An overview of Data Streaming and Stream Processing Engines

Vincenzo Gulisano

Chalmers University
of technology

Agenda

- Introduction
  - Motivation
  - System Model
- Centralized SPEs
- Distributed SPEs
- Parallel-Distributed SPEs
- Elastic SPEs
- Conclusions

Agenda

- Introduction
  - Motivation
  - System Model
- Centralized SPEs
- Distributed SPEs
- Parallel-Distributed SPEs
- Elastic SPEs
- Conclusions

Motivation

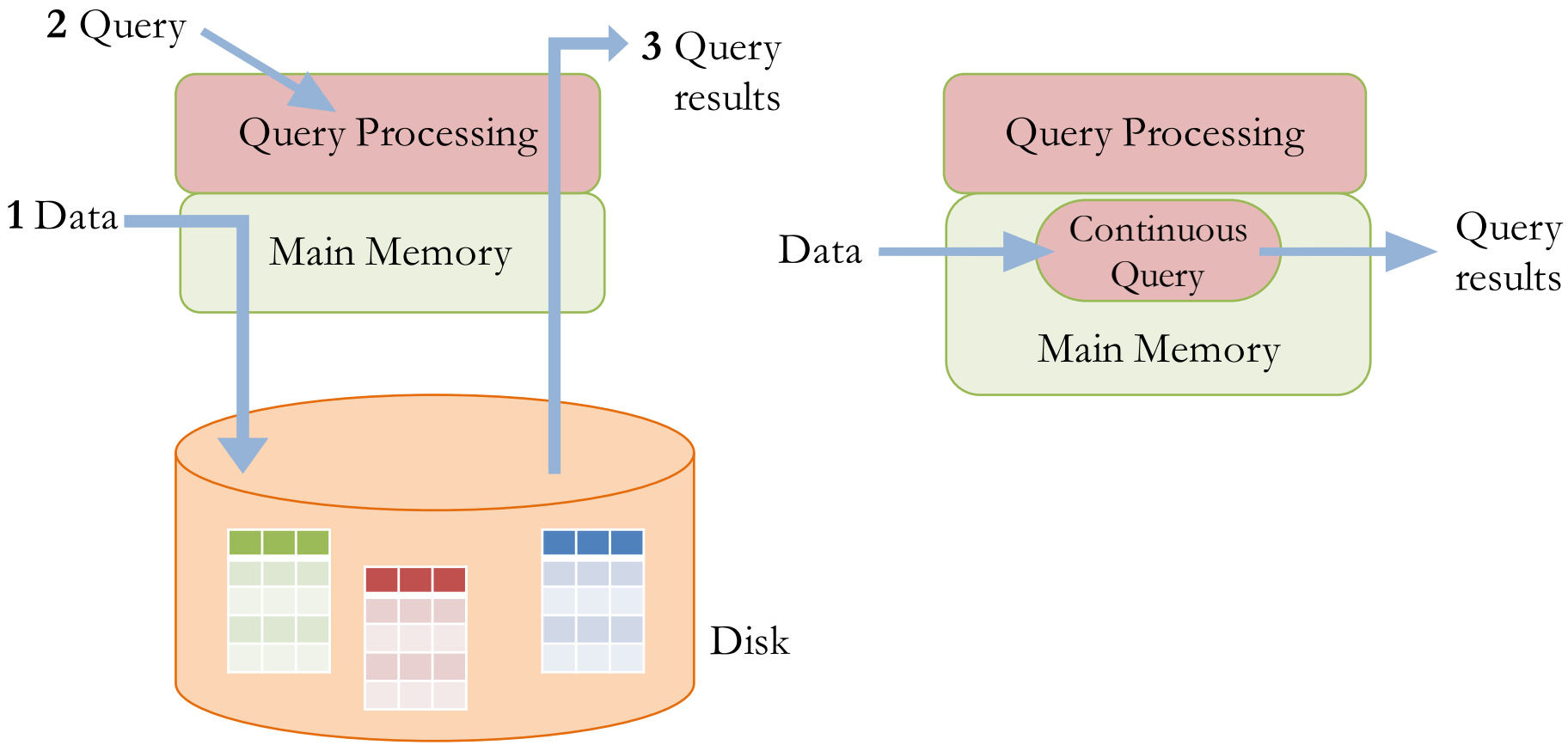
- Applications such as:
  - Sensor networks
  - Network Traffic Analysis
  - Financial tickers
  - Transaction Log Analysis
  - Fraud Detection
- Require:
  - Continuous processing of data streams
  - Real Time Fashion

Motivation

- 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

Motivation

- DBMS vs. DSMS



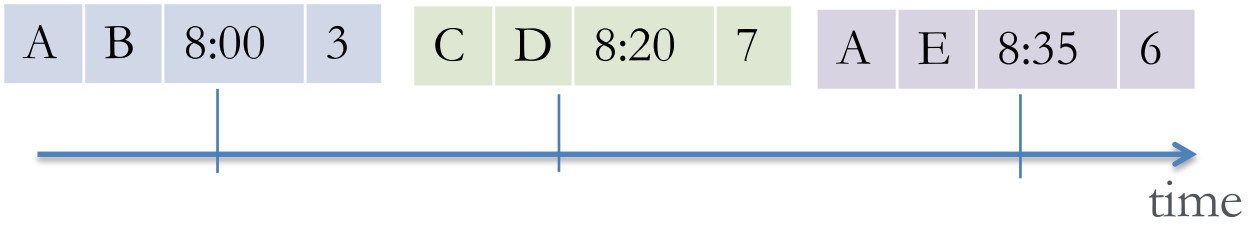
The 8 Requirements of Real-Time Stream Processing

1. Keep the data moving
2. Query interface, e.g., extended SQL
3. Handle imperfections
4. Generate predictable outcomes
5. Integrate stored and streaming data
6. Guarantee data safety and availability
7. Partition and scale applications automatically
8. Process and respond instantaneously

System Model

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

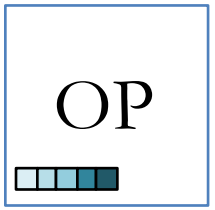
Field	Field
Caller	text
Callee	text
Time (secs)	int
Price (€)	double



System Model

- Operators:

Stateless
1 input tuple
1 output tuple

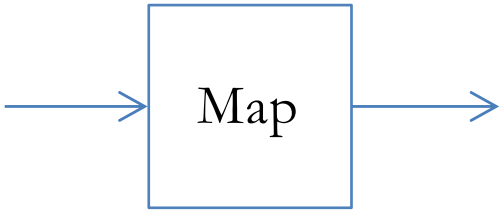


Stateful
1+ input tuple(s)
1 output tuple

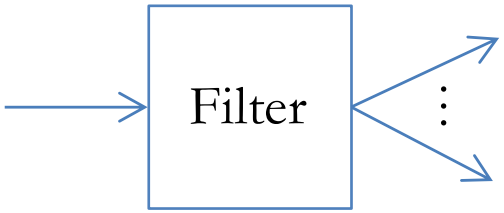
System Model

Stateless Operators

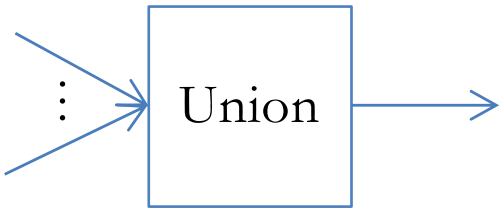
Map: transform tuples schema
Example: convert price € → $



Filter: discard / route tuples
Example: route depending on price



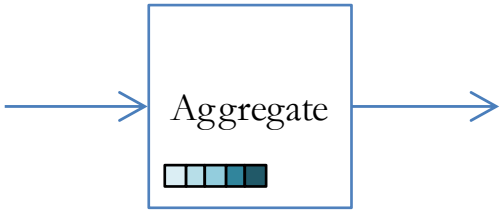
Union: merge multiple streams (sharing the same schema)
Example: merge CDRs from different sources



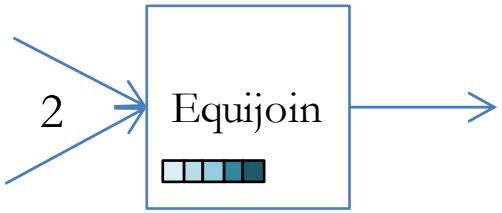
System Model

Stateful Operators

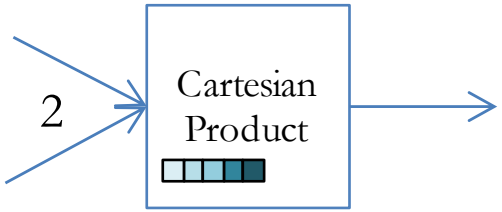
Aggregate: compute aggregate functions (group-by)
Example: compute avg. call duration



Equijoin: match tuples from 2 streams (equality predicate)
Example: match CDRs with same price

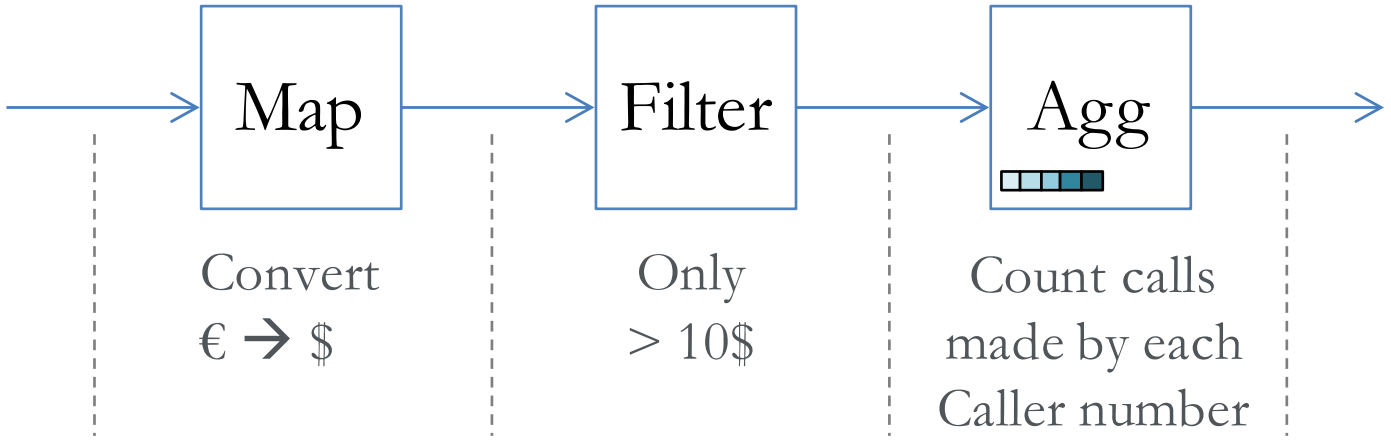


Cartesian Product: merge tuples from 2 streams (arbitrary predicate)
Example: match CDRs with prices in the same range



System Model

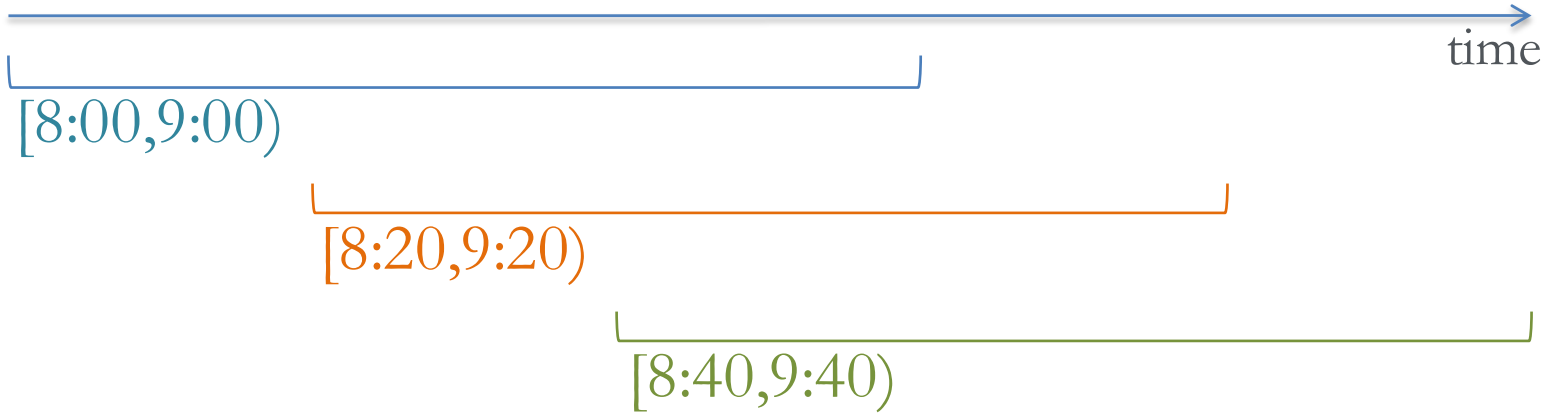
- Continuous Query: graph operators/streams



System Model

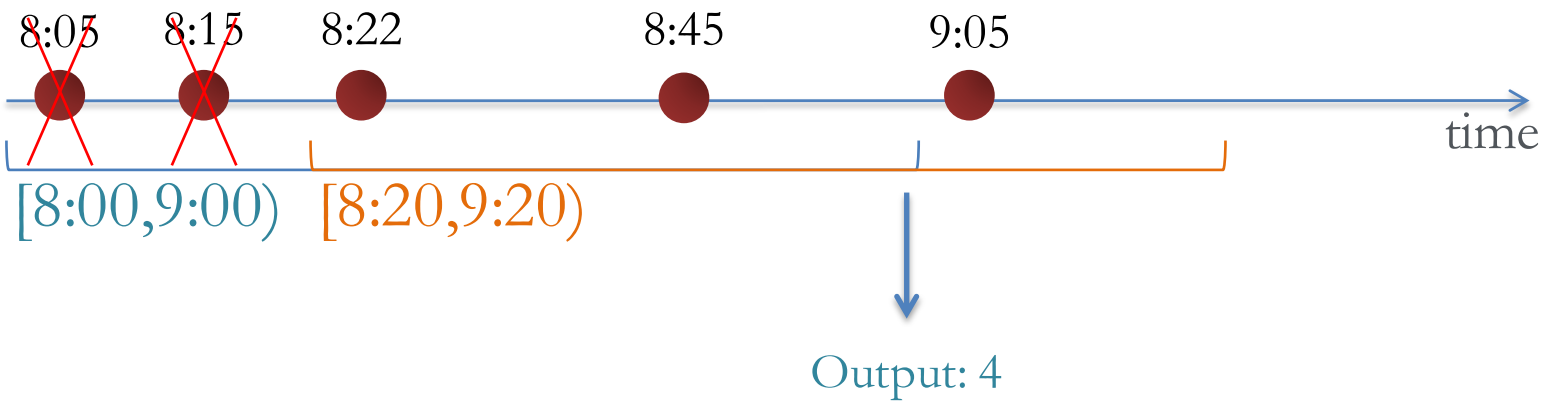
- Infinite sequence of tuples / bounded memory

  → windows
- Example: 1 hour windows



System Model

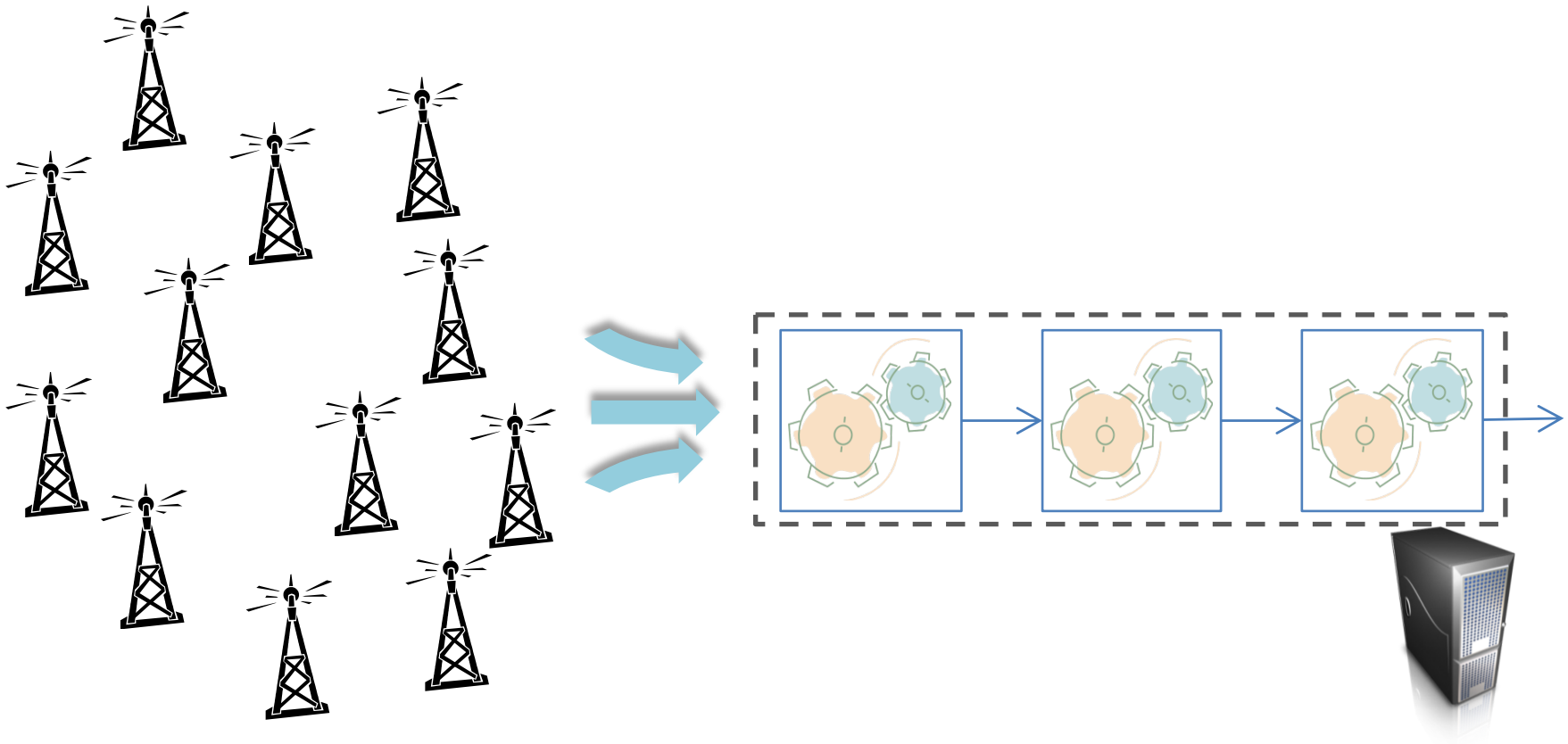
- Infinite sequence of tuples / bounded memory
  → windows
- Example: count tuples - 1 hour windows



Agenda

- Introduction
  - Motivation
  - System Model
- Centralized SPEs
- Distributed SPEs
- Parallel-Distributed SPEs
- Elastic SPEs
- Conclusions

Centralized SPEs



Pioneer SPEs

- Cougar
  - Distributed processing of sensors data
- NiagaraCQ
  - Queries over distributed XML data
- TelegraphCQ
  - Fjord API middleware
- STREAM (Stanford stREam datA Manager)
  - CQL, declarative language that extends SQL
- Aurora
  - Focus on providing operators to compose rich queries
  - Later evolved to Aurora*, Medusa, Borealis, used also in StreamCloud

Still rely on DBs

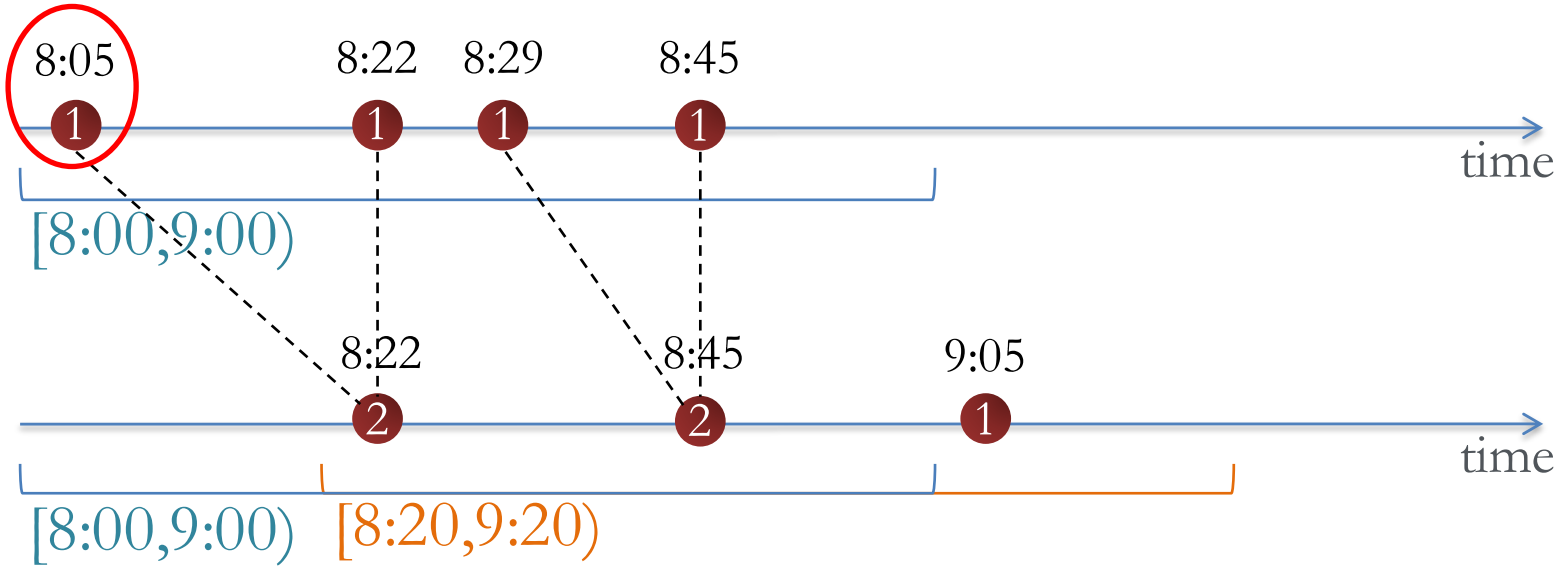
No DBs

Research topics – Centralized SPEs

- Approximation
  - Due to limited resources, how to approximate results to reduce space or time complexity?
- Load Shedding
  - If close to saturation, which information to discard in order to maximize the Quality of Service?
- Operator scheduling
  - In which order to run operators in order to minimize memory consumption?

Approximation – Exponential Histograms

- Aggregate information of multiple tuples
  - Example: count ones

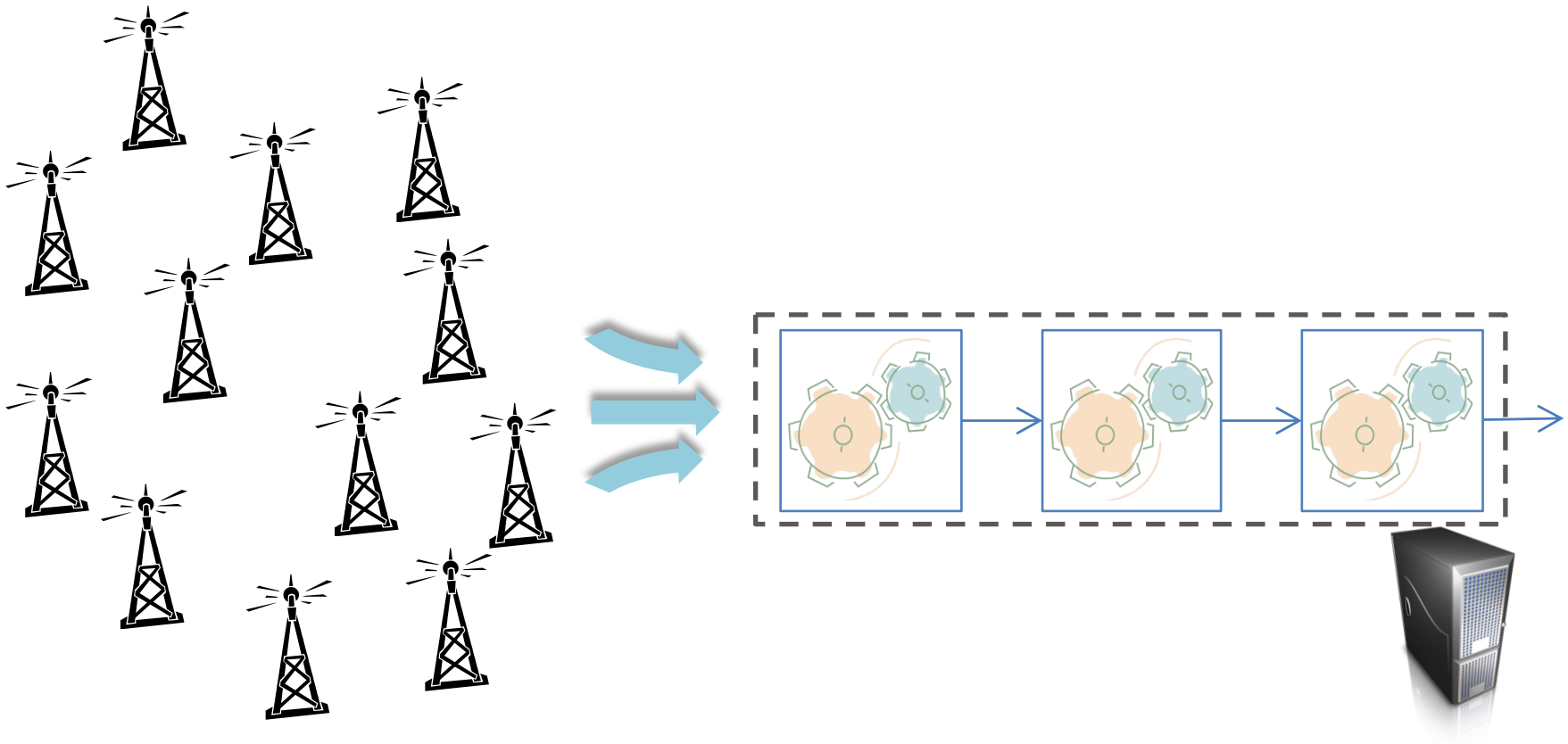


The approximation error can be bounded

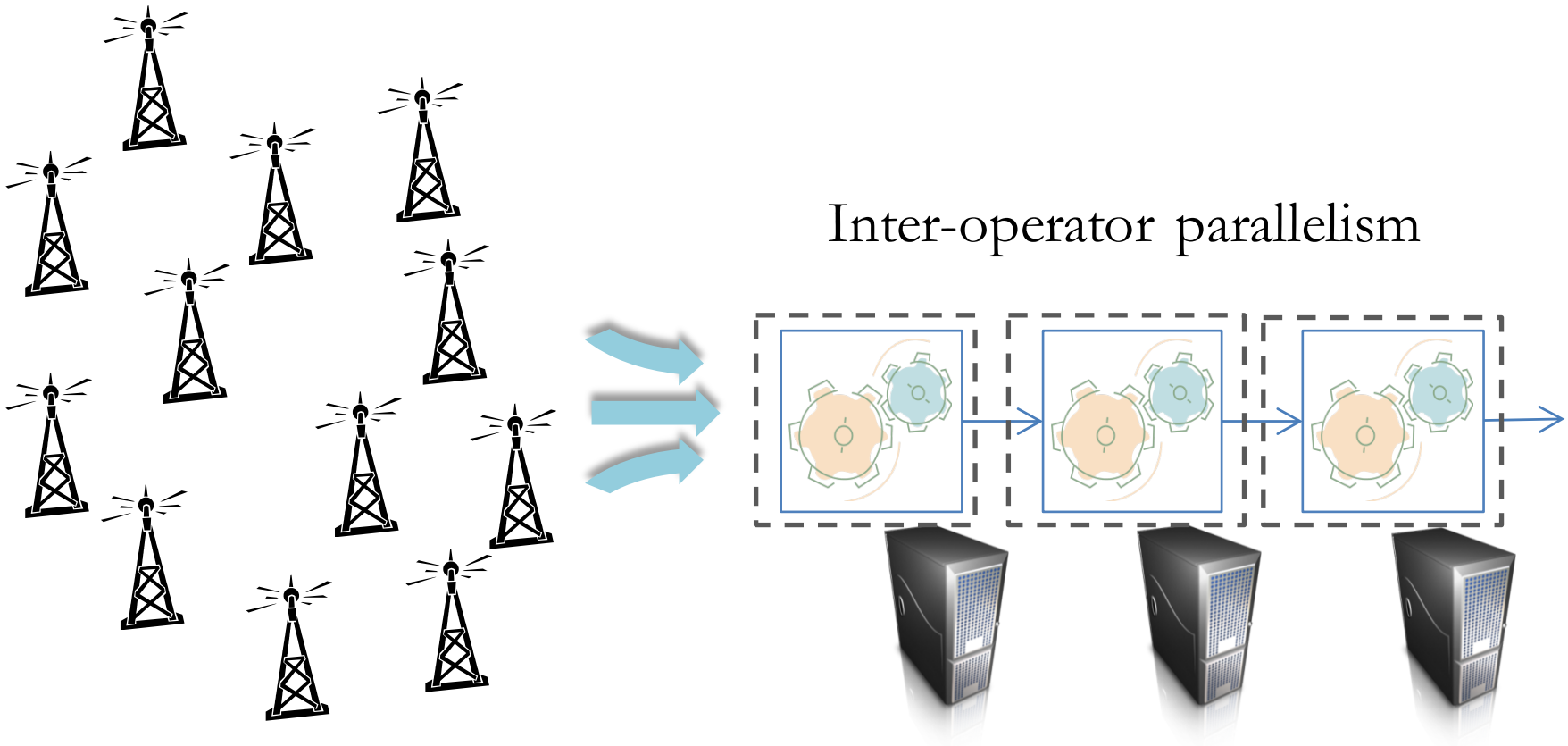
Agenda

- Introduction
  - Motivation
  - System Model
- Centralized SPEs
- Distributed SPEs
- Parallel-Distributed SPEs
- Elastic SPEs
- Conclusions

Centralized SPEs



Distributed SPEs



Distributed SPEs

- Challenges in distributed SPEs (among others)
  - Load Balancing
  - Fault Tolerance

Distributed SPEs

- Load Balancing: how to distribute operators to nodes in order to maximize throughput?
  - Offline load balancing
    - Load correlation between operators
  - Dynamic load balancing
    - Box sliding

Load correlation between operators

- Given two operators A-B, correlation in [-1,1]
  - Correlation -1 → burst in A load, decrease in B load
  - Correlation +1 → burst in A load, burst in B load

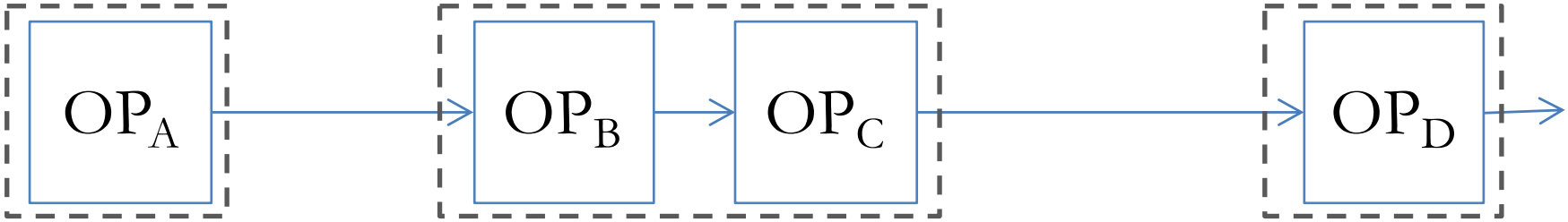
- Goal of the load balancing algorithm
  - Find the assignment of operators to nodes that maximizes the overall correlation
  - Burst in the system results in increased load in all the nodes

Box Sliding

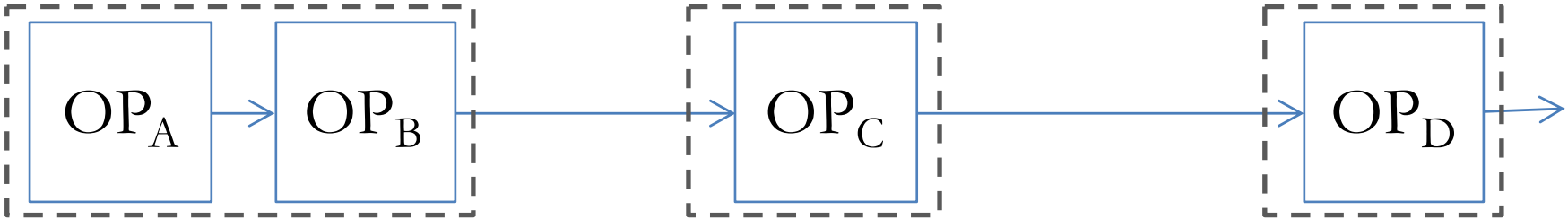
- Given selectivity(OP): $\frac{Output\ rate}{Input\ Rate} > 0$

- Map, Union $= 1$
- Filter < 1
- Aggregate < 1
- Join > 0

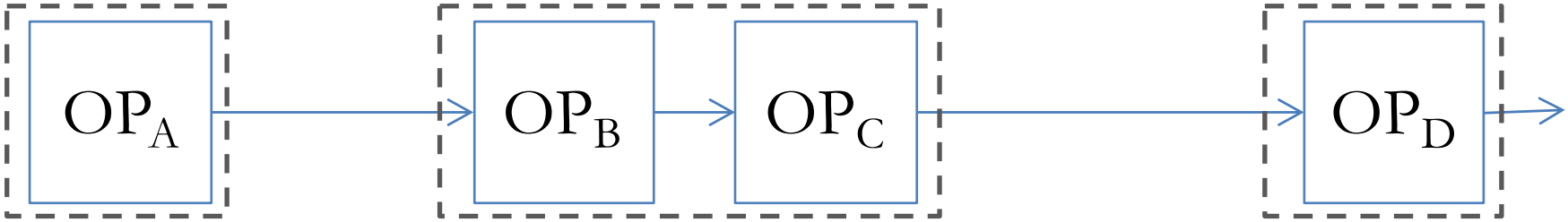
Box Sliding



If selectivity(OP_A) >> selectivity(OP_B)



Box Sliding



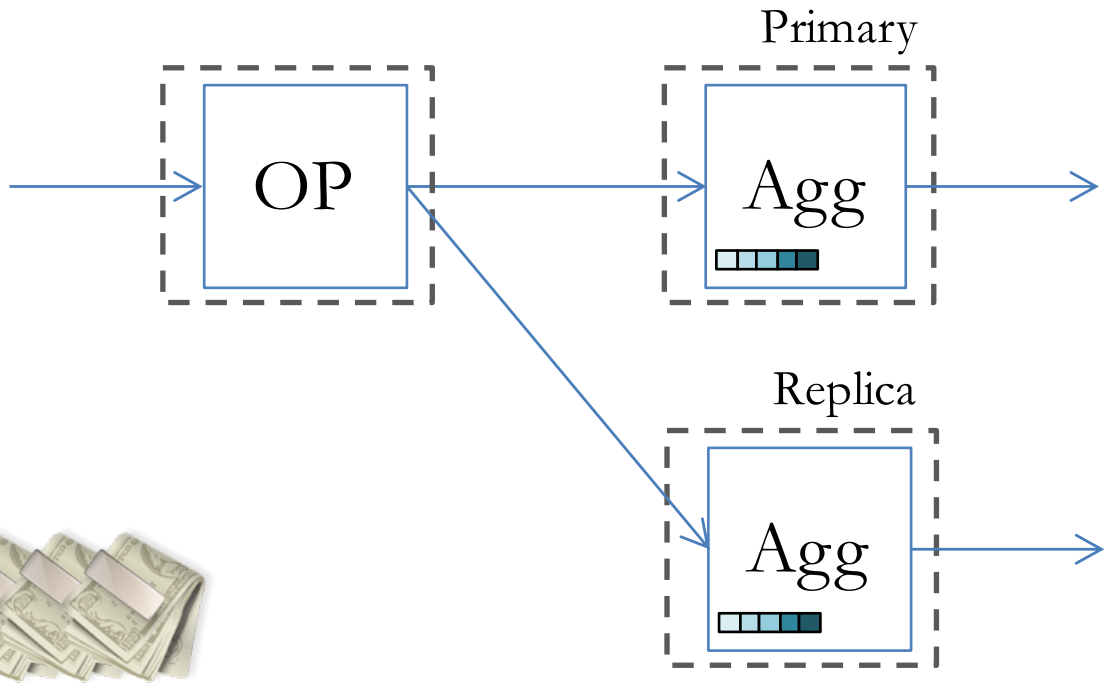
If selectivity(OP_C) >> selectivity(OP_B)

Distributed SPEs – Fault Tolerance

- Existing techniques:
  - Active standby
  - Passive standby
  - Upstream backup
- Guarantees:
  - Precise
  - Rollback recovery
  - Gap

Distributed SPEs – Fault Tolerance

- Active standby

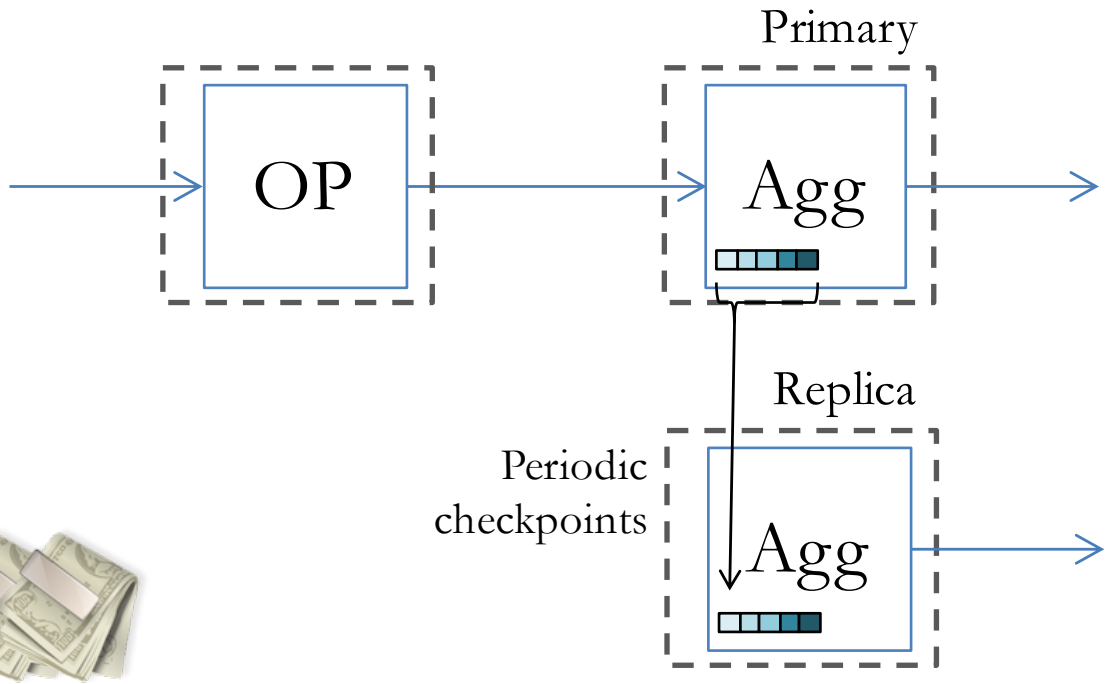


Cost:

Recovery Time:

Distributed SPEs – Fault Tolerance

- Passive standby

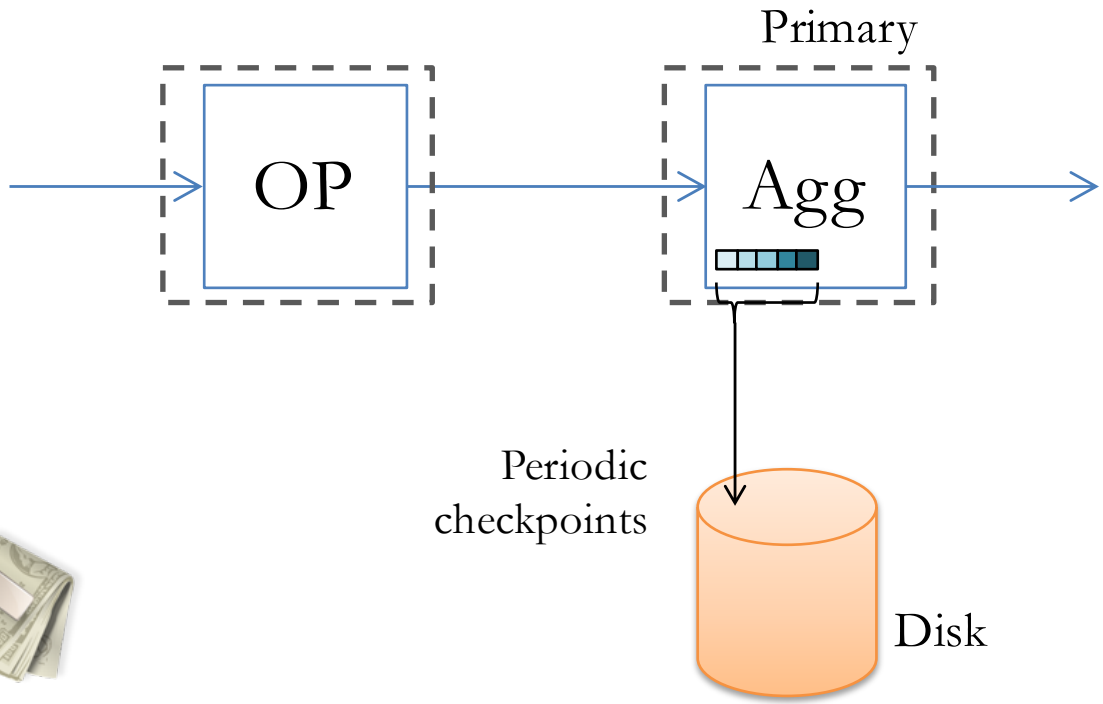


Cost:

Recovery
Time:

Distributed SPEs – Fault Tolerance

- Passive standby

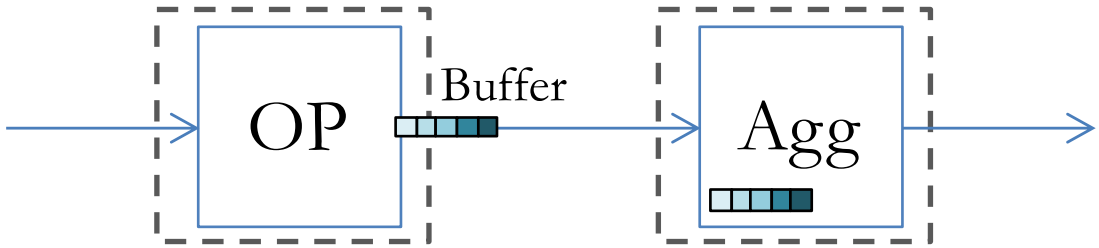


Cost:

Recovery
Time:

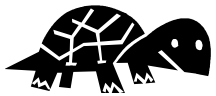
Distributed SPEs – Fault Tolerance

- Upstream Backup



Cost:

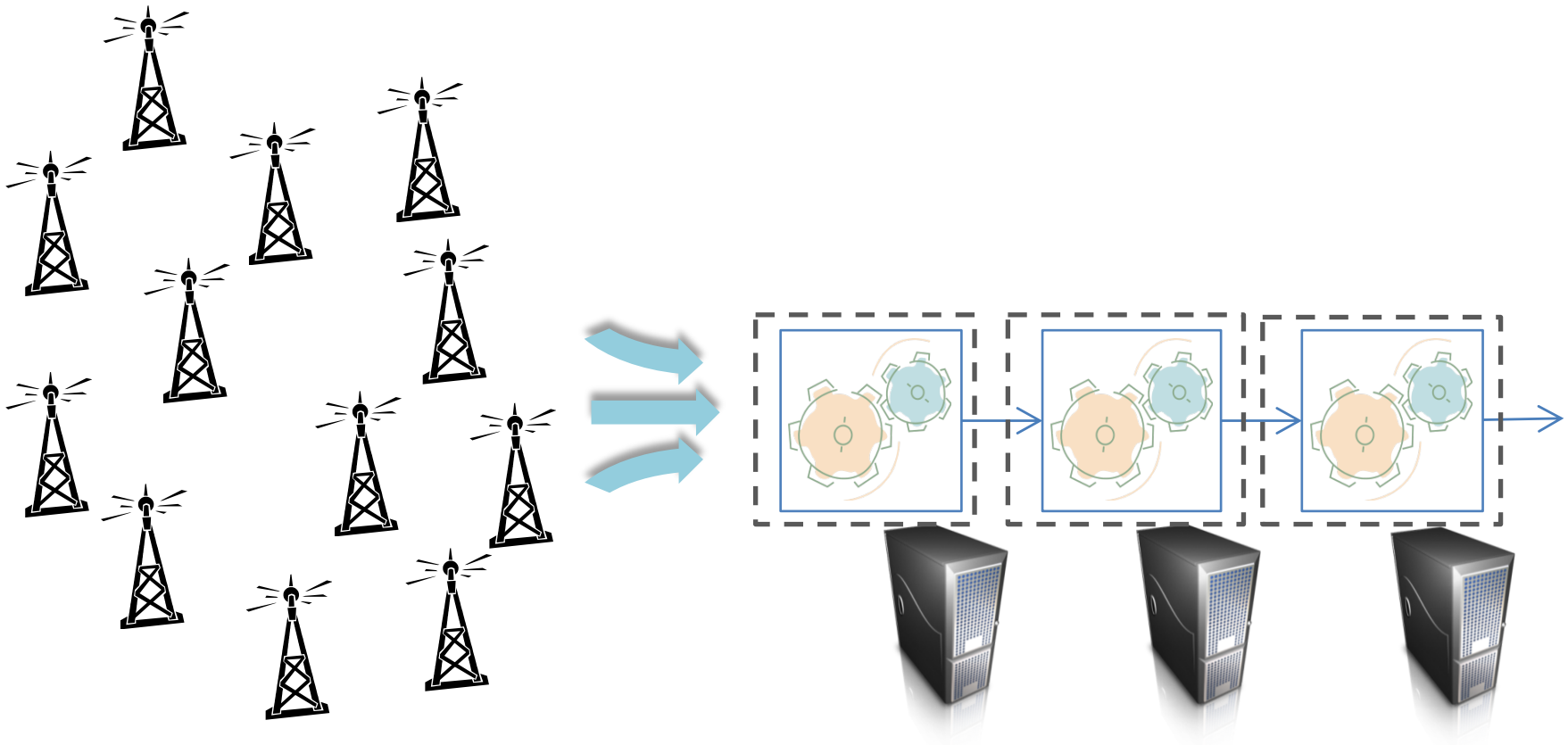
Recovery Time:



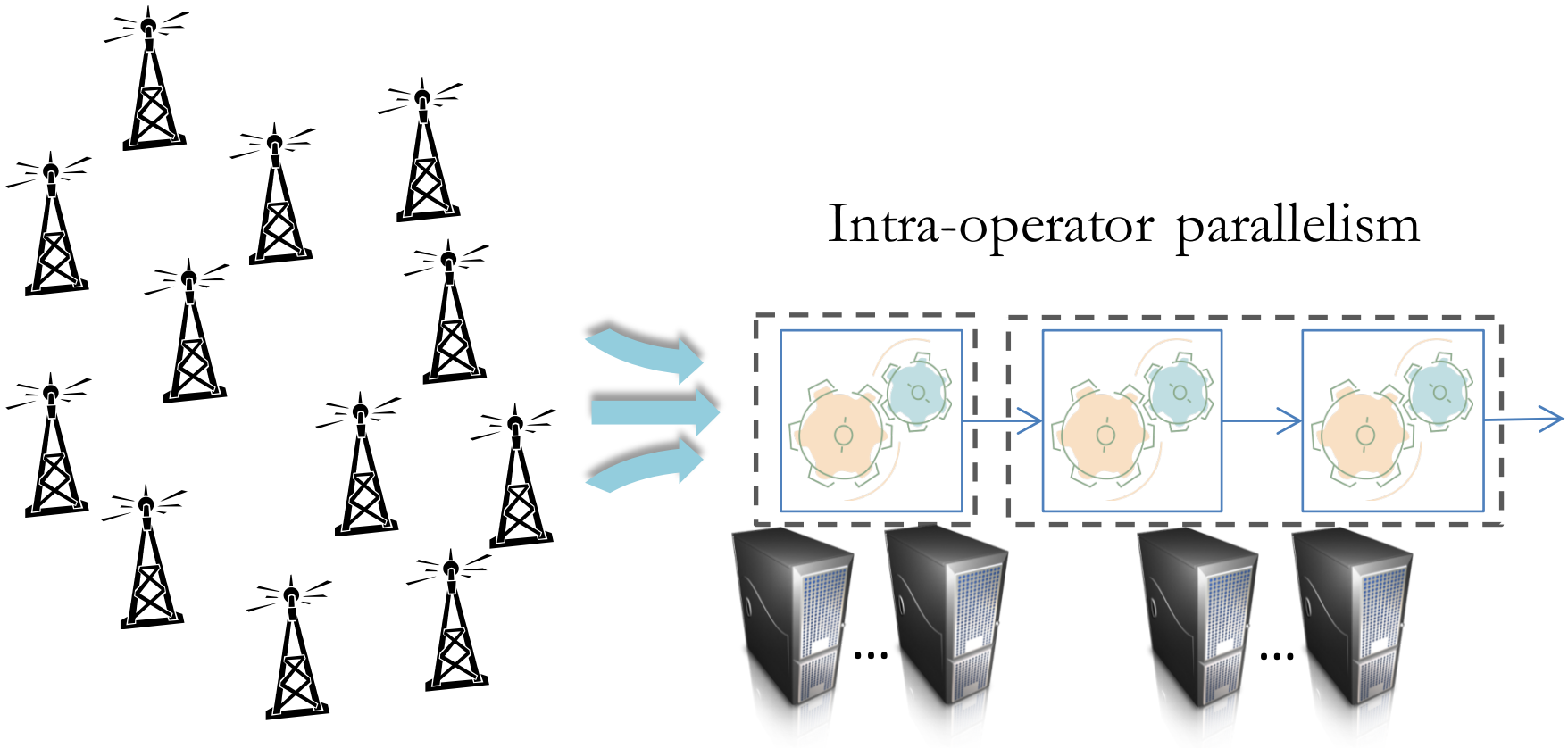
Agenda

- Introduction
  - Motivation
  - System Model
- Centralized SPEs
- Distributed SPEs
- Parallel-Distributed SPEs
- Elastic SPEs
- Conclusions

Distributed SPEs



Parallel SPEs



Parallel-distributed SPEs

- Parallel-distributed SPE
  - Deploy single data streaming operators at multiple SPE instances

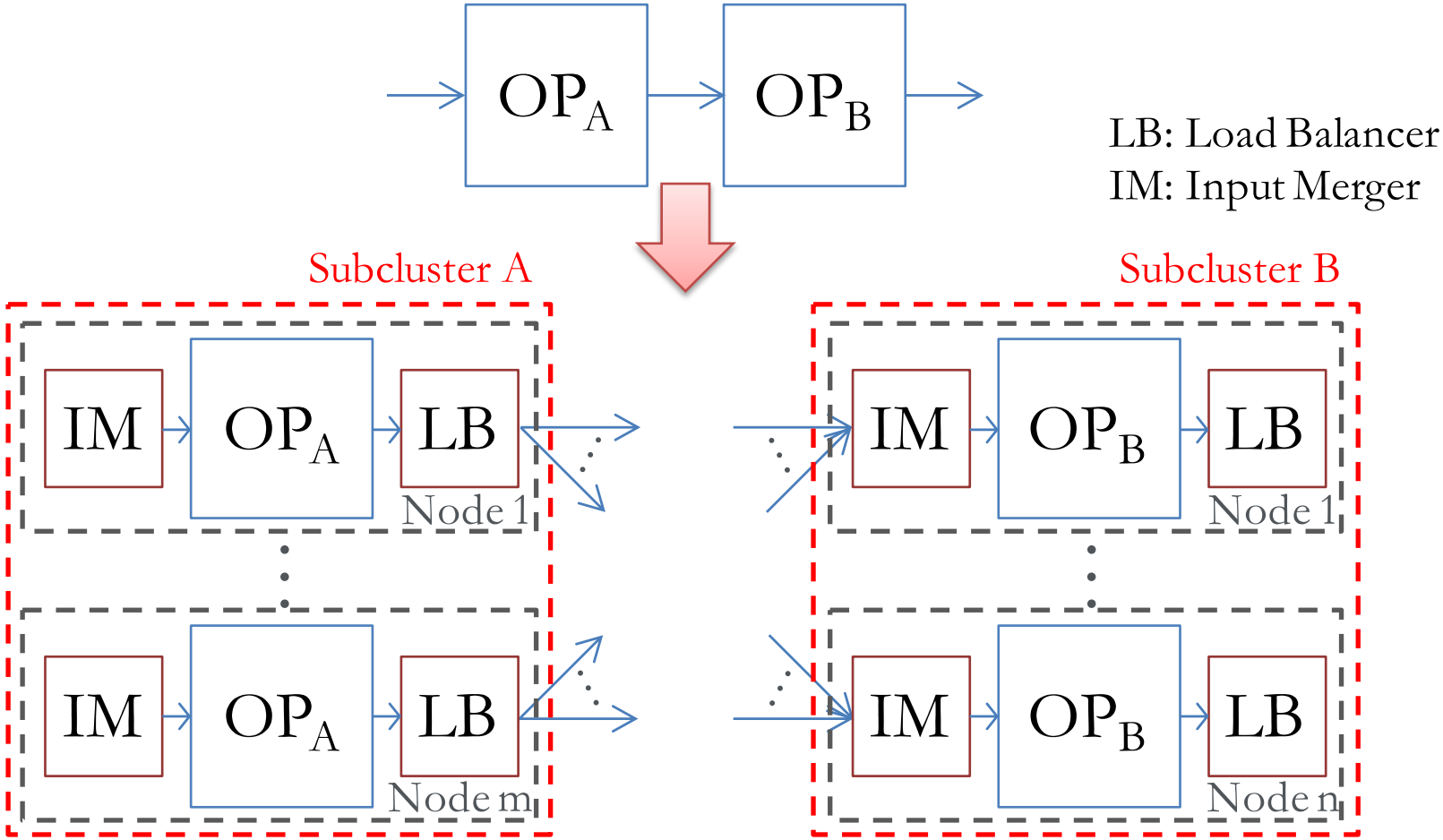
- Provided by state-of-the-art SPEs
  - Storm
  - Yahoo s4
  - StreamCloud ←

Parallel-Distributed SPEs

- Challenges:
  - Semantic Transparency
    - Results produced by parallel-distributed execution equal to centralized or distributed execution
  - Throughput maximization
    - How to maximize throughput when distributing and parallelizing data streaming operators?

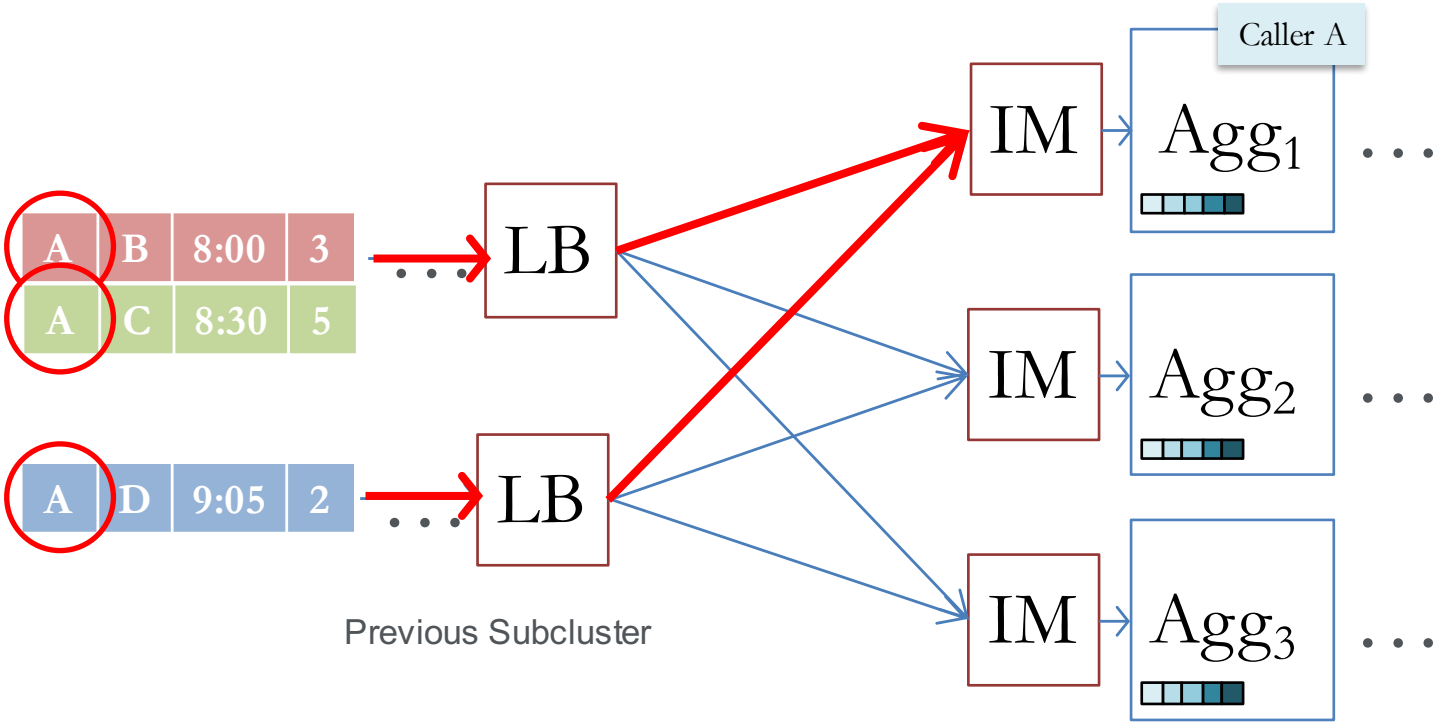
StreamCloud - Parallelization

- General approach



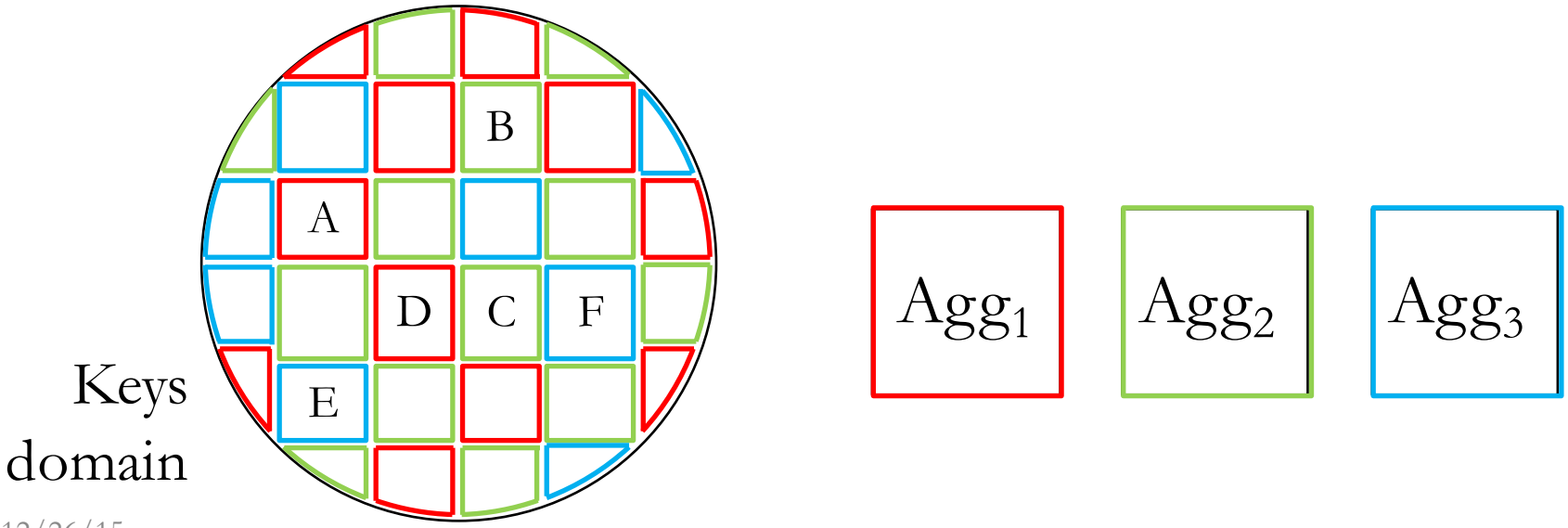
StreamCloud - Parallelization

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



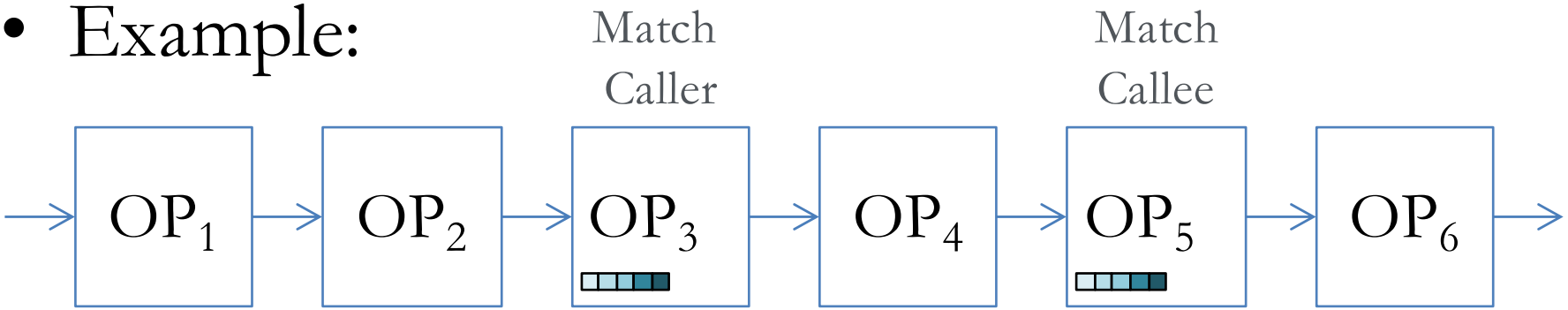
StreamCloud - Parallelization

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



StreamCloud - Parallelization

- Example:

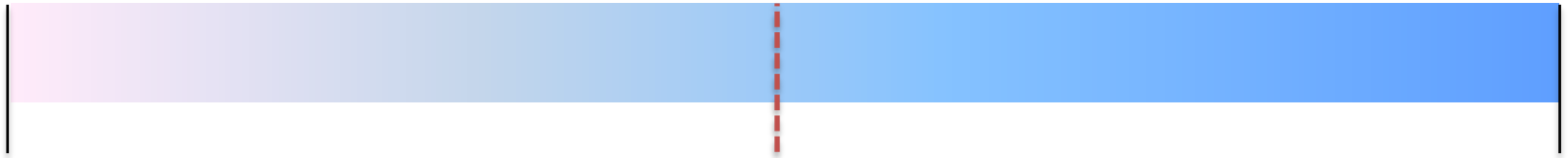


  - Cluster composed by 90 nodes

How to parallelize/distribute the query to maximize throughput?

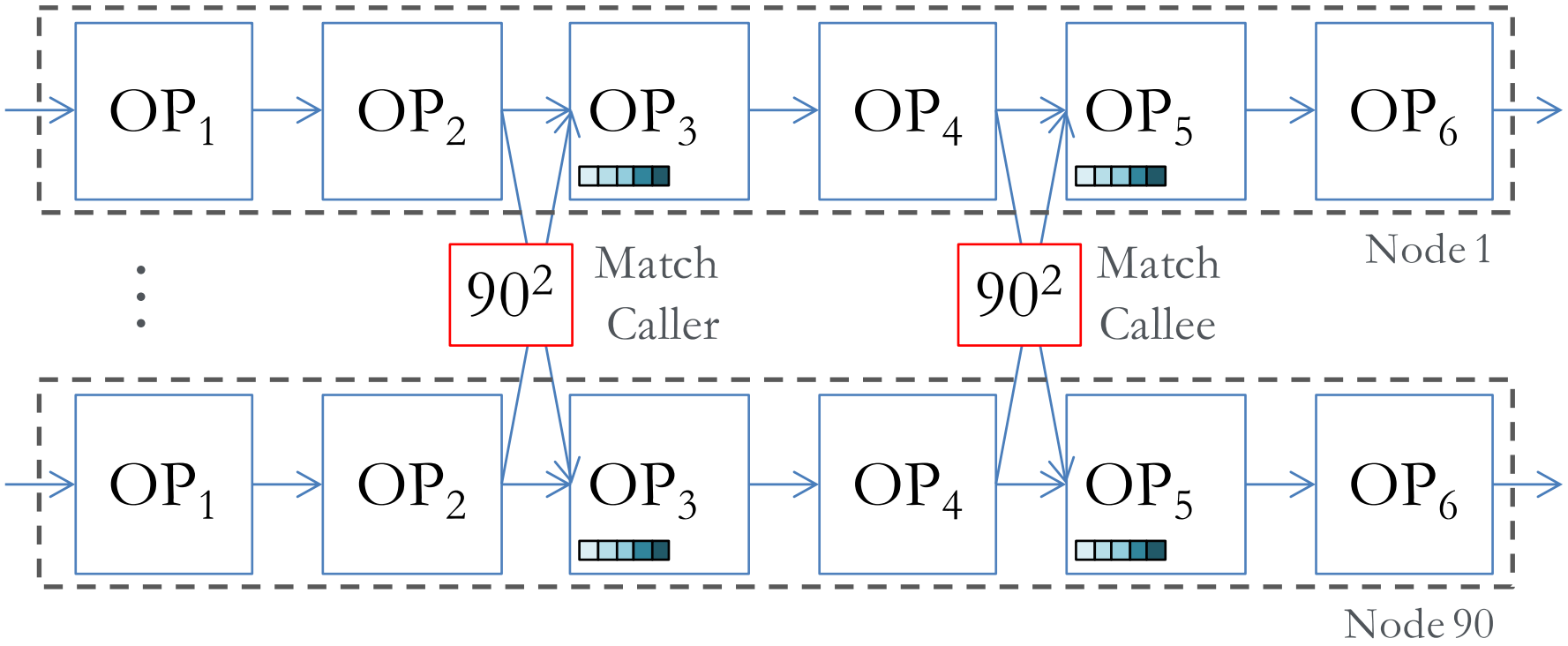
StreamCloud - Parallelization

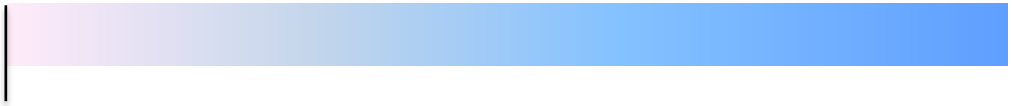
- Parallelization Strategies



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

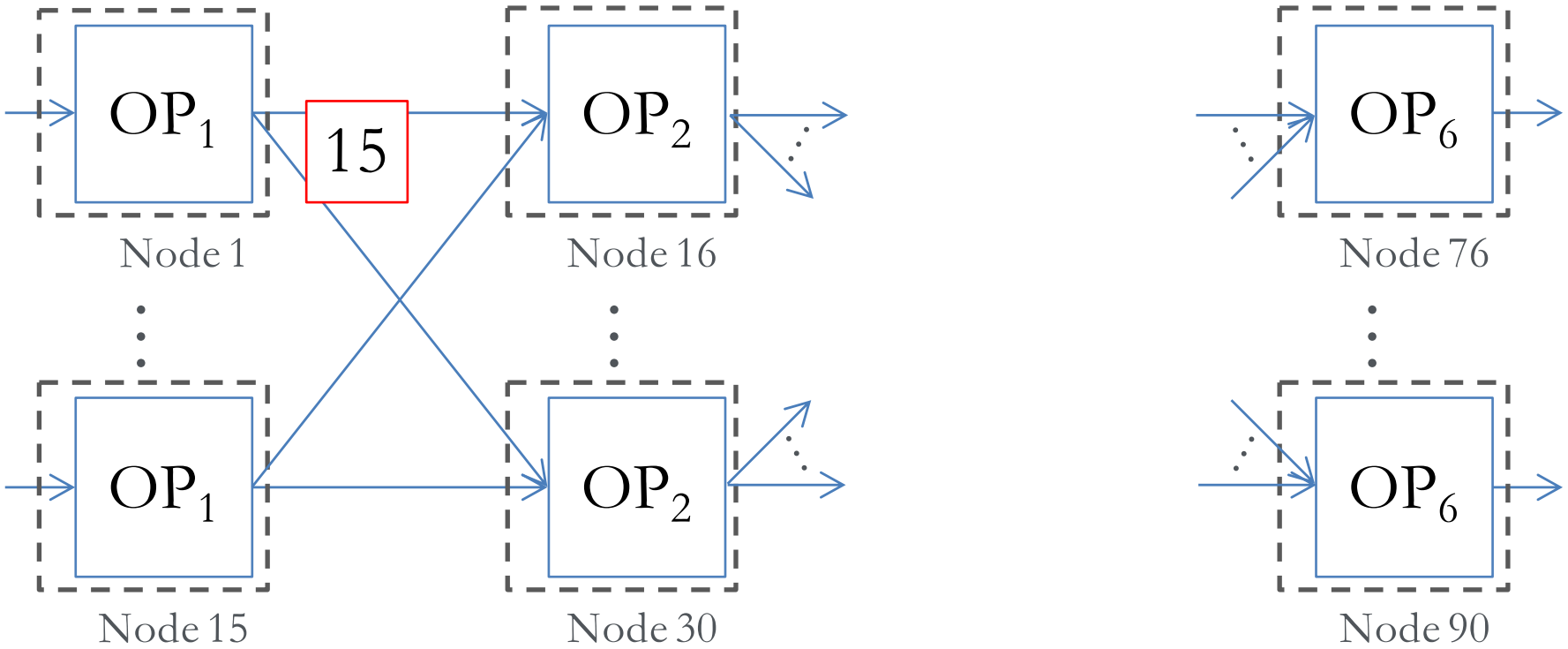
StreamCloud - Parallelization





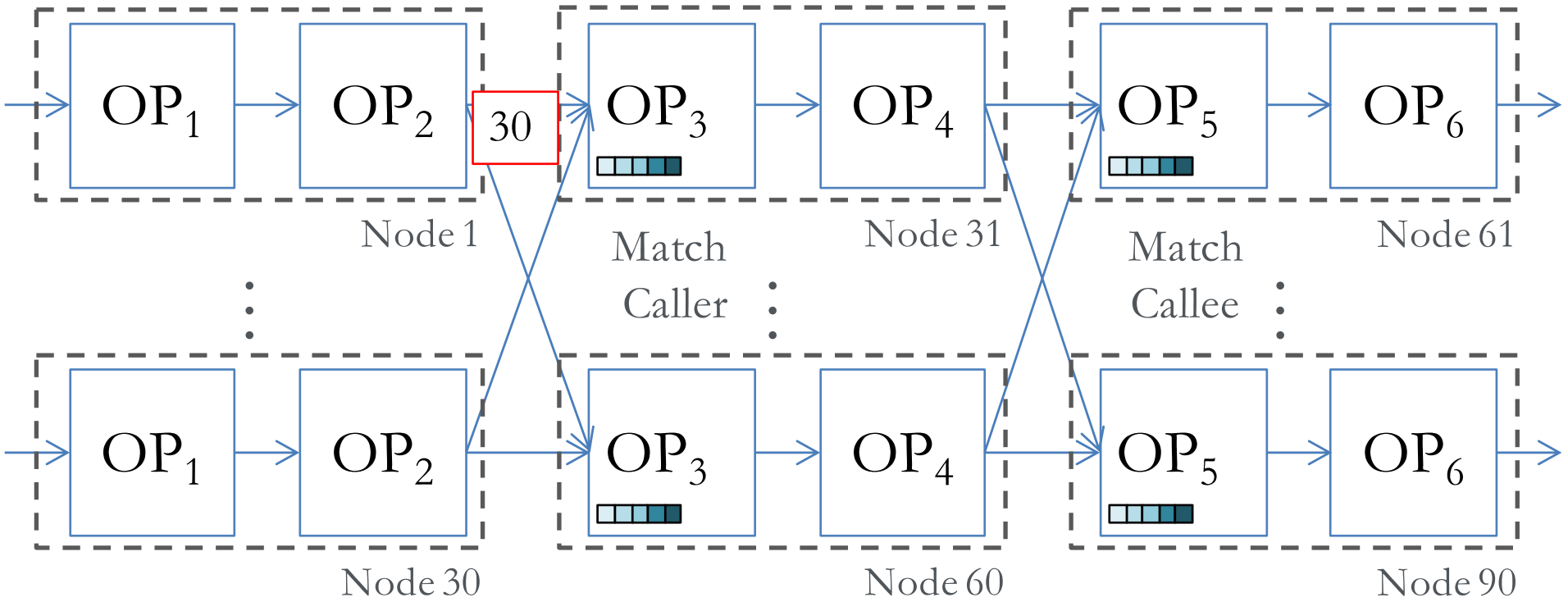
Full query at all nodes

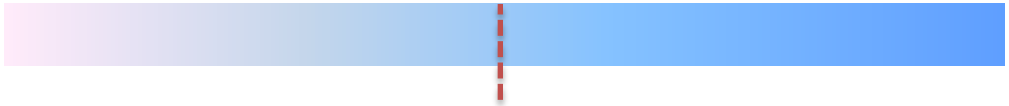
StreamCloud - Parallelization



1 operator per node

StreamCloud - Parallelization



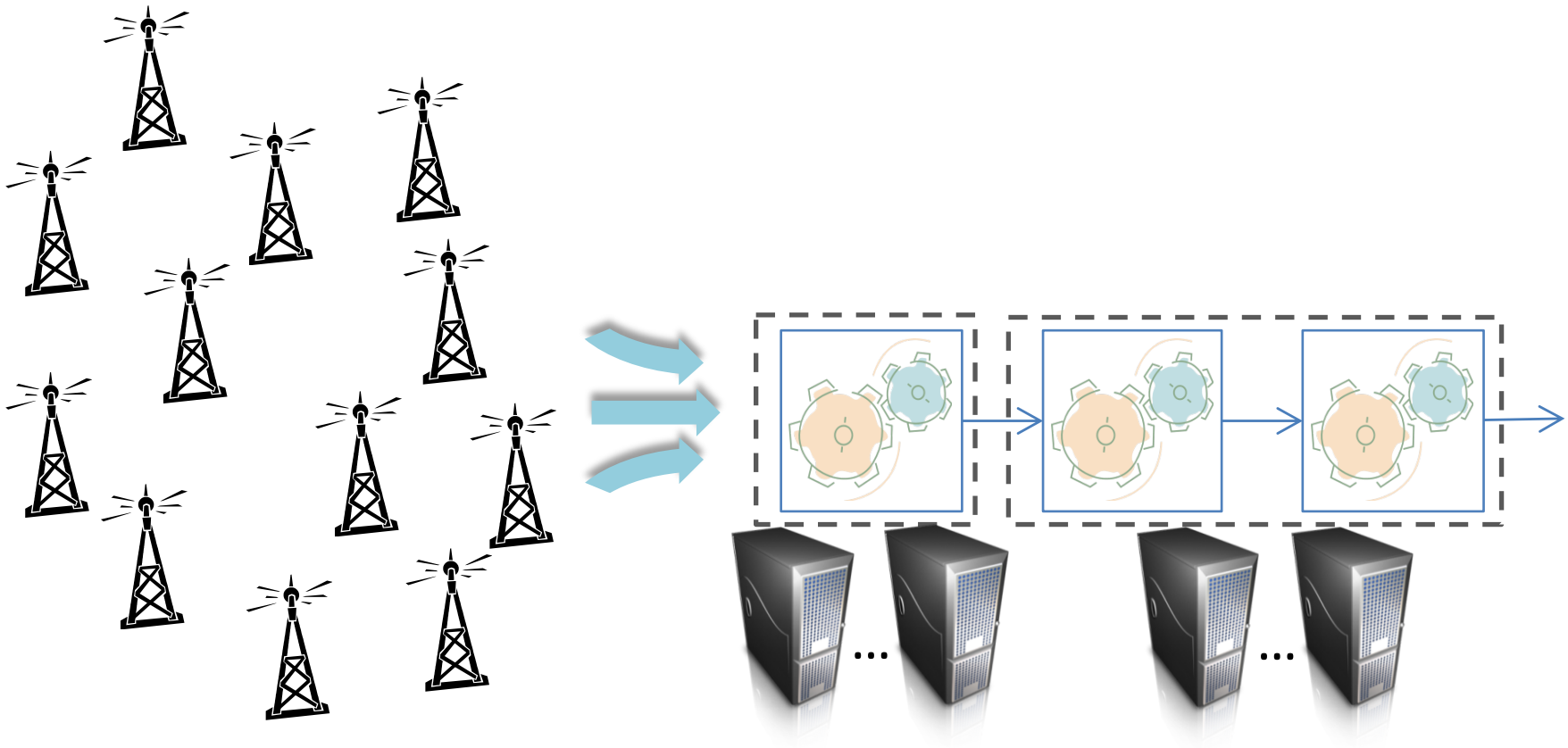


1+ operators per node

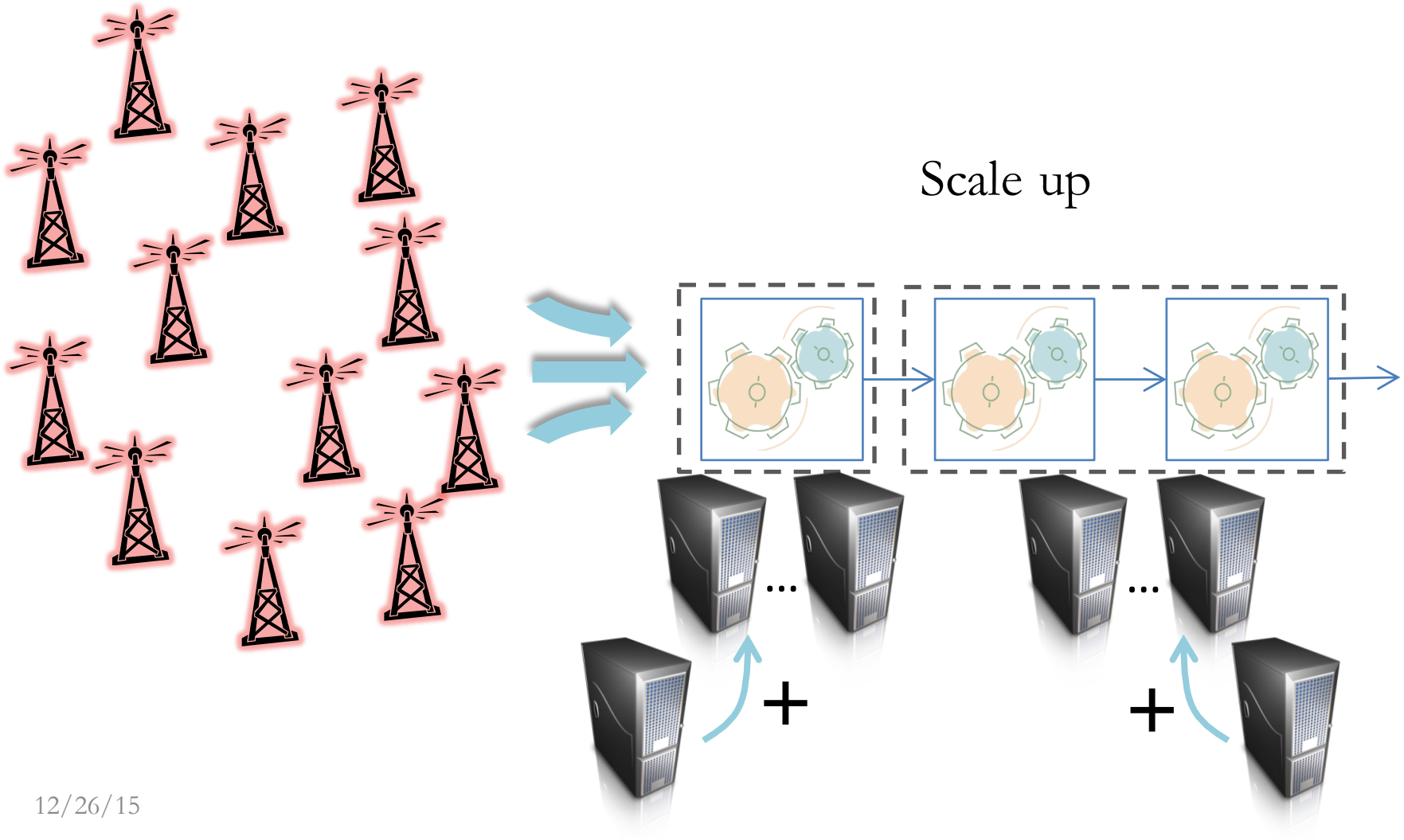
Agenda

- Introduction
  - Motivation
  - System Model
- Centralized SPEs
- Distributed SPEs
- Parallel-Distributed SPEs
- Elastic SPEs
- Conclusions

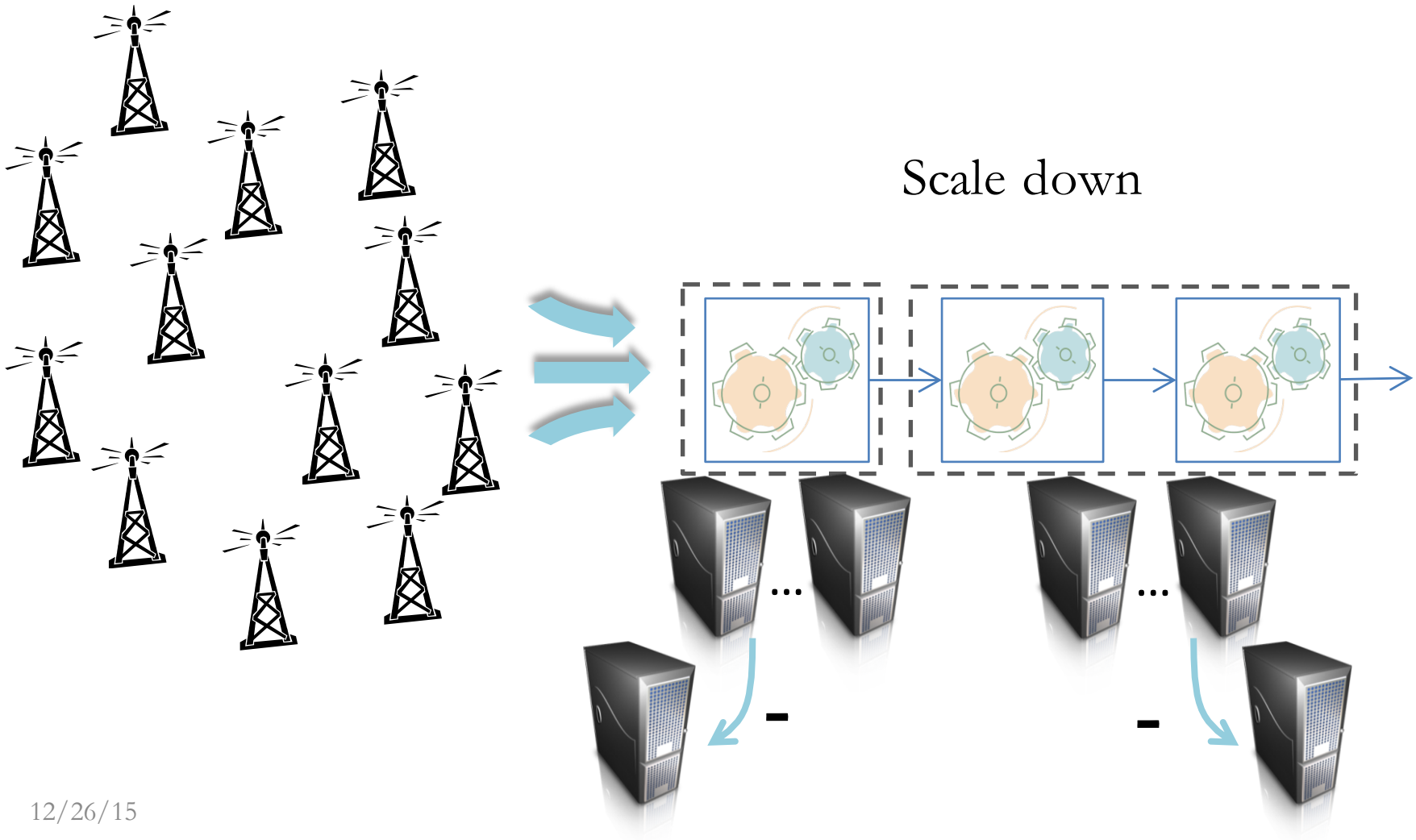
Parallel SPEs



Elastic SPEs



Elastic SPEs



Elastic SPEs

- Elastic SPEs and the Cloud environment

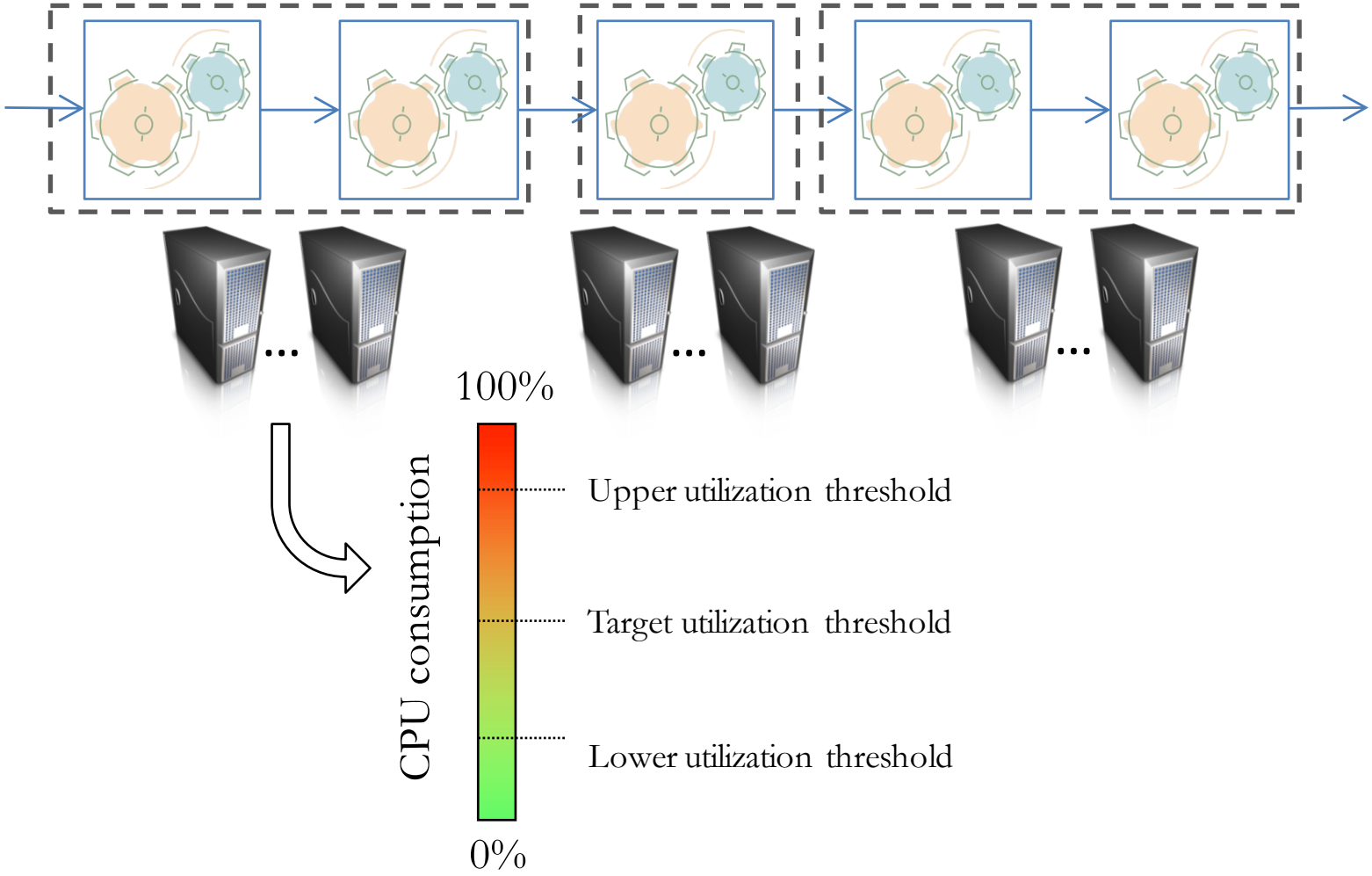
- Elasticity
  - Variable number of threads
  - Variable number of nodes, scaling with queries
  - Variable number of nodes, scaling with operators
    - StreamCloud

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



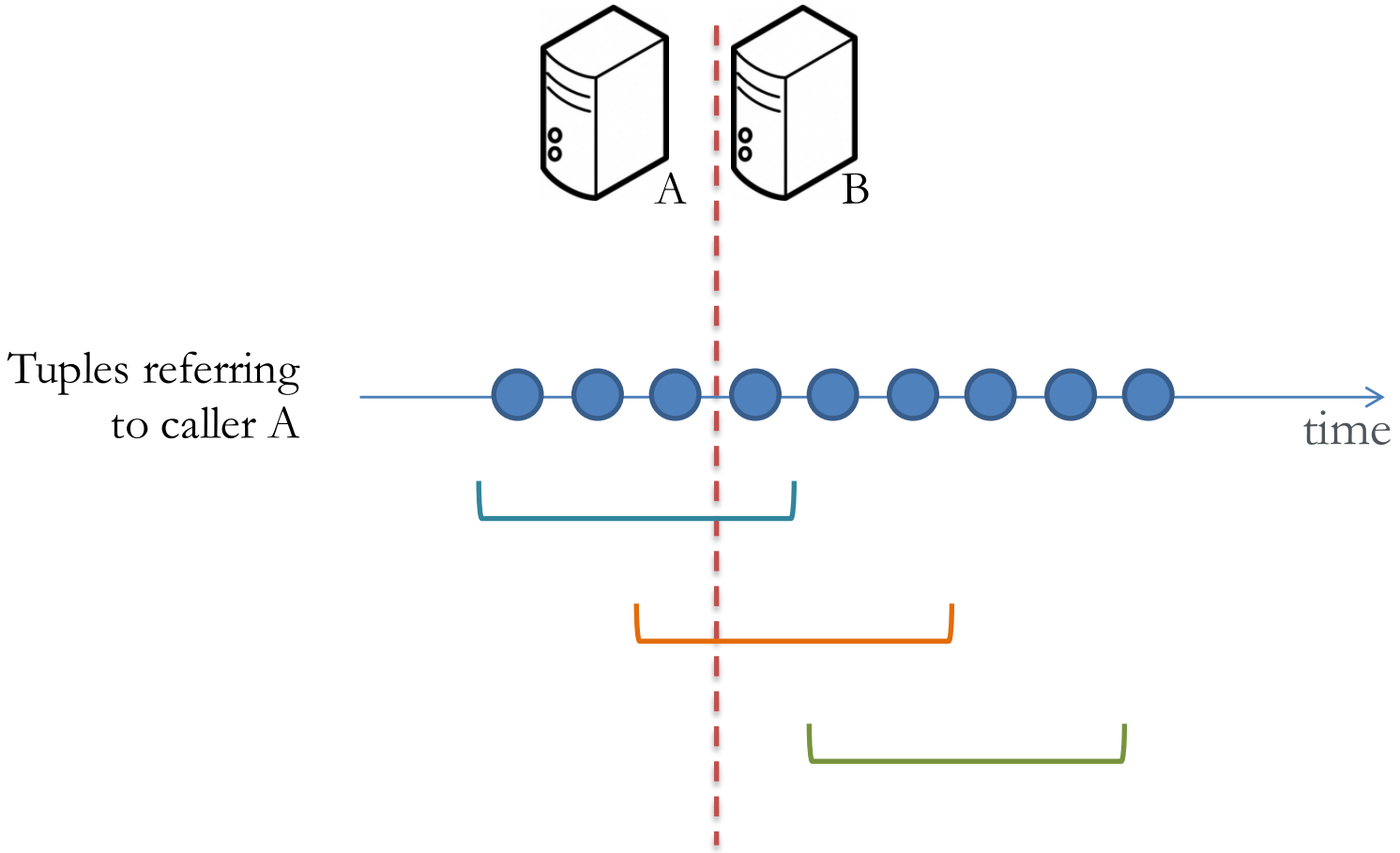
StreamCloud - Elasticity

- Transfer state between nodes
- Load balancing algorithm

- Minimum parallelization unit → bucket
  - Transfer buckets between instances

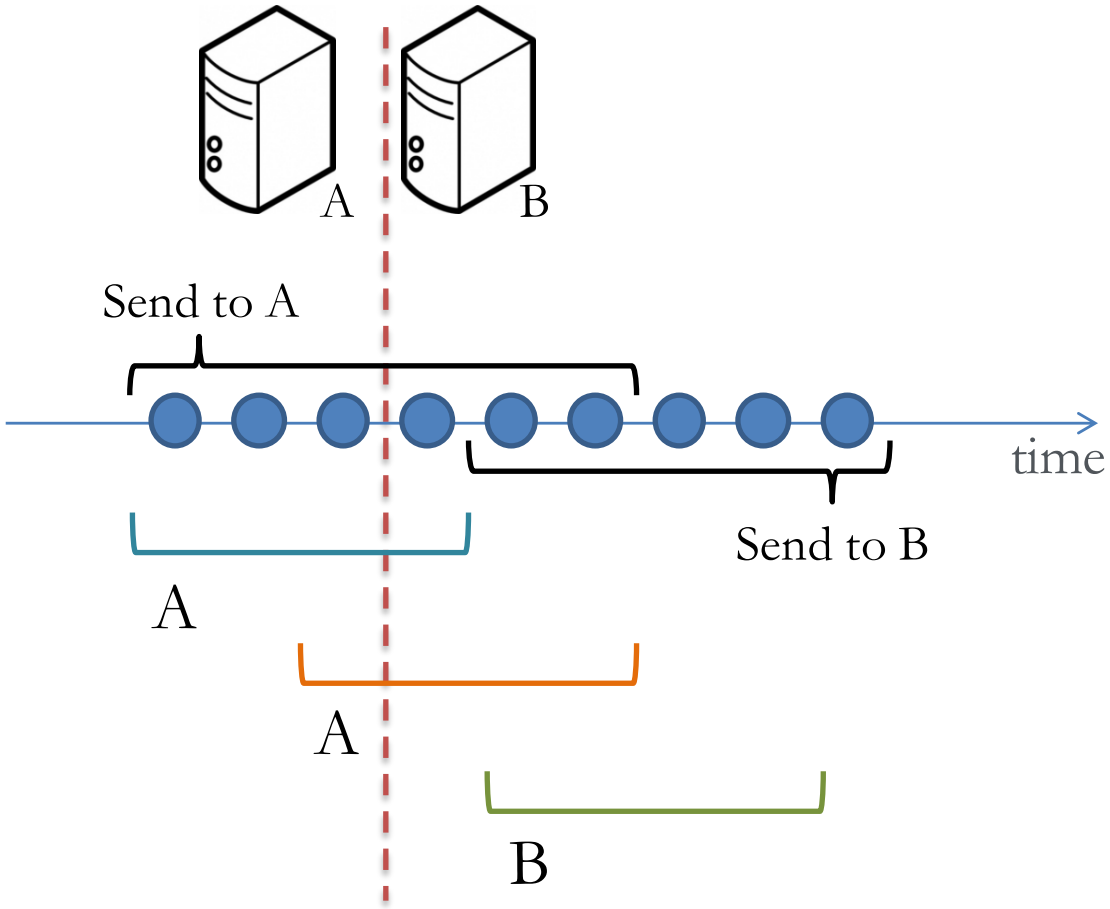
StreamCloud - Elasticity

- State transfer challenging for stateful operators



StreamCloud - Elasticity

- Window Recreation Protocol

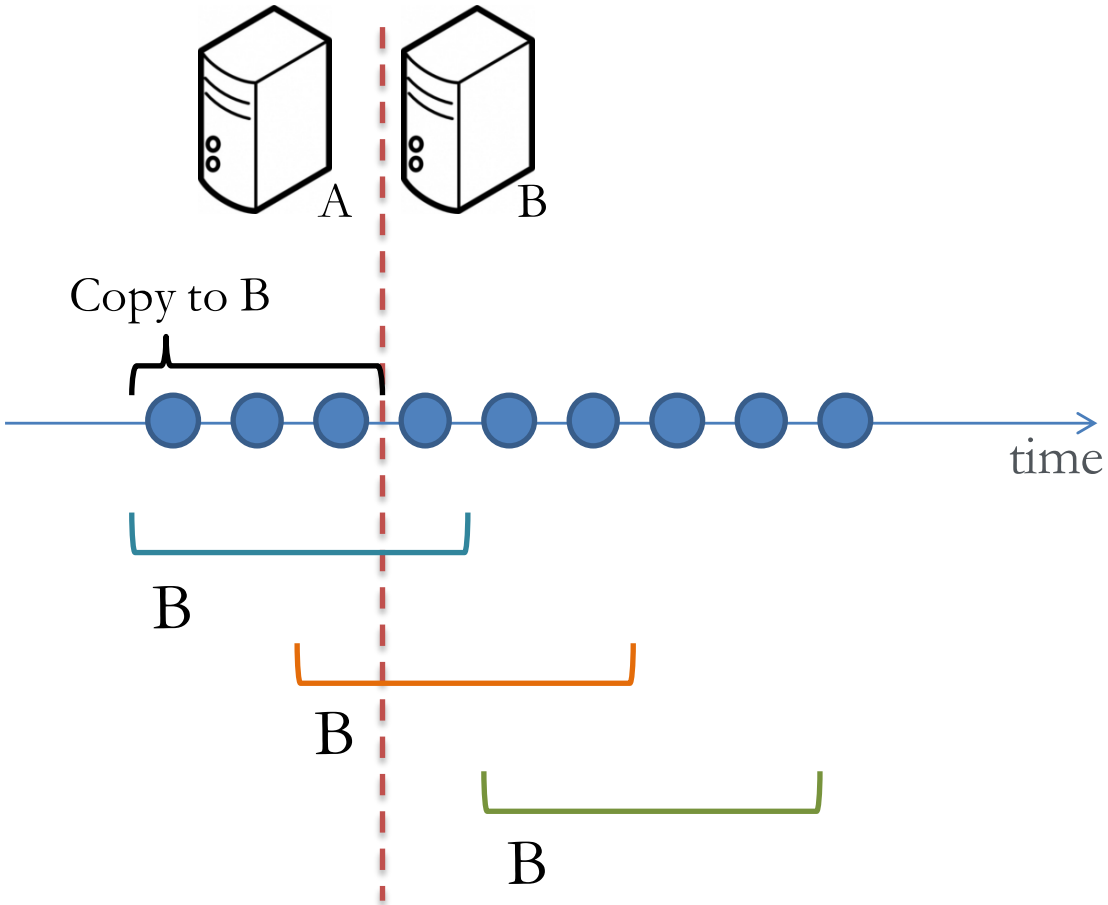


+ No communication between nodes

- Completion time proportional to window size

StreamCloud - Elasticity

- State Recreation Protocol



+ Minimizes completion time

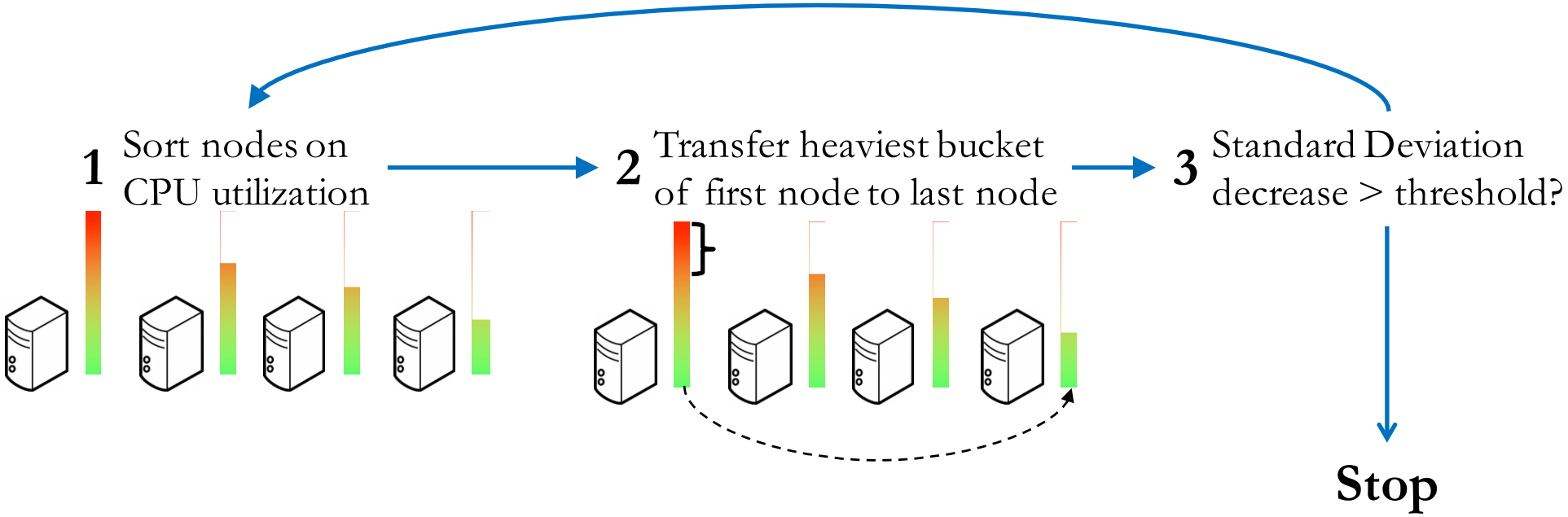
- Communication between nodes

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

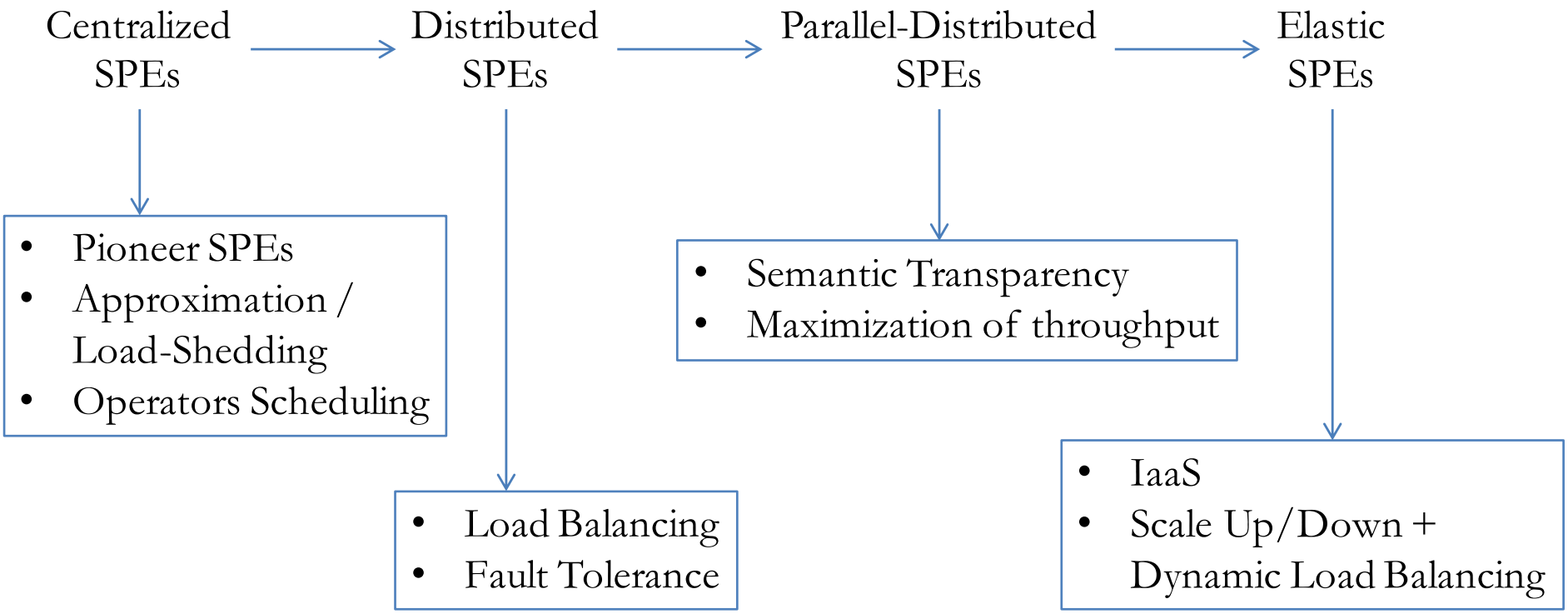
- Greedy algorithm:
  - Balance load → Minimized CPU standard deviation



Agenda

- Introduction
  - Motivation
  - System Model
- Centralized SPEs
- Distributed SPEs
- Parallel-Distributed SPEs
- Elastic SPEs
- Conclusions

Conclusions



http://www.cse.chalmers.se/~vinmas

Bibliography

- Motivation / System Model

1. Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. Models and issues in data stream systems. In Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, PODS '02, New York, NY, USA, 2002. ACM.
2. Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. Models and issues in data stream systems. In Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, PODS '02, New York, NY, USA, 2002. ACM.
3. Michael Stonebraker, Uˇgur Çetintemel, and Stan Zdonik. The 8 requirements of realtime stream processing. SIGMOD Rec., 34(4), December 2005.
4. Nesime Tatbul. QoS-Driven load shedding on data streams. In Proceedings of the Worshops XMLDM, MDDE, and YRWS on XML-Based Data Management and Multimedia Engineering-Revised Papers, EDBT '02, London, UK, UK, 2002. Springer-Verlag.

Bibliography

- Centralized SPEs

1. Arvind Arasu, Brian Babcock, Shivnath Babu, John Cieslewicz, Keith Ito, Rajeev Motwani, Utkarsh Srivastava, and Jennifer Widom. Stream: The stanford data stream management system. Springer, 2004.
2. Arvind Arasu, Shivnath Babu, and Jennifer Widom. The CQL continuous query language: semantic foundations and query execution. The VLDB Journal, 15(2), June 2006.
3. Daniel J. Abadi, Don Carney, Ugur Cetintemel, Mitch Cherniack, Christian Convey, Sangdon Lee, Michael Stonebraker, Nesime Tatbul, and Stan Zdonik. Aurora: a new model and architecture for data stream management. The VLDB Journal, 12(2), August 2003.
4. Nesime Tatbul and Stan Zdonik. Window-aware load shedding for aggregation queries over data streams. In Proceedings of the 32nd international conference on Very large data bases, VLDB '06. VLDB Endowment, 2006.

Bibliography

- Distributed SPEs

1. Daniel J. Abadi, Yanif Ahmad, Magdalena Balazinska, Ugur Çetintemel, Mitch Cherniack, Jeong-Hyon Hwang, Wolfgang Lindner, Anurag Maskey, Alex Rasin, Esther Ryvkina, Nesime Tatbul, Ying Xing, and Stanley B. Zdonik. The design of the borealis stream processing engine. In CIDR, pages 277–289, 2005.
2. Magdalena Balazinska, Hari Balakrishnan, Samuel R Madden, and Michael Stonebraker. Fault-tolerance in the borealis distributed stream processing system. ACM Trans. Database Syst., 33(1), March 2008. ACM ID: 1331907.
3. Philippe Bonnet, Johannes Gehrke, and Praveen Seshadri. Towards sensor database systems. In Proceedings of the Second International Conference on Mobile Data Management, MDM '01, London, UK, UK, 2001. Springer-Verlag.
4. Jeong-hyon Hwang, Magdalena Balazinska, Alexander Rasin, Ugur Cetintemel, Michael Stonebraker, and Stan Zdonik. A comparison of stream-oriented highavailability algorithms. Technical report, Brown CS, 2003.
5. Jeong-Hyon Hwang, Magdalena Balazinska, Alexander Rasin, U˘gur Çetintemel, Michael Stonebraker, and Stan Zdonik. High-Availability algorithms for distributed stream processing. In Data Engineering, International Conference on, volume 0, Los Alamitos, CA, USA, 2005. IEEE Computer Society.

Bibliography

- Parallel SPEs

1. Vincenzo Gulisano, Ricardo Jiménez-Peris, Marta Patiño-Martínez, and Patrick Valduriez. Streamcloud: A large scale data streaming system. In ICDCS 2010: International Conference on Distributed Computing Systems, pages 126–137, June 2010.
2. Mehul Shah Joseph, Joseph M. Hellerstein, Sirish Ch, and Michael J. Franklin. Flux: An adaptive partitioning operator for continuous query systems. In In ICDE, 2002.

Bibliography

- Elastic SPEs

1. Vincenzo Gulisano, Ricardo Jimenez-Peris, Marta Patino-Martinez, Claudio Soriente, and Patrick Valduriez. Streamcloud: An elastic and scalable data streaming system. IEEE Transactions on Parallel and Distributed Systems, 99(PrePrints), 2012.
2. Thomas Heinze. Elastic complex event processing. In Proceedings of the 8th Middleware Doctoral Symposium, MDS '11, New York, NY, USA, 2011. ACM.
3. Simon Loesing, Martin Hentschel, Tim Kraska, and Donald Kossmann. Stormy: an elastic and highly available streaming service in the cloud. In Proceedings of the 2012 Joint EDBT/ICDT Workshops, EDBT-ICDT '12, New York, NY, USA, 2012. ACM.
4. Scott Schneider, Henrique Andrade, Bugra Gedik, Alain Biem, and Kun-Lung Wu. Elastic scaling of data parallel operators in stream processing. In Proceedings of the 2009 IEEE International Symposium on Parallel&Distributed Processing, IPDPS '09, Washington, DC, USA, 2009. IEEE Computer Society.