Contributions to Stream Processing

An Analysis of Existing Contributions

January 10, 2019

Nathan Woods

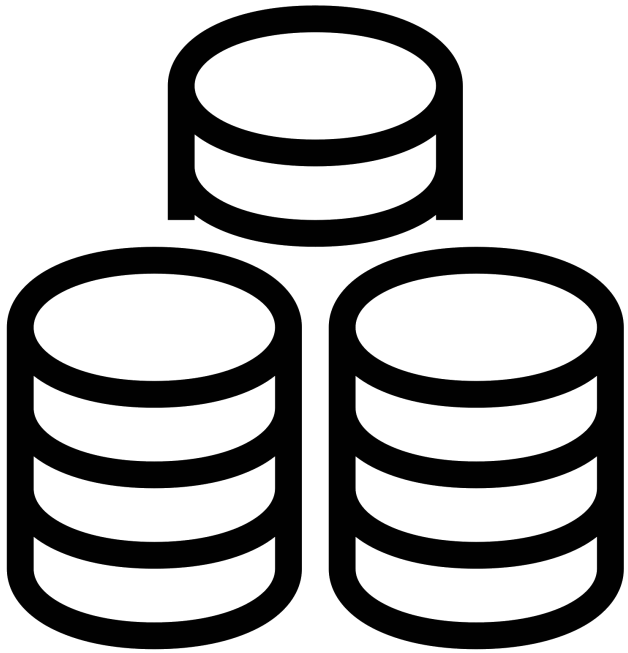
Gianforte School of Computing
Montana State University

MONTANA STATE UNIVERSITY

Introduction

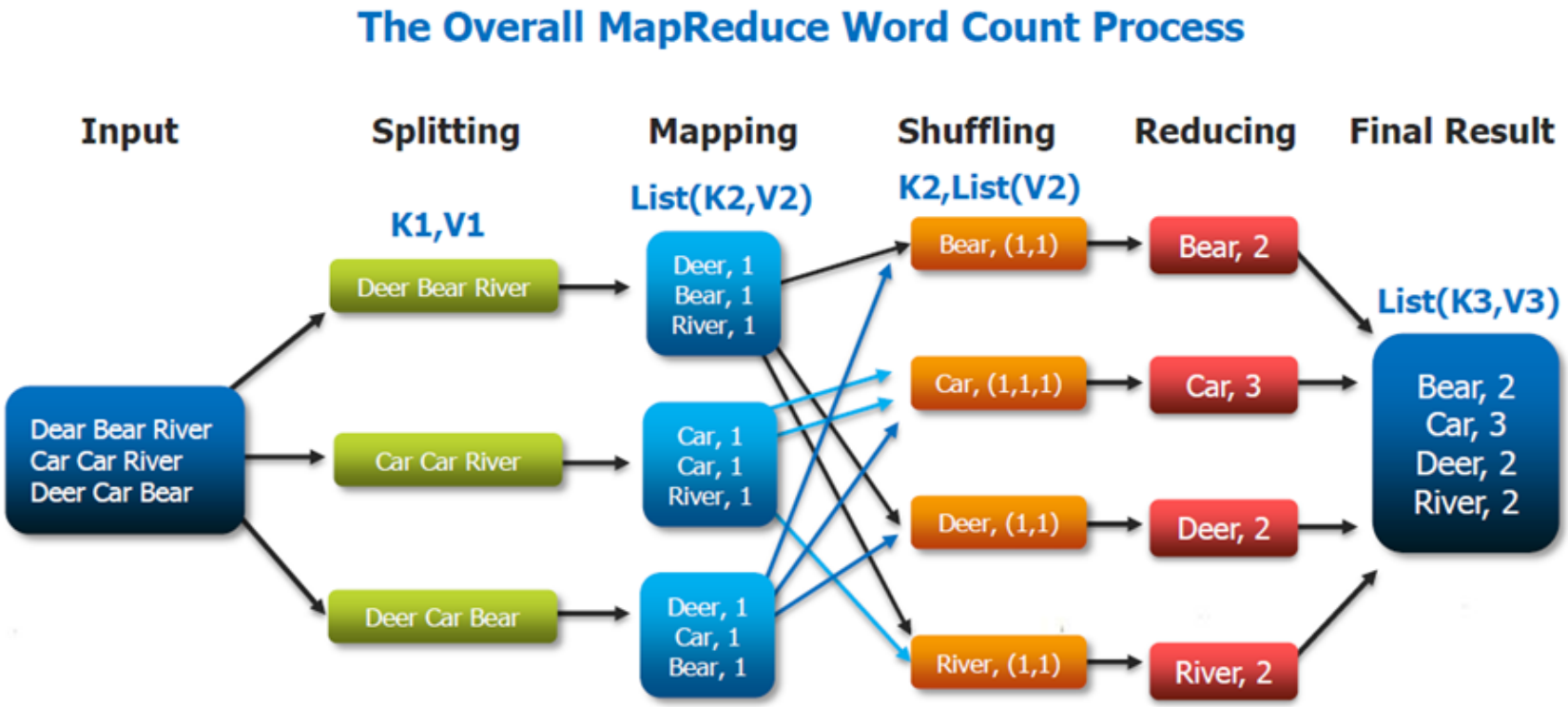
- 2,744,774 emails
- 8,262 tweets
- 70,634 searches
- 64,201 GB traffic

http://www.internetlivestats.com/



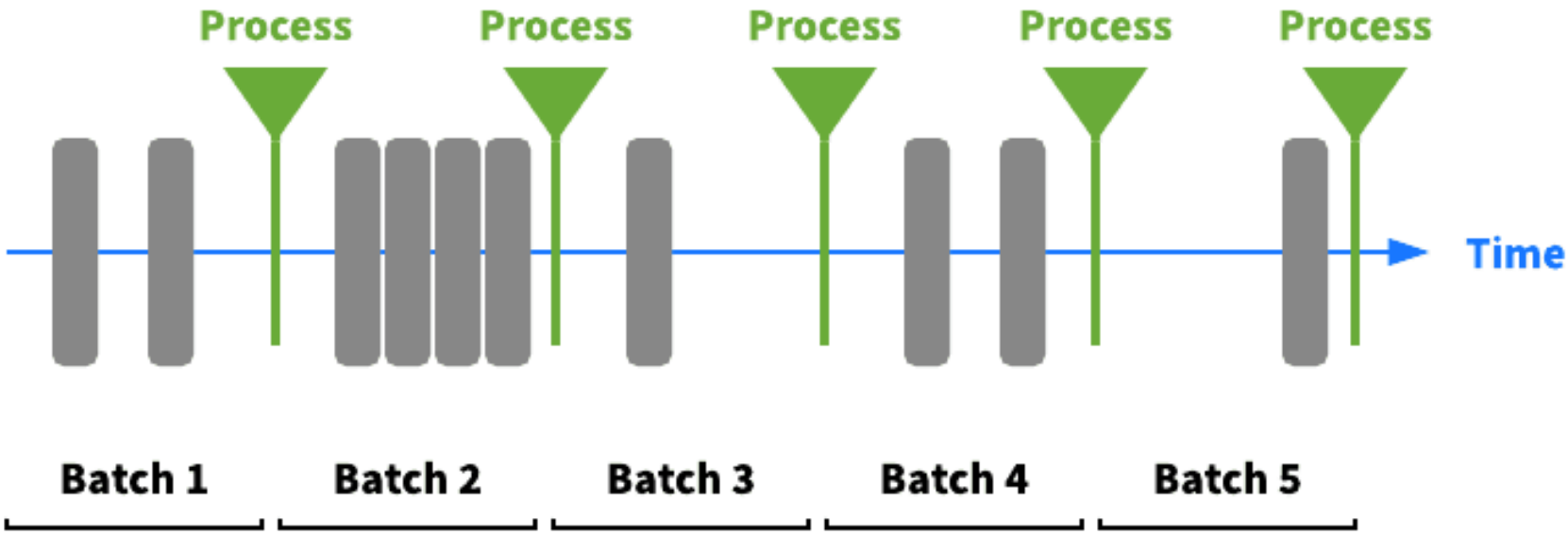
https://icons8.com/icon/65568/big-data

MapReduce



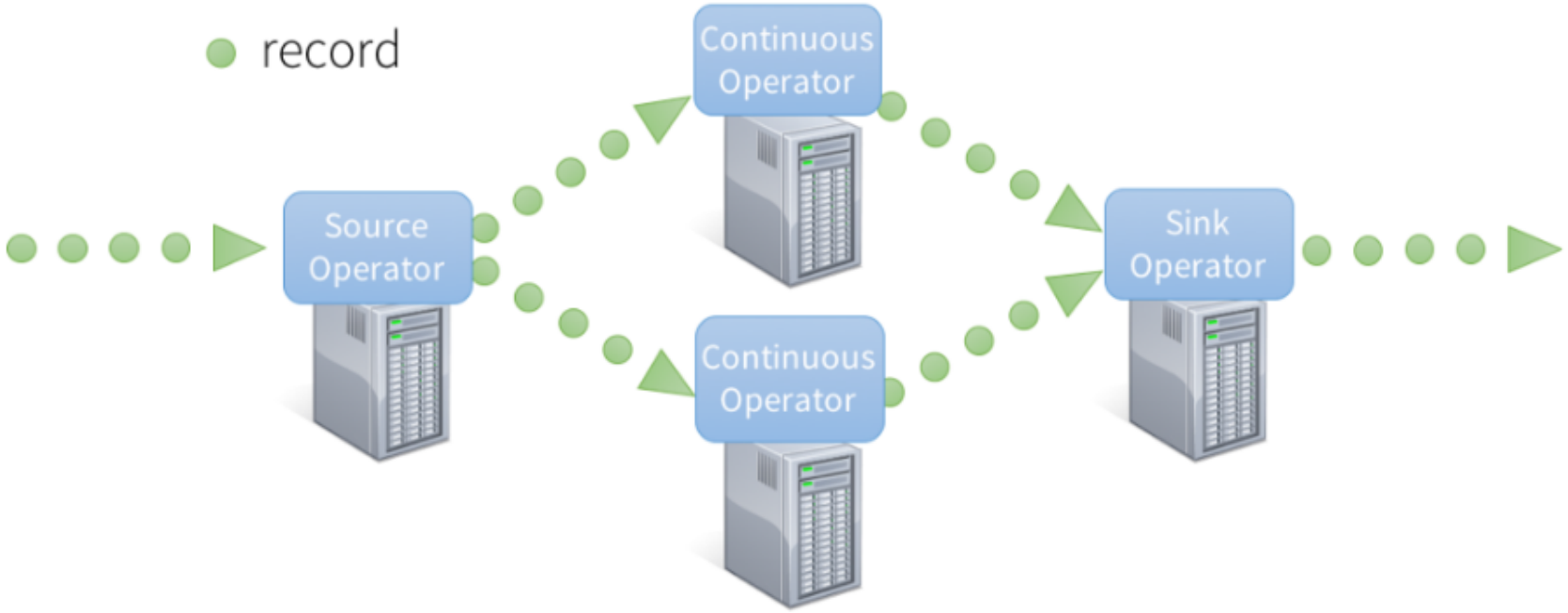
https://i.stack.imgur.com/199Q1.png

Micro-Batching



https://streaml.io/media/img/batch-processing.png

Continuous Operators



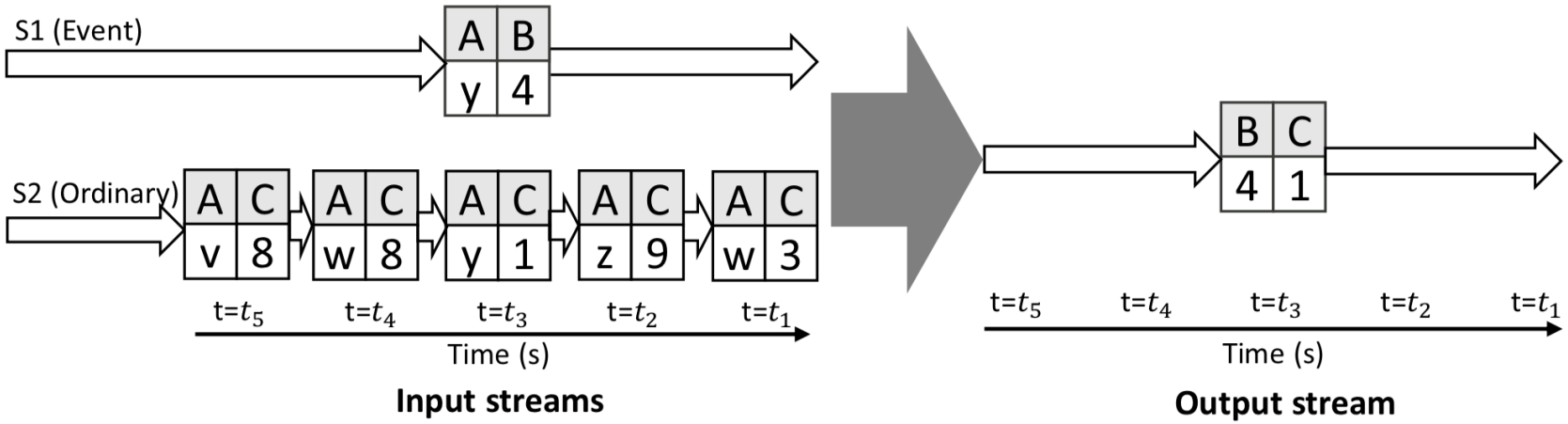
https://pangbw.files.wordpress.com/2017/04/15.png

Outline

- Introduction
- Contributions
  - Smart Windows
  - Drizzle
  - Spade
  - Real-Time Analytics
  - Consistency
- Synthesis

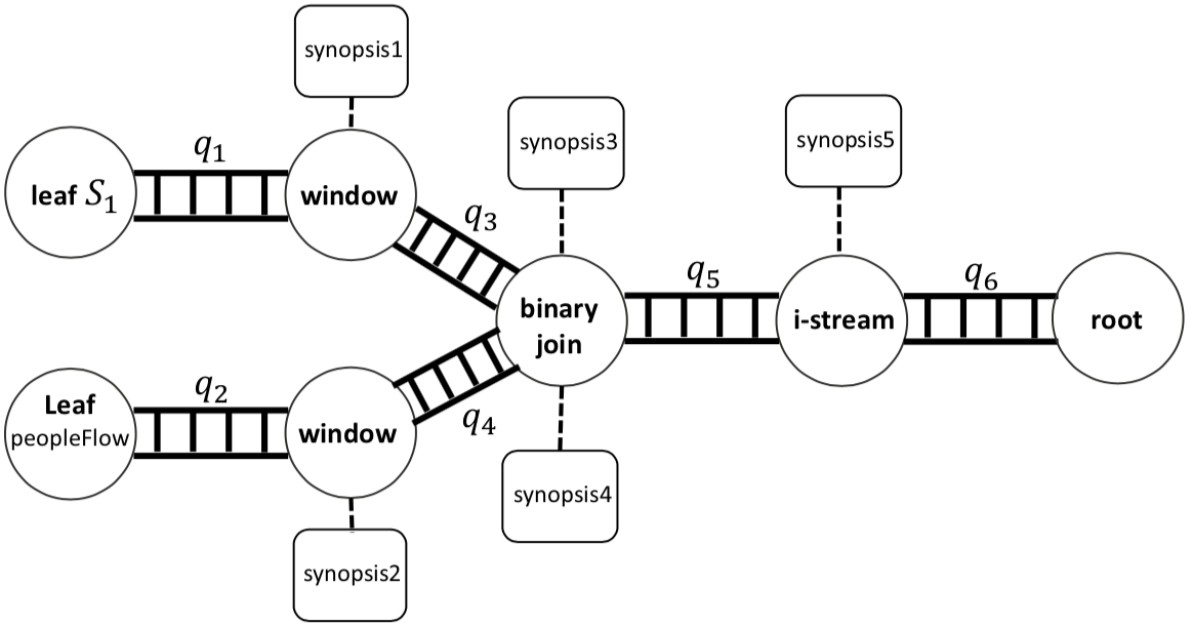
Smart Windows

S.A.Shaikh, Y. Watanabe, Y. Wang, and H. Kitagawa
Smart Query Execution for Event-driven Stream Processing
Multimedia Big Data (BigMM), 2016 IEEE Second International Conference on. IEEE, 2016, pp. 97-104



Event-driven Query / Event Stream / Active

Event Windows



- **Tuple-based Window:** Given integer n, return the n most recent tuples from stream S.
- **Time-based Window:** Given τ, at any time t return the tuples with timestamps between $t - \tau$ and t from stream S.
- **Incremental Computation:** Annotate events with "+" or "-" to signal events being added or removed from the window.

Smart Windows

Algorithm 1 Smart Window (W_o): When new tuple arrives

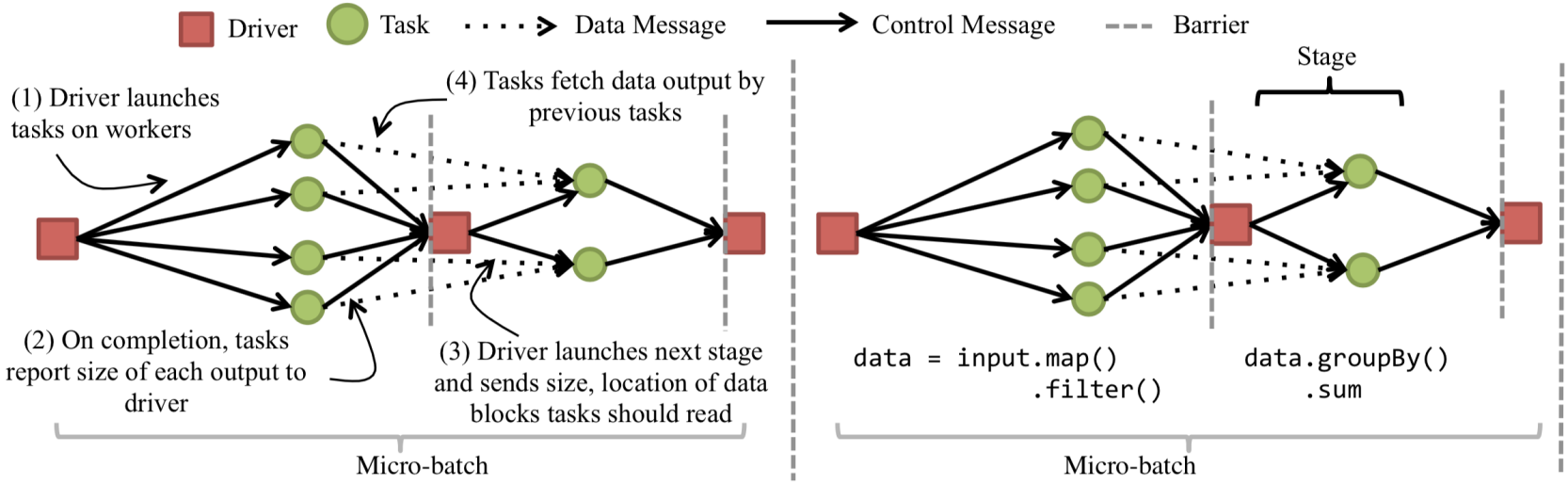
1: **for each** arrival of ordinary stream tuple $e \in Q$ at timestamp t **do**
2: **if** isActive(Q) **then**
3: Insert e in the output part and send $\langle e, t, + \rangle$ downstream
4: **else**
5: Buffer e in the suspended part
6: **end if**
7: **if** # of elements $\in W_o >$ size of W_o **then**
8: Find e'; $\{e'$: oldest element in $W_o\}$
9: **if** $e' \in$ suspended part **then**
10: delete e'
11: **else**
12: delete e' and send $\langle e', t, - \rangle$ downstream
13: **end if**
14: **end if**
15: **end for**

Smart Windows Summary

- Smart Windows reduce Processing Load
- Small scale testing
  - single system
  - 1 minute tests
- Detect conditional queries
- Expand concept to include conditional joins

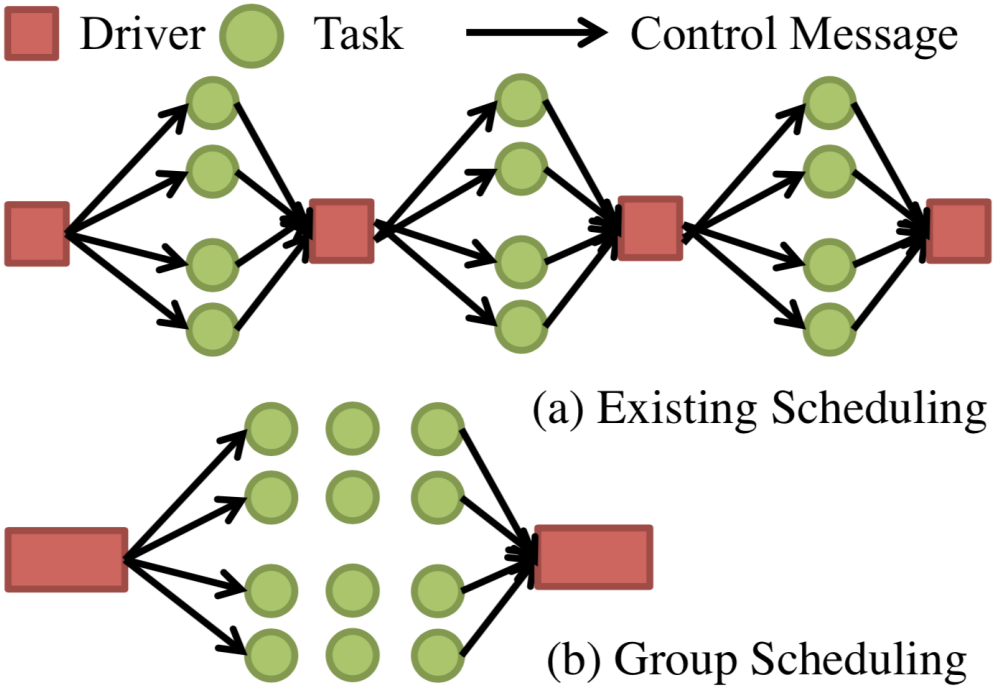
Drizzle

S.Venkataraman, A.Panda, K.Ousterhout, M.Armbrust, A.Ghodsi, M.R.Franklin, B.Recht, and I.Stoica
Drizzle: Fast and Adaptable Stream Processing at Scale
Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles. ACM, 2017, pp. 374-389



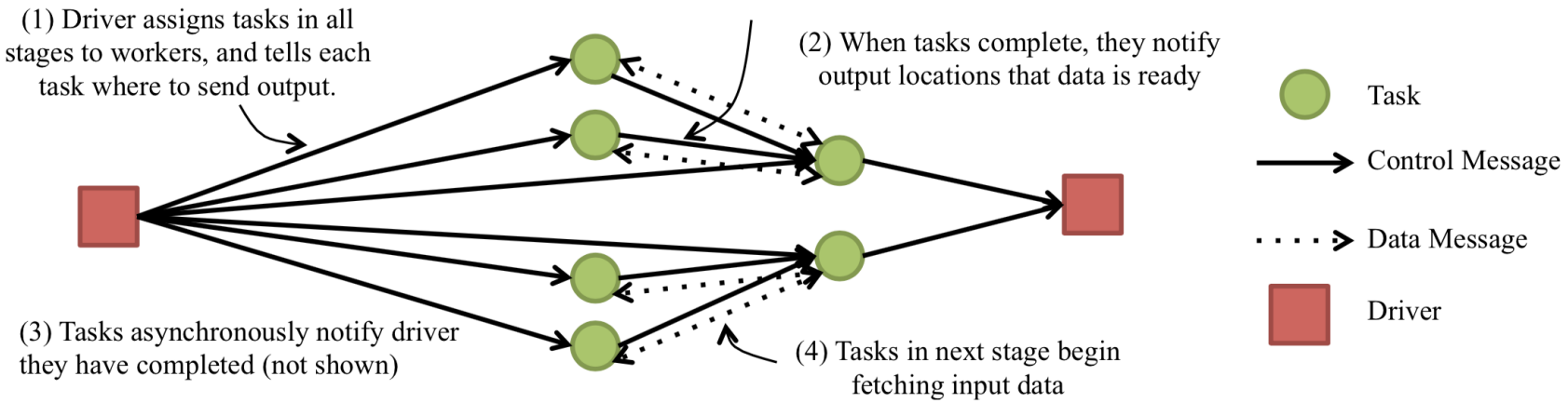
- **Failures:** Communication, Process, Hardware
- **Recovery:** Micro-Batch vs Continuous Operators

Group Scheduling



Same scheduling instructions across multiple micro-batches.

Pre-Scheduling Shuffles



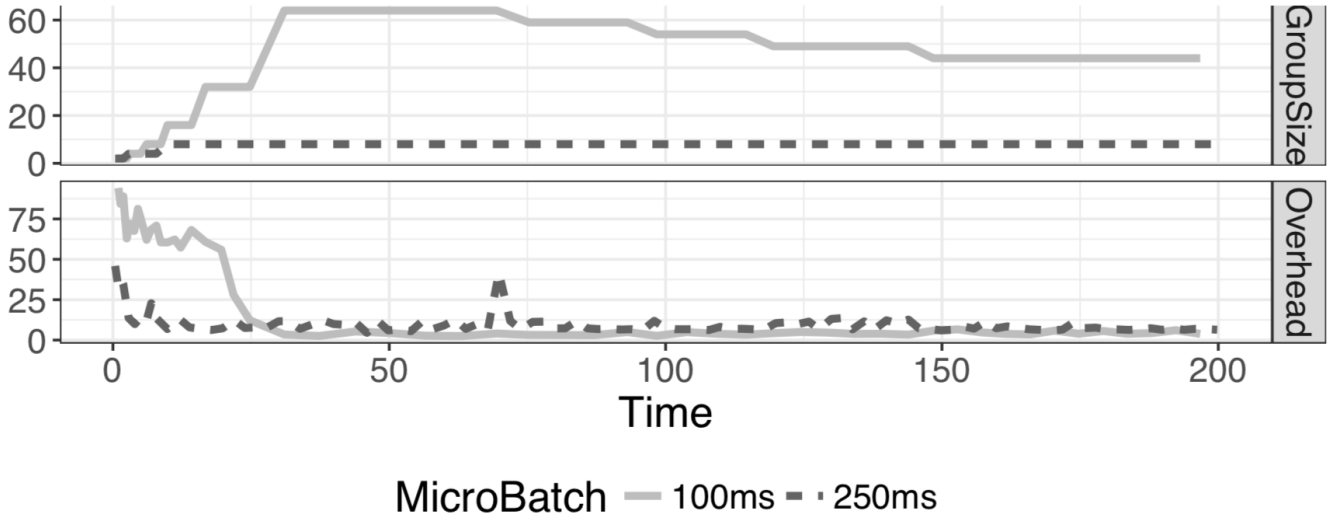
Scheduling dormant downstream tasks first.
Upstream can send data directly to consumer.

Adaptability

Fault Tolerance: Worker Heartbeat, Parallel Retries, Checkpoints
Elasticity: Adjust plan at boundaries based on available resources.

Tune group size based on TCP congestion control

$$gs(t+1) = \begin{cases} gs(t) \times a & \text{if scheduling overhead} > \text{upper bound} \\ gs(t) - b & \text{if scheduling overhead} < \text{lower bound} \end{cases}$$



Data-Plane Changes

Within Batch

Vectorized CPU Operations

Minimize trafic with partial merges

Across Batches

Metrics to measure query performance

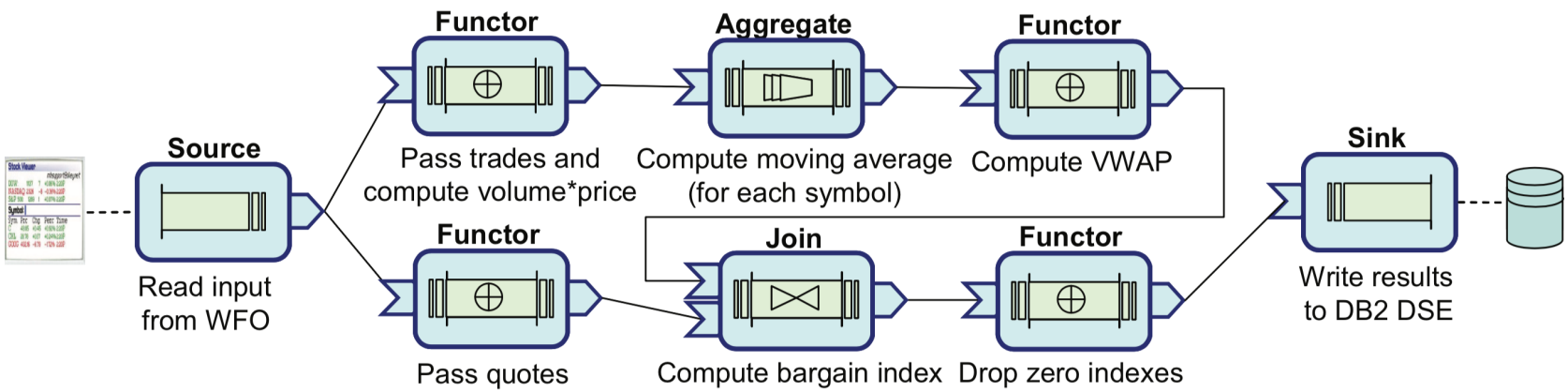
Reuse results across different queries

Drizzle Summary

- Low Latency Scheduling
- Data-Level Optimizations
- Good testing
  - 128 `r3.xlarge`
  - Included JVM warmup
  - Single node failures
- Multiple node failures?

Spade

B.Gedik, H.Andrade, K.-L. Wu, P.S.Yu, and M.Doo
Spade: The System S Declarative Stream Processing Engine
Proceedings of the 2008 ACM SIGMOD international conference on Management of data. ACM, 2008, pp. 1123-1134



Stream Processing Ecosystem

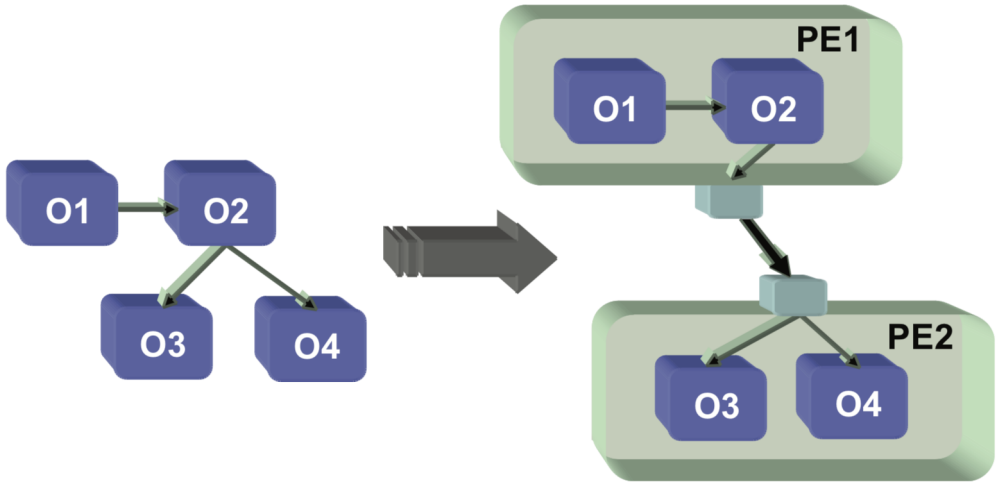
System S

- Large-scale distributed data stream processing middleware
- Distributes jobs across a cluster (DAGs and PEs)
- Handles Reliablity, Scheduling and Placement Optimization, Distributed Job Management, Storage Services, Security, etc

Spade

- Declarative Programming Language
- Fundamental Unit is a Stream
- Deploys Programs to System S
- Includes Compilers, Optimizers and Generators (UDOPS)
- Differences: Pre-grouping, Vectorized Operations, Edge Adapters, Windowing Schemes

Operator Fusion



Combine Input Queues for nodes in PEs

Spade Summary

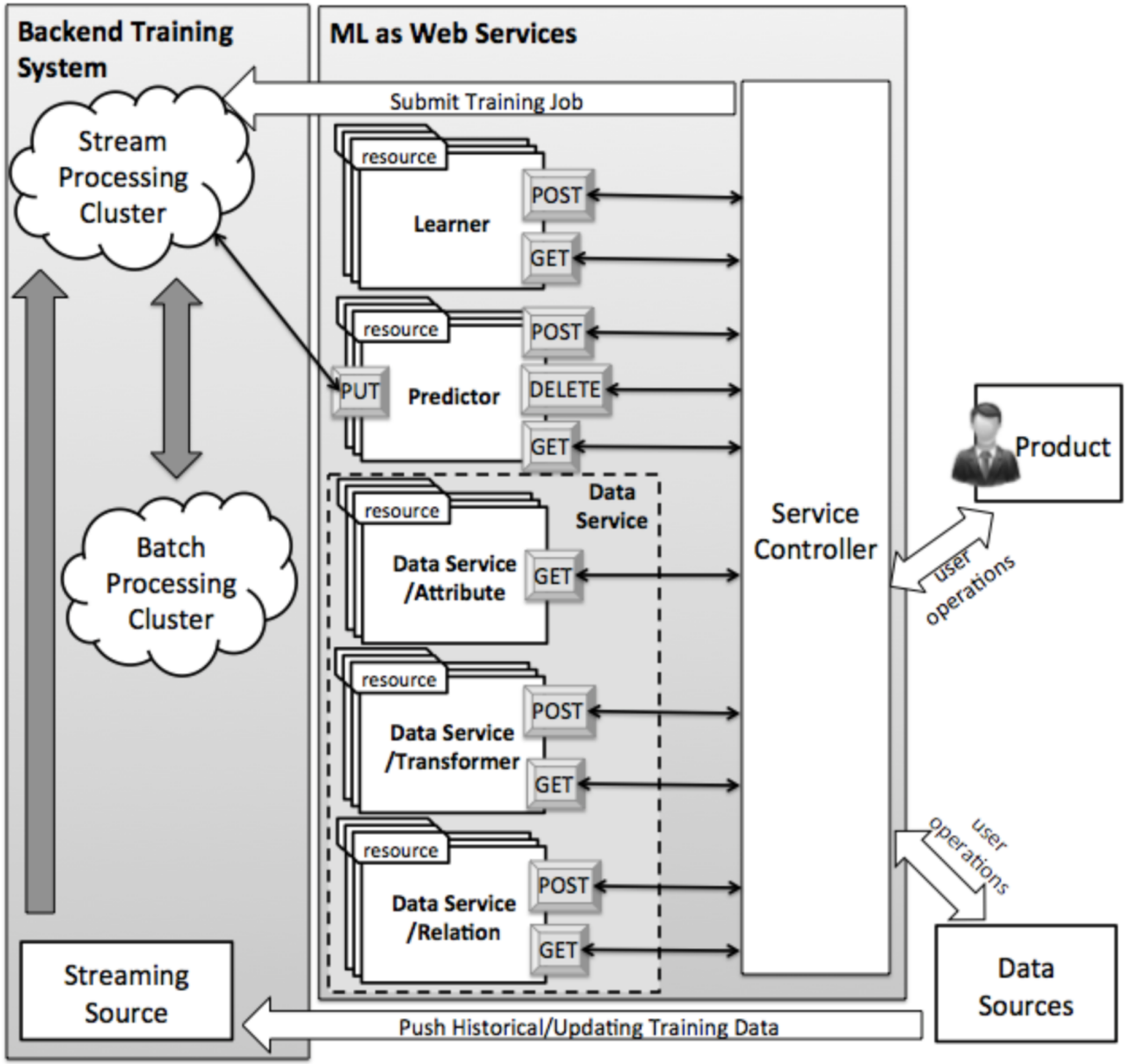
- Stream Processing Ecosystem (2 years before Spark)
- Small test set of 20GB
  - 20GB of data
  - 16 node cluster
  - 1.6 Million tuples/second
  - 3.5 minutes
- System could report under-performing queries for review

Real-Time Analytics

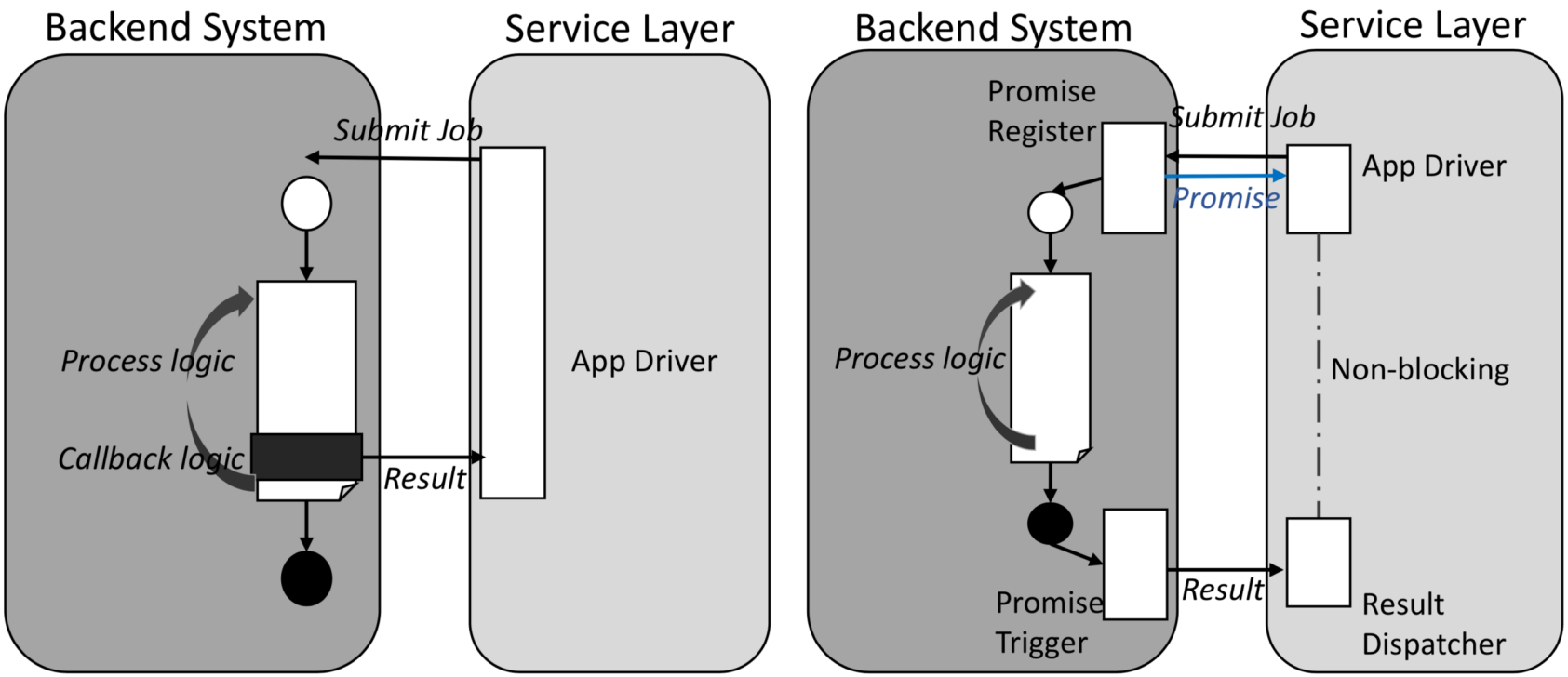
D.Xu, D.Wu, X.Xu, L.Zhu, and L.Bass
Making Real Time Data Analytics Available as a Service
Quality of Software Architectures (QoSA), 2015 11th International ACM SIGSOFT Conference on. IEEE, 2015, pp. 73-82

- Scaling data processing to meet demand
- ML models to update with live data
- Combining big data processing with ML training

Components


Backend Training System
ML as Web Services
Submit Training Job
Stream Processing Cluster
Batch Processing Cluster
resource
Learner
POST
GET
resource
PUT
Predictor
POST
DELETE
GET
Data Service
resource
Data Service /Attribute
GET
resource
Data Service /Transformer
POST
GET
resource
Data Service /Relation
POST
GET
Service Controller
Product
user operations
user operations
Data Sources
Streaming Source
Push Historical/Updating Training Data

Processing Integration

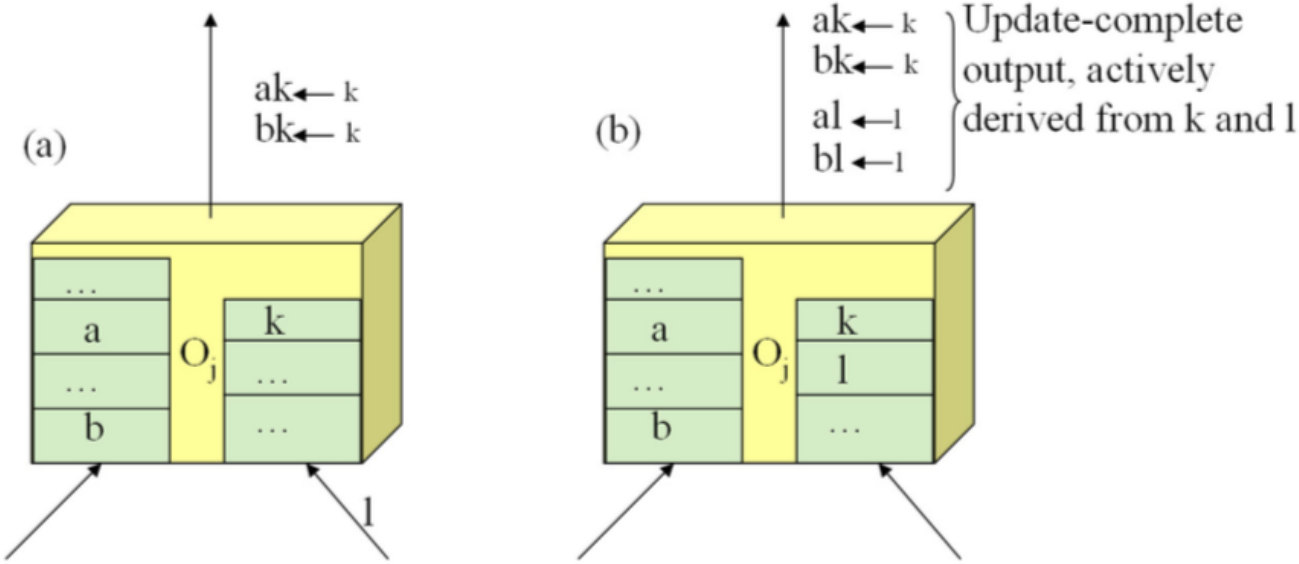


Real-Time Analytics Summary

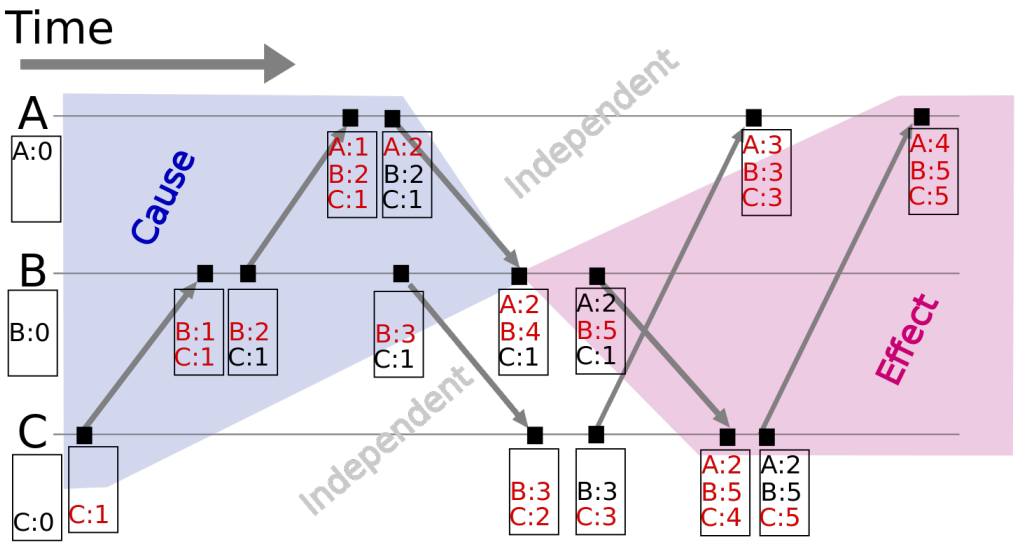
- Wrapped ML behind a REST API
- Adapted Stream Processing Systems to Support ML + API
- Small testing bit-rates
  - 9 `t2.medium`
  - low bit-rate testing (20MB/s)
- Initialize state with parallel processing
  Update models with incremental changes

Consistency

G.A.Mihaila, I.Stanoi, and C.A.Lang
Anomaly-Free Incremental Output in Stream Processing
Proceedings of the 17th ACM Conference on Information and knowledge management. ACM, 2008, pp. 359-368



Heeding

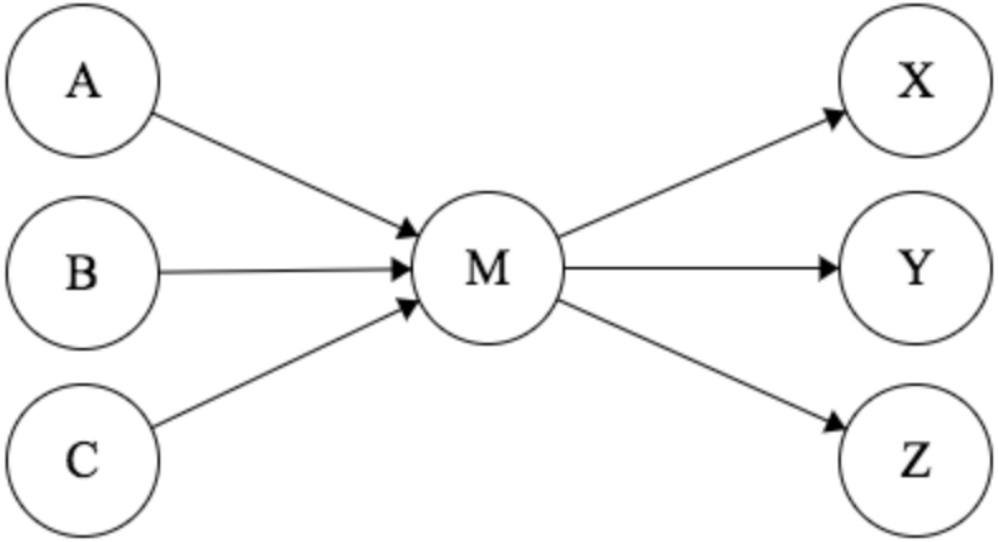


`https://en.wikipedia.org/wiki/Vector_clock`

Also labels active contributor to an event

Wait-and-See, Passive Consistency

Periodic Draining



http://madebyevan.com/fsm/

Synchronization Tokens injected every δT

Active Consistency

Consistency Summary

- Ensure Consistent Results in Stream Processing Engines
- Rather simple testing scenarios
  - 3 joins
  - $[1, 5000]$ integers
  - single machine processing
- Integrate consistency with recovery

Open Research Questions

- Large-Scale Testing
- Integrating Features
- Failure Recovery
- Multiple Cluster

Thank You

Questions?

Slides available at `https://bign8.info/msu/qual.pdf`