X-Tracing Hadoop

Andy Konwinski, Matei Zaharia,
Randy Katz, Ion Stoica
CS division, UC Berkeley, andyk@cs.berkeley.edu

Motivation & Objectives

- **Motivation:**
  - Hadoop style parallel processing masks failures and performance problems
- **Objectives:**
  - Help Hadoop *developers* **debug** and **profile** Hadoop
  - Help *operators* **monitor** and **optimize** MapReduce jobs

About the RAD Lab

Reliable, Adaptive Distributed Systems

Setup for X-Tracing

- Instrument Hadoop using X-Trace framework
- Trace analysis
  - Visualization via web-based UI
  - Statistical analysis and anomaly detection
- Identify potential problems

Outline

- X-Trace overview
- Applications of trace analysis
- Conclusion

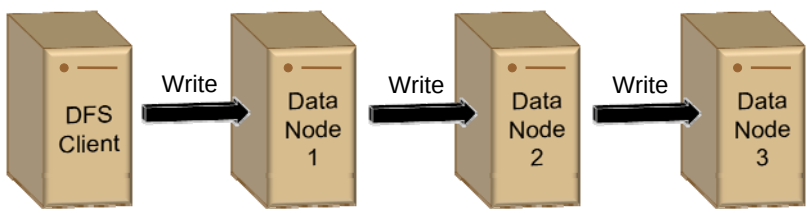
Checkpoint

- X-Trace overview
- Applications of trace analysis
- Conclusion

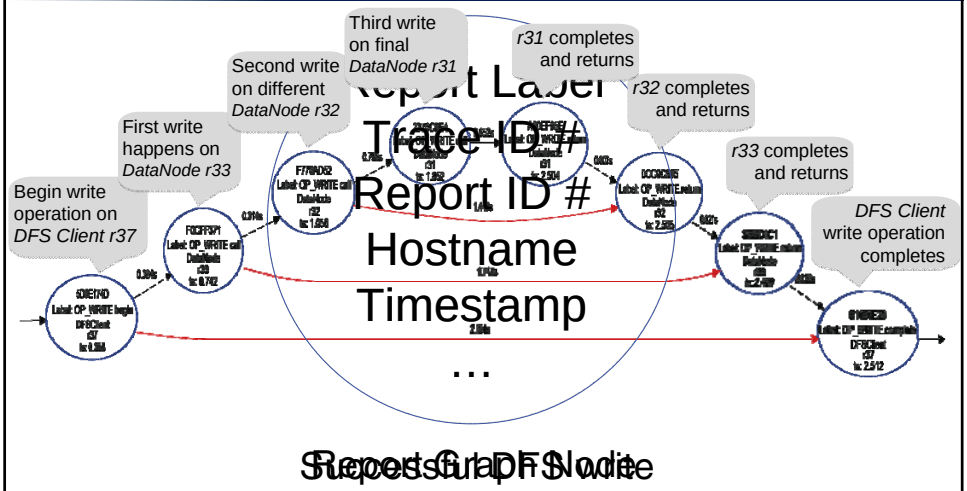
How X-Trace Works

- *Path based* tracing framework
- Generate event graph to capture causality of events across network
  - Examples of events: RPCs, HTTP requests
- Annotate *messages* with trace metadata (16 bytes) carried along execution path
  - Instrument Protocol APIs and RPC libraries

DFS Write Message Flow



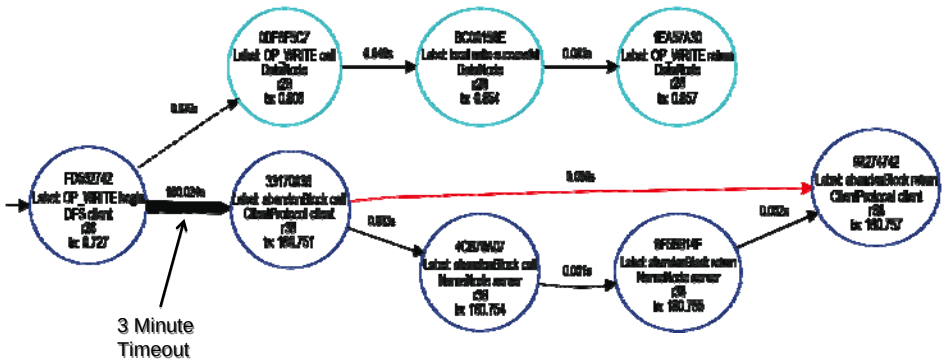
Example DFS Write



Request for DFS write

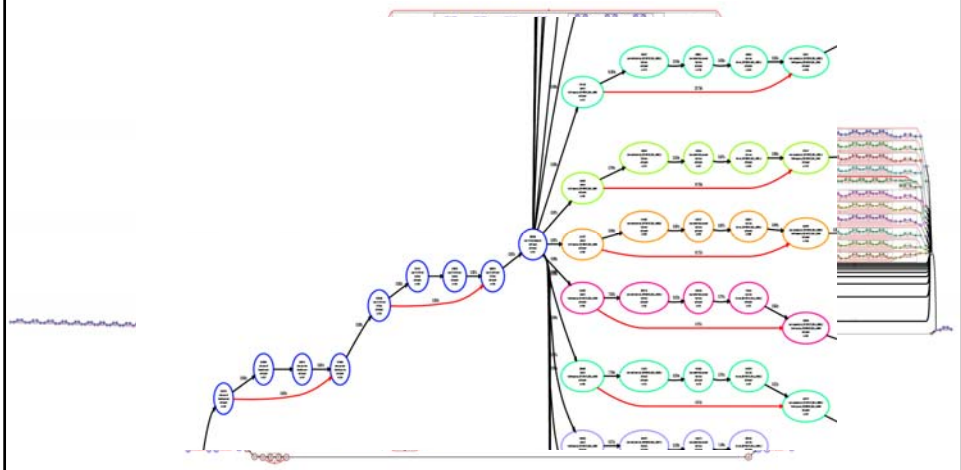
Report Graph

Failed DFS Write



Failed DFS write

MapReduce Event Graph

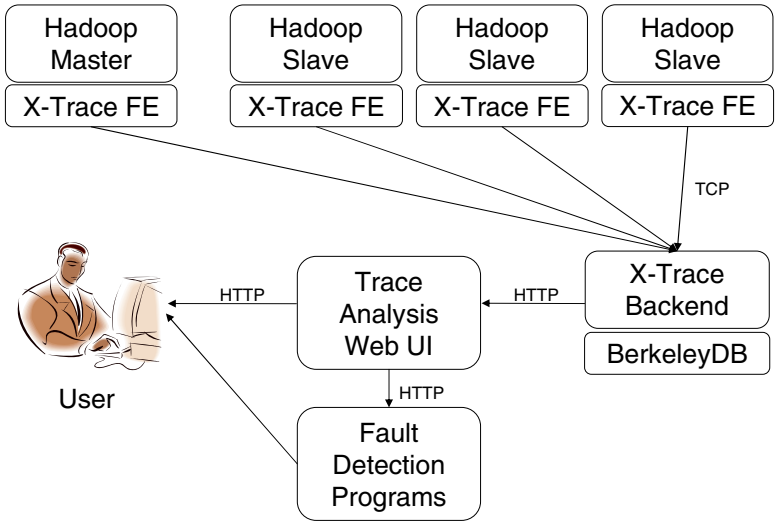


Example MapReduce graph

Properties of Path Tracing

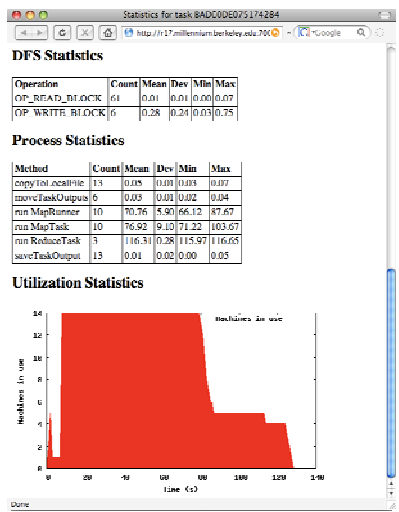
- Deterministic causality and concurrency
- Control over which events get traced
- Cross-layer
- Low overhead
- Modest modification of app source code
  - Fewer than 500 lines for Hadoop

Our Architecture



Trace Analysis UI

- Web-based
- Provides
  - Performance statistics
  - Graphs of utilization
  - Critical path analysis



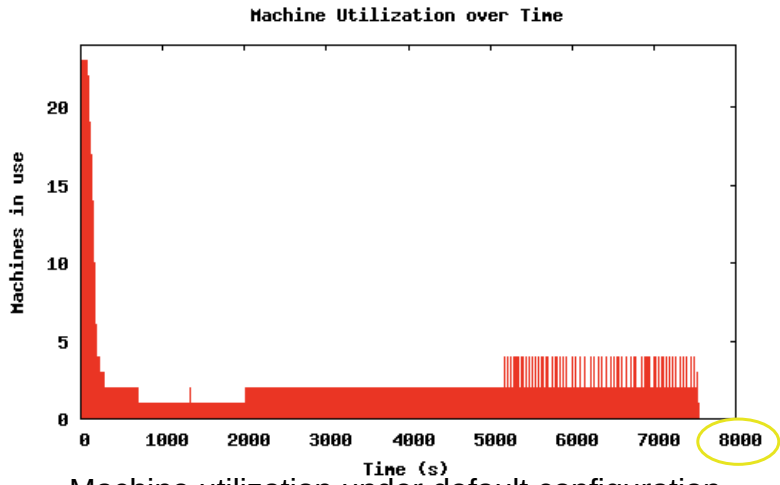
Checkpoint

- X-Trace overview
- Applications of trace analysis
- Conclusion

Optimizing Job Performance

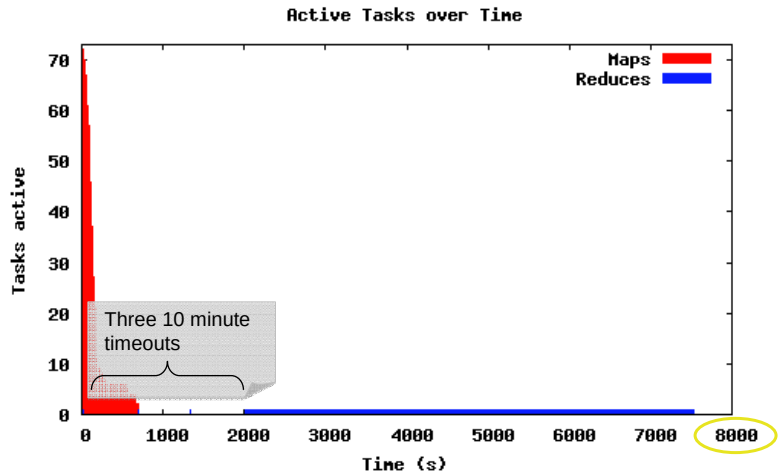
- Examined performance of Apache Nutch web indexing engine on a Wikipedia crawl
- Time to creating an inverted link index of a 50 GB crawl
  - With default configuration, 2 hours
  - With optimized configuration, 7 minutes

Optimizing Job Performance



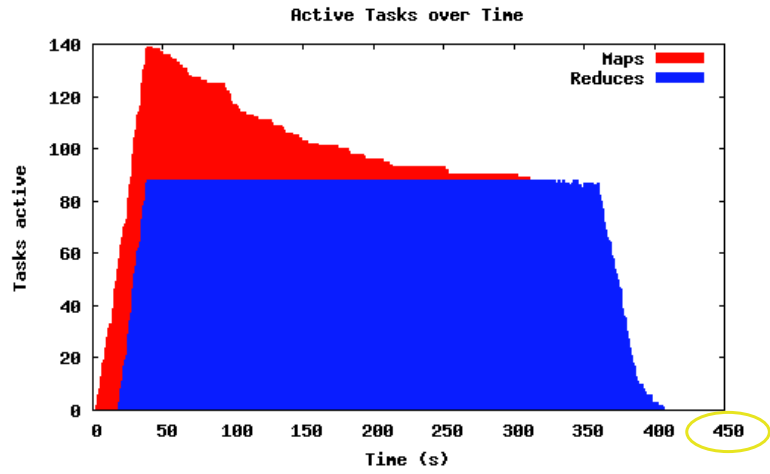
Machine utilization under default configuration

Optimizing Job Performance



Problem: One single Reduce task, which actually fails several times at the beginning

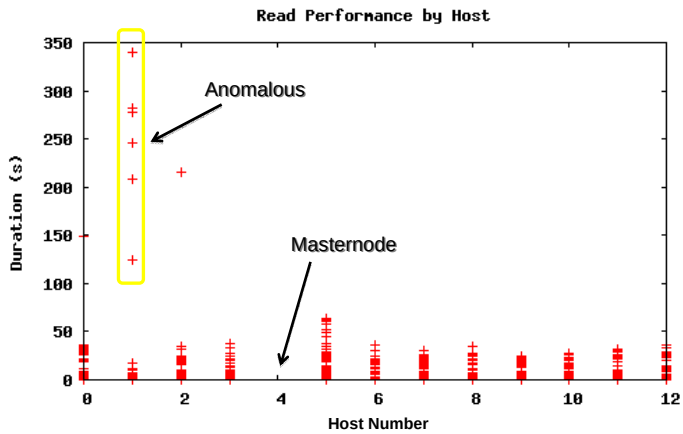
Optimizing Job Performance



Active tasks vs. time with improved configuration
(50 reduce tasks instead of one)

Faulty Machine Detection

- Motivated by observing slow machine



Diagnosed to be failing hard drive

Statistical Analysis

- Off-line Machine Learning
  - Faulty machine detection
  - Buggy software detection
- Current Work on graph processing and analysis

Future Work

- Tracing more production MapReduce applications
  - Larger clusters + real workloads
- More advanced trace processing tools
- Migrating our code into Hadoop codebase

Conclusion

- Efficient, low overhead tracing of Hadoop
- Exposed multiple interesting behaviors

andyk@cs.berkeley.edu
matei@cs.berkeley.edu

http://x-trace.net

Questions?

? ? ?

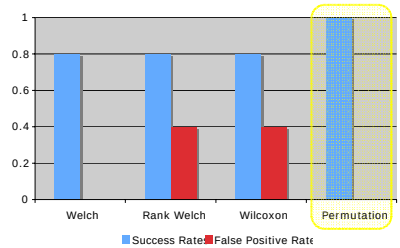
Andy's ToDo Slide:

- Hadoop Summit prep TODO:
  - A section on telemetry collection/correlation
  - Add url where they can go to check out x-trace (more about "official release")
  - ~~get feedback~~
  - ~~Practice talk~~
  - ~~Update all outline slides~~
  - Change green to blue on all graphs (color blind bastards!)
  - Email yahoo and joydeep about referencing our work with them

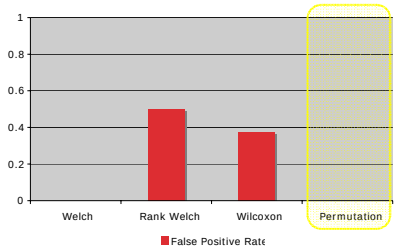
Overhead of X-Trace

- Negligible overhead in our 40-node test cluster
- Because Hadoop operations are large
- E.g. 18-minute Apache Nutch indexing job with 390 tasks generates 50,000 reports (~20 MB of text data)

Faulty Machine Detection

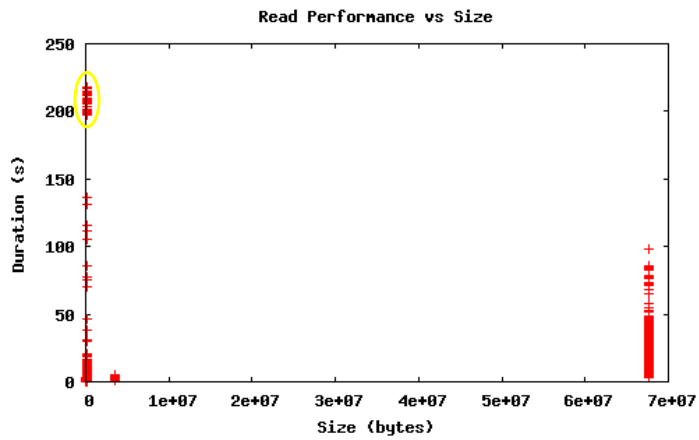


Failing disk, significance = 0.01



No failing disk, significance = 0.01

Observations about Hadoop

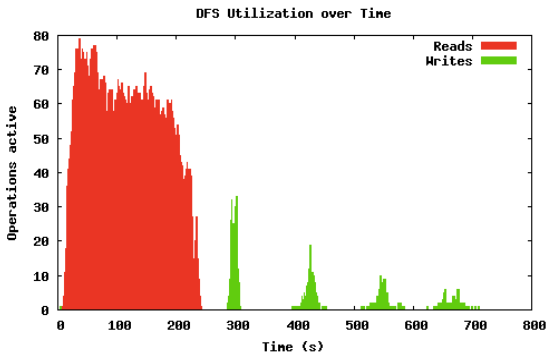


Unusual run - very slow small reads at start of job

Optimizing Job Performance

Init	0.0s (0.1%)
Start JVM	1.6s (4.8%)
Init runner	4.5s (13.4%)
Run map	13.1s (39.3%)
Cleanup	14.1s (42.4%)

Breakdown of longest map with 3x more mappers



Tracing also illustrates the inefficiency of having too many mappers and reducers

Related Work

- Per-machine logs
  - Active tracing: log4j, etc
  - Passive tracing: dtrace, strace
- Log aggregation tools: Sawzall, Pig
- Cluster monitoring tools: Ganglia

Event Graphs

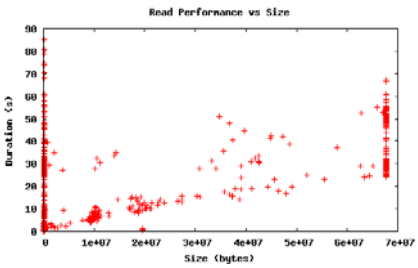
- Events are nodes in a “causal” directed graph
  - Captures causality (graph edges)
  - Captures concurrency (fan-out, fan-in, parallel edges)
- Spans layers, applications, administrative boundaries

Observations about Hadoop

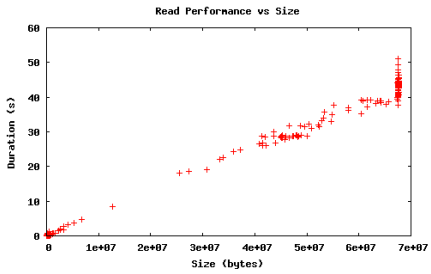
- Hardcoded timeouts may be inappropriate in some cases (e.g. long reduce task)
  - Solution: expose timeouts in configuration file?
- Highly variable DFS performance under load (on *our* cluster), which can slow the entire job
  - Solution: multiple hard disks per node?

Hardware Provisioning

Node with 1 Disk
(high variance typical)



Node with 4 Disks
(variance typically lower)



Faulty Machine Detection

- Approach: Compare performance of each machine vs. others in that trace
- Statistical Anomaly detection tests