

Towards Automated Problem Analysis of Large-Scale Storage Systems

Priya Narasimhan
Greg Ganger
Chuck Cranor

Carnegie Mellon University



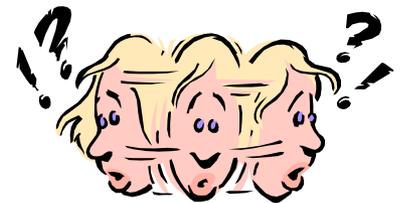
Automated Failure Analysis

- **Failure analysis difficulty...**
 - Creates major problems for administrators
 - Worsens as scale and complexity grows
- **Goal: automate it and get proactive**
 - Failure detection and prediction
 - Problem determination (or “fingerpointing”)
- **How: Instrumentation plus statistical-analysis tools**



Challenges in Problem Analysis

- **Goal: identify failed components and root cause**
 - Guiding repair and prevention of recurrence
- **Challenging in 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
- **Fingerpointing**
 - Pinpoint the faulty server or node in the system



Goals

- Runtime data collection
- Runtime data analysis
- Performance – low false-positive rate, low overheads
- Work for various workloads and under workload changes
- To make this practical
 - Flexibility to attach/detach any data source or analytics
 - Work in **production environment** – no luxury to modify application code
 - Support online and offline analyses
- Non-goals (for now)
 - Root-cause analysis



Exploration of Fingerprinting

- **Previous targets: Distributed fault-tolerant storage systems**
 - CMU's Self-* Storage (supported by the PASIS read/write protocols)
 - MIT's Castro-Liskov BFS (supported by BFT protocols)
 - Presented last year
- **Current explorations**
 - Hadoop (somewhat mature)
 - Yahoo's open-source implementation of Map/Reduce
 - PVFS (early exploration)
- **Injected various types of problems**



Target List of Problems

- Poor performance
 - Caching disabled, working poorly
 - Blocksize, striping, rotation factor (configuration error)
 - Increased serialization of I/O and network requests
 - Data structures or algorithms run slower than expected
- Resource leaks
 - Leak of protocol resource (e.g. file handles)
 - Memory leak
- Correctness
 - Cache consistency
 - Files end up empty or have wrong contents
 - Directory entries are missing
- Hangs
 - Infinite loop on operation
- Miscellaneous - Server won't start
 - Port in use, bad saved state

Culled from
bug-tracking
databases for

- Self-* Storage
- CODA
- CITI Linux NFSv4
- OpenAFS

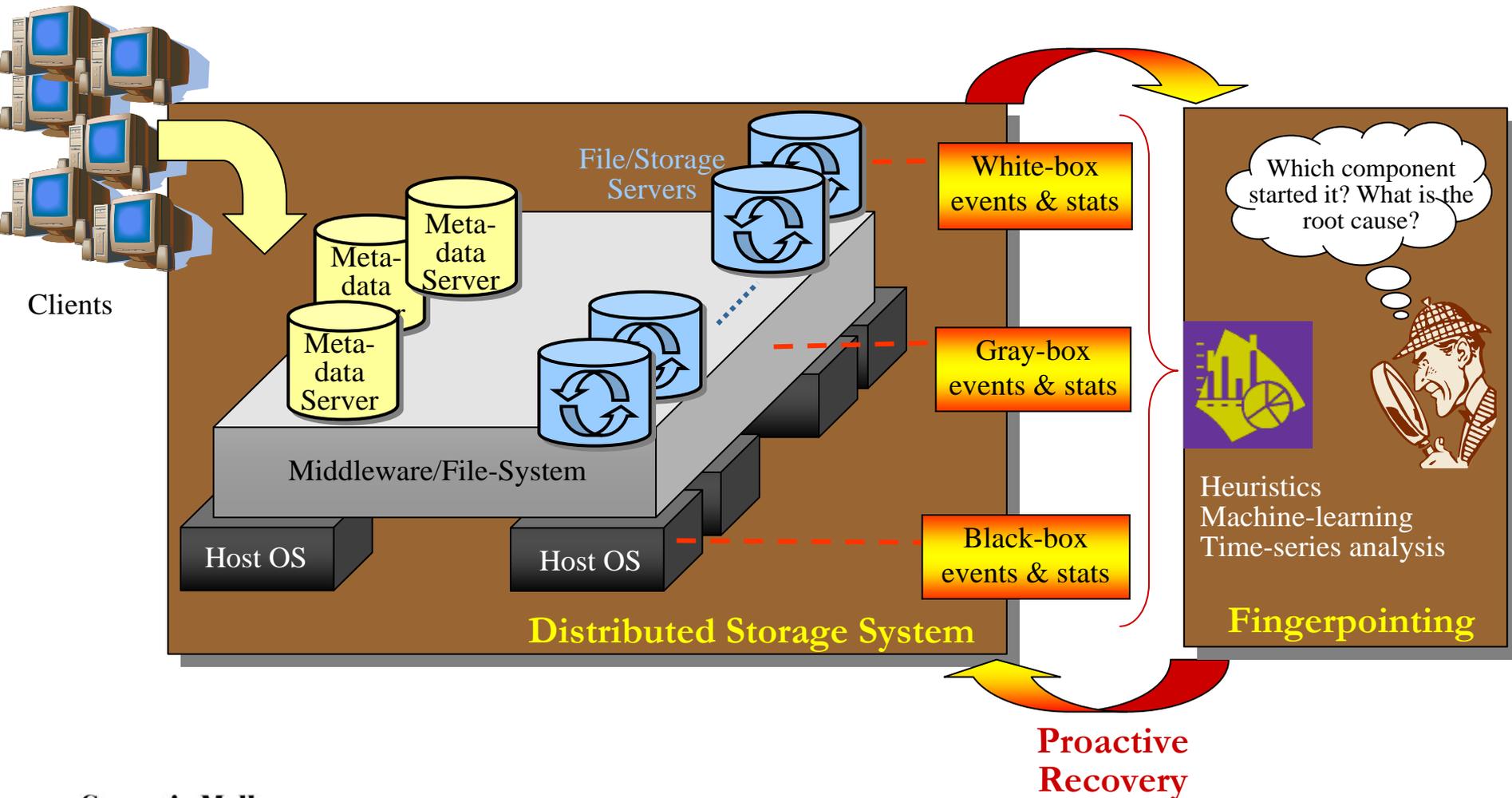


Online Fingerprinting Framework

- **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



Fingerpointing Framework



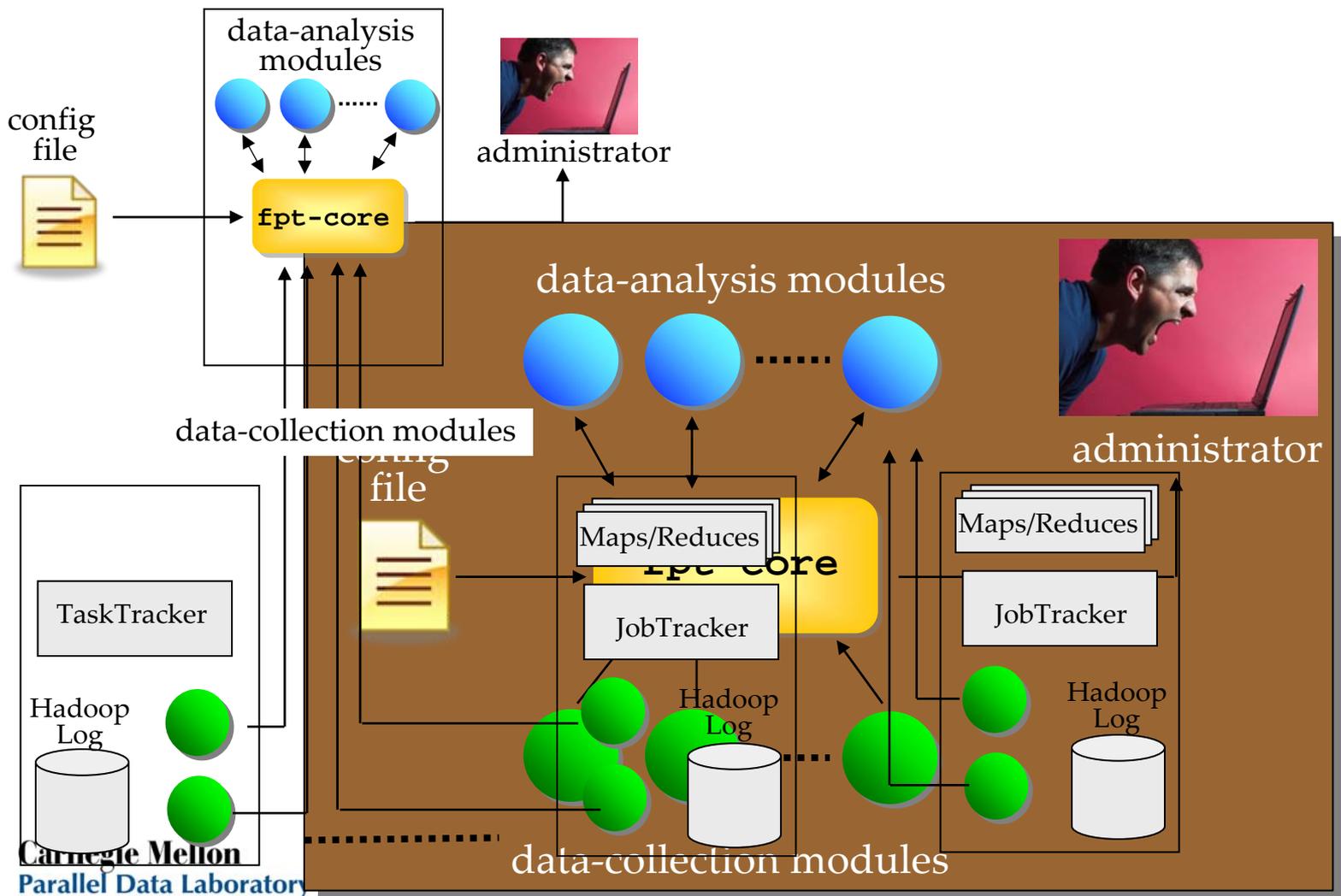
Towards More Instrumentation

- **Black-box** metrics
 - Network traffic-rate, memory usage, context-switch rate, CPU usage, paging, disk activity
- **Gray-box** metrics
 - HTTP requests/second, cache hits/misses
 - Number of threads in the thread pool, number of connections
 - Number of exceptions/second
- **White-box** metrics
 - Application-level response times
 - Application-level information, e.g., workload changes
 - System configuration monitored through system logs
 - Application-level native logs

Architecture

- Core demux engine (**fpt-core**)
 - Data sources and analysis techniques are *modules*
 - Plug-in/out modules conform to the same API
- Current data modules
 - `sadc`, `pidstat`, `strace`, `netstat`
 - Hadoop logs, Apache logs
- Current analysis modules
 - Moving average, horizontal (cross-node) correlation, vertical (intra-node) correlation, machine learning (k -NN)
- Generate DAG from configuration files
 - “Wiring diagram” for connecting data sources to sinks
 - Scheduling data sources as needed

Target 1: Hadoop



Some Hadoop Problems of Interest

| Manifestation | Example of problem |
|--------------------------------|--|
| Crash/Abort | Hadoop 1748: Task Trackers fail to launch tasks when they have relative log directories configured causing the task tracker to look for logs in wrong directory. |
| Hang | Hadoop 1036: Infinite loop at task tracker due to unhandled exception. The process hang is not detected by the job tracker heartbeats due to error in handling task cleanup. |
| | Hadoop 1255: Infinite loop at name node if a data node registers twice then subsequently dies. The name node hangs when attempting to clear the duplicate registration from the queue. |
| Performance degradation | M45 cluster: Nutch job depletes memory on cluster and causes nodes to be unresponsive. |
| | Hadoop users: Sep 13, 2007; 05:37pm: CPU bottleneck caused by running name node and data node on the same machine. |
| | Hadoop users Sep 26, 2007: Excessive messages logged to file during job tracker startup. |
| Error messages | The name node continuously throws exceptions when replicating corrupted block since the receiving side rejects the block due to checksum error and the namenode keeps on retrying. The periodicity of the retry is low so there is little impact on performance. |
| Value faults | Random corruption of data between map and reduce due to incorrect setting of a field. |
| | Datanode corruption if machine dies while writing VERSION file |

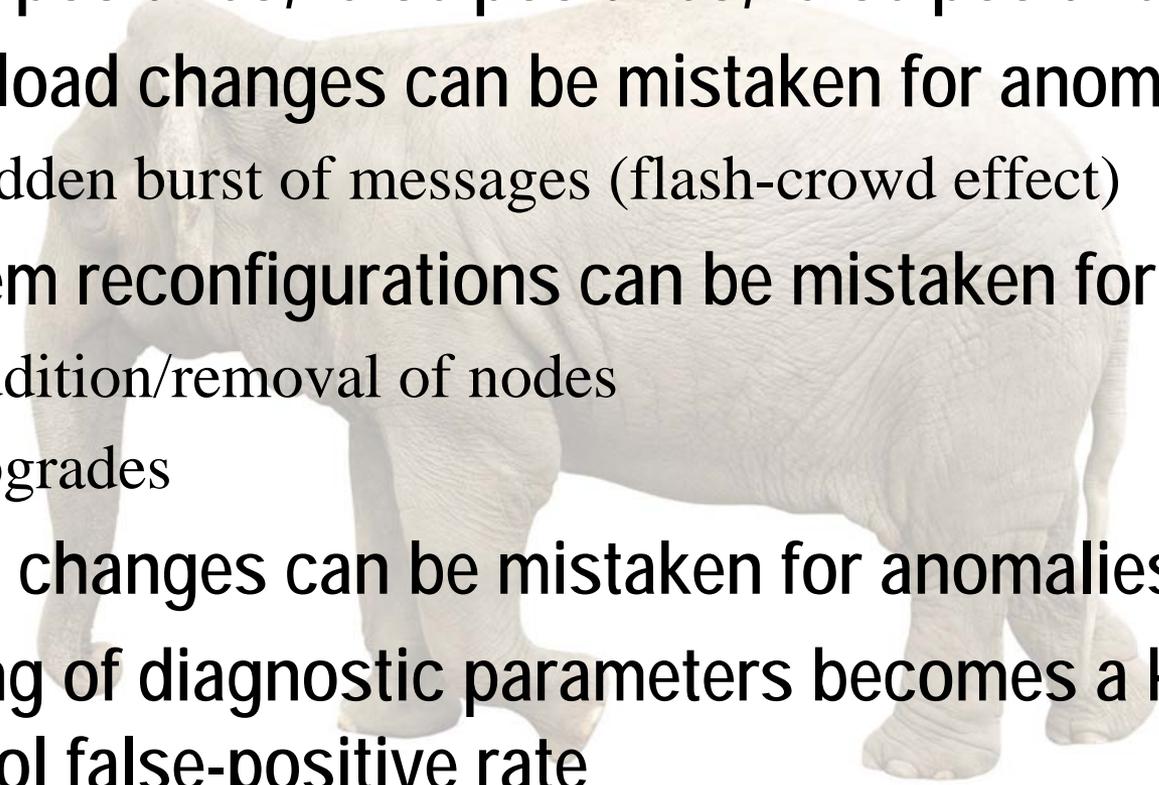
Hadoop: Results So Far

- Testbed: 7 nodes
- Problems: Crash, abort, hang, misconfiguration
- Workloads: Random writer, sort, Nutch
- Most relevant data sources
 - `sysstat`, `netstat`
 - Hadoop logs (capturing events and state transitions within Hadoop)
 - Localizes problems while Hadoop and its applications are running
- Algorithms for analyzing data
 - White-box data gathered from existing Hadoop logs
 - Simple machine-learning techniques [USENIX SysML 2007]
 - Time-series analysis, correlation [SRDS 2008, CMU-PDL-TR-08-104]

Experiences

- **Fingerpointing latency**
 - White-box and black-box analysis modules detected injected performance problems with < 1 min latency over multiple runs
- **False-positive rate**
 - Black-box false-positive rate over problem-free runs: $< 1\%$
 - White-box false-positive rate over problem-free runs: $< 1\%$
- **Instrumentation overheads**
 - `hadoop_log_rpc`: 0.0245% CPU
 - `sadc_rpcd`: 0.3553% CPU
 - `fpt-core`: 0.8063% CPU

The Elephant in the Room

- False positives, false positives, false positives,
 - Workload changes can be mistaken for anomalies
 - Sudden burst of messages (flash-crowd effect)
 - System reconfigurations can be mistaken for anomalies
 - Addition/removal of nodes
 - Upgrades
 - Mode changes can be mistaken for anomalies
 - Tuning of diagnostic parameters becomes a key to control false-positive rate
- 

Target 2: PVFS

- Testbed: 1 client node, 3 metadata servers, 3 I/O servers
- Problems: Disk hog, packet loss
- Workloads: dd, postmark, IOzone
- Most relevant data sources
 - `sysstat`
 - `/proc/net/snmp`, `/proc/net/netstat`, `/proc/net/tcp`
- **Initial approach**
 - Black-box analysis of `sysstat`/network data for hardware faults
 - White-box/gray-box analysis of the PVFS state machines, and possibly BMI requests to model what the system should be doing and identify when the system is not doing as it should
 - System-call analysis of clients/server to discover value faults

Hard Problems

- **Generating automatic configurations tailored to specific classes of faults**
 - For example, if we wanted to pursue misconfigurations, which data sources would we “wire up” to which analysis modules?
- **Understanding the limits of black-box fingerprinting**
 - What kinds of failures are outside the reach of a pure black-box approach?
- **Scalability**
 - Scaling the ASDF infrastructure to run across large systems and understanding the “growing pains”
 - New algorithms that can scale and that are hierarchical
 - 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?
 - What level of instrumentation is “good enough” for accurate diagnosis?
- **Virtualized environments**
 - Do these environments help/hurt problem diagnosis?

Next Steps

- **More experimentation**
 - Expand failure-injection campaign for PVFS
 - Expand instrumentation sources for PVFS
 - Deploy online analysis for PVFS
- **Scalability**
 - Migrate ASDF to large-scale environments
 - Possible target: M45, Yahoo! data-center
 - 4,000 processors, 3 TB of memory, 1.5 PB of disks
 - Multiple applications running
 - Large-scale production environment
 - Another target: EC2
- **Improve algorithms**
 - Leverage topology and dependency information

Summary

- Focus on automated online & offline problem determination
- Previous targets: BFS, Replicated-CoreFS
- Current targets: Hadoop, PVFS
- Online fingerprinting framework
 - Plug in black-box and white-box data sources & analytics
- Initial set of failures from real-world bug databases
- Next
 - Scalability, scalability, scalability,
 - Expand failure-injection campaign
 - Identify “better” data sources

For more information:

<http://www.pdl.cmu.edu/>

priya@cs.cmu.edu

ganger@ece.cmu.edu

