Change Point Detection in Software Performance Testing

David Daly, William Brown, Henrik Ingo, Jim O’Leary, David Bradford
Performance Testing Goals (In CI)

Know if and when the performance changes
  • If it gets slower, quickly fix it
  • If it gets faster, lock in the improvement

Part of our release process
  • The quicker the notification, the easier it is to:
    • Isolate the cause of the change
    • Fix or backout the responsible change
Performance Testing in Continuous Integration

Setup a system under test
Run a workload
Report the results
Decide (and alert) if the performance changed
Visualize the result
Automate everything/Keep noise down
Performance Testing in Continuous Integration (V0)

Setup a system under test
Run a workload
Report the results

Decide (and alert) if the performance changed
  • Human looking at graphs – there are a lot of graphs

Visualize the result
Automate everything/Keep noise down
ADD_NODE
56f001f
Sep 21 2019
ops_per_sec:
1,205
bfs:
ACK  HIDE
UNMARK
COMPARE

UPDATE_NODE
56f001f
Sep 21 2019
ops_per_sec:
1,053
bfs:
ACK  HIDE
UNMARK
COMPARE
Performance Testing in Continuous Integration (V1)

- Setup a system under test
- Run a workload
- Report the results

**Decide (and alert) if the performance changed**
  - Alert if performance drops more than 10% from baseline

- Visualize the result
- Automate everything/Keep noise down
Oct 2 2019
ops_per_sec:
bfs:
[Buttons: ACK, HIDE, UNMARK, COMPARE]
Thresholds Are Awful

But better than version 0!

Problems

• False positives – some tests are noisier than others
• False negatives – miss any change less than the threshold
• Identifying regressions at the wrong time
  • E.g., 8% drop doesn’t cross threshold, but a week later 8% drop + 3% noise cross the threshold
Problem Statement

*Detect which commits change the performance of the software (as measured by our performance tests) in the presence of the noise from the testing infrastructure.*

Change Point Detection

“Change point analysis is the process of detecting distributional changes within time-ordered observations.”
Support For Change Point Detection

- Calculate the change points
- Visualize change points on trend graphs
- Change point dashboard for triage
  - Verify and isolate
  - Create JIRA tickets
<table>
<thead>
<tr>
<th>Revision</th>
<th>Hazard Level</th>
<th>Variant</th>
<th>Task</th>
<th>Test</th>
<th>Thread Level</th>
<th>Create Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>03c012B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>&gt;2019-08-29</td>
</tr>
<tr>
<td>-1%</td>
<td></td>
<td>atlas-like-M60</td>
<td>industry_benchmarks_secondar...</td>
<td>ycas_50read50update_secondar...</td>
<td>32</td>
<td>2019-09-04T04:17:46Z</td>
</tr>
<tr>
<td>6f30bb</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2019-08-31T10:30:45Z</td>
</tr>
<tr>
<td>-2%</td>
<td></td>
<td>linux-1-node-replSet</td>
<td>change_streams_throughput</td>
<td>15_lookup_1c_update</td>
<td>20</td>
<td></td>
</tr>
</tbody>
</table>
Impact: Does it Work?

Yes – Game Changing for us (but could still be better)

Qualitatively

- A human can process all the results
- Finding changes with smaller magnitude
- Finding changes faster $\rightarrow$ Regressions fixed sooner
- Recognizing improvements

Quantitatively

- E-divisive didn’t miss any real changes caught by the threshold system
- From 1% of notifications being useful to 67%
Work with (Help) Us

We have real world problems and would love to work with the community

• **Noise Reduction work**
• DBTest.io: “Automated System Performance Testing at MongoDB”
• LTB Talk: “How to Waste Time and Money Testing the Performance of a Software Product.”

Our code is open source: [signal-processing-algorithms](#), [infrastructure code](#)

Our **regression environment** is open, and **the platform** is open source

Our performance data is not open source, but we’re working to share it with academics
Thank you