Hi!

- Aeronautics
- Electronics
- Development
- Operations
- Data science
- Grey beard
What’s Complex?

• Multiple interacting components
• Emergent behavior
  • Dynamics occur which were not intentionally designed in
• Usually larger than a single piece of software on a single host
Incident Response

- Project Management
- Temporary Mitigation
- External Communication
- Diagnosis

This Talk

@terranmelconian
Applicability

- To any system which once worked or sometimes works and then does not
- Which has been or can be observed and measured in both states
Core Principles

• Question beliefs
  • You believed the system worked. It doesn’t.
  • Your other beliefs are not magically better.

• Divide the problem space
  • Binary search beats linear search

• Fast measurements first
  • Look up a measurement we have: 5 min
  • Write, review, deploy, wait for peak: 24+ hours
Teaching the Skill

• Uncorrelated with software development
• Contrast actual states - avoid distracting contrasts with idealized “shoulds”
  • smallest set of changes to reproduce
• Enforce shared, written records and diagrams
• Practice in advance of critical failures
Domains

- Computer systems
- Your car
- Health
- Leaks in your house

- Almost any kind of system which *used to work*
Start with Symptom

- Errors
- Slow Response Time
- Server crash
- GC hell
- Bad data
Mechanism

- Draw a tree of possible causality
  - Rooted at symptom
  - Possible causes point to root
  - Causes of those causes and so on
- Take organized data samples
  - Add columns as you expand your analysis
Example: GC Hell

- Gradual Overload
- Killer Request
- WTF?
# GC Hell Data

<table>
<thead>
<tr>
<th>Time</th>
<th>GC CPU %</th>
</tr>
</thead>
<tbody>
<tr>
<td>13:46 (failure)</td>
<td>100%</td>
</tr>
<tr>
<td>13:40</td>
<td>3%</td>
</tr>
<tr>
<td>13:00</td>
<td>4%</td>
</tr>
<tr>
<td>12:00</td>
<td>3%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time</th>
<th>GC CPU %</th>
</tr>
</thead>
<tbody>
<tr>
<td>13:46 (failure)</td>
<td>100%</td>
</tr>
<tr>
<td>13:40</td>
<td>92%</td>
</tr>
<tr>
<td>13:00</td>
<td>34%</td>
</tr>
<tr>
<td>12:00</td>
<td>3%</td>
</tr>
</tbody>
</table>
GC Hell Step 2

- Slower Response
- More Requests
- WTF?

Gradual Overload → Killer Request → GC Hell

13
## Step 2 Data

<table>
<thead>
<tr>
<th>Time</th>
<th>GC CPU %</th>
<th>Response Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>13:46</td>
<td>100%</td>
<td>16423 ms</td>
</tr>
<tr>
<td>13:40</td>
<td>92%</td>
<td>2473 ms</td>
</tr>
<tr>
<td>13:00</td>
<td>34%</td>
<td>844 ms</td>
</tr>
<tr>
<td>12:00</td>
<td>3%</td>
<td>192 ms</td>
</tr>
</tbody>
</table>

**GC Hell**

- Gradual Overload
- Killer Request

**WTF?**

- Slower Response
- More Requests
GC Hell Step 2 bis

- Slower Response
- More Requests
- Killer Request
- Gradual Overload
- WTF?
- GC Hell
- WTF?

15
# Step 2 Data bis

<table>
<thead>
<tr>
<th>Time</th>
<th>GC CPU %</th>
<th>Response Time</th>
<th>Req/Min</th>
</tr>
</thead>
<tbody>
<tr>
<td>13:46</td>
<td>100%</td>
<td>16423 ms</td>
<td>3</td>
</tr>
<tr>
<td>13:40</td>
<td>92%</td>
<td>2473 ms</td>
<td>352</td>
</tr>
<tr>
<td>13:00</td>
<td>34%</td>
<td>844 ms</td>
<td>1630</td>
</tr>
<tr>
<td>12:00</td>
<td>3%</td>
<td>192 ms</td>
<td>850</td>
</tr>
</tbody>
</table>
GC Hell Step 3

- Bad Balance
- More Users
- WTF?
- Slower Response
- More Requests
- Gradual Overload
- Killer Request
- GC Hell
- WTF?
## Step 3 Data

<table>
<thead>
<tr>
<th>Time</th>
<th>Server A</th>
<th>Server B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GC CPU %</td>
<td>Response Time</td>
</tr>
<tr>
<td>13:46 (failure)</td>
<td>100%</td>
<td>16423 ms</td>
</tr>
<tr>
<td>13:40</td>
<td>92%</td>
<td>2473 ms</td>
</tr>
<tr>
<td>13:00</td>
<td>34%</td>
<td>844 ms</td>
</tr>
<tr>
<td>12:00</td>
<td>3%</td>
<td>192 ms</td>
</tr>
</tbody>
</table>
GC Hell Step 4

- Bad Balance
- More Users
- WTF?

Uneven balance at other times?
Uneven balance across other servers?

- Slower Response
- More Requests
- Gradual Overload

WTF?
Killer Request
GC Hell
Anti-Example 1

Add More Servers

Change Garbage Collector

Add a Rate Limit

GC Hell
Anti-Example 2

L1 Cache

That new feature from Thursday

Maybe bad data?

GC Hell
Anti-Example 3

- Slowdown
- Crash
- Network Failure

→ GC Hell
Partitioning

• Do
  • By system component
  • By service
  • By time of checkin/deployment
    • beware, blind to causes which are not a code change

• Don’t
  • By ways to mitigate
  • By listing individual pieces of code
  • By ignoring the information in the symptoms
Extensions

• Weight the tree by prior beliefs and partition weight instead of node count
  • Default first steps such as rolling back release
  • but limit your temptation to repeatedly pursue high-confidence guesses
  • disagreements over weight more likely
• Give tree and process to others for diagnosis
• Plan your logging and dashboards
Summary

• Start with a symptom
• Draw a tree of possible causes
• Take measurements to partition the tree
• Prefer observing to mutating and waiting
• Record all your data in one place
• Suspect everything you believe
Questions?

terran@airnetsim.com       twitter @terranmelconian

slides: from O'Reilly site or http://www.airnetsim.com/terran/