CS 744: NAIAD

Shivaram Venkataraman
Fall 2020
- Course Project Proposal feedback today!
- Midterm grading in progress
- Assignment regrades?
Scalable Storage Systems

Datacenter Architecture

Resource Management

Computational Engines

Applications

Machine Learning

SQL

Streaming

Graph

Applications

Machine Learning

SQL

Streaming

Graph

Computational Engines

Scalable Storage Systems

Resource Management

Datacenter Architecture
Sales Dashboard

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

Week of Date Closed
December 6, 2020 - December 25, 2020

Voltage

Revenue Over Time

<table>
<thead>
<tr>
<th>Year</th>
<th>2013</th>
<th>2014</th>
<th>2015</th>
<th>2016</th>
</tr>
</thead>
<tbody>
<tr>
<td>Revenue</td>
<td>246.6M</td>
<td>555.2M</td>
<td>846.7M</td>
<td>931.1M</td>
</tr>
<tr>
<td>Running Total</td>
<td>0.0M</td>
<td>246.6M</td>
<td>555.2M</td>
<td>846.7M</td>
</tr>
</tbody>
</table>

Sales Team Performance

Sales Team | Salesperson | Q1 | Q2 | Running Sum of Revenue
---|---|---|---|---
Enterprise | Susan Olson | 0.5M | 0.4M | $147,043
| Steve Watkins | 0.4M | 0.4M |
| Raymond Hawk | 0.4M | 0.4M |
| Robert Hudson | 0.3M | 0.3M |
| Preston Ross | 0.3M | 0.3M |
| Sophia Willis | 0.3M | 0.3M |
| Dan Rivera | 0.3M | 0.3M |
| Sarah Stephens | 0.3M | 0.3M |
| Ross Spencer | 0.3M | 0.3M |
| Ellie Price | 0.3M | 0.3M |

Revenue by Quarter

- Q1: $14.6M
- Q2: $79.8M
STREAMING + ITERATIVE COMPUTATION

User queries are received

Low-latency query responses are delivered

Updates to data arrive

Queries are joined with processed data

Complex processing incrementally re-executes to reflect changed data
TIMELY DATAFLOW

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

<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></td>
</tr>
<tr>
<td>Feedback</td>
<td>((e, \langle c_1, \ldots, c_k \rangle))</td>
<td></td>
</tr>
</tbody>
</table>
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)
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
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
MICRO STRAGGLERS

What is different from stragglers in MapReduce?

Sources of stragglers
Network
Concurrency
Garbage Collection
// 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/qn8AzcHMT9VtyEc37
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!