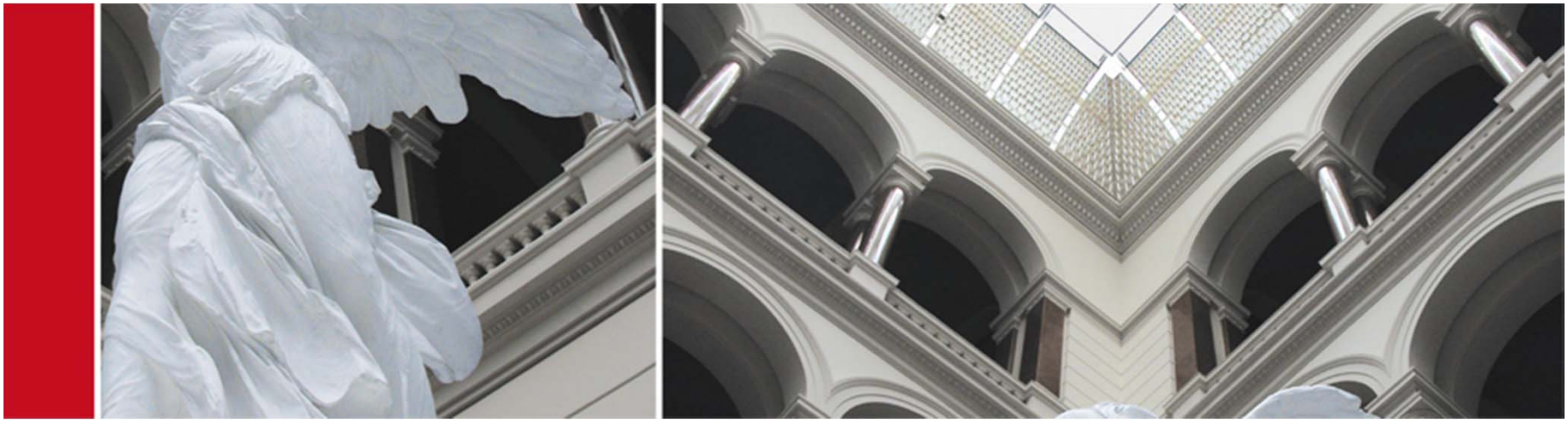
Technologie-Workshop „Big Data“
FZI Karlsruhe, 22. Juni 2015

Technische Universität Berlin



BBDC BERLIN BIG DATA CENTER

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

Introduction to Apache Flink

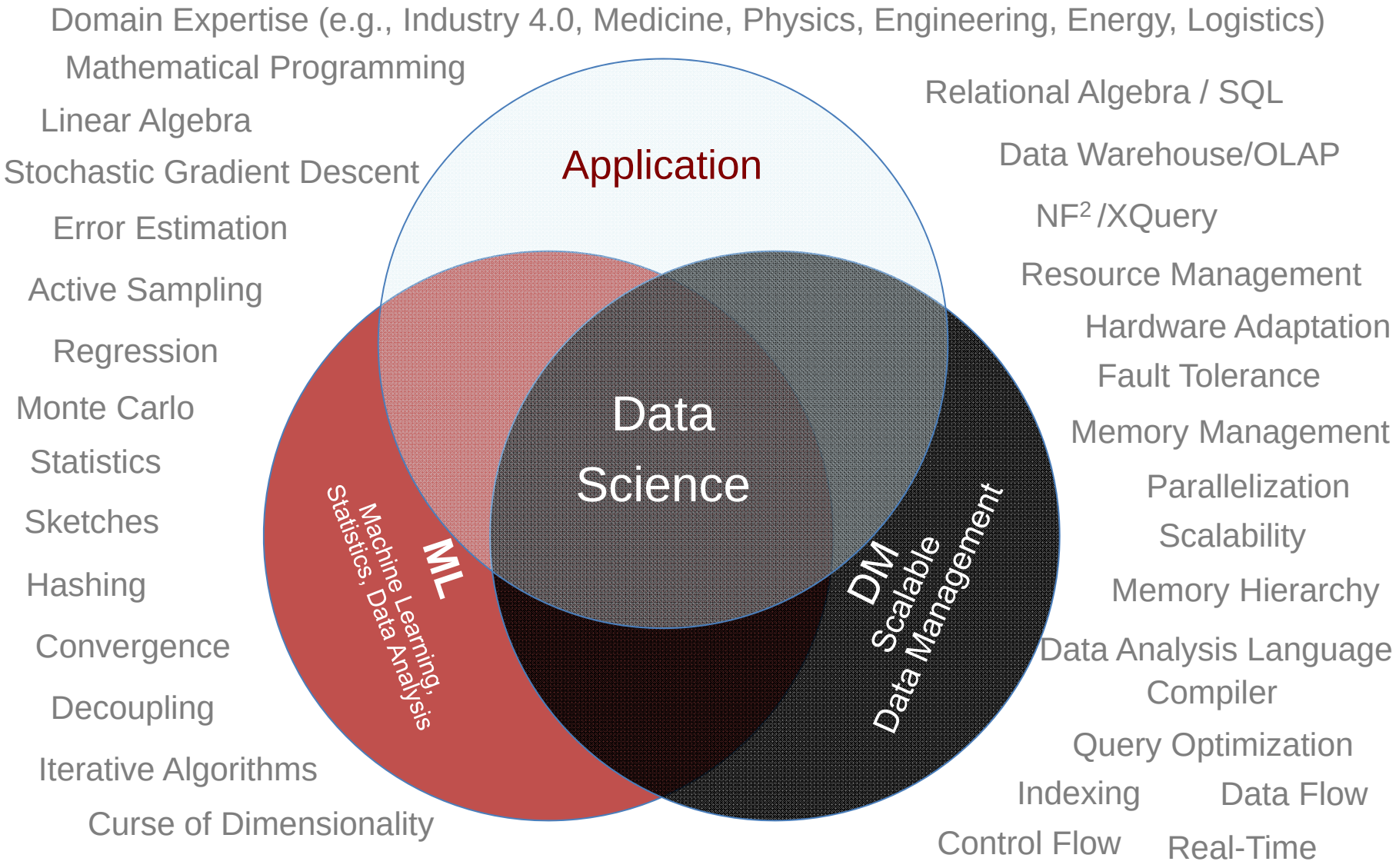
Christoph Boden | stv. Projektkoordinator Berlin Big Data Center (BBDC) | TU Berlin DIMA

Introducing the

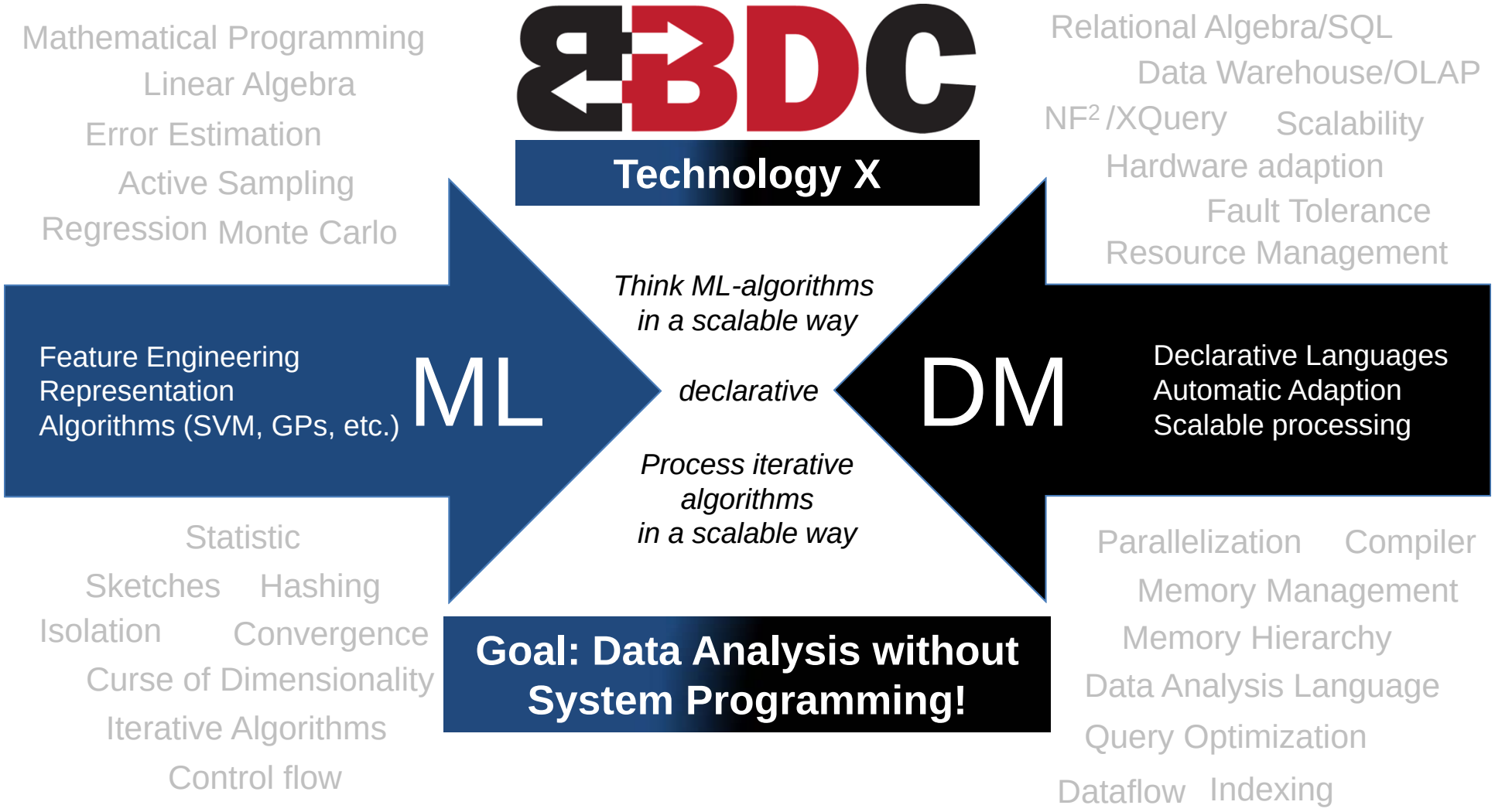
BERLIN BIG DATA CENTER

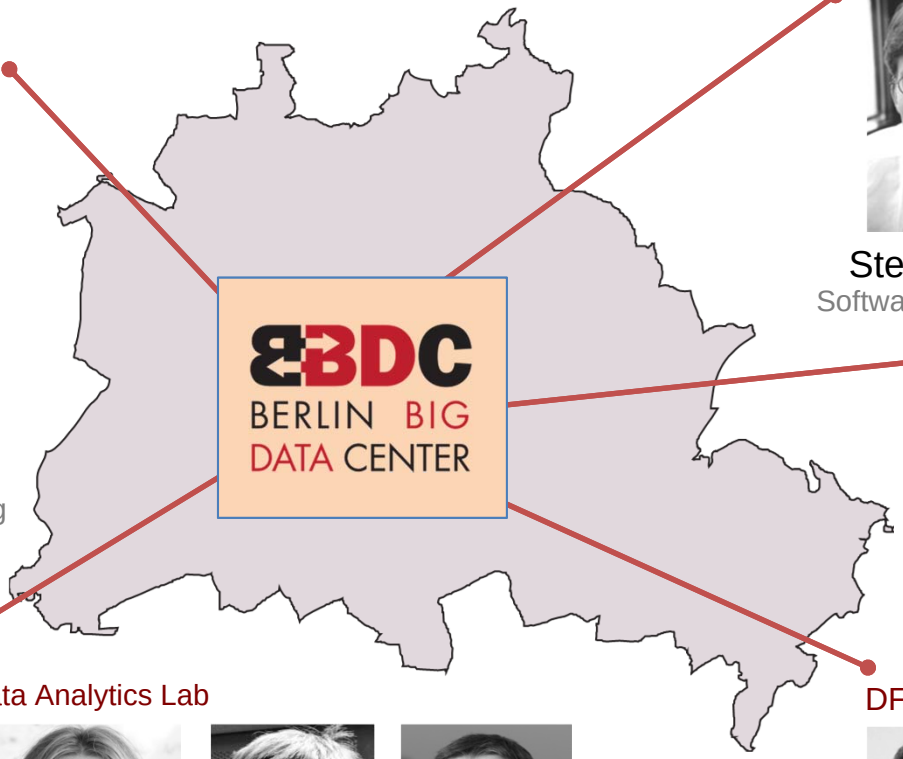
http://bbdc.berlin

“Data Scientist” – “Jack of All Trades!”



Machine Learning + Data Management = X





ZIB Berlin

Christof Schütte

Information-based Medicine

Alexander Reinefeld

File Systems, Supercomputing

Beuth Hochschule

Stefan Edlich

Software Engineering

Fritz-Haber-Institut, Max-Planck-Gesellschaft

Matthias Scheffler

Material Science

Technische Universität Berlin, Data Analytics Lab

Volker Markl

Data Management

Klaus R. Müller

Machine Learning

Anja Feldmann

Computer Networks

Odej Kao

Distributed Systems

Thomas Wiegand

Video Mining

DFKI

Hans Uszkoreit

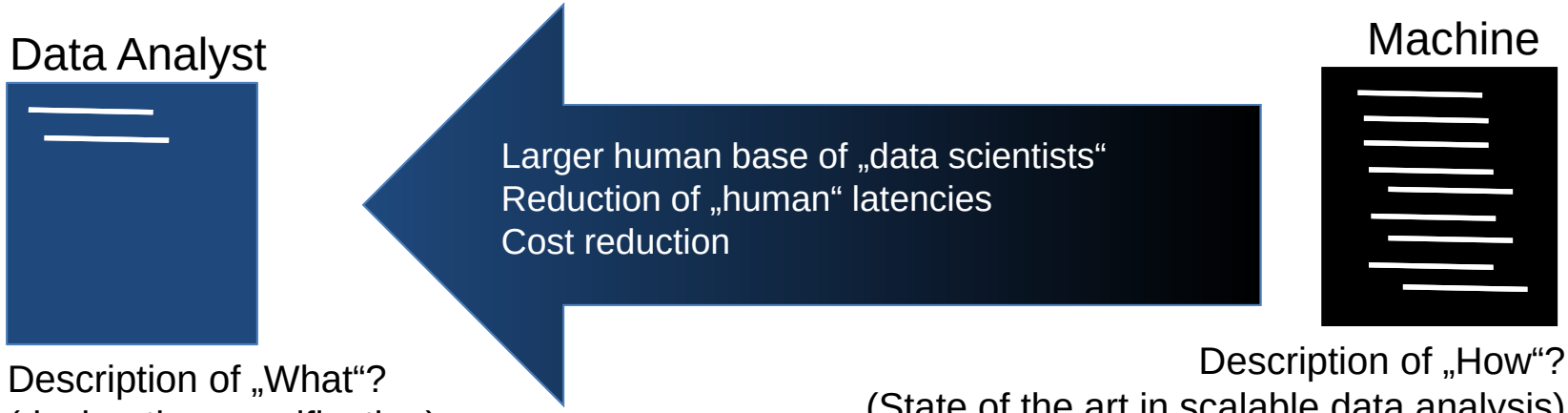
Language Technology

SPONSORED BY THE

Federal Ministry of Education and Research

http://bbdc.berlin

X = Big Data Analytics – System Programming!
(„What“, not „How“)



Technology X

Think ML-algorithms in a scalable way

Analysis of “data in motion”

Multimodal analysis

Numerical stability

Declarative specification

Automatic optimization, parallelization and hardware adaption of dataflow and control flow with user-defined functions, iterations and distributed state

Scalable algorithms and debugging

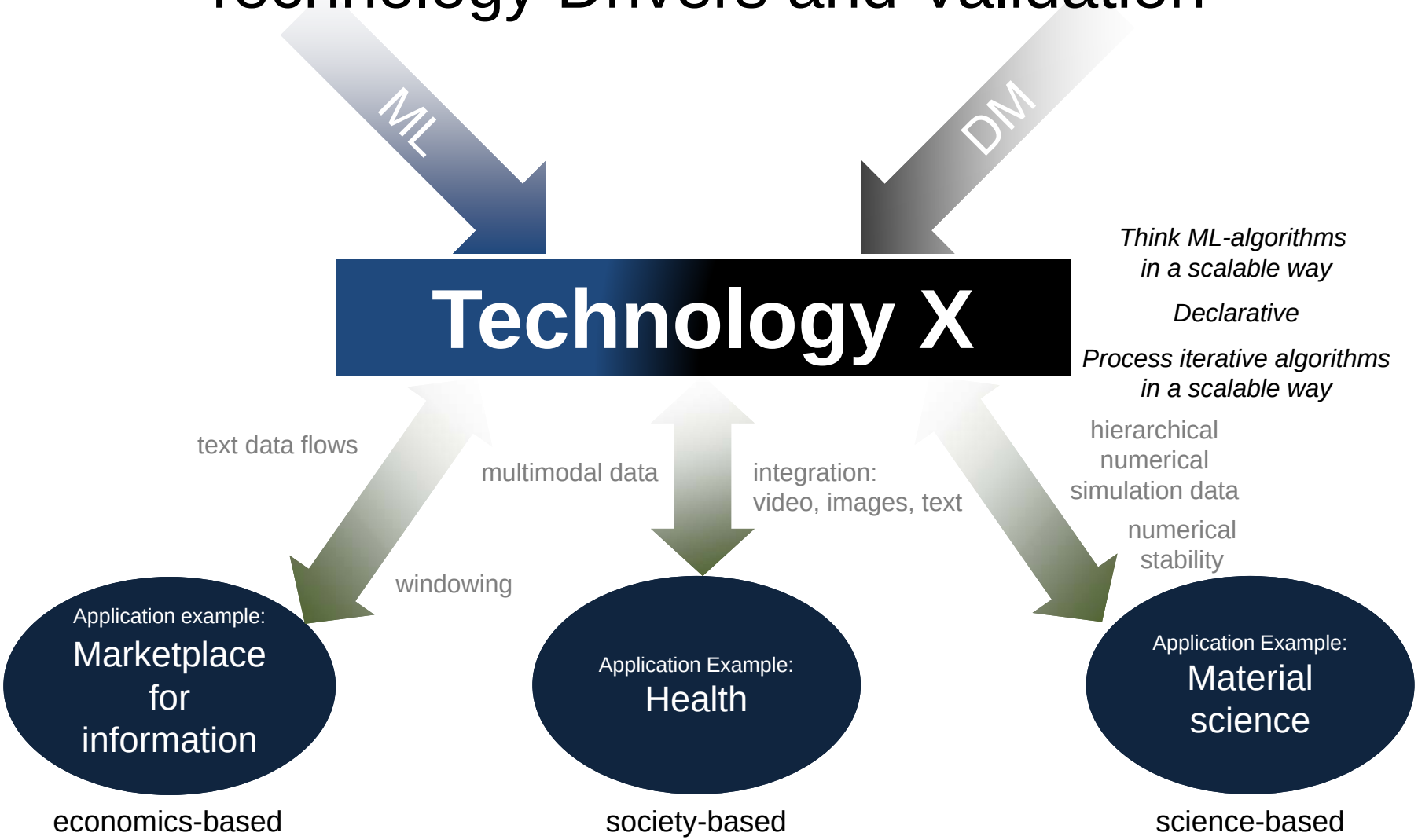
Algorithmic fault tolerance

Consistent intermediate results

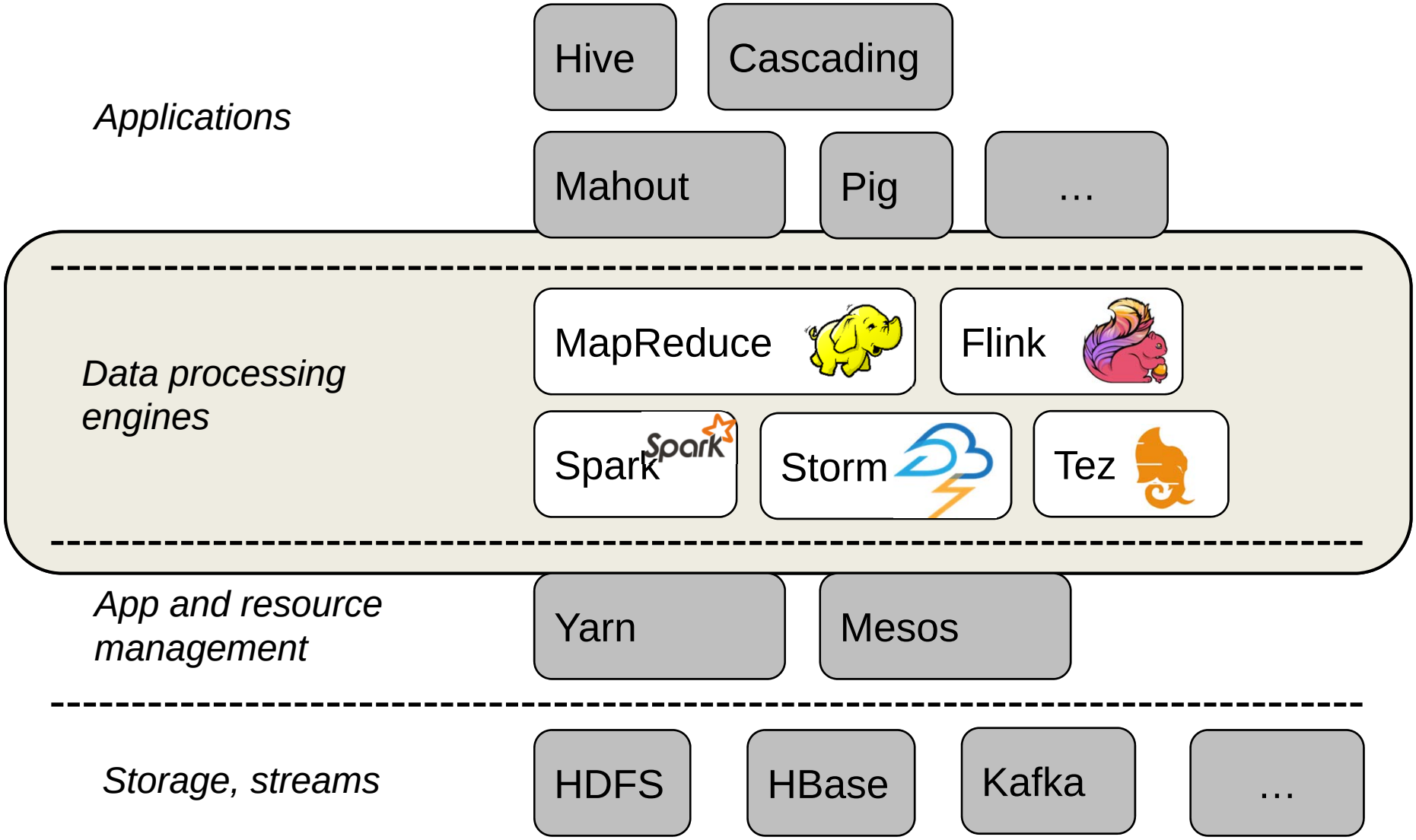
Software-defined networking

process Iterative algorithms in a scalable way

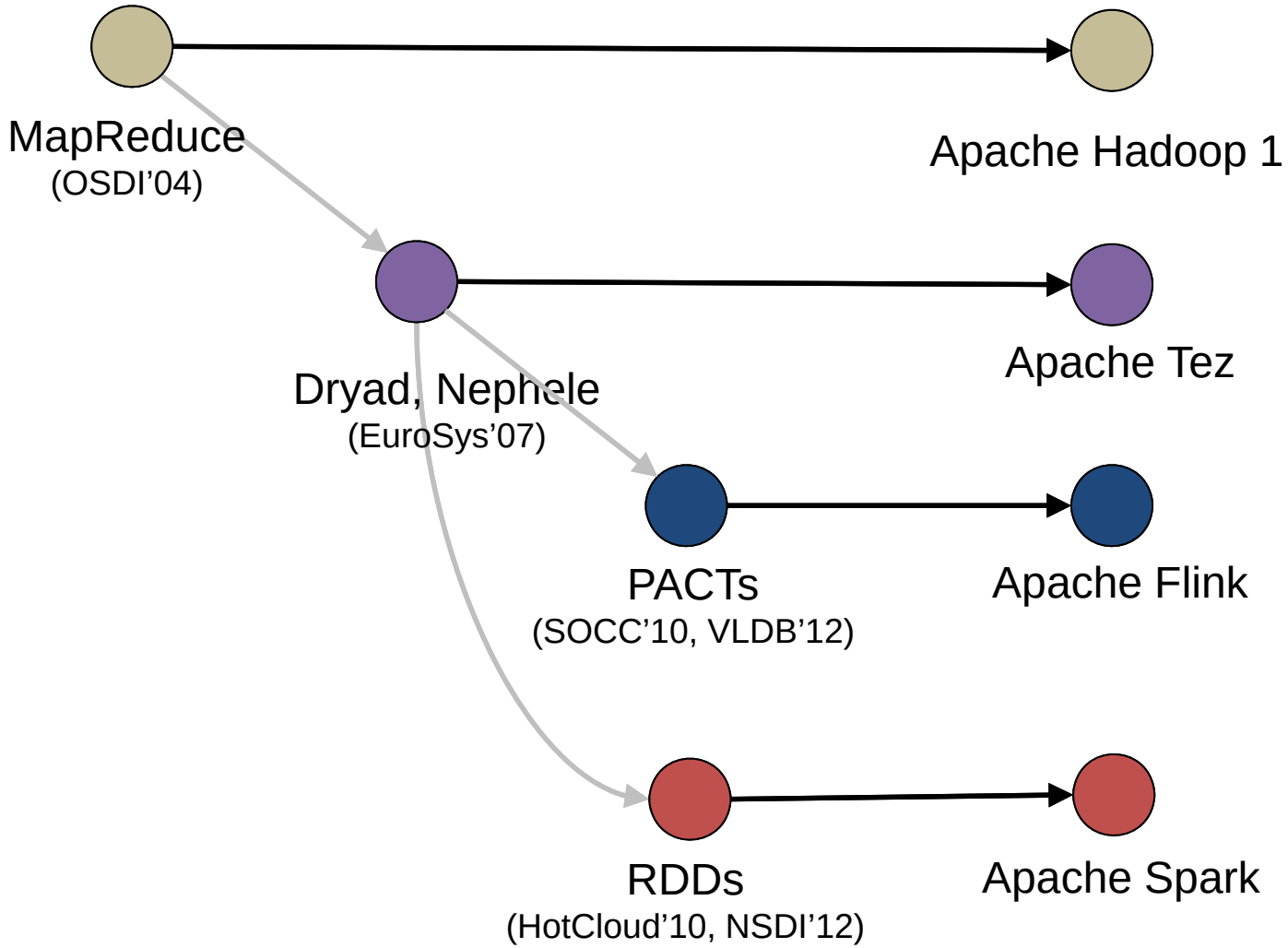
Application Examples: Technology Drivers and Validation



Open source data infrastructure



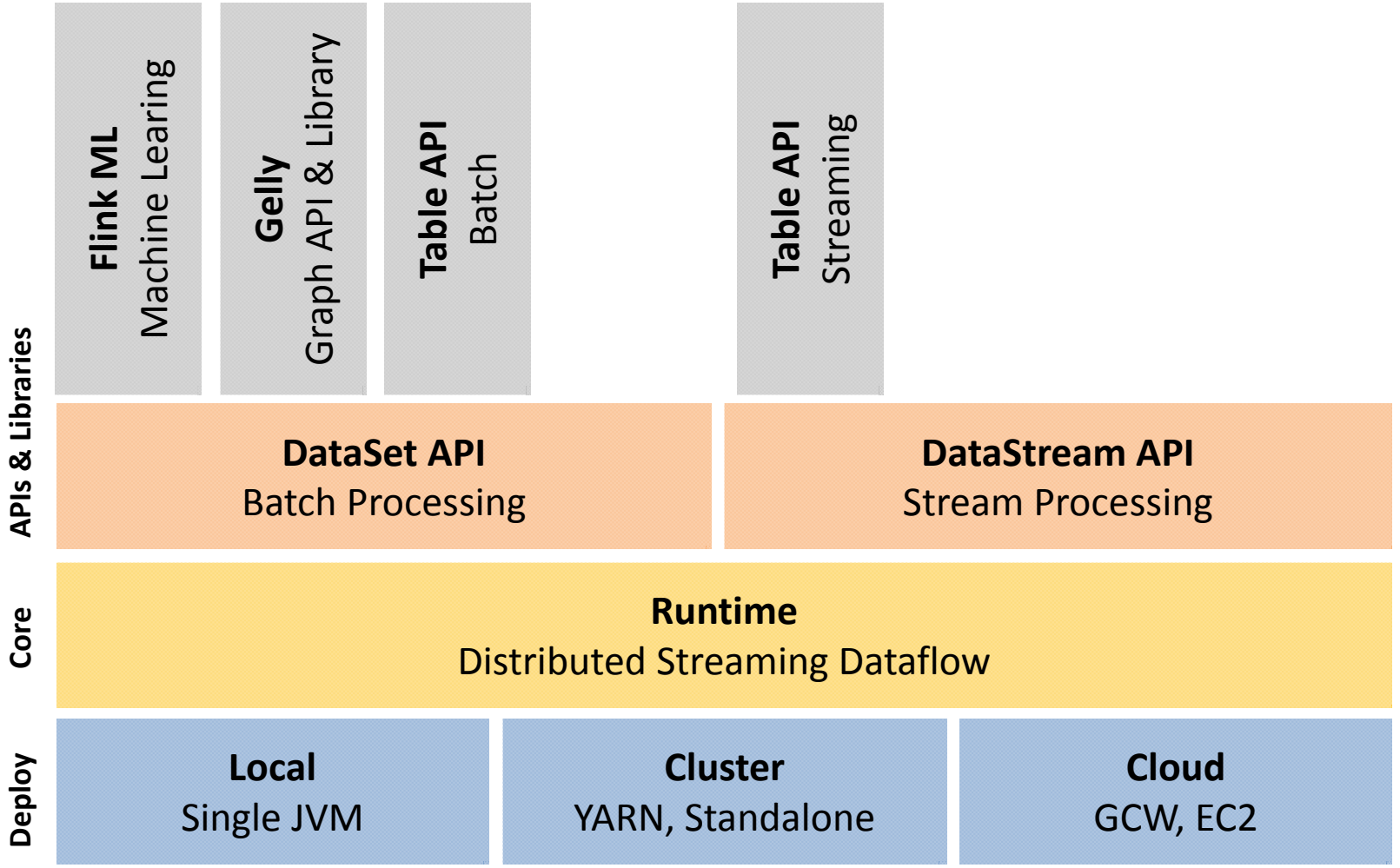
Engine paradigms & systems



Engine comparison

API	MapReduce on k/v pairs	Transformations on k/v pair collections	Iterative transformations on collections
Paradigm	MapReduce	RDD	Cyclic dataflows
Optimization	none	Optimization of SQL queries	Optimization in all APIs
Execution	Batch sorting	Batch with memory pinning	Stream with out-of-core algorithms

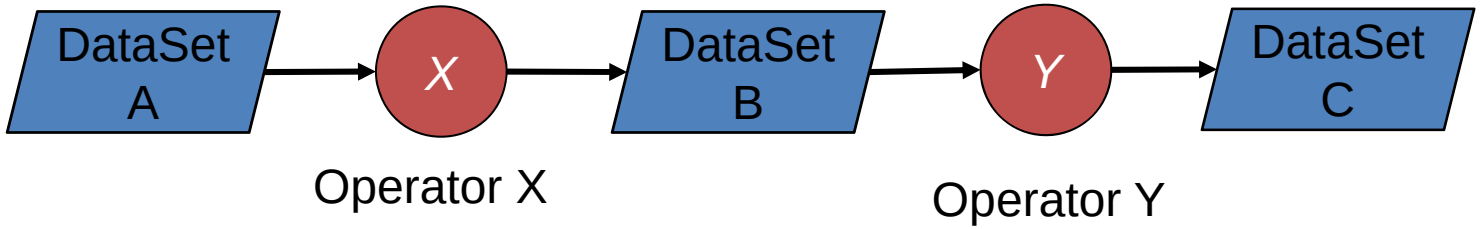
APACHE FLINK



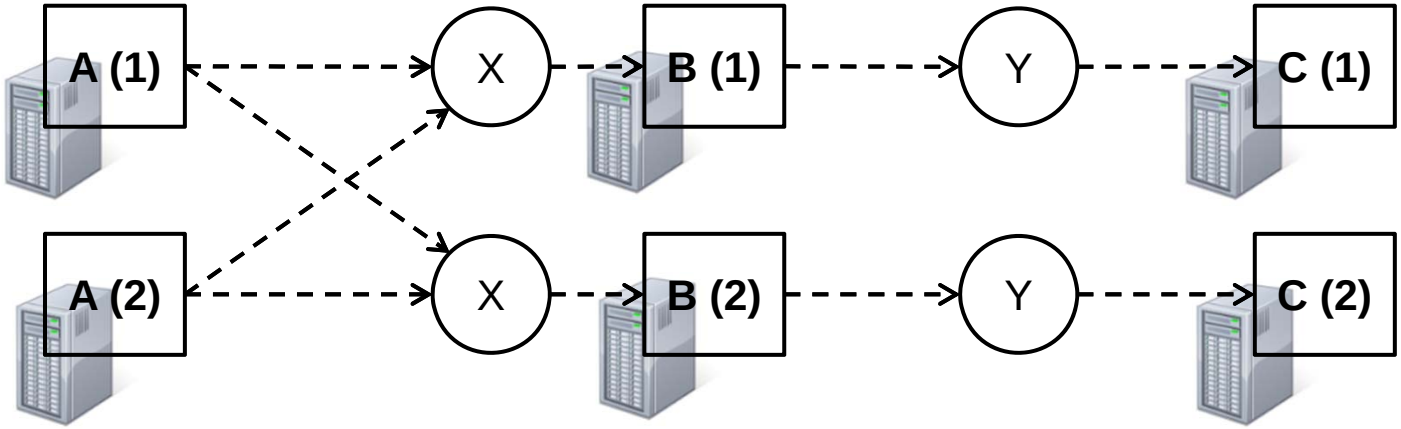
An open source platform for scalable batch and stream data processing.

Data sets and operators

Program

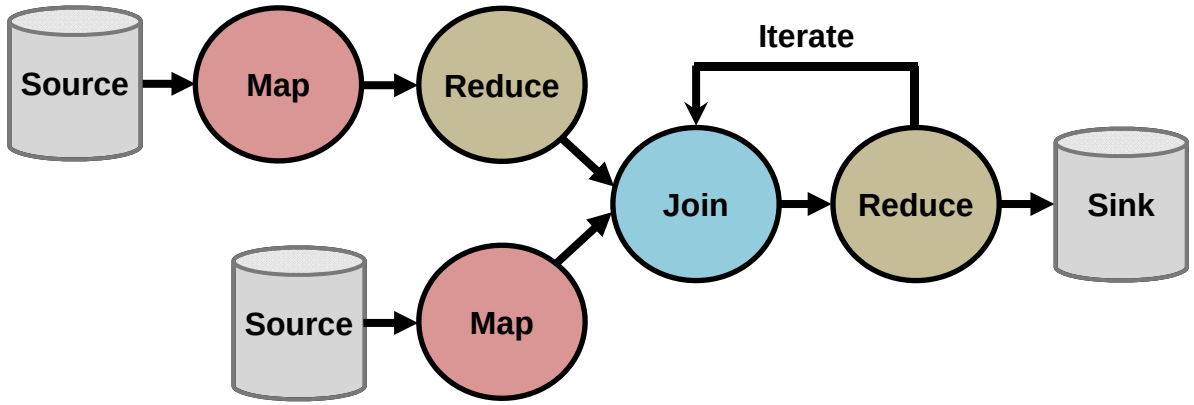


Parallel Execution

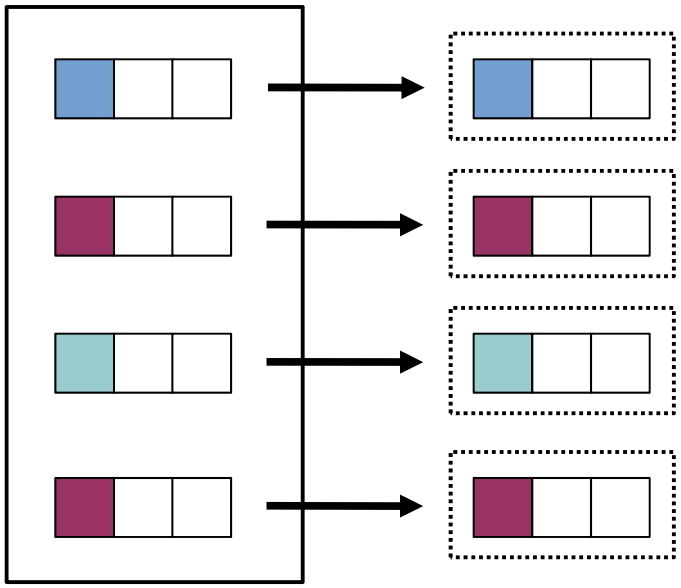


Rich operator and functionality set

Map, Reduce, Join, CoGroup, Union, Iterate, Delta Iterate, Filter, FlatMap, GroupReduce, Project, Aggregate, Distinct, Vertex-Update, Accumulators

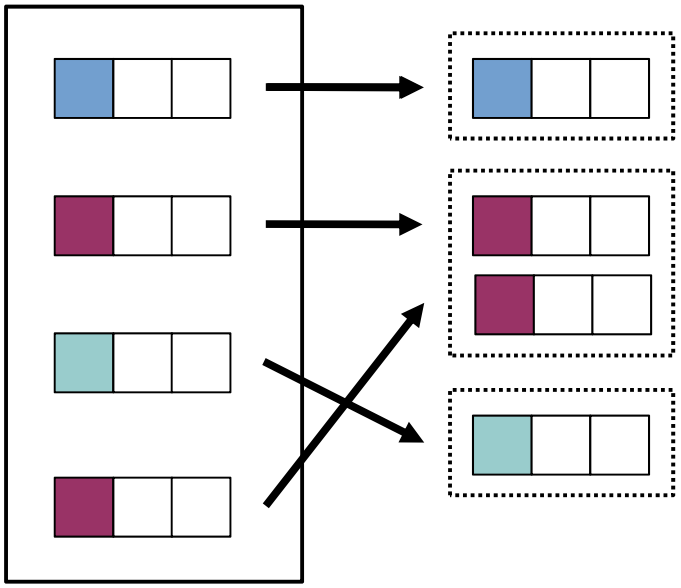


Base-Operator: **Map**



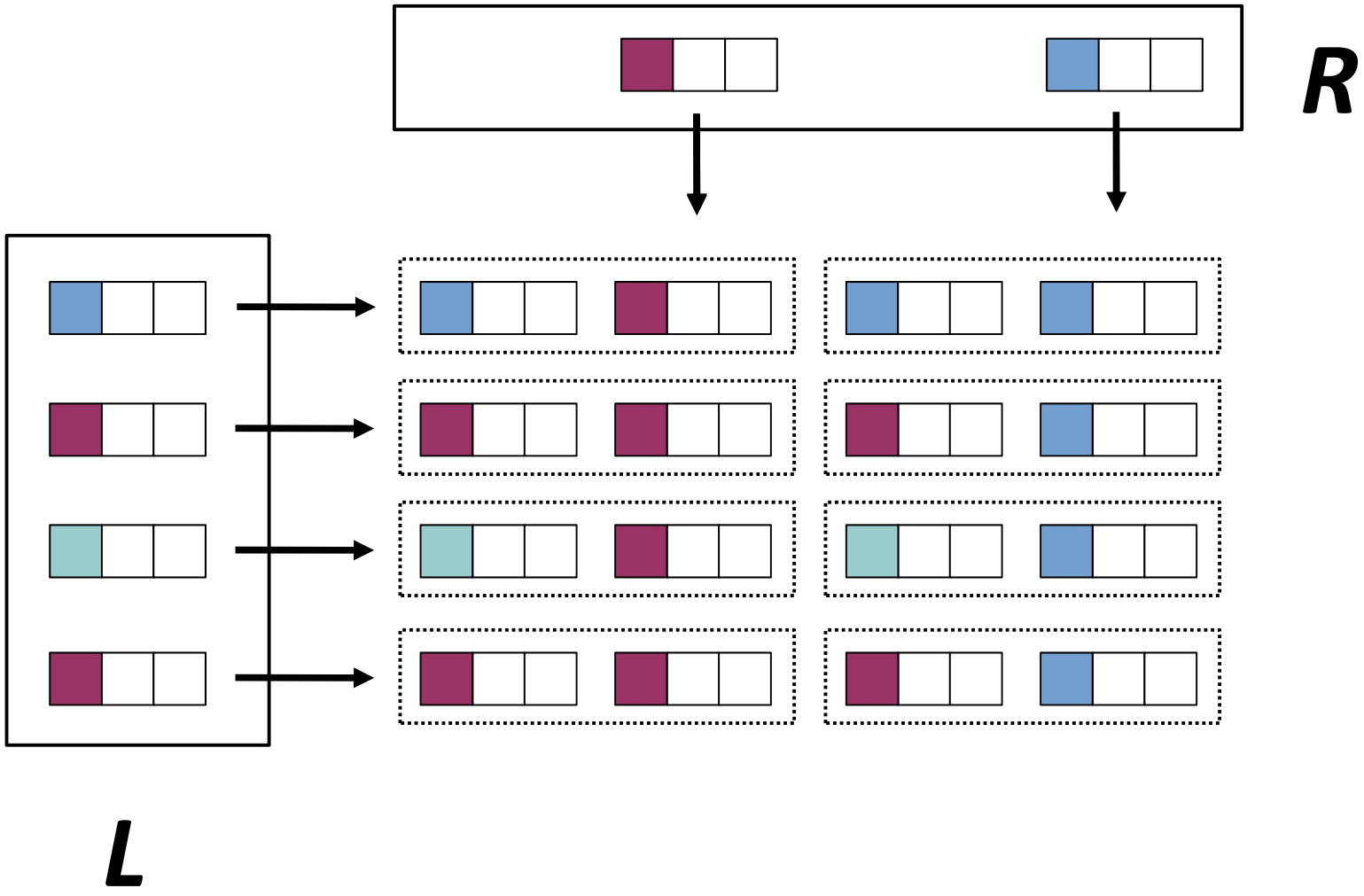
D

Base-Operator: **Reduce**

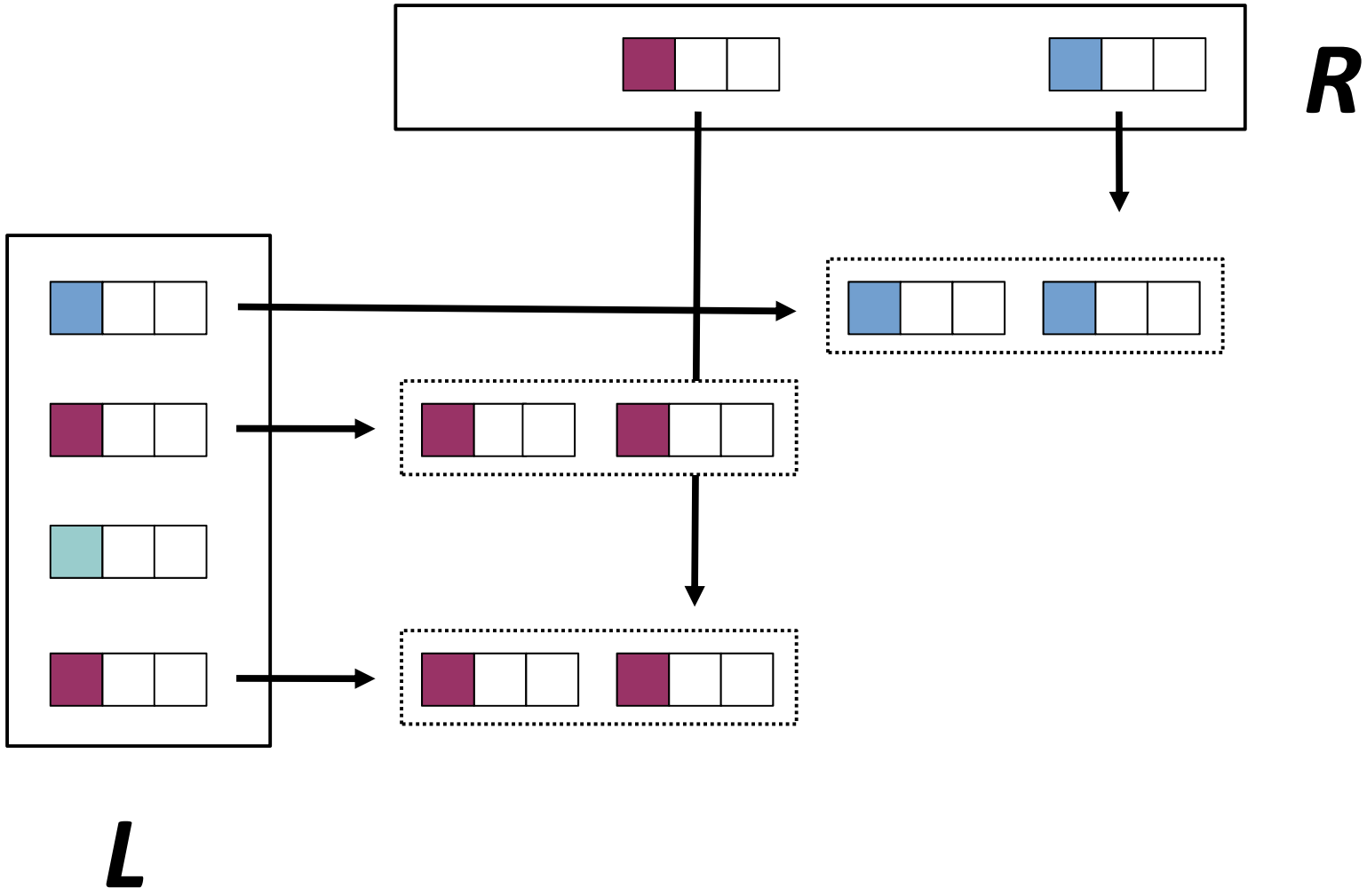


D

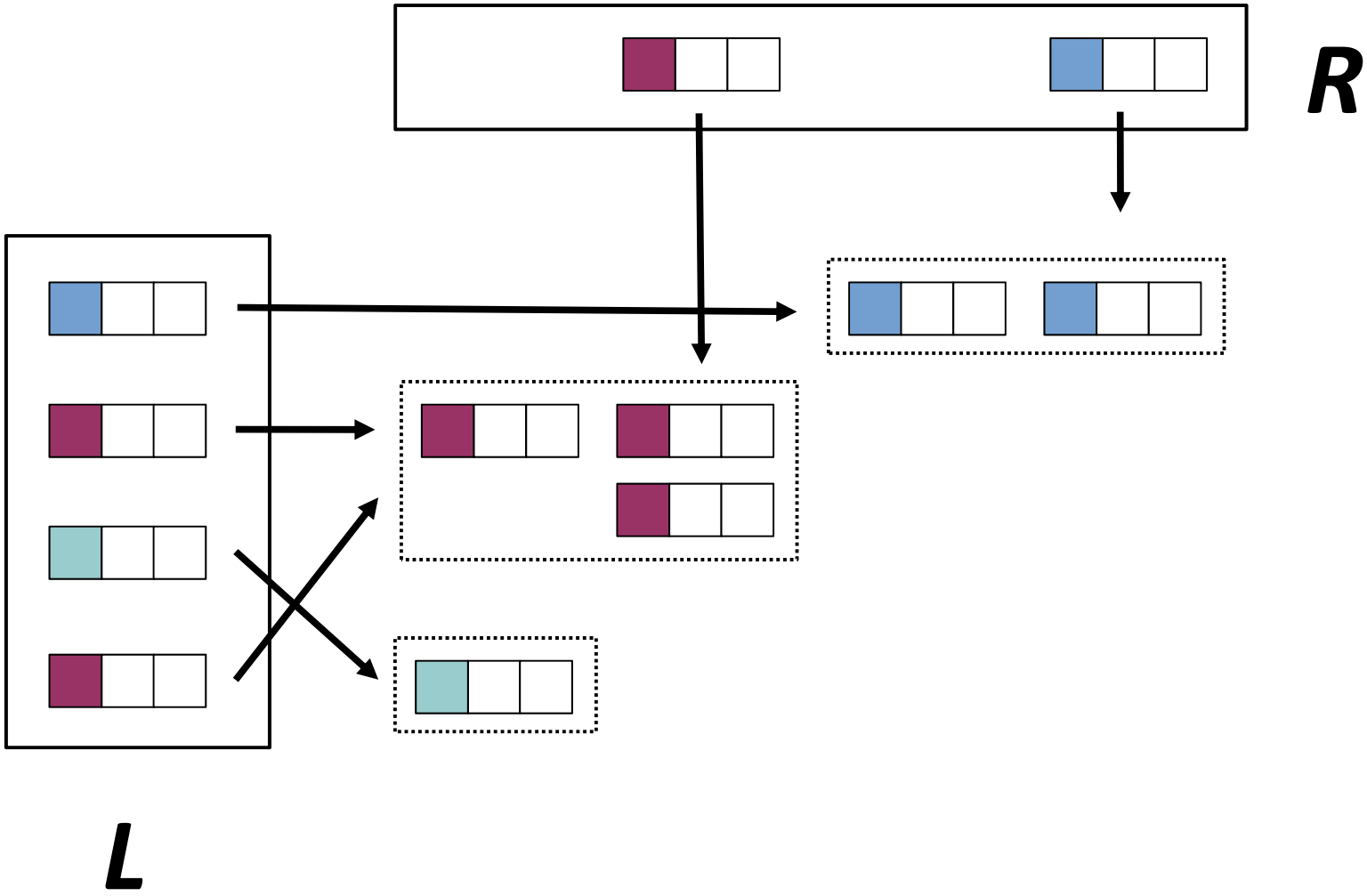
Base-Operator: **Cross**



Base-Operator: **Join**

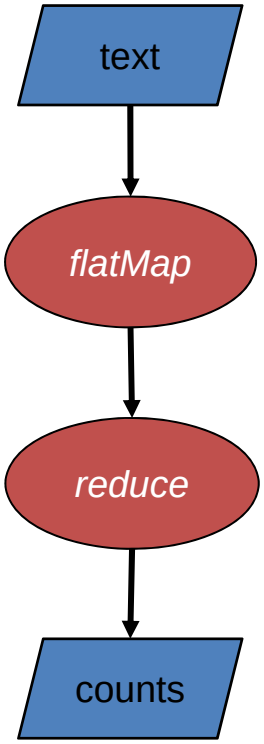


Base-Operator: **CoGroup**



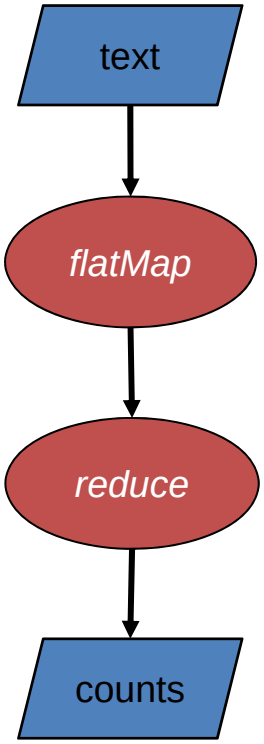
WordCount in Java

```
ExecutionEnvironment env =
  ExecutionEnvironment.getExecutionEnvironment();

DataSet<String> text = readTextFile (input);

DataSet<Tuple2<String, Integer>> counts= text
  .map (l -> l.split("\\W+"))
  .flatMap ((String[] tokens,
             Collector<Tuple2<String, Integer>> out) -> {
    Arrays.stream(tokens)
    .filter(t -> t.length() > 0)
    .forEach(t -> out.collect(new Tuple2<>(t, 1)));
   })
  .groupBy(0)
  .sum(1);

env.execute("Word Count Example");
```



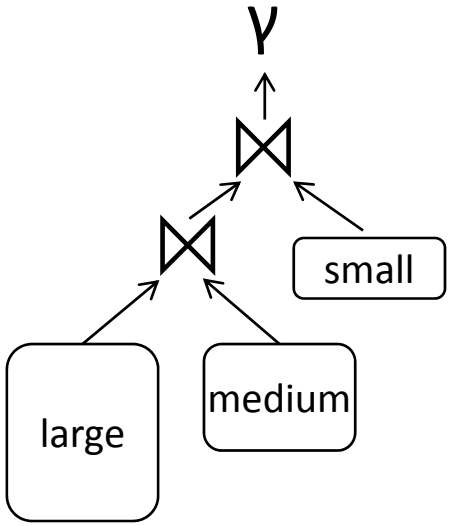
WordCount in Scala

```
val env = ExecutionEnvironment
  .getExecutionEnvironment

val input = env.readTextFile(textInput)

val counts = text
  .flatMap { line => line.split("\\W+") }
  .filter { term => term.nonEmpty }
  .map { term => (term, 1) }
  .groupBy(0)
  .sum(1)

env.execute()
```



Long operator pipelines

```
DataSet<Tuple...> large = env.readCsv(...);
DataSet<Tuple...> medium = env.readCsv(...);
DataSet<Tuple...> small = env.readCsv(...);

DataSet<Tuple...> joined1 =
        large.join(medium)
             .where(3).equals(1)
             .with(new JoinFunction() { ... });

DataSet<Tuple...> joined2 =
         small.join(joined1)
              .where(0).equals(2)
              .with(new JoinFunction() { ... });

DataSet<Tuple...> result = joined2.groupBy(3)
                                  .max(2);
```



Beyond Key/Value Pairs

```
DataSet<Page> pages = ...;
DataSet<Impression> impressions = ...;

DataSet<Impression> aggregated =
 impressions
  .groupBy("url")
  .sum("count");

pages.join(impressions).where("url").equalTo("url")


// custom data types

class Impression {
    public String url;
    public long count;
}
```

```
class Page {
    public String url;
    public String topic;
}
```

Flink's optimizer

- inspired by optimizers of parallel database systems
  - cost models and reasoning about interesting properties

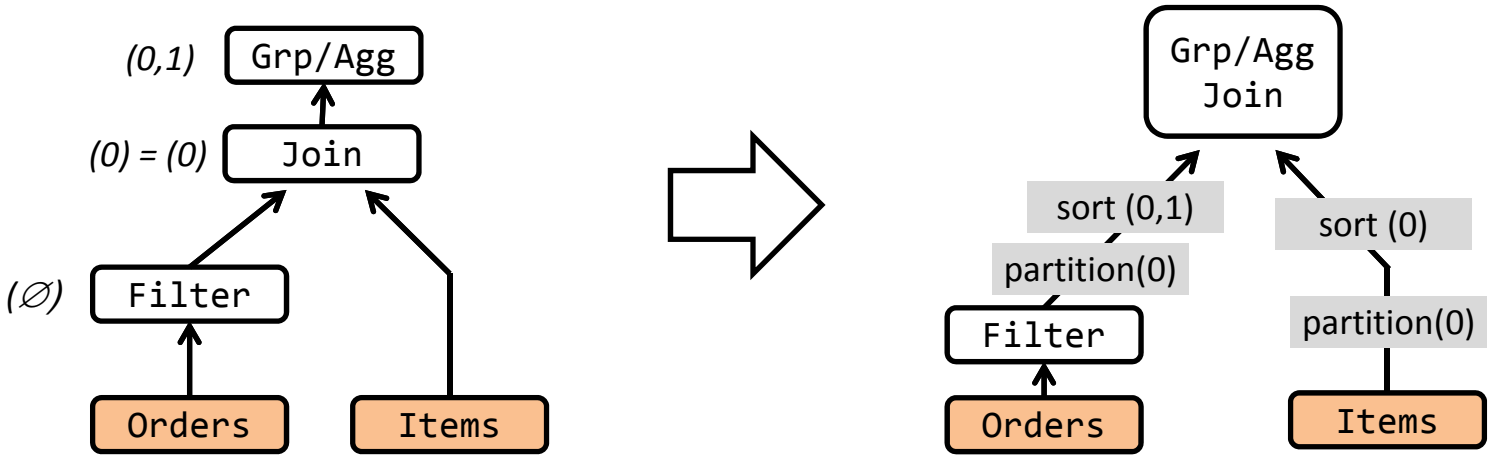
- physical optimization follows cost-based approach
  - Select data shipping strategy (forward, partition, broadcast)
  - Local execution (sort merge join/hash join)
  - keeps track of interesting properties such as sorting, grouping and partitioning

- optimization of Flink programs more difficult than in the relational case:
  - no fully specified operator semantics due to UDFs
  - unknown UDFs complicate estimating intermediate result sizes
  - no pre-defined schema present

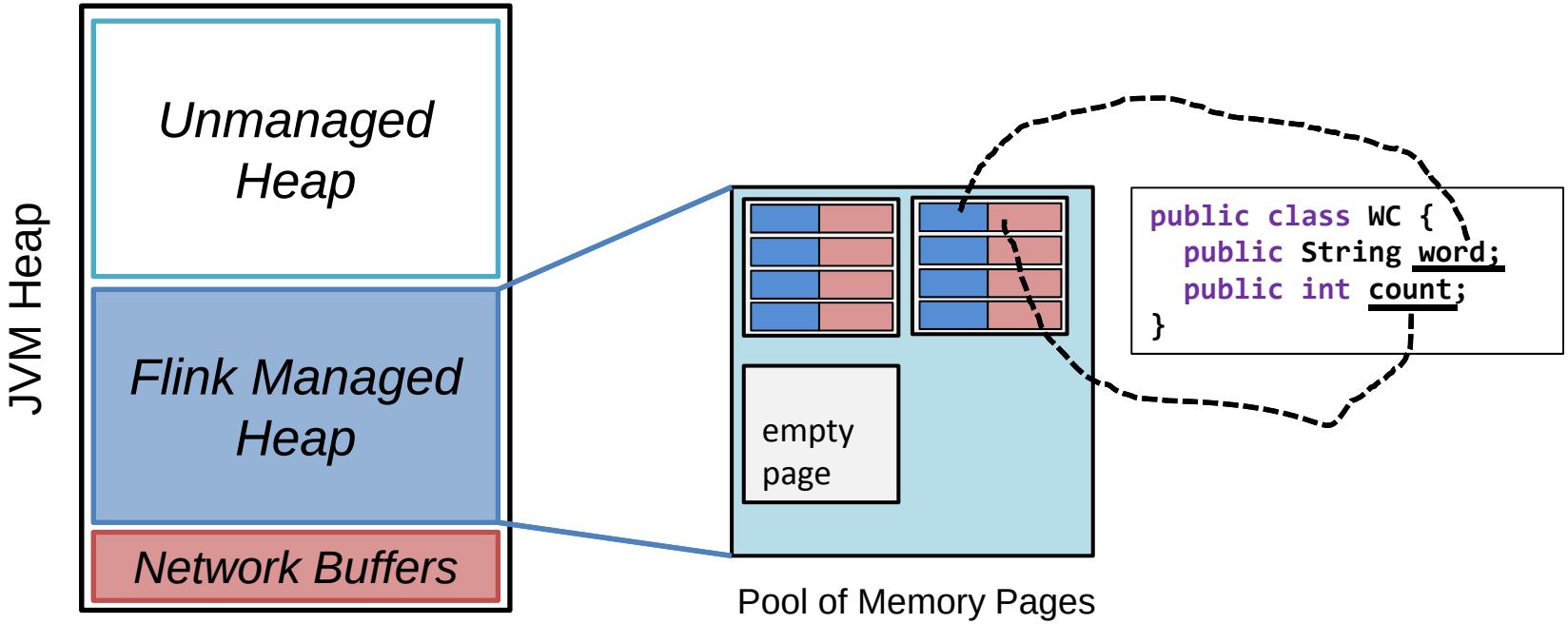
Optimization example

```
case class Order(id: Int, priority: Int, ...)
case class Item(id: Int, price: double, )
case class PricedOrder(id, priority, price)
```

```
val orders = DataSource(...)
val items = DataSource(...)

val filtered = orders filter { ... }

val prio = filtered join items where { _.id } isEqualTo { _.id }
                    map {(o,li) => PricedOrder(o.id, o.priority, li.price)}

val sales = prio groupBy {p => (p.id, p.priority)} aggregate ({_.price},SUM)
```

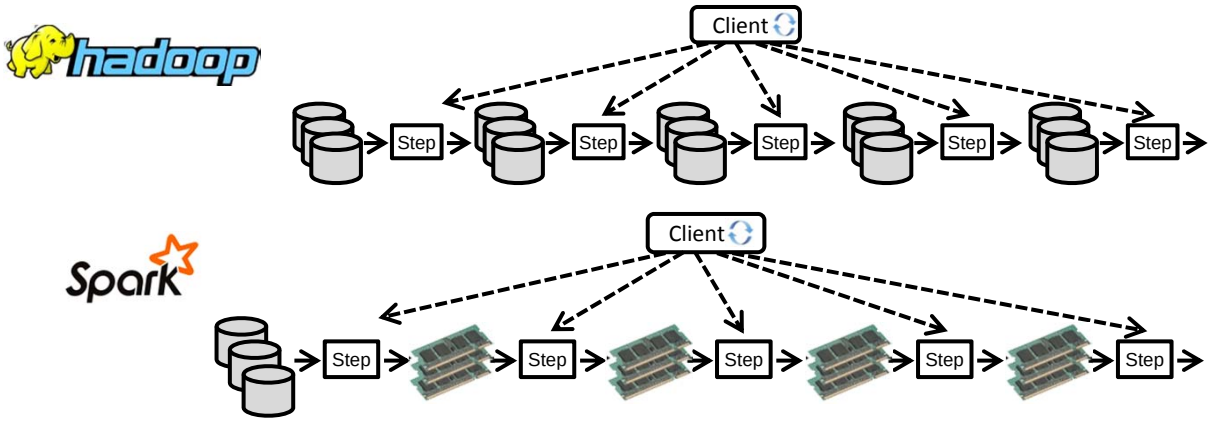


Memory management



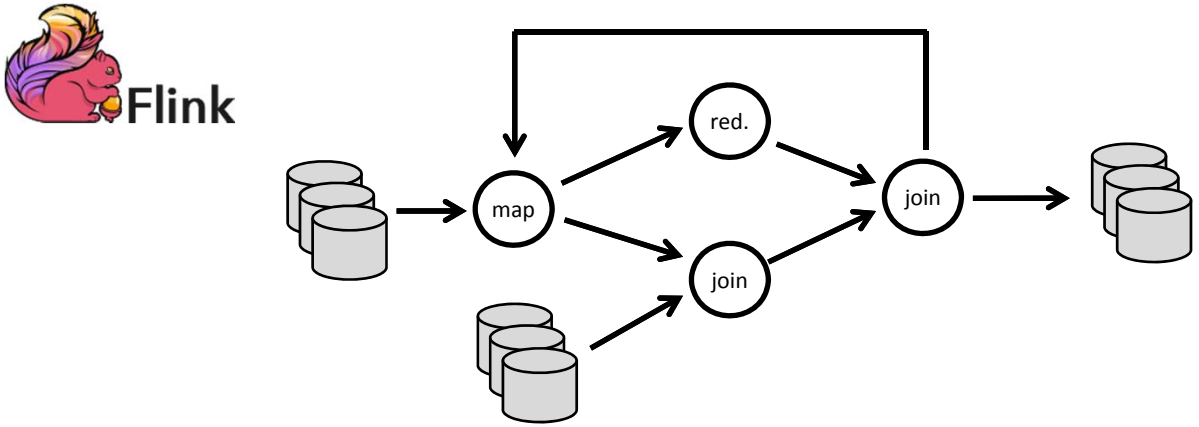
- Flink manages its own memory
- User data stored in serialize byte arrays
- In-memory caching and data processing happens in a dedicated memory fraction
- Never breaks the JVM heap
- Very efficient disk spilling and network transfers

Built-in vs. driver-based iterations



Loop outside the system, in driver program

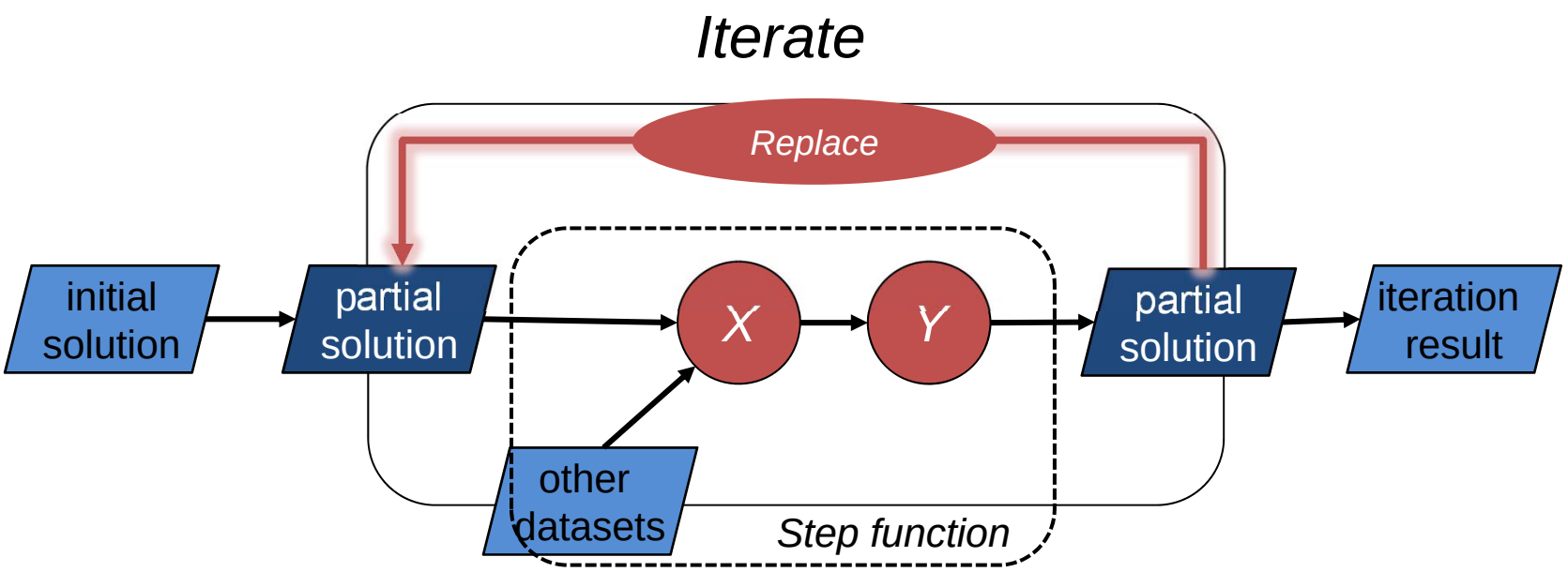
Iterative program looks like many independent jobs



Dataflows with feedback edges

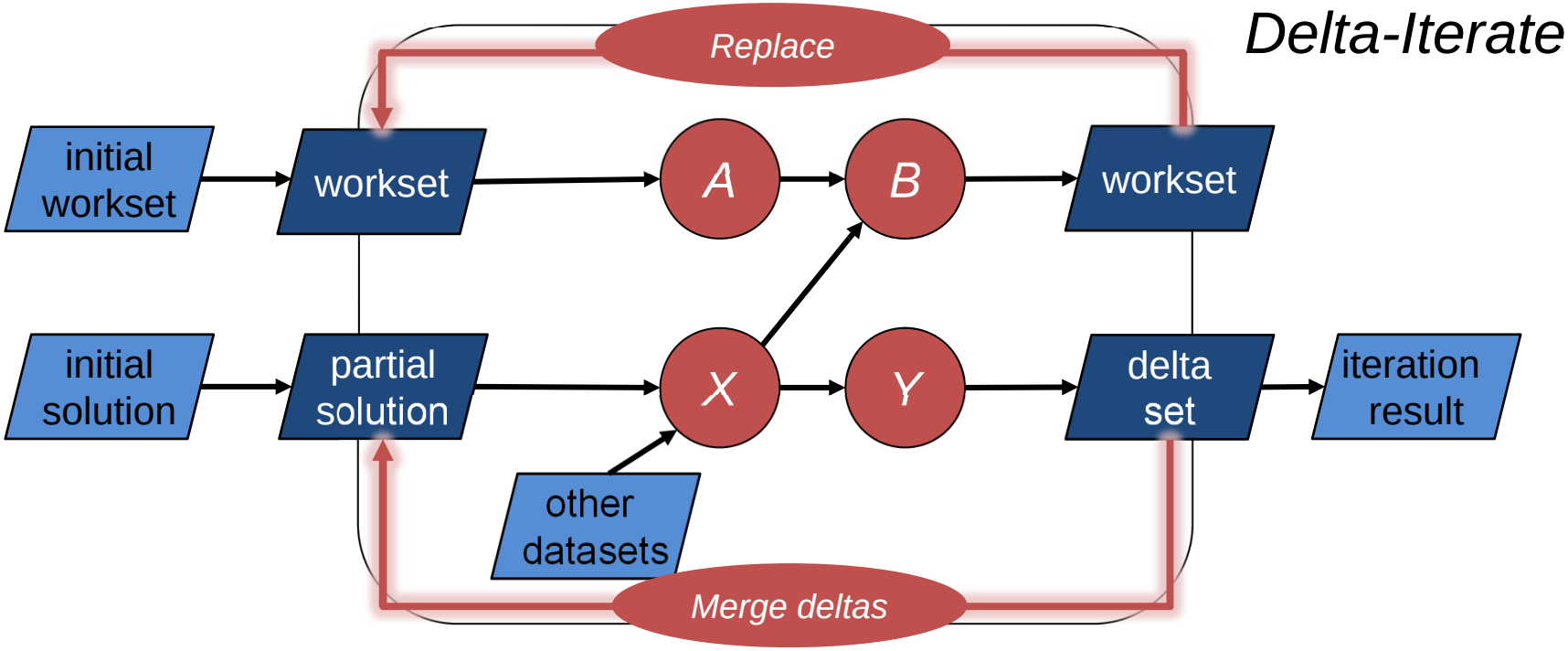
System is iteration-aware, can optimize the job

“Iterate” operator



- Built-in operator to support looping over data
- Applies step function to partial solution until convergence
- Step function can be arbitrary Flink program
- Convergence via fixed number of iterations or custom convergence criterion

“Delta Iterate” operator



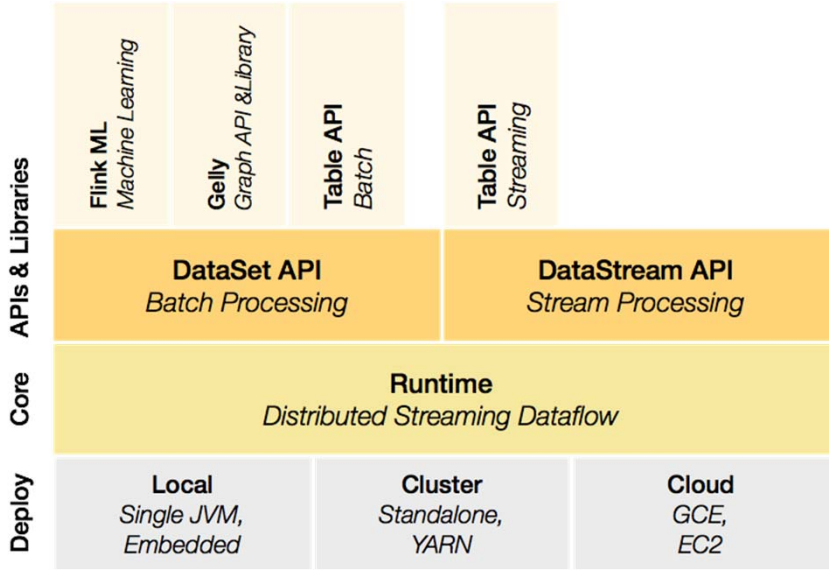
- compute next workset and changes to the partial solution until workset is empty
- generalizes vertex-centric computing of Pregel and GraphLab

ReCap: What is Apache Flink?

Apache Flink is an open source platform for scalable batch and stream data processing.

- The core of Flink is a distributed streaming dataflow engine.
  - Executing dataflows in parallel on clusters
  - Providing a reliable foundation for various workloads

- **DataSet** and **DataStream** programming abstractions are the foundation for user programs and higher layers

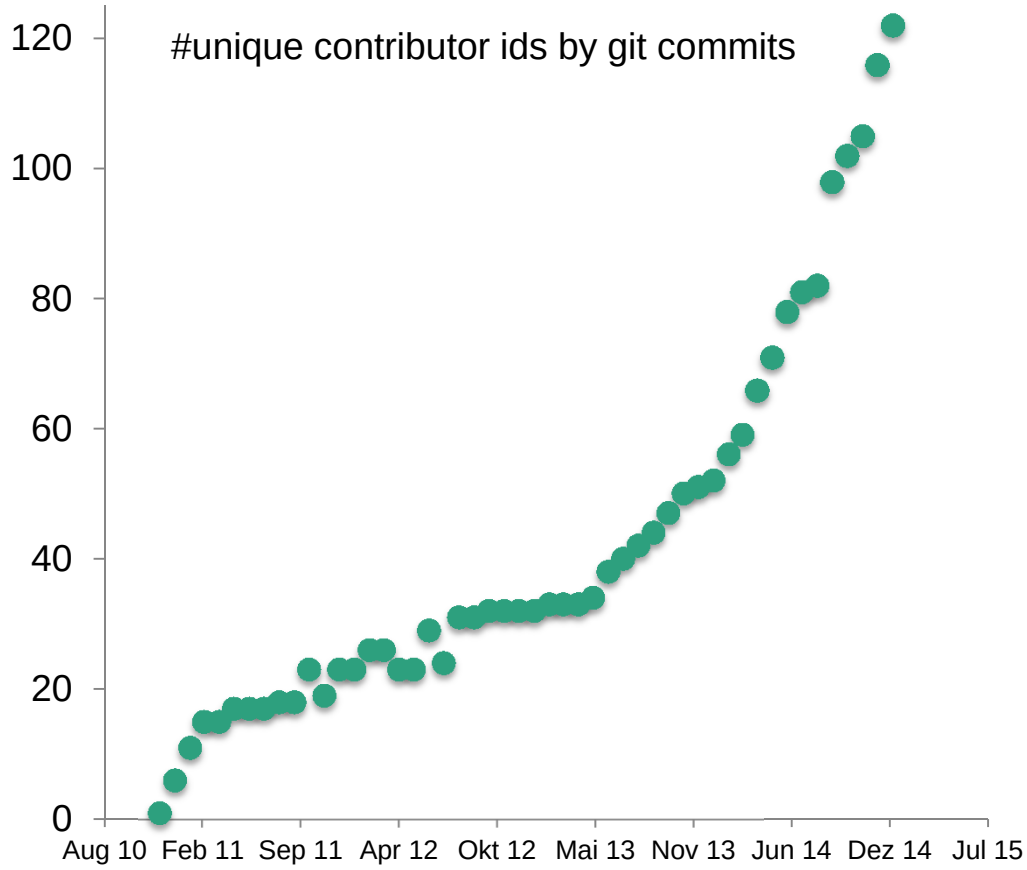


http://flink.apache.org

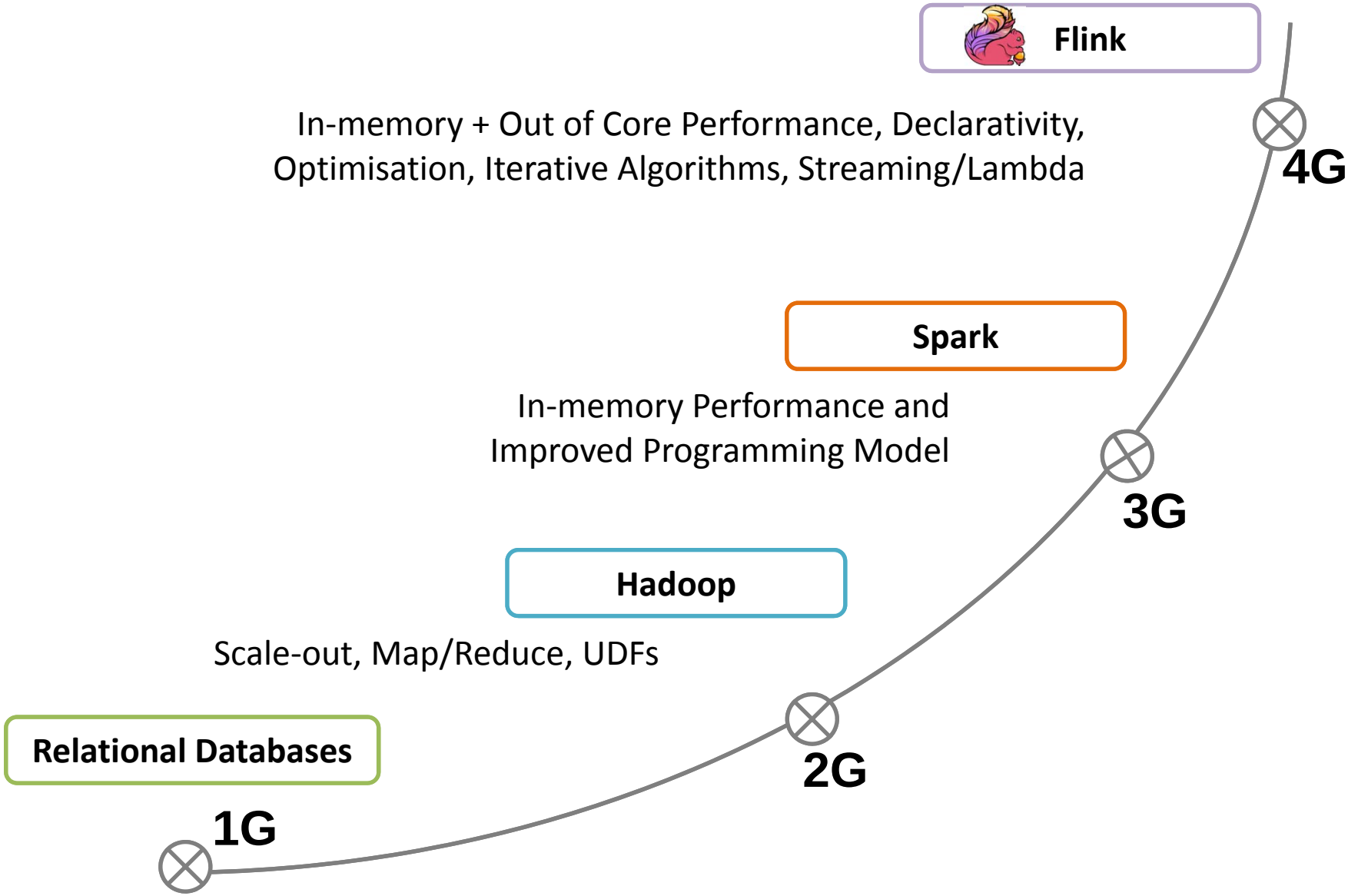
Working on and with Apache Flink

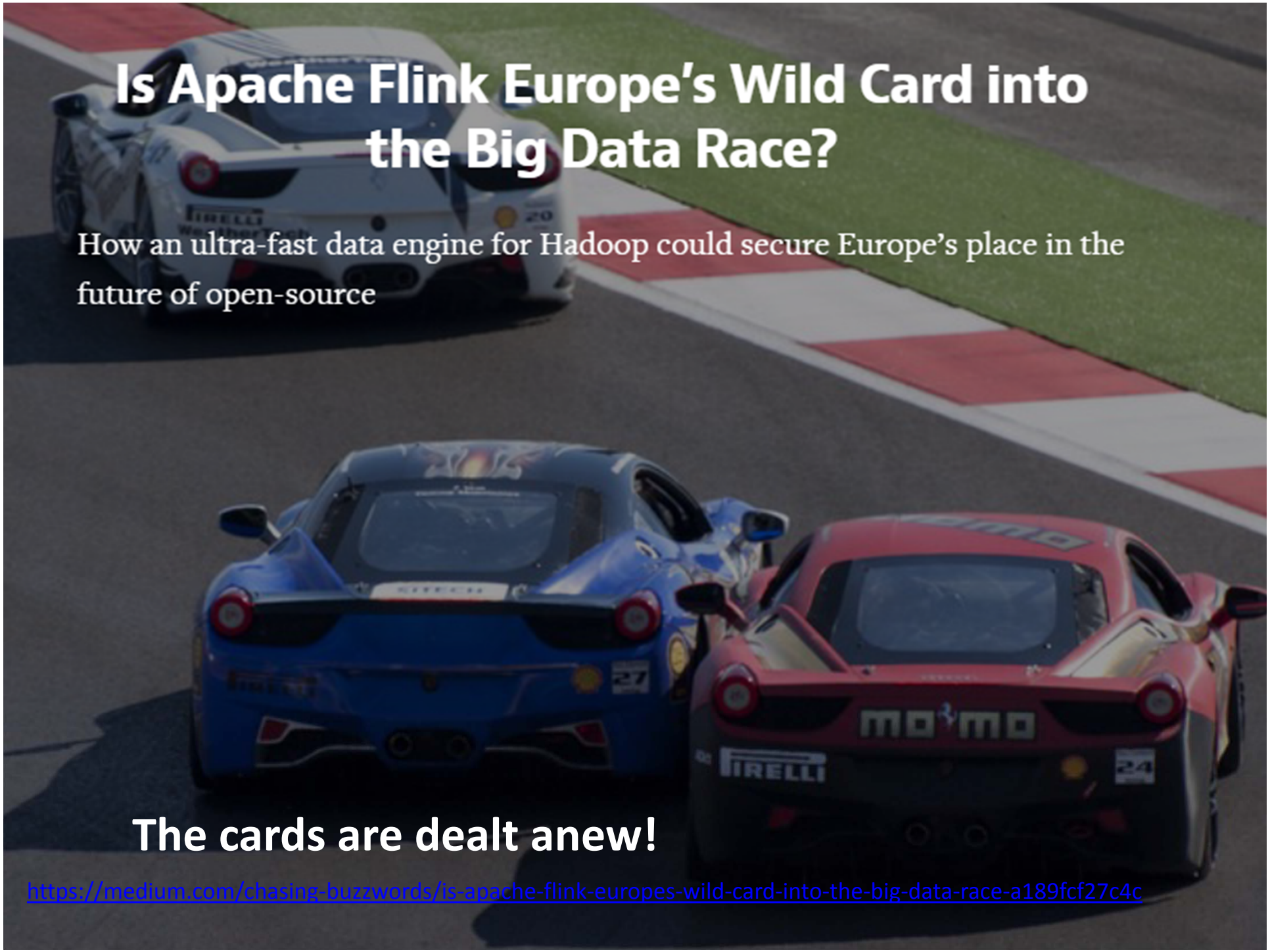
- Flink homepage
  https://flink.apache.org
- Flink Mailing Lists
  https://flink.apache.org/community.html#mailing-lists
- Flink Meetup in Berlin
  http://www.meetup.com/de/Apache-Flink-Meetup/

Flink community



Evolution of Big Data Platforms




Is Apache Flink Europe's Wild Card into the Big Data Race?
How an ultra-fast data engine for Hadoop could secure Europe's place in the future of open-source
The cards are dealt anew!
https://medium.com/chasing-buzzwords/is-apache-flink-europes-wild-card-into-the-big-data-race-a189fcf27c4c

Forbes on Apache Flink:

- *„[…] Flink, which is also a top-level project of the Apache Software Foundation, has just recently begun to attract many of the same admiring comments directed Spark's way 12-18 months ago. Despite sound technical credentials, ongoing development, big investments, and today's high-profile endorsement from IBM, it would be unwise (and implausible) to crown Spark as the winner just yet. […]"*

  http://www.forbes.com/sites/paulmiller/2015/06/15/ibm-backs-apache-spark-for-big-data-analytics/

- http://www.infoworld.com/article/2919602/hadoop/flink-hadoops-new-contender-for-mapreduce-spark.html

- http://www.datanami.com/2015/06/12/8-new-big-data-projects-to-watch/

- **Two day developer conference** with in-depth talks from
  - developers and contributors
  - industry and research users
  - related projects

- **Flink training** sessions (in parallel)
  - System introduction, real-time stream processing, APIs on top

- **Flink Forward registration & call for abstracts is open now at** http://flink-forward.org/