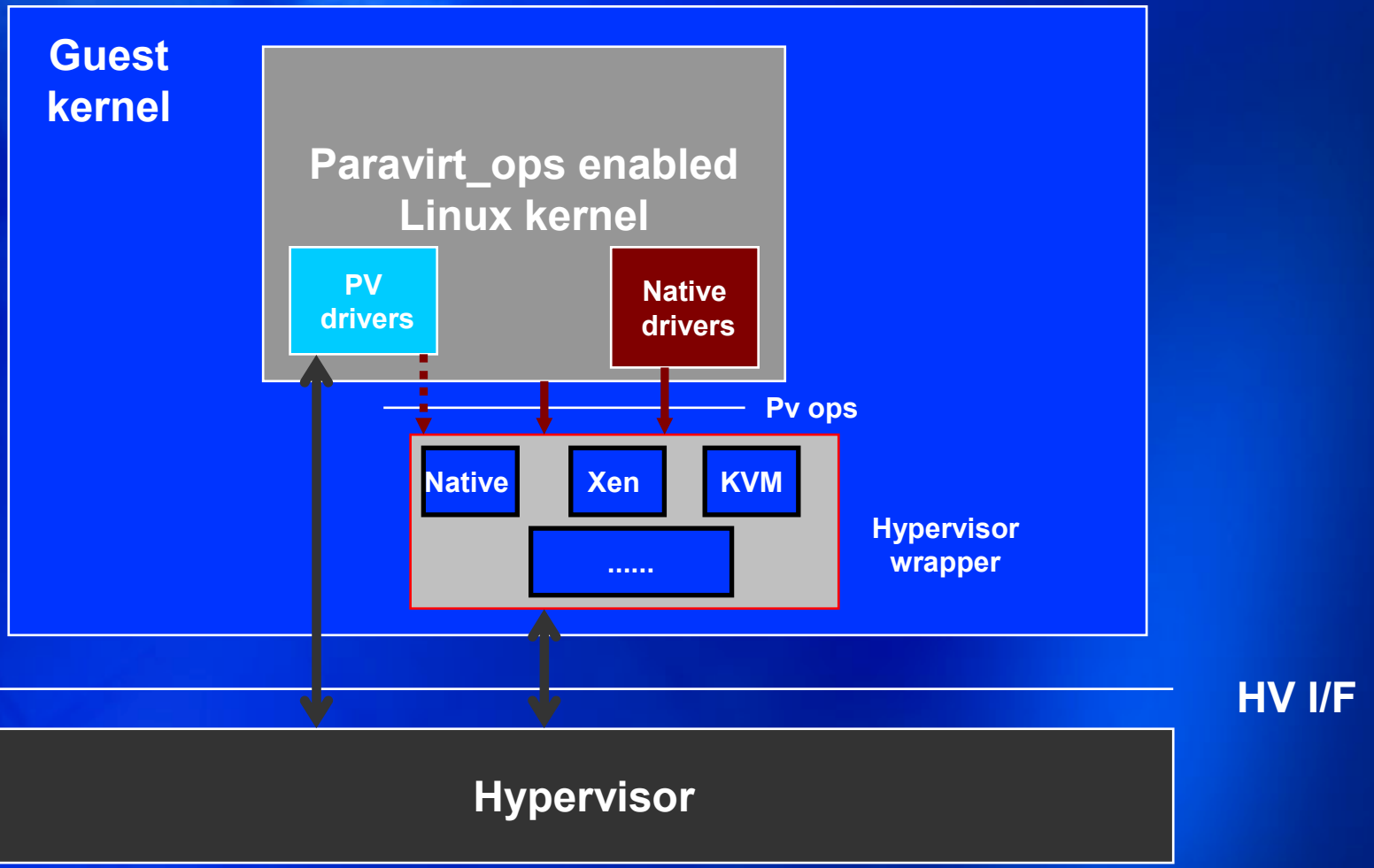
IPF paravirt_ops work effort estimation

Paravirt_ops framework



I/Fs

- Pv ops (paravirt_ops)
  - Wrapper to convert between pv ops & HV I/F
    - Native is abstracted too
  - PV drivers may talk to HV directly
- HV I/F
  - Hypercall / Event channel / Share memory etc.
- Hypervisor wrapper
  - Hook for each hypervisor or native

Pv_ops work

- Vanilla Linux code change to use pv_ops
  - Replace kernel sensitive instructions
    - Include MMU, timer, interrupt: All CRs
  - PMC/PMD
    - Could be later: Check if X86 did
  - Irqchip
    - Xen_irq_chip or vSAPIC
    - IOSAPIC -> paravirt_ops
      - Today kernel/iosapic.c already doing similar
  - Dma, i.e. p2m or swiotlb issue
  - Kexec/Kdump: could be later
- Xen wrapper for pv ops
  - keep current Xen HV I/F for now
    - No change to PV drivers

Xen machine vector

- Native machvec: sn2, hpsim, dig, hpzx1, hpzx1_swiotlb
- Xen machvec:
  - Dom0, or driver domain, need the physical machine dedicated code support
    - Unique virtual machine can't satisfy all different host platform
  - Pv_ops provide possibility for each host platform to run both natively and virtually
- Suggestion:
  - Adopt pv_ops in each machine vector eventually for upstream
  - Temporary leave in xen wrapper for now to minimize effort

IVT.s paravirtualization

- Performance critical
  - Same source, multiple compile, multiple IVT table
    - least impact to bare metal performance
  - Binary patching
    - Single IVT, but could leave more spare “nop” after patching
    - IVT entrance code size may be not enough
      - Need br.cond and thus additional Bx register save/restore

Staged pv_ops impl.

- Stage1: Replace sensitive instruction with pv_ops
  - Native wrapper, vITC
  - Result: works for native
- Stage2A: Basic Xen wrapper
  - Sal/PAL/init/domain builder/Xen entrance
  - vITC, vIOSAPIC, vSAPIC, vConsole
  - Result: pv_ops enabled domU works w/ ramdisk
- Stage2B: Extended Xen wrapper
  - vITC -> xen timer
  - Xen_irq_chip
    - PV FE driver works
  - Result: domU works w/ FE driver.
- Stage 2C: Advanced Xen wrapper
  - Kexec/Kdump support
  - Perfmon/xenoprof
  - mca

Staged pv_ops impl. - 2

- Stage 3A: dma
- Stage 3B: Integration
  - Integrate with PV drivers, tools, native drivers
  - Result: pv_ops enabled dom0 works
- Stage 4: new HV I/F with less hypervisor maintenance effort (post RHEL6)
  - Convert machine vector to pv_ops
  - Simplify hypervisor and merge more VTi code & pv code
    - Reduce maintenance effort
  - Investigate hybrid mode
    - Close user level sensitive instruction issue
      - THASH/TTAG
      - ITC read
    - Performance for Tukwila and forward processor
  - Run Xen on Xen

Pv_ops calling convention

- Static registers
  - For ASM code such as IVT.s where stacked registers are not available.
  - Br.cond only
- Stacked registers
  - C calling convention

Static registers pv_ops

- Clobber registers:
  - In: R8/R9, Out: R8 --- Current Xen
    - Need extra save/restore for R8/R9, R8/R24 pair ?
  - In: R24/R25, Out: R24 ???
    - Need HV I/F changes, could future work
  - Return address in B0 (rp)
  - PR : P12 / P13
    - IVT.S doesn't use it and always save/restore PR
- Complicate hand coding for IVT.s to get clobber registers
- Binary patching vs. branch call ?
  - X86 like style, maybe at some time later
    - If the IVT space is enough

Example static pv_ops

Static pv_ops priority - 1

- P0 ops : - must do, architecture hole
  - Thash/Ttag/fc
  - Cover
    - If vPSR.ic != PSR.ic, behavior differently
- P1 ops : should do, performance critical
  - Share memory ops
    - IFS, IFA, ISR, IIM, ITIR, IPSR, IIP, IHA
    - RSM PSR.i, RSM/SSM psr.ic
    - XSI_BANKNUM (XSI_BANK1_R16 & XSI_B1NAT)
      - For bsw.1 only now
      - Bsw.0 needs to preserve original bank0 registers too per SDM
        - Address for correctness
  - Rfi, rsm psr.dt, ssm psr.dt, ssm psr.i
  - From psr, from IVR, to EOI

Static pv_ops priority - 2

- P2 : do after P1, can trap & emulate
  - To/from RR, to/from TPR, to PSR
  - ITC/ptc, to/from ITM, set KR
- P3A (No need, trap & emulate)
  - ltr.i/itr.d/ptr.i/ptr.d
  - Others
- P3B (leave to future acceleration)
  - DBR/IBR, PKR, PMC/PMD, CPUID
  - To IVR, from EOI
  - TPA, TAK

Static pv_ops priority - 3

- IA32 related (P4)
  - To/from AR.24 (EFLAG)
  - ???

Binary patching for static pv_ops

Example patching static pv_ops

Stacked pv_ops

- C convention for parameter
  - Do later after P1/P2 static pv_ops
  - Can cover all, but could rely on trap & emulate too

Physical mode

- Hypervisor (Xen) use dedicated rid to emulate guest physical mode.
  - Machine PSR.dt/rt/it are always = 1.
- Dt/rt/it=0, pure guest physical
  - Initial code only, not performance critical
  - Could pv_ops, but no needs
- Dt/rt/it=1, pure guest virtual
  - Entered thru rfi, capture & emulated
  - Hypervisor rid = guest rid now
- Dt=0, rt/it=1, in some IVT code
  - Kernel code & RSE use other than rr0/rr4.
  - Xen shared memory in rr7
  - No extra effort
- Other combination: No support

Backups