

Visually Programming Dataflows for Distributed Data Analytics

Lauritz Thamsen, Thomas Renner, Marvin Byfeld,
Markus Paeschke, Daniel Schröder, Felix Böhm

Technische Universität Berlin

3rd ASH Workshop at IEEE Big Data

December 5, 2016

Technische
Universität
Berlin



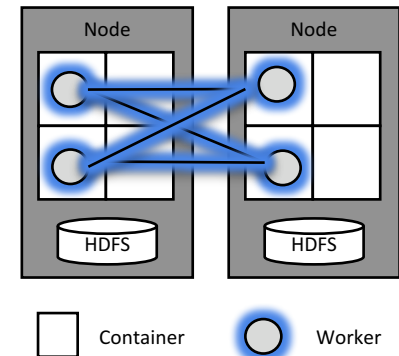
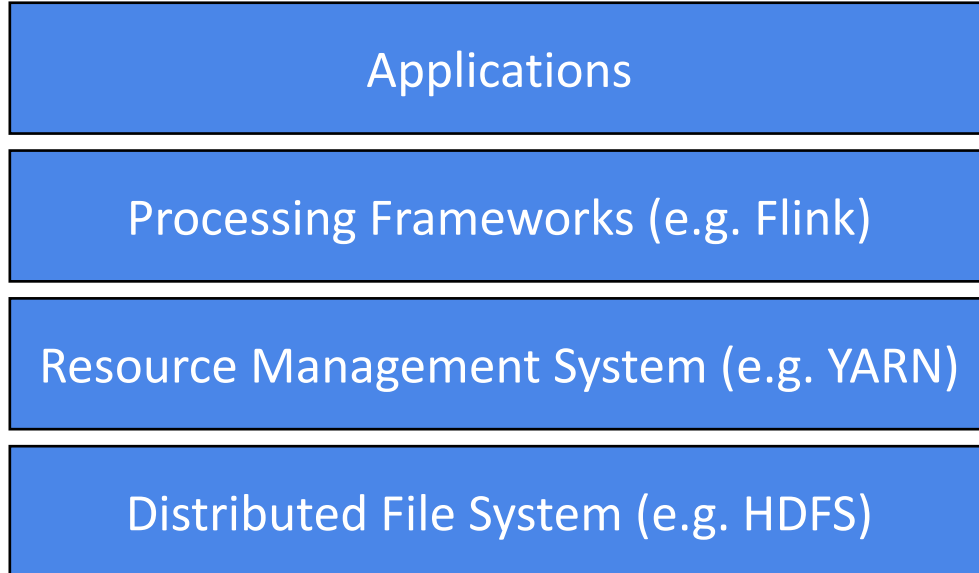
*Complex and
Distributed
IT Systems*



StratoSphere
Above the Clouds

Distributed Data Analytics

- Systems like MapReduce, Spark, and Flink allow to analyze large datasets in parallel using clusters of computers:



Apache Flink

- Open Source distributed dataflow system for general-purpose data analysis:
 - suited for batch and stream processing,
 - native support for iterative jobs,
 - efficient distributed execution,
 - high-level programming abstractions



2010: Stratosphere

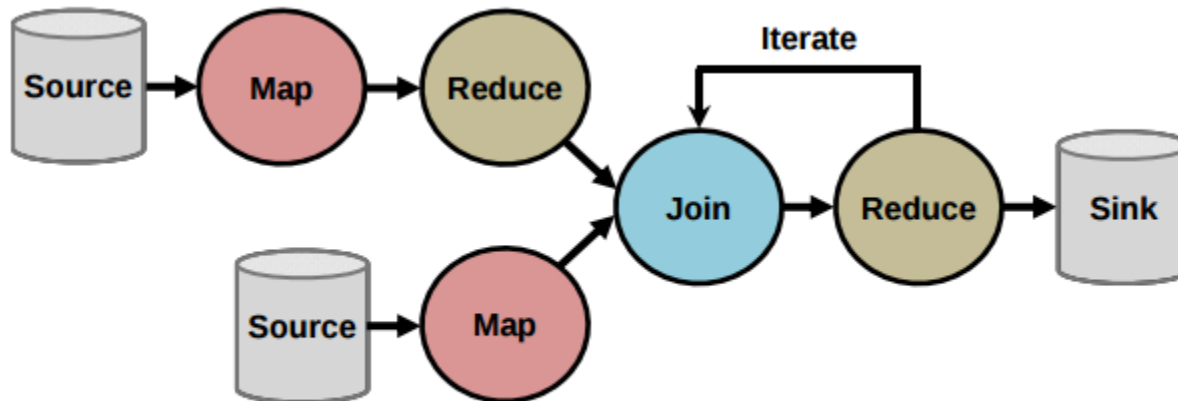


2004: MapReduce 2008: Apache Hadoop (DFG Forschergruppe)



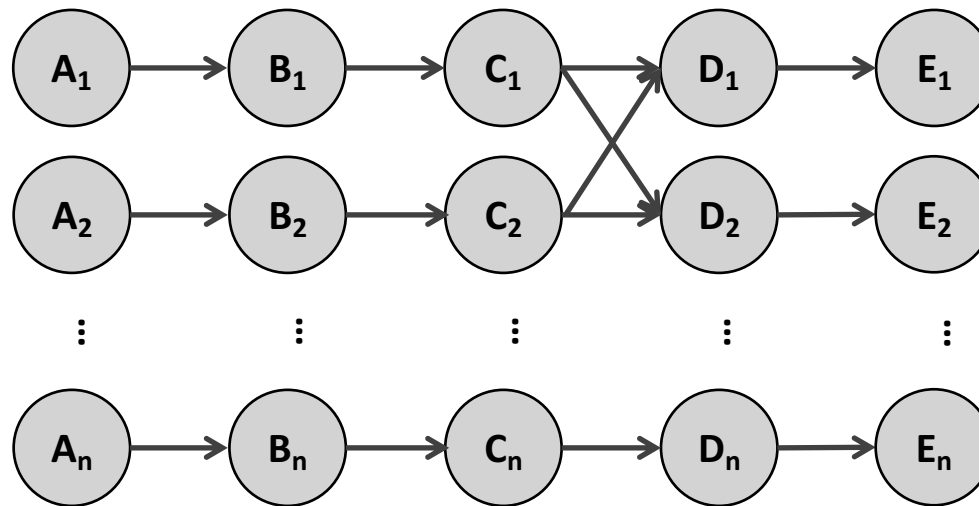
Jobs

- Jobs are directed graphs of transformations tasks
- Available transformations: Map, Reduce, Join, CoGroup, Union, Iterate, Filter, FlatMap, ...



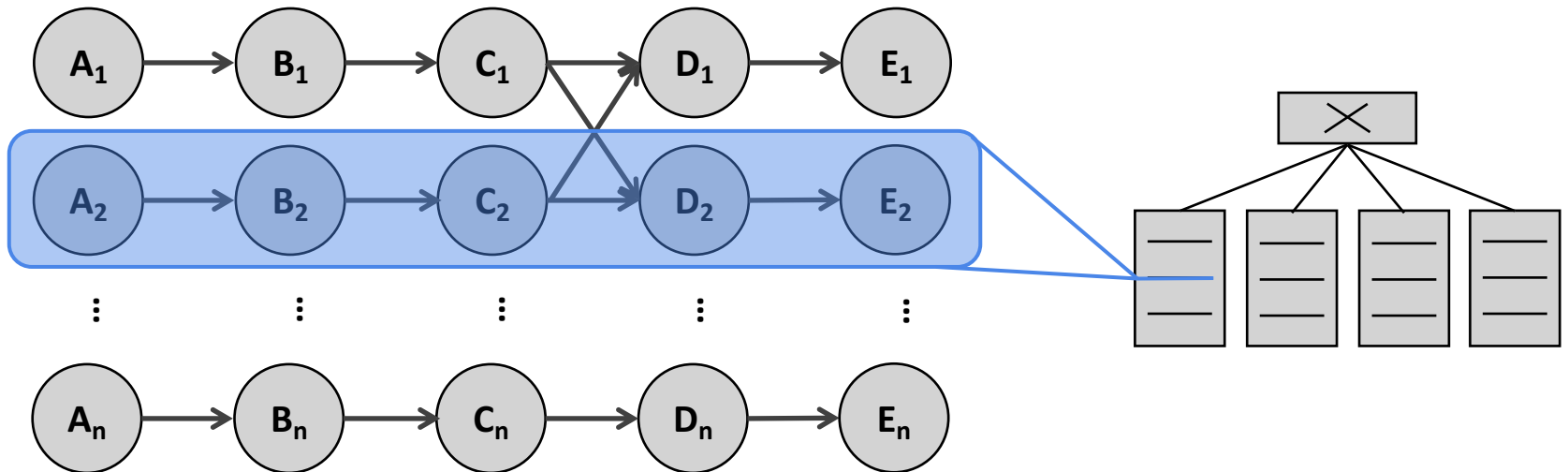
Parallelism

- Data partitioning and data-parallel task instances
- Synchronized when necessary for transformations, otherwise also pipeline parallelism



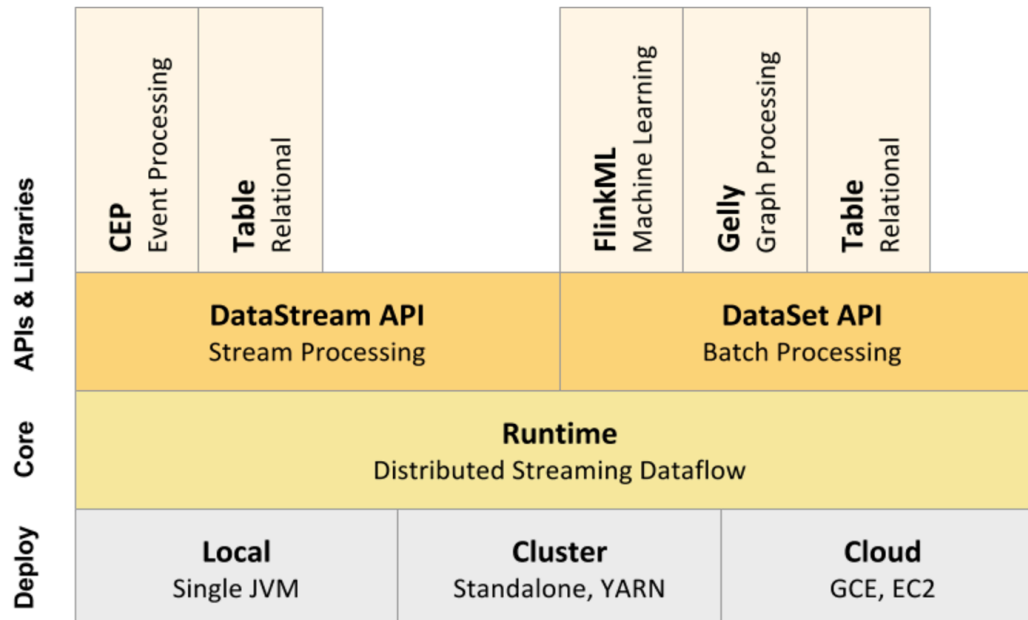
Distributed Execution

- Parallel pipelines scheduled on to shared nothing workers



Programming Abstractions

- General-purpose APIs for batch and stream
- Domain-specific abstractions for relational data, graph processing, and machine learning



<http://flink.apache.org/>

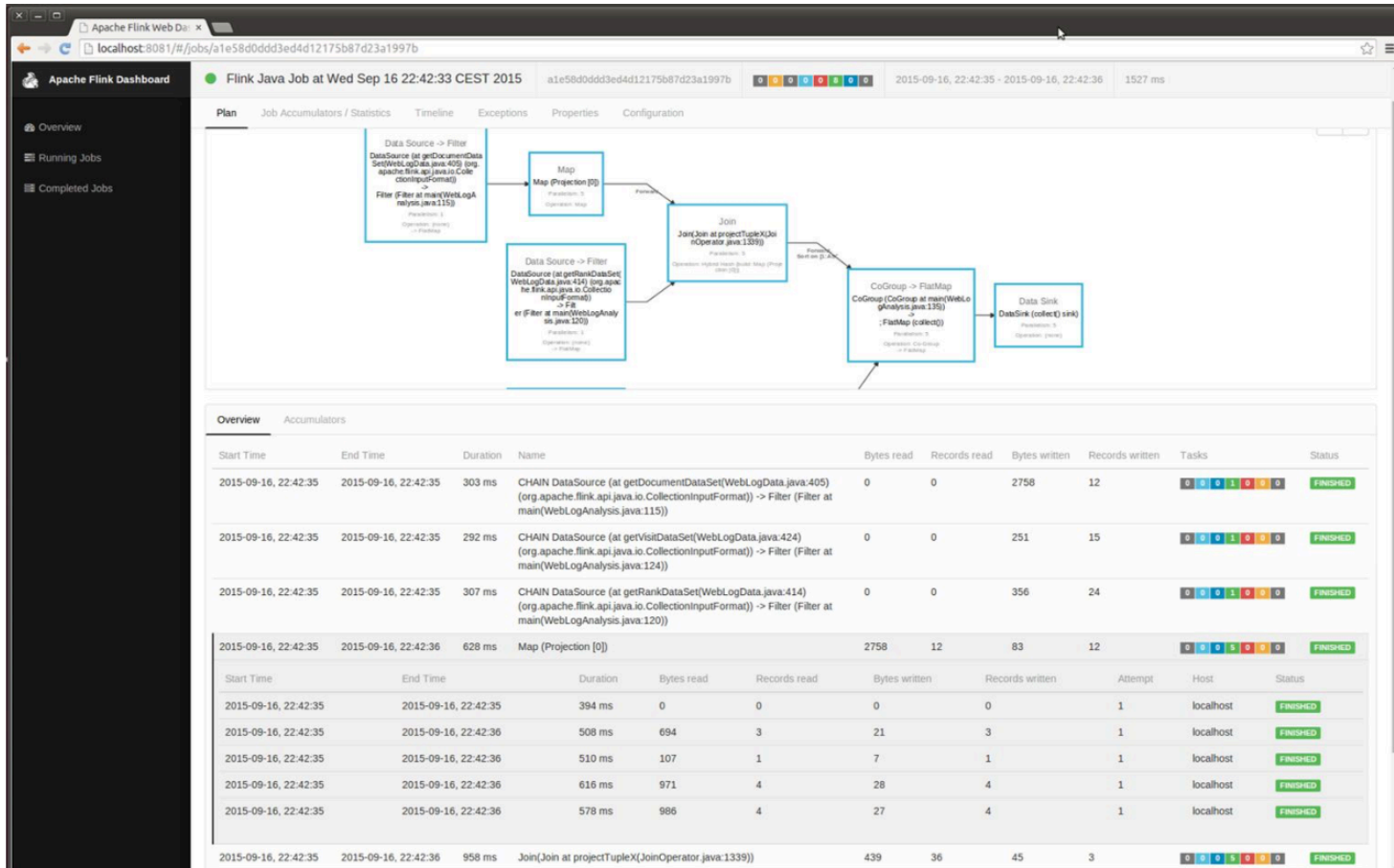
Flink WordCount Example

```
public static void main(String[] args) throws Exception {
    final ParameterTool params = ParameterTool.fromArgs(args);
    final ExecutionEnvironment env =
        ExecutionEnvironment.getExecutionEnvironment();
    env.getConfig().setGlobalJobParameters(params);

    DataSet<String> text = env.readTextFile(params.get("input"));
    DataSet<Tuple2<String, Integer>> wc =
        text.flatMap(new Tokenizer()).groupBy(0).sum(1);
    wc.writeAsCsv(params.get("output"), "\n", " ");

    env.execute("WordCount Example");
}
```


Web Dashboard

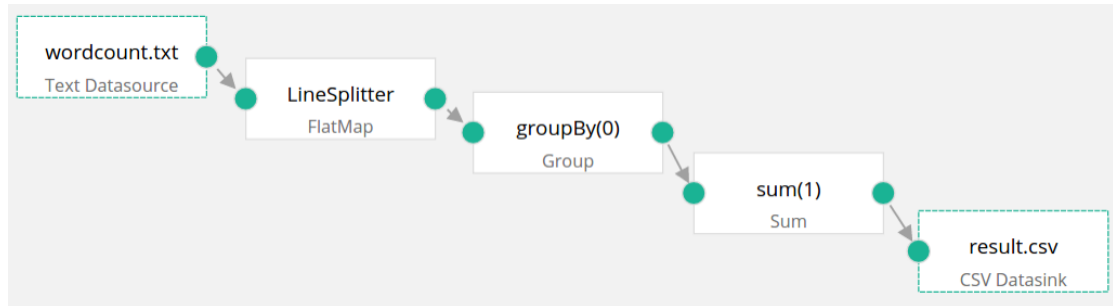


<https://cloud.githubusercontent.com/assets/1727146/9918196/beccacfa-5cc5-11e5-89f4-b26ffbd2ef9.png>

Identified Issues

- **Representation mismatch:** job graph vs. linear text
- **Invisible operator set:** code completion, no graphical view of available components
- **UDF clutter:** user code either inline or defined elsewhere
- **General-purpose coding environment:** no job submission and monitoring

Visually Programming Dataflows



```
import org.apache.flink.api.common.*;
import org.apache.flink.api.common.functions.*;
import org.apache.flink.api.java.aggregation.Aggregations;
import org.apache.flink.api.java.tuple.*;

public class Tokenizer implements FlatMapFunction<String, Tuple2> {
    @Override
    public void flatMap(String value, Collector<Tuple2<String, Integer>> out) {
        String[] tokens = value.toLowerCase().split("\\W");
        for (String token : tokens) {
            out.collect(new Tuple2<String, Integer>(token, 1));
        }
    }
}
```

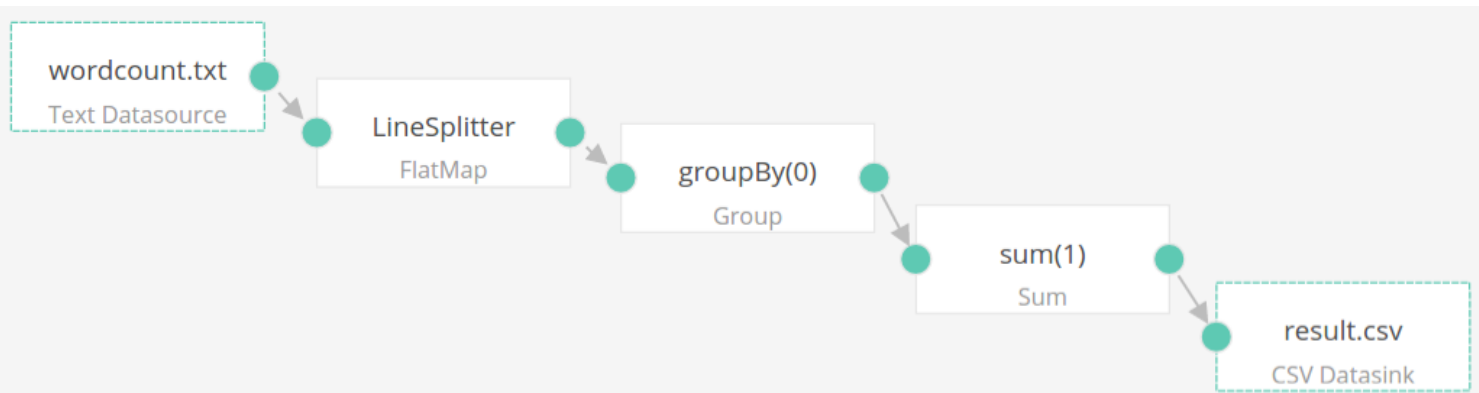
- Use visual programming for programming distributed dataflow systems:
 - Job graph view
 - Visible operator set
 - Operator configuration and UDF coding in windows
 - Integrated job submission and monitoring

Flision Prototype

- Web-based visual programming environment for Apache Flink, developed by a student project

The screenshot displays the FLINK VISUAL web-based visual programming environment. The interface includes a top toolbar with actions like Undo, Redo, Deploy & Run, Code, Jar, Reset, Export, Import, and Config. A left sidebar contains a search bar and a list of components: Datasources, Join, Filter, Sum, Group, Map, FlatMap, and Reduce. The main workspace shows a workflow diagram with the following components: 'wordcount.txt' (Text Datasource), 'LineSplitter' (FlatMap), 'groupBy(0)' (Group), 'sum(1)' (Sum), and 'result.csv' (CSV Datasink). A 'Console Output' window at the bottom right shows the following log output:

```
[INFO] Building jar: /tmp/fc0e4485-eb7d-4bfb-8094-5fd1b0c81c1c/FlinkProject/target/FlinkJob-1.0.jar
[INFO] --- maven-shade-plugin:2.4.1:shade (default) @ FlinkJob ---
[INFO] No artifact matching filter org.apache.flink:*
[INFO] Replacing original artifact with shaded artifact.
[INFO] Replacing /tmp/fc0e4485-eb7d-4bfb-8094-5fd1b0c81c1c/FlinkProject/target/FlinkJob-1.0.jar with /tmp/fc0e4485-eb7d-4bfb-8094-5fd1b0c81c1c/FlinkProject/target/FlinkJob-1.0-shaded.jar
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 3.055 s
[INFO] Finished at: 2016-03-17T22:41:09+01:00
[INFO] Final Memory: 28M/216M
[INFO] -----
```



Demo

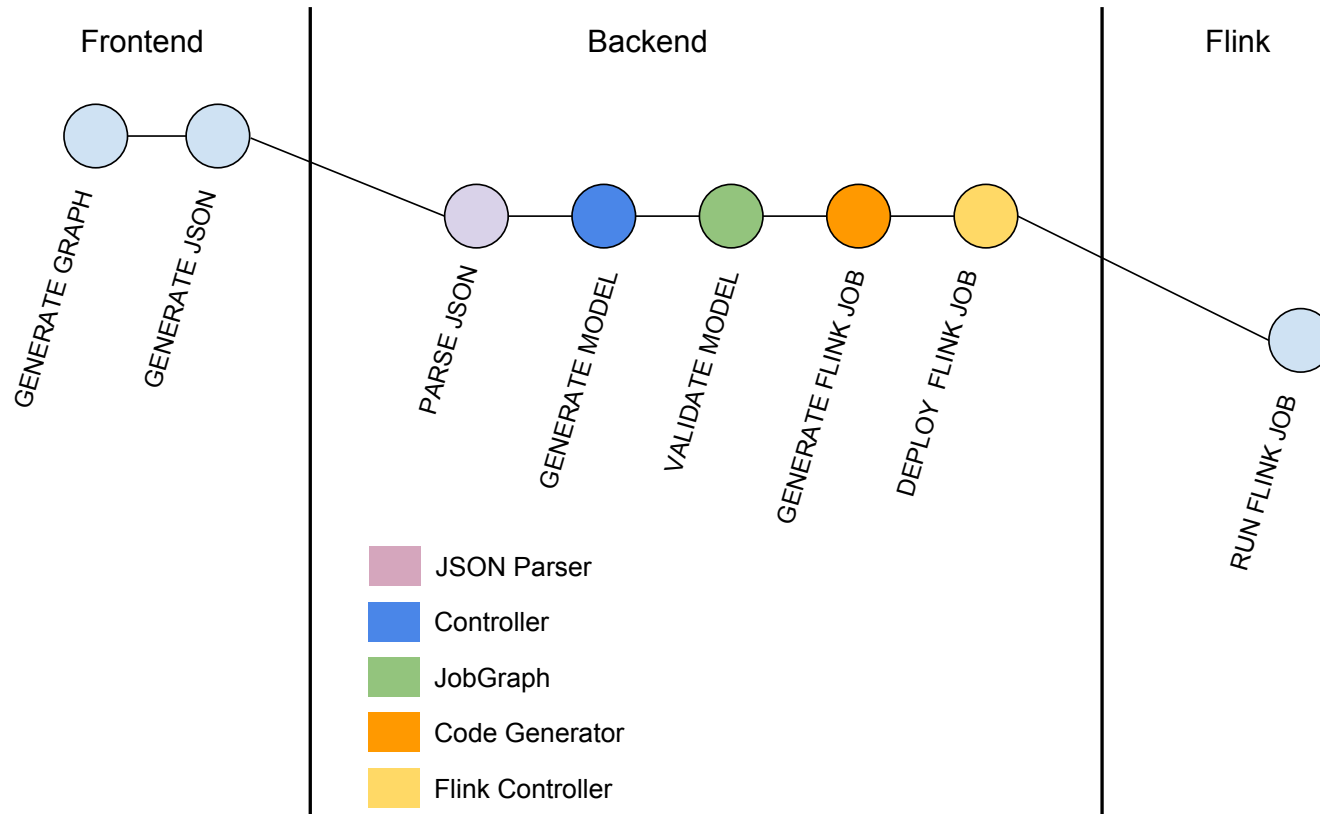
Console Output

```
[INFO] Building jar: /tmp/fc0e4485-eb7d-4bfb-8094-5fd1b0c81c1c/FlinkProject/target/FlinkJob-1.0.jar
[INFO] --- maven-shade-plugin:2.4.1:shade (default) @ FlinkJob ---
[INFO] No artifact matching filter org.apache.flink:*
[INFO] Replacing original artifact with shaded artifact.
[INFO] Replacing /tmp/fc0e4485-eb7d-4bfb-8094-5fd1b0c81c1c/FlinkProject/target/FlinkJob-1.0.jar with /tmp/fc0e4485-eb7d-4bfb-8094-5fd1b0c81c1c/FlinkProject/target/FlinkJob-1.0-shaded.jar
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 3.055 s
[INFO] Finished at: 2016-03-17T22:41:09+01:00
[INFO] Final Memory: 28M/216M
[INFO] -----
```

→ <https://vimeo.com/166030396>

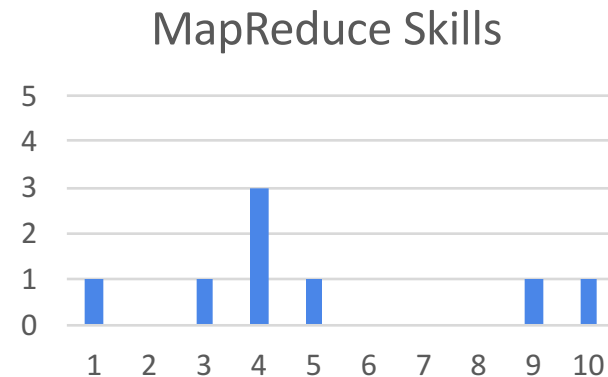
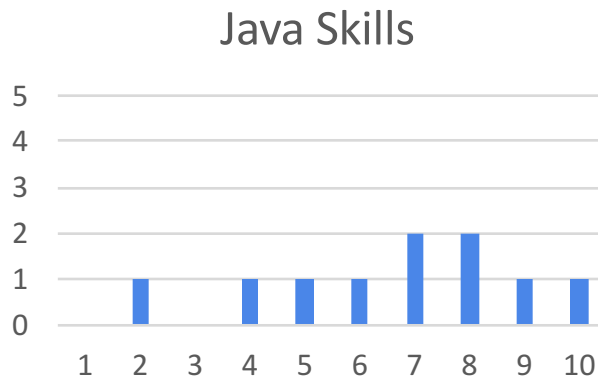
Prototype Implementation

- Implemented as a Web application



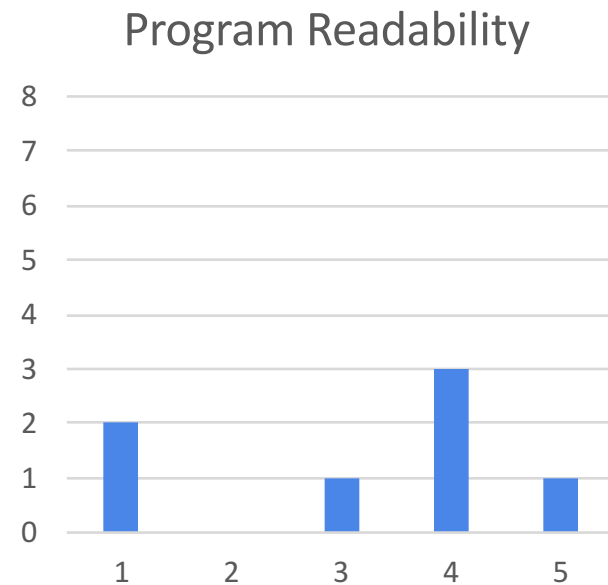
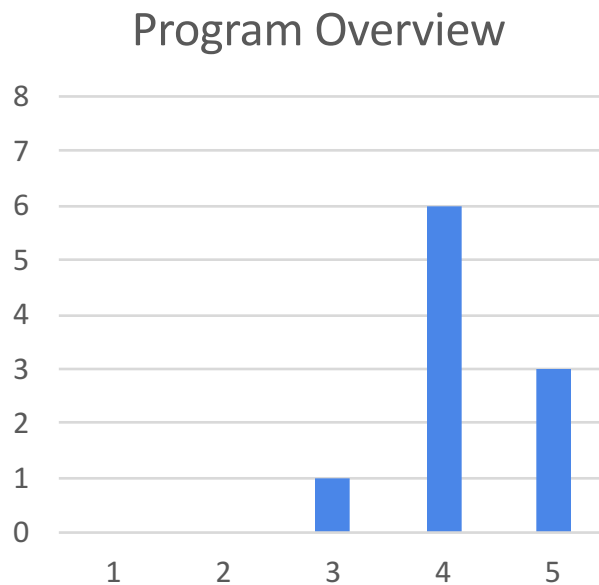
Evaluation Setup

- Qualitative user testing:
 - implementation of WordCount and a questionnaire
 - with ten participants, with between 3.5 and 20 years of programming experience (average: 7.85 years)



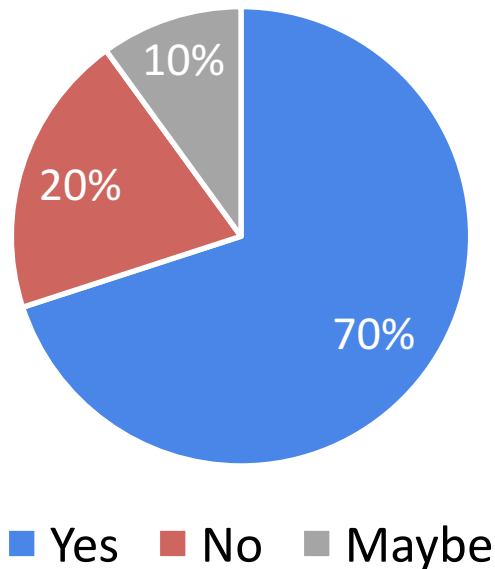
Evaluation Results (1/2)

- Good overview, good for new users, yet some developers still might prefer their textual IDEs



Evaluation Results (2/2)

- Would you be willing to use such an environment in future data analysis projects?



Conclusion

- Visual programming might be a good fit for distributed dataflow systems
- User feedback: Increased overview, easier for new users and non-programmers, but maybe not the best choice for professional developers
- Interesting future directions:
 - Abstract entire subgraphs
 - Shared component repositories
 - Job progress and debugging