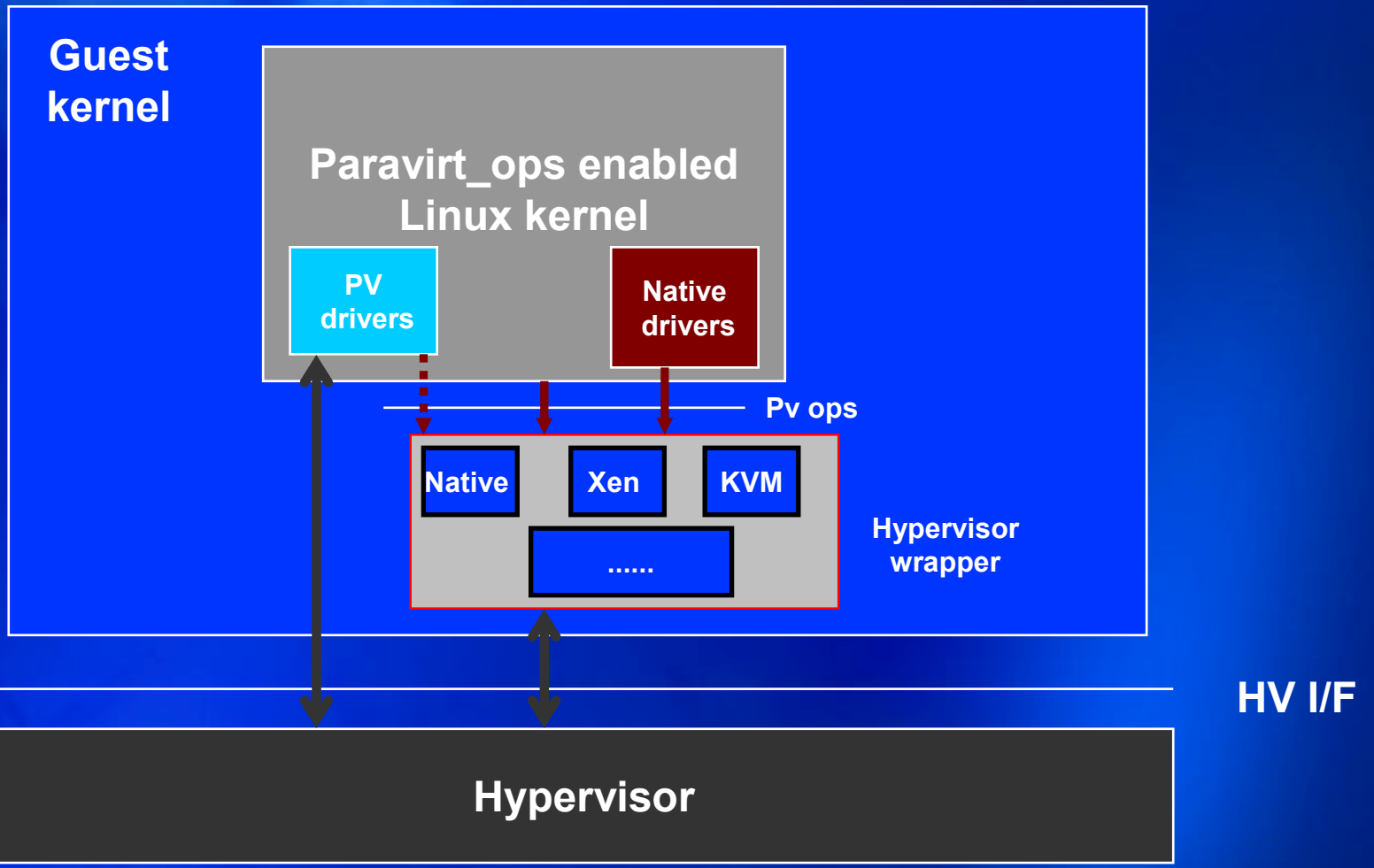
IPF paravirt_ops gap analysis -- preliminary

Paravirt_ops framework



I/Fs

- Pv 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.

Paravirt_ops work

- Vanilla Linux code change to use paravirt_ops
  - Replace kernel sensitive instructions
    - Include MMU, timer, interrupt: All CRs
  - Irqchip
    - Xen_irq_chip or
    - vSAPIC
    - IOSAPIC -> paravirt_ops
      - Today kernel/iosapic.c already doing similar
  - Dma, i.e. p2m or swiotlb issue
- Xen wrapper for pv ops
  - keep current Xen HV I/F for now
    - No change to PV drivers
    - Tools?

Alternative CPU virtualization

- Alt1: Pre-virtualization
  - Use script + compiler to replace sensitive instruction to hypercall
- Alt2: Unmodified Linux (as dom0)
  - VTi to capture sensitive instructions
- Alt3: Privify (vblade)
  - Tool to convert binary no trap sensitive instructions to undefined instruction to trap & emulate
- All above method are similar and performance difference is minor too
  - Difference only in instruction cost among break (hypercall), VT-I HW trap & undefined instruction fault

Issues to alternative CPU virtualization

- No intermediate layer of hypervisor wrapper
  - vSAPIC & Xen irqchip have to co-exist
    - PV drivers need xen irqchip
    - Vanilla Linux kernel need SAPIC
    - Extra effort
  - Need new HV I/Fs -> extra HV complexity
  - Performance concern
    - No shared memory -> excess hypercall
- Still need Linux change for DMA
  - Either push this part of paravirt_ops
  - Or works for VT-d only platform
    - Unacceptable

Staged CPU paravirt_ops implementation

- **Stage1: Replace sensitive instruction with pv ops**
  - Native wrapper
  - Result: works for native
- **Stage2: Xen wrapper**
  - No extra HV I/Fs
  - Result: paravirt_ops enabled domU works w/ ramdisk
- **Stage2B: irq_chip + PV FE drivers**
  - Xen_irq_chip
  - Result: domU works w/ FE driver.
- **Stage 3: Integration**
  - Pre-dependency: dma pv ops patch
  - Integrate with PV drivers, tools, native drivers
  - Result: Paravirt_ops enabled dom0 works

Staged CPU paravirt_ops implementation

- Stage 4: new HV I/F with less hypervisor maintenance effort (post RHEL6)
  - 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 ?
      - No trap for KR write
    - Performance for Tukwila and forward processor
  - Run Xen on Xen

Issues

- Tools: TBD
- Clobber register for paravirt_ops
  - ASM invoking of paravirt_ops
  - 2 sets of pv_ops like PAL call?
    - One for ASM code using static registers
      - Need art of re-structuring Linux ASM code
    - Another one for C code using stacked registers
- Physical mode
  - Dt, rt, it
  - No shared memory access?
- Xen_irq_chip