Measuring & Improving Build Speeds

London Build Meetup

Vaibhav Shah

Why Measure Build Speeds?

- “You can't fix what you don't measure.”
- Unhappy developers
  - Unable to iterate and make changes quickly
  - Context switching
- Unhappy business
  - More bugs
  - Longer release timelines
- Builds aren’t everything, but they are critical

EngFlow

Raise your hand

if you measure build times

Raise your hand

if you measure builds at the task or action-level

A Typical Build Journey

- Stage 0: Chaos
  - No one really cares.
- Stage 1: Awareness
  - Complaints about latency and productivity hits
- Stage 2: Measure a little bit
  - Spot-check builds & relevant logs to determine slowness
- Stage 3: Improve build speeds
  - Throw more compute at the problem and shard jobs in CI
- Stage 4: Measure again
  - Collect CI timing via logs/dashboards and instrument local builds

Design Matters

- Not all build systems are the same
  - Dependency tracking
  - Invalidation strategy
  - Execution modes
- What you can measure depends on what the system tracks

What should we ideally measure?

- Wall time → latency
- Critical path → bottlenecks
- Cache hits → reuse efficiency
- Action or task time → bottlenecks by type
- CI trends → regressions
- Resource utilization (CPU, RAM, I/O, network)

Builds in CI

- Job duration (i.e. wall time)
  - Timing for individual pipelines
- Queue time
- Concurrency
- Exit codes and statuses
- Resource usage (CPU/RAM)

Best resources for measurement

- Profiling
  - Task/action execution times
  - Spawn strategy
  - Critical path breakdown
  - Memory usage
- Compilation commands
- Query language
- External Infrastructure

By Build System

- Bazel: Build Event Protocol, remote execution
- Buck2: profiling, remote execution
- Gradle: build scans, profiler plugins
- CMake: log parsing, profiling, compilation commands
- JS tools: plugins like SpeedMeasure, logging
- Others: `time`, `dstat`, `iostat`, `netstat`

Collect metrics

- Log total duration + timestamp to file if no profiling
- Emit profiles and/or metrics to something like Prometheus or BigQuery
- Parse profiles for action/task level timings and resource utilization
- Aggregate and visualize with Grafana or custom dashboards

Improving Build Speeds

- Provide more resources to the build
  - Use better CPUs and/or more RAM
  - Increase parallelism
- Enable caching
  - Fix hermeticity issues
- Break down bottlenecks in the critical path
- Modularization

Thank You!