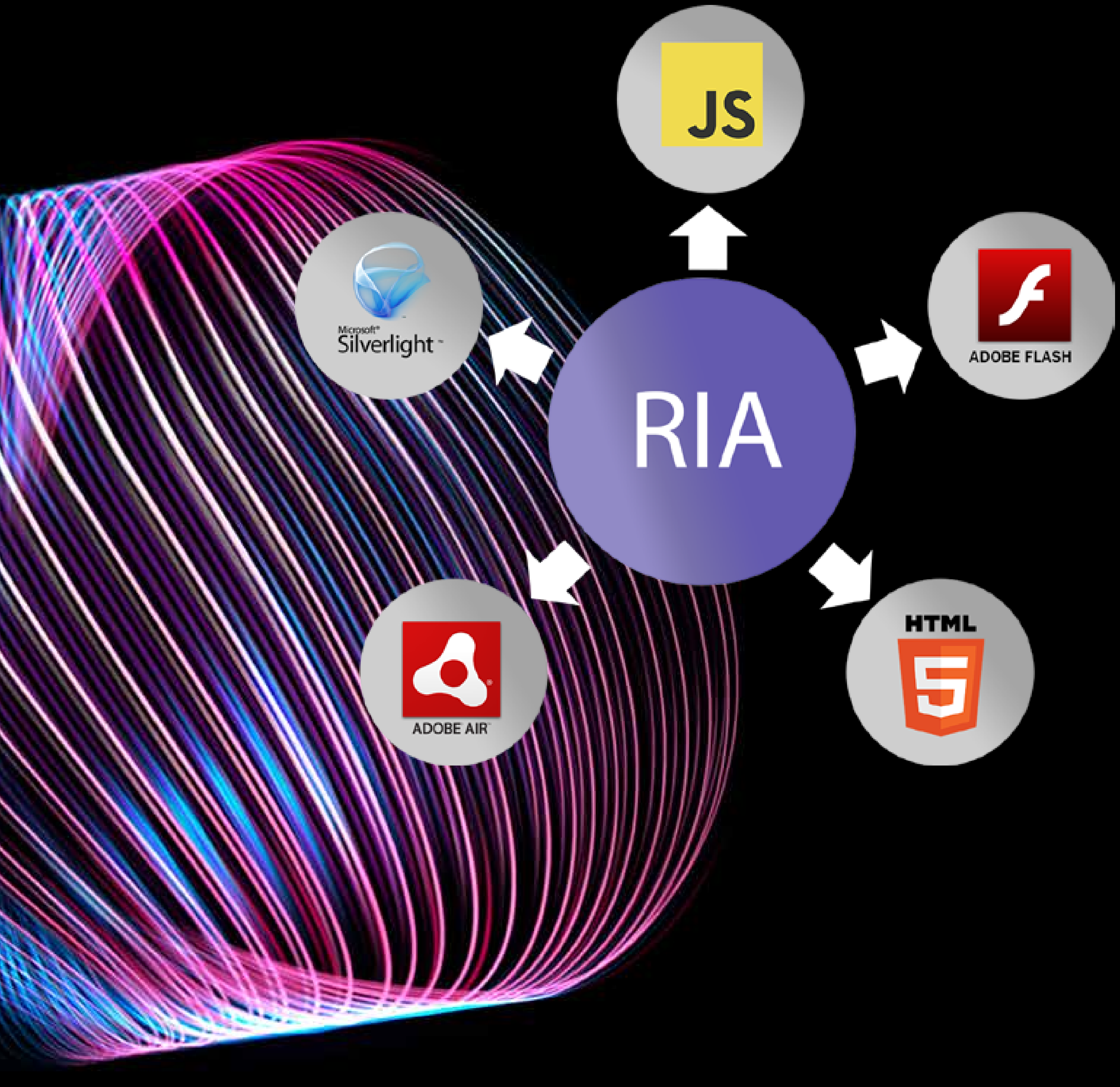
Ten10

“Performance testing Rich Internet Applications (RIAs) and other complex clients”



+44 (0) 203 697 1444

www.ten10.com

As more and more organisations are using increasingly complex applications that make us of Rich Internet Applications (RIA) and other technologies such as HTML5 and Javascript, they are finding that traditional performance testing approaches are no longer sufficient.

RIA technologies often include functionality that it is not possible to performance test using traditional approaches, but which can have an impact on the overall performance experienced by the user, potentially resulting in greater bounce rates and lower conversion. Traditional approaches only look at the server performance, and not the client side performance where these technologies are implemented. Another common challenge is the lack of comprehensive tooling available to test the performance of complex graphical user interface applications (GUI).

To solve these challenges a new approach is needed; one which realistically measures the performance all the way through to the client, and also provides scalability and flexibility that traditional hardware based performance solutions do not provide.

Challenges

- Increasingly complex applications including Rich Internet Applications (RIA) include functionality not performance tested by traditional approaches, but which can have an impact on overall performance experienced by the user
- A lack of comprehensive tooling to support complex GUI based performance testing
- Scaling tests when using traditional approaches, that uses physical hardware in a datacentre, can be costly
- Being able to understand and incorporate client side efficiencies into the overall user experience

The Solution

- A bespoke solution that meets the performance testing challenges Rich Internet Applications present - True Performance Testing
- Tests are performed through the actual user interface using real user interactions, measuring the true performance of the system, including the latency from multiple geographical locations
- Infinitely scalable and cost efficient compared to traditional approaches
- Includes GUI based performance measures and traditional client-server measures to provide a clear picture of the system performance and to help diagnose issues

Testing application performance typically involves replicating the traffic between client and server and checking the time for messages to be returned from the server following a message being sent from the client. Replicating the user activity in this way proves that the server is performing to your needs. While this is very important, it does not provide a true picture of the performance observed by the end user, especially in today’s Rich Internet Applications that can include extended functionality through plugins and increasingly through technology like HTML5 and javascript. The client side technologies, when performing badly, can make the difference between a system that performs to a user’s expectations or not, and may mean the difference between a user converting into a transaction of some sort like a trade in finance, or a conversion in retail.

Increasingly feature-rich GUIs resulting in performance issues and degraded overall user experience

Although measuring the performance of a server in a performance test is very important in understanding the performance of a system, the increasingly complex applications of today present unique challenges.

It is possible to fool a server into accepting scripts replicating the client server traffic from a user on an internal injector farm. As more application functionality is executed at the client, the use of traditional performance testing methodologies to simulate message-level interaction with the server for Rich Internet Applications (RIAs) is becoming more complex and potentially less representative of the behaviour of the application itself. Many Modern web applications now depend on the use of application frameworks such as HTML5, Flash and JavaFX or ever increasingly complex javascript, and their compatibility with browsers such as IE, Firefox or Chrome. The overall user experience and performance could be degraded without traditional performance tests even knowing about it.

Measuring the performance of a system just through the user interface is not enough. You can measure the performance against the client but you still need to measure the server performance as you would with a traditional performance test. Not many tools or libraries allow you to do this out of the box.

Tools, approaches and solutions for GUI based performance testing

The traditional approach to performance testing presents challenges, and fundamentally risk, when applied to systems with rich clients. Many solutions and tools available are geared towards this approach and have typically focussed on replicating the messaging layers that enable applications to communicate. The technology exists to drive more complicated rich clients but no one tool or library exists that can successfully bring the technology required to drive a rich client together with the ability to understand the traffic between the client and server, while also taking into account a sound performance engineering approach.

For this reason a new approach and custom built tools had to be adopted that also enables large scale tests to be performed that don’t rely on a performance testing lab built around physical infrastructure.

Scalability without using a traditional injector farm

Scaling the test without using a traditional injector farm (as not possible in the traditional way when using GUI automation)

Typically these labs would include a number of injectors that provide the traffic that replicates the real users using the system. These injectors would be physical servers located in a data centre and would be one of the key prohibitors in developing a live like performance solution, as the number of Virtual Users (VUs) that a single server can support is a limiting factor on each server. For complex Rich Internet Applications (RIAs) each Virtual User (VU) could consume a large amount of memory, limiting even further the number of VUs each injector can support. This presents a further challenge around costs of running enough injectors to support a live like performance test.

True performance engineering

To remain competitive companies must engage optimal true performance engineering methodologies to ensure they are fully aware of and are able to manage any latency and potential bottleneck issues with their system. Competitive advantage may also be gained as a consequence of being able to demonstrate to clients that optimal levels of performance are consistently achieved.

The True Performance Engineering approach has clear advantages. It enables handling of dynamic data, multiple applications, complex hardware (injector farm capability), software dependencies and huge volumes. It also allows better measurement of budget, risk, resource scalability and availability.

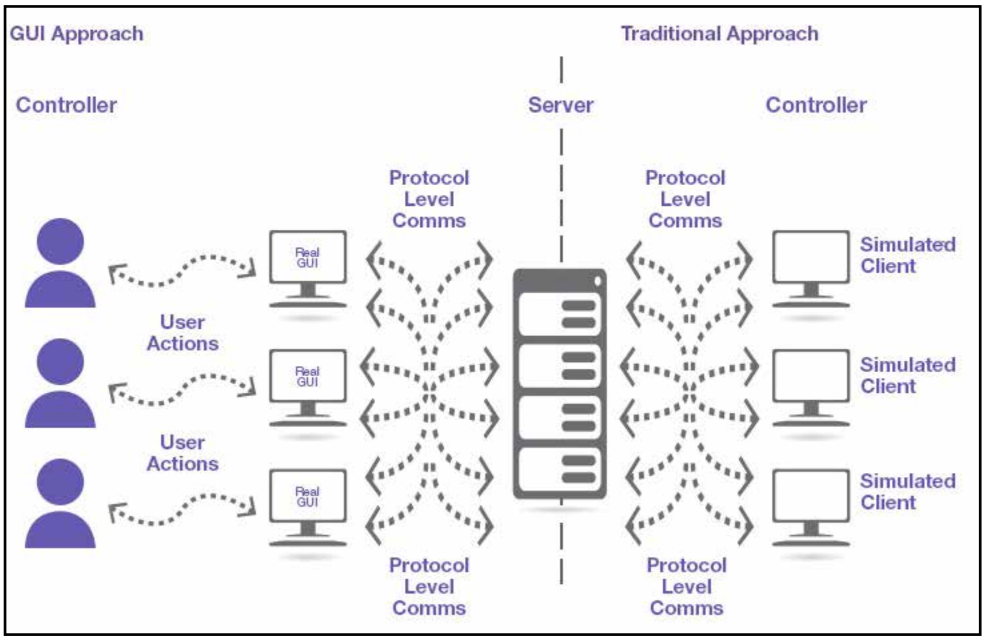
True performance engineering traditionally demanded massive infrastructure and resource utilisation at substantial cost, however, the cloud now makes it possible to access immense, on demand, cost-effective, scalable cloud based injector farms, leased solely for the duration of the test.

Performance tests driven by GUI-based automation

The need to drive performance tests from the GUI not only necessitated a change in approach, but also a change in tooling. Tools exist to drive the GUI for the purposes of test automation, but these are not designed to be used in large scale performance tests and lack the type of interaction and instrumentation needed to properly recognise and manage potential issues being encountered.

Further to this many applications are developed in technologies that are inherently not automatable. For example Flex applications don’t include the normal hooks that a HTML website contains that allows you to control specific elements on the page. Although libraries can be built to enable automation in Flex it is often seen as an obtrusive solution and can lead to challenges with configuration and version control when deploying to Live.

Ten10 re-engineered open source Optical Character Recognition (OCR) technology to build ground breaking tools that drive the GUI, mimicking user actions in a non intrusive and efficient way, capturing statistics and issues as the test takes place. The tools include the capability to control any application on any type of operating system (injector) and tools to manage the injectors and aggregate the data collected in real time. They have a very low performance overhead on the machine it is running on and is completely decoupled from any application it controls. As the actual interface is being controlled, the response times being captured are true to what the user will see, rather than the transaction time clock being stopped when a message is received on the client from the server.



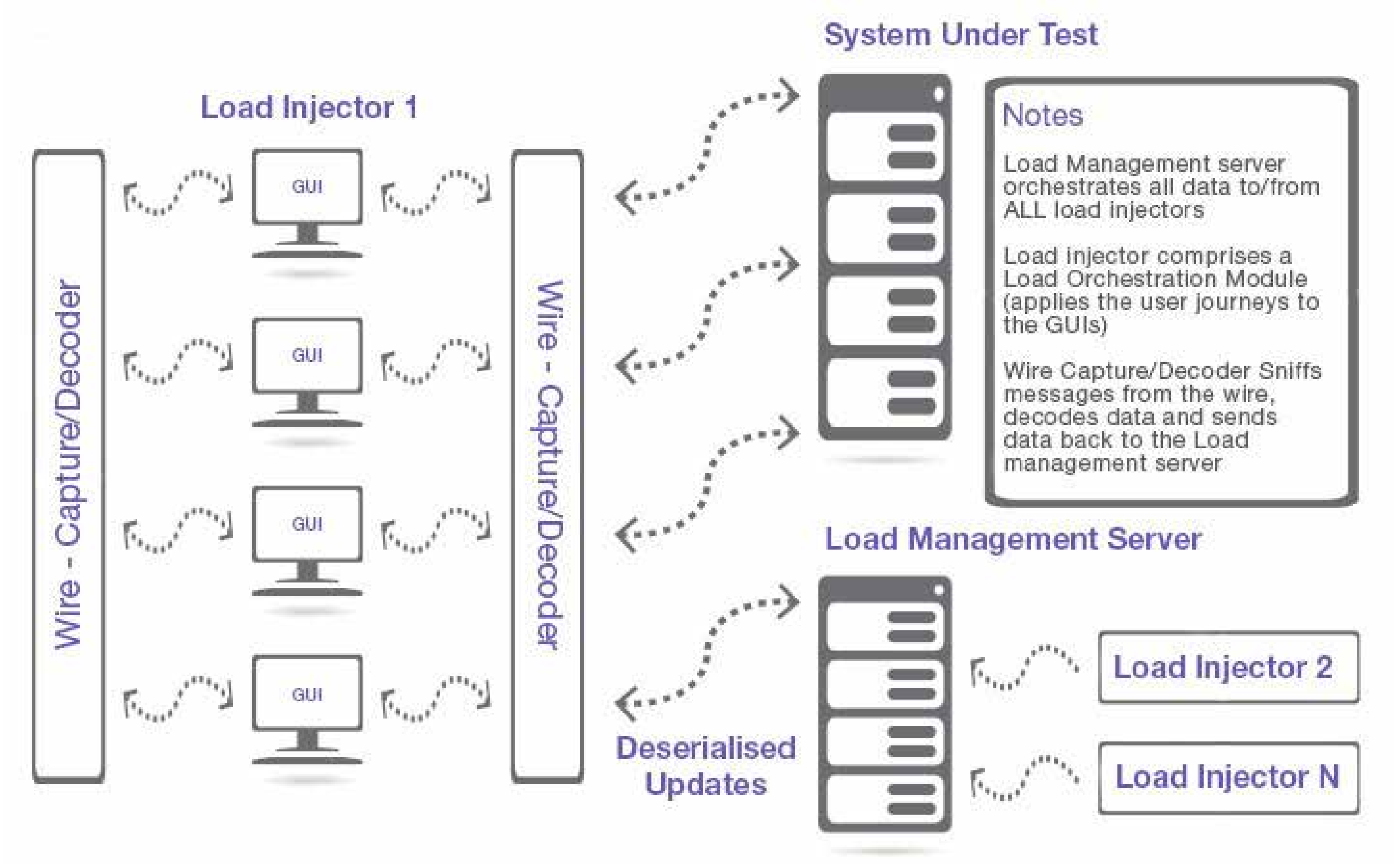
Rapid and low-cost availability in the cloud

The traditional performance testing approach, that involves simulating client messages to the server, utilises injectors to provide this load. These injectors can each provide the load of a large number of users known as Virtual Users (VU). Each Virtual User will have a typical memory footprint meaning the number of VUs an injector can handle will be limited by the capacity of the injector. For a GUI based performance test this approach doesn’t fit - you can’t reliably load and control a high number of applications on a single machine without causing contention between the instances of the application. Also, typically only a single instance of an application can be loaded at a time, meaning the relationship between Virtual User (VU) and injector becomes one to one.

Using a traditional onsite approach where hundreds or even thousands of servers would need to be procured and set up is clearly not a viable solution. To provide the scalability and flexibility needed, the cloud was used. This offers “pay for what you use” operations, enabling ‘on demand’ expansion of load infrastructures without the need for scheduled load management, often months in advance. In addition to the technical advantages, this model offers end-users true business benefits in terms of flexible project prioritisation and reductions in hardware budgets, which directly affect bottom line return on investment.

Load management, messaging capture and decoding

Although providing a clear measure of the end to end performance as the user would observe has clear advantages, understanding the performance of the server by measuring the server response times is still important. It can also make diagnosing issues much simpler than interpreting unexpected events on a complex GUI. For this reason the messages between the client and server are monitored, and if needed decoded, so that assertions can take place and server transactions times can be logged. This provides visibility of underlying messaging between client and server so you know how the whole systems performs and where the issues are.


System Under Test
Load Injector 1
GUI
GUI
GUI
GUI
Wire - Capture/Decoder
Wire - Capture/Decoder
Notes
Load Management server orchestrates all data to/from ALL load injectors
Load injector comprises a Load Orchestration Module (applies the user journeys to the GUIs)
Wire Capture/Decoder Sniffs messages from the wire, decodes data and sends data back to the Load management server
Load Management Server
Load Injector 2
Load Injector N
Deserialised Updates

Benefits

Handling complex, dynamic systems

Being able to performance test today’s complex applications just wouldn’t be possible with a traditional approach. A combination of expertise, experience and custom tooling now makes this possible. As applications and platforms become more complex and rich, there is an increasing need to be able to drive applications closer to the user source. This is becoming more important as the digital world expands beyond just mobile.

Measuring the true performance of the end-user experience

Being able to performance test a system through the GUI provides a clearer picture of the true end to end performance of a system. For rich internet applications this becomes vital as the complex features provided through rich clients are increasingly skewing the overall performance of the system. Engineering good performance into the server side of a system is no longer enough to ensure your user experience and performance is good enough to ensure you convert traffic into revenue and don’t lose customers through high bounce rates.

Cloud-based cost efficiencies

Paying as you go for cloud based infrastructure that can be quickly and easily scaled is one of the most cost effective solutions now being employed for performance testing. This solution also fits perfectly when measuring the true performance of a system as driving a graphical user interface can be even more restrictive when using a traditional hardware or virtual machine based approach to purchasing performance testing infrastructure.

Conclusion

The traditional server based approach to performance testing no longer covers all bases. To truly understand the system performance that a user will observe increasingly requires you to performance test the client also. With the right tooling, performance engineering approach and making the use of the cloud for flexible costs you can understand the performance of your systems and ensure your applications provide the business benefits they were set out to achieve.