CS 744: NAIAD

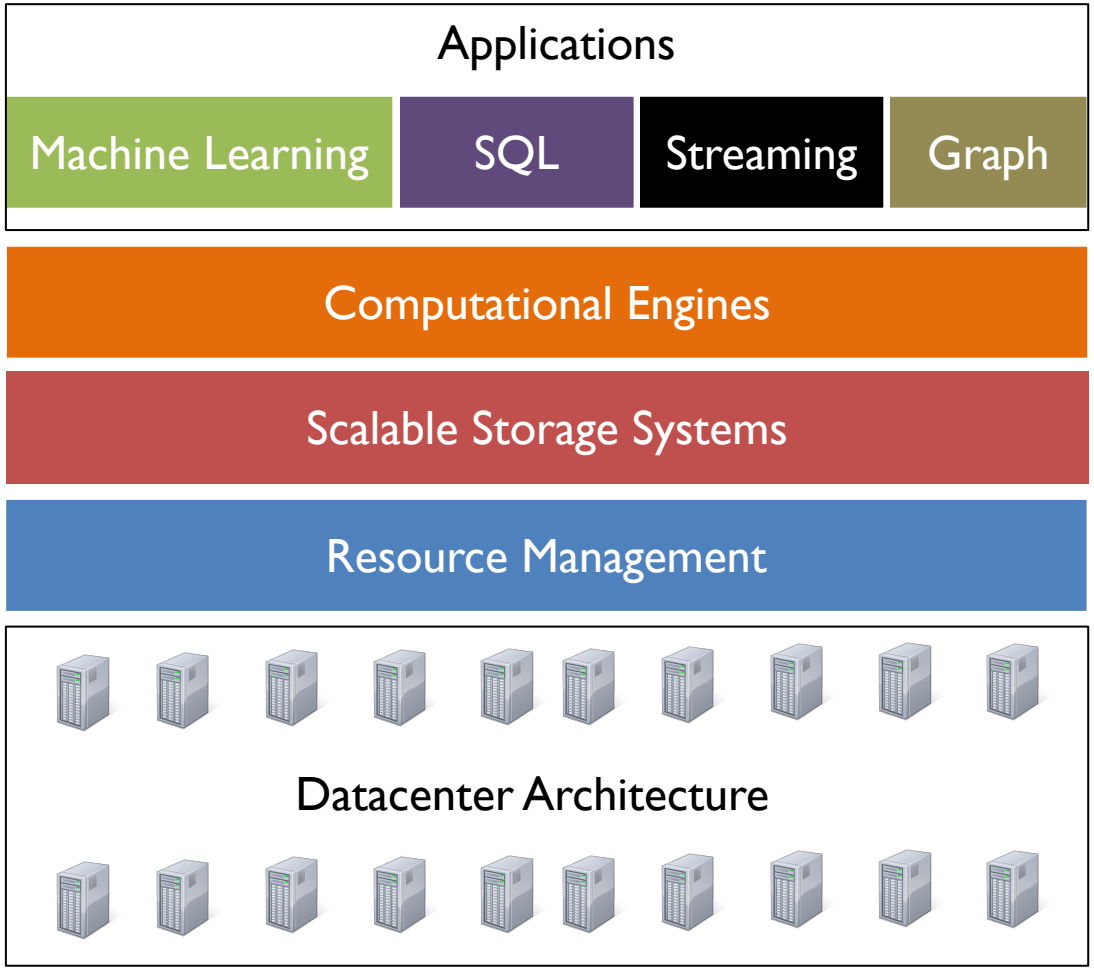
Shivaram Venkataraman

Fall 2021

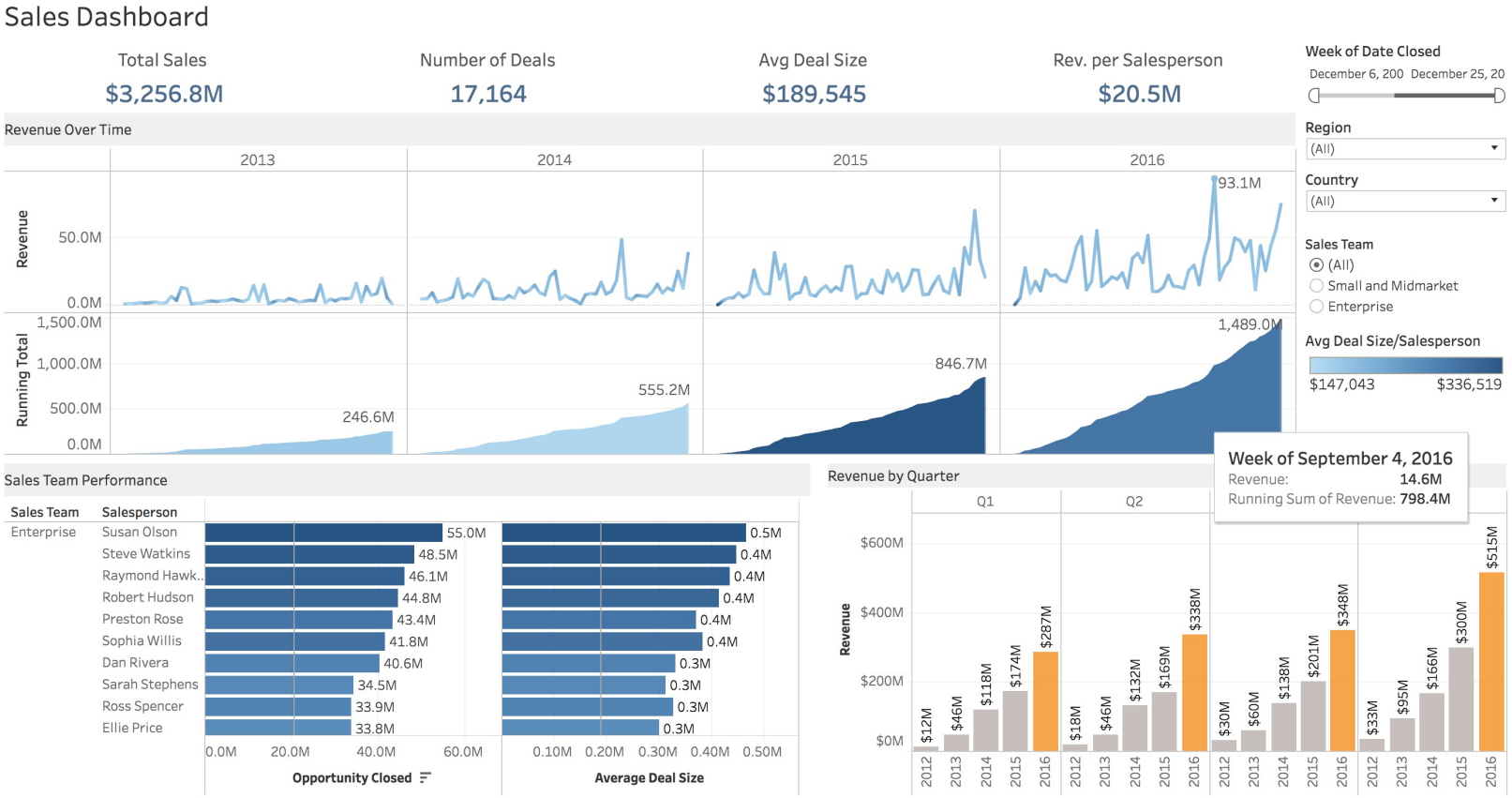
ADMINISTRIVIA

In Progress

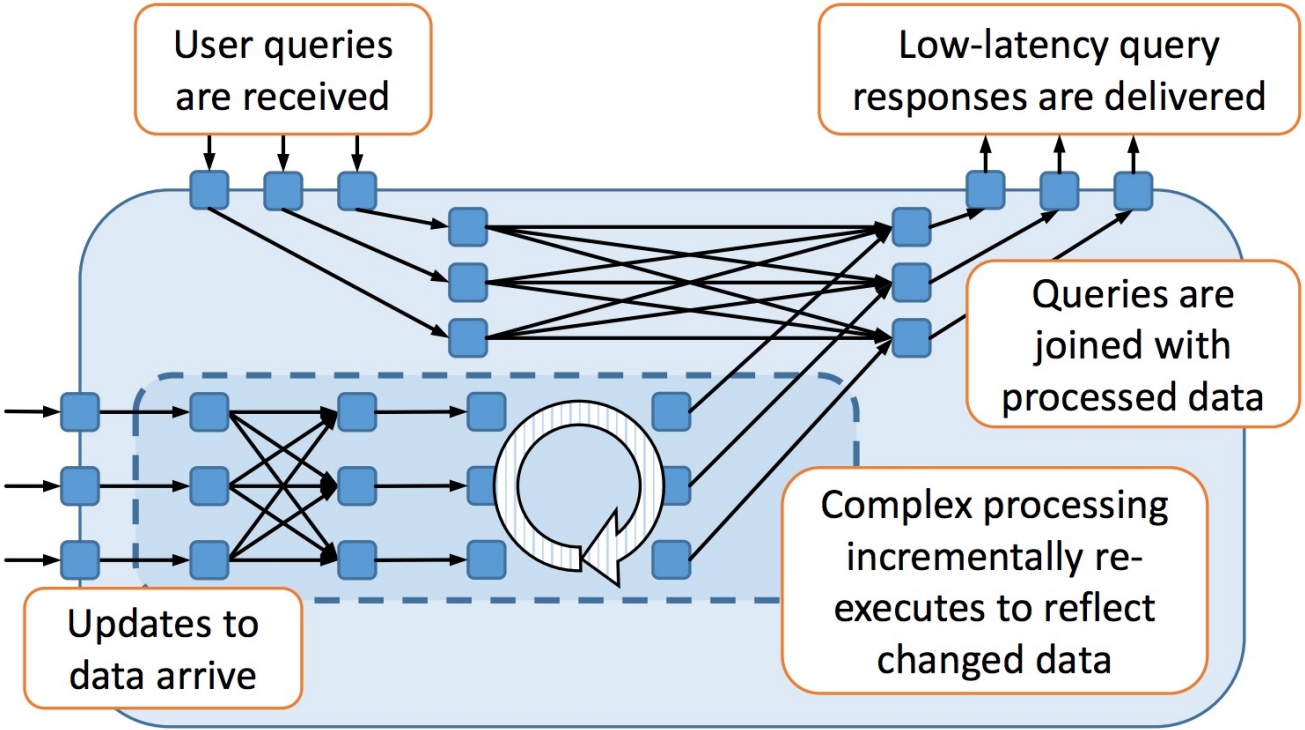
- Course Project Proposal feedback
- Midterm grading


Applications
Machine Learning
SQL
Streaming
Graph
Computational Engines
Scalable Storage Systems
Resource Management
Datacenter Architecture

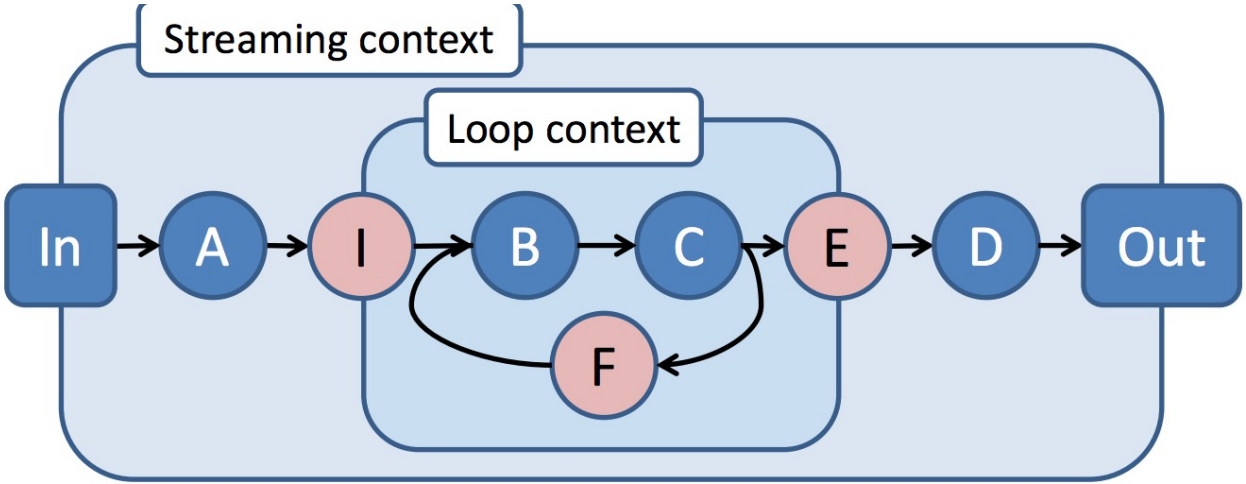
DASHBOARDS


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, 200 December 25, 20
Region
(All)
Country
(All)
Sales Team
(All)
Small and Midmarket
Enterprise
Avg Deal Size/Salesperson
$147,043
$336,519
Revenue Over Time
2013
2014
2015
2016
Revenue
50.0M
0.0M
93.1M
Running Total
1,500.0M
1,000.0M
500.0M
0.0M
246.6M
555.2M
846.7M
1,489.0M
Week of September 4, 2016
Revenue: 14.6M
Running Sum of Revenue: 798.4M
Sales Team Performance
Sales Team
Salesperson
Enterprise
Susan Olson 55.0M 0.5M
Steve Watkins 48.5M 0.4M
Raymond Hawk.. 46.1M 0.4M
Robert Hudson 44.8M 0.4M
Preston Rose 43.4M 0.4M
Sophia Willis 41.8M 0.4M
Dan Rivera 40.6M 0.3M
Sarah Stephens 34.5M 0.3M
Ross Spencer 33.9M 0.3M
Ellie Price 33.8M 0.3M
0.0M 20.0M 40.0M 60.0M
Opportunity Closed
0.10M 0.20M 0.30M 0.40M 0.50M
Average Deal Size
Revenue by Quarter
Q1
Q2
Revenue
$600M
$400M
$200M
$0M
$12M $46M $118M $174M $287M
$18M $46M $132M $169M $338M
$30M $60M $138M $201M $348M
$33M $95M $166M $300M $515M
2012 2013 2014 2015 2016

STREAMING + ITERATIVE COMPUTATION

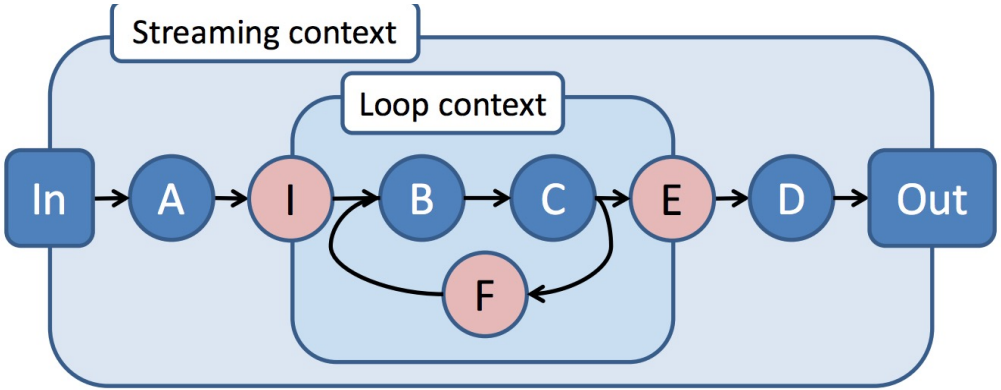


TIMELY DATAFLOW



TIMELY DATAFLOW

$$\text{Timestamp} : (\overbrace{e \in \mathbb{N}}^{\text{epoch}}, \overbrace{\langle c_1, \ldots, c_k \rangle \in \mathbb{N}^k}^{\text{loop counters}})$$



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

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)
```

EXAMPLE

```
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

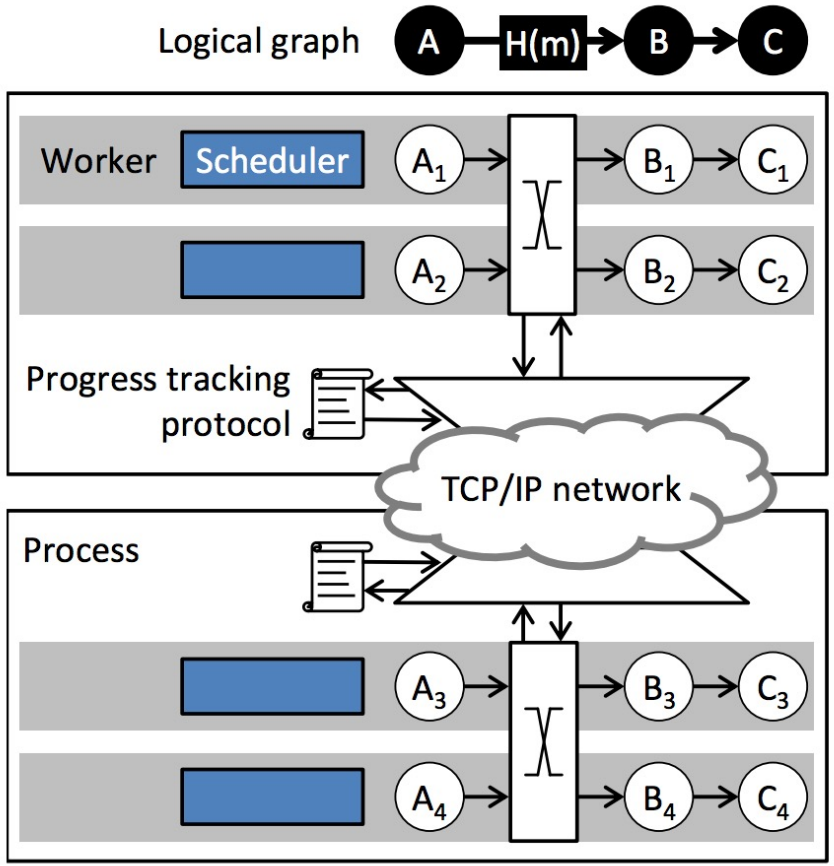
Precursor count = 0 → Frontier

ARCHITECHTURE

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 ($p \in$ Pointstamp, $\delta \in Z$)
- 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

HIGH LEVEL API

```
// 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

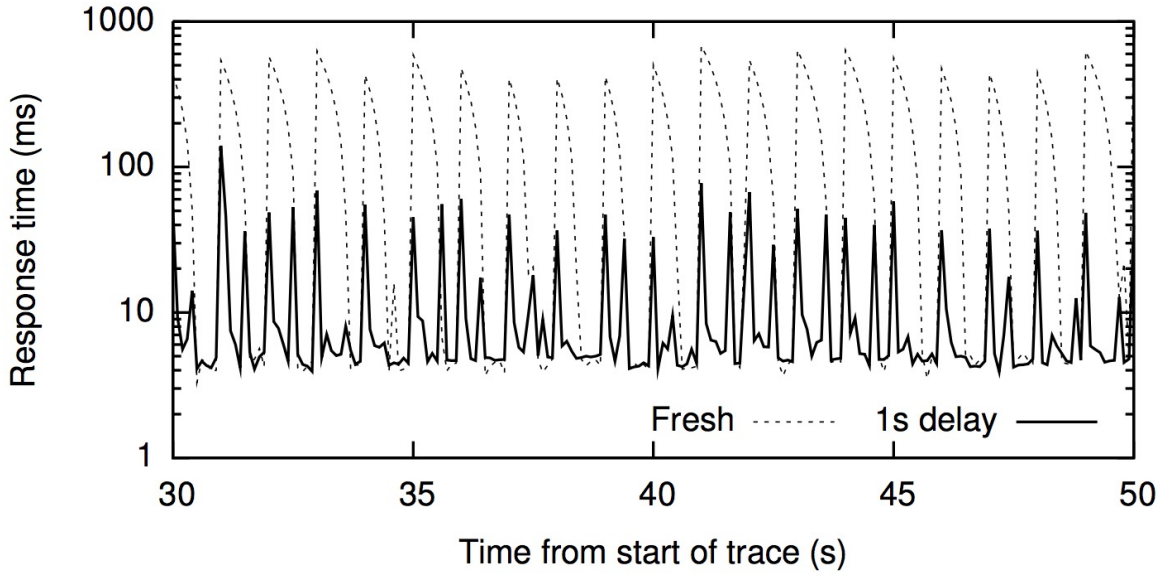
Timely dataflow: Principled approach to model batch, streaming together

Vertex message model

- Compute frontier
- Distributed progress tracking

DISCUSSION

https://forms.gle/cYYy14JfX71qBzY66


1000
100
10
1
Response time (ms)
30
35
40
45
50
Time from start of trace (s)
Fresh
1s delay

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?

SUMMARY

Next class: Spark Streaming

Course project peer feedback due tonight!