Systems Infrastructure for Data Science

Web Science Group
Uni Freiburg
WS 2012/13

Data Stream Processing

Today’s Topic

- **Stream Processing**
  - Model Issues
  - System Issues
  - **Distributed Processing Issues**

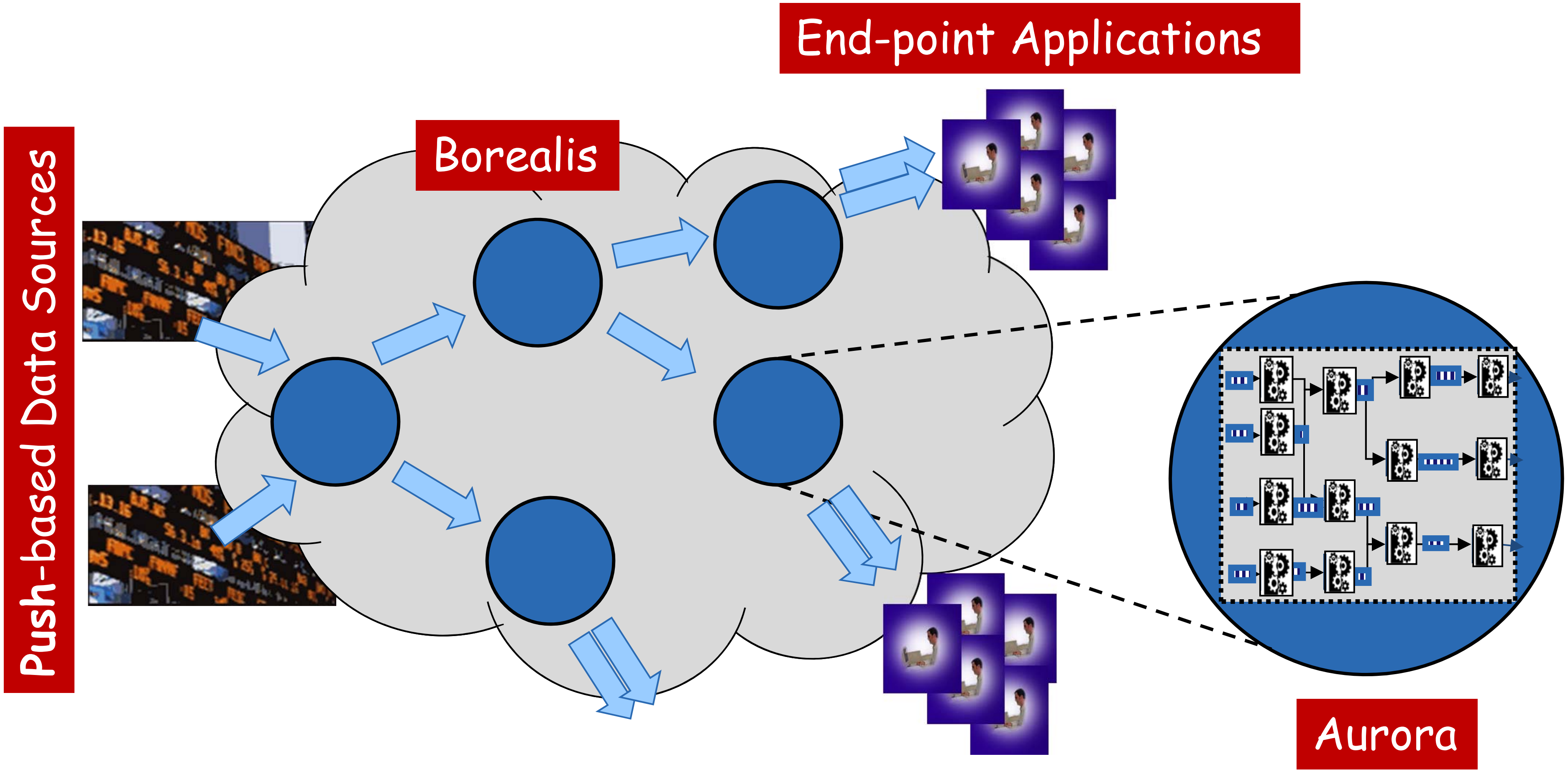
Distributed Stream Processing
Motivation

- Distributed data sources
- Performance and Scalability
- High availability and Fault tolerance

Design Options for Distributed DSMS

- Almost same split as with distributed databases vs cloud databases
- Currently, most of the work is on fairly tightly coupled, strongly maintained distributed DSMS
- We will study a number of general/traditional approaches for most of the lecture, look at some ideas for cloud-based streaming
- As usual, distributed processing is about tradeoffs!

Distributed Stream Processing
Borealis Example



Distributed Stream Processing
Major Problem Areas

- Load distribution and balancing
  - Dynamic / Correlation-based techniques
  - Static / Load-resilient techniques
  - (Network-aware techniques)
- Distributed load shedding
- High availability and Fault tolerance
  - Handling node failures
  - Handling link failures (esp. network partitions)

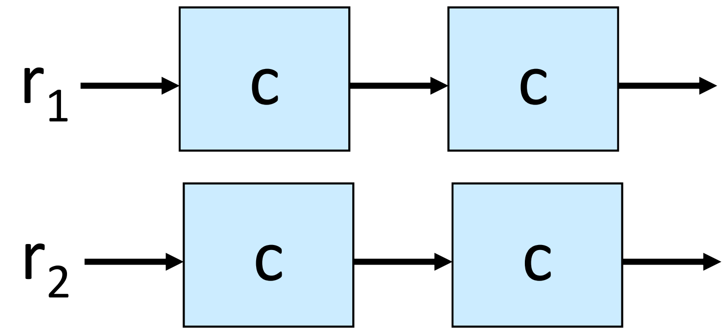
Load Distribution

- Goal: to distribute a given set of continuous query operators onto multiple stream processing server nodes
- What makes an operator distribution good?
  - Load balance across nodes
  - Resiliency to load variations
  - Low operator migration overhead
  - Low network bandwidth usage

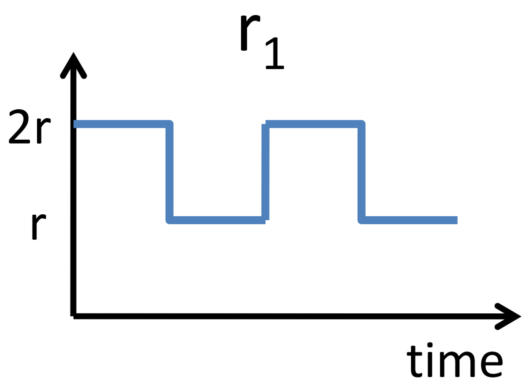
Correlation-based Techniques

- Goals:
  - to minimize end-to-end query processing latency
  - to balance load across nodes to avoid overload
- Key ideas:
  - Group boxes with small load correlation together
    - ⇒ helps minimize the overall load variance on that node
    - ⇒ keeps the node load steady as input rates change
  - Maximize load correlation among nodes
    - ⇒ helps minimize the need for load migration

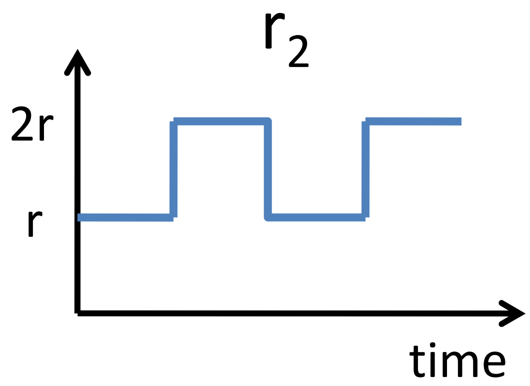
Example



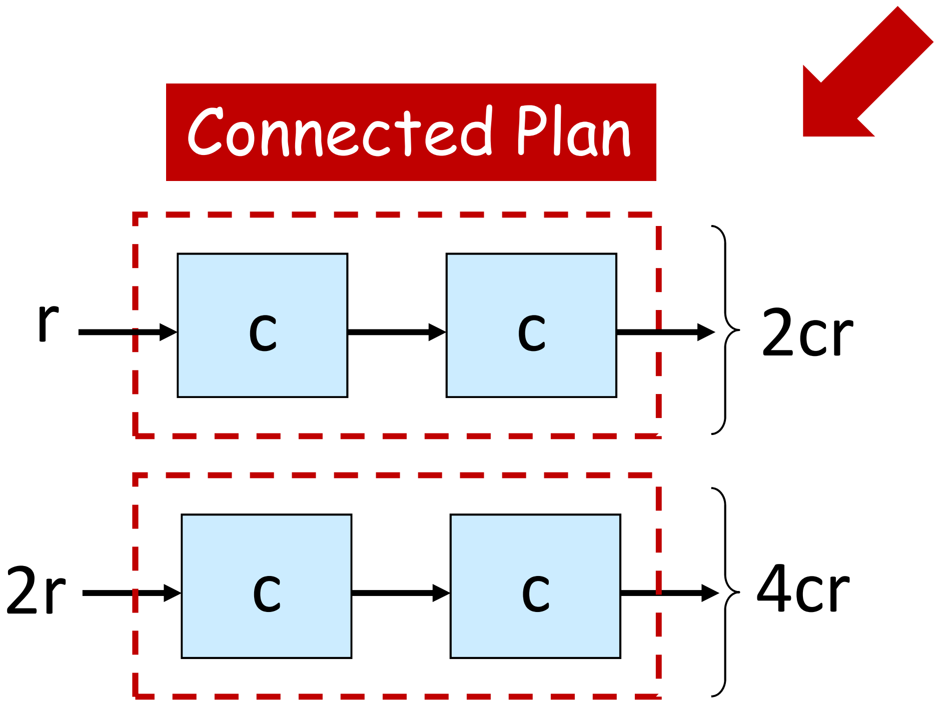
r_1
c
c
r_2
c
c



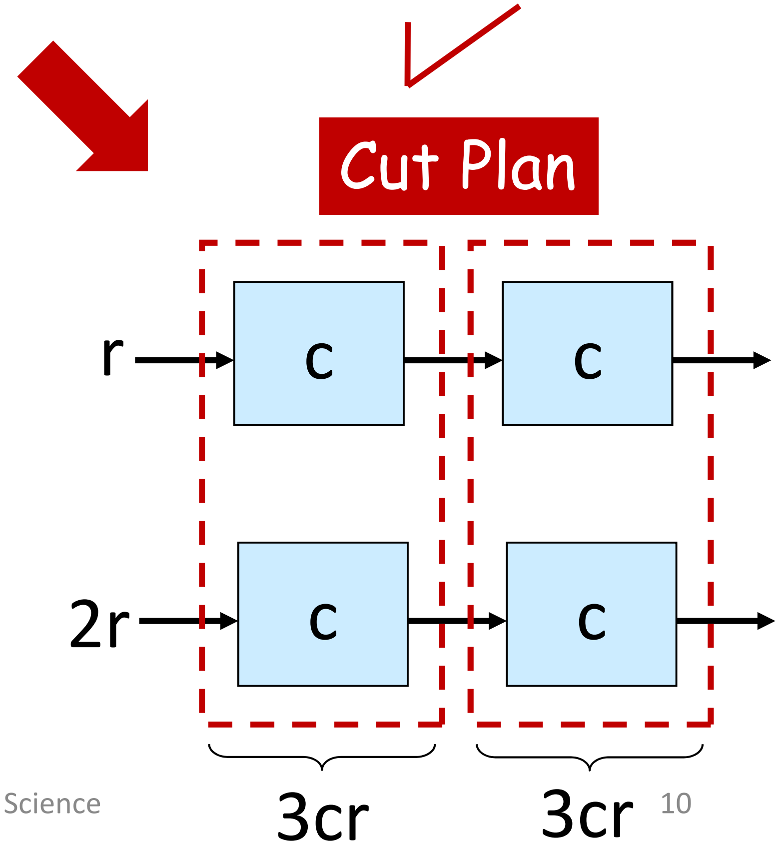
r_1
2r
r
time



r_2
2r
r
time

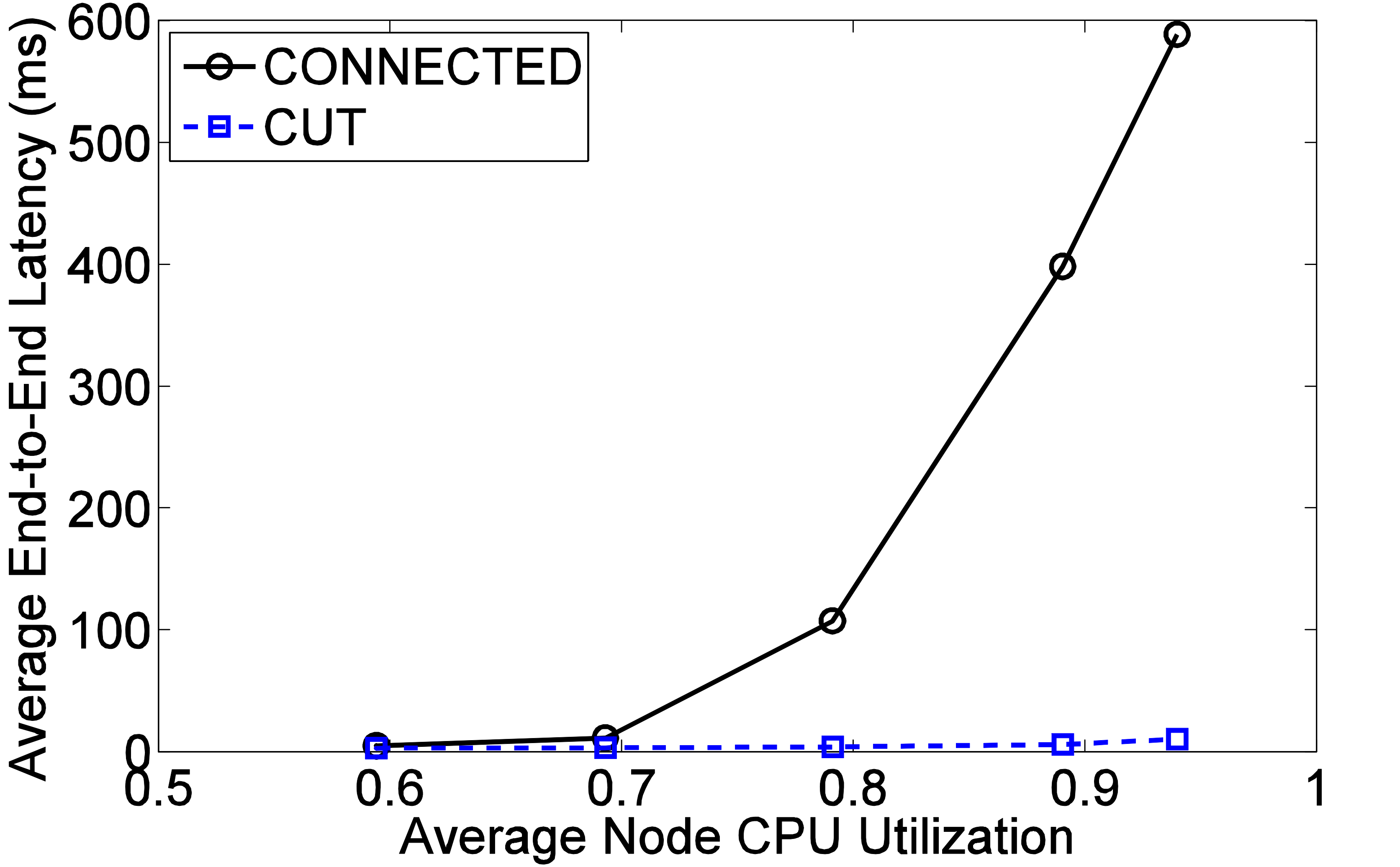


Connected Plan
r
c
c
2cr
2r
c
c
4cr



Cut Plan
r
c
c
2r
c
c
3cr
3cr

Example: Cut Plan beats the Connect Plan



Formal Problem Definition

- n: number of server nodes
- X_i: load time series of node N_i
- ρ_{ij}: correlation coefficient of X_i and X_j, $1 \le i, j \le n$
- Find a plan that maps operators to nodes with the following properties:
  - $EX_1 \approx EX_2 \approx \ldots \approx EX_n$
  - $\frac{1}{n}\sum_{i=1}^{n} \operatorname{var} X_i$ is minimized, or
  - $\sum_{1 \le i < j \le n} \rho_{ij}$ is maximized.

Dynamic Load Distribution Algorithms

- Periodically repeat:
  1. Collect load statistics from all nodes.
  2. Order nodes by their average load.
  3. Pair the i^{th} node with the $(n-i+1)^{th}$ node.
  4. If there exists a pair (A, B) such that |A.load – B.load| ≥ threshold, then move operators between them to balance their average load and to minimize their average load variance.
- Two load movement algorithms for pairs in Step 4:
  - One-way
  - Two-way

One-way Algorithm

- Given a pair (A, B) that must move load, the node with the higher load (say A) offloads half of its excess load to the other node (B).

- Operators of A are ordered based on a score, and the operator with the largest score is moved to B until balance is achieved.

- Score of an operator O is computed as follows:

  correlation_coefficient(O, other operators at A)

  – correlation_coefficient(O, other operators at B)

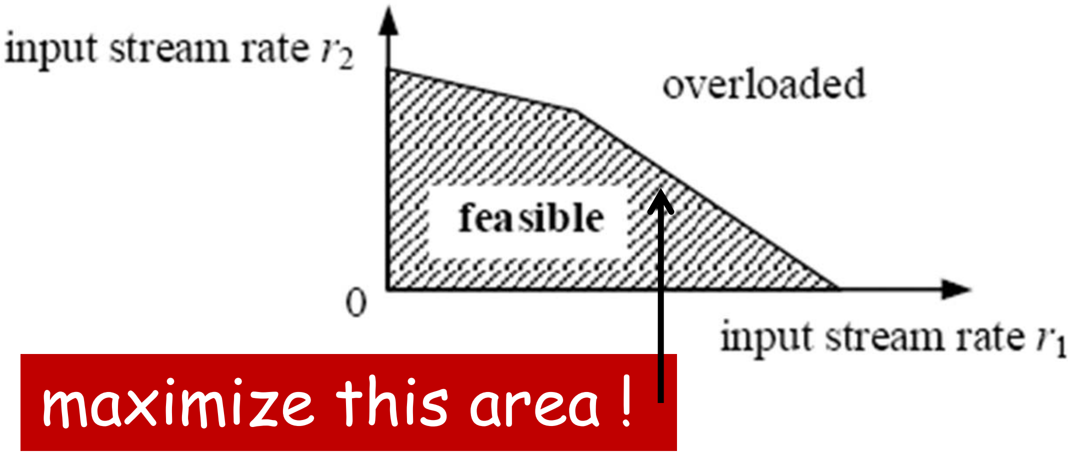
Two-way Algorithm

- All operators in a given pair can be moved in both ways.
- Assume both nodes are initially empty.
- Score all the operators.
- Select the largest score operator and place it at the less loaded node.
- Continue until all operators are placed.

- Two-way algorithm could results in a better placement.
- But, load migration cost would be higher.

Load-resilient Techniques

- Goal: to tolerate as many load conditions as possible without the need for operator migration.
- Resilient Operator Distribution (ROD)
  - ROD does not become overloaded easily in the face of fluctuating input rates.
  - Key idea:



Comparison of Approaches

Correlation-based

- Dynamic
- Medium-to-long term load variations
- Periodic operator movement

Load-resilient

- Static
- Short-term load fluctuations
- No operator movement

Distributed Stream Processing
Major Problem Areas

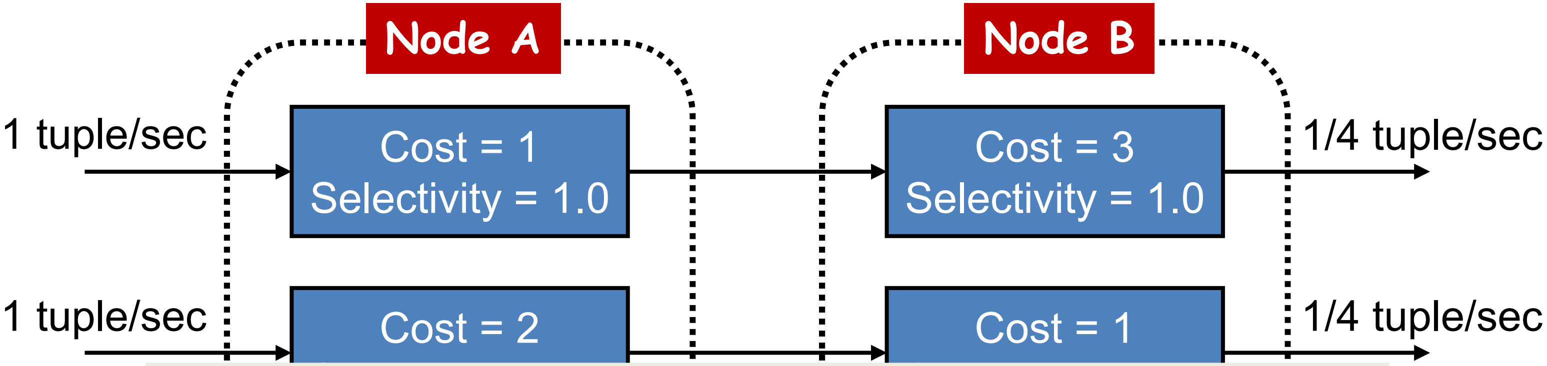
- Load distribution and balancing
  - Dynamic / Correlation-based techniques
  - Static / Load-resilient techniques
  - (Network-aware techniques)
- Distributed load shedding
- High availability and Fault tolerance
  - Handling node failures
  - Handling link failures (esp. network partitions)

Distributed Load Shedding

- Problem: One or more servers can be overloaded.
- Goal: Remove excess load from all of them with minimal quality loss at query end-points.
- There is a load dependency among the servers.
- To keep quality under control, servers must coordinate in their load shedding decisions.

Distributed Load Shedding

Load Dependency



Server nodes must coordinate to achieve high-quality results!

Plan	Rates at A	A.load	A.throughput	B.load	B.throughput
0	1, 1	3	1/3, 1/3	4/3	1/4, 1/4
1	1, 0	1	1, 0	3	1/3, 0
2	0, 1/2	1	0, 1/2	1/2	0, 1/2
3	1/5, 2/5	1	1/5, 2/5	1	1/5, 2/5

optimal for A → Plan 1

feasible for both → Plans 2 and 3

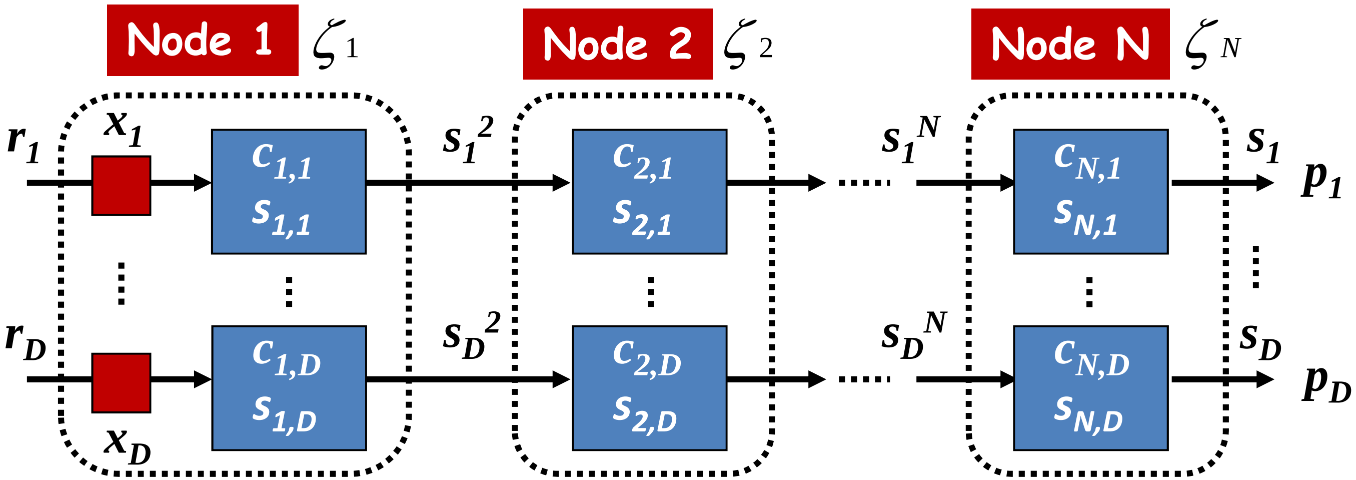
optimal for both → Plan 3

A.load: ≤ 1

B.load: ≤ 1

B.throughput: maximize !

Distributed Load Shedding
as a Linear Optimization Problem



Find x_j such that for all nodes $0 < i \le N$:

$$\sum_{j=1}^{D} r_j \times x_j \times s_j^i \times c_{i,j} \le \zeta_i$$

$$0 \le x_j \le 1$$

$$\sum_{j=1}^{D} r_j \times x_j \times s_j \times p_j \text{ is maximized.}$$

Distributed Stream Processing
Major Problem Areas

- Load distribution and balancing
  - Dynamic / Correlation-based techniques
  - Static / Load-resilient techniques
  - (Network-aware techniques)
- Distributed load shedding
- High availability and Fault tolerance
  - Handling node failures
  - Handling link failures (esp. network partitions)

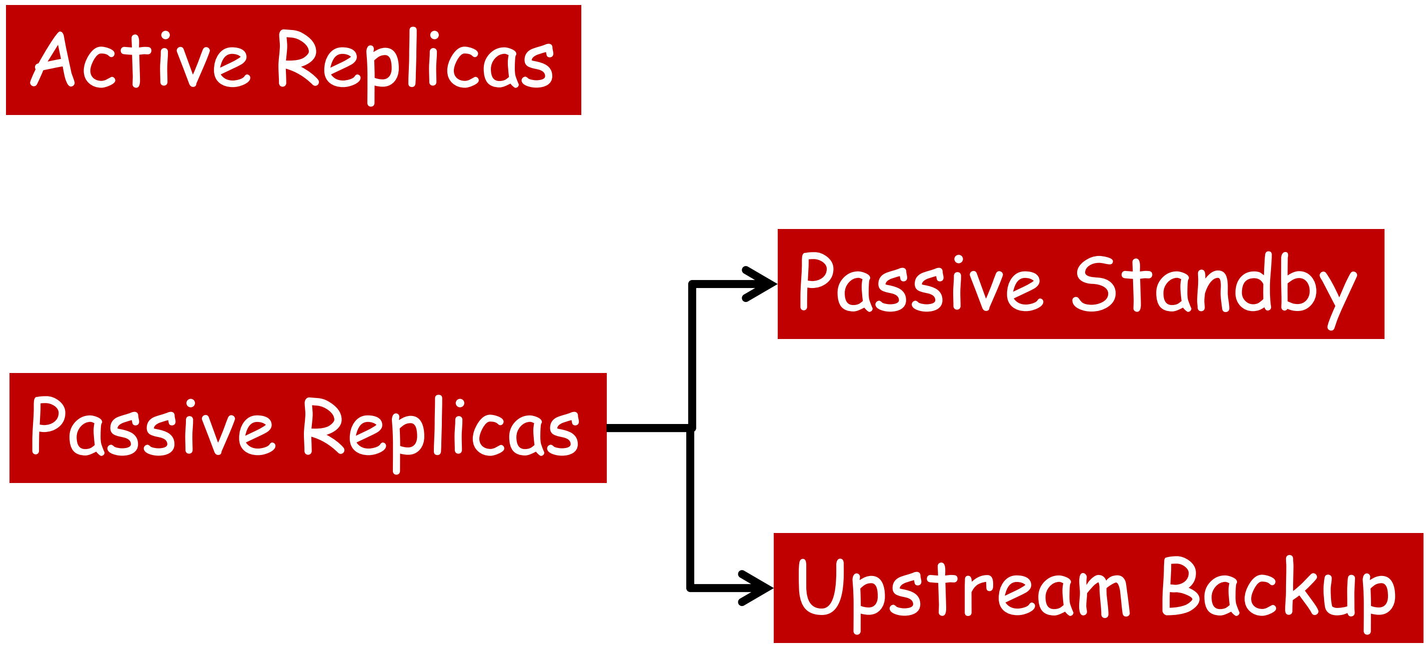
High Availability and Fault Tolerance
Overview

- Problem: node failures and network link failures
  - ➢ Query execution stalls
  - ➢ Queries produce incorrect results
- Requirements:
  - – Consistency -> Avoid lost, duplicate, or out of order data
  - – Performance -> Avoid overhead during normal processing
    + overhead during failure recovery
- Major tasks:
  - – Failure preparation -> Replication of volatile processing state
  - – Failure detection -> Timeouts
  - – Failure recovery -> Replica coordination upon failure

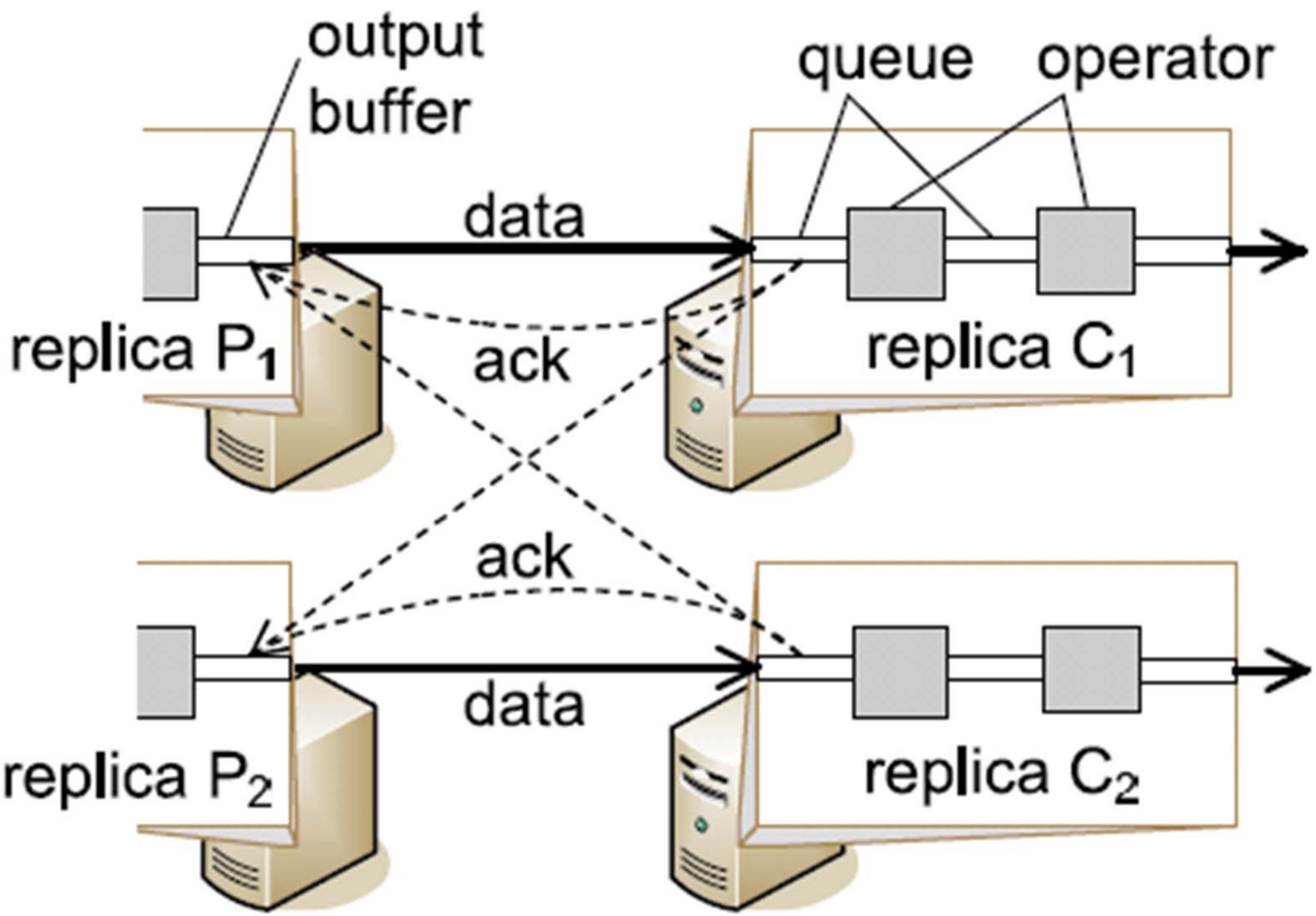
High Availability and Fault Tolerance
General Approach

- Adapt traditional approaches to stream processing
- Two general approaches:
  - State-machine approach
    - Replicate the processing on multiple nodes
    - Send all the nodes the same input in the same order
    - Advantage: Fast fail-over
    - Disadvantage: High resource requirements
  - Rollback recovery approach
    - Periodically check-point processing state to other nodes
    - Log input between check-points
    - Advantage: Low run-time overhead
    - Disadvantage: High recovery time
- Different trade-offs can be made among:
  - Availability, Run-time overhead, and Consistency

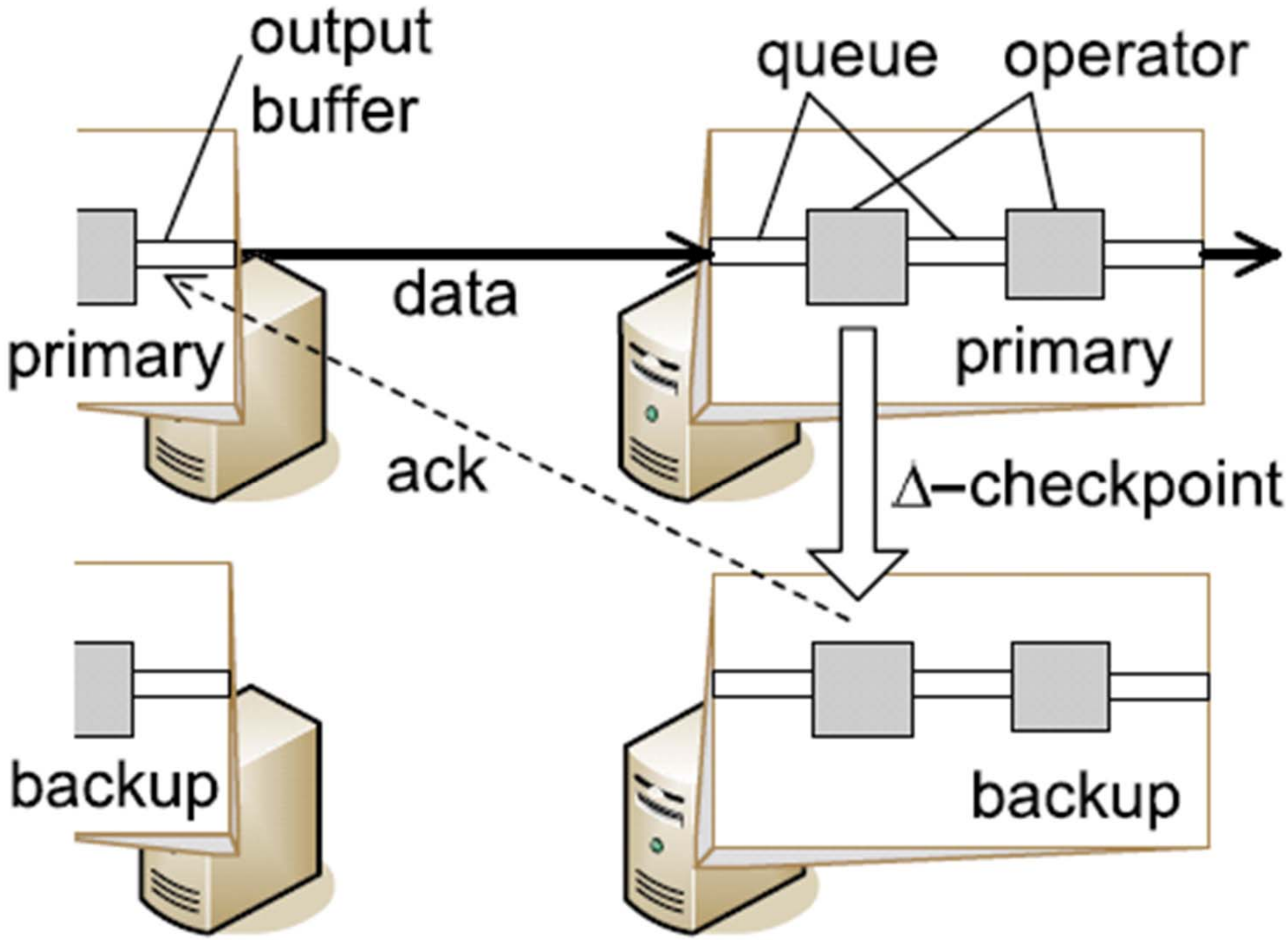
Handling Node Failures



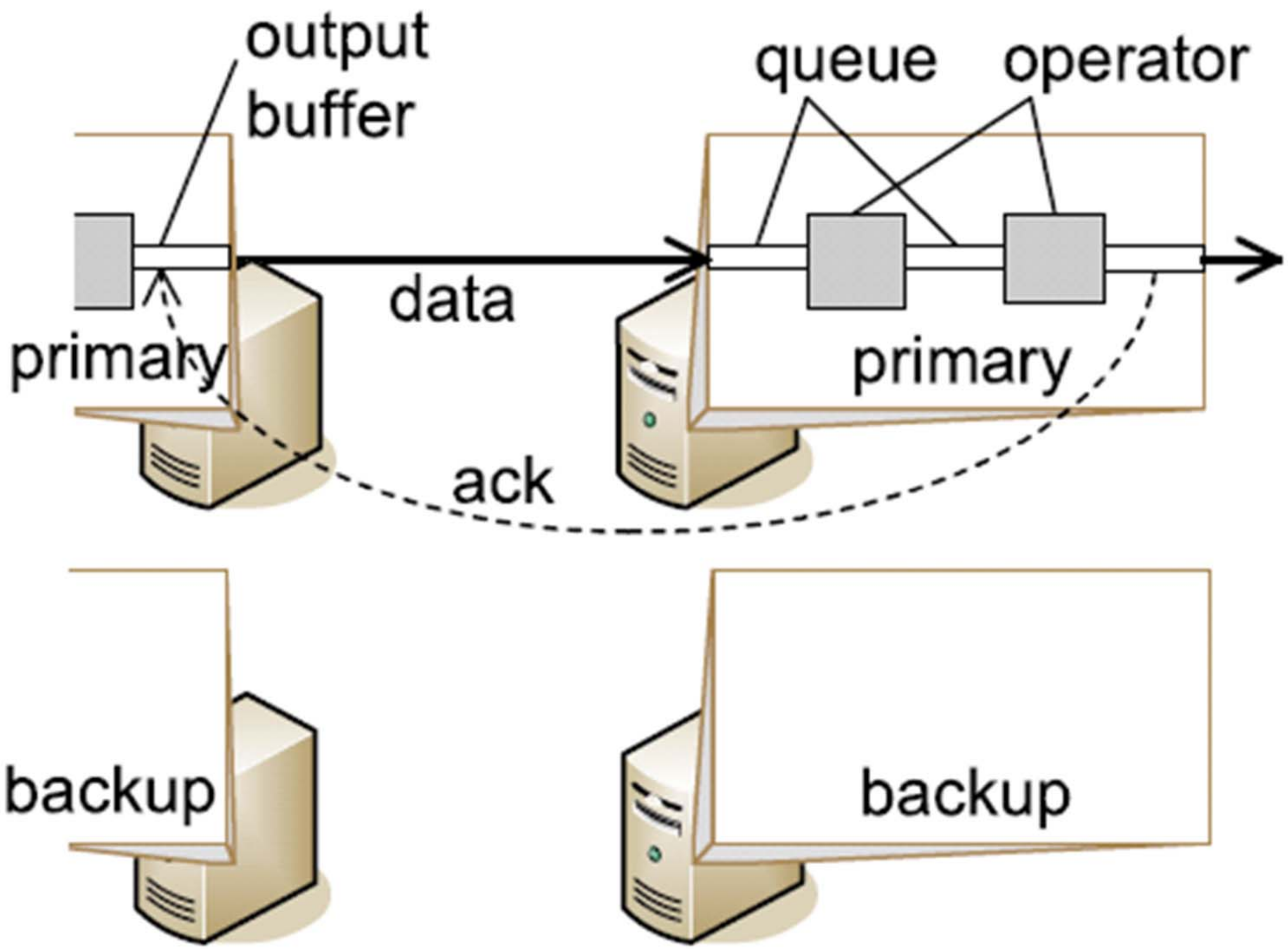
Active Replicas



Passive Standby



Upstream Backup

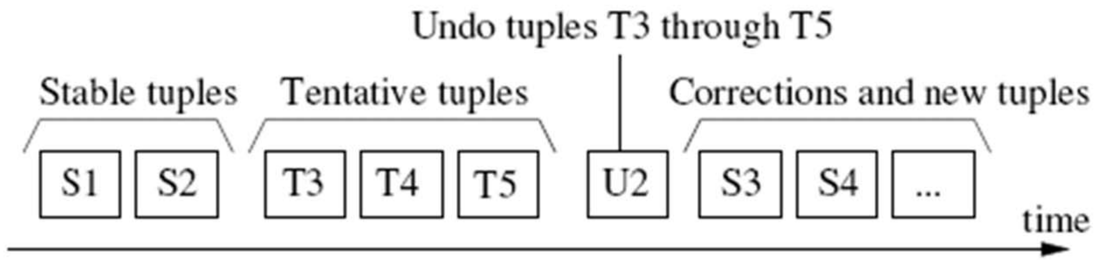


Run-time Overhead vs. Recovery Time Trade-off

- Active Replicas:
  - High run-time overhead
  - Fast fail-over (i.e., low recovery time)
- Passive Standby:
  - Check-point interval can be flexibly adjusted
- Upstream Backup:
  - Low run-time overhead
  - Recovery time is proportional to the size of the upstream buffers

Handling Network Partitions

- "Network Partitions" occur when data sources, processing nodes, and clients are split into disconnected partitions due to network failures.
- Two general options:
  - Suspend processing to avoid inconsistency.
  - Continue processing to avoid unavailability.
- Delay-Process-Correct (DPC) Protocol
  - Adjust the trade-off btw consistency and availability using maximum tolerable latency threshold and tentative tuples.



Other Advanced HA Techniques

- Cooperative and Self-configuring HA [Borealis]
  - Each server node is backed up by multiple servers in a cooperative fashion, which can take over processing in parallel.
  - Backup assignment dynamically changes to balance HA load.
  - Wide-area extensions
- Integrating Fault Tolerance with Load Balancing [Flux]
  - Fine-granularity dataflow partitions
  - Rebalance load after failure recovery