The Increasing Importance of Effective Technical Containerisation

As the use of virtualisation continues to increase, the problems with a simplistic approach are becoming obvious. It is now time for organisations to ensure that their virtualisation strategy is coherent – and fit for the future.

The days of running an application on a single physical host or cluster are fast disappearing. System virtualisation is now widespread, enabling resources to be shared across a range of different workloads.

However, as the market for virtualisation matures, the use of virtual machines (VMs) as a means of implementing workloads is showing its own problems – the multiplication of actions that are essentially shared across different workloads is still wasting resources and resulting in higher costs for the overall IT platform than should be required.

The current move is toward application containerisation, yet problems are becoming apparent in this approach as the sharing of a single underlying operating system enables security breaches to occur too easily.

This then leads to the need for system containerisation – an approach that brings together the strengths of VMs and application containers to enable a highly secure, massively dense and highly flexible operational and management platform for any workload.

Report Authors

Clive Longbottom
Tel: +44 118 948 3360
Email: Clive.Longbottom@Quocirca.com

Rob Bamforth
Tel: +44 7802 175796
Email: Rob.Bamforth@Quocirca.com

Sponsored by:

Virtuozzo

Virtuozzo

Physical, virtual and advanced platforms

Modern IT platforms are becoming more dependent on virtualisation. The use of hypervisors is now commonplace, but the way that virtualisation is being used is still going through many changes.

There are two basic types of virtualisation – hypervisors and containers.

A hypervisor sits directly on top of the hardware, enabling multiple operating systems to be hosted on it, and then, multiple applications to be run on those operating systems. Examples of this type of hypervisor include KVM, VMware ESX, Microsoft Hyper-V, and Citrix XenServer.

A container sits on top of a host operating system (OS) and enhances it with kernel updates/packages. This allows a lightweight copy of the host OS to be run multiple times on the same hardware in an isolated environment with its own set of applications. Examples of this type of virtualisation include Virtuozzo, OpenVZ and LXC.

Within the container architectures there are now two differing implementation models depending on the application itself (see figure 1). The system container model provides a full copy of the OS, allowing for any type of application to be run within it, such as legacy, persistent databases, web applications and so on. The alternate, application container model (such as Docker, Mesos and Rkt) uses a shared OS paradigm, providing a much more lightweight run-time designed for a single process or a micro-services application, typically for applications designed specifically to work in an application container environment.

Historically, virtual machines (VMs) have been used to run applications in a virtualised (Type 1 Hypervisor) environment, but there is an increasing move to the use of application containerisation.

VMs contain everything that an application needs to run including virtual hardware, a complete OS, and an application stack. This provides good security – each VM is a self-contained system that is addressing its own virtual resources that no other VM has default access to. Although a VM on the platform may be compromised, this does not impact other VMs on the same platform.

VMs have been seen as the more portable means of implementing workloads, as there is less dependency on any underlying physical resources.

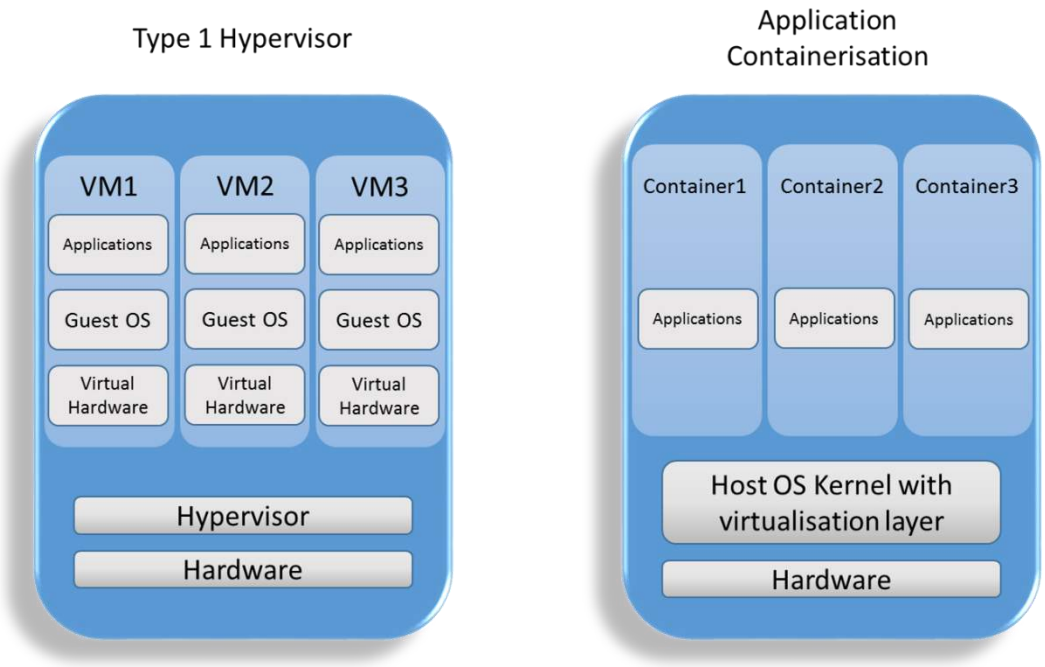

Figure 1

APPLICATION CONTAINERS

Application containers are ideal for modern, cloud-based applications, based around the use of microservices.

Where they start to fail is in the support of 'legacy' applications, built for client server or other application architecture models.

The need for simple application containers to have such a modern architectural approach can be a problem – although it is possible to encapsulate an older application within an application container, it will perform poorly and the significant benefits of encapsulation will be lost.

Falling back onto a VM model then reintroduces the many issues around resource usage of the VM model.

Only full system containerisation can provide the support for legacy and modern applications in one environment.

However, with the right choice of containerisation and the right tooling, both VMs and containers can now be moved from one environment to another as required. By ensuring that all required logical dependencies are encapsulated within the container, guest workloads can easily be moved across different platforms.

Application containers provide for higher densities of workloads, low latency and faster response times compared to VMs on the same set of physical resources. This is due to the fact that application containers share more of the underlying resource – particularly around the OS. By sharing the OS, the application container does not need to hold it itself – so containers are by default smaller. Also, where two different workloads are essentially doing the same thing, an application container can share the function (what is known as using 'shared actions') – whereas a VM will replicate it and double the workload.

The smaller size of an application container also makes them faster to deploy – but it does lead to more security concerns. It is reasonably easy to compromise an application container such as Docker through accessing it via a privileged mode. With privileged access, a user can then drill through the container to the underlying physical OS and platform. This can lead to a compromise of the underlying (shared) OS – and so every single container that is running on top of that OS becomes compromised as well. So, managing system isolation is of paramount importance – but is not easy to do.

"To be able to combine the benefits of VMs and application containers in a simple and effective manner is the optimum end point. This is available - through the use of system containers."

This is not ideal – and has led to ivory-tower arguments amongst those adhering to VMs and those to application containers.

To be able to combine the benefits of VMs and application containers in a simple and effective manner is the optimum end point.

This is available - through the use of system containers. Here, a virtual environment is created that contains a secure, virtualised Linux environment where applications can be run. By such abstraction, the application can talk to virtual OS constructs, and the security issues can be dealt with – while the workload density can still be maintained.

Through the choice of the right system container model, application containers can still be used – but they gain enhanced security and portability in the process.

The pros and cons of application containers

Essentially, the idea of an application container is that it holds only those bits that are required for a workload to be deployed and run. An application container can therefore be deployed on top of a physical OS and can share the capabilities and resources of that OS.

"A final – and in many cases, show-stopping – issue is that many applications and management functions require privileged access to the operating system. As application containers have a dependency on a physical OS, then opening up such privileged access gives that application or management function full access not only to the OS (and the platform's physical resources) itself, but often through to the other application containers on the platform. This is dangerous, as maintaining control of sysadmins has to be a high priority, and this drives a coach and horses through that need."

The main player in the application container market has been Docker, with others such as Mesos and Rkt being alternatives.

The benefits of application containerisation are that the OS can be more easily patched, upgraded and replaced as required, with all the application container workloads on that OS gaining the benefits of such changes immediately. The other benefit is that the density of workloads that can be applied to a given IT platform increases dramatically – as the OS and core resources are all shared, an application container is a fraction of the size of a VM.

Also, due to that smaller size, application containers can generally be spun up more quickly than VMs, and so offer greater benefits when it comes to business continuity and disaster recovery.

The downside is that if the underlying OS or other resources are compromised, then all application containers dependent on that platform are also compromised.

Containers can offer good levels of portability due to the level of abstraction they use. As long as each host has the same underlying OS type, containers can simply be migrated across from one platform to another. However, the dependencies of a specific workload on a specific version or patch level of an OS can still cause issues.

This dependency on a specific OS version or patch level means that application containerisation struggles. The sharing of the underlying OS means that all application containers on the platform need to be able to be run against that version and patch level of the OS. In some cases, this may not be a problem, but Quocirca's research has shown that most organisations are running applications that have such dependencies.

"This dependency on a specific OS version or patch level means that application containerisation struggles. The sharing of the underlying OS means that all application containers on the platform need to be able to be run against that version and patch level of the OS. In some cases, this may not be a problem, but Quocirca's research has shown that most organisations are running applications that have such dependencies."

A final – and in many cases, show-stopping – issue is that many applications and management functions require privileged access to the operating system. As application containers have a dependency on a physical OS, then opening up such privileged access gives that application or management function full access not only to the OS (and the platform's

physical resources) itself, but often through to the other application containers on the platform. This is dangerous, as maintaining control of sysadmins has to be a high priority, and this drives a coach and horses through that need.

However, a strong benefit of an application container over a VM is that it can have a high level of granularity. There is a slow but sure move within the industry toward a micro-services approach to application development – a looser means of creating a composite application where functions can be swapped in and out rapidly to meet the business' changing process needs. Application (or functional) containers are ideal for this, whereas the large size of VMs make this an unlikely means of creating composite applications.

It should be noted that an application container can act as a full VM – but the main advantages of the application container model will not then be obtained. A VM cannot effectively operate as a granular container; an application container should not be seen as a means of creating and managing full, run time application stacks. At the moment, it is a matter of 'horses for courses' – both VMs and containers make sense for different use cases.

However, if a suitable means of encapsulating application containers is used, then containerisation can replace VMs while maintaining the benefits of both approaches.

This brings us to how system containers work.

The concept of a system container

A system container provides a virtualised environment that enables an application container to see the underlying systems in an abstract manner. Rather than being dependent on the underlying operating system, a system container creates a 'proxy namespace' (see figure 2). This namespace abstracts calls made from application containers to a set of virtual resources rather than through to the physical resource.

By doing so, such a virtualised resource can be deemed to be dedicated to the application container.

As such, all application containers become fully removed from the underlying operating system – almost to the same extent as with a VM.

This means that each system container has its own 'copy' (although in a virtual manner) of an operating system, along with all the privileged access required to carry out actions that many applications and management systems require.

SYSTEM CONTAINERS

A system container can create what is essentially a highly efficient and dense virtual machine in which containers can securely be run.

By combining the functions of a hypervisor and a container in one system, the best of all worlds can be gained.

A system container abstracts the container away from the physical environment – whereas a native container will be dependent on e.g. physical ports, storage and so on.

By creating an abstracted environment, the system container increases security, systems availability and management of the overall environment while maintaining container densities at levels far higher than VMs on specialised hypervisors can attain by themselves.

No privileged access is allowed between containers, unless specifically and expressly allowed. Compromising the security of a system container is therefore far more difficult than that of an application container.

"However, as the functions are taking place through a proxy, the system container itself still does not carry a full copy of the underlying OS. Therefore, it maintains workload densities while providing the needed levels of performance and security."

However, as the functions are taking place through a proxy, the system container itself still does not carry a full copy of the underlying OS. Therefore, it maintains workload densities while providing the needed levels of performance and security.

The use of system containers also has other benefits. As each container is now essentially self-contained, legacy applications that are dependent on specific versions or patch revisions of operating systems can be run, as the proxy ensures that the right libraries and so on are loaded.

Provisioning and portability of the workloads is far easier – as the system container acts as a wrapper around the application container, the namespace can manage all the settings that are required as the workload moves from one place to another.

As such, system containers fit very effectively into a DevOps strategy – as the application moves from development through test and then into the production environment, the proxy namespace ensures that this is carried out seamlessly.

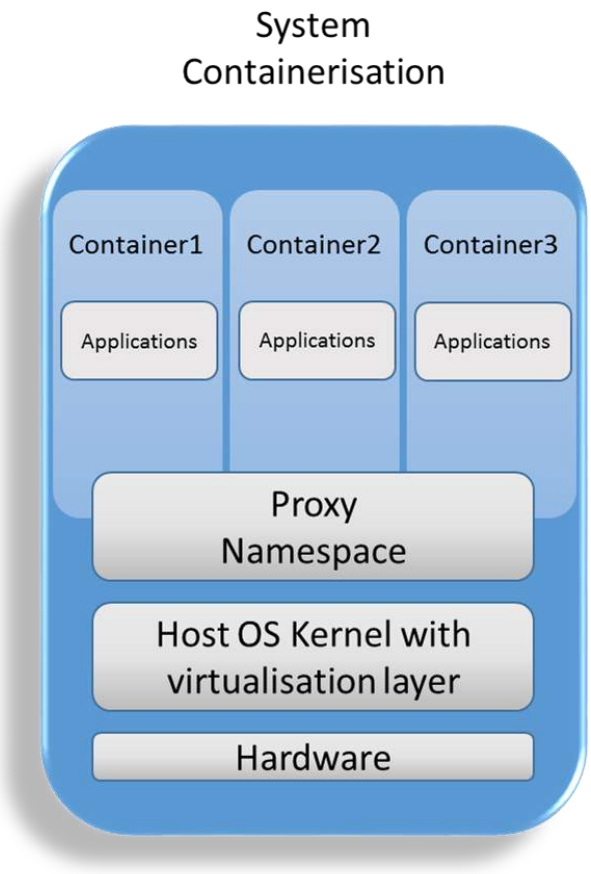

Figure 2

Further, for organisations looking to move to a hybrid cloud model, system containers allow for workloads to be moved to and from different cloud platforms as required.

"This also makes container portability much more flexible: an application container wrapped in a systems container can be moved across different OS levels without problems."

The level of abstraction also makes systems management easier. The main core of the OS can still be updated as required, with all containers using shared actions benefiting from this. The namespace ensures that those applications that are dependent on specific patch and version levels remain on those levels.

This also makes container portability much more flexible: an application container wrapped in a systems container can be moved across different OS levels without problems.

Due to projects such as CRIU.org this has also enabled namespace checkpoint and restore which in turn provides zero downtime migration across clouds and not just hosts.

Conclusions

It is clear that the old model of 'one application per physical host' is dying out. The low rates of resource utilisation and the lack of resource elasticity to meet the needs of peak workloads means that such a physical model no longer provides the required capabilities to support the business.

The move has been to an increased virtualisation of underlying resources, and this is now being bolstered through a move toward the use of containerisation.

However, the use of VMs has shown that there can still be a lot of resource wastage through having to replicate functions that are shared across multiple different workloads. The use of application containers has uncovered risks and problems around the need for core, often privileged access requirements to the underlying operating system.

To be able to combine all the advantages of the VM and application container model requires the use of an abstraction model which still enables the sharing of common actions – but with complete security.

By using a proxy namespace, this can be done.

It is not just a change in how applications are being delivered that will drive the need for system containerisation, however. There are multiple dynamics in the market which are coming together as the 'perfect storm' which will give a stark business differential to those who are effectively managing their IT platform effectively and efficiently, and those who attempt to adhere to older-style approaches.

The growth in the use of DevOps and continuous delivery will require greater granular control of code, functions, microservices and applications.

The evolution towards complex hybrid cloud environments will need much greater mobility of workloads, along with a strong business need for high availability and zero-time recovery.

Trying to manually balance a mix of VMs and application containers across a complex underlying IT platform will likely overstress many IT departments. Now is the time to plan to avoid such issues, rather than wait for the issues to impact business performance.

Quocirca recommends that any organisation that is looking at the use of application containerisation ensures that system containerisation is also on their requirements list.

Case Study

Thomas Cook/netclusive

Travel leader, Thomas Cook has to manage the travel and leisure needs of its more than 22 million customers. Thomas Cook needed a single platform to manage an e-commerce project, with a main aim of the platform being both flexible and highly available, with the need of dealing with seasonal demand peaks without adversely impacting the customer's experience.

Thomas Cook's existing hosting partner, netclusive, advised Thomas Cook that Virtuozzo was the only platform that could provide on these needs.

Thomas Cook adopted a broad set of Virtuozzo systems, using Virtuozzo containers, virtual machines and storage for this vital e-commerce platform. Through the adoption of a Virtuozzo platform, Thomas Cook has been able to seamlessly grow and shrink capacity against demand using Virtuozzo's elastic resource capabilities, while it has also realised greater reliability and better performance than before.

"Virtuozzo's flexible, demand-based resources enabled us to fully support our applications with our hosting partner, netclusive," said Axel Becker, Solution Architect Central Europe at Thomas Cook.

For netclusive, the use of Virtuozzo has also had its benefits. Whereas netclusive was primarily a web hosting company, it has now been able to diversify into complete infrastructure as a service (IaaS) offerings as well. The combination of Virtuozzo's unified storage and virtualisation capabilities along with its high availability and zero-disruption updates has allowed them to offer customers a truly elastic computing experience at low cost and high reliability.

"Virtuozzo's unified products mean the price and performance we can offer our customers is significantly better," said Cliff Simon, Managing Director, netclusive. "We can put up to three times as many virtual machines on a single physical server and run them alongside containers to maximise the use of hardware resources. Virtuozzo's ability to efficiently pool unused storage resources allows us to lower costs for ourselves and our customers."

About Virtuozzo

An industry pioneer, Virtuozzo developed the first commercially available container technology in 2001, which today is used by over 700 service providers, ISVs and enterprises to enable over 5 million virtual environments running mission-critical cloud workloads.

Today, Virtuozzo continues to innovate in areas ranging from industry-leading virtualised object storage to cloud-optimised Linux distributions to ground-breaking container migration technologies. A significant force in the open source community, Virtuozzo sponsors and/or is a contributor to numerous open source projects including OpenVZ, CRIU, KVM, Docker, OpenStack, and the Linux kernel.

Further information can be seen at www.virtuozzo.com.

About Quocirca

Quocirca is a research and analysis company with a primary focus on the European market. Quocirca produces free to market content aimed at IT decision makers and those that influence them in business of all sizes and public sector organisations. Much of the content Quocirca produces is based on its own primary research. For this primary research, Quocirca has native language telephone interviewing capabilities across Europe and is also able to cover North America and the Asia Pacific region. Research is conducted one-to-one with individuals in target job roles to ensure the right questions are being asked of the right people. Comparative results are reported by geography, industry, size of business, job role and other parameters as required. The research is sponsored by a broad spectrum of IT vendors, service providers and channel organisations. However, all Quocirca content is written from an independent standpoint and addresses the issues with regard to the use of IT within the context of an organisation, rather than specific products. Therefore, Quocirca's advice is free from vendor bias and is based purely on the insight gained through research, combined with the broad knowledge and analytical capabilities of Quocirca's analysts who focus on the "big picture". Quocirca is widely regarded as one of the most influential analyst companies in Europe. Through its close relationships with the media, Quocirca articles and reports reach millions of influencers and decision makers. Quocirca reports are made available through many media partners.

To see more about Quocirca's analysts, click here

To see a list of some of Quocirca's customers, click here

To contact Quocirca, please click here.