SALSA: Analyzing Logs as StAte Machines

Jiaqi Tan

Xinghao Pan, Soila Kavulya,

Rajeev Gandhi, Priya Narasimhan

PARALLEL DATA LABORATORY

Carnegie Mellon University

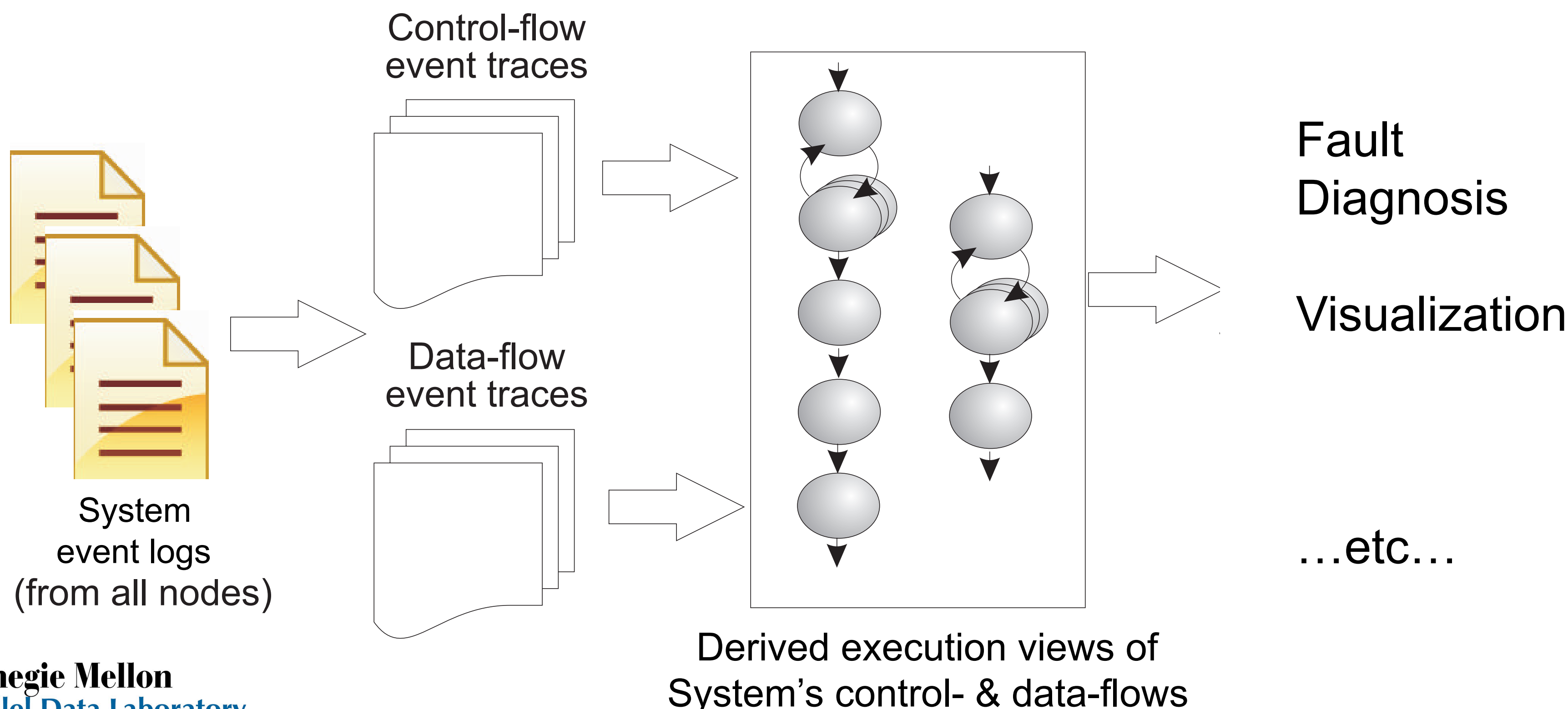
Carnegie Mellon
Parallel Data Laboratory

Motivation

- Ubiquitous, rich logs
  - Many software systems generate logs
  - Semantically-rich data, but difficult to analyze: Typically unstructured text
- Current techniques
  - Study statistical properties of log events
  - Adds little application semantics to analysis
- Want to incorporate semantics in analysis

SALSA

- General technique to extract execution views from system event logs
- Execution structure: *a priori* knowledge



Outline

- SALSA: Log Analysis Technique
- Applying SALSA: Analyzing Hadoop Logs
- Use-case 1: Visualization
- Use-case 2: Diagnosis

SALSA: State Extraction (1)

- Technique to extract state-machine views from logs
- Assumptions :
  - System has multiple concurrent, activities
  - Each activity emits log messages
  - Consider tasks as producers and consumers
- Extracts control-flow and data-flow views of execution
- Sample idealized log:

```
[ t1 ] Begin Task P1
[ t2 ] Begin Task C1
[ t3 ] Task P1 does some work
[ t4 ] Task C1 waits for data from
P1 and P2
[ t5 ] Task P1 produces data
[ t6 ] Task C1 consumes data from
P1 on host X
[ t7 ] Task P1 ends
[ t8 ] Task C1 consumes data from
P2 on host Y
[ t9 ] Task C1 ends
```

SALSA: State Extraction (2)

- Model control-flow: states extracted from log messages
  - Interpret each log message as an event
  - Impose semantics: Interpret events as starts/ends of execution states

```
[ t1 ] Begin Task P1
[ t2 ] Begin Task C1
[ t3 ] Task P1 does some work
[ t4 ] Task C1 waits for data from P1 and P2
[ t5 ] Task P1 produces data
[ t6 ] Task C1 consumes data from P1 on host X
[ t7 ] Task P1 ends
[ t8 ] Task C1 consumes data from P2 on host Y
[ t9 ] Task C1 ends
```

Carnegie Mellon
Parallel Data Laboratory

SALSA: State Extraction (3)

- Model data-flow:
  - Extract producer-consumer interactions e.g. when log messages indicate source for consumer
  - Can extract inter-node data-flows

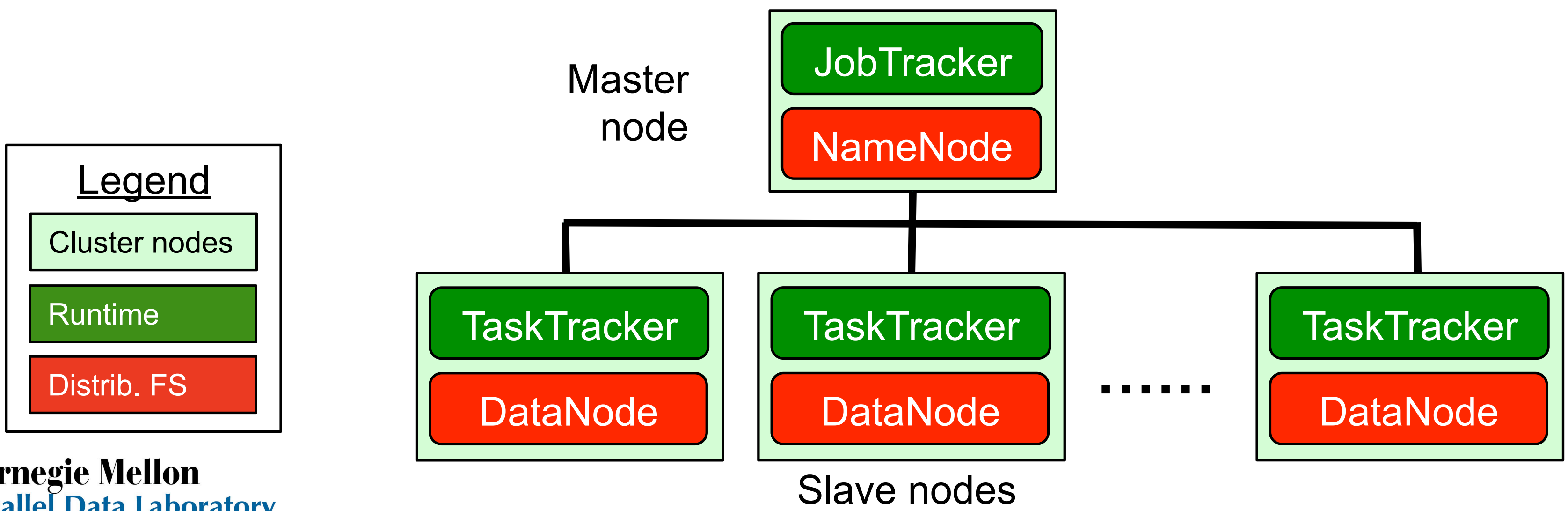
```
[ t1 ] Begin Task P1
[ t2 ] Begin Task C1
[ t3 ] Task P1 does some work
[ t4 ] Task C1 waits for data from
P1 and P2
[ t5 ] Task P1 produces data
[ t6 ] Task C1 consumes data from
P1 on host X
[ t7 ] Task P1 ends
[ t8 ] Task C1 consumes data from
P2 on host Y
[ t9 ] Task C1 ends
```

Outline

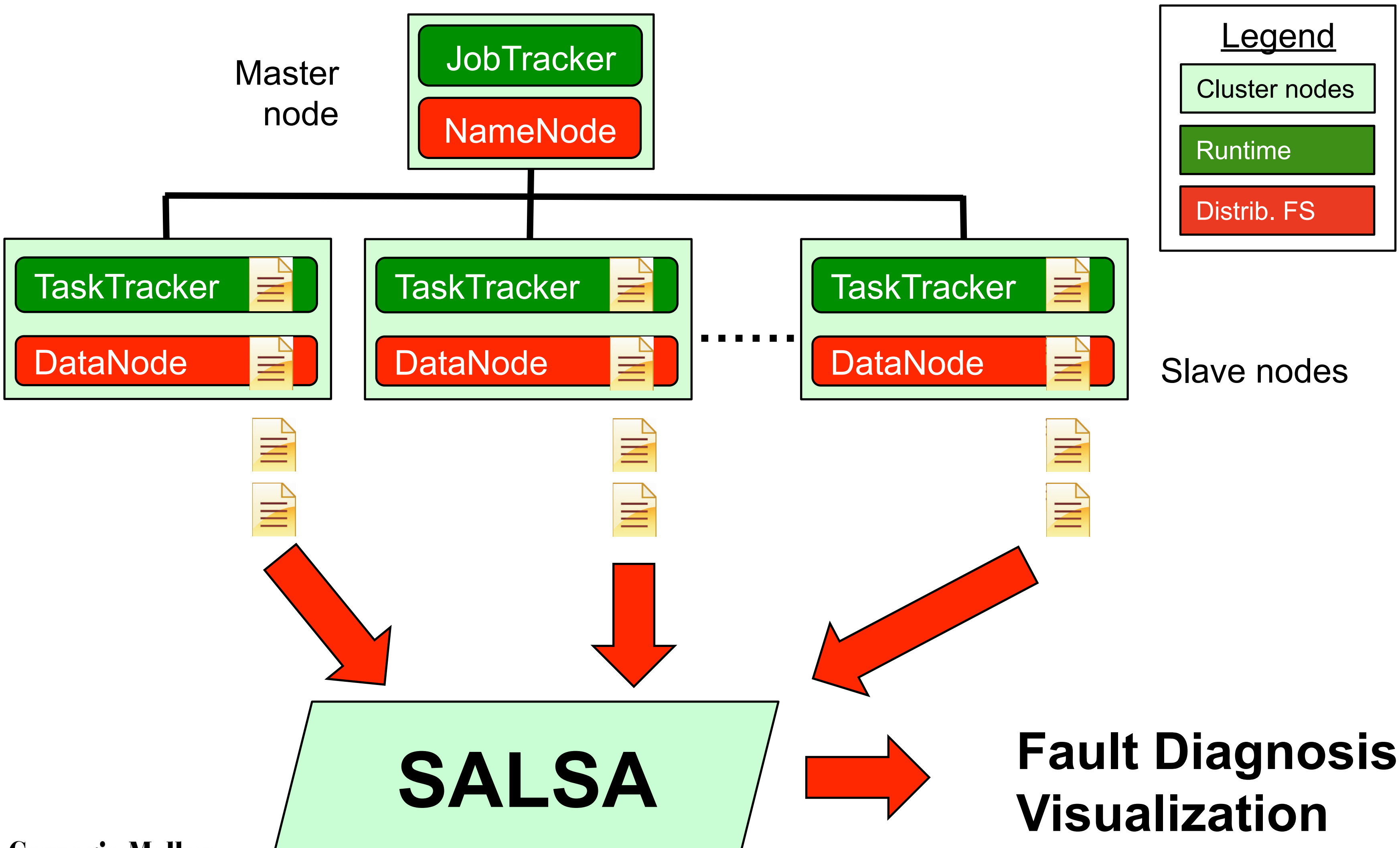
- SALSA: Log Analysis Technique
- Applying SALSA: Analyzing Hadoop Logs
- Use-case 1: Visualization
- Use-case 2: Diagnosis

Architecture of Hadoop

- MapReduce Runtime + Distributed Filesystem
- Master/Slave architecture
- Focus on slave node logs:
  - One log for each TaskTracker and DataNode
- Logs record processing activities



SALSA for Hadoop



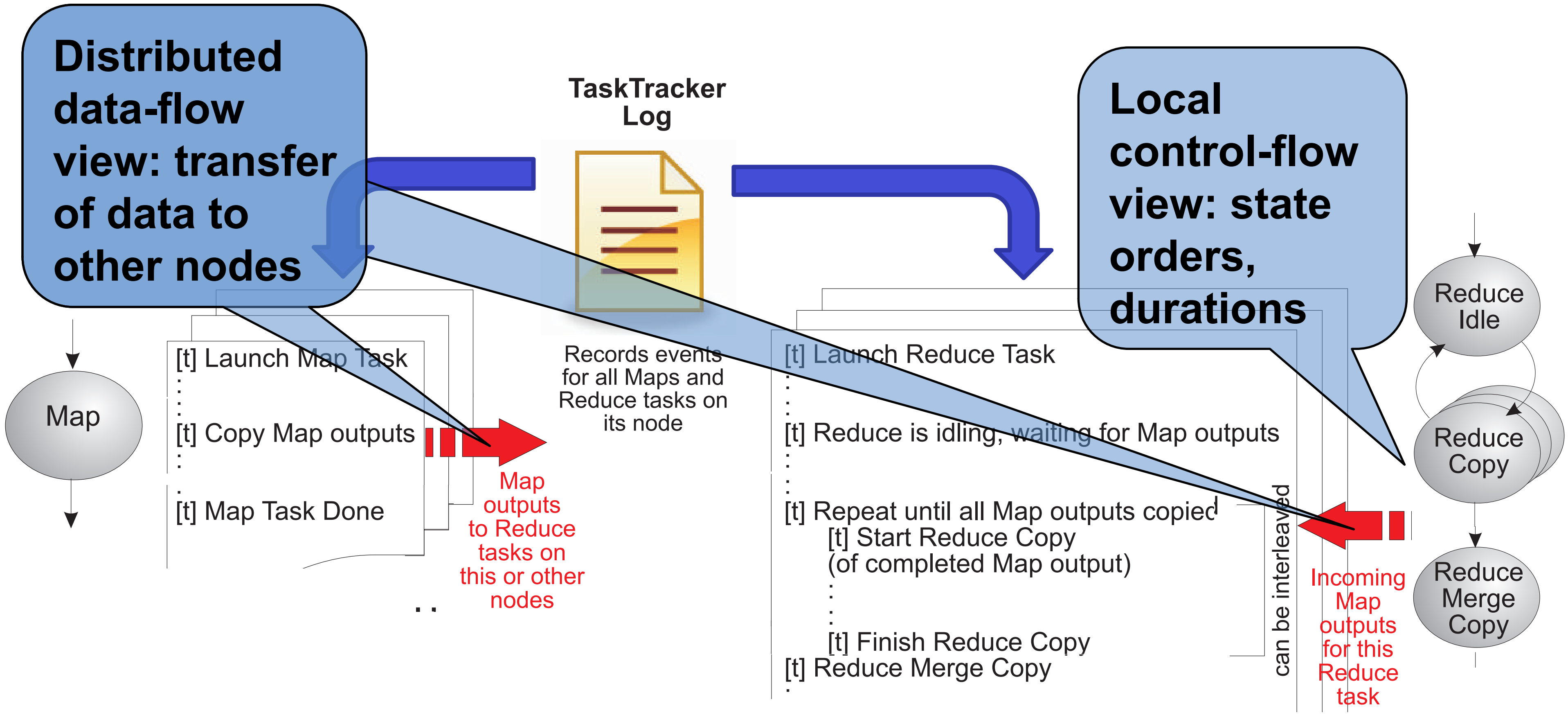
Hadoop-specific Semantics

- Identify events from log messages
  - Match tokens in message
- Map events to starts/ends of execution states
  - Using mapping from *a priori* knowledge, e.g.:

Activity/State	Start Token	End Token
Map	LaunchTaskAction [MapID]	Task [MapId] is done.
ReduceCopy	[ReduceID] Copying [MapID] output from [Hostname]	[ReduceID] done copying [MapID] output from [Hostname].

- Inputs: Hadoop application semantics
  - Message token → event mapping
  - Event → State start/end mapping

Applying SALSA to Hadoop

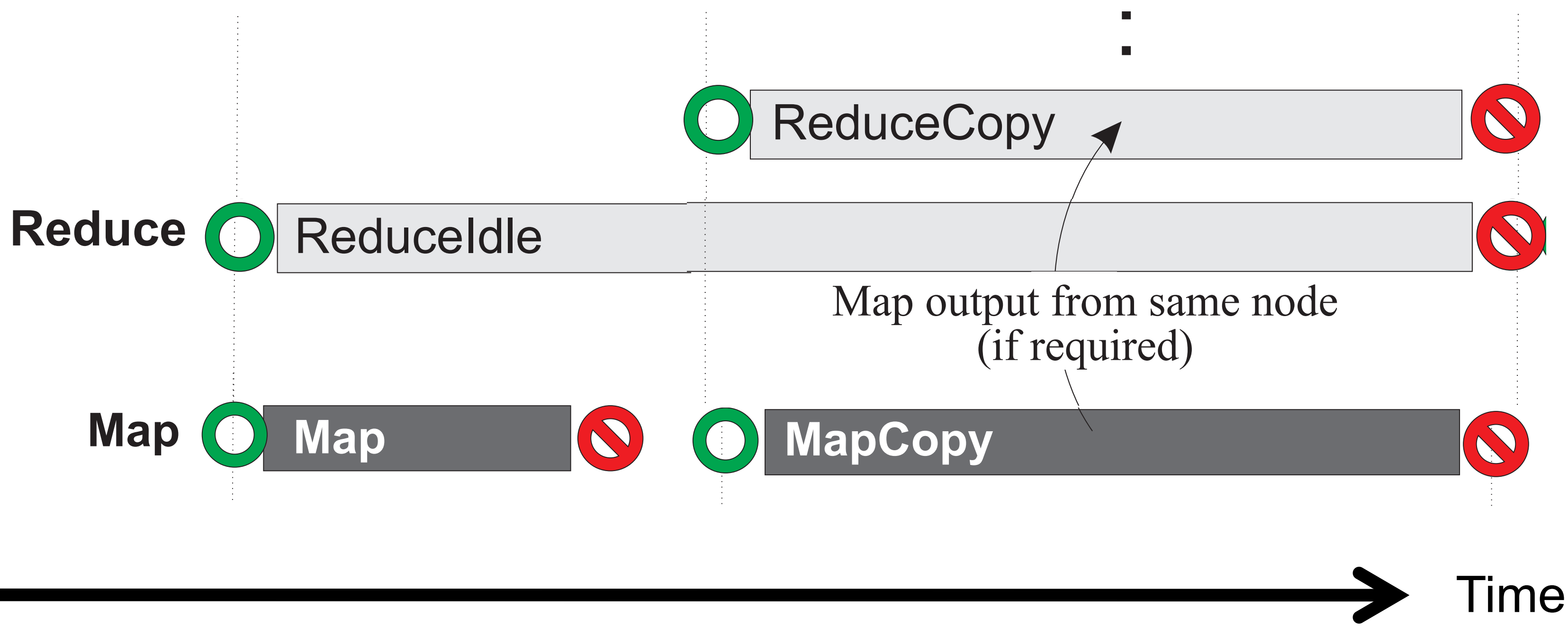


Outline

- SALSA: Log Analysis Technique
- Applying SALSA: Analyzing Hadoop Logs
- Use-case 1: Visualization
- Use-case 2: Diagnosis

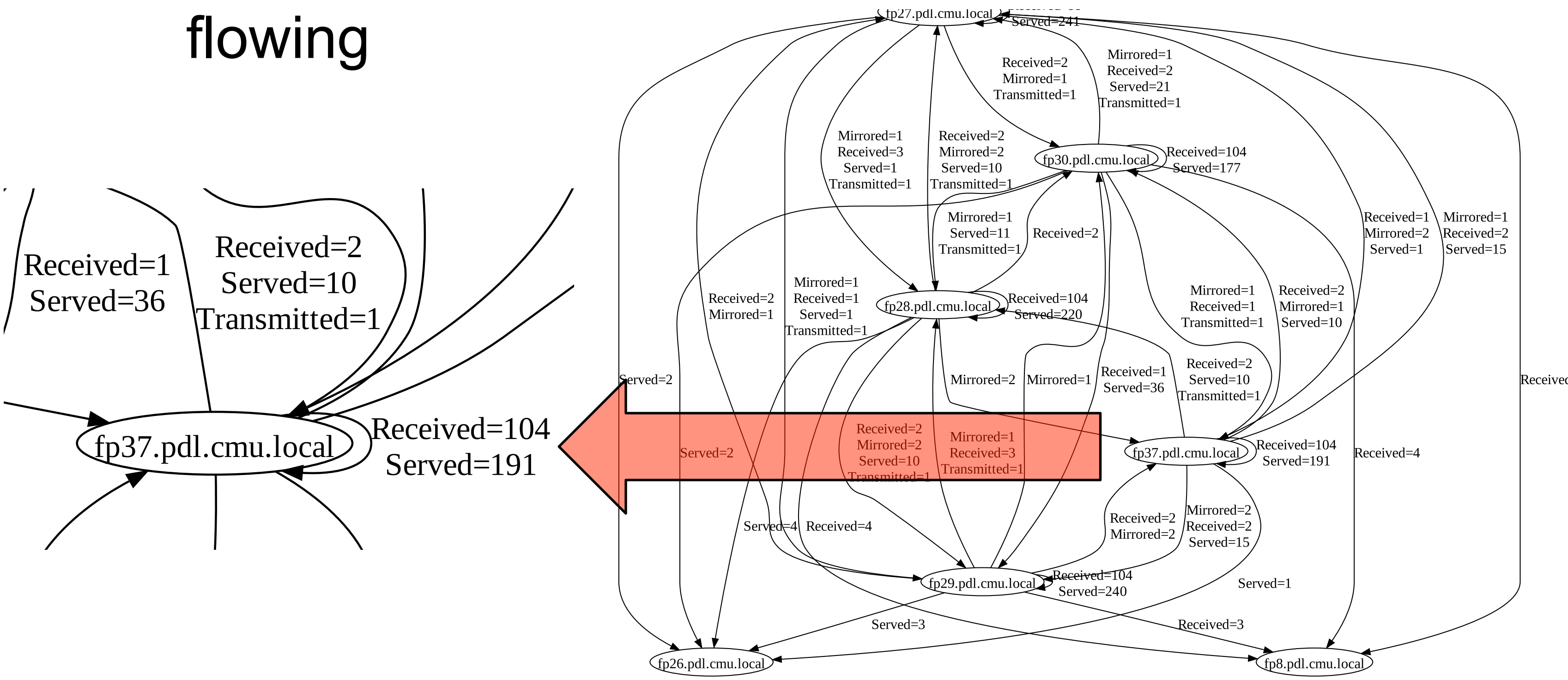
Control-flow Visualization

- Modeled execution of Hadoop TaskTracker: state machine view of logs
- Can be augmented to show data-flows



Data-flow Visualization

- Aggregate DataNode data-flows
  - Each node: one DataNode
  - Each edge: annotated with number of blocks flowing



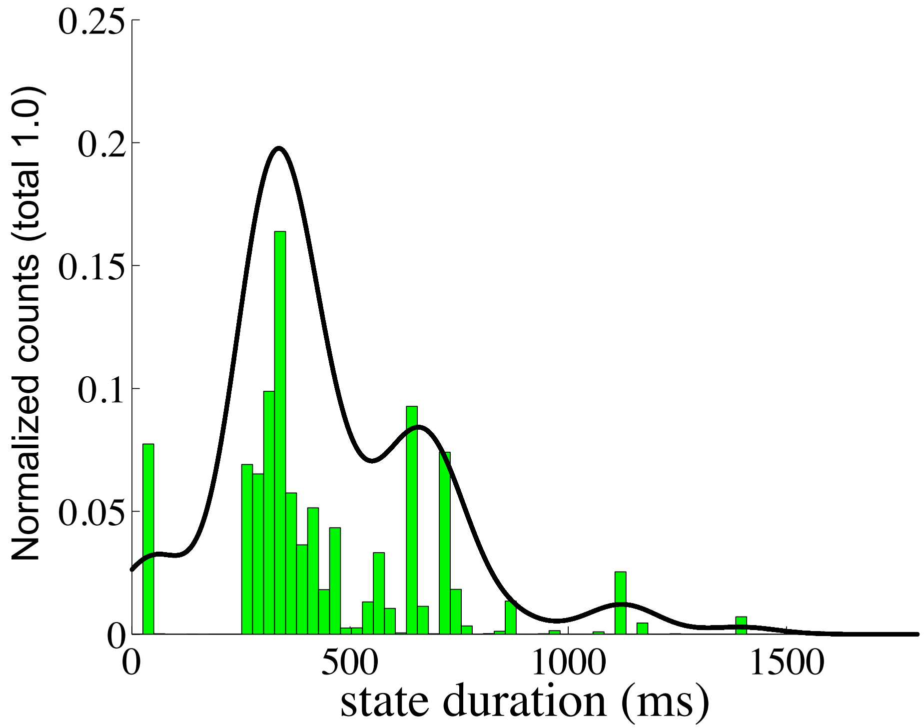
Carnegie Mellon
Parallel Data Laboratory

Outline

- SALSA: Log Analysis Technique
- Applying SALSA: Analyzing Hadoop Logs
- Use-case 1: Visualization
- Use-case 2: Diagnosis

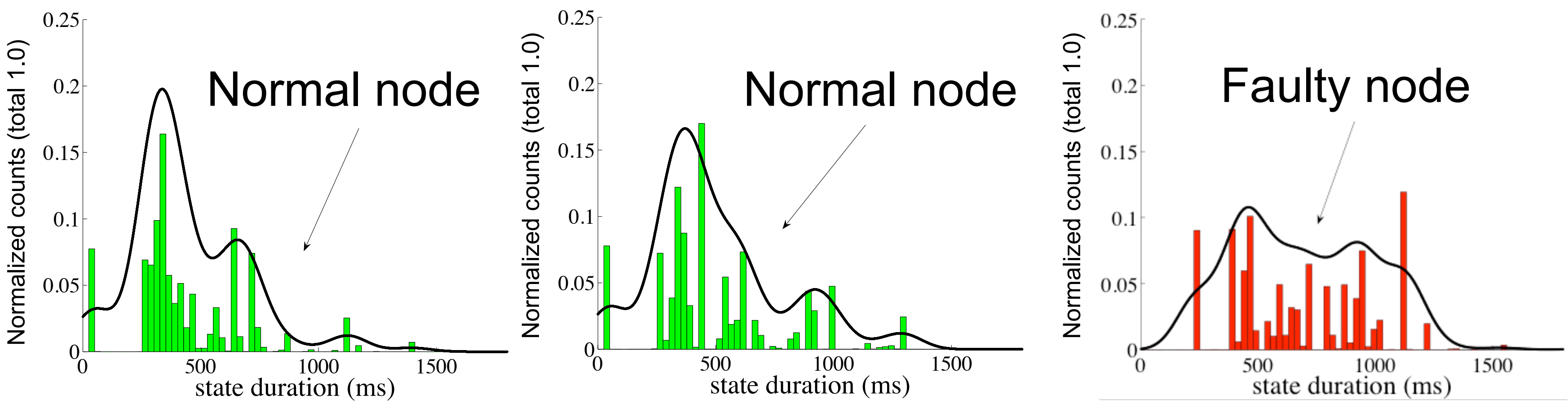
Durations of States

- Expressing aggregate control-flow: Build a histogram of state-durations for each node



Histogram (distributions) of durations of the `WriteBlock` state over a 30-second window

Intuition: Peer Comparison



Histograms (distributions) of durations of the `WriteBlock` state over a 30-second window

- In fault-free conditions, durations of a state are similar across nodes
- Faulty nodes: durations of the state are different than non-faulty nodes

Carnegie Mellon
Parallel Data Laboratory

Diagnosis Algorithm

- Diagnosis metrics: State durations of:
  - TaskTracker: `Map`, `ReduceMergeCopy`
  - DataNode: `ReadBlock`, `WriteBlock`
- Build a histogram of durations per node
  - One histogram in every window
- Compare histograms across nodes
  - Compute statistical measure of distance (Jensen-Shannon Divergence) between histograms
  - Indict nodes whose histograms have distances greater than threshold to more than half the other nodes

Experimentation

- Hadoop 0.12.3 cluster
  - 5-slaves, 1-master, identically configured nodes
- Workloads:
  - RandWriter: writes 32GB random data to disk
  - Sort: sorts 3GB of records of random data
  - Nutch: distributed web crawler for Hadoop
- Data Collection
  - Hadoop logs harvested and processed offline
- Faults Injected: External resource hogs
  - Disk Hog, CPU Hog

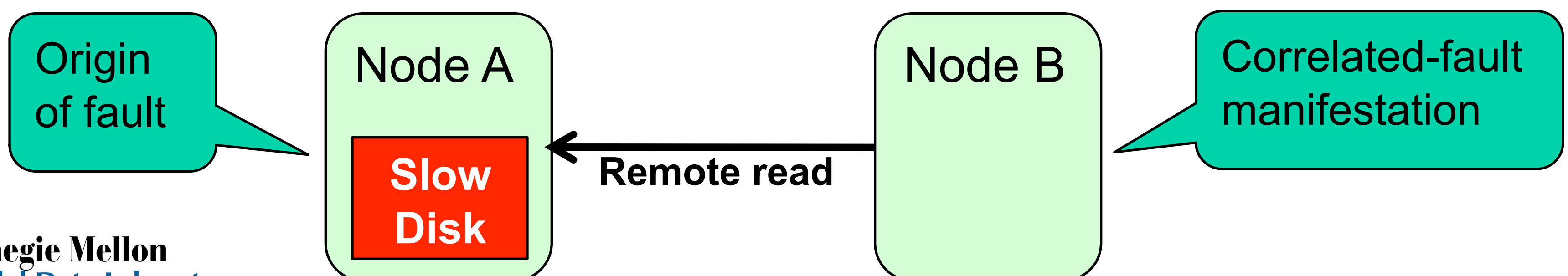
Selected Diagnosis Results

- Different states effective at diagnosing different classes of faults
  - Dependent on type of activity of state
- `Map` state: effective for CPU and Disk Hogs
  - (TP > 0.8, FP < 0.25)
- `ReduceMergeCopy`: primarily disk activity
  - Disk Hog: TP = 1.0, FP < 0.05
    - Less effective on CPU-related issues
  - CPU Hog: TP > 0.3, FP < 0.15

TP = True-positive rate [0.0,1.0]
FP = False-positive rate [0.0,1.0]
Perfect diagnosis: TP=1.0, FP=0.0

Correlated Fault-Manifestations

- Correlated fault-manifestations
  - Fault originates on one node
  - Other non-faulty nodes exhibit manifestations
  - Symmetric behavioral change: need technique other than peer-comparison
- Example: Disk Hog injected on one node
  - Reads from affected node slowed down
  - Show up as slower reads on other nodes

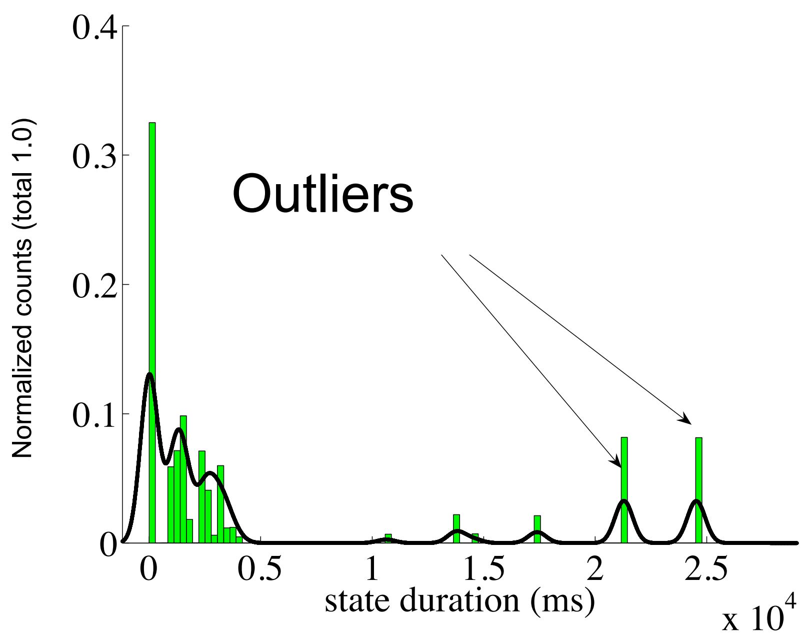


Data-flow Augmented Diagnosis

- Key idea:
  - Non-faulty nodes must somehow interact with faulty node for fault-manifestation to spread
- SALSA extracts data-flow from logs
- Localizing correlated fault-manifestations
  - Compare node's histogram of phase durations with its own past histograms
  - Identify outliers in current histogram
  - If majority of outlier phases associated with single node, indict that node

Historical Comparison

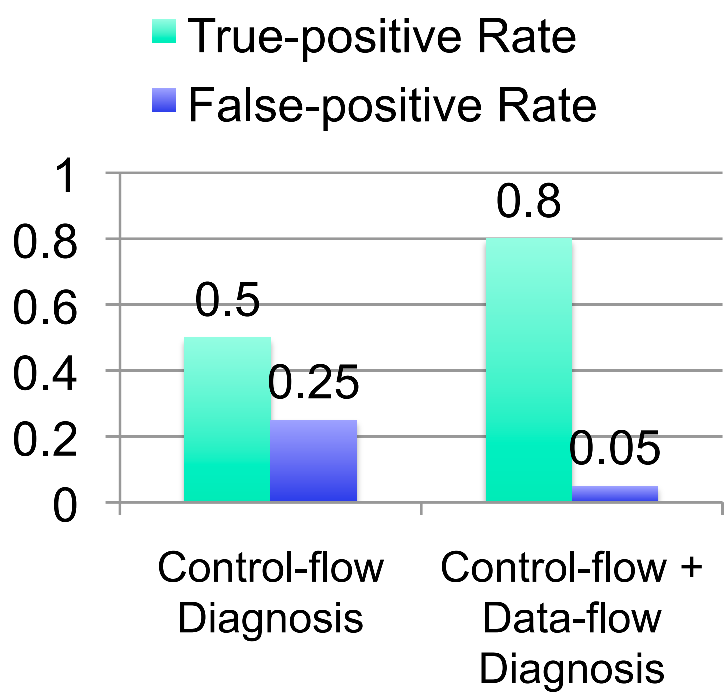
- Detect outlier phases as compared to past
- Indict node associated with many outliers



Histograms (distributions) of durations of the `ReadBlock` phase over a 30-second window

Correlated Fault-Manifestation: Results

- Improved Disk Hog diagnosis using `ReadBlock` on Sort workload



Summary

- SALSA: Log Analysis Technique
  - Extract state-machine views of execution
  - Augment analysis with application semantics
  - Provides control-flow and data-flow views
- Applying SALSA: Analyzing Hadoop Logs
- Use-case 1: Visualization
- Use-case 2: Diagnosis
  - Peer comparison of state durations for Hadoop
  - Detected resource hogs
  - Detected source of correlated-fault manifestation

Reaching us

- Our Hadoop diagnosis efforts:
  - ASDF: Automated, Online Fingerpointing for Hadoop: CMU-PDL-08-104
  - Ganesha: Black-box Fault Diagnosis for MapReduce Environments: CMU-PDL-08-112, Also a poster at SysML '08: Come talk to us!
  - SALSA: Analyzing Logs as State Machines: Longer version as CMU-PDL-08-111
- My email: jiaqit at andrew dot cmu dot edu
- Our website: http://www.ece.cmu.edu/~fingerpointing