Synoptic: Summarizing System Logs with Refinement

Sigurd Schneider
Ivan Beschastnikh
Slava Chernyak
Michael D. Ernst and Yuriy Brun

Saarland University
University of Washington
Google, Inc
University of Washington

Sunday, October 3, 2010
Synoptic overview

- Coarsening \([1,2]\) is standard practice
- Inefficient
- Inaccurate

Coarsen by merging nodes

Input

Output

[1] BIERMANN et al.,

[2] LORENZOLI et al.,
ICSE 2008

Ivan Beschastnikh
U. Washington

Sunday, October 3, 2010
**Synoptic overview**

- uses refinement for higher efficiency
- preserves mined log invariants for accuracy

**Inputs:**
- Log of events
- Relation over events
- Events to event types map
- Invariant mining granularity

**Refine by splitting nodes**

**Input**

**Output**

---

**Ivan Beschastnikh**

**U. Washington**

**Sunday, October 3, 2010**
Talk outline

• Synoptic Overview

• Synoptic’s Design
  • From logs to graphs
  • Graph coarsening and refinement
  • Guiding exploration with log invariants
  • The full picture

• Evaluation

• Related Work
From logs to graphs

- Given a log of events and an event relation generate an event graph

<table>
<thead>
<tr>
<th>time</th>
<th>src</th>
<th>dst</th>
<th>data</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>propose 0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>2</td>
<td>propose 0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>abort 0</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>0</td>
<td>commit 0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>tx-abort 0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>2</td>
<td>tx-abort 0</td>
</tr>
</tbody>
</table>
From logs to graphs

- To compact such graphs we must relate events
- Require the user to specify a mapping from log events to event types
- Definition depends on the analysis
- Induces a partition graph

<table>
<thead>
<tr>
<th>src</th>
<th>dst</th>
<th>data</th>
<th>event type</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>*</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

2PC Mapping 1

<table>
<thead>
<tr>
<th>src</th>
<th>dst</th>
<th>data</th>
<th>event type</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>0</td>
<td>*</td>
<td>to-0</td>
</tr>
<tr>
<td>0</td>
<td>*</td>
<td>*</td>
<td>from-0</td>
</tr>
</tbody>
</table>

2PC Mapping 2
From logs to graphs

Event Graph

0, 1, propose 0
0, 2, propose 0
1, 0, abort 0
2, 0, commit 0
0, 1, tx-abort 0
0, 2, tx-abort 0

2PC Mapping 1

propose
propose
abort
commit
tx-abort

2PC Mapping 2

from-0
to-0
from-0
from-0
From logs to graphs
Talk outline

• Synoptic Overview

• Synoptic’s Design
  • From logs to graphs
  • Graph coarsening and refinement
  • Guiding exploration with log invariants
  • The full picture

• Evaluation

• Related Work
Graph coarsening and refinement

Coarsening

Refinement
Graph refinement: Bisim

1. Initial Partitioning

2. tx-commit

3. tx-abort

4. tx-commit

commit

abort

tx-commit

commit

abort

tx-commit

commit

abort

tx-commit

commit

abort

tx-commit

commit

tx-abort
Graph coarsening and refinement

More Concise

More Accurate

Coarsening

Refinement
Graph coarsening and refinement

• Exploration policy questions
  • Which graph to choose at each step, if multiple options exist?
  • When to terminate?
  • How to preserve important log properties?
Talk outline

• Synoptic Overview

• Synoptic’s Design
  • From logs to graphs
  • Graph coarsening and refinement
  • Guiding exploration with log invariants
  • The full picture

• Evaluation

• Related Work
Log invariants

• Logs often have patterns, which reflect system invariants
  • Temporal invariants
  • Structural invariants

• Synoptic does not assume that the user knows these invariants, instead it mines them
  • This better supports the goal of enabling log exploration
Log invariants

• Synoptic mines three kinds of log invariants:

<table>
<thead>
<tr>
<th>Invariant</th>
<th>LTL formula</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>x AlwaysFollowedBy y</td>
<td>□(x → ♦y)</td>
<td>liveness</td>
</tr>
<tr>
<td>y AlwaysPrecededBy x</td>
<td>♦y → ¬y U x</td>
<td>safety</td>
</tr>
<tr>
<td>x NeverFollowedBy y</td>
<td>□(x → □¬y)</td>
<td>safety</td>
</tr>
</tbody>
</table>
## Log invariants

- Synoptic mines three kinds of log invariants:

<table>
<thead>
<tr>
<th>Invariant</th>
<th>LTL formula</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>x AlwaysFollowedBy y</td>
<td>x → y</td>
<td>liveness</td>
</tr>
<tr>
<td>y AlwaysPrecededBy x</td>
<td>x → y</td>
<td>safety</td>
</tr>
<tr>
<td>x NeverFollowedBy y</td>
<td>x → y</td>
<td>safety</td>
</tr>
</tbody>
</table>
Log invariants: satisfiability

All Satisfied

Some Satisfied

None Satisfied

Most concise model satisfying all invariants
Guiding exploration

• Exploration conditions:
  • Given a choice of two splits, perform the split that satisfies a previously unsatisfied invariant
  • Given a choice of merges, perform the merge that maintains all the log invariants

• Termination condition:
  • A log summary must preserve all log invariants
Talk outline

• Synoptic Overview
• Synoptic’s Design
  • From logs to graphs
  • Graph coarsening and refinement
  • Guiding exploration with log invariants
  • The full picture
• Evaluation
• Related Work
BisimH

- Hybrid Bisim
  - Combines refinement (Bisim) with coarsening (kTail)
All Invariants Satisfied

Some Invariants Satisfied

Synoptic finds the local optimum, not the global one.

Refine

Coarsen

Refine

Refine

Sunday, October 3, 2010
Talk outline

• Synoptic Overview
• Synoptic’s Design
• Evaluation
  • case studies
  • accuracy
  • efficiency
• Related Work
Evaluation: two-phase commit

Anomalous

Correct
Evaluation: Peterson algorithm

• Leader election in a uni-directional ring network

• Nodes are issued a random id and start as active

• In each round of the protocol, an active node becomes a relay by comparing its ids with that of its two predecessors

• The last active node to remain is the leader
Evaluation: Peterson algorithm

relay
recv
send
relay
recv
send
active
recv
send
leader
recv
leader
Evaluation: Reverse Traceroute
Evaluation: Reverse Traceroute

(a) Initialization
- Issuing traceroutes
- Connecting to tr_server
- Issuing spoofed recordroutes
- Checking for TRs to source
- Initializing RR VPs
- Rev Seg found
- Connecting to controller

(b) Resuming Operation
- Initializing RR VPs
- Found Recent Spoof To Use
- Issuing spoofed recordroutes
- RR VPs left to try
- using nonSpoofed Issuing recordroutes
- TS adjacents left to try
- Adjacents for + are
- Issue spoofed timestamps
- Current hop unresponsive
- No adjacents found
- Issuing timestamps
- Issuing spoofed recordroutes

(c) Record-route Option
- Using nonSpoofed
- RR VPs left to try
- Issuing recordroutes
- Rev Seg found
- Connection to adjacency server
- TS adjacents left to try
- No Adjacents found
- Issuing traceroutes
- Current hop TS unresponsive

(d) Timestamp Option
- Issuing timestamps
- Current hop unresponsive
- No adjacents found
- Issuing spoofed timestamps

(e) Success
- Reached
- Issuing spoofed timestamps
- Current hop TS unresponsive
- Issuing recordroutes
- Issuing spoofed recordroutes
- Adjacent for + are

KATZ-BASSETT et. al. NSDI 2010

Sunday, October 3, 2010
Evaluation: accuracy

- BisimH
- More accurate than kTail [1], a popular coarsening algorithm

<table>
<thead>
<tr>
<th></th>
<th>Peterson nodes</th>
<th>Peterson accurate</th>
<th>Two-phase commit nodes</th>
<th>Two-phase commit accurate</th>
</tr>
</thead>
<tbody>
<tr>
<td>kTail (k=1)</td>
<td>5</td>
<td>no</td>
<td>2</td>
<td>no</td>
</tr>
<tr>
<td>kTail (k=2)</td>
<td>14</td>
<td>no</td>
<td>7</td>
<td>yes</td>
</tr>
<tr>
<td>BisimH</td>
<td>9</td>
<td>yes</td>
<td>7</td>
<td>yes</td>
</tr>
</tbody>
</table>

Evaluation: efficiency

- BisimH
  - slower than coarsening without invariants
  - faster than coarsening with invariants

<table>
<thead>
<tr>
<th></th>
<th>Peterson</th>
<th>Reverse Traceroute</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>nodes</td>
<td>time</td>
</tr>
<tr>
<td>kTail with invariants</td>
<td>12</td>
<td>3,743s</td>
</tr>
<tr>
<td>BisimH</td>
<td>9</td>
<td>9s</td>
</tr>
</tbody>
</table>
Evaluation: efficiency

![Graph showing time (s) vs input events for different processes: total, mine invariants, refinement, coarsening, extraction. The graph indicates an increase in time as the number of input events increases.](image-url)
Related work

- Distributed systems
  - Process tracing [Magpie, X-Trace]
  - Visualizing Hadoop logs [Mochi, SALSA]
- Temporal log properties [Perracotta]
- Coarsening
  - k-Tail [1] and GK-Tail [2]
- Partition refinement [3]

Synoptic

- Uses graph refinement for efficiency
- Preserves key log invariants for accuracy
- A multi-purpose tool intended for exploration
- Can improve developers’ system understanding

http://code.google.com/p/synoptic/