Real-time Application Monitoring and Diagnosis for Service Hosting Platforms of Black Boxes

Huadong Liu (U. of Tennessee)
Hui Zhang, Rauf Izmailov, Guofei Jiang,
Xiaoqiao Meng (NEC Labs America)

Presented by: Hui Zhang
Motivation
SRAMD architecture
Application component dependency discovery
Evaluation
Conclusions
Motivation

Service hosting systems
- Web farms, service-oriented utility computing networks, Peer-to-Peer service composition based computing grids, …

Service management
- Fault diagnosis, capacity planning, performance analysis, impact analysis, etc.

Challenges
- Application components are usually delivered as black-boxes w/o sufficient instrumentation
- The huge amount of logging information in large-scale systems makes real-time monitoring and debugging unrealistic with a centralized approach
An intuition of the SRAMD Art

Source: www.pictureMOSAICs.com

©NEC Laboratories America
SRAMD: an extensible tool that is
- easy to deploy
- scalable, and
- able to effectively profile the intricate dependency relationships among interacting application components seen as black boxes.

Our approach
- uses low level packet traces instead of high level event traces to get insight into application components
- Has end-system instrumentation for close observation on the correlation between application performance and local resource utilization, and for enabling a rich set of queries for diagnosis
- understands the overall system/application behavior and performance by aggregating and correlating summarizations from distributed components
SRAMD in Operation

An extensible framework for application topology discovery, capacity planning and performance debugging

An application level passive resource monitor with active summarization

©NEC Laboratories America
The SRAMD Controller

- **Collector**
  - passively collects summarization data from distributed monitors through UDP.

- **Aggregator**
  - retrieves, validates information blocks available in the repository, and organizes them into per-application groups.

- **Visualizer**
  - constructs in-memory DOT files [DOT] using outputs from the aggregator and calls the Grappa [Grappa] to visualize application topologies enriched with component traffic statistics and causal probabilities.

- **Diagnosis**
  - generates probing requests to related monitors with operator interaction to get detailed information about application components and to isolate possible bottlenecks for performance debugging.
The SRAMD Controller snapshot
The SRAMD Monitor

- Periodically probe for CPU, memory and disk usage of every registered application component.
- Passively capture network traffic and associate captured packets to registered application components,
- Actively calculate useful local application statistics and dependencies from packet traces
- Temporarily perform diagnosis tasks on-demand to assist performance diagnosis and debugging.
Given two application components A and B in the system, we want to discover the following real-time dependency relationships between A and B during a time interval:

- are the input requests of one components caused by another one (directly or indirectly)? and in what percentage if yes?
Local Dependency Discovery (LDD)

Find IDs of peer application components that local ones talked to in the last report interval. Every SRAMD monitor sends a list of \((\text{LocalPort}, \text{AppCompID})\) to the monitor at every hosting server that the communicating application components are running on.

Count the number of requests (including nesting requests) between application components and calculate the probability of their causal dependency.

Although requests appear to be nested by accident, if the same nesting relationship appears with a high probability, it is highly possible that the nesting represents a causal dependency of application components.
Dealing with persistent connections and connectionless communications

- Traffic Regulation based Component Dependency Discovery (TRCDD)
  - Divert socket based traffic regulation. Under investigation.

![Diagram showing traffic regulation and request flow between A->B and B->C over time T.](image-url)
Evaluation: SRAMD overhead (1)

- Experiment setup

![Diagram showing experiment setup with blocks labeled as Sender thrulay, SRAM Controller, UDP Packets over giga ethernet, Receiver thrulayd, SRAM Monitor, and Intel 2.8GHz SMP.]
CPU overhead of the SRAMD monitor with bulk UDP traffic using different packet sending rates and packet sizes.
Evaluation: SRAMD overhead (3)

- CPU overhead of packet-application matching and sniffing
  - data rate 100Mb/s and packet size 1500 Bytes.

<table>
<thead>
<tr>
<th>Association Probability</th>
<th>1/250k</th>
<th>0.01</th>
<th>0.02</th>
<th>0.03</th>
<th>0.04</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Overhead %</td>
<td>4.22</td>
<td>5.53</td>
<td>6.15</td>
<td>7.12</td>
<td>7.86</td>
</tr>
</tbody>
</table>
Evaluation: LDD algorithm (1)

Experiment setup

Logic view

physical view
Evaluation: LDD algorithm (2)

- Causal probability as observed on application server A1 with different number of concurrent clients
Conclusions and Future Work

- An unobtrusive application-level monitoring and diagnosis tool that does not make any assumptions about the traced applications.
- Two schemes to infer dependency relationships of application components in different scenarios.
- An initial assessment of the quality and overhead of application-level packet tracing and an evaluation of the statistical dependency discovery scheme.
- Possible extensions
  - A kernel module to obtain per-application disk read / write statistics
  - Application of data mining techniques to packet traces
Questions?
Calculate Response Time from Traces

a

b

c

©NEC Laboratories America