Bulletproof your CI pipeline

Using APM to augment your automated performance testing
Brad Stoner

- Senior Engineer, AppDynamics
- brad.stoner@appdynamics.com
- @sandbreak80
Key topics

- Segment a test strategy to find performance issues faster and earlier in the SDLC
- Reducing human involvement and manual root cause isolation from CI/CD deployment pipelines
- Executing a test strategy that decouples infrastructure testing from code testing
- Using APM to augment your automated performance testing
What is performance testing
Legacy performance testing
Keeping up with Agile / DevOps

• “It takes 2 weeks to script all our use cases and we get releases every 3 days”
• “The application is too difficult to test”
• “We are moving to agile on our legacy waterfall project. How we get started?”
• “QA will always be the bottleneck”
• “Issues are difficult to reproduce and our environment is unstable”
• “We don’t have visibility into our infrastructure”
• “If we find an issue, it still takes a week to fix”
• “We have no idea what changed in the application or why we are testing it again”
QA / Dev

- Short test cycle
- Low resources / cost
- Rapid feedback
- Component testing
Staging / Pre-prod

- Increased resources
- Long test durations
- Additional resources
- Multiple test elements
- Integrated systems
- Build validation
Prod / Perf

- High complexity
- Multiple integrated systems
- Environmental variance
- Resource intensive
- High cost
Mobile web/app example

QA testing – API flows

Build automation

APIs released/ BE functionality

Dev testing - APIs

Mobile site built

Optimize app chatter and network resources

Baseline Pre-Prod / Staging - platform

Mobile site released

Mobile app built

Staging testing – Capacity w/ UI and API

Front End Optimization

Mobile app released

Prod / Perf testing (inside firewall) – stability / scalability

Prod / Perf testing (outside firewall) – network / load balancing
What if legacy test principles were applied?

- Staging testing – Capacity w/ UI and API
- Prod / Perf testing (inside firewall) – stability / scalability
- Prod / Perf testing (outside firewall) – network / load balancing
- Front End Optimization
- Mobile app built
- Mobile app released
- Mobile site built
- Mobile site released
- Optimize app chatter and network resources
- Baseline Pre-Prod / Staging - platform
- APIs released/ BE functionality
Getting started with automated performance testing

- Establish baseline and tune infrastructure
  - Stabilize platform first
- Isolate functionality - API vs UI tests
- Develop apps with testability in mind
- Automation – nightly tests (build validation)
- Capacity and scalability testing is still needed
- Golden set of core functionality and performance
- Additional tests for scoped functionality
- Focus on code performance vs platform performance
  - Build focused
- Automate what makes sense
  - 2 weeks to script vs 2 weeks to setup and update automation
- Develop tests cases as close to development as possible / reuse functional test cases where possible
How does APM enable agility?

1. As our code and architecture rapidly changes, does our monitoring dynamically adapt?
2. Can we quantify the impact of changes, can we drive improvements from our observations?
3. Can we proactively detect degradations, and notify across multiple channels?
4. Can we derive observation into business performance?
Integrating release markers into CI pipelines

```
# Execute a command to inform App about the build event
sudo -u centos -i $(JAVA_HOME)/bin/bash

# Execute a command to send an event to the App
curl -s -L -X GET http://example.com/server/ping
```
Dynamic discovery of our application topology

AppDynamics dynamically discovers our web application, and its dependencies to the legacy payments platform.
AppDynamics discovers the business transactions our application delivers. We can constantly manage customer experience by function, on top of code and infrastructure.

Business Transactions:
- Account Overview
- Adding a Card
- Submitting Payment
- Account History
Understanding our first updated code drop

After the first set of code changes, AppDynamics dynamically updates the map to reflect the new application topology.

Immediately we can see that our backend team added:

- REST call to JBOSS
- JDBC call to MySQL

Dev

#VelocityConf
Measuring the impact of our release

AppDynamics can compare the performance of test, or production, code drops to quantify what changed and by how much.

What’s changed?

- Release 1 – No events / errors
- Release 2
  - Code problems
  - Error rate increased to 6.9%

Dev
Troubleshooting a failed build or code drop

Because we integrated into our CI pipeline, our development team *can quickly identify the code drop, and Jenkins Build, that introduced the error.*

<table>
<thead>
<tr>
<th>Type</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Deploy</td>
<td>LOAD_TEST_STOP-Load Test - FrontEnd-237</td>
</tr>
<tr>
<td>Application Deploy</td>
<td>LOAD_TEST_START-Load Test - FrontEnd-237</td>
</tr>
<tr>
<td>Application Deploy</td>
<td>LOAD_TEST_STOP-Load Test - FrontEnd-236</td>
</tr>
<tr>
<td>Application Deploy</td>
<td>LOAD_TEST_START-Load Test - FrontEnd-236</td>
</tr>
<tr>
<td>Application Deploy</td>
<td>LOAD_TEST_STOP-Load Test - FrontEnd-235</td>
</tr>
</tbody>
</table>

Dev

#VelocityConf
Getting additional context on our code change

Our team can isolate the application function that is impacted, `accountLookup`, as well as measure the severity of the defect.
Real-time transaction scorecards

AppDynamics scores all transactions in real-time, our DevOps team can immediately see the scope of the performance problem or errors

Understanding the impact
• Know when the failures started
• Know how many transactions are impacted
Source line of code, root-cause diagnostics

With a single click we can see all affected transactions, and quickly drill-down to the SLOC root-cause failure
Proactively alert on performance and health

Checkout is a revenue critical transaction, when there's an issue we want AppDynamics to raise a ticket so our team is alerted to ACT immediately

Proactive alert notification:

• Machine learning dynamic baselines, alert on deviation from healthy

• Proactively notify and manage issues in the “yellow”, before they become critical

• Native integration with SNOW, and other ticketing platforms
Seamlessly launch from SNOW to AppDynamics

Deep URLs and a rich web platform make collaboration across tools and teams seamless

- Immediately share data between Ops, Support, QA, and Dev
- Enable all teams to execute from the same dataset, instead of silo’d tools and environments
Now that you've decided to take the first step, it's time to, well, take that first step—by signing up for AppDynamics Lite, a completely free, full-featured version of our robust Application Intelligence Platform. Once you sign up for AppDynamics Lite, you can use it free for a day, a month, a year—or forever. Even better, your first 15 days includes a free upgrade to AppDynamics Pro, which includes more monitoring agents and longer data retention. So take that first step, and take advantage of AppDynamics Lite today. It's the last first step you'll ever need to take.

https://www.appdynamics.com/lite/
Questions
Screenshots from live demo
Application under test
Use Jenkins to manage tests
Trend release performance over time
Integrate with functional and performance tools
### Jenkins job details

#### Default NeoLoad Trend Graphs

| Avg | Display Trend Graph: Average Response Time (all pages) |
| Errors | Display Trend Graph: Error Rate % |

#### Define NeoLoad Trend Graphs

- **Graphs**: Add Graph

#### Execute shell

**Command**

```
curl --verbose -s -c /tmp/session.dat --header 'Authorization: Basic YWRtaW44MDBydXN0b21lcjE6YXNzZGxhYmY=' -X GET http://devops.lab.appspHERE:8099/PARAMS="events\&eventtype=CUSTOM\&customeventtype=LOAD\_TEST\&summary=LOAD\_TEST\_STOP\-\$\{JOB\_NAME\}\$\{BUILD\_ID\}\&severity=INFO\&tier=Neo\_Server"
curl -s -b /tmp/session.dat --data "$(PARAMS)" http://devops.lab.appspHERE:8090/controller/rest/applications/KonaKart/events
```

See the list of available environment variables

#### Execute shell

**Command**

```
#!/bin/sh
END=$(/bin/date +%s)
echo "$END"
END_MS=$((END * 1000))
echo "$END_MS"
echo "$BUILD\_TIMESTAMP"
jobStart=$BUILD\_TIMESTAMP
START=$(/bin/date +%s)
echo "$START"
START_MS=$(($START * 1000))
echo "$START_MS"
curl --verbose -s -b /tmp/session.dat --header "Content-Type: application/json;charset=utf-8" --header "Authorization: Basic YnjhXCUW6N1c3Rv6WVymTo=
```
Git and AppDynamics integration

Build

Execute shell

**Command**

cd /home/ubuntu/neoload_projects/konakart
git fetch --all
git reset --hard origin/master

See the list of available environment variables

Execute shell

**Command**
curl --verbose -s -c /tmp/session.dat --header "Authorization: Basic YWRtaW41ND0wMD1lcjE6TXBwZmluYW1pY3M=" -X GET http://devopslabapppsphep:8090/par...Kona_Server"
curl -s -b /tmp/session.dat --data "${PARAMS}" http://devopslabapppsphep:8090/controller/rest/applications/KonaKart/events

See the list of available environment variables

Execute a NeoLoad Scenario

**Neoload Executable**

/home/ubuntu/neoload/bin/NeoloadCmd

Project Type

- Local Project

**Project File**

/home/ubuntu/neoload_projects/konakart/KonaKart.nlp

- Shared Project
AppDynamics Jenkins plugin

<table>
<thead>
<tr>
<th>AppDynamics Performance Publisher</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AppDynamics REST URL</strong></td>
<td><a href="http://devopslabappsshare:8090/">http://devopslabappsshare:8090/</a>**</td>
</tr>
<tr>
<td><strong>Username</strong></td>
<td>admin</td>
</tr>
<tr>
<td><strong>Password</strong></td>
<td>**********</td>
</tr>
<tr>
<td><strong>Application Name</strong></td>
<td>KonaKart</td>
</tr>
</tbody>
</table>

**Advanced AppDynamics Integration Options**

<table>
<thead>
<tr>
<th>Threshold Metric</th>
<th>Overall Application Performance/Average Response Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AppDynamics Metric</strong></td>
<td>that will be used to decide if thresholds are reached</td>
</tr>
<tr>
<td><strong>Lower is better</strong></td>
<td>$&gt;$</td>
</tr>
<tr>
<td><strong>Minimum Measure Time</strong></td>
<td>20</td>
</tr>
<tr>
<td><strong>Performance Threshold</strong></td>
<td>(80% Unstable, 65% Failed)</td>
</tr>
</tbody>
</table>

These thresholds provide the boundaries for when the build is marked as unstable or failed.
Performance job execution

Started by user appd
Building in workspace /var/lib/jenkins/workspace/Load Test - FrontEnd
[Load Test - FrontEnd] 5 /bin/sh -xe /tmp/hudson7337830548044031084.sh
- /home/ubuntu/neoload/bin/neoloadCmd -project /home/ubuntu/neoload_projects/konakart/KonaKart.nlp -launch Build -testResultsName $Date:hh:mm - dd MMM yyyy (build 23) -report /var/lib/jenkins/workspace/Load Test - FrontEnd/neoload-report/report.html /var/lib/jenkins/workspace/Load Test - FrontEnd/neoload-report/report.xml -SLAUUnitResults /var/lib/jenkins/workspace/Load Test - FrontEnd/neoload-report/junit-sla-results.xml -noGUI
Logging to /home/ubuntu/neotype/neoload/v5.2/logs, as user: ubuntu
Loading project: /home/ubuntu/neoload_projects/konakart/KonaKart.nlp
Project KonaKart loaded
Launching scenario: Build
Initializing...
Initializing Monitors...
Pre-Monitoring...
Running

Stoping...
Post-Monitoring...
Stopping Monitors...
Stopped.
Generating report /var/lib/jenkins/workspace/Load Test - FrontEnd/neoload-report/report.html... Done.
Generating report /var/lib/jenkins/workspace/Load Test - FrontEnd/neoload-report/report.xml... Done.
Generating report /var/lib/jenkins/workspace/Load Test - FrontEnd/neoload-report/junit-sla-results.xml... Done.
Verify connection to AppDynamics REST Interface ...
Connection successful, continue to fetch measurements from AppDynamics Controller ...
Ready building AppDynamics report
Verifying for improving or degrading performance, main metric: Overall Application Performance/Average Response Time (ms) where lower is better = true
Performance degradation greater or equal than 80% sets the build as unstable
Performance degradation greater or equal than 65% sets the build as failure
Number of old reports located for average: 4
Calculated average from previous reports: 7773.0
Current report average: 36904.0
Current average as percentage of total average: 21.062757424669414%
Metric: Overall Application Performance/Average Response Time (ms) reported performance compared to average of 21.062757424669414. Build status is: FAILURE
Build step 'AppDynamics Performance Publisher' changed build result to FAILURE
Archiving artifacts
Recording test results
Finished: FAILURE
## Track releases and deployment data

### KonaKart Dashboard

<table>
<thead>
<tr>
<th>Type</th>
<th>Summary</th>
<th>Time</th>
<th>Business Transaction</th>
<th>Tier</th>
<th>Node</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Deployment</td>
<td>LOAD_TEST_STOP-Load Test - FrontEnd-237</td>
<td>05/21/17 10:18:30 AM</td>
<td>-</td>
<td>Kona_Ser...</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Application Deployment</td>
<td>LOAD_TEST_START-Load Test - FrontEnd-237</td>
<td>05/21/17 10:06:05 AM</td>
<td>-</td>
<td>Kona_Ser...</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Application Deployment</td>
<td>LOAD_TEST_STOP-Load Test - FrontEnd-236</td>
<td>05/21/17 10:01:22 AM</td>
<td>-</td>
<td>Kona_Ser...</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Application Deployment</td>
<td>LOAD_TEST_START-Load Test - FrontEnd-236</td>
<td>05/21/17 9:48:58 AM</td>
<td>-</td>
<td>Kona_Ser...</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Application Deployment</td>
<td>LOAD_TEST_STOP-Load Test - FrontEnd-235</td>
<td>05/21/17 9:42:04 AM</td>
<td>-</td>
<td>Kona_Ser...</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Application Deployment</td>
<td>LOAD_TEST_START-Load Test - FrontEnd-235</td>
<td>05/21/17 9:29:40 AM</td>
<td>-</td>
<td>Kona_Ser...</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Application Deployment</td>
<td>LOAD_TEST_STOP-Load Test - FrontEnd-234</td>
<td>05/21/17 9:28:24 AM</td>
<td>-</td>
<td>Kona_Ser...</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Application Deployment</td>
<td>LOAD_TEST_START-Load Test - FrontEnd-234</td>
<td>05/21/17 9:16:00 AM</td>
<td>-</td>
<td>Kona_Ser...</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Measure release performance
Easily compare build performance
Rapidly isolate performance issues
### Identify systemic performance issues

#### Slow Response Times

**Slow Transactions**

- **Slowest DB & Remote Service Calls**

  These are the calls with largest observed individual execution time (Max Time) during the specified time range.

<table>
<thead>
<tr>
<th>Call</th>
<th>Avg. Time</th>
<th>Number of Calls</th>
<th>Max Time</th>
<th>Snapshots</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELETE FROM CUSTOMERS_BASKET WHERE CUSTOMERS_BASKET.CUSTOMERS_ID=?</td>
<td>721.7</td>
<td>481</td>
<td>1910</td>
<td>View snapshots</td>
</tr>
<tr>
<td>SELECT CUSTOMERS_BASKET.CUSTOMERS_BASKET_ID, CUSTOMERS_BASKET.CUSTOMERS_BASKET_DATE_ADDED, CUSTOM</td>
<td>427.6</td>
<td>691</td>
<td>691</td>
<td>View snapshots</td>
</tr>
<tr>
<td>SELECT CUSTOMERS_BASKET.CUSTOMERS_BASKET_ID, CUSTOMERS_BASKET.CUSTOMERS_BASKET_DATE_ADDED, CUSTOM</td>
<td>381.1</td>
<td>256</td>
<td>646</td>
<td>View snapshots</td>
</tr>
<tr>
<td>SELECT ORDERS.ORDERS_ID, ORDERS.ORDERS_NUMBER, ORDERS.LIFECYCLE_ID, ORDERS.TRACKING_NUMBER, ORDERS.ST</td>
<td>190</td>
<td>237</td>
<td>287</td>
<td>View snapshots</td>
</tr>
<tr>
<td>SELECT ORDERS_PRODUCTS.ORDERS_PRODUCTS_ID, ORDERS_PRODUCTS.PRODUCTS_ID, ORDERS_PRODUCTS.PRODUCTS</td>
<td>66.4</td>
<td>135</td>
<td>104</td>
<td>View snapshots</td>
</tr>
<tr>
<td>SELECT PRODUCTS.PRODUCTS_ID, MAX(ORDERS.DATE_PURCHASED) FROM ORDERS_PRODUCTS OPA, ORDERS_PRODUCTS</td>
<td>52.9</td>
<td>51</td>
<td>103</td>
<td>View snapshots</td>
</tr>
<tr>
<td>SELECT CUSTOMERS.CUSTOMERS_ID, CUSTOMERS.CUSTOMERS_FIRSTNAME, CUSTOMERS.CUSTOMERS_LASTNAME, CUSTC</td>
<td>61.8</td>
<td>33</td>
<td>96</td>
<td>No snapshots</td>
</tr>
<tr>
<td>SELECT ORDERS_STATUS_HISTORY.DATE_ADDED, ORDERS_STATUS_HISTORY.CUSTOMER_NOTIFIED, ORDERS_STATUS_HIST</td>
<td>28.9</td>
<td>130</td>
<td>52</td>
<td>View snapshots</td>
</tr>
<tr>
<td>SELECT COUNT(*) FROM ORDERS, ORDERS_STATUS WHERE ORDERS.ORDERS_STATUS=ORDERS_STATUS.ORDERS_STATUS_</td>
<td>28.7</td>
<td>211</td>
<td>48</td>
<td>View snapshots</td>
</tr>
</tbody>
</table>

---

 Velvetty Conf
Resolve performance issues FAST
Track response times against releases

### Compare Releases

**Select what to Compare**
- Application
- Business Transactions
- Servers
- Kona_Server
- Node1
- VM

**View:** All Business Transactions

#### Time Range 1

<table>
<thead>
<tr>
<th>Name</th>
<th>Heap (M)</th>
<th>Time (ms)</th>
<th>Calls</th>
<th>Calls / min</th>
<th>Error</th>
<th>Error%</th>
<th>Slope</th>
<th>Verbatim</th>
<th>Stall</th>
<th>Tier</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>LoadTest.New+Buyer.Actions.confirm+on</td>
<td></td>
<td>382</td>
<td>94</td>
<td>7</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>K</td>
<td>Servlet</td>
</tr>
<tr>
<td>LoadTest.Check+Order+Status.Actions.io</td>
<td></td>
<td>247</td>
<td>26</td>
<td>2</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>K</td>
<td>Servlet</td>
</tr>
<tr>
<td>LoadTest.New+Buyer.Actions.new+custom</td>
<td></td>
<td>146</td>
<td>188</td>
<td>14</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>K</td>
<td>Servlet</td>
</tr>
<tr>
<td>LoadTest.Check+Order+Status.Actions.web</td>
<td></td>
<td>138</td>
<td>26</td>
<td>2</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>K</td>
<td>Servlet</td>
</tr>
<tr>
<td>LoadTest.New+Buyer.Actions.continue</td>
<td></td>
<td>102</td>
<td>94</td>
<td>7</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>K</td>
<td>Servlet</td>
</tr>
<tr>
<td>LoadTest.Browse.Actions.view+cart</td>
<td></td>
<td>56</td>
<td>120</td>
<td>9</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>K</td>
<td>Servlet</td>
</tr>
<tr>
<td>LoadTest.New+Buyer.Actions.select+photo</td>
<td></td>
<td>52</td>
<td>188</td>
<td>14</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>K</td>
<td>Servlet</td>
</tr>
<tr>
<td>LoadTest.New+Buyer.Actions.search</td>
<td></td>
<td>11</td>
<td>94</td>
<td>7</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>K</td>
<td>Servlet</td>
</tr>
</tbody>
</table>

#### Time Range 2

<table>
<thead>
<tr>
<th>Name</th>
<th>Heap (M)</th>
<th>Time (ms)</th>
<th>Calls</th>
<th>Calls / min</th>
<th>Error</th>
<th>Error%</th>
<th>Slope</th>
<th>Verbatim</th>
<th>Stall</th>
<th>Tier</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>LoadTest.Check+Order+Status.Actions.io</td>
<td></td>
<td>1531</td>
<td>26</td>
<td>2</td>
<td>-</td>
<td>0</td>
<td>2</td>
<td>24</td>
<td>0</td>
<td>K</td>
<td>Servlet</td>
</tr>
<tr>
<td>LoadTest.New+Buyer.Actions.confirm+on</td>
<td></td>
<td>1337</td>
<td>94</td>
<td>8</td>
<td>-</td>
<td>0</td>
<td>21</td>
<td>73</td>
<td>0</td>
<td>K</td>
<td>Servlet</td>
</tr>
<tr>
<td>LoadTest.New+Buyer.Actions.new+custom</td>
<td></td>
<td>1259</td>
<td>188</td>
<td>16</td>
<td>-</td>
<td>0</td>
<td>17</td>
<td>72</td>
<td>0</td>
<td>K</td>
<td>Servlet</td>
</tr>
<tr>
<td>LoadTest.New+Buyer.Actions.logout</td>
<td></td>
<td>1061</td>
<td>94</td>
<td>8</td>
<td>-</td>
<td>0</td>
<td>15</td>
<td>79</td>
<td>0</td>
<td>K</td>
<td>Servlet</td>
</tr>
<tr>
<td>LoadTest.Check+Order+Status.Actions.io</td>
<td></td>
<td>973</td>
<td>26</td>
<td>2</td>
<td>-</td>
<td>0</td>
<td>8</td>
<td>18</td>
<td>0</td>
<td>K</td>
<td>Servlet</td>
</tr>
<tr>
<td>LoadTest.Browse.Actions.add+to+cart</td>
<td></td>
<td>818</td>
<td>120</td>
<td>10</td>
<td>-</td>
<td>0</td>
<td>64</td>
<td>56</td>
<td>0</td>
<td>K</td>
<td>Servlet</td>
</tr>
<tr>
<td>LoadTest.New+Buyer.Actions.add+to+cart</td>
<td></td>
<td>817</td>
<td>94</td>
<td>8</td>
<td>-</td>
<td>0</td>
<td>50</td>
<td>44</td>
<td>0</td>
<td>K</td>
<td>Servlet</td>
</tr>
</tbody>
</table>

#VelocityConf
Visibility into database performance

**Dashboard**

**konakart_mysql**

- **SERVER HEALTH**: Normal
- **TOTAL EXECUTIONS**: 0.22m
- **DATABASE TYPE**: MySQL
- **TOTAL TIME IN DATABASE**: 00:16:11

**Load and Time Spent in Database**

00:16:11 Time Spent in Database

0.22m calls Load

Graph showing time spent and load over time.
DB waits between releases

### Wait State

<table>
<thead>
<tr>
<th>Wait State</th>
<th>Description</th>
<th>Total Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sending data</td>
<td>The thread is reading and processing rows for a <code>SELECT</code> statement, and sending data to the client. Because operations occurring during this state tend to perform large amounts of disk access (reads), it is often the cause of the longest wait state.</td>
<td>00:08:24</td>
</tr>
<tr>
<td>Updating</td>
<td>The thread is searching for rows to update and is updating them.</td>
<td>00:05:39</td>
</tr>
<tr>
<td>Copying to tmp table</td>
<td>The server is copying to a temporary table in memory.</td>
<td>00:01:48</td>
</tr>
</tbody>
</table>
Dashboards offer a sharable view for ops and dev
Compare system / capacity between releases
Custom name business transactions via test tools

<table>
<thead>
<tr>
<th>Name</th>
<th>Respon... Time (ms)</th>
<th>Calls / min</th>
<th>Errors / min</th>
<th>% Errors</th>
<th>% Slow Transact.</th>
<th>% Very Slow Transact.</th>
<th>% Stalled Transact.</th>
<th>CPU Used (ms)</th>
<th>Block Time (ms)</th>
<th>Wait Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LoadTest.Check+Order+Status.Actions.login+user</td>
<td>1,531</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>7.7</td>
<td>92.3</td>
<td>-</td>
<td>14</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>LoadTest.New+Buyer.Actions.confirm+order</td>
<td>1,337</td>
<td>8</td>
<td>-</td>
<td>-</td>
<td>22.3</td>
<td>77.7</td>
<td>-</td>
<td>31</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>LoadTest.New+Buyer.Actions.new+customer++for...</td>
<td>1,259</td>
<td>16</td>
<td>-</td>
<td>-</td>
<td>9</td>
<td>38.3</td>
<td>-</td>
<td>15</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>LoadTest.New+Buyer.Actions.logout</td>
<td>1,061</td>
<td>8</td>
<td>-</td>
<td>-</td>
<td>16</td>
<td>84</td>
<td>-</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>LoadTest.Check+Order+Status.Actions.logout</td>
<td>973</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>30.8</td>
<td>69.2</td>
<td>-</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>LoadTest.Browse.Actions.add+to+cart</td>
<td>818</td>
<td>10</td>
<td>-</td>
<td>-</td>
<td>53.3</td>
<td>46.7</td>
<td>-</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>LoadTest.New+Buyer.Actions.add+to+cart</td>
<td>817</td>
<td>8</td>
<td>-</td>
<td>-</td>
<td>53.2</td>
<td>46.8</td>
<td>-</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>LoadTest.Browse.Actions.homepage</td>
<td>336</td>
<td>30</td>
<td>-</td>
<td>-</td>
<td>6.1</td>
<td>22.2</td>
<td>-</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>LoadTest.Check+Order+Status.Actions.homepage</td>
<td>320</td>
<td>7</td>
<td>-</td>
<td>-</td>
<td>6.4</td>
<td>23.1</td>
<td>-</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>LoadTest.New+Buyer.Actions.homepage</td>
<td>315</td>
<td>24</td>
<td>-</td>
<td>-</td>
<td>7.1</td>
<td>21.3</td>
<td>-</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>LoadTest.New+Buyer.Actions.continue</td>
<td>210</td>
<td>8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Directly compare tier performance
Track GC and memory utilization between releases

- Average Utilization: 25%
- Current Utilization: 17%
- Current Usage: 213 MB
- Current Committed: 779 MB
- Max Available: 1244 MB
- Free: 1031 MB
Slack, PagerDuty, Service-Now integrations
Slack notifications for dev teams

```
incoming-webhook  APP  8:24 AM
incident_key: $latestEvent.node.name - KonaKart
    description: Application Deployment on $latestEvent.node.name
    client: AppDynamics

Details:

    Event Name: Application Deployment
    Summary: LOAD_TEST_START-Load Test - FrontEnd-232
    Event ID: 64758
    Event Time: Sun May 21 15:24:40 UTC 2017
    Event Type: APPLICATION DEPLOYMENT
    Event Type Key: APPLICATION DEPLOYMENT
    Application Name: KonaKart
    Node Name: $latestEvent.node.name
    Message: LOAD_TEST_START-Load Test - FrontEnd-232
    Severity: INFO

View this transaction in AppDynamics: http://ec2-52-205-164-180.compute-1.amazonaws.com:8090/controller/#location=APP_EVENT_VIEWER_MODAL&eventSummary=64758
```