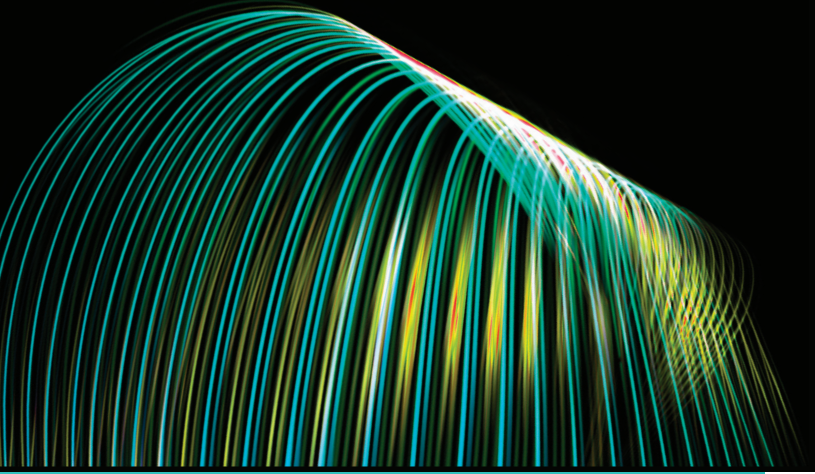
Effective optimisation of systems through early performance testing



The client: A leading, global, professional services firm

Industry: Professional services

Technologies: SaaS, Drupal CMS, MuleSoft ESB, proxy and caching technologies

Ten10 services: Performance testing

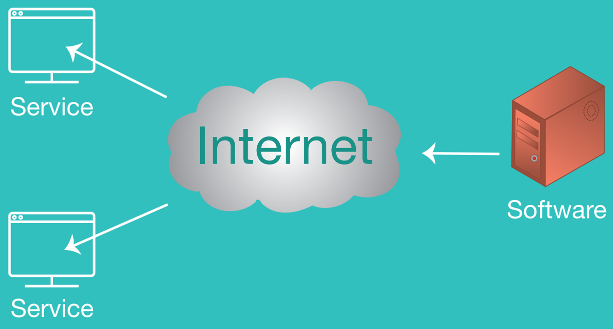
The client

The client is a leading, global, professional services firm. Their aim is to help organisations and individuals create the value they're looking for, by delivering quality in assurance, tax and advisory services.

The project

The client was in the process of developing a new Software-as-a-Service (SaaS) solution that would be used by local councils to offer services to the public, with the additional aim of building best processes and practices.

The development team also saw performance as a key success criterion. The plan was that one particular council would be the initial, pilot user, although the system was designed to support concurrent use by multiple councils in the future.



The client was actively developing the solution during Ten10's involvement. An external supplier managed the infrastructure on which the solution was to be deployed, and made available a number of virtual environments for development and testing.

The solution relied on the Drupal CMS for the front-end and on MuleSoft ESB for the integration with third-party and client services. ForgeRock was used for authentication purposes. The solution also employed proxy and caching technologies, such as Varnish and Memcached, to improve server performance.

The client was looking to performance-test the solution in order to ensure that the production system would be able to handle expected and peak loads, while identifying the capacity of the application under gradually increasing load. Initially, traffic would be coming from the piloting council, but the system was designed to handle to up to five councils concurrently.

Goals

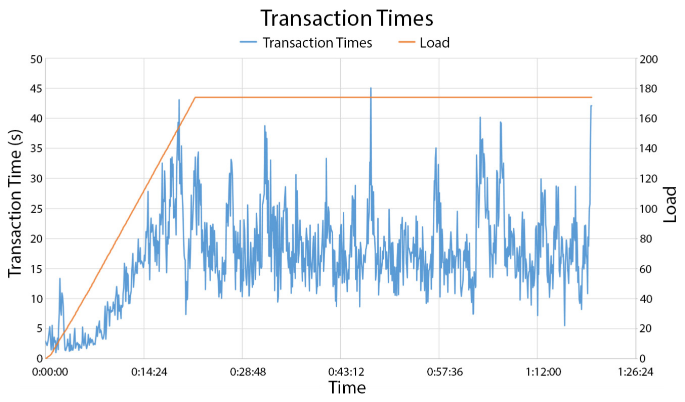
We needed to understand the performance, stability, and capacity of the CMS component of the solution, with expected peak user volumes of 175 concurrent users and a peak load of 300 concurrent users. Specific performance metrics for the system were that 80 percent of all server transaction times should be under three seconds under peak load, that average CPU utilisation was under 70 percent, and that the time taken for the enterprise service bus to process a message was under 40ms end-to-end under peak load.

Further, we wanted to investigate the capacity of the application by gradually increasing the load until the system became unresponsive or the load generation servers become saturated. Following the system stress testing, we wanted to analyse the results, diagnose potential performance issues, recommend actions, and investigate the effect of the identified solutions.

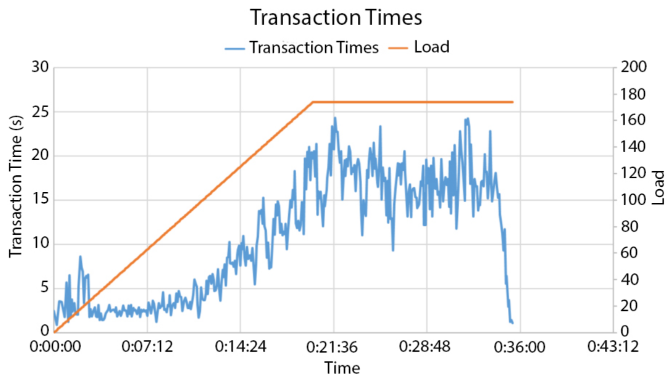
Challenges

Among the challenges were the complexity of the system and the fact that Ten10 was involved at a very early stage of the project, when environments and system features aren't always stable enough to undergo performance testing.

We integrated with the team early on, gathering requirements and aligning the performance testing approach to the development effort. To overcome issues with features not being delivered on time or environments being unstable, and limit delays and disruptions in testing, we followed an agile approach by adjusting load tests to the existing stable functionality. This allowed the early provision of feedback to the development team, the quick discovery and efficient diagnosis of performance issues, and the prompt resolution of bugs.



Before Suggested Fix - Transaction Time Graph and load



After Suggested Fix - Transaction Time Graph and load

Success story

Ten10 integrated well with the development team, which was very supportive, and the load test rig was well provisioned.

Delivery was massively improved due to our extensive analysis and our recommended changes to improve performance. For example, based on our analysis, we recommended a configuration change that improved the system's response times by about 40 percent. We also provided our scripts to the client to allow them a repeatable execution, so that they could validate additional fixes.