Qemu support for Windows on Arm

Mohamed Mediouni

About me

- EC2 @ AWS
- My opinions are mine

AGENDA

A primer on what's going on under the hood in WSL2

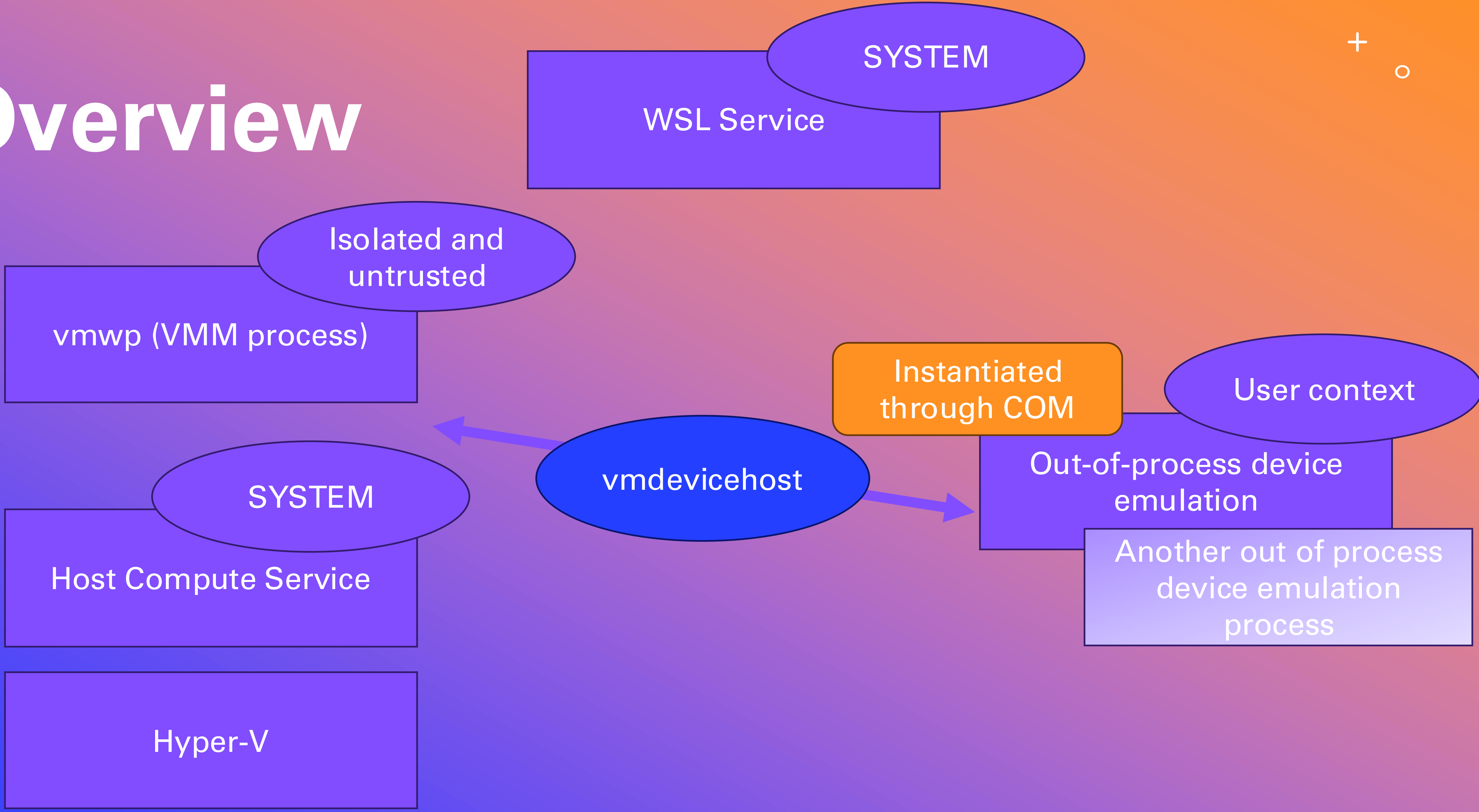
Hyper-V out-of-process device emulation: a public API

Windows Hypervisor Platform on Arm64

WSL2 AND WSA

HCS and virtio device emulation

Overview


SYSTEM

WSL Service

Isolated and untrusted

vmwp (VMM process)

SYSTEM

Host Compute Service

Hyper-V

vmdevicehost

Instantiated through COM

User context

Out-of-process device emulation

Another out of process device emulation process

File sharing

- virtiofs emulation is *not* part of WSL2. It is provided by Windows in v9pfs.dll.
- 8GB virtiofs BAR for caching.
- Can be initialised via HCS APIs (Plan9 device type notably)
- For Windows Sandbox/Windows guests, VSMB is used instead.

wsldevicehost

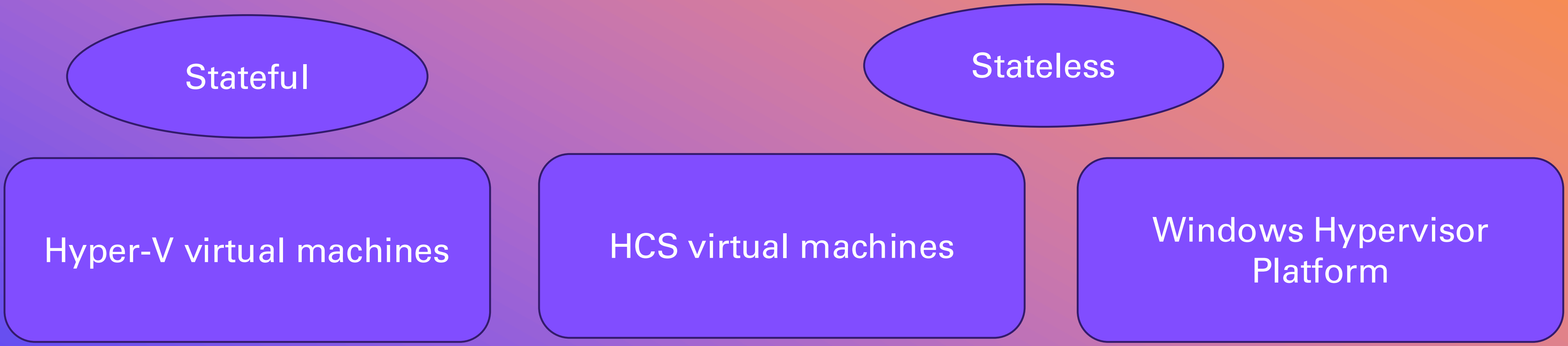
- Component written in Rust. This is part of the WSL2 and WSA packages. There’s also an implementation of virtio-net present in there, and virtio-pmem too.
- This component uses vmdevicehost, but does not leverage vmvirtio.dll, which is a private DLL which implements common virtio infrastructure.

virtio-gpu

- Part of the Windows Subsystem for Android.
- gfxstream_backend.dll, loaded from WsaClient.exe.
- Backed by gfxstream, supported across GPU vendors.
- Completely separate from the vGPU infrastructure on WSL2.
- Windows Subsystem for Android *did* have an experimental WSL2-like vGPU infrastructure for it, but that never reached production status. WSA itself is currently deprecated.

HOST COMPUTE SERVICE

Multiple types of Hyper-V VMs



Stateful

Stateless

Hyper-V virtual machines

HCS virtual machines

Windows Hypervisor Platform

Restricted to Professional SKUs upwards.

All SKUs.

Close to no good public documentation – a lot of trial and error.

Actually, easier to get started with.

HCS

- Supports both virtual machines and containers (Silos in NT parlance)
- JSON given with settings (which are ephemeral)
- Support for custom I/O devices.
- https://github.com/M2Team/NanaBox is a good starting point to start from.
- Documented to some extent, but the docs aren’t easy to parse & lack of good samples/guides.

HYPER-V OUT-OF-PROCESS DEVICE EMULATION

Security model… using COM to start the virtual devices

COM

- Device emulation for virtio devices is done out of process via the HCS device virtualization APIs.
- dllhost.exe process, with using COM to start device emulation in a different user context in a secure manner.
- Those security measures are at the end quite optional if you run as administrator, which made experiments much easier to get started with.

vmvirtio

- A small layer on top of vmdevicehost.
- Undocumented API – but the virtiofs implementation is built on top of it

vmdevicehost

- Provides an API for access to guest memory.
- For the config space: those are blocking reads/writes
- A doorbell design where it’s possible to trap writes to parts of the BARs.
- For trapped writes: the write is *not* done unlike the virtual PCIe devices implementations in some other places. Has to be done manually if wanted.
- Suitable for implementing virtio and nvme.

Supporting it in Qemu

- Prototype of a qemu-oopdevice binary, but that does not cover using the stack with COM security enabled.
- Would it be worthwhile to continue on this road or are there no big benefits in using Hyper-V's VMM to make it worthwhile?
- Should we have out-of-process emulation of arbitrary* PCIe devices on Linux?

WINDOWS HYPERVISOR PLATFORM

Windows Hypervisor Platform

- An API to support third-party VMMs on Windows, while still running on top of Hyper-V.
- *One* VM allowed per process.
- On arm64: in preview since Windows 11 version 24H2
- On x86: has been there since quite a while now, and already supported by Qemu, VirtualBox and a few others
- Documentation is not ideal either for this one, but things are quite a bit easier to figure out.

GIC

- State on 24H2 (26100): no way to move the GIC MMIO address ranges to another location.
- Worked around for those builds via hacking in to not overlap with the GIC location in conventional Hyper-V VMs.
- Not an upstreamable configuration – except if we’re going to do an -M hyperv.
- Addressed in a prerelease build – for which the SDK not publicly available yet.
- GICv3, with no ITS

Hypercalls

- Always exposes Hyper-V hypercalls.
- The PSCI implementation in Hyper-V is always used.
- Hyper-V VMs have virtio support despite not having an ITS for MSI-X, with a custom irqchip implemented inside of the Hyper-V PCIe code.
- Consequence: no MSI-X devices are currently functional with Qemu on Windows Hypervisor Platform for arm64. WHPX has WHv vPCI APIs, but I didn’t start using those yet.

Future plans

- Getting MSI-X working so that virtio devices become functional.
- Writing a proper -M hyperv target to be enable external use before the new SDK becomes public.
- Publish an RFC

THANK YOU

mediou@amazon.de