Welcome!

CS 744: NAIAD

Shivaram Venkataraman

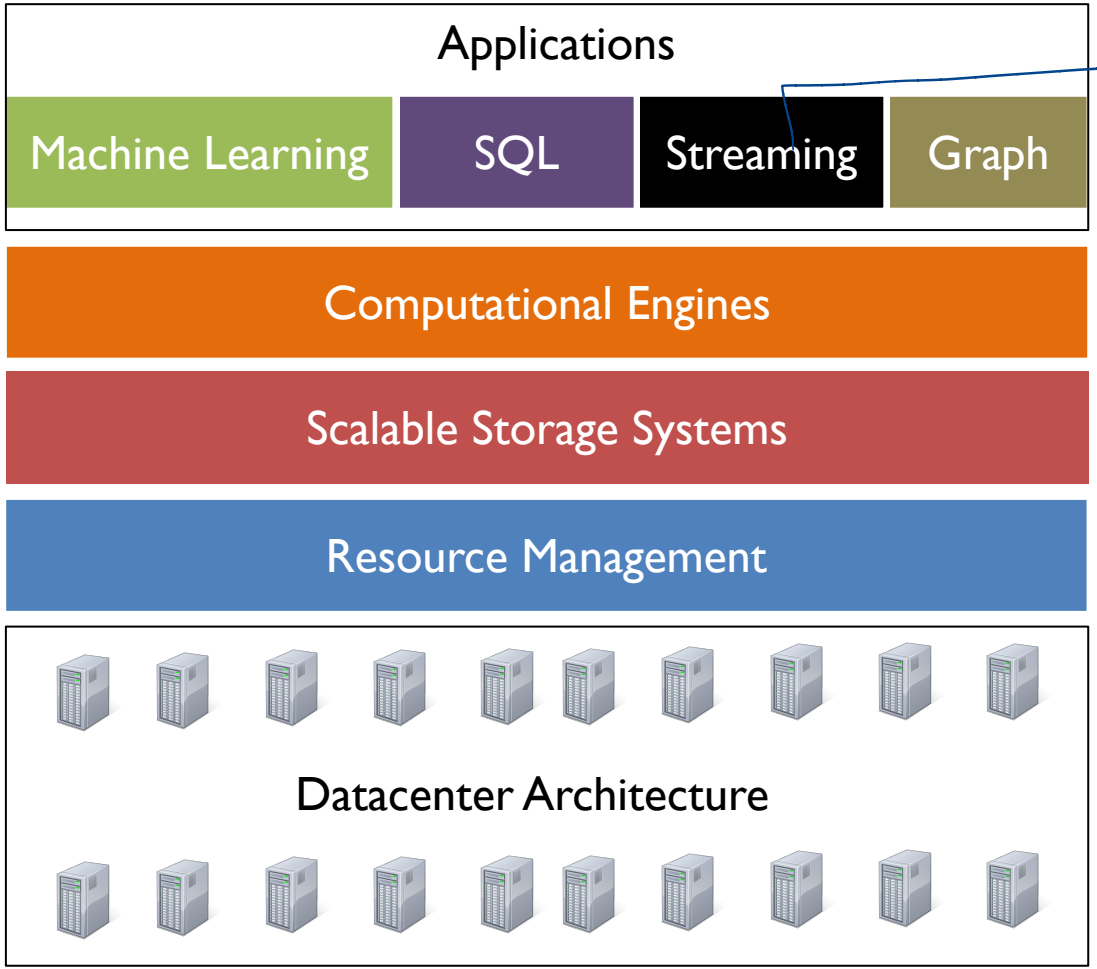
Fall 2020

ADMINISTRIVIA

- Course Project Proposal feedback today!
- Midterm grading in progress
- Assignment regrades?

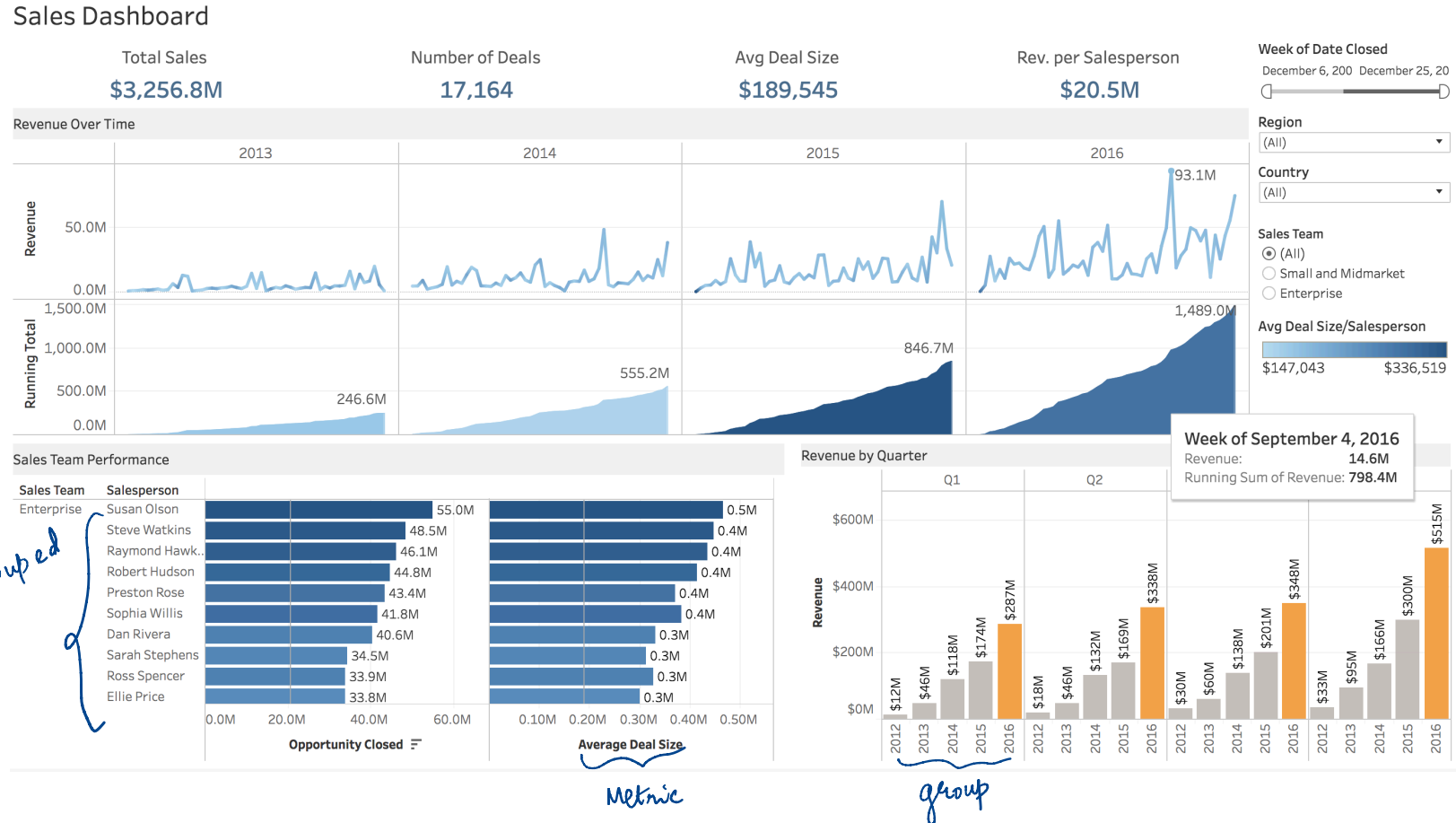
↳ HotCRP system

- Shivaram OH poll → Piazza



Unbounded data
↓
Dataflow model
↙ (Streaming)
↘ Batch

DASHBOARDS

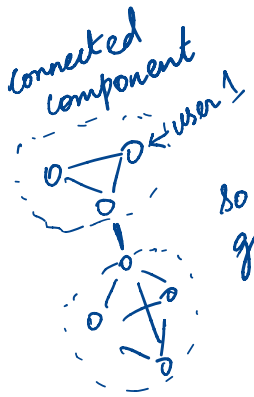


grouped

Metric

group

STREAMING + ITERATIVE COMPUTATION



→ Page Rank
→ ML algorithms

Incremental processing !

User queries are received

What is top hashtag for a user? (recommendation)

Low-latency query responses are delivered

Tweets streaming in to the system

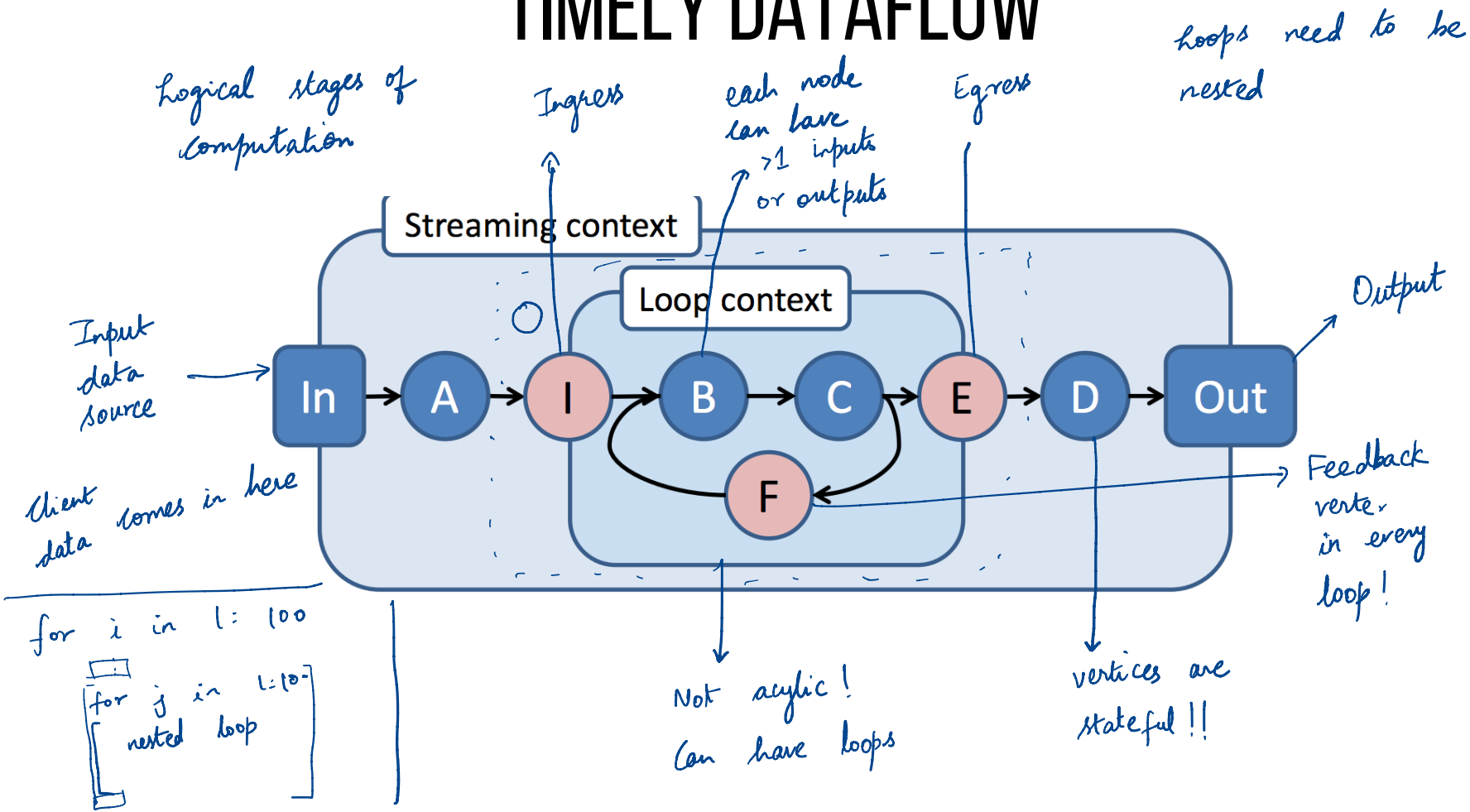
Queries are joined with processed data → output from connected components

Updates to data arrive

Complex processing incrementally re-executes to reflect changed data

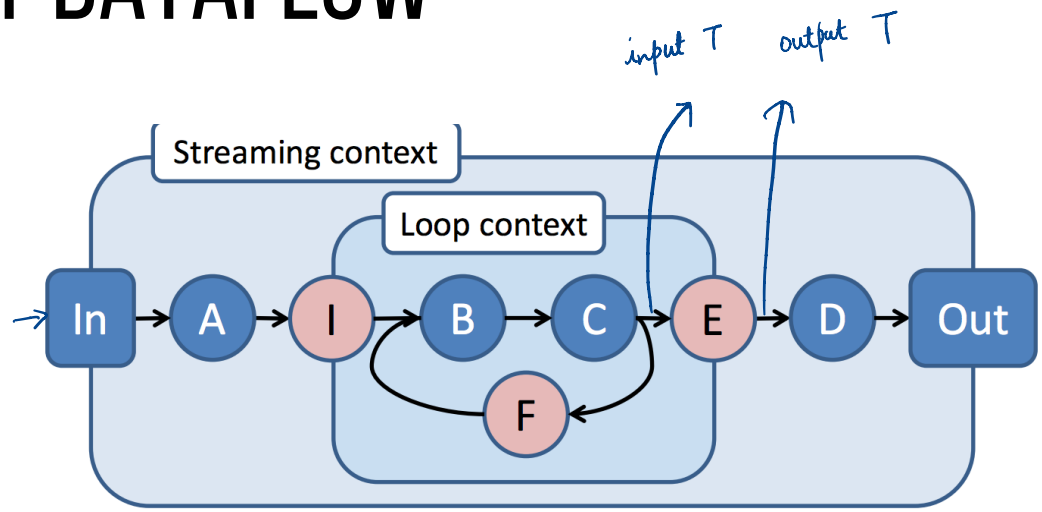
Connected components in the social group

TIMELY DATAFLOW



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

TIMELY DATAFLOW

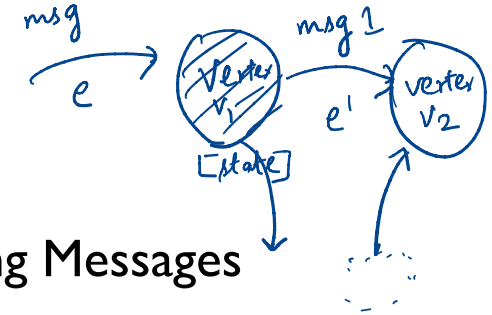


Timestamp : $(\underbrace{e \in \mathbb{N}}_{\text{epoch}}, \underbrace{\langle c_1, \ldots, c_k \rangle \in \mathbb{N}^k}_{\text{loop counters}})$

Example : (0, <0, 0, 1>)
<0, 0, 2>
<0, 1, 0>
⋮

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

→ for this new loop context
→ done with k+1 th loop
→ increment last loop counter



VERTEX API

Actor Model = Similar

Receiving Messages

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

v.OnNotify(t :Timestamp)

↳ called when all messages with ts <= t have been delivered

Sending Messages

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

this.NotifyAt(t :Timestamp) → will lead to onNotify(t)

Useful to finalize the values for an epoch

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

```
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'.
  }
}
```

$t' >= t$

IMPLEMENTING TIMELY DATAFLOW

one process

Need to track when it is safe to notify

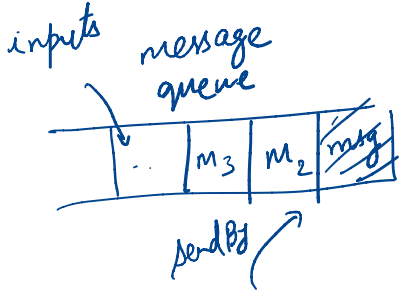
Path Summary

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

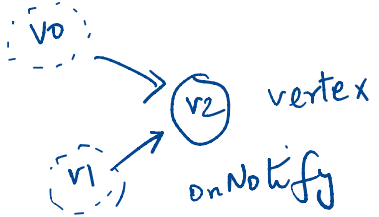
Scheduler

Occurrence and Precursor count

Precursor count = 0 → Frontier



Scheduler
pops a msg
calls v.onRecv( )



All precursors must be done with timestamp ⊤ before onNotify can be called.

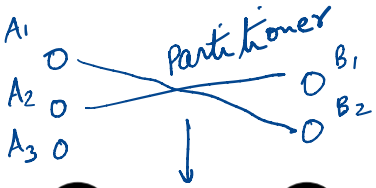
when occurence = 0
update precursor for downstream vertices

sendBy → increments occurrence count

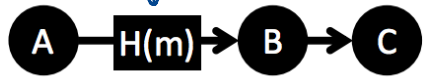
recv → decrements

long running vertices
↳ mutable state
inside them

ARCHITECHTURE



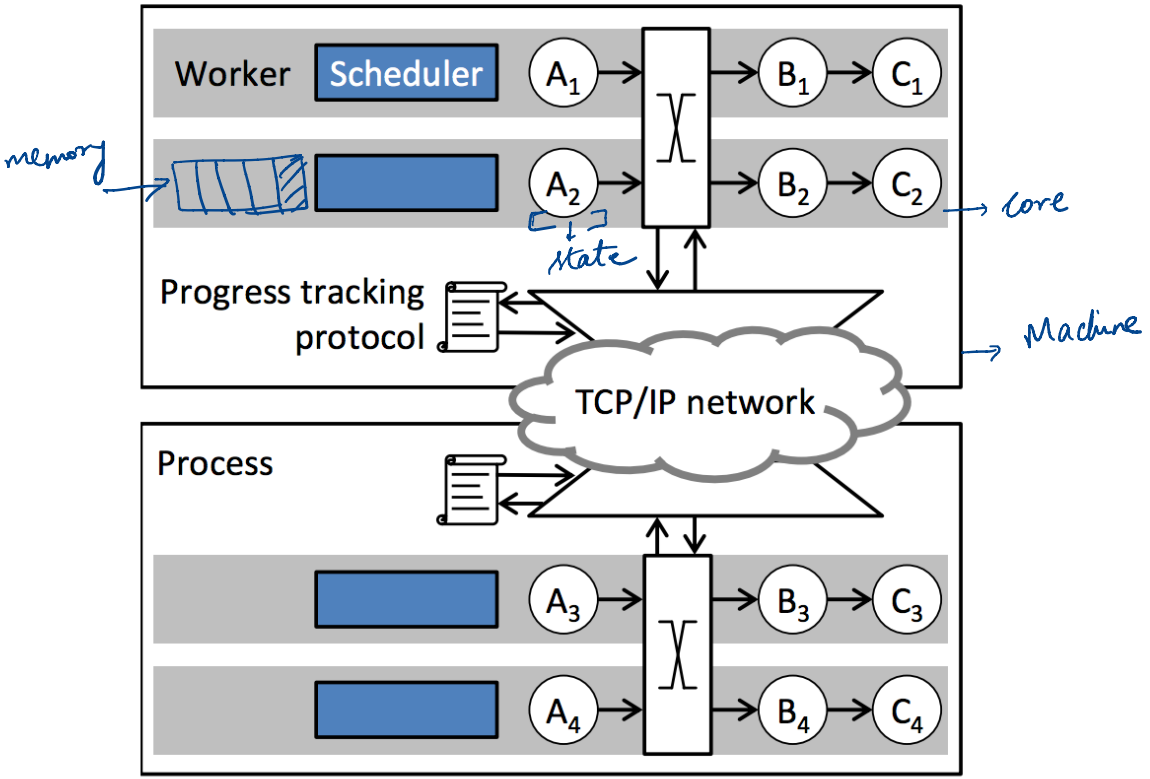
Logical graph



Workers communicate using Shared Queue

Batch messages delivered
Account for cycles

Vertex single threaded



DISTRIBUTED PROGRESS TRACKING

Broadcast-based approach

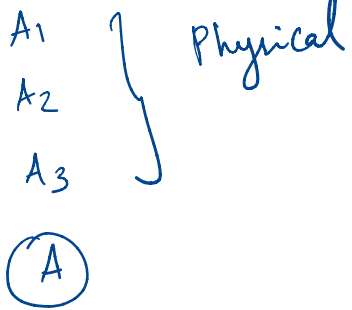
- Maintain local precursor count, occurrence count
- Send progress update (p ∈ Pointstamp, δ∈ Z)
- Local frontier tracks global frontier

when is it safe to notify

send updates to all workers before updating locally

Optimizations

- Batch updates and broadcast
- Use projected timestamps from logical graph



FAULT TOLERANCE

Mutable data in vertices

Checkpoint

- Log data as computation goes on
- Write a full checkpoint on demand

Pause worker threads

Flush message queues OnRecv

Restore

not just failed ones!

- Reset all workers to checkpoint
- Reconstruct state

Resume execution → re-doing computation!

↓

Inputs can be replayed !!

Actors in Ray / Pytorch

How frequently checkpoint ⇒ Overhead vs. Time to recovery

MICRO STRAGGLERS

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.

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();
```

Streaming

SCOPE / Spark ?

Timely Dataflow

feed inputs

SUMMARY

Stream processing → Increasingly important workload trend

Timely dataflow: Principled approach to model batch, streaming together

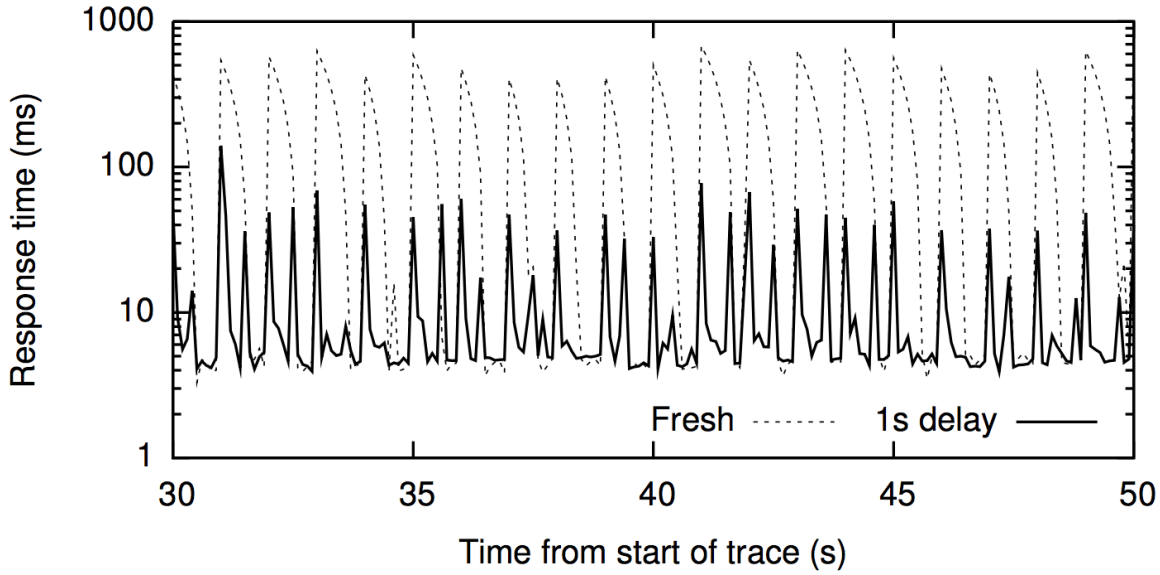
Vertex message model

- Compute frontier
- Distributed progress tracking

long running stateful vertices

DISCUSSION

https://forms.gle/qn8AzxHMT9VtyEc37


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

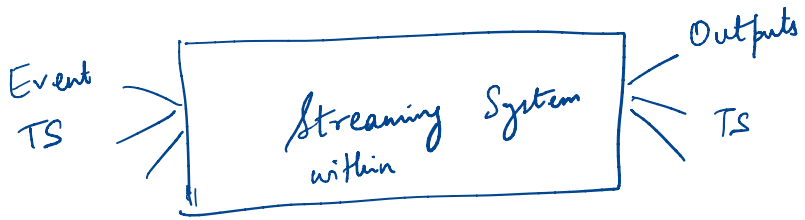
Course project peer feedback due tonight!

Timely dataflow vs. Dataflow Model

<epoch, <counters>>

event_ts

<processing_ts>



Limitations of Naiad

→ Epoch boundaries need to be defined.