StormForge

Case Study

Large technology and consulting firm sees 20% response time improvement, reduced risk of failures with StormForge

Initial testing reveals massive potential improvements for cloud-native app optimization

As a global technology and consulting firm helping clients move to cloud-native architectures and DevOps practices, balancing trade-offs between application performance and cost is always top-of-mind. As the lead DevOps engineer for a large technology consulting organization told us recently, "If you want to create a highly available, highly scalable solution, which pretty much all of our clients do, there's always a cost/value trade-off to that. If we can optimize our customers' apps to run leaner, faster, and more efficiently, that's a great benefit to our clients."

That's why the team decided to look at StormForge. Testing with a simple front-end application, the team saw significant response time improvement of 20% and also identified a number of unstable configurations that would have likely resulted in application downtime had they been deployed to production.

> "We were impressed. We were quickly able to see the value of running StormForge against this application."

Experiment setup

The team began testing with a simple front-end UI app running on an OpenShift cluster. They used Locust for performance load testing, with a run time of 3 minutes and 5,000 users (500 per second).

For simplicity, they optimized for two primary Kubernetes parameters - CPU and memory - with goals of improving 95th percentile response time and identifying configurations that were unstable and would likely fail in production.

With initial settings of 100 m CPU and 128 Mi memory, the starting performance test benchmark measured 95th percentile response time at 35ms.

> "We found installation and startup to be very quick and easy. Installing StormForge on the OpenShift cluster went very smoothly. It was a simple command to install it all. The range of examples was also great, allowing us to quickly jump in and get started."

Results

After running the StormForge experiment, the team chose optimized settings of 187 m CPU and 61 Mi memory. This allowed them to improve 95th percentile response time from 35 to 28 ms, an improvement of 20%. During the course of their testing, StormForge also identified many failed configurations that were filtered out of their configuration options. Finding failures during performance testing helps to identify and avoid configurations that are likely to be unstable in production.

Performance bench test results

Locust performance test parameters

Run time: 3 minutes
Users: 5,000 (500 per second)

Original settings	**Optimized settings**
CPU: 100 m Memory: 128 Mi	CPU: 187 m Memory: 61 Mi
Results:	**Results:**
95%tile response time: 35 ms	95%tile response time: 28 ms

Recommendations

Asked to describe their learnings and recommendations to other organizations getting started with StormForge, the team provided this feedback:

- Choose a small number of trials to start. That will ensure everything is working smoothly and ultimately result in faster time to value.
- Choose a relatively broad range for your parameters. Allow the machine learning to do its work and home in on the optimal app configuration.
- Use the parameter drill-downs to analyze and explore. This will help you better understand your application and how each parameter affects performance.
- Perform both before and after testing. Benchmark performance before starting to experiment and then run the same test after to compare and objectively understand the benefits.

About StormForge

StormForge is changing the way organizations approach application performance.

We combine cloud-native performance testing with application optimization powered by machine learning to proactively improve application performance, stability, and efficiency. StormForge lets you release with confidence to accelerate your move to cloud native and ensure success of your digital transformation while efficiently managing costs.