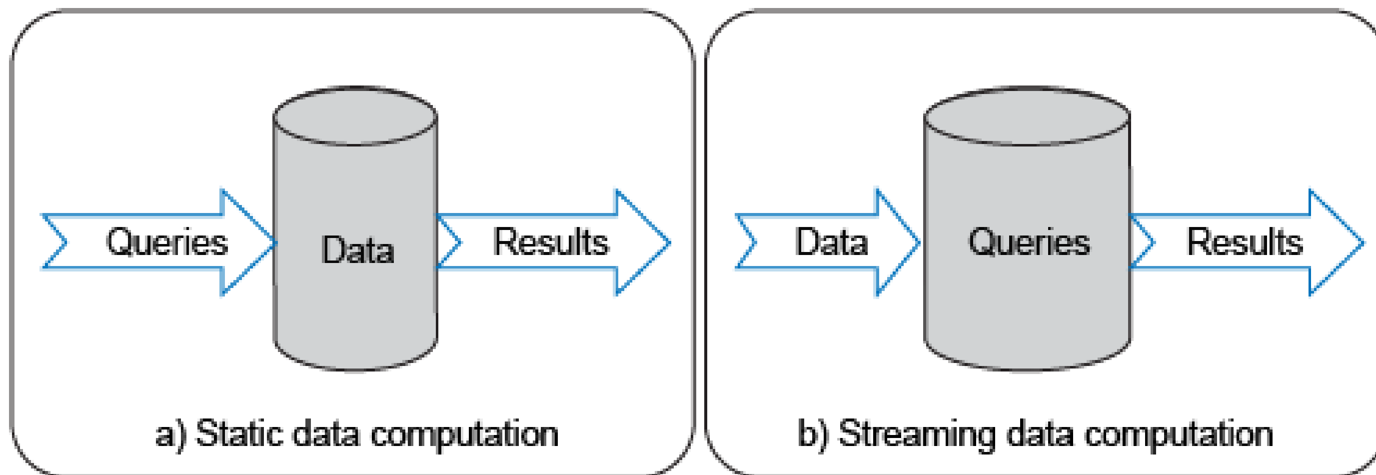
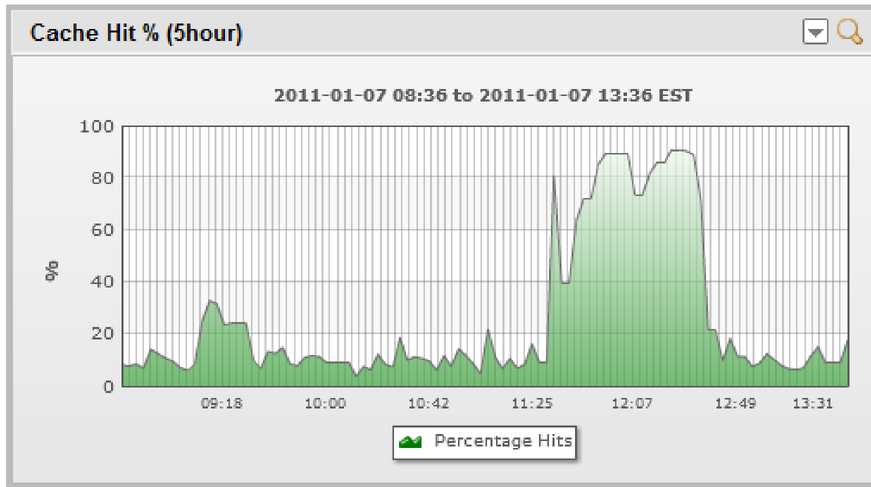
*by Matei Zaharia, Tathagata Das, Haoyuan Li,
Timothy Hunter, Scott Shenker, Ion Stoica*

Streaming Computations

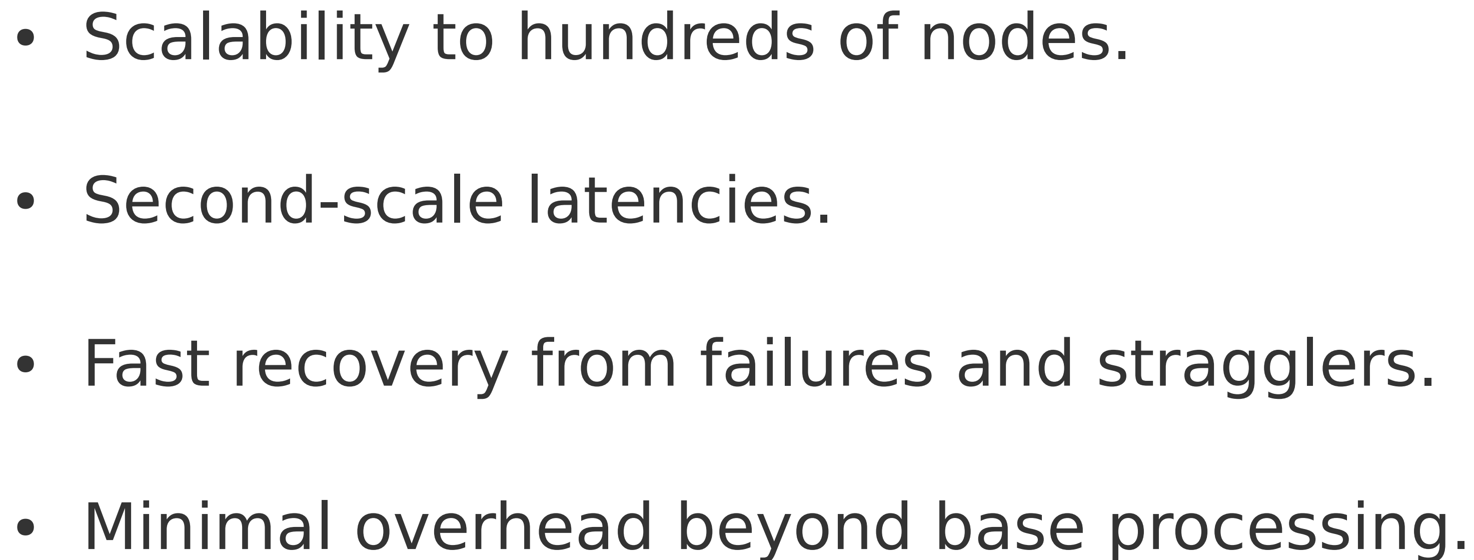
Processing large data sets in real time with low latency.



Motivation

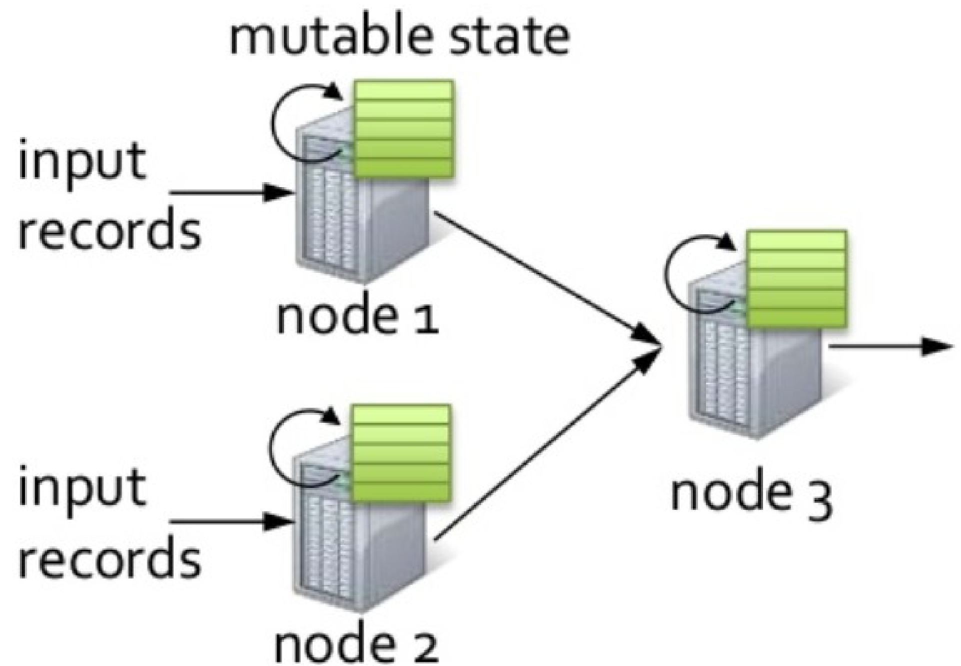


Goals

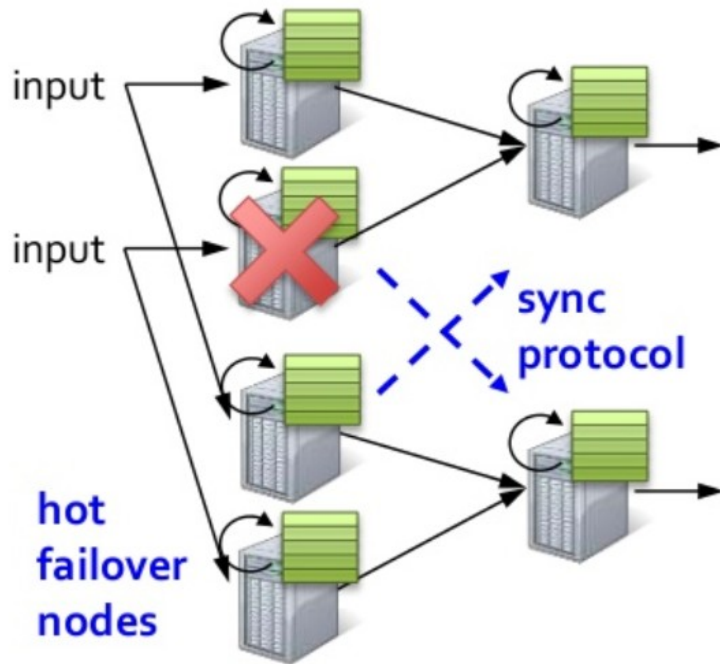


Continous Operator Model

- Computations are divided into *long-living*, *stateful* operators.
- Operator processes input records.
- Changes its' state.
- Sends new records in response.



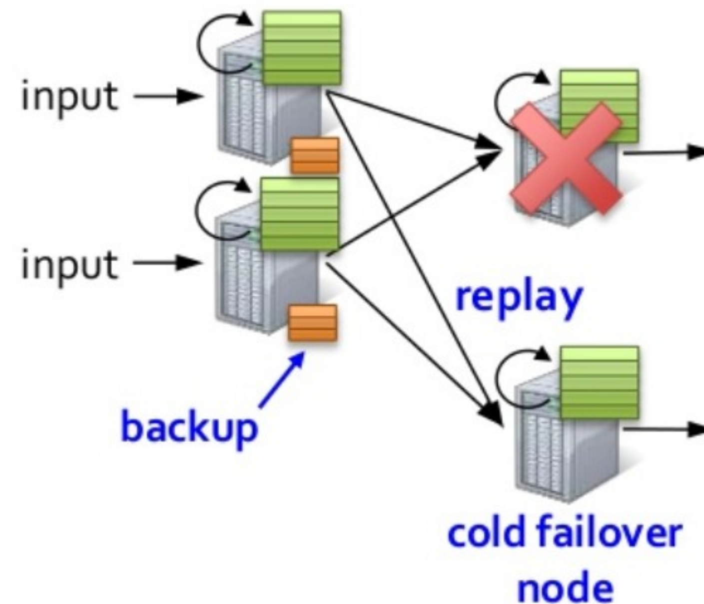
Node Replication



- Nodes are duplicated.
- Synchronization protocols ensure data ordering.
- On failure we switch to other node.
- Fast Recovery
- **2x Costs**

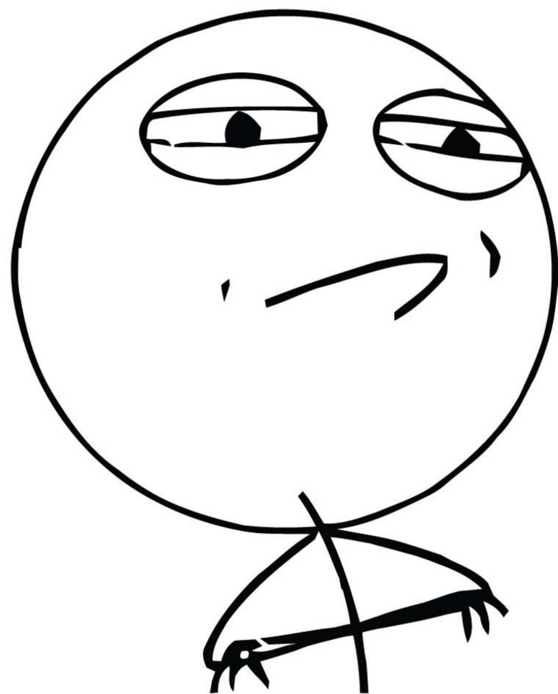
Upstream Backup

- Each node buffers send data.
- On failure state is recovered by resending data to hot standb node.
- **No Fast Recovery**



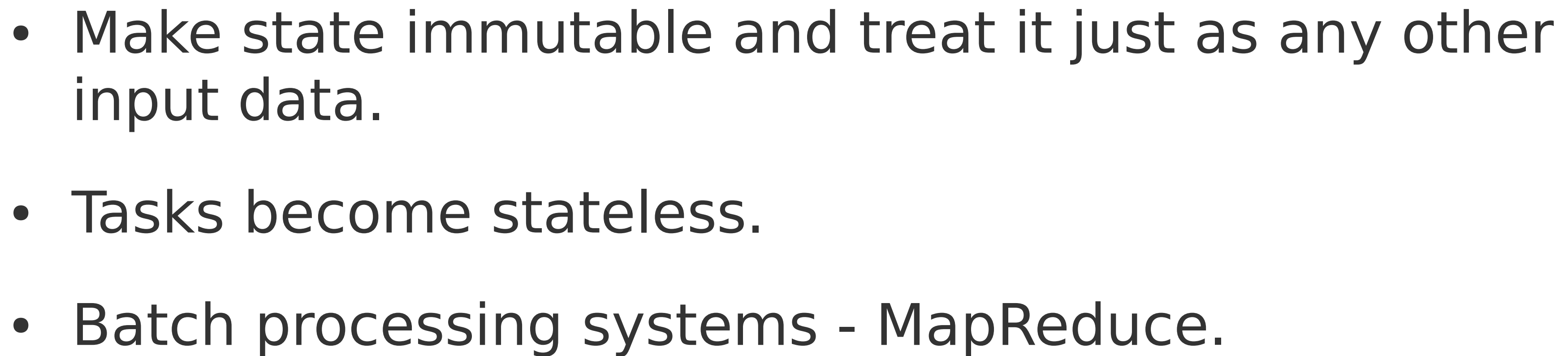
Problem

Computations are tightly integrated with mutable state which is hard to move around.



CHALLENGE ACCEPTED

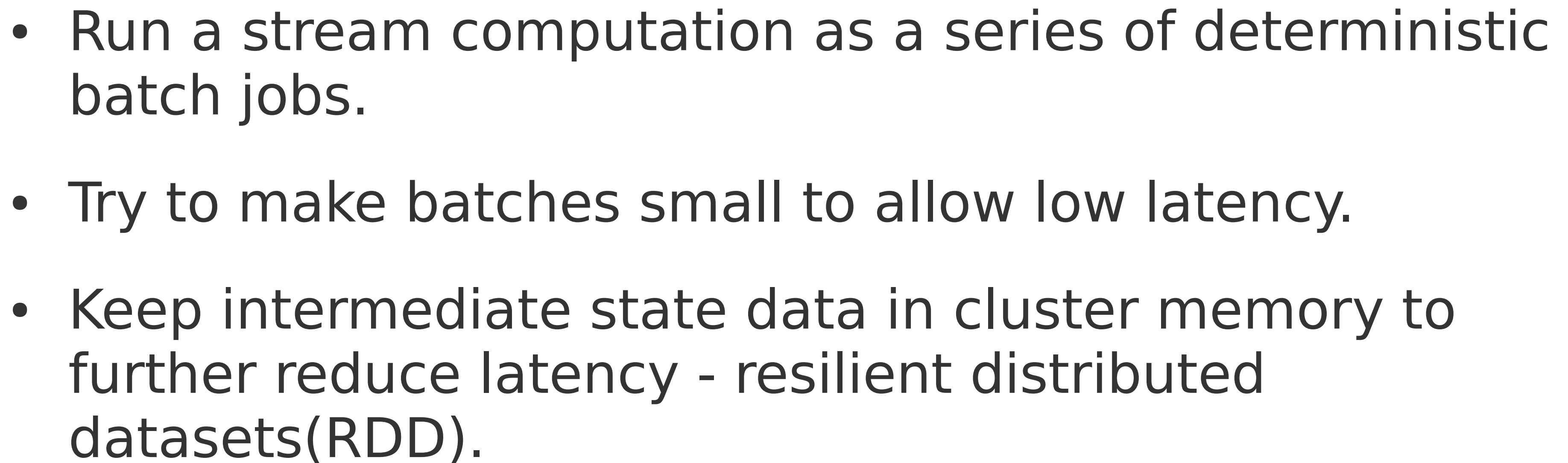
Remedy



Discretized Streams - DStreams

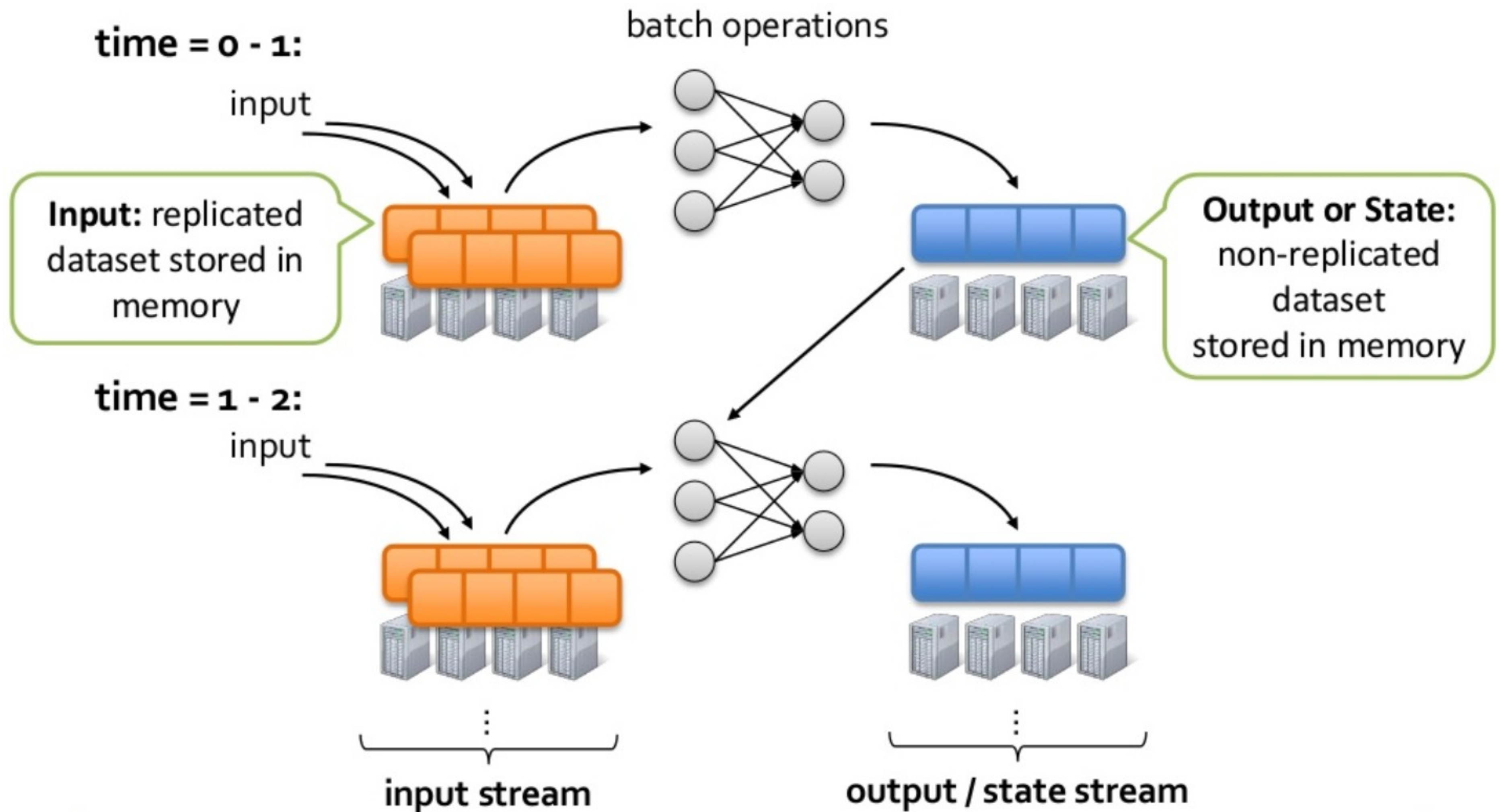
- Created by gathering streaming data from small time intervals.
- Allow small overhead to gather data.
- Seqeunce of immutable, small partitioned datasets.
- Can also be created:
  - by applying transformations on other DStreams.
  - from stored data.
  - by combining few DStreams.

Discretized Stream Processing



- Run a stream computation as a series of deterministic batch jobs.
- Try to make batches small to allow low latency.
- Keep intermediate state data in cluster memory to further reduce latency - resilient distributed datasets(RDD).

Discretized Stream Processing

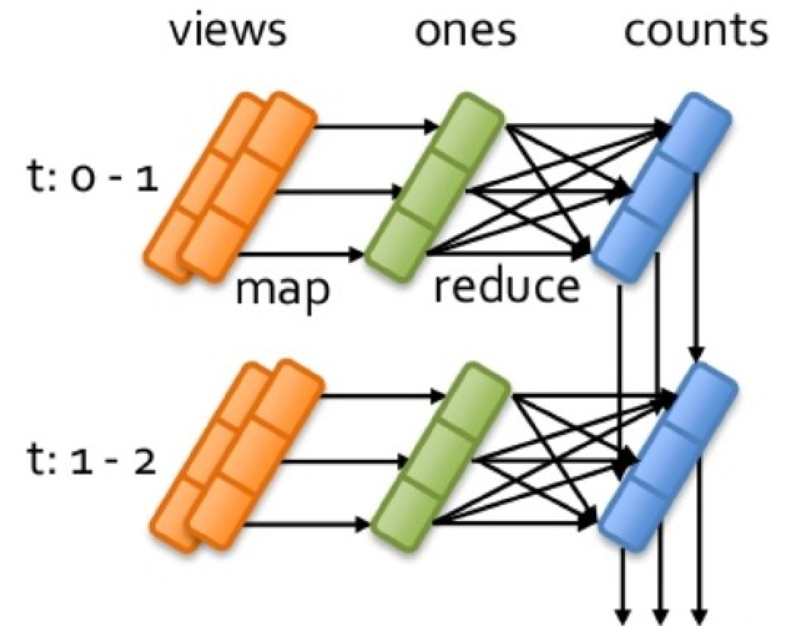


Page view count

creating a DStream

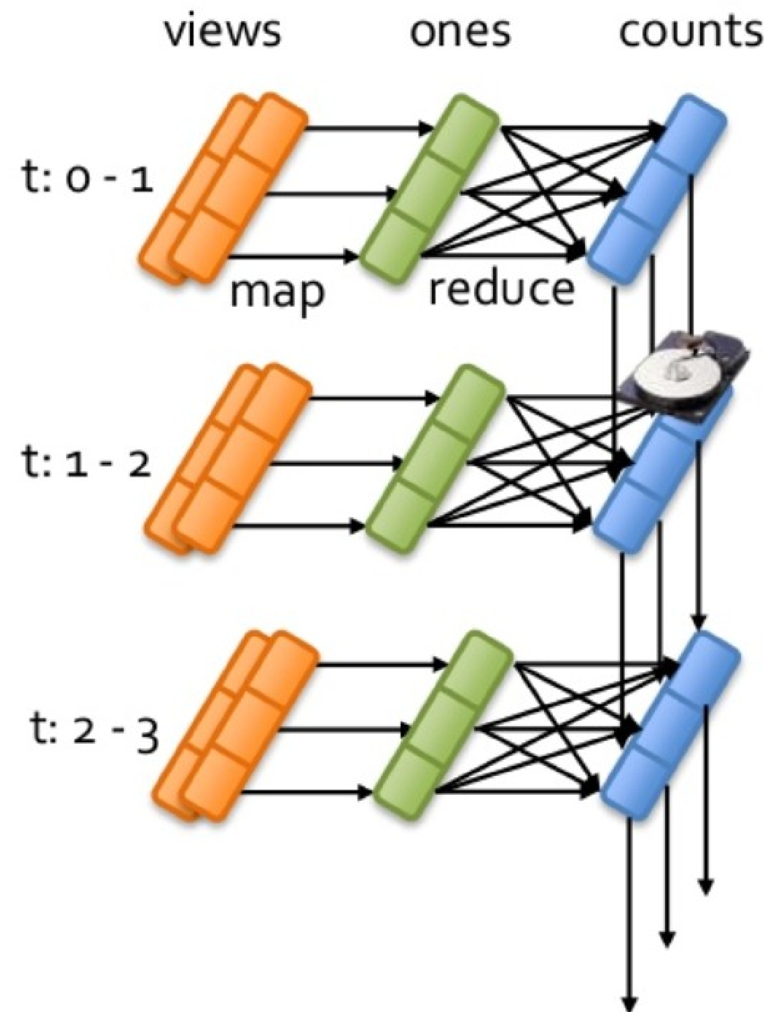
```
views = readStream("http:...", "1 sec")

ones = views.map(ev => (ev.url, 1))

counts = ones.runningReduce((x,y) => x+y)
```

transformation



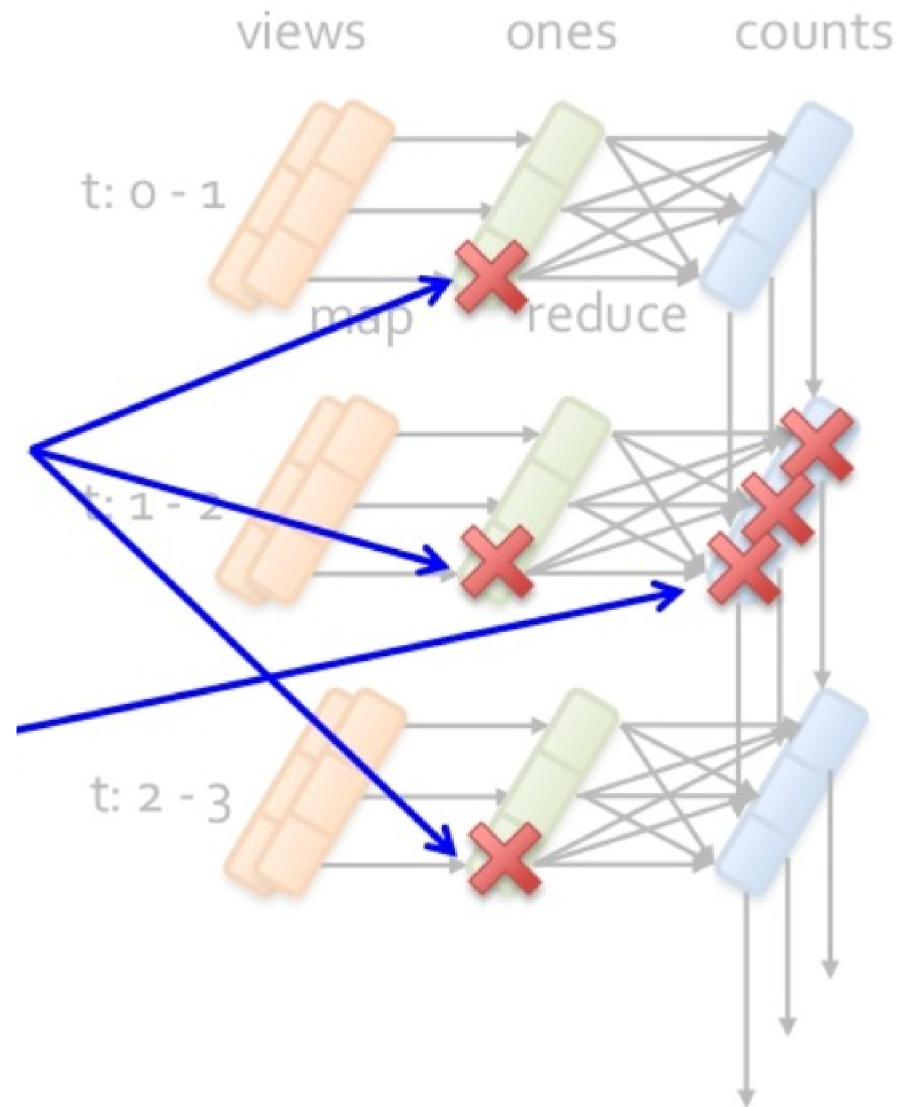
Lineage graph

- Lineage - a set of tasks used to build certain data.
- DStreams and RDD's track their lineage.
- When node fails or slows down lineage allows us to recompute lost data by re-running tasks used to build them.
- Data is being periodically checkpointed to prevent long recomputations and lineages.

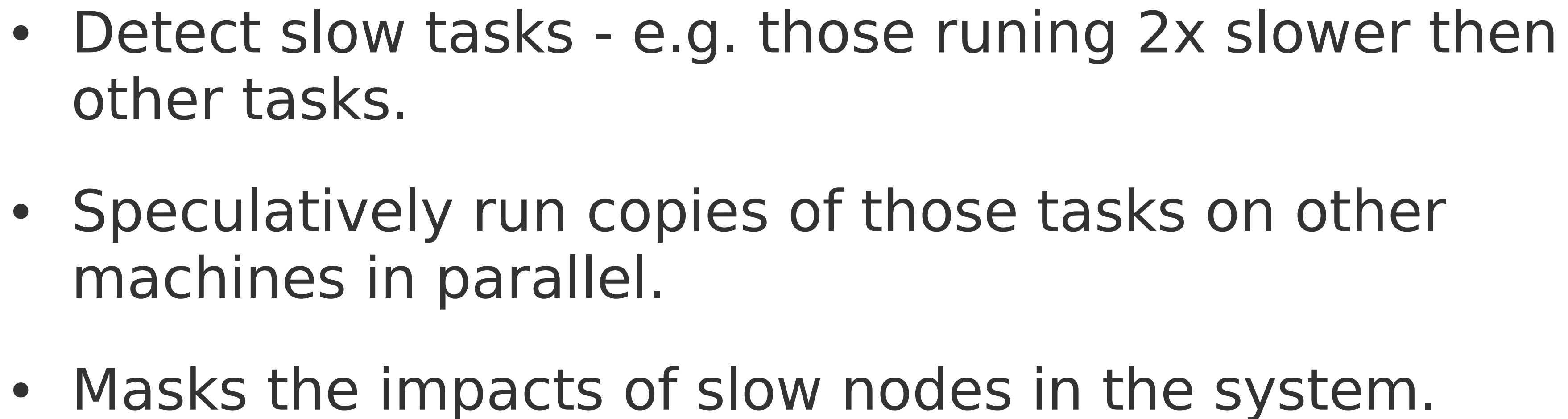


Parallel Recovery

- Data from different timesteps to be recomputed in parallel.
- Partitions within datasets can be recomputed in parallel.

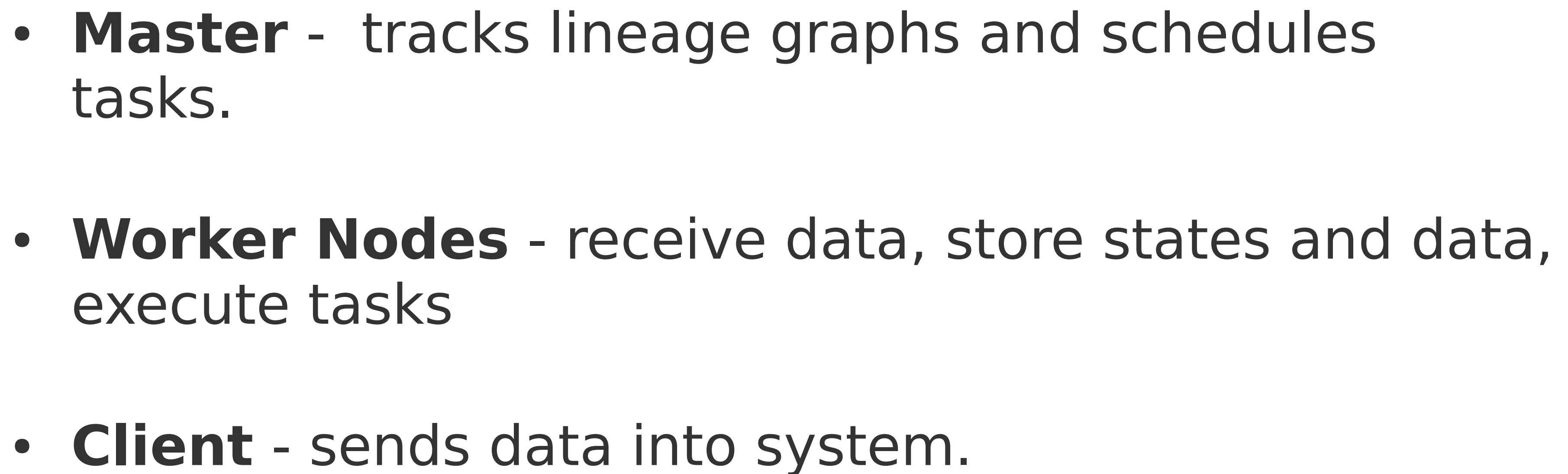


Straggler Recovery

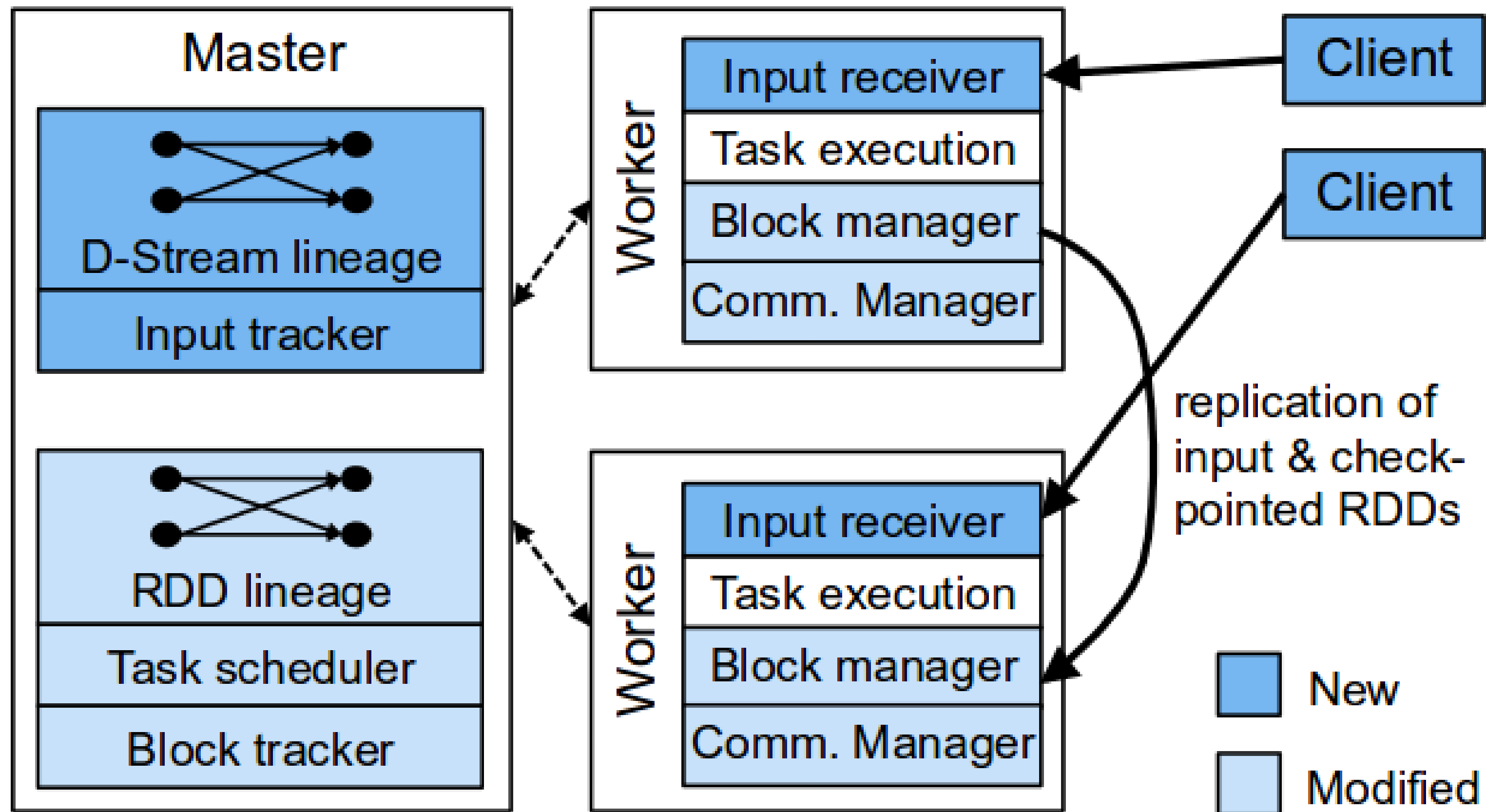


- Detect slow tasks - e.g. those runing 2x slower then other tasks.
- Speculatively run copies of those tasks on other machines in parallel.
- Masks the impacts of slow nodes in the system.

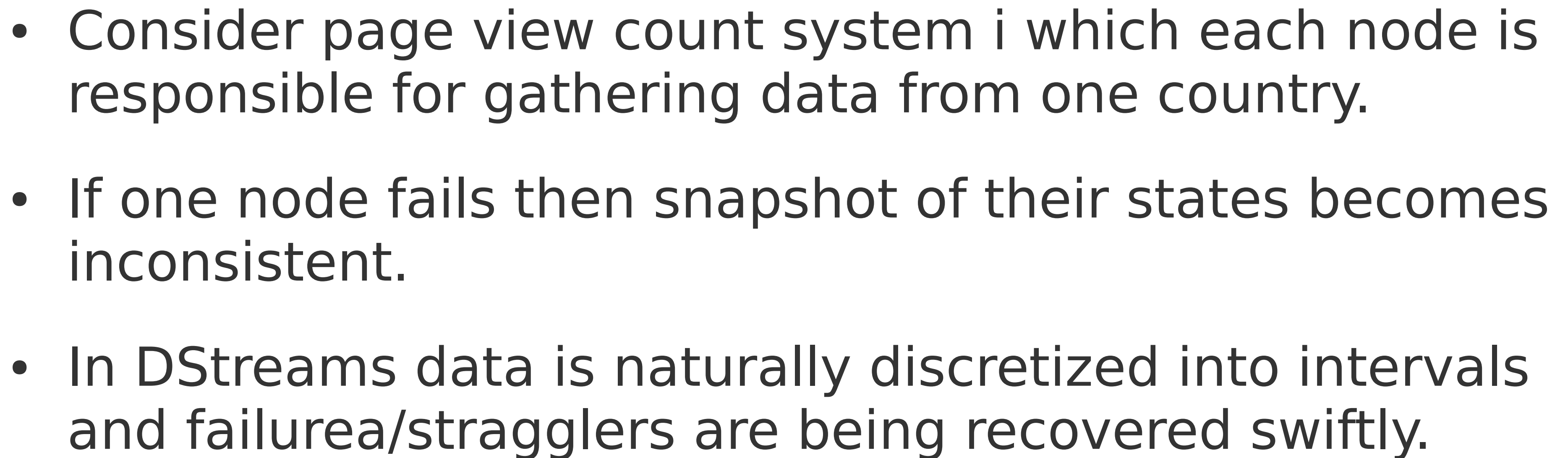
Spark Implementation



Spark Implementation



Consistency



- Consider page view count system i which each node is responsible for gathering data from one country.
- If one node fails then snapshot of their states becomes inconsistent.
- In DStreams data is naturally discretized into intervals and failurea/stragglers are being recovered swiftly.

Late Records

- In DStreams record is placed in batch when it arrives at the system.
- Data can be sorted by external timestamp.
- System can wait before processing each batch for late records.
- We can recompute old interval in future as if the node has failed.
- Also we can use incremental reduce operations.

Summary

- Latency - 0.5 - 2s.
- Consitency - Records processed atomically with interval they arrive.
- Late records - Slack time or app-level correction.
- Fault recovery - Fast parallel recovery.
- Straggler recovery - speculative execution.

Thank You