Toward Automatic Policy Refinement in Repair Services for Large Distributed Systems

M. Goldszmidt, M. Budiu, Y. Zhang, M. Pechuk
Microsoft
The problem we are addressing

Cluster

Monitoring system

Logs

Repair service

Signals

Repair actions

State

Policy manager

Policies

Analysis

Policy refinement
The repair service

Watchdogs:
Asynchronously monitoring machines and sending signals

Each machine has a state associated with it

E.g.: healthy, probation, faulty, rebooted_once, etc.

State transitions are regulated by an automaton.
A signal or a repair action will cause a state transition

A policy is a function from State to Repair Action

E.g.:
- If probation do_nothing.
- If rebooted_once reboot.
- If dead call tier_1 operator.
Logs

Log consists of 3 months of data collected from ~2k machines

Reason for transition e.g. = e8382

Time of the event 2009-02-21 02:09:07
Research questions

Given the data in the logs:
1. Estimate the ‘effectiveness’ of a repair action
   What is a “successful” repair action?
2. Suggest alternative (better) policies (without intervention)
Effectiveness and success

- Effectiveness $\rightarrow$ time that a machine is ‘usable’
- Estimate the survival curve of the repair action

Successful repair = threshold on $P$ of survival and time
Modeling successful repairs

Automatically find a function from watchdog-signals to success

Machine learning to the rescue:
classification with feature selection.
Logistic regression with L1 regularization
Models of success

# selected signals: 9
CV BA: 0.872

CV confusion matrix:

<table>
<thead>
<tr>
<th></th>
<th>below</th>
<th>above</th>
</tr>
</thead>
<tbody>
<tr>
<td>pred below</td>
<td>89</td>
<td>14</td>
</tr>
<tr>
<td>pred above</td>
<td>11</td>
<td>71</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>coeffs</th>
<th>ind</th>
<th>threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>e50202</td>
<td>-0.79</td>
<td>0.965</td>
<td>0.00</td>
</tr>
<tr>
<td>e8240</td>
<td>-0.89</td>
<td>0.942</td>
<td>0.00</td>
</tr>
<tr>
<td>e8383</td>
<td>0.31</td>
<td>0.692</td>
<td>1.00</td>
</tr>
<tr>
<td>e8506</td>
<td>-0.84</td>
<td>0.861</td>
<td>0.00</td>
</tr>
</tbody>
</table>

185 samples with 42 signals
Refining policies

A policy is a function from State and Signal to Repair Action

QoS, Availability costs
Money, QoS, Availability costs

Cost increase
Data processing (with Artemis)

1. Use regular expression to extract segments of data
2. Extract duration and censoring events
3. Estimate survival curves
4. Define success
5. Extract the signals before the repair action
6. Induce models of success/fail
7. Present relevant signals
Data visualization (with Artemis)
Results

- Comparing different datacenters
  - Statistical tests on the different survivability curves
  - Visualization (correlation graphs)
- Models for different repair actions
The bad sensor case

How come 1 signal was predicting with 98% accuracy the failure to repair?

Further investigation ➞ faulty sensor!!

New models (3 months after the fix) have a mixture of many signals and E8382 appears as evidence for success...
Faulty repair procedure

Snippet of the T1-REPAIR model

<table>
<thead>
<tr>
<th></th>
<th>coeffs</th>
<th>ind</th>
<th>threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>-0.79</td>
<td>0.965</td>
<td>0.00</td>
</tr>
<tr>
<td>S2</td>
<td>-0.89</td>
<td>0.942</td>
<td>0.00</td>
</tr>
<tr>
<td>S4</td>
<td>-0.84</td>
<td>0.861</td>
<td>0.00</td>
</tr>
</tbody>
</table>

S2 is indicative of an easy fix... Why was not effective?

Bug in the repair instructions.... Fixed!

What about S1 and S4?
Final Remarks

• Models directed the debugging of the repair service.
  – Signals that are strong indications of failed repair
  – Signals that are irrelevant

• In two weeks the results helped improve a system that was “hand-tuned” during 6 months

• Further automate the whole workflow

• Induce models of correlated watchdogs

• Correlate to performance data