
William Jones
Background - The rise of streams

- In the last 10 years, stream processing has become prevalent.
- Came out of the demand of real time data analysis.
- Instead of static data, streams are continuous real-time data sources.
Background - What is System S

- SPADE runs on System S, necessary to understand what this is first.
- System S is a distributed stream processing middleware, developed by IBM.
- Abstracts away commonly seen distributed computing issues. e.g. Placement and scheduling, distributed job management, failure-recovery, and security.
System S continued

- User inputs a data-flow graph of **processing elements** (PEs) connected by streams containing stream data objects (SDOs).
- Includes all important input stream(s) **source(s)** and output stream(s) **sink(s)**.
Problem for inexperienced programmers...

- System S offers INQ, a simple DSL where users can pose simple ‘inquiries’ to their streams.
- This automatically generates data-flow graphs consisting of existing PEs - quite inflexible.
... Customisation only available to experienced programmers

- For custom data-flow graphs and PEs, the user needs to write C++ or Java code to interact with the PE APIs.
- They need to specify PE behaviour in terms of input and output port, configuration files to specify topology of data-flow graph etc...

This is a headache.
Where SPADE fits in

Figure 1: System S from an application developer’s perspective
Introducing SPADE

- SPADE solves this problem by providing a declarative intermediate language.
- Basic building block objects are streams.
- Able to specify arbitrary data-flow graphs, and compose streams with them operators and stream adaptors.
SPADE - further details

- Code is compiled automatically and generates code running natively on System S.
- Performance is optimised automatically.
Example operators

- **Functor**
  - filtering, projection, mapping etc...
- **Aggregate**
  - summarization of incoming tuples
- **Join**
  - correlating two streams.
- Also possible to create udops (user defined operations)
Operators are partitioned amongst PEs

- In a way to minimise inter-PE communication, but also ensure PEs are within capacity.
- Hard to do deterministically, especially for user-defined operations.
- They describe a statistical learning approach. They compile SPADE code twice, first to collect statistics, then to optimise operator partitions for performance based on these statistics.
Example SPADE application

Calculating the bargaining index using real time financial data.

Figure 5: Bargain Index computation for all stock symbols
Criticisms

- Details of statistical learning not clear. IBM restricting information?
- Doesn’t explicitly compare performance with native code written for System S. Doesn’t even attempt to quantify these optimisations.
- Doesn’t give a clear evaluation with other stream processing platform (if any exist).
- No example code written in SPADE.