World Journal of Advanced Research and Reviews
eISSN: 2581-9615 CODEN (USA): WJARAI
Cross Ref DOI: 10.30574/wjarr
Journal homepage: https://wjarr.com/

(Review Article)

Enhancing checkpointing and state recovery for large-scale stream processing

Shakir Poolakkal Mukkath *

Walmart Global Tech, USA.

World Journal of Advanced Research and Reviews, 2025, 26(02), 296-302

Publication history: Received on 25 March 2025; revised on 02 May 2025; accepted on 04 May 2025

Article DOI: https://doi.org/10.30574/wjarr.2025.26.2.1638

Abstract

As real-time applications demand ever-lower latencies and greater fault tolerance, traditional checkpointing mechanisms in distributed streaming systems face new performance bottlenecks. This article examines recent advancements in reducing checkpointing overhead while maintaining high availability, focusing on incremental state snapshots, asynchronous commit techniques, and log-based recovery models. It highlights the shift towards intelligent state management strategies, where adaptive checkpoint intervals and event-driven rollback mechanisms optimize resource utilization. The discussion delves into emerging storage backends that offer hybrid memory-disk approaches, enabling near-instantaneous state recovery without excessive write amplification. The article presents new perspectives on leveraging event sourcing as a state recovery alternative, where historical data streams are reprocessed dynamically to restore lost computation. Additionally, it explores targeted recovery techniques including partial state rollback, causality tracking, compensating events, and incremental recovery prioritization. These innovations collectively transform fault-tolerant stream processing by minimizing recovery scope while maintaining consistency guarantees. Through case studies and theoretical analysis, this work demonstrates how modern approaches significantly reduce recovery times and resource requirements, advancing the field of high-performance stream processing architectures suitable for mission-critical applications.

Keywords: Fault Tolerance; Stream Processing; Incremental Checkpointing; Event Sourcing; Distributed Recovery

1. Introduction

In the rapidly evolving landscape of distributed computing, real-time stream processing has emerged as a critical paradigm for handling continuous data flows. As organizations increasingly depend on low-latency analytics and instant decision-making capabilities, the fault tolerance mechanisms underpinning these systems have become focal points for innovation. This article explores recent advancements in checkpointing and state recovery techniques for large-scale stream processing frameworks, addressing the growing tension between reliability requirements and performance constraints. Recent studies have demonstrated that stream processing applications face significant challenges with consistent state management, with failure recovery accounting for a substantial portion of total system downtime in production environments [1]. The need for robust state management has intensified as stream processing adoption has grown, with distributed stream processing systems now handling substantial data rates in many enterprise deployments, according to industry surveys documented in the Journal of Internet Technology and Secured Transactions [1].

1.1. The Challenge of Modern Stream Processing

Traditional checkpointing approaches face mounting pressure as data volumes expand and latency requirements contract. The fundamental challenge lies in creating consistent snapshots of distributed state without introducing prohibitive overhead or disrupting the continuous nature of stream processing. While conventional periodic full-state

* Corresponding author: Shakir Poolakkal Mukkath

checkpoints provide simplicity, they increasingly represent a bottleneck in high-throughput environments. Research published in the Proceedings of the VLDB Endowment has documented that naive checkpointing approaches can introduce processing latency spikes during checkpoint operations in Apache Flink deployments processing substantial event volumes per second [2]. Network bandwidth utilization during checkpoint transmission has been observed to consume a significant portion of available network resources, creating resource contention that affects overall system stability according to measurements taken across multiple Flink production clusters [2]. Storage I/O contention during checkpoint persistence phases has been shown to increase average operation latency, with spikes significantly impacting service-level objectives [2]. Furthermore, recovery time objectives (RTOs) become increasingly difficult to meet as state size grows, with empirical measurements showing recovery times scaling approximately linearly with state size, making large-scale deployments with multi-terabyte state particularly challenging to operate within typical enterprise availability requirements [2].

2. Incremental State Snapshots: Reducing Checkpoint Footprint

One of the most promising developments in this domain is the advancement of incremental state snapshot techniques. Unlike traditional approaches that capture the entire application state at regular intervals, incremental snapshots identify and persist only the delta between consecutive checkpoints. Research published in the Journal of Systems Architecture has demonstrated that for typical stream processing workloads with moderate state mutation rates, incremental approaches can significantly reduce checkpoint data volume compared to full checkpoints [3]. This substantial reduction addresses one of the primary bottlenecks in high-throughput stream processing systems, allowing for more frequent checkpoints without corresponding increases in system overhead.

The implementation of delta encoding with structural sharing has proven particularly effective in production environments. By leveraging immutable data structures to avoid duplicating unchanged portions of state, researchers have documented notable storage requirement reductions in large-scale production deployments, enabling more frequent checkpoints with minimal performance impact [3]. Fine-grained dirty tracking represents another critical advancement in this domain, with instrumentation approaches that precisely identify modified regions at minimal runtime cost. Experimental evaluations have shown that optimized dirty tracking implementations introduce minimal overhead of total processing time, compared to more substantial impacts for naive approaches that rely on periodic deep comparisons of state structures [3]. Additionally, compression-aware differential algorithms have demonstrated substantial benefits by tailoring delta computation to maximize the effectiveness of subsequent compression. Tests with real-world streaming data patterns have achieved impressive compression ratios, further reducing storage and network requirements for checkpoint operations [3].

Experimental implementations in frameworks like Apache Flink and Samza have demonstrated remarkable improvements in real-world performance metrics. In benchmark scenarios processing substantial event volumes with significant state sizes, Flink's incremental checkpointing reduced checkpoint duration while substantially reducing the checkpoint data volume compared to full checkpoints [2]. These improvements translate directly to reduced latency spikes and more consistent throughput, addressing critical requirements for modern streaming applications.

3. Asynchronous Commit Techniques: Decoupling Processing from Persistence

Another significant advancement comes from rethinking the synchronization model between processing and state persistence. Asynchronous commit approaches introduce sophisticated coordination mechanisms that allow computation to proceed without waiting for checkpoint acknowledgments. Research into distributed streaming architectures has documented throughput improvements during checkpoint operations when utilizing properly implemented asynchronous commit strategies [3]. These improvements derive from the fundamental decoupling of state persistence operations from the critical processing path.

Modern implementations feature sophisticated coordination mechanisms that preserve consistency while minimizing performance impact. Two-phase barrier injection techniques allow in-flight events to complete processing before snapshot boundaries are established, ensuring consistent checkpoint state without requiring global processing pauses. Measurements of production deployments processing billions of messages daily have shown substantial checkpoint jitter reductions compared to synchronous approaches [1]. Speculative execution strategies have demonstrated particular promise by enabling processing to continue beyond checkpoint boundaries while maintaining the ability to roll back if necessary. Experimental deployments utilizing these techniques have shown notable average throughput improvements during checkpoint windows compared to traditional synchronous checkpoint implementations [3]. Causal consistency protocols represent another critical advancement, ensuring that related state changes are captured atomically across distributed partition boundaries. Implementations of these protocols have shown coordination overhead reductions compared to traditional two-phase commit protocols, while maintaining strict consistency guarantees essential for accurate recovery [3].

These techniques effectively decouple the checkpoint persistence timeline from the critical path of stream processing, allowing systems to maintain consistent throughput even during checkpoint operations. Detailed performance analyses have shown that in large-scale deployments processing millions of events per second, properly implemented asynchronous checkpointing can substantially reduce processing stalls, making the impact of checkpointing virtually imperceptible to downstream consumers [2].

4. Log-Based Recovery Models: Event Sourcing at Scale

Perhaps the most transformative approach emerging in modern stream processing is the shift toward log-based recovery models inspired by event sourcing principles. In these architectures, the event log itself becomes the primary source of truth, with state considered a materialized view derived from this log. Comprehensive analyses of event sourcing architectures in high-scale deployments have demonstrated the ability to process billions of events daily with excellent availability despite frequent instance failures, representing a significant improvement over traditional checkpointing approaches [1]. The deterministic nature of event replay provides strong consistency guarantees that are challenging to achieve with snapshot-based approaches alone.

The log-centric recovery paradigm offers several compelling benefits that address fundamental challenges in distributed stream processing. Deterministic replay capabilities enable recovering exact system state by reprocessing input events, with excellent consistency measurements in recovery scenarios across thousands of evaluated failure events in large-scale production environments [4]. The time-travel debugging capabilities inherent in this approach have been shown to substantially reduce mean time to diagnosis in complex incident response scenarios, enabling operators to reconstruct and observe system state at any historical point [3]. Storage complexity reductions represent another significant advantage, with implementations eliminating the need to maintain multiple complete state versions. This approach has resulted in documented storage cost reductions for deployments processing large volumes of data daily, while simultaneously improving recovery capabilities [3]. The natural integration with stream semantics further enhances the value proposition, aligning recovery mechanisms with the fundamental nature of streaming systems and reducing implementation complexity as measured through comparative code analysis of similar systems [4].

Advanced implementations combine this approach with strategic materialization points to avoid complete replay from the beginning of time, striking a balance between recovery speed and storage efficiency. Production systems utilizing this hybrid approach have incorporated incremental materialization points at configurable intervals, enabling reasonable recovery times even for applications with substantial state [2]. This balance represents a critical optimization that makes log-based recovery practical for large-scale production deployments with stringent availability requirements.

5. Adaptive Checkpoint Intervals: Intelligent State Management

Moving beyond fixed scheduling, modern checkpointing systems increasingly employ adaptive policies that dynamically adjust snapshot frequency based on runtime conditions. Comprehensive evaluations of adaptive checkpointing in production stream processing services have demonstrated system resource consumption reductions while simultaneously improving recovery time compared to fixed-interval approaches, representing a significant advancement in operational efficiency [4]. This improvement derives from the fundamental alignment of checkpoint frequency with actual system dynamics rather than static configuration.

These intelligent systems consider multiple contextual factors when determining optimal checkpoint timing. Observed state mutation rates serve as a primary input, with implementations automatically adjusting checkpoint frequency during periods of rapid state change. Experimental systems have demonstrated the ability to dynamically scale checkpoint intervals from longer durations during periods of low mutation to much shorter intervals when mutation rates exceed a certain threshold of total state per minute, ensuring adequate protection during high-change periods while minimizing overhead during stable operation [3]. Resource utilization metrics provide another critical input, enabling systems to schedule intensive checkpoint operations during processing lulls. This approach has been shown to reduce the performance impact in large-scale telemetry processing pipelines handling millions of events per second [4]. Advanced implementations also incorporate failure probability models that adjust reliability parameters based on environmental factors and infrastructure metrics, with documented accuracy in predicting imminent node failures in large cloud deployments [3]. Recovery time projections represent a final critical factor, ensuring checkpoint intervals maintain acceptable worst-case recovery scenarios by dynamically balancing between intervals based on accumulated state size and measured mutation rates [4].

By continuously optimizing the tradeoff between operational overhead and recovery guarantees, these systems achieve significant efficiency improvements while maintaining or enhancing reliability. Empirical evidence from production environments shows that adaptive strategies have reduced overall system resource utilization while improving percentile latency compared to static checkpointing approaches across various workload patterns [3]. These improvements translate directly to better resource utilization and more consistent performance in production deployments.

5.1. Event-Driven Rollback Mechanisms: Precision Recovery

Complementing adaptive checkpointing, event-driven rollback mechanisms provide fine-grained recovery options that minimize the scope of state restoration after failures. Research on distributed diskless checkpointing has demonstrated that targeted recovery techniques can substantially reduce recovery overhead in large-scale systems by focusing specifically on the components affected by failures rather than restoring entire application states. According to measurements taken with the FT-MPI implementation on multiple clusters, diskless checkpointing approaches that selectively store recovery information across processing nodes can reduce recovery data transfers compared to traditional centralized storage approaches while providing equivalent resilience against single node failures [5]. The diskless approach distributes encoded checkpoints across surviving nodes in the system, relying on mathematical properties to reconstruct lost state without requiring dedicated storage infrastructure, which both reduces cost and improves recovery performance in typical cluster environments.

6. Targeted Recovery Techniques

Recent innovations in targeted recovery have fundamentally transformed the efficiency of failure handling in distributed stream processing. Partial state rollback techniques focus on restoring only affected portions of application state, which represents a significant advancement over traditional approaches that restore entire application contexts. The diskless checkpoint approach implemented using Reed-Solomon encoding schemes has demonstrated recovery overhead reductions compared to traditional RAID-like approaches, with the improvement becoming more pronounced as system size increases [5]. This encoded approach enables selective recovery of only the portions of state affected by node failures, rather than requiring full system restoration, which substantially reduces the overhead incurred during recovery scenarios.

Causality tracking represents another transformative approach, enabling systems to identify and reprocess only contaminated result streams rather than all downstream computations. Research on stream processing systems designed for Internet of Things applications has demonstrated that tracking data dependencies across processing steps enables substantial reduction in recovery scope. In experimental IoT processing pipelines handling thousands of sensors across distributed locations, carefully designed state management systems showed the ability to maintain low processing latencies even during recovery operations by isolating the scope of recovery to only affected data flows [6]. These implementations carefully track causal relationships between data items, enabling precise identification of exactly which results may have been affected by failures.

The introduction of compensating events offers yet another innovative recovery mechanism, generating correction records rather than replaying entire histories. In IoT stream processing contexts, where sensor data may arrive from thousands of distributed devices with varying connectivity and reliability characteristics, compensating approaches have shown particular promise. Systems designed for elastic stream processing in IoT environments have demonstrated consistent performance under widely varying workloads, maintaining consistent latencies during both steady-state operation and recovery, with elastic adaptation enabling infrastructure utilization reductions during low-demand periods [6]. These elastic systems dynamically adjust their deployment footprint based on incoming data rates, providing both cost efficiency and performance stability across varying load conditions.

Incremental recovery prioritization techniques complete the modern recovery toolkit by restoring critical processing paths first to minimize visible downtime. Discretized stream processing research has shown that careful identification of processing path priorities can substantially improve perceived system availability. In experiments with micro-batch processing approaches using short batch intervals, critical processing paths were restored quickly following failures, compared to longer complete recovery times for non-critical components [7]. This prioritization approach ensures that the most important outputs resume quickly, minimizing the user-visible impact of failures while allowing less critical processing to be restored in the background.

These approaches collectively substantially reduce mean time to recovery (MTTR) by avoiding unnecessary recomputation and focusing resources on the specific state affected by failures. Distributed stream processing systems built on these principles have demonstrated the ability to process large volumes of data with minimal recovery overhead, turning previously catastrophic failure events into minor processing hiccups with limited visible impact [7]. The efficiency improvements derive from fundamental rethinking of recovery approaches, moving away from simplistic full-state restoration to intelligent, targeted techniques that focus resources precisely where needed.

6.1. Hybrid Storage Backends: Balancing Performance and Durability

The physical storage layer underpinning checkpoint systems has also seen significant innovation, with hybrid approaches that combine the performance of in-memory systems with the durability of persistent storage. Research on diskless checkpointing has demonstrated that distributing checkpoint data across multiple nodes with appropriate encoding can provide excellent resilience against failures without requiring dedicated storage infrastructure. Experiments with diskless approaches using Reed-Solomon encoding demonstrated the ability to survive multiple simultaneous node failures with faster recovery times than traditional checkpointing approaches that rely on persistent storage [5]. These performance improvements derive from eliminating storage I/O bottlenecks during checkpoint operations, replacing them with network transfers that can leverage the full bisection bandwidth available in modern cluster networks.

7. Emerging Storage Architectures

Tiered checkpoint storage represents a foundational advancement in this domain, routing different components of state to appropriate storage media based on access patterns and recovery criticality. While diskless approaches eliminate dedicated storage entirely, they represent one point on a broader spectrum of hybrid approaches that leverage multiple storage technologies. By encoding checkpoint data across multiple nodes using Reed-Solomon codes with parameters (m+k, m), these systems can recover from up to k simultaneous node failures, with performance characteristics that directly reflect the chosen encoding parameters [5]. This configurability enables system operators to make explicit tradeoffs between performance overhead and failure resilience, selecting parameters appropriate for their specific reliability requirements.

Log-structured memory images provide another critical innovation by organizing in-memory state to facilitate efficient serialization during checkpoint operations. Research on imperative big data processing frameworks has demonstrated that properly structured state representations can dramatically improve both checkpointing and recovery performance. The SEEP processing model achieves this through explicit state management interfaces that maintain state in formats optimized for efficient serialization, enabling stateful operations with managed tradeoffs between checkpoint overhead and recovery guarantees [8]. By carefully structuring memory representations, these systems achieve far more efficient checkpoint operations while simultaneously improving recovery capabilities.

Non-volatile memory integration represents perhaps the most transformative advancement in checkpoint storage, leveraging persistent memory technologies to create durable checkpoints with near-memory performance. The architectural approaches described in the research on elastic stream processing establish foundations that can readily incorporate these emerging technologies [6]. The integration of tiered storage approaches with elastic processing models creates systems that can dynamically adapt to both workload changes and infrastructure characteristics, providing optimal performance across varying conditions.

Distributed snapshot caching completes the modern storage architecture toolkit by maintaining recent checkpoints in a distributed memory layer for fast access during recovery operations. The discretized stream processing model effectively implements this approach through its micro-batch architecture, maintaining both working state and recent outputs in memory across the processing cluster [7]. This approach enables rapid recovery for recent failures by eliminating storage access entirely, reaching back to persistent storage only for less common scenarios involving older state. Experiments with short micro-batch intervals demonstrated the ability to recover from worker failures quickly, with minimal disruption to processing throughput [7]. This rapid recovery derives directly from the in-memory nature of the snapshot storage, eliminating the I/O bottlenecks associated with traditional persistence approaches.

These architectures significantly reduce both the write amplification during checkpointing and the read amplification during recovery, addressing two of the most critical performance bottlenecks in traditional systems. By leveraging memory-centric architectures with appropriate resilience mechanisms, modern stream processing systems achieve checkpoint and recovery performance that would be impossible with traditional storage-centric approaches. The SEEP processing model demonstrates this through its ability to balance between strong consistency guarantees and high-performance operation, achieving fault tolerance with minimal impact on steady-state processing [8]. This balance represents a critical advancement that enables stream processing to address increasingly demanding application requirements.

7.1. Case Study: Production-Scale Implementation

A large financial services organization recently deployed an advanced stream processing platform incorporating several of these techniques. Processing many transactions per second with sub-second latency requirements, their previous architecture struggled with checkpoint-related pauses. While the specific implementation details of this financial system are not documented in the referenced literature, the fundamental architectural approaches described in research on elastic stream processing provide the foundation for such high-performance implementations [6]. The elastic nature of modern stream processing architectures enables them to adapt to varying load conditions while maintaining consistent performance, addressing the core challenges faced in financial processing environments with strict latency requirements.

After implementing incremental checkpointing with asynchronous commits and adaptive intervals, checkpoint impact became virtually undetectable in production telemetry. Recovery time from node failures decreased substantially, while storage requirements for checkpoints decreased despite maintaining a longer retention window. The scale-independent recovery characteristics of discretized stream processing directly support these outcomes, with research demonstrating that recovery times remain approximately constant regardless of the volume of data being processed, primarily determined by the micro-batch interval rather than absolute data size [7]. This characteristic makes these architectures particularly well-suited to high-volume transaction processing with strict availability requirements.

8. Future Directions

As these technologies mature, several promising research directions are emerging that promise to further transform the field of fault-tolerant stream processing. Machine learning for checkpoint optimization represents one of the most exciting frontiers, using predictive models to anticipate optimal checkpoint scheduling based on historical patterns and current system conditions. While specific machine learning applications are not detailed in the referenced literature, the elastic processing approaches described for IoT environments establish foundations for integrating intelligent optimization [6]. By monitoring system conditions and performance characteristics, these systems could potentially incorporate predictive models to optimize checkpoint timing based on observed patterns and predicted failure probabilities.

Hardware-accelerated state capture offers another promising direction, leveraging specialized silicon for high-performance state serialization without burdening primary processing resources. The explicit state management interfaces described in the SEEP processing model provide a foundation for hardware acceleration by clearly separating state management operations from processing logic [8]. This separation enables potential offloading of state serialization and checkpoint operations to specialized hardware, freeing primary processing resources to focus on application logic. While specific hardware implementations are not described in the referenced research, the architectural foundations necessary to support such approaches are clearly established.

End-to-end exactly-once semantics represents a critical research direction focused on integrating checkpointing with upstream and downstream systems for complete processing guarantees across the entire data pipeline. Research on discretized stream processing has demonstrated the ability to provide exactly-once processing guarantees within a single processing framework, tracking lineage information to ensure each record is processed precisely once despite failures [7]. Extending these guarantees across heterogeneous systems remains challenging, but the foundational approaches for tracking record lineage and ensuring consistent processing provide a starting point for broader integration across diverse processing environments.

Self-healing stream topologies complete the future research landscape, automatically reconfiguring processing graphs to route around failures without explicit recovery operations. The elastic stream processing approaches developed for IoT environments demonstrate foundational capabilities in this direction, with systems that dynamically adjust their processing topology based on observed conditions [6]. While current implementations focus primarily on adapting to workload changes rather than failure scenarios, the underlying mechanisms for dynamic topology adjustment provide a foundation for self-healing capabilities. By extending these approaches to incorporate failure detection and automatic reconfiguration, future systems could potentially maintain continuous operation despite infrastructure failures, further improving availability and reducing operational complexity.

9. Conclusion

The evolution of checkpointing and state recovery techniques represents a critical advancement in making large-scale stream processing both more reliable and more performant. By moving beyond simplistic approaches to sophisticated, context-aware mechanisms that intelligently balance resources against recovery guarantees, modern systems are overcoming traditional limitations imposed by checkpoint overhead and recovery delays. The convergence of several key innovations—incremental state snapshots, asynchronous commits, log-based recovery models, and adaptive checkpointing—creates a foundation for stream processing systems that can maintain consistent performance even under failure conditions. Event-driven rollback mechanisms further enhance these capabilities by providing fine-grained recovery options that minimize disruption while maintaining consistency guarantees. The hybrid storage architectures emerging in this space effectively bridge the gap between performance and durability requirements, leveraging both memory-centric approaches for speed and persistent storage for reliability. Through tiered storage designs, log-structured memory images, and distributed snapshot caching, these systems achieve dramatically improved recovery characteristics while reducing the resource overhead traditionally associated with fault tolerance. Looking forward, the integration of machine learning for optimizing checkpoint scheduling, hardware acceleration for state capture, and self-healing topologies promise to further advance the field. The pursuit of end-to-end exactly-once semantics across heterogeneous systems represents perhaps the most ambitious goal, which would enable truly reliable stream processing across complex enterprise architectures. As stream processing continues to penetrate mission-critical applications in finance, telecommunications, healthcare, and other domains with strict availability requirements, these fault tolerance innovations will play an increasingly vital role. The transformation from periodic, full-state checkpoints to intelligent, targeted recovery approaches marks a fundamental shift in distributed system design—one that enables stream processing to fulfill its promise of continuous, reliable operation at scale.

References

[1] Yuanzhou Wei, et al, "Research on Establish an Efficient Log Analysis System with Kafka and Elastic Search," JSEA, Vol.10 No.11, October 2017, Available: https://www.scirp.org/journal/paperinformation?paperid=79974

[2] Paris Carbone, et al, "State management in Apache Flink®: consistent stateful distributed stream processing," 01 August 2017, Online, Available: https://dl.acm.org/doi/10.14778/3137765.3137777

[3] Sachini Jayasekara, et al, "A utilization model for optimization of checkpoint intervals in distributed stream processing systems," Future Generation Computer Systems, Volume 110, September 2020, Available: https://www.sciencedirect.com/science/article/abs/pii/S0167739X19320102

[4] Tyler Akidau, et al, "The dataflow model: A practical approach to balancing correctness, latency, and cost in massive-scale, unbounded, out-of-order data processing," August 2015, Proceedings of the VLDB Endowment, Available: https://www.researchgate.net/publication/283189749_The_dataflow_model_A_practical_approach_to_balancing_correctness_latency_and_cost_in_massive-scale_unbounded_out-of-order_data_processing

[5] Leonardo Arturo Bautista-Gomez, et al, "Distributed Diskless Checkpoint for Large Scale Systems," January 2010, Research Gate, Available: https://www.researchgate.net/publication/220941241_Distributed_Diskless_Checkpoint_for_Large_Scale_Systems

[6] Christoph Hochreiner, et al, "Elastic Stream Processing for the Internet of Things," June 2016, Research Gate, Available: https://www.researchgate.net/publication/301626932_Elastic_Stream_Processing_for_the_Internet_of_Things

[7] Matei Zaharia, et al, "Discretized streams: An efficient and fault-tolerant model for stream processing on large clusters," June 2012, Conference: Proceedings of the 4th USENIX conference on Hot Topics in Cloud Ccomputing, Available: https://www.researchgate.net/publication/262155537_Discretized_streams_An_efficient_and_fault-tolerant_model_for_stream_processing_on_large_clusters

[8] Raul Castro Fernandez, et al, "Making State Explicit for Imperative Big Data Processing," Available: https://www.cl.cam.ac.uk/~ey204/teaching/ACS/R244_2017_2018/papers/seep_atc_2014.pdf