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

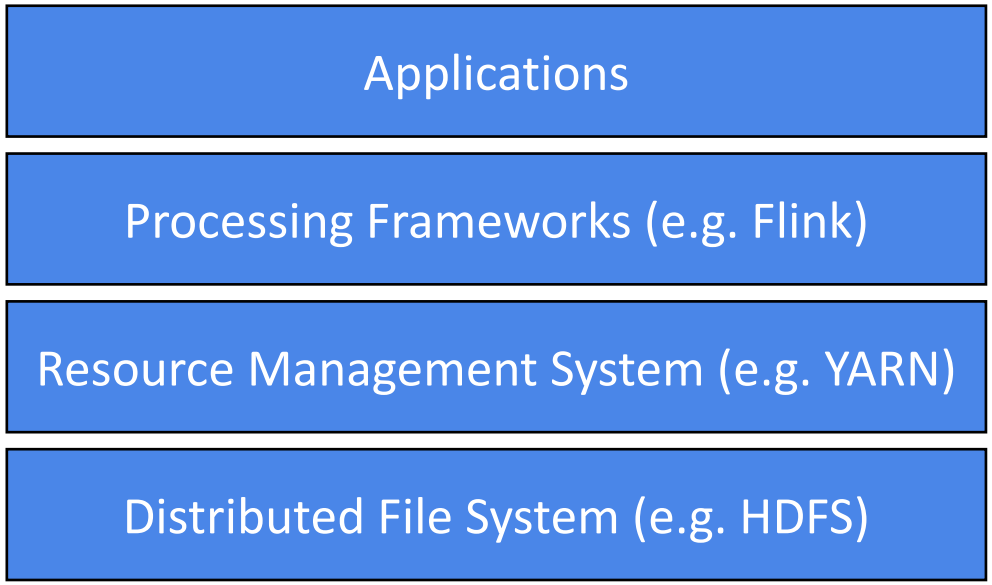
December 5, 2016

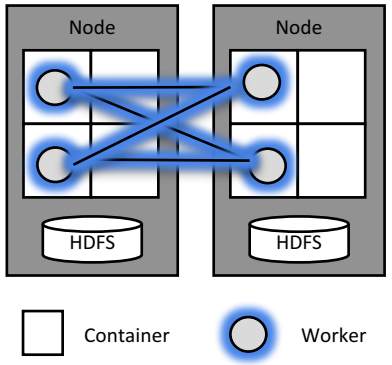
Technische Universität Berlin | Complex and Distributed IT Systems

StratoSphere Above the Clouds

Distributed Data Analytics

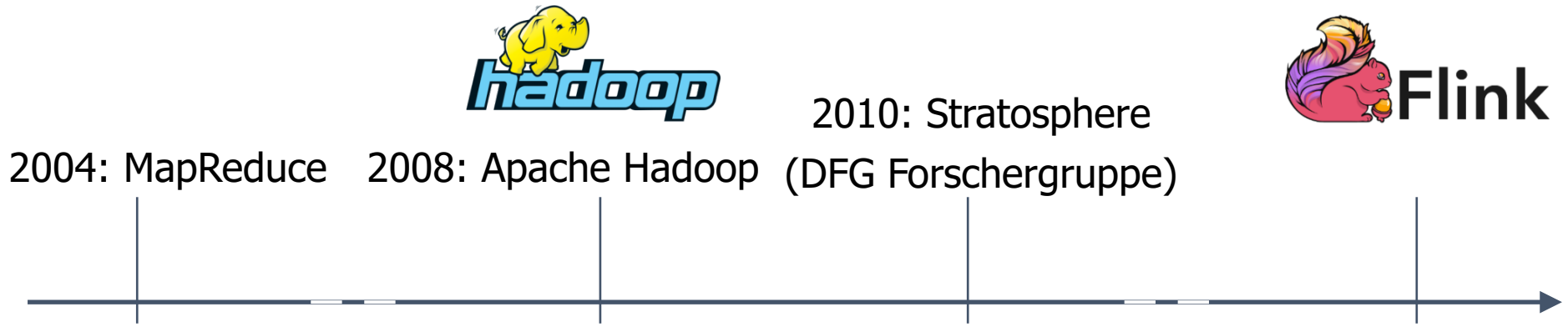
- Systems like MapReduce, Spark, and Flink allow to analyze large datasets in parallel using clusters of computers:





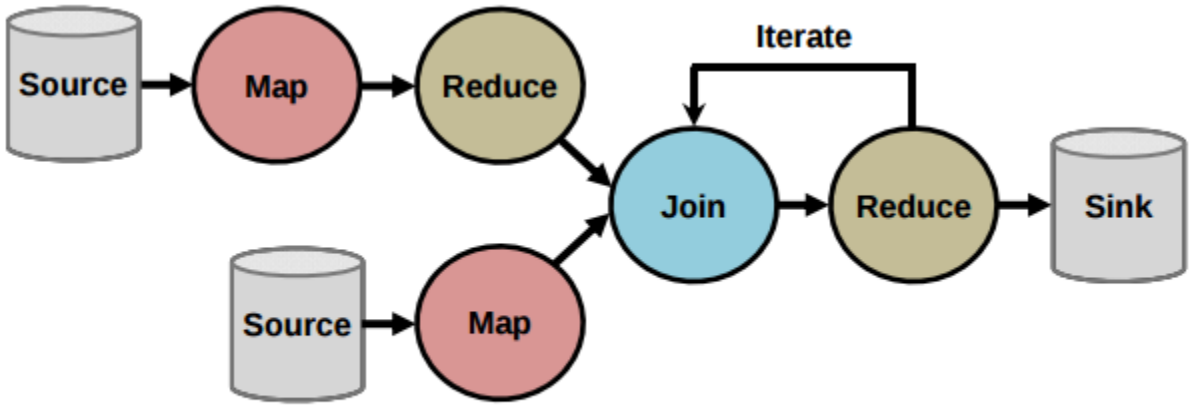
Apache Flink

- 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



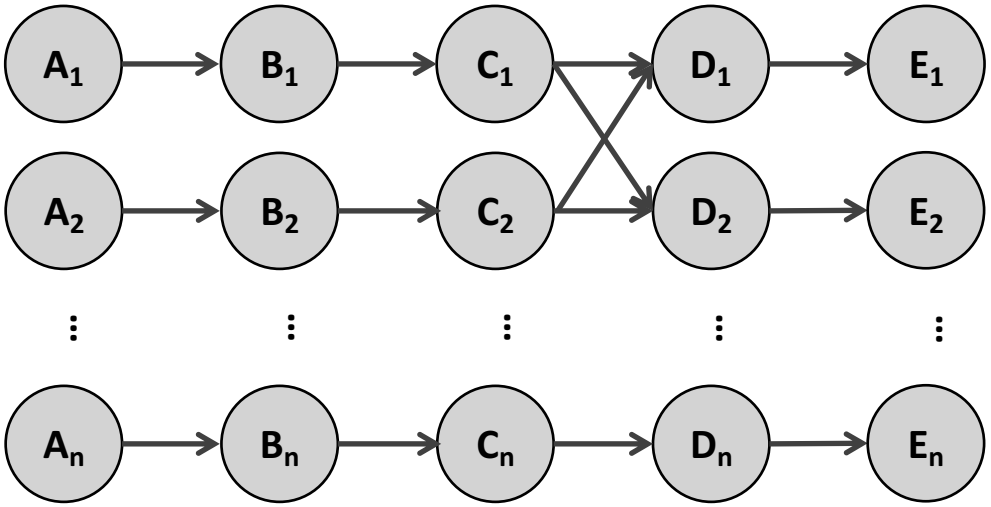
Jobs

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



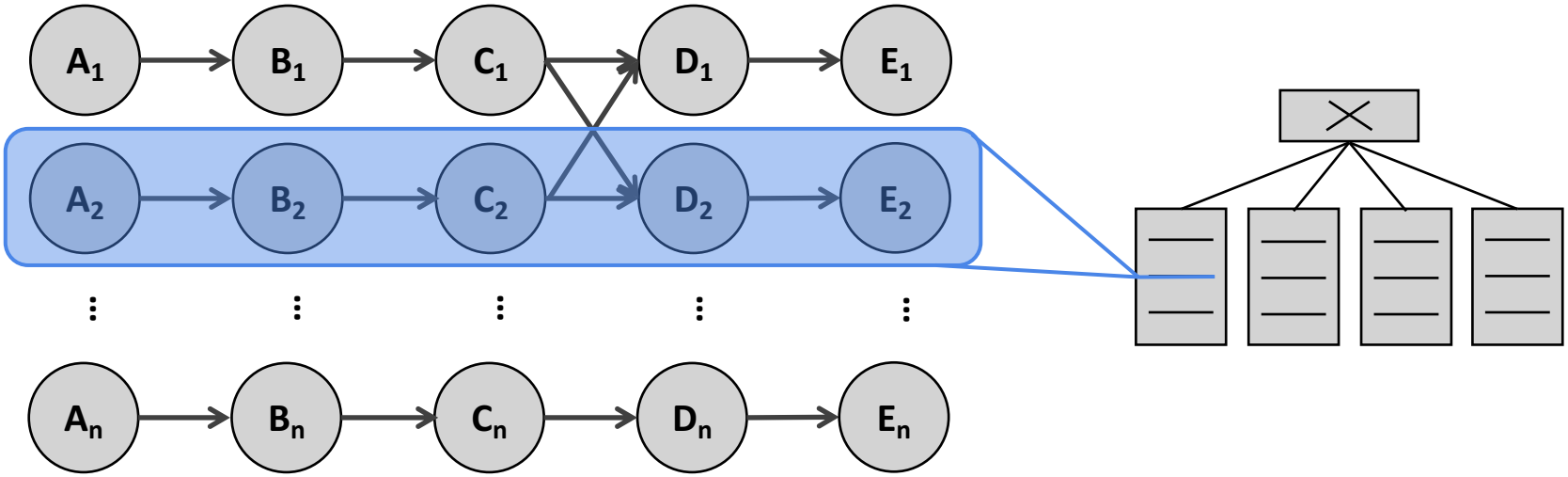
Parallelism

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



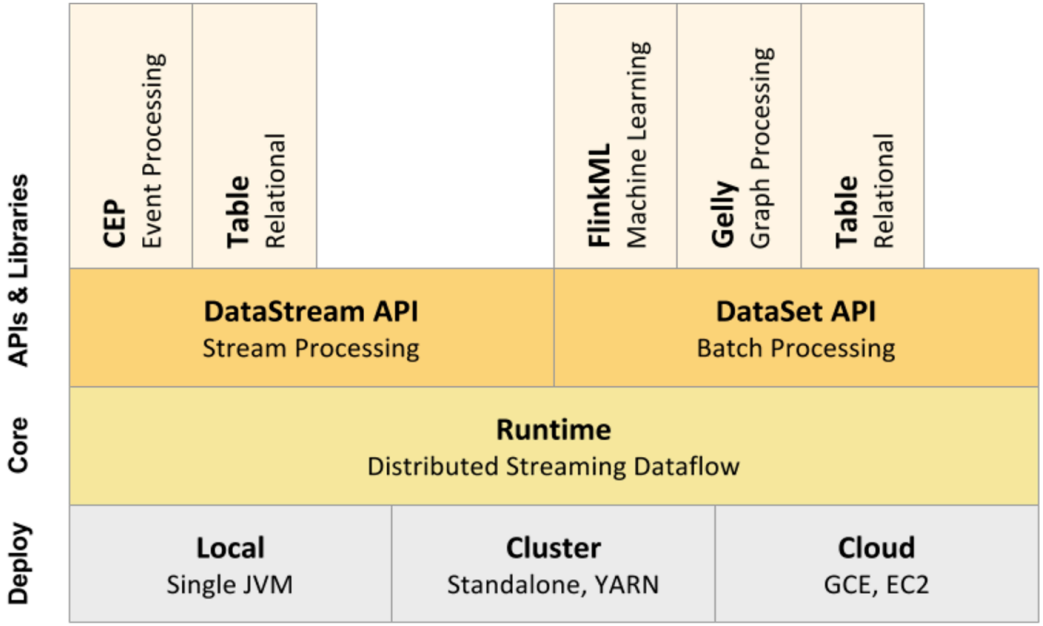
Distributed Execution

- 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/

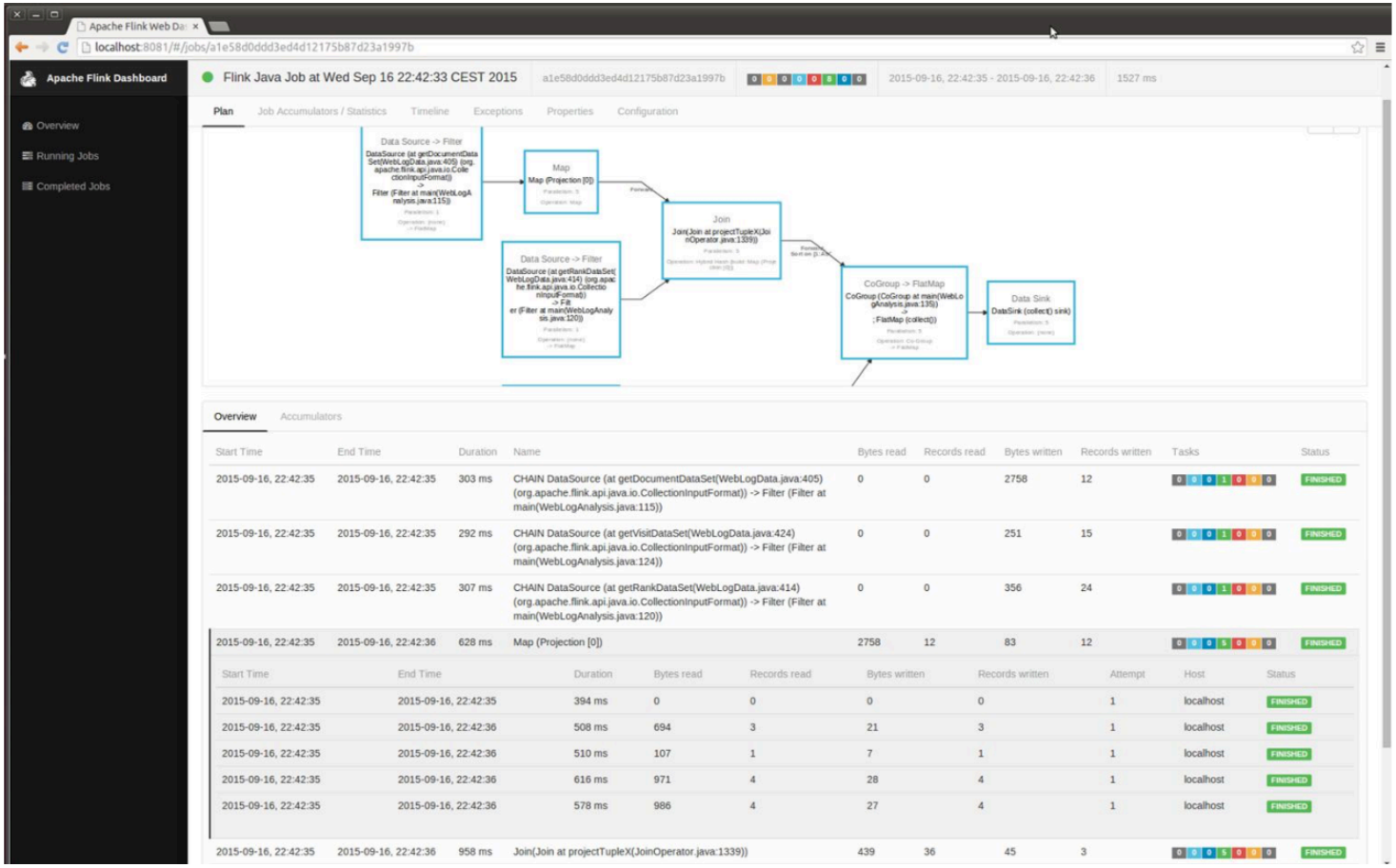
Flink WordCount Example

```
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

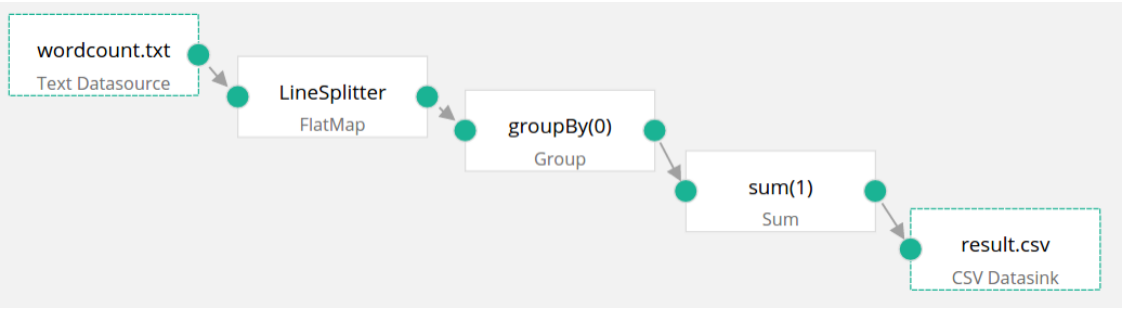


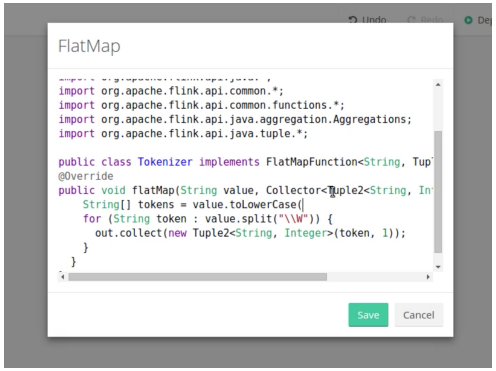
https://cloud.githubusercontent.com/assets/1727146/9918196/beccacfa-5cc5-11e5-89f4-b26ffbdf2ef9.png

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

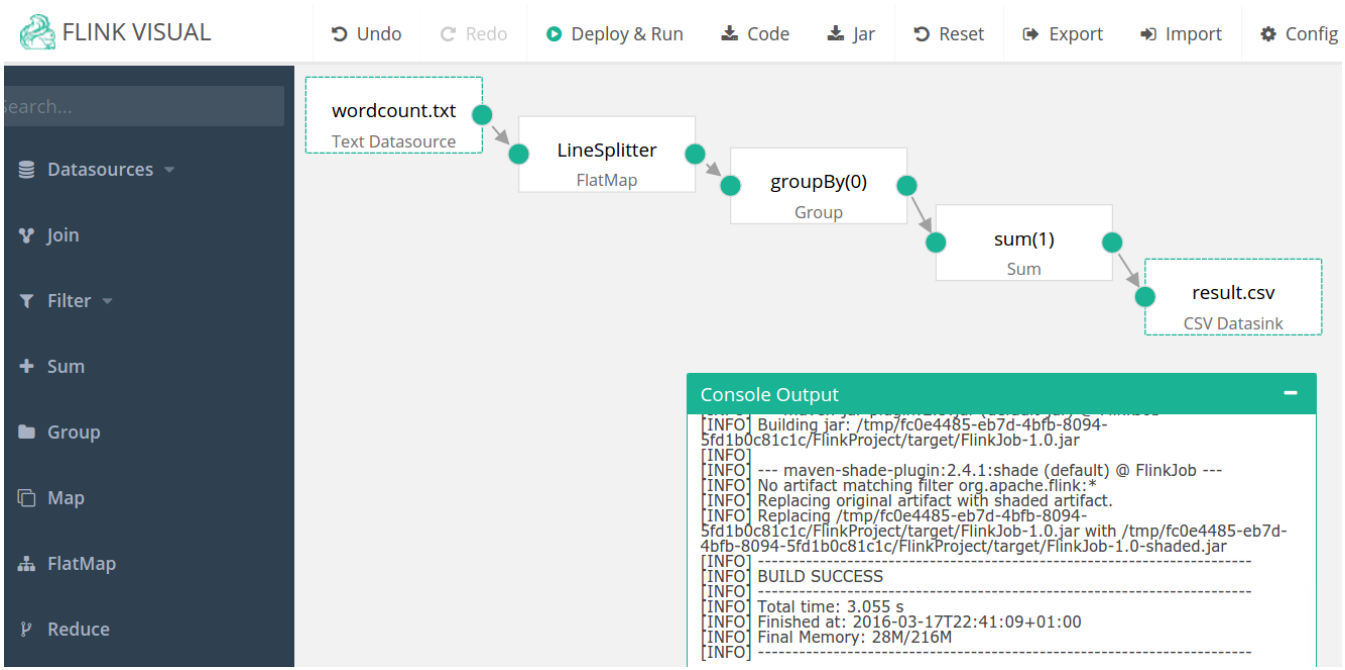


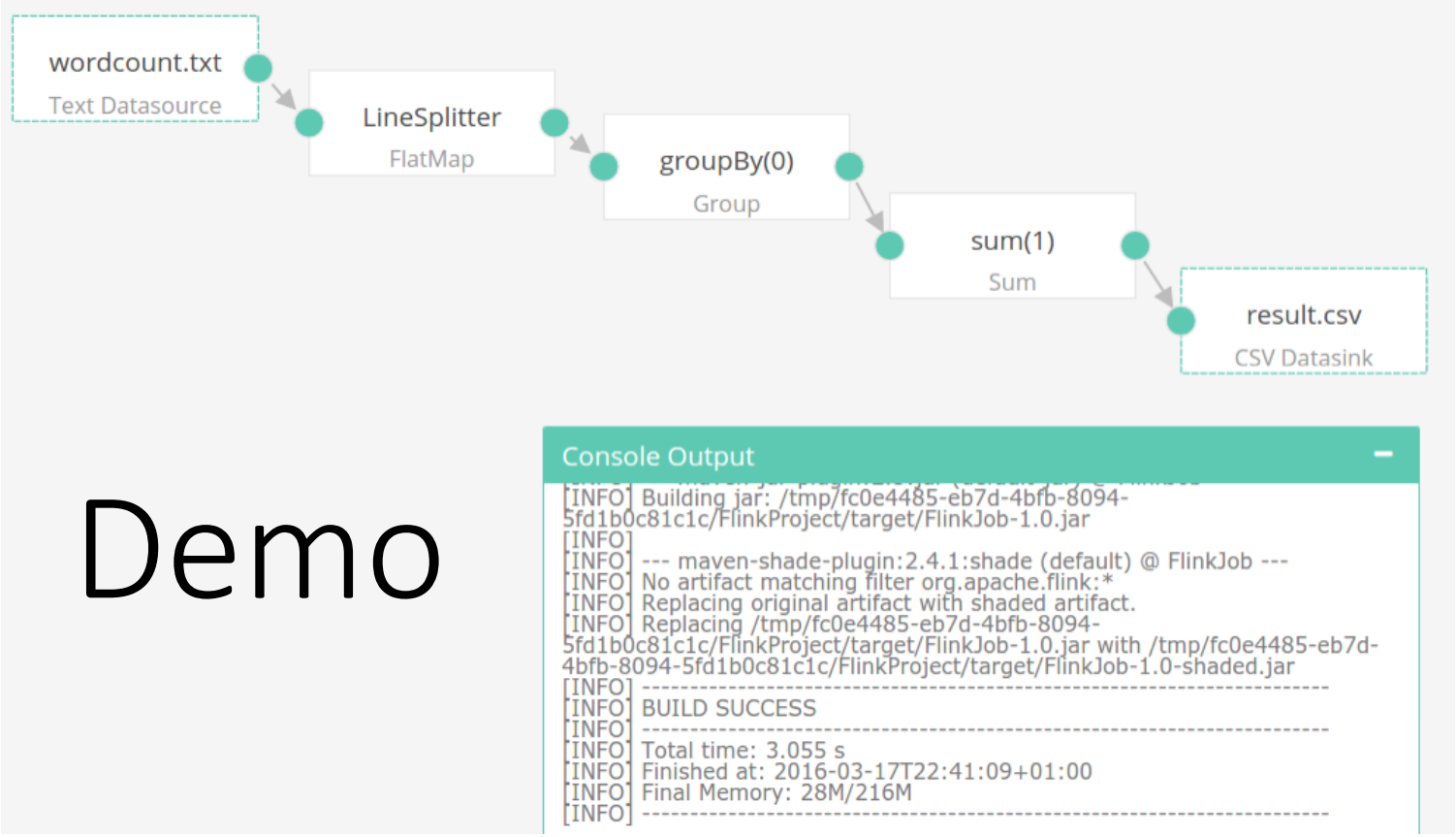


- Use visual programming for programming distributed dataflow systems:
  - Job graph view
  - Visible operator set
  - Operator configuration and UDF coding in windows
  - Integrated job submission and monitoring

Flision Prototype

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



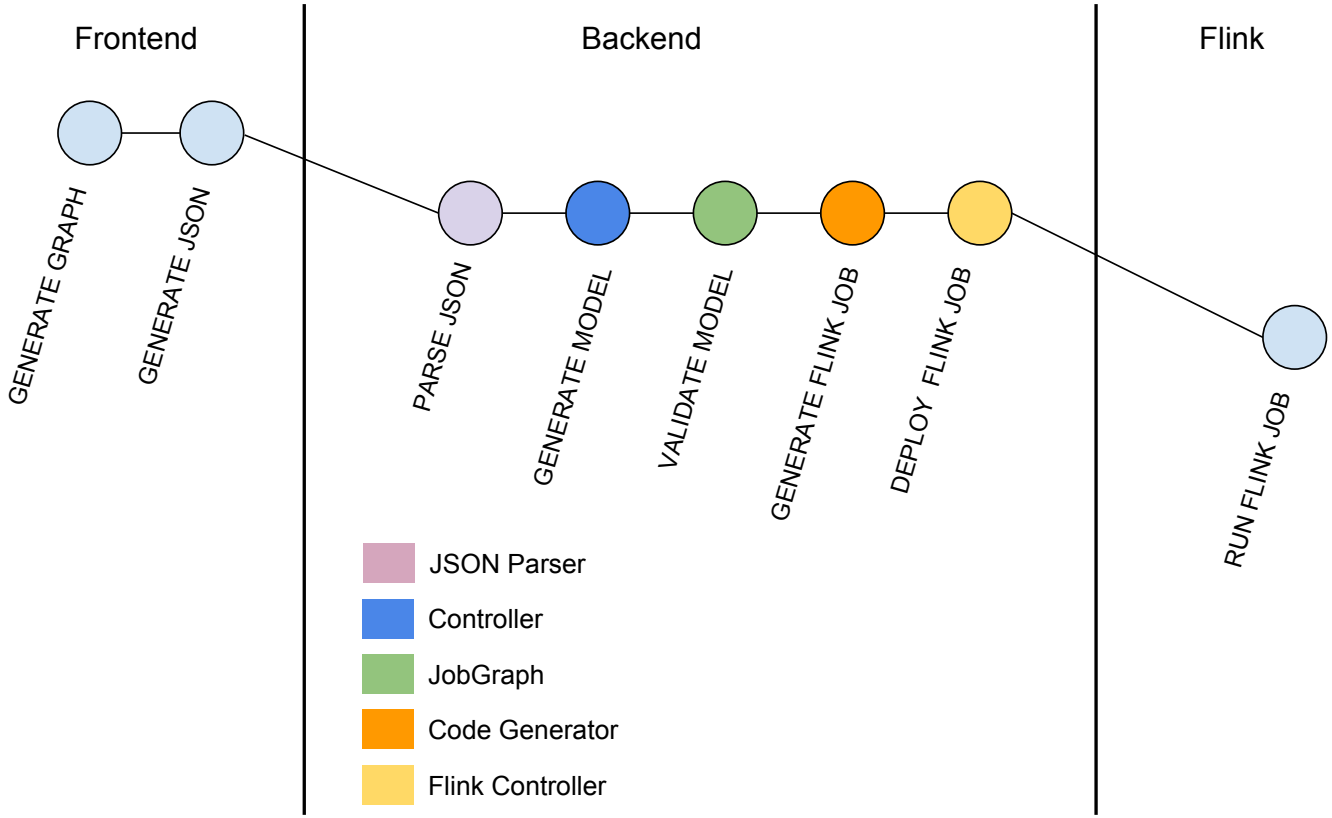


Demo

→ https://vimeo.com/166030396

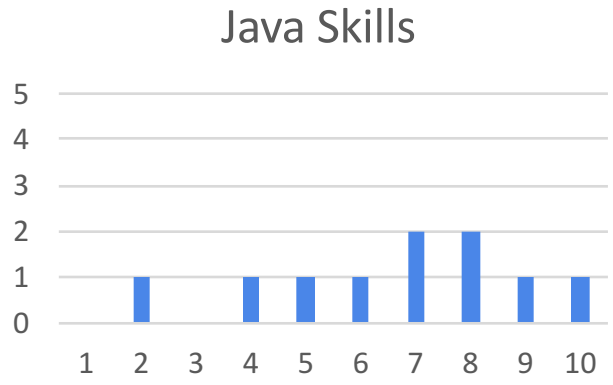
Prototype Implementation

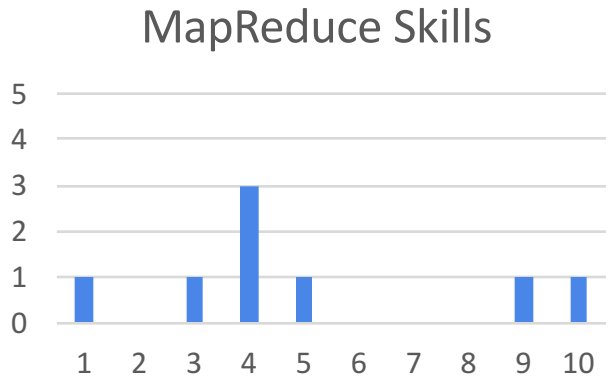
- Implemented as a Web application



Evaluation Setup

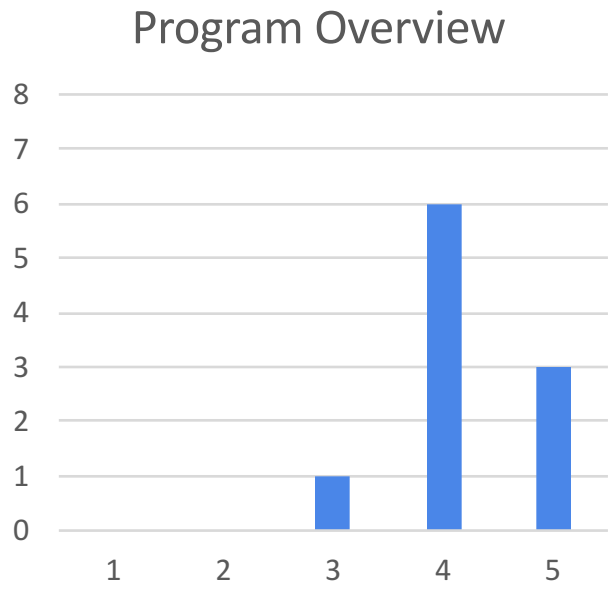
- Qualitative user testing:
  - implementation of WordCount and a questionnaire
  - with ten participants, with between 3.5 and 20 years of programming experience (average: 7.85 years)

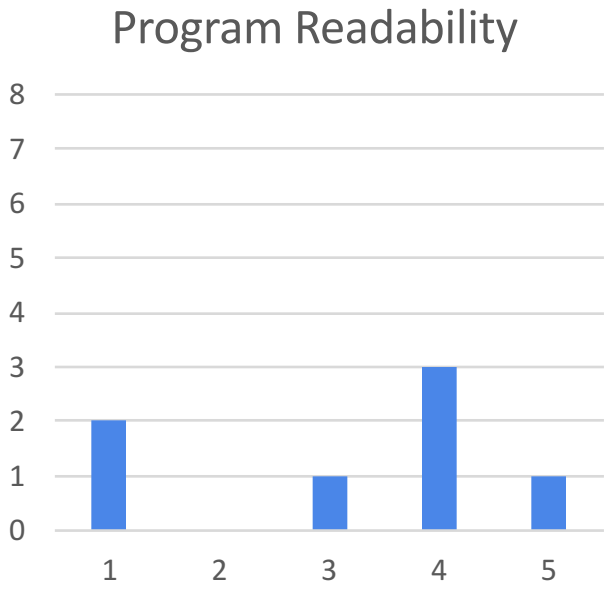




Evaluation Results (1/2)

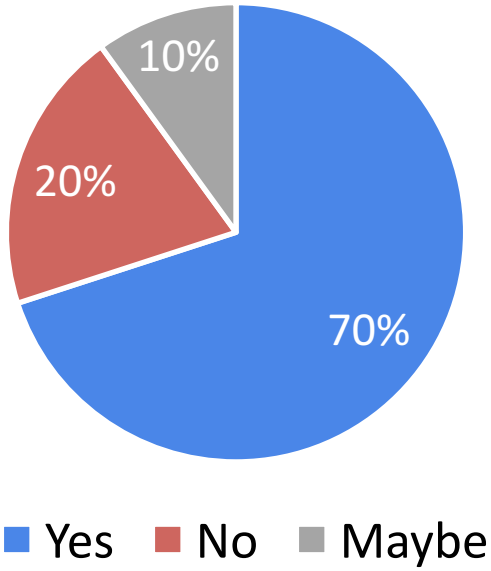
- Good overview, good for new users, yet some developers still might prefer their textual IDEs





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