Is your performance analysis approach as cutting edge as your application architecture?

Jon C. Hodgson
APM Subject Matter Expert
Advanced Technology Group, Office of the CTO

@JonQ  @Riverbed  #VelocityConf
See the full video for this presentation at:
Challenge #1: Think Logically, Not Physically
Modern Challenges for APM users

- Increasingly complex architecture
  - “Web-Scale”
  - Microservices

- Dynamic infrastructure
  - Elasticity
  - Transience of OS/app instances

- Abstraction
  - Cloud, containers
Modern Challenges for APM users

- What is my app architecture?
  - Can change minute-to-minute
  - Physical concepts like “servers” make less sense in dynamic environments
  - Impractical to visualize based on physical servers & app instances

- What are my key dependencies?
  - Answering based on physical architecture offers limited value
  - Microservices and shared code libraries require looking deeper than servers & instances

- What should I focus on to have the greatest impact on the business?
  - Interdependency between many transactions across many apps & LOBs
Physical Application Map: Just 2 App Servers
“Simple” App in QA
Traditional Physical Application Map
Challenge #2: Think holistically - Solve overarching problems
Multinational financial services company

30+ Tier 1 Applications

1,000s of different types of transactions

Account details page frequently extremely slow

For months they tried analyzing the slowest transactions and optimizing the slowest methods

The problem persisted
analyzing the **slowest** transactions

optimizing the **slowest** methods

--

analyzing all transactions

optimizing the **overall** slowest methods
Real-World Benefits

1.1 sec
Response Time
Before Optimization

95%
Faster

53ms
Method
Optimized

7 Million
Transactions / Day

62ms
Response Time
After Optimization

2,000
Hours Saved / Day
We knew this could be improved.

We just didn’t realize how bad it was.

How long has it been like this?

Forever.

What else uses this code?

EVERYTHING.
Everything.

30+ Tier 1 Applications

1,000s of different types of transactions

All shared that common monitoring method

All saw significant improvements in performance

Tens of millions of transactions per day
Challenge #3: Leverage True Big Data APM
### TOP URLs

<table>
<thead>
<tr>
<th>URL</th>
<th># TXs</th>
<th>Resp Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>/page/c.jsp</td>
<td>134</td>
<td>8.63</td>
</tr>
<tr>
<td>/dir/a.aspx</td>
<td>792</td>
<td>6.27</td>
</tr>
<tr>
<td>/d.do</td>
<td>1,219</td>
<td>4.92</td>
</tr>
<tr>
<td>/pages/b.jsp</td>
<td>876</td>
<td>3.12</td>
</tr>
<tr>
<td>/dir/e.aspx</td>
<td>12,418</td>
<td>1.07</td>
</tr>
</tbody>
</table>

### TOP SQL

<table>
<thead>
<tr>
<th>SQL</th>
<th># Calls</th>
<th>Resp Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT x FROM y WHERE ...</td>
<td>2,641</td>
<td>2.15</td>
</tr>
<tr>
<td>INSERT INTO z VALUES ...</td>
<td>376</td>
<td>1.97</td>
</tr>
<tr>
<td>SELECT a FROM b WHERE ...</td>
<td>11,847</td>
<td>0.78</td>
</tr>
<tr>
<td>UPDATE c SET ...</td>
<td>192</td>
<td>0.63</td>
</tr>
<tr>
<td>SELECT d FROM e WHERE ...</td>
<td>34,433</td>
<td>0.19</td>
</tr>
</tbody>
</table>
### TOP URLs

<table>
<thead>
<tr>
<th>URL</th>
<th># TXs</th>
<th>Resp Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>/page/c.jsp</td>
<td>134</td>
<td>8.63</td>
</tr>
<tr>
<td>/dir/a.aspx</td>
<td>792</td>
<td>6.27</td>
</tr>
<tr>
<td>/d.do</td>
<td>1,219</td>
<td>4.92</td>
</tr>
<tr>
<td>/pages/b.jsp</td>
<td>876</td>
<td>3.12</td>
</tr>
<tr>
<td>/dir/e.aspx</td>
<td>12,418</td>
<td>1.07</td>
</tr>
</tbody>
</table>

### TOP SQL

<table>
<thead>
<tr>
<th>SQL</th>
<th># Calls</th>
<th>Resp Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT x FROM y WHERE ...</td>
<td>2,641</td>
<td>2.15</td>
</tr>
<tr>
<td>INSERT INTO z VALUES ...</td>
<td>376</td>
<td>1.97</td>
</tr>
<tr>
<td>SELECT a FROM b WHERE ...</td>
<td>11,847</td>
<td>0.78</td>
</tr>
<tr>
<td>UPDATE c SET ...</td>
<td>192</td>
<td>0.63</td>
</tr>
<tr>
<td>SELECT d FROM e WHERE ...</td>
<td>34,433</td>
<td>0.19</td>
</tr>
</tbody>
</table>
### TOP URLs

<table>
<thead>
<tr>
<th>URL</th>
<th># TXs</th>
<th>Resp Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>/page7.c.jsp</td>
<td>134</td>
<td>8.63</td>
</tr>
<tr>
<td>/dir/a.aspx</td>
<td>792</td>
<td>6.27</td>
</tr>
<tr>
<td>/dir/e.aspx</td>
<td>1,119</td>
<td>4.92</td>
</tr>
<tr>
<td>/pages/b.jsp</td>
<td>876</td>
<td>3.12</td>
</tr>
<tr>
<td>/dir/e.aspx</td>
<td>12,418</td>
<td>1.07</td>
</tr>
</tbody>
</table>

### TOP SQL

<table>
<thead>
<tr>
<th>SQL</th>
<th># Calls</th>
<th>Resp Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT x FROM y WHERE ...</td>
<td>2,641</td>
<td>2.15</td>
</tr>
<tr>
<td>INSERT INTO z VALUES ...</td>
<td>376</td>
<td>1.97</td>
</tr>
<tr>
<td>SELECT a FROM b WHERE ...</td>
<td>11,847</td>
<td>0.78</td>
</tr>
<tr>
<td>UPDATE c SET ...</td>
<td>192</td>
<td>0.63</td>
</tr>
<tr>
<td>SELECT d FROM e WHERE ...</td>
<td>34,433</td>
<td>0.19</td>
</tr>
</tbody>
</table>
conflation

noun | con·fla·tion | \kən-ˈflā-shən\

- The practice of treating two distinct concepts as if they were one
- Often produces errors or misunderstandings
- May be an intentional tool of deception by information providers
- May unwittingly occur when assumptions are made by the information consumer
(Unintentional) conflation example

RFI questionnaire for a vehicle manufacturer:

- ✔ 2-Wheeled Vehicles
- ✔ 0-60 in less than 2 seconds
- ✔ Cargo capacity of 10,000 pounds or more
- ✔ Seats 8 People
- ✔ Sunroof
### How Big is your Big Data? (Captured in high-volume production)

<table>
<thead>
<tr>
<th>Sort-of Big Data</th>
<th>True Big Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>microdata captured for <em>some</em> TXs</td>
<td>microdata captured for all TXs</td>
</tr>
<tr>
<td>most/all microdata discarded</td>
<td>long-term retention of microdata</td>
</tr>
</tbody>
</table>

#### DIAGNOSE

**Per-transaction microdata**
- Persisted for ad-hoc analysis

- Client-side EUE w/ Parameters
- Server-side URL w/ Parameters
- Method Call Stack w/ Parameters
- SQL w/ Bind Variables
- Messages
- Web Service calls w/ Methods
- Remote socket calls w/ details
- Wire details (packets)
- Multi-tier stitched transaction map

#### DETECT

**Aggregated macrodata**
- Summary EUE stats
- Summary URL stats
- Summary Method stats
- Summary SQL stats
- Message Queue stats
- Summary Web Service stats
- Summary Network stats
- Aggregate transaction flow map

### On Demand
- Single Transactions
- Triggered TXs slower than X
- Sampled Every Nth Txn
- Complete All Transactions

### False Positives
- High volume
- Bursty TXs
- Low-priority TXs
- TXs dominated by network latency

### True Positives
- High-priority TXs
- Critical business transactions
- TXs with high data volume

---

© 2016 Riverbed Technology. All rights reserved.
Did I mention BIG?

<table>
<thead>
<tr>
<th>Storage</th>
<th>754 GiB of 2 TiB is currently in use</th>
</tr>
</thead>
</table>

- **Detailed Usage**

<table>
<thead>
<tr>
<th></th>
<th>Oldest Available</th>
<th>Latest</th>
<th>Duration</th>
<th>Count</th>
<th>Size On Disk</th>
<th>Percent Of Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>End-to-End Transactions</td>
<td>2016-03-10 04:59</td>
<td>2016-06-02 12:03</td>
<td>84 days 6 hours 4 minutes</td>
<td>3,702,483,721</td>
<td>265.4 GiB</td>
<td>12.704%</td>
</tr>
<tr>
<td>Aggregated Metrics</td>
<td>2016-03-09 16:00</td>
<td>2016-06-02 12:03</td>
<td>84 days 19 hours 3 minutes</td>
<td>n/a</td>
<td>21.1 GiB</td>
<td>1.011%</td>
</tr>
</tbody>
</table>
Challenge #4: Don’t settle for inadequate data
Coarse Sampling: Where’d those spikes go?

100% chance
Catching all 5 spikes

AppInternals: 1 sec data
5x 1-second spikes

Mocked-up examples of other tools:
15 sec samples

71% chance
Missing all spikes

7% chance
Catching a specific spike

0.0001% chance
(1 in 759,375 chance)
Catching all 5 spikes

This is not an exaggeration!

100% chance
Showing all 5 spikes
(but values deflated)
Lightning Example

Please watch the video for this example
Challenge #5: Beware the Flaw of Averages
A classic example of the Flaw of Averages involves the Statistician who drowned crossing a river that was, on average, 3 ft. deep.

Source: The Flaw of Averages: Why We Underestimate Risk in the Face of Uncertainty by Sam L. Savage, with illustrations by Jeff Danziger – http://flawofaverages.com
“What’s the Response Time of MyPage.aspx?”

<table>
<thead>
<tr>
<th>#</th>
<th>Start Time</th>
<th>URL</th>
<th>RespTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>00:00:00.000</td>
<td>MyPage.aspx</td>
<td>3.277</td>
</tr>
<tr>
<td>2</td>
<td>00:00:02.413</td>
<td>MyPage.aspx</td>
<td>3.875</td>
</tr>
<tr>
<td>3</td>
<td>00:00:04.040</td>
<td>MyPage.aspx</td>
<td>2.825</td>
</tr>
<tr>
<td>4</td>
<td>00:00:06.520</td>
<td>MyPage.aspx</td>
<td>69.954</td>
</tr>
<tr>
<td>5</td>
<td>00:00:08.028</td>
<td>MyPage.aspx</td>
<td>35.047</td>
</tr>
<tr>
<td>6</td>
<td>00:00:10.382</td>
<td>MyPage.aspx</td>
<td>4.194</td>
</tr>
<tr>
<td>7</td>
<td>00:00:12.222</td>
<td>MyPage.aspx</td>
<td>5.171</td>
</tr>
<tr>
<td>8</td>
<td>00:00:14.074</td>
<td>MyPage.aspx</td>
<td>4.679</td>
</tr>
<tr>
<td>9</td>
<td>00:00:15.500</td>
<td>MyPage.aspx</td>
<td>3.795</td>
</tr>
<tr>
<td>10</td>
<td>00:00:17.119</td>
<td>MyPage.aspx</td>
<td>5.159</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

9,993 04:02:05.774 MyPage.aspx 3.778
9,994 04:02:07.170 MyPage.aspx 34.376
9,995 04:02:08.433 MyPage.aspx 24.971
9,996 04:02:10.480 MyPage.aspx 4.004
9,997 04:02:12.082 MyPage.aspx 3.552
9,998 04:02:14.869 MyPage.aspx 10.735
10,000 04:02:19.266 MyPage.aspx 5.200

There are over 10,000 answers to this question!
Not all charts are created equal
Big Data: Details for 10,000 Individual Transactions
Big Data: Details for 10,000 Individual Transactions
Big Data: Analyzing *Sets* of Transactions

- **Exclude:** TXs with no exceptions
- **60-65 sec** <0.5% Web Service Timeout
- **30-35 sec** ~2% DB Timeout
- **15-20 sec** ~3% Auth Timeout
- **100-500 ms** ~1% Initialization Failure
- **10-60 sec** ~2% ~3% ~1%
Challenge #6: Beware “Performance Phantoms”
Consistency vs. Inconsistency

Response Time (seconds)

<30 sec 1:30pm-2:30pm
2,162 TXs

Overall: Method C

© 2016 Riverbed Technology. All rights reserved.
Consistency vs. Inconsistency

- Response Time (seconds)
- >40 sec
- 330 TXs
- Overall: NONE

© 2016 Riverbed Technology. All rights reserved.
The Application’s Ecosystem

Java Application

- Java Code
- JVM Heap (Reserved RAM)
- Java JVM (OS Process)

.NET Application

- .NET Code
- CLR Heap (Reserved RAM)
- .NET CLR (OS Process)

OS Resources (CPU, RAM, I/O etc.)

Operating System

The Application’s Ecosystem

Java Application

- Java Code
- JVM Heap (Reserved RAM)
- Java JVM (OS Process)
- OS Resources (CPU, RAM, I/O etc.)

Operating System

.NET Application

- .NET Code
- CLR Heap (Reserved RAM)
- .NET CLR (OS Process)
- OS Resources (CPU, RAM, I/O etc.)

Operating System

The Application’s Ecosystem

Java Application VM

- Java Code
- JVM Heap (Reserved RAM)
- Java JVM (OS Process)
- Virtual OS Resources (CPU, RAM, I/O etc.)
- Guest Operating System

.NET Application VM

- .NET Code
- CLR Heap (Reserved RAM)
- .NET CLR (OS Process)
- Virtual OS Resources (CPU, RAM, I/O etc.)
- Guest Operating System

Hypervisor

Physical OS Resources (CPU, RAM, I/O etc.)

Performance Phantom

Same delay causes 3x – 11x relative execution time increase

Method A Executions – Thread 1

Method B Executions – Thread 2

Method C Executions – Thread 3

Method D Executions – Thread 4

Method E Executions – Thread 5

JVM Garbage Collection State

Method execution time increase

11x
6x
4.3x
3.5x
3x

Same delay causes 3x – 11x relative execution time increase

Performance Phantom

Same delay causes 3x – 11x relative execution time increase

Method execution time increase

11x

6x

4.3x

3.5x

3x

SYMPTOM

ROOT CAUSE

TruePlot – tells the true story

Multi-type spike: “Performance Phantom” (Shared dependency issue or environmental issue)

Single-type pattern: True code issue
Challenge #7: Be \textit{(business)} relevant
Business-relevant APM --- $$$ not seconds
Optimization & troubleshooting based on Financial Impact

% of Total Delay (seconds)

Top methods/sql based on seconds of delay

% of Total COST of Delays ($$$)

Top methods/sql based on cost of those seconds
53ms
Optimized

2K hr
Saved / Day

79,463 Matching Transactions
Response Time (sec.)
Try it for free at
www.appinternals.com