Automated Online Failure Diagnosis in Data Centers

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

Motivation

- Failure diagnosis
- Important in distributed systems
- Systems consist of third-party software and distributed components
- Failures and performance problems cannot be reasoned about in isolation
- Fingerpointing = problem localization = identifying culprit nodes

Challenges

- Multiple possible root causes, single problem manifestation
- Multiple problem manifestations, yet single root cause
- Problems and/or their manifestations might change their “shape” as they traverse inter-connected components

Goals

- Online data collection
- Online data analysis
- Performance – low false-positive rate, low overheads
- Work for various workloads and under workload changes
- Be 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)
  - Identifying the bug or offending line of code

NEW! Online Fingerpointing 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

Results So Far

- Transparent online fingerpointing
  - Discovers culprit nodes by analyzing data traces
  - Localizes problems while the target system is running
  - Does not require any modification of the applications
- Algorithms for analyzing data
  - Black-box fingerpointing
    - Data gathered transparently from the OS/network
  - White-box fingerpointing
    - Existing application-specific logs

Target Systems

- Initial focus: Dagnosis in data-centers
- Currently working with Yahoo! data-center
  - 4,000 processors, 3 TB of memory, 1.5 PB of disks
  - Multiple applications running
  - Large-scale production environment
- Initial target application
  - Hadoop, Yahoo's implementation of Google's Map/Reduce scalable programming architecture
- Other targets of interest (ongoing)
  - Large-scale storage systems, multi-tier enterprise systems, virtual machines

Details

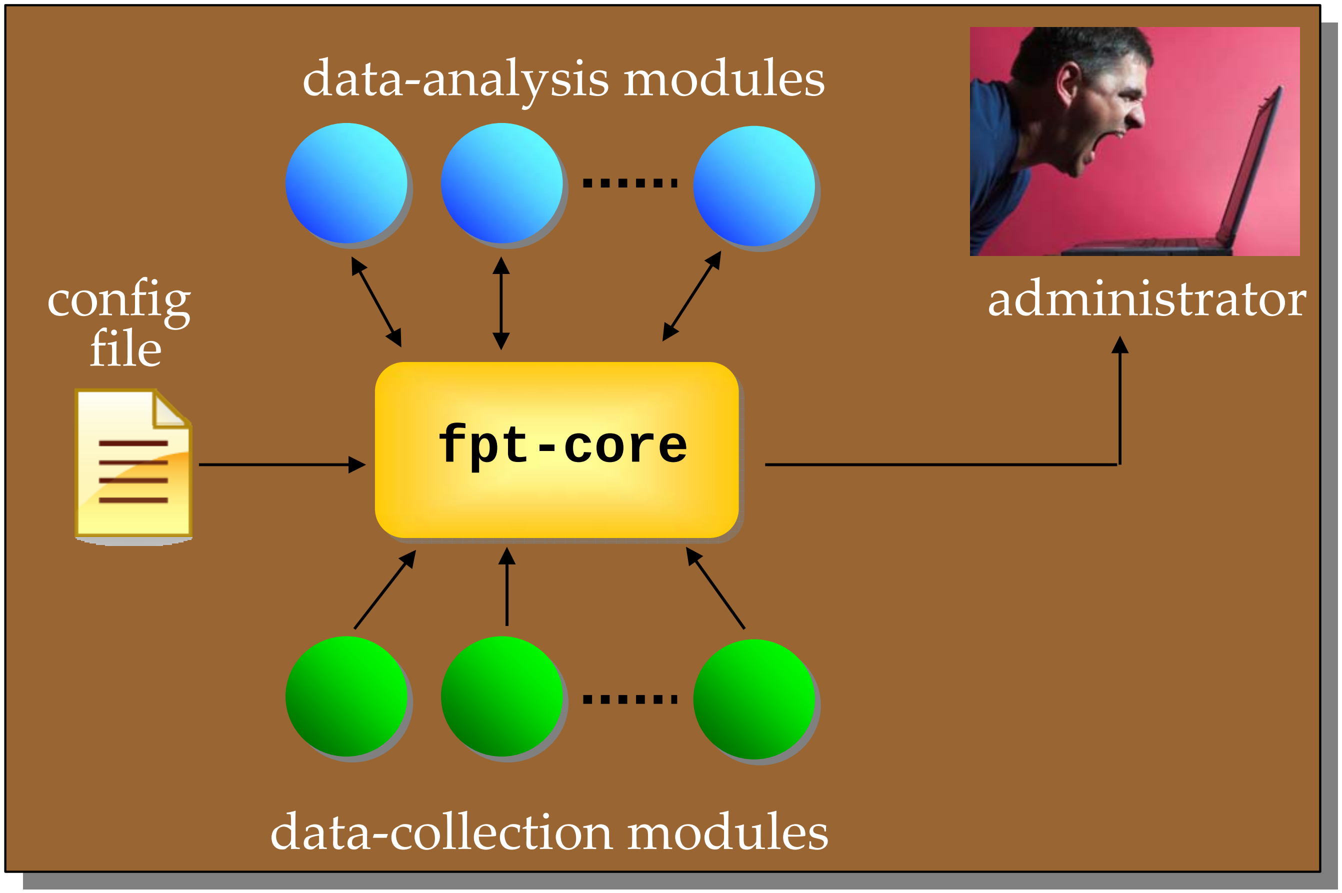
- Runtime data collection
  - Centralized location for accessing arbitrary data sources collected at varying rates
  - Allow synthesis of multiple data sources
  - Handle collection of data generated at varying rates
- Runtime data analysis
  - Simultaneous analysis of data using different algorithms
  - Process incoming data streams in real time to produce list of suspected problem nodes

ASDF

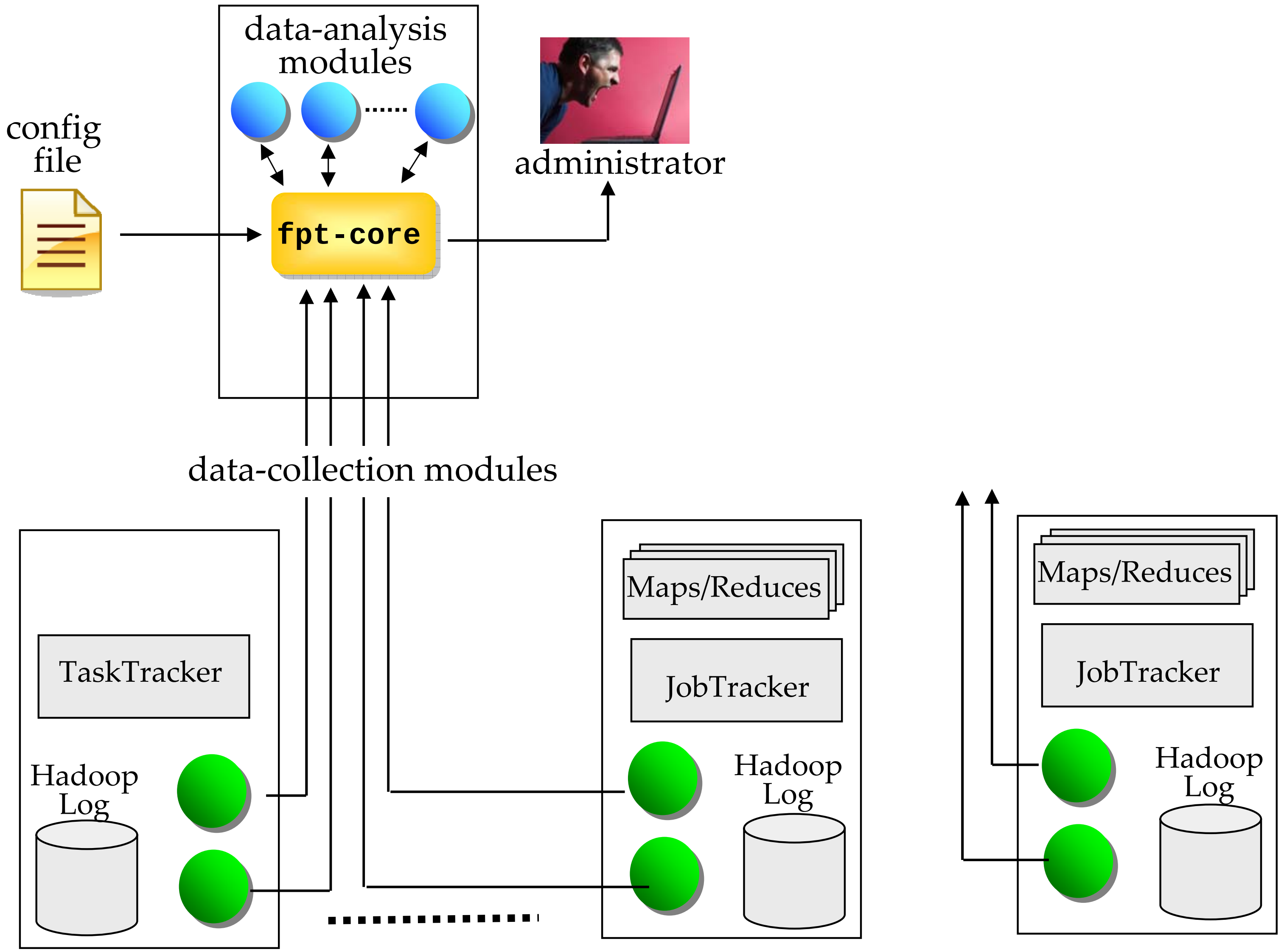
Architecture

- 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`
- 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

ASDF Architecture



ASDF Architecture



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

Experiences

- Target set of faults (from the Apache Hadoop bug database)
  - Crash, packet-loss, deadlock, out-of-memory, cache misconfiguration, hang, CPU overload, log overrun, data corruption, exceptions
- White-box and black-box analysis modules fingerpointed the culprit for injected faults with < 1 min latency over multiple runs
- Black-box and white-box false-positive rate over problem-free runs: <1%
- CPU usage of ASDF
  - On each node: 0.0245% CPU
  - On the `fpt-core` node: 0.8063% CPU

Diagnostic Experiences

- Fault manifestations tend to "travel" within systems
  - Performance problem observed on one (faulty) node can "travel" to other (non-faulty) nodes
  - This traveling phenomenon shows up in various metrics
  - No such thing as "independent failures" in practice
- No single performance metric is sufficient for root-cause analysis
  - Combined analysis of multiple metrics is required
  - Single metric leads to many false-positives and false-negatives
- Different failures have different "signatures"
  - Possible to perform black-box diagnosis of some failures by identification of these signatures

What's Next?

- 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 fingerpointing
  - What kinds of failures are outside the reach of a black-box approach, attractive as such an approach might be?
- Scalability
  - Scaling the ASDF infrastructure to run across the Yahoo! cluster of 4000 processors and understanding the “growing pains”
  - New algorithms that can scale and that are hierarchical
- 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?
- Visualization
  - Really difficult for system adminstrators to visualize problems at scale

Summary

- ASDF – Online fingerpointing framework
- Pluggable data sources
- Pluggable analysis modules
- Initial results with Hadoop for documented performance problems in public bug database
- Next steps
  - Visualization
  - Scalability
  - More algorithms, more data sources
  - Limits of a black-box approach
- Potentially a future IFIP workshop? DSN workshop?