Using Containers to Deliver an Efficient Private Cloud

TidalScale

Software-Defined Servers

Contents

1 Solving the 3 Challenges of Containers

Every few years, a disruptive force comes along that prompts us to reframe our understanding of what something means, or how it works. For years, the notion of what a computer is and how you make one went pretty much unchallenged. Then virtualization came along, followed by cloud computing, and most recently containers. Suddenly the old rules no longer seemed to apply, or at least they didn't always apply. These disruptors made us reconsider our IT worldview.

Software-Defined Servers are having a similar effect on our notion of virtualization and containers. Software-Defined Servers combine all the resources of multiple servers – CPUs, memory, storage and network – into one or more virtual machine. This allows data analysts, data scientists and others to handle scale-up workloads that previously would have required the purchase of a larger system. In the data center, Software-Defined Servers complement other software-defined resources, including storage and networks, turning a previously fixed resource into one that can be "right-sized" on the fly.

This is a fundamental change for data centers and the private clouds they host. At TidalScale, we've seen this transformation firsthand. We've seen how manufacturers, financial services companies and research institutions are able to use their existing data center assets to create virtual servers large enough to accommodate challenging in-memory computing workloads.

2 The Fit with Containers

By introducing Software-Defined Servers to data centers, TidalScale is doing for the rack what virtualization did for the motherboard. But how does this fit with the recent IT design trend of breaking systems into their smallest services so they can be isolated in containers? How does a scale-up technology work with this scale-out methodology? Let's take a look.

Compared with traditional virtualization, containers can support a much higher load much more efficiently by sharing a common kernel where containers are provisioned as isolated processes. This means memory management, filesystems, networking and other core pieces of a server can easily be shared rather than duplicated. This also enables the allocations of the server's memory and CPU to be set independently from each other and mixed with one another to fit workloads together.

But containers aren't without their downsides. Challenges exist in three main areas:

- **Mobility** – Containers are executing processes, so live migration of a process from machine to machine isn't possible as it is with traditional virtualization. Containers don't take advantage of the virtualization extensions in modern processors that facilitate efficient mobility. Access to storage and networking is an additional challenge when many containers need to work in concert in a "swarm" deployment.
- **Orchestration** – Adding and removing containers while servers are running introduces other complex optimization problems. For instance, how to fit in new containers onto the servers that are available under their current load? This somehow has to be achieved efficiently while still allowing all networking and storage resources to remain connected to selected servers, with related containers placed so they experience minimal latency.
- **Security** – Many data center designs are underpinned by access controls that separate physical resources. But there is no hardware enforced separation between multiple containers running on the same server. Meanwhile, the orchestration challenges introduced by containers can prohibit the usual hardware techniques of limiting network and storage access.

3 How Container Deployments Benefit from TidalScale

Because TidalScale's HyperKernel (a distributed hypervisor) extends and shares an operating system kernel beyond the bounds of a single physical machine, it naturally fits with the underlying efficiency of containers. By breaking these scalability limitations, it enables the implementation of applications that resist migration to containers today because they need too many disks, memory or CPU resources. However, even for small containers, HyperKernel enables the mobility, orchestration, and security benefits of virtualization while preserving the efficiency and scalability of containers.

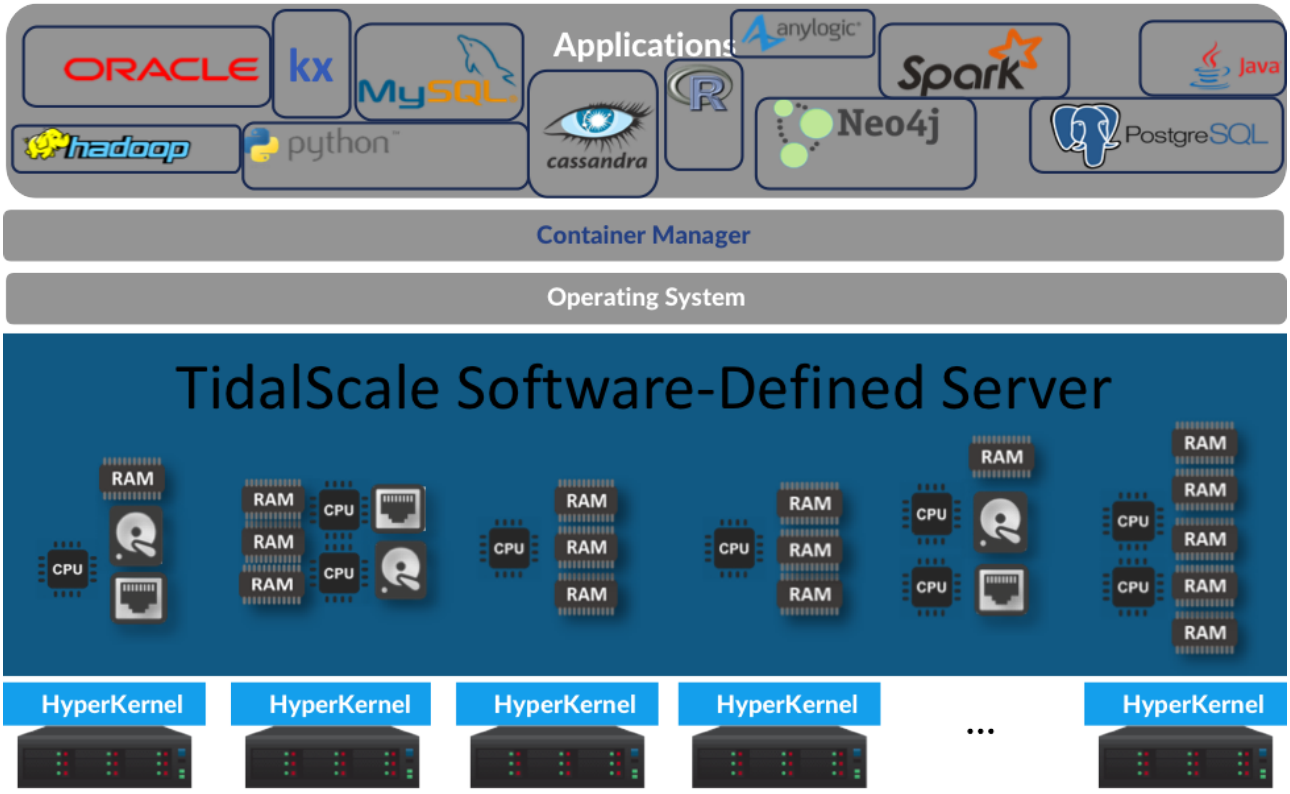


Figure 1: Containers on TidalScale Increase Utilization

Several components of TidalScale enable a container-based solution to become a complete private cloud solution:

- Leveraging a distributed hypervisor solves the mobility challenges of containers. TidalScale's HyperKernel software virtualizes of all the hardware in a server. This enables resources, such as a virtual CPU (vCPU), to move quickly and seamlessly across nodes. The result is that HyperKernel can improve efficiency by moving the vCPU to wherever the data resides in memory – a fundamental requirement in a high-performance shared design. Using the hardware below the kernel, TidalScale HyperKernel takes care of this transparently. HyperKernel also presents virtual networking and disk resources to the guest, so the OS and application both see all resources as part of a single system. These resources can be configured and mobilized as needed.
- Leveraging rack-level virtualization simplifies orchestration by nearly two orders of magnitude. It's common to have a large number of server and storage resources combined in a datacenter rack with a top-of-rack switch. Under normal circumstances, managing switching and storage provisioning can be complex. TidalScale's WaveRunner control panel simplifies this by managing the switching and storage provisioning, along with operating system images. This eliminates much of the provisioning complexity. TidalScale's ability to pool all of the resources in a rack while maintaining locality helps orchestrate containers. To put another way, because a TidalScale Software-Defined Server aggregates resources at the rack level (say, between 40 and 96 servers), it's much easier to orchestrate storage, CPU, memory and related container than on the individual server level.
- Leveraging a data center control plane preserves the security of physical resources access. The software-defined networking, storage, and server control plane exists outside of the guest operating system and is managed via a network that is not available to the guest. In addition, the containers running in the guest are even a step further removed from access to the core datacenter components.

4 How TidalScale Benefits from Containers

TidalScale benefits from the flexibility that containers provide. A common vision for many of our customers is to be able to use a small machine, and as needed scale it up to be huge. Containers fundamentally enable this. A container can grow to the size of the physical resources that are available on a single physical machine – and with TidalScale, the resources available can easily outscale any single system in a particular datacenter. The other containers on the rack can be quickly migrated to other machines to enable a large job, and then migrated back once it completes.

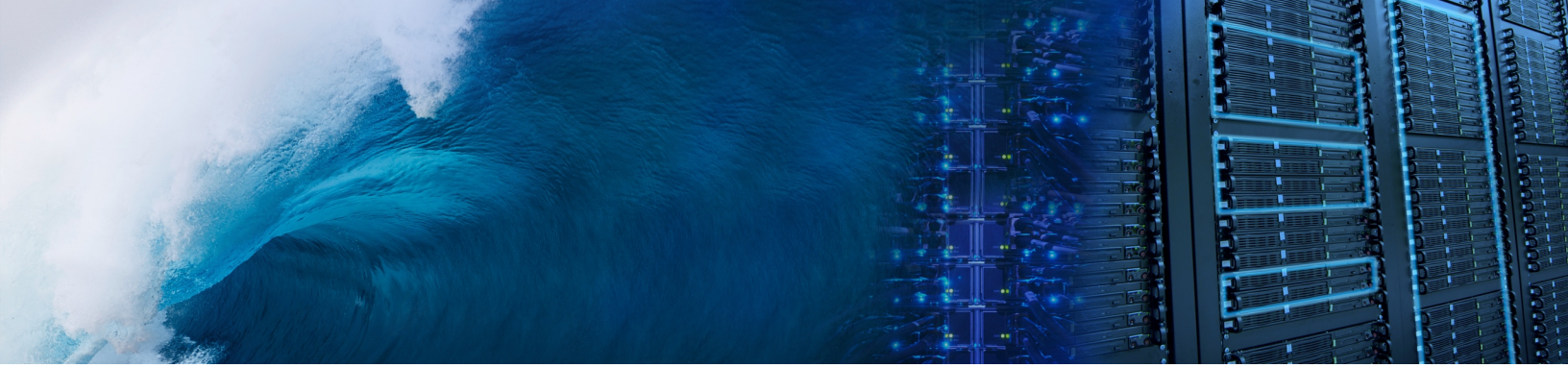
All this happens behind the scenes, at the resource management layer. The limits that can be placed on containers around memory and CPU resources enable prioritization of workloads and oversubscription of the physical resources. TidalScale's mobilization of physical resources enables hardware to be called up when an oversubscribed guest actually needs it. This is the essence of right-sizing servers on the fly.

TidalScale also benefits from the segmentation of workloads into containers. TidalScale identifies and co-locates working sets of memory, CPU, and I/O resources amongst the physical hardware it has control over. This enables high levels of performance as the vast majority of activity is local to a physical machine and has no remote access overhead.

With a large number of containers running on a TidalScale deployment, the workloads that require access to a large contiguous segment of memory are easier to co-locate on a physical machine, while the workloads with well-defined segments of activity can be spread across multiple machines with the vCPU migrating to each working set as needed. HyperKernel software continually optimizes how virtual resources are mapped in response to the actual activity of the containers, resulting in the ideal mix of containers or portions of a container across several machines.

5 Conclusion

By combining containers with a distributed hypervisor like TidalScale's HyperKernel, you can maximize the scalability and efficiency benefits of both traditional virtualization and containerization. Both at the kernel and HyperKernel levels, you can realize the mobility, orchestration and security of traditional virtualization. This enables numerous micro service style containers to run alongside large containers that appear and operate more like a traditional virtual machine. Related containers reduce communication overhead by automatically migrating to the same machine, and large applications are broken up based on their activity to run across several machines. This enables a datacenter to become an efficient private cloud with truly fluid resources where memory, CPUs, and I/O can be assigned as needed and allocated only based on actual use. To learn more about how TidalScale can help you right-size servers on the fly while taking full advantage of containers, visit tidalscale.com



TidalScale

Software-Defined Servers

TidalScale, Inc. 1694 Dell Avenue, Campbell CA USA Tel +1-877-399-0680 Fax +1-408-628-0312
www.tidalscale.com. Date: 01-Aug-17