TRIGENT

CASE STUDY

Performance Testing for a Pittsburgh Based National Nonprofit Organization

Customer

A Pittsburgh based national nonprofit organization with a charitable mission that is in complete alignment with non-profit communities and organizations. The Company's DonorEdge on-line research engine helps build community knowledge, learn about its mission, programs, leadership and financial information and to make it accessible to inspire charitable giving.

Non-profits and government organizations use DonorEdge's Match Day Performance website to raise funds for charities and communities during specific events, on a day or a few hours, during a race or game or any other activity.

Project Objective

The client helps charities drive donors to the DonorEdge Match Day website during specific events through advertising media. Motivated by radio and other ad campaigns, donors visit the Match Day site and donate to charities. For each donation there is a matching contribution by an organization. This motivates donors and increases the overall contribution. The matching amount is often capped by a limit. As donors donate money, the matching cap reduces by the amount donated. There is a race for donations and the site experiences high and peak loads during these events.

On a match day, it is anticipated that at least 20,000 users would access the application, search for organization/s and donate funds and therefore would like to test the performance of this website for load using a tool that models reality and identify the potential risks by studying the behavior of the system during peak loads. It would also like to ascertain the cluster arrangements or the hardware configuration are capable to handle the load, the user experiences when load increases and also determine the peak load beyond which the site may stop responding to users.

Customer Challenges

The customer was interested to study the performance of the application, the hardware capability and the cluster arrangements on which the applications was deployed. The more interesting scenario was to load the system to maximum of 20,000 users where one-fifth of the users to be loaded initially with no loss of time and rest of the users to be loaded incrementally in 1000 every 60 seconds thereafter. So, within an hour period the entire users access the system and should complete the

Overview

Client

A Pittsburgh based charity foundation, a community to find detailed information about the Western Pennsylvania's nonprofit organizations

Project Objective

The customer was interested to study the performance of the application, the hardware capability and the cluster arrangements on which the applications was deployed

Technology Stack

- ✓ Amazon EC2
- ✓ The Grinder 3.0

Benefits

- ✓ Trigent helped the client understand their application performance on the high user volume
- ✓ Helped to fine tune their cluster arrangements and some of the application configuration settings
- ✓ Recommendation of Amazon EC2 as a load injector machines had both cost benefit and realistic performance testing through rigorous testing

transactions. This necessitated some expertise in performance testing and who could help them understand their application performance on varying load scenario.

Trigent's Solution

Are we solving two issues - cluster load balancing, and performance? If so we need to deal them separately.

Trigent recommended “The Grinder 3.0” - an open source tool that could best fit their requirements. This involved translating the requirements to sense a real world scenario.

Modeled real world scenarios with constant interaction with the client. The servers were not exposed to Trigent and were monitored by the client themselves. Likewise, the Client was not exposed of the tool setup that Trigent were executing. The focal point was the data that were exchanged after every execution of the scenario. The data was analyzed to decide the next course of action under mutual agreement. Below are some of key discussions that exemplifies on how a given problem were translated to viable solution.

➢ Having wait or sleep time between transactions deemed logical

Client Proposed

By design of Grinder it adds a default wait time or sleep time between the transactions as we record the actions. The client wanted all these wait time to be removed and need the transactions be executed instantaneously as it completes.

Outcome

The Grinder was configured to load 5,000 users. As the users loaded to the system, this induced a linear load pattern both on Application and the database server. Either the activity on the server side peaked to 100% or dropped to 0%. Each run exhibited too many failures at different transaction and at varying proportion.

Trigent’s Recommendation

Instead of the fixed default wait or no wait time, based on educated guesses Trigent recommended having a random wait between I second to 10 seconds.

Outcome

The same user load of 5,000 was loaded to the system and this induced a saw-tooth load pattern both on application and the database server. The activities on the server were fluctuating with 30% to 75%.

➢ Seeking the edge of stability for the Application

Well, this was another important and interesting question to be answered. How the execution to be planned, should it be started with the highest or the minimum load to conclusively state the threshold load beyond which the application would start to degrade. Starting with the minimum load seems warranted for the reason that application stability can be studied methodically. For example, the applications was stable and were no failures at some 'x' load and later on as the load increased there were proportionate failures. With this, it would be easy to identify the area of instability in the application.

The application was stable with no failures with the load of 3,000. As we increased the load to 5,000, the application was stable but with some failures in the payment transaction though not significant. We further increased the load to 6,000 and it was observed, the activity of the CPU was 10% ~ 15% more relative to its previous run and the transactions failures were 30% more compared to the earlier run. As we further increased the load to 7,000, and now we saw more than 50% failures and the failures were more in the login page and the payment process. There were failures in the other transactions like search of organization, providing credit card information, add to cart but the failures here were negligible and can be disregarded in this case.

We have been seeking the edge of stability for the Application. The testing of that edge was complicated by the NFG – a third party payment processor. Finally, it was proved conclusively that the bottleneck was in the payment processor which NFG acknowledged is not very fast.

The other significant failure was in the Login page, as we increased the load, the average time at 5000 user load to bring the page up was 3 minutes or even more. Also, the Grinder worked threads were killed while doing a GET request of the image files. A perfunctory inspection revealed that the rich in images in the login page caused degradation.

➢ Throttle to limit the load on extent

We have identified the area of instability in the application - one is in the login screen and the other is in payment processor. After some discussion, it was decided to have a throttle mechanism that would restrain the web users instead of making them wait ending up with no result - as it said a web users have little patience and lots of options. A Governor was setup was set in login screen and at the payment processor to restrain the user load surpassing the throttle value. And many tests was performed at different throttle value and arrived at 7,000 as a safe cut-off. This approach or workaround was more viable for the reason that it did not

involve any development that may demand a re-design to the login screen or find alternate payment process and re-evaluate the performance.

- Recommended Amazon EC2 instances for load injector machines
- Flexibility in script level changes that cater to varying scenarios and changes to application configuration
- Running the test client on discrete machine for realistic performance testing

Client Benefits

Trigent helped the client understand the behavior of the application on the high user load in an easy and pragmatic approach.

Some of the benefits include:

- Trigent helped client understand and ensured their business and mission-critical transactions scale under load and deliver quality web experiences
- Helped in identifying bottle necks in some of the key transactions like login page, payment processor that they believed would handle 20,000 users otherwise
- Helped to identify deadlocks in the database server
- Helped to fine tune their cluster arrangements and some of the application configuration settings
- Recommendation of Amazon EC2 as a load injector machines in conjunction to “The Grinder 3.0” tool had both cost benefit and realistic performance testing
- All the aforementioned details helped the client in knowing the unknowns of the application and potential risks on load ahead of time rather than being a surprise on the last day. This even facilitated to demonstrate of their product to other clients