Good morning!

CS 744: NAIAD

Shivaram Venkataraman
Fall 2021
In Progress
- Course Project Proposal feedback
- Midterm grading
Scalable Storage Systems

Datacenter Architecture

Computational Engines

Resource Management

Applications

Machine Learning

SQL

Streaming

Graph

✓ Workload API

Dataflow model
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

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

Running Total:
- 2013: 0M
- 2014: 246.6M
- 2015: 555.2M
- 2016: 1,480.0M

Sales Team Performance

Sales Team: Enterprise
Salesperson: Susan Olson, Steve Watkins, Raymond Hawk, Robert Hudson, Preston Ross, Sophia Willis, Dan Rivera, Sarah Stephens, Ross Spencer, Ellie Price

Revenue by Quarter

Week of September 4, 2016
Revenue: $14.6M
Running Sum of Revenue: $798.4M

Dashboards are continuously updated as data arrives. The input of events, latency to update this.
Streaming + Iterative Computation

Top hashtag over last N minutes

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

Tweets arriving

Updates to data arrive

Complex processing incrementally re-executes to reflect changed data

Connected components (finding topics that are connected)
TIMELY DATAFLOW

Computation run in iteration of loop

Vertices

Ingress

Messages between vertices

Feedback

Loop exit vertex

Loop entry vertex

Egress / Output vertex
TIMELY DATAFLOW

Epoch input events have an epoch which is given

Epoch numbers with timestamp?

→ flexibility / Out or order

Epoch

→ notifications / output when computation for epoch finishes

→ flexibility

loop counters

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

for \( i = 0 : 10 \)

for \( j = 0 : 10 \)


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

**Actor System**

- **Receiving Messages**
  - `v.OnRecv(e : Edge, m : Msg, t : Time)`
  - `v.OnNotify(t : Timestamp)`
    - Called when all msgs \( \leq t \) have been delivered.

**Sending Messages**

- `this.SendBy(e : Edge, m : Msg, t : Time)`
- `this.NotifyAt(t : Timestamp)`
  - Send a msg \( m \) on this edge \( e \), with \( t \) as timestamp.
class DistinctCount<S,T> : Vertex<T> {
    Dictionary<T, Dictionary<S,int>> counts;
    void OnRecv(Edge e, S msg, T time) {
        if (!counts.ContainsKey(time)) {
            counts[time] = new Dictionary<S,int>();
            this.NotifyAt(time);
        }
        if (!counts[time].ContainsKey(msg)) {
            counts[time][msg] = 0;
            this.SendBy(output1, msg, time);
        }
        counts[time][msg]++;
    }
    void OnNotify(T time) {
        foreach (var pair in counts[time])
            this.SendBy(output2, pair, time);
        counts.Remove(time);
    }
}
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
Workers communicate using Shared Queue

Batch messages delivered

Account for cycles

Vertex single threaded

Identical workers
(no centralized Master)
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
  - redo work from the checkpoint

- It is safe to checkpoint / consistent when failure is infrequent, get better perf when no failures!
What is different from stragglers in MapReduce?

Sources of stragglers
- Network
- Concurrency
- Garbage Collection

\[ \overset{\text{minimize prob of straggler!}}{\Rightarrow \text{duplicate or back up}} \]
// 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();
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/cYYy14JfX71qBzY66
Input:
32,000 tweets/second, Queries every 100ms

1. $t_{\text{Fresh}}$ suffers from bigger slowdown than $t_{1s}$
2. Stale data already been processed
3. Query is waiting for incremented connected components

materialize: 10
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!