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

Jamf

Helping organizations succeed with Apple

engineering.jamf.com

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

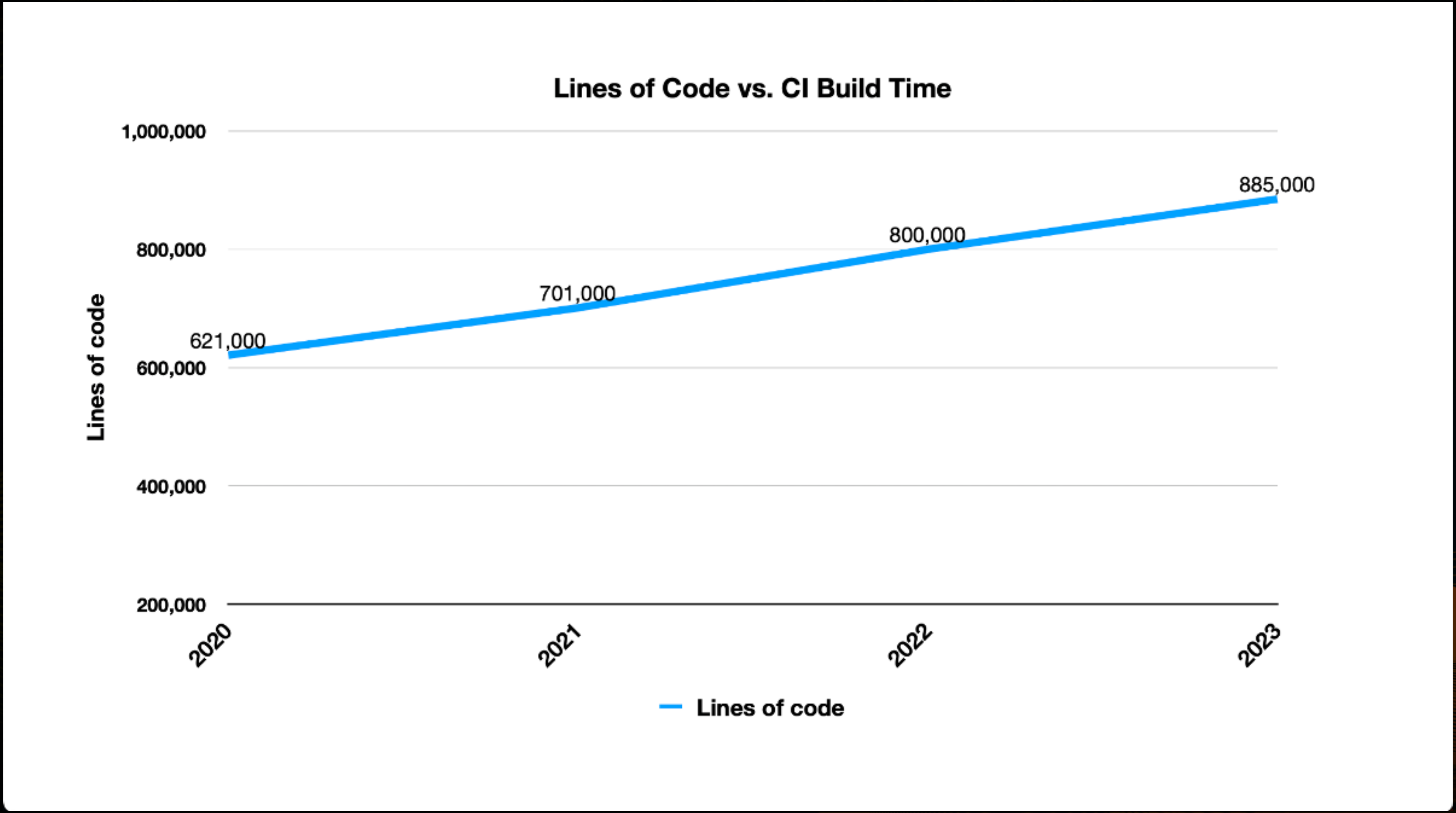
Java
Server Backend

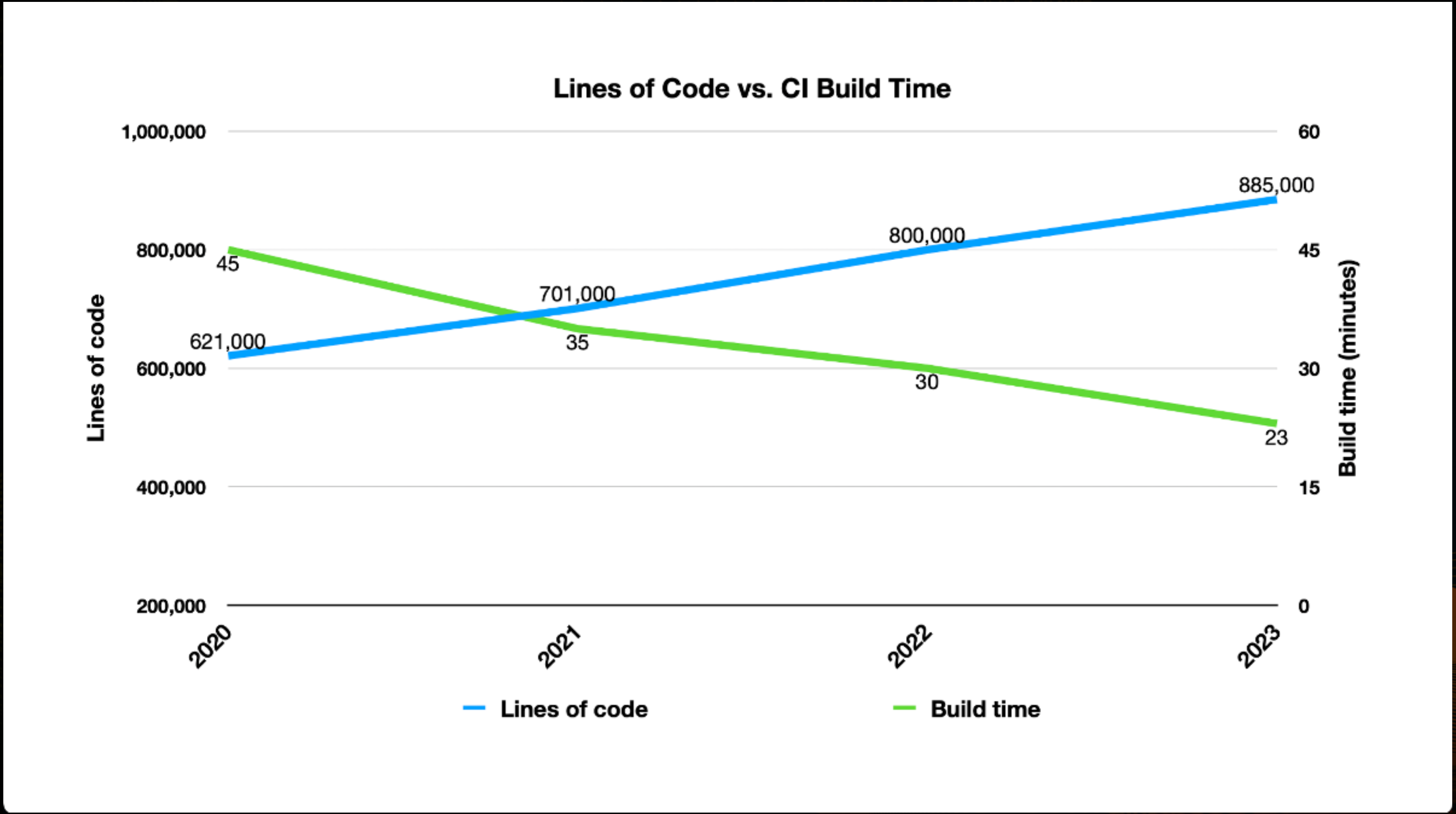
Gradle
Build

TypeScript
UI Frontend

JUnit
Unit + Integration Tests

Bamboo
CI Server


Lines of Code vs. CI Build Time
1,000,000
800,000
600,000
400,000
200,000
Lines of code
621,000
701,000
800,000
885,000
2020
2021
2022
2023
Lines of code


Lines of Code vs. CI Build Time
1,000,000
800,000
600,000
400,000
200,000
Lines of code
60
45
30
15
0
Build time (minutes)
621,000
701,000
800,000
885,000
45
35
30
23
2020
2021
2022
2023
Lines of code
Build time

Build Times by Year

45	35	30	23
2020	2021	2022	2023
Maven → Gradle Build Cache Remote Build Cache	Build Cache Improvements	Predictive Test Selection Build Cache Optimization	Test Distribution Configuration Cache

Zoom in: 2022-2023 YoY

⬆ 11%	⬆ 11%	⬆ 14%	⬇ 23%
Lines of Code	Unit Tests	Integration Tests	Build Time

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

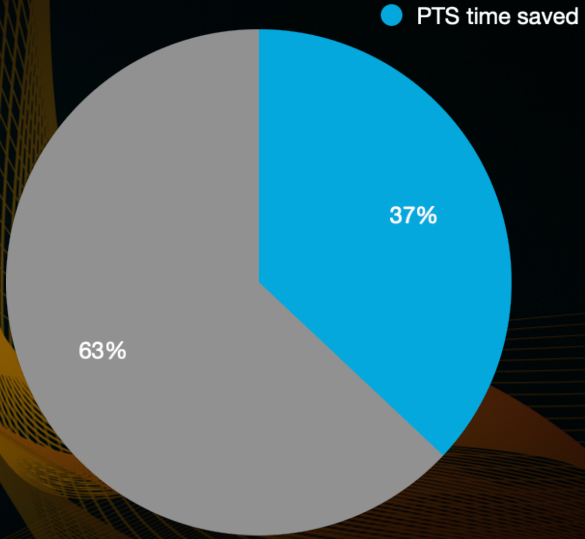
Predictive Test Selection

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:

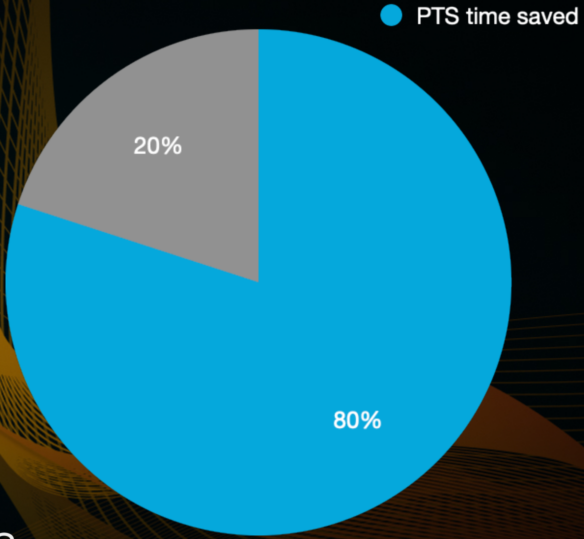
- 93% of unit test time
- 64% of integration test time

→ **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

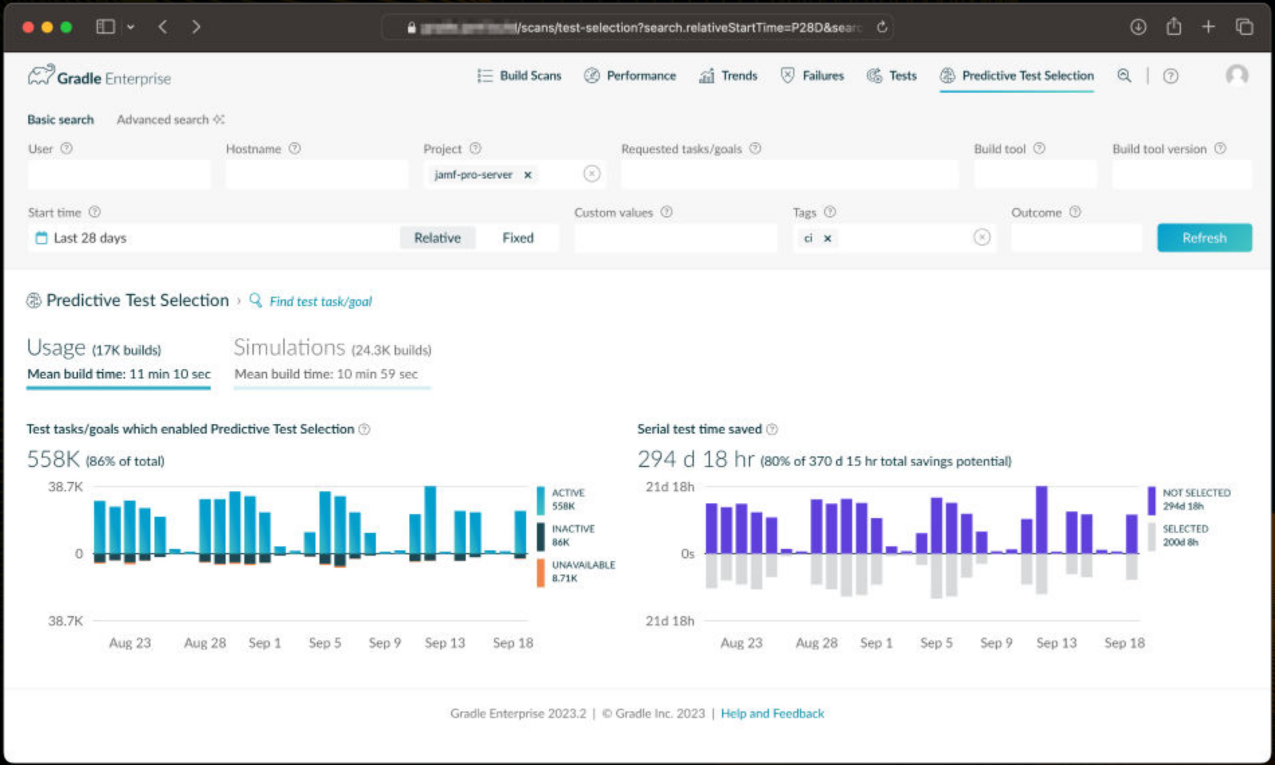
Predictive Test Selection

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



[redacted]/scans/test-selection?search.relativeStartTime=P28D&searc

Gradle Enterprise

Build Scans | Performance | Trends | Failures | Tests | Predictive Test Selection

Basic search | Advanced search

User	Hostname	Project	Requested tasks/goals	Build tool	Build tool version
		jamf-pro-server ×			

Start time		Custom values	Tags	Outcome	
Last 28 days	Relative / Fixed		ci ×		Refresh

Predictive Test Selection › Find test task/goal

Usage (17K 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)

Serial test time saved
294 d 18 hr (80% of 370 d 15 hr total savings potential)

Gradle Enterprise 2023.2 | © Gradle Inc. 2023 | Help and Feedback

Build Cache Optimization

Make tasks cacheable and keep cache misses low

Build Cache Optimization

Keeping cache misses low

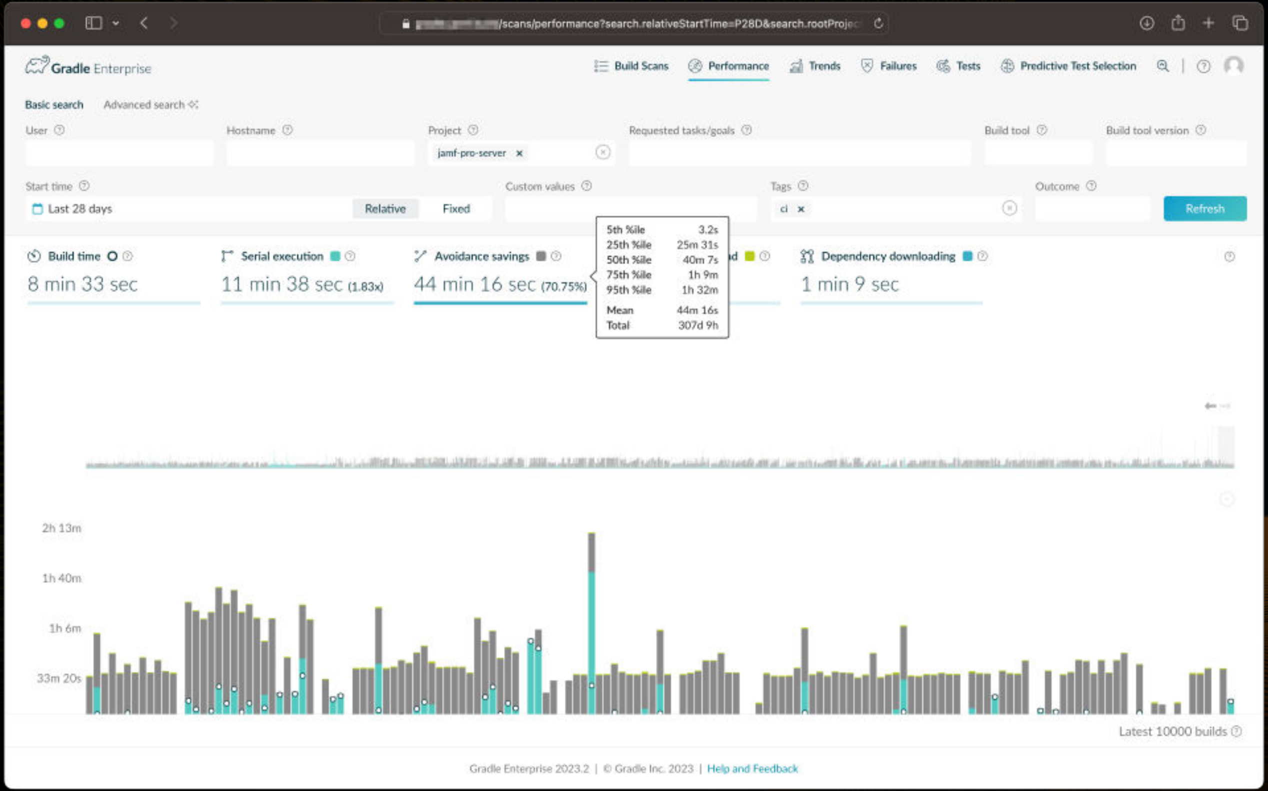
Fixed cache misses for a couple long-running test suites

Cache hit rate: 98%

Cache optimization maintenance this year: ~2 weeks

Build cache avoidance savings: 60%

Build Cache Optimization



[redacted]/scans/performance?search.relativeStartTime=P28D&search.rootProjec

Gradle Enterprise

Build Scans | Performance | Trends | Failures | Tests | Predictive Test Selection

Basic search | Advanced search

User	Hostname	Project	Requested tasks/goals	Build tool	Build tool version
		jamf-pro-server			

Start time	Custom values	Tags	Outcome
Last 28 days (Relative / Fixed)		ci	

Refresh

Build time	Serial execution	Avoidance savings	Dependency downloading
8 min 33 sec	11 min 38 sec (1.83x)	44 min 16 sec (70.75%)	1 min 9 sec

5th %ile	3.2s
25th %ile	25m 31s
50th %ile	40m 7s
75th %ile	1h 9m
95th %ile	1h 32m
Mean	44m 16s
Total	307d 9h

Latest 10000 builds

Gradle Enterprise 2023.2 | © Gradle Inc. 2023 | Help and Feedback

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 m / 60 m/hr / 8 hrs/workday * 20% waiting * 28,977 builds/year = 785 days lost/year

→ 3.0 engineering years lost, per year

Developer Productivity Engineering

Real-world savings @ Jamf

Incorporating PTS + Build Cache, CI builds average 23 minutes

23 m / 60 m/hr / 8 hrs/workday * 20% waiting * 28,977 builds/year = 278 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

Developer Productivity Engineering

The developer experience impact

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