State management in distributed stream processing systems

Kostas Magoutis

Computer Science Department
University of Crete, Greece

Institute of Computer Science (ICS)
Foundation for Research and Technology – Hellas (FORTH)

FORTH
Institute of Computer Science

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
UNIVERSITY OF CRETE

Agenda

- Distributed stream processing
  - What is state? How is it typically managed?

- Fault tolerance
  - Types of checkpointing
  - Focus on continuous incremental checkpointing
    - CEC
    - LinkedIn Samza
    - Recent experience with Samza

- Exactly-once semantics

References

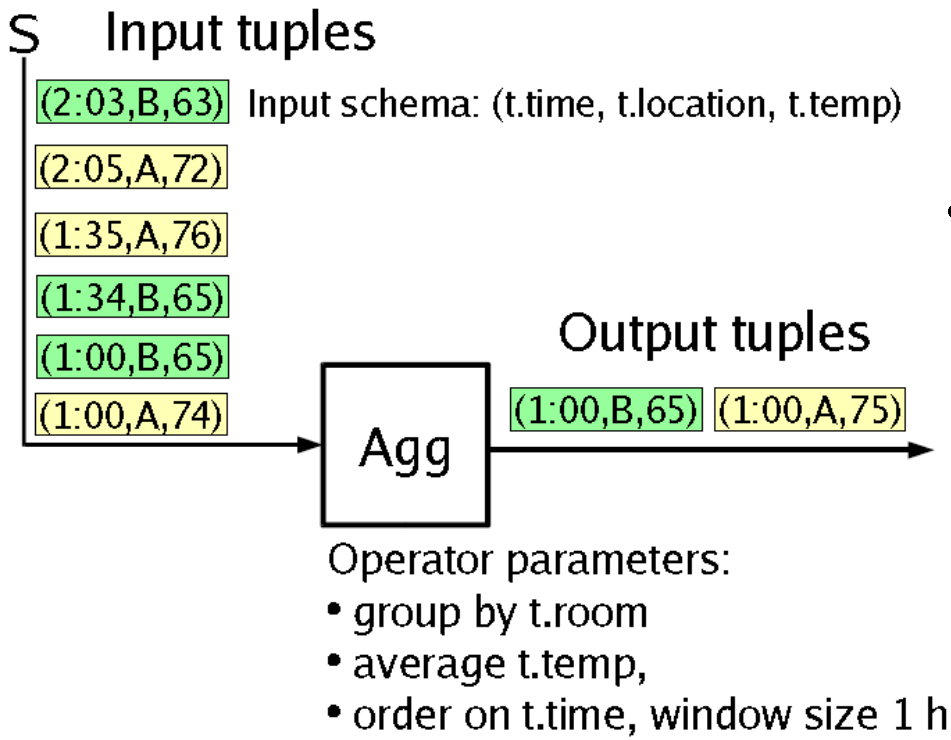
- Z. Sebepou, K. Magoutis. "Continuous Eventual Checkpointing for Data Stream Processing Operators", IEEE DSN 2011, Hong Kong, China, July 6-9, 2011

- S. Noghabi, K. Paramasivam, Y. Pan, N. Ramesh, J. Bringhurst, I. Gupta, R. H. Campbell. "Samza: stateful scalable stream processing at LinkedIn", Proc. VLDB Endow. 10, 12, Aug. 2017

- P. Carbone, S. Ewen, G. Fóra, S. Haridi, S. Richter, K. Tzoumas. "State management in Apache Flink: consistent stateful distributed stream processing", Proc. VLDB Endow. 10, 12, Aug. 2017

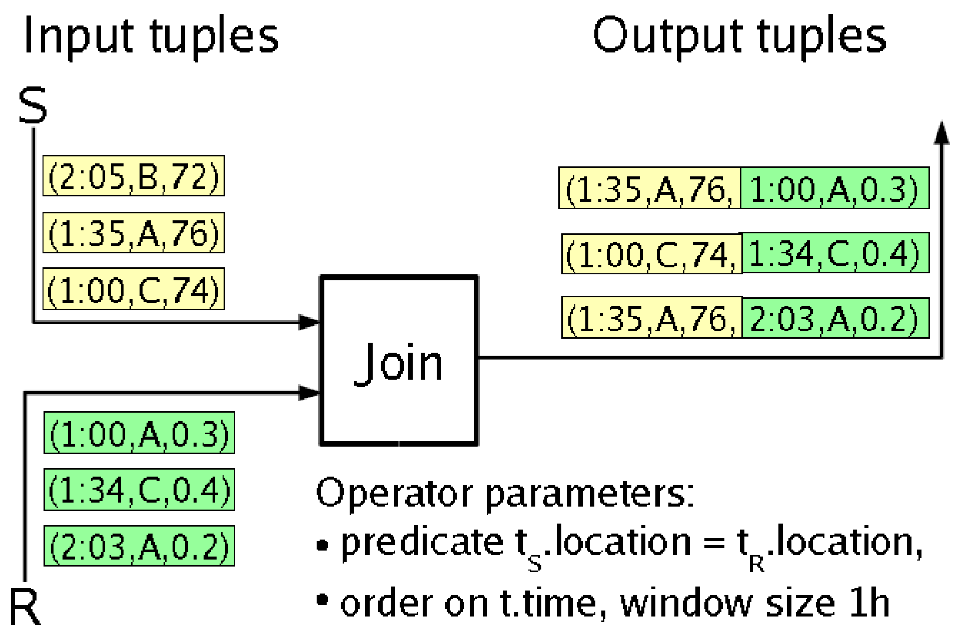
- A. Chronarakis, A. Papaioannou, K. Magoutis, "On the impact of log compaction on incrementally checkpointing stateful stream-processing operators", Proc. DRSS'19, to be held in conjunction with SRDS'19, Lyon, France, October 1, 2019

General principles: Aggregate operator



- Operator state (per window) may be
  - One value (accumulating state)
  - All tuples that enter the window

General principles: Join operator



Fault-tolerance in stream processing systems

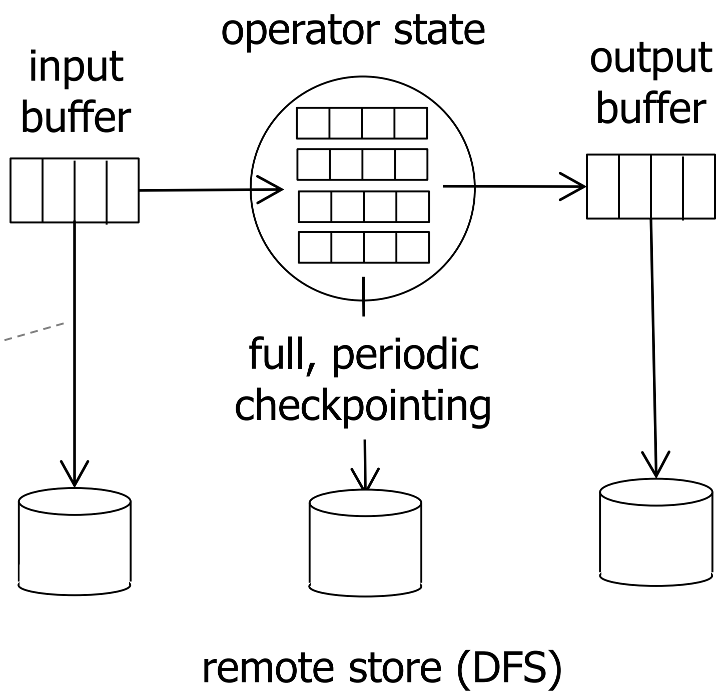
- State replication
  - Maintain full replicas of operator state across nodes
  - High availability, memory requirements

- Checkpoint roll-backward
  - Checkpoint to remote disk
  - On recovery, load most recent checkpoint

- Types of checkpointing
  - Full, periodic
  - Delta (incremental), periodic
  - Continuous incremental (log of updates)

Full, periodic checkpoints

Trim after operator checkpoints



Store durably only

- for replay at a later time
- spill-over

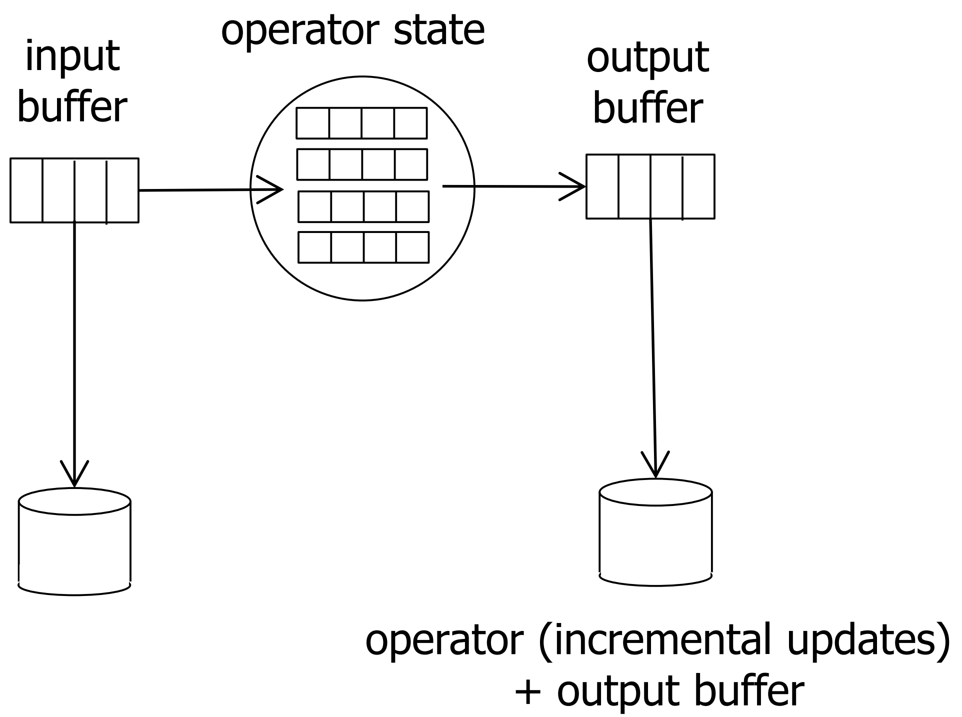
Operator freezes during checkpoint

- Large response-time spikes

Efficient implementations use copy-on-write (COW)

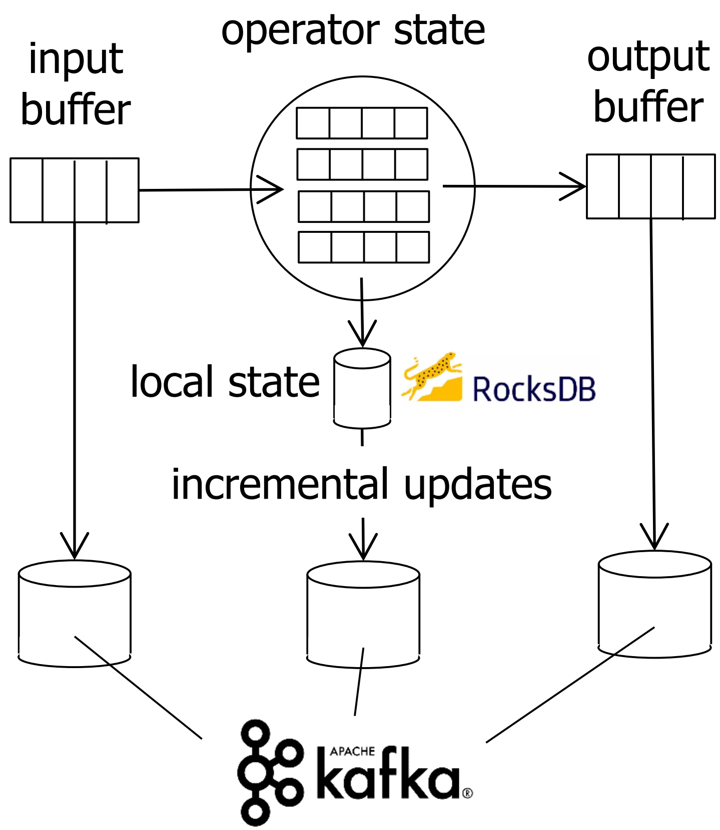
- Complex to implement
- Overhead to compute what needs to be checkpointed
- Overhead handling exceptions during protection fault

Incremental checkpointing (CEC, DSN'11)

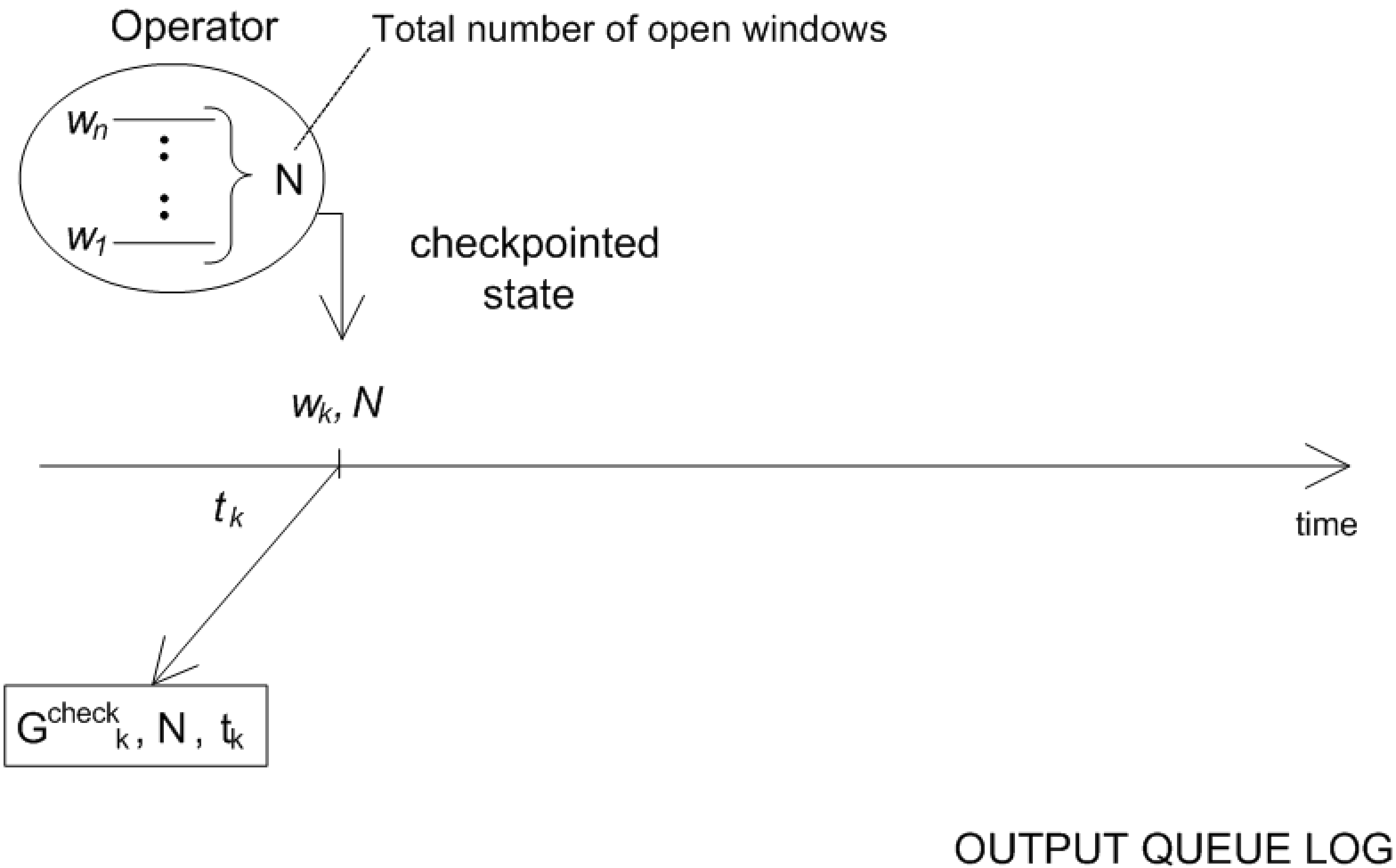


Z. Sebepou, K. Magoutis, Continuous eventual checkpointing for data stream processing operators, in *Proc. of IEEE DSN'11*

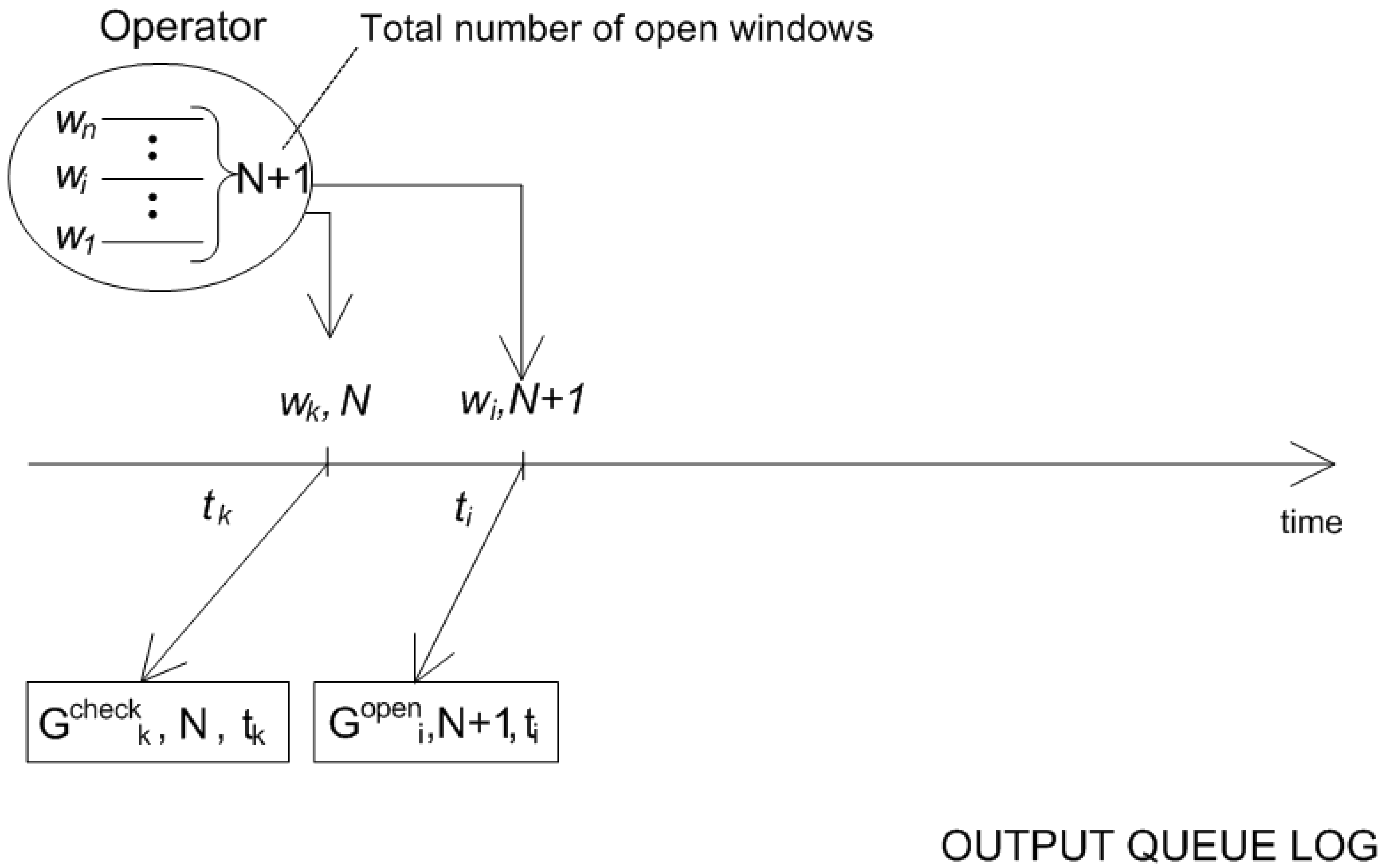
Incremental + local state



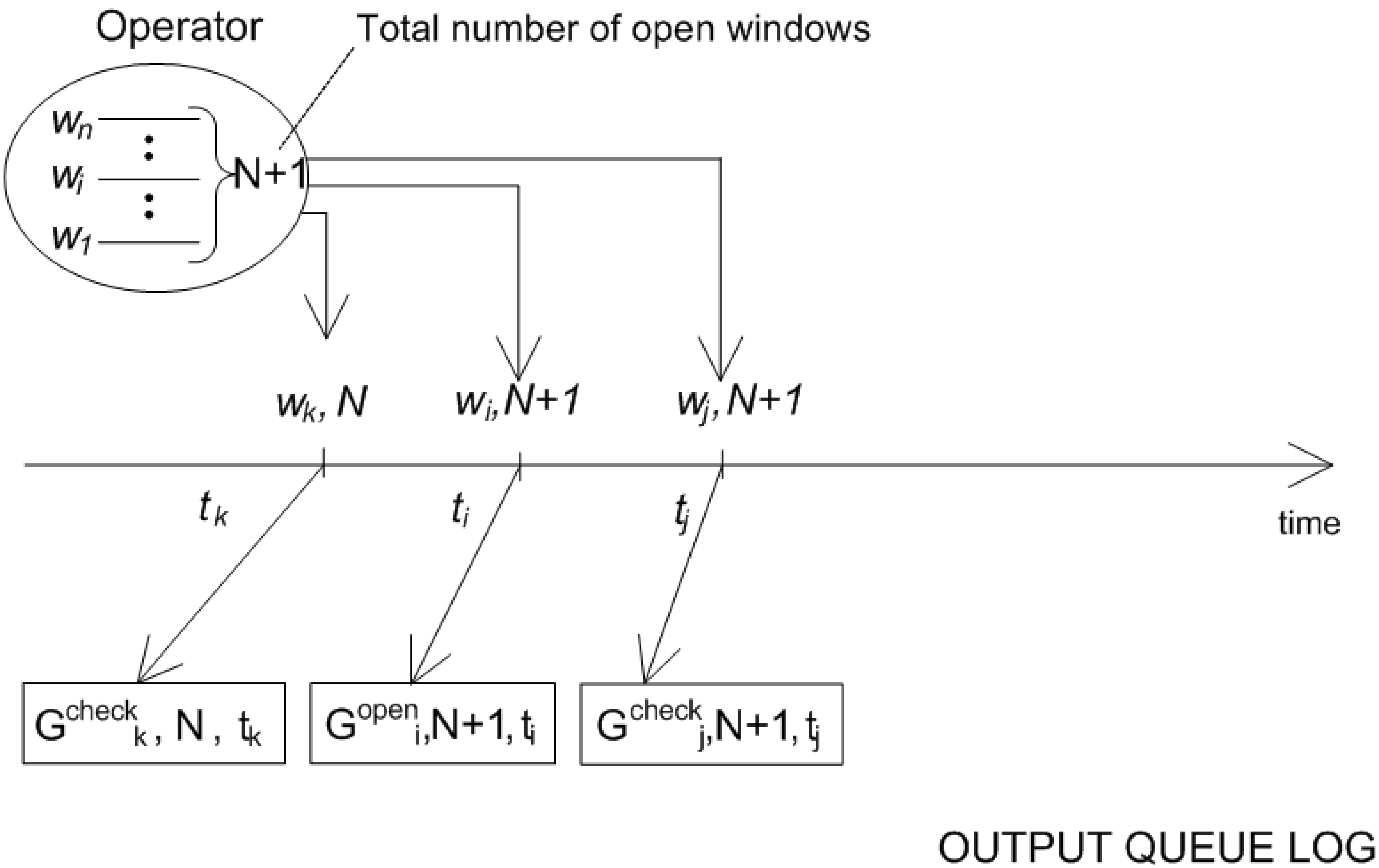
Continuous eventual checkpointing (CEC)



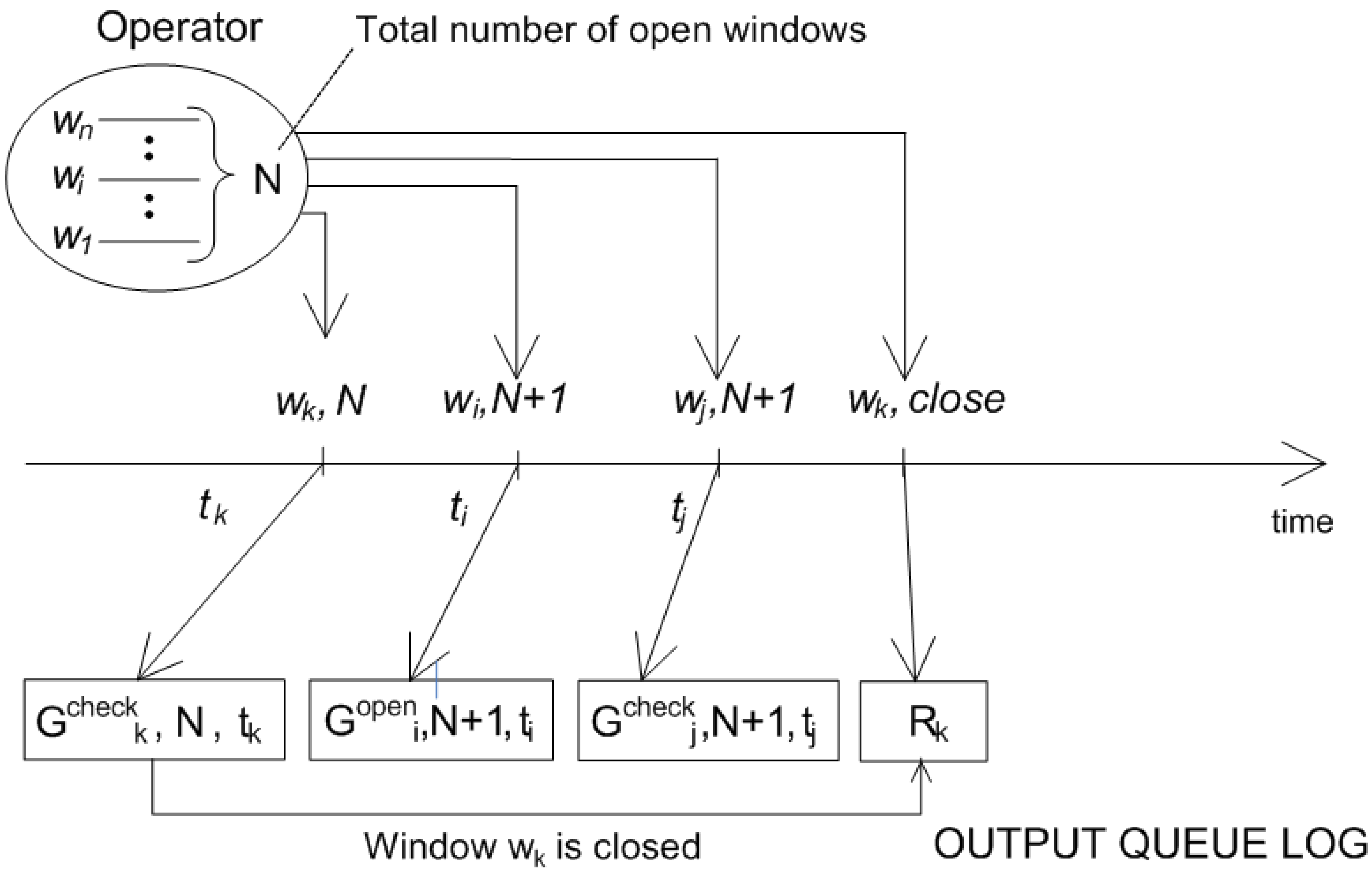
Opening of a new window



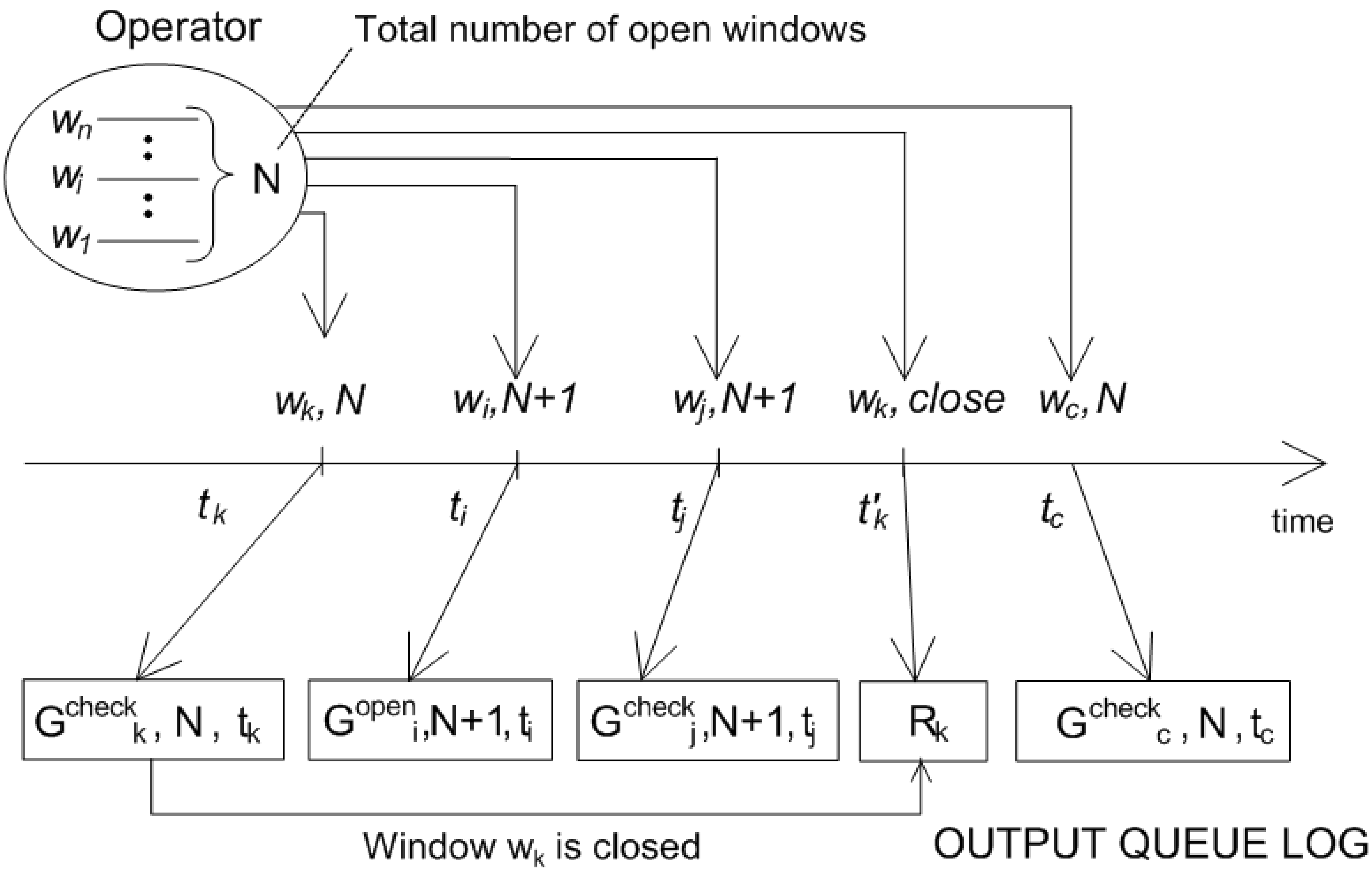
Another checkpoint of an open window



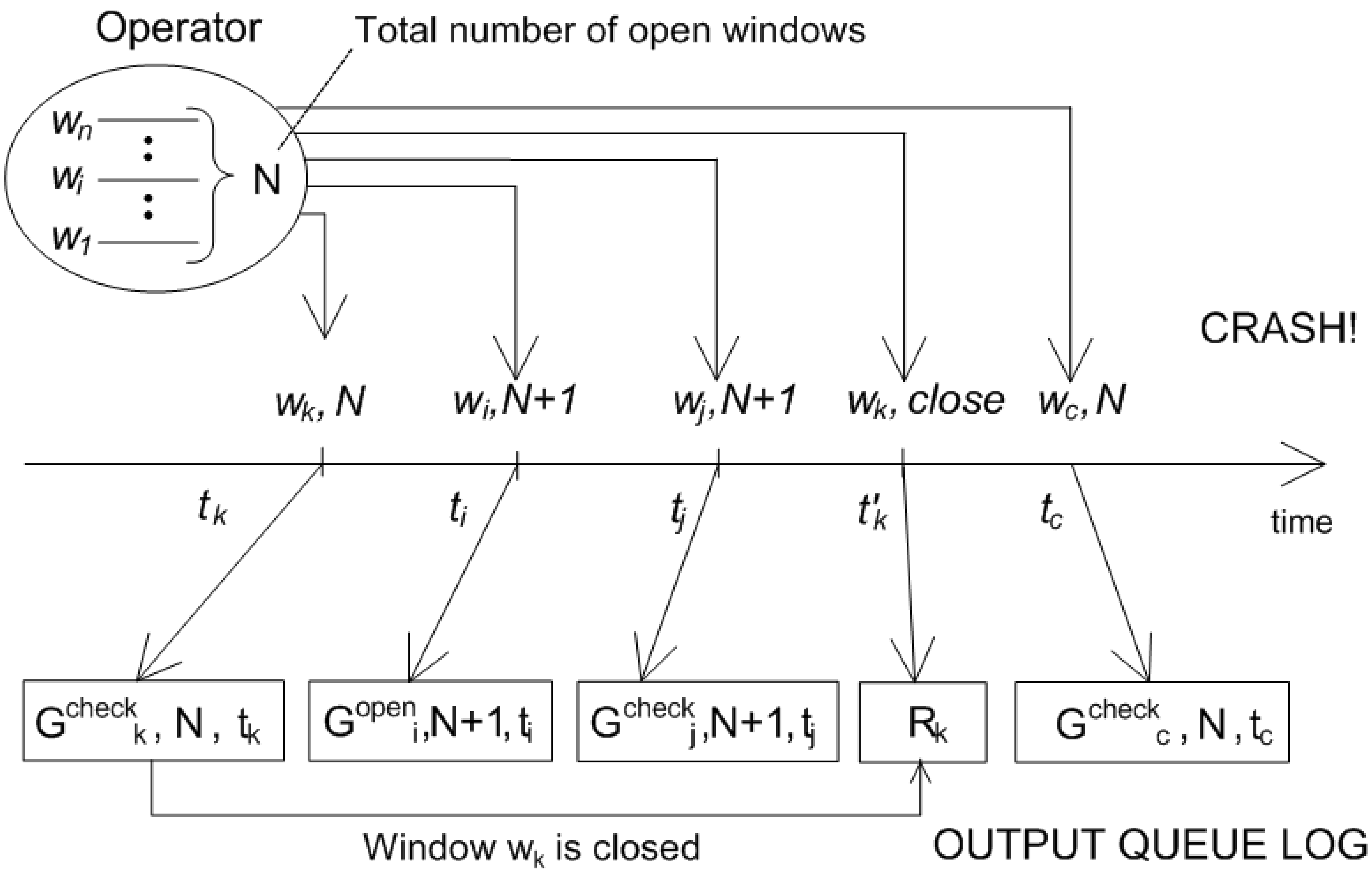
Closing of a window



Another checkpoint of an open window

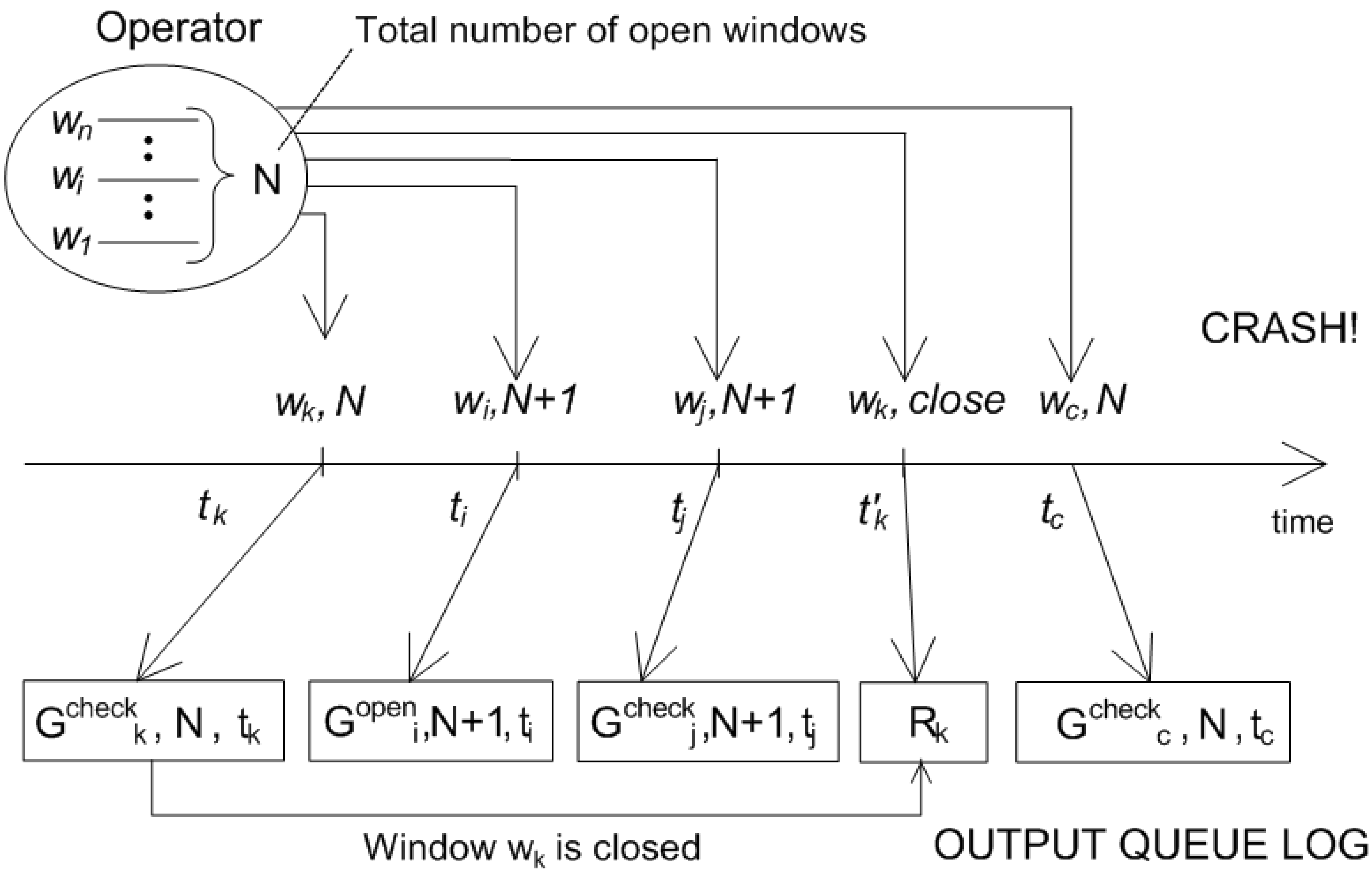


Crash



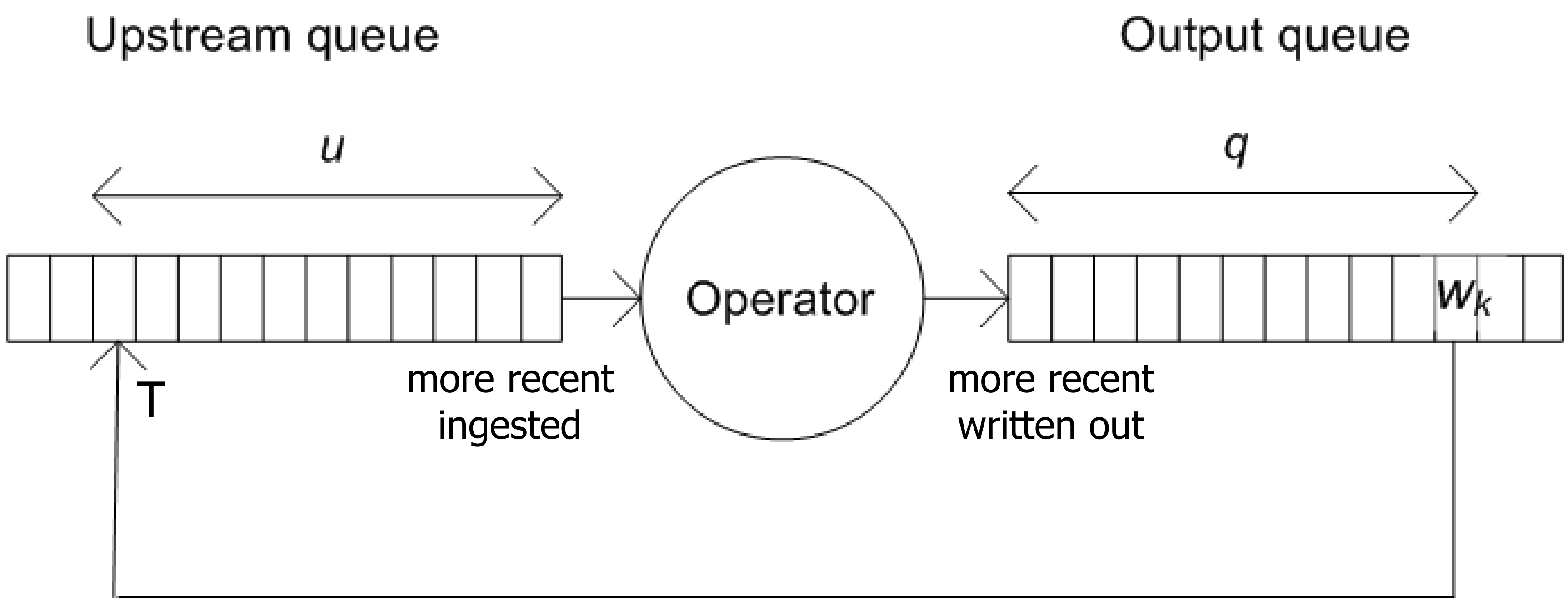
Operator
Total number of open windows
w_n
w_i
w_1
N
CRASH!
w_k, N
w_i, N+1
w_j, N+1
w_k, close
w_c, N
t_k
t_i
t_j
t'_k
t_c
time
G^{check}_k, N, t_k
G^{open}_i, N+1, t_i
G^{check}_j, N+1, t_j
R_k
G^{check}_c, N, t_c
Window w_k is closed
OUTPUT QUEUE LOG

Recovery



Operator
Total number of open windows
w_n
w_i
w_1
N
CRASH!
w_k, N
w_i,N+1
w_j,N+1
w_k, close
w_c, N
t_k
t_i
t_j
t'_k
t_c
time
G^{check}_k, N, t_k
G^{open}_i,N+1, t_i
G^{check}_j,N+1, t_j
R_k
G^{check}_c, N, t_c
Window w_k is closed
OUTPUT QUEUE LOG

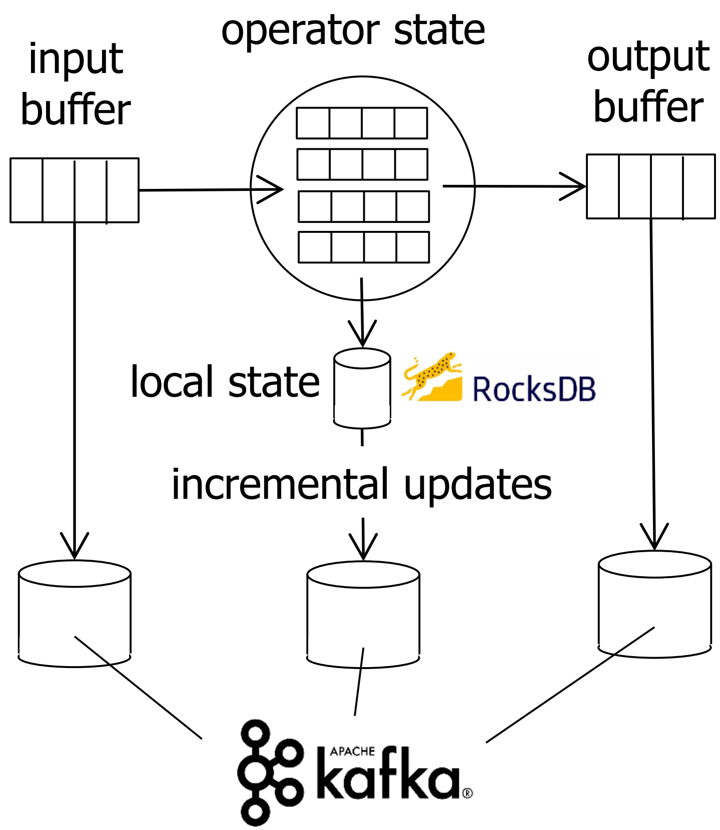
Overall view



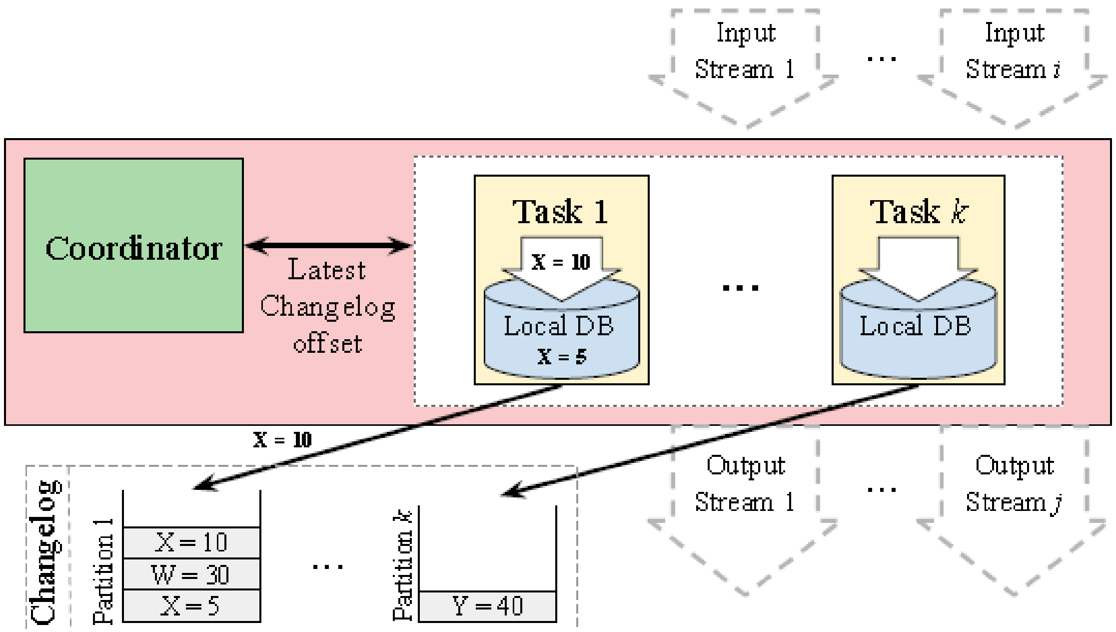
- Window w_k has oldest checkpoint in output queue log
- Producing a checkpoint for it will reduce *q*
- It will also reduce the number of tuples to replay *u*

Incremental checkpointing: LinkedIn Samza

Incremental + local state



Use of changelog



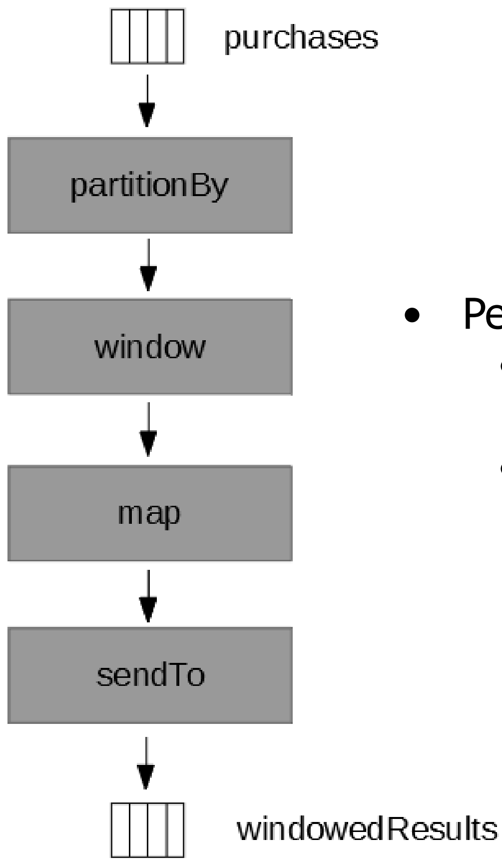
Sync changelog, then update "successfully processed" input offset (=>*at least once*)

S. Noghabi et al. "Samza: stateful scalable stream processing at LinkedIn", Proc. VLDB Endow. 10, 12, Aug. 2017

Recent experience with incremental checkpointing in Samza

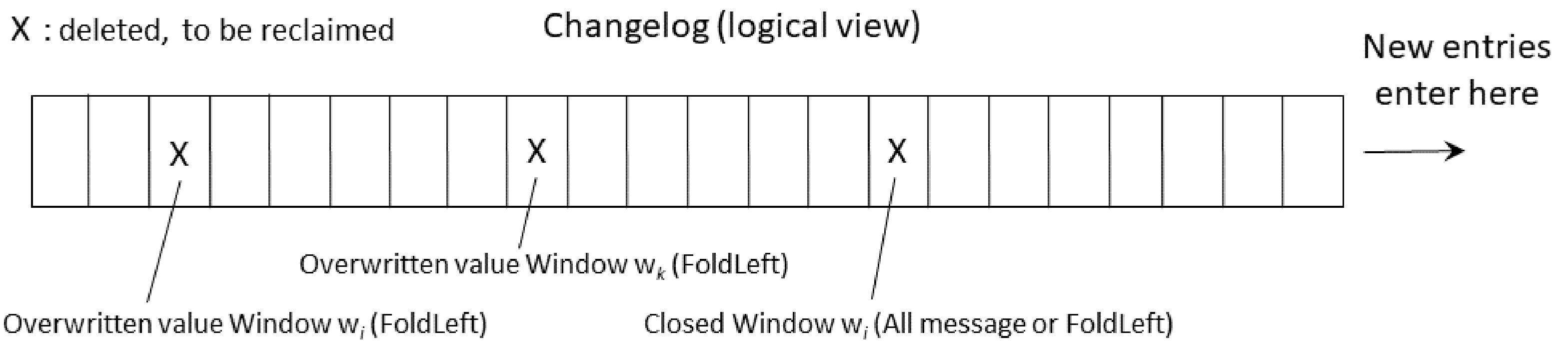
Window-based streaming application

purchases.
partitionBy(...).
window(...).
map(...).
sendTo(windowedResults);



- Per-window operator state may contain
  - One value (accumulating state)
    - FoldLeftFunction (FLF)
  - All tuples that enter the window
    - Retain all (RetainALL)

Changelog and compaction



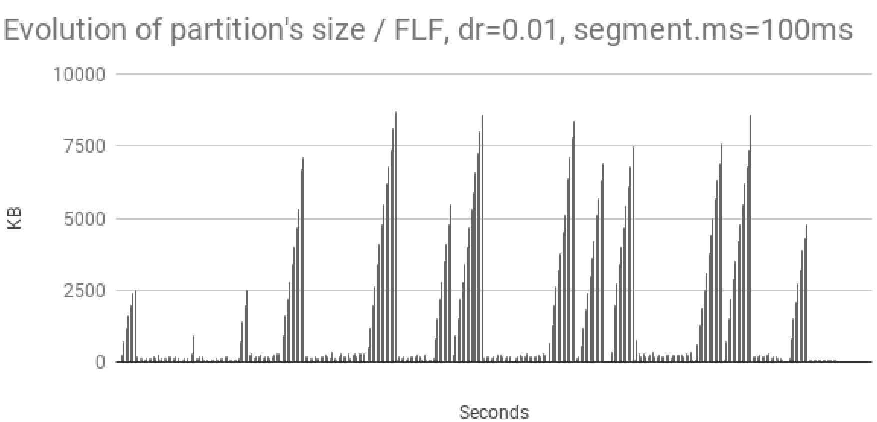
Research questions

- How does recovery time depend on changelog size?
- What policies can be used to limit changelog size ?

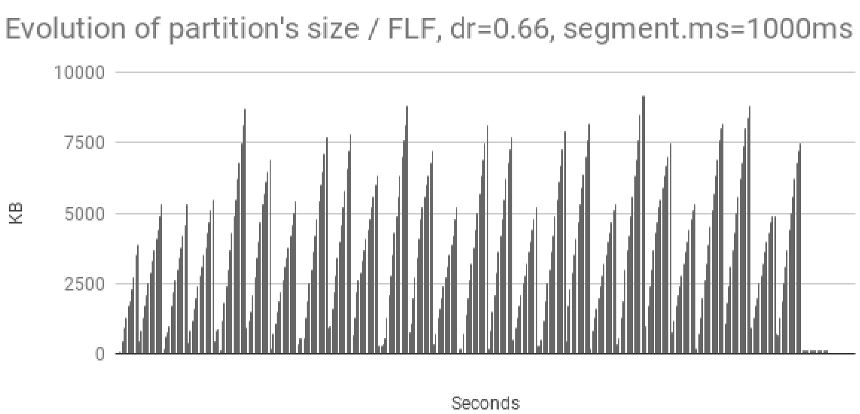
Changelog-size vs. overhead

- Compaction parameters (policies) affect
  - Size of changelog, CPU usage of broker
- Trade-off between restore time and overhead

Aggressive compaction



Relaxed compaction



- Experiments with segment.ms= {100ms,1000ms} & dirty ratio= {0.01, 0.33, 0.66}

Overall

- Appropriate tuning of compaction configuration parameters needed to achieve recovery time goals
- What about (exactly-once / at-least-once) semantics?

Exactly-once semantics

- Flink’s pipelined (asynchronous barrier) checkpointing
- Reminiscent of Chandy-Lamport global snapshots
  - Inject markers at input streams
  - Take local operator snapshots after having accounted for all input prior to snapshot time

References

- Z. Sebepou, K. Magoutis. “Continuous Eventual Checkpointing for Data Stream Processing Operators”, IEEE DSN 2011, Hong Kong, China, July 6-9, 2011

- S. Noghabi, K. Paramasivam, Y. Pan, N. Ramesh, J. Bringhurst, I. Gupta, R. H. Campbell. “Samza: stateful scalable stream processing at LinkedIn”, Proc. VLDB Endow. 10, 12, Aug. 2017

- P. Carbone, S. Ewen, G. Fóra, S. Haridi, S. Richter, K. Tzoumas. “State management in Apache Flink: consistent stateful distributed stream processing”, Proc. VLDB Endow. 10, 12, Aug. 2017

- A. Chronarakis, A. Papaioannou, K. Magoutis, “On the impact of log compaction on incrementally checkpointing stateful stream-processing operators”, Proc. DRSS'19, to be held in conjunction with SRDS’19, Lyon, France, October 1, 2019

Questions?

FORTH
Institute of Computer Science

UNICORN

H2020 GA no. 731846 EU project

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
UNIVERSITY OF CRETE