Data Analysis With Apache Flink

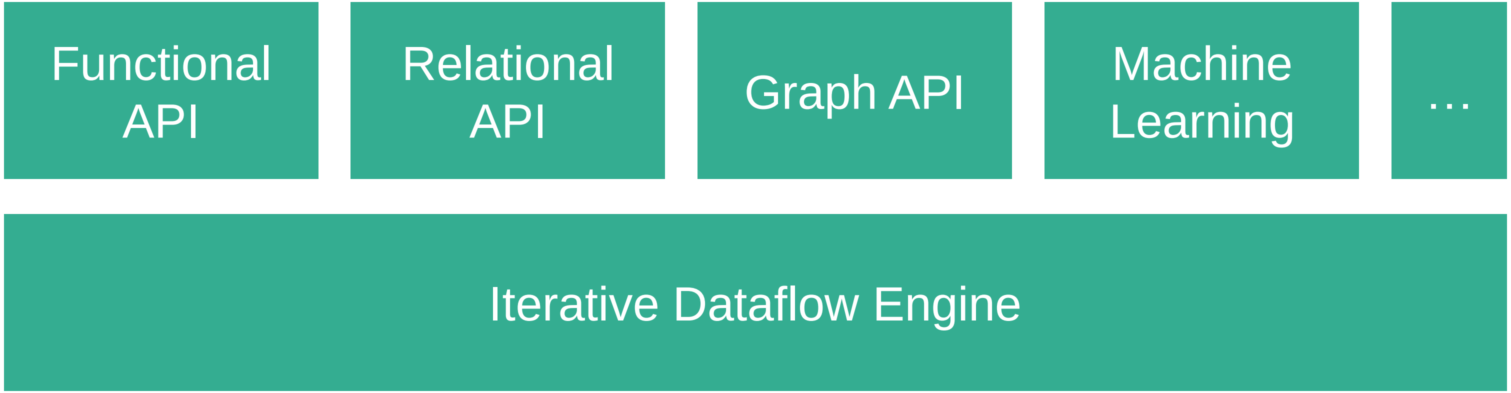
Aljoscha Krettek / Till Rohrmann

Flink committers

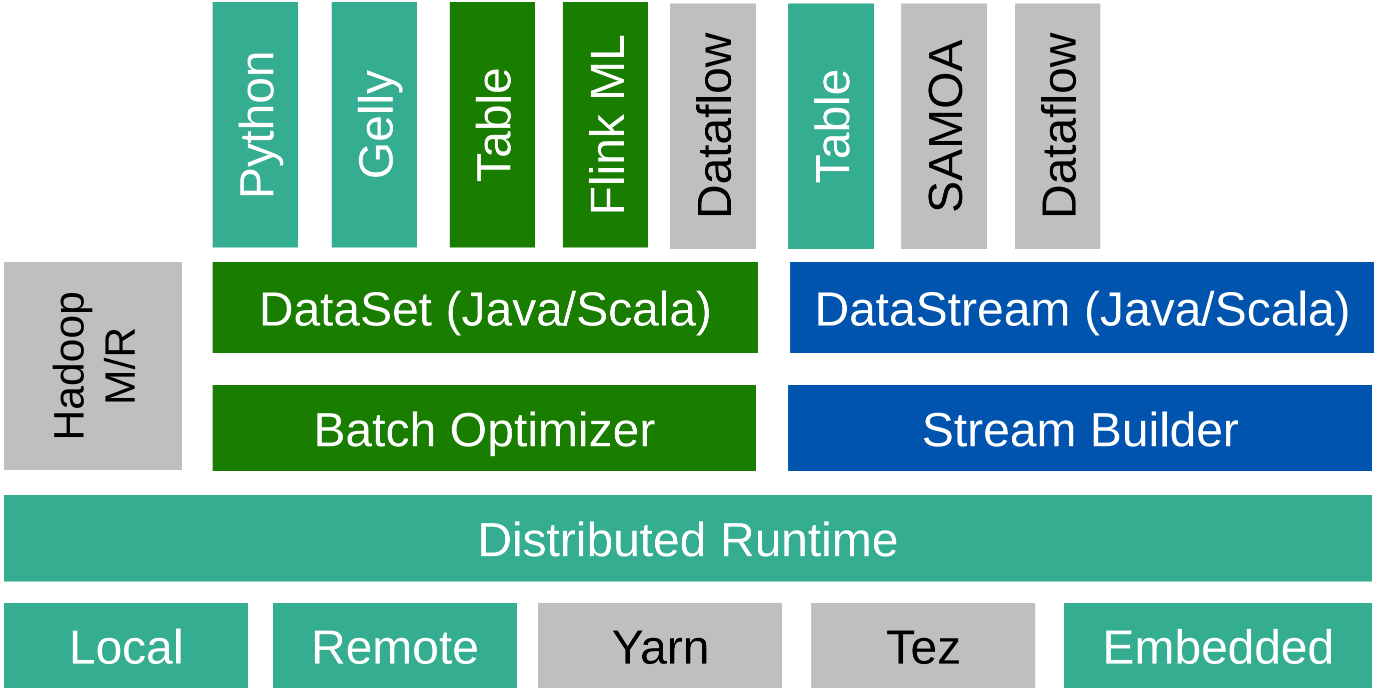
Co-founders @ *data Artisans*

aljoscha@apache.org / trohrmann@apache.org

What is Apache Flink?



Apache Flink Stack

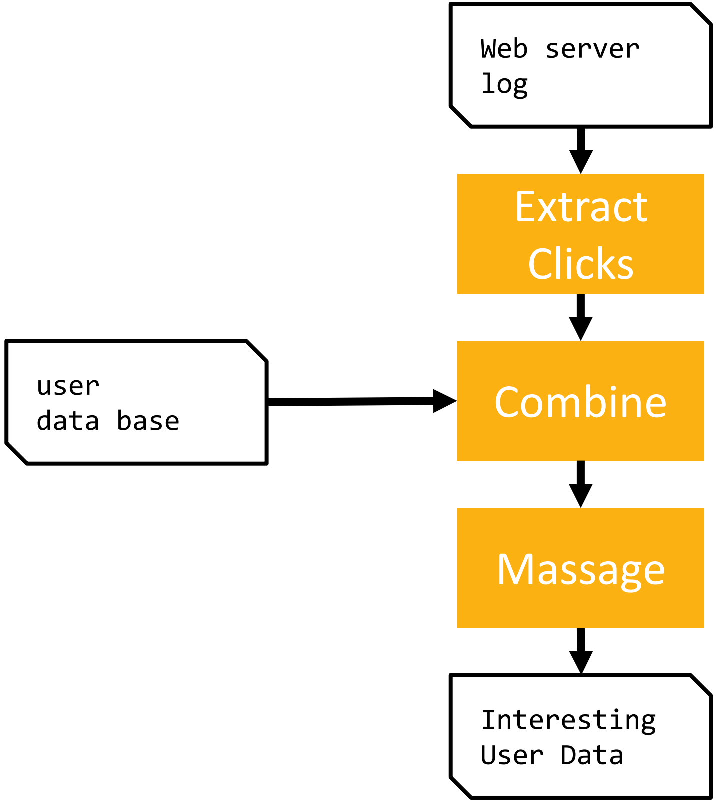


**current Flink master + few PRs*

Example Use Case: Log Analysis

What Seems to be the Problem?

- Collect clicks from a webserver log
- Find interesting URLs
- Combine with user data



The Execution Environment

- Entry point for all Flink programs
- Creates DataSets from data sources

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

Getting at Those Clicks

```
post /foo/bar… 313
get /data/pic.jpg 128
post /bar/baz… 128
post /hello/there… 42
```

```
DataSet<String> log = env.readTextFile("hdfs:///log");

DataSet<Tuple2<String, Integer>> clicks = log.flatMap(

 (String line, Collector<Tuple2<String, Integer>> out) ->
  String[] parts = in.split("*magic regex*");
  if (isClick(parts)) {
   out.collect(new Tuple2<>(parts[1],Integer.parseInt(parts[2])));
  }
 }
)
```

The Table Environment

- Environment for dealing with Tables
- Converts between DataSet and Table

```
TableEnvironment tableEnv = new TableEnvironment();
```

Counting those Clicks

```
Table clicksTable = tableEnv.toTable(clicks, "url, userId");

Table urlClickCounts = clicksTable
  .groupBy("url, userId")
  .select("url, userId, url.count as count");
```

Getting the User Information

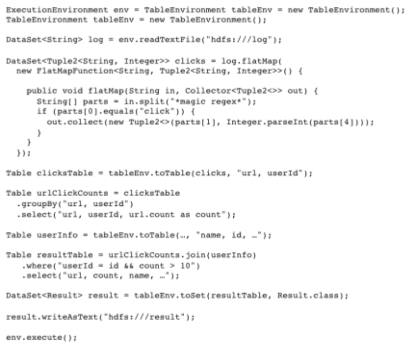
```
Table userInfo = tableEnv.toTable(…, "name, id, …");

Table resultTable = urlClickCounts.join(userInfo)
   .where("userId = id && count > 10")
   .select("url, count, name, …");
```

The Final Step

```
class Result {
  public String url;
  public int count;
  public String name;
  …
}

DataSet<Result> set =
  tableEnv.toSet(resultTable, Result.class);

DataSet<Result> result =
  set.groupBy("url").reduceGroup(new ComplexOperation());

result.writeAsText("hdfs:///result");
env.execute();
```

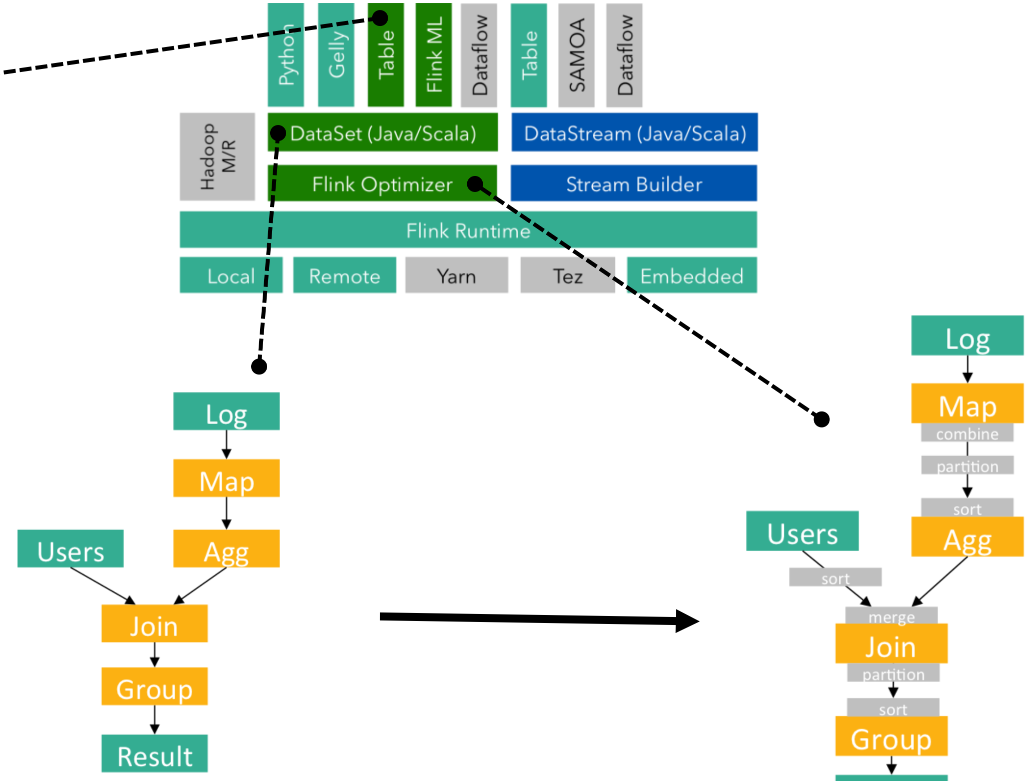
What happens under the hood?

From Program to Dataflow



```
ExecutionEnvironment env = TableEnvironment tableEnv = new TableEnvironment();
TableEnvironment tableEnv = new TableEnvironment();

DataSet<String> log = env.readTextFile("hdfs:///log");

DataSet<Tuple2<String, Integer>> clicks = log.flatMap(
  new FlatMapFunction<String, Tuple2<String, Integer>>() {

    public void flatMap(String in, Collector<Tuple2<>> out) {
      String[] parts = in.split("*magic regex*");
      if (parts[0].equals("click")) {
        out.collect(new Tuple2<>(parts[1], Integer.parseInt(parts[4])));
      }
    }
  });

Table clicksTable = tableEnv.toTable(clicks, "url, userId");

Table urlClickCounts = clicksTable
  .groupBy("url, userId")
  .select("url, userId, url.count as count");

Table userInfo = tableEnv.toTable(…, "name, id, …");

Table resultTable = urlClickCounts.join(userInfo)
   .where("userId = id && count > 10")
   .select("url, count, name, …");

DataSet<Result> result = tableEnv.toSet(resultTable, Result.class);

result.writeAsText("hdfs:///result");

env.execute();
```

Flink Program



Python
Gelly
Table
Flink ML
Dataflow
Table
SAMOA
Dataflow
Hadoop M/R
DataSet (Java/Scala)
DataStream (Java/Scala)
Flink Optimizer
Stream Builder
Flink Runtime
Local
Remote
Yarn
Tez
Embedded

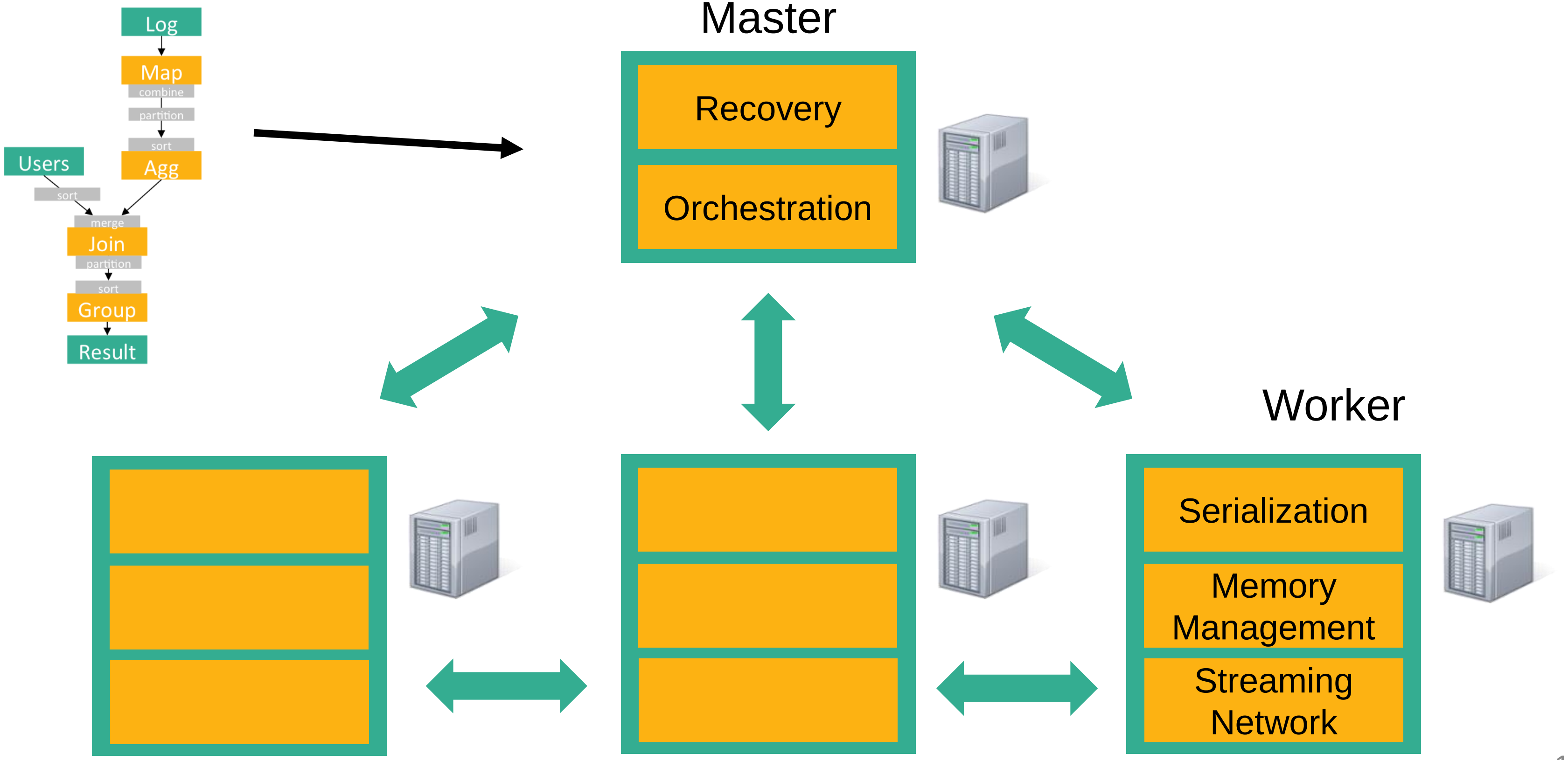
Log
Map
Users
Agg
Join
Group
Result

Log
Map
combine
partition
sort
Users
Agg
sort
merge
Join
partition
sort
Group
Result

Optimized Plan

Dataflow Plan

Distributed Execution



Log

Map

combine

partition

sort

Agg

Users

sort

merge

Join

partition

sort

Group

Result

Master

Recovery

Orchestration

Worker

Serialization

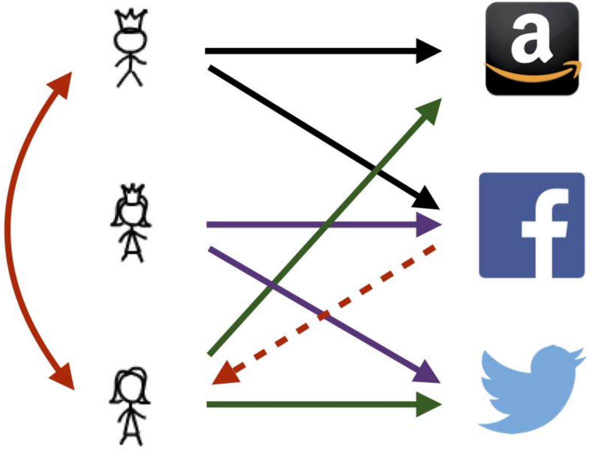
Memory Management

Streaming Network

Advanced Analysis:
Website Recommendation

Going Further

- Log analysis result: Which user visited how often which web site
- Which other websites might they like?
- Recommendation by collaborative filtering

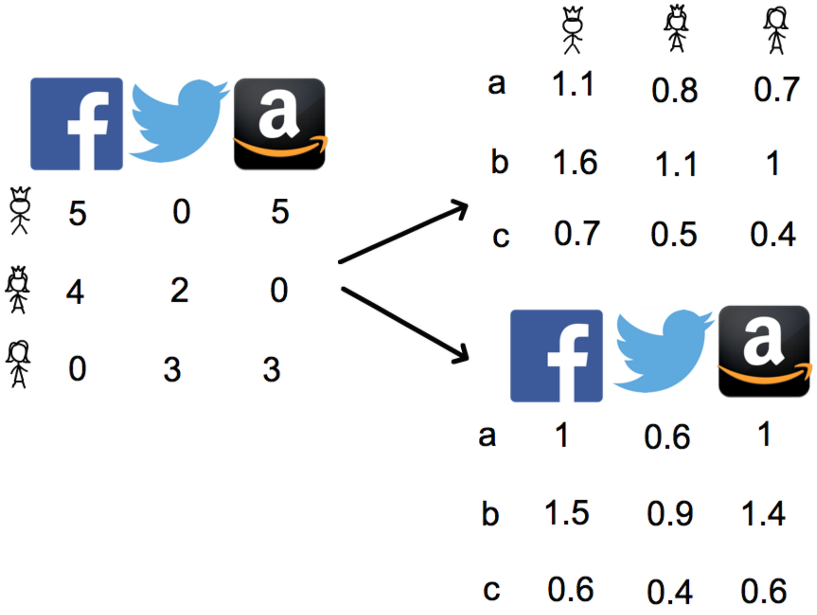


Collaborative Filtering

- Recommend items based on users with similar preferences
- Latent factor models capture underlying characteristics of items and preferences of user
- Predicted preference: $\hat{r}_{u,i} = x_u^T y_i$

Matrix Factorization

$$R \approx X^T Y$$



f	🐦	a
5	0	5
4	2	0
0	3	3

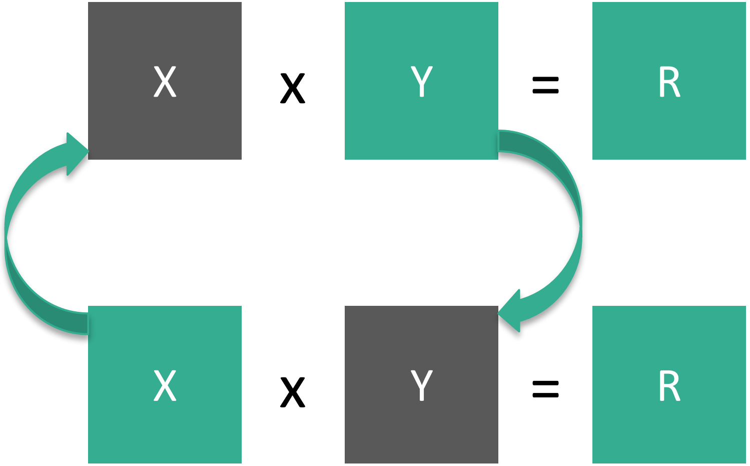
a	1.1	0.8	0.7
b	1.6	1.1	1
c	0.7	0.5	0.4

	f	🐦	a
a	1	0.6	1
b	1.5	0.9	1.4
c	0.6	0.4	0.6

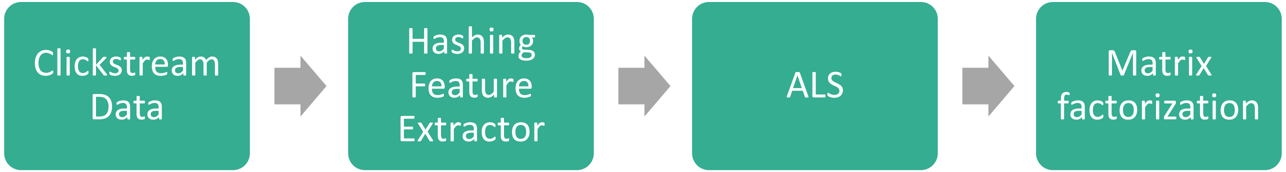
$$\min_{X,Y} \sum_{r_{u,i} \neq 0} \left(r_{u,i} - x_u^T y_i\right)^2 + \lambda \left(\sum_u \nu_u \|x_u\|^2 + \sum_i \nu_i \|y_i\|^2 \right)$$

Alternating least squares

- Iterative approximation
  1. Fix X and optimize Y
  2. Fix Y and optimize X
- Communication and computation intensive



Matrix Factorization Pipeline



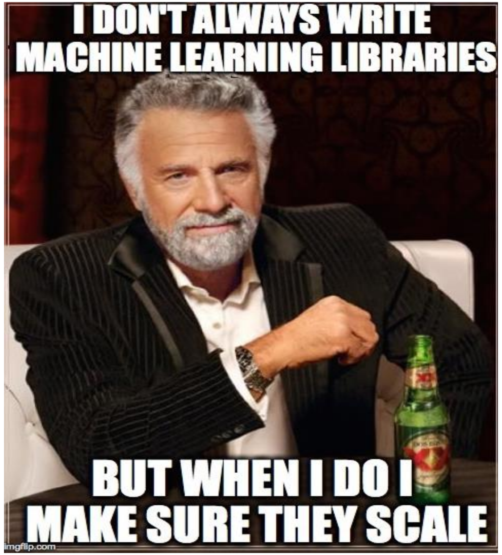
Clickstream Data

Hashing Feature Extractor

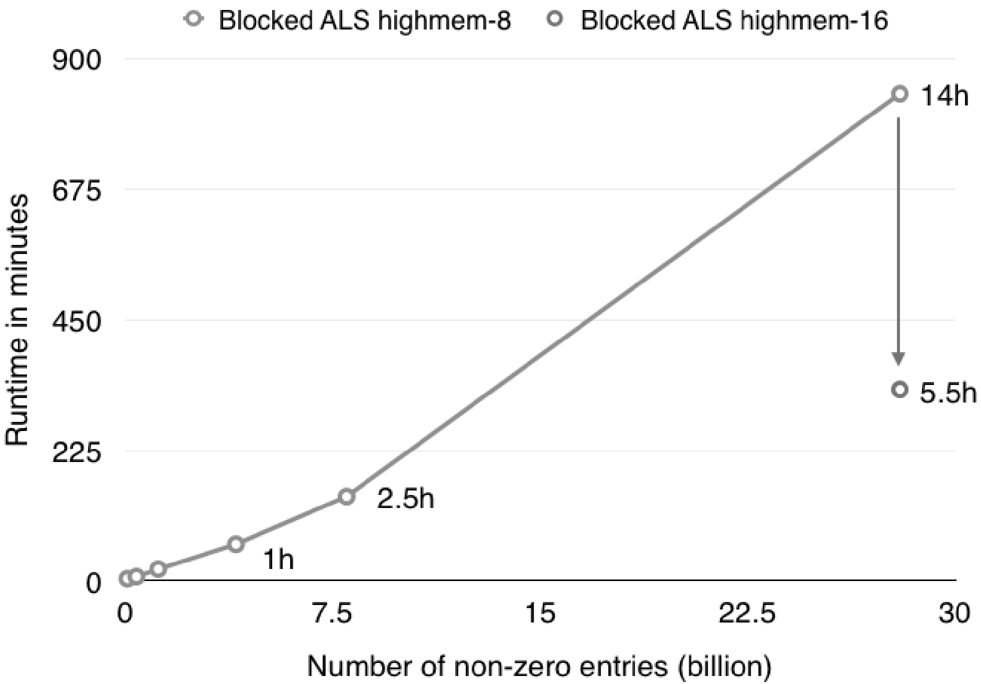
ALS

Matrix factorization

```
val featureExtractor = HashingFT()
val factorizer = ALS()

val pipeline = featureExtractor.chain(factorizer)

val clickstreamDS =
  env.readCsvFile[(String, String, Int)](clickStreamData)

val parameters = ParameterMap()
  .add(HashingFT.NumFeatures, 1000000)
  .add(ALS.Iterations, 10)
  .add(ALS.NumFactors, 50)
  .add(ALS.Lambda, 1.5)

val factorization = pipeline.fit(clickstreamDS, parameters)
```


I DON'T ALWAYS WRITE MACHINE LEARNING LIBRARIES
BUT WHEN I DO I MAKE SURE THEY SCALE
imgflip.com

Does it Scale?

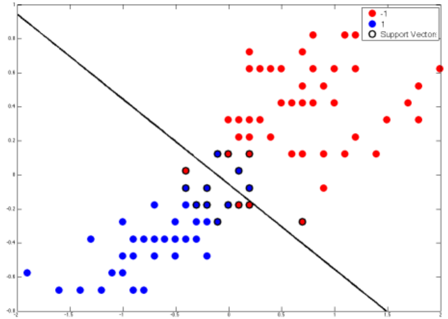


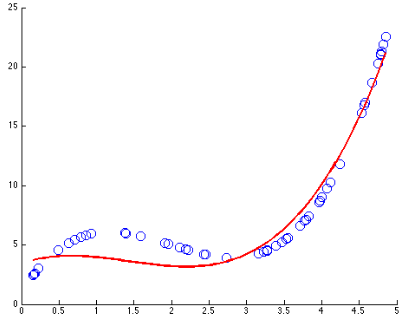
Scale of Netflix or Spotify

- 40 node GCE cluster, highmem-8
- 10 ALS iteration with 50 latent factors
- Based on Spark MLlib’s implementation

What Else Can You Do?

- Classification using SVMs
  - Conversion goal prediction
- Clustering
  - Visitor segmentation
- Multiple linear regression
  - Visitor prediction





Closing

What Have You Seen?

- Flink is a general-purpose analytics system
- Highly expressive Table API
- Advanced analysis with Flink’s machine learning library
- Jobs are executed on powerful distributed dataflow engine

Flink Roadmap for 2015

- Additions to Machine Learning library
- Streaming Machine Learning
- Support for interactive programs
- Optimization for Table API queries
- SQL on top of Table API

2015

Flink Forward

BERLIN 12/13 OCT

flink.apache.org
@ApacheFlink

Backup Slides

WordCount in DataSet API

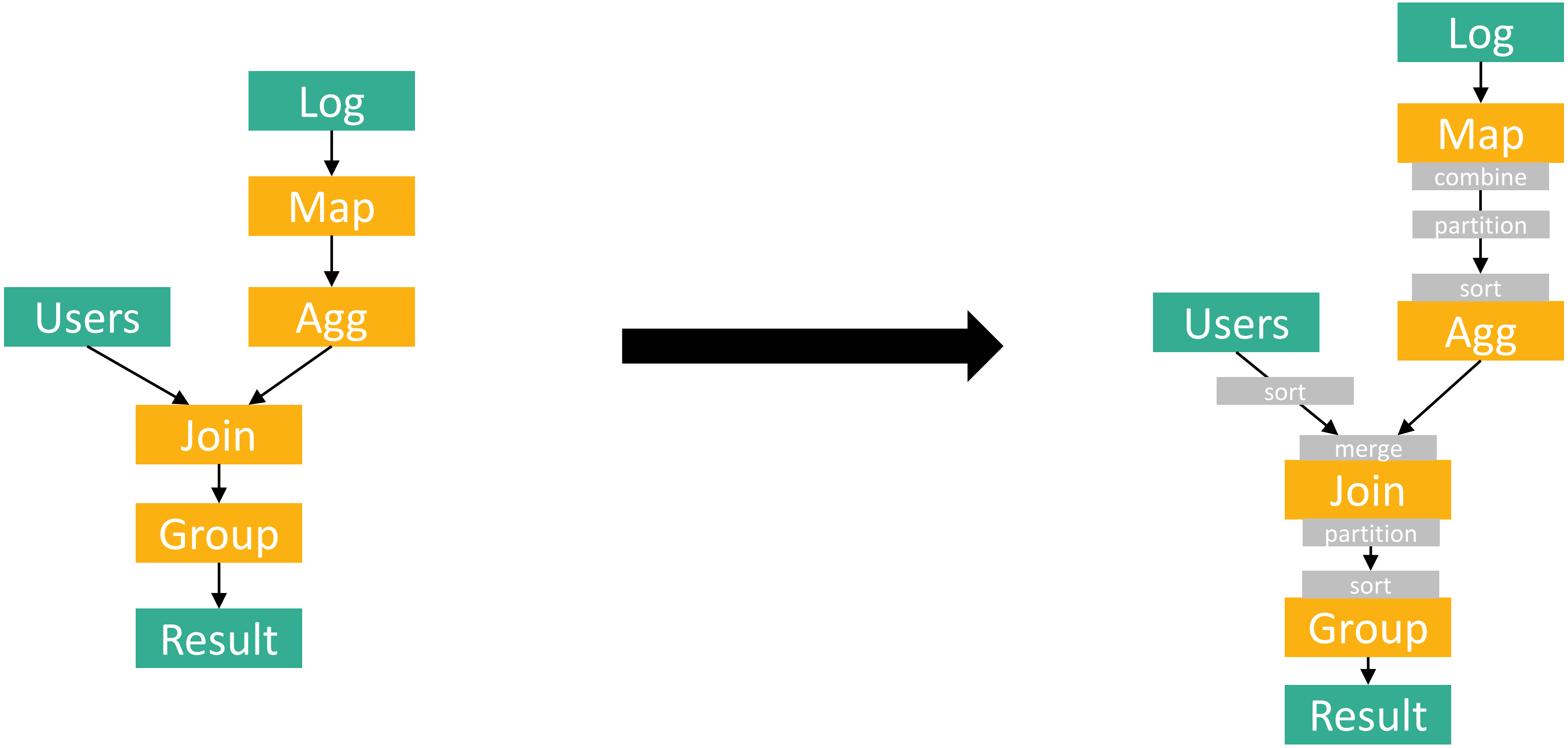
```
case class Word (word: String, frequency: Int)

val env = ExecutionEnvironment.getExecutionEnvironment()

val lines = env.readTextFile(...)

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

env.execute()
```

Java and Scala APIs offer the same functionality.

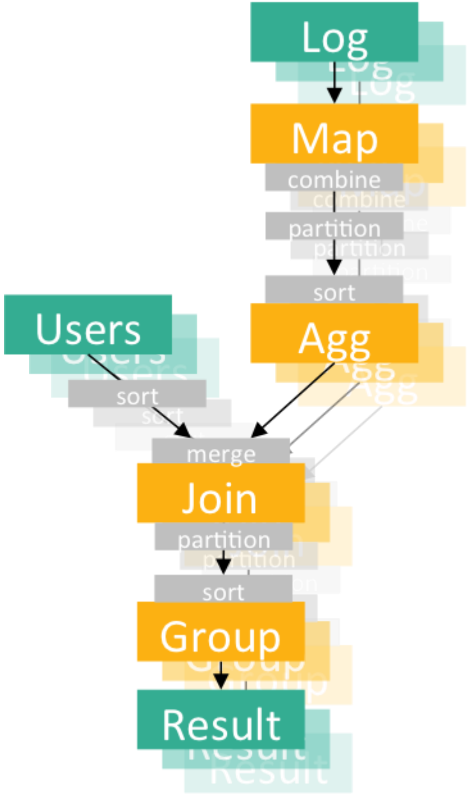
Log Analysis Code

```
ExecutionEnvironment env = TableEnvironment tableEnv = new TableEnvironment();
TableEnvironment tableEnv = new TableEnvironment();

DataSet<String> log = env.readTextFile("hdfs:///log");

DataSet<Tuple2<String, Integer>> clicks = log.flatMap(
  new FlatMapFunction<String, Tuple2<String, Integer>>() {

    public void flatMap(String in, Collector<Tuple2<>> out) {
      String[] parts = in.split("*magic regex*");
      if (parts[0].equals("click")) {
        out.collect(new Tuple2<>(parts[1], Integer.parseInt(parts[4])));
      }
    }
  });

Table clicksTable = tableEnv.toTable(clicks, "url, userId");

Table urlClickCounts = clicksTable
  .groupBy("url, userId")
  .select("url, userId, url.count as count");

Table userInfo = tableEnv.toTable(…, "name, id, …");

Table resultTable = urlClickCounts.join(userInfo)
   .where("userId = id && count > 10")
   .select("url, count, name, …");

DataSet<Result> result = tableEnv.toSet(resultTable, Result.class);

result.writeAsText("hdfs:///result");

env.execute();
```

Log Analysis Dataflow Graph



Pipelined Execution



Only 1 Stage
(depending on join strategy)

Note: Intermediate DataSets are not necessarily "created"!

Data transfer in-memory
and disk if needed

API in a Nutshell

- Element-wise
  - **map, flatMap, filter**
- Group-wise
  - **groupBy, reduce, reduceGroup, combineGroup, mapPartition, aggregate, distinct**
- Binary
  - **join, coGroup, union, cross**
- Iterations
  - **iterate, iterateDelta**
- Physical re-organization
  - **rebalance, partitionByHash, sortPartition**
- Streaming
  - **window, windowMap, coMap, ...**