INFUSE

PERFORMANCE ENGINEERING 101

Performance Engineering

An abstract on a modern approach to ensuring software applications scale and remain performant under load.

Chapter 1 :

Introduction

Welcome!,

This short article looks at Performance Engineering as it relates to software design and seeks to provide a suggested best practice approach to achieving highly scalable and performant applications embracing new application architecture choices such as Microservices and Containers.

Performance engineering is a natural evolution of the simple static performance testing model historically part of QA, to encompass the entire Software Development Lifecycle (SDLC). The emphasis has shifted (left?) to focus on ensuring release quality and accelerating the speed of delivery as enterprises seek to deploy software changes over ever decreasing time-scales, which requires a strategic rather than purely tactical approach to application performance management.

In the following chapters we will discuss:

- Changing application architecture choices
- Performance Engineering rather than just Performance Testing
- Future trends and how they may further disrupt how we approach Performance Engineering

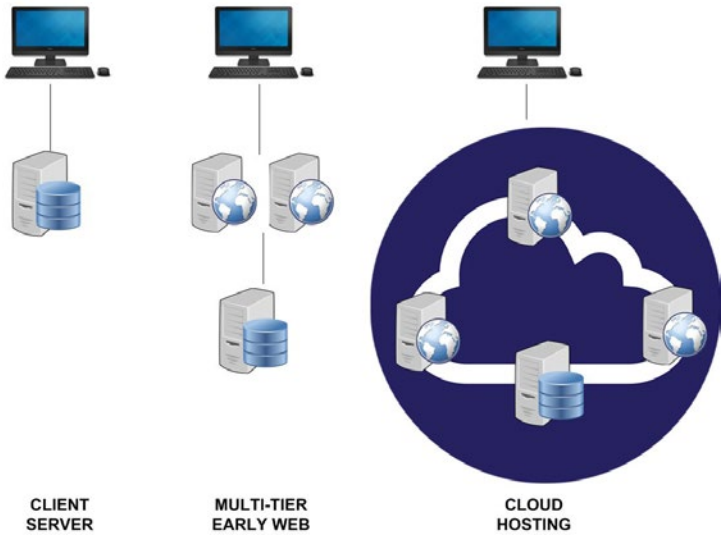
Chapter 2:

Changing Application Architecture

Application design has evolved significantly over the last 10 years finally dispensing with the legacy client-server model in favour of highly distributed architectures based around Cloud and the Web.

The increasing dominance of Cloud has made (Web) Applications easier than ever to deploy and the number of cloud hosting providers has increased to offer public, private and hybrid offerings to suit most business models.

In terms of performance engineering these new architectures present new challenges. Highly distributed design means there are lot of interacting components and services often provided by 3rd-parties. Any of these could impact application performance and scalability.



New Design Choices

The desire to improve the speed and simplicity of deployment has led to the adoption of new design choices like microservices which breaks application functionality down into small highly specialised components and containerization which packages application components into small self-contained deployable modules that can be easily managed and scaled. Both these technologies allow enterprises to break up and re-engineer often monolithic applications in efficient and agile ways so that changes can be rapidly designed, tested and deployed in days rather than weeks or months.

This is especially important in eCommerce where business success or even survival now depends on the ability to rapidly react to changes in customer buying patterns and competitor offerings.

Node.js from Google is another design choice gaining in popularity taking JavaScript to another level as part of enterprise application deployment. Node works well with containers complementing the self-contained and highly scalable modular deployment model.

Everything's Agile

The evolution from Waterfall to Agile methodologies has changed the way we design and build software applications. It is arguably easier to implement performance engineering in an agile shop particularly with the move to componentizing application functionality.
With the increasing popularity of microservice based architecture which has a relatively simple testing model (consumer and endpoint) this makes it straightforward to implement for example; performance testing in Dev.

Risks to Accelerating the Speed of Delivery

As discussed, embracing new design choices make it easier and quicker to code and deploy software applications but this brings it own risks. The more frequently you make changes to the code-base the greater the chance of accidentally introducing performance defects. This makes it increasingly important your performance engineering approach is appropriate and effective.

Performance regression can be surprisingly subtle where individual releases appear fine but the cumulative effects of many releases slowly degrades the end user experience. You have to identify and correct performance defects early and regularly performance benchmark your production deployment to ensure that this doesn't happen.

Performance Pinch-Points

Modular and highly distributed application design also leads to increased potential for performance bottlenecks. Every API call whether to an external or internal service has the potential to impact application performance. Affiliate content embedded in application web pages is a notorious source of end-user slow-down invisible to data-centre based monitoring solutions.

Often the responsibility for maintaining service availability and responsiveness is down to 3rd parties so it's important that applications are engineered to deal with service outage or slow-down. This is particularly true for mobile application clients where cellular and network outage is a frequent occurrence.

Packaged Applications

Something slightly off topic although still relevant to performance engineering is the performance of packaged applications, typically CRM or ERP, and their stated and actual ability to support load.

The underlying architecture of many of these software packages remains legacy so they do not necessarily scale well in response to load. An understanding of modern performance engineering concepts can provide potential clients with much greater insight into the suitability of a given vendors solution for their needs. It would not be unreasonable for example, to expect a vendor to be able to demonstrate support for a given level of load by providing a recent performance benchmark.

Monitoring Visibility

The scope of monitoring visibility has also evolved and now needs to cover last-mile/ first-mile with particular emphasis on understanding the quality of the end-user-experience. No longer are data-centre only monitoring solutions sufficient to provide the performance visibility required.

New technology choices like Containers can present monitoring challenges as the internals may not be visible to your current monitoring technology. With highly distributed architectures it is important to understand how all your application components interact so consider deploying one of the current generation of Application Performance Monitoring solutions. Most have specific support for new technologies like Containers to give you the visibility you need

Chapter 3:

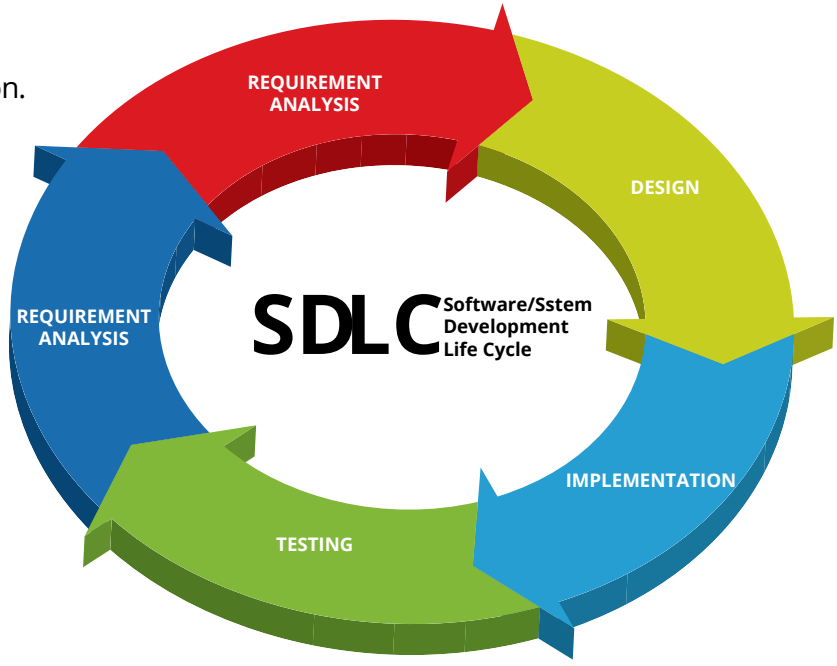
Performance Engineering rather than just Performance Testing

Modern performance engineering is (or should be) a full SDLC discipline embracing DevOps, Automation and Business planning. Cloud simplifies the creation and management of test environments allowing rapid spin-up / tear-down and service virtualization (SV) enables early testing of application functionality.

It has long been established that you cannot effectively performance test without automation. The same is now true for defect tracking, build control and release deployment.

Automation servers like Jenkins have become increasingly popular and the move to simplify integration through the use of published API has provided a lot of flexibility in the choice and makeup of automation solutions.

Automation focus in performance engineering terms is really about enabling performance testing and continuous performance benchmarking, across Dev, QA and Ops.



Design / Dev

Performance (by) Design

True performance engineering begins with application design and should always take the following into account:

- Geo distribution and access choice (i.e. Mobile, Browser) of end users
- A sync requirement of UX design
- Release cycle frequency once deployed
- Key Performance Indicators (KPI) for the application and hosting infrastructure
- Monitoring solution for EUE and Application performance
- The theoretical and expected maximum end-user concurrency
- The theoretical and expected maximum throughout for web / application / database servers, message queues and service endpoints (internal and external)
- Horizontal and Vertical scaling of infrastructure and application components to meet increased demand
- Performance ramifications of interaction with other systems (internal and external)

As you can see there are many design choices that can influence application performance and scalability but the focus should be on maintaining end-user experience quality at peak load which is principally a function of capacity and smart UX design. The application needs to:

- Remain available
- Maintain core functionality
- Provide an acceptable level of responsiveness.
- Scale!

Performance Defects and Work Items

Monitoring the quality of release is a primary measure of the effectiveness of performance engineering therefore performance defects should be captured and tracked in the same manner as functional defects. The aim should always be to reduce the number and shorten the time to identify - in other words shift-left.

Sprint planning for agile shops should always include performance (NFR) work items where they exist. Sprint exit criteria should evidence that any performance targets have been met and there has been no unexplainable performance regression across builds.

Performance Testing

Developers should be testing for performance and scalability as soon as a viable software component is available. The move to service-based design makes this increasingly straight-forward as service virtualization techniques allow easy mock of service consumer or end-points. It is entirely possible to test against production performance targets and SLA's even at this early stage.

Performance testing should be automated where-ever possible using Open Source offerings like JMeter or even bespoke test-harness and included in the overnight Build/Deploy/Test.

Alternatively if you have already invested in a licensed performance testing solution like Neoload or Load Runner consider making this available to Dev. The goal is to make performance metrics readily available for comparison and trending across builds.

QA / Test

Performance testing has traditionally been confined to the QA function within IT, identifying and building test assets (scripts and test scenarios) that reflect discrete sets of end user functionality. By extension this requires a partial or full application deployment to be available before meaningful testing can take place.

Delaying performance testing until this stage greatly increases the risk of performance defects creeping into release candidates and importantly reduces the time available to identify and resolve defects.

Combined with an absence of performance testing in Dev this increases the likelihood that performance defects will be surfaced delaying the release whilst they are investigated or risking a go-live decision with known defects in the hope that they can successfully be dealt with post-deployment. This leads to:

- Increasing the amount of testing required
- Slowing the velocity of release
- Release deployment with known defects
- Reduction in quality of release

Shift-Left Performance Engineering Benefits

By incorporating performance engineering into Design and Dev the number and severity of performance defects will be reduced increasing the quality of release candidates passed on to QA. This reduces the amount of system performance testing required and makes it much more likely that release candidates will meet performance targets for scalability and response time.

Deployment / Operations

Release Deployment

It is important that those who make the release decision have accurate performance data available for the release candidate. Implementing performance engineering across the full SDLC makes this happen and importantly ensures that performance benchmark data is always available for comparison with future releases and to inform production monitoring of the application.

Production Performance Benchmarking

Often referred to as BAU testing the importance of regular performance benchmarking of production websites cannot be overstated. This is the only reliable way of detecting any creeping regression and to ensure that your core systems are ready for peak loading events, scheduled or unscheduled.

Whilst it may not be possible test every aspect of core functionality in production, test assets created as part of system testing in QA are the starting point for this process and should be carefully curated so that BAU testing can be safely executed as required. For eCommerce websites this typically means deprecating activity that involves payment gateways (unless a reliable and scalable sandbox is available) and exclusion of links to the supply chain.

BAU performance testing can also help to validate key replatforming decisions - for example; migrating core application hosting to cloud, delivering a comparison of before and after performance to confirm that there has been no response time or scalability regression.

The Importance of Monitoring

It is now more important than ever to have a comprehensive monitoring strategy in place so all application components can be monitored under load. To address this requirement Application Performance Monitoring solutions (APM) have come to the fore in recent years. These solutions seek to offer real-time 24x7 monitoring of application and infrastructure performance - last mile to first mile - automatically alerting and initiating remedial action should problems occur.
The latest generation of these toolsets has also seen the incorporation of Artificial Intelligence techniques (AI) to accelerate fault domain identification and root cause analysis.

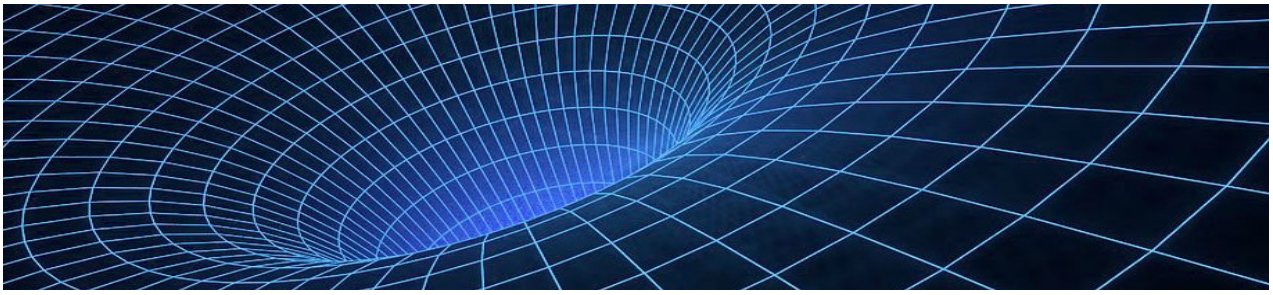
Current leading solutions include:

- DynaTrace
- Appdynamics
- New Relic

Chapter 4:

Future Trends

We have discussed the recent move to microservices and containerization both of which simplify scalability and increase speed of delivery. However new design choices in themselves do not change the basic principles of effective performance engineering.



They may influence your automation tooling choice for example; to ensure that you have appropriate monitoring coverage, or trigger a re-think of coding standards to ensure that your dev teams use them effectively.

Regardless, you still need to ensure your application design is sound from a capacity and scalability perspective and you still need to track performance defects and carry out regular performance benchmarks.

Disruptive Technology

Internet of Things (IOT)

Embedding software into devices. Not necessarily a new concept but now easier than ever to implement and covering devices as diverse as smoke alarms, vacuum cleaners and cars. The big change has been the integration possible using the IFTTT standard where common protocols allow you to do things like turn your house lights red in response to a smoke alarm detection event.

IOT devices tend to operate as self-contained entities but may periodically need to connect to native vendor systems for software updates and data exchange. Such devices invariably use API's to communicate so the same performance risks apply; endpoint availability and responsiveness and the growing proliferation of IOT devices only increases the pressure on supporting vendor systems to scale in response. Interestingly security concerns have also been prominent with several high profile instances of DDOS attacks using IOT devices weaponised as load injectors (performance testing for the wrong reasons!) IOT protocols can now be tested directly with native support present in recent releases of mainstream performance test tooling.

Artificial Intelligence (AI)

The emergence of AI as part of application design adds a new level of complexity to self-learning and expert systems. Arguably early AI systems were little more than implementations of linear math however they are rapidly becoming more sophisticated particularly where this is applied to big data and fault domain analysis.

Performance engineering considerations for AI are still evolving however they are likely to encompass distributed processing requirements either loosely or tightly coupled (you may remember grid-computing) so have the potential to require rapid exchange of large amounts of data across network connections by way of API.

AI is also having a impact on automation tooling with the emergence of:

- Automated detection and resolution of functional and non-functional defects where applications effectively test themselves. (A natural evolution of BDD/TDD and CI)
- Increased sophistication of auto-scale management in response to peak load events.
- Automated identification and resolution of fault domains as part of production APM monitoring solutions.

Increased Adoption

Compared to just 5 years ago there is clearly a much greater industry appreciation of the importance of good design in ensuring core application performance and scalability. More businesses need to embrace full SDLC performance engineering.

The benefits are clear for all to see and the implementation process can be staged and gradual. Simply moving to recording performance defects and introducing a level of performance testing in Dev can reap significant rewards in terms of increasing software quality and speed of delivery.

In Conclusion

Hopefully this short discussion on performance engineering has been interesting and thought-provoking. Infuse as a company specialise in helping clients achieve performance excellence so if you would like to discuss how we can help your own performance engineering journey please don't hesitate to get in touch.

INFUSE

ABOUT INFUSE

Infuse is a UK software testing company that provides modern software testing, transformation consulting and test environment management. We specialise in test automation and performance engineering.

Our strong alliance and partner network enables us to deliver the right solution for every client. Infuse is an Micro Focus Partner in Application Delivery Management (ADM), an Oracle Gold Partner in Application Quality Management (AQM), a CA Partner in Dev and Test and SAP Partner. We have a global partner network to enhance our delivery capabilities beyond that of a typical UK software testing company.

For more information, email _info@infuse.it or visit infuse.it

Infuse Consulting Ltd | QEII Conference Centre
Broad Sanctuary | London SW1P 3EE | UK
Tel: +44 (0)20 3755 5135

facebook.com/infuse-consulting
@infusegroup
linkedin.com/company/infuseconsulting

infuse.it