



Improved Event Processing Performance through Parallel Event Transformation

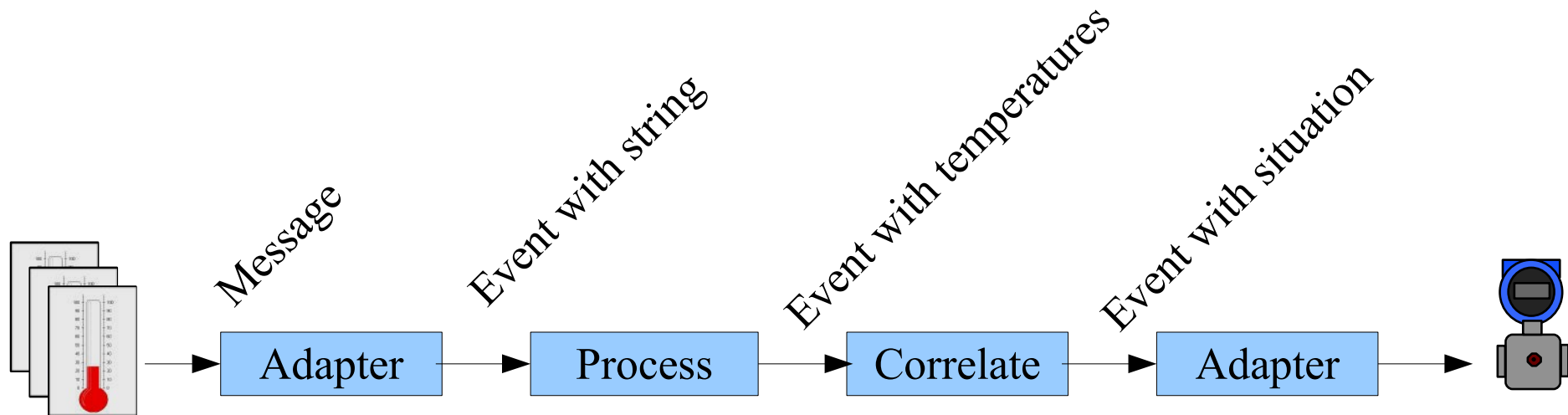
Andrey Brito, Christof Fetzer

andrey.brito@inf.tu-dresden.de, christof.fetzer@inf.tu-dresden.de

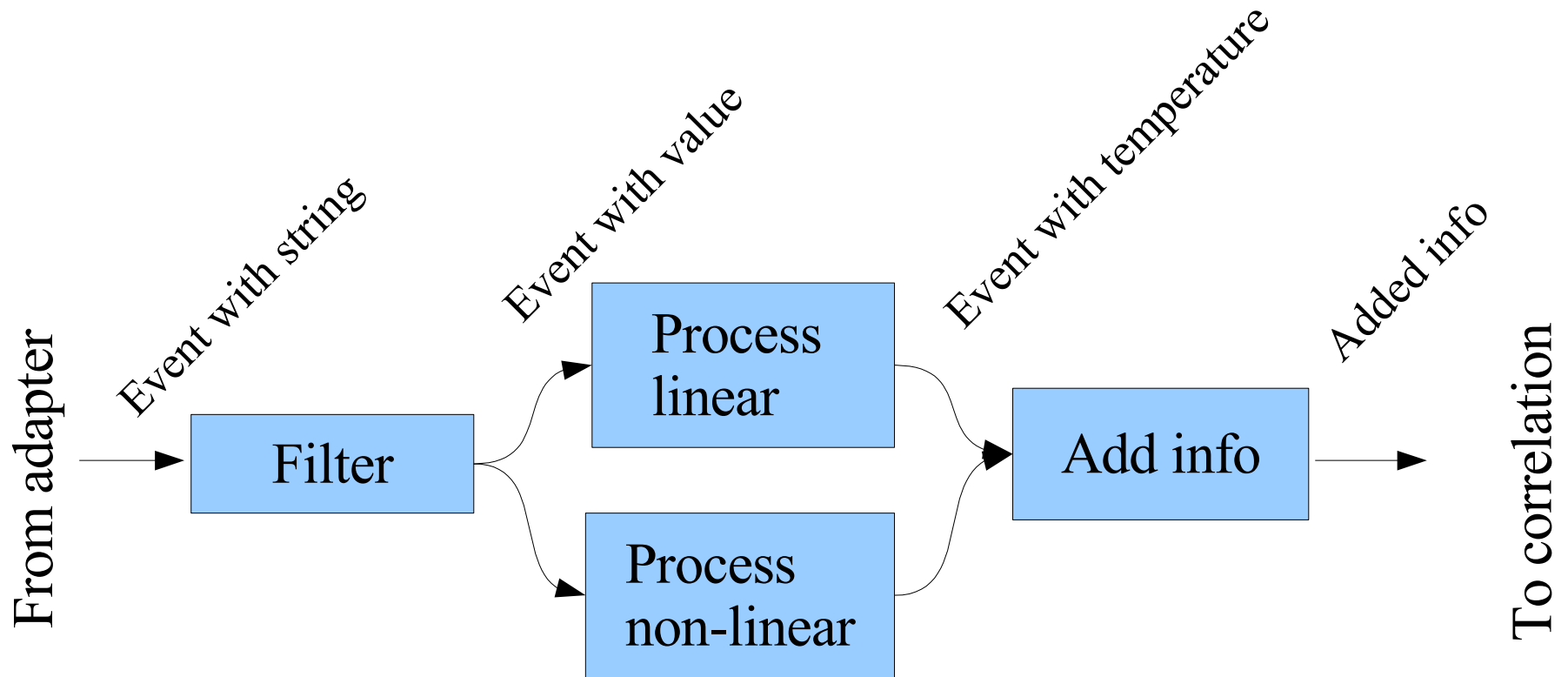
EDA-PS'07, 24.9.2007

Introduction

- A general architecture for event processing



Process Phase



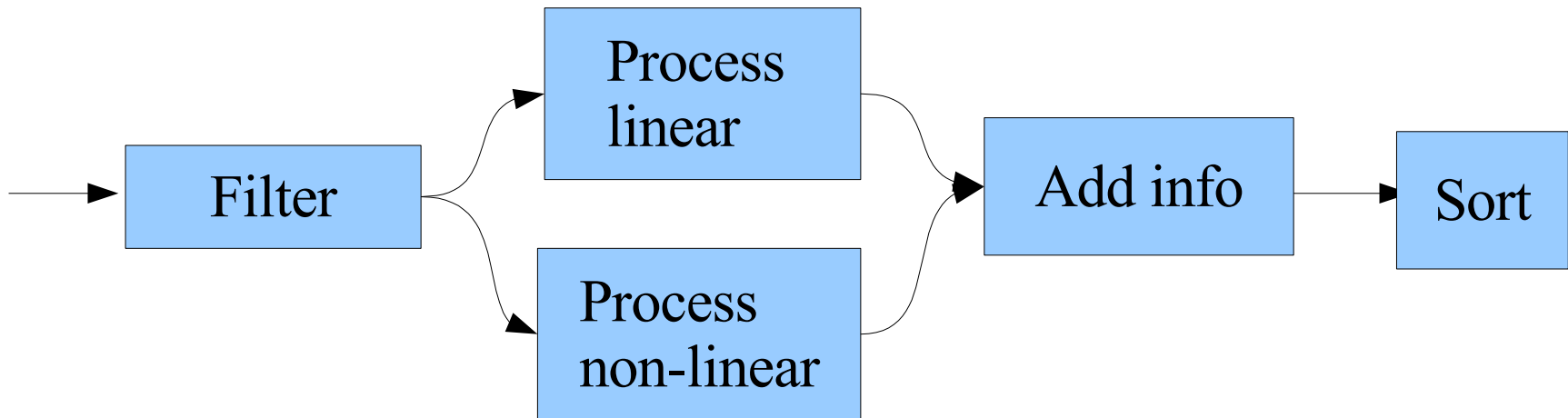
Sequential versus Parallel

- Sequential processing may be not enough
 - Transformations may require expensive computations
 - Computation time may be greater than inter-arrival time
- Parallel processing is difficult
 - User-Defined Functions are difficult to parallelize
 - Semantics must be the same as sequential
 - Event orders
 - States dependencies (rules have local states)

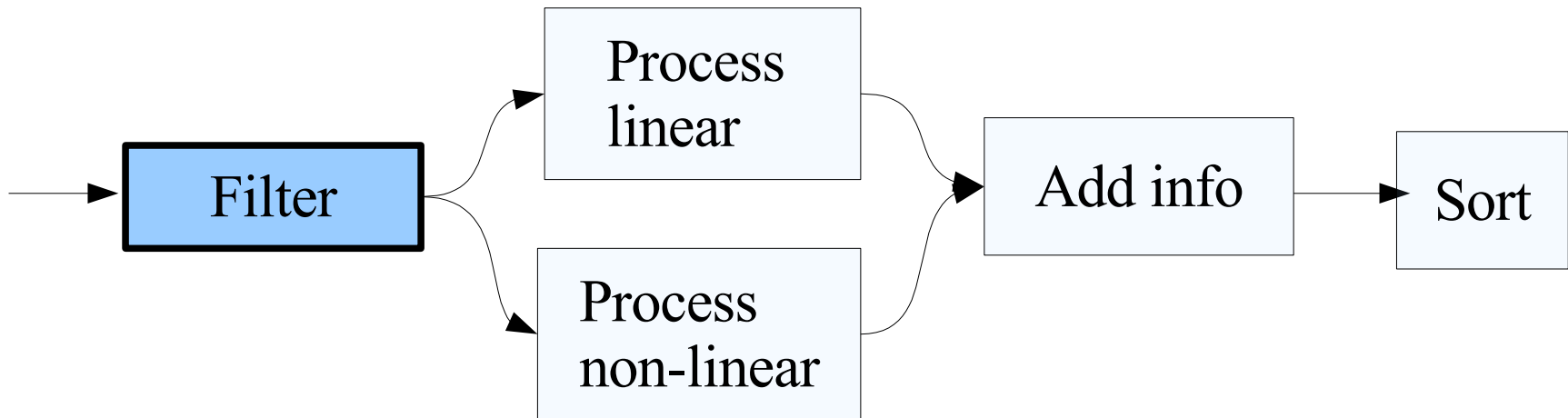
When is parallelization possible?

- Consider events types A and B
 - First, do static analysis on the rule specifications
 - If processing of A can modify some state that affects B
 - Then, create a synchronization between A and B
 - If not, care just about the order

Process Phase



Process Phase

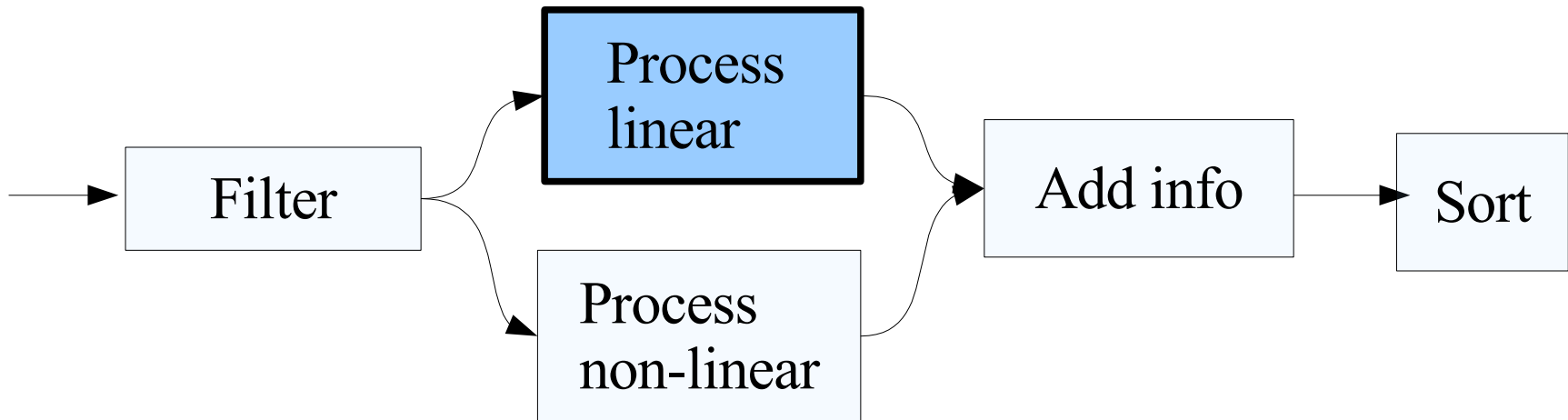


No local state

→ Processing order not important

→ Events may be processed in parallel

Process Phase

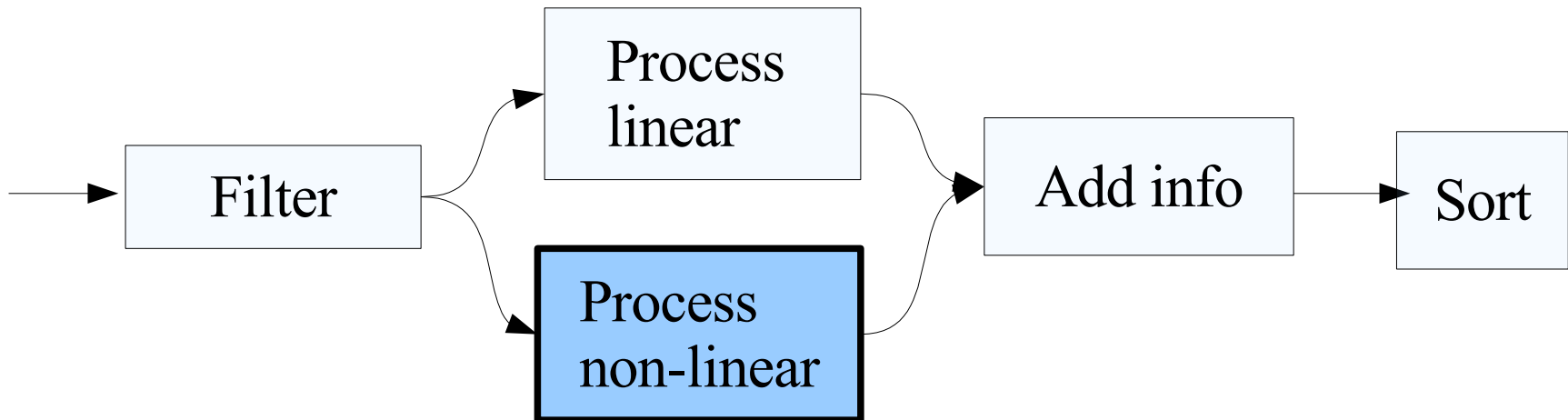


No local state

→ Processing order not important

→ Events may be processed in parallel

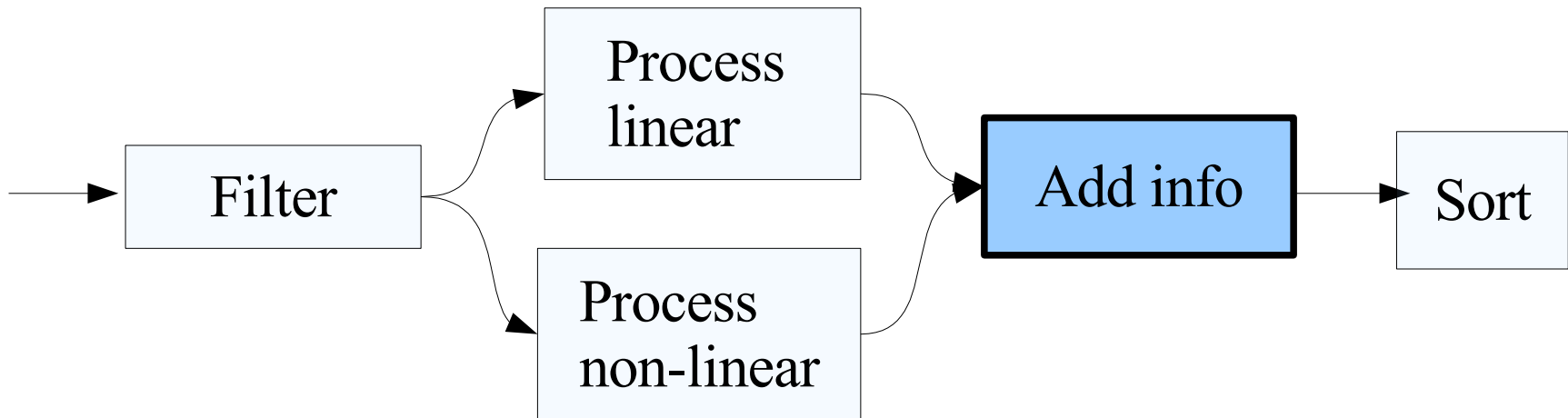
Process Phase



Local state

→ Processing order *is* important

Process Phase

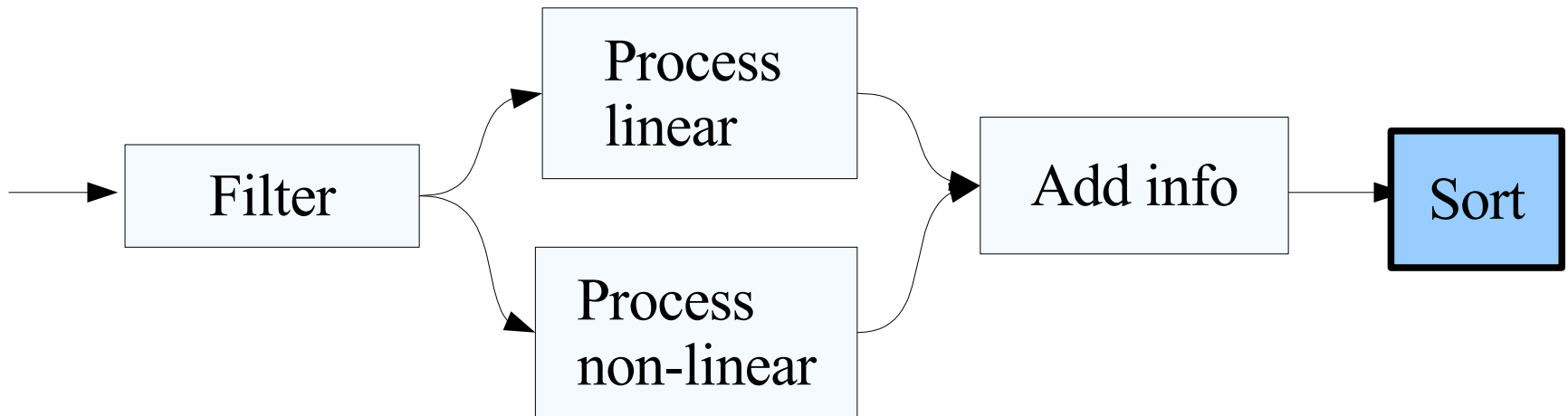


Local state

→ Processing order *is* important

→ Wait for *all* predecessors to make progress

Process Phase



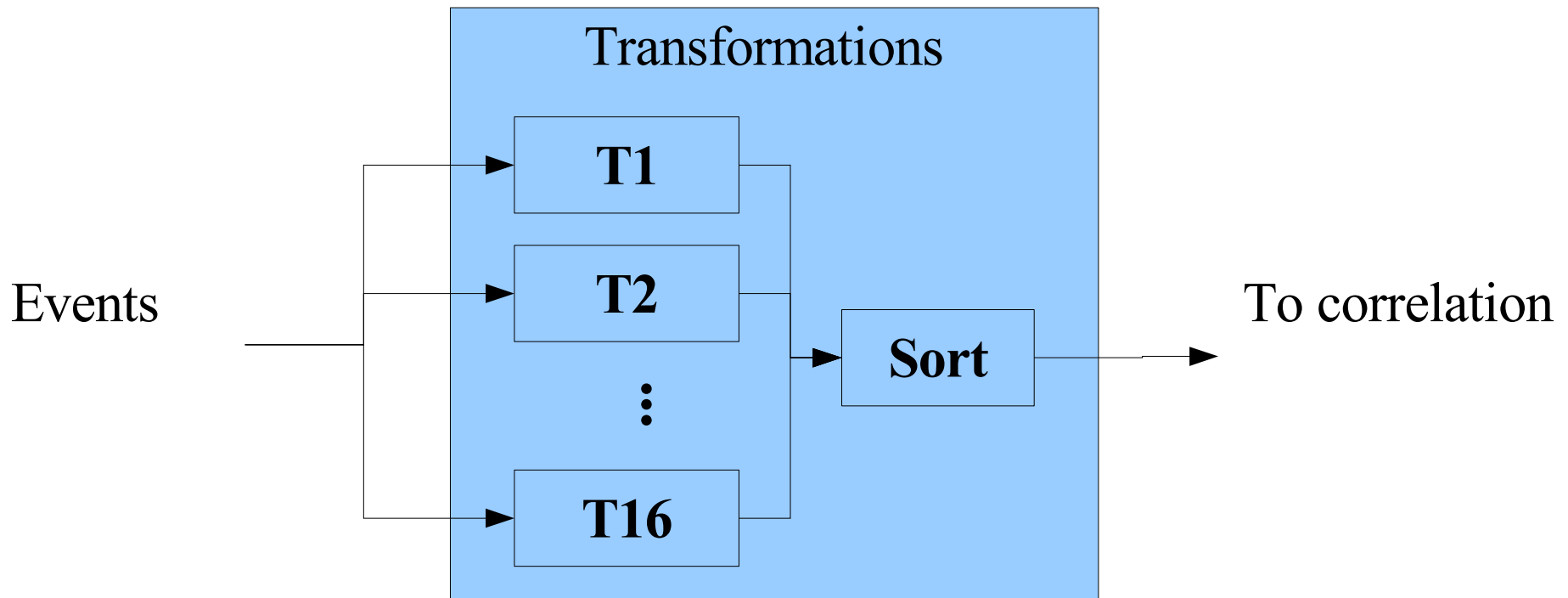
Common successor to all transformations

→ Hold all events with TS x until all events with TS $x-1$ are outputted

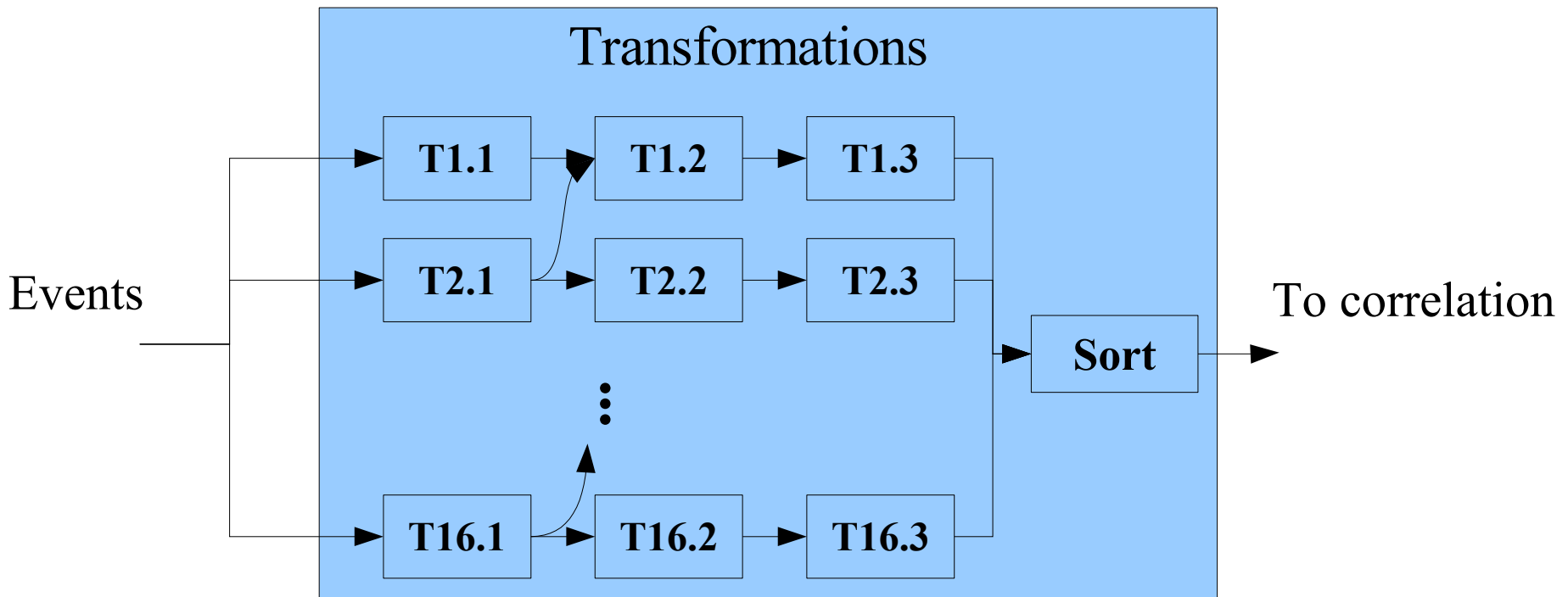
Possible Execution Cases

- Case 1
 - Independent events activating independent rules
- Case 2
 - One type of event triggers multiple rules and/or rules do not have local state
- Case 3
 - Event is processed by a rule with local state

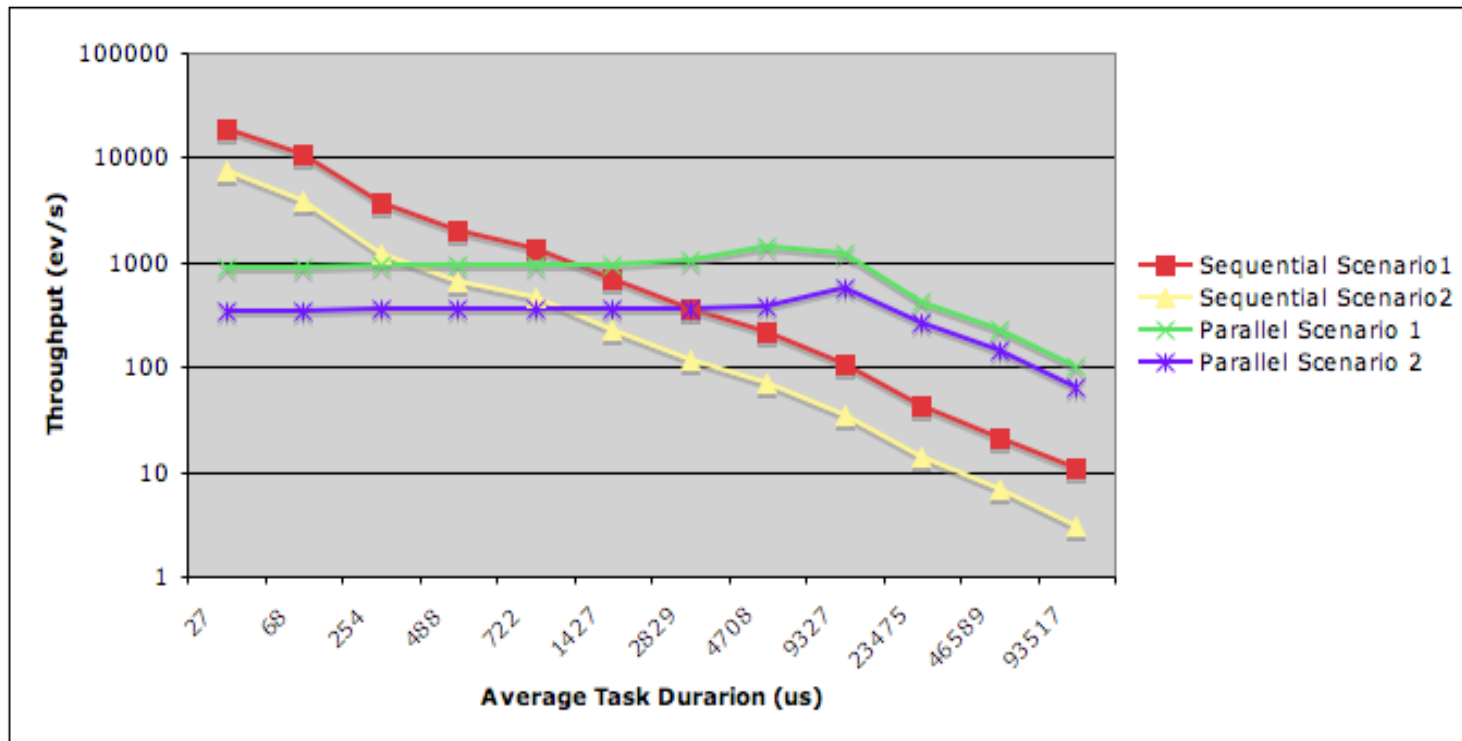
Performance Scenario 1



Performance Scenario 2

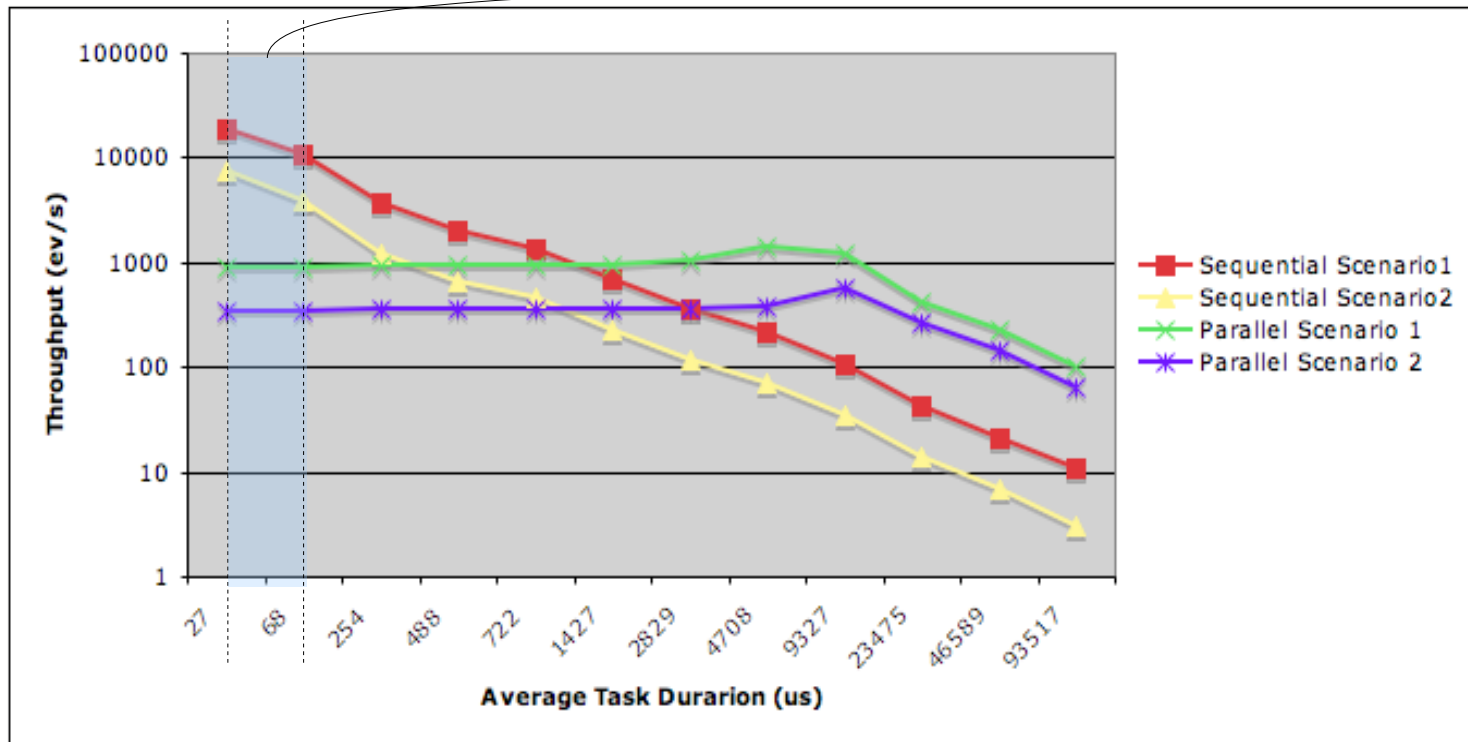


Performance



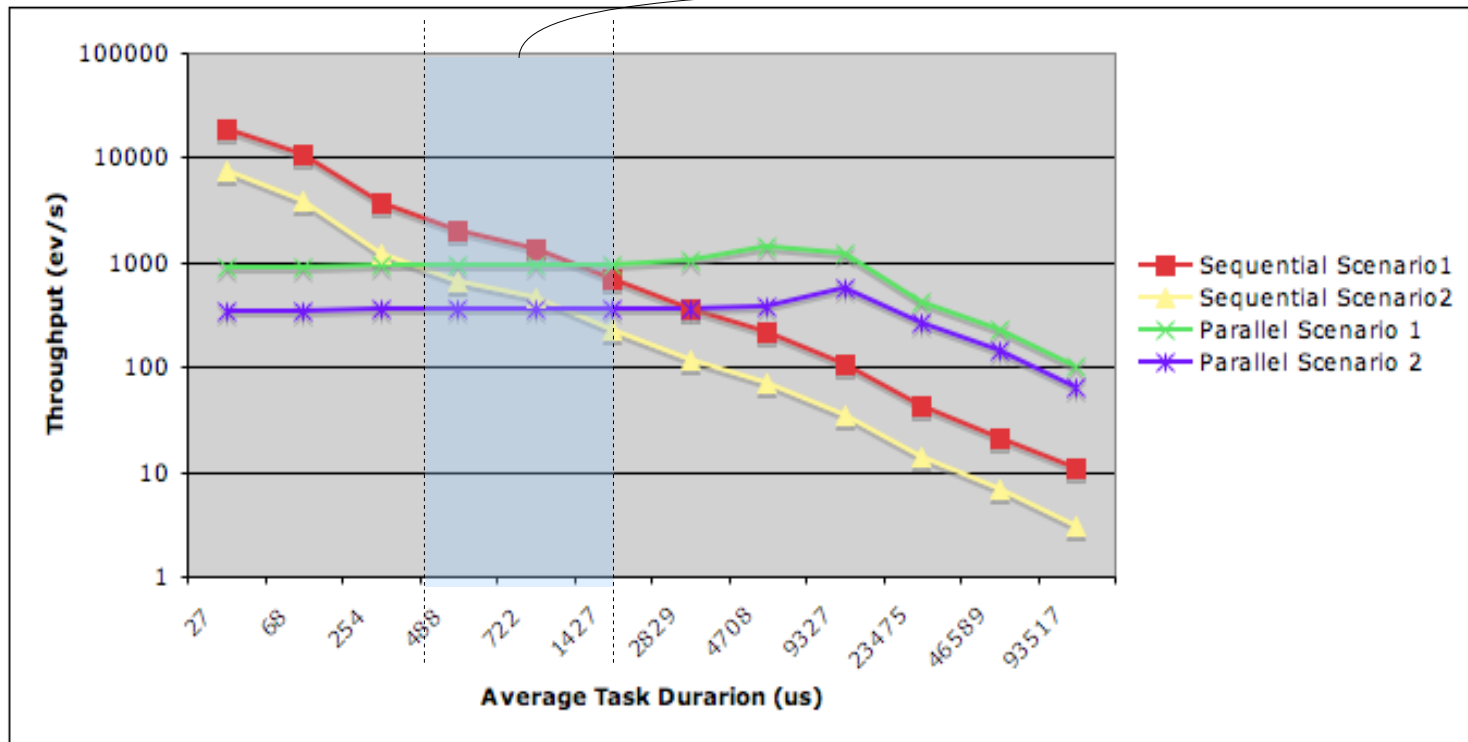
Performance

Simple computation → Speedup: 0.10

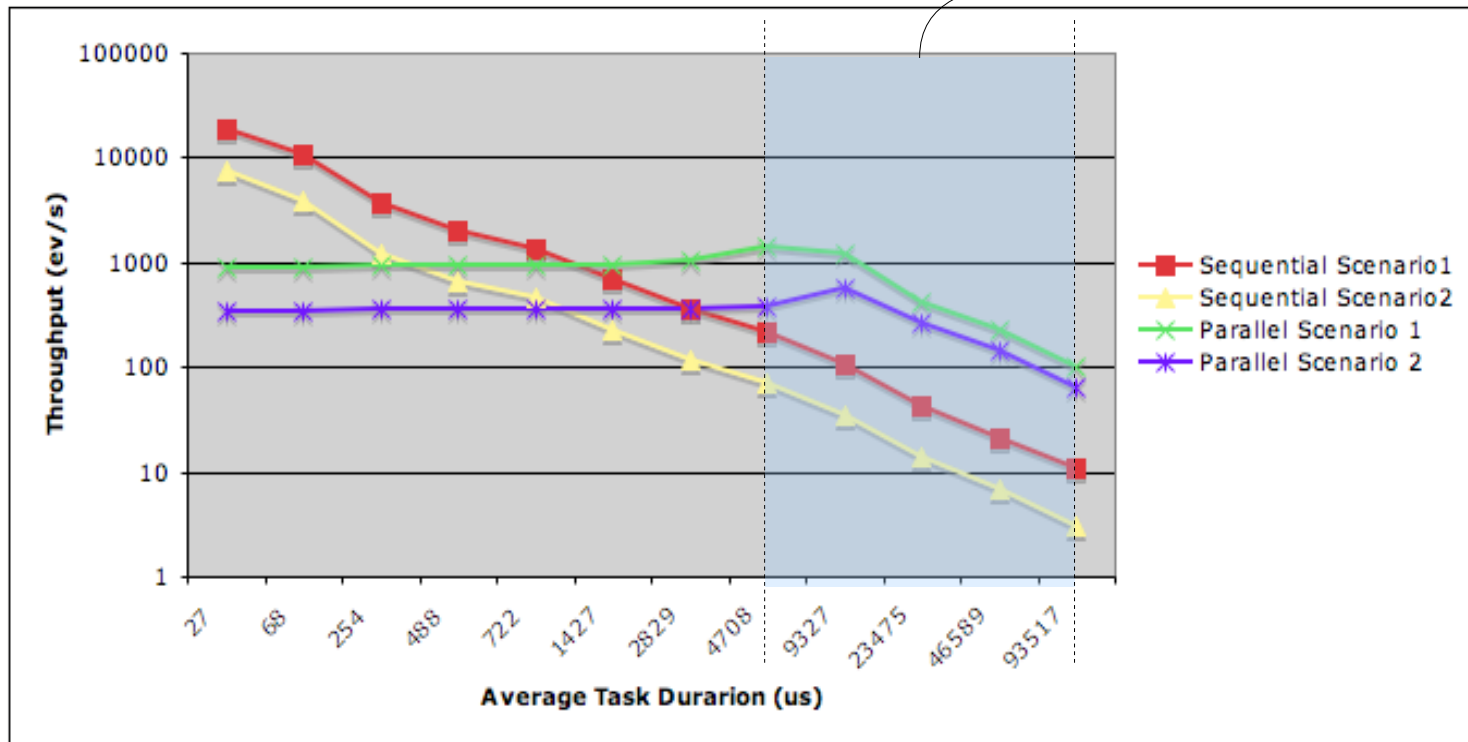


Performance

IO bounded ops. → Speedup: 0.5 - 1.5



Performance Complex computation → Speedup: 6 - 20



Current Work

- **Adaptation**
 - Sequential processing is some times faster
- **“Helping”**
 - How faster threads could help slower ones?
- **Speculation**
 - The relevant state does not always change
- **More flexible synchronization**
 - Improve dependency analysis towards “per event” instead of “per type”

Conclusion

- Transformations with User-Defined Functions are likely to be bottlenecks
 - Our approach achieve over 20x speed-up
- No need for explicit parallelization code
- Only change to the event processing language: no global variables

Thank you!



<http://wwwse.inf.tu-dresden.de>