

How VirtusLab Used Bazel Remote Execution to Cut PR Testing Costs by 20% and Project \$3M+ in Annual Savings

Client: **Global Technology Company**

Executive Summary

VirtusLab is helping a global technology company move critical monorepo workloads to **Bazel Remote Execution** with **EngFlow**. The work has already reduced the average cost of its main pull request tests by 20% and is projected to generate more than \$3 million in annual savings from PR testing alone.

The client's existing setup relied on large AWS EC2 workers assigned to groups of tests. These machines took around 15 minutes to start and were not always available in the required AWS region. Tests were divided into large shards, so the duration of the entire job was determined by the slowest shard. Uneven execution times could therefore delay the entire job. The environment also offered limited isolation between executions. Files, caches, or other state left behind by one build could affect the next, creating failures that were difficult to reproduce and investigate.

The client had already tested Bazel Remote Execution on a smaller repository. VirtusLab was brought in to extend the approach to the company's main monorepo and move its most important build and test workloads onto RBE. The migration involved thousands of tests initially incompatible with RBE, resource-heavy workloads, non-hermetic behavior, infrastructure issues, and a codebase that continued to change throughout the project.

Key Project Results

- **20% lower average cost** of the main PR tests
- **More than \$3 million in projected annual savings** from PR testing alone
- Tests excluded from RBE reduced from approximately **11,000 to under 3,000**
- **RBE enabled for all pull requests**, with the main pre-merge workflows and key post-merge test suites already migrated
- Post-merge compilation job made **3× faster**

The Challenges

Moving a large monorepo to remote execution is not simply a matter of connecting Bazel to a new group of machines. The existing workflows, tests, resource allocation, and infrastructure assumptions all need to work in a distributed and isolated execution environment.

Large Workers and Long Startup Times

The legacy system assigned large EC2 machines to entire test shards. Starting a worker took around **15 minutes**, generating cost and delay before useful work began. The required instance types were also not always available in the selected region, which could leave builds waiting for capacity. Once a worker was running, it remained assigned to the whole shard. If one test ran significantly longer than the others, the machine stayed occupied until that test finished - even when most of its work was already complete. This made it difficult to match infrastructure use closely to the actual work being performed.

Large Workers and Long Startup Times

Bazel Remote Execution runs individual actions in isolated environments. To work reliably, those actions need clearly defined inputs and cannot depend on undeclared files, services, network behavior, or machine-specific state.

Many tests in the client's monorepo had been built around assumptions that were valid in the legacy environment but did not hold under remote execution. These included expectations about DNS behavior, fixed Docker gateway IP addresses, IPv4 and IPv6 configuration, access to particular services or devices, and workloads incompatible with Docker-in-Docker setups. Some tests also assumed that they would always run in a specific AWS environment.

Remote execution exposed other issues that had previously remained hidden. Unlike the legacy environment, where CPU and memory could be assigned more granularly, RBE relied on a predefined set of executor configurations. Some tests therefore had to run with fewer resources than before, while others needed to move to a significantly larger configuration. This surfaced concurrency problems, timeouts, and incorrect assumptions about CPU and memory requirements.

A failed test could therefore point to a problem in the test itself, its Bazel configuration, EngFlow, Docker handling, network setup, resource allocation, or the surrounding infrastructure. Identifying which assumption had been broken was sometimes more difficult than applying the eventual fix.

The Target Kept Moving

The monorepo remained under active development during the migration. New tests were added, existing tests changed, and infrastructure updates introduced new behavior. VirtusLab was not working through a fixed backlog that simply became smaller every week. The team needed to make progress while the test suite itself continued to grow. That made reliable monitoring essential. It was not enough to know how many tests were failing at a given moment. The team also needed to understand why they were failing, how the situation was changing, and whether individual improvements were having the expected effect.

The Solution: Moving Critical CI Workloads to Bazel Remote Execution with EngFlow

Because the monorepo and its test suite continued to evolve throughout the engagement, VirtusLab combined measurement and diagnostics with test remediation, resource tuning, Docker optimization, and the migration of critical workloads to RBE.

Making the Migration Measurable

VirtusLab combined purpose-built migration dashboards with the client’s existing monitoring and targeted diagnostic scripts to build a reliable view of how selected workloads behaved across the legacy and RBE environments.

The dashboards created as part of the migration focused on:

Tests excluded from remote execution and the reasons behind those exclusions

Job duration and failure rates for workloads being prepared for migration

Differences between RBE and the legacy environment, including tests that failed remotely but passed locally

Failures affecting pull request workflows and the tests or actions associated with them;

Action types that were slower or more expensive on RBE than in the previous environment;

Migration progress and changes in workload performance over time.

The team also used the client’s existing monitoring to review executor-pool utilization, cache behavior, existing infrastructure cost metrics, and other platform-level indicators. Where more detailed information was needed, targeted scripts were used for selected workloads to inspect CPU and memory consumption, action input and output sizes, and changes in resource usage over time.

Together, these sources gave the engineers a shared view of the migration and helped them decide where to focus next. They also made it easier to distinguish issues introduced by remote execution from unrelated changes elsewhere in the repository. This was particularly important because the total number of tests was still growing. Looking only at the absolute number of failures would not have shown the full picture.

Fixing Common Causes Before the Long Tail

With approximately 11,000 tests initially incompatible with RBE, investigating every test independently would not have been practical. VirtusLab grouped failures according to their likely root causes and looked for changes that could unlock larger sets of tests.

The team worked through issues including:

Hidden or undeclared dependencies

Unintended network access

Timeouts and concurrency problems

Incorrect memory or CPU allocation

Excessive data transfer

Docker image handling

Build rule and infrastructure configuration errors

Part of the migration involved identifying hidden assumptions and improving test hermeticity so that workloads could run reliably in RBE.

The new remote execution infrastructure also made individual worker failures less disruptive. EngFlow automatically detects and restarts unhealthy workers, while affected actions can be retried elsewhere, limiting the impact on subsequent jobs.

Remote execution did not make worker-level issues impossible - subtle problems in Bazel, EngFlow, or the surrounding infrastructure could still occur - but it provided a more resilient recovery model and reduced their impact on the overall build.

Some changes fixed an entire group of tests, others required individual investigation. **LLM-based** tools supported parts of this work, particularly log analysis and repetitive test remediation. They were used to assist the team, while experienced engineers remained responsible for diagnosing the underlying problems and validating the changes. Over the course of the engagement, the number of tests excluded from RBE fell from approximately 11,000 to under 3,000. One of the most significant improvements came among the heavier workloads. The share of heavy tests excluded from remote execution dropped from more than 55% to under 9%. For unit tests, it fell from 10% to around 1.2%.

Matching Tests to the Right Resources

Not every test required the same amount of CPU or memory. Some workloads failed because their assigned executors were too small, while others were running on more expensive configurations than they actually needed.

For selected tests, VirtusLab evaluated their actual resource requirements and adjusted the executor configuration accordingly. The team also analyzed broader workload patterns and recommended changes to existing resource pools, as well as additional configurations for tests that exceeded the available capacity.

VirtusLab also explored a more automated approach to resource allocation. An analysis of one executor pool showed that approximately 95% of its tests could run with the same CPU allocation but half the memory. This made it possible to move those workloads to lower-cost AWS instances without reducing their CPU capacity.

Based on the results, VirtusLab recommended broader adoption of this approach and developed a plan for organizing executor pools around CPU requirements while adjusting memory more dynamically.

Optimizing Docker Image Builds and Data Movement

VirtusLab worked to reduce unnecessary data transfers and improve how Docker-related workloads were handled in the remote execution environment. The team also optimized the process of loading container images into the Docker runtime, matching the performance of the previous environment.

The largest improvement came from adjusting Docker rules and moving Docker-related actions fully to remote execution. This allowed more of the image-building work to benefit from caching and avoid unnecessary execution. As a result, the post-merge compilation job became three times faster.

Further improvements to Docker image handling and data movement remain part of the ongoing optimization work.

Moving Production Workloads onto RBE

VirtusLab gradually migrated the client's key pre-merge and post-merge workflows to RBE, focusing on the jobs responsible for most of the CI workload. These included linting, compilation, pull request tests, and the main post-merge test suites. The rollout was incremental, with workloads moved and validated progressively before running fully on RBE.

The Results:

Result	Measured or Estimated Impact
Average cost of the main PR tests	20% lower on RBE
Projected annual PR testing savings	More than \$3M
Tests excluded from RBE	Reduced from approximately 11,000 to under 3,000
Heavy tests excluded from RBE	Reduced from more than 55% to under 9%
Unit tests excluded from RBE	Reduced from 10% to approximately 1.2%
Pull requests running on RBE	100%
Post-merge compilation job	3× faster
Test flakiness	Approximately 10% lower

What Comes Next

The migration continues. With the key pre-merge and post-merge workflows already running on RBE, the next priorities are to optimize their performance and cost, work through the remaining incompatible tests, improve Docker handling and resource allocation, and assess which additional jobs would benefit most from migration.

As the monorepo continues to evolve, the remaining work is being addressed iteratively. The direction is already clear: more tests are running successfully on RBE, the new environment is becoming more stable, and the cost of the main PR testing workload has fallen. The team expects further savings as additional workloads are migrated or optimized and the legacy infrastructure footprint is reduced.

Conclusion

At this scale, successful adoption of remote execution depends on treating it as a continuous engineering process across the build system, test suite, and infrastructure. Reliable observability, systematic remediation of recurring failure patterns, and resource allocation based on actual workload behavior make it possible to improve cost, performance, and reliability together as the codebase evolves.

VirtusLab combined Bazel and EngFlow expertise with detailed monitoring, infrastructure optimization, and carefully applied AI support to turn a broad migration goal into measurable results. The migration is still underway, but the cost improvement is already visible before all eligible workloads have moved to RBE and before the client has significantly reduced its legacy infrastructure footprint.

About VirtusLab

VirtusLab helps engineering-driven organizations improve developer productivity and modernize complex build systems at scale. **Our Bazel experts** design, migrate, stabilize, and optimize build and test environments across large monorepositories, CI/CD platforms, remote execution systems, and developer tooling. We combine build-system expertise with production observability, infrastructure optimization, and practical AI-assisted engineering workflows.

The goal is straightforward: faster and more reliable feedback, better use of infrastructure, and **build systems that can continue to scale as engineering demand grows**.

Looking to Reduce the Cost of Your CI Infrastructure?

A growing cloud bill is often only a symptom. The underlying problem may be inefficient sharding, oversized workers, unreliable tests, hidden dependencies, or a build system that no longer matches the scale of the organization. VirtusLab can help you identify those bottlenecks and build a practical path toward faster, more stable, and more cost-efficient CI.

Talk to our Bazel and Developer Experience experts.