CS 744: NAIAD

Shivaram Venkataraman
Fall 2019
- Course Project Proposal feedback
- Midterm grades
- Checkins?
STREAMING + ITERATIVE COMPUTATION

User queries are received

Low-latency query responses are delivered

Queries are joined with processed data

Complex processing incrementally re-executes to reflect changed data

Updates to data arrive

Dashboard

SQL

Results from streaming

Unification
TIMELY DATAFLOW

[Diagram with nodes labeled A, B, C, D, E, F, and edges indicating flow and context]

- Timestamp
- A new message
- "track"
- Loop context
- Streaming context
- "notification"
- DASS
- Stateful
- TensorFlow
TIMELY DATAFLOW

Timestamp: \( (e \in \mathbb{N}, \langle c_1, \ldots, c_k \rangle \in \mathbb{N}^k) \)

Vertex
- Ingress
- Egress
- Feedback

Input timestamp
- \( (e, \langle c_1, \ldots, c_k \rangle) \)
- \( (e, \langle c_1, \ldots, c_k, c_{k+1} \rangle) \)
- \( (e, \langle c_1, \ldots, c_k \rangle) \)

Output timestamp
- \( (e, \langle c_1, \ldots, c_k, 0 \rangle) \)
- \( (e, \langle c_1, \ldots, c_k \rangle) \rightarrow \text{Inc loop counter} \)
- \( (e, \langle c_1, \ldots, c_k, c_{k+1} \rangle) \rightarrow \text{Inc loop counter} \)
**VERTEX API**

**Receiving Messages**
- `v.OnRecv(e : Edge, m : Msg, t : Time)`
- `v.OnNotify(t : Timestamp)`

**Sending Messages**
- `this.SendBy(e : Edge, m : Msg, t : Time)`
- `this.NotifyAt(t : Timestamp)`

Not specifying any vertex
IMPLEMENTING TIMELY DATAFLOW

Need to track when it is safe to notify

Path Summary
Check if \((t_1, l_1)\) could-result-in \((t_2, l_2)\)

Scheduler
Occurrence and Precursor count
Precursor count = 0 \(\rightarrow\) Frontier

\(\rightarrow\) know that all messages with \(t' \leq t\) have been delivered before Object

\(v_1\) \(\text{send-by} (t_1, e)\)
\(\text{oc}[t_1, e] + = 1\)
\(\text{recv} (t_1, e)\)
\(\text{oc}[t_1, e] - = 1\)
Workers communicate using Shared Queue

Batch messages delivered Account for cycles

Vertex single threaded
DISTRIBUTED PROGRESS TRACKING

Broadcast-based approach

Maintain local precursor count, occurrence count
Send progress update \((p \in \text{Pointstamp}, \delta \in \mathbb{Z})\)

Local frontier tracks global frontier

Optimizations

Batch updates and broadcast

Use projected timestamps from logical graph
FAULT TOLERANCE

Checkpoint
- Log data as computation goes on
- Write a full checkpoint on demand
- Pause worker threads
- Flush message queues OnRecv

Restore
- Reset all workers to checkpoint
- Reconstruct state
- Resume execution

Vertices Stateful

Recovery time could be large
Failures are rare?

Centralized point of contention
Simplifies
Micro Stragglers

What is different from stragglers in MapReduce?

Stateful

Sources of stragglers
Network
Concurrency
Garbage Collection

Systems Tricks
Preventive
Differential Dataflow

// 1a. Define input stages for the dataflow.
var input = controller.NewInput<string>();

// 1b. Define the timely dataflow graph.
// Here, we use LINQ to implement MapReduce.
var result = input.SelectMany(y => map(y))
    .GroupBy(y => key(y),
        (k, vs) => reduce(k, vs));

// 1c. Define output callbacks for each epoch
result.Subscribe(result => { ... });

// 2. Supply input data to the query.
input.onNext(/* 1st epoch data */);
input.onNextCompleted();
Stream processing → Increasingly important workload trend

Timely dataflow: Principled approach to model batch, streaming together

Vertex message model
- Compute frontier
- Distributed progress tracking
DISCUSSION

https://forms.gle/v3YsW1HvnqsxCuPu5
What are some example scenarios discussed in the dataflow paper that are NOT a good fit for implementation using Naiad?

Stale updates → watermark

Triggering →

Good but not perfect is fine!
Consider you are implementing a micro-batch streaming API on top of Apache Spark. What are some of the bottlenecks/challenges you might have in building such a system?