

Connecting the Dots: Using Runtime Paths for Macro Analysis

Mike Chen, Emre Kiciman, Anthony Accardi,
Armando Fox, Eric Brewer
<http://pinpoint.stanford.edu>



Motivation

- Difficult to monitor and debug dynamic, distributed systems
 - Divide and conquer, layering, and replication disperse execution context throughout the system
 - E.g., Internet services, P2P, and sensor networks
- Lots of existing low-level tools that help with debugging individual components, but not a collection of them
- Need tools that help us understand how components/peers/sensors interact and depend on each other



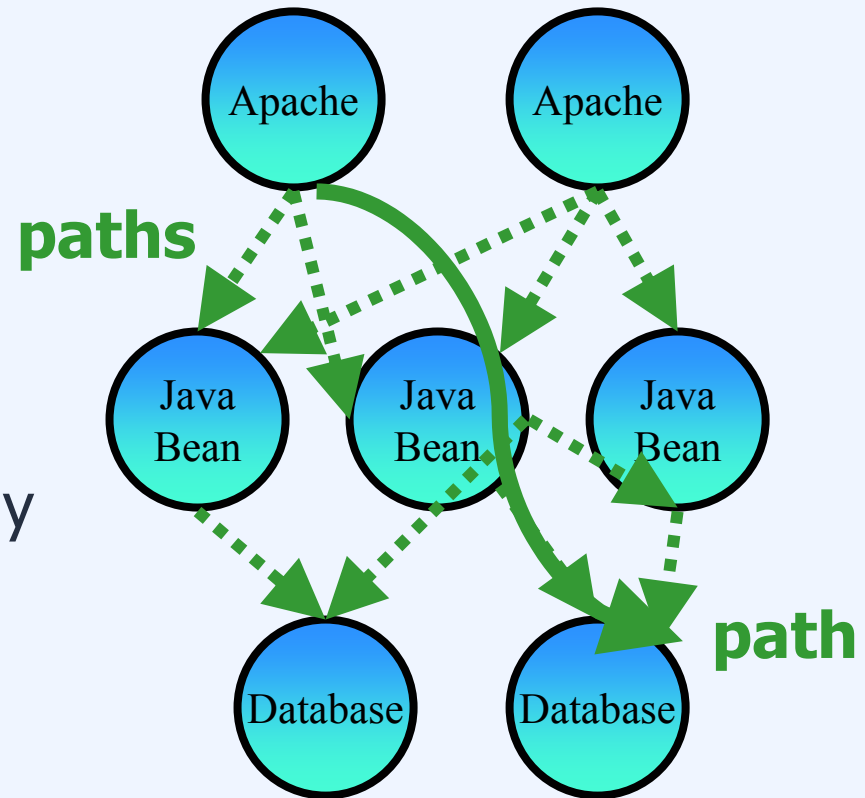
Macro Analysis

- Macro analysis exploits non-local context to improve **reliability** and performance
 - Performance examples: Scout, ILP, Magpie
 - Statistical view is essential for large, complex systems
- Beehive Analogy:
 - micro analysis allows you to understand the details of individual honeybee; macro analysis is needed to understand how the bees interact to keep the beehive functioning
- Open challenges macro analysis helps with:
 - Deducing system structure
 - Detecting and diagnosing application-level failures (improve MTTR)
 - Verifying macro invariants



Our Approach

- **Use runtime paths to connect the dots!**
 - dynamically captures the component interactions
 - Analyze *many* paths to get the overall system behavior
- Components are only partially known (“gray boxes”)



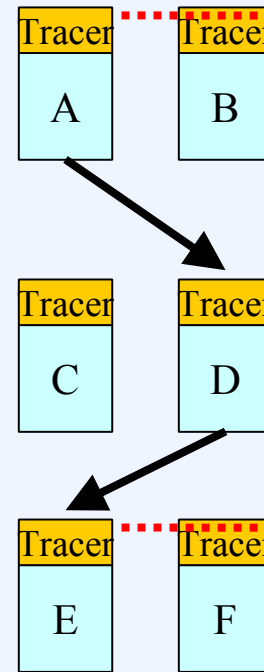
Runtime Paths

- Instrument code to dynamically trace requests through a system at the component/peer/sensor level
 - record call path + the runtime properties
 - e.g. components, latency, success/failure, and resources used to service each request
- Use statistical analysis to detect and diagnose problems
 - e.g., data mining, machine learning, etc.
- Runtime analysis tells you how the system is *actually* being used, not how it *may* be used
- Complements existing micro analysis tools

Architecture

- Tracer
 - Tags each request with a unique ID, and carries it with the request throughout the system
 - Report observations (component name + resource + performance properties) for each component
- Aggregator + Repository
 - Reconstructs paths and stores them
- Declarative Query Engine
 - Supports statistical queries on paths
 - Data mining and machine learning routines
- Visualization

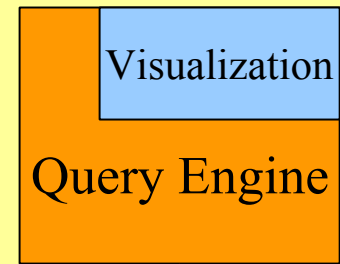
request



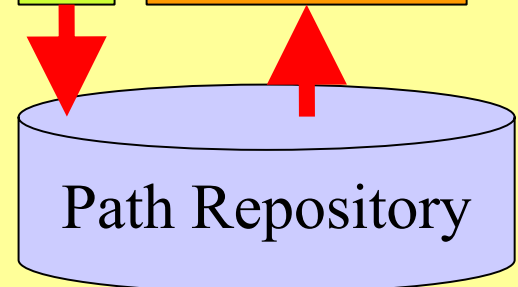
Reusable Analysis Framework



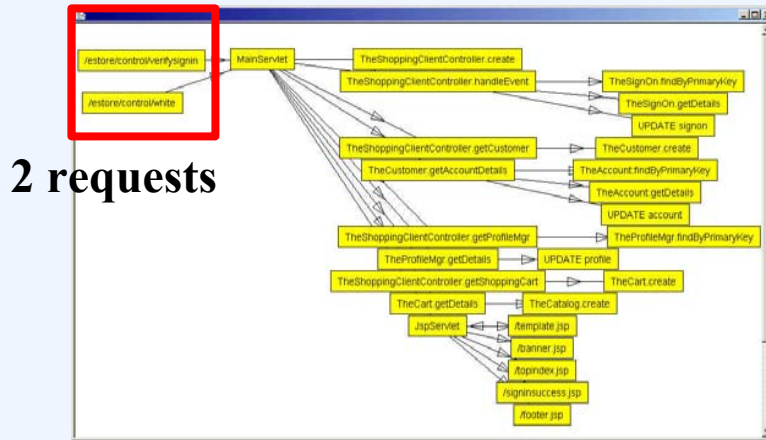
Developers/
Operators



Aggregator



Inferring System Structure



2 requests

- Key idea: paths directly capture application structure

Request types

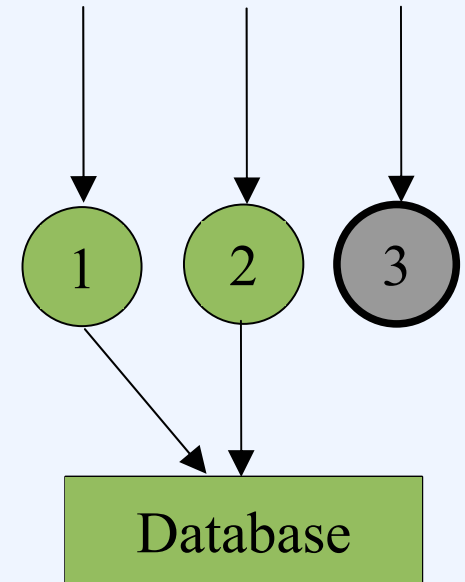
Database tables

Request types	Product	Item	Signon	Account	Bannerdata	Profile	Sequence	Inventory
/verifysignin	R	R	R	R		R		
/cart	R	R			R			R/W
/product	R	R						
/commitorder	R	R					R/W	W
/category	R							
/language	R	R						
/search	R	R			R			
/productdetails	R	R						R/W
/main					R			
/validatenewaccount			R	R		R		
/checkout								W

- Key idea: paths associate requests with internal state
- Track shared state across requests

Anomalies as Likely Failures

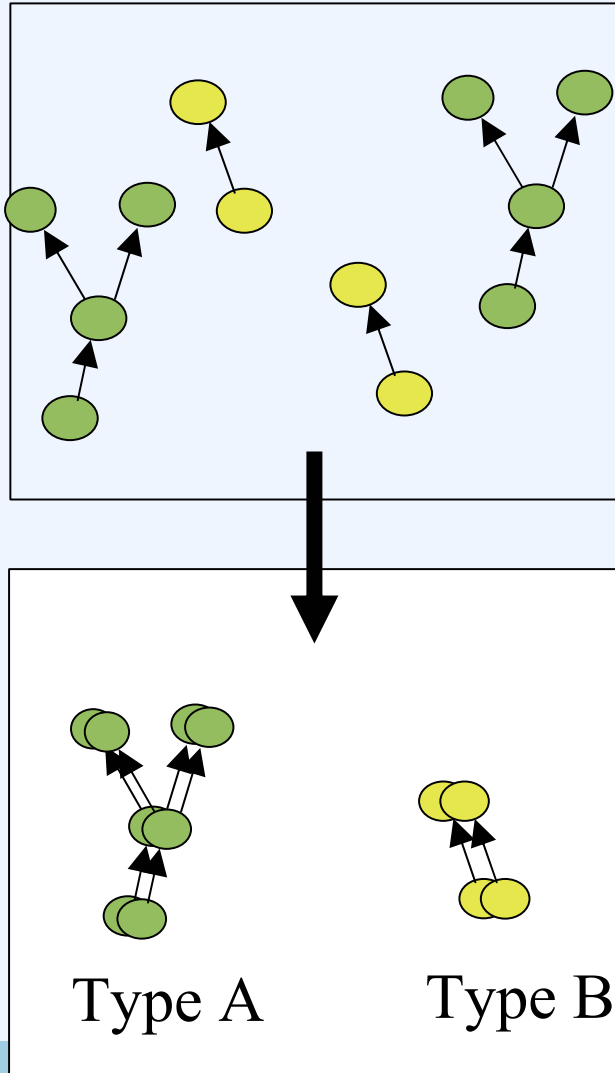
- What is an anomaly?
 - Paths: deviant *structures* or latencies
 - Components: performance/behavior variation
 - Compared to historical norms, or current peers
- Generic method of detecting likely errors



Component
Behavior Variation



Detecting Anomalies in Paths



1. Generate path traces from observations
2. Separate paths by request type
3. Cluster similar structures together
 - 4a. Peer comparison: differences between clusters
 - But, differences are often normal
 - 4b. Historical comparison: compare number, structure and size of clusters to history

Future Directions: P2P and Sensors

- **Key idea: violation of macro invariants are signs of buggy implementation or intrusion**
- Message paths in P2P and sensor networks
 - a *general mechanism* to provide visibility into the collective behavior of multiple nodes
 - micro or static approaches by themselves don't work well in dynamic, distributed settings
 - e.g. algorithms have upper bounds on the # of hops
 - Although hop count violation can be detected locally, paths help identify nodes that route messages incorrectly
 - e.g. detecting nodes that are slow or corrupt msgs



Sensor Networks – Network Topology

- Use message paths to infer network topology and membership
 - directed messaging may reduce resource consumption
- Coping with limited bandwidth
 - each message records a subset of the nodes
 - statistically reconstruct the paths

Conclusion

- Macro analysis fills the need when monitoring and debugging systems where local context is of insufficient use
- Runtime path-based approach dynamically traces request paths and statistically infer macro properties
- A shared analysis framework that is reusable across many systems
 - Simplifies the construction of effective tools for other systems and the integration with recovery techniques like RR
- <http://pinpoint.stanford.edu>
 - Paper includes a commercial example from Tellme! (thanks to Anthony Accardi and Mark Verber)