Transparent System Call Based Performance Debugging for Cloud Computing

Nikhil Khadke, Michael P. Kasick, Soila P. Kavulya, Jiaqi Tan, and Priya Narasimhan,

PARALLEL DATA LABORATORY

Carnegie Mellon University

Introduction

- Automated problem diagnosis is challenging
  - **Complex system interactions:** Concurrency and distributed interactions can obscure root-cause
  - **Scale:** Large volume of monitoring data
  - **Production systems:** Lack luxury to modify instrumentation in legacy systems
- Focus of talk
  - Explore feasibility of **system calls** as **transparent instrumentation source** for debugging of **performance problems**

Related Work: Instrumentation

- End-to-end tracing of request flows
  - Instrumentation frameworks [Sambasivan11, Sigelman10]
  - Black-box approach [Aguilera03]
  - Analysis of production logs [Kavulya12, Tan09]

- Performance and event log analysis
  - OS-level logs [Kasick10]
  - Generation of problem signatures [Bodik10,Cohen05]
  - Alarm correlation [Oliner10, Kandula09]

Why System Calls?

- Captures interaction between application & OS
- Rich source of statistical and semantic data
  - Statistical: Disk/network transfer times
  - Semantic: Programs/files accessed, hangs/failures
- Higher reliability than application logs
  - Quality of logs dependent on application developer
  - Logs can become obsolete as system evolves
- Transparency
  - No need to modify to application source code
  - No dependence on hardware architecture

Goals and Assumptions

- Goals
  - Learn profiles of normal behavior using system calls
  - Localize problematic node and type of problem
- Assumptions
  - Majority of nodes exhibit fault-free behavior
  - Workload evenly balanced across nodes
  - Hardware configurations similar across nodes
  - Clocks on nodes are synchronized
- Initial exploration
  - Diagnose resource contention in MapReduce systems

Target System: Hadoop

- Hadoop: Open-source implementation of MapReduce
  - Long running jobs (> 100s): Hard to label failures
  - Large, distributed: Hard to isolate failures
  - Leased cloud infrastructure: Slowdowns are costly
- MapReduce job consists of two main abstractions
  - **Maps**: Process smaller partition of large job in parallel
  - **Reduces:** Fetch, merge, and sort output of map tasks

Overview of Approach

Instrument System

- Collect per-node system call traces

Localize Problem

- Identify nodes whose system call profiles differ significantly from peers
- Studied two peer-comparison approaches

System Call Instrumentation

- strace(), a UNIX utility captures system calls invoked by program
- We record:
  - Average and total time spent in each system call
  - Number of total and failed system call invocations
  - Percentage of time of spent in the system call
  - Verbose trace of system calls

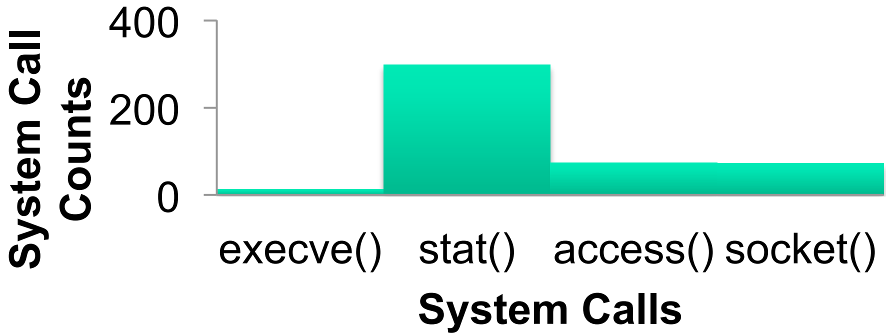
System Call Scope and Choice

- Network related:
  - accept()
  - connect()
  - bind()
  - socket()
- File System related:
  - access()
  - stat()
- Process related:
  - execve()

- Omitted send/receive calls due to large variance in data processed
- Overhead of naïve implementation: 1.2x

Diagnostic Approach

- Learn per-node profiles of normal behavior



- Two-step threshold for generating alarms
  - **Local alarms:** Accounts for normal variance between nodes (min. and max. bin counts for thresholds)
  - **Global alarms:** Pairwise histogram-comparison flags nodes which differ significantly from peers
- Use heuristics to identify type of problem

Diagnostic Approaches

- Statistical diagnostic approach
  - Histogram of invocations of system call per time window

% time	seconds	usecs/call	calls	errors	syscall
96.15	0.001524	117	13	7	execve
3.85	0.000061	0	1705	1198	stat
0.00	0.000000	0	74	44	access
0.00	0.000000	0	72	1	socket

- Semantic diagnostic approach
  - Uses verbose system traces
  - Identify unique system call invocations
    - Unique call → system calls, arguments, return code
    - E.g., stat(fileX) = -1 ENOENT (No such file or dir)

Experimental Setup

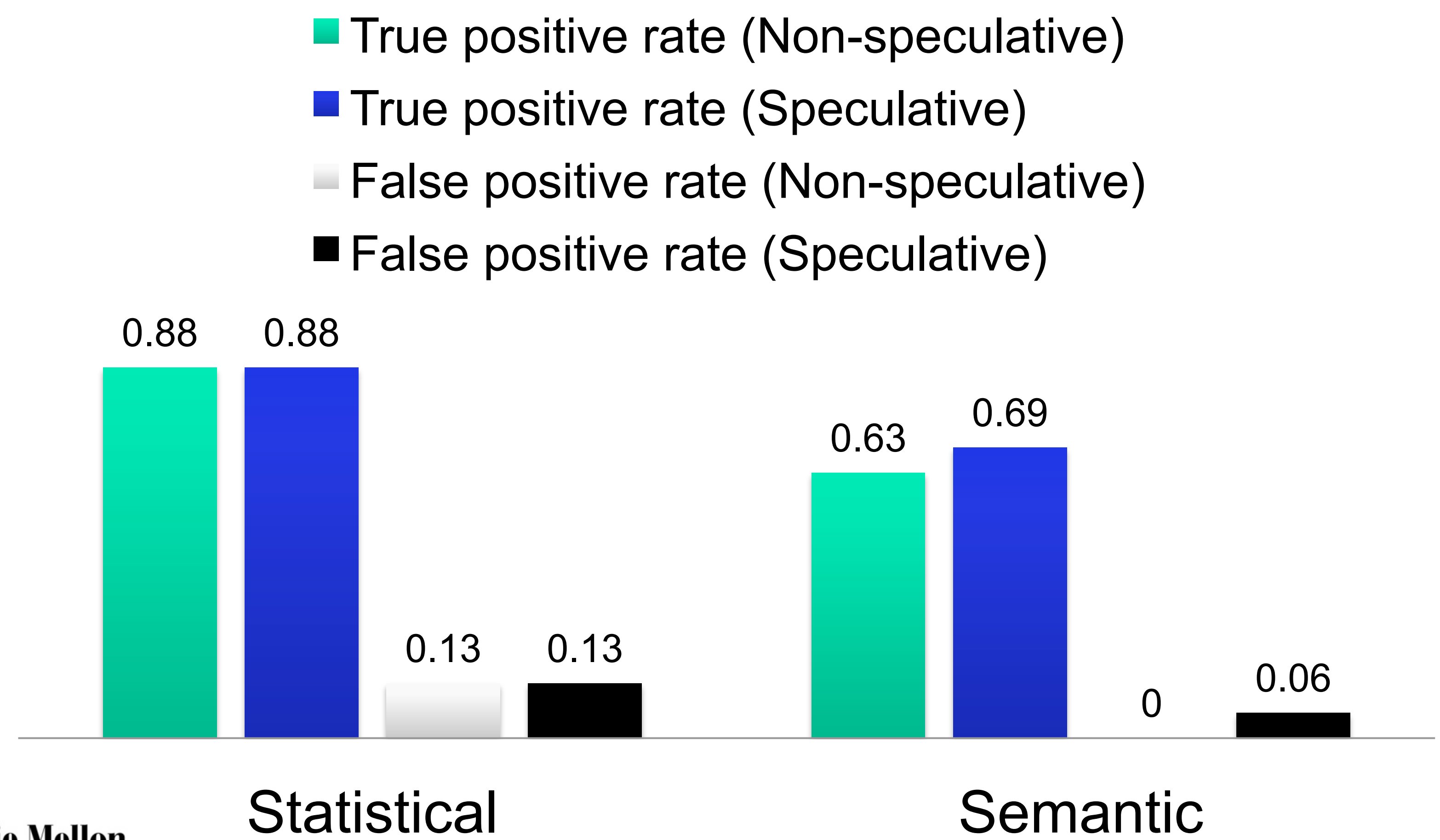
- Experimental Setup
  - 5 identical machines: Single master and 4 slaves
- Hadoop workloads
  - wc – count the frequency of every word in a text corpus of 100,000 words
  - sort – sorts 100,000 randomly generated numbers
- Performance problem injection
  - Disk-hog – write 2GB chunks continually to disk
  - Packet-loss – drop 5%, 20% and 50% of the network packets on a node

Results: Terminology

- True positive
  - Faulty node and root-cause correctly identified
- False positive
  - Incorrectly identify a node or provide a wrong root-cause
- Output node set
  - Set of all the nodes that are outputted by our algorithms (and believed to be faulty).

Results: Disk hog

Disk hog signature: More time in stat() + less time in execve()

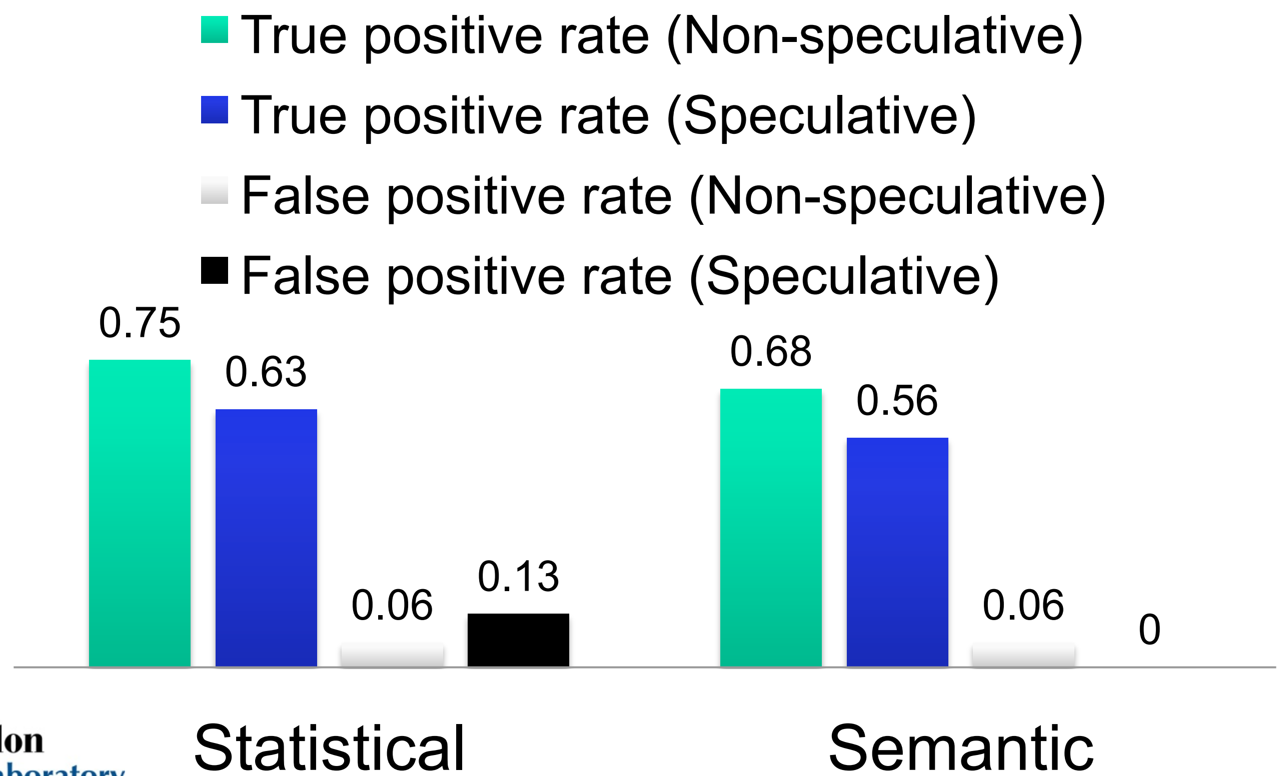


Carnegie Mellon
Parallel Data Laboratory

Results: Packet Loss

Packet loss signature: More time in connect()
+ less time in accept()

Results for 50% Packet Loss



Lessons Learned

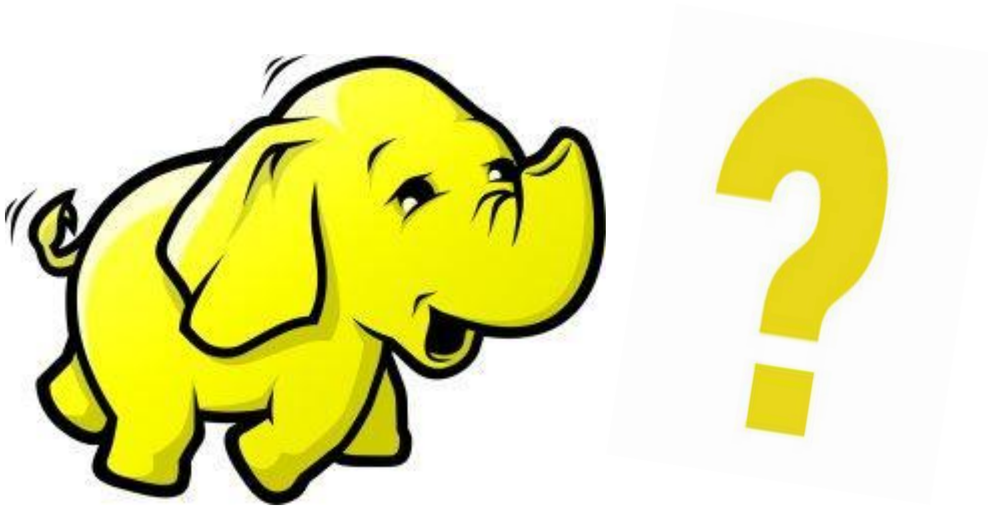
- Diagnostic approaches
  - Statistical approach more effective semantic approach
  - Diagnostic accuracy independent of Hadoop workload
  - Speculative execution introduces variance that reduces diagnostic accuracy

- Performance Problems
  - Disk-related issues were easier to detect
  - Lower accuracy for network-related problems probably due to correlated problem manifestation across nodes

Future Work

- How do cope with heterogeneous systems?
  - Normalize behavior across heterogeneous nodes?
  - Limit peer comparison to nearest neighbors?
- How do we reduce instrumentation overhead?
- Can we distinguish between application-level problems and infrastructural problems?

Questions?



Related Work (1)

- **[Aguilera03]** Performance debugging for distributed system of black boxes. M. K. Aguilera, J. C. Mogul, J. L. Wiener, P. Reynolds, and A. Muthitacharoen. SOSP 2003
- **[Barham04]** P. Barham, A. Donnelly, R. Isaacs, and R. Mortier. Using Magpie for request extraction and workload modelling. SOSP 2004.
- **[Bodik10]:** Fingerprinting the datacenter: automated classification of performance crises. P. Bodík, M. Goldszmidt, A. Fox, D. B. Woodard, H. Andersen. EuroSys 2010.
- **[Cohen05]:** Capturing, indexing, clustering and retrieving system history. Ira Cohen, Steve Zhang, Moises Goldszmidt, Julie Symons, Terence Kelly, Armando Fox. SOSP, 2005.
- **[Kandula09]:** Detailed diagnosis in enterprise networks. Srikanth Kandula, Ratul Mahajan, Patrick Verkaik, Sharad Agarwal, Jitendra Padhye, Paramvir Bahl. SIGCOMM 2009.
- **[Kasick10]:** Black-Box Problem Diagnosis in Parallel File Systems. Michael P. Kasick, Jiaqi Tan, Rajeev Gandhi, Priya Narasimhan. FAST 2010.

Related Work (2)

- **[Kavulya12]:** Draco: Statistical Diagnosis of Chronic Problems in Large Distributed Systems. Soila Kavulya, Scott Daniels, Kaustubh Joshi, Matti Hiltunen, Rajeev Gandhi, Priya Narasimhan. DSN 2012
- **[Oliner2010]** Using correlated surprise to infer shared influence. A. J. Oliner, A. V. Kulkarni, and A. Aiken. DSN 2010.
- **[Sambasivan11]:** Diagnosing Performance Changes by Comparing Request Flows. Raja R. Sambasivan, Alice X. Zheng, Michael De Rosa, Elie Krevat, Spencer Whitman, Michael Stroucken, William Wang, Lianghong Xu, and Gregory R. Ganger. NSDI 2011.
- **[Sigelman10]** B. H. Sigelman, L. A. Barroso, M. Burrows, P. Stephenson, M. Plakal, D. Beaver, S. Jaspan, and C. Shanbhagy. Dapper, a large-scale distributed systems tracing infrastructure. Google Technical Report, April 2010.
- **[Tan09]:** Mochi: Visual Log-Analysis Based Tools for Debugging Hadoop. J. Tan, X. Pan, S. P. Kavulya, R. Gandhi, P. Narasimhan. HotCloud09.