CS856 Fall 2005 Presentation

Fault-Tolerance in the Borealis Distributed Stream Processing System

Weihan Wang

November 23, 2005

About this paper

- Magdalena Balazinska, MIT
- Hari Balakrishnan, MIT
- Samuel Madden, MIT
- Michael Stonebraker, MIT

- In *Proc. ACM SIGMOD Int. Conf. on Management* of Data, 2005.

2

Agenda

- Background
- System overview
- Upstream failure
- Stabilization
- Evaluation

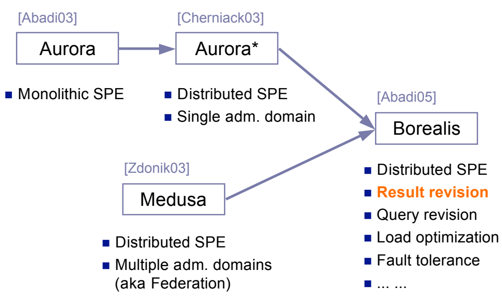
3

Agenda

- **Background**
- System overview
- Upstream failure
- Stabilization
- Evaluation

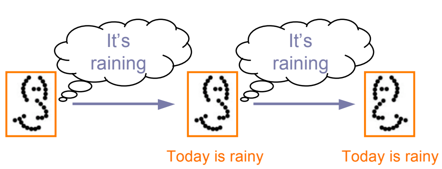
4

A piece of history



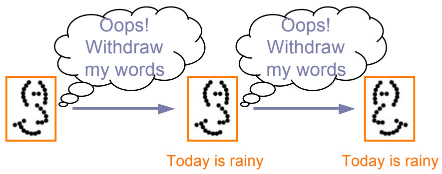
5

Result revision in Borealis



6

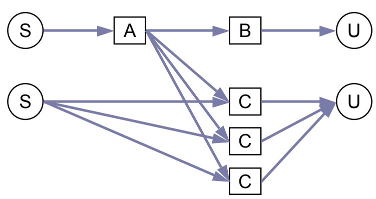
Result revision in Borealis



- Error fixing from data source
- Load shedding
- Time-travel into the past or future
- Fault tolerance

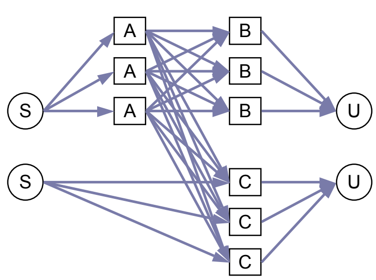
7

Common solution for replication-based FT



8

Common solution for replication-based FT



9

Comparison with [Hwang05]

- They don't distinguish between HA & FT.
- They are parallel to each other.
- Compared to [Hwang05]:
  - Approach of this paper is similar to [Hwang05]'s active standby.
  - This paper uses result revision.
  - This paper addresses network failures.
  - This paper avoids inter-replica communications.
  - ……

10

Agenda

- Background
- **System overview**
- Upstream failure
- Stabilization
- Evaluation

11

Design goal

- User's preference:

During failure	*After failure*	*User*
No outputs	Correct outputs	☹
Approximation	-	☺
Approximation	Error correction	☺ ☺

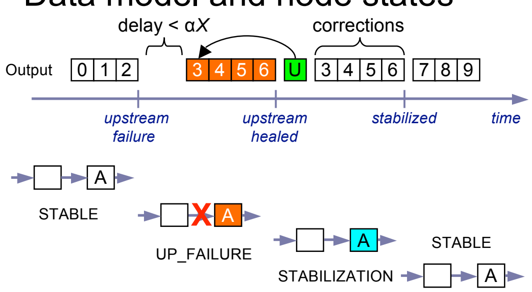
12

Design goal

- Goal: to minimize the number of approximated outputs during failure, subject to a delay constraint, and to revise them after failure.
- For each nodes, the user-defined delay constraint is *X*, and data processing time is (1-α)*X*. So we can hold input tuples up to α*X* sec.

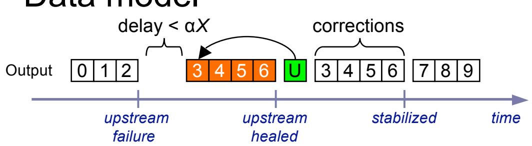
13

Data model and node states



14

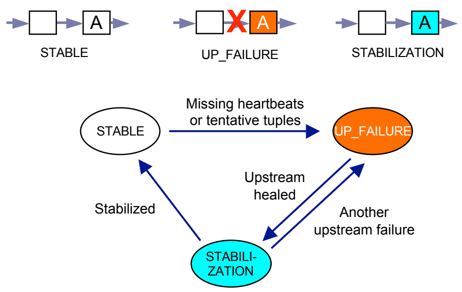
Data model



- Tuple format: (*type, id, time,* a_1, …, a_m)
  - 1 STABLE tuples
  - 1 TENTATIVE tuples
  - U UNDO tuples
  - B BOUNDARY tuples

15

Node states



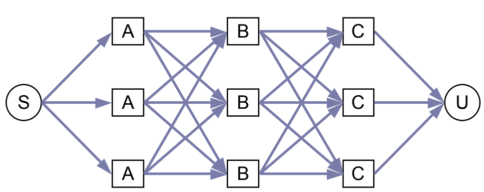
16

Agenda

- Background
- System overview
- **Upstream failure**
- Stabilization
- Evaluation

17

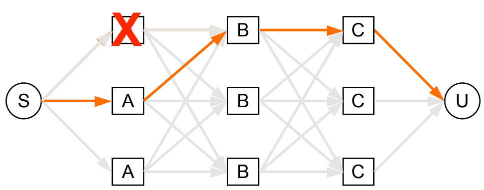
Scenario 1



Every node monitors *all* upstream nodes
and does stream processing simultaneously.

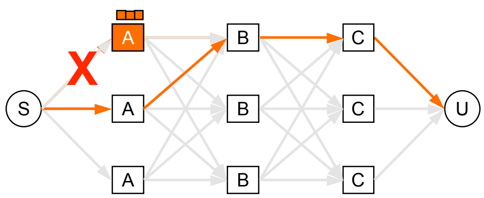
18

Scenario 1



19

Scenario 2



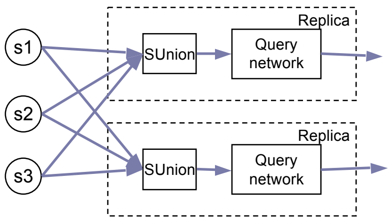
20

Issues of upstream switching

- All replicas must have consistent outputs.
- No inter-replica communication.
- Solution
  - Use *deterministic* operators
  - Use *SUnion* & boundary tuples to sort inputs from multiple streams deterministically.

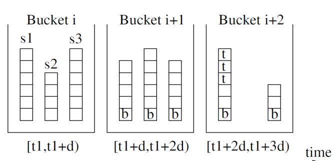
21

SUnion



22

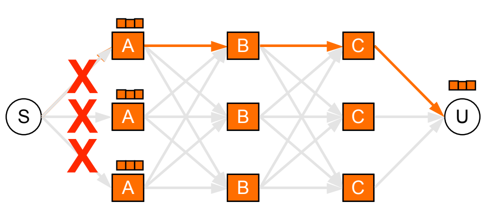
SUnion



- If all boundaries arrive in time, SUnion sorts & forwards the whole bucket as STABLE tuples.
- If boundaries don't arrive in time, or there are TENTATIVE tuples, SUnion stores & forwards the bucket as TENTATIVE tuples

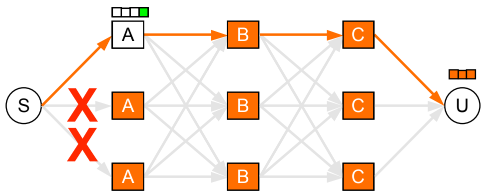
23

Scenario 3



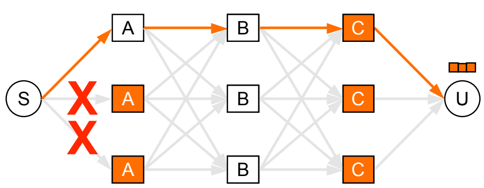
24

Scenario 3



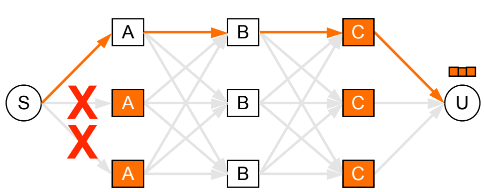
25

Scenario 3



26

Scenario 3



27

Agenda

- Background
- System overview
- Upstream failure
- **Stabilization**
- Evaluation

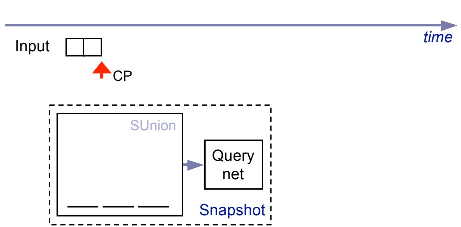
28

Stabilization

- State reconciliation
  - Checkpoint / redo
  - Undo / redo
  - How to satisfy delay constraint if stabilization takes long?
- Output stabilization
- Failed node recovery

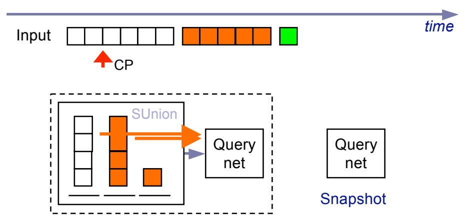
29

State reconciliation: Checkpoint / Redo



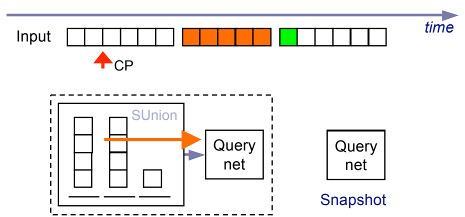
30

State reconciliation: Checkpoint / Redo



31

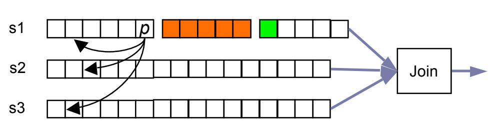
State reconciliation: Checkpoint / Redo



32

State reconciliation: Undo / Redo

- The stream markers of tuple *p* identify the oldest tuples on each input stream that still contribute to the operator's state when the operator processes *p*.



- To undo all tuples after *p*, reset the operator and restart from the markers of *p*.

33

Processing new tuples during reconciliation

- A node suspends its outputs for state reconciliation. But it may take longer than *X*.
- Solution:
  - The node requests another replica to postpone its own reconciliation.
  - The downstream nodes turn to that replica for TENTATIVE outputs.
  - They switch back to the original node when reconciliation done.

34

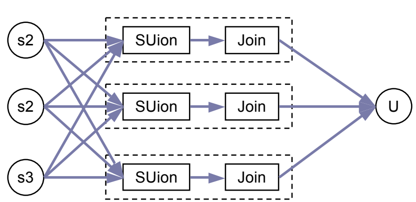
Agenda

- Background
- System overview
- Upstream failure
- Stabilization
- **Evaluation**

35

Evaluation setup

- Single-node evaluation
- Multiple-node evaluation



36

Evaluation results

- The best approach is to process new tuples without delay in both UP_FAILURE and STABILIZATION states.
- Checkpoint/redo is better than undo/redo.
- Memory overhead is proportional to:
  - # of SUion
  - SUion's bucket sizes
  - SUion's input rates

37

Conclusion

- The approach favors availability but guarantees eventual consistency.
- It uses result revision to achieve final consistency.
- It uses SUion to synchronize replicas without inter-replica communication.
- Checkpoint/redo and undo/redo are used for state reconciliation.

38

Discussion

- Long failures may cause output/input buffers overrun.
- No enough explanation on output buffer truncation strategies.
- No enough explanation on relationship between boundary tuples and SUnion bucket size.
- How to recover failed node with divergent operators?
- No evaluations on failed node recovery and replica switching during reconciliation.

39

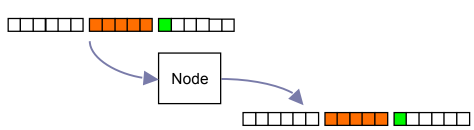
References

- [Abadi03] D. Abadi, D. Carney, U. Cetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, and S. Zdonik. **Aurora: A new model and architecture for data stream management**. *The VLDB Journal*, 12(2):120-139, Aug 2003.
- [Cherniack03] Mitch Cherniack, Hari Balakrishnan, Magdalena Balazinska,Don Carney, Ugur Cetintemel, Ying Xing, and Stan Zdonik. **Scalable Distributed Stream Processing**, CIDR 2003
- [Zdonik03] Stan Zdonik, Michael Stonebraker, Mitch Cherniack, Ugur Cetintemel, Magdalena Balazinska, and Hari Balakrishnan, **The Aurora and Medusa Projects**, IEEE Computer Society. March 2003. p.3-10
- [Abadi05] Daniel J. Abadi, Yanif Ahmad, Magdalena Balazinska, Ugur Centintemel, Mitch Cherniack, Jeong-Hyon Hwang, Wolfgang Lindner, Anurag S. Maskey, Alexander Rasin, Esther Ryvkina, Nesime Tatbul, Ying Xing, and Stan Zdonik, **The Design of the Borealis Stream Processing Engine**, CIDR 2005
- [Hwang05] J-H. Hwang, M. Balazinska, A. Rasin, U. Cetintemel, M. Stonebraker, and S. Zdonik. **High-availability algorithms for distributed stream processing**. In Proc. 21st Int. Conf. on Data Engineering, pages 779-790, 2005.

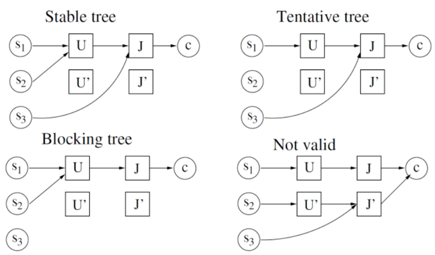
Backup slides

Output stabilization

- Every node shall propagate UNDO tuples during stabilization.
- Checkpoint/redo nodes use *SOutput* operators to help produce UNDO tuples.



Query network trees



43

Implementation

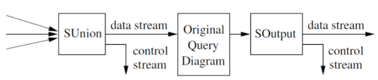
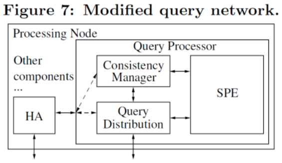


Figure 7: Modified query network.



44

Operator / wrapper interface

- For checkpoint / redo
  - packState()
  - unpackState()
- For undo / redo
  - clear()
  - findOldestTuple(int stream_id)
- For boundary tuple
  - findOldestTimestamp()

45