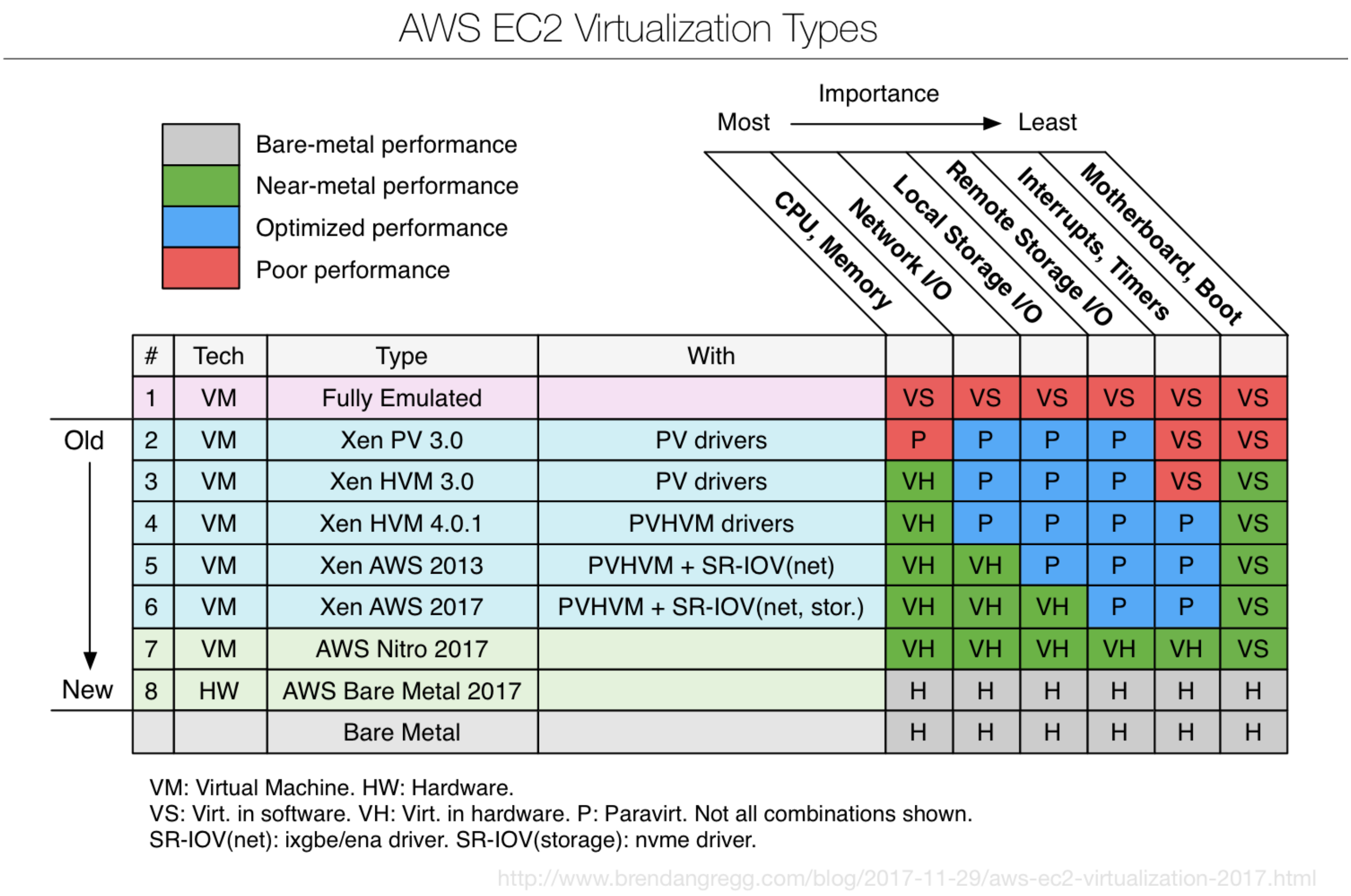
Brendan Gregg's Blog home

AWS EC2 Virtualization 2017: Introducing Nitro

29 Nov 2017



AWS EC2 Virtualization Types

⬜ Bare-metal performance
🟩 Near-metal performance
🟦 Optimized performance
🟥 Poor performance

Importance: Most → Least

	#	Tech	Type	With	CPU, Memory	Network I/O	Local Storage I/O	Remote Storage I/O	Interrupts, Timers	Motherboard, Boot
	1	VM	Fully Emulated		VS	VS	VS	VS	VS	VS
Old	2	VM	Xen PV 3.0	PV drivers	P	P	P	P	VS	VS
	3	VM	Xen HVM 3.0	PV drivers	VH	P	P	P	VS	VS
	4	VM	Xen HVM 4.0.1	PVHVM drivers	VH	P	P	P	P	VS
	5	VM	Xen AWS 2013	PVHVM + SR-IOV(net)	VH	VH	P	P	P	VS
	6	VM	Xen AWS 2017	PVHVM + SR-IOV(net, stor.)	VH	VH	VH	P	P	VS
	7	VM	AWS Nitro 2017		VH	VH	VH	VH	VH	VS
New	8	HW	AWS Bare Metal 2017		H	H	H	H	H	H
			Bare Metal		H	H	H	H	H	H

VM: Virtual Machine. HW: Hardware.
VS: Virt. in software. VH: Virt. in hardware. P: Paravirt. Not all combinations shown.
SR-IOV(net): ixgbe/ena driver. SR-IOV(storage): nvme driver.

http://www.brendangregg.com/blog/2017-11-29/aws-ec2-virtualization-2017.html

Hardware virtualization for cloud computing has come a long way, improving performance using technologies such as VT-x, SR-IOV, VT-d, NVMe, and APICv. At Netflix, we've been using these technologies as they've been made available for instance types in the AWS EC2 cloud. The latest AWS hypervisor, Nitro, uses everything to provide a new hardware-assisted hypervisor that is easy to use and has near bare-metal performance. It's an exciting development in cloud computing: hardware virtualization is now fast.

I've summarized hypervisor developments in EC2 with the above diagram. The columns show dimensions of instance performance, ordered in importance for typical workloads at Netflix (CPU-bound being the most important). The rows of this table are virtualization types, and the cells show the type of virtualization and are colored by the expected performance.

You may first notice that, over time, green is creeping in from the left. That's deliberate engineering: optimizing the most important workloads first. Each dimension has progressed through these stages:

1. **Virtualized in Software**: While this can support an unmodified guest OS, many operations are emulated and slow. Apps may run 2x to 10x slower, or worse.
2. **Paravirtualization**: The hypervisor provides efficient hypercalls, and the guest OS uses drivers and kernel modifications to call these hypercalls. It's using software and coordination between the hypervisor and guest to improve performance. I'd expect measurable overhead of 10% to 50% (depending on the PV type and workload).
3. **Virtualized in Hardware**: Hardware support for virtualization, and near bare-metal speeds. I'd expect between 0.1% and 1.5% overhead.

I'll summarize each row in the diagram, in chronological order:

1. Fully Emulated

Remember the original VMware x86 hypervisor from 1998? It's amazing to recall that it was even possible to virtualize x86 before processors had hardware-assisted virtualization (Intel VT-x and AMD-V), which were added in 2005 and 2006.

This first x86 hypervisor used emulation and binary translation for privileged operations, such as syscalls and page table operations. This had a noticeable performance cost, especially for I/O-heavy workloads. First impressions endure, and many of us were introduced to hardware virtualization by something that was known for being "slow." But that was nearly two decades ago, and I don't think this hypervisor type ever ran on EC2.

2. Xen PV 3.0

(With Xen, there are many different possible configurations, including some I'm leaving out. I'm including this one as it helps tell the story of virtualization.)

Enter paravirtualization, originally introduced in Xen, where the guest has been modified to be aware of the hypervisor and to make efficient hypercalls. In this configuration, the AMI and boot is paravirt (PV), the kernel is making hypercalls instead of privileged instructions, and the system is using paravirt network and storage drivers. This provides a performance improvement, but there are still significant overheads with privileged operations (syscalls and page table events, which also slows I/O) – although it is faster than before, so perhaps I should have colored the CPU cell yellow instead of red. This is before processor hardware virtualization for CPUs and memory (Intel VT-x, AMD-V).

The first instances on EC2 were configured like this, m1.small (thanks @cperciva).

3. Xen HVM 3.0

This row shows a more recent hypervisor configuration, running on a processor with hardware virtualization for CPUs and memory (VT-x), and using paravirt drivers for network and storage devices. The AMI and boot are now HVM. Interrupts and timers haven't been paravirtualized yet. I also started coloring "Motherboard and Boot" green, because instances of this type can boot faster than a bare metal machine can, despite software virtualization.

4. Xen HVM 4.0.1

This configuration boots HVM and uses PVHVM drivers. Those drivers are also called PV on HVM, which are paravirt drivers that use HVM features. While this is using an HVM AMI, to help differentiate it, I call these instances "PVHVM", after the drivers. This improves some kinds of workloads including interrupts and timers.

Things started to become confusing here for two reasons. Instead of glossing over these details, let me dig in:

The first source of confusion was the AMI types. AWS EC2 uses a different image type and boot process for PV and HVM, as described on the Linux AMI Virtualization Types page. People then began referring to the running instances as PV or HVM, but it's more complex than that, because HVM can boot and then run paravirt drivers (PV), and can also run paravirt on HVM Drivers (PVHVM). Most (or all) of the "HVM" instances we use on EC2 are HVM with PVHVM drivers.

The second source of confusion was performance. Earlier "HVM" versions didn't use as much paravirt, and could even be slower than "PV," causing many to recommend "PV" over "HVM." This recommendation quickly became out of date. It's all a bit confusing, and I wrote about this in 2014: Xen Modes.

When I joined Netflix in 2014, we had begun transitioning from PV to HVM (PVHVM). I came from the world of containers, and expected to be busy wrestling with the overheads of hardware virtualization and details such as PV vs PVHVM, but I found that workloads were already running pretty fast on these instances. This was because most of our workloads are CPU and memory bound, which were already virtualized in hardware. But not all workloads: some are network bound (proxies) and storage bound (databases).

5. Xen AWS 2013

Starting in 2013, some EC2 instance types began supporting hardware virtualization for network interfaces: Single Root I/O Virtualization (SR-IOV). The first was c3. AWS called this enhanced networking. This was initially used via the ixgbe driver for speeds up to 10 Gbps, then the ena driver for speeds up to 25 Gbps.

My colleague Amer Ather was responsible for testing it and making it work at Netflix, and posted some of his results: 2 Million Packets Per Second on a Public Cloud Instance. Impressive! With a solution in hand for network performance, the next target was storage.

6. Xen AWS 2017

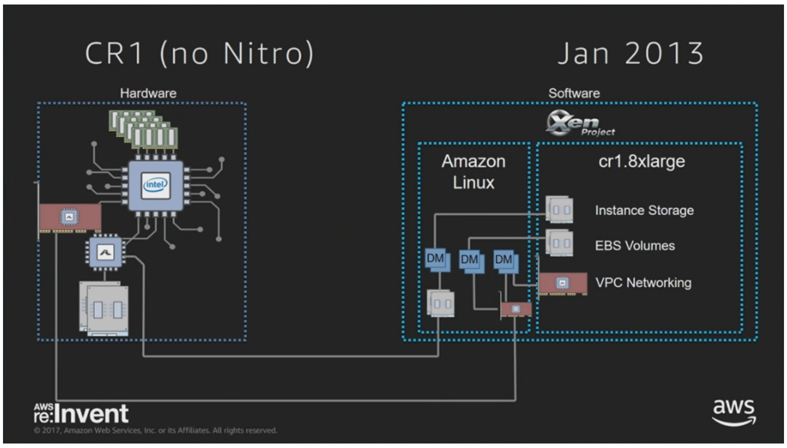
In 2015, AWS launched c4, which used hardware virtualization for EBS volumes. This was extended to instance storage devices for the x1.32xlarge in 2016. This finally came to storage-optimized instance types in 2017 with the i3 instance type, which used SR-IOV and the nvme storage driver. Amer has tested and deployed this as well, and shared some results: 3 Million Storage IOPS on AWS Cloud Instance.

This is a great development, and I've been working on a blog post to describe these new instance types: with hardware virtualization for CPU, memory, network, *and* storage. Before I could even finish this post, AWS launched the Nitro hypervisor.

7. AWS Nitro 2017

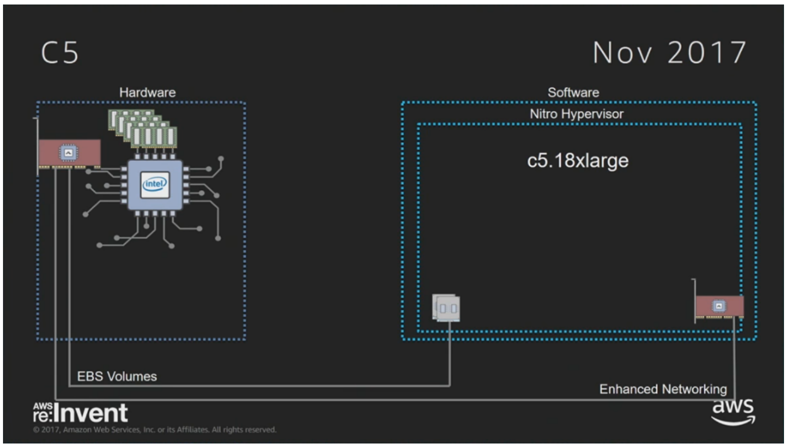
As was announced at AWS re:Invent last night, and covered in Anthony Liguori's talk today (CMP332: video), and the bare metal talk (CMP330: video), the c5 instance type uses a new hypervisor called Nitro. Nitro is lightweight. It is based on the KVM core kernel module, but does not use many of the other KVM components, such as QEMU. There is also no domO or IDD involved in the I/O path. Direct metal access.

The aim of Nitro is to provide performance that is "indistinguishable from metal." It not only uses SR-IOV for hardware virtualization of network and storage I/O (provided by custom silicon cards by annapurnalabs), but it also has hardware virtualization support of interrupts: using posted interrupts and APICv to reduce the number of VM exits. Improving interrupt performance has been described as the last battleground for hardware virtualization performance.



Before Nitro: I/O initialized via dom0 (CMP332)

As Anthony explained in his talk, these previous hardware virtualization developments were components of Nitro. They were being launched piecemeal, improving performance of pre-Nitro systems. Nitro is easier to use, as it uses all these technologies by default (that's why I left the "With" cell blank). The c5 is pictured right is EBS-only, so that diagram doesn't show direct metal access for ephemeral drives (which we'll see in other Nitro instances).



After Nitro: Direct metal I/O access (CMP332)

I've been investigating the overhead of Nitro, and have so far found it to be miniscule, often less than 1% (it's hard to measure). Nitro's performance is near-metal.

I'm also excited about Nitro because it exposes all PMC counters. I previously posted The PMCs of EC2: Measuring IPC, covering the architectural set of PMCs that were recently exposed to certain instance types in EC2. That was only seven PMCs. On the c5 Nitro instances, you have hundreds of PMCs, and can truly analyze low-level CPU performance in detail. This should help find wins of 5%, 10%, and more.

The c5s were the first to use the Nitro hypervisor, but also at re:Invent the m5 instance type was launched, also based on Nitro. AWS have said that eventually most (or all) instances will use the Nitro hypervisor, with the exception of the new Bare Metal instances.

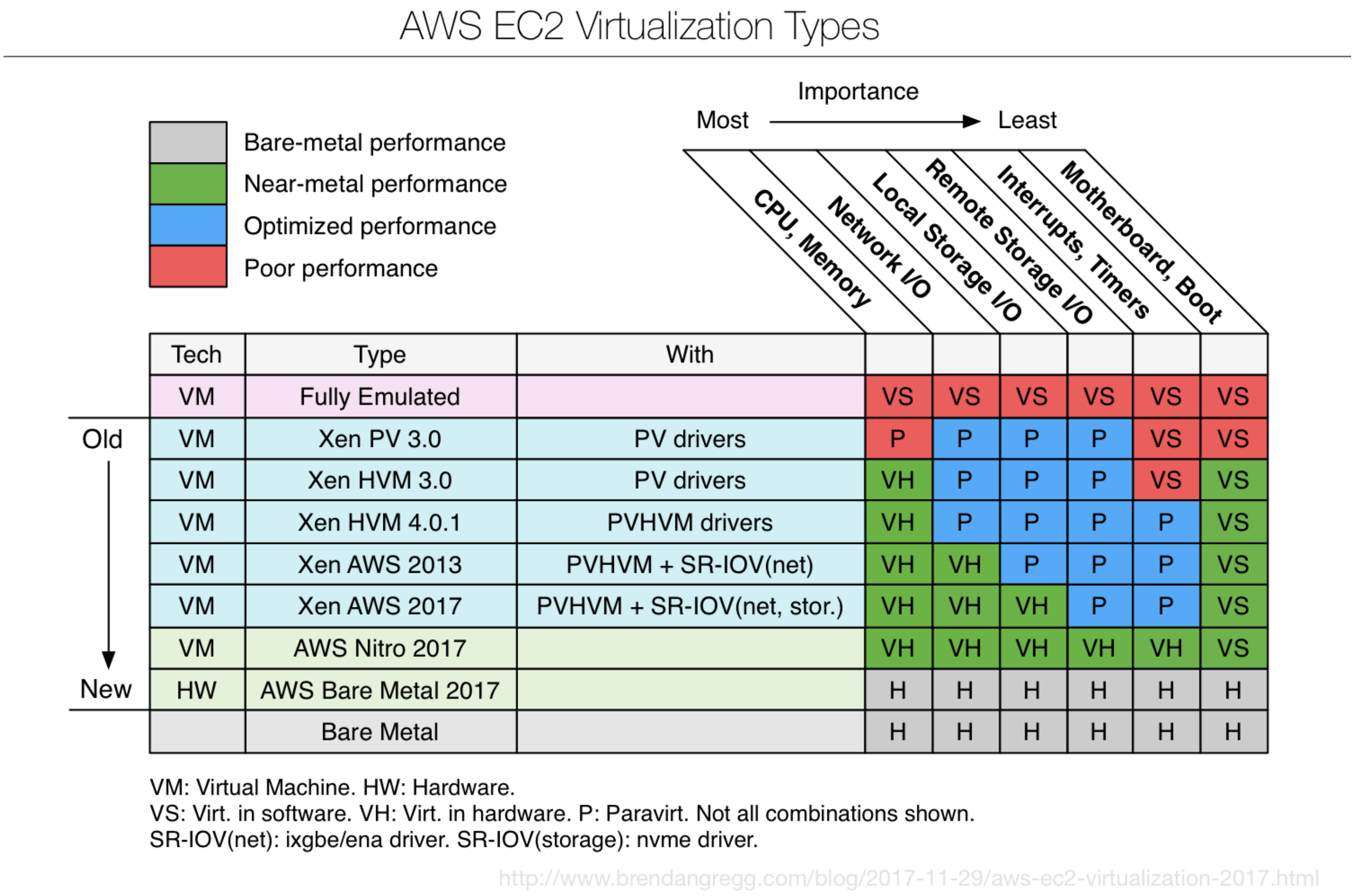
8. AWS Bare Metal 2017

Also announced at AWS re:Invent: the Amazon EC2 Bare Metal instances, which are just that – bare metal servers. 0% performance overhead. Run whatever you like: Xen, KVM, containers. Access all PMCs, and other processor features. It was covered in detail today in a talk by Matthew Wilson and Aaron Blasius (CMP330: video).

AWS has a name for their Bare Metal platform: it's the "Nitro system". The "Nitro" I discussed earlier for the c5 and m5s is the "Nitro hypervisor", which also runs on a Nitro system. It sounds a bit confusing at first, but you can hear Matt use both these terms in the CMP330 talk: start watching here.

In summary

I hope the diagram now makes more sense, and summarizes the virtualization development journey:



AWS EC2 Virtualization Types

⬜ Bare-metal performance
🟩 Near-metal performance
🟦 Optimized performance
🟥 Poor performance

Importance: Most → Least

	Tech	Type	With	CPU, Memory	Network I/O	Local Storage I/O	Remote Storage I/O	Interrupts, Timers	Motherboard, Boot
	VM	Fully Emulated		🟥 VS	🟥 VS	🟥 VS	🟥 VS	🟥 VS	🟥 VS
Old	VM	Xen PV 3.0	PV drivers	🟥 P	🟦 P	🟦 P	🟦 P	🟥 VS	🟥 VS
	VM	Xen HVM 3.0	PV drivers	🟩 VH	🟦 P	🟦 P	🟦 P	🟥 VS	🟩 VS
	VM	Xen HVM 4.0.1	PVHVM drivers	🟩 VH	🟦 P	🟦 P	🟦 P	🟦 P	🟩 VS
	VM	Xen AWS 2013	PVHVM + SR-IOV(net)	🟩 VH	🟩 VH	🟦 P	🟦 P	🟦 P	🟩 VS
	VM	Xen AWS 2017	PVHVM + SR-IOV(net, stor.)	🟩 VH	🟩 VH	🟩 VH	🟦 P	🟦 P	🟩 VS
	VM	AWS Nitro 2017		🟩 VH	🟩 VH	🟩 VH	🟩 VH	🟩 VH	🟩 VS
New	HW	AWS Bare Metal 2017		⬜ H	⬜ H	⬜ H	⬜ H	⬜ H	⬜ H
		Bare Metal		⬜ H	⬜ H	⬜ H	⬜ H	⬜ H	⬜ H

VM: Virtual Machine. HW: Hardware.
VS: Virt. in software. VH: Virt. in hardware. P: Paravirt. Not all combinations shown.
SR-IOV(net): ixgbe/ena driver. SR-IOV(storage): nvme driver.

http://www.brendangregg.com/blog/2017-11-29/aws-ec2-virtualization-2017.html

I'll leave you with one parting thought: the Bare Metal instance types are huge (eg, 72 CPUs), and you'll probably want to divide them up into cloud instances. Do you think you can setup a hypervisor to do that with <1% overhead? Even containers can have hidden overheads (like Docker's use of overlayfs or bridge networks). I'd personally find that a fun and interesting challenge, but I think it will be hard to beat Nitro. I imagine that, for most people, Nitro is exactly what they want.

Acknowledgements & References

My EC2 virtualization diagram began as a Xen modes diagram from Lars Kurth at LinuxCon EU and also published by George Dunlap on xenproject.org in 2012, which I developed further here in 2014, and now Lars is updating it on xen.org. In my new diagram, to show the importance of hardware virtualization, I've made that green, and paravirtualization is now blue.

Thanks to Amer Ather of Netflix for help and input, and from AWS, Matt Wilson, Anthony Liguori, Greg Dunn, and others, for their work on virtualization and helping me understand how it all works. Thanks to Deirdré Straughan for edits. Disclaimer: this is not an officially-endorsed AWS post.

More reading about these virtualization topics:

- https://wiki.xen.org/wiki/Xen_Project_Software_Overview
- (older) https://wiki.xenproject.org/wiki/Virtualization_Spectrum
- https://wiki.xenproject.org/wiki/Understanding_the_Virtualization_Spectrum
- https://wiki.xen.org/wiki/PV_on_HVM
- http://www.brendangregg.com/blog/2014-05-07/what-color-is-your-xen.html
- http://www.brendangregg.com/blog/2014-05-09/xen-feature-detection.html
- http://techblog.cloudperf.net/2016/05/2-million-packets-per-second-on-public.html
- http://techblog.cloudperf.net/2017/04/3-million-storage-iops-on-aws-cloud.html
- https://www.slideshare.net/AmazonWebServices/sdd406-amazon-ec2-instances-deep-dive-aws-reinvent-2014
- Where I introduced this diagram: https://www.slideshare.net/brendangregg/how-netflix-tunes-ec2-instances-for-performance
- AWS re:Invent 2017: Amazon EC2 Bare Metal Instances (CMP330): https://www.youtube.com/watch?v=o9_4uGvbvnk
- AWS re:Invent 2017: C5 Instances and the Evolution of Amazon EC2 Virtualization (CMP332): https://www.youtube.com/watch?v=LabltEXk0VQ

Comments for this thread are now closed ×

5 Comments **Brendan Gregg's Blog** 1 Login

♡ Recommend 13 Tweet Share Sort by Best

Ivan Glushkov • a year ago

Excellent!!!
Brendan, do you have any info, why they get rid of Xen? Why KVM is better?

1 ^ | v • Share ›

Noteworthy ➦ Ivan Glushkov • a year ago

Asking myself the same question.

^ | v • Share ›

Bill Metangmo ➦ Noteworthy • a year ago

Not sure but as with AWS 2017, I/O is handled with SR-IOV & cpu/mem is handled with VT-x/ VT-d, I think that the need of PV drivers with a dom0 is less important to cahieve performance...

^ | v • Share ›

Piyush Kansal • a year ago

Excellent post. Loved the "Virtualization Types" diagram

^ | v • Share ›

YuvarajLoganathan • a year ago

Excellent Post. Loved it :D

^ | v • Share ›

✉ Subscribe Ⓓ Add Disqus to your siteAdd DisqusAdd 🔒 Disqus' Privacy PolicyPrivacy PolicyPrivacy

You can comment here, but I can't guarantee your comment will remain here forever: I might switch comment systems at some point (eg, if disqus add advertisements).

About this blog