Optimal Usage of the Computer Farm for Regression Testing

Daniel Hansson, Verifyter
Patrik Granath, Verifyter
Background

Why does the need for computer farm resources increase so fast?

We run more stuff

...and my jobs are still queued. Buy more!

We need some science!
Optimal Setup

Which regression test setup is the most efficient?

At the end of the presentation you will know
## Defining Metrics (1/2)

<table>
<thead>
<tr>
<th>Metric</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cost</strong></td>
<td>Total CPU test time</td>
</tr>
<tr>
<td><strong>Bug Identification Time</strong></td>
<td>Time from a bug is committed to the revision control system until a failure is reported</td>
</tr>
<tr>
<td><strong>Bug Fix Time</strong></td>
<td>Bug Identification Time + debug time + committing fix</td>
</tr>
<tr>
<td><strong>Test Fail Ratio (Quality)</strong></td>
<td>The number of failing tests / total number of tests run, for a given period</td>
</tr>
</tbody>
</table>
Defining Metrics (2/2)

Launch of Test Suite

Test Failure Reported

Bug Committed

Tests

Fix Committed

Bug Identification Time (BIT)

Debug Time

Bug Fix Time
Quality

- Better Test Fail Ratio (Quality) means Earlier Release
- Shorter BIT and Shorter Bug Fix Time => Earlier Release
- What is the optimal setup?
Doubling the cost (run time) does not double the chance of finding a bug (coverage)
Longer test suite means longer bug identification time at a higher cost, but only gives some better coverage.
Launch Frequency

#Sanity Runs (2h) / Day

Running twice instead of once per day matters a lot!

For higher launch frequencies it is a case of diminishing returns
Equations for 1 Test Suite

# Sanity Runs (2h) / Day

BIT = \frac{\text{cycle}}{2 \times L} + \text{TSL}

\text{Cost} = L \times \text{TSL} \times \text{CPUs}

L \quad \text{Launches}

\text{TSL} \quad \text{Test Suite Length}

\text{CPUs} \quad \text{Average \#CPU's}
Conclusions for 1 test suite

- The BIT looks similar for different cases
- The reason: the BIT depends mainly on #launches
- **Conclusion**: 2-5 launches/cycle has a good cost-benefit ratio
2 Test Suites

- **Assumption:** Functional coverage of *sanity* is a sub-set of the *nightly* coverage

\[
\Delta \text{coverage}_{\text{sanity}} = \frac{\text{coverage}_{\text{sanity}}}{\text{coverage}_{\text{nightly}}}
\]

\[
\Delta \text{coverage}_{\text{nightly}} = 1 - \Delta \text{coverage}_{\text{sanity}}
\]

\[
\text{BIT}_{\text{total}} = (\Delta \text{coverage}_{\text{sanity}} \times \text{BIT}_{\text{sanity}}) + (\Delta \text{coverage}_{\text{nightly}} \times \text{BIT}_{\text{nightly}})
\]
2 Test Suites

- There are 3 test suite sequences
  - `sanity` => `sanity`
  - `sanity` => `nightly`
  - `nightly` => `sanity`

- Probability for a bug to be committed during a test sequence

\[
\text{Max Commit To Launch (CTL)} \\
\text{cycle}
\]

- BIT for each test sequence

\[
\frac{\text{Max CTL}}{2} + TSL
\]
2 Test Suites

Average BIT for each of the 3 sequences

\[ BIT_{\text{sanity}} = \]
\[ = \left( \frac{\max x \ CTL_{\text{StoS}} + TSL_{\text{sanity}}}{2} \right) \times \left( \frac{\max x \ CTL_{\text{StoS}} \times (L_{\text{sanity}} - L_{\text{nightly}})}{\text{cycle}} \right) + \]
\[ + \left( \frac{\max x \ CTL_{\text{StoN}} + TSL_{\text{nightly}}}{2} \right) \times \left( \frac{\max x \ CTL_{\text{StoN}} \times L_{\text{nightly}}}{\text{cycle}} \right) + \]
\[ + \left( \frac{\max x \ CTL_{\text{NtoS}} + TSL_{\text{sanity}}}{2} \right) \times \left( \frac{\max x \ CTL_{\text{NtoS}} \times L_{\text{nightly}}}{\text{cycle}} \right) + \]

Probability for a bug to appear in this part of the cycle

\[ BIT_{\text{nightly}} = \frac{\text{cycle}}{2 \times L_{\text{nightly}}} + TSL_{\text{nightly}} \]

Cost total

\[ \text{Cost}_{\text{total}} = L_{\text{sanity}} \times TSL_{\text{sanity}} \times CPUs_{\text{sanity}} + \]
\[ L_{\text{nightly}} \times TSL_{\text{nightly}} \times CPUs_{\text{nightly}} \]
Optimal Scheduling

• The optimal launch schedule is achieved when all max BIT’s are equal
• This keeps the max BIT’s too a min

Optimal free time between two sanity runs

\[ T_{\text{freeSanity}} = \frac{T_{\text{freeTot}} + (L_{\text{tot}} - L_{\text{SanToSan}}) \ast (TSL_{\text{nightly}} - TSL_{\text{sanity}})}{L_{\text{tot}}} \]

Optimal free time between one sanity run and one nightly run

\[ T_{\text{freeTrans}} = \frac{T_{\text{freeTot}} - L_{\text{SanToSan}} \ast (TSL_{\text{nightly}} - TSL_{\text{sanity}})}{L_{\text{tot}}} \]
Optimal Scheduling (2/2)

- If the nightly test suite length is equal or larger than $\text{max BIT}_{StoS}$ then nightly will never first find a bug in the sanity coverage tranche.
- In this case it can be scheduled independently of the sanity runs.
Interleaving

- Interleaving test suites reduces the overall BIT
- This is because the nightly runs will be better spread out (lower $BIT_{\text{nightly}}$)
- $BIT_{\text{sanity}}$ is not order dependent if it is optimally scheduled
- With 3 test suites, run the shortest at least every second run
1 Nightly Run, X Sanity Runs

- 1 Nightly and 2-4 Sanity: good cost-benefit ratio
- Better than running 2 Nightly runs
3 Test Suites

Optimal Launch Frequencies (in red) Per Week for 3 Test Suites

Same pattern as for 2 Test Suites

<table>
<thead>
<tr>
<th>Test Suite Length (h)</th>
<th>Sanity (S)</th>
<th>Nightly (N)</th>
<th>Weekend (W)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>10</td>
<td>40</td>
</tr>
</tbody>
</table>

| Functional Coverage (%) | 40 | 60 | 70 |

S=1 N=1 W=1 Cheapest
Worst choices
S=1 N=1 W=2
S=1 N=1 W=3
S=58 N=1 W=1 (bad)
S=13 N=6 W=1 Running 75% of the time
S=9 W=7 W=2

Shortest Bug Identification Time Running 100% of the time
Summary

• Detect Bugs Fast, Fix Bugs Fast => Earlier release
• Launch Frequencies with good cost-benefit ratios:
  – 1 Test Suite: Run the test suite 2-5x per cycle
  – Multiple Test Suites: Run the largest test suite once and the smaller test suites 2-4x for each larger run
• Optimal Scheduling:
  – All max BIT’s should be equal for lowest coverage tranche
  – If a test suite is longer than the max BIT of a shorter test suite then you can schedule the test suites independently
  – Interleave test suites
  – With 3 test suites, run the shortest at least every second run

To find your optimal setup you need to use the equations
Which regression test setup is the most efficient?

Answer: A