StreamScope: Continuous Reliable Distributed Processing of Big Data Streams

Wei Lin*, Haochuan Fan*, Zhengping Qian (ZP)*, Junwei Xu, Sen Yang,
Jingren Zhou*, Lidong Zhou

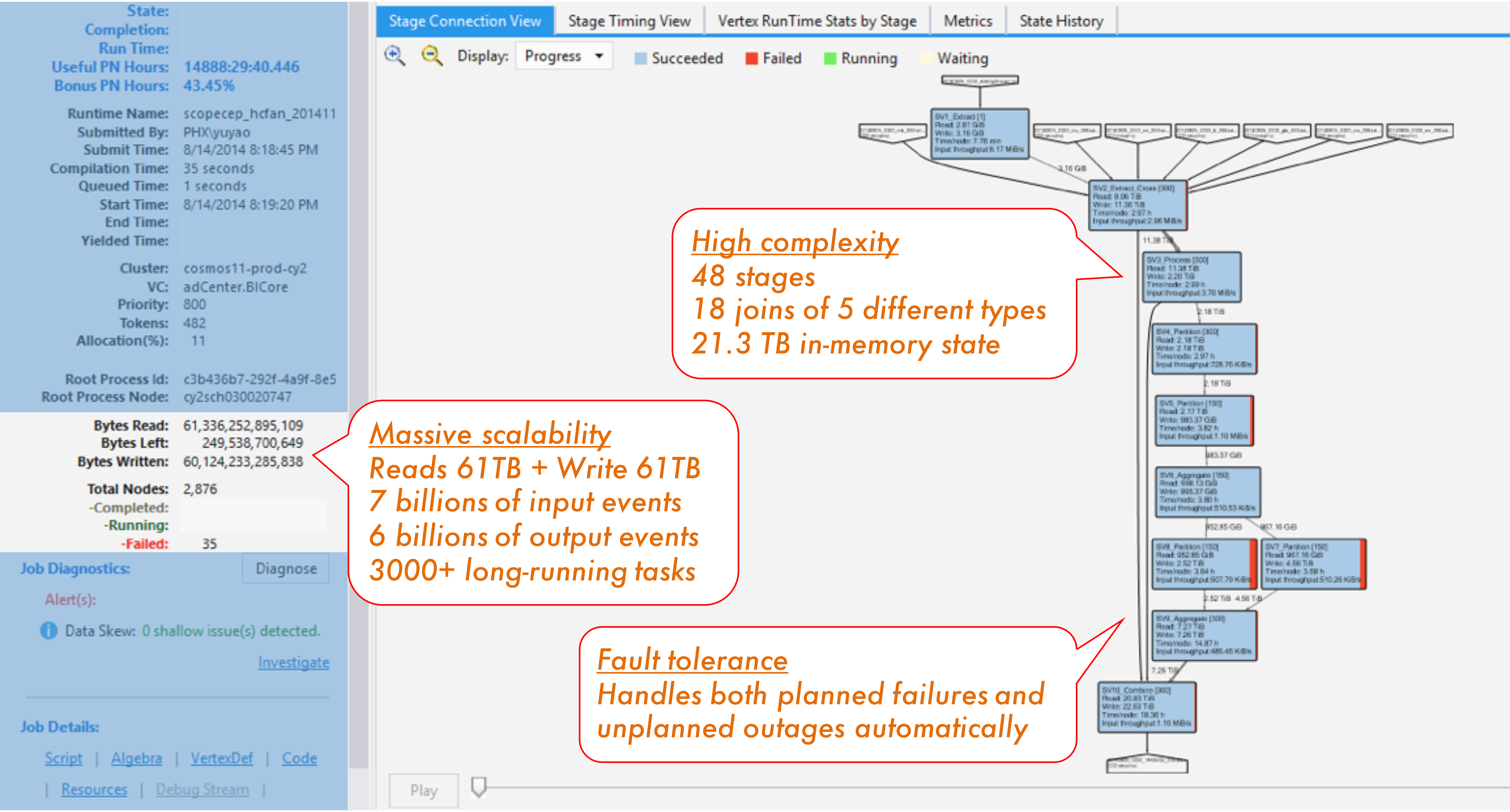
Microsoft

*Now with Alibaba Group

@NSDI'16

2015 天猫双11全球狂欢节
2015 TMALL 11.11 GLOBAL SHOPPING FESTIVAL
双11全球狂欢节
2015 11.11 GLOBAL SHOPPING FESTIVAL
00:35:31
¥20,428,378,110
$3,211,908,133
GMV Generated in USD on 2015 11.11
无线成交占比:74%
Mobile GMV Share
初步数据，未经审计。All data on the screen are preliminary and unaudited
全球交易国家/地区排行
Top Countries/Regions by GMV
全球商品热卖榜
Top Import Products and Top Export Product Categories
193
已成交国家/地区总数
# of Countries/Regions with Completed Transactions
"一带一路"沿线国家/地区交易热
"Belt and Road Initiative" Countries/Regions Transaction
海外旗舰店和进口国家
Overseas Merchant Flagship Stores and Top Markets Selling

A new transaction got reflected in the output within 3s
The system processed up to 50 million events/s



State:
Completion:
Run Time:
Useful PN Hours: 14888:29:40.446
Bonus PN Hours: 43.45%

Runtime Name: scopecep_hcfan_201411
Submitted By: PHX\yuyao
Submit Time: 8/14/2014 8:18:45 PM
Compilation Time: 35 seconds
Queued Time: 1 seconds
Start Time: 8/14/2014 8:19:20 PM
End Time:
Yielded Time:

Cluster: cosmos11-prod-cy2
VC: adCenter.BICore
Priority: 800
Tokens: 482
Allocation(%): 11

Root Process Id: c3b436b7-292f-4a9f-8e5
Root Process Node: cy2sch030020747

Bytes Read: 61,336,252,895,109
Bytes Left: 249,538,700,649
Bytes Written: 60,124,233,285,838

Total Nodes: 2,876
-Completed:
-Running:
-Failed: 35

Job Diagnostics: Diagnose

Alert(s):

Data Skew: 0 shallow issue(s) detected.

Investigate

Job Details:

Script | Algebra | VertexDef | Code

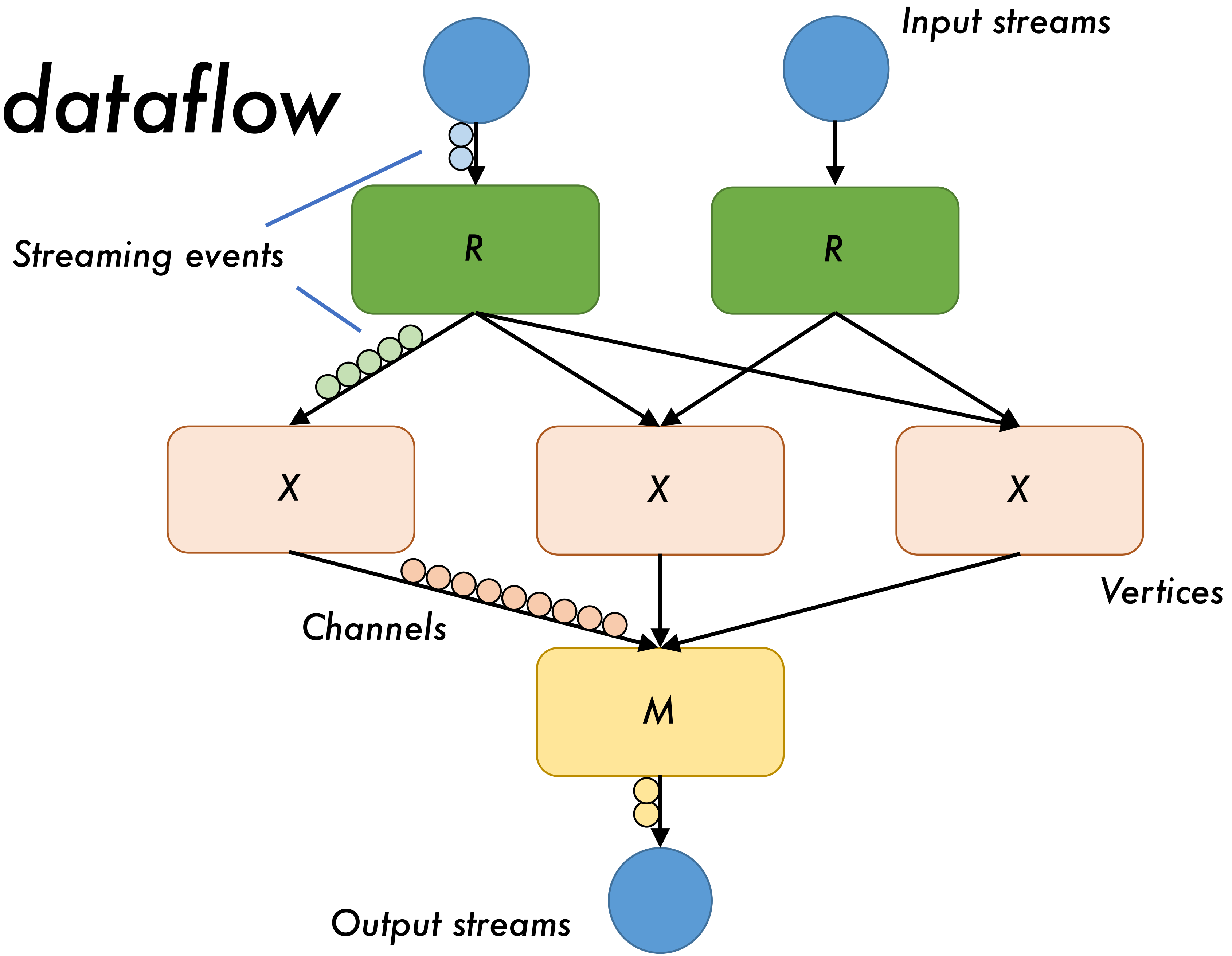
| Resources | Debug Stream |

High complexity
48 stages
18 joins of 5 different types
21.3 TB in-memory state

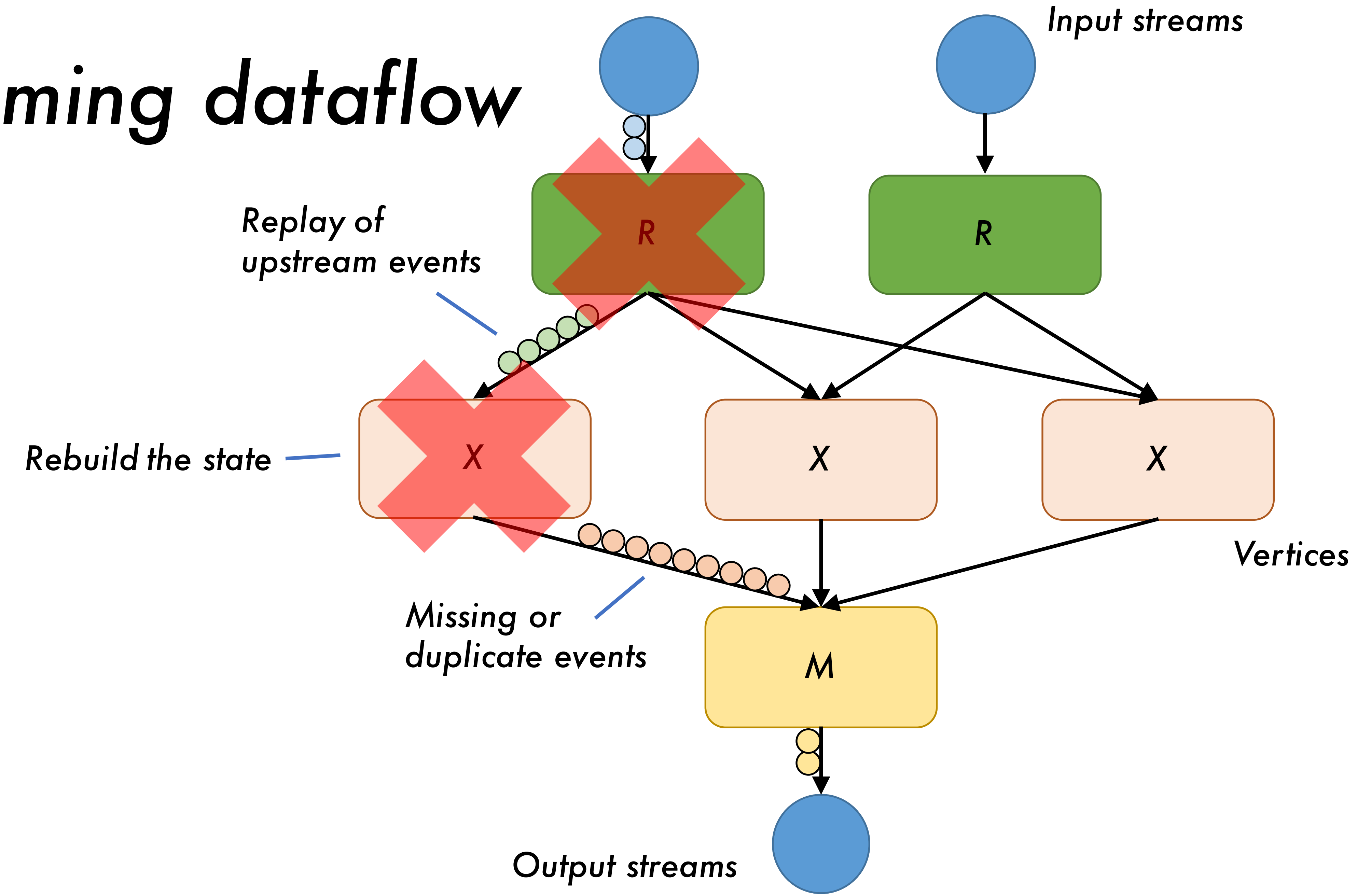
Massive scalability
Reads 61TB + Write 61TB
7 billions of input events
6 billions of output events
3000+ long-running tasks

Fault tolerance
Handles both planned failures and unplanned outages automatically

Streaming dataflow

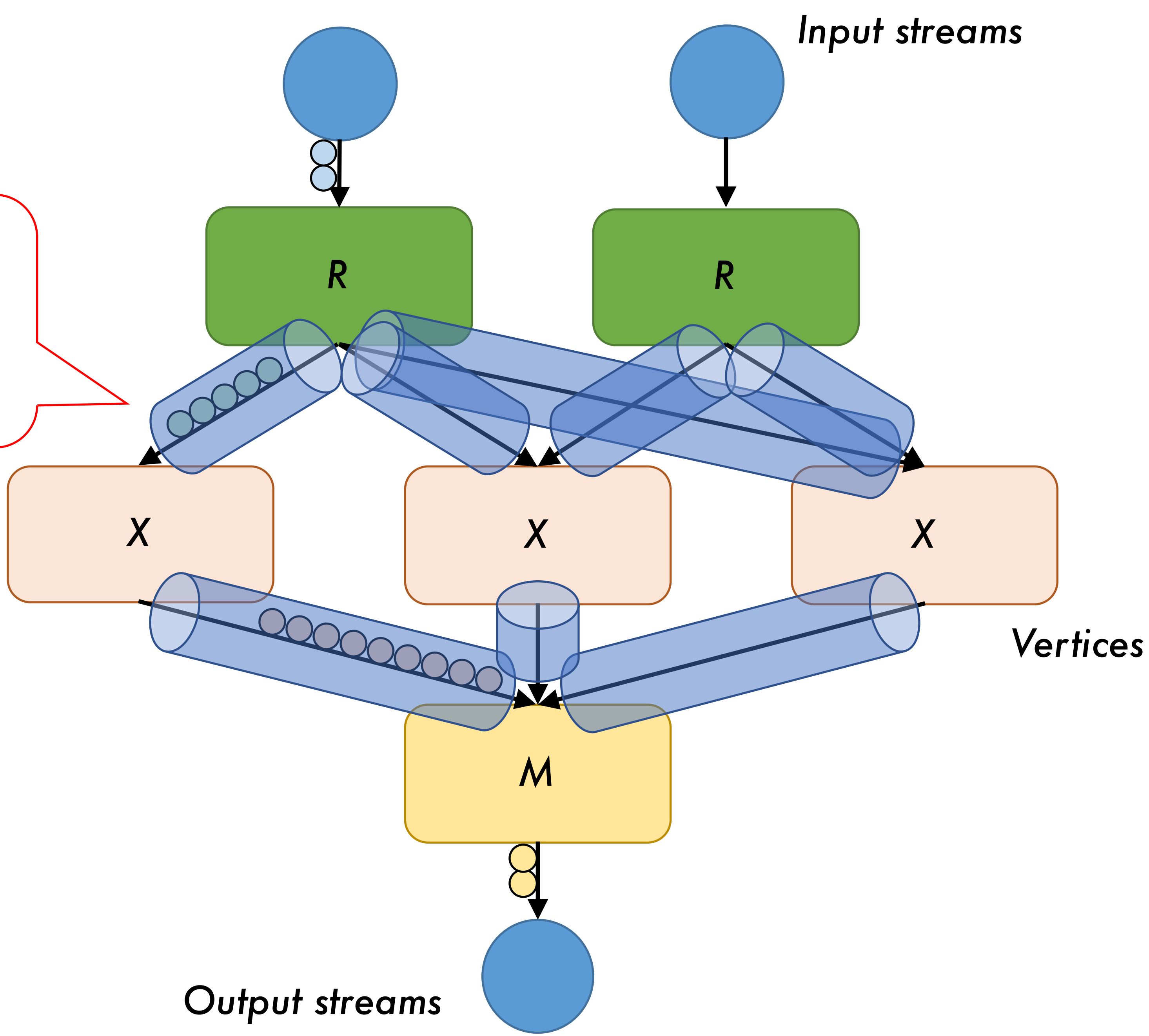


Streaming dataflow

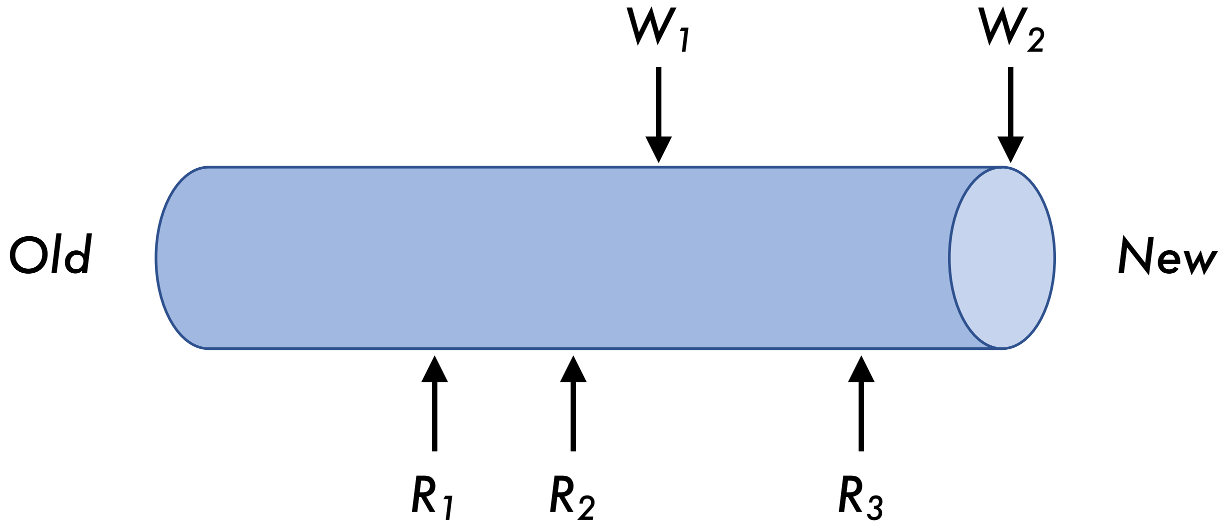


Decoupling

rStream
Provides the illusion of reliable and asynchronous communication channels

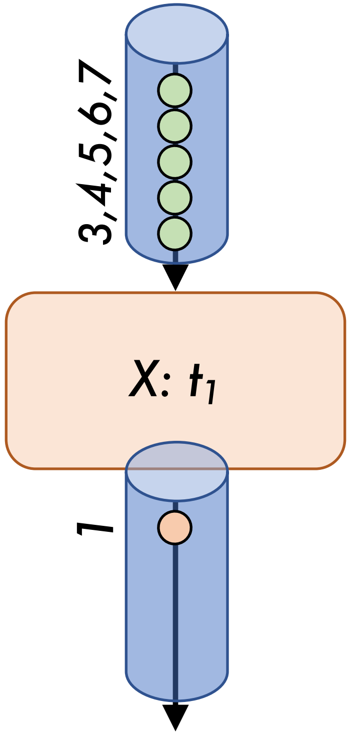


rStream



- Properties
  - There is a unique value associated with each sequence number
  - A read returns only after a successful write, for the same `seq`
  - If a write of `(seq,e)` succeeds, then for the following reads that reach position `seq`, they eventually return `(seq,e)`

Execution of a vertex

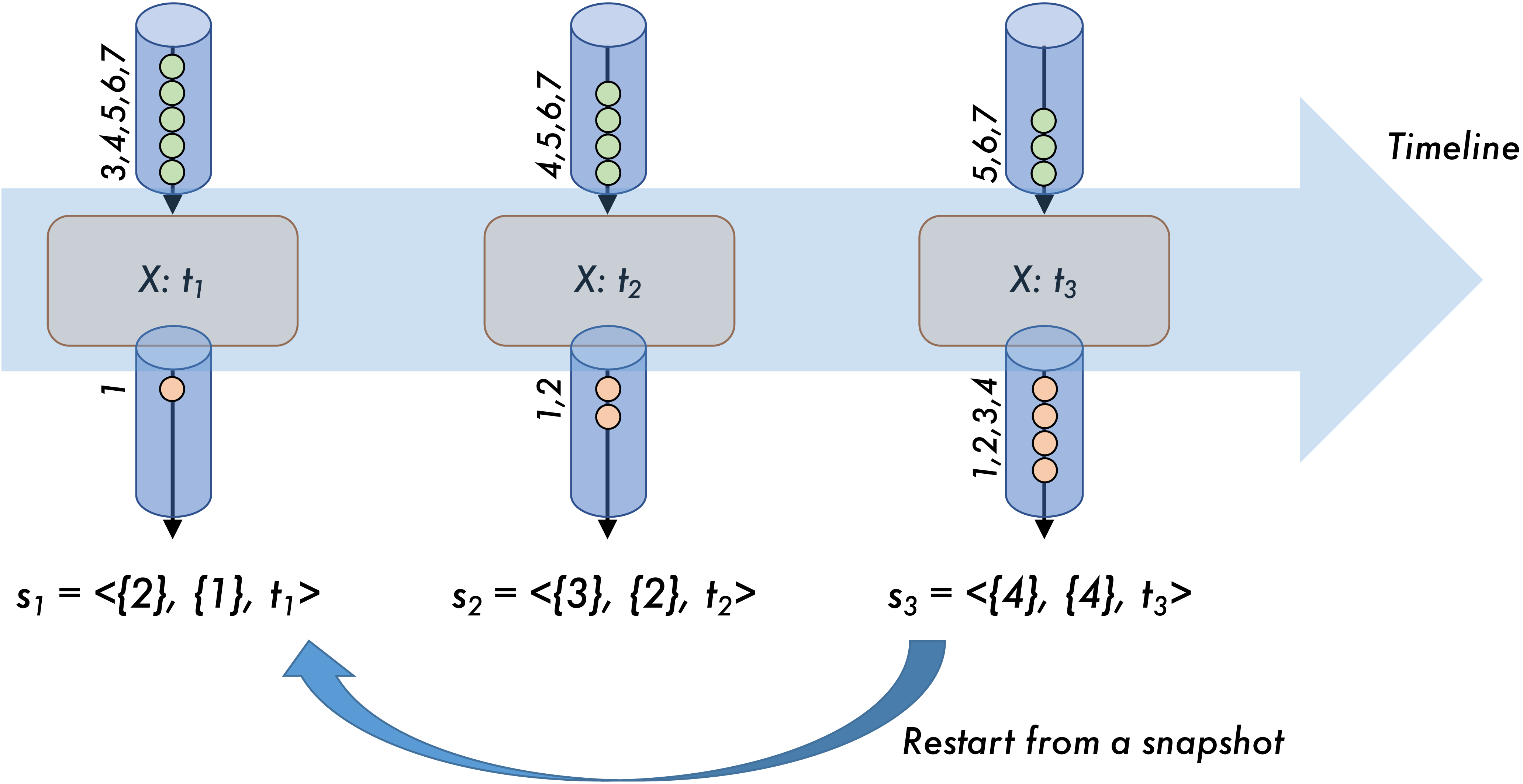


s_1 = <{2}, {1}, t_1>

<u>Snapshot</u>

- Sequence numbers of its input streams
- Sequence numbers of its output streams
- Computation state

rVertex

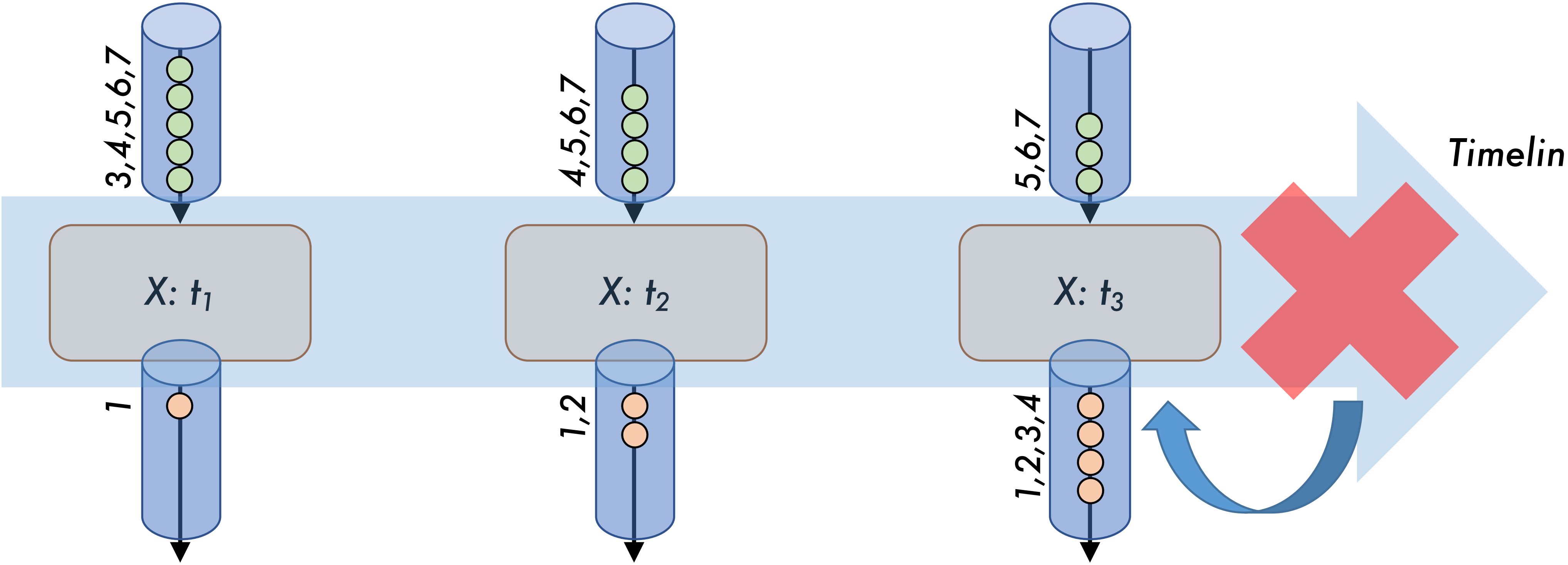


s_1 = <{2}, {1}, t_1>

s_2 = <{3}, {2}, t_2>

s_3 = <{4}, {4}, t_3>

Failure recovery



s_1 = <{2}, {1}, t_1>

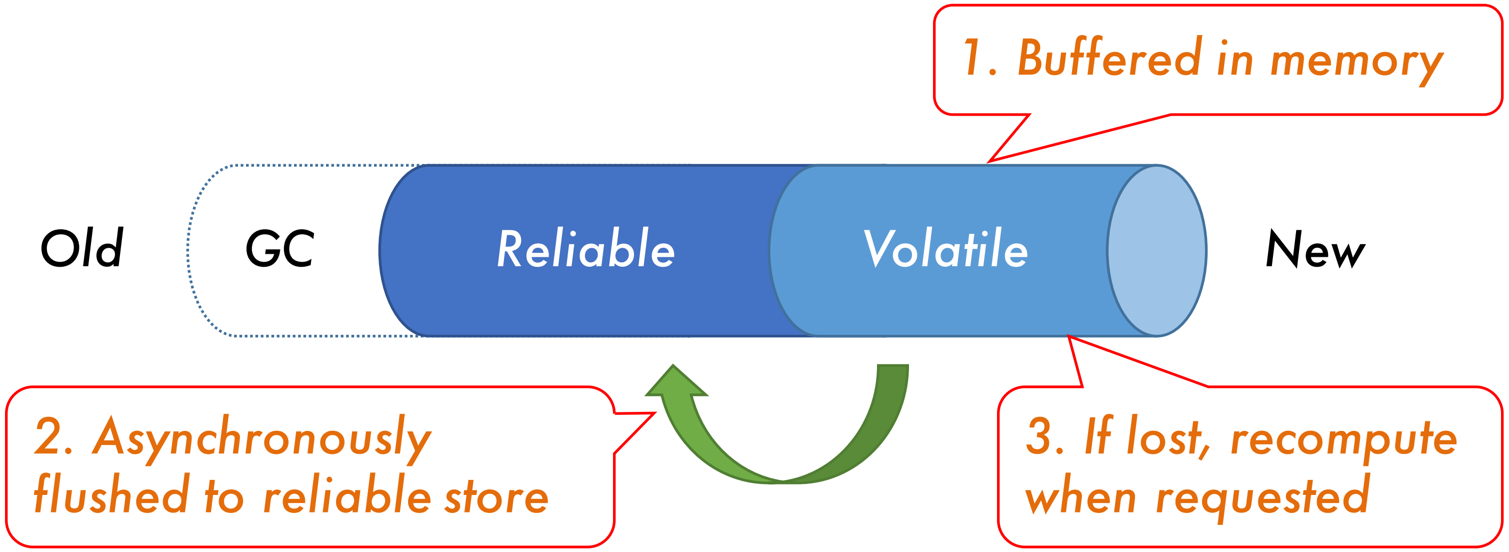
s_2 = <{3}, {2}, t_2>

s_3 = <{4}, {4}, t_3>

Optimization

- *Naïve implementation of rStream: writing events to reliable store*
  - *Synchronous writes introduce significant latencies*

- *Uses a hybrid scheme that moves writes out of the critical path while providing the illusion of reliable channels*



1. Buffered in memory

2. Asynchronously flushed to reliable store

3. If lost, recompute when requested

Different failure recovery strategies

- Recomputation using dependency tracking at runtime
- Checkpoint/log replay
- Persistent state/streams
- Hybrid

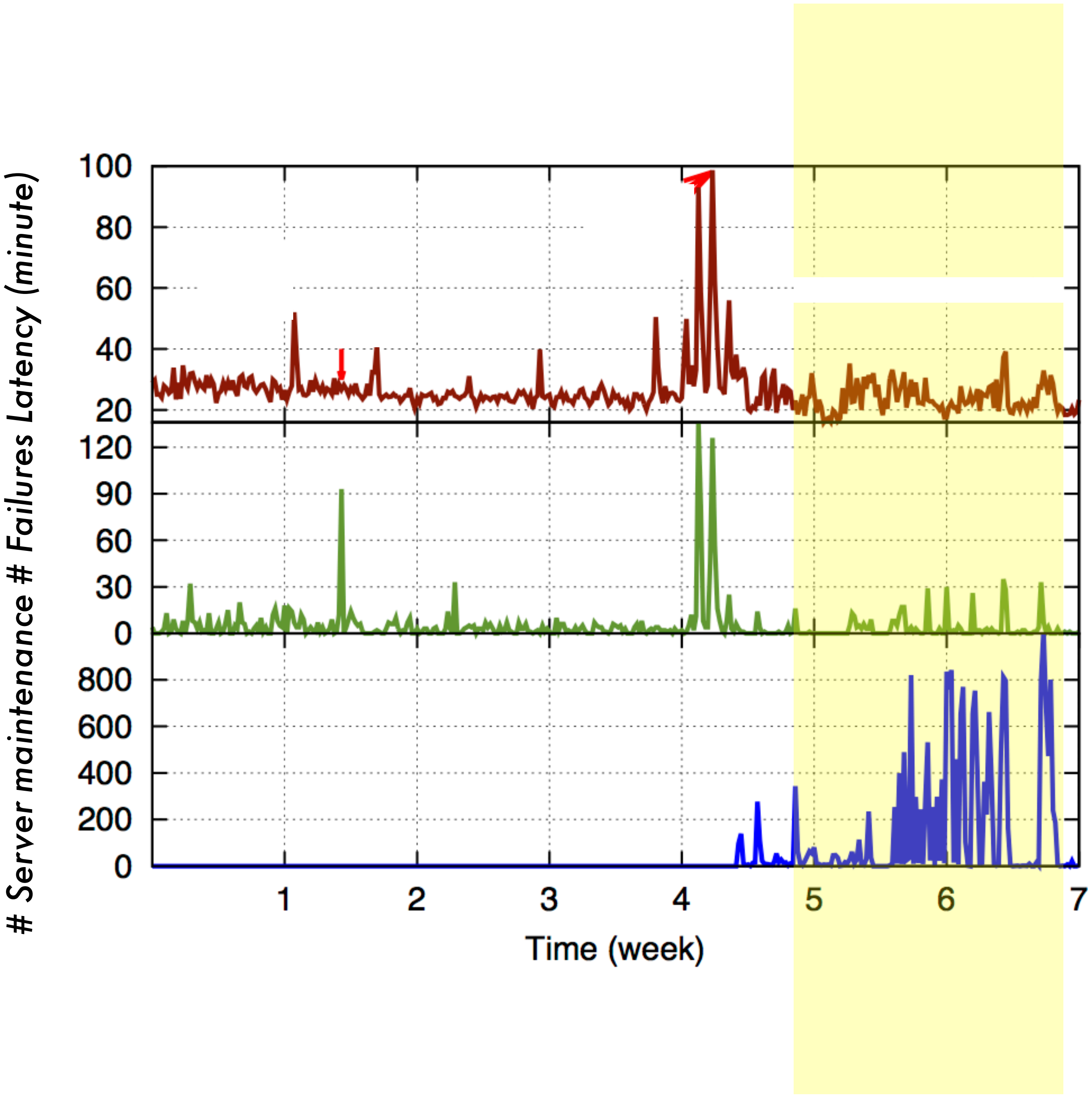
Development/debugging

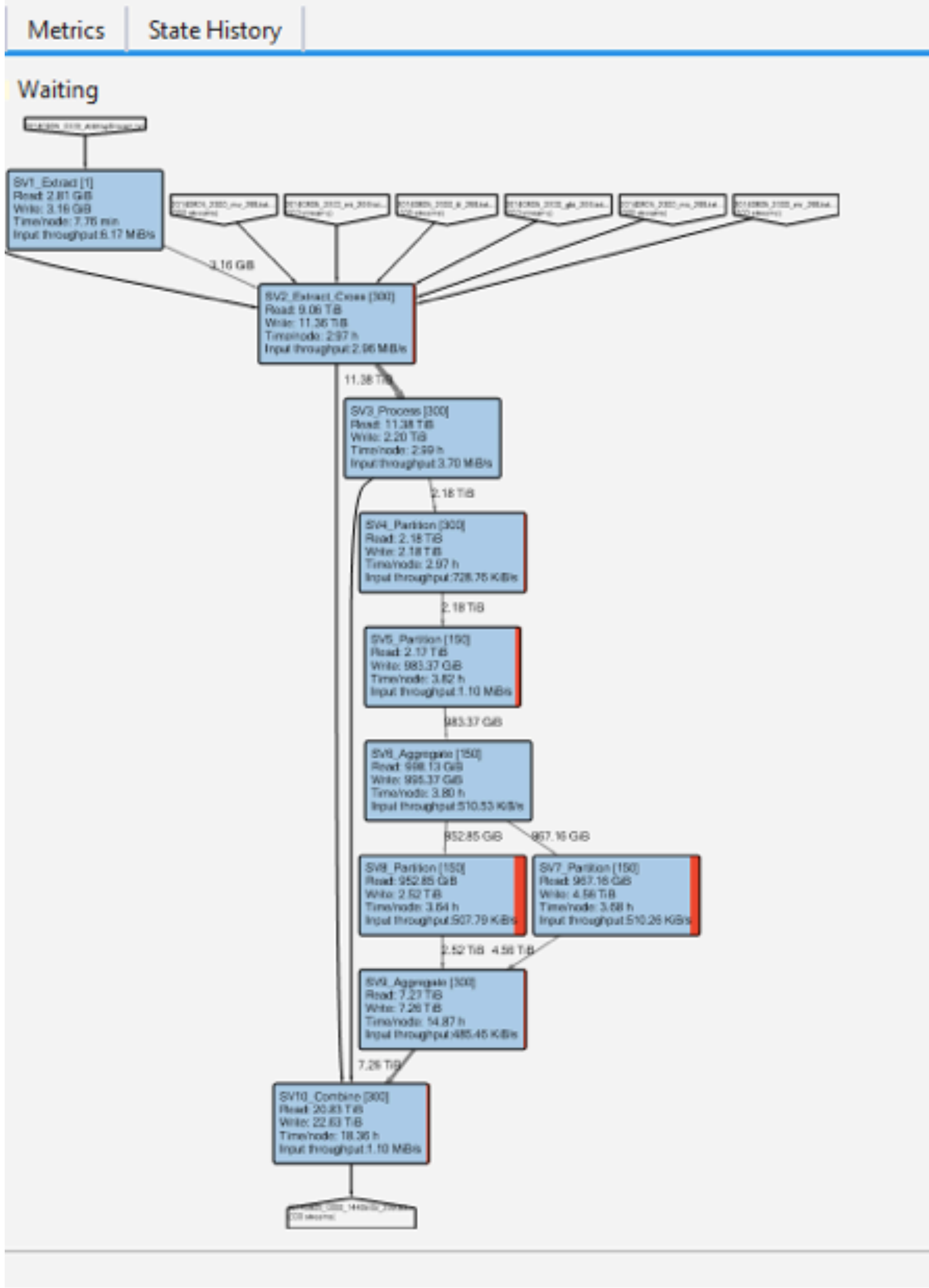
- *Greatly leveraged and tightly integrated with existing system*
  - *Integrated language, optimizer, scheduling, etc.*

- *Distributed streaming made easy*
  - *Off-line mode: starting with finite inputs with minimum resources to validate/debug a streaming application*
  - *Later switched to on-line, live execution transparently*
  - *Greatly improves developer productivity in lifecycle of an application*
    - *E.g., Can even debug/profile a vertex without impacting the running job*

Deployment

- *Re-examination of segments of execution in the past for auditing*
- *Dynamic scaling and robustness to load fluctuation*
- *Continuous operation during system maintenance*
- *Straggler handling*
- *Dynamic reconfiguration/patching to resolve data anomalies*





SV1_Extract [1]
Read: 2.81 GiB
Write: 3.16 GiB
Time/node: 7.76 min
Input throughput:6.17 MiB/s

SV2_Extract_Cross [300]
Read: 9.06 TiB
Write: 11.36 TiB
Time/node: 2.97 h
Input throughput:2.96 MiB/s

SV3_Process [300]
Read: 11.38 TiB
Write: 2.20 TiB
Time/node: 2.99 h
Input throughput:3.70 MiB/s

SV4_Partition [300]
Read: 2.18 TiB
Write: 2.18 TiB
Time/node: 2.97 h
Input throughput:728.75 KiB/s

SV5_Partition [150]
Read: 2.17 TiB
Write: 983.37 GiB
Time/node: 3.82 h
Input throughput:1.10 MiB/s

SV6_Aggregate [150]
Read: 998.13 GiB
Write: 995.37 GiB
Time/node: 3.80 h
Input throughput:510.53 KiB/s

SV8_Partition [150]
Read: 952.85 GiB
Write: 2.52 TiB
Time/node: 3.64 h
Input throughput:507.79 KiB/s

SV7_Partition [150]
Read: 967.16 GiB
Write: 4.56 TiB
Time/node: 3.68 h
Input throughput:510.26 KiB/s

SV9_Aggregate [300]
Read: 7.27 TiB
Write: 7.26 TiB
Time/node: 14.87 h
Input throughput:485.45 KiB/s

SV10_Combine [300]
Read: 20.83 TiB
Write: 22.63 TiB
Time/node: 18.36 h
Input throughput:1.10 MiB/s

Conclusion

- *Cloud-scale stream computation is challenging due to the complexity of dependencies*

- *StreamScope introduces two new abstractions, rVertex and rStream, to manage the complexity through decoupling*

- *The abstractions separate system properties from the actual implementation to,*
    - *Enable powerful optimizations*
    - *Develop different failure recovery strategies*
    - *Better support the lifecycle of streaming applications in production*