Naiad: A Timely Dataflow System

Derek G. Murray, Frank McSherry, Rebecca Isaacs, Michael Isard, Paul Barham, Martín Abadi

A Paper Review by:
Luca Choteborsky

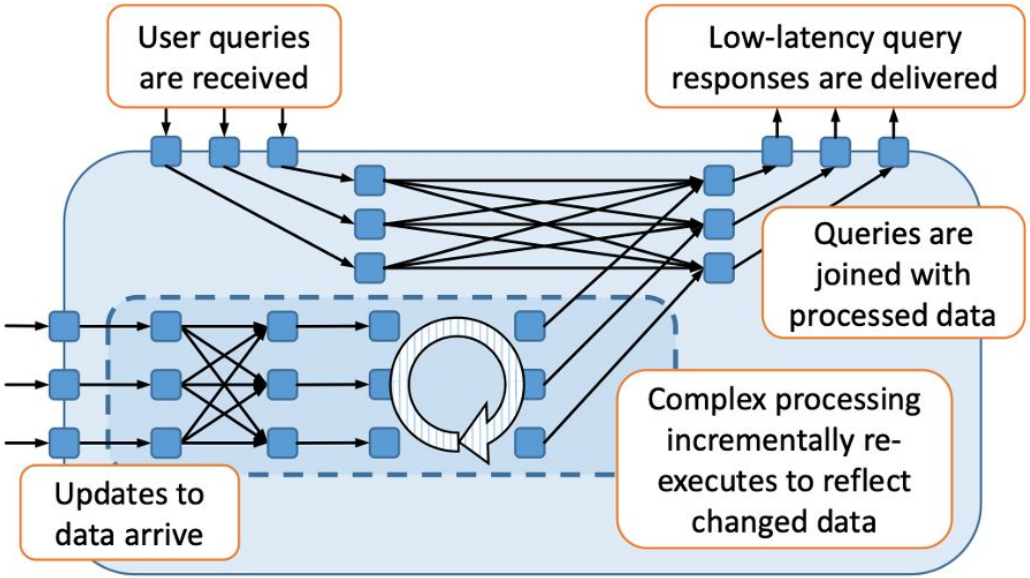
Background Context

A new type of application

- Iterative processing on real-time data stream
- Supports interactive queries on fresh results

Existing Systems

- Streaming, Batch and Trigger-based
- Other systems could not guarantee all properties



Background Context

Earlier paper introduced differential dataflow framework

- Changes to collections are described using a partial order
- This allows collections to change in multiple independent dimensions (e.g. loop indices or input vertices)

Implements differential dataflow in Naiad

- Extends the language of Naiad, the data-parallel runtime system
- Results in the amount of work performed being approximately proportional to the number of records that changed in the previous iteration

Background Context

Changes over the past years

- Large focus on ML training
- Movement to heterogeneous clusters

Problem Tackled

Wanted to implement a distributed system that achieved:

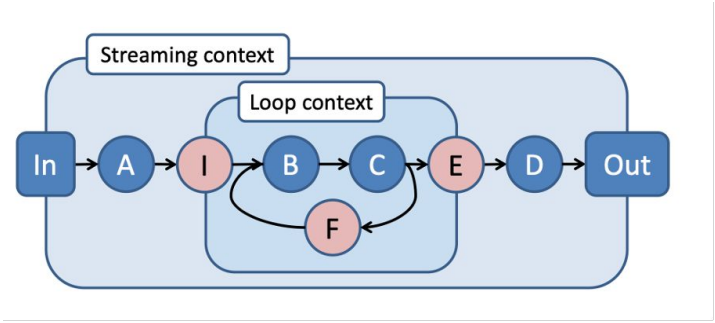
- High throughput
- Low latency
- Ability for iterative and incremental computation

No other system existed

Problem Tackled

Timely Dataflow

- A new computational model that modified dataflow
- Attaches timestamps to messages (vector timestamps for loop iteration and an epoch number)
- Specific vertices in a loop (Ingress, Egress and Feedback) modify the vector timestamps



Vertex	Input timestamp	Output timestamp
Ingress	$(e, \langle c_1, \ldots, c_k \rangle)$	$(e, \langle c_1, \ldots, c_k, 0 \rangle)$
Egress	$(e, \langle c_1, \ldots, c_k, c_{k+1} \rangle)$	$(e, \langle c_1, \ldots, c_k \rangle)$
Feedback	$(e, \langle c_1, \ldots, c_k \rangle)$	$(e, \langle c_1, \ldots, c_k + 1 \rangle)$

Problem Tackled

Timely Dataflow (continued)

- Vertices will receive a notification that they have received all messages bearing a specific timestamp
- Each vertex also implements two callback functions:

$$v.\textsc{OnRecv}(e : \text{Edge}, m : \text{Message}, t : \text{Timestamp})$$
$$v.\textsc{OnNotify}(t : \text{Timestamp}).$$

- And from their context they may call:

$$this.\textsc{SendBy}(e : \text{Edge}, m : \text{Message}, t : \text{Timestamp})$$
$$this.\textsc{NotifyAt}(t : \text{Timestamp}).$$

Problem Tackled

Timely Dataflow (continued)

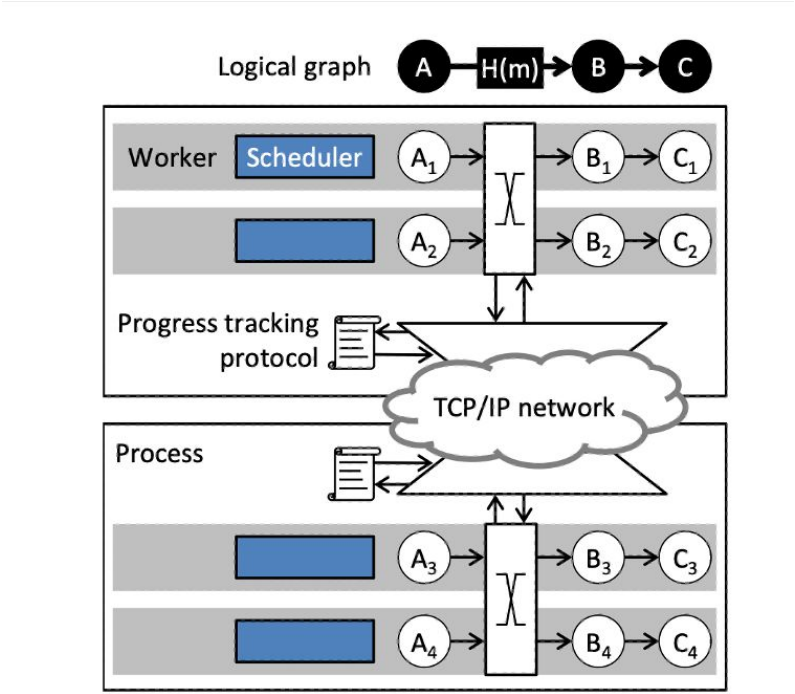
- To ensure correct delivery of notifications system must reason about impossibility of future messages bearing a given timestamp
- Each event has a timestamp and a location (referred to as a pointstamp)
- Uses structural constraints on timely dataflow graphs to induce an order on poinstamps know as *could-result-in*
- Presented a single-threaded scheduler to handle frontier of active pointstamps

Problem Tackled

Naiad: high-performance distributed implementation of timely dataflow

- Runs on a cluster that consists of a group of processes hosting workers that manage a partition of the timely dataflow vertices
- Uses specify logical timely dataflow graph of stages linked by typed connectors
- At run-time, logical graph is expanded to physical graph
- Uses specified partitioning functions to route messages
- Multiple workers coordinate (broadcasts) independent sets of events using a local view of the global state
- Tackled mitigating sources of micro-stragglers: networking, data structure contention and garbage collection

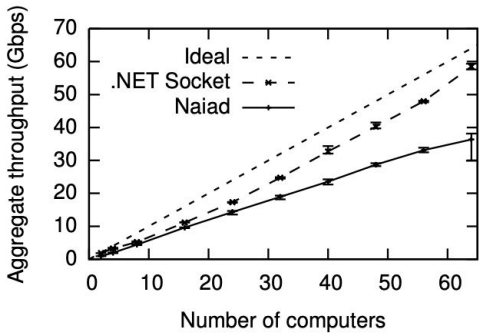
Problem Tackled



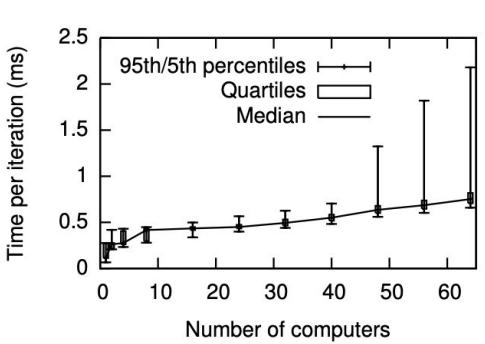
Problem Tackled

Evaluation

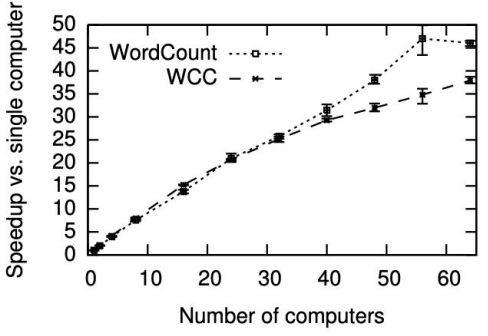
- Used microbenchmarks to evaluate baseline Naiad system performance



(a) All-to-all exchange throughput (§5.1)



(b) Global barrier latency (§5.2)

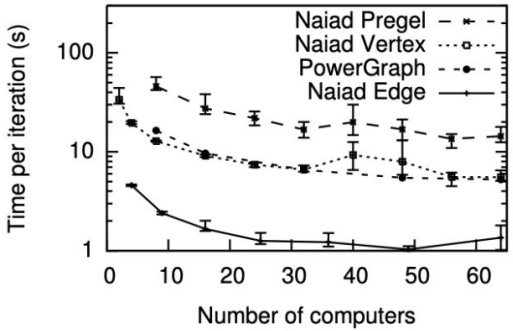


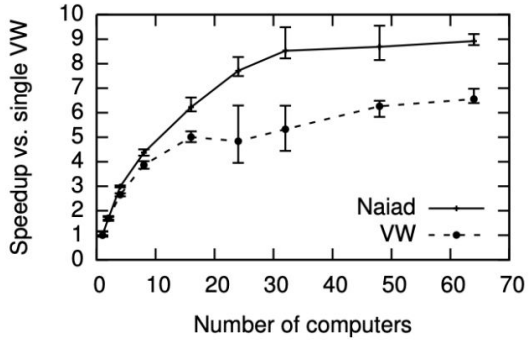
(d) Strong scaling (§5.4)

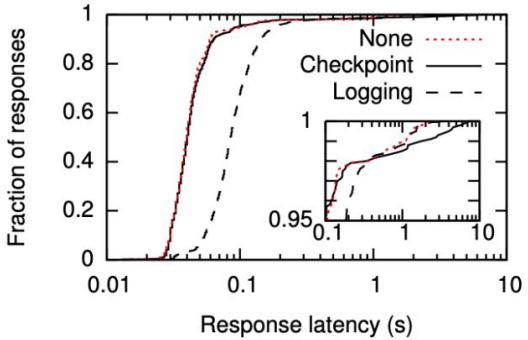
Problem Tackled

Evaluation

- Compared performance against existing systems when applied to applications drawn from the literature on batch, streaming and graph computation







Opinion of Paper

I mainly agree with the paper

- Agree with the need of a system to deliver high throughput, low latency and the ability for iterative and incremental computations
- I agree that the timely dataflow model has its merits
- I partially agree with the comment that combining existing systems could not yield the same performance, however proper evidence would have been nice

Opinion of Paper

Strengths

- Timely Dataflow is based on an already established computational model and gives crucial properties for parallel iterative computations
- Naiad can be applied to already existing data processing applications with exceptional performance
- Only Naiad can serve as the platform for sophisticated applications

Opinion of Paper

Weakness

- Lack of evaluation on different cluster architectures
- Does not mention how work is scheduled across heterogeneous architectures
- Uses a global checkpointing for fault tolerance; does not allow faulted processes to rejoin
- Vulnerable to micro-stragglers; only attempts to prevent them from occurring but does not handle recovery
- Potential memory inefficiencies: “Shared Arrangements” paper shares in-memory state between concurrent queries to address this problem

Opinion of Paper

Key Takeaway

- Timely dataflow is a promising computational model for building distributed systems
- Implementation of Naiad can be applied to already existing processing applications, competing with specialised systems
- Naiad’s programming language is highly expressive

Key Impact

- Well-established distributed data processing system; commonly used in benchmarks
- TensorFlow borrow Naiad’s vertices for branching and looping
- Additional frameworks have been added to Naiad (e.g. GraphLINQ)