CS 744: SPARK STREAMING

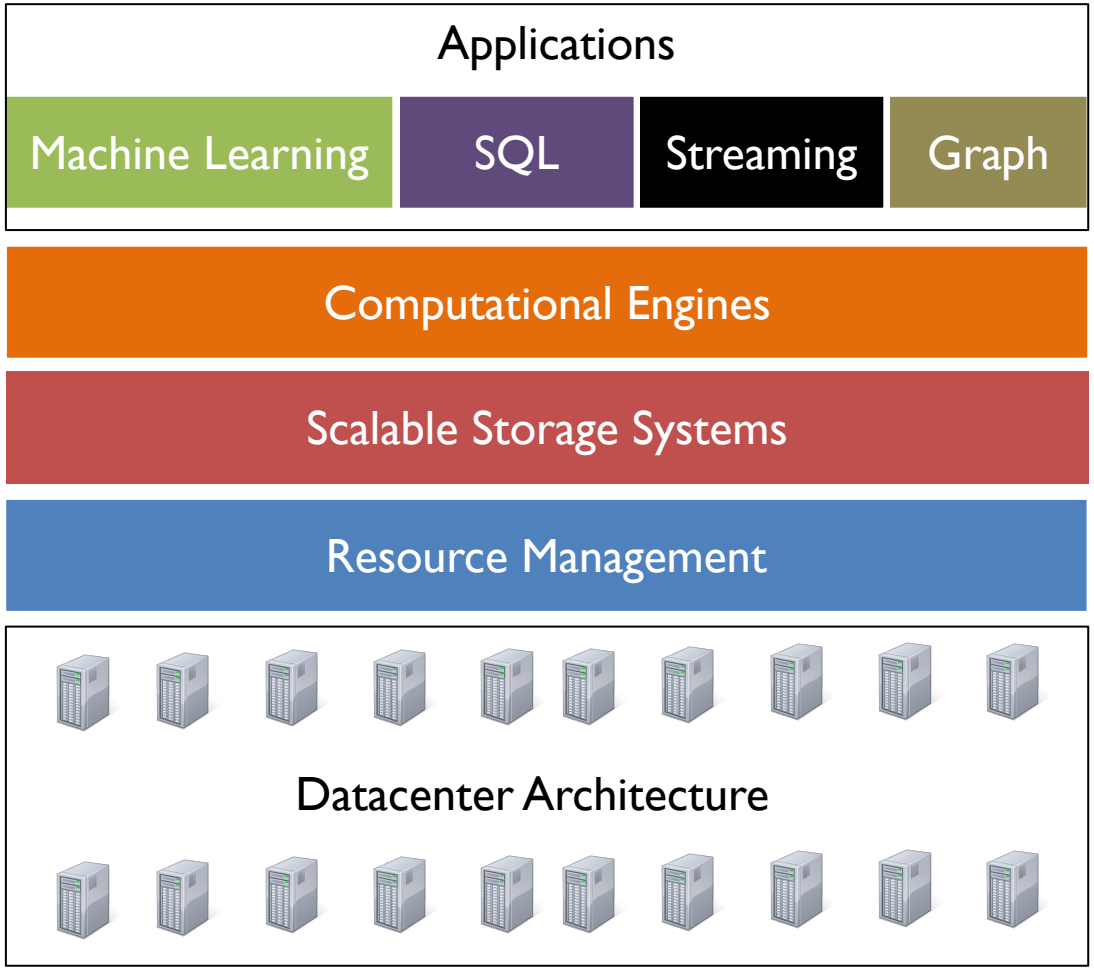
Shivaram Venkataraman

Spring 2024

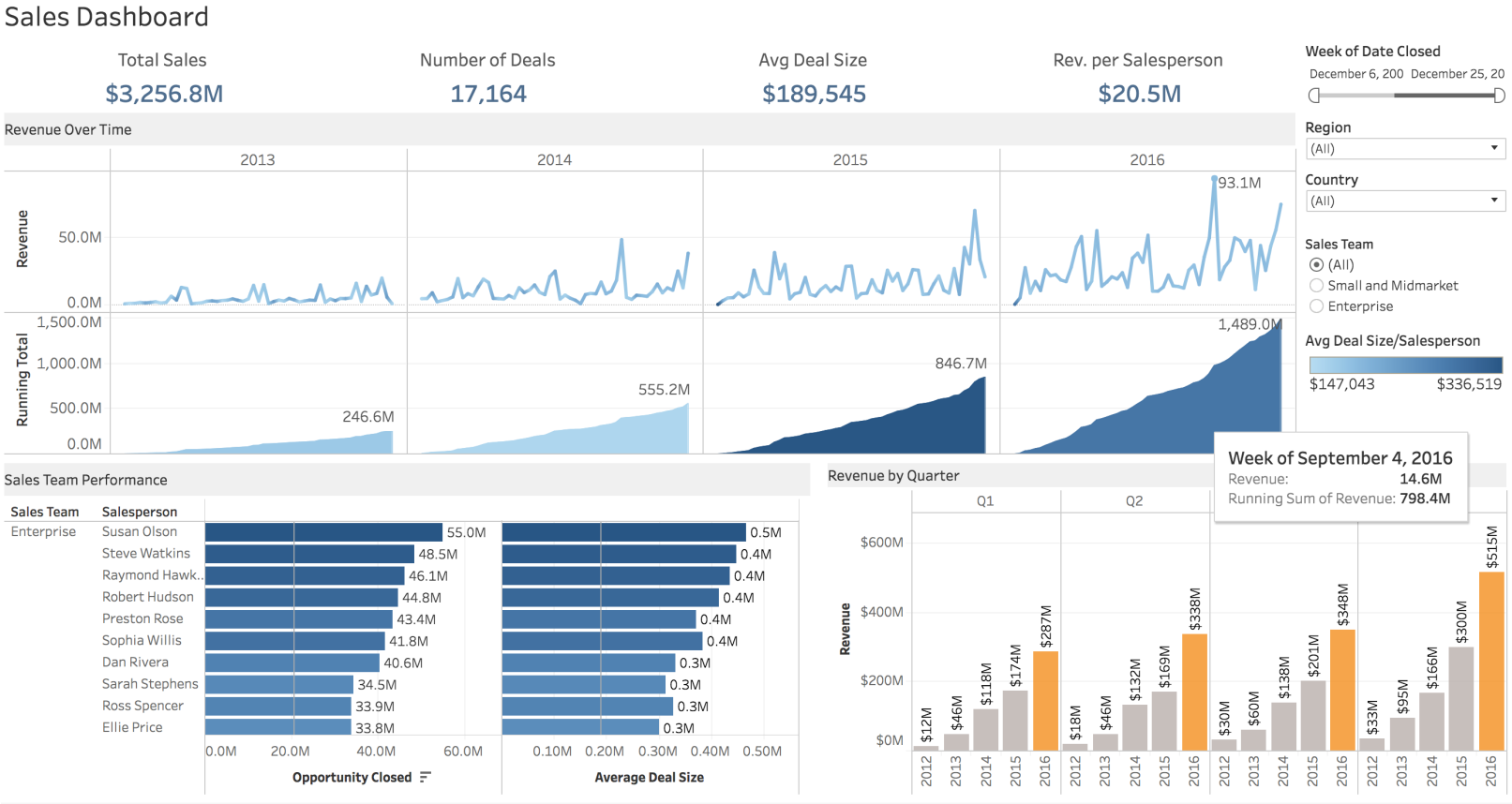
ADMINISTRIVIA

- Course Projects feedback
- Midterm grades – this week?

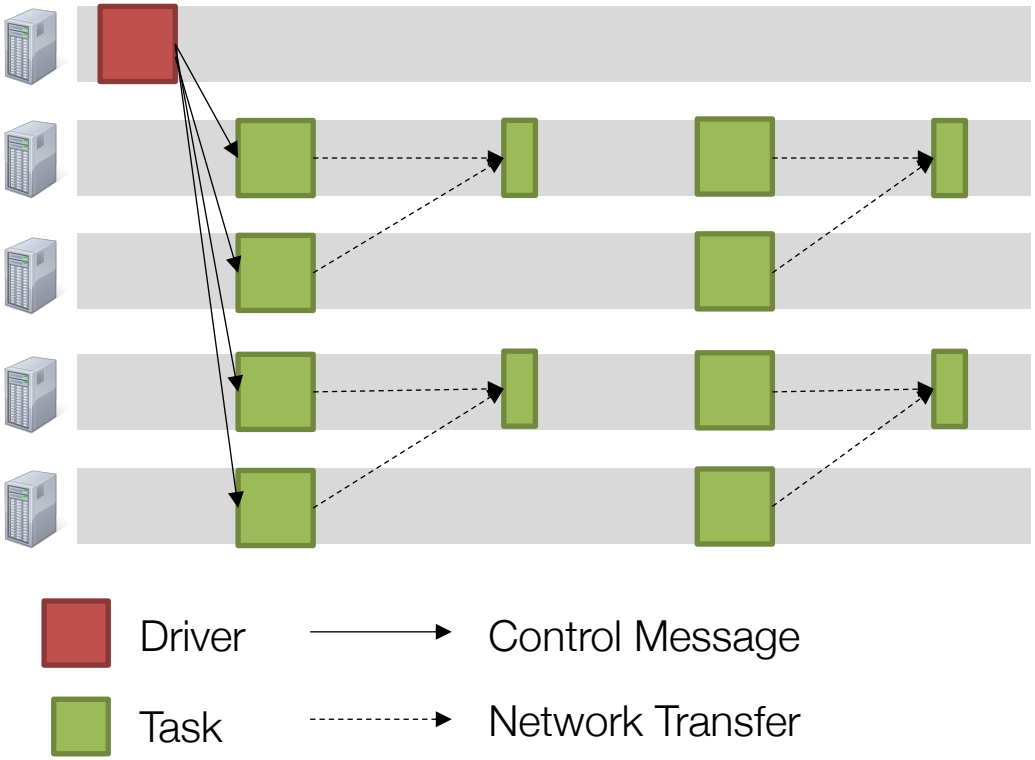
- Cloudlab reservations
  - Per-user from now


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
Steve Watkins
Raymond Hawk..
Robert Hudson
Preston Rose
Sophia Willis
Dan Rivera
Sarah Stephens
Ross Spencer
Ellie Price
55.0M
48.5M
46.1M
44.8M
43.4M
41.8M
40.6M
34.5M
33.9M
33.8M
0.5M
0.4M
0.4M
0.4M
0.4M
0.4M
0.3M
0.3M
0.3M
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

CONTINUOUS OPERATOR MODEL

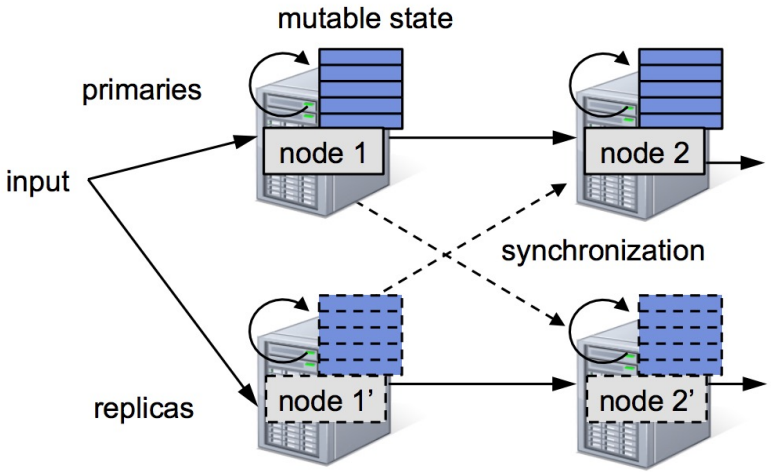


- Long-lived operators
- Mutable State
- Distributed Checkpoints for Fault Recovery
- Stragglers ?

Flink

Naiad

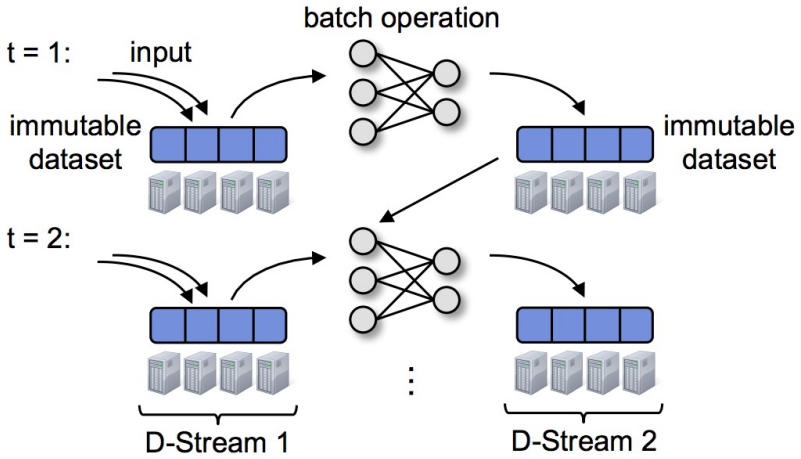
CONTINUOUS OPERATORS



SPARK STREAMING: GOALS

1. Scalability to hundreds of nodes
2. Minimal cost beyond base processing (no replication)
3. Second-scale latency
4. Second-scale recovery from faults and stragglers

DISCRETIZED STREAMS (DSTREAMS)

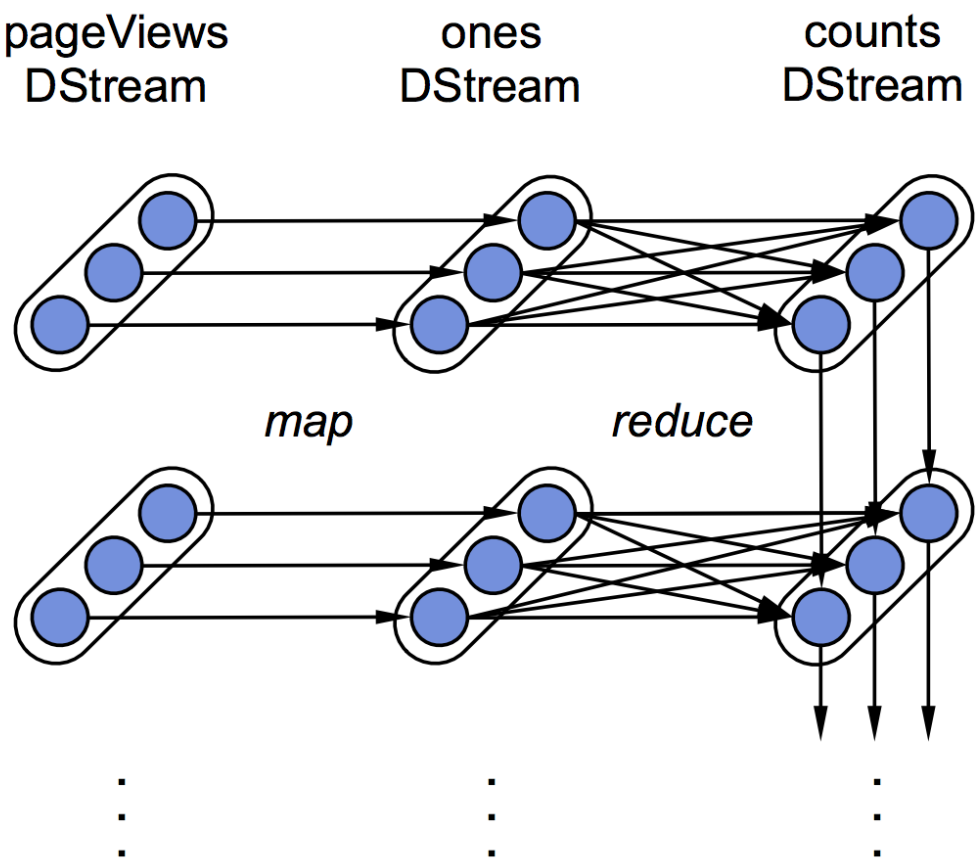


EXAMPLE

```
pageViews =
  readStream(http://...,
             "1s")

ones = pageViews.map(
   event =>(event.url, 1))

counts =
    ones.runningReduce(
        (a, b) => a + b)
```



DSTREAM API

Transformations

Stateless: map, reduce, groupBy, join

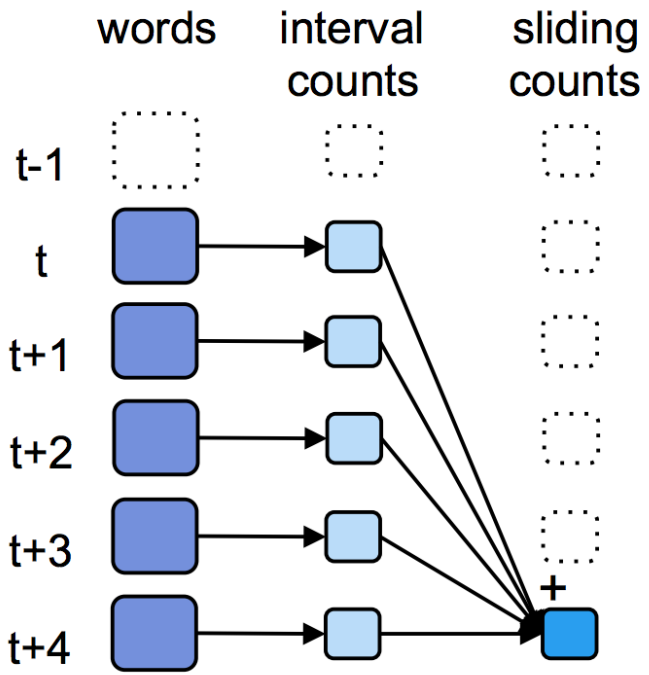
Stateful:

Sliding window("5s") → RDDs with data in [0,5), [1,6), [2,7)

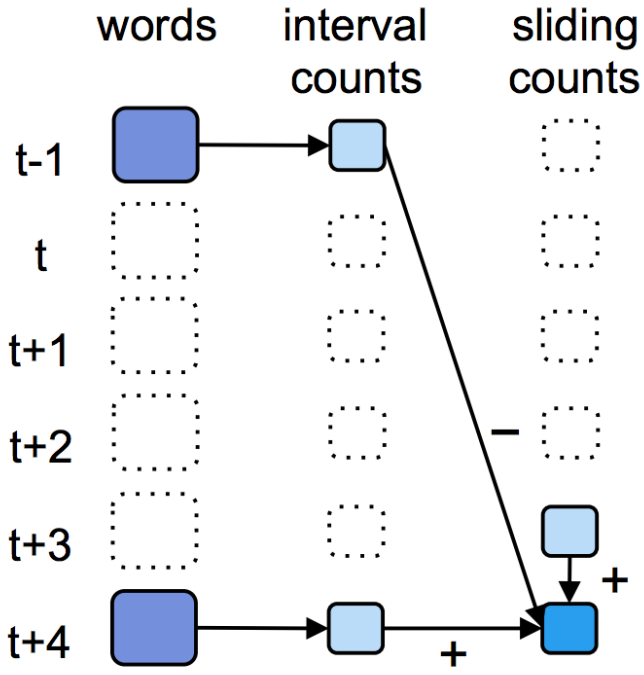
reduceByWindow("5s", (a, b) => a + b)

SLIDING WINDOW

Add previous 5 each time



(a) Associative only



(b) Associative & invertible

STATE MANAGEMENT

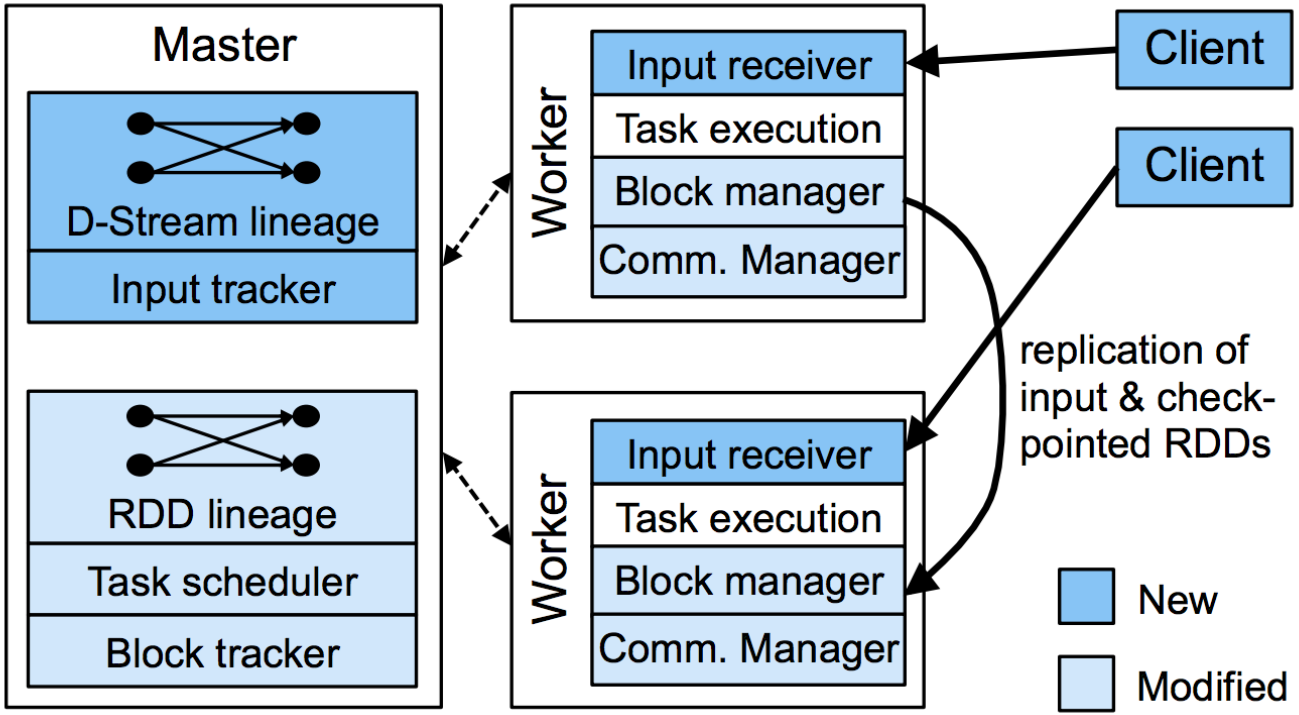
Tracking State: streams of (Key, Event) → (Key, State)

```
events.track(
  (key, ev) => 1,

  (key, st, ev) => ev == Exit ? null : 1,

  "30s”)
```

SYSTEM IMPLEMENTATION



OPTIMIZATIONS

Timestep Pipelining

- No barrier across timesteps unless needed
- Tasks from the next timestep scheduled before current finishes

Checkpointing

- Async I/O, as RDDs are immutable
- Truncate lineage after checkpoint

FAULT TOLERANCE: PARALLEL RECOVERY

Worker failure

- Need to recompute state RDDs stored on worker
- Re-execute tasks running on the worker

Strategy

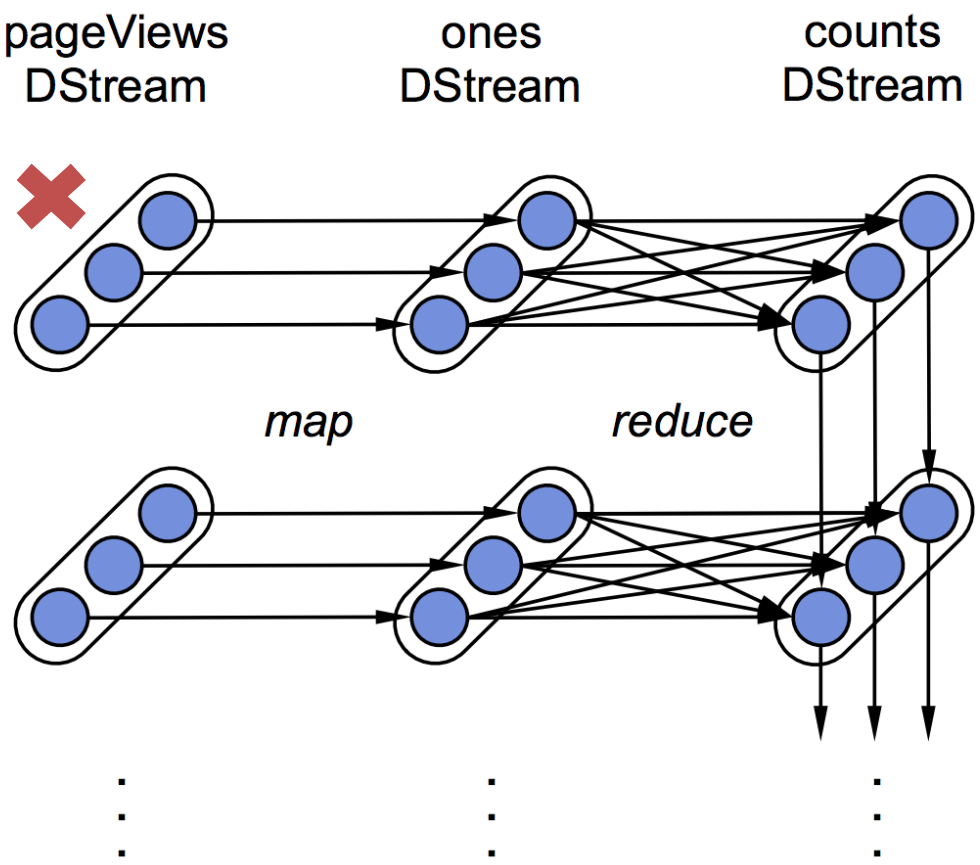
- Run all independent recovery tasks in parallel
- Parallelism from partitions *in timestep* and *across timesteps*

EXAMPLE

```
pageViews =
  readStream(http://...,
             "1s")

ones = pageViews.map(
   event =>(event.url, 1))

counts =
    ones.runningReduce(
        (a, b) => a + b)
```



FAULT TOLERANCE

Straggler Mitigation: Use speculative execution

Driver Recovery

- At each timestep, save graph of DStreams and Scala function objects
- Workers connect to a new driver and report their RDD partitions
- Note: No problem if a given RDD is computed twice (determinism).

SUMMARY

Micro-batches: New approach to stream processing

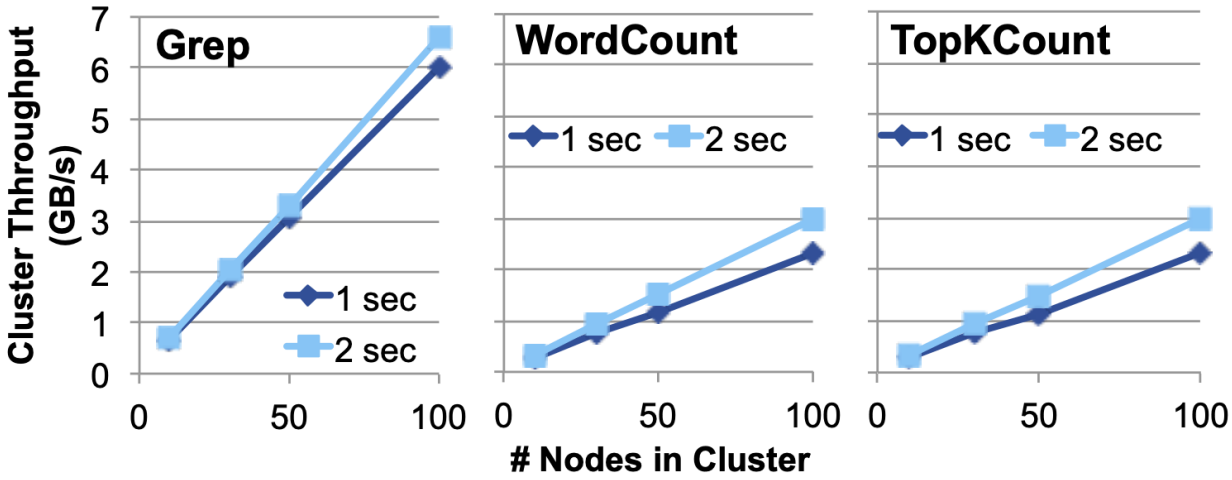
Simplifies fault tolerance, straggler mitigation

Unifying batch, streaming analytics

DISCUSSION

https://forms.gle/RVtChgDQzbX16tqT7

If the latency bound was made to 100ms, how do you think the above figure would change? What could be the reasons for it?



Consider the pros and cons of approaches in Flink vs Spark Streaming. What application properties would you use to decide which system to choose?

NEXT STEPS

Next class: Graph processing!

Midterm grades soon!