Codebase Growth and the Developer Productivity Impact

Making the case for investment in Developer Productivity Engineering
About Me

Brian Stewart
Staff Systems Development Engineer @ Jamf
The Story
Incremental improvements to the developer experience
Jamf Pro project stats

Jamf Pro server monorepo

1 million+ lines of code

150 engineers contributing code

28,977 CI builds

- 2,415 builds/month, or 80 builds/day

23 minute average CI build time across all branches
Lines of Code vs. CI Build Time

<table>
<thead>
<tr>
<th>Year</th>
<th>Lines of Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>2020</td>
<td>621,000</td>
</tr>
<tr>
<td>2021</td>
<td>701,000</td>
</tr>
<tr>
<td>2022</td>
<td>800,000</td>
</tr>
<tr>
<td>2023</td>
<td>885,000</td>
</tr>
</tbody>
</table>
Build Times by Year

- 2020: Maven → Gradle, Build Cache, Remote Build Cache
- 2021: Build Cache Improvements
- 2022: Predictive Test Selection, Build Cache Optimization
- 2023: Test Distribution, Configuration Cache
The build time is actually 35% better than just doing nothing and letting it grow at the natural rate.
How we did it

Predictive Test Selection + Build Cache Optimization

Simplicity — the art of maximizing the amount of work not done — is essential.

- The 10th Principle of the Agile Manifesto
Predictive Test Selection

Intelligently run only the most useful subset of tests for a particular change
Predictive Test Selection

Machine learning applied to run only relevant tests

POC during Summer 2022, rolled out to full test suite in October 2022

Main branch runs all tests post-merge to keep full test coverage

Implementation effort was minimal - only a few hours of looking at simulated results to ensure accuracy
Predictive Test Selection

Our results after 6 months

PTS is saving us:
- 36% of unit test time
- 39% of integration test time

→ 111 days of build time saved per month*

* Wall clock build time, not serial execution time
Predictive Test Selection

Our results after 1 year

PTS is saving us:
- 93% of unit test time
- 64% of integration test time

→ 165 days of build time saved per month

What about uncaught test failures on the main branch?
- In 1 year and 2000+ merges, only 3 test failures slipped past PTS

* Wall clock build time, not serial execution time
Predictive Test Selection

Developer time and cost savings

Assuming developers actively wait on 20% of builds:

20% x 165 days saved/month = 33 days saved/month

33 days saved/month / 22 engineering days/month = 1.5 engineering months saved/month

Extrapolated, that is 1.5 engineering years (and cost) saved per year
Predictive Test Selection

CI agent cost savings

Running on Amazon EC2 agents (m5.xlarge) @ $0.192/hour:

$0.192/hour * 165 days saved/month * 24 hours/day * 12 months/year = $9,124 saved/year
Predictive Test Selection

- Usage (79K builds)
  - Mean build time: 11 min 10 sec
- Simulations (24.3K builds)
  - Mean build time: 10 min 59 sec

Test tasks/goals which enabled Predictive Test Selection

558K (86% of total)

- Active
- Predictive
- Unavailable
- Inactive

Savings test time saved:

294 d 18 h (80% of 370 d 15 h total savings potential)
Build Cache Optimization

Make tasks cacheable and keep cache misses low
Build Cache Optimization

Keeping cache misses low

- Fixed cache misses for a couple long-running test suites
- Cache hit rate: 98%
- Cache optimization maintenance this year: ~2 weeks
- Build cache avoidance savings: 60%
What’s the ROI?
Or, is the DPE investment justified?
Developer Productivity Engineering

The dystopian world

Without PTS + Build Cache, CI builds average 65 minutes

\[
65 \text{ m} / 60 \text{ m/hr} / 8 \text{ hrs/workday} \times 20\% \text{ waiting} \times 28,977 \text{ builds/year} = 785 \text{ days lost/year}
\]

\[\rightarrow 3.0 \text{ engineering years lost, per year}\]
Developer Productivity Engineering

Real-world savings @ Jamf

Incorporating PTS + Build Cache, CI builds average 23 minutes

\[
23 \text{ m} / 60 \text{ m/hr} / 8 \text{ hrs/workday} \times 20\% \text{ waiting} \times 28,977 \text{ builds/year} = 278 \text{ days lost/year}
\]

→ 1.1 engineering years lost, per year

The difference with DPE: 3.0 - 1.1 = 1.9 engineering years saved, per year
Developer Productivity Engineering

What’s the Return on Investment?

Total effort to maintain PTS + Build Cache going forward:
~10% FTE capacity, or 0.1 engineering years

What’s the ROI?
0.1 engineering years to save 1.9 engineering years:

19x ROI
What’s next?

2023 and beyond
Current and future optimizations

Relentless improvement

- Test Distribution
  - Already rolled out for unit tests
  - Integration tests in progress
- Configuration Cache
- Local IDE & workflow revamp
- Onboarding all Gradle projects at Jamf into Gradle Enterprise
Without any build acceleration features, developers would be waiting for builds an average of 5.2 days per developer per year, over a week lost per developer! Not only is the waiting time lost, but additional developer productivity is lost as developers lose their mental flow, increasing context switching and frustration.
Developer Productivity Engineering

The developer experience impact

Happy developers are creative, innovative developers.

DPE is helping the Jamf make our developer experience awesome, one step at a time.
fin

brian.stewart@jamf.com

engineering.jamf.com