Comparison of Traces to Diagnose Performance Variations

Presented by François Doray
Introduction | Motivation

- Distribution of the duration of remote procedure calls

- Distribution of request time for 2 different versions of a Web service
Possible causes of performance variation:

- Different requests.
- Different binary (application or external library).
- Different code path (e.g. garbage collection).
- Different system load (CPU, memory, disk, network).
- Different system configuration.
- Different resource sharing.
- ...
Introduction | Motivation

- **Trace Viewer:**
  - Huge amount of information not related to the observed problem.

- **Trace Filters:**
  - Rules written for a specific problem.
Automatically identify the root cause of a performance variation between multiple executions of the same task by comparing traces.
TraceDiff: Visualization of a comparison between function call traces.
(Trümper and al., 2013)

Matching instructions between different versions of the same code using the dynamic data dependence graph.
(X. Zhang and R. Gupta, 2005)
Literature

- **Spectroscope / Magpie / Pinpoint:** Extract and *cluster distributed request flow graphs.*
  
  (R. R. Sambasivan and al., 2011)

- **Compare executions on different OS using high-level concepts** to detect attacks.

  (A. Hamou-Lhadj and al, 2013)
Dapper: Query identifier recorded with each event.  
(B.H. Sigelman and al., 2010)

Magpie and TraceCompass critical path: 
Control flow retrieved using a model based on system events.  
(R. Isaacs and al., 2004) (F. Giraldeau, 2014)
Associate Events to Tasks | Solution

- Abstract control flow builder that can find dependencies between thread segments using:
  - Low-level kernel events.
  - User-space events.

![Diagram showing event flow between clone, wait, Sched_wakeup, and exit]

 Franco 10/12/2014
The second “wait” system call in the third execution is abnormal.
Dynamic Programming
\[ O(n^2) \]

Dynamic Programming, by level of call stack
(M. Weber, 2012)

Depth-First Search and Dynamic Programming
(R. R. Sambasivan, 2011)
Events Matching | Solution
Remove irrelevant states: pre-empted, interrupted.
Collapse loops using a sliding window.
Automatically match unambiguous events.
Slice the graph at the position of matched events.
Apply the dynamic programming algorithm on depth-first search traversals of the sub-graphs.

Events Matching | Solution
\( \chi^2 \) hypothesis test to check whether invariants are respected.

(M.A. Munawar, 2008)

Control chart (Shewhart) to detect machines whose performance deteriorates.

(T.H. Nguyen, 2012)

Mogul at Netflix Automatic correlation of metrics.

(B. Gregg, 2014)
Compare Metrics | Solution

- Distribution View
- Distribution Comparison View
Demo
Road Ahead | Automatic Segmentation

- Automatically find **recurrent patterns**.

  ```
  open [var/log/app/log.txt]
  write
  close
  ```

- Compare the **frequency of occurrence** of recurrent patterns in different traces (different machines, different days).
Conclusion

Objective:

Automatically identify the root cause of a performance variation between multiple executions of the same task by comparing traces.