

CS 744: SPARK STREAMING

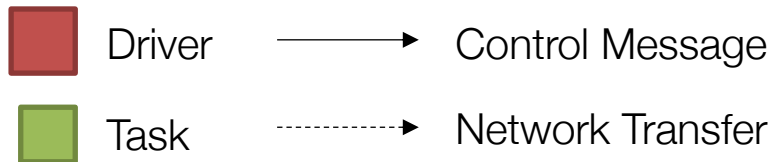
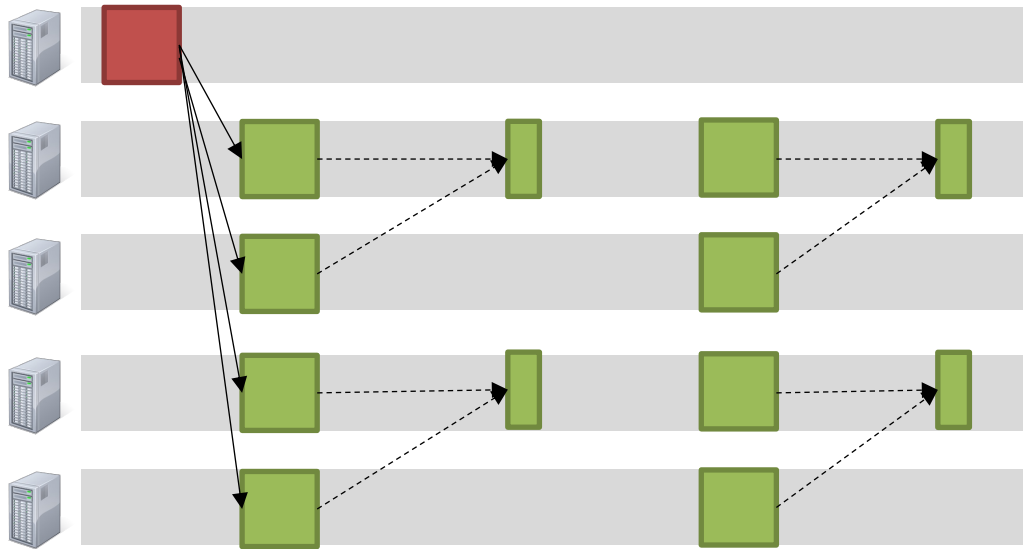
Shivaram Venkataraman

Fall 2022

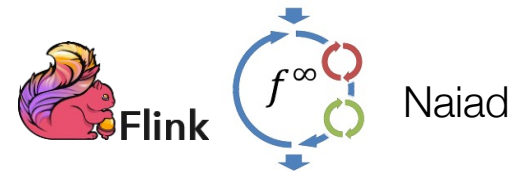
ADMINISTRIVIA

- Course Projects feedback
- Assignment2 grades
- Midterm grades – this week?

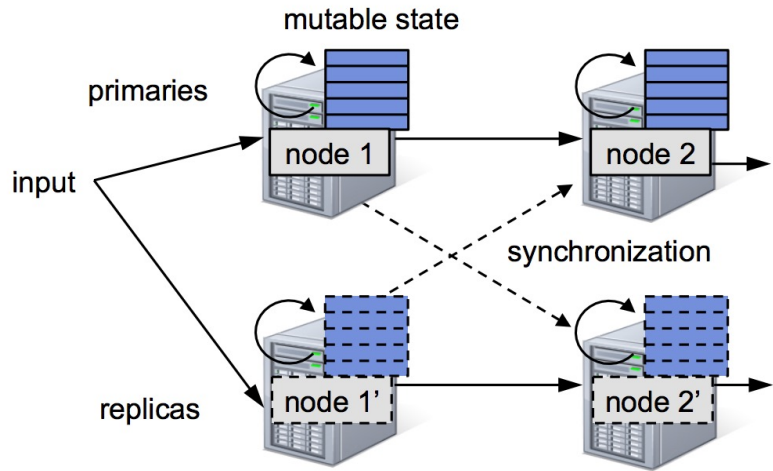
CONTINUOUS OPERATOR MODEL



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



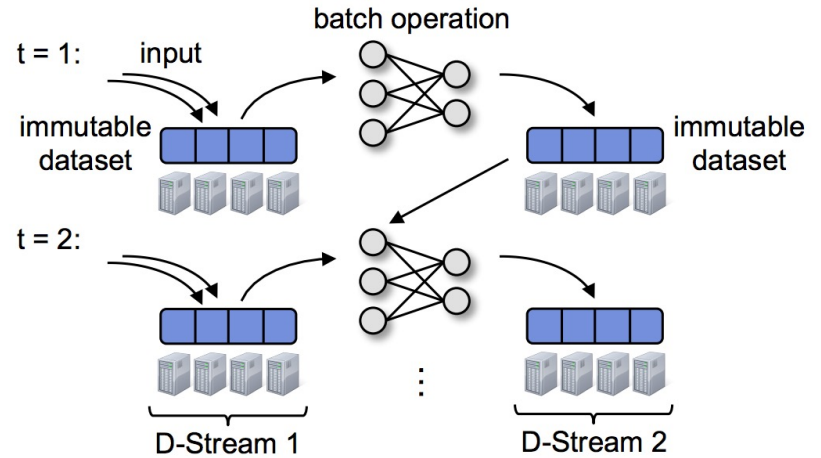
CONTINUOUS OPERATORS



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)

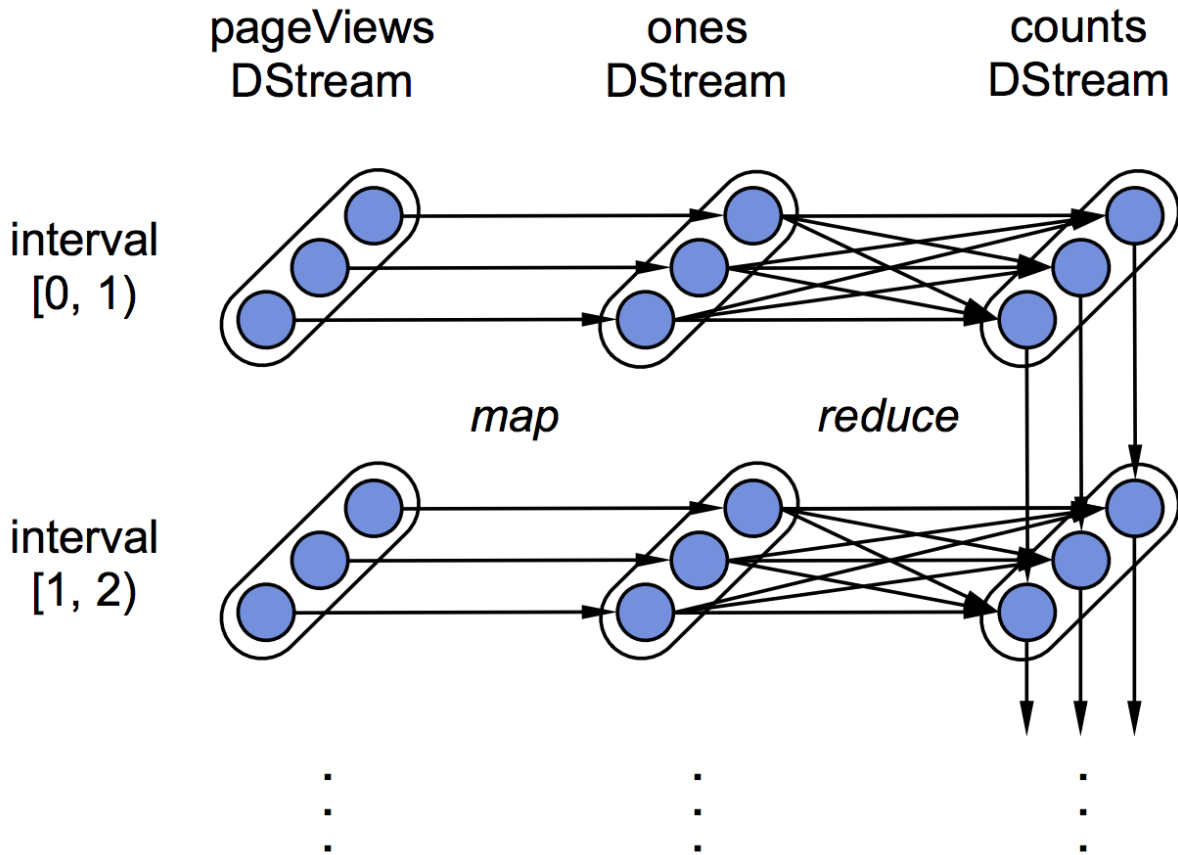


EXAMPLE

```
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

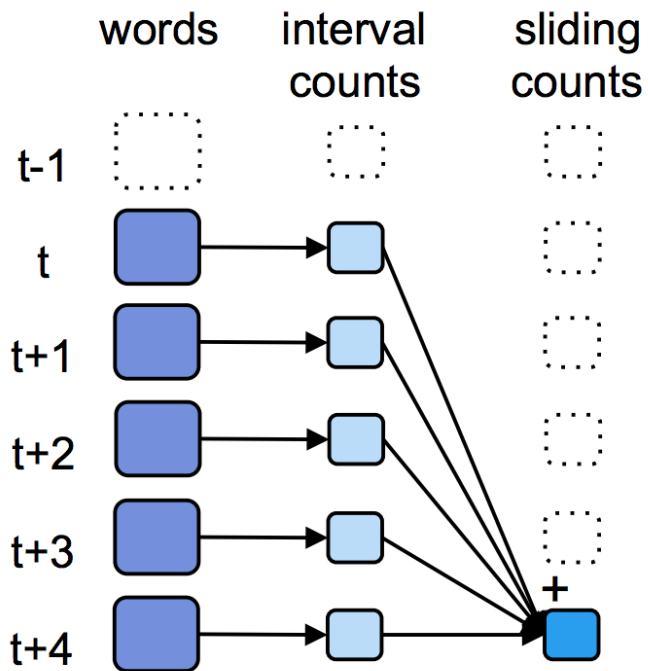
Stateful:

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

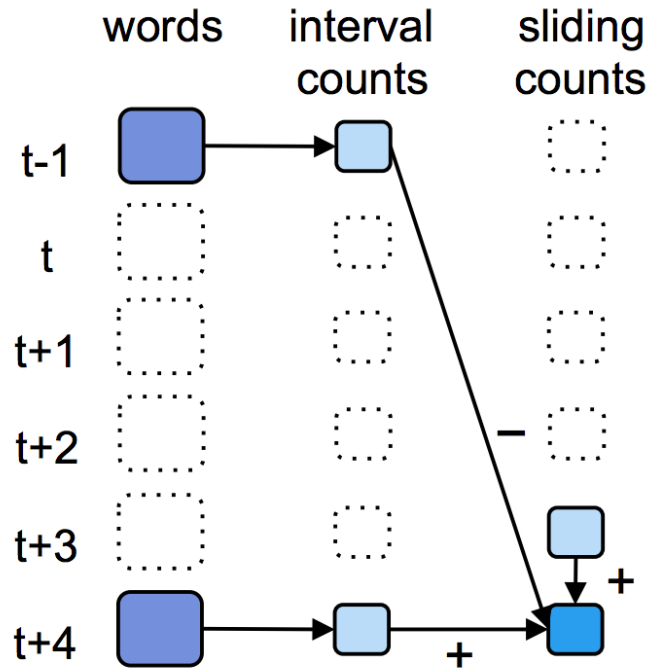
reduceByWindow("5s", (a, b) \Rightarrow a + b)

SLIDING WINDOW

Add
previous 5
each time



(a) Associative only



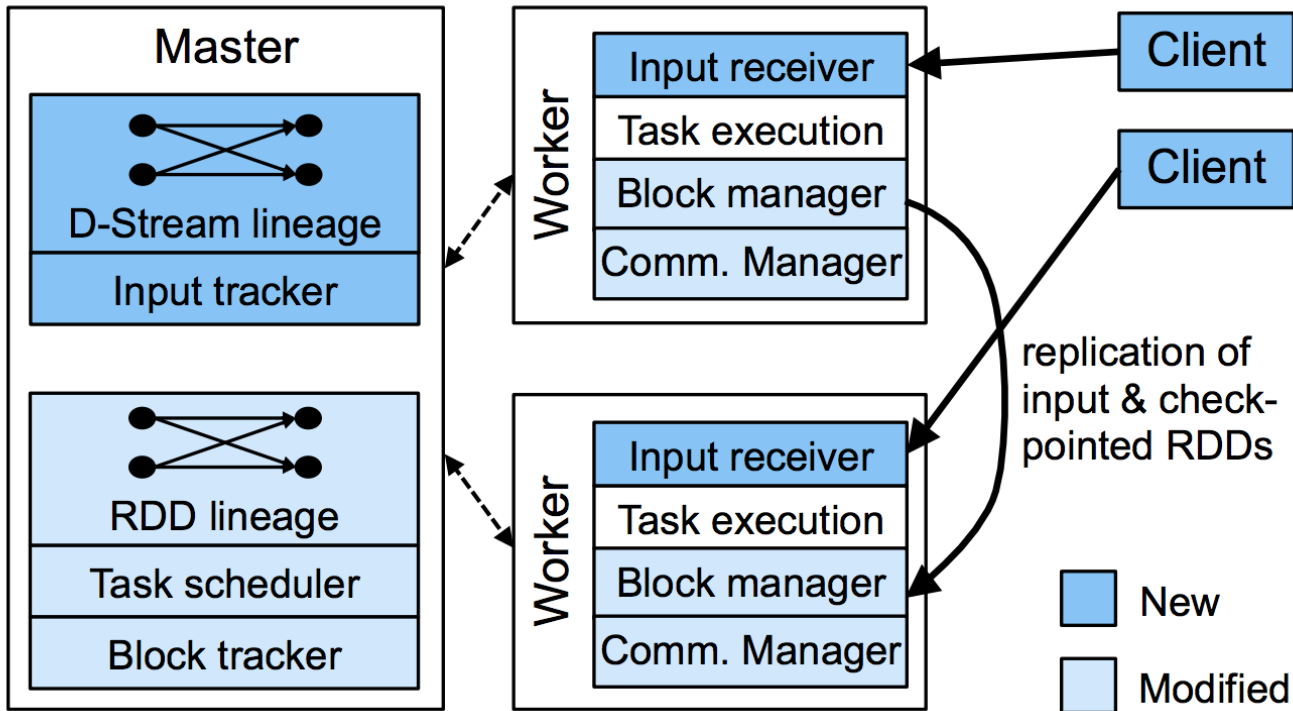
(b) Associative & invertible

STATE MANAGEMENT

Tracking State: streams of (Key, Event) \rightarrow (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

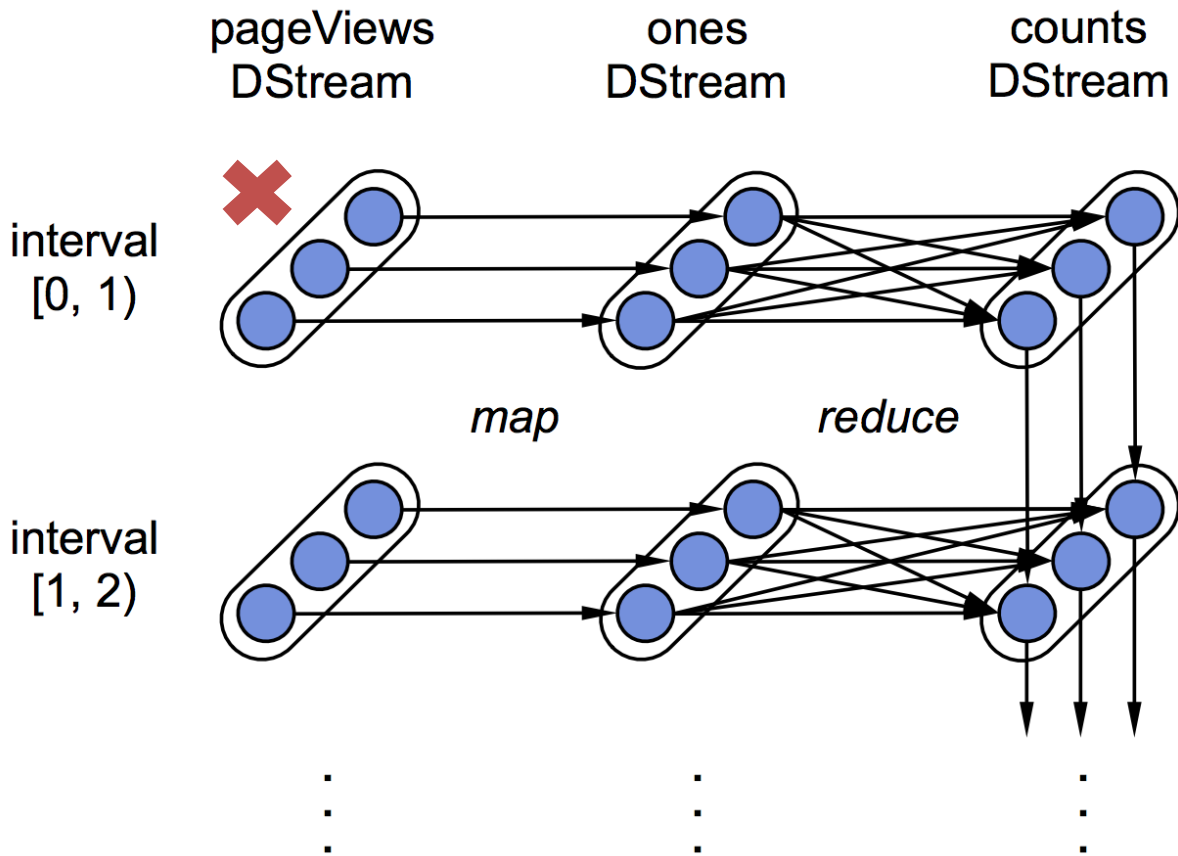
- Run all independent recovery tasks in **parallel**
- Parallelism from partitions *in timestep* and *across timesteps*

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

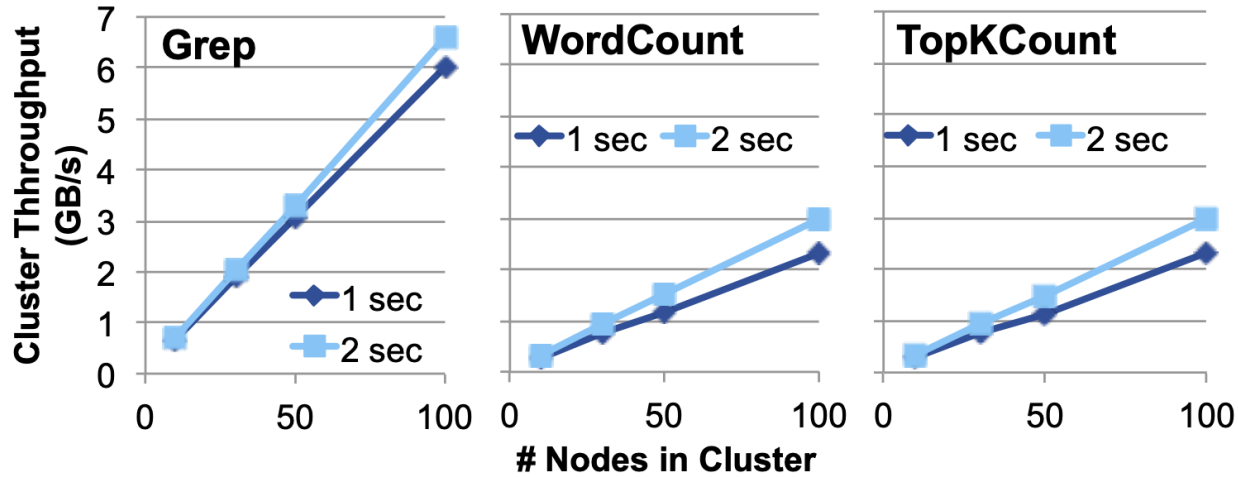
Simplifies fault tolerance, straggler mitigation

Unifying batch, streaming analytics

DISCUSSION

<https://forms.gle/rkBykWeSgiQhPjf57>

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 Naiad vs Spark Streaming. What application properties would you use to decide which system to choose?

NEXT STEPS

Next class: Graph processing!

Midterm grades soon!