Predicting Global Failure Regimes in Complex Information Systems

Kevin Mills, NIST
July 9, 2015
Project Research Goals

- Develop *design-time methods* that system engineers can use to detect existence and causes of costly failure regimes prior to deployment

- Develop *run-time methods* that system managers can use to detect onset of costly failure regimes in deployed systems, prior to collapse
Topics

• Some past results on design-time methods

• Example → Applying one design-time method to seek failure scenarios in a cloud system

• Ongoing work on run-time methods

• Where to find more information
Some Past Results

State-space reduction techniques and their application to clouds

Directed and self-directed search techniques and their application to clouds

- Identifying Failure Scenarios in Complex Systems by Perturbing Markov Chains
- Using Spectral Methods to Streamline Search for Failure Scenarios
- Combining GAs & Simulations to Search for Failure Scenarios in System Models

Kevin Mills, NIST
Method: Genetic Algorithm (GA) steers a population of simulators to search for parameter combinations that lead to system failure.

**Model Simulators**

- **Model Parameter Specifications**
  - List of parameters and for each parameter a MIN, MAX and precision.

**Growing Collection of Tuples:**

\[
\{\text{Generation, Individual, Fitness, Parameter 1 value, \ldots, Parameter N value}\}
\]

**Selection based on Anti-Fitness**

**Recombination & Mutation**

**Principal Components Analysis, Clustering, ...**

**MULTIDIMENSIONAL ANALYSIS TECHNIQUES**

In our following example, we use the Koala cloud simulator, and we define *anti-fitness* as the proportion of users not served, and we use differential probability analysis on the collection of tuples.

Kevin Mills, NIST
## Summary of *Koala* Parameters to Search Over

### Test Case – Can GA find VM Leakage *due to message loss and lack of orphan control?*


<table>
<thead>
<tr>
<th>Model Element</th>
<th>Behavior</th>
<th>Structure</th>
<th>Asymmetry</th>
<th>Failure</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td>28</td>
<td>2</td>
<td>4</td>
<td>0</td>
<td>34</td>
</tr>
<tr>
<td>Cloud Controller</td>
<td>21</td>
<td>4</td>
<td>5</td>
<td>0</td>
<td>30</td>
</tr>
<tr>
<td>Cluster Controllers</td>
<td>11</td>
<td>5</td>
<td>3</td>
<td>0</td>
<td>19</td>
</tr>
<tr>
<td>Nodes</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>14</td>
<td>20</td>
</tr>
<tr>
<td>Intra-Net/Inter-Net</td>
<td>4</td>
<td>11</td>
<td>2</td>
<td>9</td>
<td>26</td>
</tr>
<tr>
<td>Totals</td>
<td>70</td>
<td>22</td>
<td>14</td>
<td>23</td>
<td>129</td>
</tr>
</tbody>
</table>

Average # values per parameter is about 6, so search space is $\approx 6^{129}$ i.e., $\approx 10^{100}$ scenarios are possible

- adapted 125-parameter *Koala* IaaS simulator to be GA controllable
- added 4 *Koala* parameters to turn on/off logic to control (a) creation orphans, (b) termination orphans, (c) relocation orphans and (d) administrator actions

---

Kevin Mills, NIST
**Koala** GA Search over 500 Generations

**GENETIC ALGORITHM CONTROL PARAMETERS**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generations</td>
<td>500</td>
</tr>
<tr>
<td>Population Size</td>
<td>200 Individuals</td>
</tr>
<tr>
<td>Elite Per Generation</td>
<td>16 Individuals</td>
</tr>
<tr>
<td>Reboot After</td>
<td>200 Generations</td>
</tr>
<tr>
<td>Selection Method</td>
<td>Stochastic Uniform Sampling</td>
</tr>
<tr>
<td># Crossover Points</td>
<td>3</td>
</tr>
<tr>
<td>Mutation Rate</td>
<td>0.001 ≤ Adaptive ≤ 0.01</td>
</tr>
</tbody>
</table>

**Average Anti-Fitness**

- Average anti-fitness oscillates around 65% of users not served

**Maximum Anti-Fitness Discovered**

- For Koala simulator, failure scenarios appear within first 100-200 generations

**Frequency Distribution of Anti-Fitness**

- Only 8% of scenarios are duplicate (equals elite-selection percentage)
- 84% of scenarios exhibit anti-fitness ≥ 0.50

Kevin Mills, NIST
Differential Probability Analysis

Let $C$ be the set of collected tuples, each containing a vector of parameter value (PV) pairs and a corresponding anti-fitness value, $f$

Segment $C$ into high-pass ($H$) and low-pass ($L$) subsets, where:

$H = \{ x \in C \mid f_x > 0.70 \}$ and $L = \{ x \in C \mid f_x < 0.15 \}$

For each PV estimate the probability of occurrence in $H$ and $L$:

$P(PV_i \mid f > 0.70) = PV_i \in H \setminus H$ and $P(PV_i \mid f > 0.15) = PV_i \in L \setminus L$

Then compute the estimated differential probability:

$D = P(PV_i \mid f > 0.70) - P(PV_i \mid f < 0.15)$

Plot $D$ for each PV pair

Outliers contributing to failure scenarios

PV pairs sorted by decreasing $D$

Outliers contributing to success scenarios

PVs exerting little influence on success or failure

Kevin Mills, NIST
Analysis of Results from *Koala* GA Search 1 – 500 Generations

Seeking Known Failure Scenario – search duration 30 days

![Graph showing analysis results](image)

**Known failure scenario found,**
*but under previously unknown circumstances*

$D$ (y-axis) for 684 PV pairs (x-axis) for first GA search—outlier PV pairs labeled.

Kevin Mills, NIST
Ongoing Work: Do published findings on the spread of congestion hold for realistic network models?

Abstract Network Model

Add 7 Realism Factors

34 Valid Combinations

Y: Network Disruption

X: Increasing Load

Kevin Mills, NIST
To Learn More

Project Team (the core four)

- **Kevin Mills**, computer scientist – kmills@nist.gov
- **Chris Dabrowski**, computer scientist – cdabrowski@nist.gov
- **Jim Filliben**, statistician – jfilliben@nist.gov
- **Sandy Ressler**, information visualization specialist – sressler@nist.gov

Project Page