POLITÉCNICA

StreamCloud: an Elastic Parallel-Distributed Stream Processing Engine

Ph.D. Student
Vincenzo Massimiliano Gulisano

Director: Ricardo Jiménez Peris
Co-director: Patrick Valduriez

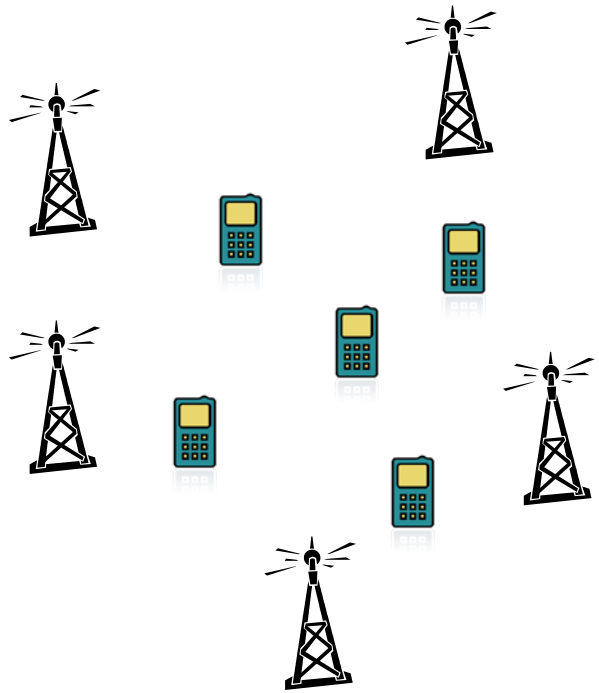
Lsd Distributed Systems Laboratory

December 20, 2012

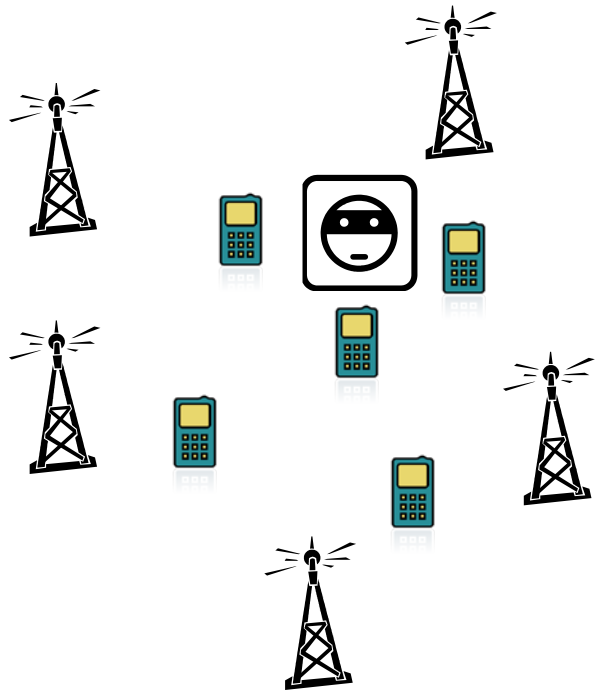
StreamCloud in a Nutshell

- Sample data streaming application (fraud detection)
- Pioneer Stream Processing Engines (SPEs)
- Contributions

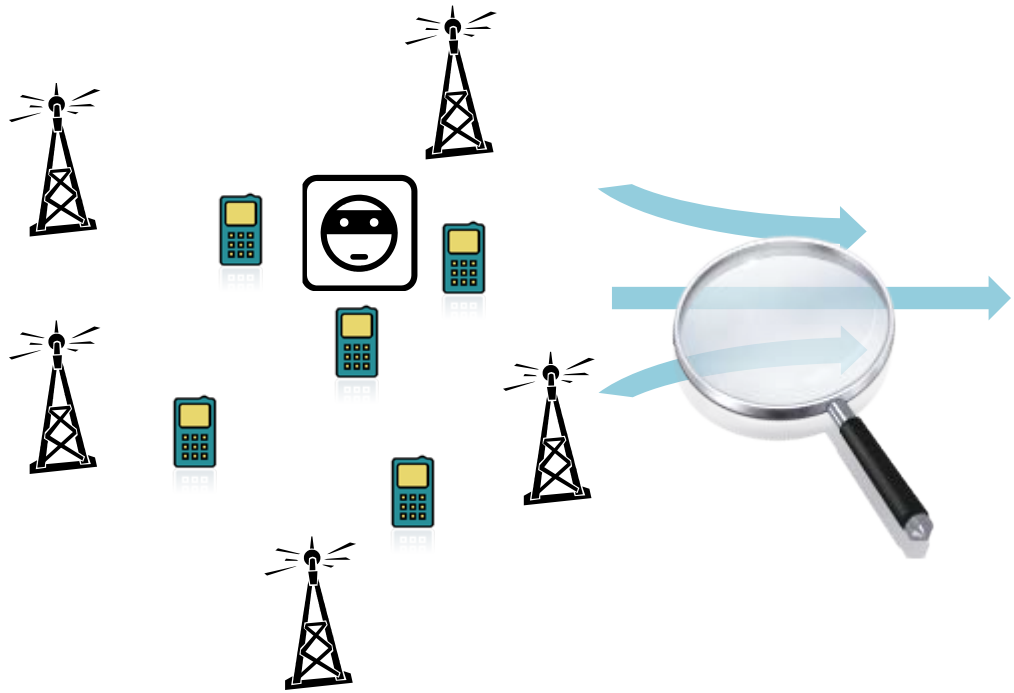
Motivation - Fraud detection



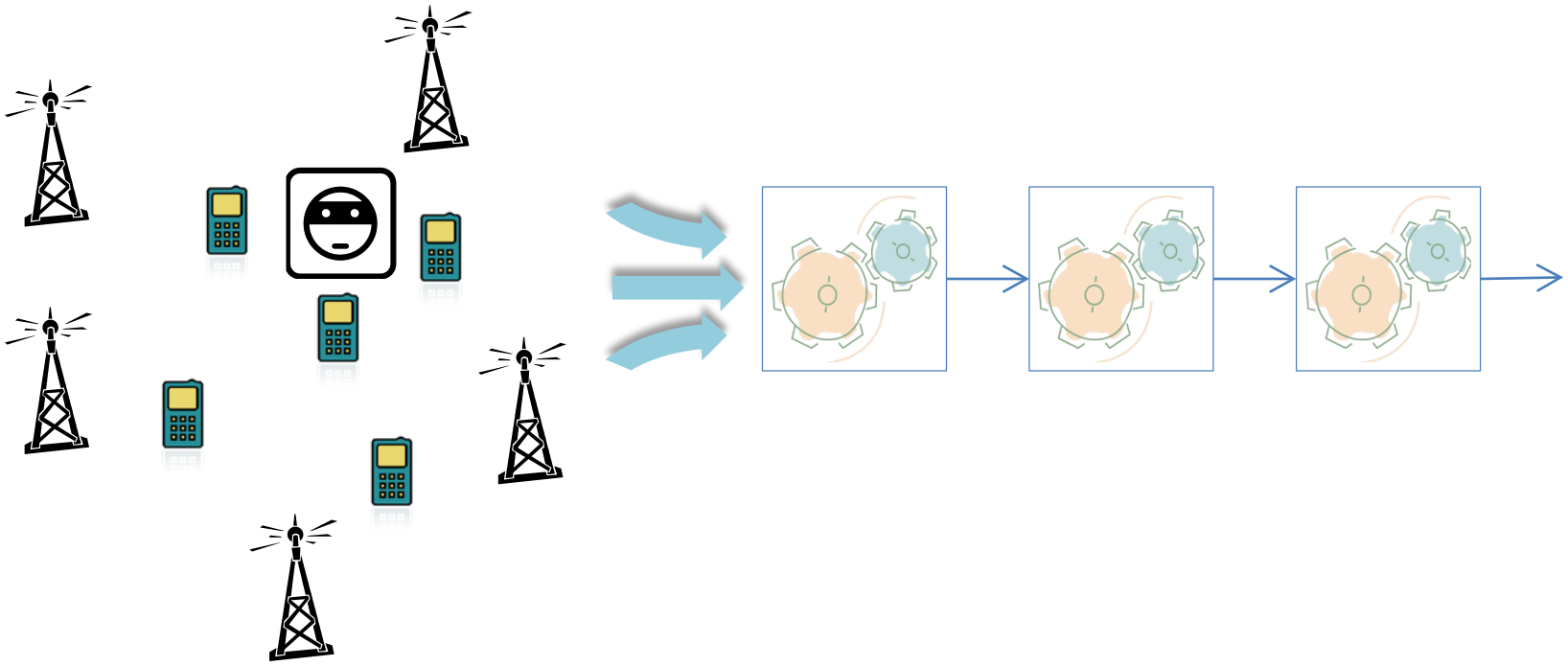
Motivation - Fraud detection



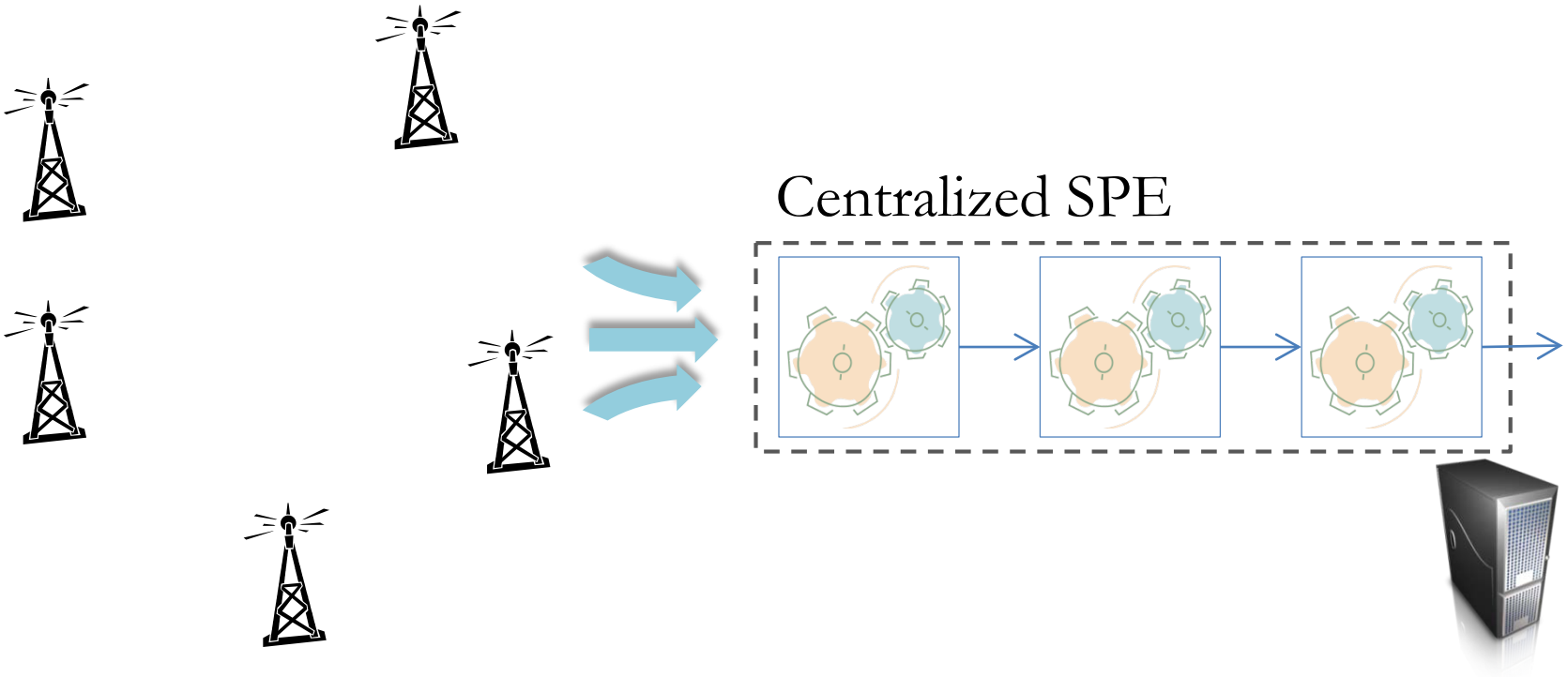
Motivation - Fraud detection



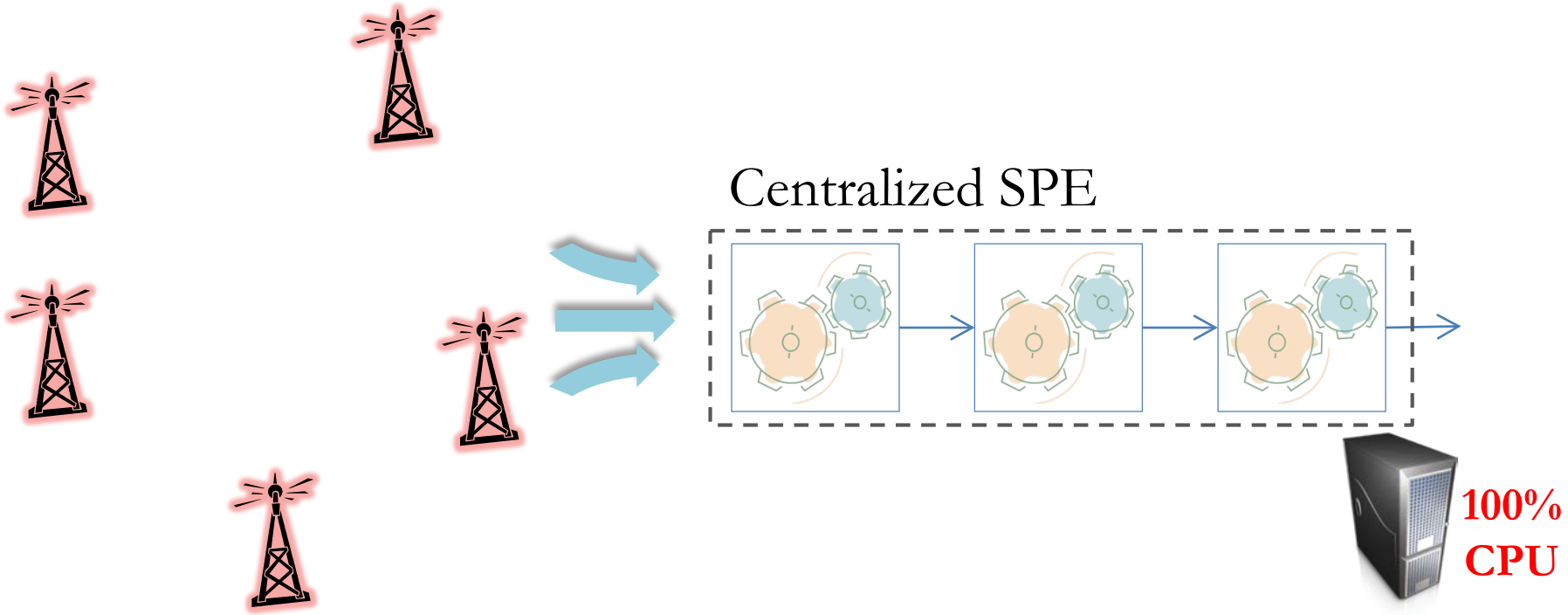
Motivation - Fraud detection



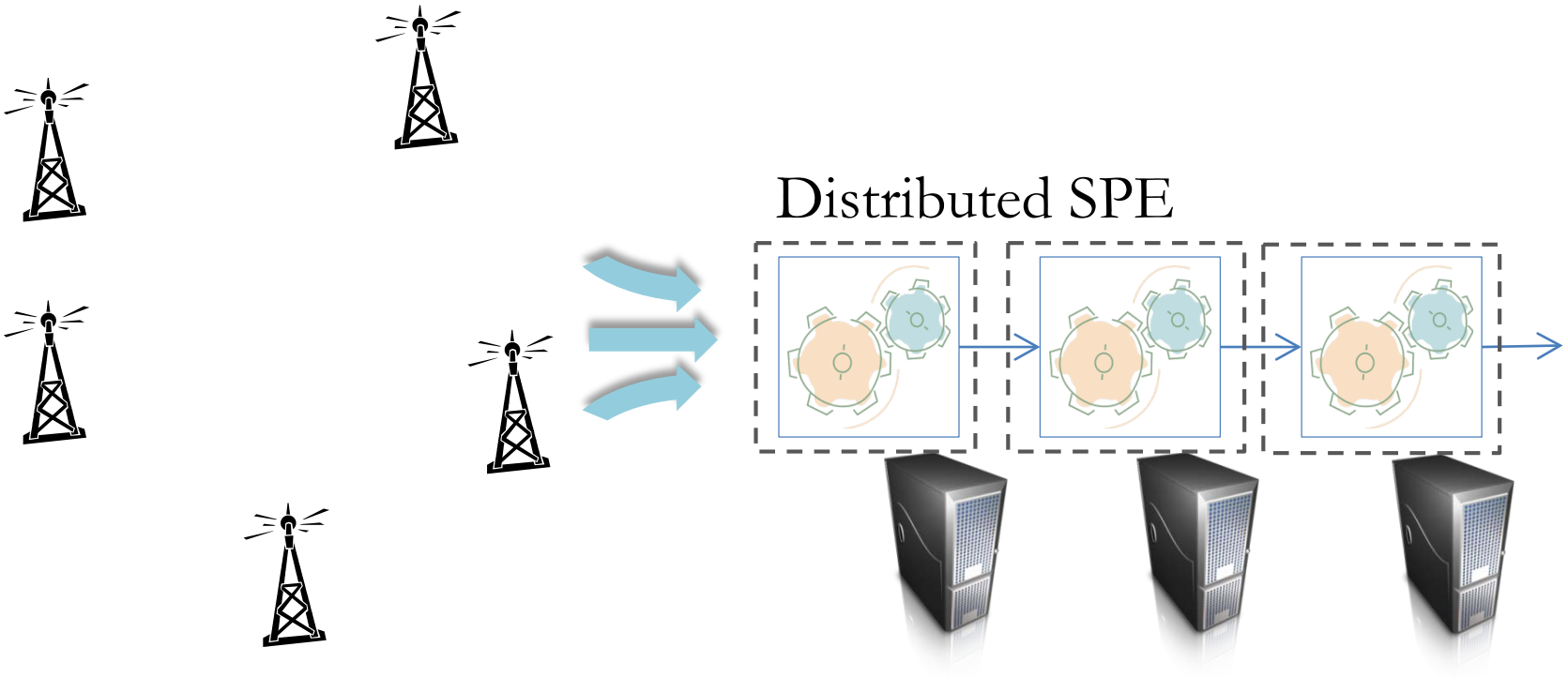
Background - Pioneer SPEs



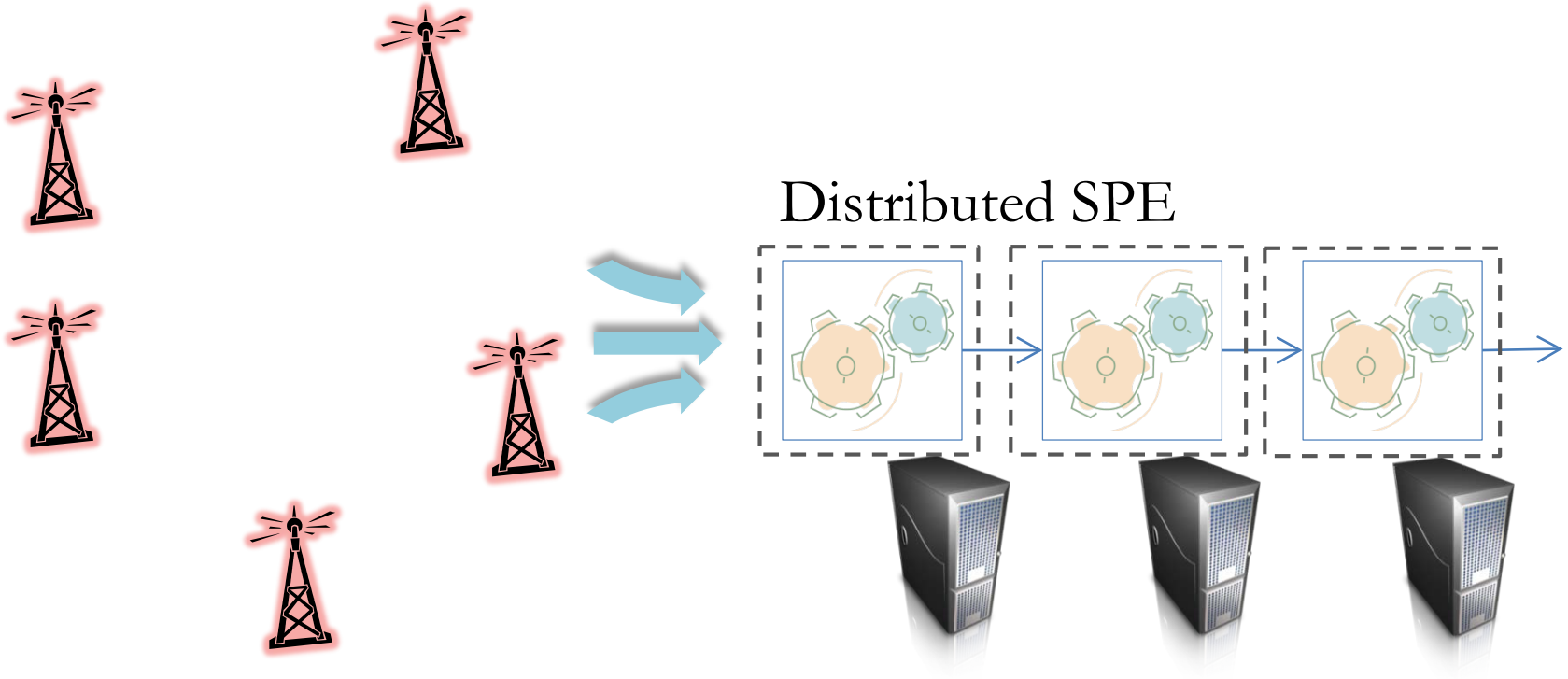
Background - Pioneer SPEs



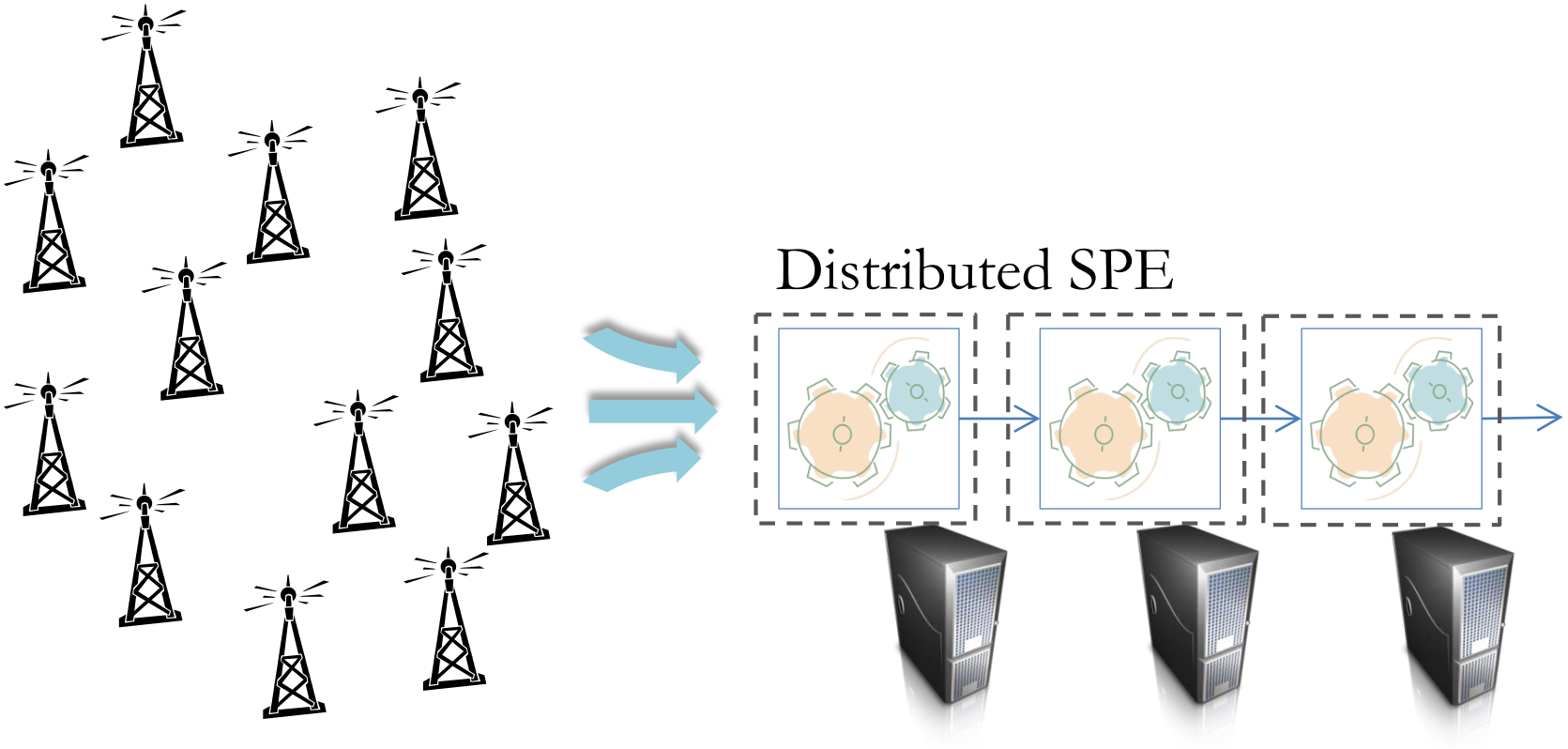
Background - Pioneer SPEs



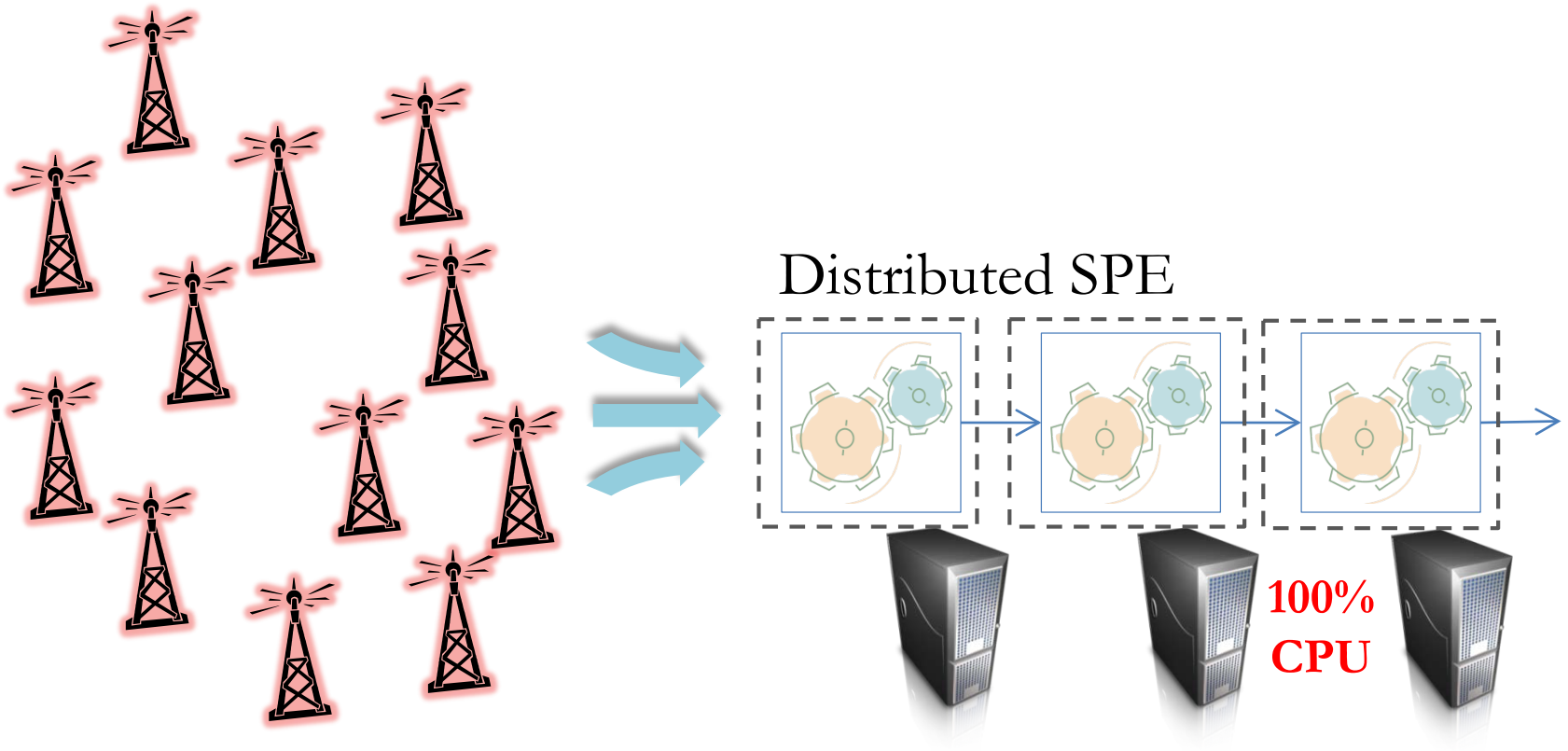
Background - Pioneer SPEs



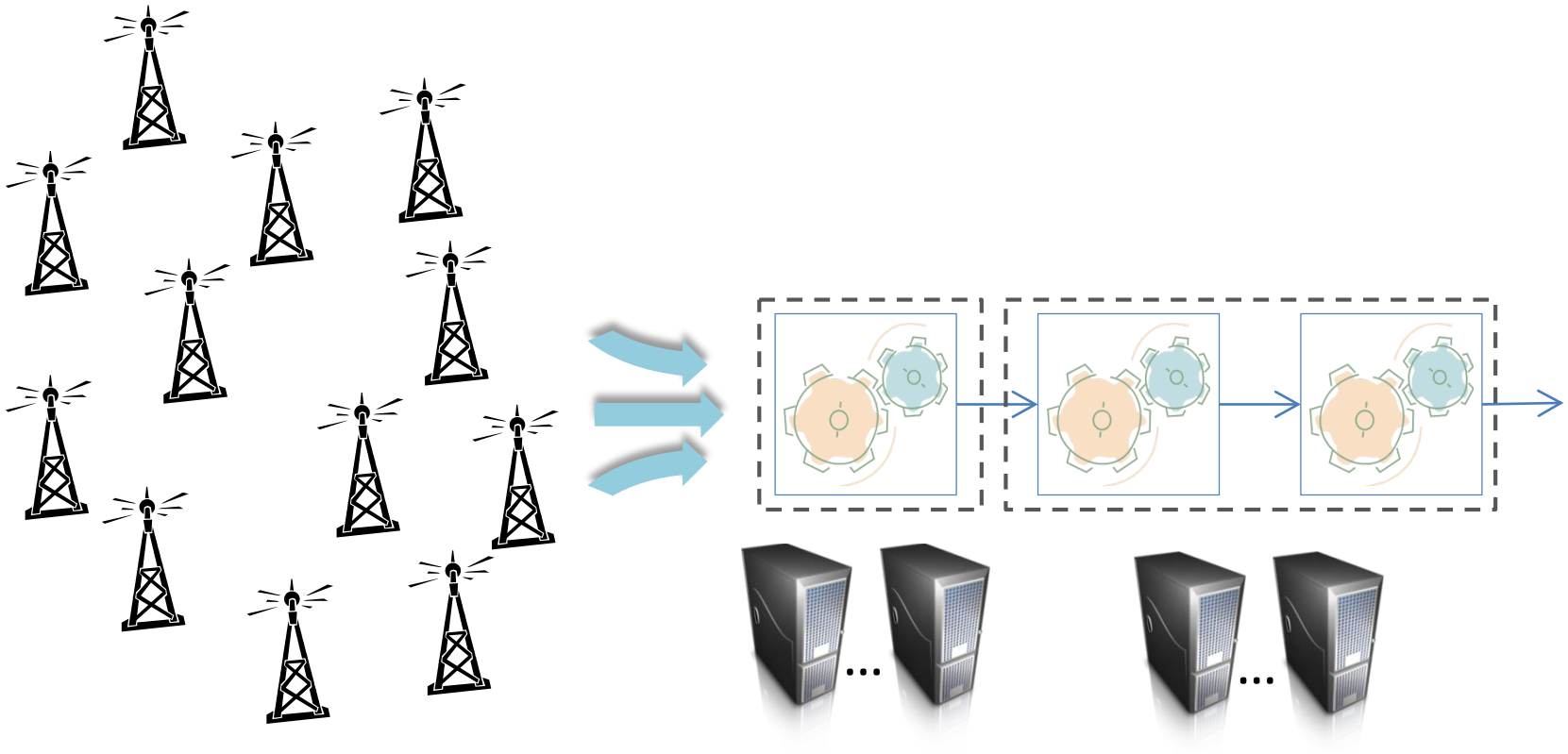
Background - Pioneer SPEs



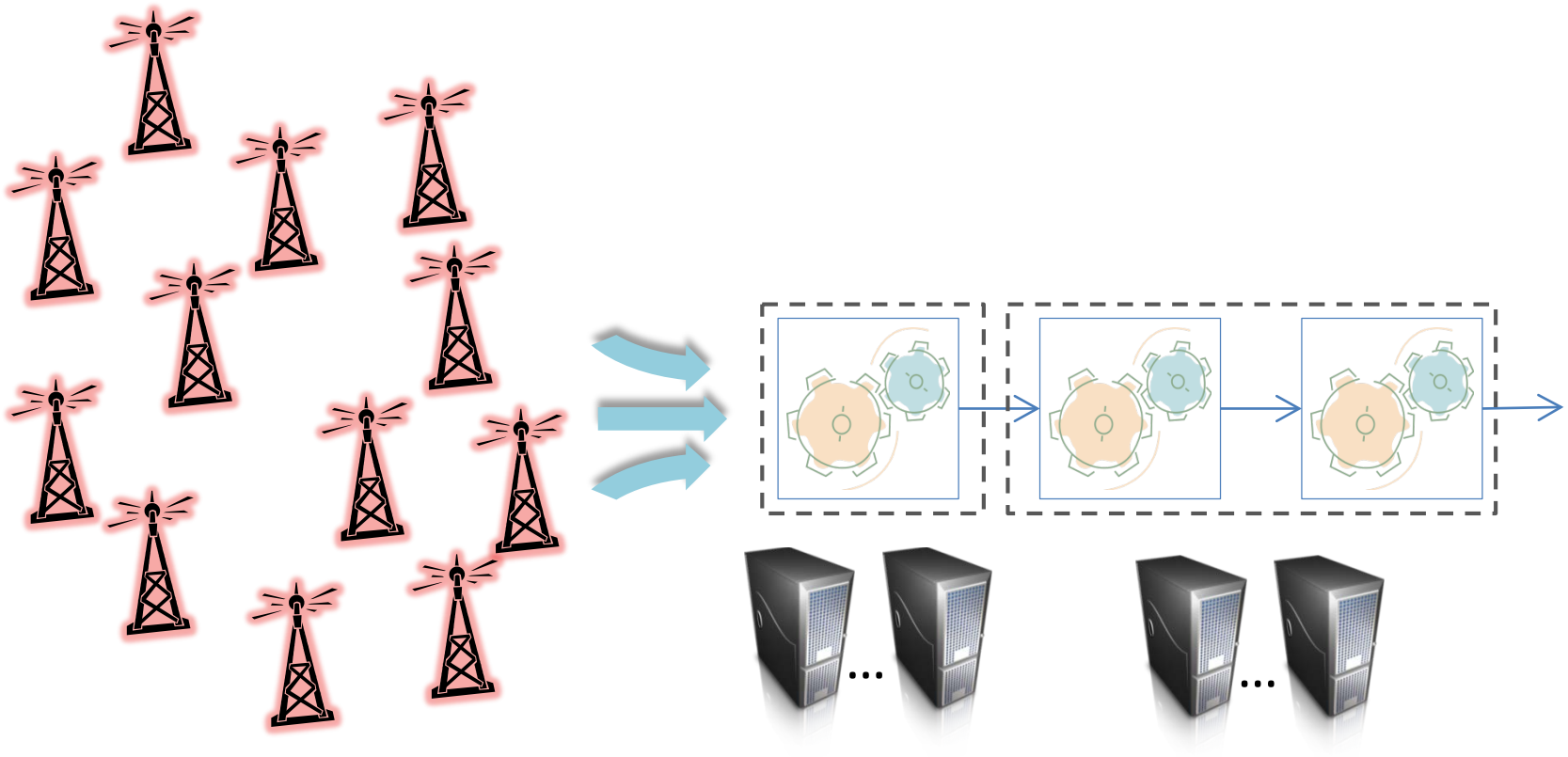
Background - Pioneer SPEs



Contributions - StreamCloud

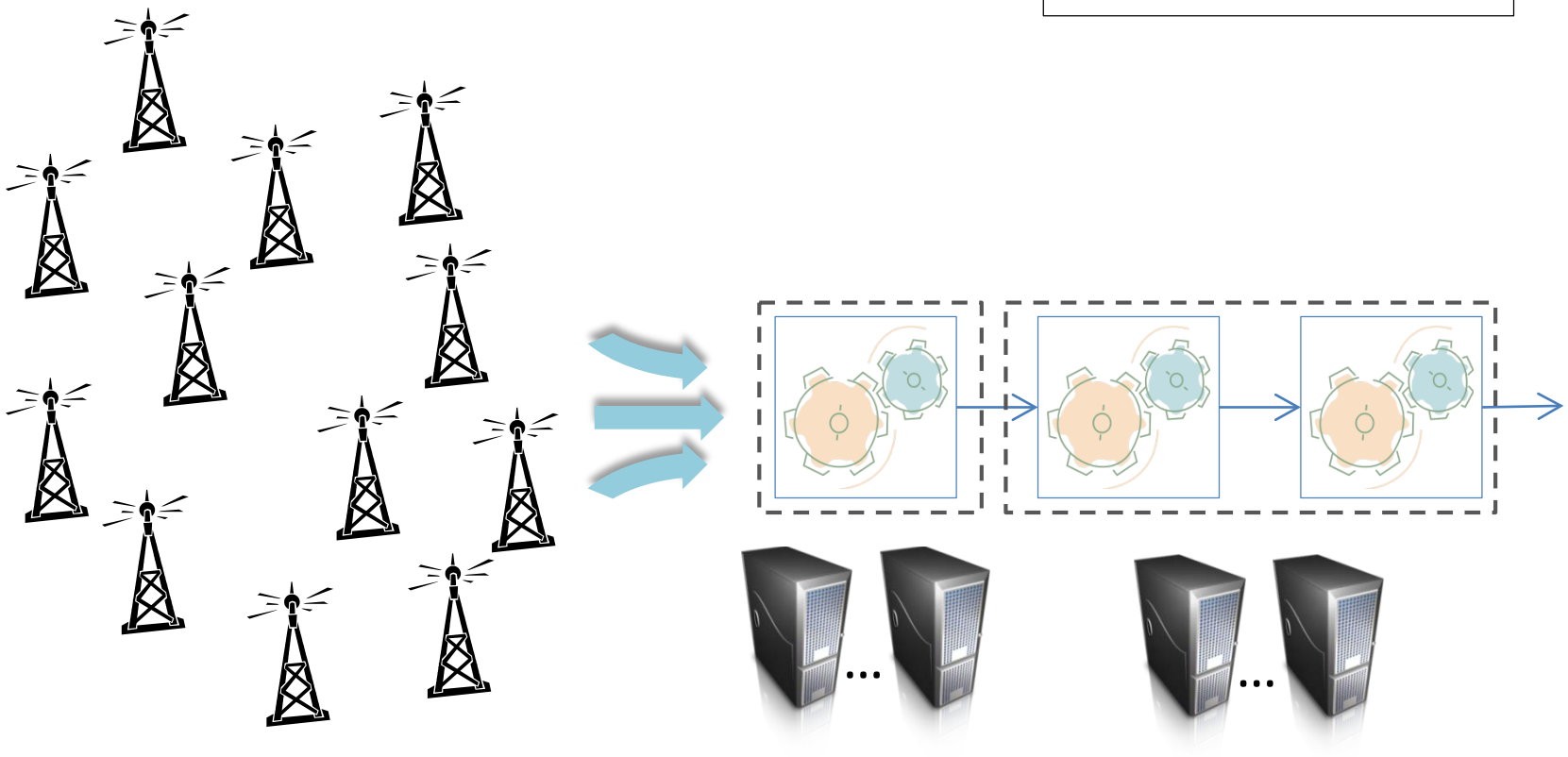


Contributions - StreamCloud

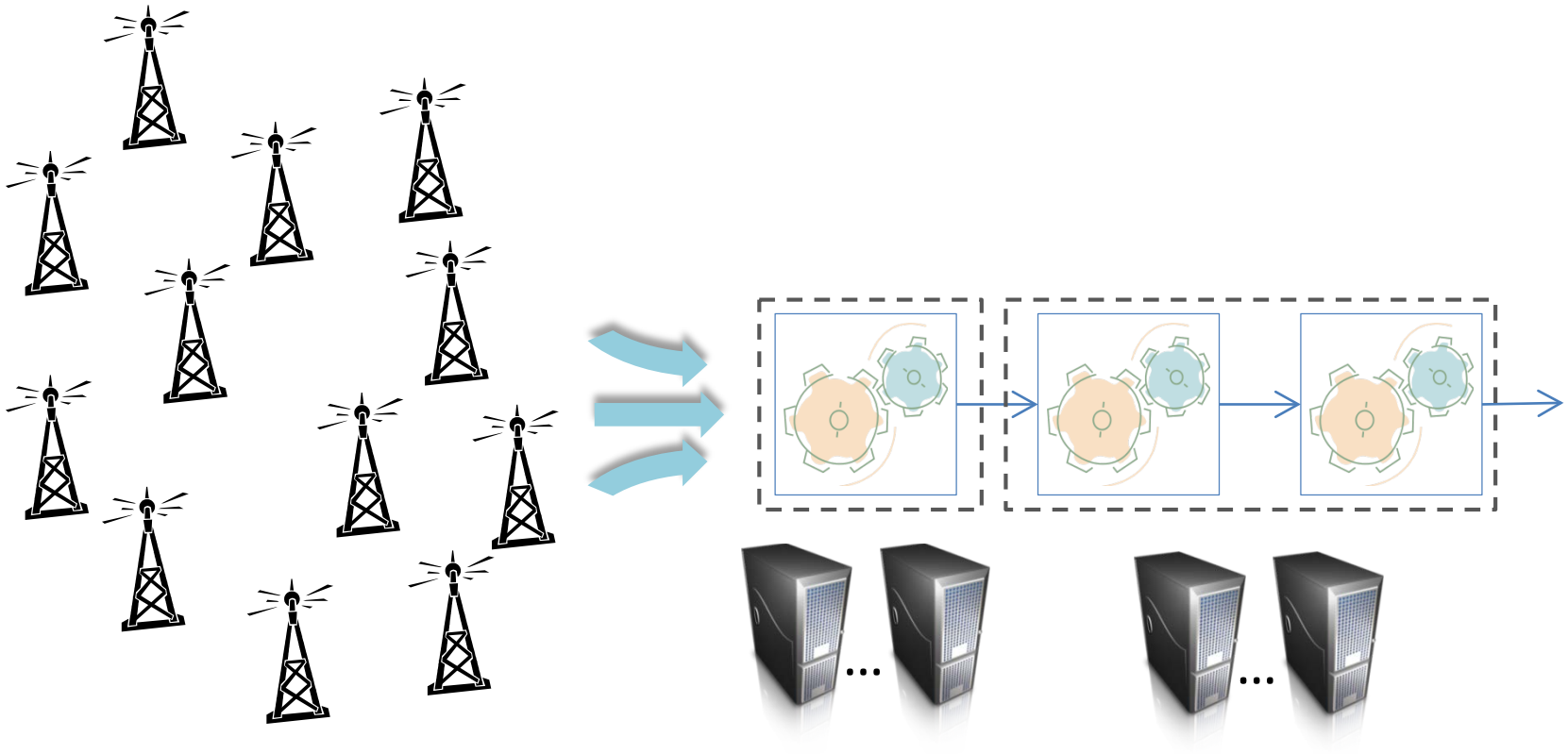


Contributions - StreamCloud

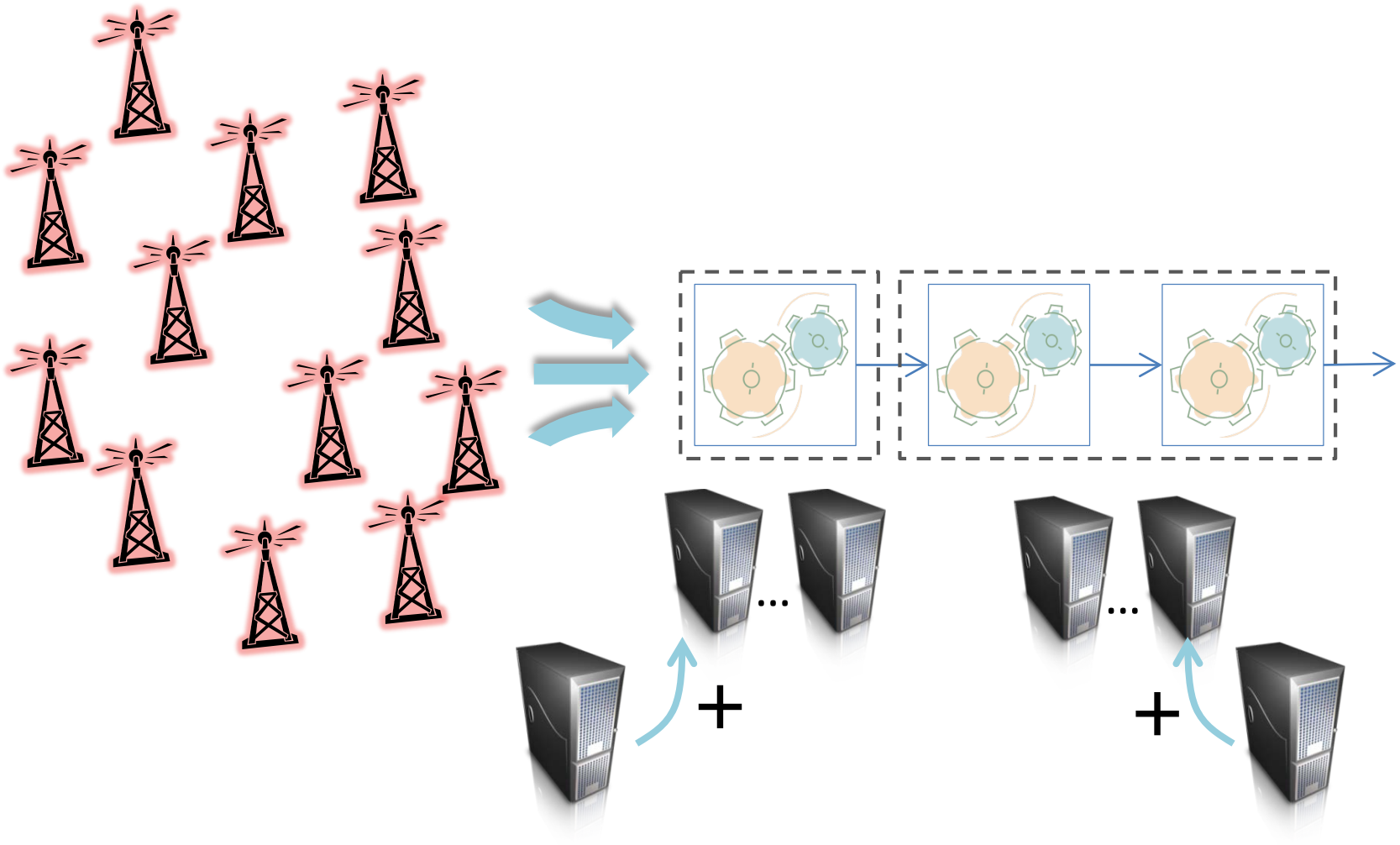
1 Parallelization



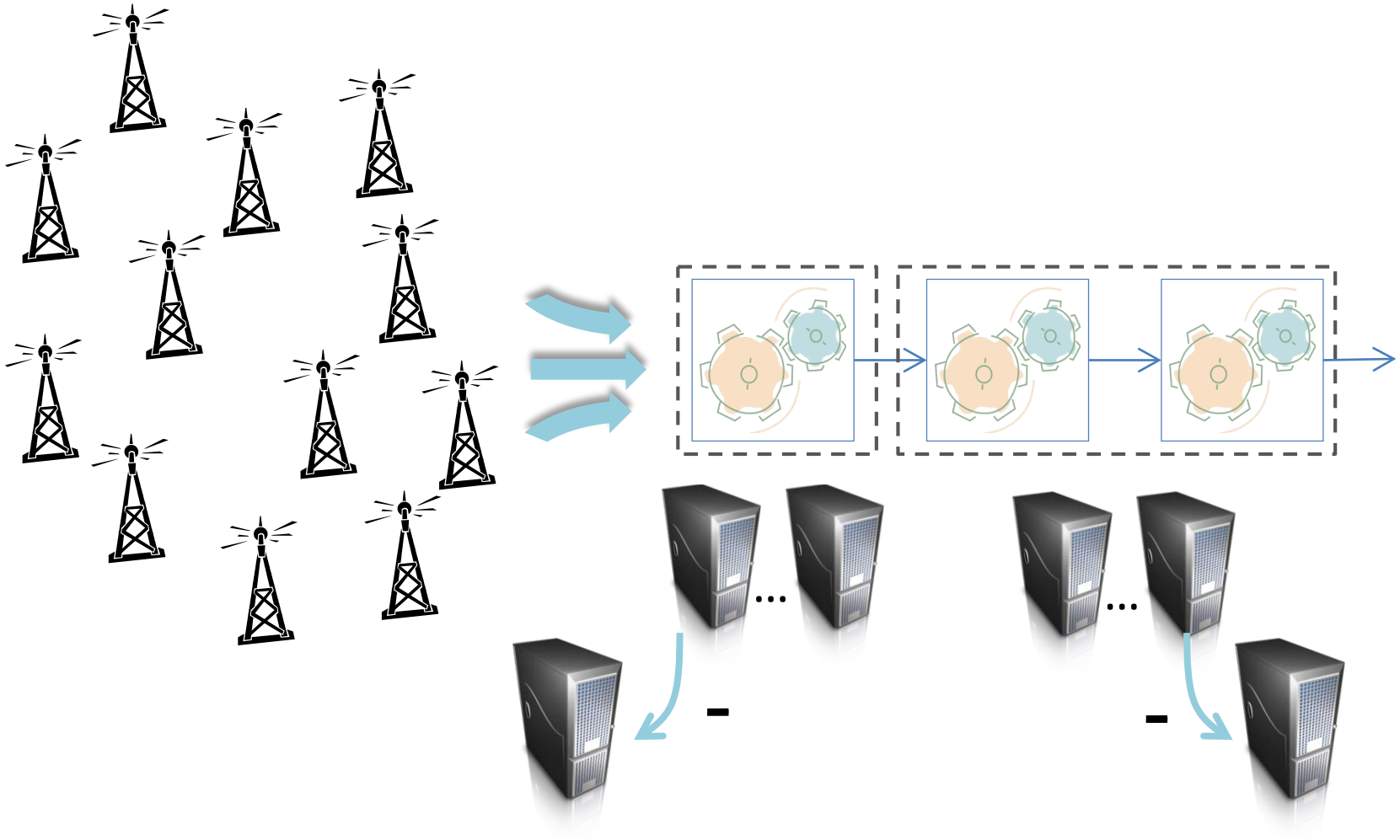
Contributions - StreamCloud



Contributions - StreamCloud

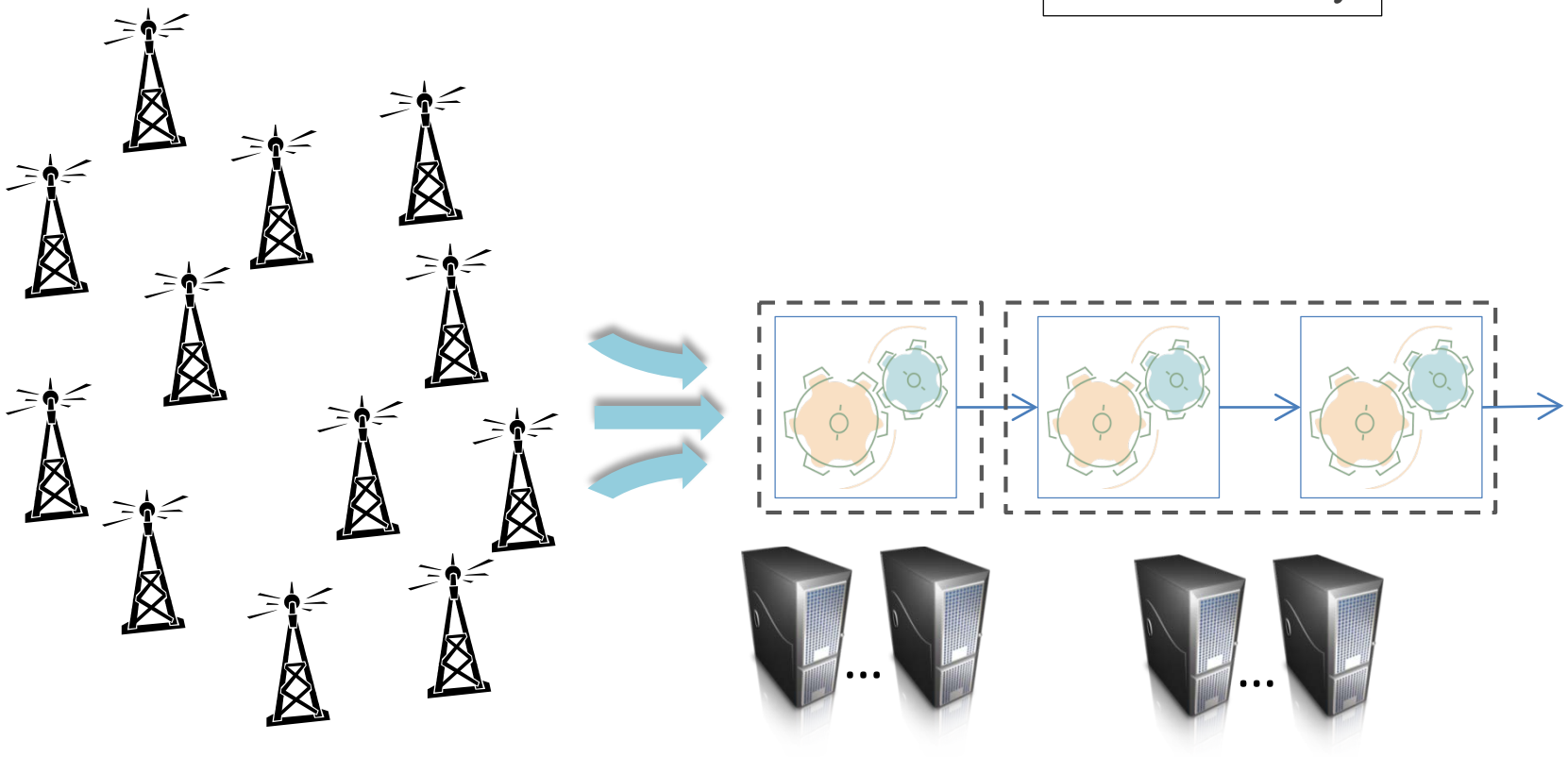


Contributions - StreamCloud

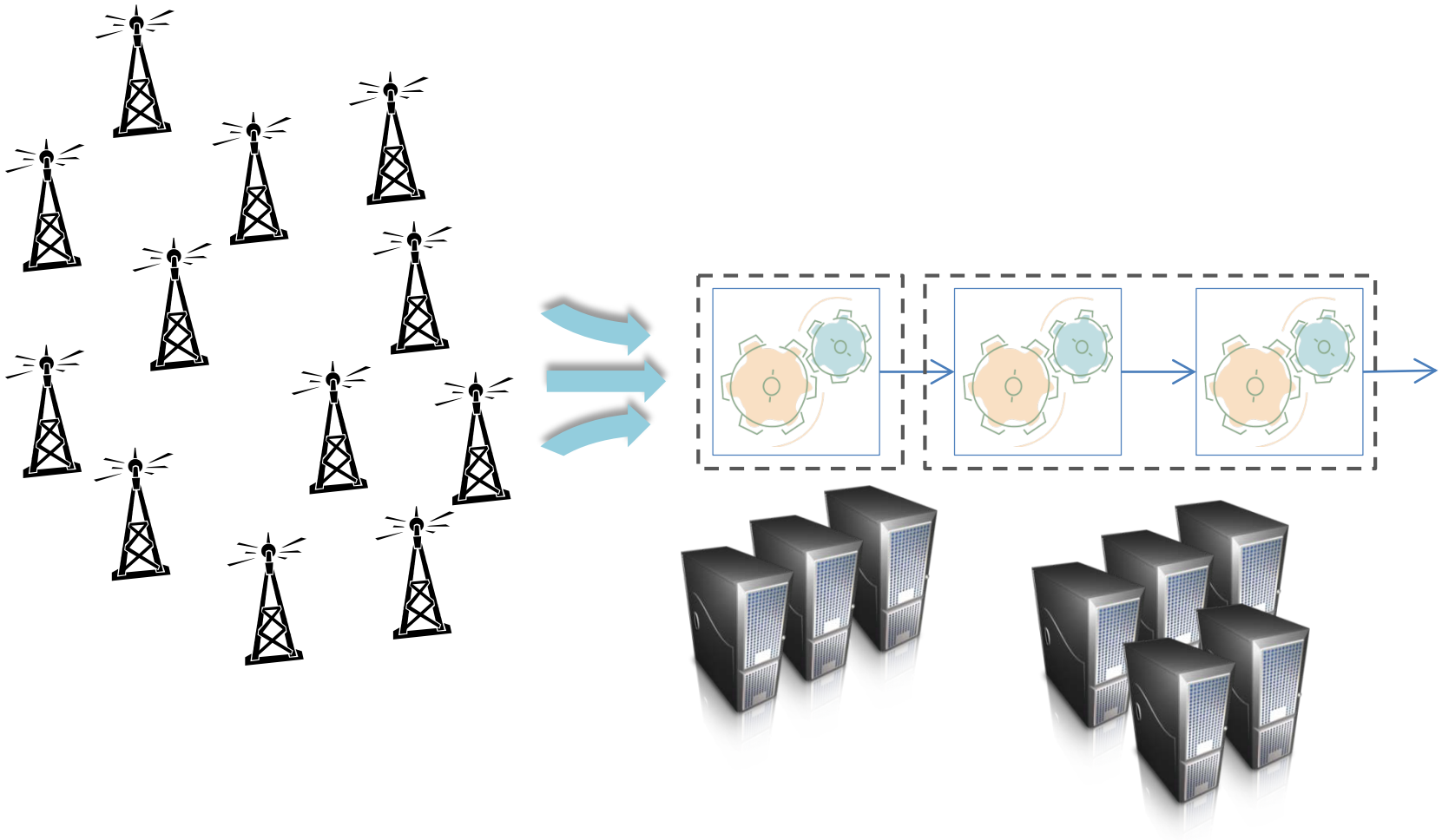


Contributions - StreamCloud

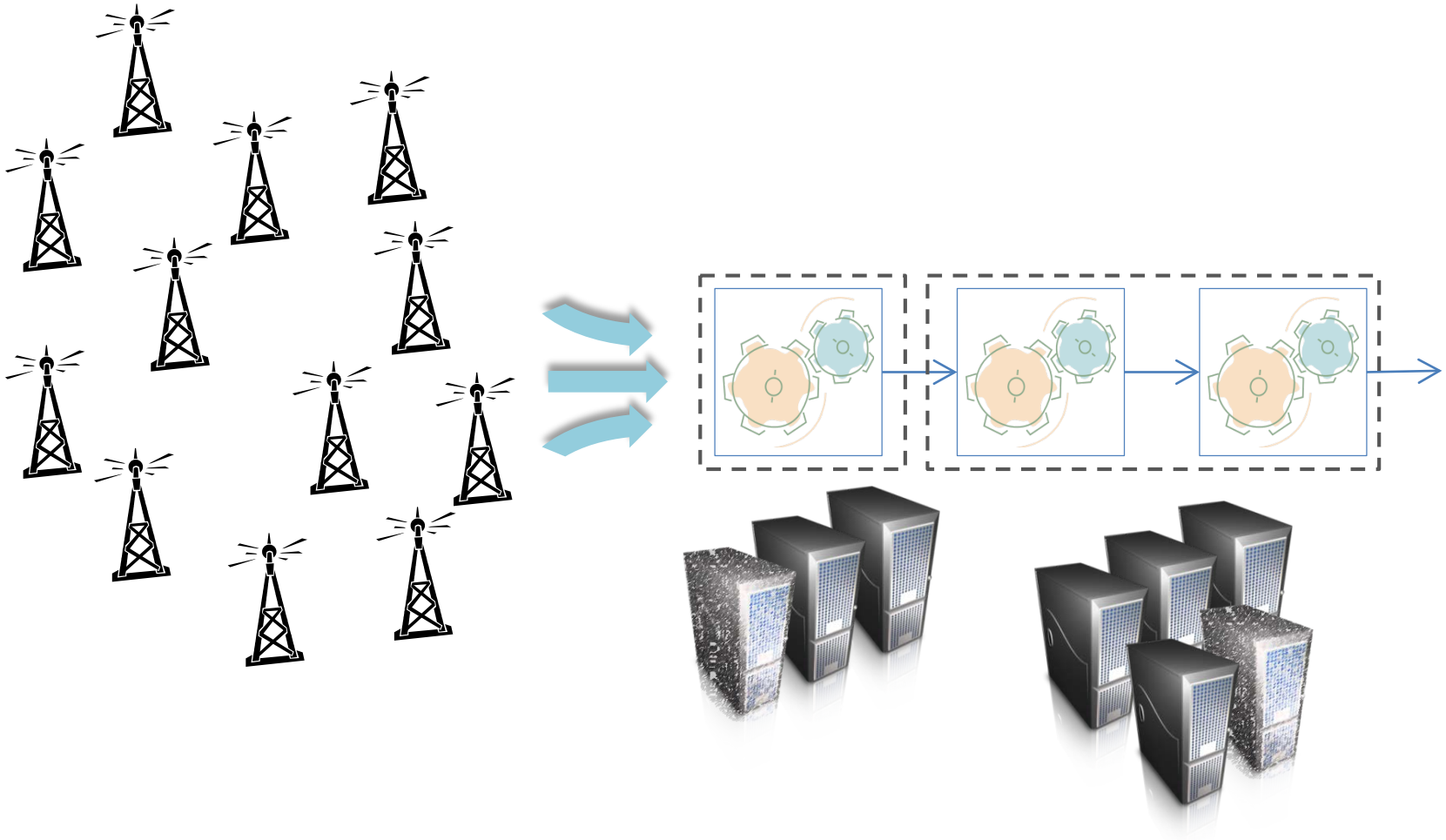
2 Elasticity



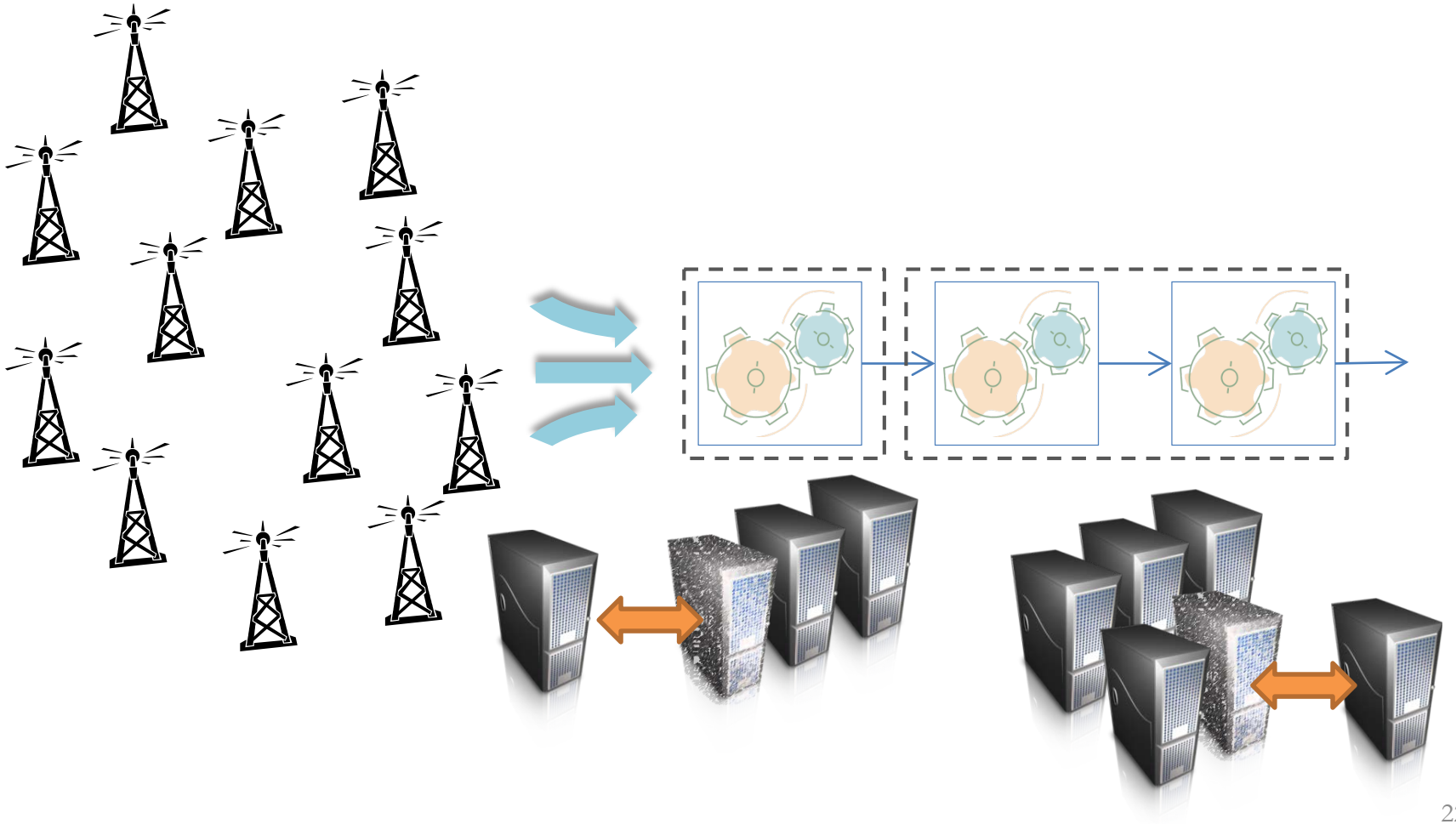
Contributions - StreamCloud



Contributions - StreamCloud

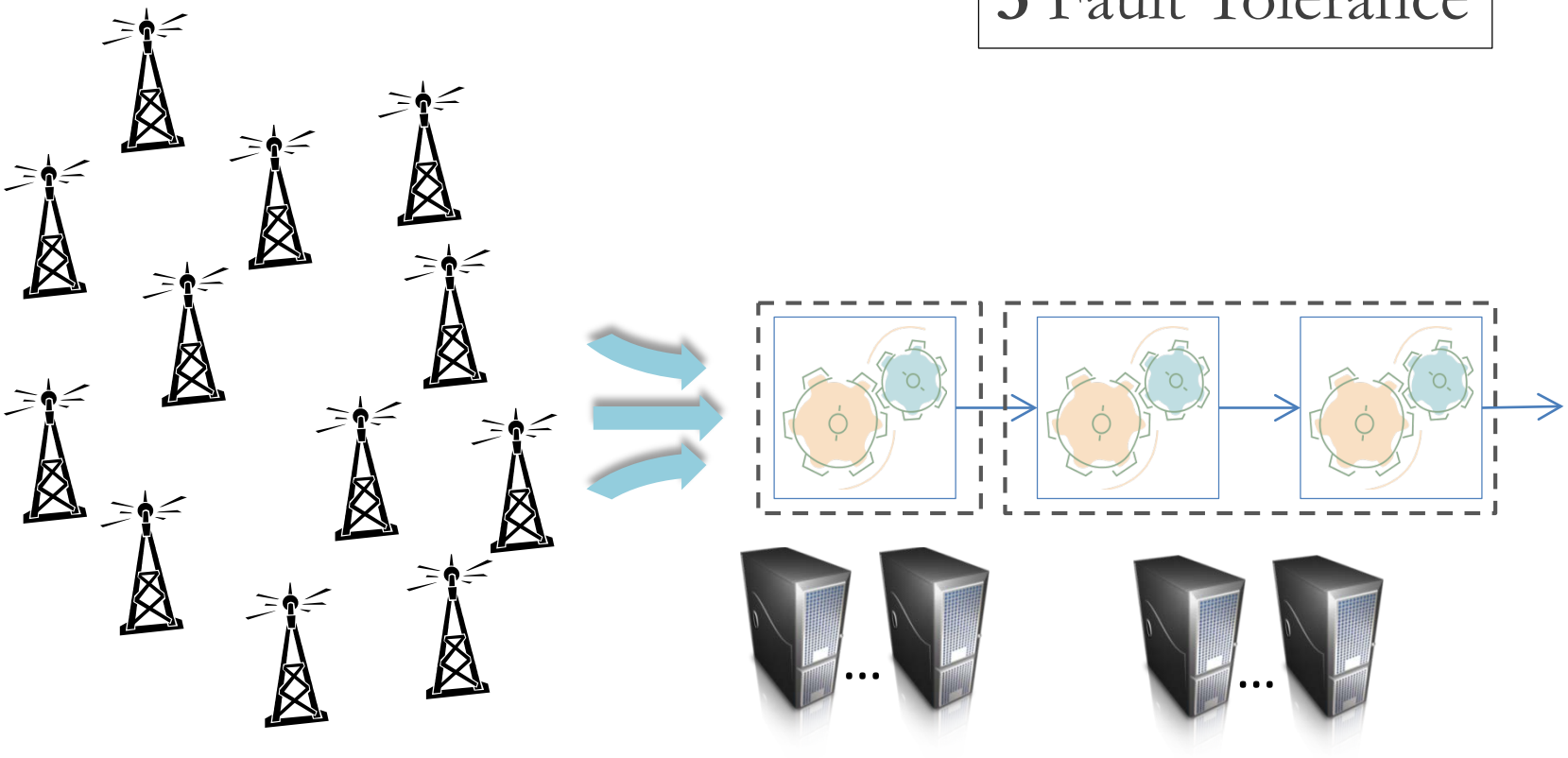


Contributions - StreamCloud

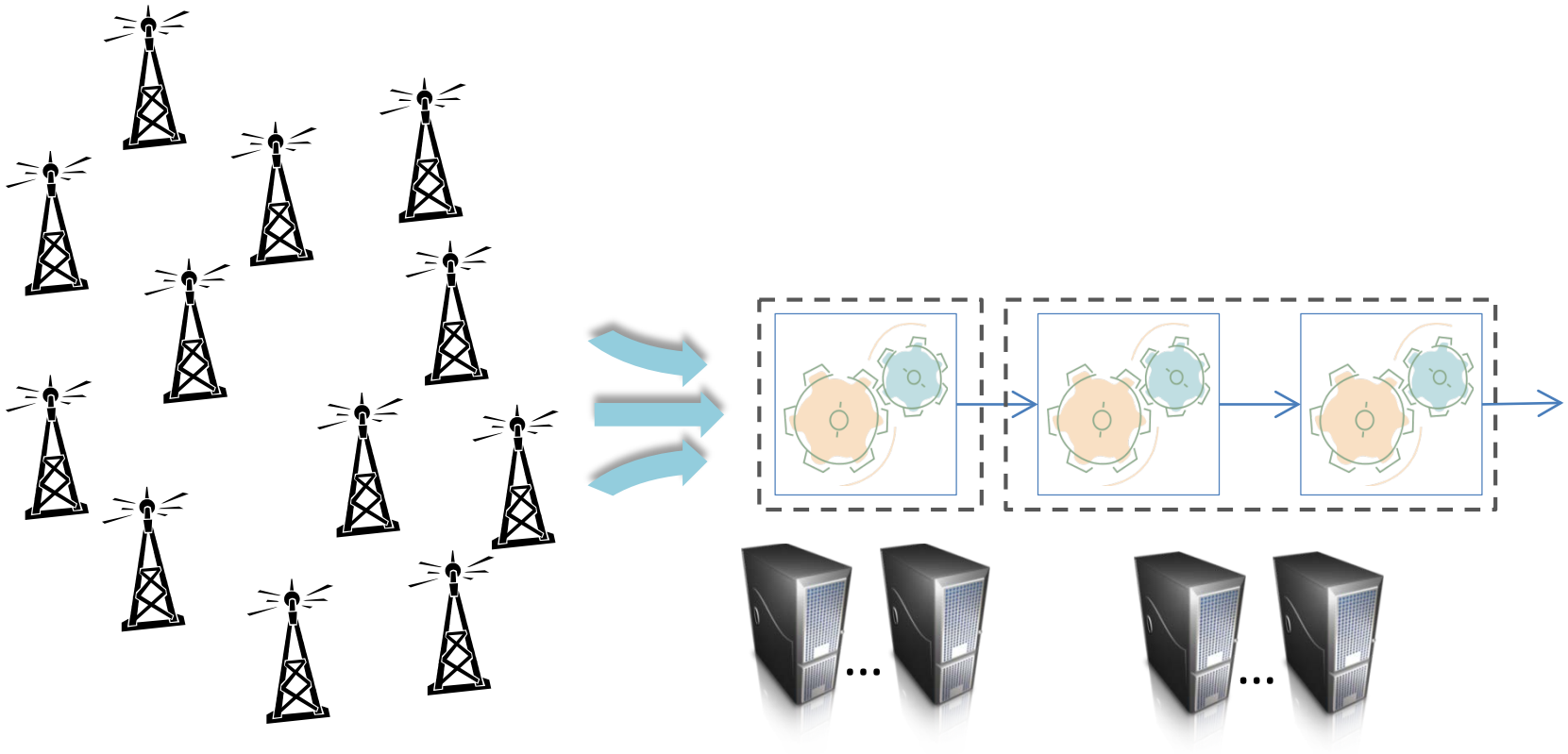


Contributions - StreamCloud

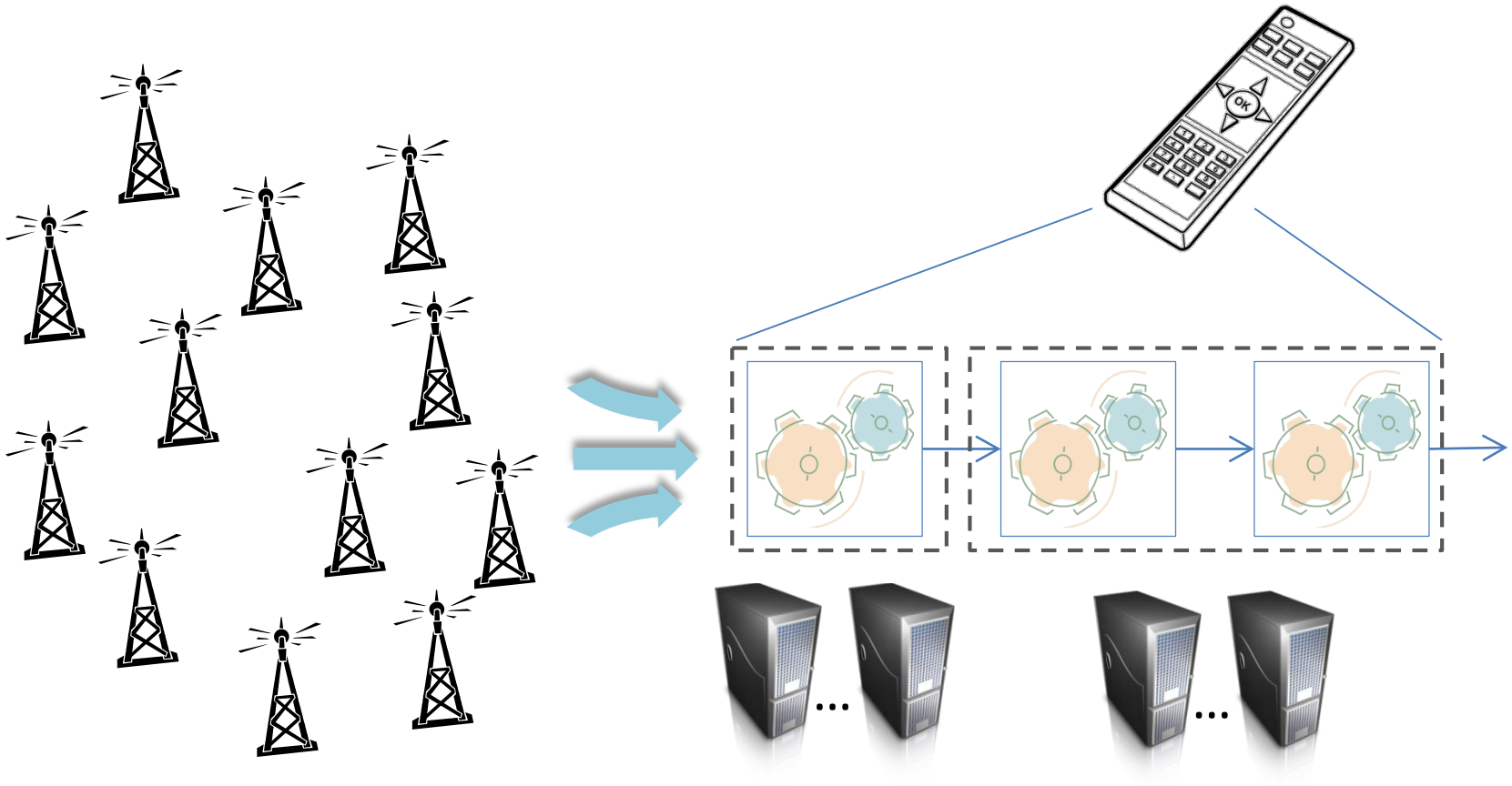
3 Fault Tolerance



Contributions - StreamCloud

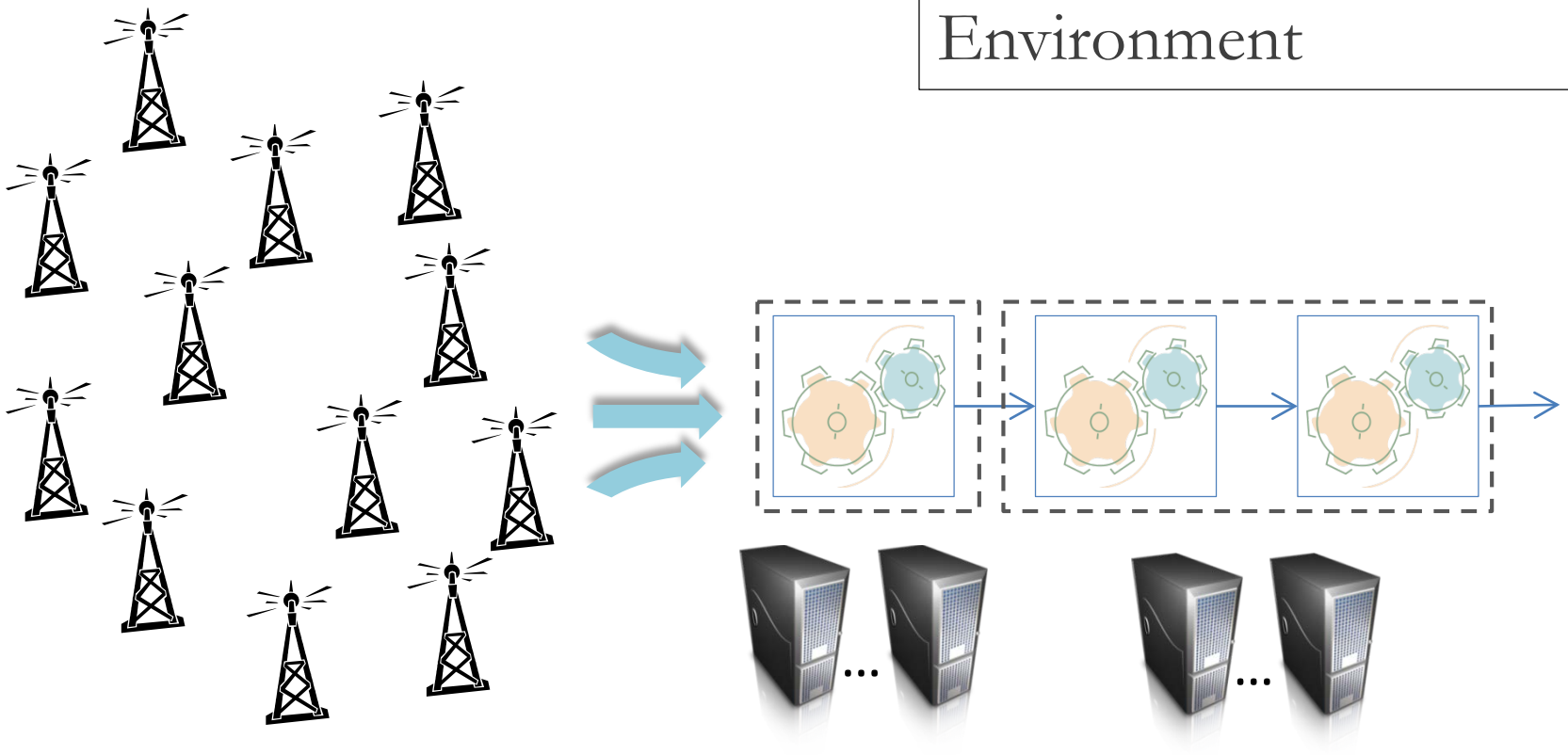


Contributions - StreamCloud



Contributions - StreamCloud

4 Integrated Development Environment



Agenda

- Introduction
  - Motivation
  - System Model
- Parallelization
- Elasticity
- Fault tolerance
- Integrated Development Environment
- Conclusions

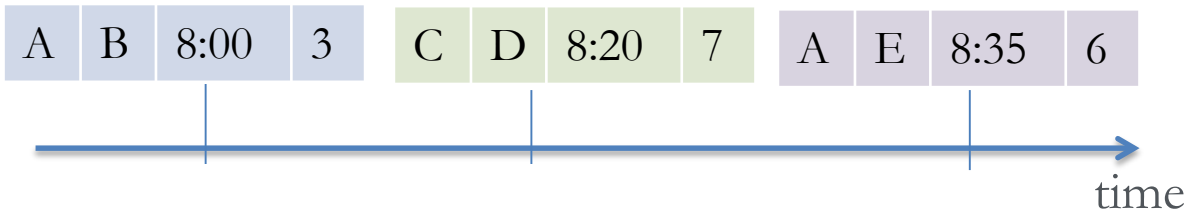
Motivation

- Financial applications, sensor networks monitoring, … require
  - Continuous processing of data streams
  - Real Time fashion
- Store and process is not feasible
  - high-speed networks, nanoseconds to handle a packet
  - ISP router: gigabytes of headers every hour,…
- Data Streaming:
  - In memory
  - Bounded resources
  - Efficient one-pass analysis

System Model

- Data Stream: unbounded sequence of tuples
  - Example: Call Description Record (CDR)

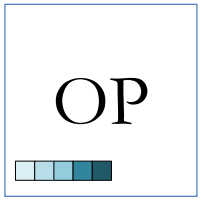
Field	Field
Caller	text
Callee	text
Time (secs)	int
Price (€)	double



System Model

- Operators:

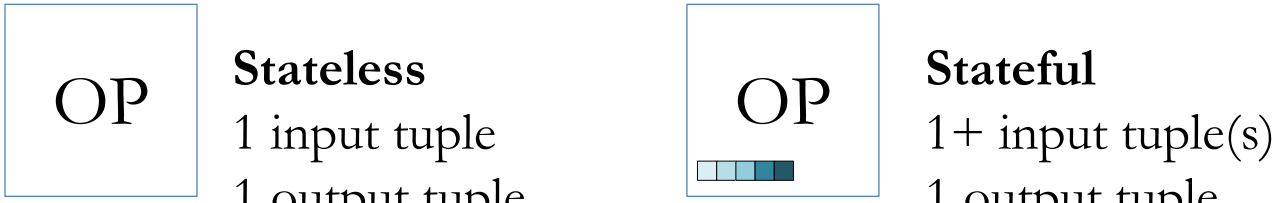
Stateless
1 input tuple
1 output tuple



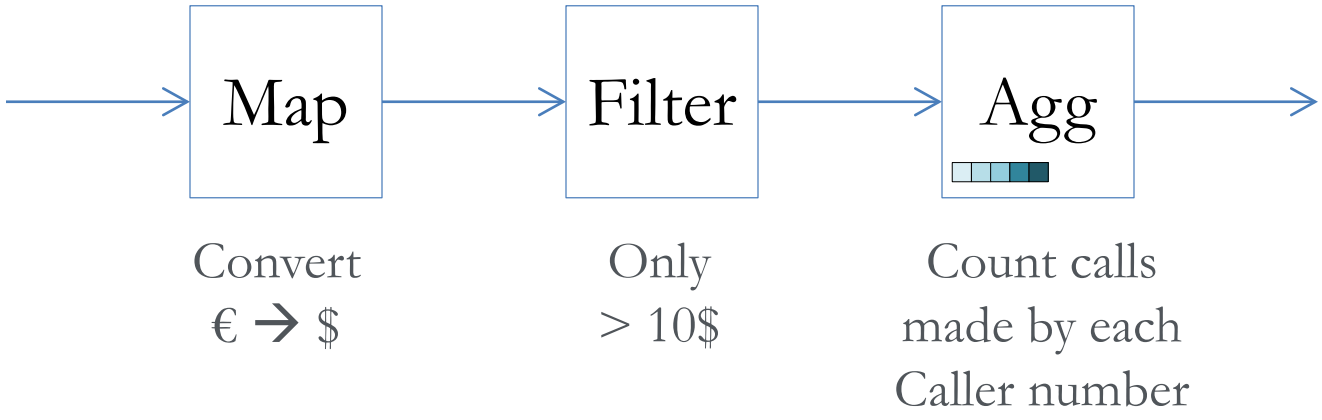
Stateful
1+ input tuple(s)
1 output tuple

System Model

- Operators:

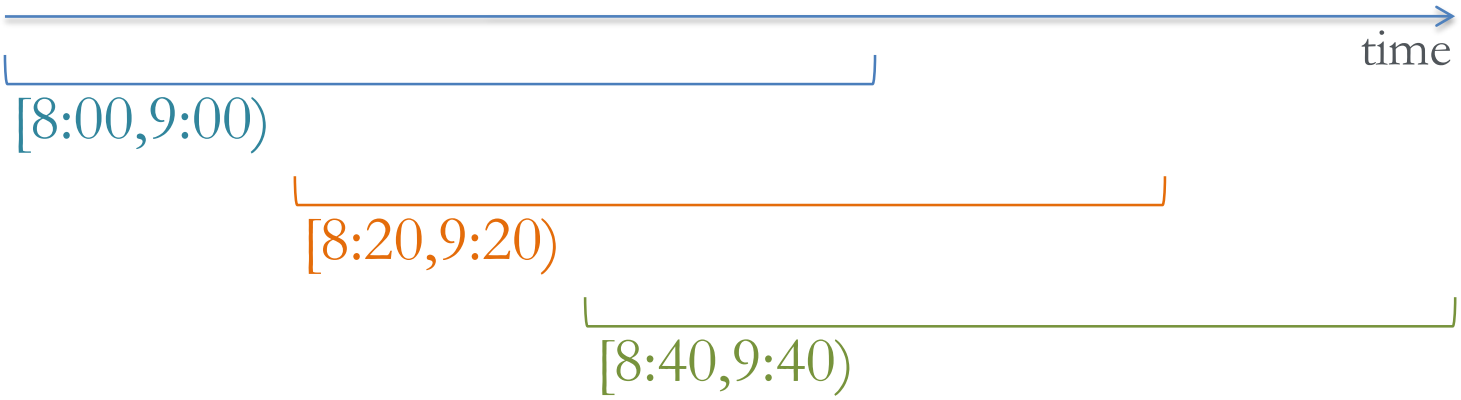


- Continuous Query: graph operators/streams



System Model

- Infinite sequence of tuples / bounded memory
  → windows
- Example: 1 hour windows



System Model

- Infinite sequence of tuples / bounded memory
  → windows
- Example: 1 hour windows

System Model

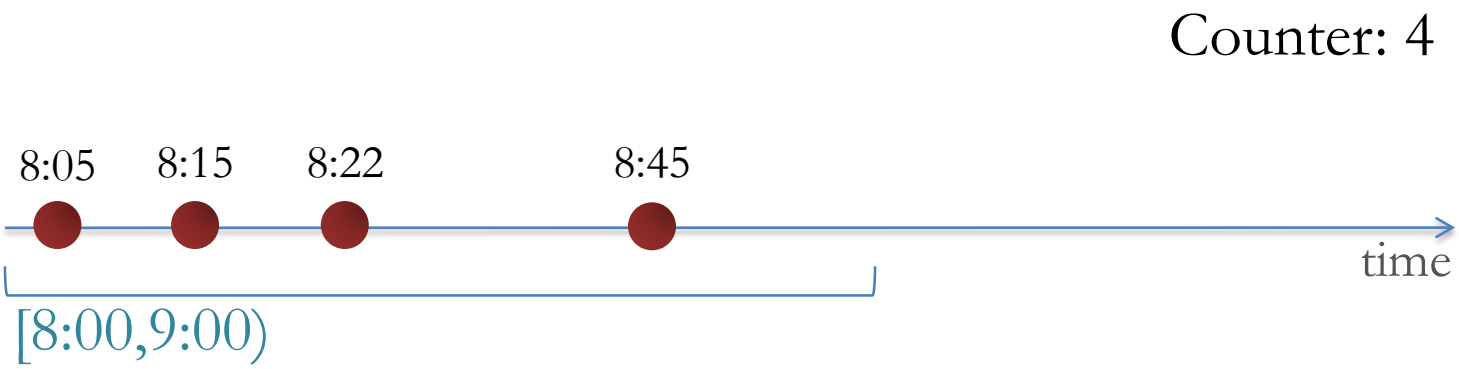
- Infinite sequence of tuples / bounded memory
  → windows
- Example: 1 hour windows

System Model

- Infinite sequence of tuples / bounded memory
  → windows
- Example: 1 hour windows

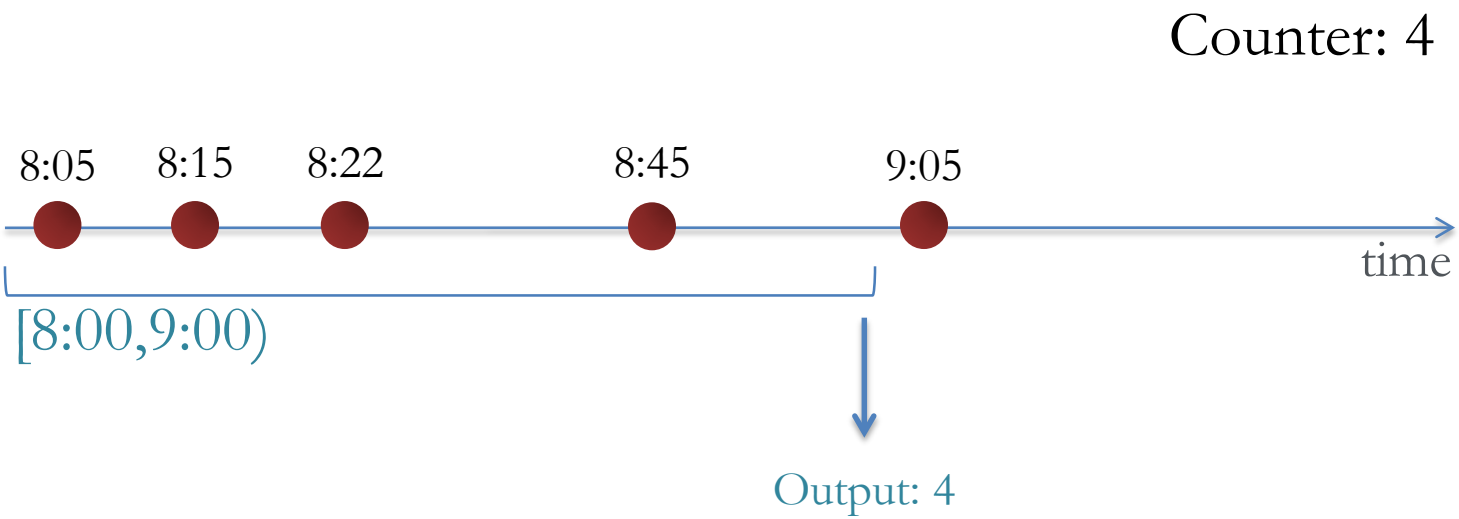
System Model

- Infinite sequence of tuples / bounded memory
  → windows
- Example: 1 hour windows



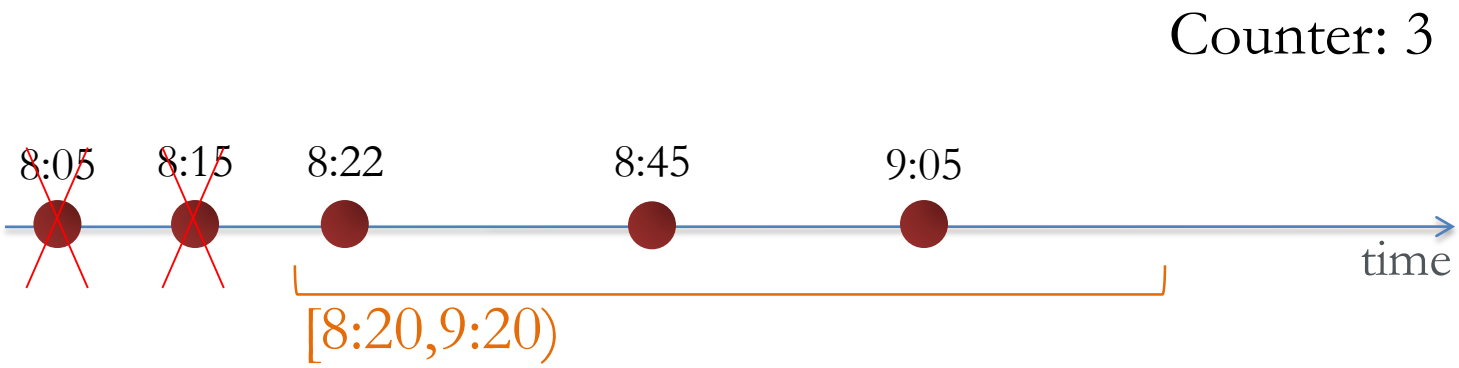
System Model

- Infinite sequence of tuples / bounded memory
  → windows
- Example: 1 hour windows



System Model

- Infinite sequence of tuples / bounded memory
  → windows
- Example: 1 hour windows



Agenda

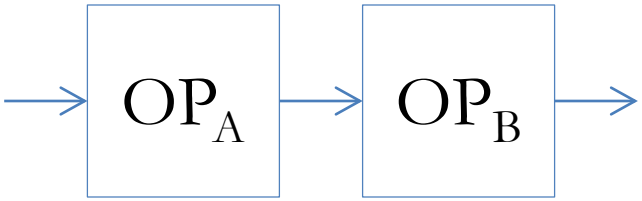
- Introduction
  - Motivation
  - System Model
- Parallelization
- Elasticity
- Fault tolerance
- Integrated Development Environment
- Conclusions

StreamCloud - Parallelization

- Building blocks:
  - Parallelization of data streaming operators
  - Parallelization and Distribution strategy

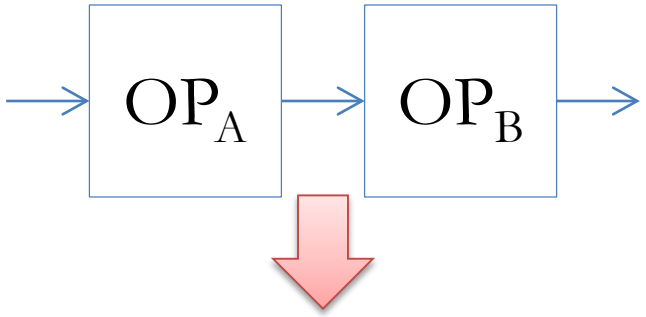
StreamCloud - Parallelization

- General approach



StreamCloud - Parallelization

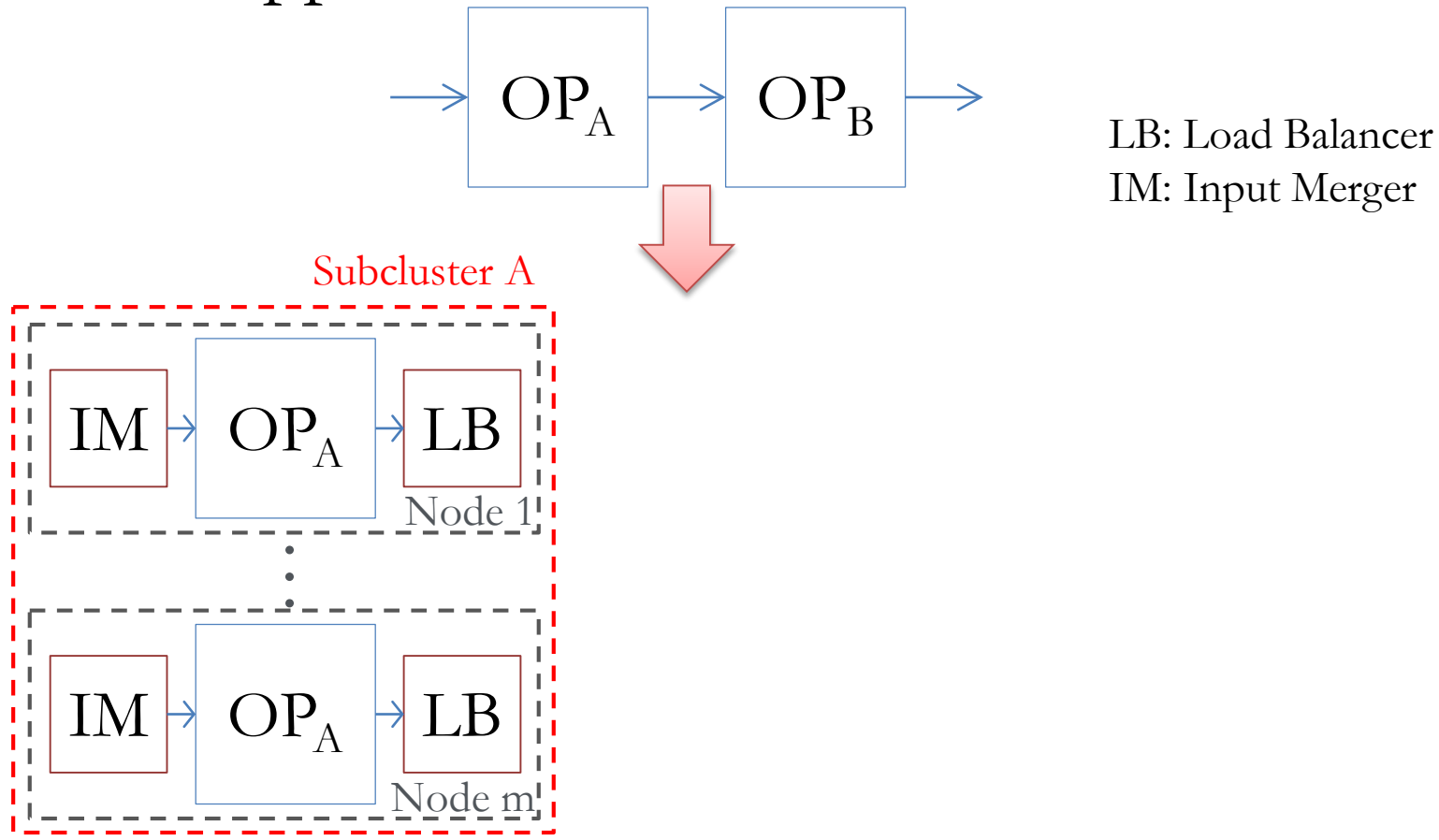
- General approach



LB: Load Balancer
IM: Input Merger

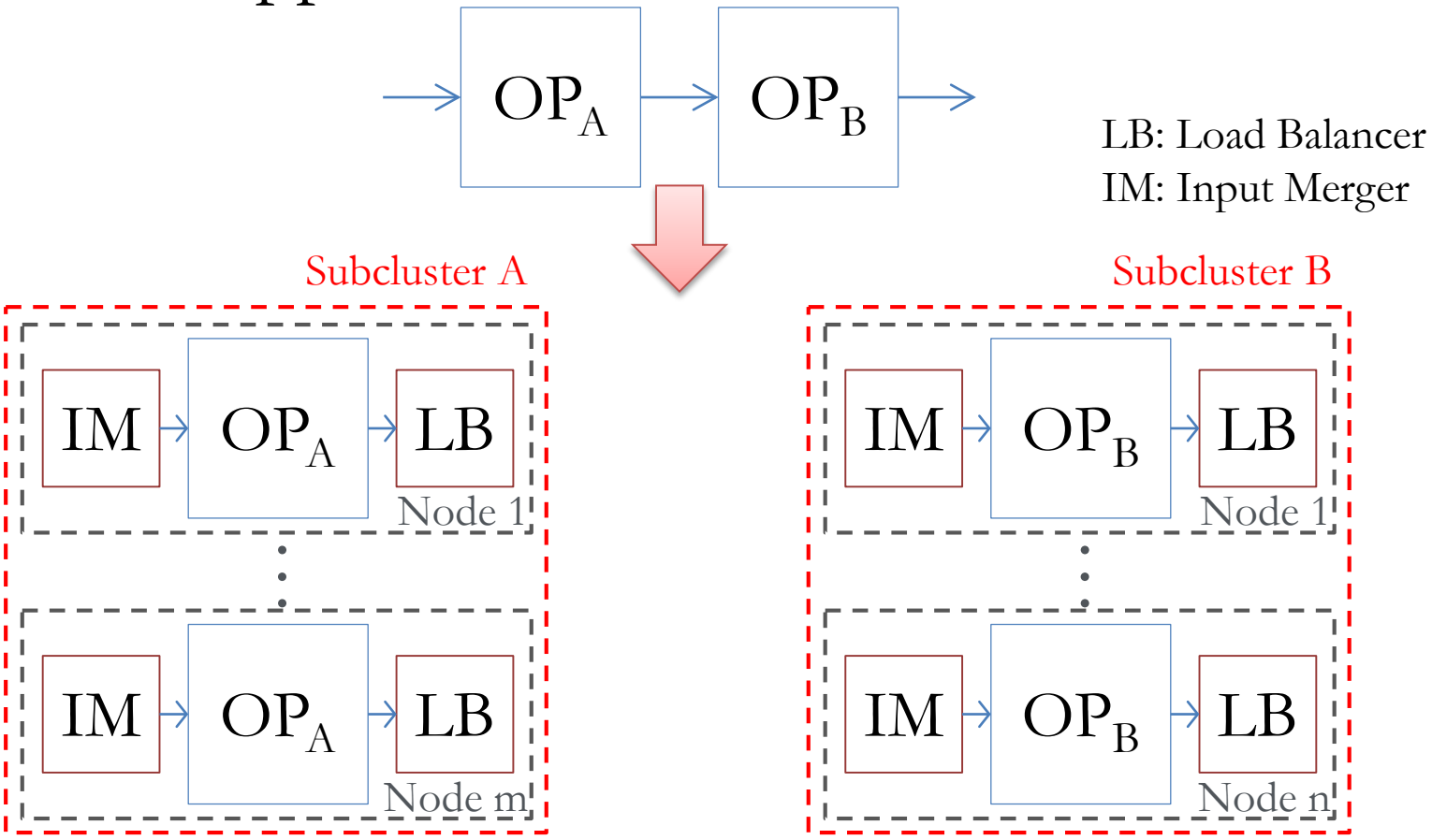
StreamCloud - Parallelization

- General approach



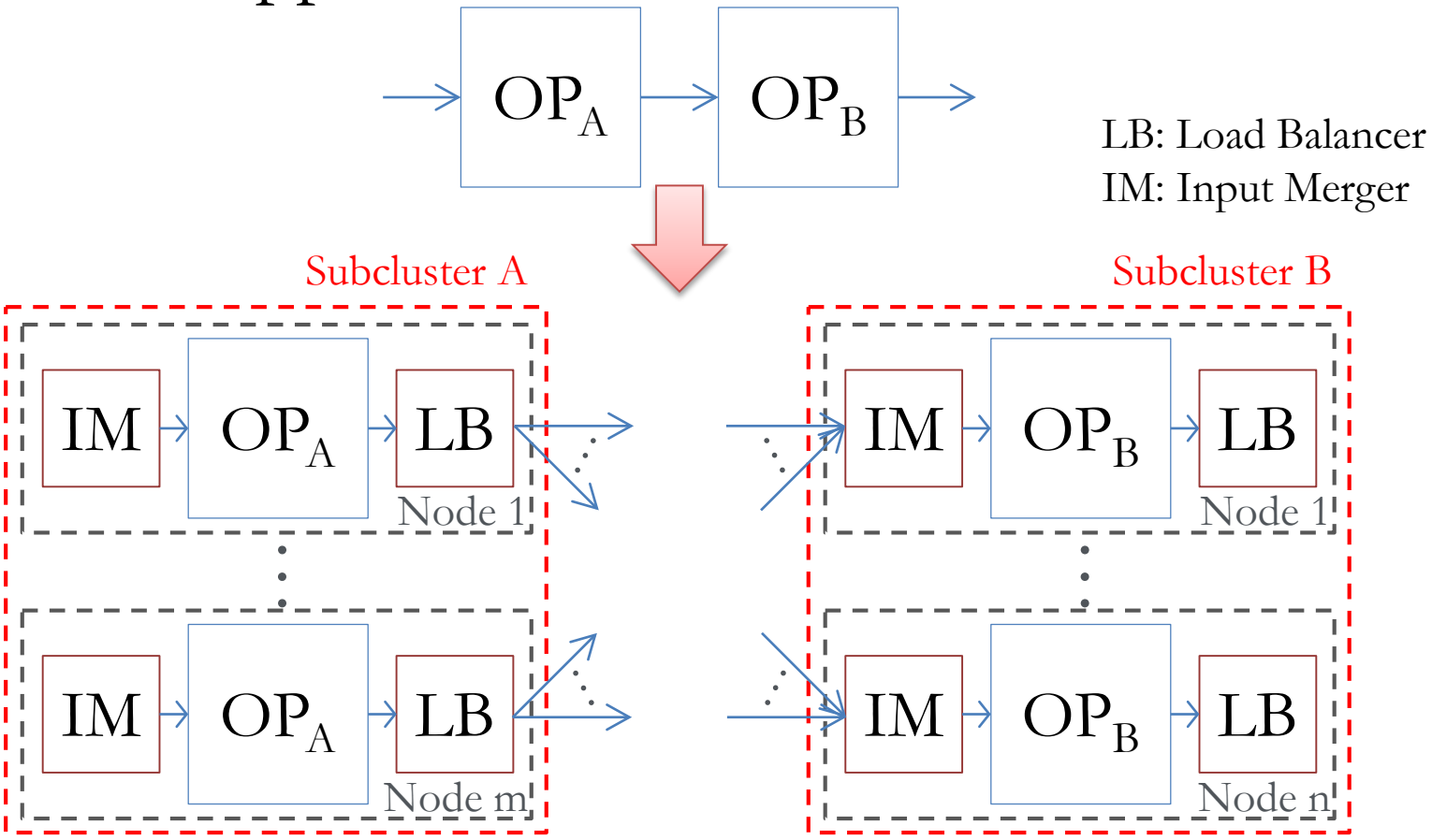
StreamCloud - Parallelization

- General approach



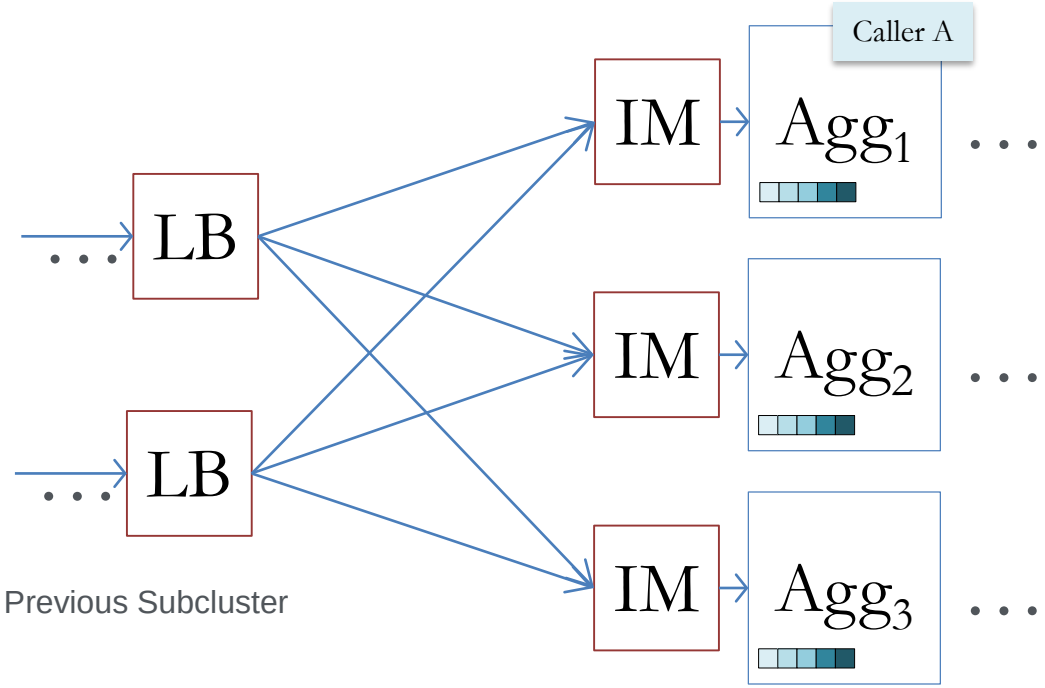
StreamCloud - Parallelization

- General approach



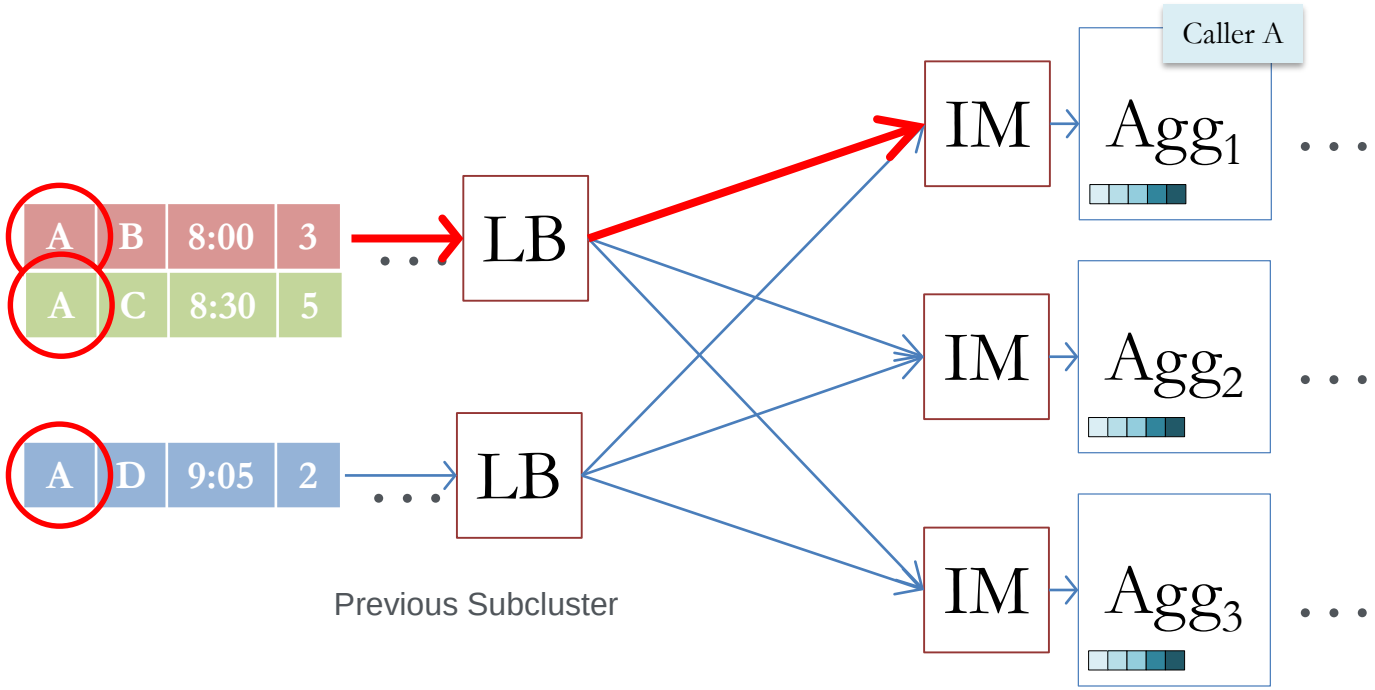
StreamCloud - Parallelization

- Stateful operators: Semantic awareness
  - Aggregate: count within last hour, group-by caller number



StreamCloud - Parallelization

- Stateful operators: Semantic awareness
  - Aggregate: count within last hour, group-by caller number

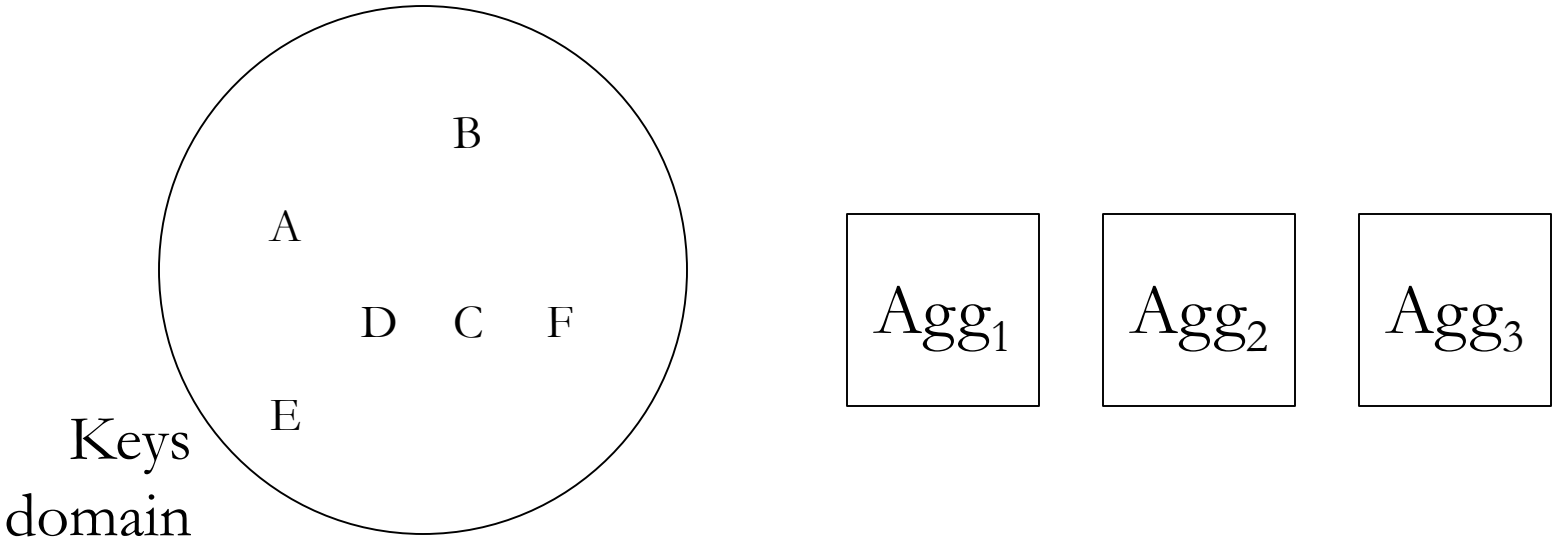


StreamCloud - Parallelization

- Depending on the stateful operator semantic:
  - Partition input stream into **buckets**
  - Each bucket is processed by 1 node
- # buckets >> # nodes

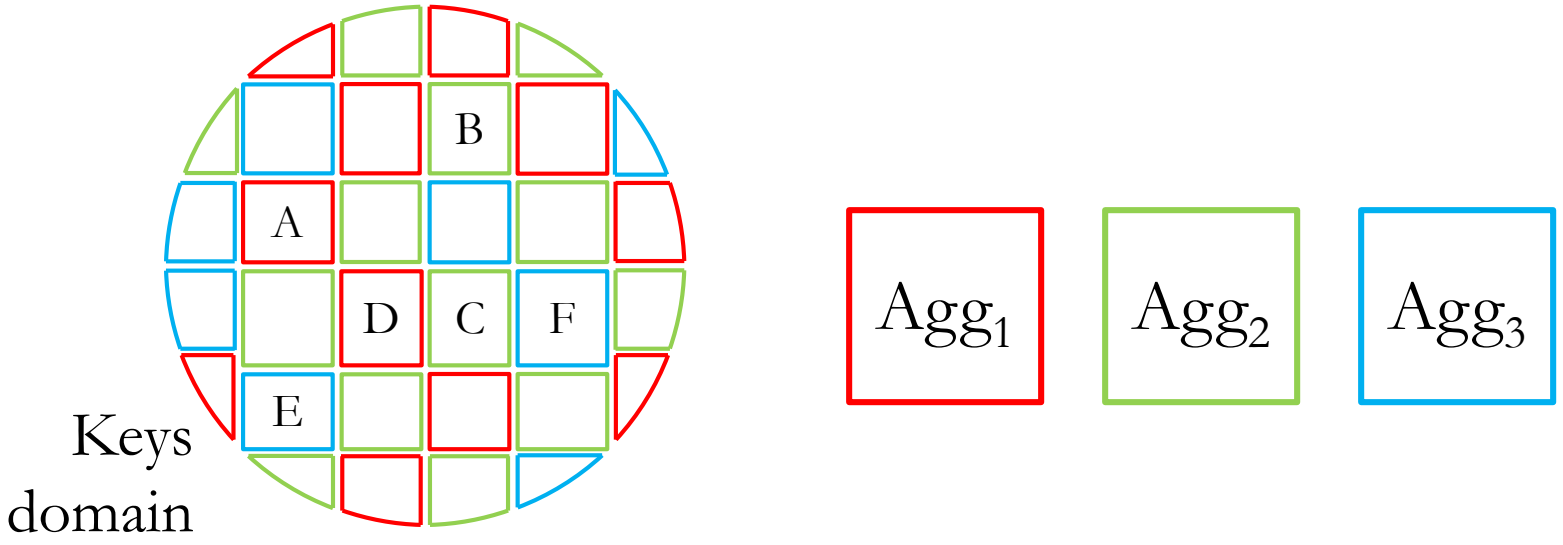
StreamCloud - Parallelization

- Depending on the stateful operator semantic:
  - Partition input stream into **buckets**
  - Each bucket is processed by 1 node
- # buckets >> # nodes



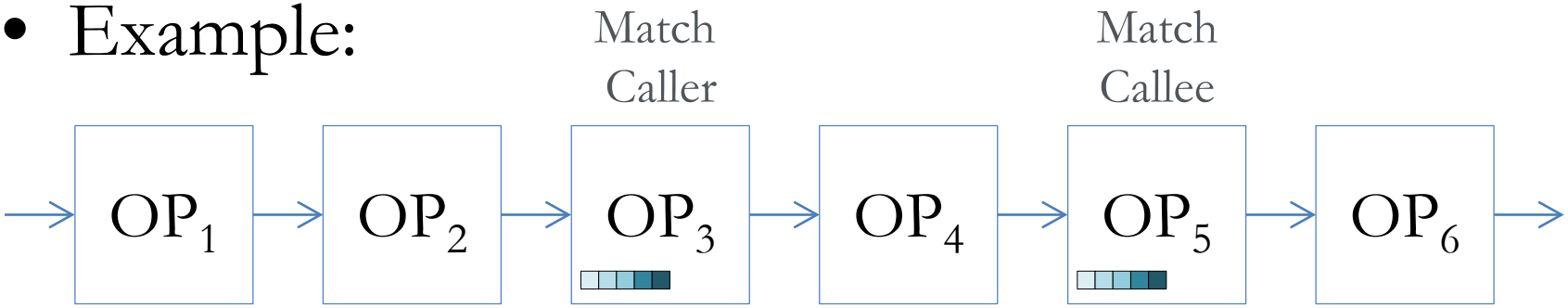
StreamCloud - Parallelization

- Depending on the stateful operator semantic:
  - Partition input stream into **buckets**
  - Each bucket is processed by 1 node
- # buckets >> # nodes



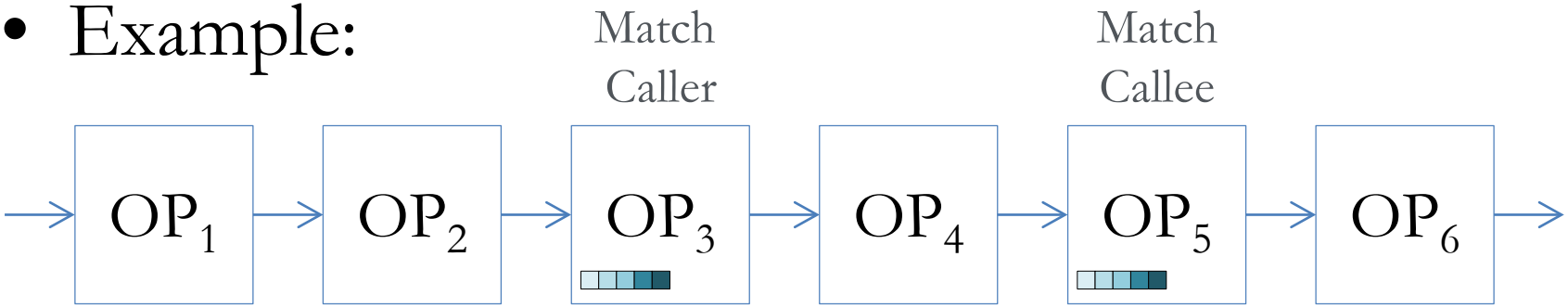
StreamCloud - Parallelization

- Example:



StreamCloud - Parallelization

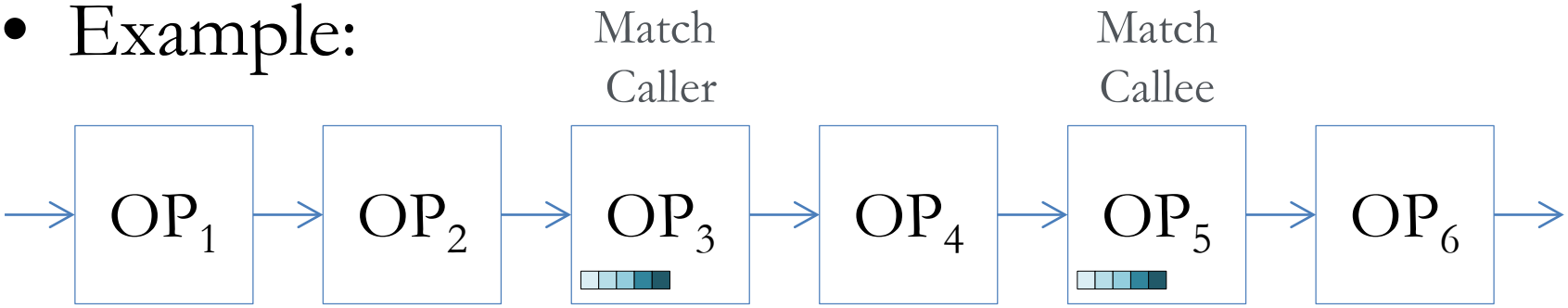
- Example:



  - Cluster composed by 90 nodes

StreamCloud - Parallelization

- Example:

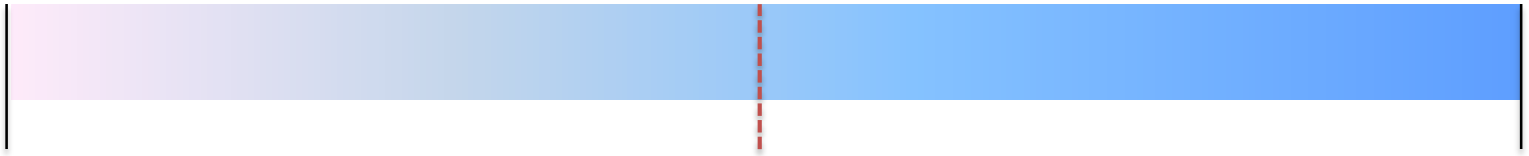


  - Cluster composed by 90 nodes

How to parallelize/distribute the query to maximize throughput?

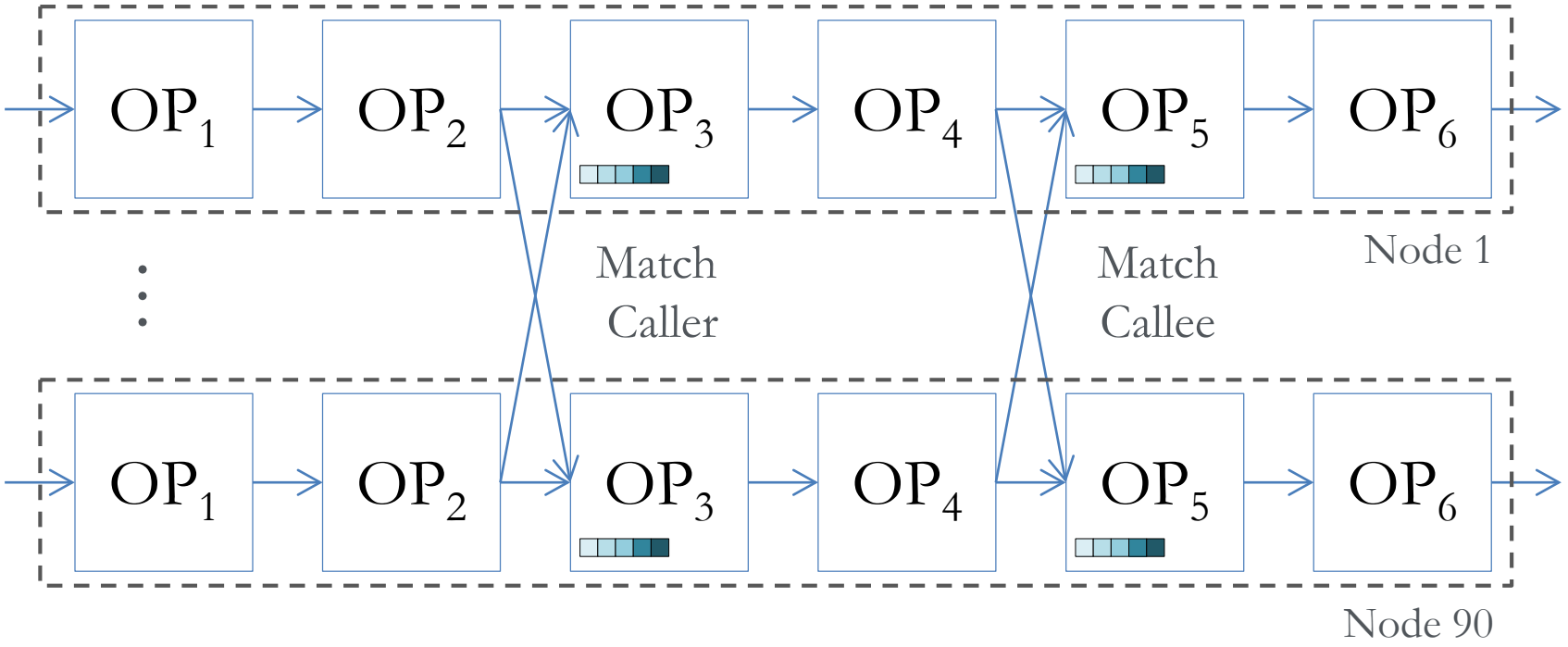
StreamCloud - Parallelization

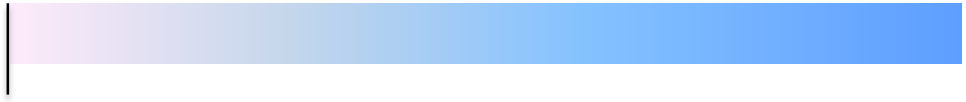
- Parallelization Strategies



Full query at all nodes | 1+ operators per node | 1 operator per node

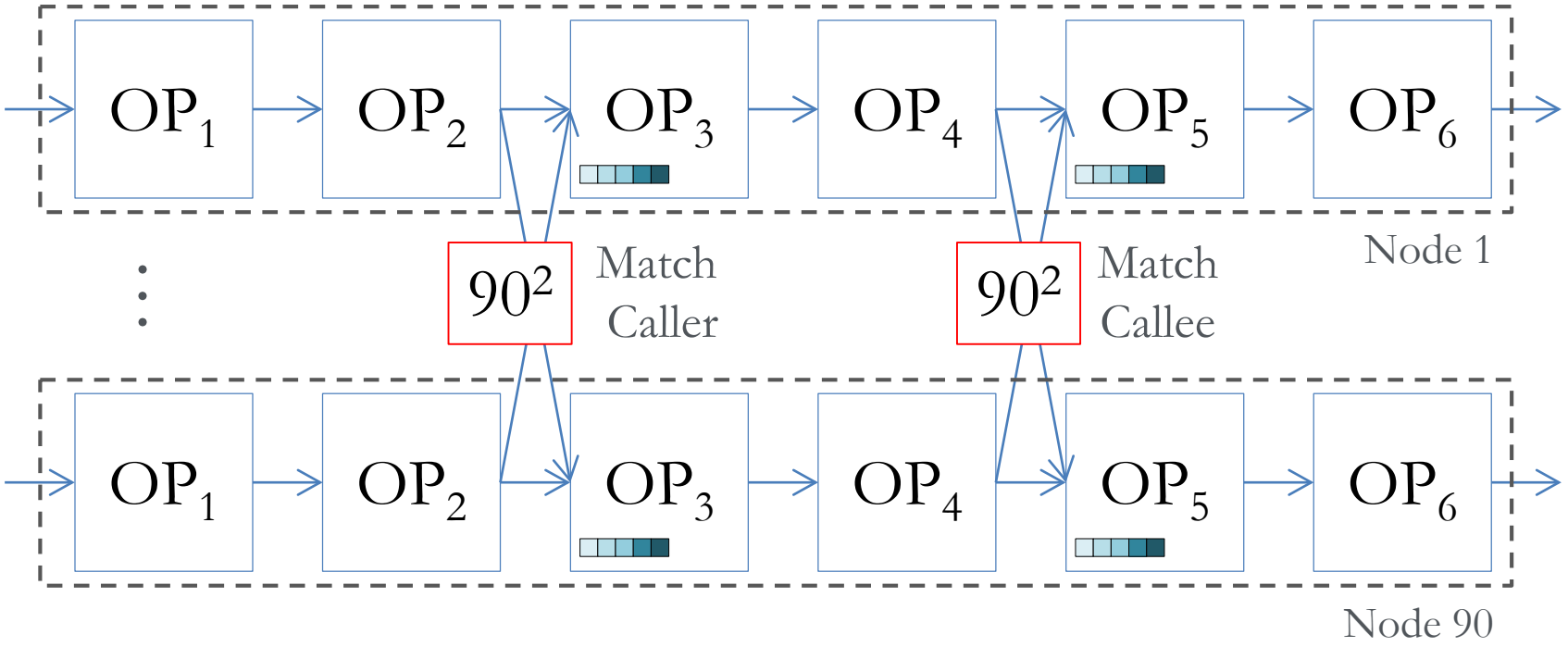
StreamCloud - Parallelization

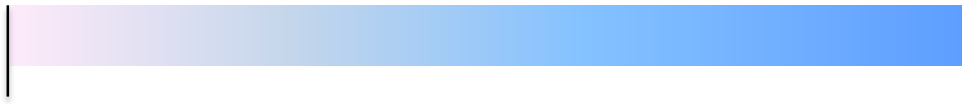




Full query at all nodes

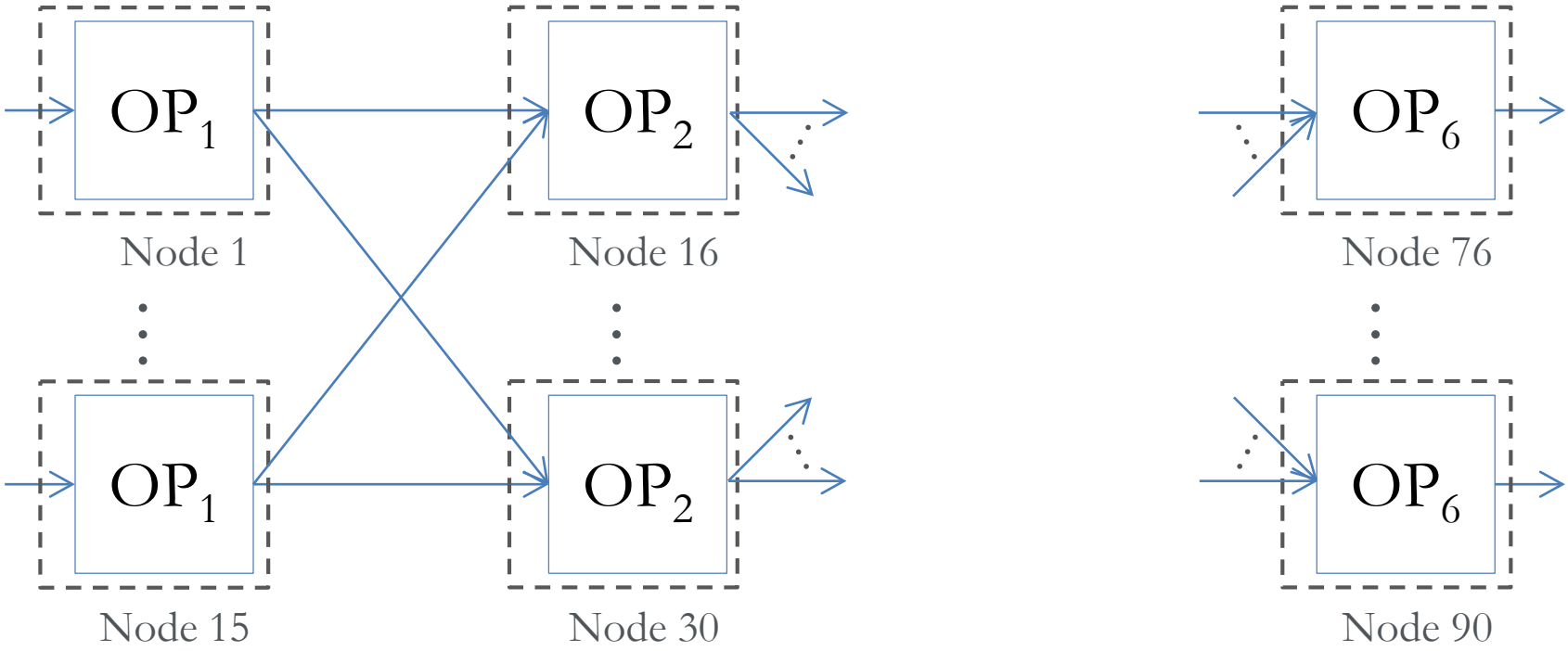
StreamCloud - Parallelization





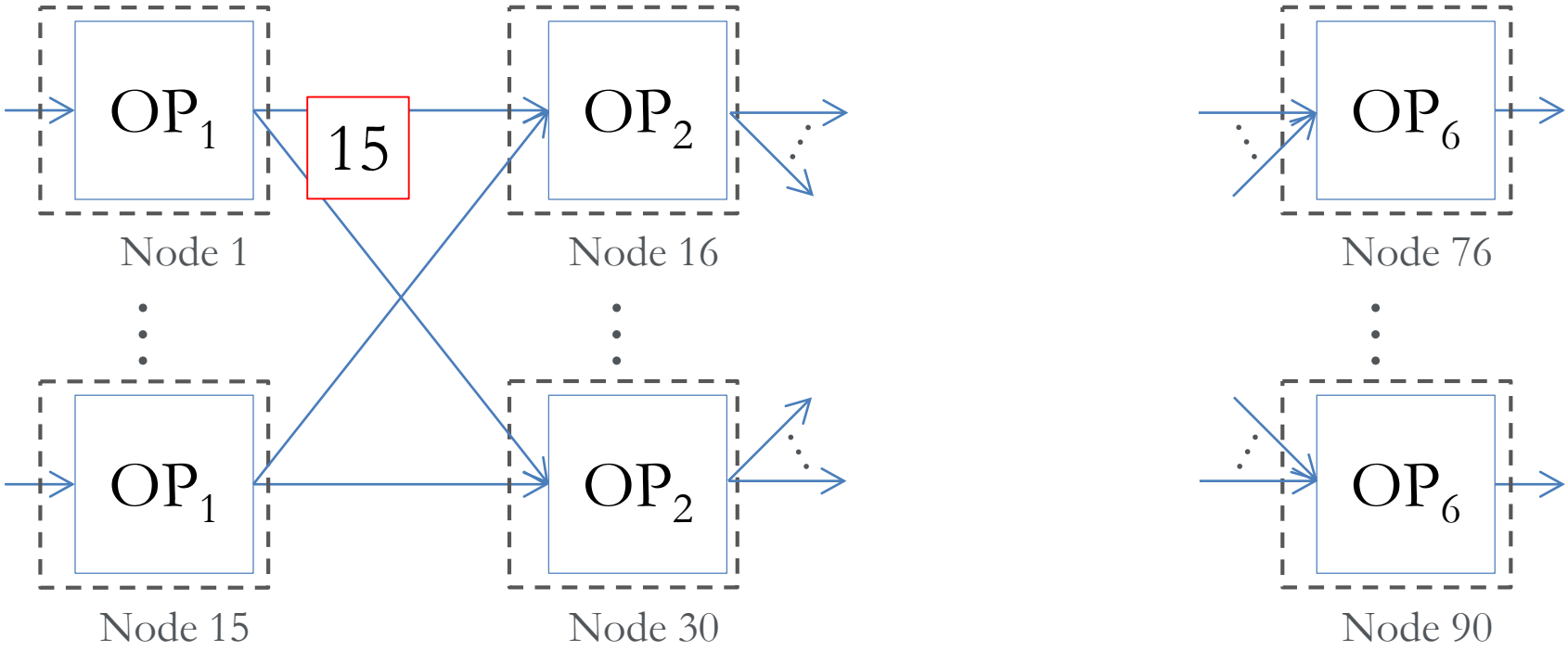
Full query at all nodes

StreamCloud - Parallelization



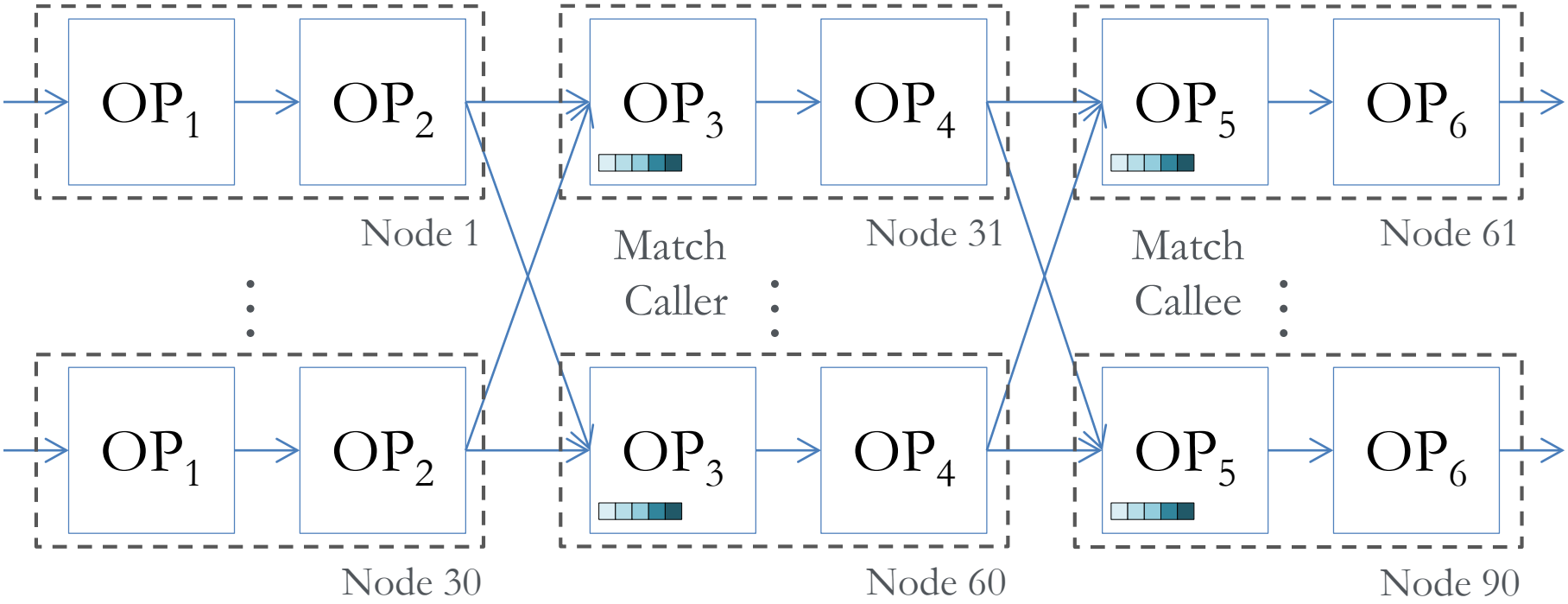
1 operator per node

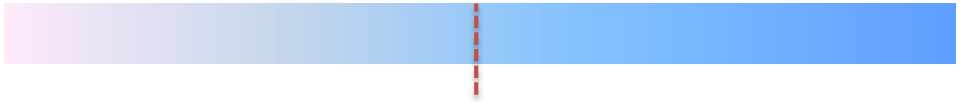
StreamCloud - Parallelization



1 operator per node

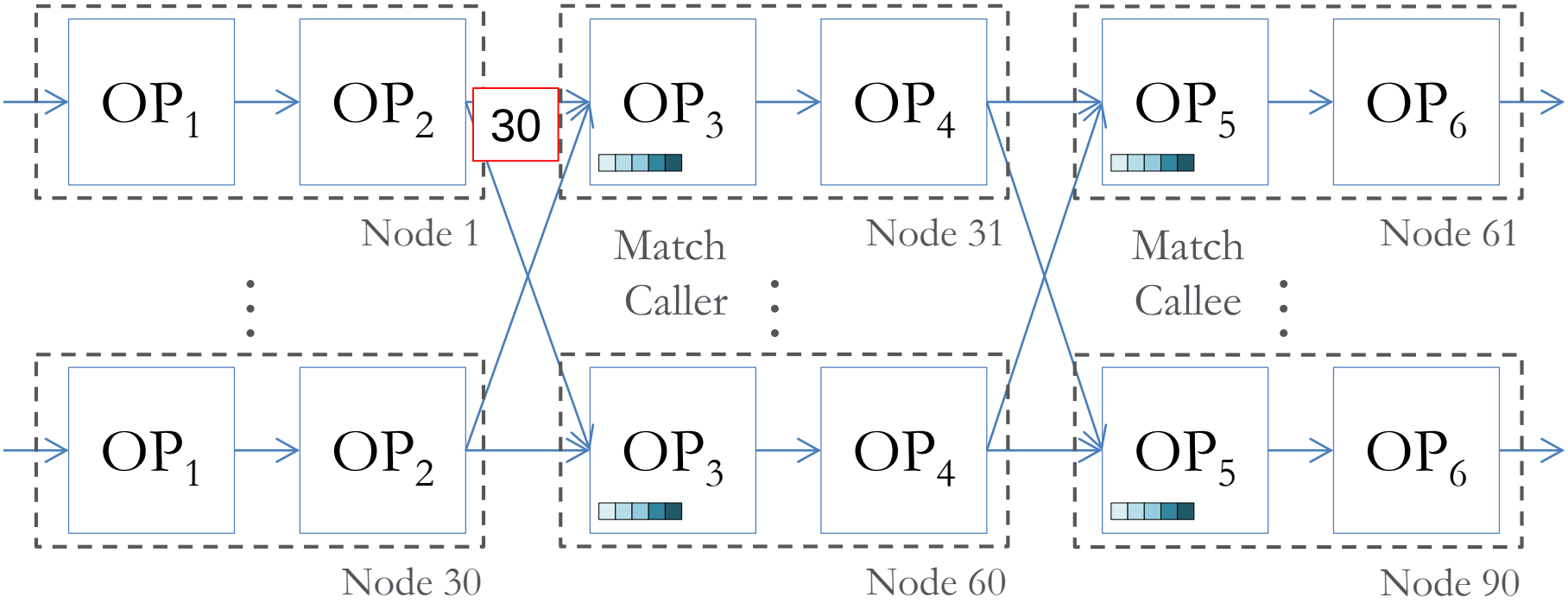
StreamCloud - Parallelization

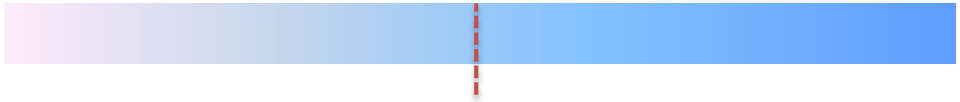




1+ operators per node

StreamCloud - Parallelization





1+ operators per node

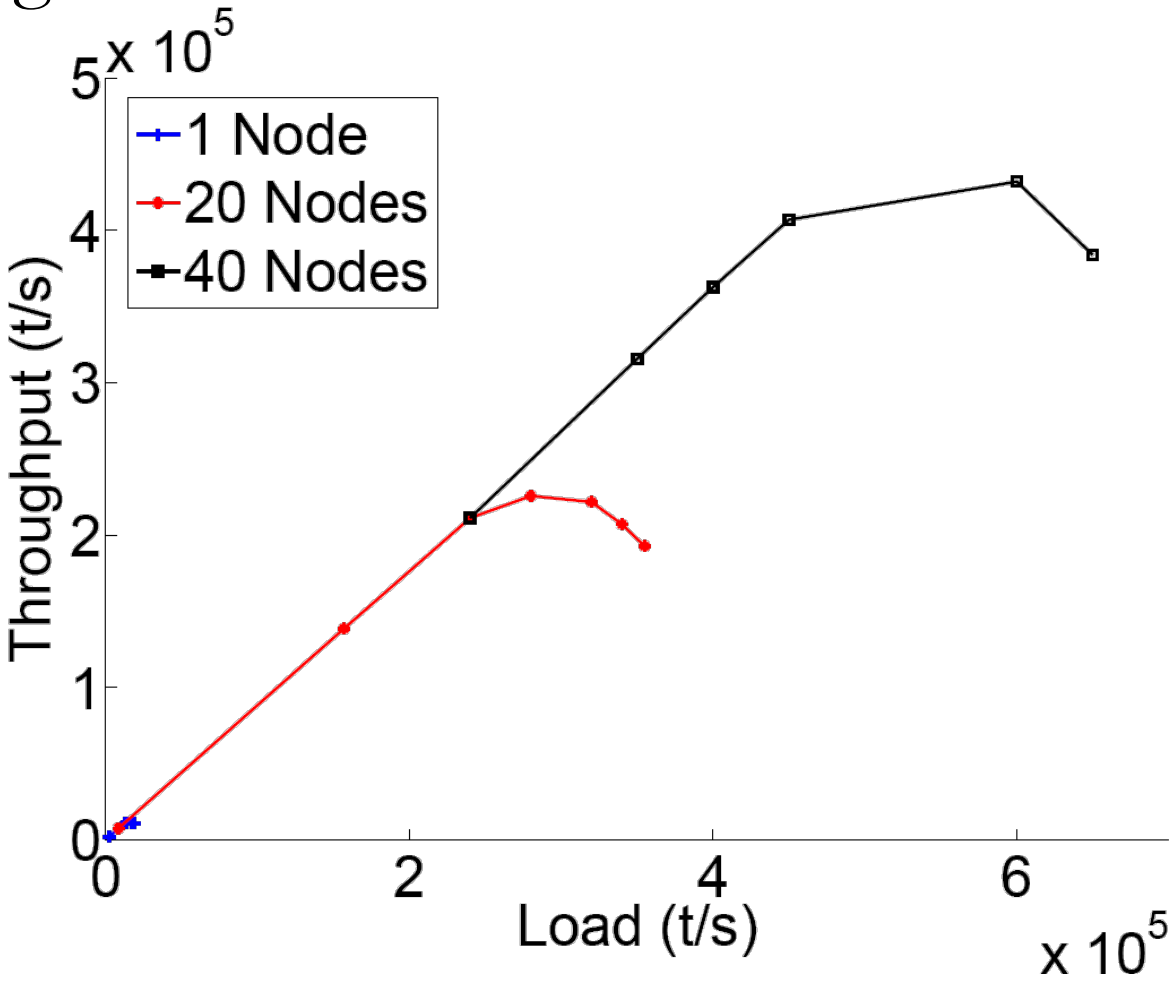
StreamCloud - Parallelization

- Scalability of data streaming operators
- Aggregate operator:

  Compute average call duration, number of calls for each phone number
- Setup:
  - 1 / 20 / 40 nodes

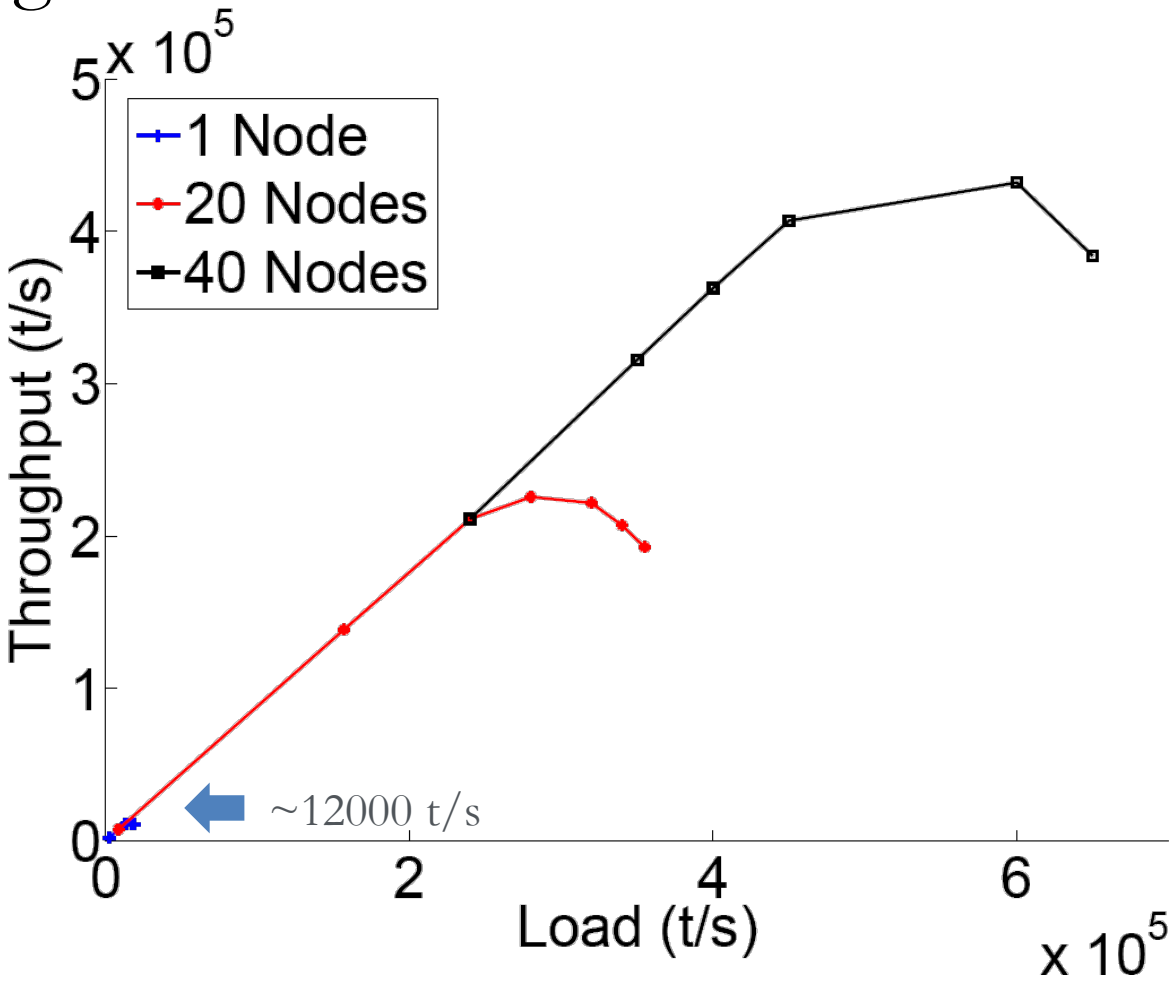
StreamCloud - Parallelization

- Aggregate scale out evaluation



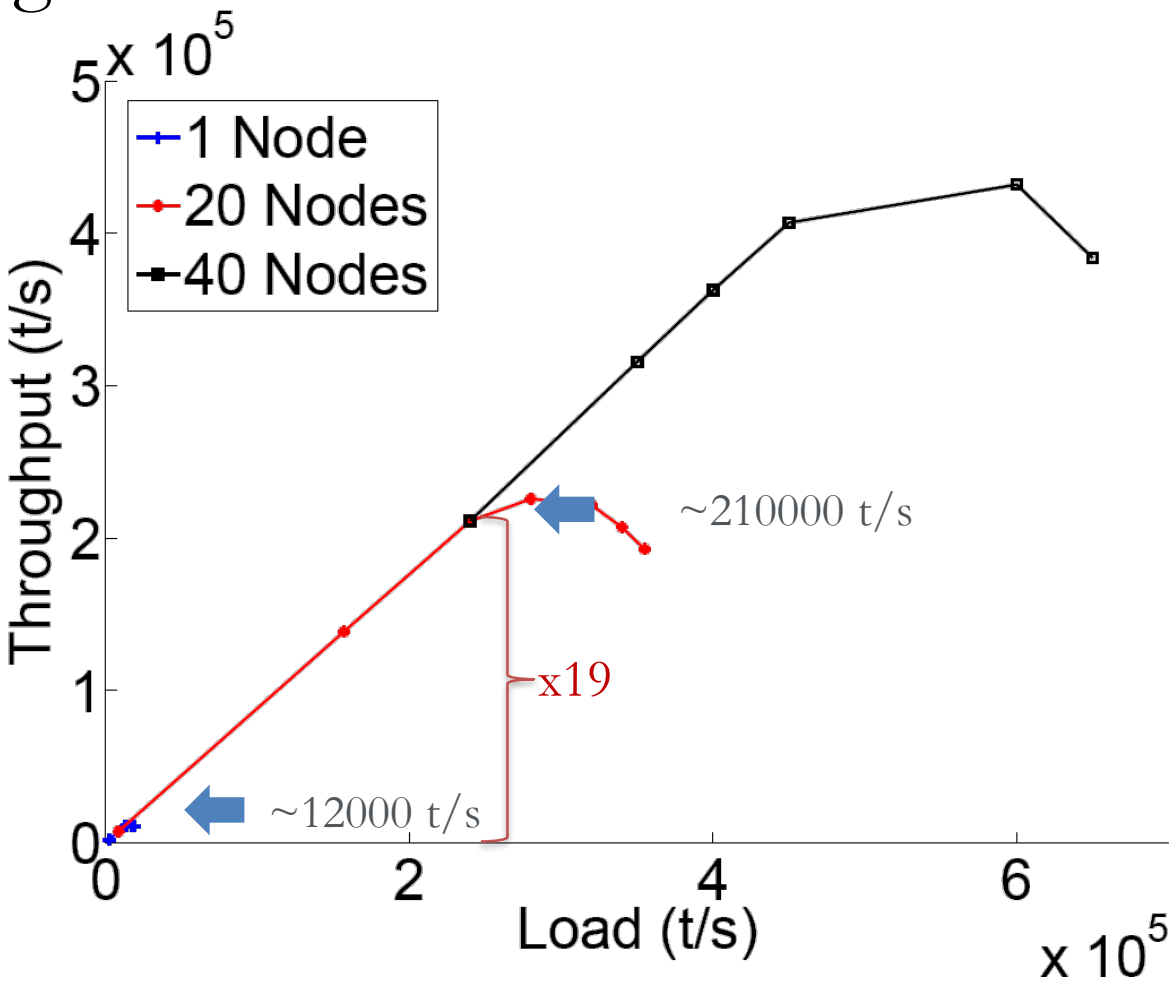
StreamCloud - Parallelization

- Aggregate scale out evaluation



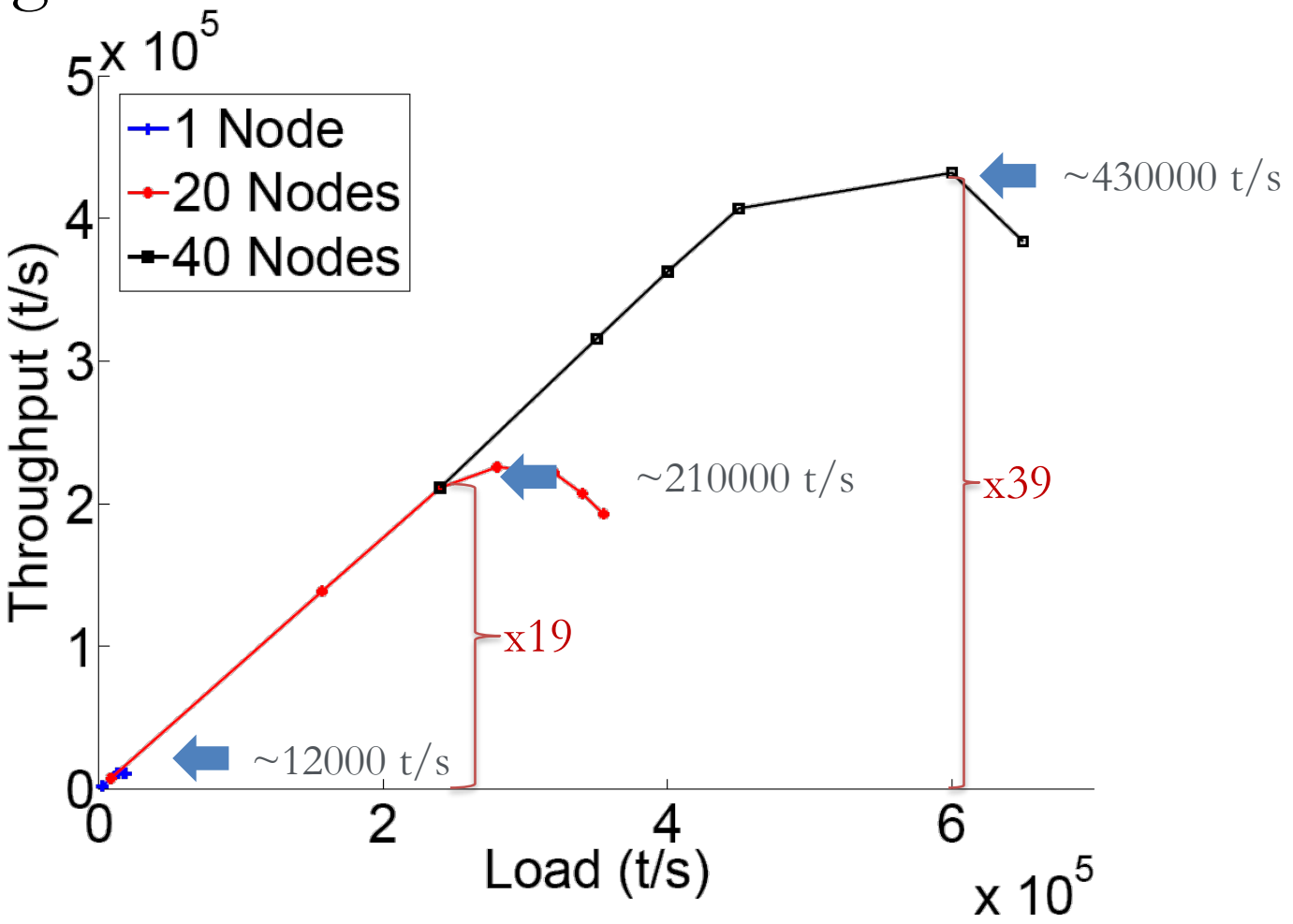
StreamCloud - Parallelization

- Aggregate scale out evaluation



StreamCloud - Parallelization

- Aggregate scale out evaluation



Agenda

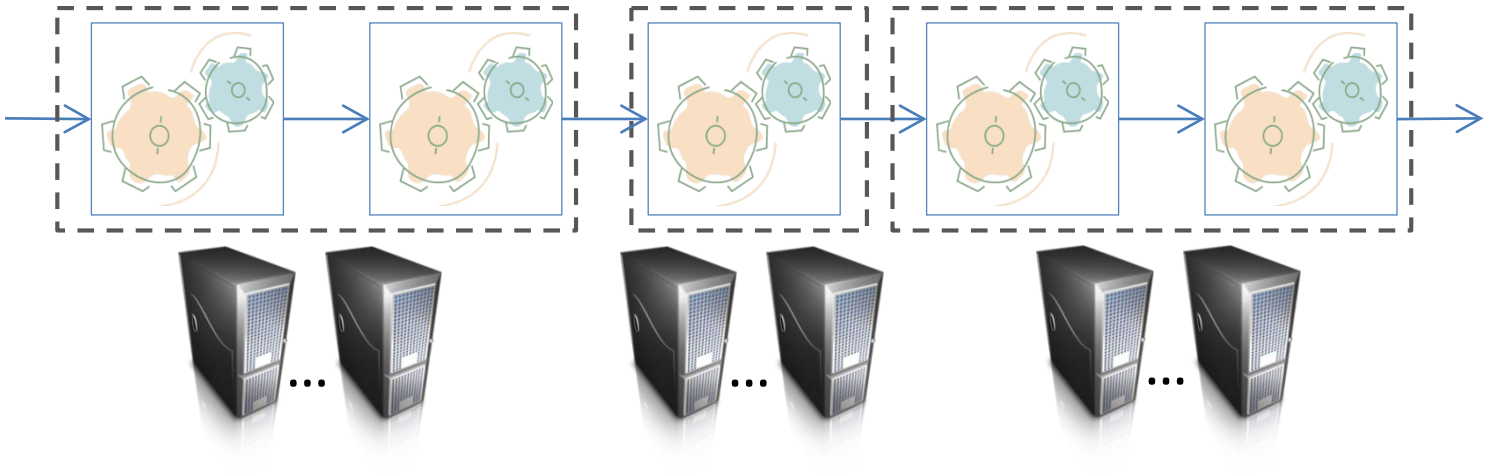
- Introduction
  - Motivation
  - System Model
- Parallelization
- Elasticity
- Fault tolerance
- Integrated Development Environment
- Conclusions

StreamCloud - Elasticity

- Building blocks:
  - Elasticity rules
  - Dynamic Load Balancing combined with provisioning /decommissioning
  - Load-aware provisioning / decommissioning

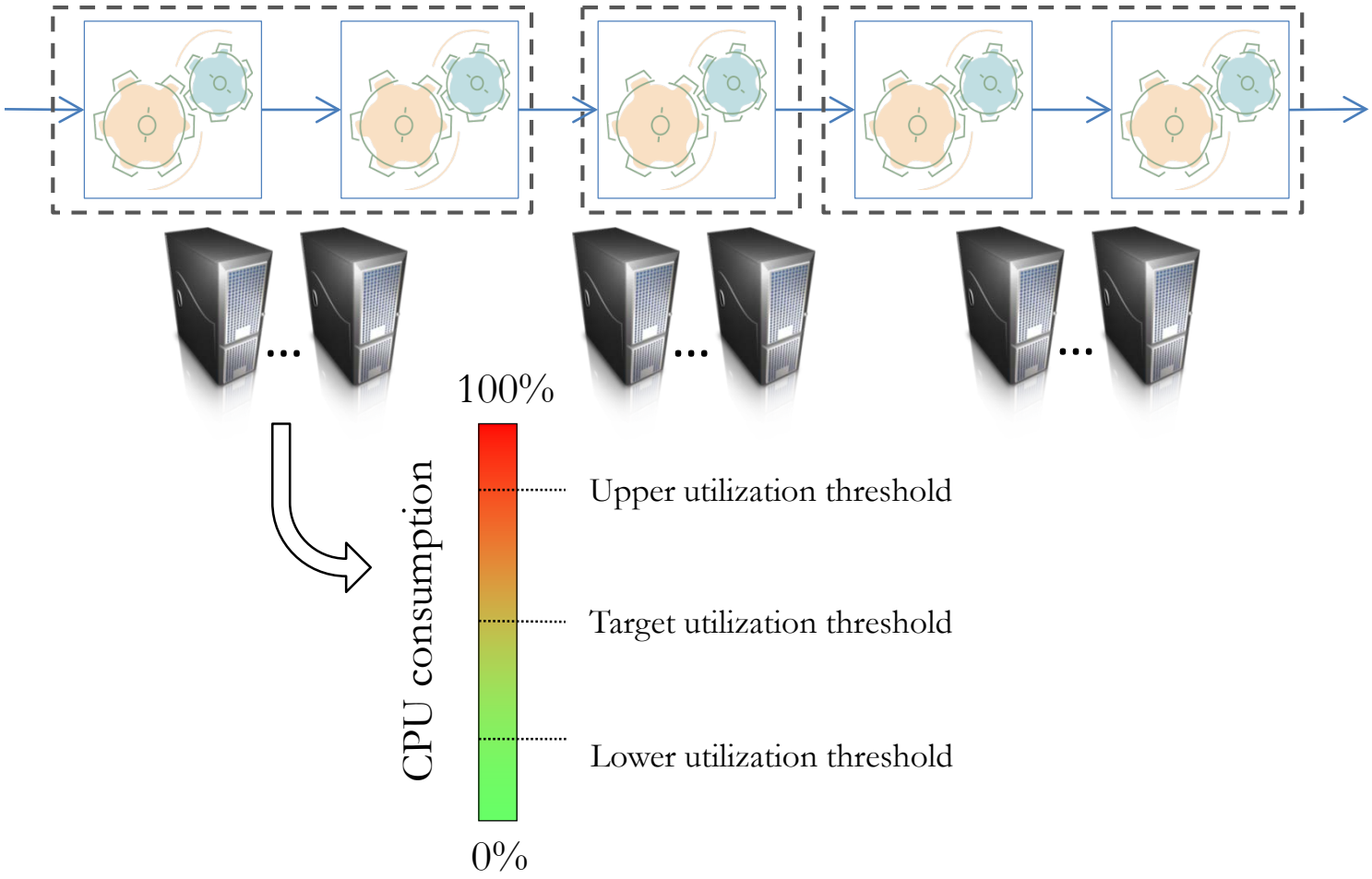
StreamCloud - Elasticity

Elasticity and load balancing actions on per-subcluster basis



StreamCloud - Elasticity

Elasticity and load balancing actions on per-subcluster basis



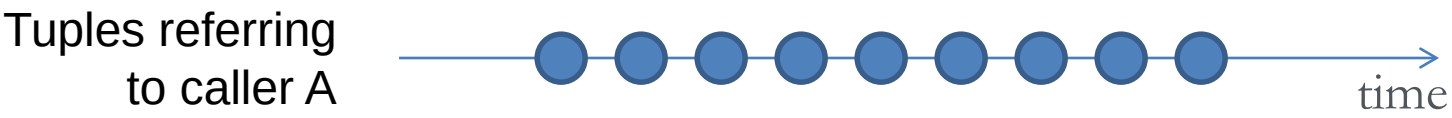
StreamCloud - Elasticity

- Transfer state between nodes
- Load balancing algorithm

- Minimum parallelization unit → bucket
  - Transfer buckets between instances

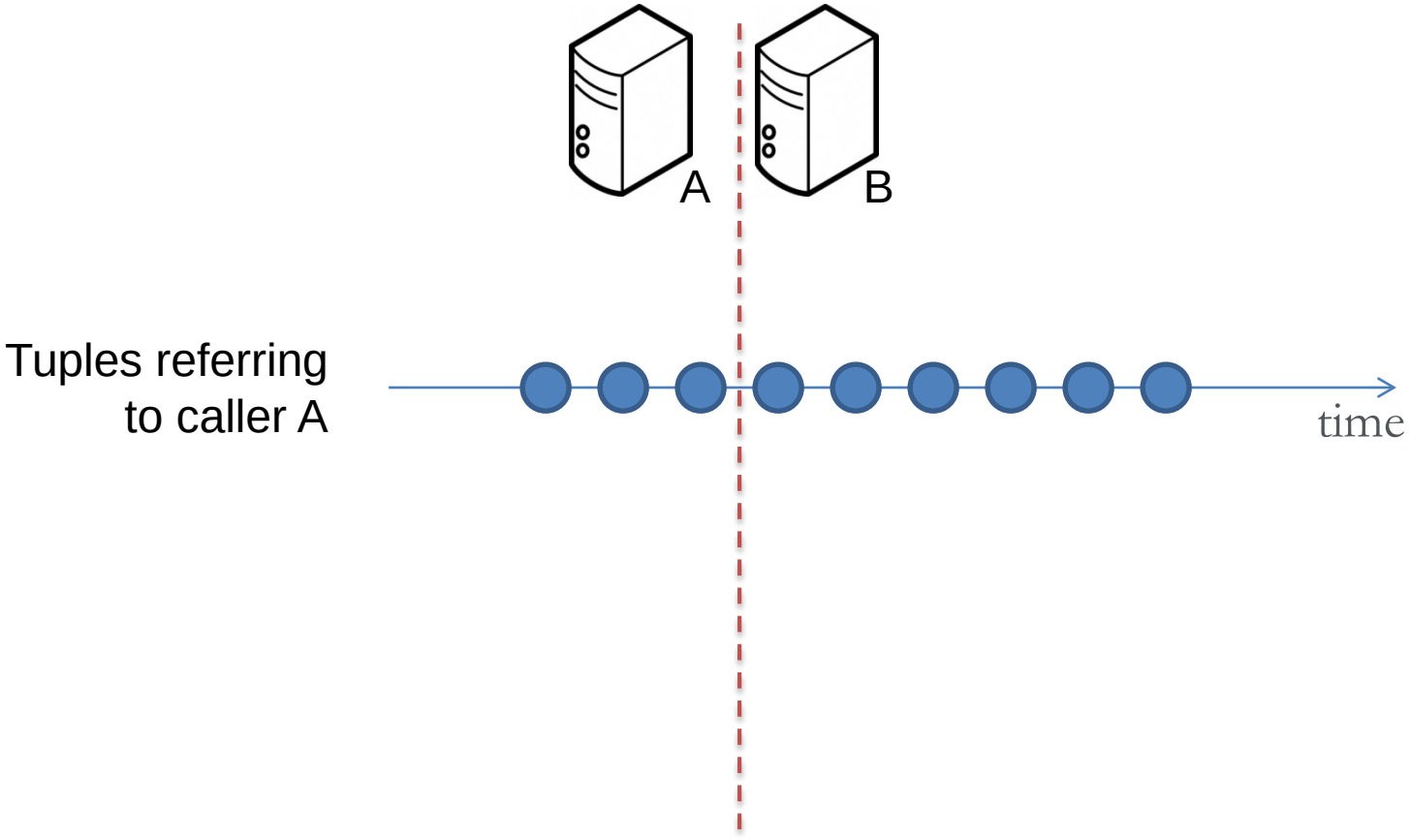
StreamCloud - Elasticity

- State transfer challenging for stateful operators



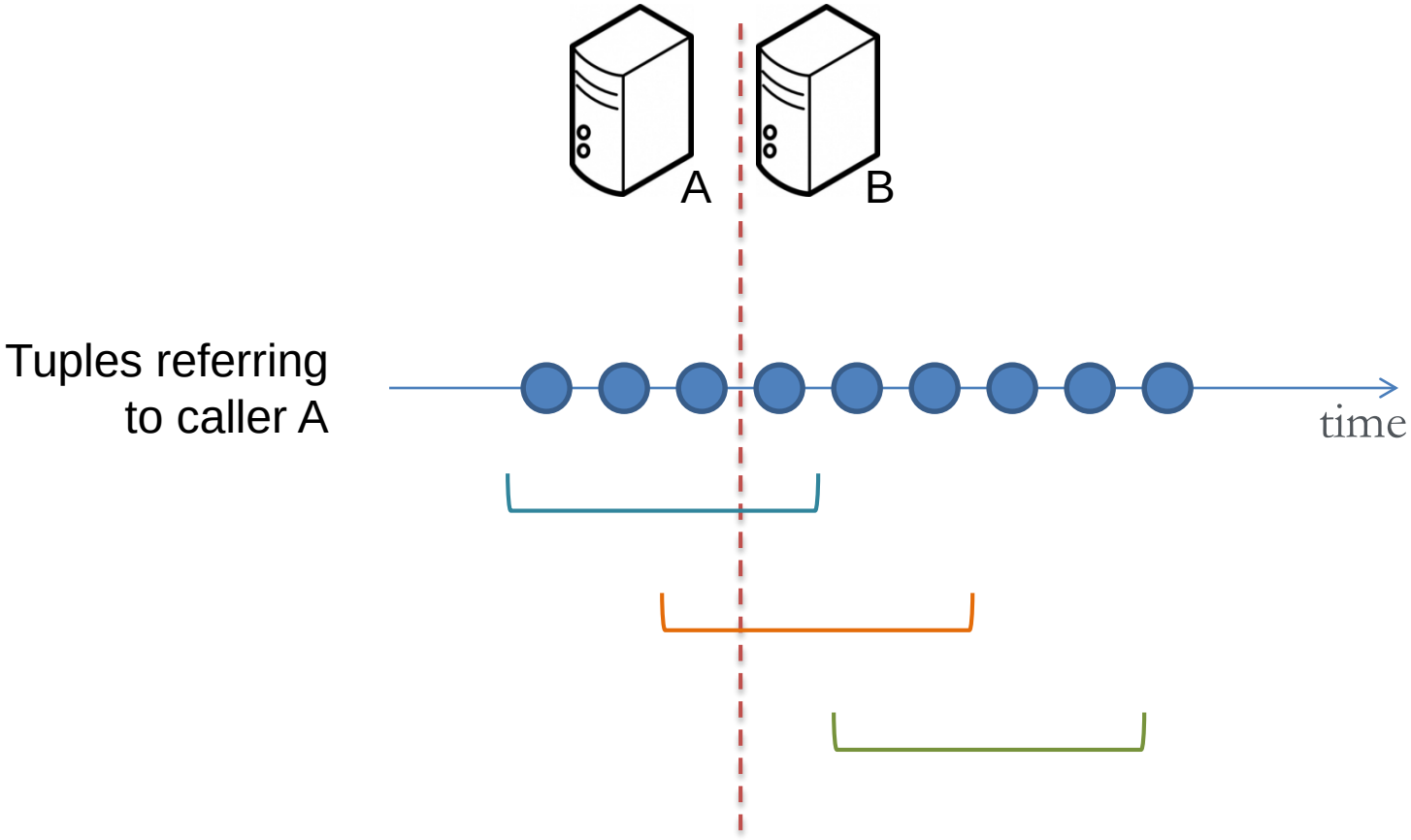
StreamCloud - Elasticity

- State transfer challenging for stateful operators



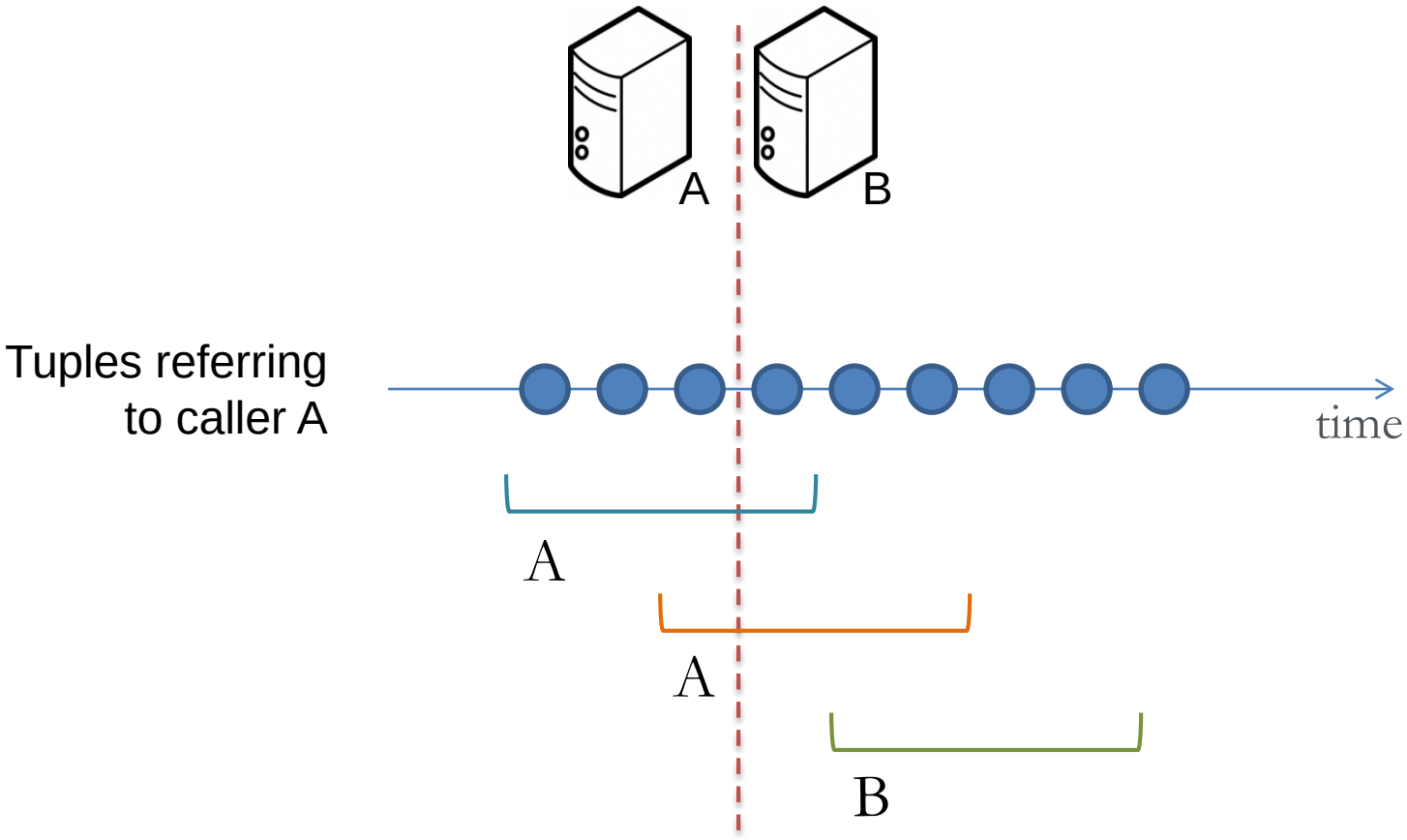
StreamCloud - Elasticity

- State transfer challenging for stateful operators



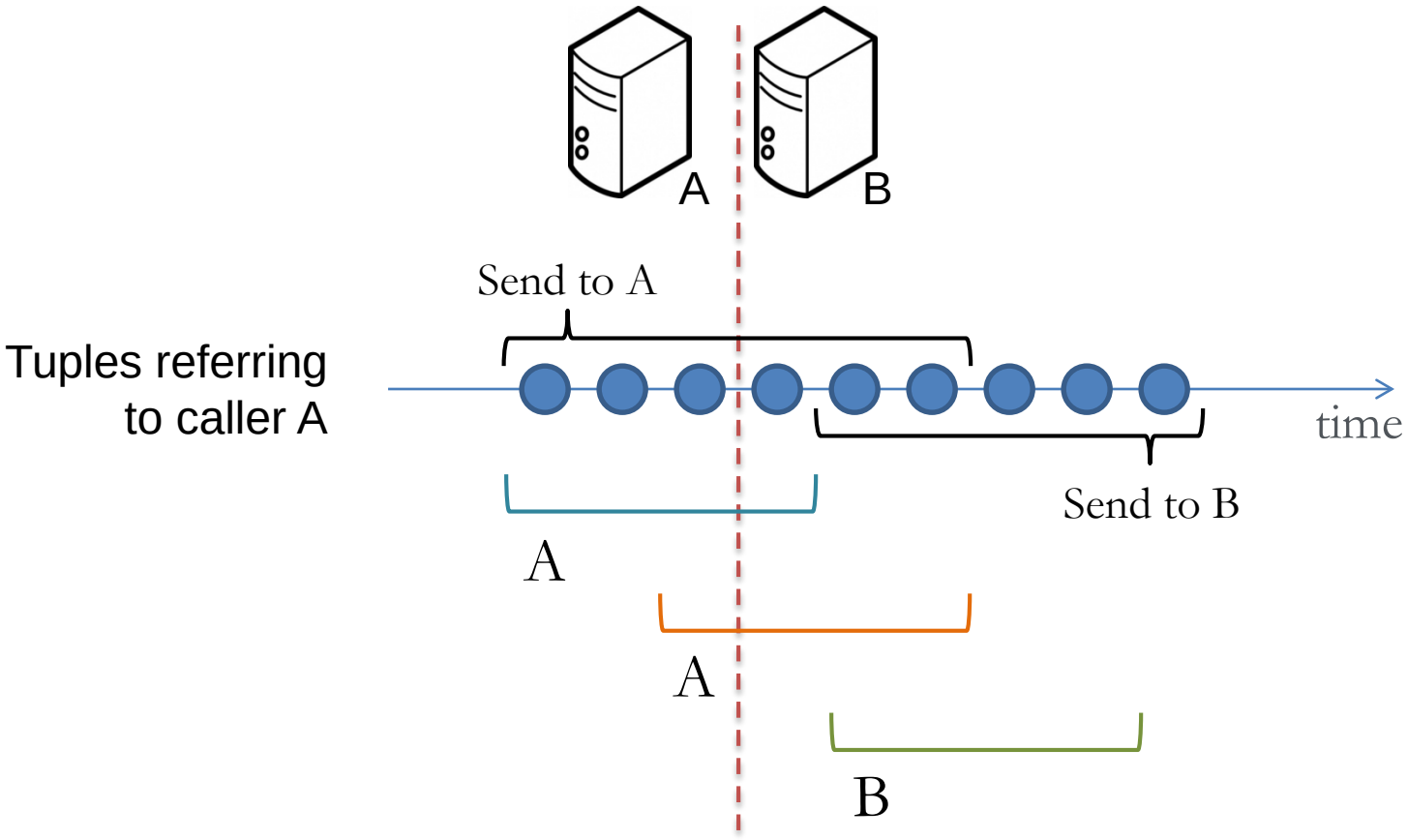
StreamCloud - Elasticity

- Window Recreation Protocol



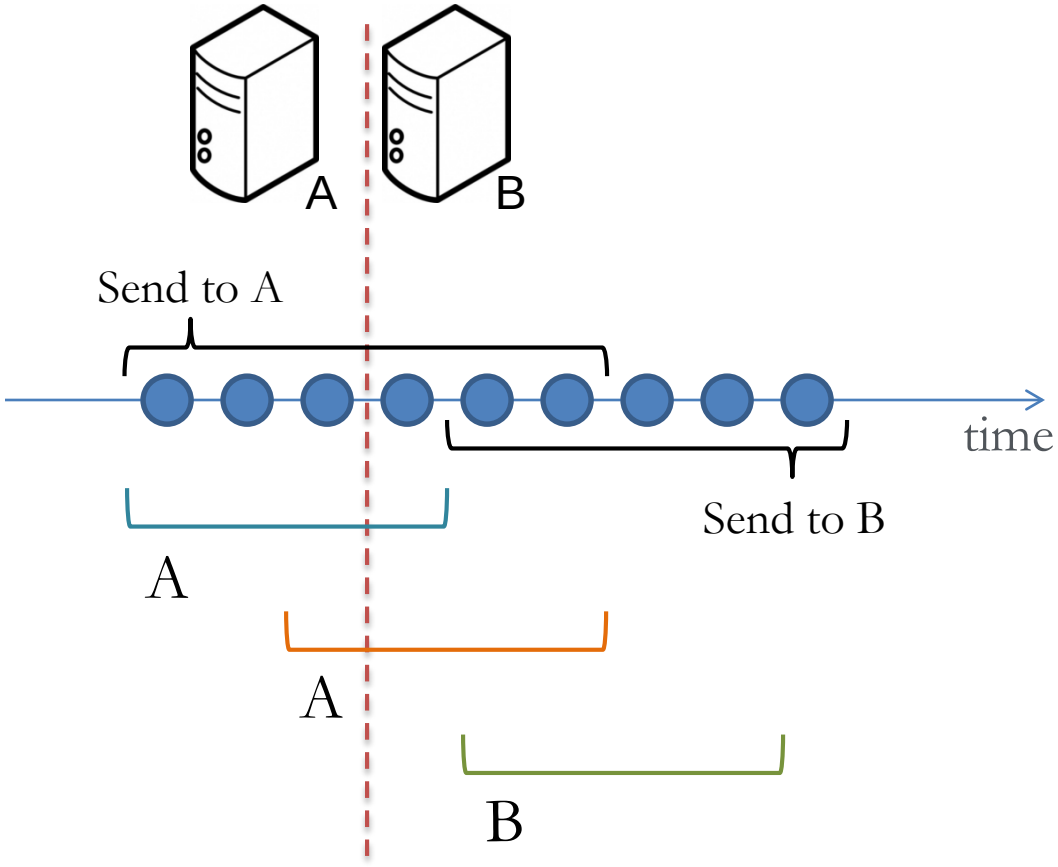
StreamCloud - Elasticity

- Window Recreation Protocol



StreamCloud - Elasticity

- Window Recreation Protocol

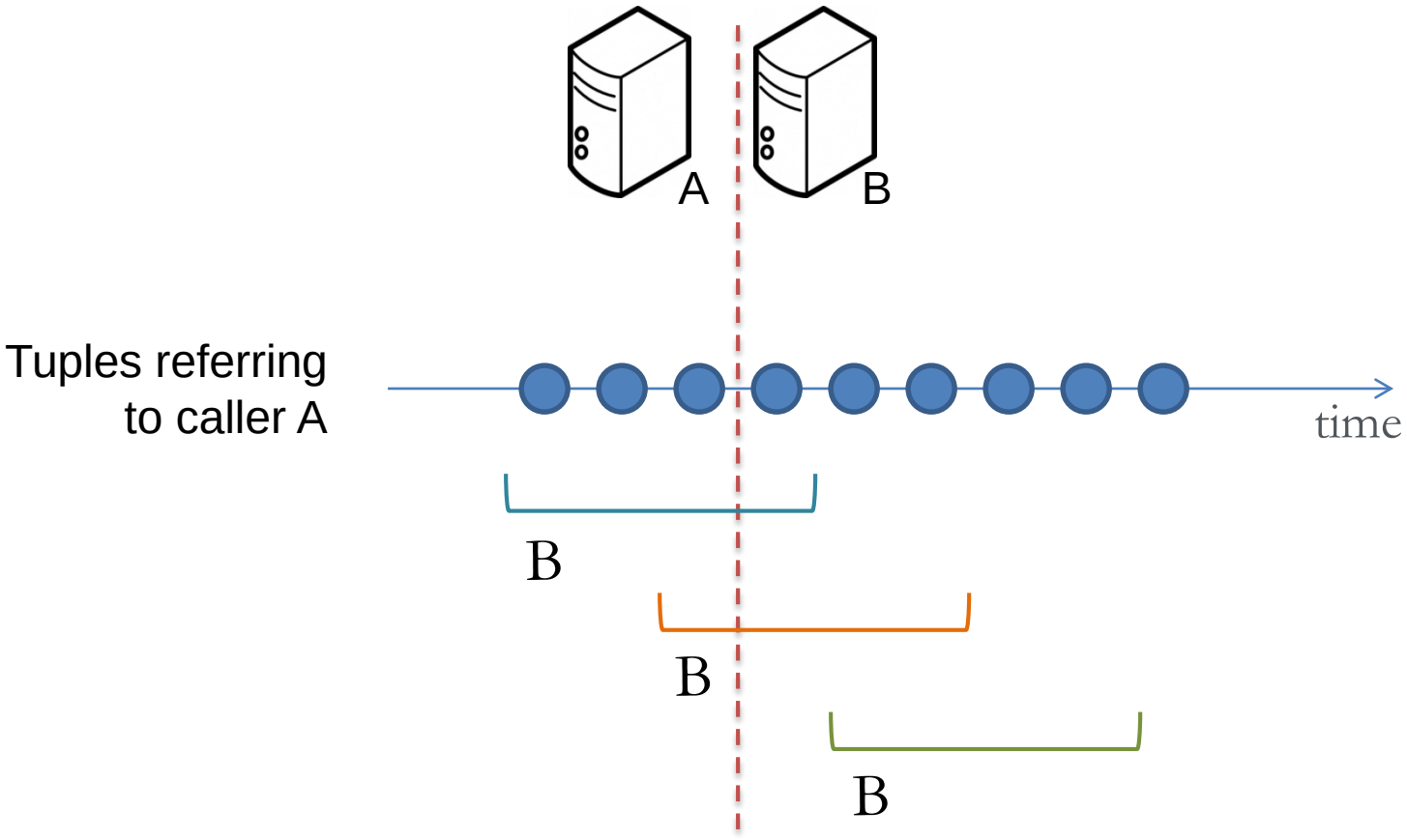


+ No communication between nodes

- Completion time proportional to window size

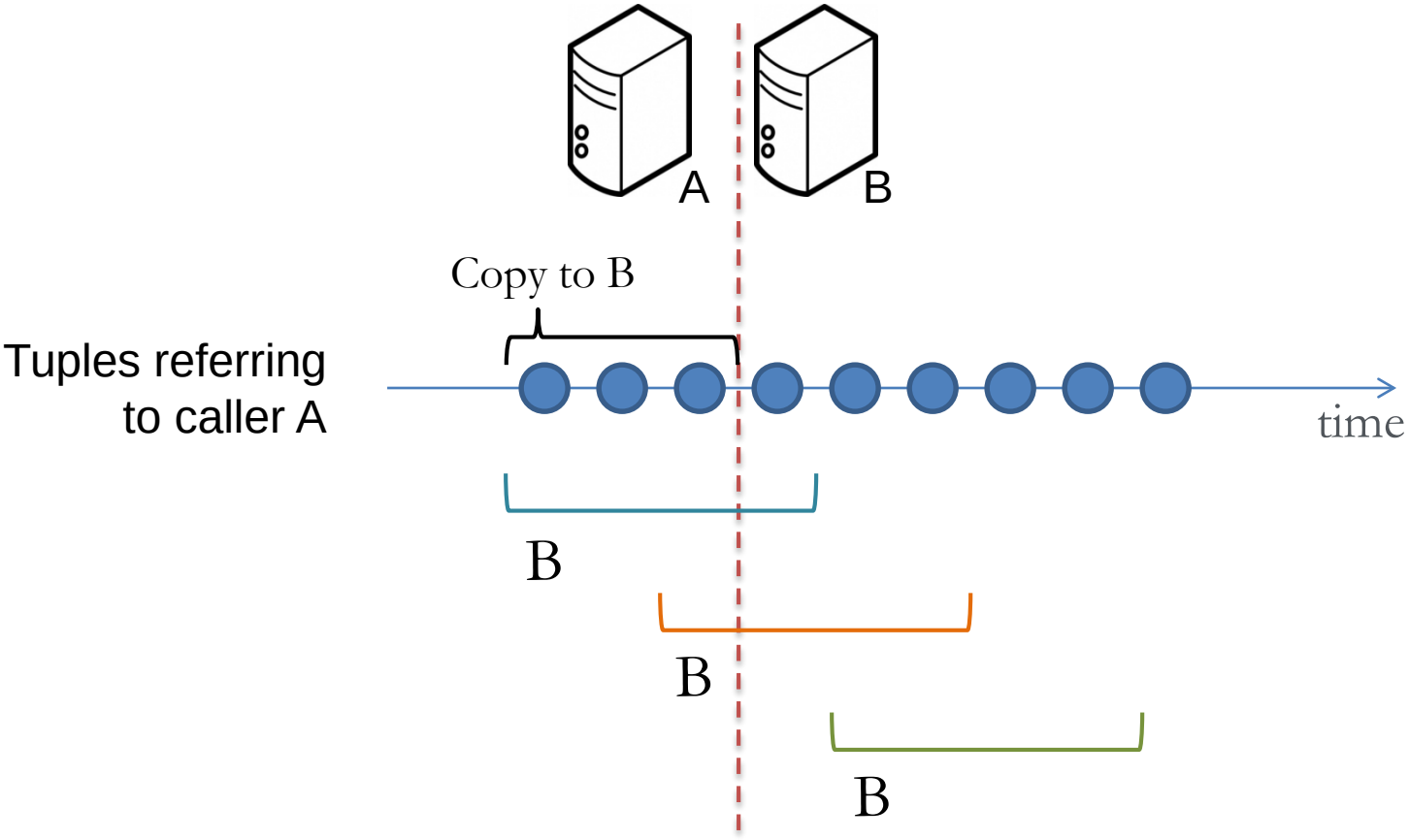
StreamCloud - Elasticity

- State Recreation Protocol



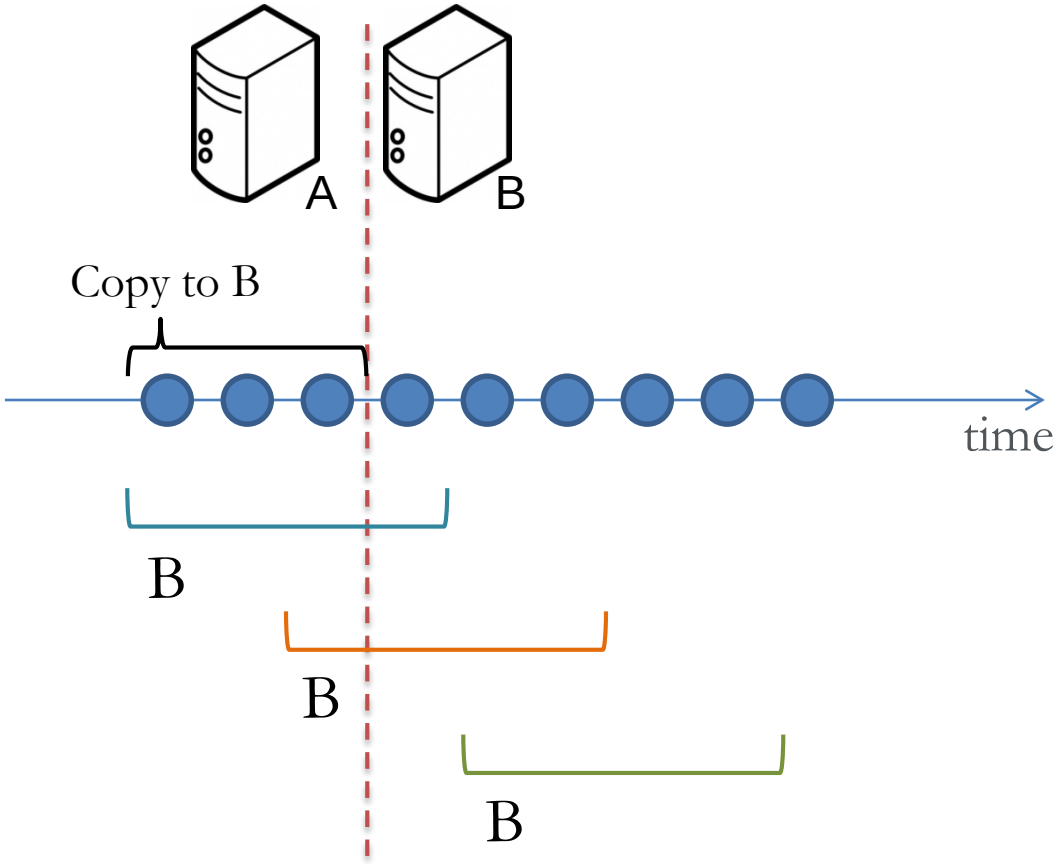
StreamCloud - Elasticity

- State Recreation Protocol



StreamCloud - Elasticity

- State Recreation Protocol



+ Minimizes completion time

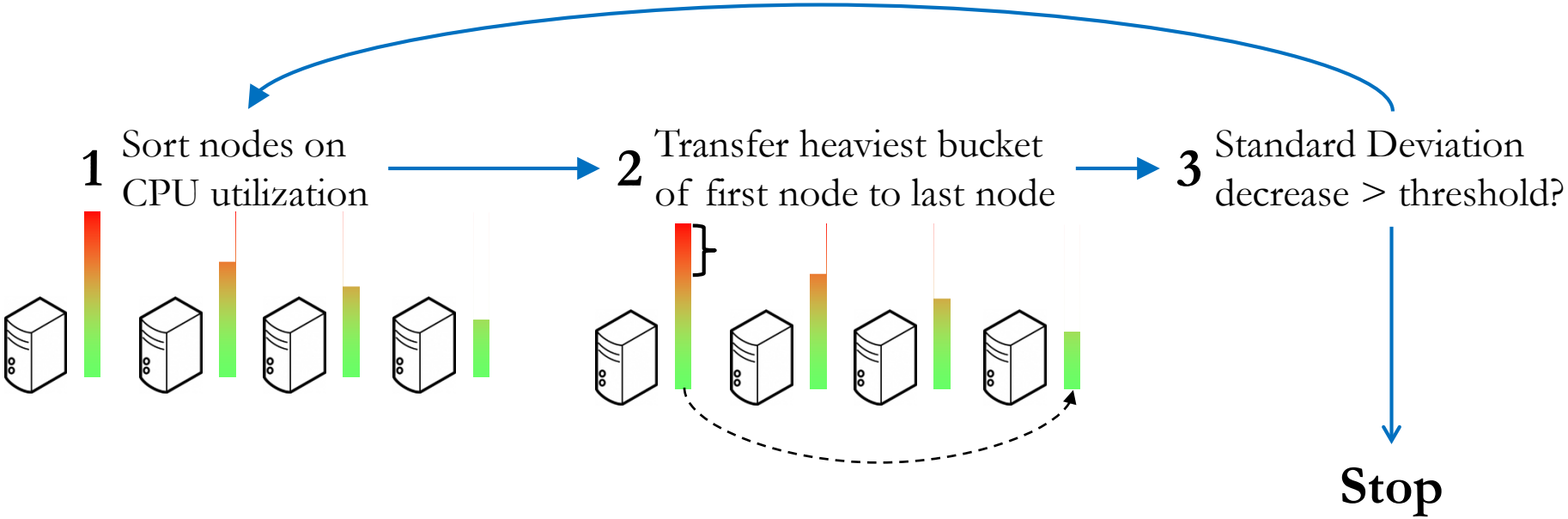
- Communication between nodes

StreamCloud - Elasticity

- Load balancing algorithm:
- Finding the right assignment of buckets to machines → bin packing problem (NP hard)
- Should reduce the number of state transfers (cannot reallocate all buckets)

StreamCloud - Elasticity

- Greedy algorithm:
  - Balance load → Minimized CPU standard deviation

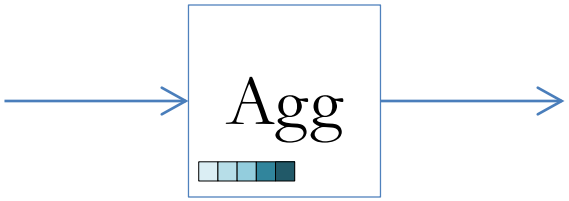


StreamCloud - Elasticity

- Load-aware provisioning evaluation
- Query: High Mobility (cellular telephony fraud detection)

StreamCloud - Elasticity

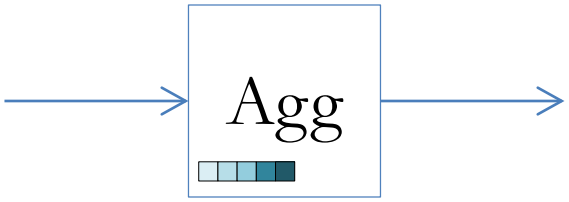
- Load-aware provisioning evaluation
- Query: High Mobility (cellular telephony fraud detection)



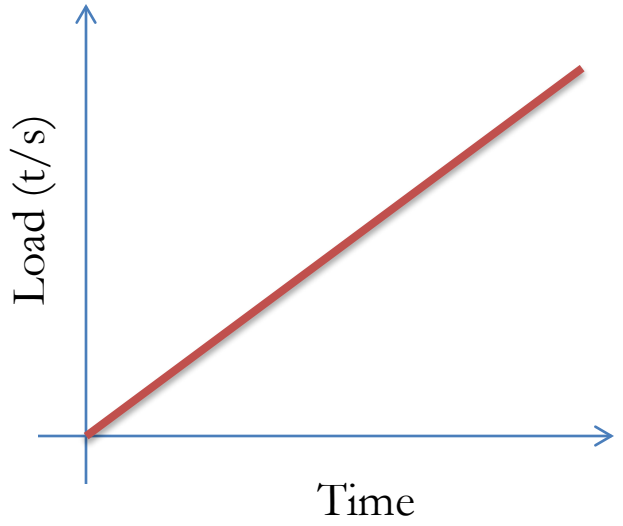
Extract time and position for each pair of consecutive CDR from the same Caller

StreamCloud - Elasticity

- Load-aware provisioning evaluation
- Query: High Mobility (cellular telephony fraud detection)

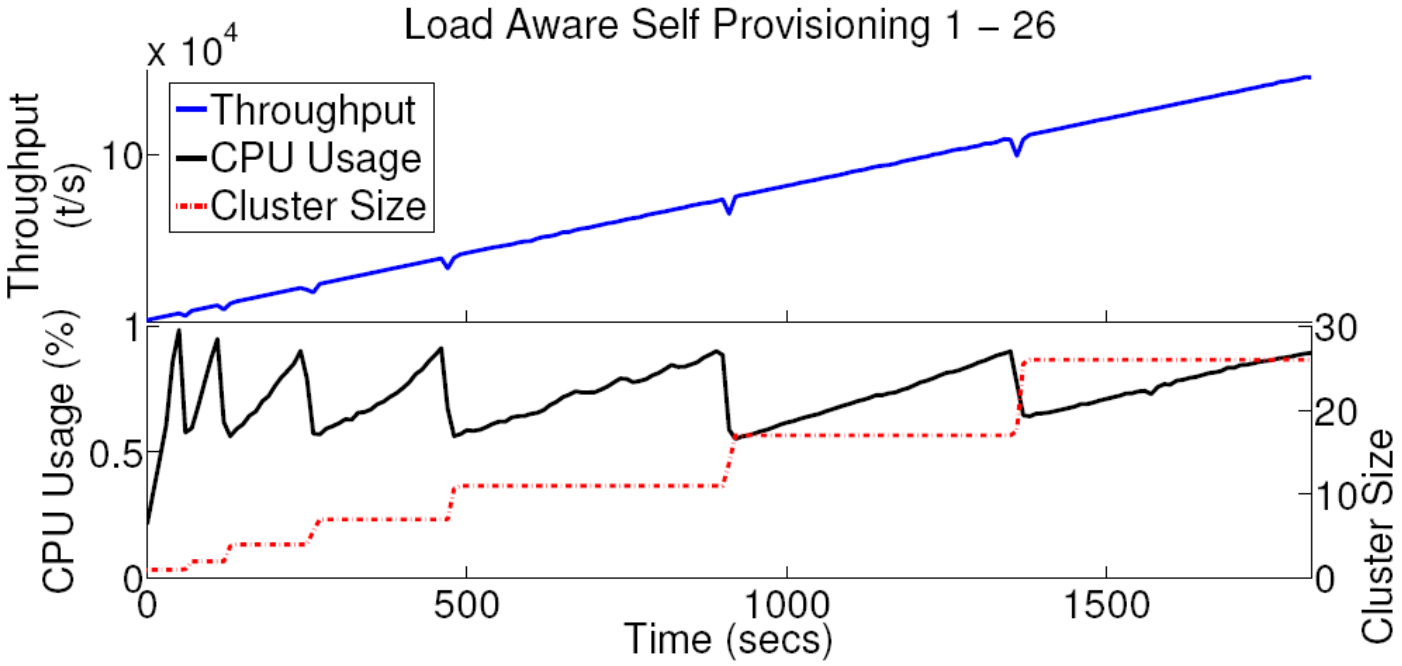


Extract time and position for each pair of consecutive CDR from the same Caller



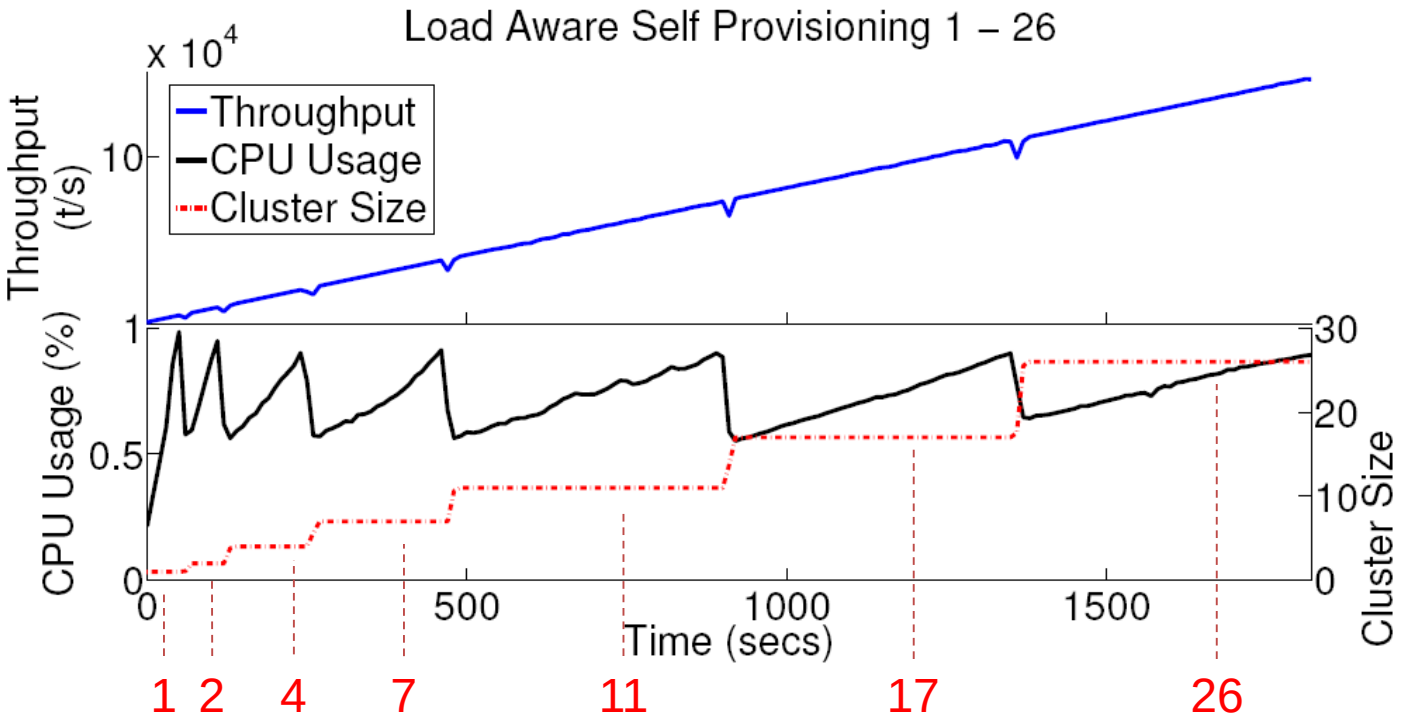
StreamCloud - Elasticity

- Load-aware provisioning



StreamCloud - Elasticity

- Load-aware provisioning



Agenda

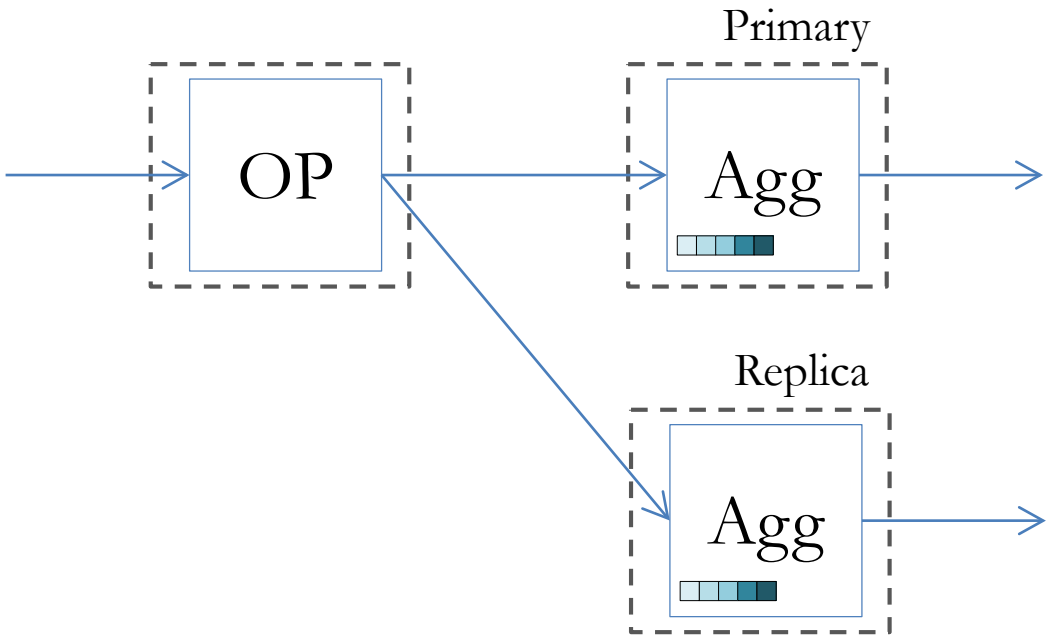
- Introduction
  - Motivation
  - System Model
- Parallelization
- Elasticity
- Fault tolerance
- Integrated Development Environment
- Conclusions

StreamCloud - Fault Tolerance

- Studied in the context of distributed SPEs
- Existing techniques:
  - Active standby
  - Passive standby
  - Upstream backup

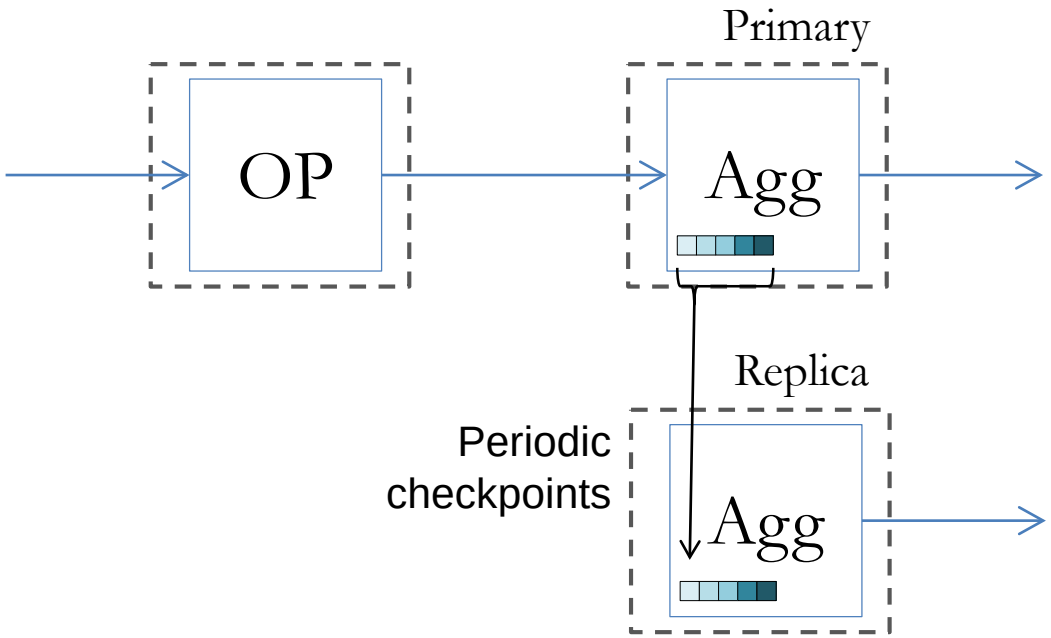
Fault Tolerance – State of the Art

- Active standby



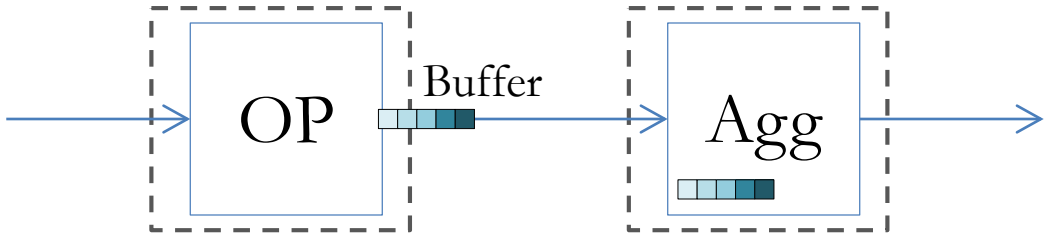
Fault Tolerance – State of the Art

- Passive standby



Fault Tolerance – State of the Art

- Upstream Backup



StreamCloud - Fault Tolerance

- Low-cost redundancy

StreamCloud - Fault Tolerance

- Low-cost redundancy
  → Move redundancy at the storage level (cheaper)

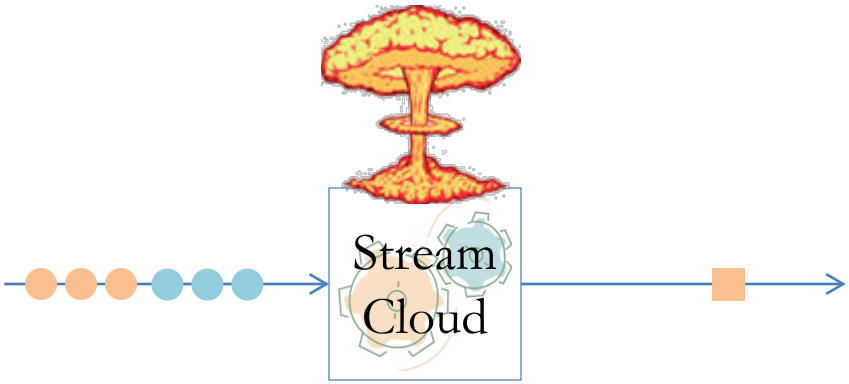
StreamCloud - Fault Tolerance

- Low-cost redundancy
  → Move redundancy at the storage level (cheaper)
- Precise Recovery

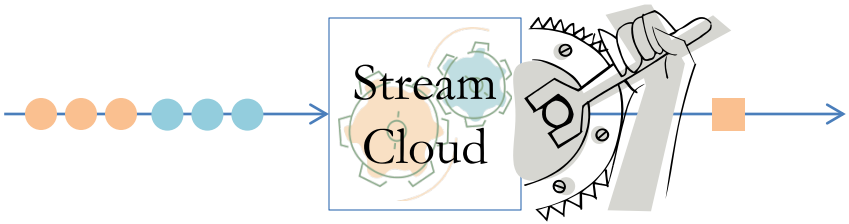
StreamCloud - Fault Tolerance

StreamCloud - Fault Tolerance

StreamCloud - Fault Tolerance



StreamCloud - Fault Tolerance



StreamCloud - Fault Tolerance

StreamCloud - Fault Tolerance

Maintain information about the last processed tuples

StreamCloud - Fault Tolerance

- Low-cost redundancy
  → Move redundancy at the storage level (cheaper)
- Precise Recovery
  → Maintain information about last processed tuples

StreamCloud - Fault Tolerance

- Low-cost redundancy
  → Move redundancy at the storage level (cheaper)
- Precise Recovery
  → Maintain information about last processed tuples
- Decouple State Maintenance – Topology

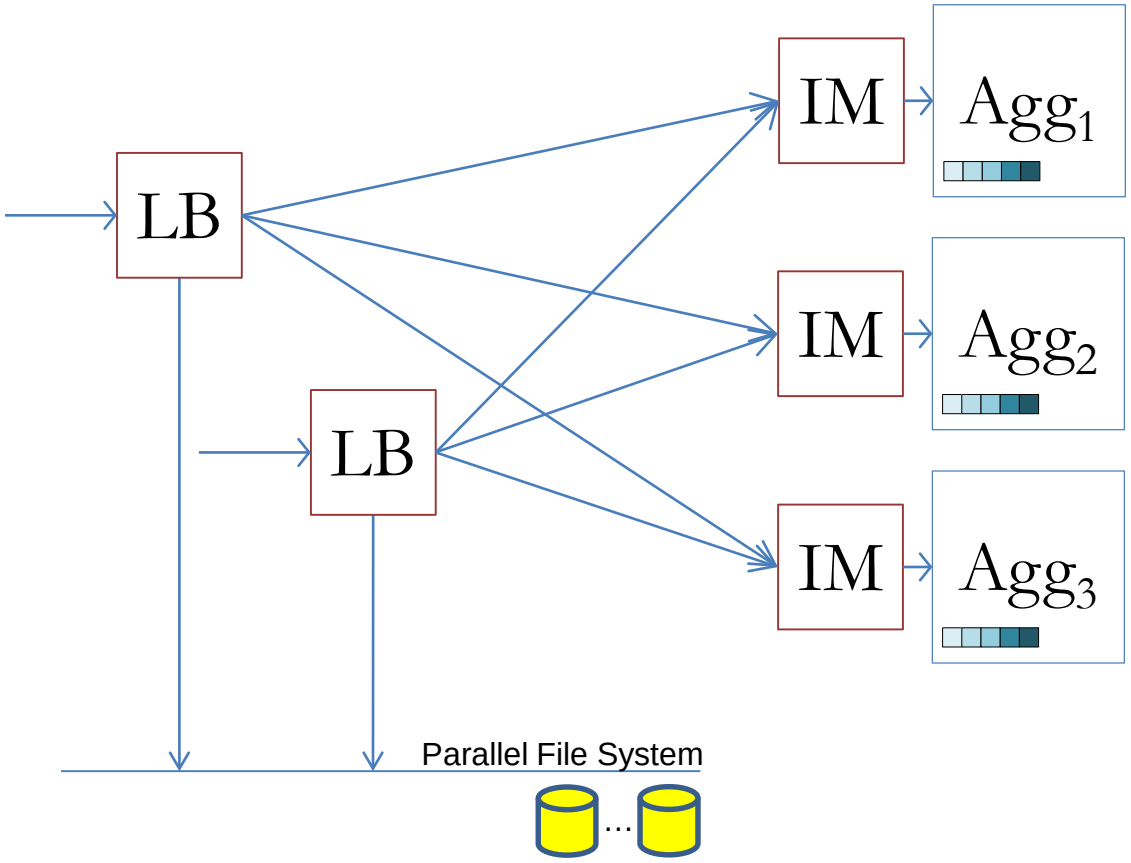
StreamCloud - Fault Tolerance

- Low-cost redundancy
  → Move redundancy at the storage level (cheaper)
- Precise Recovery
  → Maintain information about last processed tuples
- Decouple State Maintenance – Topology
  - System is not STATIC → No point in check-pointing nodes

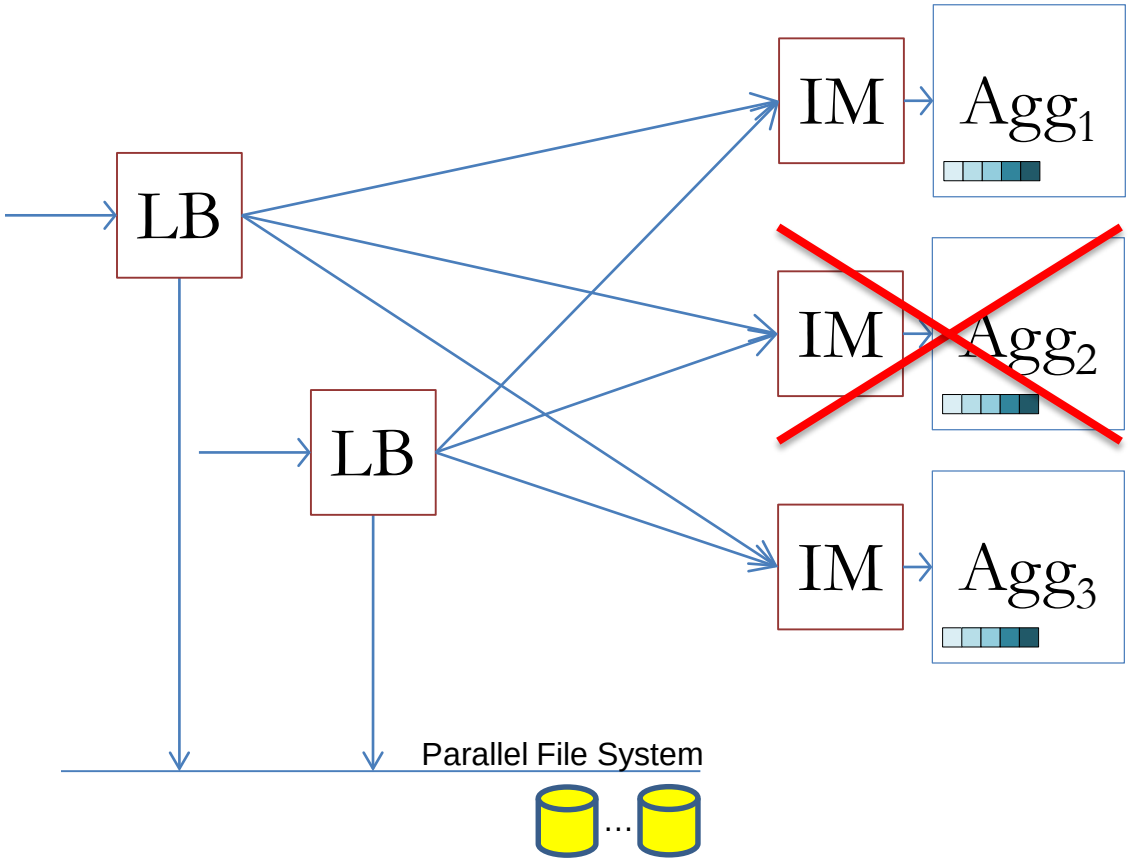
StreamCloud - Fault Tolerance

- Low-cost redundancy
  → Move redundancy at the storage level (cheaper)
- Precise Recovery
  → Maintain information about last processed tuples
- Decouple State Maintenance – Topology
  - System is not STATIC → No point in check-pointing nodes
  - Maintain past tuples of buckets

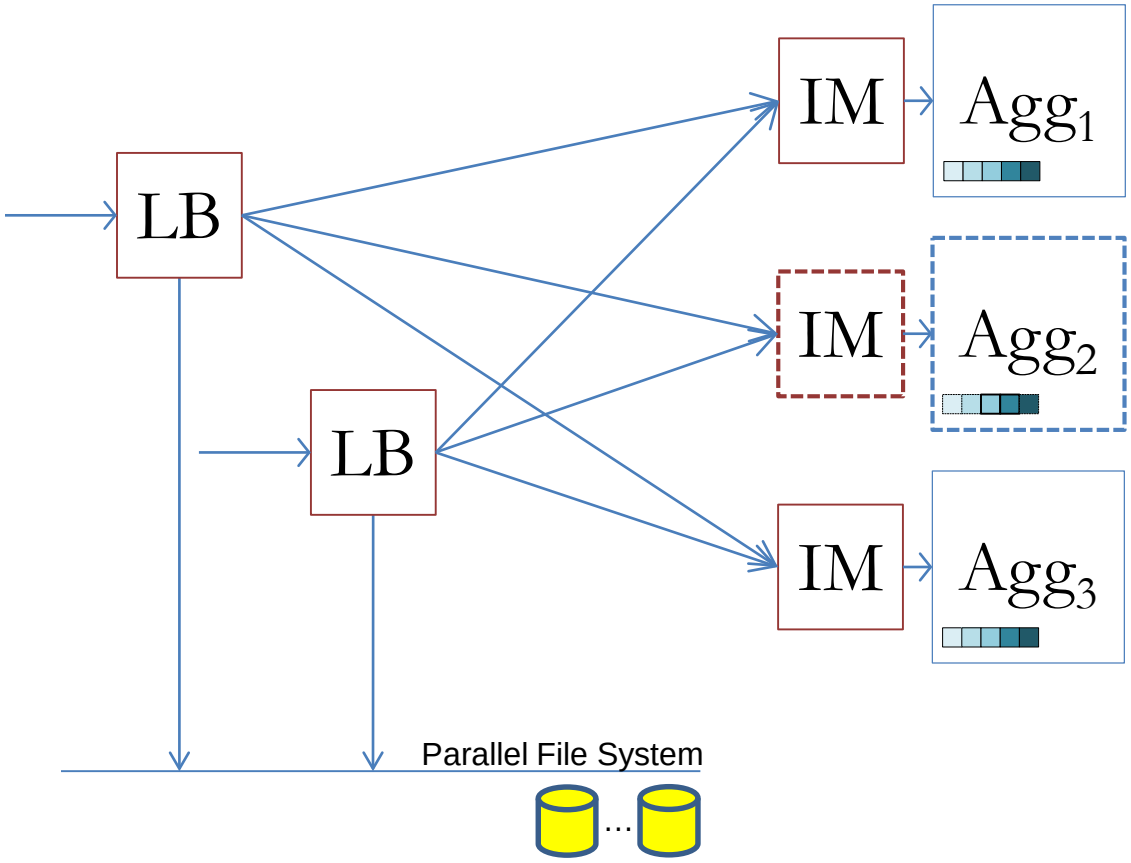
StreamCloud - Fault Tolerance



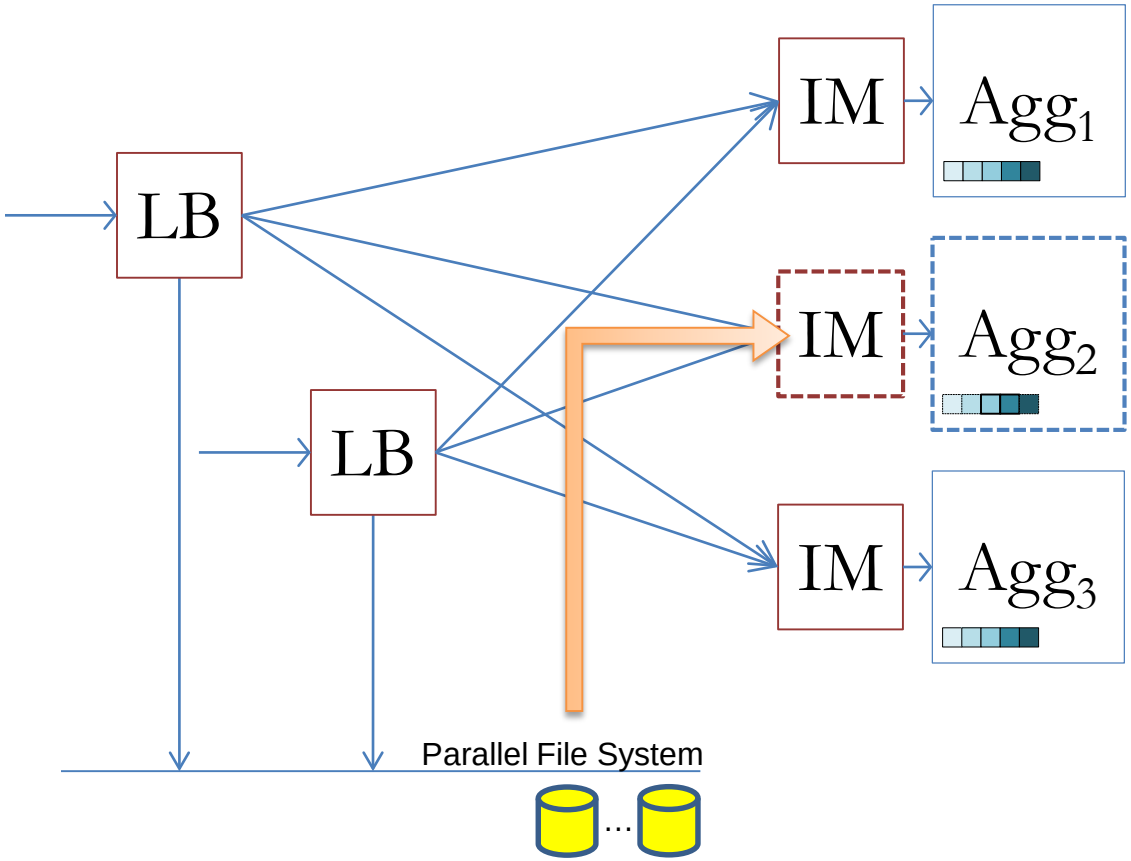
StreamCloud - Fault Tolerance



StreamCloud - Fault Tolerance



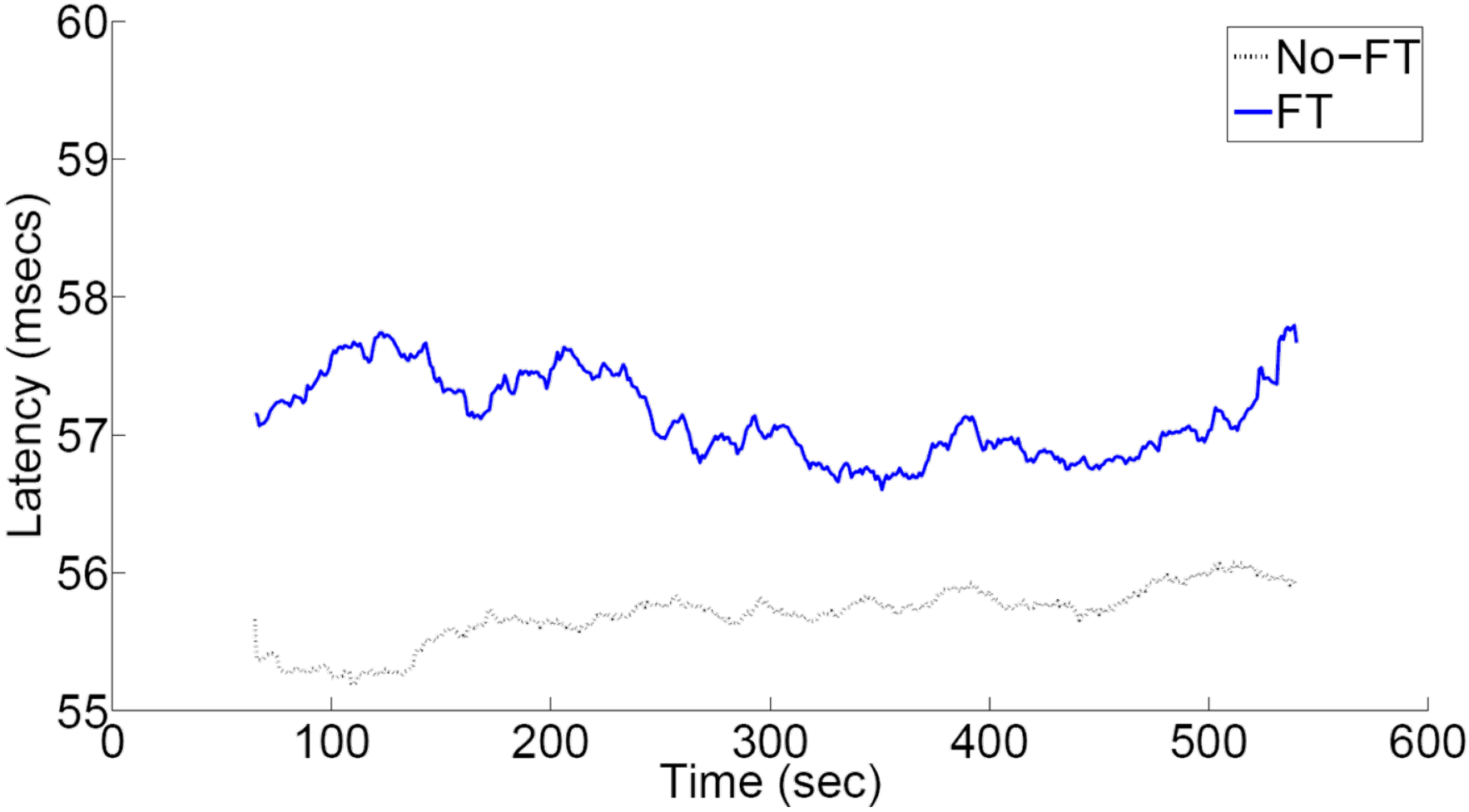
StreamCloud - Fault Tolerance



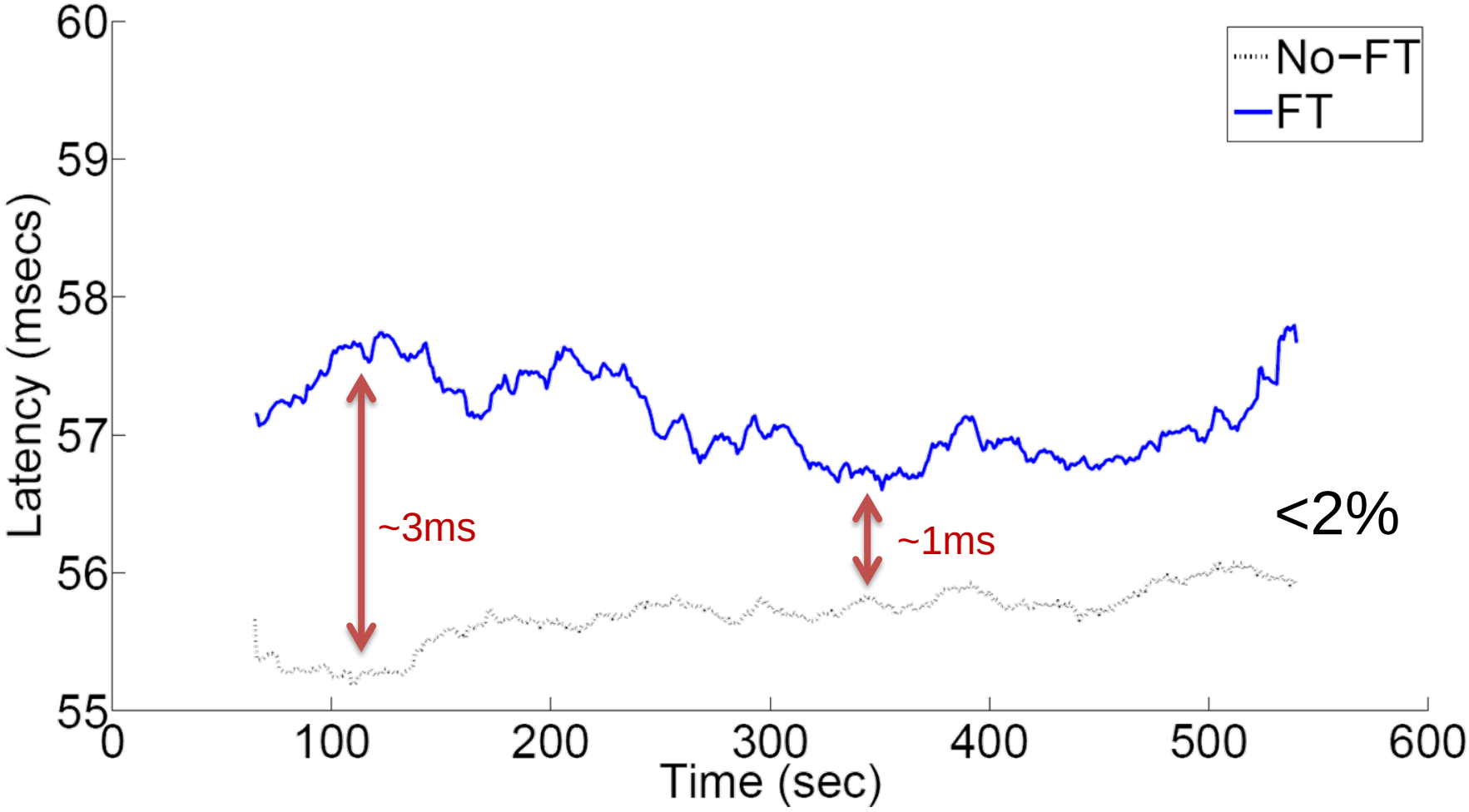
StreamCloud - Fault Tolerance

- Evaluation – Runtime overhead
- Excerpt of the Linear Road benchmark query
  - Operators used to detect accidents
  - 1 aggregate + 1 filter
- 10 nodes
- Input load ~30000 tuples / second

StreamCloud - Fault Tolerance



StreamCloud - Fault Tolerance



Agenda

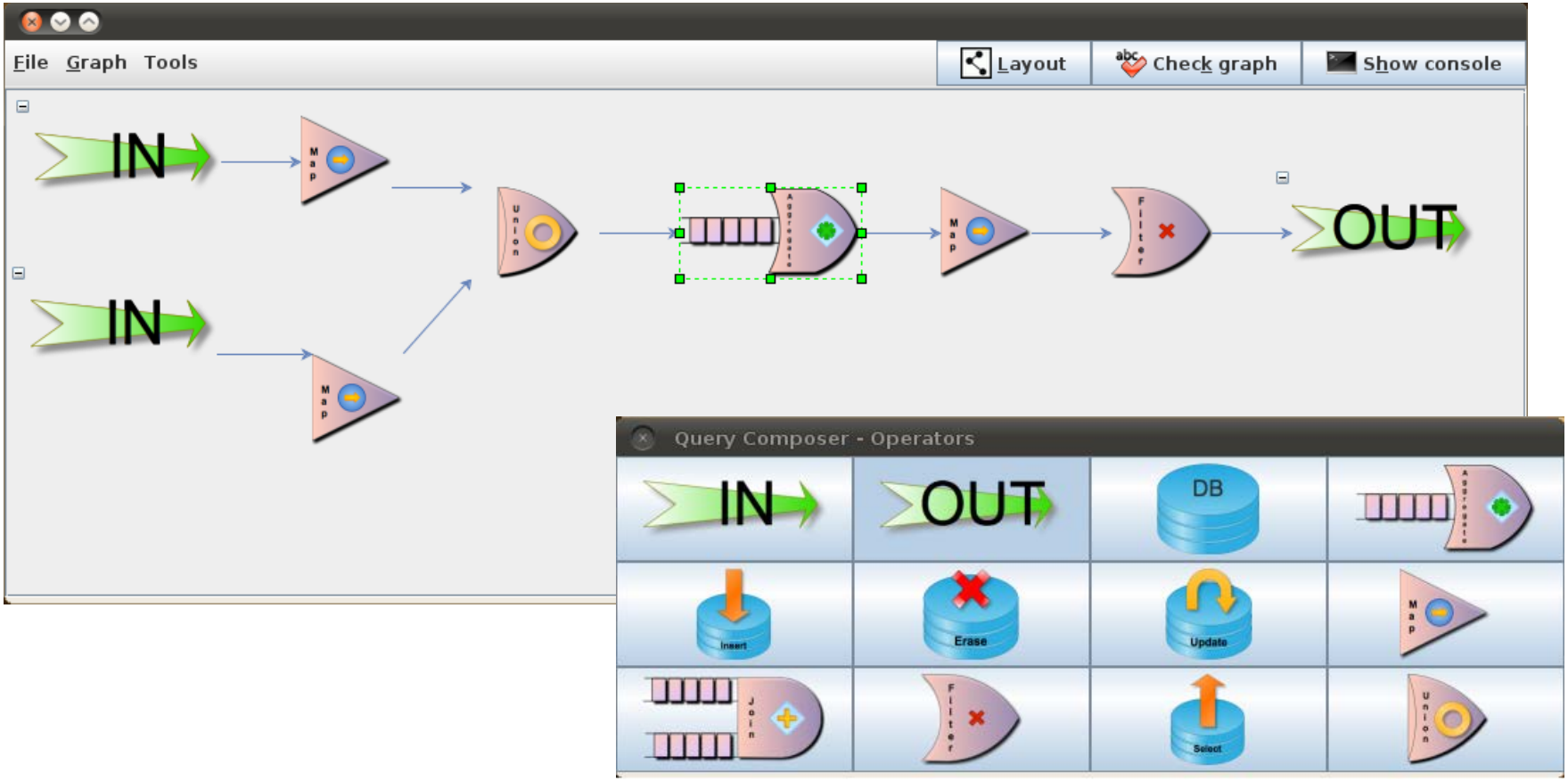
- Introduction
  - Motivation
  - System Model
- Parallelization
- Elasticity
- Fault tolerance
- Integrated Development Environment
- Conclusions

StreamCloud - IDE

- Ease the user interaction
  - Which is the min information to provide to run a data streaming application?
    - Query operators
    - Available nodes

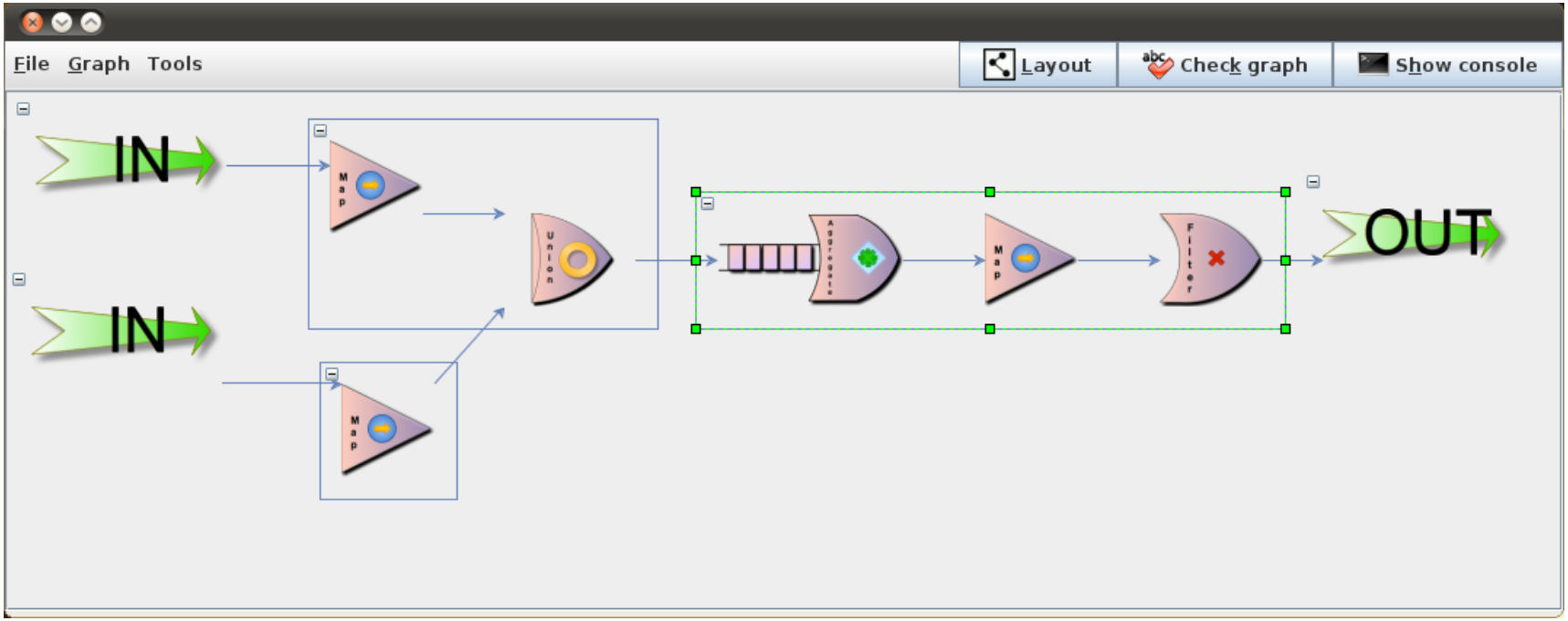
StreamCloud - IDE

Visual Query Composer



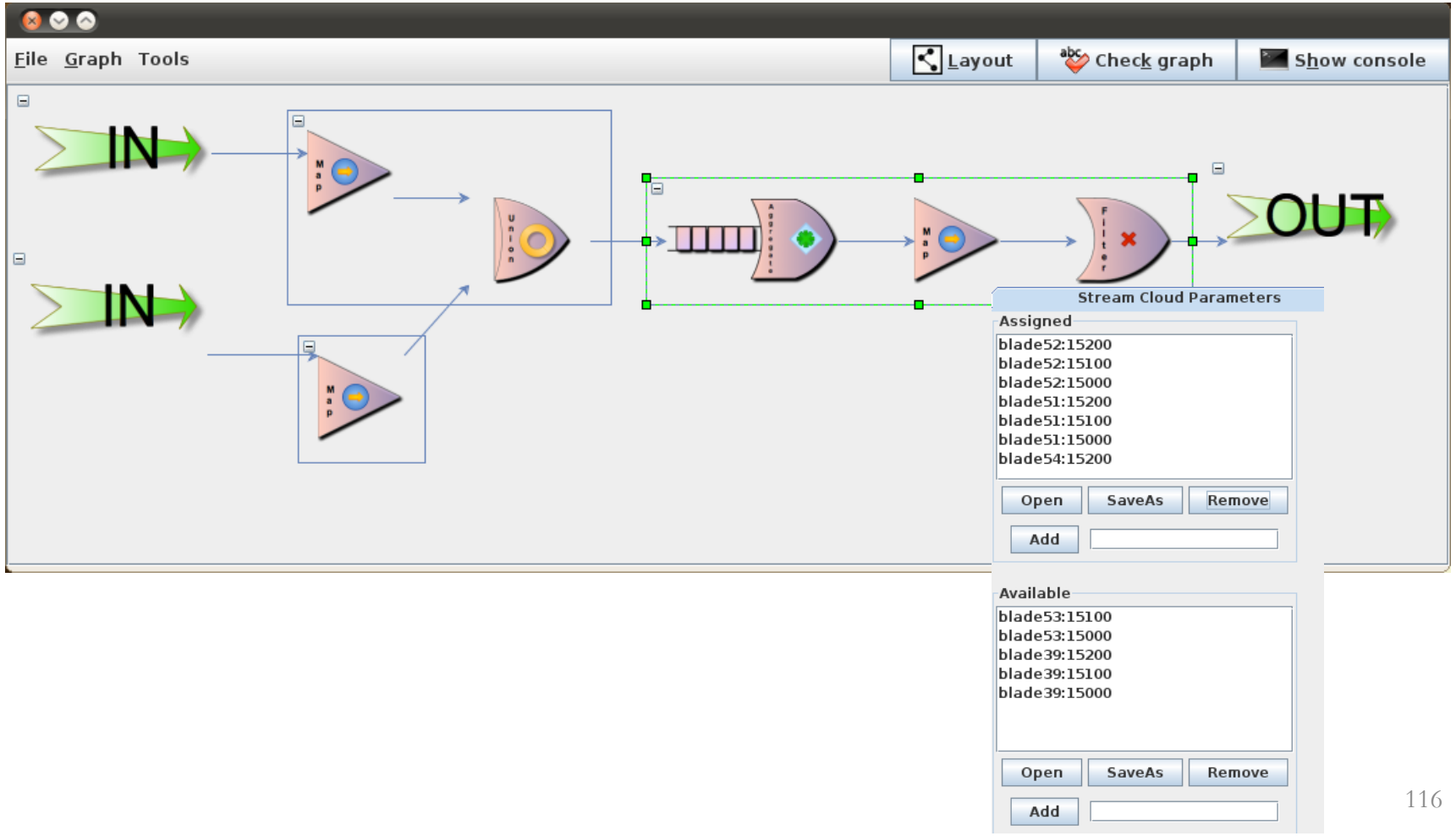
StreamCloud - IDE

Visual Query Composer



StreamCloud - IDE

Visual Query Composer



StreamCloud - IDE

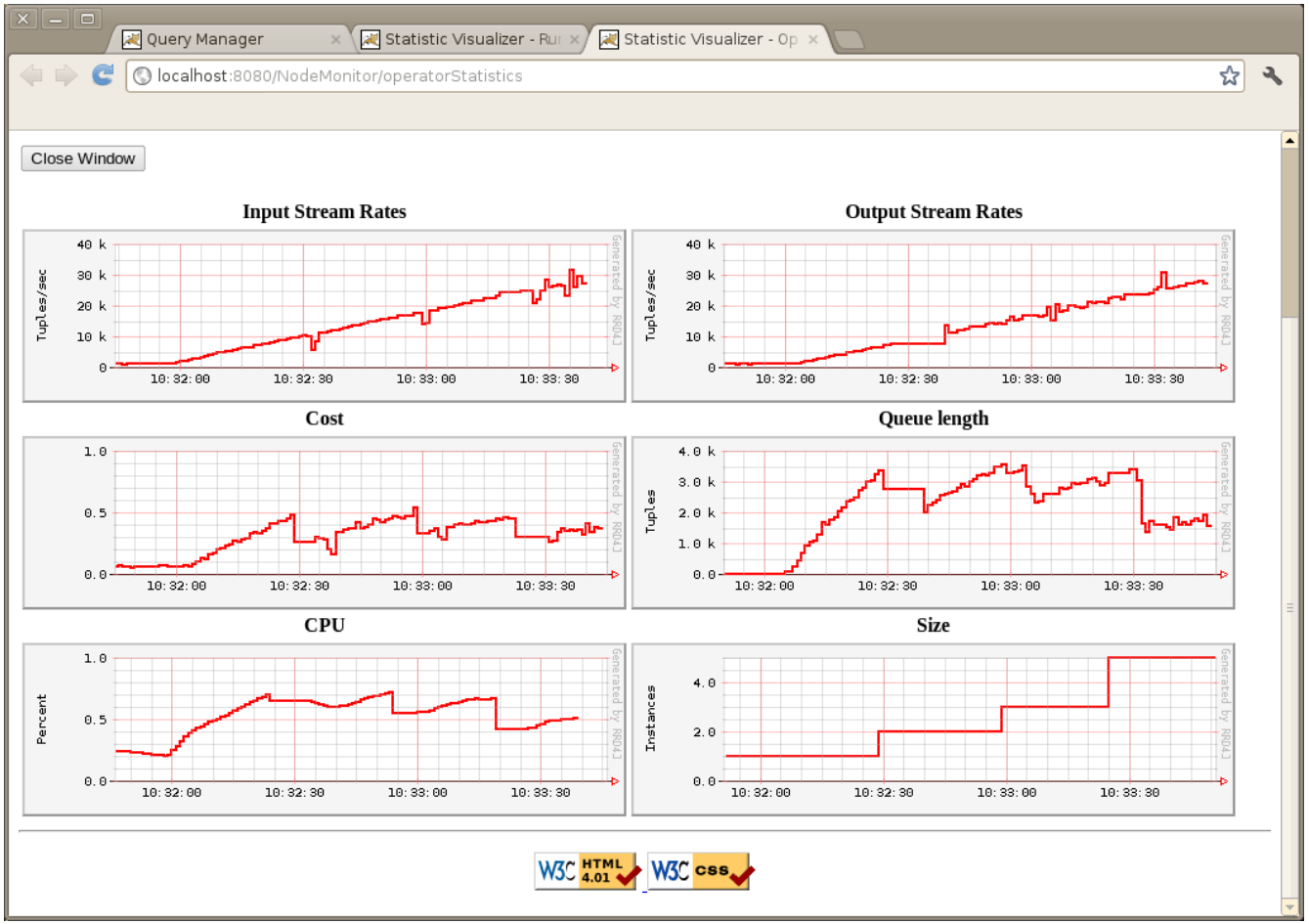
- Ease the user interaction
  - Which is the min information to provide to run a data streaming application?
    - Query operators
    - Available nodes
- Tools
  - Parallel Compiler
  - Deployer
  - Distributed Load injectors
  - Monitoring of queries

StreamCloud - IDE

- Monitoring of queries
- For each query operator, provides the following statistics:
  - Input Stream Rate
  - Output Stream Rate
  - Cost
  - Queue Length
  - CPU
  - # of Nodes

StreamCloud - IDE

- Monitoring of queries



Agenda

- Introduction
  - Motivation
  - System Model
- Parallelization
- Elasticity
- Fault tolerance
- Integrated Development Environment
- Conclusions

Conclusions

- StreamCloud
  - Parallelization of data streaming operators
  - Load-aware Self Provisioning – Decommissioning and dynamic load balancing
  - Fault Tolerance
  - IDE to ease user interaction

Conclusions

- Selected publications:
- International Journals
  - StreamCloud: An Elastic and Scalable Data Streaming System.
    V. Gulisano, R. Jimenez-Peris, M. Patiño-Martinez, C. Soriente, P. Valduriez - IEEE Transactions on Parallel and Distributed Systems (TPDS)
- International Conferences and other publications:
  - STONE: A Stream-based DDoS Defense Framework
    M.Callau, V. Gulisano, Z. Fu, R. Jimenez-Peris, M. Papatriantafilou, M. Patiño-Martinez
    28th Annual ACM Symposium on Applied Computing (SAC) 2013
  - StreamCloud: A Large Scale Data Streaming System
    V. Gulisano, R. Jimenez-Peris, M. Patiño-Martinez, P. Valduriez
    30th Int. Conf. on Distributed Computing Systems (ICDCS) 2010
  - A New Class of Services for Cloud Computing: Real-Time Services over Massive and Continuous Data Flows.
    Ricardo Jimenez-Peris, M. Patiño-Martinez, V. Gulisano
    Cloud Futures 2010 Workshop
- Book Chapters
  - Complex Event Processing Based SIEM
    V. Gulisano, R. Jimenez-Peris, M. Patiño-Martinez, C. Soriente,V. Vianello
    Advances in Security Information Management: Perceptions and Outcomes. 2011

POLITÉCNICA

StreamCloud: an Elastic Parallel-Distributed Stream Processing Engine

Ph.D. Student
Vincenzo Massimiliano Gulisano

Director: Ricardo Jiménez Peris
Co-director: Patrick Valduriez

Lsd Distributed Systems Laboratory

December 20, 2012