Good morning!

CS 744: NAIAD

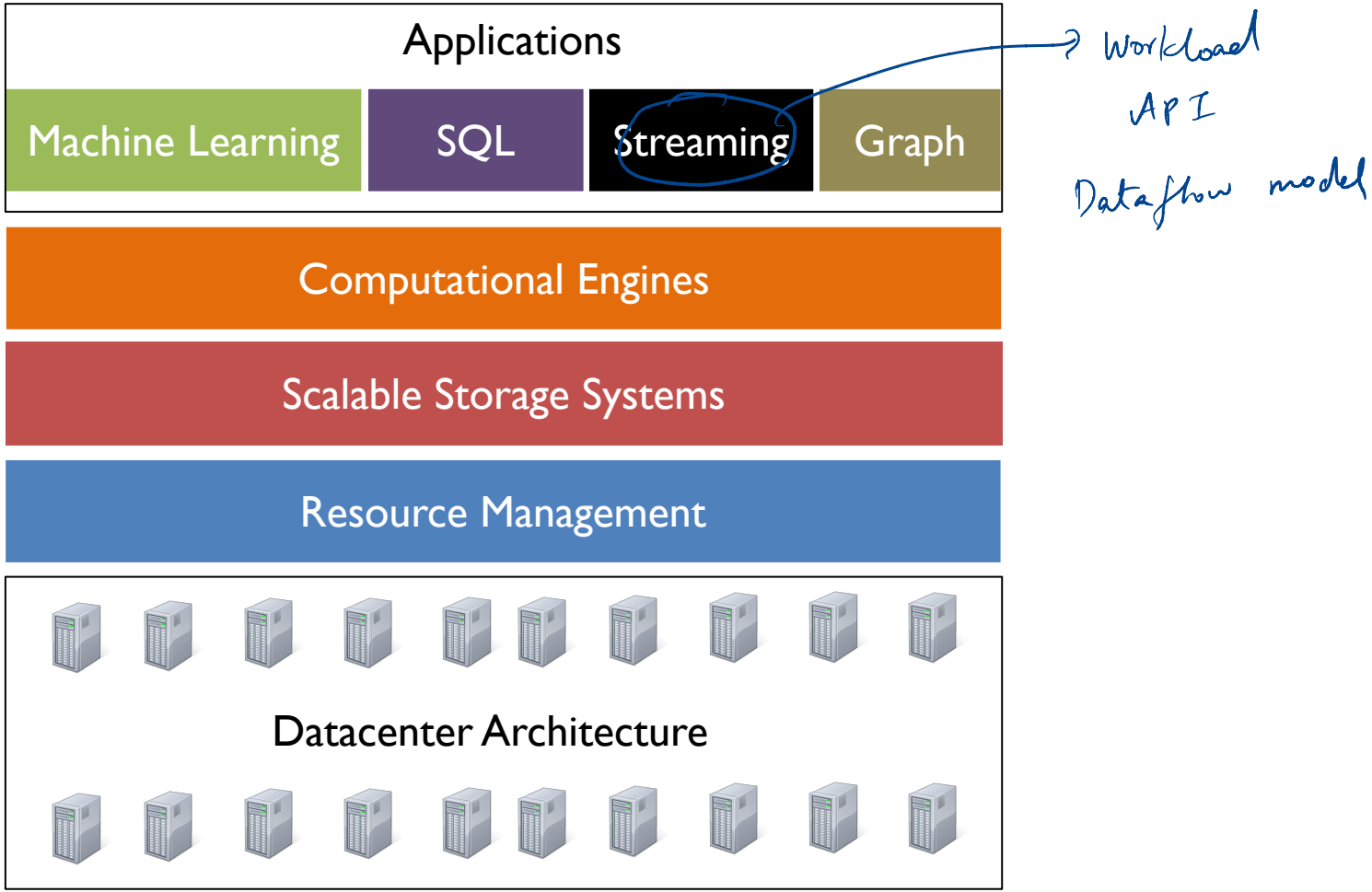
Shivaram Venkataraman

Fall 2021

ADMINISTRIVIA

In Progress

- Course Project Proposal feedback
- Midterm grading

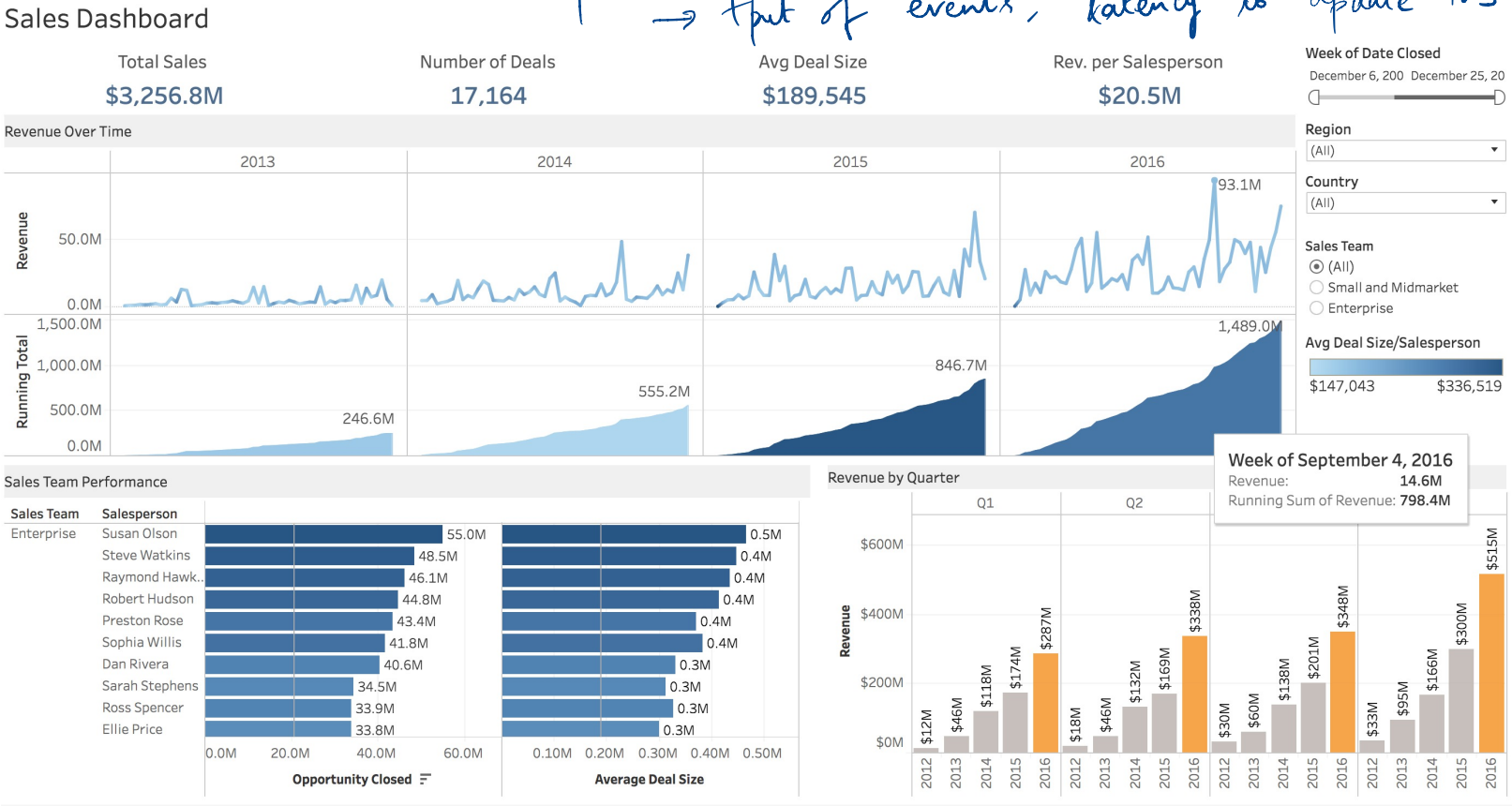


Workload API

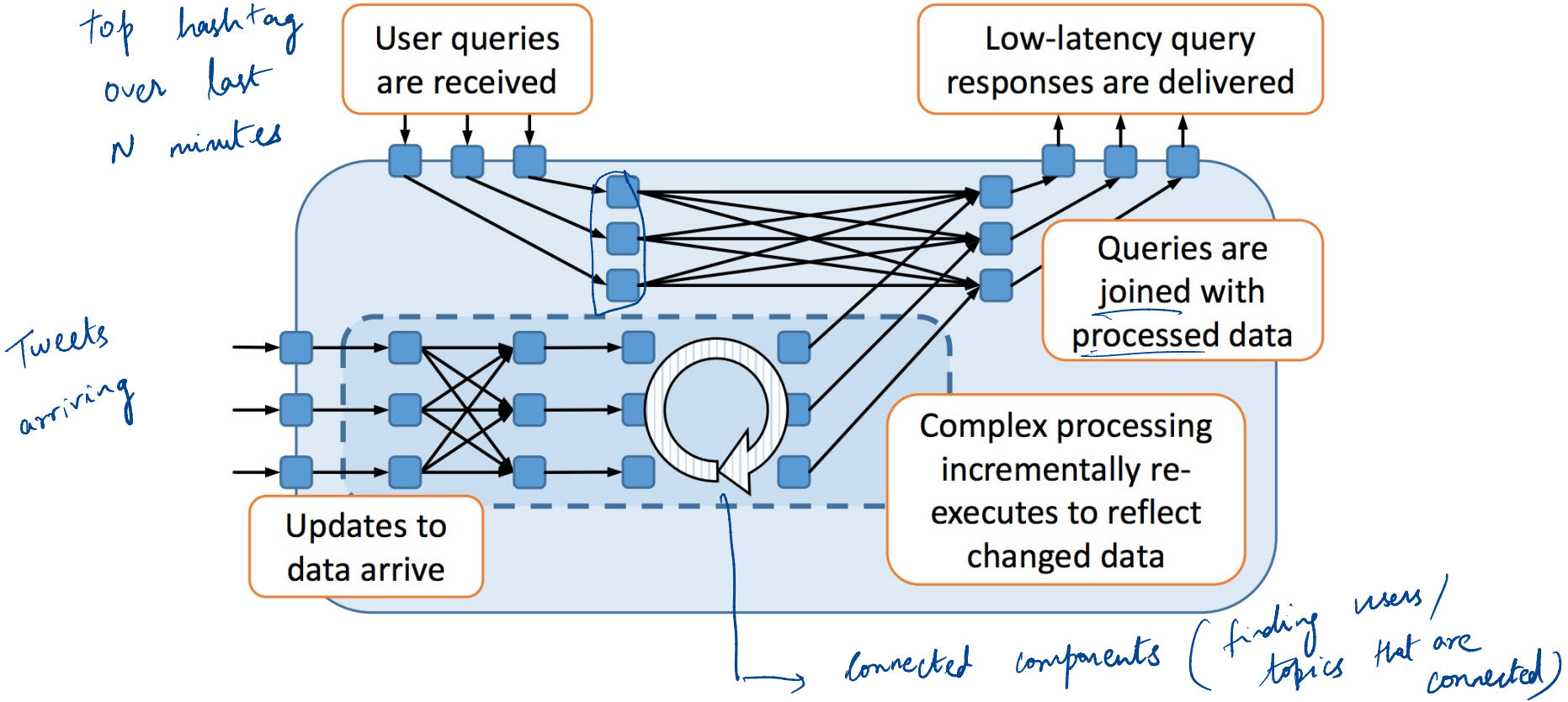
Dataflow model

DASHBOARDS

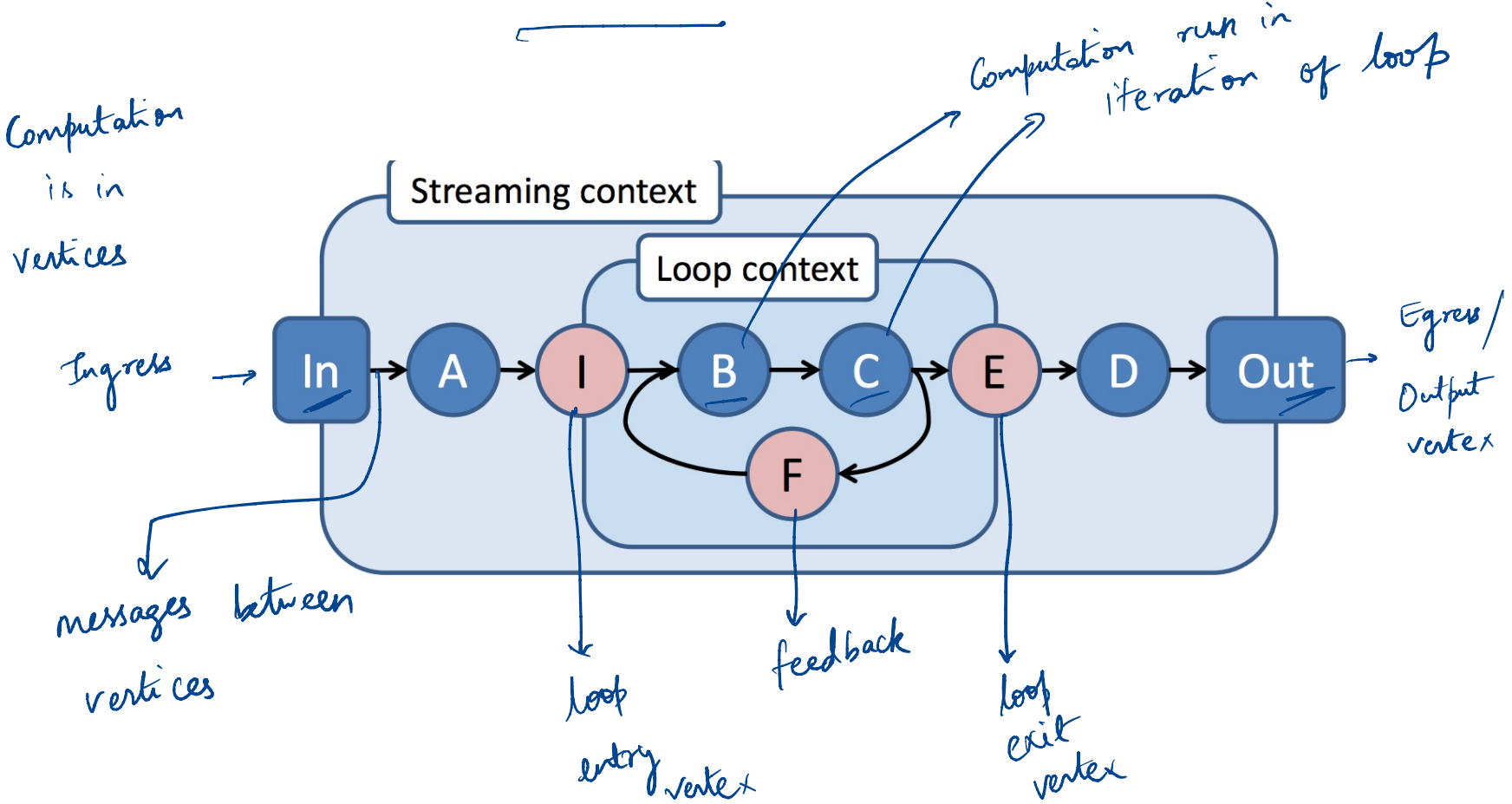
↑ Continuously updated as data arrives
→ tput of events, latency to update this



STREAMING + ITERATIVE COMPUTATION



TIMELY DATAFLOW

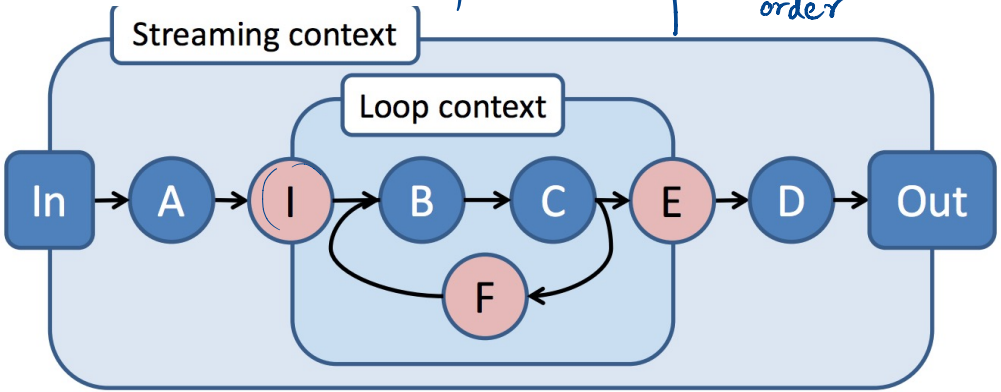


TIMELY DATAFLOW

Epoch
↳ input events have an epoch which is given

↳ notifications / output when computation for epoch finishes

epoch numbers with timestamp?
→ flexibility → Out of order



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

for i = 0 : 10
  for j = 0 : 10
    =

for loop k+1

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	$(e, \langle c_1, \ldots, c_k \rangle)$	$(e, \langle c_1, \ldots, c_k + 1 \rangle)$

↳ iter for (k+1)

nested loops

VERTEX API

Actor System (Ray)

Receiving Messages → functions that a vertex implements

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

v.OnNotify(t : Timestamp)

↳ called when all msgs <= t have been delivered.

Sending Messages → functions that vertex can call

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

↳ send a msg m on this edge e, with t as timestamp

this.NotifyAt(t : Timestamp)

↳ this vertex will not produce any outputs with < t.

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);
  }
}
```

is a vertex

"state" which is inside this vertex

notify for older timestamps

initialize dict for this timestamp

↳ tell output1 that we have a new msg!

is this a new event (msg) if so counter = 0

↳ increments counter

(msg, counter) write this as output at the end of epoch

IMPLEMENTING TIMELY DATAFLOW

Need to track when it is safe to notify

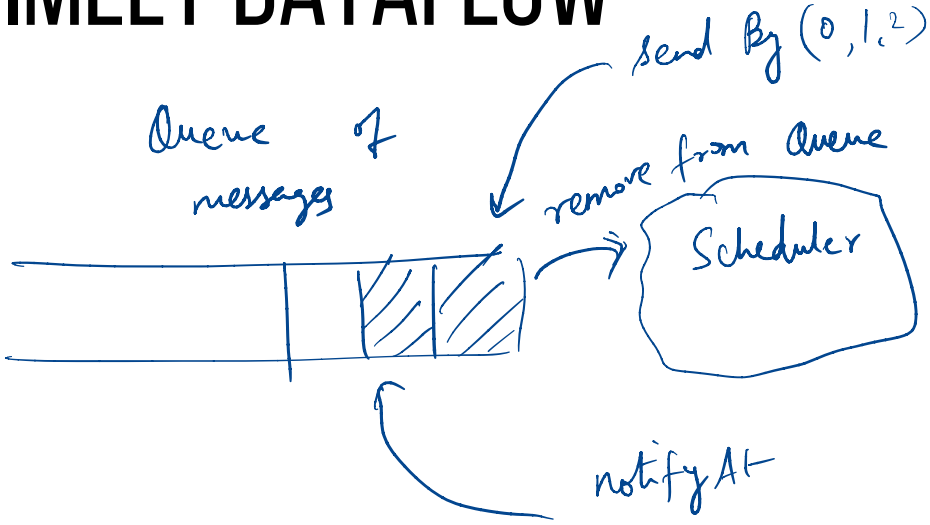
Path Summary

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

Scheduler

Occurrence and Precursor count

Precursor count = 0 → Frontier



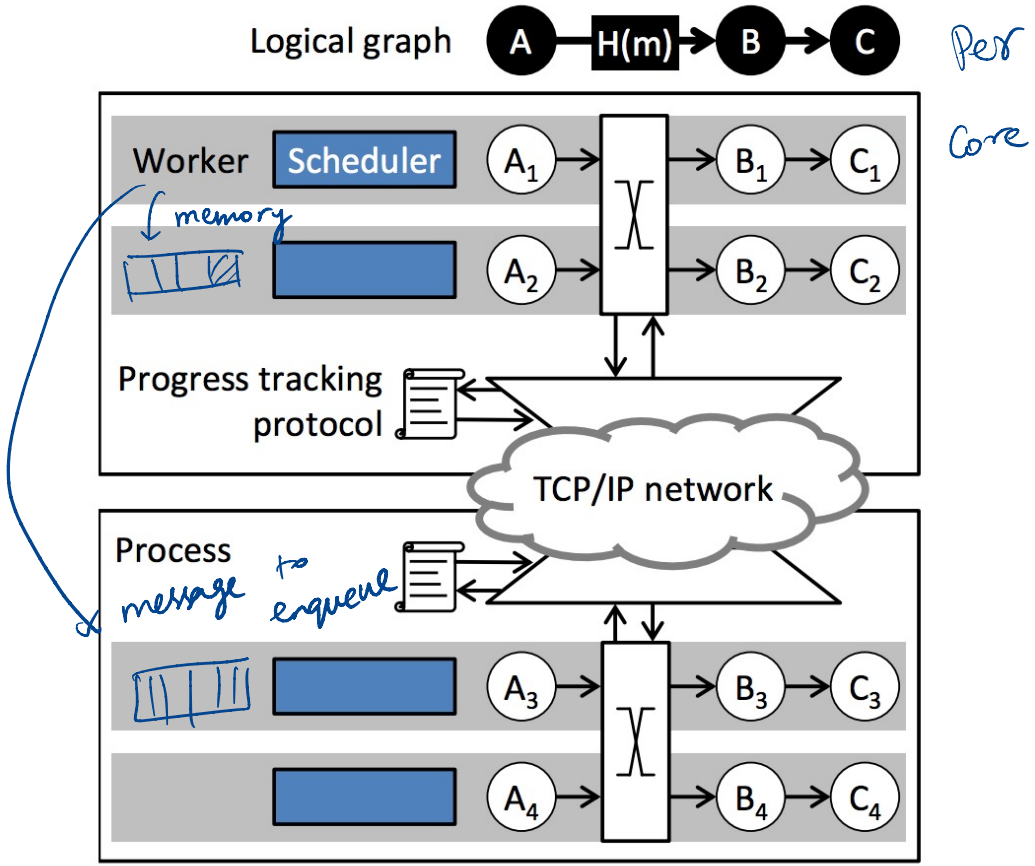
Identical workers (no centralized Master)

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

helps you derive a local frontier that tracks global

when is it safe to notify at local machine given global progress

FAULT TOLERANCE

mutable data in vertex

Checkpoint()

Log data as computation goes on

Write a full checkpoint on demand

Pause worker threads

Flush message queues OnRecv

it is safe to checkpoint / consistent

Restore

Reset all workers to checkpoint

not just the failed worker!!

Reconstruct state

Resume execution

↳ redo work from the checkpoint

When failure is infrequent get better perf when no failures!

MICRO STRAGGLERS

What is different from stragglers in MapReduce?

→ duplicate or back up tasks

Sources of stragglers

Network

Concurrency

Garbage Collection

minimize prob. of straggler!

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

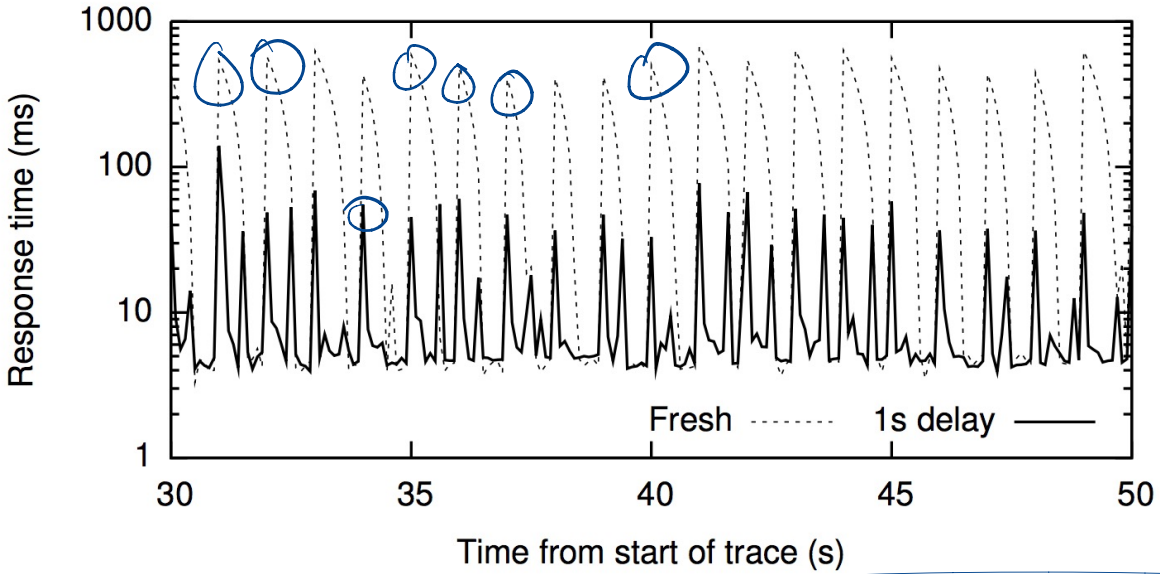
- Compute frontier
- Distributed progress tracking

DISCUSSION

https://forms.gle/cYYy14JfX71qBzY66

Input:

32,000 tweets/second, Queries every 100ms



① t = Fresh
suffers from bigger slowdown than t = 1s

stale data already been processed

② Query is waiting for incremented connected components

materialize · IO

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!~~