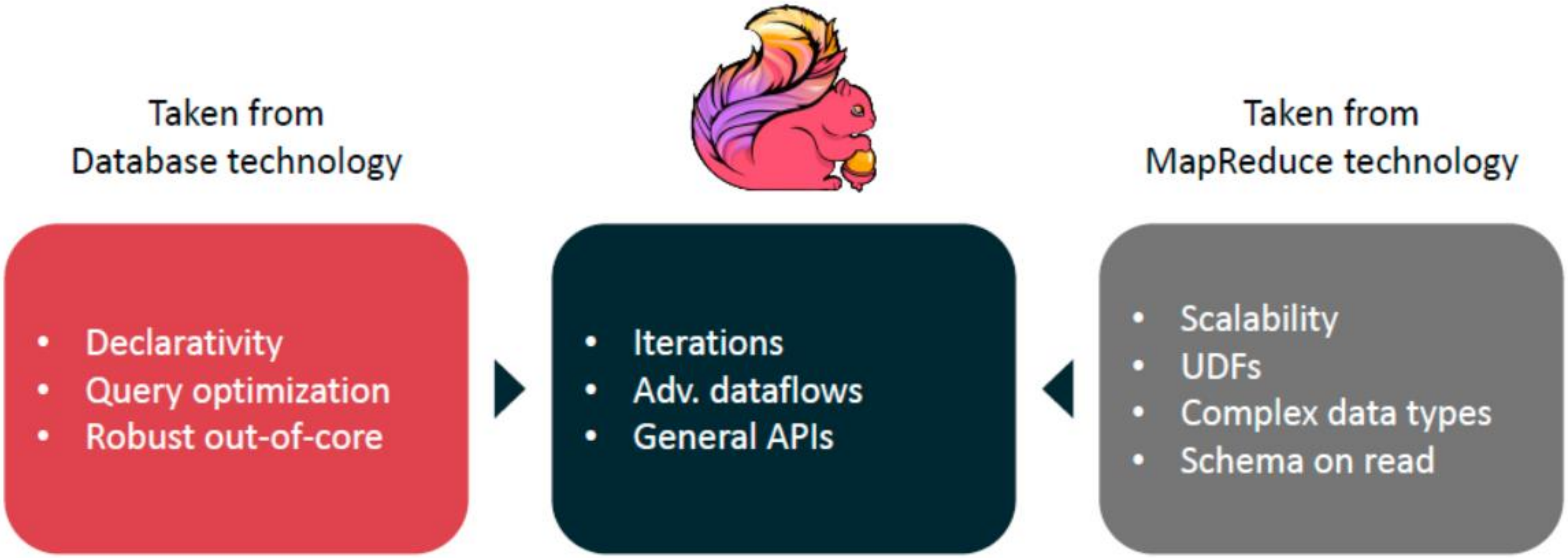
# Apache Flink

Fuchkina Ekaterina

with Material from Andreas Kunft -TU Berlin / DIMA; dataArtisans slides

## *What is Apache Flink*

Massive parallel data flow engine with unified batch-and stream-processing



Taken from Database technology

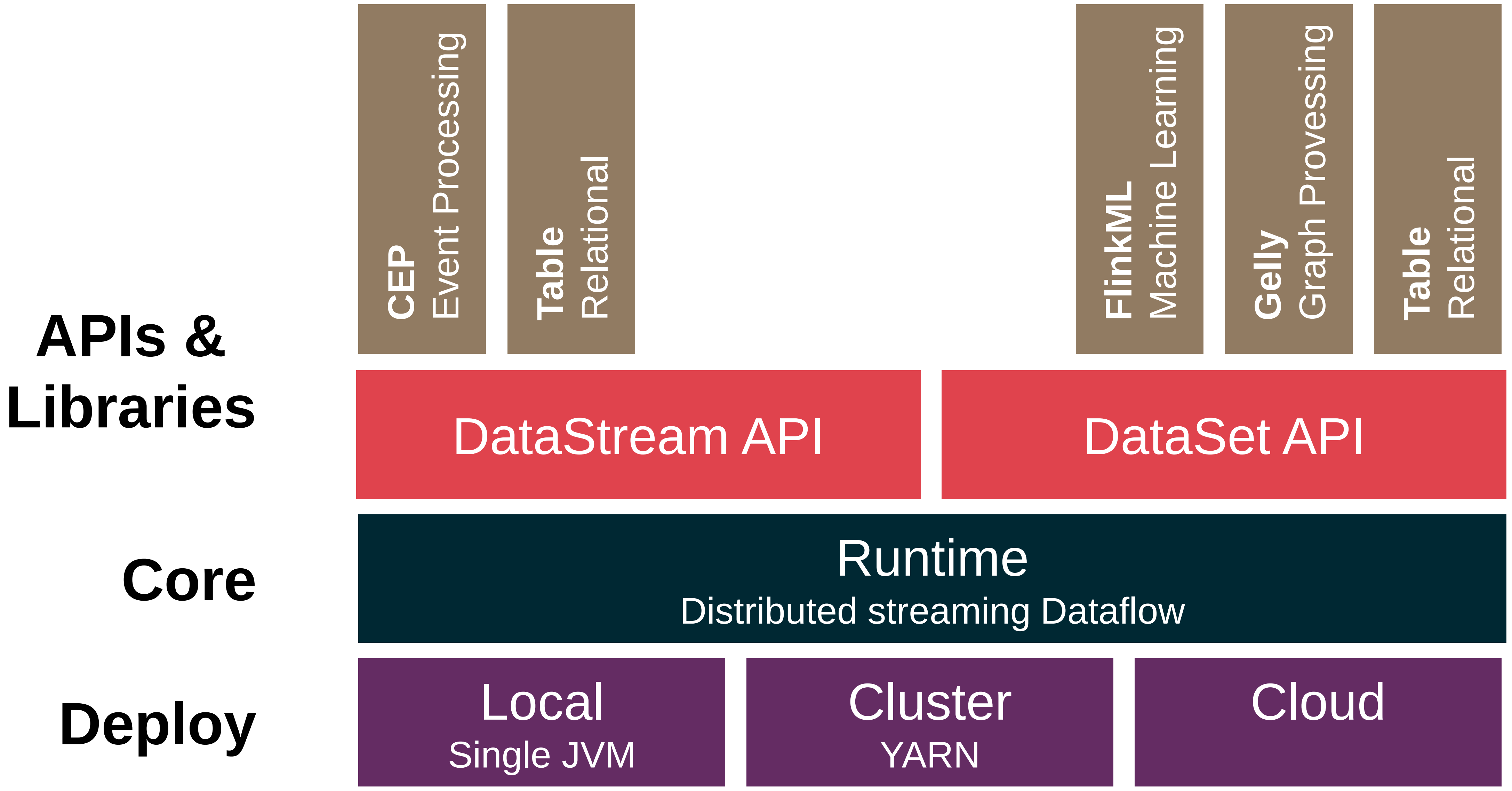
- Declarativity
- Query optimization
- Robust out-of-core

- Iterations
- Adv. dataflows
- General APIs

Taken from MapReduce technology

- Scalability
- UDFs
- Complex data types
- Schema on read

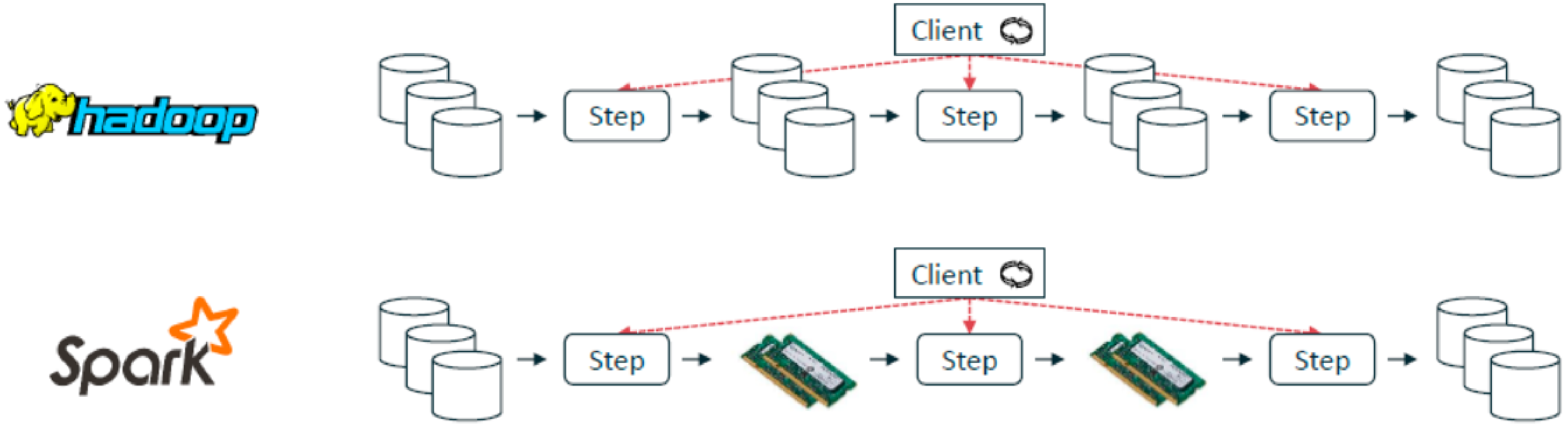
## System Stack



## *The case for Flink*

- Performance and ease of use
  - Exploits in-memory processing and pipelining, language-embedded logical APIs
- Unified batch and real streaming
  - Batch and Stream APIs on top of a streaming engine
- A runtime that "just works" without tuning
  - Custom memory management inside the JVM
- Predictable and dependable execution
  - Bird’s-eye view of what runs and how, and what failed and why

# *Built-in(native) vs. driver-based looping*



Loop outside the system, in driver program

Iterative program looks like many independent jobs

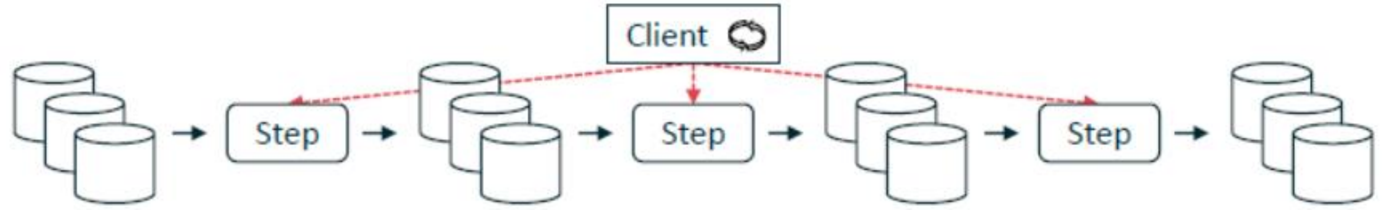
**Non-native iterations**

```
for (int i = 0; i < maxIterations; i++) {
    // Execute MapReduce job
}
```

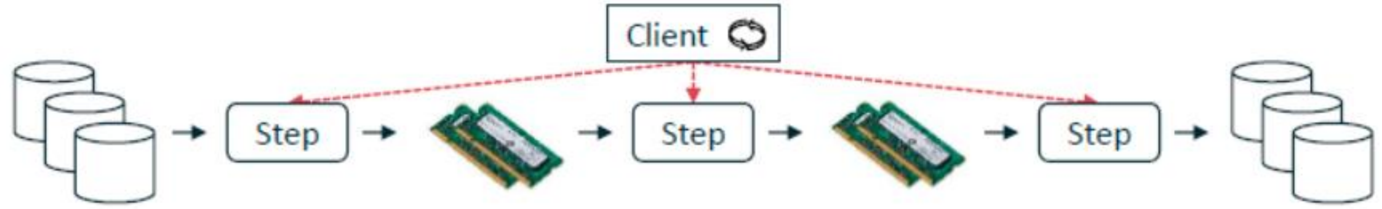
**Non-native streaming**

```
while (true) {
  // get next few records
  // issue batch job
}
```

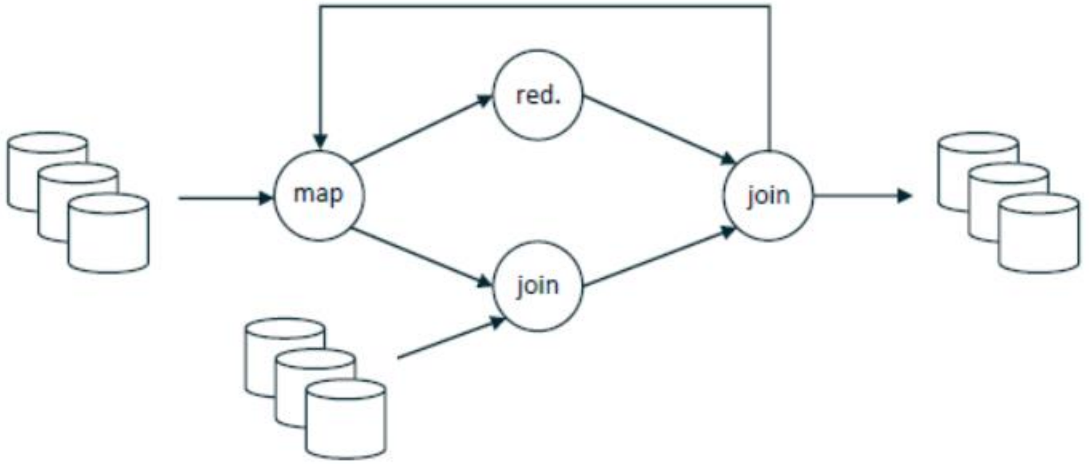
# *Built-in(native) vs. driver-based looping*



Loop outside the system, in driver program



Iterative program looks like many independent jobs



```
DataSet.flatMap(...)
       .groupBy(...)
       .reduce(...)
```

Dataflow with Feedback edges

System is iteration-aware, can optimize the job

## *Applications*

DataStream

Stream processing

Batch processing

DataSet

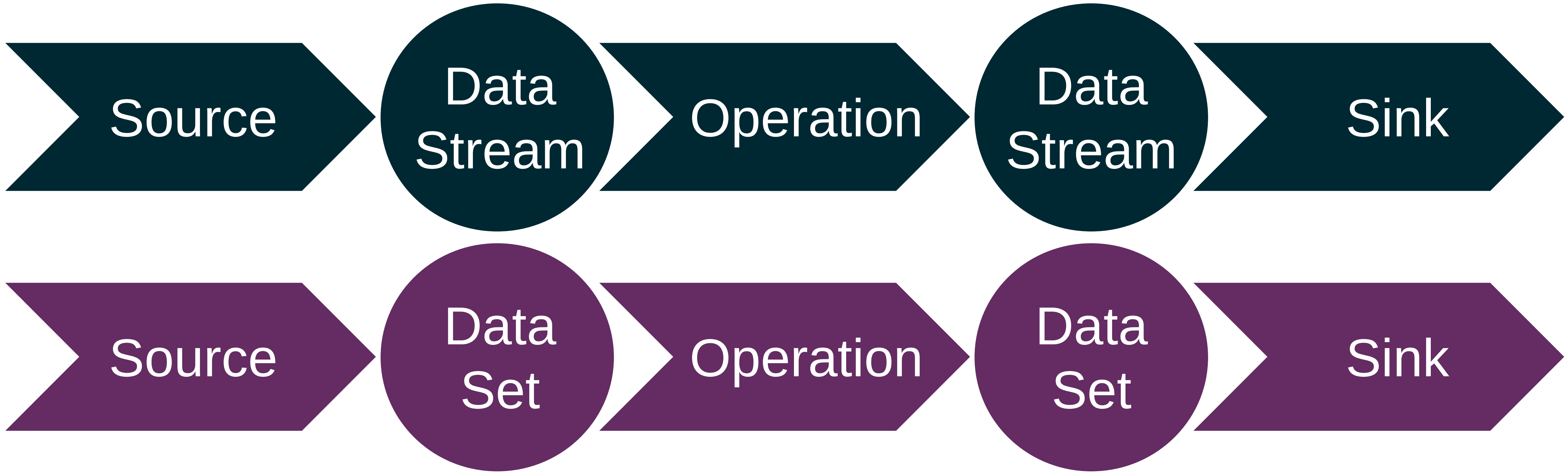
FlinkML

Machine Learning

Graph Analysis

Gelly

## *Basic API Concept*

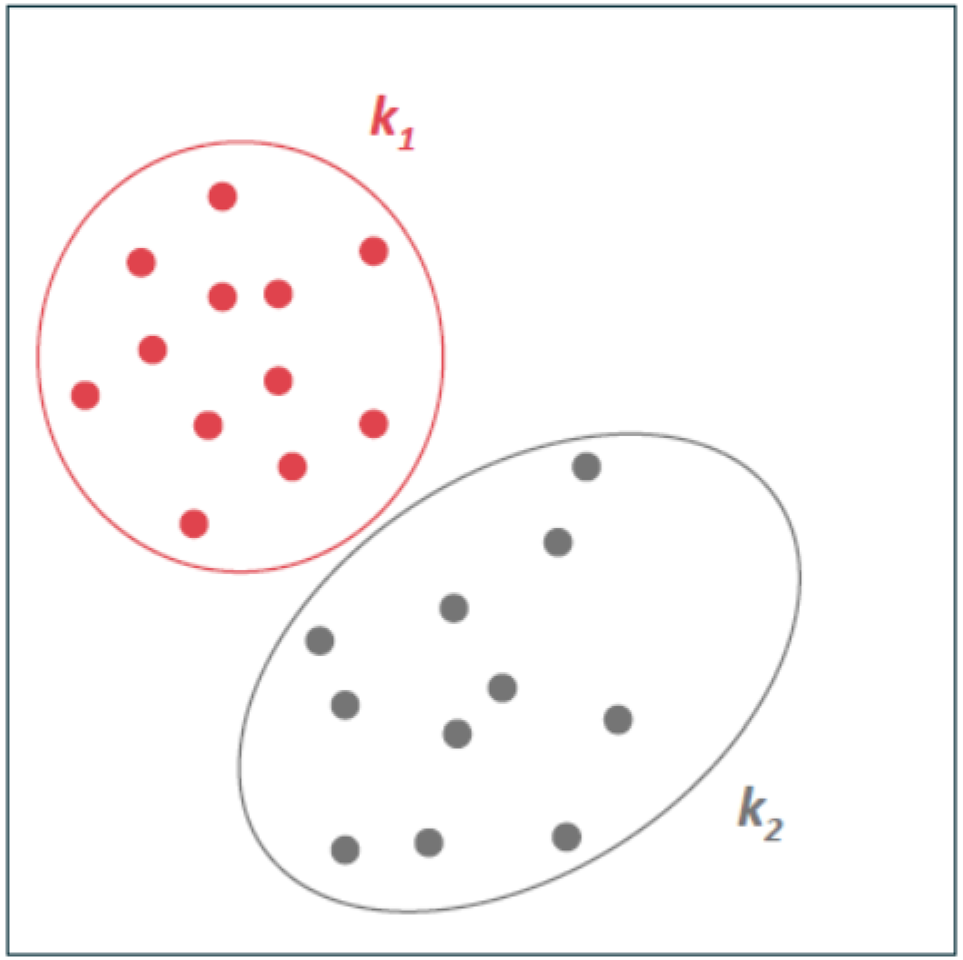


**Flink program writing:**

**1)** Bootstrap sources **2)** Apply operations **3)** Output to sink

# *Machine learning library: FlinkML*

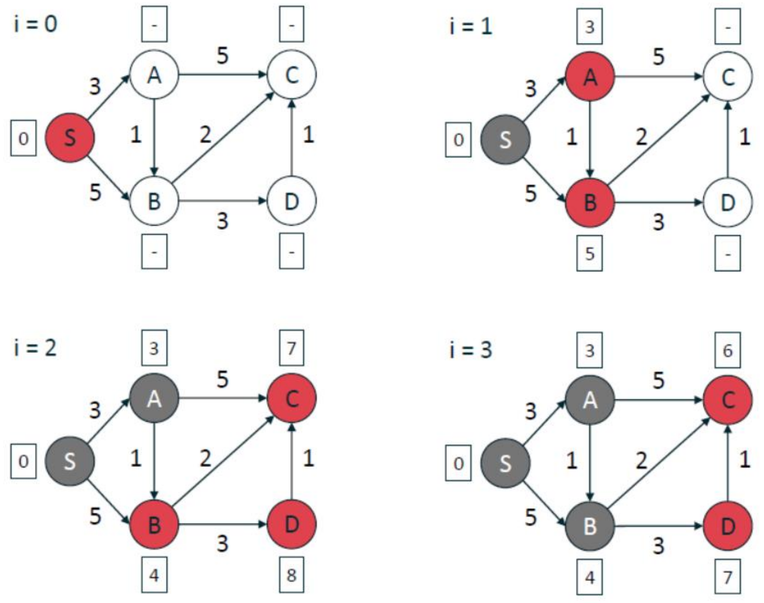
- Recently started effort
- Currently available algorithms
  - Classification
  - Logistic Regression
  - Clustering
  - Recommendation (ALS)



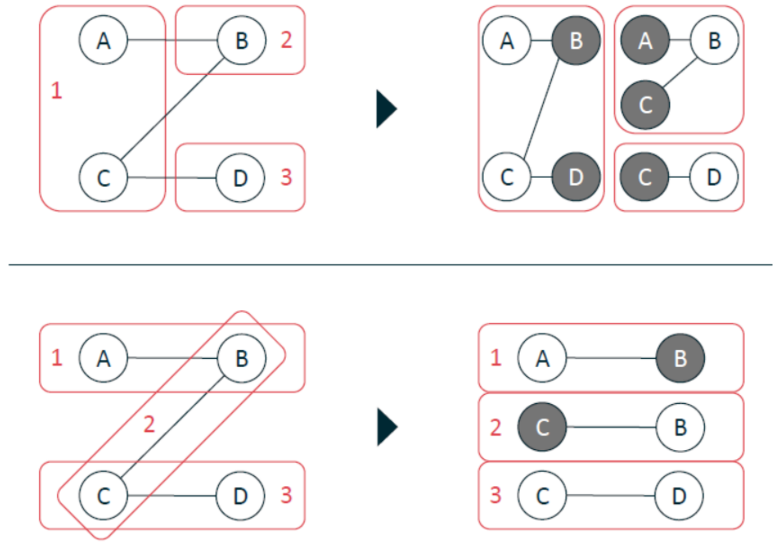
*K-Means*

# *Graph Analysis library: Gelly*

- Large-scale graph processing API
- Iterative Graph Processing
- Currently available algorithms
  - Single Source Shortest Paths
  - Weakly Connected Components
  - Community Detection
  - Page Rank
  - Label Propagation



*SSSP*



*Graph Partitioning*

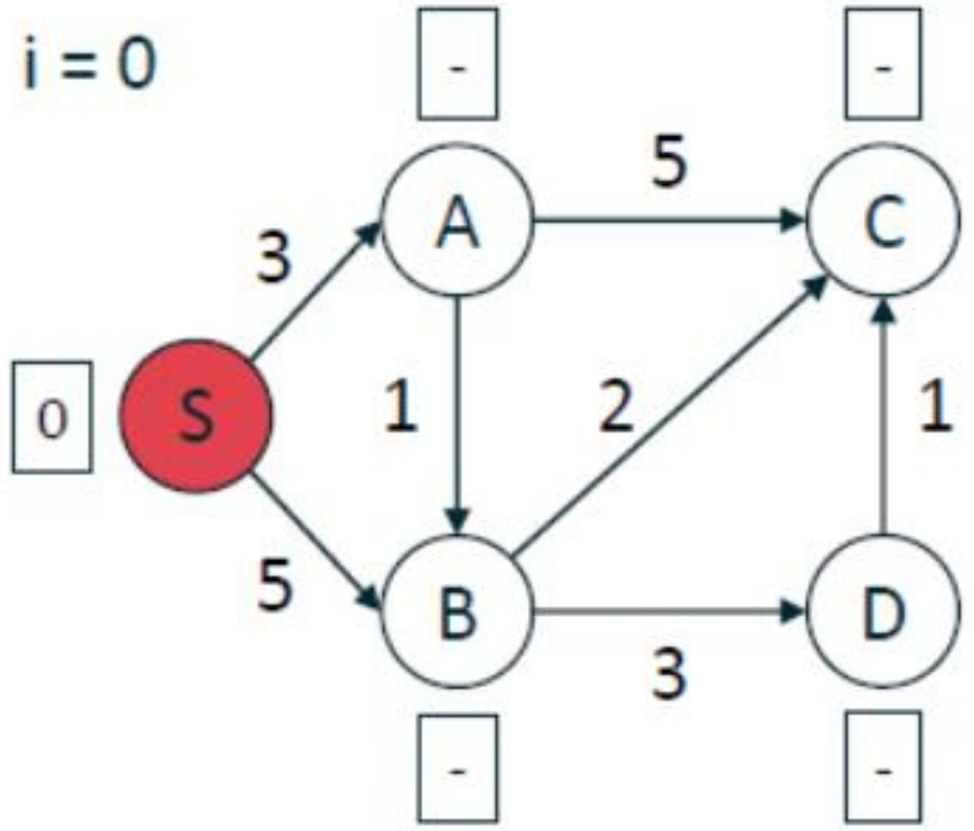
# Single Source Shortest Paths

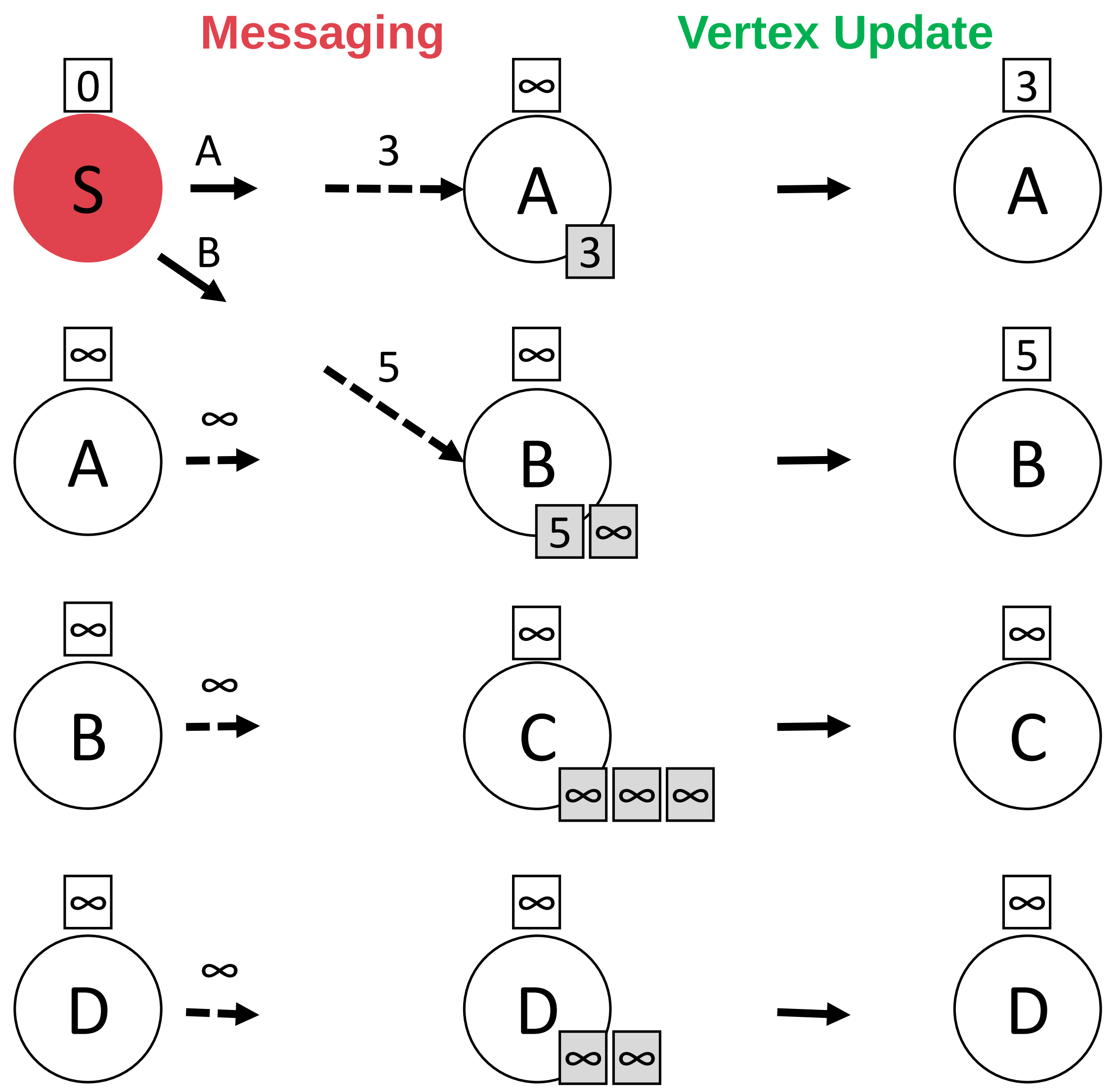
```java
ExecutionEnvironment env = ExecutionEnvironment.getExecutionEnvironment();

DataSet<Edge<Long, Double>> edges = getEdgesDataSet(env);

Graph<Long, Double, Double> graph = Graph.fromDataSet(edges, new InitVertices(), env);

// Execute the vertex-centric iteration
Graph<Long, Double, Double> result = graph.runVertexCentricIteration(
        new SSSPComputeFunction(srcVertexId), new SSSPCombiner(),
        maxIterations);

// Extract the vertices as the result
DataSet<Vertex<Long, Double>> singleSourceShortestPaths = result.getVertices();
```

**VertexUpdateFunction** **Messaging Function**

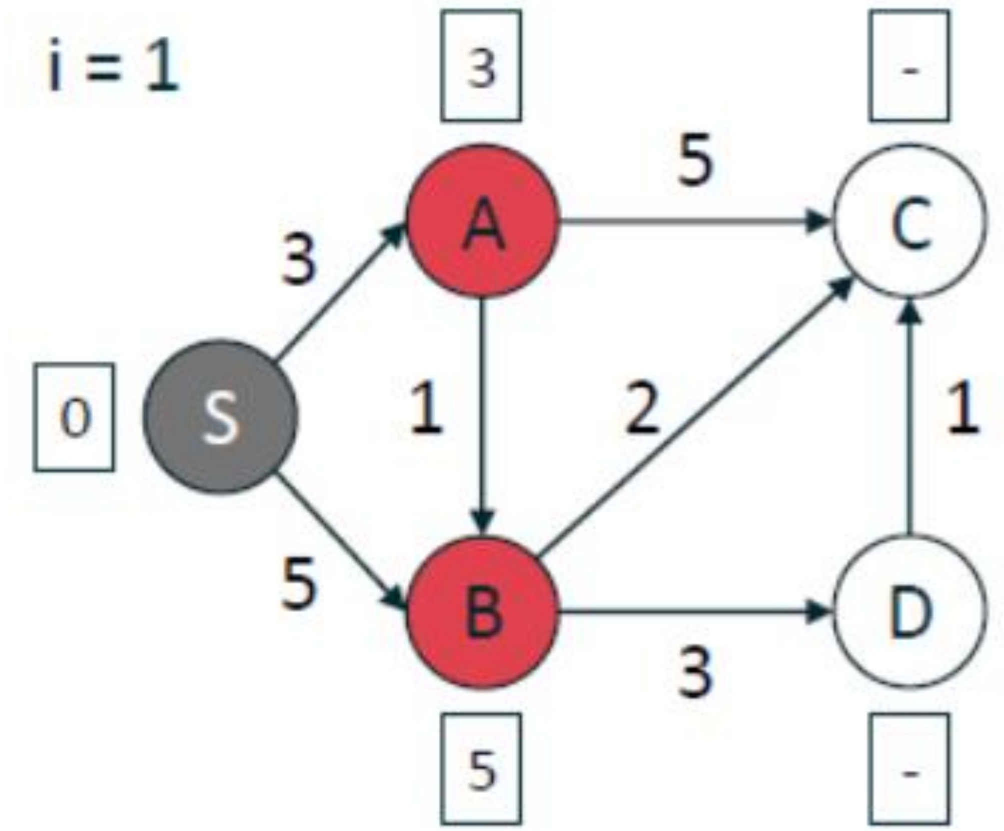
Vertex-centric

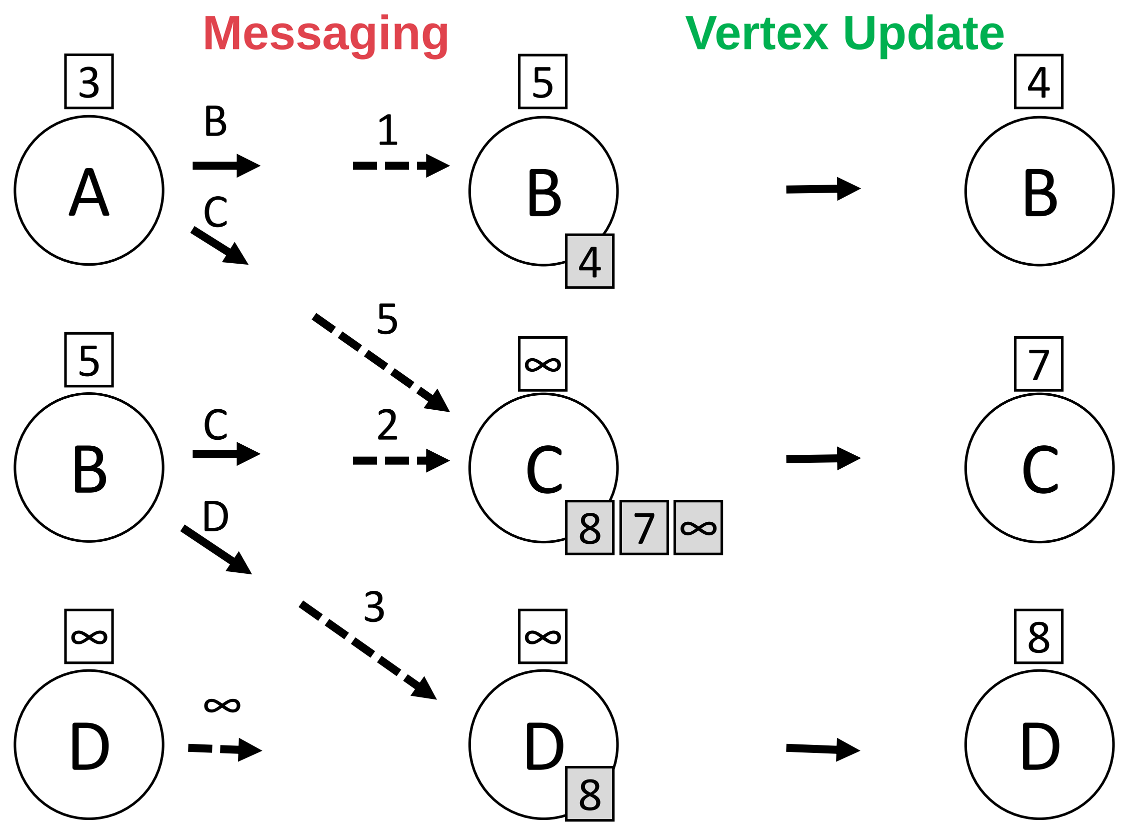
# *Single Source Shortest Paths*



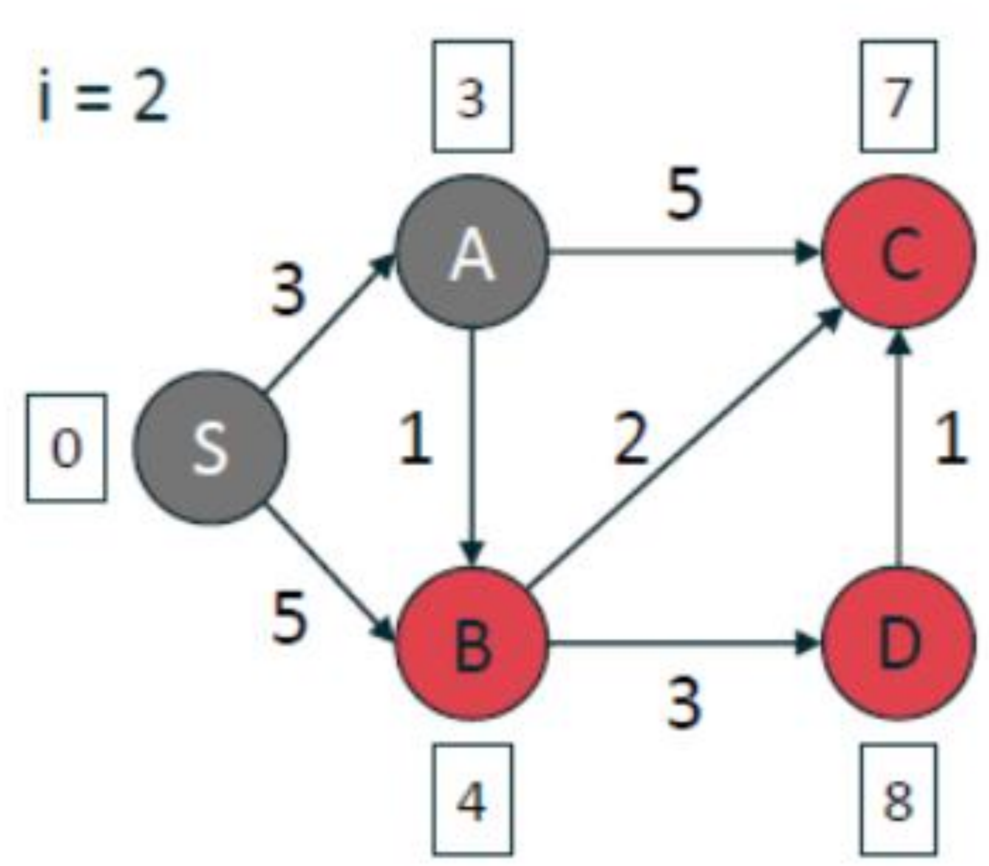


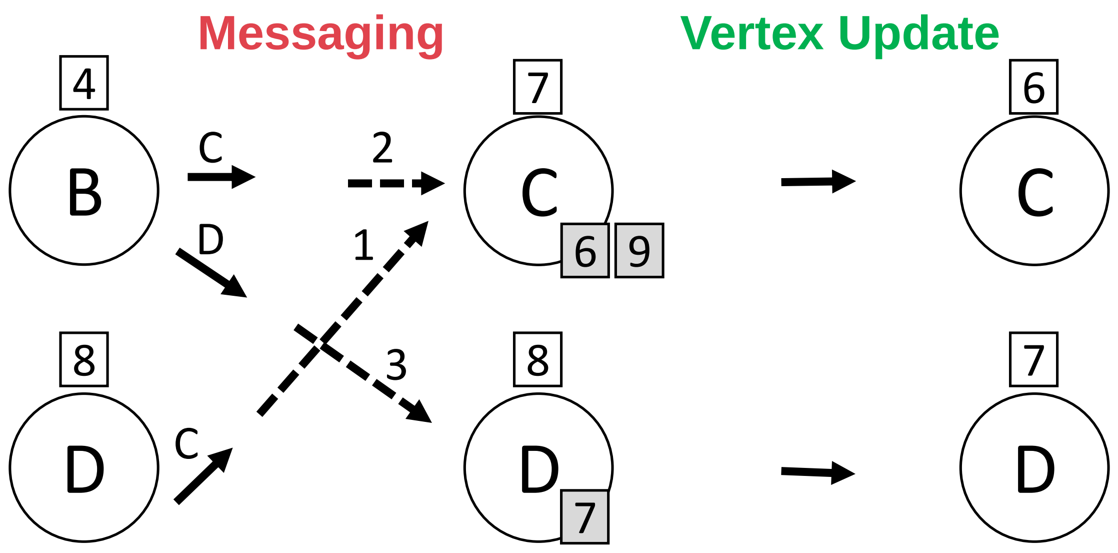
# *Single Source Shortest Paths*



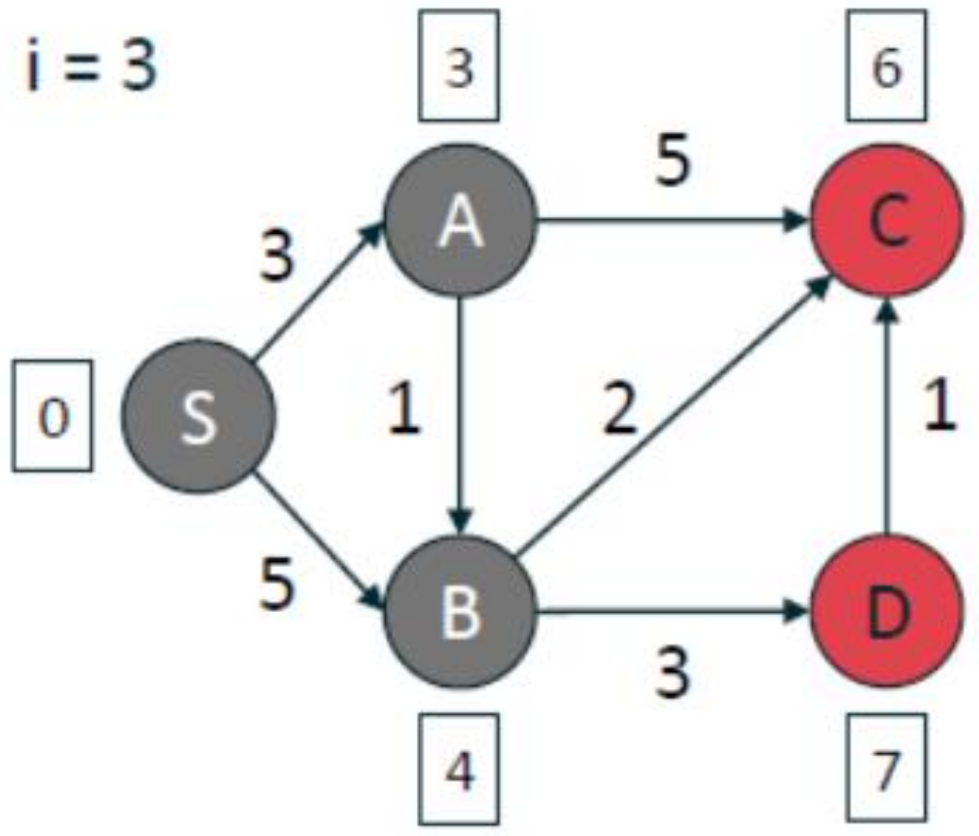


# *Single Source Shortest Paths*





# *Single Source Shortest Paths*



## *Single Source Shortest Paths*

```
public void compute(Vertex<Long, Double> vertex, MessageIterator<Double> messages) {

    double minDistance = (vertex.getId().equals(srcId)) ? 0d : Double.POSITIVE_INFINITY;

    for (Double msg : messages) {
        minDistance = Math.min(minDistance, msg);
    }

    if (minDistance < vertex.getValue()) {
        setNewVertexValue(minDistance);
        for (Edge<Long, Double> e: getEdges()) {
            sendMessageTo(e.getTarget(), minDistance + e.getValue());
        }
    }
}
```

Vertex-centric

**VertexUpdateFunction:** defines how a vertex will update its value based on the received messages

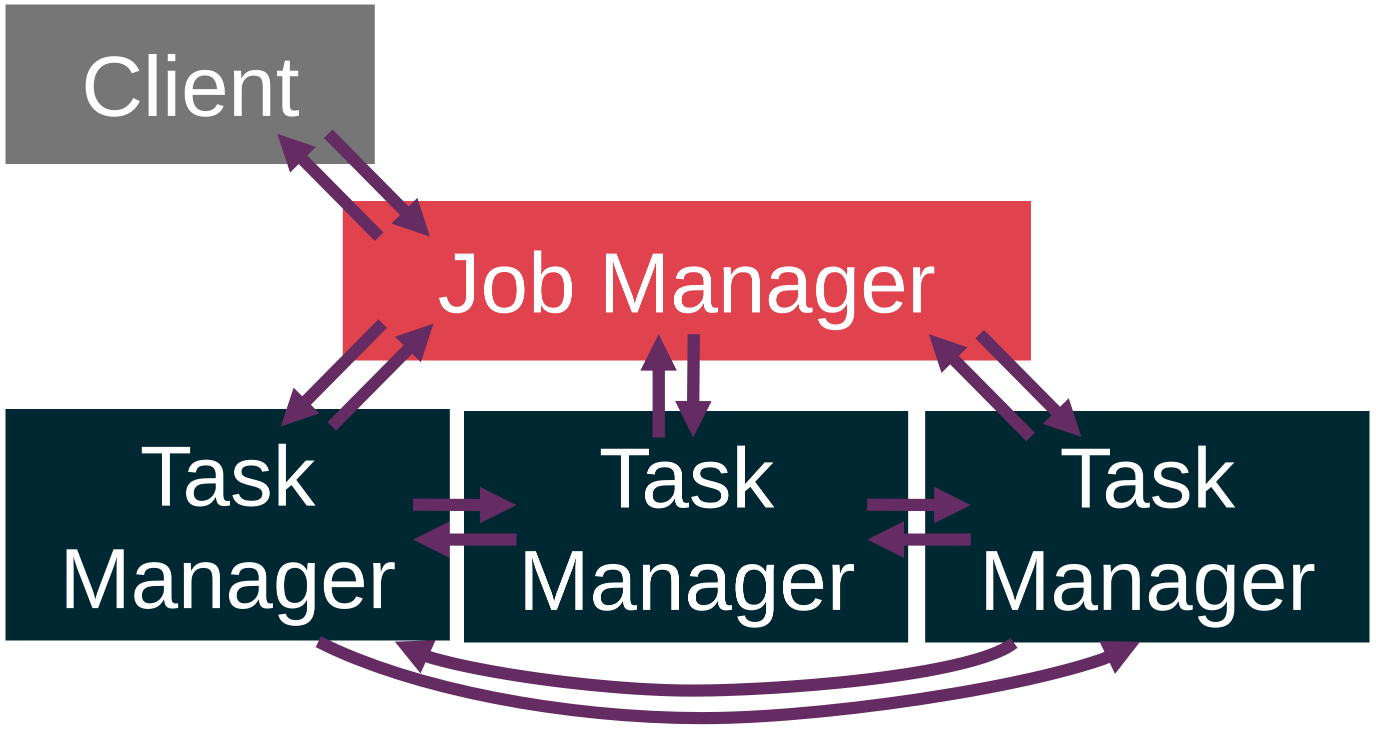
# *Single Source Shortest Paths*

```
public static final class SSSPCombiner extends MessageCombiner<Long, Double> {

    public void combineMessages(MessageIterator<Double> messages) {

        double minMessage = Double.POSITIVE_INFINITY;
        for (Double msg: messages) {
            minMessage = Math.min(minMessage, msg);
        }
        sendCombinedMessage(minMessage);
    }
}
```

Vertex-centric

**Messaging Function:** defines what messages a vertex sends out for the next superstep

# *Architecture Overview*



**1) Client**

- Optimize
- Construct job graph
- Pass job graph to manager
- Retrieve job results

**2) Master (Job Manager)**

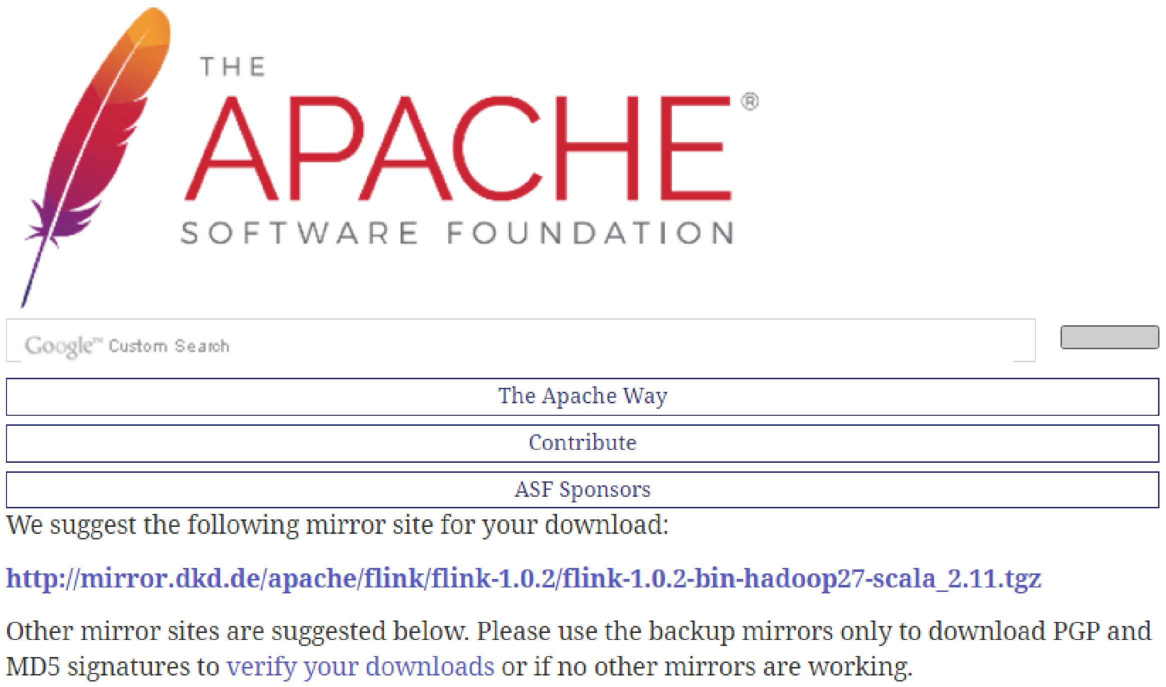
- Parallelization:
  *Create Execution Graph*
- Scheduling:
  *Assign tasks to task managers*
- State:
  *Supervise the execution*

**3) Worker (Task Manager)**

- Operations are split up into **tasks**
- Each parallel instance of an operation runs in a separate **tasks slot**

# *Installation Flink-1.0.2, Hadoop 2.7*

```
▪ wget http://ftp.fau.de/apache/flink/flink-1.0.2/flink-1.0.2-bin-hadoop27-scala_2.11.tgz
▪ tar xf flink-1.0.2-bin-hadoop27-scala_2.11.tgz
```

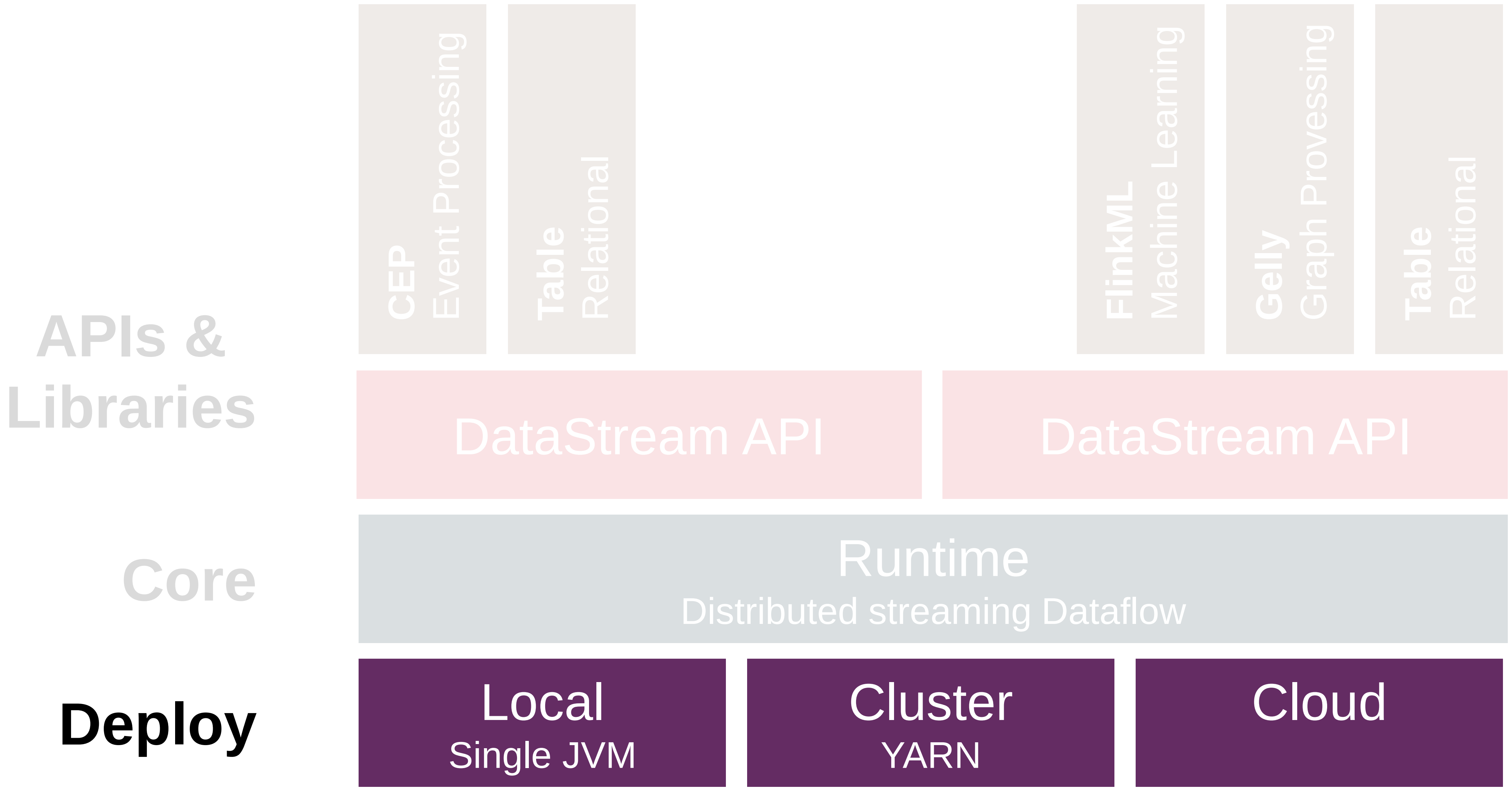


We suggest the following mirror site for your download:

**http://mirror.dkd.de/apache/flink/flink-1.0.2/flink-1.0.2-bin-hadoop27-scala_2.11.tgz**

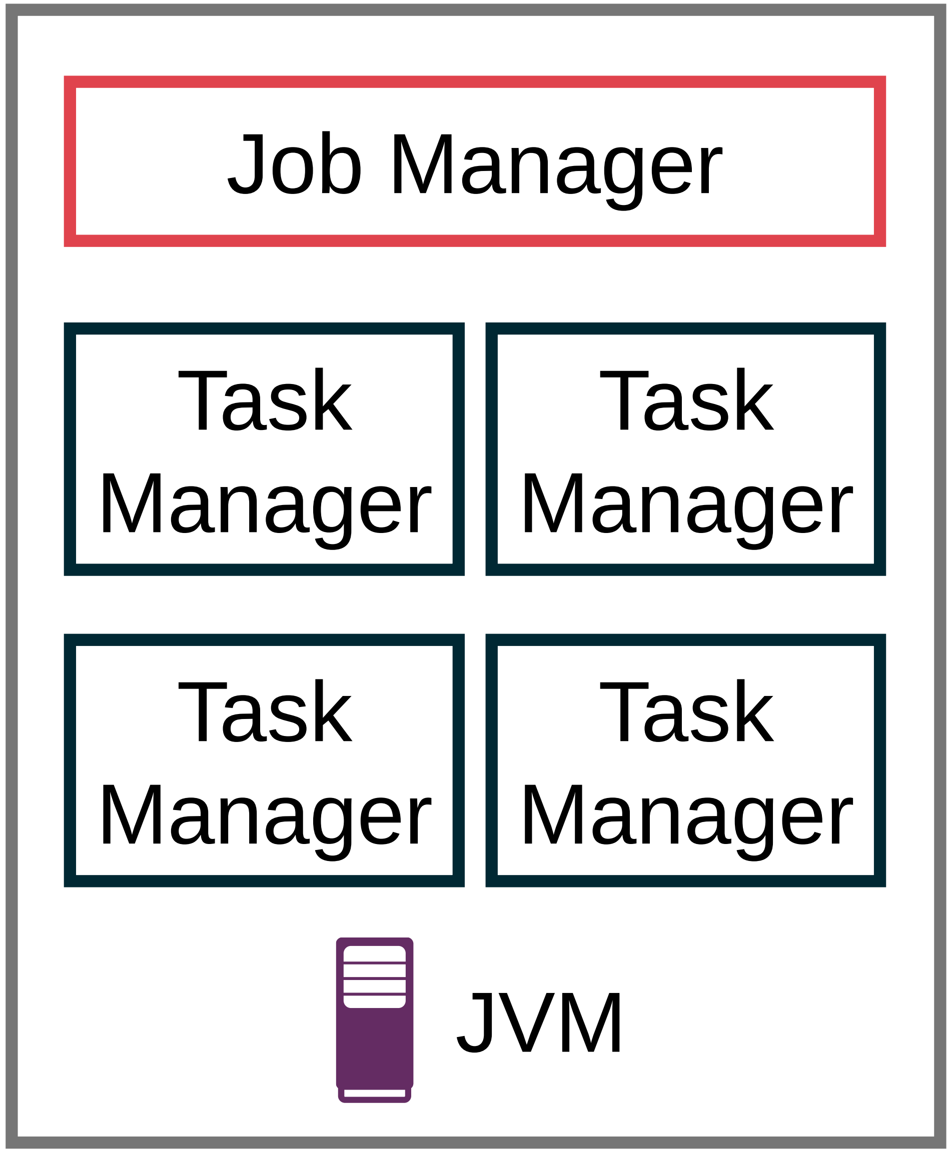
Other mirror sites are suggested below. Please use the backup mirrors only to download PGP and MD5 signatures to verify your downloads or if no other mirrors are working.

# *Ways to Run a Flink Program*



## *Local Execution*

- Starts local Flink cluster
- All processes run in the same JVM
- Behaves just like a regular Cluster
- Very useful for developing and debugging



# *Wordcount: Program*

```
case class Word (word: String, frequency: Int)

val env = ExecutionEnvironment.getExecutionEnvironment()

val lines: DataSet<String> = env.readTextFile(...)

lines
    .flatMap { line =>
      line.split(" ").map( word => Word(word, 1) )
    }
    .groupBy("word")
    .sum("frequency”)
    .print()

env.execute()
```

## *Source*

## *Sink*

*env.readCsvFile(...)*

*env.readFile(...)*

*env.readHadoopFile(...)*

*env.readSequenceFile(...)*

*env.readTextFile(...)*

*dataStream.print()*

*dataStream.writeAsText(...)*

*dataStream.writeAsCsv(...)*

**Collection-based**

**Others:**

SocketInputFormat

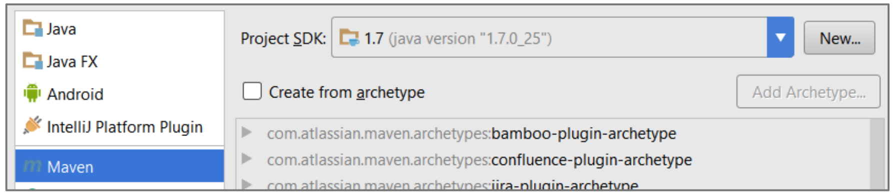
KafkaInputFormat

Databases

*JSONParser

# Creation of Flink Project

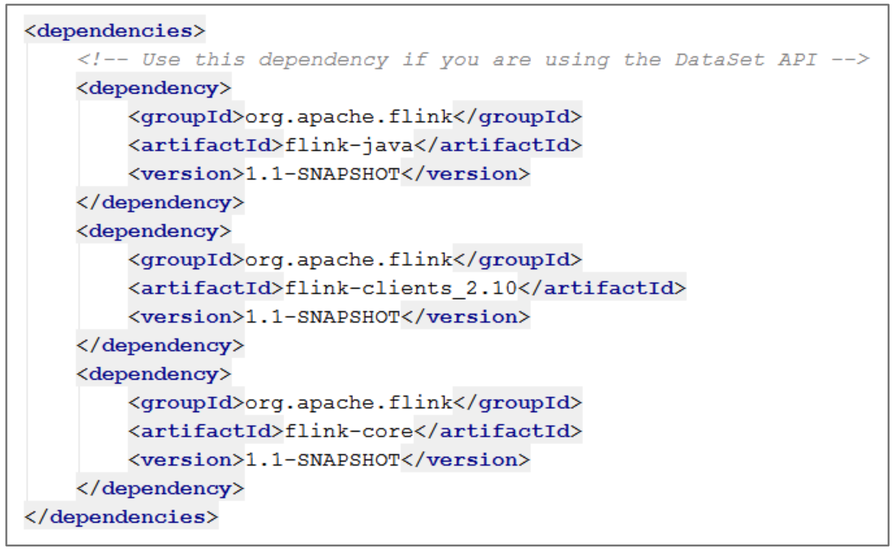
- Create empty Maven project (IntelliJ IDEA)
- Add Flink dependencies
- Change Manifest to specify Main Class
- Build to JAR



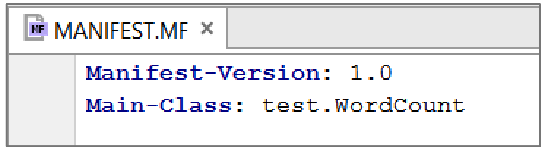
Project SDK: 1.7 (java version "1.7.0_25")

Create from archetype

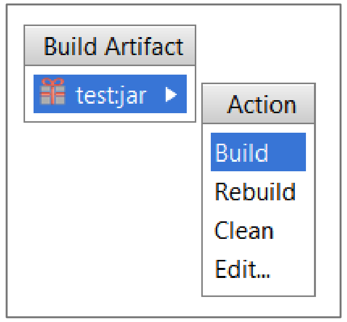
```
com.atlassian.maven.archetypes:bamboo-plugin-archetype
com.atlassian.maven.archetypes:confluence-plugin-archetype
com.atlassian.maven.archetypes:jira-plugin-archetype
```



```xml
<dependencies>
    <!-- Use this dependency if you are using the DataSet API -->
    <dependency>
        <groupId>org.apache.flink</groupId>
        <artifactId>flink-java</artifactId>
        <version>1.1-SNAPSHOT</version>
    </dependency>
    <dependency>
        <groupId>org.apache.flink</groupId>
        <artifactId>flink-clients_2.10</artifactId>
        <version>1.1-SNAPSHOT</version>
    </dependency>
    <dependency>
        <groupId>org.apache.flink</groupId>
        <artifactId>flink-core</artifactId>
        <version>1.1-SNAPSHOT</version>
    </dependency>
</dependencies>
```

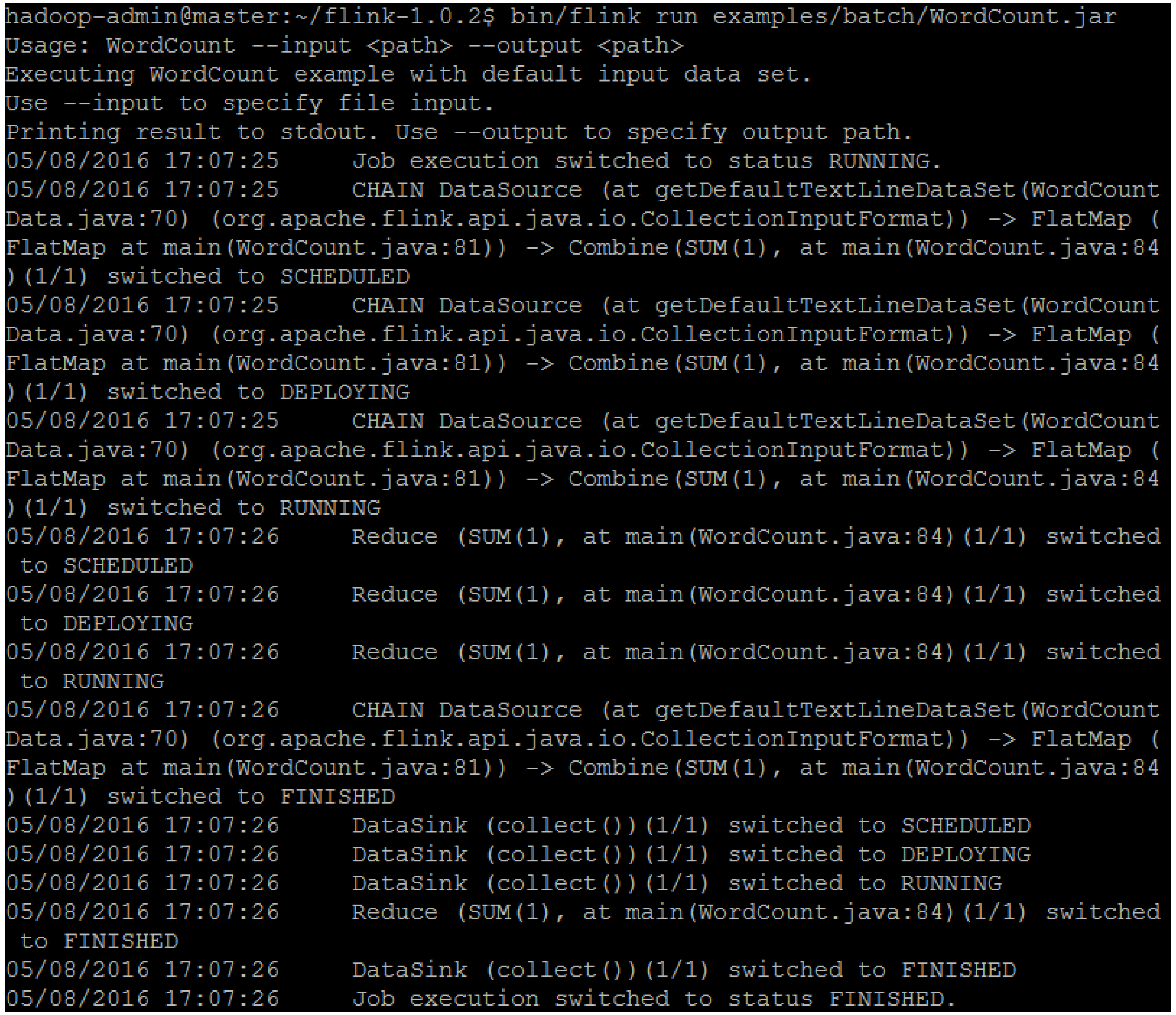


```
Manifest-Version: 1.0
Main-Class: test.WordCount
```



## *Local Execution: WordCount*

```
▪  bin/start-local.sh
▪  bin/flink run examples/batch/WordCount.jar
```



```
hadoop-admin@master:~/flink-1.0.2$ bin/flink run examples/batch/WordCount.jar
Usage: WordCount --input <path> --output <path>
Executing WordCount example with default input data set.
Use --input to specify file input.
Printing result to stdout. Use --output to specify output path.
05/08/2016 17:07:25     Job execution switched to status RUNNING.
05/08/2016 17:07:25     CHAIN DataSource (at getDefaultTextLineDataSet(WordCount
Data.java:70) (org.apache.flink.api.java.io.CollectionInputFormat)) -> FlatMap (
FlatMap at main(WordCount.java:81)) -> Combine(SUM(1), at main(WordCount.java:84
)(1/1) switched to SCHEDULED
05/08/2016 17:07:25     CHAIN DataSource (at getDefaultTextLineDataSet(WordCount
Data.java:70) (org.apache.flink.api.java.io.CollectionInputFormat)) -> FlatMap (
FlatMap at main(WordCount.java:81)) -> Combine(SUM(1), at main(WordCount.java:84
)(1/1) switched to DEPLOYING
05/08/2016 17:07:25     CHAIN DataSource (at getDefaultTextLineDataSet(WordCount
Data.java:70) (org.apache.flink.api.java.io.CollectionInputFormat)) -> FlatMap (
FlatMap at main(WordCount.java:81)) -> Combine(SUM(1), at main(WordCount.java:84
)(1/1) switched to RUNNING
05/08/2016 17:07:26     Reduce (SUM(1), at main(WordCount.java:84)(1/1) switched
 to SCHEDULED
05/08/2016 17:07:26     Reduce (SUM(1), at main(WordCount.java:84)(1/1) switched
 to DEPLOYING
05/08/2016 17:07:26     Reduce (SUM(1), at main(WordCount.java:84)(1/1) switched
 to RUNNING
05/08/2016 17:07:26     CHAIN DataSource (at getDefaultTextLineDataSet(WordCount
Data.java:70) (org.apache.flink.api.java.io.CollectionInputFormat)) -> FlatMap (
FlatMap at main(WordCount.java:81)) -> Combine(SUM(1), at main(WordCount.java:84
)(1/1) switched to FINISHED
05/08/2016 17:07:26     DataSink (collect())(1/1) switched to SCHEDULED
05/08/2016 17:07:26     DataSink (collect())(1/1) switched to DEPLOYING
05/08/2016 17:07:26     DataSink (collect())(1/1) switched to RUNNING
05/08/2016 17:07:26     Reduce (SUM(1), at main(WordCount.java:84)(1/1) switched
 to FINISHED
05/08/2016 17:07:26     DataSink (collect())(1/1) switched to FINISHED
05/08/2016 17:07:26     Job execution switched to status FINISHED.
```

Running

FlatMap *x3*

Reduce *x3*

FlatMap

DataSink *x3*

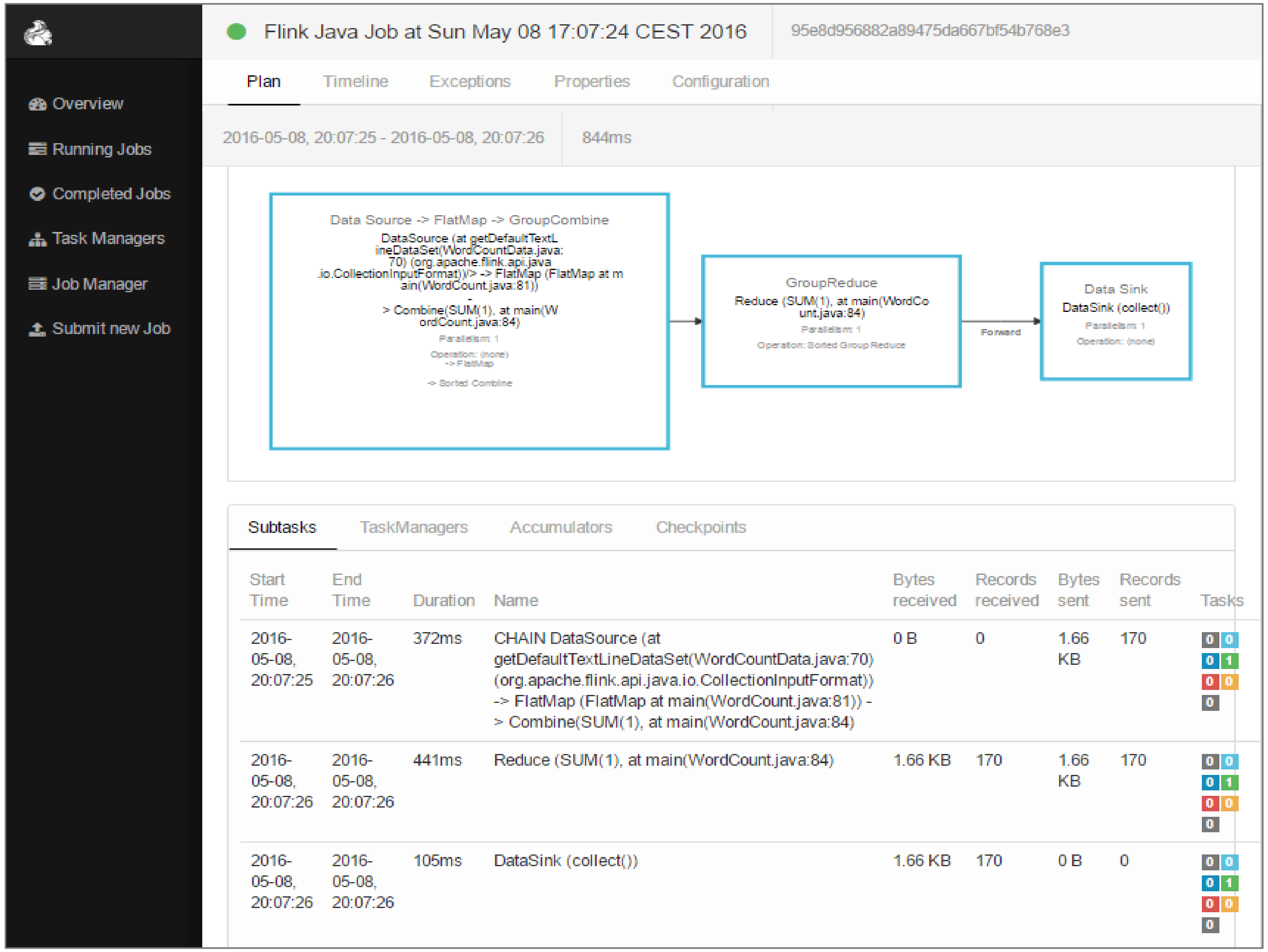
Reduce -> DataSink -> Finish

## *Local Execution: Result*

She sells sea shells on the sea shore;
The shells that she sells are sea shells I'm sure.
So if she sells sea shells on the sea shore,
I'm sure that the shells are sea shore shells

```
are 2          she 3
i 2            shells 6
if 1           shore 3
m 2            so 1
on 2           sure 2
sea 6          that 2
sells 3        the 4
```

# *Job Manager Web interface*



Overview
Running Jobs
Completed Jobs
Task Managers
Job Manager
Submit new Job
Plan
Timeline
Exceptions
Properties
Configuration
Data Source -> FlatMap -> GroupCombine
DataSource (at getDefaultTextLineDataSet(WordCountData.java:70) (org.apache.flink.api.java.io.CollectionInputFormat))/> -> FlatMap (FlatMap at main(WordCount.java:81)) -> Combine(SUM(1), at main(WordCount.java:84))
Parallelism: 1
Operation: (none) -> FlatMap -> Sorted Combine
GroupReduce
Reduce (SUM(1), at main(WordCount.java:84))
Parallelism: 1
Operation: Sorted Group Reduce
Forward
Data Sink
DataSink (collect())
Parallelism: 1
Operation: (none)
Subtasks
TaskManagers
Accumulators
Checkpoints

Flink Java Job at Sun May 08 17:07:24 CEST 2016 95e8d956882a89475da667bf54b768e3

2016-05-08, 20:07:25 - 2016-05-08, 20:07:26 844ms

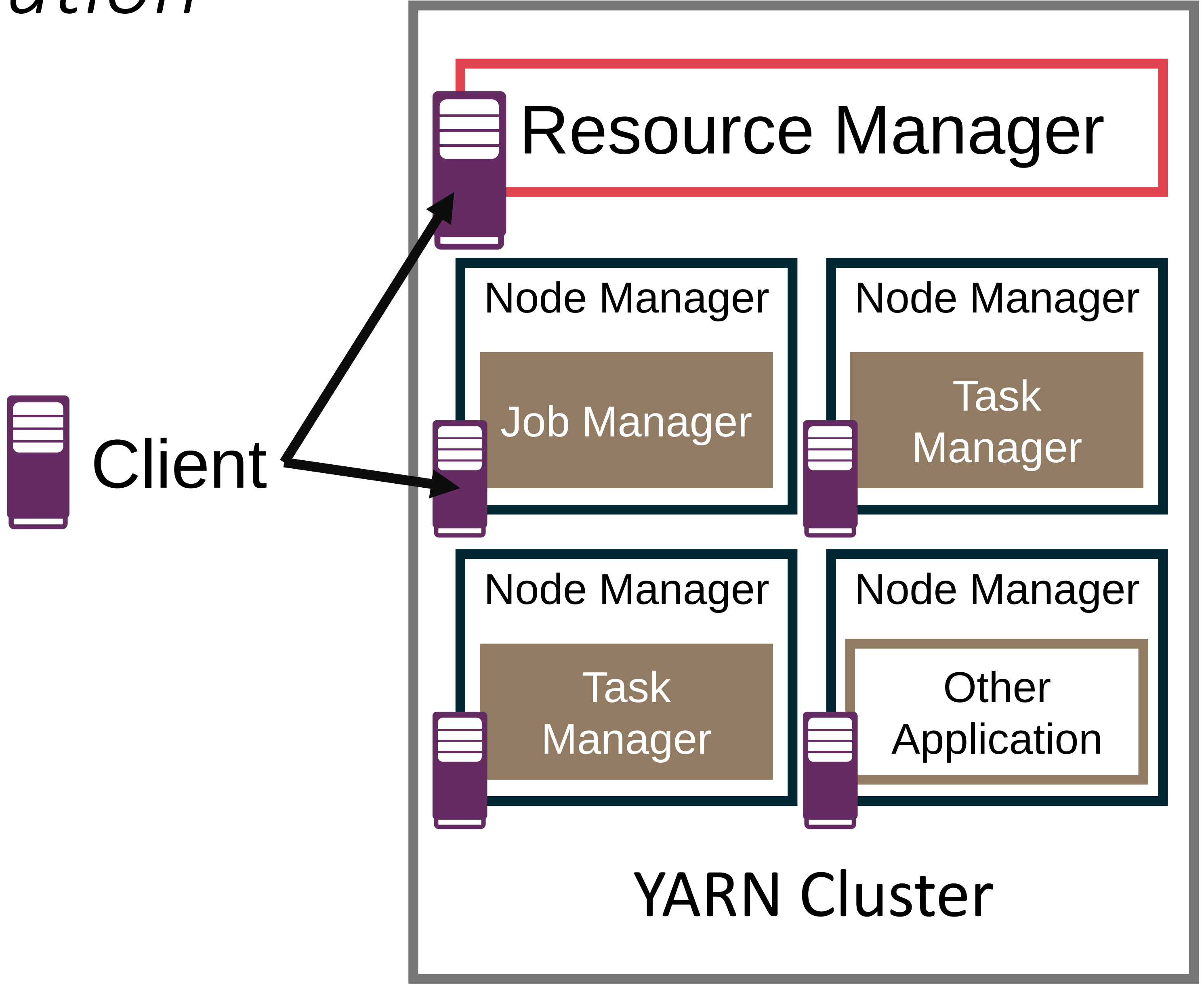
| Start Time | End Time | Duration | Name | Bytes received | Records received | Bytes sent | Records sent | Tasks |
|---|---|---|---|---|---|---|---|---|
| 2016-05-08, 20:07:25 | 2016-05-08, 20:07:26 | 372ms | CHAIN DataSource (at getDefaultTextLineDataSet(WordCountData.java:70) (org.apache.flink.api.java.io.CollectionInputFormat)) -> FlatMap (FlatMap at main(WordCount.java:81)) -> Combine(SUM(1), at main(WordCount.java:84)) | 0 B | 0 | 1.66 KB | 170 | 0 0 0 1 0 0 0 |
| 2016-05-08, 20:07:26 | 2016-05-08, 20:07:26 | 441ms | Reduce (SUM(1), at main(WordCount.java:84)) | 1.66 KB | 170 | 1.66 KB | 170 | 0 0 0 1 0 0 0 |
| 2016-05-08, 20:07:26 | 2016-05-08, 20:07:26 | 105ms | DataSink (collect()) | 1.66 KB | 170 | 0 B | 0 | 0 0 0 1 0 0 0 |

- `http://master:8081`

- Shows overall system status
- Job execution details
- Task Manager resource utilization
- Allow submit new job by form

# *Cluster. YARN Execution*

- Multi-user scenario
- Resource sharing
- Uses YARN containers to run a Flink cluster
- Easy to setup



## *Cluster. YARN Execution*

### Configure flink

```
▪ nano conf/flink-conf.yaml
    ▪ edit line >> jobmanager.rpc.address: 10.42.23.101
```

### Copy configuration on workers

```
▪ scp -r master:/home/hadoop-admin/flink-1.0.2 .
```

### Setup path to Hadoop

```
▪ export HADOOP_CONF_DIR='/opt/hadoop/etc/hadoop'
```

### Run Yarn session with 2 TaskManagers with 1GB of memory each

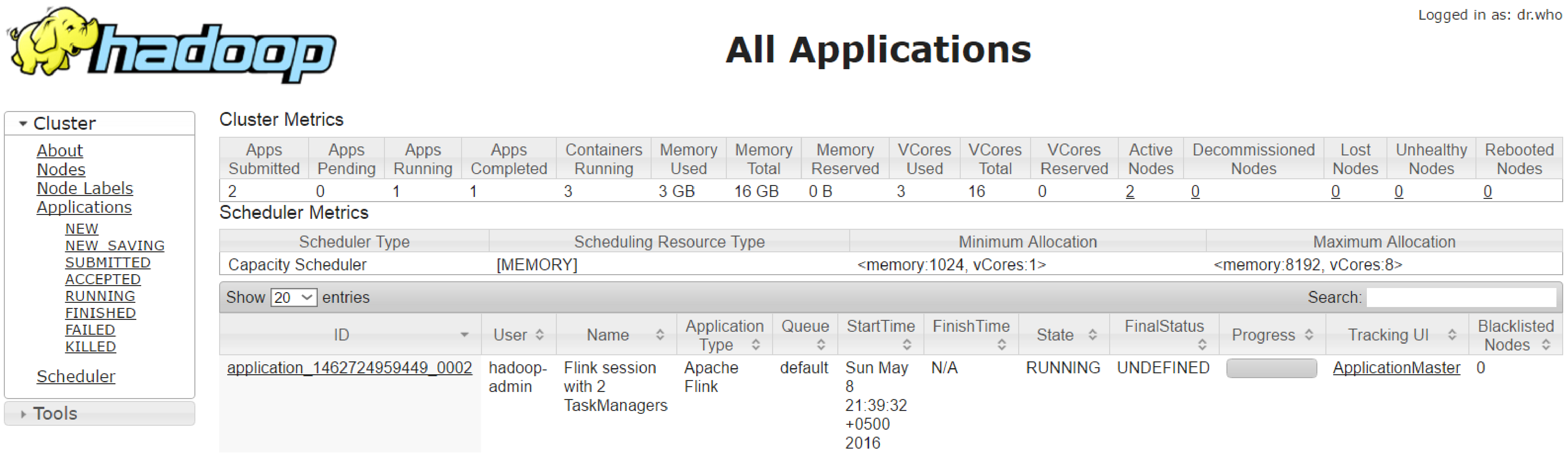
```
▪ bin/yarn-session.sh -n 2 -tm 1024
```

https://ci.apache.org/projects/flink/flink-docs-release-0.10/quickstart/setup_quickstart.html
https://ci.apache.org/projects/flink/flink-docs-release-0.8/yarn_setup.html

## *Cluster. YARN Execution*

### Run example

```
▪ bin/flink run -p 2 -m yarn-cluster -yn 2 -yjm 1024 -ytm 1024
  examples/batch/WordCount.jar
```



Logged in as: dr.who

**All Applications**

Cluster Metrics

| Apps Submitted | Apps Pending | Apps Running | Apps Completed | Containers Running | Memory Used | Memory Total | Memory Reserved | VCores Used | VCores Total | VCores Reserved | Active Nodes | Decommissioned Nodes | Lost Nodes | Unhealthy Nodes | Rebooted Nodes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 0 | 1 | 1 | 3 | 3 GB | 16 GB | 0 B | 3 | 16 | 0 | 2 | 0 | 0 | 0 | 0 |

Scheduler Metrics

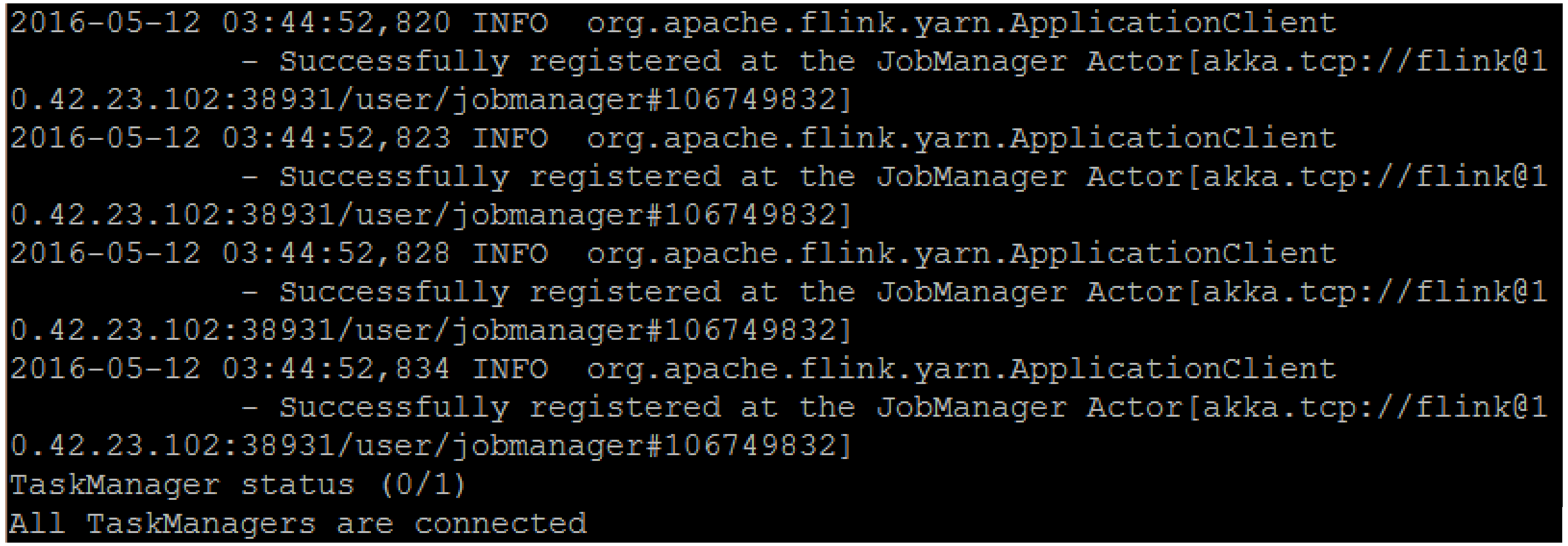
| Scheduler Type | Scheduling Resource Type | Minimum Allocation | Maximum Allocation |
|---|---|---|---|
| Capacity Scheduler | [MEMORY] | <memory:1024, vCores:1> | <memory:8192, vCores:8> |

Show 20 entries Search:

| ID | User | Name | Application Type | Queue | StartTime | FinishTime | State | FinalStatus | Progress | Tracking UI | Blacklisted Nodes |
|---|---|---|---|---|---|---|---|---|---|---|---|
| application_1462724959449_0002 | hadoop-admin | Flink session with 2 TaskManagers | Apache Flink | default | Sun May 8 21:39:32 +0500 2016 | N/A | RUNNING | UNDEFINED | | ApplicationMaster | 0 |

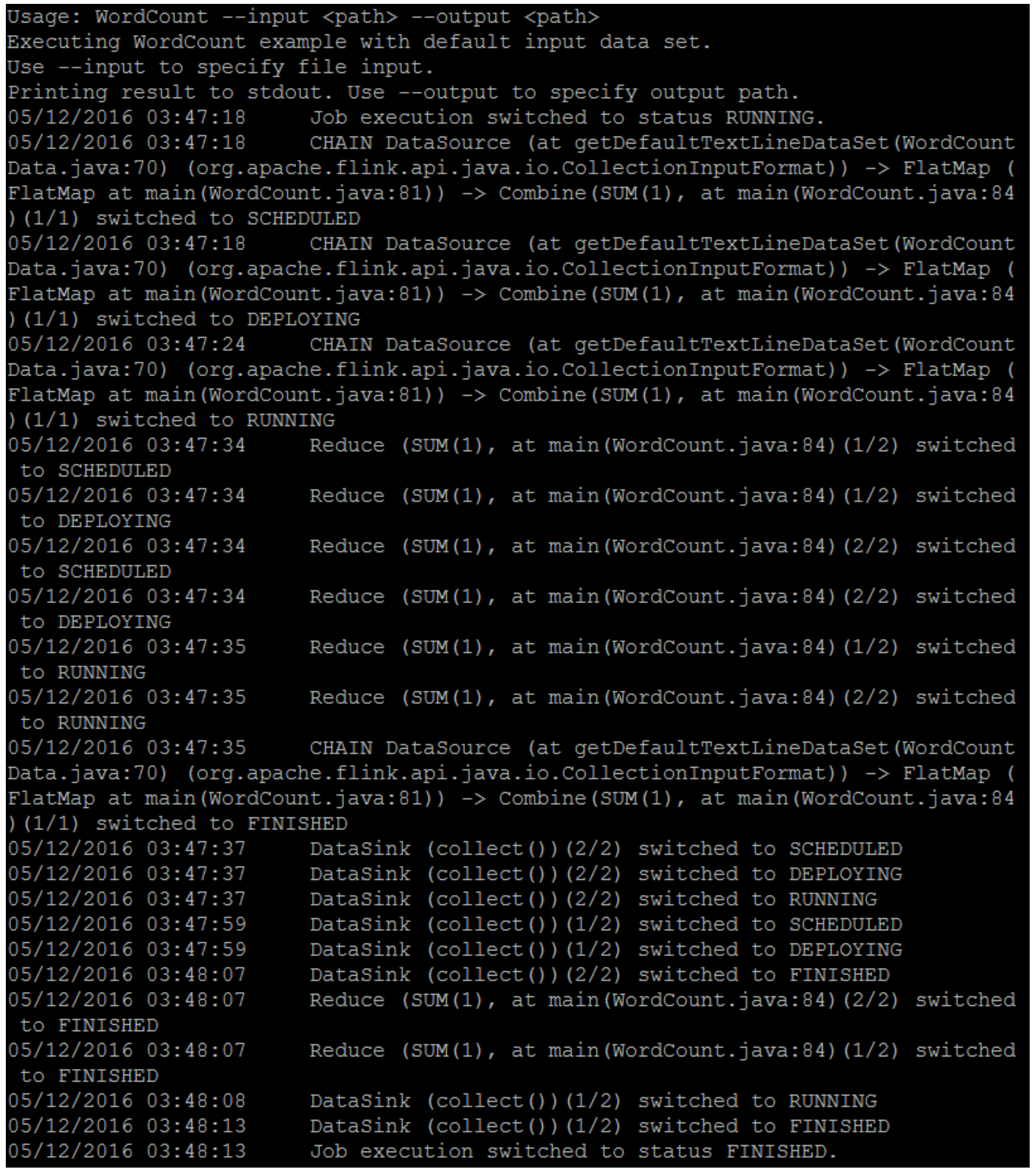
# *Cluster. YARN Execution*

## Assigning jobs to Task Managers



```
2016-05-12 03:44:52,820 INFO  org.apache.flink.yarn.ApplicationClient
            - Successfully registered at the JobManager Actor[akka.tcp://flink@1
0.42.23.102:38931/user/jobmanager#106749832]
2016-05-12 03:44:52,823 INFO  org.apache.flink.yarn.ApplicationClient
            - Successfully registered at the JobManager Actor[akka.tcp://flink@1
0.42.23.102:38931/user/jobmanager#106749832]
2016-05-12 03:44:52,828 INFO  org.apache.flink.yarn.ApplicationClient
            - Successfully registered at the JobManager Actor[akka.tcp://flink@1
0.42.23.102:38931/user/jobmanager#106749832]
2016-05-12 03:44:52,834 INFO  org.apache.flink.yarn.ApplicationClient
            - Successfully registered at the JobManager Actor[akka.tcp://flink@1
0.42.23.102:38931/user/jobmanager#106749832]
TaskManager status (0/1)
All TaskManagers are connected
```

# Cluster. YARN Execution



```
Usage: WordCount --input <path> --output <path>
Executing WordCount example with default input data set.
Use --input to specify file input.
Printing result to stdout. Use --output to specify output path.
05/12/2016 03:47:18     Job execution switched to status RUNNING.
05/12/2016 03:47:18     CHAIN DataSource (at getDefaultTextLineDataSet(WordCount
Data.java:70) (org.apache.flink.api.java.io.CollectionInputFormat)) -> FlatMap (
FlatMap at main(WordCount.java:81)) -> Combine(SUM(1), at main(WordCount.java:84
)(1/1) switched to SCHEDULED
05/12/2016 03:47:18     CHAIN DataSource (at getDefaultTextLineDataSet(WordCount
Data.java:70) (org.apache.flink.api.java.io.CollectionInputFormat)) -> FlatMap (
FlatMap at main(WordCount.java:81)) -> Combine(SUM(1), at main(WordCount.java:84
)(1/1) switched to DEPLOYING
05/12/2016 03:47:24     CHAIN DataSource (at getDefaultTextLineDataSet(WordCount
Data.java:70) (org.apache.flink.api.java.io.CollectionInputFormat)) -> FlatMap (
FlatMap at main(WordCount.java:81)) -> Combine(SUM(1), at main(WordCount.java:84
)(1/1) switched to RUNNING
05/12/2016 03:47:34     Reduce (SUM(1), at main(WordCount.java:84)(1/2) switched
 to SCHEDULED
05/12/2016 03:47:34     Reduce (SUM(1), at main(WordCount.java:84)(1/2) switched
 to DEPLOYING
05/12/2016 03:47:34     Reduce (SUM(1), at main(WordCount.java:84)(2/2) switched
 to SCHEDULED
05/12/2016 03:47:34     Reduce (SUM(1), at main(WordCount.java:84)(2/2) switched
 to DEPLOYING
05/12/2016 03:47:35     Reduce (SUM(1), at main(WordCount.java:84)(1/2) switched
 to RUNNING
05/12/2016 03:47:35     Reduce (SUM(1), at main(WordCount.java:84)(2/2) switched
 to RUNNING
05/12/2016 03:47:35     CHAIN DataSource (at getDefaultTextLineDataSet(WordCount
Data.java:70) (org.apache.flink.api.java.io.CollectionInputFormat)) -> FlatMap (
FlatMap at main(WordCount.java:81)) -> Combine(SUM(1), at main(WordCount.java:84
)(1/1) switched to FINISHED
05/12/2016 03:47:37     DataSink (collect())(2/2) switched to SCHEDULED
05/12/2016 03:47:37     DataSink (collect())(2/2) switched to DEPLOYING
05/12/2016 03:47:37     DataSink (collect())(2/2) switched to RUNNING
05/12/2016 03:47:59     DataSink (collect())(1/2) switched to SCHEDULED
05/12/2016 03:47:59     DataSink (collect())(1/2) switched to DEPLOYING
05/12/2016 03:48:07     DataSink (collect())(2/2) switched to FINISHED
05/12/2016 03:48:07     Reduce (SUM(1), at main(WordCount.java:84)(2/2) switched
 to FINISHED
05/12/2016 03:48:07     Reduce (SUM(1), at main(WordCount.java:84)(1/2) switched
 to FINISHED
05/12/2016 03:48:08     DataSink (collect())(1/2) switched to RUNNING
05/12/2016 03:48:13     DataSink (collect())(1/2) switched to FINISHED
05/12/2016 03:48:13     Job execution switched to status FINISHED.
```

Running

FlatMap *x3*

Reduce *x6*

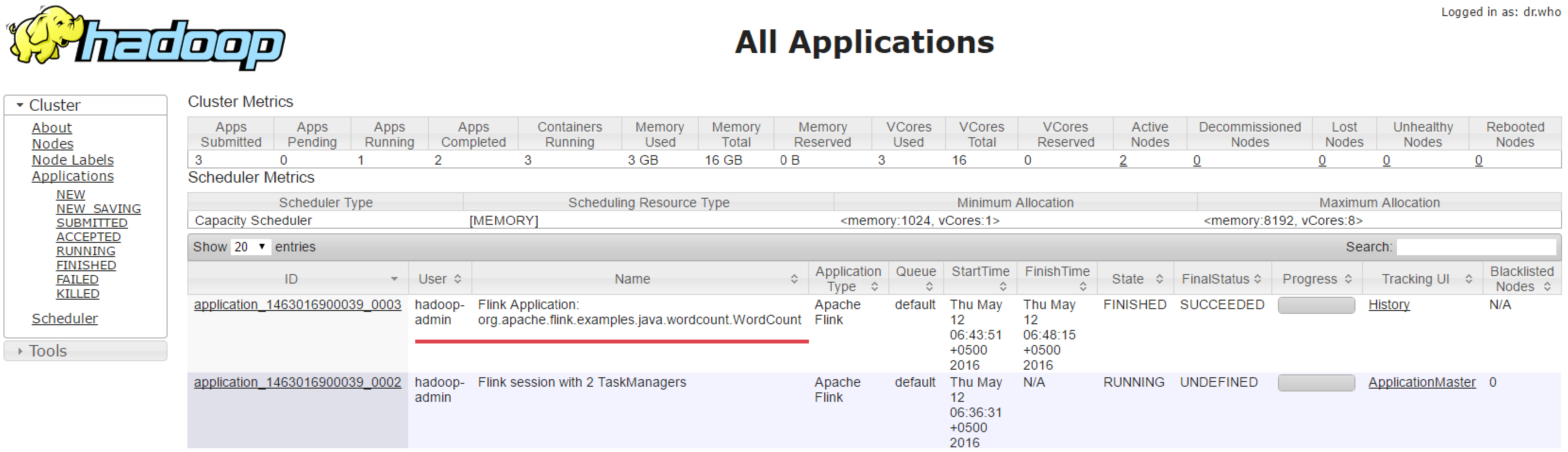
FlatMap

DataSink *x6*

Reduce *x2*

DataSink *x2 -> Finish*

# Cluster. YARN Execution



Logged in as: dr.who

## All Applications

- Cluster
  - About
  - Nodes
  - Node Labels
  - Applications
    - NEW
    - NEW_SAVING
    - SUBMITTED
    - ACCEPTED
    - RUNNING
    - FINISHED
    - FAILED
    - KILLED
  - Scheduler
- Tools

Cluster Metrics

| Apps Submitted | Apps Pending | Apps Running | Apps Completed | Containers Running | Memory Used | Memory Total | Memory Reserved | VCores Used | VCores Total | VCores Reserved | Active Nodes | Decommissioned Nodes | Lost Nodes | Unhealthy Nodes | Rebooted Nodes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 0 | 1 | 2 | 3 | 3 GB | 16 GB | 0 B | 3 | 16 | 0 | 2 | 0 | 0 | 0 | 0 |

Scheduler Metrics

| Scheduler Type | Scheduling Resource Type | Minimum Allocation | Maximum Allocation |
|---|---|---|---|
| Capacity Scheduler | [MEMORY] | <memory:1024, vCores:1> | <memory:8192, vCores:8> |

Show 20 entries Search:

| ID | User | Name | Application Type | Queue | StartTime | FinishTime | State | FinalStatus | Progress | Tracking UI | Blacklisted Nodes |
|---|---|---|---|---|---|---|---|---|---|---|---|
| application_1463016900039_0003 | hadoop-admin | Flink Application: org.apache.flink.examples.java.wordcount.WordCount | Apache Flink | default | Thu May 12 06:43:51 +0500 2016 | Thu May 12 06:48:15 +0500 2016 | FINISHED | SUCCEEDED | | History | N/A |
| application_1463016900039_0002 | hadoop-admin | Flink session with 2 TaskManagers | Apache Flink | default | Thu May 12 06:36:31 +0500 2016 | N/A | RUNNING | UNDEFINED | | ApplicationMaster | 0 |

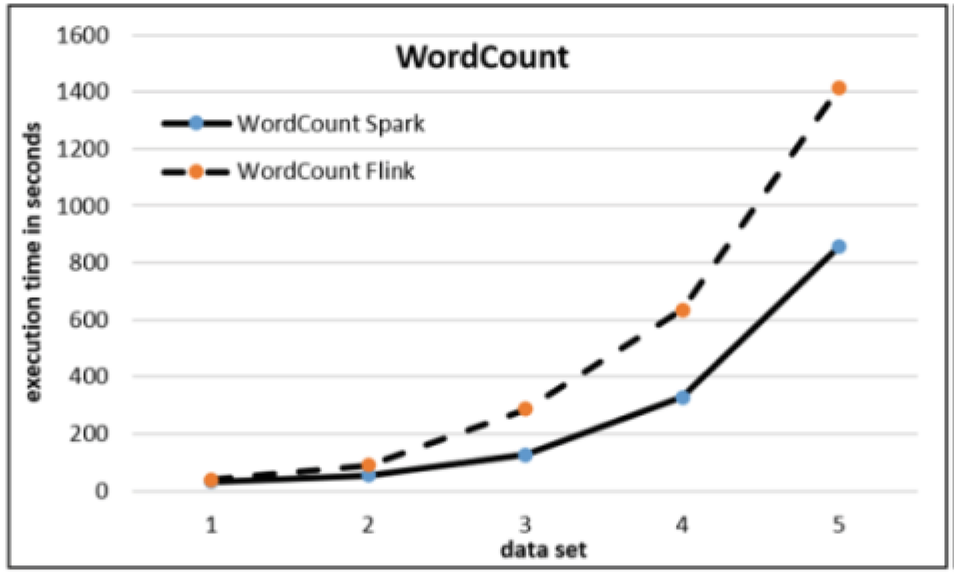
# Flink vs. Spark: Batch

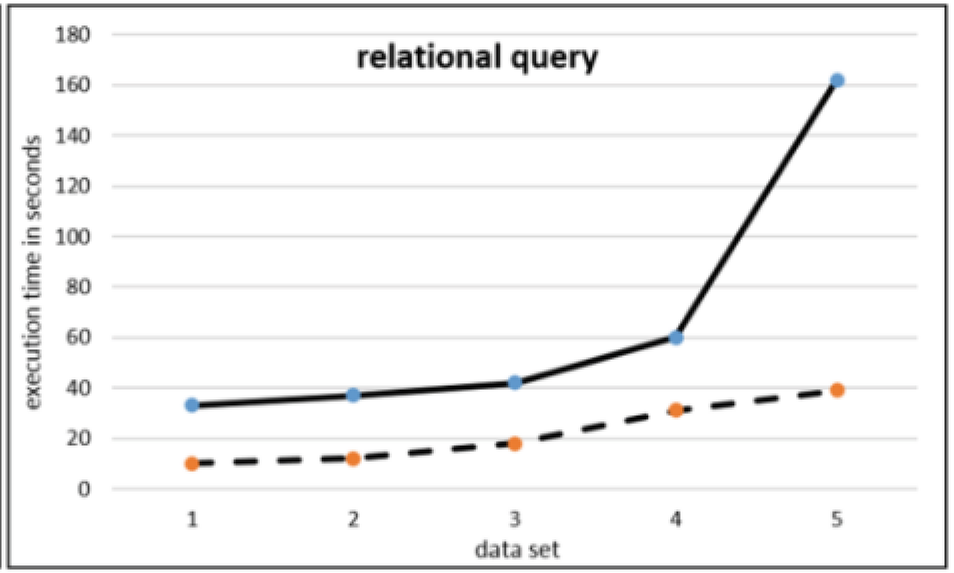
| | | |
|---|---|---|
| API | high-level | high-level |
| Data Transfer | batch | pipelined & batch |
| Memory Management | JVM-managed | Active managed |
| Iterations | in-memory cached | streamed |
| Fault tolerance | task level | job level |
| Good at | data exploration | heavy backend & iterative jobs |
| Libraries | built-in & external | evolving built-in & external |

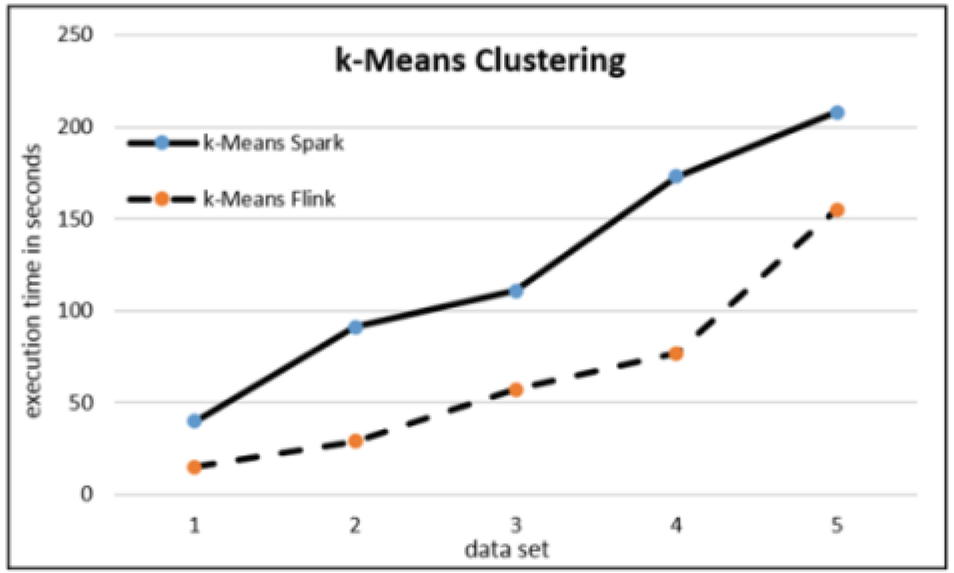
## *Flink vs. Spark: Streaming*

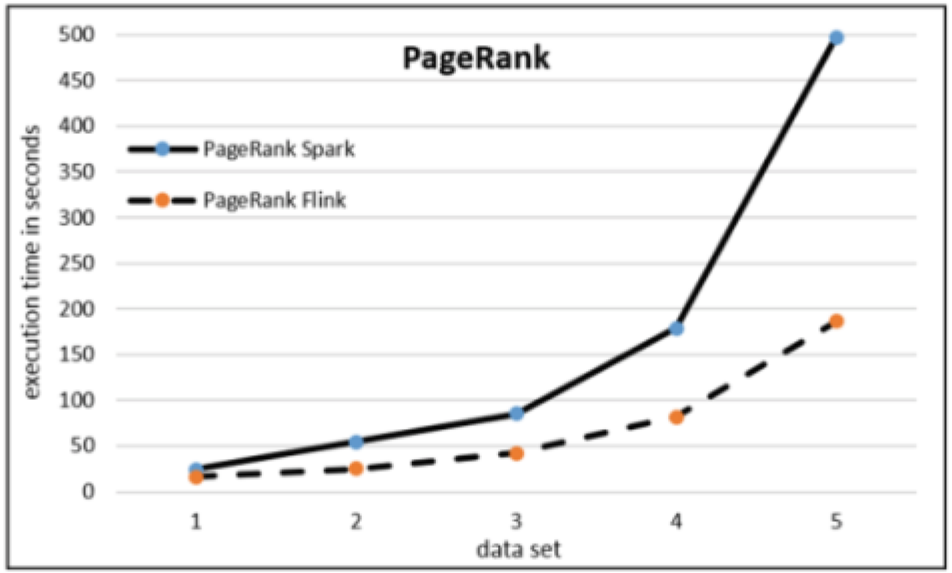
| | | |
|---|---|---|
| **Streaming** | mini batches | "true" |
| **API** | high-level | high-level |
| **Fault tolerance** | RDD-based (lineage) | coarse checkpointing |
| **State** | external | internal |
| **Exactly once** | exactly once | exactly once |
| **Windowing** | restricted | flexible |
| **Latency** | medium | low |
| **Throughput** | high | high |

# *Flink vs. Spark: Batch*









## Thank you