CS 744: NAIAAD

Shivaram Venkataraman
Fall 2020
- Course Project Proposal feedback today!
- Midterm grading in progress
- Assignment regrades?
  - Shivaram OH poll → Piazza
DASHBOARDS

Sales Dashboard

Total Sales: $3,256.8M
Number of Deals: 17,164
Avg Deal Size: $189,545
Rev. per Salesperson: $20.5M

Revenue Over Time

- 2013: 246.6M
- 2014: 555.2M
- 2015: 846.7M
- 2016: 1,480.0M

Sales Team Performance

Sales Team
- Enterprise
- Group

Salesperson
- Susan Olson: 48.5M
- Steve Watkins: 46.1M
- Raymond Haw: 44.8M
- Robert Hudson: 43.8M
- Preston Ross: 42.8M
- Sophia Willis: 42.1M
- Dan Rivera: 40.6M
- Sarah Stephens: 34.5M
- Ross Spencer: 33.9M
- Ellie Price: 33.8M

Average Deal Size

Grouped by:

Sales Team
- Small and Midmarket
- Enterprise

Week of September 4, 2016

Revenue: $14.6M
Running Sum of Revenue: $798.4M
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

Tweets streaming in to the system

What is top hashtag for a user? (recommendation)

Social graph

Incremental processing!

Output from connected components

Page rank

ML algorithms
TIMELY DATAFLOW

Logical stages of computation

Ingress

each node can have >1 inputs or outputs

Egress

Input data source

Client data comes in here

for i in 1:100

for j in 1:10:

[nested loop]

Output

Feedback vertex in every loop!

Not acrylic! Can have loops

Nested loops need to be

vertices are stateful!!

Input - A - I - B - C - E - D - Out

Stream context

Loop context

Egress
TIMELY DATAFLOW

input grows in epochs → input events already have an epoch → notify when epoch is done

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

**Example**: $(0, \langle 0, 0, 17 \rangle, \langle 0, 0, 27 \rangle, \langle 0, 1, 0 \rangle)$

<table>
<thead>
<tr>
<th>Vertex</th>
<th>Input timestamp</th>
<th>Output timestamp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ingress</td>
<td>$(e, \langle c_1, \ldots, c_k \rangle)$</td>
<td>$(e, \langle c_1, \ldots, c_k, 0 \rangle)$</td>
</tr>
<tr>
<td>Egress</td>
<td>$(e, \langle c_1, \ldots, c_k, c_{k+1} \rangle)$</td>
<td>$(e, \langle c_1, \ldots, c_{k+1} \rangle)$</td>
</tr>
<tr>
<td>Feedback</td>
<td>$(e, \langle c_1, \ldots, c_k \rangle)$</td>
<td>$(e, \langle c_1, \ldots, c_{k+1} \rangle)$</td>
</tr>
</tbody>
</table>

For the new loop context, done with $k+1$ loop:
- Increment last loop counter.
- For this new loop context:
  - Done with $k+1$ loop.
Receiving Messages

v.OnRecv(e : Edge, m : Msg, t : Time)

v.OnNotify(t : Timestamp)

called when all messages with
ks <= t have been delivered

Sending Messages

this.SendBy(e : Edge, m : Msg, t : Time)

this.NotifyAt(t : Timestamp)

Useful to finalize the values for
an epoch

Actor Model = Similar

class Vertex {

    running_sum = 0

    onRecv(e : Edge, m : Msg, t : Time) {
        // got msg m from edge E
        // at time T
        running_sum += (int)m;
        sendBy(e, running_sum, t)
    }
}

invokes onRecv on V2 with
msg = running_sum, t = t'

t' > = t

this vertex will not produce
any output for t after this.
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 → Frontier

when occurrence = 0
update precursor for downstream vertices

sendBy → increments occurrence count
recv → decrements

All precursors must be done with timestamp \(T\) before onNotify can be called.
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

\[ \text{when is it safe to notify} \]
\[ \text{send updates to all workers before updating locally} \]
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

Actor in Ray / Pytorch
Actors in Ray / Pytorch

How frequently checkpoint

Overhead vs.

Time to recovery

Inputs can be replayed!!
What is different from stragglers in MapReduce?

- Stateful vertices ← IMPORTANT
- Speculative/retry is difficult

Sources of stragglers
- Network
- Concurrency
- Garbage Collection

Preventive measures to avoid stragglers rather than mitigate them.
// 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.onCompleted();
SUMMARY

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/qn8AzxHMT9VtyEc37
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?
Next class: Spark Streaming
Course project peer feedback due tonight!

Timely dataflow vs. Dataflow Model
\(<\text{epoch, } \text{<counters>}>\) \rightarrow \text{<processing>}

Limitations of Naiad
→ Epoch boundaries need to be defined.