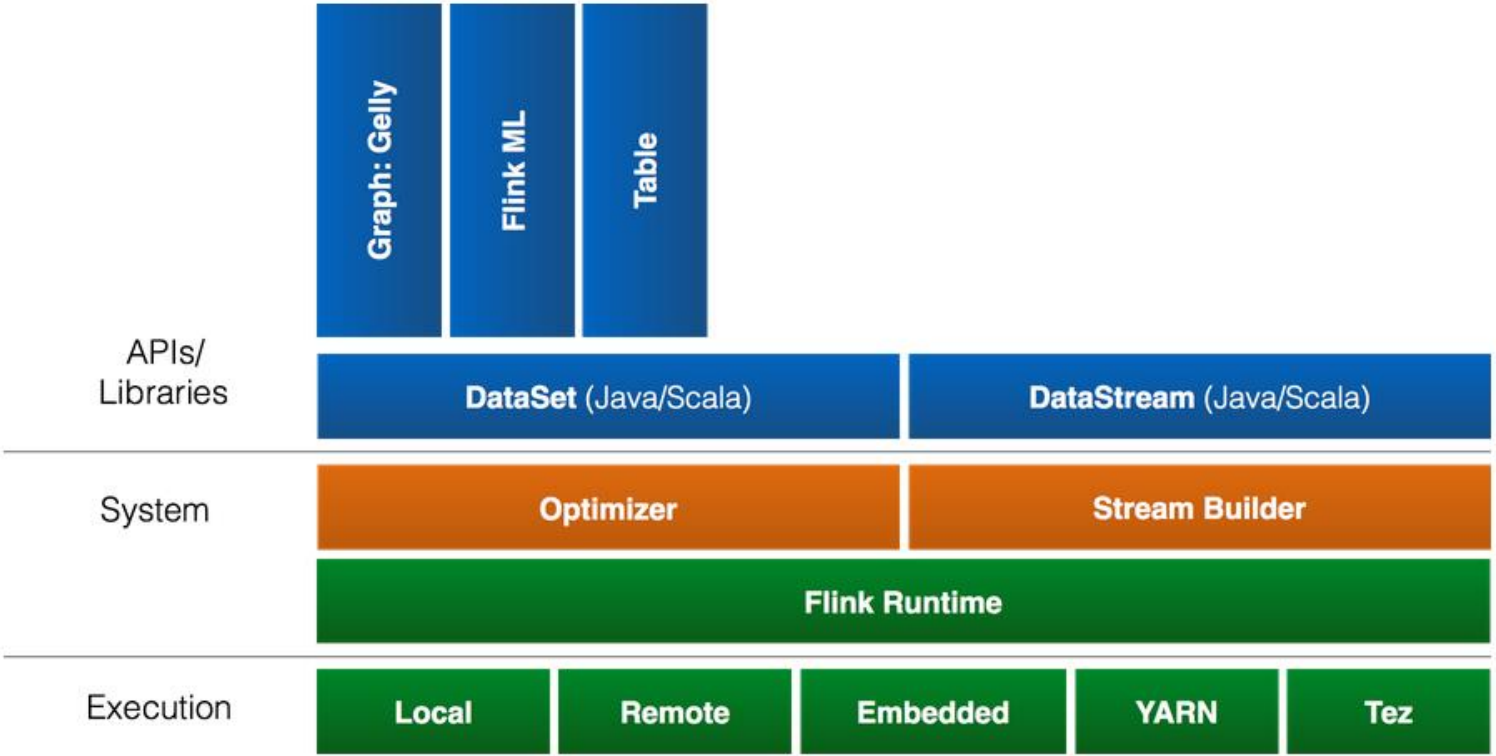
DataSet Transformations with Apache Flink

Timo Walther

Flink Committer, PMC Member
twalthr@apache.org

Overview / Flink Stack



All starts with an Environment

```
public class ExampleProgram {
    public static void main(String[] args) throws Exception {
        final ExecutionEnvironment env = ExecutionEnvironment.getExecutionEnvironment();

        DataSet<String> text = env.fromElements("Hello", "World");

        DataSet<String> moreText = env.readFromCsv("/path/to/file");

        DataSet<String> evenMoreText = env.readHadoopFile("/path/to/file");

        // add operators to your plan

        // List<Tuple2<String, Integer>> list = text.collect();
        // text.print();
        env.execute();
    }
}
```

Important Operators: .map()

```
// Takes one element and produces one element.
DataSet<Integer> tokenized = text.map(new MapFunction<String, Integer>() {
            @Override
            public Integer map(String value) {
                return Integer.parseInt(value);
            }
});

DataSet<ExperimentResult> converted = text.map(
             new MapFunction<String, ExperimentResult>() { ... });

static class ExperimentResult {
    public String name;
    public Tuple2<String, Long>[] parameters;
    public int result;
}
```

.flatMap(), .filter(),

```
// Takes one element and produces zero, one, or more elements.
data.flatMap(new FlatMapFunction<String, String>() {
            public void flatMap(String value, Collector<String> out) {
                for (String s : value.split(" ")) {
                  out.collect(s);
                }
            }
});

// Retains those elements for which the function returns true.
data.filter(new FilterFunction<Integer>() {
          public boolean filter(Integer value) { return value > 1000; }
});
```

.join()

```
DataSet<ExperimentResult> input1= ...
DataSet<ExperimentResult> input2= ...

// Joins two data sets by creating all pairs of elements that are equal on
// their keys.
DataSet<Tuple2<ExperimentResult, ExperimentResult>> =
            input1.join(input2)
                .where("name")        // key of the first input
                .equalTo("name");     // key of the second input

DataSet<ExperimentResult> =
            input1.join(input2)
                .where("name")        // key of the first input
                .equalTo("name")     // key of the second input
                .with(...);
```

.reduce(), .reduceGroup()

```
// Combines a group of elements into a single element.
number.reduce(new ReduceFunction<Integer> {
              public Integer reduce(Integer a, Integer b) { return a + b; }
});

// Combines a group of elements into one or more elements.
data
    .groupBy("name")
    .reduceGroup(new GroupReduceFunction<Integer, Integer> {
      public void reduce(Iterable<Integer> values, Collector<Integer> out) {
        int sum = 0;
        for (Integer i : values) sum++;
        out.collect(sum);
      }
});
```

.min(), .max(), .sum(), .first()

```
DataSet<Tuple2<String, Integer>> data = ...

// Perform aggregation on fields.
data = data.sum(1);

data = data.min(1);

data = data.max(1);

// Returns the first n elements of a data set.
DataSet<Tuple2<String,Integer>> result = in.groupBy(0)
    .sortGroup(1, Order.ASCENDING)
    .first(3);
```

flink.apache.org
@ApacheFlink

Or follow me on Twitter: @twalthr