Systems@ETH Zürich

Differential Durability: fault-tolerant distributed differential computation

Master Thesis Proposal

Operator state in streaming computations is very valuable and should be guarded against failure: the lack of fault-tolerance can result in incomplete or incorrect results after recovery. Additionally, streaming jobs run for long periods of time, accumulating state over several days or even months: reprocessing all input in the case of failures would be prohibitively expensive and time-consuming. The common approaches to fault-tolerance for streaming computations based on snapshots or active replication can have a significant negative impact on latency and overall performance. This Thesis will explore techniques that afford reduced impact on performance by moving the replication out of the critical path, thanks to the properties of the chosen computational model, *Differential Dataflow*[4].

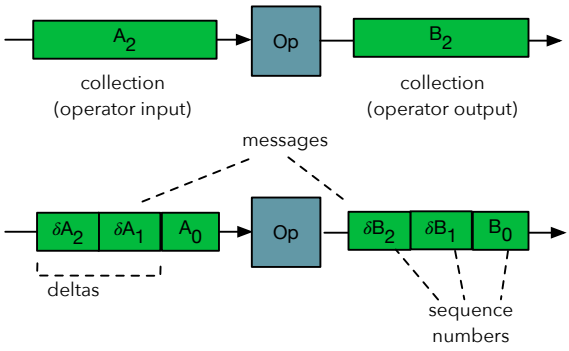

Figure 1: Differential Dataflow operator and input/output streams representing evolving collections.

Differential Dataflow is a data-parallel programming framework designed to efficiently process large volumes of data and to quickly respond to arbitrary changes in input collections. *Differential dataflow* programs are written as functional transformations of collections of data, using familiar operators like `map`, `filter`, `join`, and `group`.[1]

Collections *Differential Dataflow* represents operator inputs and outputs as collections of items that evolve with (logical) time (Fig. 1). Each collection in differential dataflow reflects an append-only log of immutable updates indexed by the logical timestamp. Updates are committed atomically every time computation for a certain (logical) time is completed (*sequence numbers* in Fig. 1). For this reason, each collection has known consistent states after each commit: these are natural rollback points in case of failure.

Thesis. This thesis aims to improve *Differential Dataflow*'s robustness against certain failure scenarios. The approach should leverage the intrinsic properties of the model to achieve asynchronous replication and fault-tolerance, with reduced overhead and recovery cost. This work will also be a key enabler for use-cases such as online re-scaling and time-travel.

Specifically, the goal is to (i) ensure timely dataflow collections can be reliably and efficiently stored or replicated, (ii) determine safe recovery points by inspecting the computation's progress and dataflow graph, and (iii) prototype a recovery mechansim for *Differential Dataflow*.

Related work Relevant for this work are RDDs[5], that share similarities with this approach, and Map Reduce[3] that's foundational for modern distributed resilient computation. [2] surveys fault-tolerance and high availability mechanisms in streaming systems, and MillWheel[1] is a data-streaming system that supports fault tolerance with durable queues and checkpoints.

[1] Tyler Akidau et al. "MillWheel: Fault-Tolerant Stream Processing at Internet Scale". In: *Very Large Data Bases*. 2013, pp. 734–746. URL: .

[2] Magdalena Balazinska, Jeong-Hyon Hwang, and Mehul A Shah. "Fault-tolerance and high availability in data stream management systems". In: (). URL: .

[3] Jeffrey Dean and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters". In: *Communications of the ACM* 51.1 (2008), pp. 107–113. URL: .

[4] Frank McSherry et al. "Differential dataflow". In: *Proceedings of CIDR 2013*. 2013. URL: .

[5] Matei Zaharia et al. "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing". In: *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*. USENIX Association. 2012, pp. 2–2. URL: .

If you are interested in this project please contact Andrea Lattuada (andreal@inf.ethz.ch). The proposed thesis will be supervised by Prof. Timothy Roscoe.

[1] From https://github.com/frankmcsherry/differential-dataflow