

CASE STUDY

Olo improves application throughput and database query times with Datadog Continuous Profiler

ABOUT OLO

Olo helps over 600 restaurant brands scale online ordering and delivery, guest engagement, and digital payments. The company's platform reaches 85 million connected guests across approximately 78,000 locations, processing more than two million orders per day on average.



Hospitality



700+ Employees



New York



“Continuous Profiler enables us to dig into the underlying code performance and quickly fix any issues.”

Mike Clark
Staff Engineer
Olo


WHY DATADOG?

- Accelerates MTTR by eliminating the need to reproduce code-level performance issues in their environments
- Reduces application latency and improves the user experience by detecting the slowest lines of code
- Lowers cloud costs by optimizing the most resource-intensive methods and lines of code

CHALLENGE

As Olo introduced an API-first approach and moved away from its monolithic architecture and databases, they experienced a decline in application throughput and an increase in database query time with no clear root cause.

USE CASE

 Continuous Profiler

KEY RESULTS

↑50%

Improved database query times

↑40%

Improved application processing throughput

Modernizing infrastructure to enable better scalability

Olo is a B2B software-as-a-service (SaaS) company whose platform helps restaurants scale online ordering and delivery, make data-driven business decisions, and personalize the guest experience. Olo currently serves 600 brands, and the company's application can service thousands of orders a minute. Olo is a .NET 6 shop, running on AWS across 1,300+ hosts and 300+ microservices. Recently, Olo has been developing and introducing APIs to facilitate better integrations with external systems and modernize its infrastructure for easier scalability.



Instantly identifying inefficiencies and their root causes

To ensure a smooth experience for the 85 million users of its platform, Olo needed a way to understand the root cause of inefficiencies across its .NET services while supporting a high volume of user requests. However, a lack of visibility into production code performance left Olo blind to inefficiencies. Staff engineer Mike Clark sought to improve that visibility, reduce outages without compromising performance, and enable his team to release new features quickly. To address these challenges, Clark turned to Datadog Continuous Profiler.

The first time Clark and his team took Continuous Profiler for a test run, they examined a business-critical service within Olo's platform responsible for propagating menu changes throughout the system. Within five minutes, they found a problem: During a routine load test, the flame graph showed that a significant amount of time was spent on utilizing reflection across the service's entire codebase, which created performance issues.

The profiler enabled the team to locate and fix every instance of inefficient reflection usage. In less than three hours, they saw improvement of more than 40 percent in the application's processing throughput. "Increasing the throughput increases our scale. It increases the volume of traffic that we can accept," says Clark. "As we roll out new features to our users, we see fewer backups, and the system is more elastic under load."

Ability to analyze and compare code performance leads to reduced latency

Clark and his team have been working to build a caching infrastructure to reduce strain on the platform database. As they did so, Clark noticed long database query times with no clear optimization the team could make to the PostgreSQL query. Olo was already using Datadog Database Monitoring, which they used to rule out the database as the root cause, leading them to conclude that the bottleneck was actually the client application. "Datadog Database Monitoring said the issue was not on the database side, and we don't need to serialize the data again, but instead, we need to check on the application side—something that we would not have thought of," says Clark.

Clark then turned to Continuous Profiler and quickly found that their Postgres enums were not mapped properly, which caused the Npgsql library to use a costly path for mapping the unmapped enums.

The team decided to stop using enums and replace it with a more efficient solution for building their cache objects. Within minutes, database query times improved by 50 percent. The long-term benefit was even more significant as Olo was able to improve the documentation of its coding standards to avoid a similar issue in the future.

The company also uses Continuous Profiler for performance optimization. "At our scale, we definitely get into high-volume situations where every little bit counts," says Clark. "Or we get into situations where something's happening in our application, and it's really hard to understand from the outside. Continuous Profiler enables us to dig into the underlying code performance and quickly fix any issues."

"As we roll out new features to our users, we see fewer backups, and the system is more elastic under load."

A go-to solution for incident response

Clark says Datadog Continuous Profiler is easier to set up than traditional profilers. "If you're using a traditional profiler in a certain environment, you have to install it on those machines, which can have a massive impact on performance," he says. "Traditional profilers also require you to know the problem beforehand, and if you don't, you could be left wondering if the investment was even worth it, or hoping it will happen again, or trying to reproduce it yourself."

The Olo team now uses Continuous Profiler as its go-to solution when they are looking for optimizations they can make to various code paths during non-production performance testing. Continuous Profiler is used to conduct deep code-level analysis in such scenarios, helping the team identify any issues and validate fixes while eliminating guesswork.

Solving these issues helps ensure Olo is using resources effectively. It also increases the resiliency of its application, as outages previously prevented teams from building and releasing new features. Finally, it reduces Olo's cloud provider costs by optimizing its most resource-intensive methods and code lines.

[GET STARTED WITH A FREE TRIAL TODAY >](#)