PROBLEM DIAGNOSIS FOR CLOUD COMPUTING

Jiaqi Tan, Soila Kavulya, Xinghao Pan, Mike Kasick, Keith Bare,
Eugene Marinelli, Rajeev Gandhi
Priya Narasimhan

Carnegie Mellon University

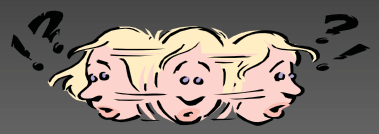
Automated Problem Diagnosis

- Diagnosing problems
  - Creates major headaches for administrators
  - Worsens as scale and system complexity grows
- Goal: automate it and get proactive
  - Failure detection and prediction
  - Problem determination (or "fingerpointing")
- How: Instrumentation plus statistical analysis

Challenges in Problem Analysis

- Challenging in large-scale networked environment
  - Can have multiple failure manifestations with a single root cause
  - Can have multiple root causes for a single failure manifestation
  - Problems and/or their manifestations can "travel" among communicating components
  - A lot of information from multiple sources – what to use? what to discard?

- Automated fingerpointing
  - Automatically discover faulty node in a distributed system



Exploration

- Current explorations
  - *Hadoop*
    - Open-source implementation of Map/Reduce (Yahoo!), popular cloud-computing platform
  - *PVFS*
    - High-performance file system (Argonne National Labs)
  - *Lustre*
    - High-performance file system (Sun Microsystems)

- Studied
  - Various types of problems
  - Various kinds of instrumentation
  - Various kinds of data-analysis techniques

Why?

- Hadoop is fault-tolerant
  - Heartbeats: detect lost nodes
  - Speculative re-execution: recover work due to lost/ laggard nodes
- Hadoop's fault-tolerance can mask performance problems
  - Nodes alive but slow
- Target failures for our diagnosis
  - Performance degradations (slow, hangs)

BEFORE: Hadoop Web Console

- Admin/user sifts through wealth of information
  - Problem is aggravated in large clusters
  - Multiple clicks to chase down a problem
- No support for historical comparison
  - Information displayed is a snapshot in time
- Poor localization of correlated problems
  - Progress indicators for all tasks are skewed by correlated problem
- No clear indicators of performance problems
  - Are task slowdowns due to data skew or bugs? Unclear

AFTER: Goals, Non-Goals

- Diagnose faulty Master/Slave node to user/admin
- Target production environment
  - Don't instrument Hadoop or applications additionally
  - Use Hadoop logs as-is (*white-box strategy*)
  - Use OS-level metrics (*black-box strategy*)
- Work for various workloads and under workload changes
- Support online and offline diagnosis

- Non-goals (for now)
  - Tracing problem to offending line of code

Target Hadoop Clusters

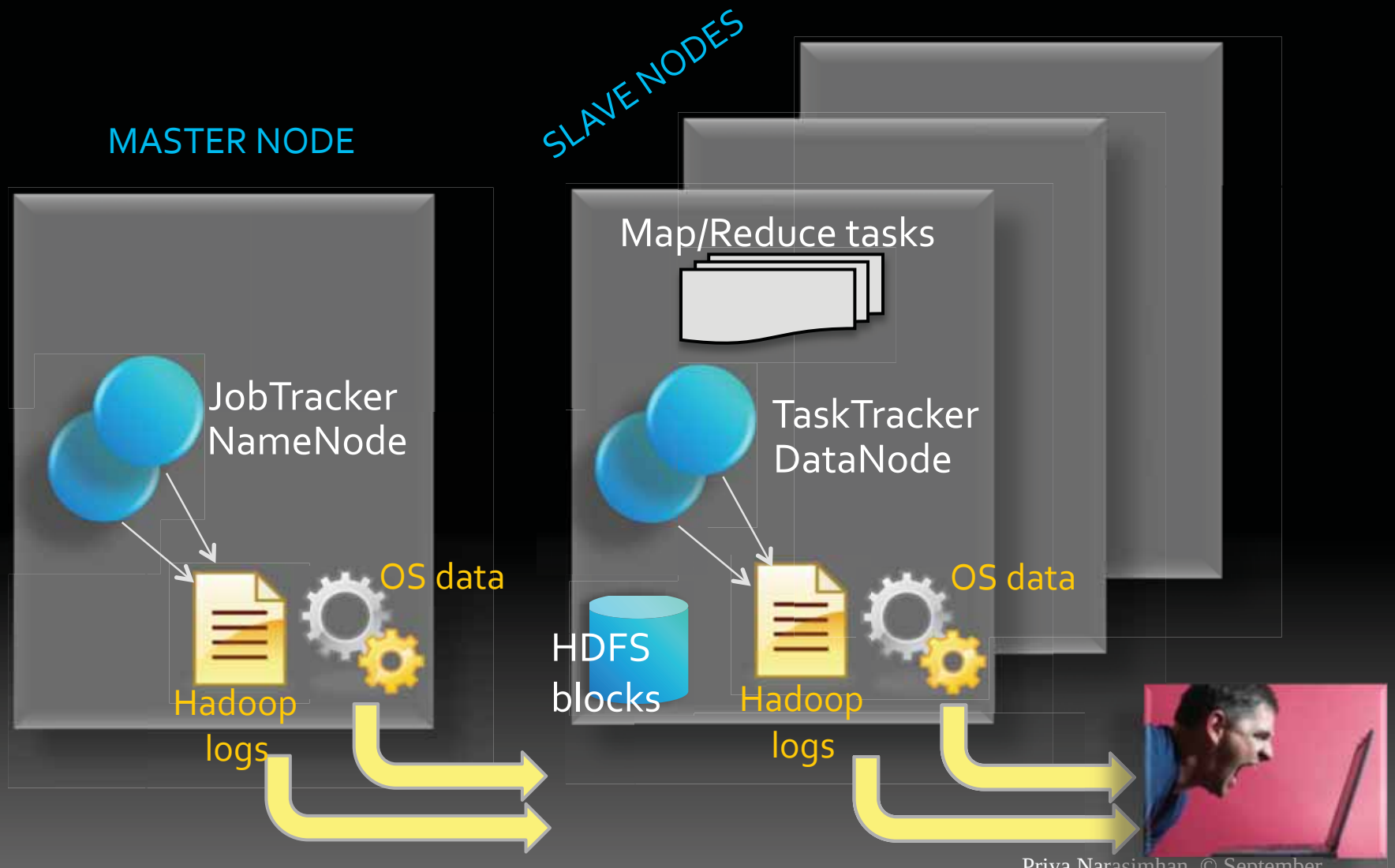
- 4000-processor Yahoo!'s M45 cluster
  - Production environment (managed by Yahoo!)
  - Offered to CMU as free cloud-computing resource
  - Diverse kinds of real workloads, problems in the wild
    - Massive machine-learning, language/machine-translation
  - Permission to harvest all logs and OS data each week
- 100-node Amazon's EC2 cluster
  - Production environment (managed by Amazon)
  - Commercial, pay-as-you-use cloud-computing resource
  - Workloads under our control, problems injected by us
    - gridmix, nutch, pig, sort, randwriter
  - Can harvest logs and OS data of only our workloads

Some Performance Problems Studied

	Fault	Description
Resource contention	CPU hog	External process uses 70% of CPU
	Packet-loss	5% or 50% of incoming packets dropped
	Disk hog	20GB file repeatedly written to
	Disk full	Disk full
Application bugs Source: Hadoop JIRA	HADOOP-1036	Maps hang due to unhandled exception
	HADOOP-1152	Reduces fail while copying map output
	HADOOP-2080	Reduces fail due to incorrect checksum
	HADOOP-2051	Jobs hang due to unhandled exception
	HADOOP-1255	Infinite loop at Nameode

Studied Hadoop Issue Tracker (JIRA) from Jan-Dec 2007

Hadoop: Instrumentation



How About Those Metrics?

- White-box metrics (from Hadoop logs)
  - Event-driven (based on Hadoop's activities)
  - *Durations*
    - Map-task durations, Reduce-task durations, ReduceCopy-durations, etc.
  - System-wide dependencies between tasks and data blocks
  - Heartbeat information: Heartbeat rates, Heartbeat-timestamp skew between the Master and Slave nodes
- Black-box metrics (from OS /proc)
  - 64 different time-driven metrics (sampled every second)
  - Memory used, context-switch rate, User-CPU usage, System-CPU usage, I/O wait time, run-queue size, number of bytes transmitted, number of bytes received, pages in, pages out, page faults

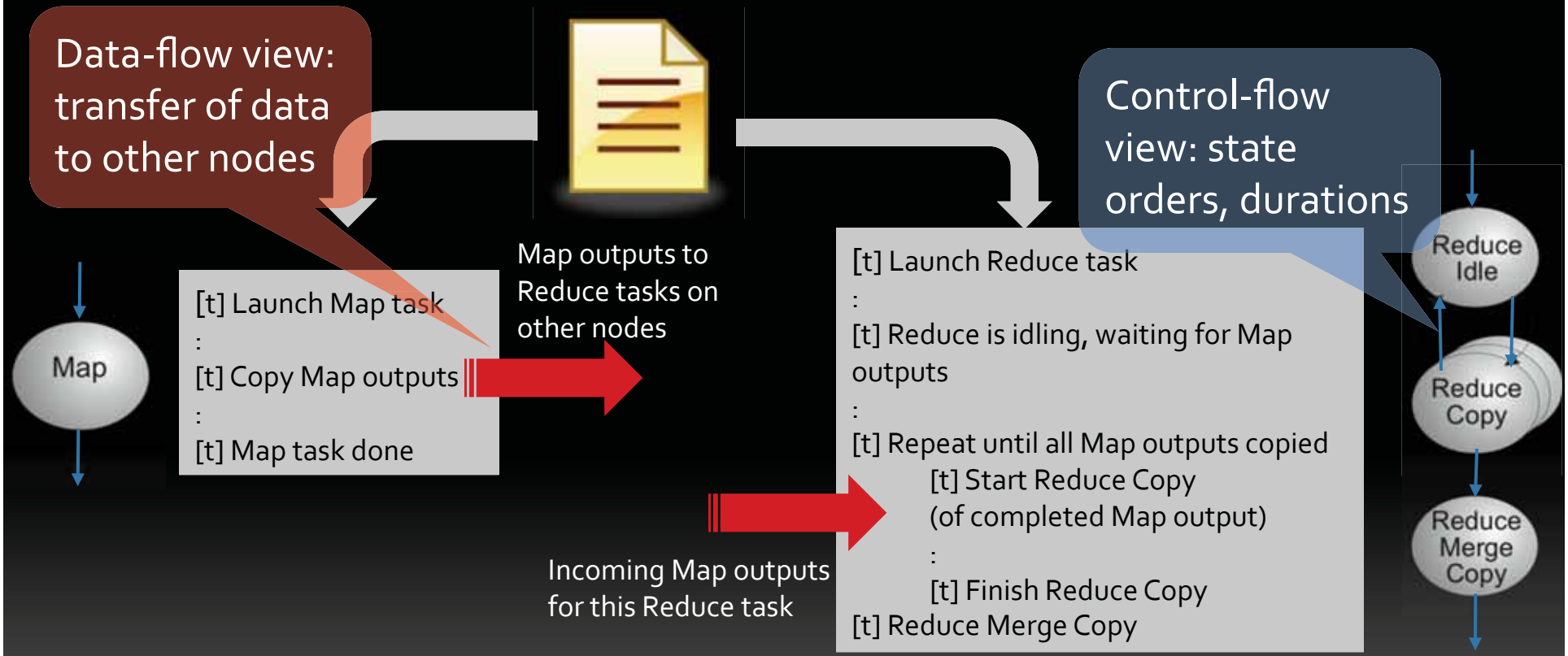
Intuition for Diagnosis

- Slave nodes are doing approximately similar things for a given job
- Use some form of "peer comparison"
  - But, peer-compare *what*?
- Gather metrics and extract statistics
  - Determine metrics of relevance
  - For both black-box and white-box data
- Peer-compare histograms, means, etc. to determine "odd-man out"

Log-Analysis Approach

- SALSA: Analyzing Logs as StAte Machines [*USENIX WASL 2008*]
- Extract state-machine views of execution from Hadoop logs
  - Distributed control-flow view of logs
  - Distributed data-flow view of logs
- Diagnose failures based on statistics of these extracted views
  - Control-flow based diagnosis
  - Control-flow + data-flow based diagnosis
- Perform analysis incrementally so that we can support it online

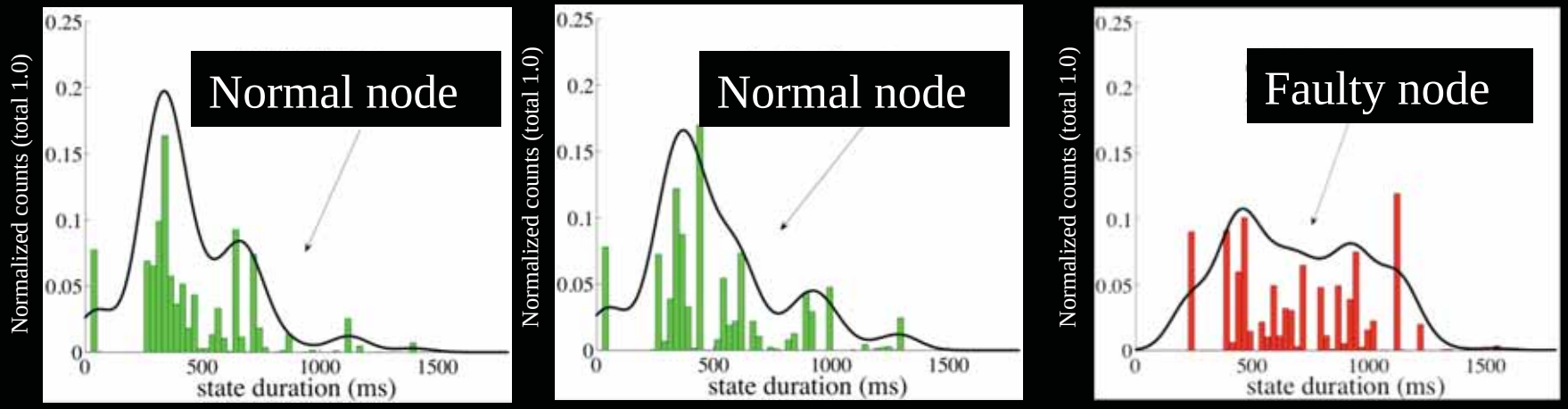
Applying SALSA to Hadoop



Distributed Control+Data Flow

- Distributed control-flow
  - Causal flow of task execution across cluster nodes, i.e., Reduces waiting on Maps via Shuffles
- Distributed data-flow
  - Data paths of Map outputs shuffled to Reduces
  - HDFS data blocks read into and written out of jobs
- Job-centric data-flows: Fused Control+Data Flows
  - Correlate paths of data and execution
  - Create conjoined causal paths from data source before, to data destination after, processing
  - Helps to trace correlated performance problems

Intuition: Peer Similarity

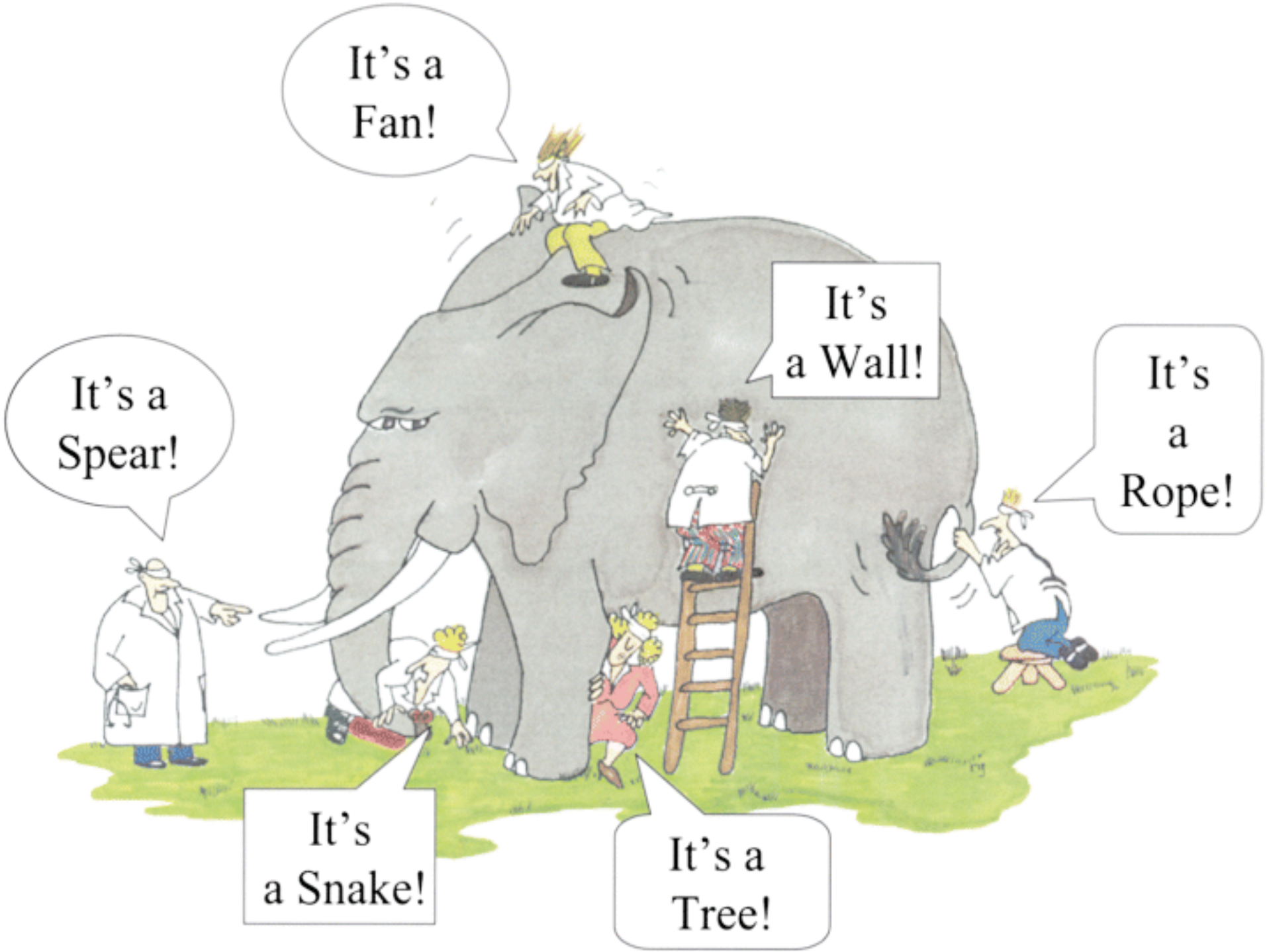


Histograms (distributions) of durations of `WriteBlock` over a 30-second window

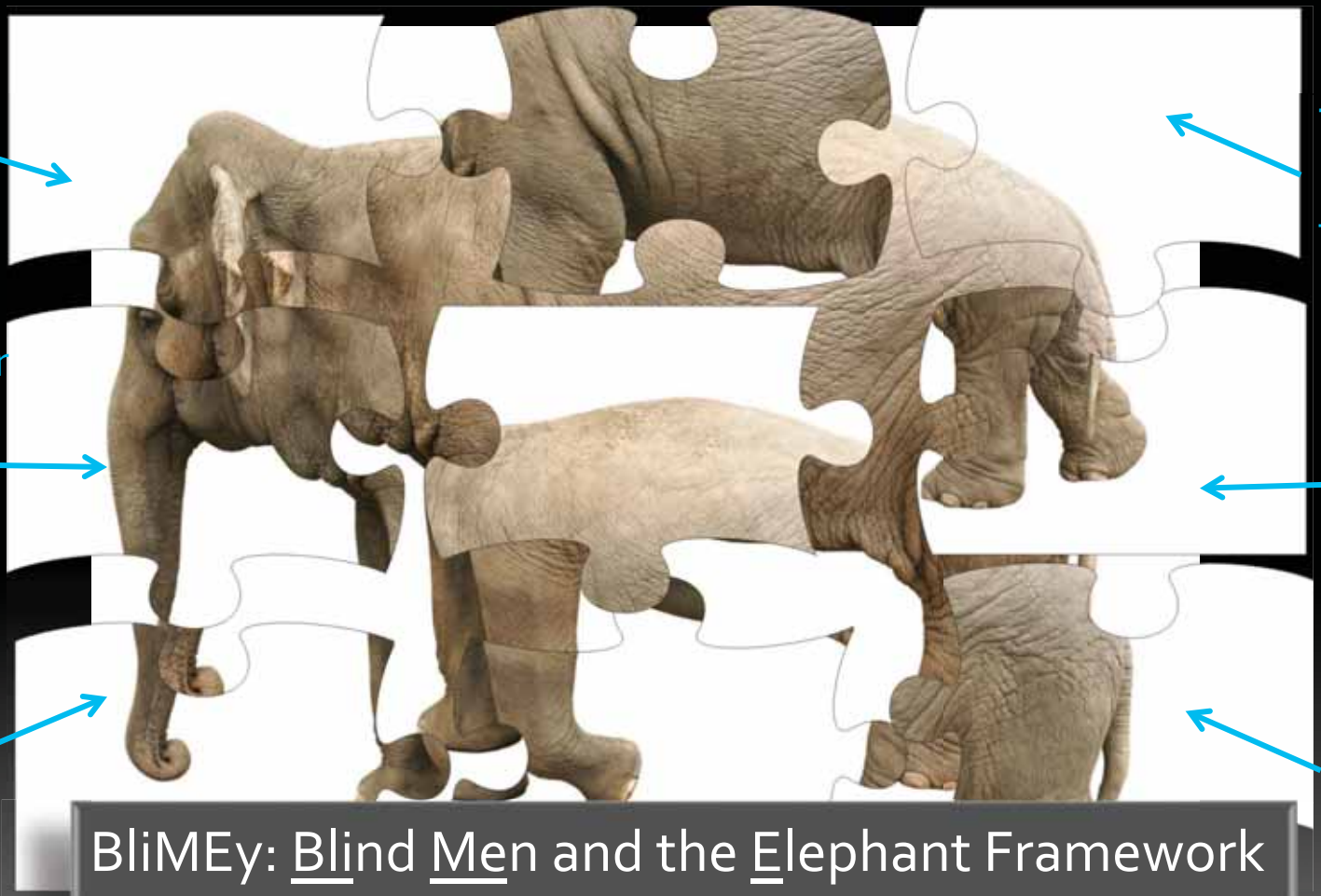
- In fault-free conditions, metrics (e.g., WriteBlock durations) are similar across nodes
- Faulty node: Same metric is different on faulty node, as compared to non-faulty nodes
- Kullback-Leibler divergence (comparison of histograms)

What Else Do We Do?

- Analyze black-box data with similar intuition
  - Derive PDFs and use a clustering approach
    - Distinct behavior profiles of metric correlations
  - Compare them across nodes
  - Technique called Ganesha [*HotMetrics 2009*]
- Analyze heartbeat traffic
  - Compare heartbeat durations across nodes
  - Compare heartbeat-timestamp skews across nodes
- Different metrics, different viewpoints, different algorithms


It's a Fan!
It's a Spear!
It's a Wall!
It's a Rope!
It's a Snake!
It's a Tree!

Putting the Elephant Together



JobTracker Durations views

TaskTracker Durations views

Job-centric data flows

TaskTracker heartbeat timestamps

JobTracker heartbeat timestamps

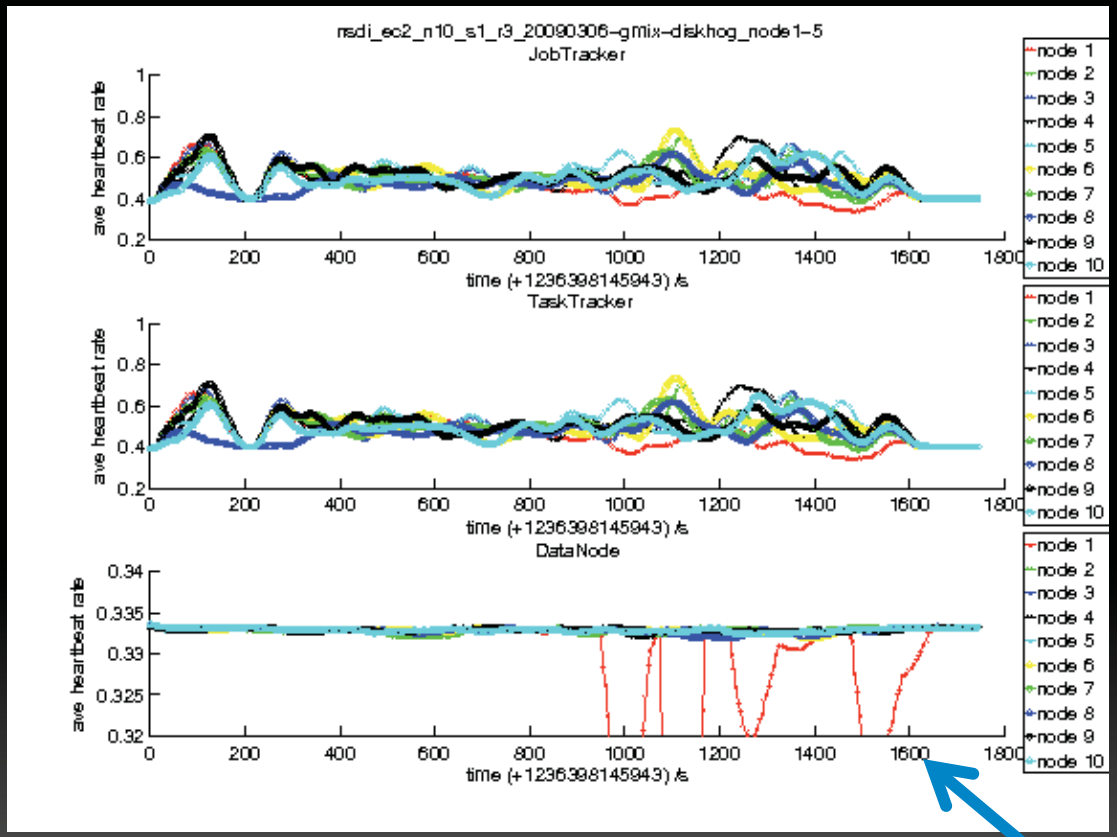
Black-box resource usage

BliMEy: Blind Men and the Elephant Framework
[*CMU-CS-09-135*]

Visualization

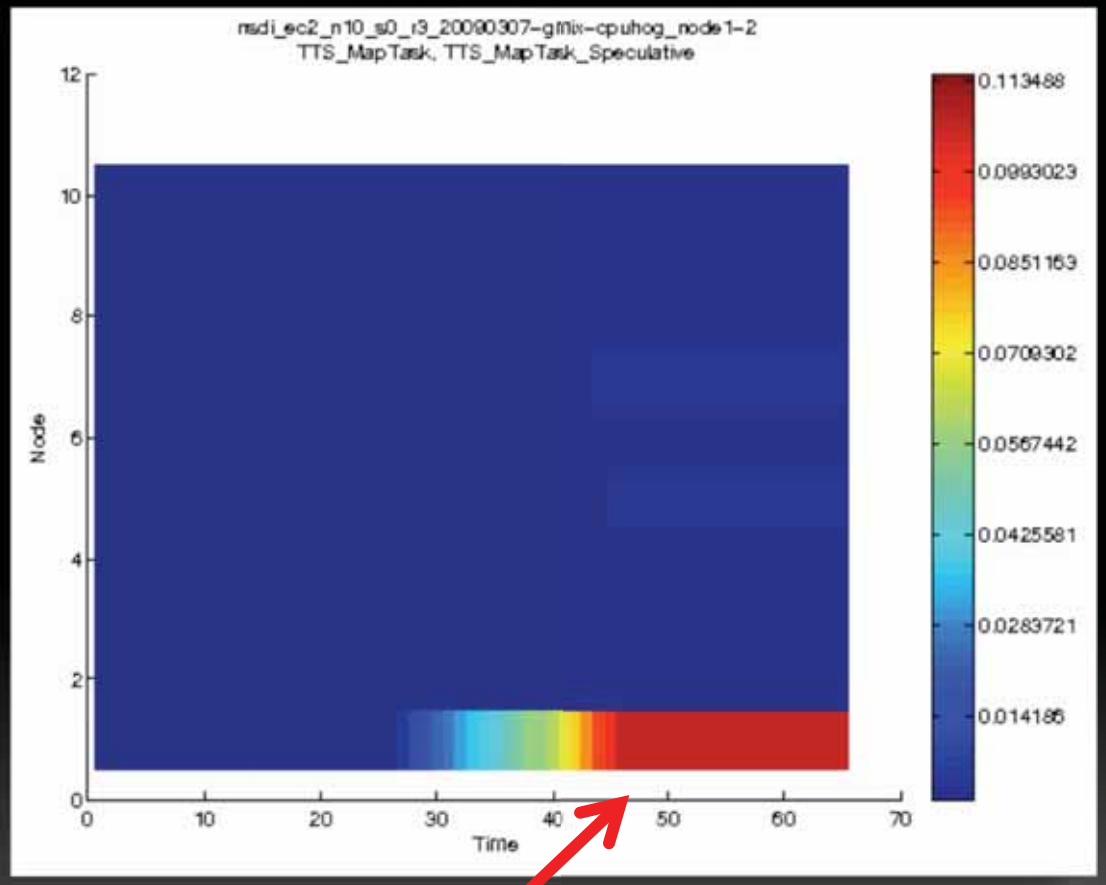
- To uncover Hadoop's execution in an insightful way
- To reveal outcome of diagnosis on sight
- To allow developers/admins to get a handle as the system scales
- Value to programmers [*HotCloud 2009*]
  - Allows them to spot issues that might assist them in restructuring their code
  - Allows them to spot faulty nodes

Visualization (*timeseries*)



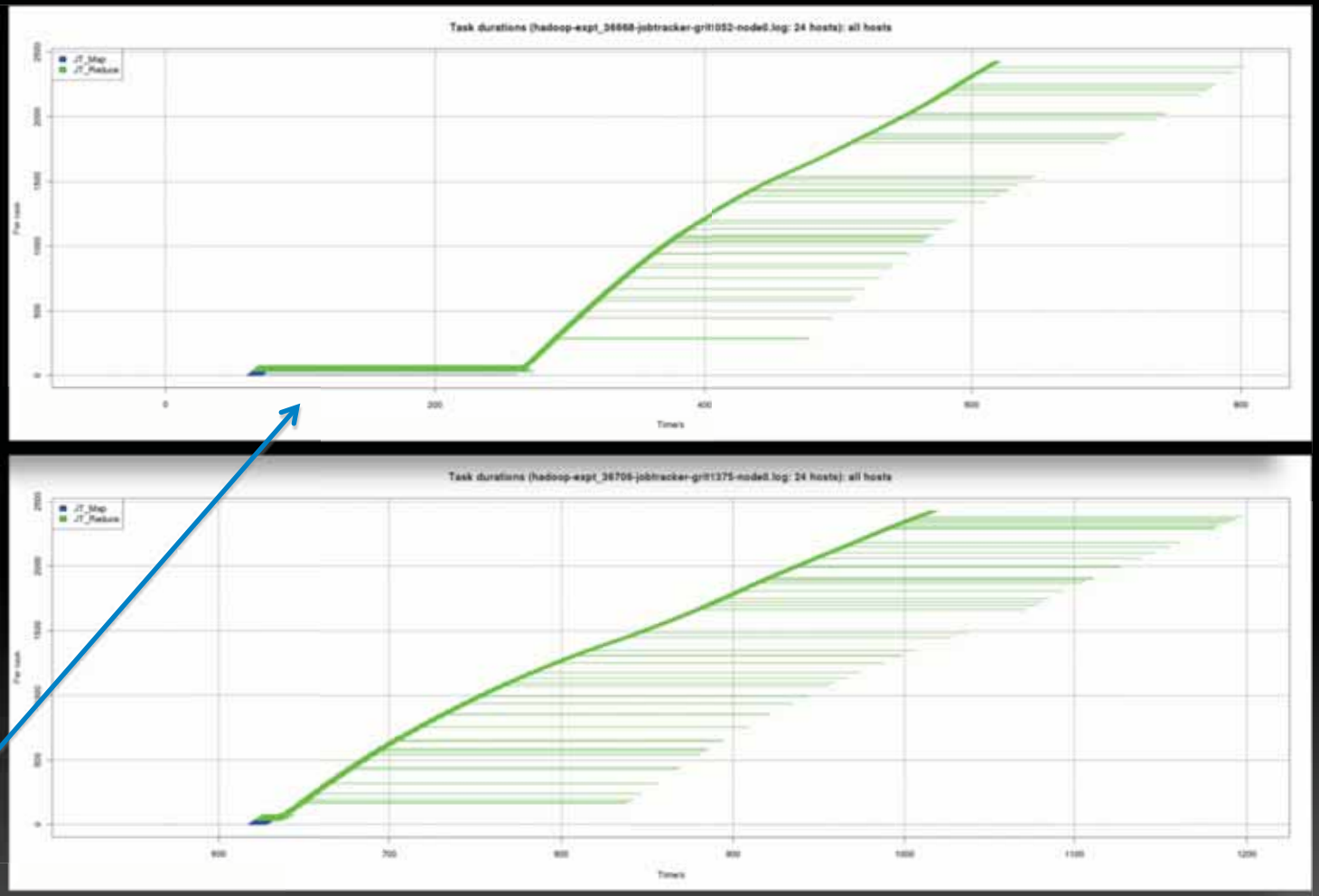
DiskHog on slave node
visible through lower heartbeat rate for that node

Visualization(*heatmaps*)



CPU Hog on node 1
visible on Map-task durations

Visualizations (*swimlanes*)



Long-tailed map
Delaying overall
job completion time

Current Developments

- State-machine extraction + visualization being implemented for the Hadoop Chukwa project
  - Collaboration with Yahoo!
- Web-based visualization widgets for HICC (Hadoop Infrastructure Care Center)
- "Swimlanes" currently available in Chukwa trunk (CHUKWA-279)

Briefly: Online Fingerpointing

- ASDF: Automated System for Diagnosing Failures
  - Can incorporate any number of different data sources
  - Can use any number of analysis techniques to process this data
- Can support online or offline analyses for Hadoop
- Currently plugging in our white-box & black-box algorithms

Hard Problems

- Understanding the limits of black-box fingerpointing
  - What failures are outside the reach of a black-box approach?
  - What are the limits of "peer" comparison?
  - What other kinds of black-box instrumentation exist?
- Scalability
  - Scaling to run across large systems and understanding "growing pains"
- Visualization
  - Helping system administrators visualize problem diagnosis
- Trade-offs
  - More instrumentation and more frequent data can improve accuracy of diagnosis, but at what performance cost?
- Virtualized environments
  - Do these environments help/hurt problem diagnosis?

Summary

- Automated problem diagnosis
- Current targets: Hadoop, PVFS, Lustre
- Initial set of failures
  - Real-world bug databases, problems in the wild
- Short-term: Transitioning techniques into Hadoop code-base working with Yahoo!
- Long-term
  - Scalability, scalability, scalability, ….
  - Expand fault study
  - Improve visualization, working with users
- Additional details
  - *USENIX WASL 2008*, USENIX SysML 2008, *HotDep 2009* *USENIX HotCloud 2009*, USENIX *HotMetrics 2009*

Our youngest fingerpointer (yet)

FOR MORE INFORMATION:

HTTP://WWW.ECE.CMU.EDU/~FINGERPOINTING

priya@cs.cmu.edu