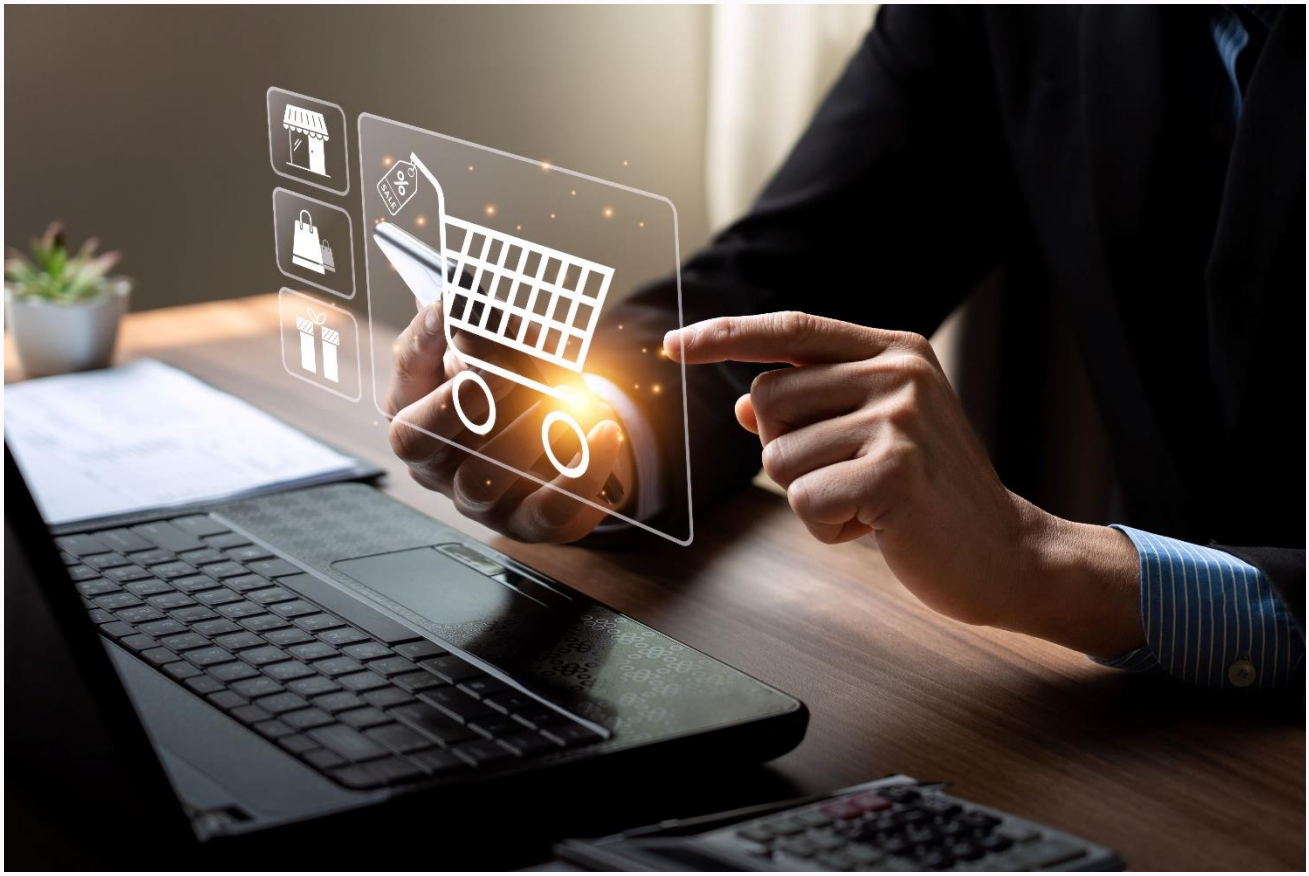
eCommerce Spike Testing Whitepaper



Author: Stephen Jenkins
Performance Testing Consultant.

Introduction

eCommerce companies face fluctuations in demand for products, as demands increase so does the impact on their technology platforms. Their technology must be able to support these fluid levels of load and concurrency while not compromising on application response time performance. We will through this whitepaper, explore how you can ensure that your organisation has a strategy to cope with these demands.

Business Problem

If you are an eCommerce company, then it is more than likely that your technology platform is going to face spikes in load, volume and concurrency.
This is the nature of eCommerce organisations where volumes are fluid and ever changing and can peak at levels well above your average levels of load and concurrency.
As an eCommerce company you need to be able to ensure that these dramatic increases in traffic on your platform can be supported seamlessly without any detrimental impact on the front-end user experience.

This whitepaper is aimed at all members of any organisations technology team and explores the topic of Spike, Stress and Scalability Testing and will look to provide you with an effective strategy to cope with these high-volume scenarios as well as discussing which tools you can use to support this strategy. You will have noticed that we are going to look at **Spike, Stress and Scalability Testing** as ways of understanding the impact of high traffic events on your technology platform and ensuring that you can support these fluctuations.
If we were to focus solely on **Spike Testing**, we may help you understand the impact of your traffic spikes on your technology platform in the short term. But to have an effective strategy to ensure that you have a platform that can support any level of load and concurrency, even those that can be generated by unexpected and unplanned events, these three performance testing disciplines need to be considered collectively. The impact of spikes in traffic to your eCommerce site can be significant, the short-term impact can be slow response times or site unavailability due to server unavailability.
This will affect your ability to sell your products when the demand is high which will have a financial impact on your organisation. The long-term impact can be brand reputational damage resulting in loss of customers, which trust is critical in the world of eCommerce where competition is fierce.

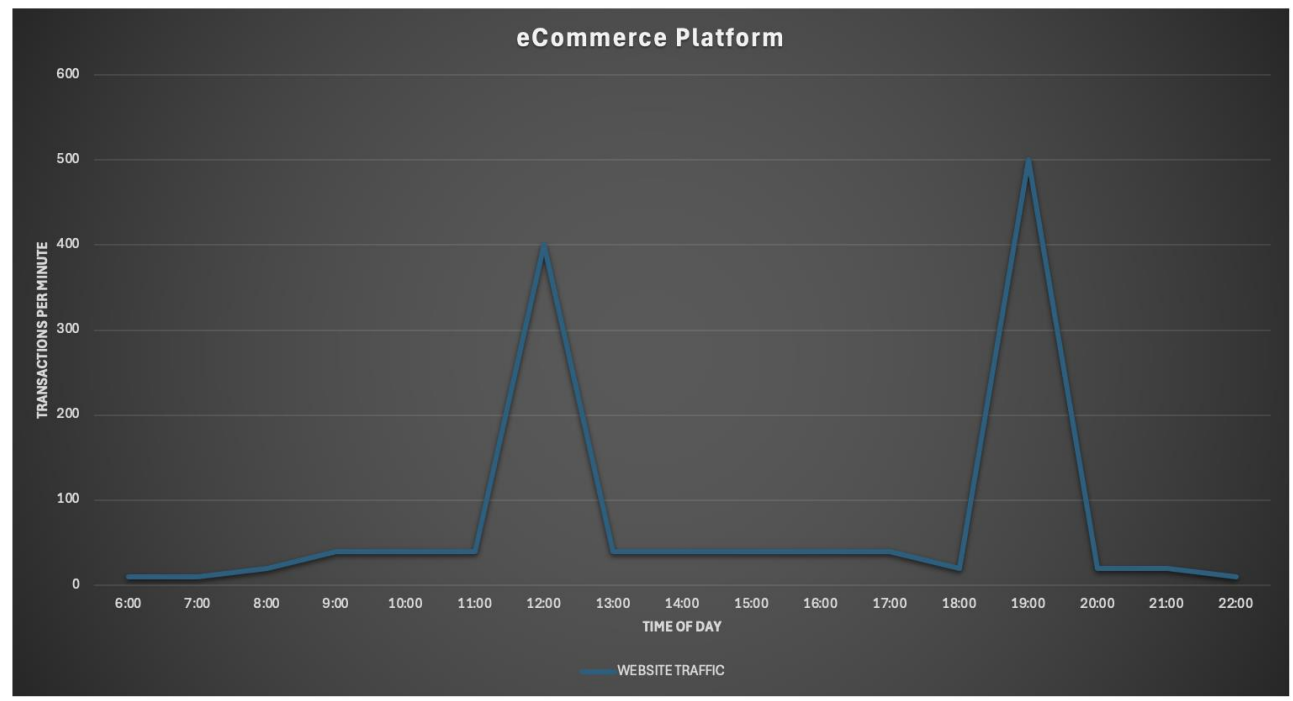
Building a set of Performance Tests to understand the impact of these traffic spikes on your eCommerce platform can be daunting and sometimes not considered high priority by organisations.
This whitepaper will help you understand how to approach this type of testing and provide you with confidence in your organisations ability to handle peaks in volumes. You'll also get real world examples of how to accomplish this and leverage the expertise of Software as a Service Companies (SaaS), such as OctoPerf, to support you in your journey.

On top of that, it will outline the risks of not building and executing this type of Non-Functional Testing with a view to ensuring that organisations see this stage of testing as a much higher priority. Finally, it will introduce you to techniques for ensuring your application is configured to deliver content quickly and efficiently under load, with a view to understand how optimizations you can make will improve your eCommerce platform's performances.

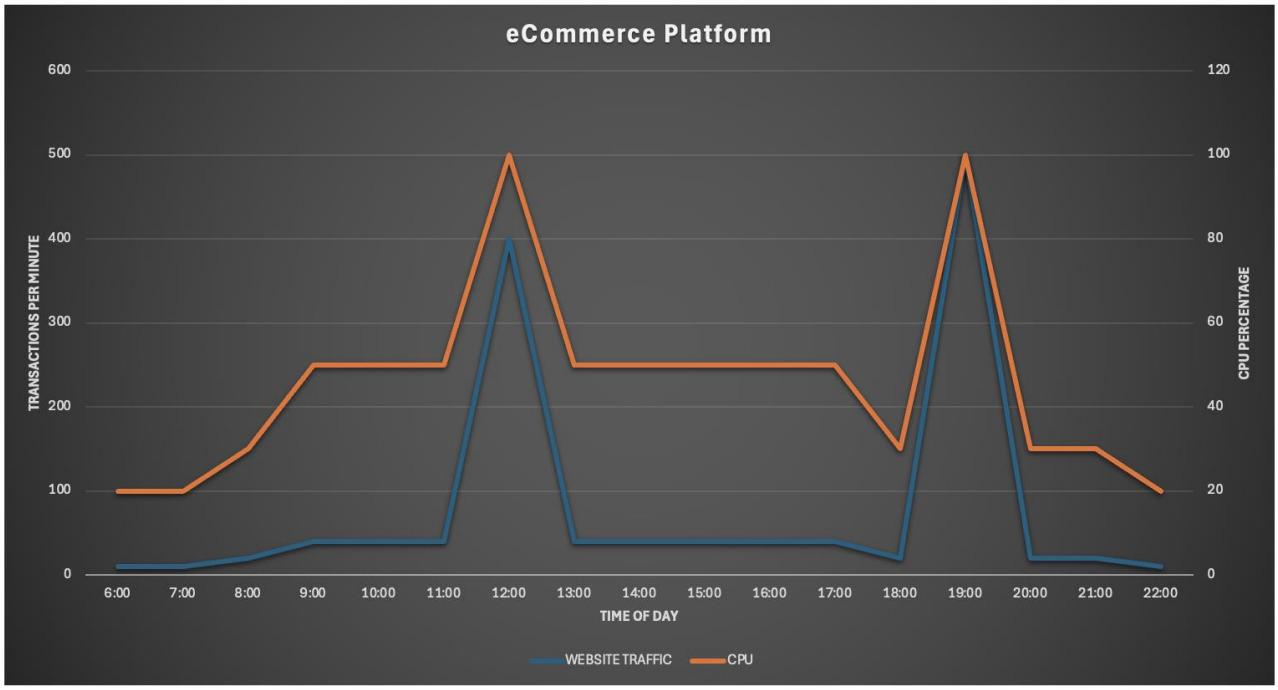
Understanding Traffic Spikes

What do we mean by traffic spikes, these are levels of load that are uncharacteristically high, possibly short lived and inconsistent. Let's look at a couple of load profile examples that will help demonstrate what we mean by traffic spikes, these examples are theoretical and are to help us demonstrate the impact of spikes in load.
We will also consider the possible causes of such spikes in traffic as well as their theoretical effect on the technology platform that host them.

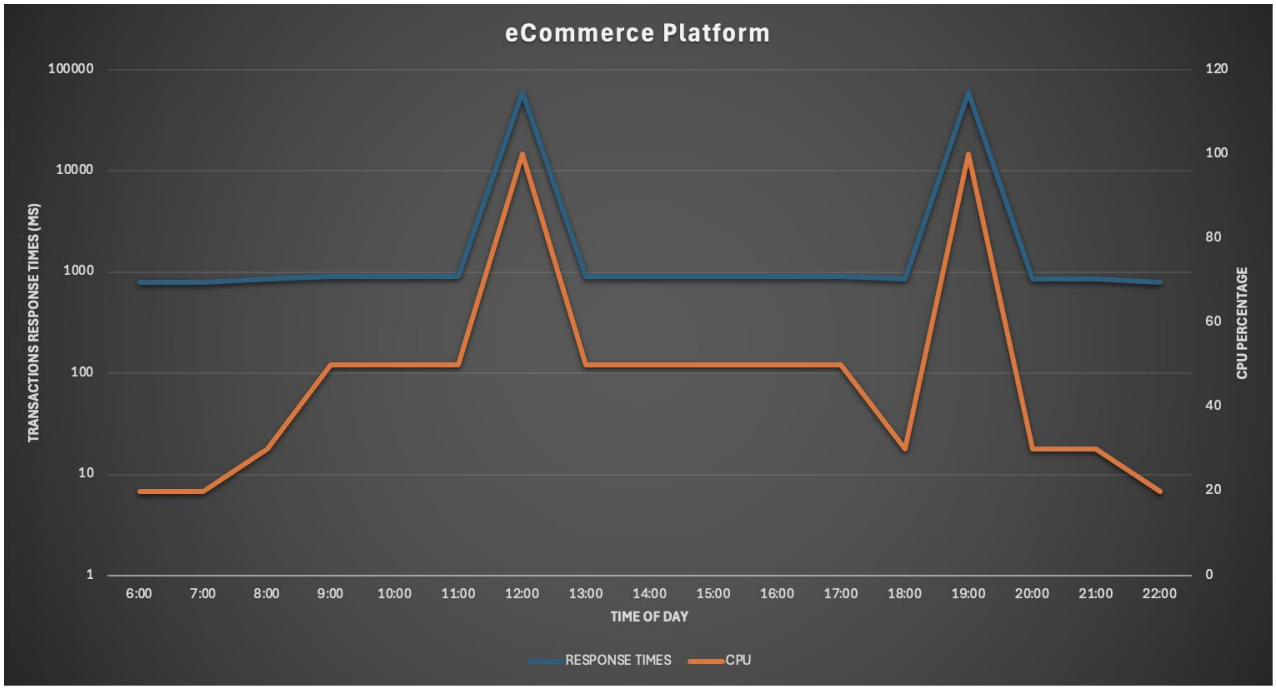
Load Profile One



The first load profile we are going to look at shows two peaks during the online day.
These peaks are **significantly above the daily average** by some margin.
Product launches or music events ticket sales happen at pre-designated times, and this is an example of what this type of user activity might look like in terms of transaction rates.
Consider the impact of the live streaming of a sports event that starts at a certain time of day and lasts a finite amount of time, this again may be the profile you see.
Let's consider the impact of this load profile on your application CPU resource profiles if you did not test these traffic spikes and put in place a strategy for dealing with them.
This is a simple graph for demonstration purposes only and does not account for the possible complexities of your architecture. Increased load across your technology estate will affect front-end, back-end, database servers etc. in different ways and we will look at this later in this whitepaper.



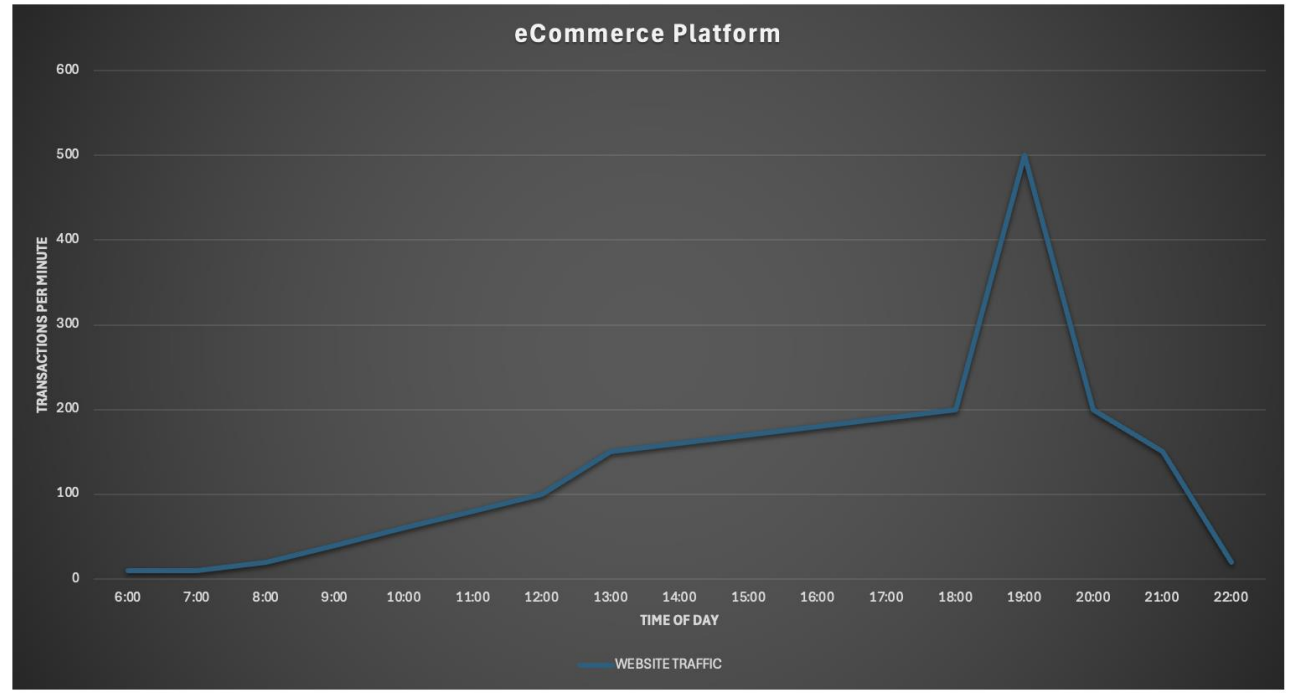
In our example we can see that the spikes in transaction rates have led to 100% of the available CPU being consumed and while our example is theoretical this is more than likely the outcome of this type of spike in load **if you have not tested and planned for this type of event**. The impact of the CPU being fully consumed is that the application response times will be affected.



Our example shows that the CPU being fully consumed has led to the application response times growing to the point that they will more than likely time out, even if they do not, they will make your eCommerce platform unusable.
In our first load profile example we have shown that there is a **direct correlation between traffic spikes and responsiveness of an application** if you have not made provision for these events in the way your application

infrastructure is configured.
This pattern will be true for all the following load profiles as the impact of traffic spikes regardless of its cause and pattern will have the same affect. We have shown, in this example, the impact on CPU, the same will be true for Memory utilisation, Message Queues, Database Connection Pools, calls to Internal and External services amongst others.
The reality is that **traffic spikes will affect any number of components in your technology stack**, and we will look at how you can put in place a strategy for mitigating these excessive loads throughout this whitepaper.

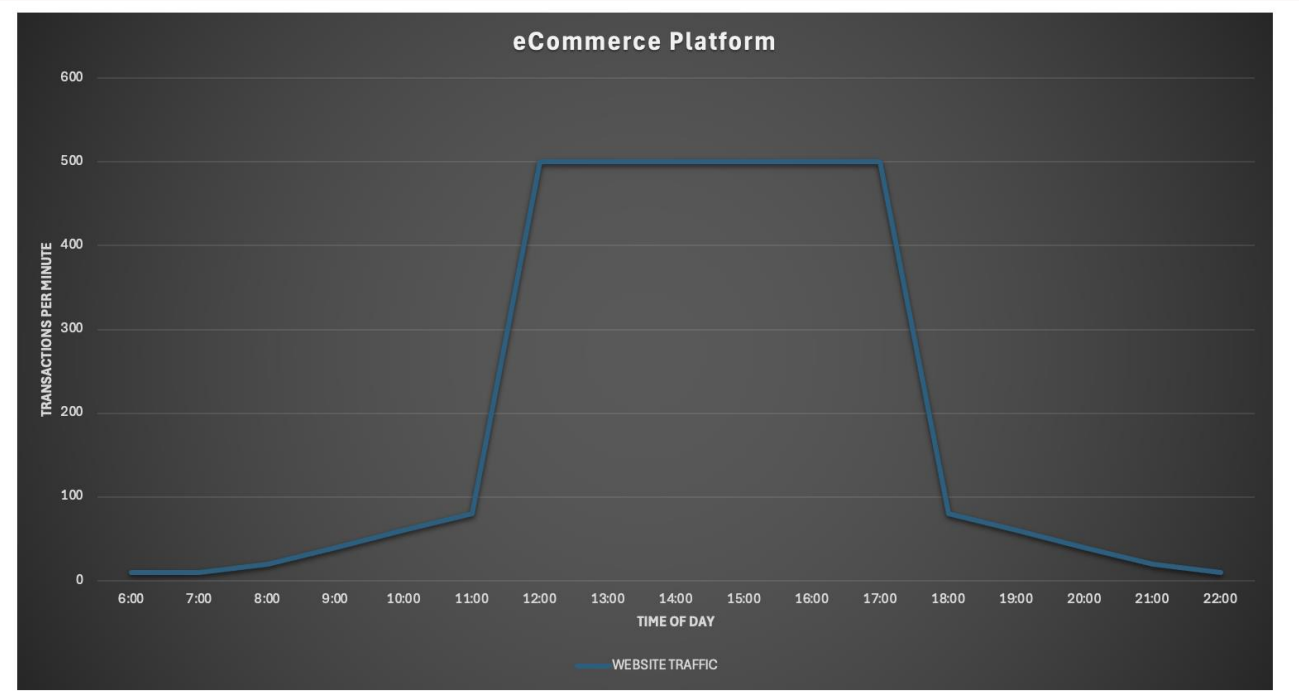
Load Profile Two



The second load profile we are going to look at shows a gradual increase load throughout the online day followed by a traffic spike in the early evening. This evening spike is a significant increase above the gradual ramp up of load that precedes it.
If your company is having a seasonal sale or one of your products goes viral throughout the course of the day on social media, you may see a significant traffic spike in the evening when many people are back home from work and are online.

Evening spikes are not an uncommon occurrence because many people find this the most convenient time to go online, and you will more than likely have a strategy in place to handle this regular increase in load. Performance issues occur when your evening traffic spike is significantly bigger than normal as in the example we have outlined above. The impact on your infrastructure and response times will be the same as **Load Profile One.**

Load Profile Three



The third load profile we are going to look at shows a single longer period of above average traffic. While it is not a traffic spike in the traditional sense it is however a period of exceptional load above what you would normally expect. You would only expect this type of exceptional load if you had an in-demand show you were streaming or a live-event that only you were hosting. Maybe you have a product that only you sell and is on sale as part of Black Friday.

It is important to understand that **traffic spikes can turn into longer periods of high-volume activity**, and you need to be aware of this and have a strategy in place to handle it.

We have shown some examples of spikes in traffic here as a way of demonstrating possible events that can cause these and the impact they would have. Each organisation is different, and each event is different, you need to consider what is best for your organisation in terms of understanding how a spike in traffic will affect you, when it might happen and what you can do to ensure your customer experience is not affected.
The remainder of this whitepaper will be devoted to helping you understand when your peaks will happen, help you build tests to understand their impact and ensure that you have done everything in your power to ensure that planned or unpanned traffic spikes do not have a detrimental effect on your production systems and your end users experience.

Preparing for Traffic Spikes

Now that we understand how traffic spikes might occur and the impact they can have, let's start to look at ways that you can prepare for them. This section will be broken down into three categories:

- Performance testing strategy
- Infrastructure readiness
- Content delivery networks (CDN's)

These three categories have a lot of commonalities and there is overlap between them, we will look at each independently whilst referencing the other categories where necessary.

We have spoken about what traffic spikes look like and their implications, while you might understand when you are likely to get them it is **hard to predict** exactly what level of additional traffic you will see on your site or the duration the traffic spike will last for.

Equally you may not expect a traffic spike at all, they can come out of the blue especially in a world where products and services can become very desirable very quickly if promoted by a celebrity or influencer.
This clearly makes testing for them hard.

You need a testing approach that helps you define a strategy for how you cope with traffic spikes whether they be known, unknown, as part of a planned marketing event or completely unexpected.

Performance Testing Strategy

Look at Historical Volumes

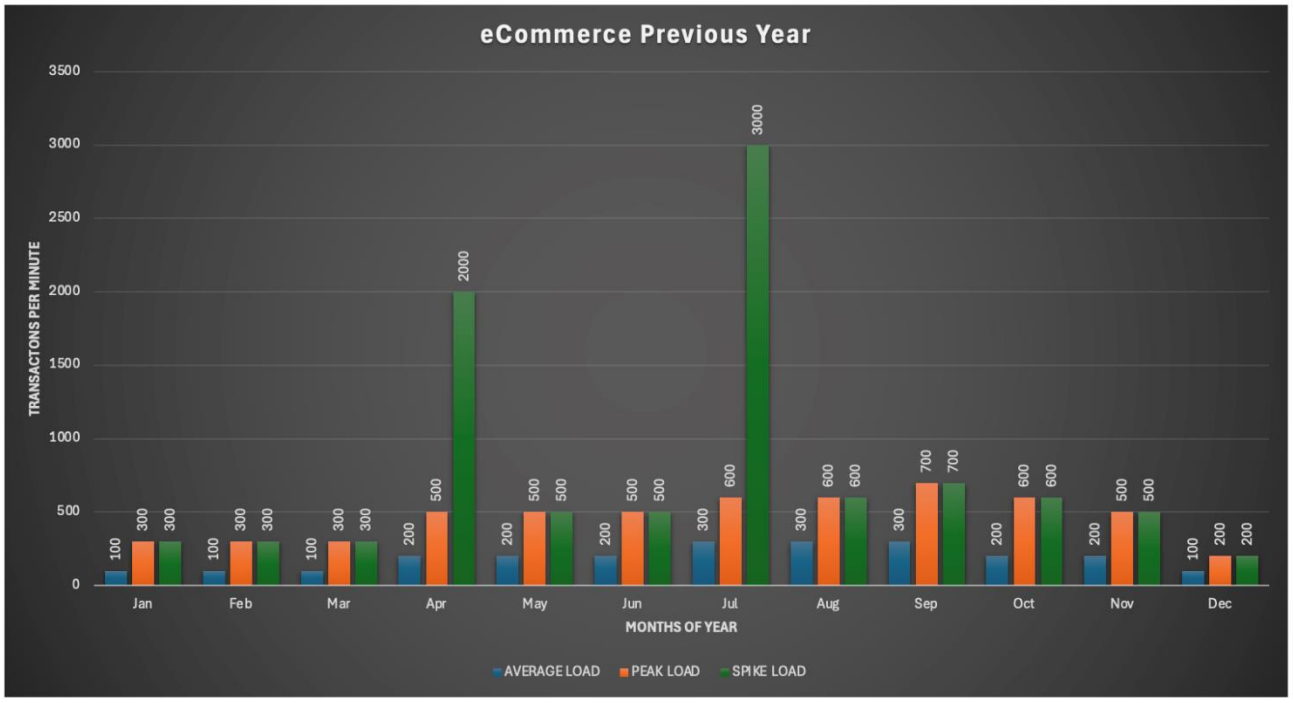
Unless you are a brand-new eCommerce company that has not sold anything before, you will have an idea about peak volumes over the last couple of years and how this has grown and changed.
You may also have data on previous traffic spikes your organisation has experienced before, **this data is invaluable** when looking to build your performance testing.
If you do not have any historical data, then we are still going to follow the same performance testing approach as you would if you did; only we may need to run more tests at varying load profiles which can be avoided with good quality historical data.
We will explain why this is the case later in this section.

Types and Definition of Performance Testing

The performance test we need to execute to allow us to understand the impact of traffic spikes on eCommerce platform are firstly Spike Testing.
The definition of a Spike Test is to place an application or system under sudden increases and decreases of load. This sounds exactly like what we want to do, so why would we need to look at any other performance tests?

You not only need to test at traffic spike levels you understand but also go beyond that and see how that affects your technology stack. If you have not experienced a spike before or are a new company just starting out in eCommerce and you have no evidence of historical levels of load, at what volumes do you start and finish your Spike Testing?

The answer is to compliment your **Spike Testing** with **Stress Testing** and **Scalability Testing**.
Stress Testing is where high levels of load and concurrency are placed on your application and Scalability Testing is where you systematically increase the levels of load and concurrency to understand the impact of each increase.
Let's look at a couple of theoretical examples.



The graph above shows **Average Load**, **Peak Load** and **Spike Load** for an eCommerce company.
Where there are no monthly spikes, we have used the same value for Peak Load and Spike Load.
Based on the above we would want to build a performance test scenario that:

- Starts at 700 transactions per minute.
- Increments at 200 transactions per minute.
- Finishes at 3000 transactions per minute plus 10%.

The reasons being that we know that our application needs to be able to handle 700 transactions per minute to be able to support the weekly peaks you expect, we will use this as our starting point.
We then increase at 200 transactions per minute. This increment is based on what feels about right based on our volumes, there is no formula for this. Increments of 200 mean we only need 13 increment to achieve the 10% increase, ultimately the value of the increment is your decision.
The reason why we increase the spike volumes by 10% is because you would, as an eCommerce business, expect growth year on year, again you may predict higher or lower growth in which case you change the growth percentage.

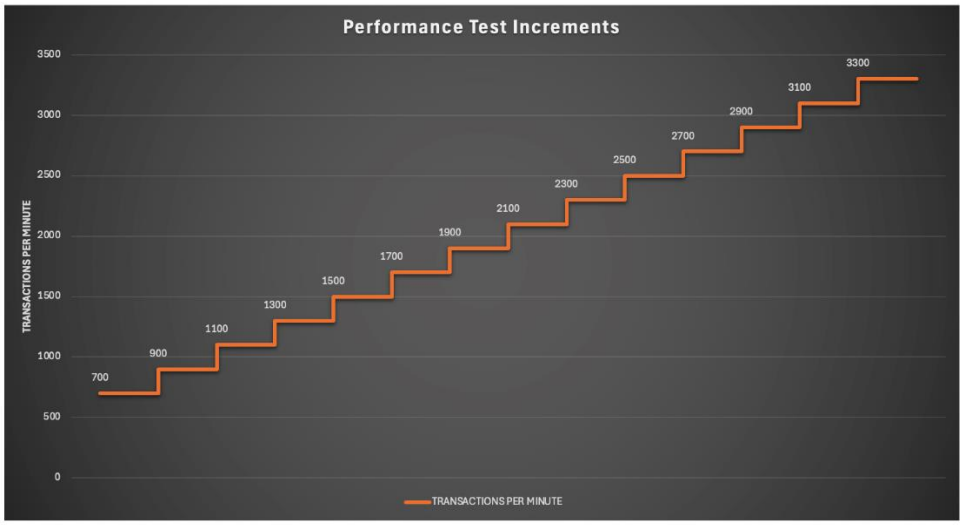
We now have our volumes that we will start our **Stress Testing** at, the increments for our **Scalability Testing** and finishing at our **Spike Test** volumes.
We have discussed already that you may not have any historical data or previously not experienced traffic spikes.

Under these conditions you still need to determine your starting, incremental and finishing transactions rates.
The reason you will not have any previous data must be because this is a new eCommerce platform, or you are selling products that you have not sold previously. This being the case there must be a set of Business Requirements which will hold figures for aspirational sales and growth, you can use these for your starting volumes.

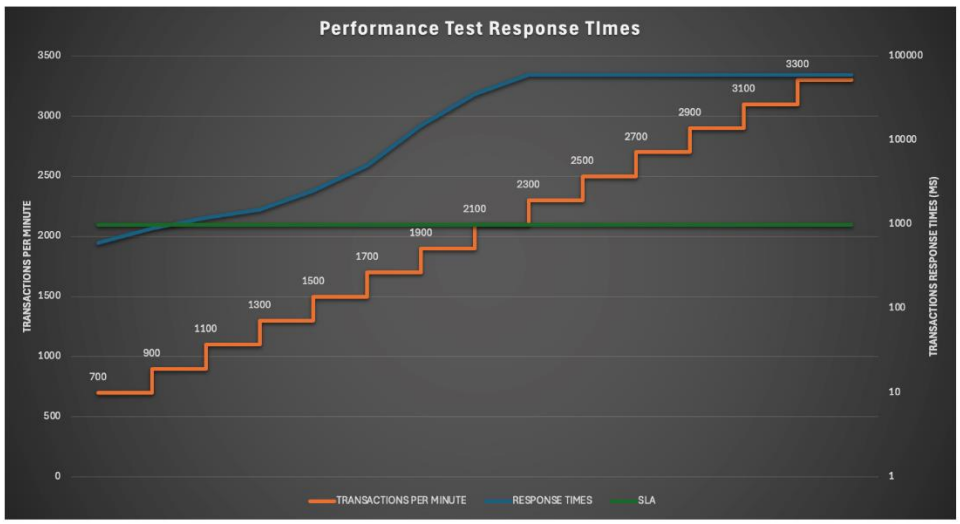
We can determine our **Stress Test** volumes if we use the sales figures derived from the requirements, what we will be unable to determine is the spike traffic volumes which means determining an incremental value is hard. The best approach if to make the increments relatively small which means you may end up running longer tests. In terms of when to stop, this is down to analysis of results and whether you can spot a linear pattern between load, response times and server resource utilisation.
We will discuss this next.

Test Execution

Whether you have figures for start, increment and finish load volumes or only a starting point the way you run the tests is the same. Let's look at an example:



The above profile shows the scenario we would need to build to satisfy our example we looked at in the previous section.
Where we incrementally increase the load from or peak volumes to our traffic spike volumes plus 10%.
Let's look at a theoretical example of the response times we might see during this test, and this will help understand why this type of Performance Testing for traffic spikes is important.

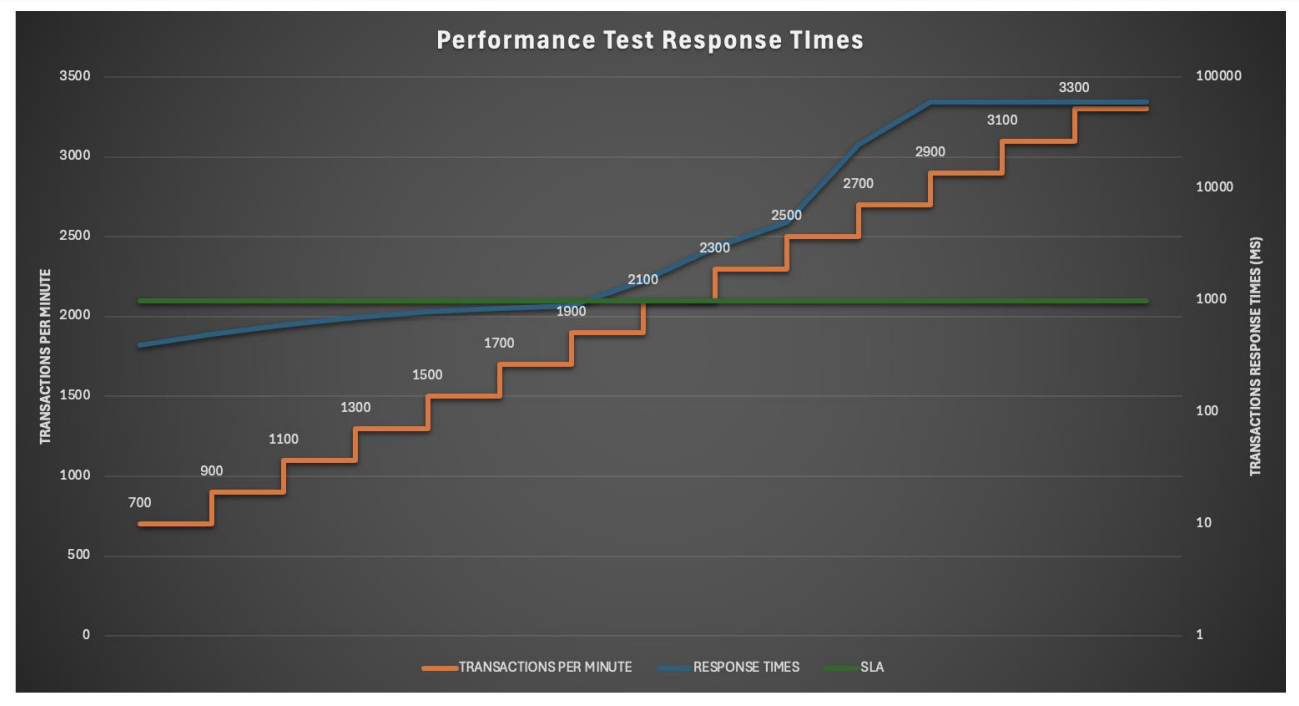


We can see that as our load profile increases so do our response times.

We have added an artificial service level agreement value of 1000ms that we must meet for all transaction response times, you should have agreed with key stakeholders through your Non-Functional Requirements what this value needs to be for each transaction you are testing.

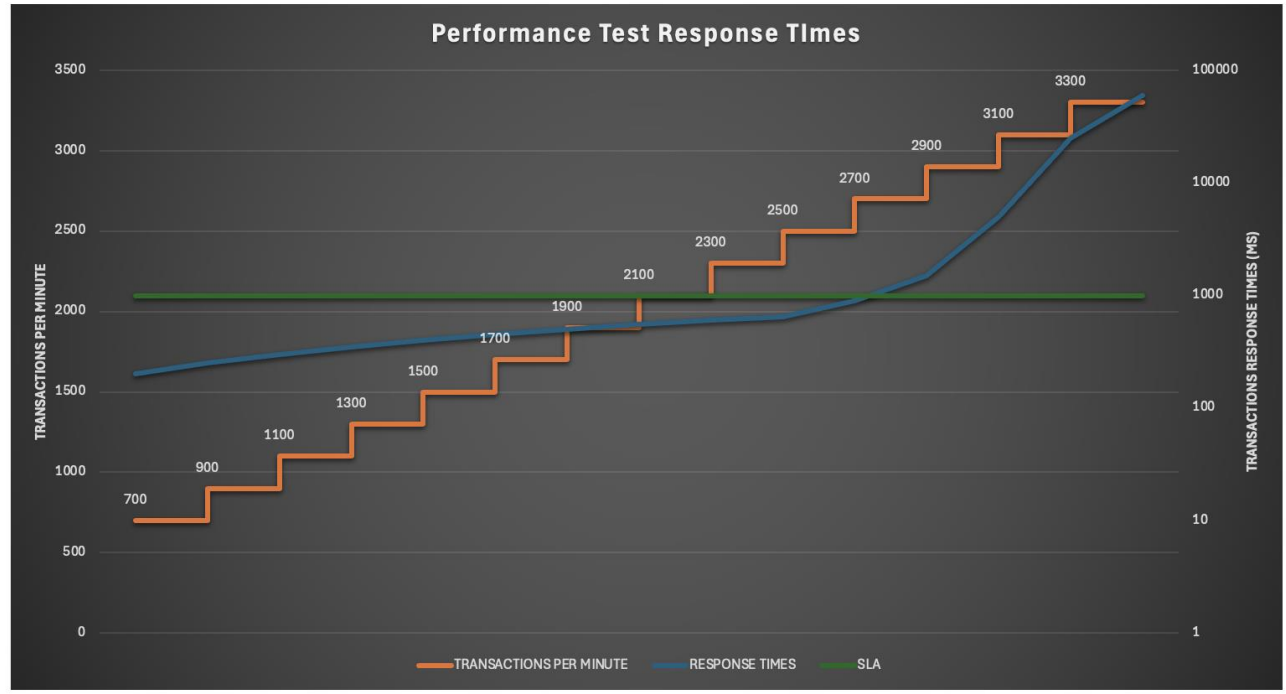
This shows that at a rate of 900 transactions per minute we meet our response time and above that we fail to meet this.
With this information, we can determine that our application as it is currently configured can reach expected peak traffic volumes and can exceed these by approximately 200 transactions per minute and still meet our requirements in terms of service level response time.
Let's assume that our application is hosted on two servers each with 4 CPU's and 10GB RAM.
We would now look to scale these resources to see the impact on response times.
When Scaling you have two options, scale horizontally by increasing the number of servers or scale vertically by increasing CPU and/or Memory of your existing servers.

For the purposes of this theoretical example, we will scale horizontally and add two additional servers with 4 CPU's and 10GB RAM.



After scaling we see that at a rate of 1800 transactions per minute, we meet our response time target of 1000ms and above that we fail to meet this. We have demonstrated that doubling the number of servers effectively doubles the number of transactions per minute we are able to achieve before our response times exceed our non-functional requirements.

We still have not reached our traffic spike volumes so we will scale our theoretical servers again by adding another two servers, to give us six in total, with the same CPU and Memory settings.



We now have managed to run our test at a rate of 2700 transactions per minute before breaching our service level threshold.
This shows that our **application scales linearly** and we can therefore assume that adding a further two servers will give us enough application resources to support our expected traffic spike.
More importantly, it shows that if for any reason your traffic volumes exceeded your growth expectations you would be able to calculate the number of servers you would need to support this level of load.
In our theoretical example we have demonstrated that we can support:

- 2 servers = approx. 900 transactions per minute
- 4 servers = approx. 1800 transactions per minute
- 6 servers = approx. 2700 transactions per minute

You would feel confident, using this data, that eight servers would give you a throughput of approx. 3600 transactions per minute.
Our testing allows us to **understand what we can expect under peak loads** and what we must do from an infrastructure perspective to support these levels of load.
Having this knowledge allows you to have a **strategy for high traffic scenarios** and allow you to be able to react to them.
Our example is simple, and your application will probably not scale in such a nice linear way as we have shown here. Whilst it may not be as simple, the techniques and principles we have discussed are still valid and non-linear response time extrapolation just requires more testing and alternative ways of scaling.

Infrastructure Readiness

We have seen in the **Performance Testing Strategy** section above that how your infrastructure is configured is paramount in ensuring your application performance response time thresholds can be achieved under any level of load.

Our theoretical example demonstrated that to achieve our desired traffic spike volumes we would need to horizontally scale our infrastructure from our starting point of two servers to eight servers.
The example we used was a very simple one and easily solved with adding more servers, your experience may be different, and your architecture will be much more complex.
In the real world your infrastructure may consist of Web Servers, Load Balancers, Database Servers, Application Servers, Message Queues amongst others.
Each of these servers will need to be assessed under load and multiple, if not all, servers scaled to support your traffic spike volumes. We will look at how you can ensure that your infrastructure is as ready as it can be to support your high-volume loads while still meeting your response time requirements.

Scalable Hosting

Scalable hosting is where your infrastructure is designed to scale as your load increases.
As we have already touched on the fact that your technology stack is likely to be complex and contain many types of servers.
And that it is possible that you not only need to scale horizontally by adding more instances but scale vertically by adding more resources such as CPU and / or Memory to your existing servers.
Your testing will help you define the extent of what components need to scale and the nature of the scaling but in practical terms you need to only scale under traffic spike conditions as you will have sized your infrastructure to support your normal levels of peak load. Let's consider the example we discussed in the previous section, where we had to increase the number of servers instances from two at our peak load volumes to eight to support the traffic spike volumes.
If we, as an example, consider that this may also be the case for the Application Servers and Load Balancers where we scale horizontally, and also assume that we need to increase the CPU and Memory on the Database Server to support the traffic spikes. In this example we have:

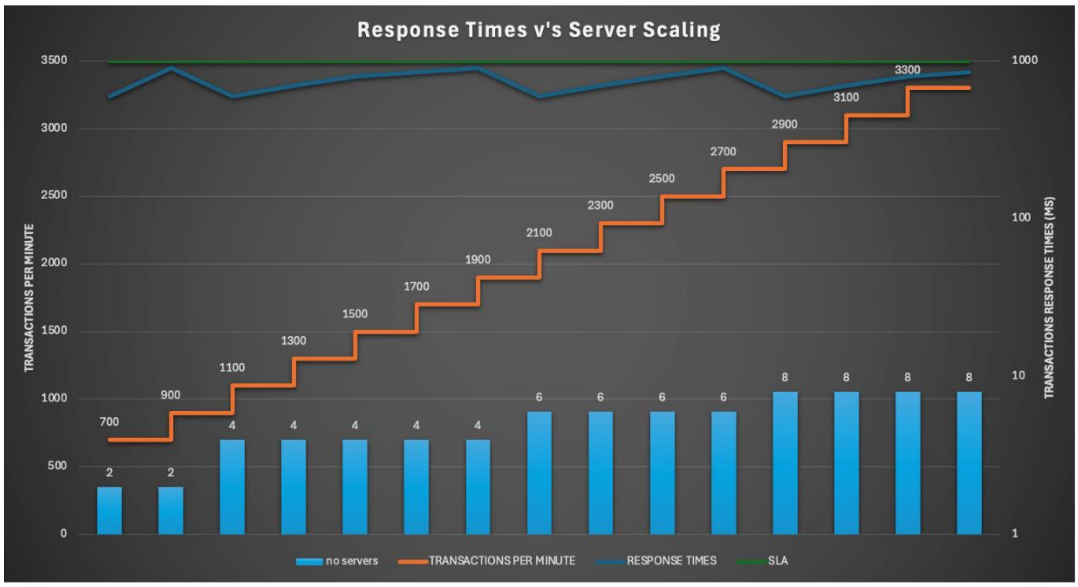
- 8 Web Servers
- 8 Application Servers
- 8 Load Balancers
- 1 Database Server with twice the amount of CPU and Memory

If you sized your environment like this then under normal peak load conditions, it would be massively underutilised.
Also, in comparison to the infrastructure you need to support your normal peak load volumes, would cost significantly more money to host.
So, you know you need to scale significantly to meet your spikes in load and concurrency on your eCommerce platform, but you don't want to keep an environment running 24 hours a day being hugely underutilised.
This is where scalable hosting can really benefit you.
Scalable hosting automatically scales both horizontally or vertically or both depending on the way you configure your infrastructure as your load increases.

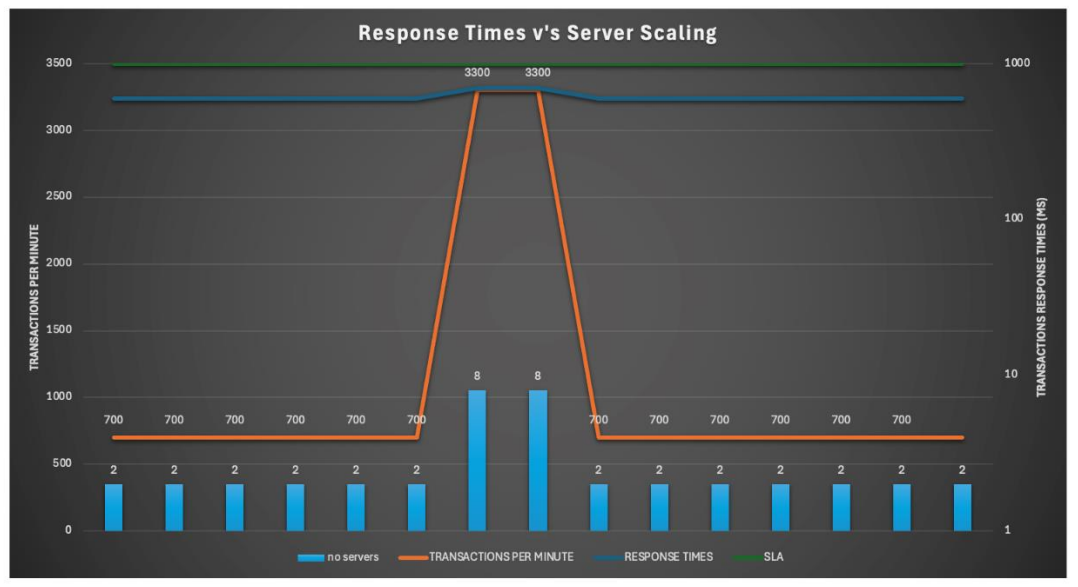
You can configure thresholds in terms of application resources and when these thresholds are breached more instances or more resources automatically become available to your server estate.
So, your technology platform scales up and down based on demand and your performance testing provide the information you need to set these thresholds.

Scalable hosting is a subject that would warrant a whitepaper of its own due to the depth of the subject, for the purposes of this whitepaper we will only discuss its benefits at a high level.

For the purposes of this whitepaper, let's extend our example and consider how the environment might scale as our load increases, and we head towards our traffic spike.



The above diagram shows how a correctly configured scalable hosting solution will increase as does your load profile.
The response times are kept below your response times defined in your non-functional requirement by the fact that when the threshold is close to being breached, more servers are added to support the load.
As with all our examples in this whitepaper, this is theoretical, but if your performance testing philosophy is well structured and robust then this should allow for these outcomes in a real-world scenario.
The diagram above shows a gently increase in load and not necessarily one that is reflective of a traffic spike.



The same principles are true with a spike in traffic as in the diagram above.

Instead of a gradual increase in the number of servers being used to host your application the number jumps from two to eight in reaction to a spike in the levels of load on your site.
When the traffic volumes reduce to a more average level of load, the number of servers scale back as quickly as they scaled up.

Dedicated and Shared Hosting

Another consideration in terms of your infrastructure is how it is hosted is whether you opt for dedicated or shared hosting.
As the names suggest dedicated hosting is a server or several servers that are dedicated to your eCommerce platform where shared hosting is where your platform is hosted alongside others on the same physical hardware.
There are benefits and drawbacks for each hosting solution, let's look at a few.

Shared Hosting

Some of the benefits of shared hosting are:

- Your infrastructure costs will be cheaper to run, with a shared hosting model the running costs are distributed amongst the companies that are hosting applications and services on them.
- The maintenance of the infrastructure will come as a service from the hosting provider as no-one company will have responsibility for this activity, this gives you peace of mind that your technology platform will always have all essential security and software patch applied.
- Shared hosting companies normally offer a reliable 24/7 support model so you can be sure that any issues can be escalated quickly and efficiently.

Some of the drawbacks of shared hosting are:

- A Cyberattack on a site that is hosted on infrastructure that you share can have a knock-on effect on your eCommerce platform. Consider a denial-of-service attack on a site that hosted on the same servers as your site. This could cause the server to crash directly impacting your technology platform.
- Updating hardware and software versions must be done in conjunction with all other organisations hosting on the shared platform. This means the upgrade paths are slower as the hosting company need to provide enough time for all parties hosting on the same platform time to test the hardware of software updates on their applications.
- Your application performance could be compromised, and this is probably the most significant in terms of what we are discussing in this whitepaper. On shared infrastructure you may find that you are unable to flex your server resources to meet you high traffic spikes due to other companies on the same platform as you looking to increase resources for their platform. Consider Black Friday where most eCommerce sites are busier than normal and the likelihood for traffic spikes is quite high, this might affect multiple companies all being hosted on the same platform, meaning you are all fighting to utilise the same resource to increase the capacity of your site.

Dedicated Hosting

Some of the benefits of dedicated hosting are:

- You have complete control over how you flex your server resources to meet your traffic spikes and regular high-volume times, you can operate your server in a way consistent with our scalable hosting

solution we discussed in the previous section. You can therefore feel confident that you can follow your strategy for traffic spikes without being compromised by the hosting solution.
- Being on a dedicated server means that you will be able to manage the server how you want to, you will have root access to install, upgrade and maintain the hardware and software version running on its in line with your development and integration strategy.
- You are not running your site on a platform shared with companies that do not implement strong security protocols. Because you have complete control of the security policies on your server you can implement the degree of security measures your type of organisation demands, which if you hold a significant amount of sensitive or regulatory data is extremely important.

Some of the drawbacks of dedicated hosting:

- It's going to cost a lot more for your hosting costs when compared to shared hosting. Ultimately you get much more flexibility on a dedicated hosting platform and certainly from a scaling perspective offers much more flexibility.
- You are going to have to supply the expertise to manage, run and maintain the platform, this is in the drawback section in as much that this is an overhead. It could be considered a benefit that you are employing skilled people that are keen on exploring the limits and possibilities of the platform and ensure that your delivery methodology meets your demands as a business.
- You will need to provide a solution for application and server monitoring and instrumentation, and again the drawback is the cost. You will at least be free to choose your application monitoring toolset and configure it as you wish. So, this could be considered a benefit.

Content Delivery Networks

Before we look further at the subject of Content Delivery Networks it is important to clarify a couple of things. When we talk about content, we are talking about Internet content such as:

- HTML pages
- JavaScript files
- Stylesheets
- Images
- Videos

This content must exist on a Web Server and a Content Delivery Network cannot host this content, a Content Delivery Network will cache this content at the network edge bringing it closer to the end users.
The content must still be served from a Web Server that meets your performance requirements under traffic spikes and high concurrency load conditions.
Additionally, the actual transactional processing on your site, a few examples being:

- Storing as a favourite
- Adding to basket
- Purchasing
- Setting up reminders and notifications
- Getting detailed product descriptions

These requires interaction with your back-end services and this is why we are looking mainly at performance testing your application to ensure that traffic spikes do not cause a degradation in performance.
While Content Delivery Networks are absolutely something that will improve the responsiveness of your site

navigation in terms of rendering content, **this must be implemented in conjunction with a robust, well tested and scalable Front-End and Back-End application**, hosted to support your peak volumes.
The benefits of using a content delivery network are:

- Improving website load times, by distributing content closer to website visitors by using a nearby content delivery network server, visitors to your eCommerce site experience faster page loading times.
- Your end users are more likely to click away from a slow-loading site, a content delivery network can reduce this and increase the amount of time that people spend on your site.
- Bandwidth consumption costs for website hosting is a primary expense for eCommerce platforms. Through caching, content delivery networks can reduce the amount of data an origin server must provide, thus reducing hosting costs for website owners.
- Content delivery networks can handle more traffic and withstand hardware failure better than many origin servers.

To improve speed and connectivity, a content delivery network will place servers at the exchange points between different networks.
These are known as Internet Exchange Points and are the primary locations where different Internet Providers connect.
By having a connection to these locations, a Content Delivery Network provider can reduce costs and transit times in high-speed data delivery.
Content delivery network also minimise down-time for your applications by:

- Load balancing distributes network traffic evenly across several servers, making it easier to scale rapid boosts in traffic.
- Failover provides uninterrupted service even if one or more of the servers go offline, the failover can redistribute the traffic to the other operational servers.

You can see by the benefits we have listed above that using a Content Delivery Network for your eCommerce site will be beneficial in ensuring your web content is served to your end users as efficiently as possible while also being able to help manager the traffic spike volumes you will encounter.

Optimising Site Performance

We have discussed the issues that you will face during periods of high traffic volumes without a clear strategy for managing these spikes.
We have theorised how you might go about performance testing these spikes and the benefits of scalable infrastructure along with the implications of how and where you host your site and its content.
We will soon start to look at how you put this theory into practical use and the tools that will benefit you.
Before we do this there are a few other techniques you can employ on your site to ensure that the best possible experience, from a performance perspective, for end-users can be achieved.
These optimisations should be employed on your test sites as well as your production platform.
Your performance tests should be executed in parallel with testing how these optimisations can improve your sites performance.
While these are all best practises for all eCommerce sites, you need to understand how they affect performance and which ones give you the most benefits.

Image and Media Optimisation

Image Optimisation

When it comes to **Image Optimisation** for your site you need to mainly consider the File Format you are using and the type and level of Compression you use.
A combination of these will ensure that your images load efficiently while making sure you do not lose any quality.
Let's start with discussing compression methods, of which there are two types:

- Lossy compression that eliminates some of the data, this will degrade the image, so you'll have to be careful of how much reduction you apply. Lossy image compression retains the most significant information for the image without keeping every single pixel, the algorithms work by removing information from the image file so that the file is comprised of fewer bytes.
- Lossless compression, this doesn't reduce the quality, but it will require the images to be uncompressed before they can be rendered. While lossless compression can reduce image file sizes by as much as 40%, it is still less effective than lossy compression for reducing file size and optimizing images for the web.

But before you start modifying your images, make sure you've chosen the best file type for displaying images on websites.
There are several types of files you can use:

- JPEG, named after the Joint Photographic Experts Group, is perhaps the most widely used image compression method. It can usually compress files by a ratio of 10:1 with a minimal reduction in image quality. More recent versions of JPEG include JPEG 2000 and JPEG XR, but many browsers do not support these formats.
- WebP which also supports lossless compression, it is more commonly used for lossy image compression. Google originally developed WebP to replace the JPEG, PNG, and GIF file formats.
- The High Efficiency Image Format (HEIF) is a type of container file for compressed images, images compressed with this method are sometimes called HEIC files.
- PNG which produces higher quality images, but also has a larger file size. It was created as a lossless image format, although it can also be lossy.
- GIF only uses 256 colours and is the best choice for animated images and only uses lossless compression.

Before moving on we should consider the concept of Lazy Loading images.
Lazy loading is the principle where images that are not immediately visible on the page, because they will only be visible when scrolled to, are not loaded immediately.
As the user scrolls and the image becomes visible the image is loaded at that point, if the use does not scroll the page to reveal the image is never loaded.
This has two main benefits:

- Reduces the number of images that need to be initially loaded on the page, this means that you are reducing the number of bytes to download and reducing the bandwidth required meaning the rendering process is much quicker.
- In terms of your eCommerce site hosting charges is costed on the number of bytes transferred. As we have already mentioned if a end-user does not fully scroll your websites pages fully then images are not loaded therefore costs are reduced.

Video Optimisation

Video optimisation follows similar principles to image optimisation.
Firstly, following on from the image section above, you should be using lazy loading with you video files as well as your images for all the same reasons. Like images you should be experimenting with the best File Format for your videos.
There are many video formats, but some either cannot or should not be used on your website, we will look at the ones that you could consider:

- MPEG-4, also known as MP4, is the most common file type for videos on websites because MP4 videos are high-quality with relatively small file sizes. The only real downside to the MP4 format is the fact that the encoding and decoding processes require a lot of resources.
- AVI files offer very high-quality audio, which is a feature you don't get in some of the other video types. Because of this, file sizes are typically much larger especially when using lossless AVI files as these have not been compressed.
- WebM was developed by Google and released in 2019 and is specifically designed for the web as they have an extremely small file size without sacrificing too much in terms of quality. However, Internet Explorer and Safari don't offer support for WebM videos unless you use additional plugins.

Let's also consider how you can use Compression to optimise your video for your eCommerce site.
Video compression is achieved using a Codec which is a specific algorithm device or program that can compress video and audio data. Its name stands for "compressor-decompressor," as it can be used to encode or decode a data stream or signal.
As with images there are two types of compression methods or codecs:

- Lossy codecs which create smaller files overall and easier for all types of digital transfer and delivery. Lossy files are compressed following a method that is unnoticeable to most of your site visitors, which means that they are a great option for any video being streamed online.
- Lossless codecs are larger files overall but still compressed and useful for performance gains on your site. Unlike lossy, though, lossless files keep all the original data intact which increases the file size, causing slower playback, but results in better quality.

Another technique to consider is to reduce the number of HTTP requests by merging multiple video files into one to improve the server response time and reduce the number of HTTP requests.
Also, you could consider optimising your thumbnails or preview clips to give the perception that videos are loaded.

Front-end Optimisation

We have already discussed techniques for optimising images, and we are now going to look at some other techniques for how you can **improve the performance of your front end** which can make all the difference, from a performance perspective, then your site is experiencing traffic spikes.

Asynchronous JavaScript Loading

Many websites load content that's written in JavaScript from top to bottom on a webpage.
So, even if a user's web browser performs more than one HTTP concurrently, the content it receives loads sequentially. This is known as Render Blocking and can make your entire webpage load more slowly because each file is waiting its turn to load in a user's web browser.

Loading your JavaScript content synchronously will allow multiple page elements to be loaded at the same time, no matter where they sit on the page.

Optimising CSS Delivery

We discussed above that JavaScript loading is by default Render Blocking, but this can be improved by using asynchronous loading techniques.
The loading of CSS files is Render Blocking and there are no techniques to avoid this, although the CSS file is cached after the first request which will help with subsequent page loads.
What you can do is consider some of these options for improving the performance of the CSS fetching and parsing:

- The larger your stylesheet, the longer it takes to download and process into a CSS Object Model, which the browser and JavaScript APIs can use to display the page. Even though CSS files are small in comparison to other files on your website, smaller is always better.
- CSS can reference other stylesheets using @import rules. These imports block the processing of the current stylesheet and load further CSS files in series, so avoid this technique is beneficial.
- Browsers have three rendering phases, layout, paint and composite, if you're not careful, CSS property changes and animations can cause all three phases to re-render.
- Combining CSS files is beneficial, a single file requires just one header and can be gzipped and cached more efficiently. Separate CSS files are only practical when you have one or more stylesheets that are changed frequently, even then mostly static CSS code can still be combined into one file.
- Tools can encode images to base64 strings which reduces the number of HTTP requests, but it harms CSS performance. These strings can be 30% larger than their binary equivalent and browsers must decode the string before an image can be used, and altering one image pixel invalidates the whole CSS file. Only consider base64 encoding if you're using very small, infrequently changing images
- The <link> tag provides an optional preload attribute that can start a download immediately rather than waiting for the real reference in the HTML.

Caching Strategies

We are going to look at how Web Caching can improve the performance of your eCommerce site. There are a couple of different techniques but before we look at these let's just familiarise ourselves with the process:

- When an end user visits your site their web browser will check for the page's data in their cache.
- If it is not there, a cache miss occurs and their web browser will then fetch the website resources directly from your servers.
- However, it is in their cache, their web browser will send an HTTP request containing an entity tag which is an HTTP header specifying which site version is in the cache.
- Your server verifies whether the cache has the latest version of the website and if it doesn't the server will send the updated resources.
- Otherwise, your end user's web browser will display the requested site using the pre-existing cache.

There are two types of web caching: **Browser Caching** and **Server Caching**. Let's discuss these.
Browser Caching, or client-side caching, stores the website's content on the end user's local machine.
After displaying a website, the page's resources are stored temporarily.
On the next visit, the website can load using the browser cache, eliminating the need to connect to the server repeatedly.
Browser Caching is effective as the site's resources are stored locally.

Server Caching saves the cached website's data on the server and after processing the requested resources into an HTML file, the server will store it temporarily.
When a user revisits the same web page, the server has the requested HTML file ready to send. It doesn't have to recompile the queried resources, shortening the process. Server caching is usually accomplished using a Content Delivery Network, the benefits of which we have already discussed. Server Caching can be broken down into different caching systems:

- Full-page
- Object
- Fragment

Full-Page Caching stores a copy of the entire page, and it allows the server to send the requested page immediately without compiling the required resources first.
This Caching system can speed up pages that most of you end users will visit, such as the homepage or product pages.
The Object Cache stores the database query results, if the server receives the same content requests, they will be served using the cache instead of the database which minimises the server load.
Fragment Caching stores specific elements of a website on the server.
These Cached elements are typically static, including the title page, widgets, and extensions.
Since the Cache already provides some of the resources, the server will send less data resulting in fewer files which speeds up data transmission.

Database Management

The final section on optimisation in on how you can make your Database as efficient as possible. Part of this involves Indexing and Query Optimisation but we will not be discussing these in this section.
Ensuring your Database tables are correctly Indexed and all Queries are as efficient as possible is part of your eCommerce performance testing, where these should always form part of your performance testing analysis and investigation.

Indexes though, after their initial creation need to be maintained because they become fragmented which happens when data is inserted, updated and deleted in a table.
Database Indexes can be fragmented in two ways, Internal Fragmentation and External Fragmentation, although the terminology may differ depending on the Database technology.

Internal fragmentation is when there is free space on a page, due to deletes, updates and inserts, and there is more free space on a page than required where External fragmentation is when database logical and physical pages are ordered differently.

An Index rebuild simply drops and recreates the index which means that Index rebuild will solve both the Internal and External fragmentation. There are several other ways to rebuild or reorganize Indexes which depends on your database technology and version. The regular checking and maintenance of Indexes will ensure that your Database Queries are as efficient as possible as they are used by your production application.

Finally, you should not forget about Archiving and Weeding data in your Databases, as the data in tables grows so does the time taken to query them and ensuring that your Database tables only contain relevant and current data can make a huge difference to your query execution times. It has the additional benefit of saving storage space which leads to cheaper hosting costs.

Leveraging Automated Testing Tools

We have discussed at the start of this whitepaper how we might approach the **Performance Testing** of our applications to determine the impact of traffic spikes on a theoretical eCommerce platform and we have considered the implications of how a scalable platform can support this.

You need to build performance tests to satisfy your performance testing goals in line with your performance testing strategy.

Only by executing these tests and analysing the results can you determine if your application under high load and traffic spike loads can meet your response time and throughput requirements for performance.
It also ensures that visitors to your eCommerce site continue to have a great experience whatever the load on your system.

The best, and most convenient way, to generate load on your platform is to use one of the many Performance Testing Tools that are available, some Commercial and some Open Source.
We will discuss in more detail the Apache JMeter tool shortly, but regardless of the tool you use the principles of how you approach Performance Testing will be the same.

Introduction to Automated Testing

In this section we are going to look at automation at a high level, from building tests to their execution and benefits they can provide.

Building Your Tests

You need to consider what functionality or journeys through your eCommerce platform you want to automate, depending on the complexity of your site.
It is not a good idea to build performance tests for all functionality, unless of course it has a very simple Interaction Model with only a few pages.

All automated test scripts, whether for the purposes of **Performance Testing** or Functional Testing, require maintenance. If you build a significant amount of automation this can lead to the activity of maintaining scripts becoming so time consuming that they outweigh the benefits they bring when running.
With Performance Testing you do not need vast coverage of your eCommerce application as the testing is mainly about load and concurrency. While each page or piece of functionality on your eCommerce site will potentially load your application and the infrastructure its hosted on in different ways.

A good rule of thumb, assuming you have a functionally rich eCommerce application, is to automate the **High Volume, High Revenue Earning** functionality for your Performance Tests.
Once you have Built, Correlated and Parameterised your tests to make then re-usable and re-executable you can then run them under load to simulate your high traffic spike volumes.

Executing Your Tests

Once you have your performance tests built and stable you can generate the load profiles that simulates the high traffic spikes in a repeatable fashion. If you are executing performance tests and then re-running based after application or infrastructure tuning or optimisation, you want your tests to always generate the same

load in the same way.
If you do not, it is impossible to compare results generated under different load conditions or profiles.

Analysing Results

Once your tests have been executed this is where you can determine the impact of the load and concurrency on the response times of your tests.

This is also where you can see how the CPU and Memory as well as other server resources has been impacted by the high traffic volumes. If your response times from the journey simulated in your tests do not meet the non-functional requirements that you have defined, then at this point you **need to search for the bottlenecks in your application or infrastructure**.

CPU and Memory across all parts of your infrastructure is a good place to start as well as how your Database Queries are performing under load. Other considerations are Network Latency and any single threaded Message Queues or Topics that might be part of your application process. A point to consider here is that some business functionality that relies on several sequential processes may never be able to meet your non-functional requirements due to their complexity.

If you are unable to determine any obvious points of contention on your technology platform that may be the reason for any slow transactions it may be that they are tuned as efficiently as they can be, and your non-functional requirement is unattainable. You can always redefine what your expected response time should be in your non-functional requirements, the critical thing about this testing is that under traffic spike conditions your eCommerce site should not regress in terms of how it performs and remains consistent regardless of load. An eCommerce site that becomes unresponsive or slow will result your end-users going elsewhere for their products.

Having an eCommerce platform that **delivers consistent performance regardless of load** is what you can achieve using the techniques and optimisations we are discussing in this whitepaper.

Re-execution and Comparison

This is where your performance tests really pay dividends as you can, once the tests are built, execute and re-execute tests at any frequency you want. Once you have a performance test built that simulates your traffic spikes then with your automation you can execute after each new release or upgrade and contrast your results against the previous ones.
This way you are **regularly maintaining a baseline** of response times under these high traffic conditions and can easily tell if a potential change or platform enhancement will affect response times.
With automation it is very easy to schedule overnight tests and get results in the morning or to run performance tests over quiet traffic weekends. Consider the fact that it is possible that your test environment and your production environment could share parts of your infrastructure, the network for example.
High volume tests on your network that your production application also uses may see your performance testing impacting your production services. Having automated tests and having the ability to schedule tests when you production system is quiet is very useful.

Types of Testing

We are focussing our attention on building tests to understand the impact of traffic spikes on eCommerce

platforms.
With performance test automation once you have built your tests you can execute them under any load profile conditions you want meaning that the tests script capability you build for traffic spike testing can also be used for other performance tests, examples being:

- Load testing, where you execute your tests at levels of high volume outside of that seen in your traffic spikes.
- Soak testing, where you execute lower volumes but for longer periods of time to understand the impact of long periods of sustained load on your infrastructure.
- Volume testing, where you execute performance tests against an environment that has the same volume of data in the database as you have, or will expect, in production.
- Regression testing is running your performance test suit, or subset of it, on a regular basis to understand the impact of small changes on performance.

JMeter Overview

We have already stated above that there are several performance testing tools available to support your performance testing requirements.
We are going to discuss Apache JMeter in this whitepaper, the reason we are focussing on this tool are listed below and we will expand on these in the next sections:

- Open source
- Supports many protocol
- Large online community
- 3rd Party extension and plugins
- The OctoPerf SaaS solution uses the JMeter engine

Overview

This is directly from the Apache JMeter page:
The Apache JMeter™ application is open source software, a 100% pure Java application designed to load test functional behavior and measure performance.
It was originally designed for testing Web Applications but has since expanded to other test functions.
JMeter is not a browser, it works at protocol level. As far as web-services and remote services are concerned, JMeter looks like a browser (or rather, multiple browsers); however JMeter does not perform all the actions supported by browsers.
In particular, JMeter does not execute the JavaScript found in HTML pages. Nor does it render the HTML pages as a browser does (it's possible to view the response as HTML etc., but the timings are not included in any samples, and only one sample in one thread is ever displayed at a time).

This whitepaper is not the right place to discuss in detail how Apache JMeter works and how it can be leveraged to build and support your load testing activities.
There are many posts on the OctoPerf blog pages that provide clear guidance on how you can use Apache JMeter to build tests and how most of the functionality it offers can be used in a practical way with numerous real-world examples.
These posts can be found on the OctoPerf Blog Post pages.

Protocol Support

Apache JMeter has rich Protocol, Server and Application support, again taken from the Apache JMeter homepage these are those covered natively:

- Web - HTTP, HTTPS (Java, NodeJS, PHP, ASP.NET, …)
- SOAP / REST Webservices
- FTP
- Database via JDBC
- LDAP
- Message-oriented middleware (MOM) via JMS
- Mail - SMTP(S), POP3(S) and IMAP(S)
- Native commands or shell scripts
- TCP
- Java Objects
- JMS
- GraphQL

Online Community

There is an extensive **Online Community** supporting Apache JMeter and the tool continues to be regularly upgraded with at least one release a year dating back to 2007.
There is an active Developer GIT Repository and Mailing List that is continually evolving.
This ensures that Apache JMeter is just as relevant as it was when it was first released and whilst not the only Open Source Performance Testing Tool available, it certainly has one of the best support communities and continually looks to evolve to support new technologies.

3rd Party Extensions and Plugins

Apache JMeter contains a plug-ins manager which allows the extension of Apache JMeter functionality by 3rd parties. The 3rd party support available to JMeter consists of:

- Samplers
- Listeners
- Graphs and Reporting
- Throughput Controllers
- Protocol Support and Enhancement
- Security and Authentication
- Database Interaction

Benefits of OctoPerf

When it comes to Performance Testing there are benefits of using a specialist company, one that provides a SaaS approach to delivering your performance testing requirements.
OctoPerf can provide expertise and guidance on Performance Testing which is a discipline that can be complicated and expensive to run and understand.

OctoPerf delivers the ability to quickly and easily take Apache JMeter tests and execute these performance

tests under a variety of load profiles in a repeatable and consistent way which is critical to large scale performance testing.

The ability to ensure that you have enough server resources to support very high levels of load and concurrency is a very important factor in load and performance testing and one that employing OctoPerf will ensure is fully met.

One of the most important factors in performance testing is ensuring that your Load Injectors are not the bottleneck, you would be surprised to learn of the number of times that the performance issues were due to the way the load was being delivered to the application under test rather than the application itself.
The inability for the servers generating the load being the point of contention because they did not have enough resources in terms of CPU or Memory to support the concurrent request volumes is a common mistake.

What follows is the perception that the software does not perform or scale where it's the load injectors being the point of contention.

SaaS companies remove this concern by being able to provide load injectors with resources that can scale to support any load you want to generate meaning that you will have no concerns about not being able to load you application correctly and accurately.
And you will not need to worry that your load injectors are potentially the reason for slow application response times.

OctoPerf stands alone in offering a simple interface to mask some of the complexity of Apache JMeter, they offer the ability to scale effortlessly coupled with easy of analysis and integration with your organisations on-premise tools. This unique blend of simplicity, agility and power to simulate high traffic volumes from anywhere makes OctoPerf a leading proposition for Performance Testing in the SaaS & On-Premise space.
Once your tests are uploaded, OctoPerf provides the framework to build Performance Test Scenarios to match any of your load profiles regardless of whether is for Load Testing, Scalability Testing or to simulate spikes in traffic.

The Cloud Platform that OctoPerf uses allows you to effortlessly scale your concurrent users easily to above one million. Your load tests can originate from many geographic locations to simulate where your user base is located while also utilising your local hardware to support service you want to performance test that are sat behind Company Firewalls.

Integration with your Continuous Integration and Continuous Delivery tooling allows you to leverage your internal DevOps development approach while still ensuring your Performance Testing runs seamlessly.
Analysing you load tests with OctoPerf is both simple and powerful, the ability to visualise your test results both as your tests are running and after testing completes. It brings all the information you need to determine if your application performs in line with your non-functional requirements under high traffic volumes.

Providing historical comparison of results provides you with the capacity to contrast response times across previous tests and look for signs of regression or improvement.
Results can easily be shared with your project and development teams using native integration with many industry-standard collaboration tools, such as:

- Microsoft Teams

- Slack
- Jira

The OctoPerf platform also seamless integration with your internal Application Performance Management platform to aggregate performance and server metrics side by side, examples of supported platforms are:

- Dynatrace
- New Relic
- AppDynamics
- GitLab
- Azure Devops
- Instana
- GitHub
- Jenkins
- Jira

The ability to push all the performance results to one of your internal data sources is also supported natively and allows you monitor performance and provide reports using your internal technology stack should you wish.

Case Study

There are some other real-world examples of how these companies have successfully used the OctoPerf SaaS platform to fulfil their performance testing needs:

- Adeo is the European leader in the home improvement and DIY market.
- Decathlon is a network of innovative retail chains and brands providing enjoyment for all sports people.
- Privalia holds short brand oriented flash sales on several markets like Brazil, Mexico, Spain and Italy.

Let's look at a brief theoretical case study detailing how an eCommerce company successfully used OctoPerf to prepare for Black Friday. We are going to use a company called PerformaShoe for our case study.

Introduction

PerformaShoe is a market leading eCommerce site specialising in sports shoes and designer trainer brands. During Black Friday last year their eCommerce platform was unable to keep up with the number of visitors and the volume of purchases which meant that their webservers became unresponsive and crashed.
This led to a loss of revenue as PerfomaShoe had not only invested heavily in a marketing campaign ahead of Black Friday but had purchased additional stock in anticipation of high demand which did not sell.

The damage was more than just financial as PerformaShoe suffer reputational damage with customer retention at an all-time low.

After steadily growing the brand name and gaining the trust of consumers again PerformaShoe turned to OctoPerf to help them ensure that this Black Friday would not be a similar to the last one.

The Challenge

PerfomaShoe had an experienced in-house technology department with their eCommerce site hosted on a cloud platform.

They had built a set of performance tests using JMeter that were run from a laptop that was unable to support the load volumes they wanted to achieve, and decided that they would run low volume tests and extrapolate the results.

What PerformaShoe did not appreciate was that their eCommerce platform did not scale linearly and therefore as volumes increased so did demand on their platform resources in an exponential fashion.

To further complicate the issue, they had underestimated the volumes they would experience and did not have a technology platform that could be easily scaled.

The JMeter tests also did not cover the high-volume journeys through the application and were focused solely on product purchasing where the highest traffic volumes were all around product searches and adding items to a favourites list.

The Solution

Ahead of the next Black Friday the QE manager at PerformaShoe engaged with OctoPerf as they wanted to run a series of performance tests to accurately represent the transaction volumes and levels of concurrency they saw last year and to then push these volumes up incrementally.

PerformaShoe did not want to invest in an in-house performance testing solution to support this testing a,s these Black Friday volumes were easily the highest they would experience this year. An ability to support these would mean they could support any volumes their platforms would experience.

The PerformaShoe performance testers were able to upload their existing JMeter tests to the OctoPerf platform and develop further tests around searching and adding to favourites directly using the OctoPerf scripting engine.

Several load profiles were then created using the OctoPerf platform, each one of them made of a variety of loads and concurrent users replicating the load varieties that PerformaShoe saw during the year, from their quietest time in terms of sales volumes up to the peak of last year's Black Friday.

PerformaShoe uses a Content Delivery Network and therefore wanted the load on the servers to originate from a variety of locations to ensure this part of their architecture was caching content correctly. This was easy to configure using the geo-location functionality that OctoPerf has at its disposal.

Each load profile was run using OctoPerf against the PerformaShoe platform, and the volumes increase systematically.

At each stage of the incremental process the application resources were monitored, and responses times visualised in both the OctoPerf analytics tool and PerformaShoe's in-house Data Warehouse for further analysis.

As the PerformaShoe platform was scaled and resources added, the tests were re-executed by triggering from PerformaShoe's CI/CD Jenkins Pipelines which integrate seamlessly in OctoPerf, meaning that the tests could be scheduled to run outside of the busy times of day to avoid impacting other users of the test platform.

The Results

PerformaShoe now fully understood how they needed to scale their infrastructure to support volumes depending on the variations in load and concurrency.

And importantly the understood what they needed to do to react to high traffic spikes.

During the next Black Friday event, again on the back of a marketing campaign, PerformaShoe scaled their infrastructure based on their performance testing executed using the OctoPerf platform and they suffered none of the issues they saw the previous year.

At one point on Black Friday the demand started to increase even above the tested volumes of load and concurrency but as PerformaShoe understood how their platform scaled, they were able to easily increase their eCommerce platform resources to cope with the additional peak in traffic volumes.

Action

PerformaShoe used the load volumes it saw from this Black Friday event and plans to use them to generate higher loads using the OctoPerf SaaS solution.

Real-Time Monitoring and Incident Management

Application Management Tools

There are many **Management Tools that** are available to organisations to monitor their production systems. We are not going to list them here although a few have been mentioned in the Benefits of OctoPerf section above.

While the integration between the tools and your application, services and infrastructure may be different along with the way they display and articulate results, the management tool you choose does not change the way you need to approach your monitoring techniques.
We are going to look at ways of **ensuring you can use this real time monitoring** to ensure your application can support the high traffic spikes your eCommerce platform might encounter.

Compare against Tests

When implementing an Application Performance Management Tool, you will find huge benefits using your Performance Test environment and your Performance Testing scenarios to test your Threshold Monitoring and the building of Dashboards to monitor for system resource utilisation and response times etc.
We will discuss what we mean by Threshold Monitoring and how you can use Dashboards to support your monitoring of your environment in the next sections.

Firstly, let's discuss the benefits of using your Performance Testing to establish your APM, and the benefits that this can bring to your production application monitoring as you start to experience traffic spikes.
We have spoken about, at the start of this whitepaper, the approach you should look to take with your Performance Testing where you systematically increase the load based on analysis of previous results. How you look to understand how your system resources and your responses times are affected as you increase your load profile towards your expected traffic spikes.

Earlier we discussed how increasing load and concurrency can affect response times on your eCommerce platform and the action you need to take in terms of increasing your platform resources to try and counter this.

We did not discuss in detail how you would gather the data you need to understand where the application bottlenecks exist.

Now there are ways you can do this using Windows or Unix tools to monitor infrastructure resources and response times, which are available from Apache JMeter using its native output in the form of .jtl files.
A better option would be to use your APM to do this work.

This has dual benefits, the first being that you can understand clearly how your application performs under the various degrees of load and concurrency you place on it.

And secondly, you can use the APM functionality to trace your transactions through your application and understand which parts of your technology stack are causing the bottleneck.

APM's are functionally very rich and can pinpoint exactly where your response times is spending the most time and where the points of contention are. We have touched on the fact that traffic spikes are sometimes expected, Black Friday for example, however some are not.

As you have the information about how your application behaves from your Performance Testing ,you can take this information and look for similarities in your production monitoring.

If you start to **spot a trend in application behaviour** as load increases, you will have the knowledge to understand what needs to be done to support the increase in load, whether this is manually scaling your infrastructure, ensuring that your platform is autoscaling based on the load profile or whether you have an infrastructure problem and need to investigate the root cause.

Using your APM as part of your performance testing allows you to test your production process for dealing with increased traffic spikes.

Thresholds

When you configure your APM you can set thresholds on many aspects of your application and the platform that supports it.

Examples can be the amount of CPU, the Memory being consumed, the number of items in Message Queues or Database Query execution times.

Once these thresholds have been exceeded, then the APM can alert you through pre-configured alerting methods or if your APM is part of your cloud service provider, it can automatically scale your platform to

ensure the thresholds are no longer being exceeded.
These thresholds, especially for traffic spikes that can happen quicky, can be defined and tested as part for your Performance Testing approach so you can determine what the course of action should be in the event this happens in production.

One of the critical takeaways from this is that APM's provide a significant amount of data and metrics and can be the **difference between your platform surviving traffic spikes**.
However, you need to test these processes and understand the action to take when you see them, and using high volume performance testing to replicate real-world high traffic scenarios is an important part of this process.

Build Dashboards

Comparison of response times and server resource utilisation at different times of the year, month, week or day helps you understand how your customers interact with your eCommerce platform.
This knowledge can help you predict when your traffic spikes might happen down to the time of year or even the hour of the day.

This information is not only useful for real time monitoring and predicting future event volumes but can be useful in designing your Performance Tests.

We spoke at the start of this whitepaper about using **historical data to determine your Performance Testing volumes** starting point and understanding any historical traffic spike volumes.

APM reporting can provide all this information which you can use to define your Performance Testing volumes for future tests. Performance Testing is not a one-off exercise and needs to be run regularly, the volumes you run your performance testing at can be defined using the information your get from your APM and present in your Dashboards and Reports.

Collaboration Tools

Just as OctoPerf integrates with collaboration tools such as Microsoft Team, Slack or Jira so do most APM's. Collaboration is key is most technology organisations and providing visibility to all around application performance and alerting promotes this culture. Performance is not just the job of the performance testing team. It is a **mark of quality that all members of the technology department should be responsible for**, and your APM is how you gain visibility of your production eCommerce servers and how it is performing.

The sharing of this data promotes ideas and thoughts around high volume management and ensuring your platform provides your end-users with the best possible experience.

Incident Response Strategies

You can plan, test and prepare for a Production Incident related to performance of your eCommerce platform but sometimes even with all your hard work and analysis of Performance Test results you will have a production issue where the responsiveness of your platform is affected.
The critical thing is how you respond to the incident and how you learn from it, we will discuss this here.

Immediate Actions

Scale your infrastructure, if you have demonstrated through Performance Testing that you have a scalable platform then in the event of high volumes of load on your eCommerce platform that your system is unable to support, then the first action should be to scale your infrastructure beyond where it currently is.
You should do this across all Application Servers and Database Servers, to try and rectify your end-users experience and to ensure you can continue to sell your products.

You should look to over-scale in the first instance, what I mean by this is to not be concerned by increasing your system resources incrementally until performance returns to acceptable limits.

You should scale big first, doubling or even trebling your server resources and if performance returns to an acceptable level, incrementally scale back down until you find the point that application performance and server resources are optimised.

We have spoken about not all organisations hosting their platform on cloud-based infrastructure which means that scaling quickly may not be an option, or you maybe you found that scaling does not solve your production performance issues.

If this is the case then in the first instance it is good practice to put up a static landing page stating that you are experiencing issues, this way your end-users are not being faced with time-outs or server error messages which gives you some time to investigate the incident further.

Communication Policies

If your eCommerce platform is being affected by a performance issues or application failure you need to effectively communicate to both internal and external customers.

Social media is powerful and bad publicity for your eCommerce platform can quickly go viral. If you are facing issues with the performance of your platform and you are unable to rectify the issue quickly then proactive communication from you organisation allows you to control the narrative and avoid, to the best of your ability, negative publicity.

Internally, the quicker you communicate to internal stakeholders and key management then the quicker you can start to act on the problem.

Your Application Monitoring Tool will be pivotal in understanding why your eCommerce site is not performing and should help you to pinpoint the root cause of the issue.

Post Event Analysis

Assuming you regularly execute or executed during the initial release of your eCommerce platform, a set of Performance Tests, the post event analysis needs to centre on why the issue was not replicated in test.
It could be that your production volumes were well in advance of anything that you predicted, and you did not scale your tests that high.

If this is the case, then the critical thing is to ensure you can **replicate the issue in test** using the volumes you saw in production during the incident.

If you can replicate the production incident then you can understand why and where the issue occurred and put in place a permanent fix, whether this is an application code fix, further front-end optimisations or an understanding of how you need to size your infrastructure to support your traffic spikes.

If you are unable to replicate the incident in test then you need to understand why.
If your eCommerce platform scaling strategy to be able to support traffic spikes is based on results from an environment that does not behave as production, then you need to determine why this is.

It may be that your testing scenarios are not accurate and the volume of load you are generating is inaccurate, it may be that your test environment uses a different network or stubs 3rd parties.

Another explanation could be that your Test Environments do not match production in terms of server resources, or that your test Database does not contain the same volume, and/or diversity of data as production does. The possibilities are numerous, and the focus of your post event analysis should be to replicate what happened in production and then revise your eCommerce server scaling policy accordingly.

Security During Peak Traffic

Protecting your eCommerce platform against **Security Threats** is paramount as most organisations will face some sort of Cyberattack at some point. The impact of high volumes or load on your security are only really an issue if the source of the load is a Denial of Service **(DOS)** or **Dedicated Denial of Service (DDOS)** attack.
The load generated by a DOS or DDOS attack might affect your eCommerce platform in several ways.
If you are in a period of low volume customer activity, you might find that your infrastructure can scale to support this additional load, especially if you have infrastructure that scales dynamically.

However, you are in a period of high traffic volume then the additional load generated by the attack is going to cause your platform to experience the type of slowdown or failure that you have been trying to avoid with your Performance Testing strategy.

Web Application Firewalls

A Web Application Firewall **(WAF)** will help you avoid most cyberattacks if you regularly maintain the firewall and ensure it is patched regularly. A WAF helps protect Web Applications by filtering and monitoring HTTP traffic between a Web Application and the Internet. It typically protects web applications from attacks such as:

- Cross-Site Forgery
- Cross-Site-Scripting
- File Inclusion
- SQL injection

By deploying a WAF in front of a **Web Application**, a shield is placed between the web application and the Internet.
While a Proxy Server protects a client machine's identity by using an intermediary.
A WAF is a type of Reverse Proxy, protecting the server from exposure by having clients pass through the WAF before reaching the server.
A WAF operates through a set of rules often called policies and these policies aim to protect against vulnerabilities in the application by filtering out malicious traffic.

Conclusion

If you want to ensure that your eCommerce platform can support traffic spikes you need to run **Performance Tests at volumes indicative of these spikes**.

To do this, you need to invest time in defining what these volumes of load and concurrency are likely to be and analysis of previous load volumes will help with this.

Once you have a clear indication of volumes, you can run a series of **Scalability Tests to incrementally load your platform** with load ranging from volumes that are indicative of average load up to the expected traffic spike peaks.

As you increase your load volumes, you can start to understand how your servers react to the increases and how you need to increase your server resources to support these additional levels of load.
This is also an opportunity to address any application defects that might be associated with high traffic volumes.

Using you **Application Monitoring Tools to** support your testing and help define Service Thresholds will not only help with your testing but will ensure that your **Service Thresholds** for alerting against your production systems are accurate.

Your performance testing will allow you to define a strategy for managing high traffic spikes whilst still maintaining a platform that is responsive and performs consistently.

Performance testing requires you to have the server power to generate load, it is important to not let your load injectors become the bottleneck. Using a **SaaS/On-Premise company such as OctoPer**f not only ensures you can generate load at any volume you need with any level of concurrent connection, it will also provide you with the critical analysis you need to understand how response times, server resources and throughput is affected as load increases.

It will bring comprehensive analytical data to support the information you gather from your APM.
Comprehensive, targeted performance testing against your eCommerce platform can make the difference between a successful high traffic period and one that does your organisation more damage than good.