PVH:
Faster, improved guest model for Xen

Elena Ufimtseva

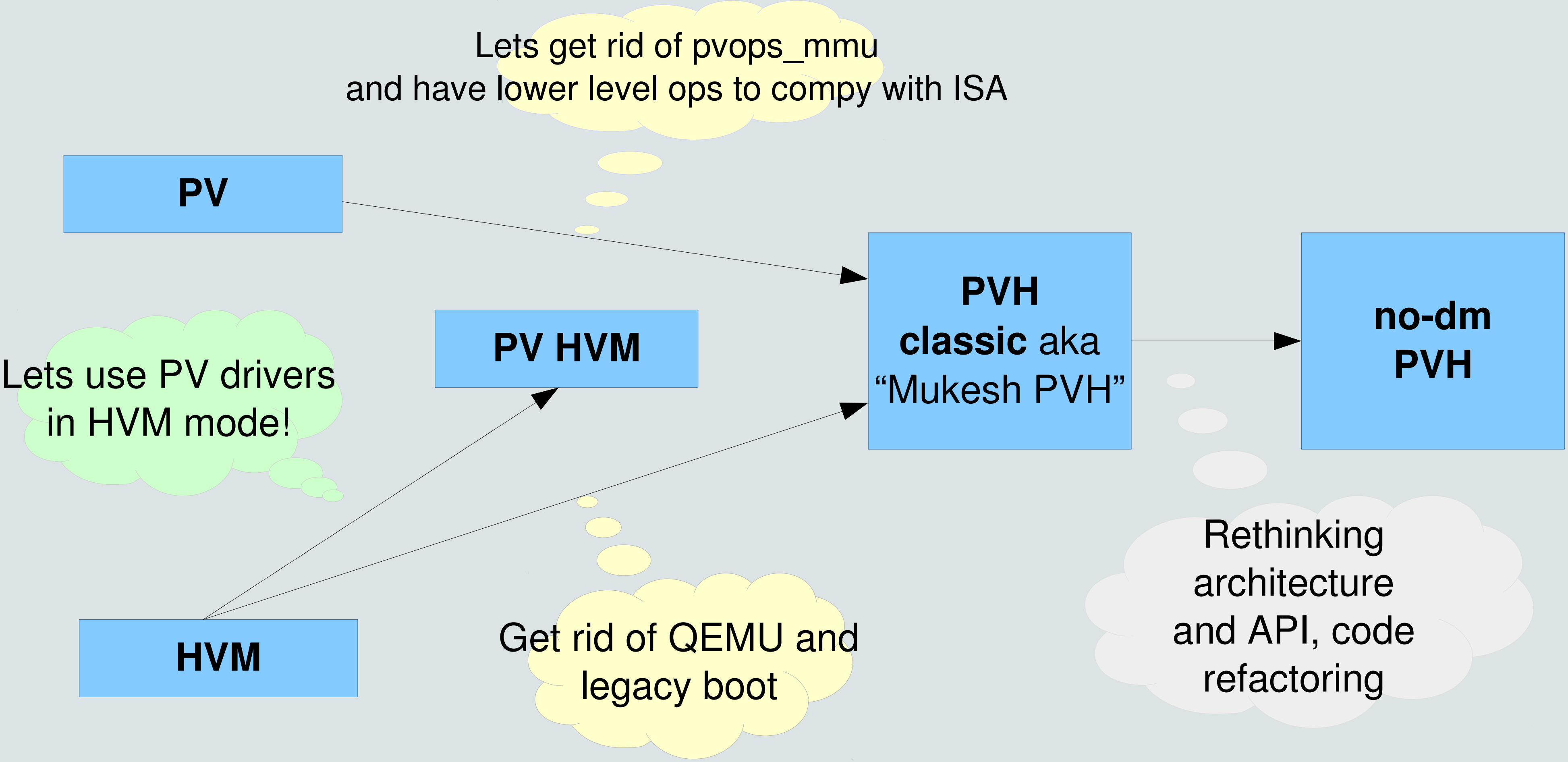
Software Engineer

Oracle Corp

Agenda

- What is PVH guest model?
- Development advances.
- Technical details.
- Advantages/disadvantages.
- Future plans.

Xen guest models development history



Virtualization spectrum

	HVM*	PV	PVH
Boot	emulated	paravirt	paravirt
Memory	hardware	paravirt	hardware
Interrupts	paravirt	paravirt	paravirt
Timers	paravirt	paravirt	paravirt
Spinlocks	paravirt	paravirt	paravirt
Disk	paravirt	paravirt	paravirt
Network	paravirt	paravirt	paravirt
Devices	emulated	paravirt	paravirt/hardware

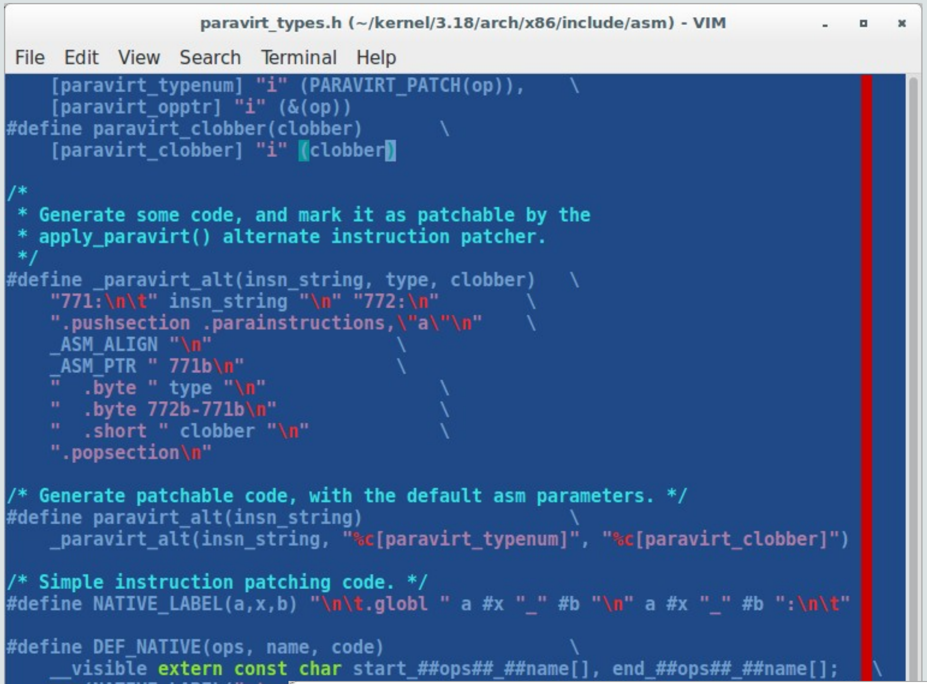
hardware – best performance using cpu feautures

ORACLE

PV limitations

- MMU pv_ops
  - page tables are mapped R/O, traps to hypervisor
- Slow 64 syscalls
  - CPL 3 for both kernel and userspace
  - need to bounce to Xen to switch context to guest kernel on system calls
- X86 maintainers and API documentation
  - Absence of specifications makes it hard to upstream;

PVOPS and binary patching


paravirt_types.h (~/kernel/3.18/arch/x86/include/asm) - VIM
File Edit View Search Terminal Help

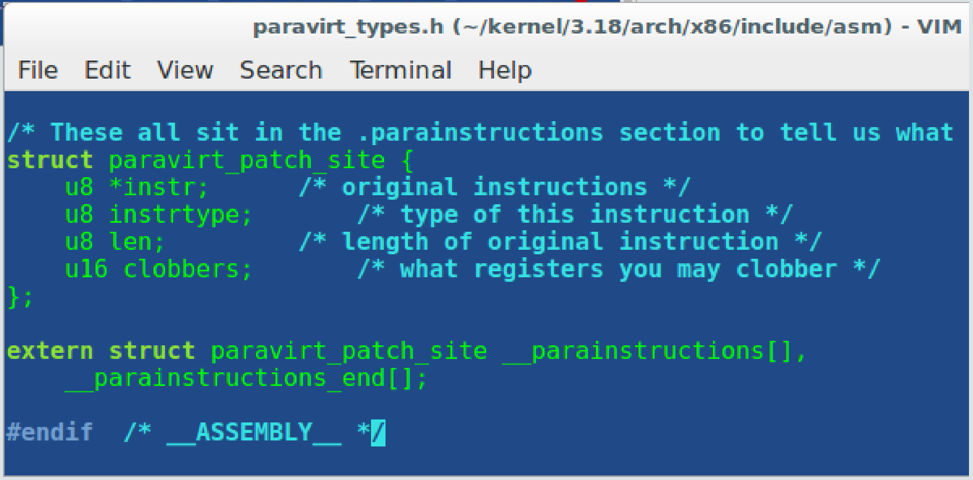
```
    [paravirt_typenum] "i" (PARAVIRT_PATCH(op)),     \
    [paravirt_opptr] "i" (&(op))
#define paravirt_clobber(clobber)          \
    [paravirt_clobber] "i" (clobber)

/*
 * Generate some code, and mark it as patchable by the
 * apply_paravirt() alternate instruction patcher.
 */
#define _paravirt_alt(insn_string, type, clobber)    \
    "771:\n\t" insn_string "\n" "772:\n"          \
    ".pushsection .parainstructions,\"a\"\n"      \
    _ASM_ALIGN "\n"                         \
    _ASM_PTR " 771b\n"                      \
    "  .byte " type "\n"                        \
    "  .byte 772b-771b\n"                       \
    "  .short " clobber "\n"                    \
    ".popsection\n"

/* Generate patchable code, with the default asm parameters. */
#define paravirt_alt(insn_string)                        \
    _paravirt_alt(insn_string, "%c[paravirt_typenum]", "%c[paravirt_clobber]")

/* Simple instruction patching code. */
#define NATIVE_LABEL(a,x,b) "\n\t.globl " a #x "_" #b "\n" a #x "_" #b ":\n\t"

#define DEF_NATIVE(ops, name, code)                      \
    __visible extern const char start_##ops##_##name[], end_##ops##_##name[];  \
    asm(NATIVE_LABEL("star
```


paravirt_types.h (~/kernel/3.18/arch/x86/include/asm) - VIM
File Edit View Search Terminal Help

```
/* These all sit in the .parainstructions section to tell us what
struct paravirt_patch_site {
    u8 *instr;      /* original instructions */
    u8 instrtype;       /* type of this instruction */
    u8 len;         /* length of original instruction */
    u16 clobbers;       /* what registers you may clobber */
};

extern struct paravirt_patch_site __parainstructions[],
    __parainstructions_end[];

#endif  /* __ASSEMBLY__ */
```

- Binary patching to optimize path.
- We want cpuid() to be replaced with native_cpuid() (or xen_cpuid) cpuid (..)→ native_cpuid(..).
- We need to know where in the kernel and with what call to replace it with (offset).

ORACLE

PVOPS and binary patching

```
/* See if we can find out some more. */
if (cpuid_eax(0x80000000) >= 0x80000005) {
          /* Yes, we can. */
```

```
.text
ffffffff81432725:      ff 14 25 90 01 82 81     callq  *0xffffffff81820190
```

generate some code and mark code as patchable with paravirt_alt()

```
  #define _PVSITE(ptype, clobbers, ops, word, algn)   \
771:;                    \
   ops;                  \
772:;                    \
   .pushsection .parainstructions,"a"; \
    .align algn;              \
    word 771b;             \
    .byte ptype;              \
    .byte 772b-771b;          \
    .short clobbers;          \
   .popsection
```

```
.parainstructions:
  443  ffffffff81921c40 25274 ffffffff 1f07ff01 00000000  %'C............
```

Overwrite @ffffffff81921c40 with ffffffff8102849a

```
In pv_cpu_ops:
 ffffffff8102849a <native_cpuid>:
```

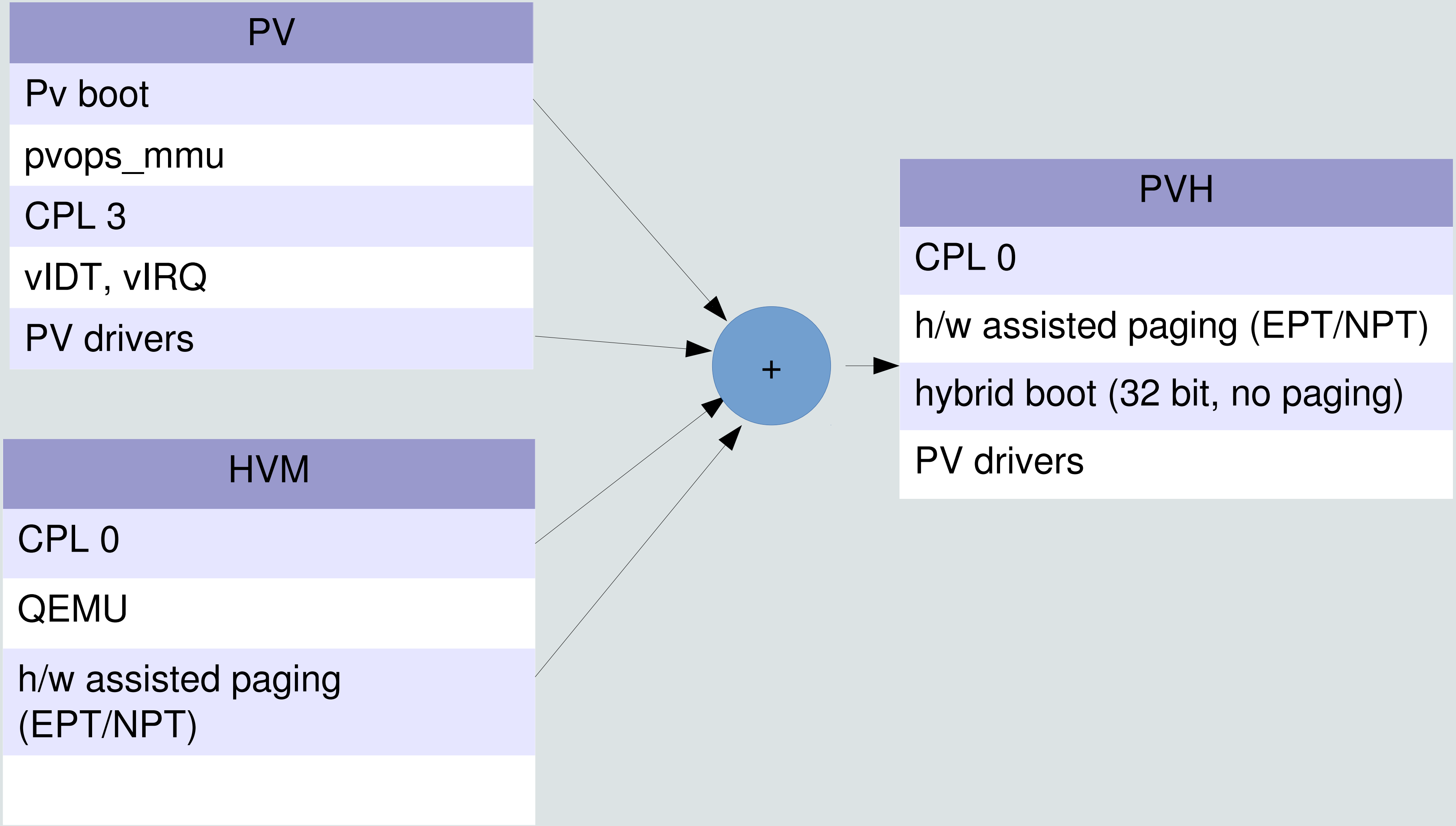
during boot, native_patch calls paravirt_patch_insn which figures out the delta and makes it a call ffffffff81003062

```
ffffffff81003062 <xen_cpuid>:
```

PVHVM limitations

- QEMU:
  - emulated devices.
  - QEMU runs in user space of Dom0 or as MiniOS guest.
- Legacy boot, emulated firmware.
- Platform emulation – ISA, PCI bus, PIIX4, etc.

PVH - best of both worlds



PV
Pv boot
pvops_mmu
CPL 3
vIDT, vIRQ
PV drivers

HVM
CPL 0
QEMU
h/w assisted paging (EPT/NPT)

PVH
CPL 0
h/w assisted paging (EPT/NPT)
hybrid boot (32 bit, no paging)
PV drivers

PVH – best of both worlds

- Disable QEMU – no emulated devices – such as legacy drivers (floppy, VENOM), or serial port.
- Xen manages P2M, guest manages page tables.
- HVM hypercalls (smaller subset than PV hypercalls).
- Special APs booting (PVH specific).

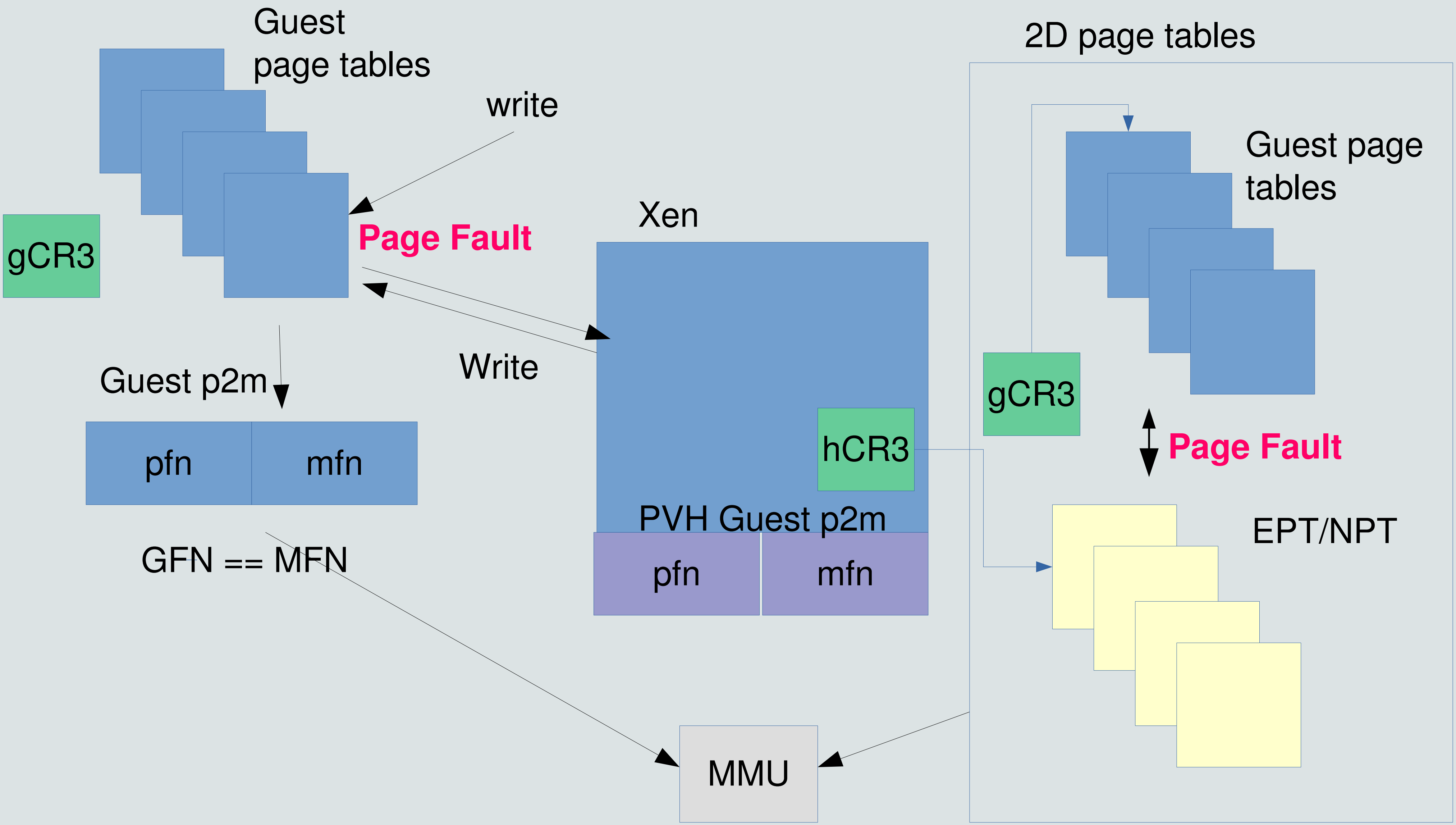
PVH – best of both worlds

- No BIOS, no firmware:
  - Start Linux kernel in 32-bit, no paging. Execute at defined physical offset within Linux binary.
  - E820 map from hypervisor.
- No MMU pv_ops, native paging.
- No hypercalls for LDT,GDT, FPU, Crx, CPUID, etc.
- P2M managed by Xen.
- CPL 0.
- Hybrid boot path.

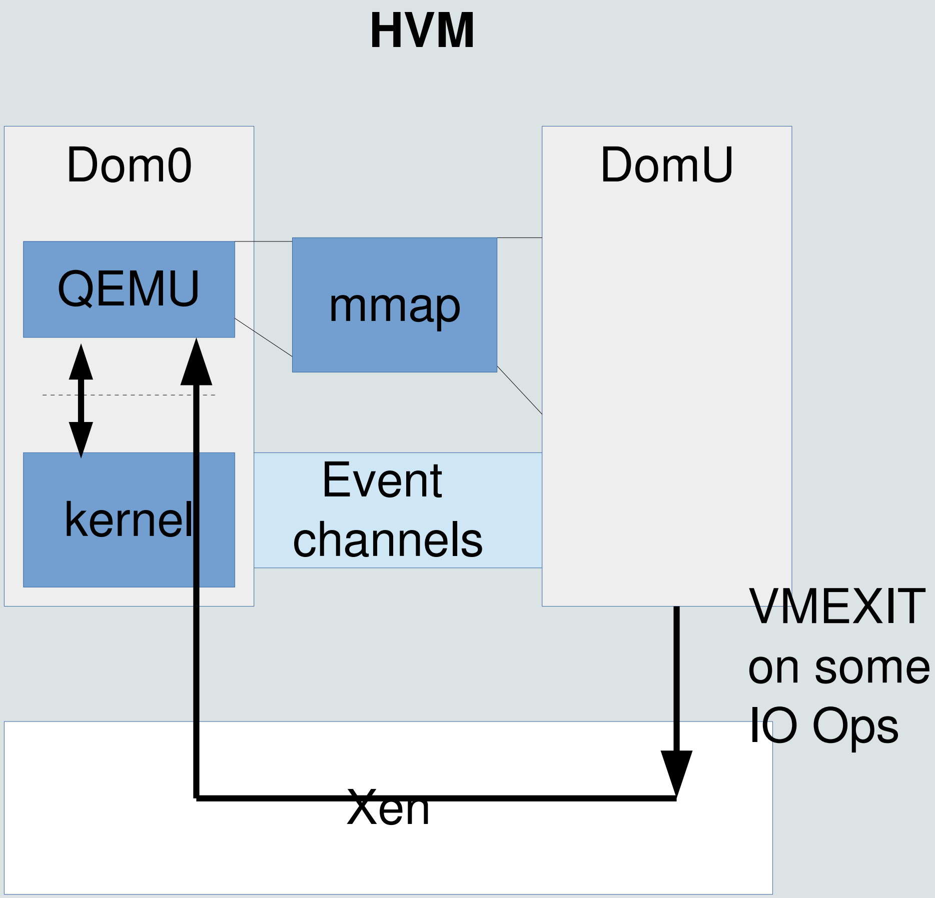
PVH Linux impact

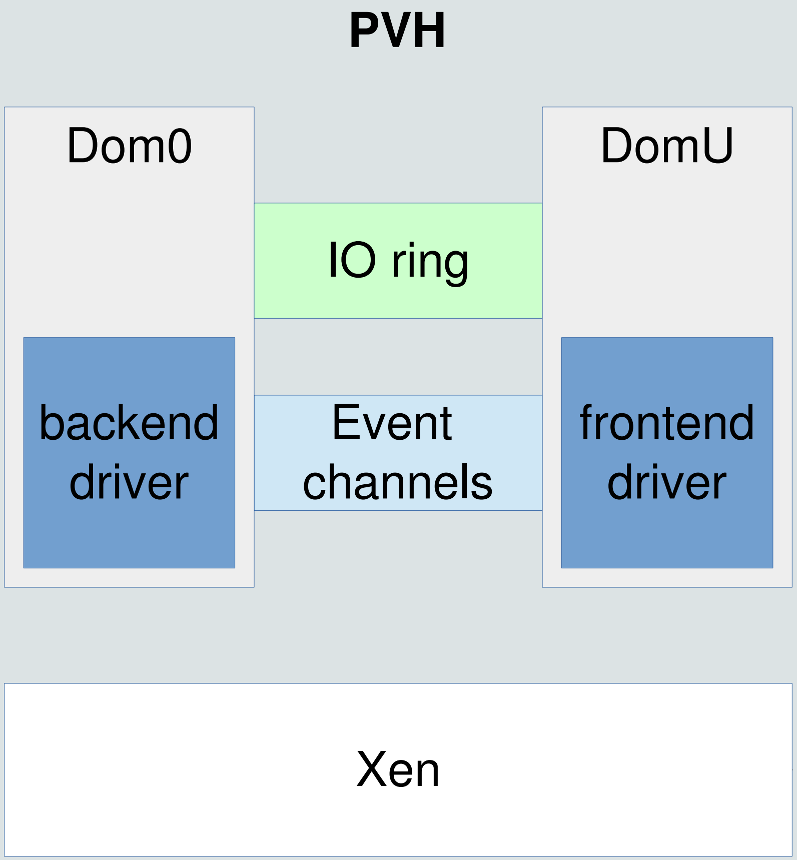
- Tons of pvops disabled, low level calls follow x86 ISA semantics.
- Tons of pv specific code paths can be disabled.
- Guest is in charge of page tables.
- tlb_flush_others (optimization to deal with sleeping vCPUs).

Memory management in PV and PVH

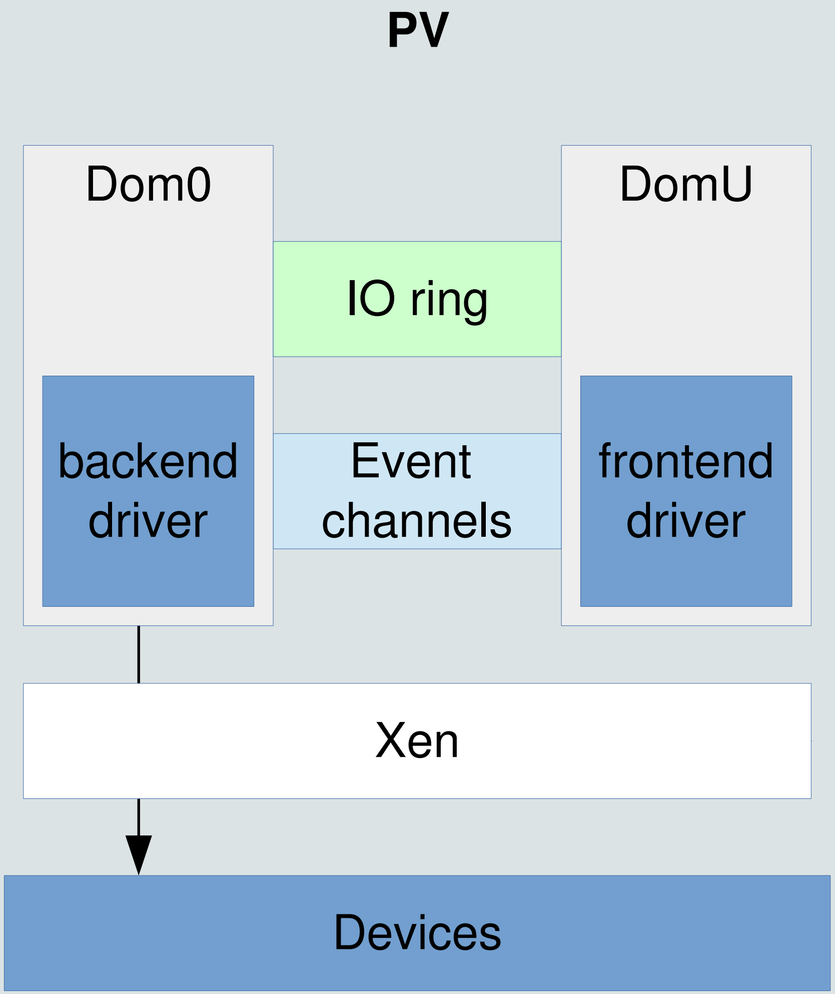


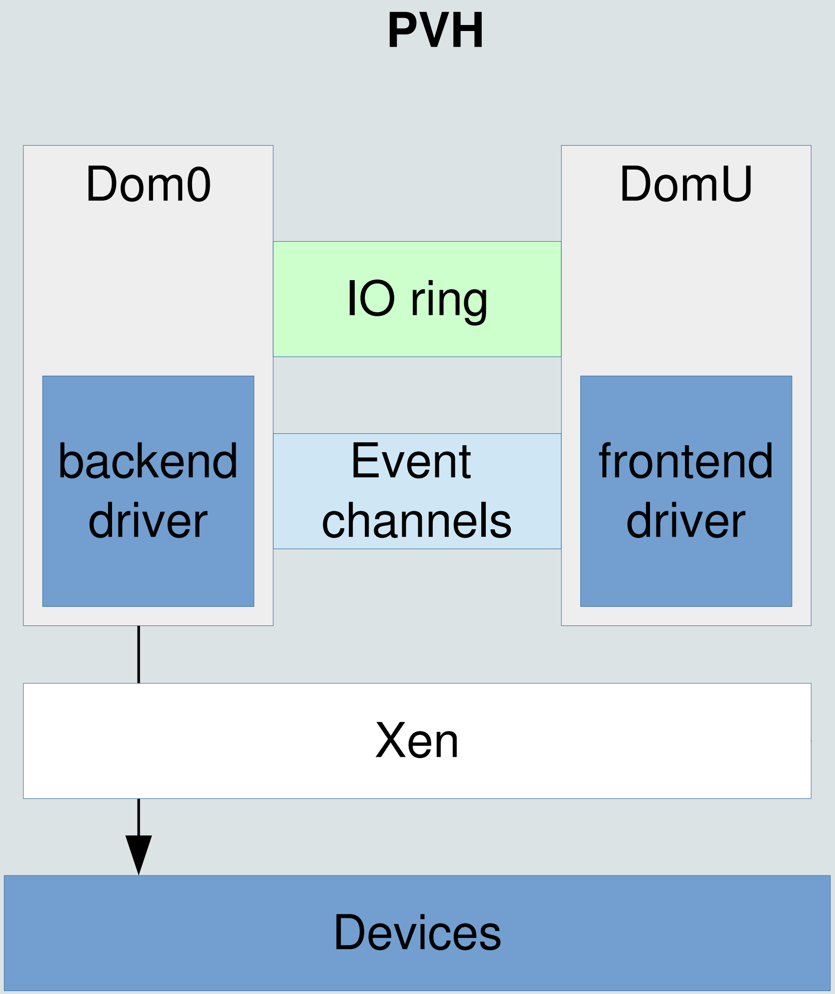
IO ops on HVM and PVH





IO Ops on PV and PVH guest



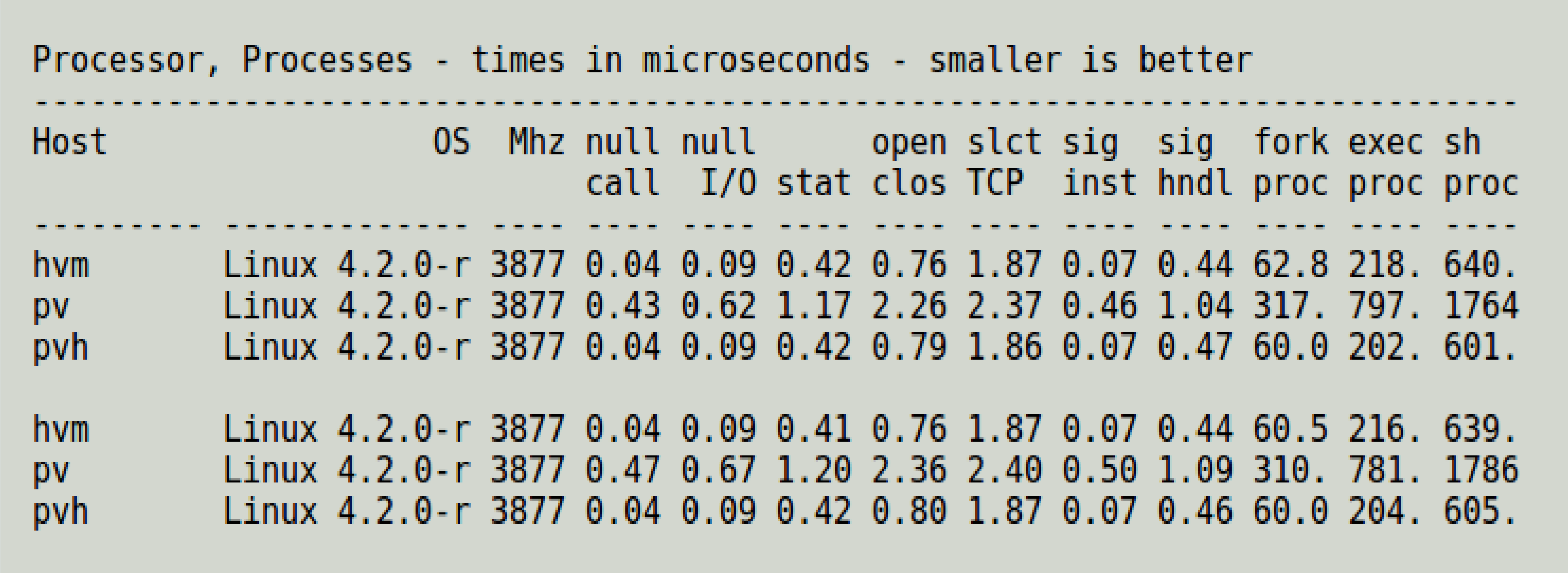


Advantages/disadvantages

- Fast!
- Small security impact – no QEMU!
- Bad things:Only works on Linux, MiniOS and FreeBSD.
- Legacy deployments (older hyper visors) can't boot this guest.

Performance (lmbench)

Lenovo ThinkCentre M93p, Intel x86-64, i7-4770 CPU @ 3.40GHz, 8CPUs, 16GB
Dom0 configuration: 2GB,8x vCPU, Linux 4.2.0-rc6+
guest configuration: 1GB, 1x vCPU, Linux 4.2.0-rc6+
Xen 4.6-unstable

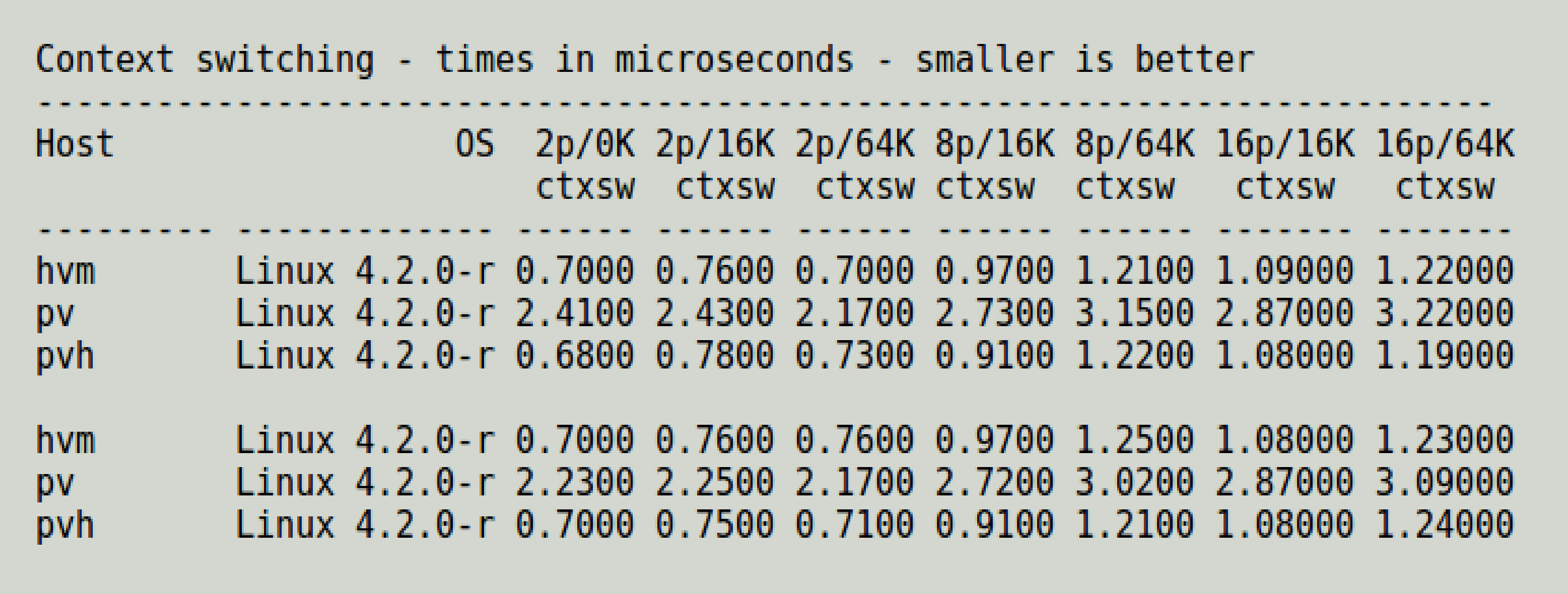


```
Processor, Processes - times in microseconds - smaller is better
------------------------------------------------------------------------------
Host                 OS  Mhz null null      open slct sig  sig  fork exec sh
                             call  I/O stat clos TCP  inst hndl proc proc proc
--------- ------------- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
hvm       Linux 4.2.0-r 3877 0.04 0.09 0.42 0.76 1.87 0.07 0.44 62.8 218. 640.
pv        Linux 4.2.0-r 3877 0.43 0.62 1.17 2.26 2.37 0.46 1.04 317. 797. 1764
pvh       Linux 4.2.0-r 3877 0.04 0.09 0.42 0.79 1.86 0.07 0.47 60.0 202. 601.

hvm       Linux 4.2.0-r 3877 0.04 0.09 0.41 0.76 1.87 0.07 0.44 60.5 216. 639.
pv        Linux 4.2.0-r 3877 0.47 0.67 1.20 2.36 2.40 0.50 1.09 310. 781. 1786
pvh       Linux 4.2.0-r 3877 0.04 0.09 0.42 0.80 1.87 0.07 0.46 60.0 204. 605.
```

Performance (lmbench)

Lenovo ThinkCentre M93p, Intel x86-64, i7-4770 CPU @ 3.40GHz, 8CPUs, 16GB
Dom0 configuration: 2GB,8x vCPU, Linux 4.2.0-rc6+
guest configuration: 1GB, 1x vCPU, Linux 4.2.0-rc6+
Xen 4.6-unstable



```
Context switching - times in microseconds - smaller is better
-------------------------------------------------------------------------
Host                 OS  2p/0K 2p/16K 2p/64K 8p/16K 8p/64K 16p/16K 16p/64K
                         ctxsw  ctxsw  ctxsw ctxsw  ctxsw   ctxsw   ctxsw
--------- ------------- ------ ------ ------ ------ ------ ------- -------
hvm       Linux 4.2.0-r 0.7000 0.7600 0.7000 0.9700 1.2100 1.09000 1.22000
pv        Linux 4.2.0-r 2.4100 2.4300 2.1700 2.7300 3.1500 2.87000 3.22000
pvh       Linux 4.2.0-r 0.6800 0.7800 0.7300 0.9100 1.2200 1.08000 1.19000

hvm       Linux 4.2.0-r 0.7000 0.7600 0.7600 0.9700 1.2500 1.08000 1.23000
pv        Linux 4.2.0-r 2.2300 2.2500 2.1700 2.7200 3.0200 2.87000 3.09000
pvh       Linux 4.2.0-r 0.7000 0.7500 0.7100 0.9100 1.2100 1.08000 1.24000
```

Future of PVH

- cpuid filtering
- dom0 support (IOMMU required)
- tsc scaling on amd
- HVM instance construction- w/o toolstack
- MMIO accelarators
- pci passtrough (IOMMU required)
- bugfixes
- timer_mode support
- check shadow paging?

References

Xen source code

Kernel source code

Intel SDM

AMD Developers Manual

http://www.linuxplumbersconf.org/2012/wp-content/uploads/2012/09/2012-lpc-virt-intel-vt-feat-nakajima.pdf

http://www.cs.rochester.edu/~sandhya/csc256/seminars/hedayati_vm_npt.pdf

http://www-archive.xenproject.org/files/summit_3/stub-xensummit-sept06.pdf

http://www.slideshare.net/RampantJeff/qemu-binary-translation

Thank you!