The hitch hiker's guide to Xen

山幡 為佐久
<yamahata@private.email.ne.jp>
<yamahata@valinux.co.jp>

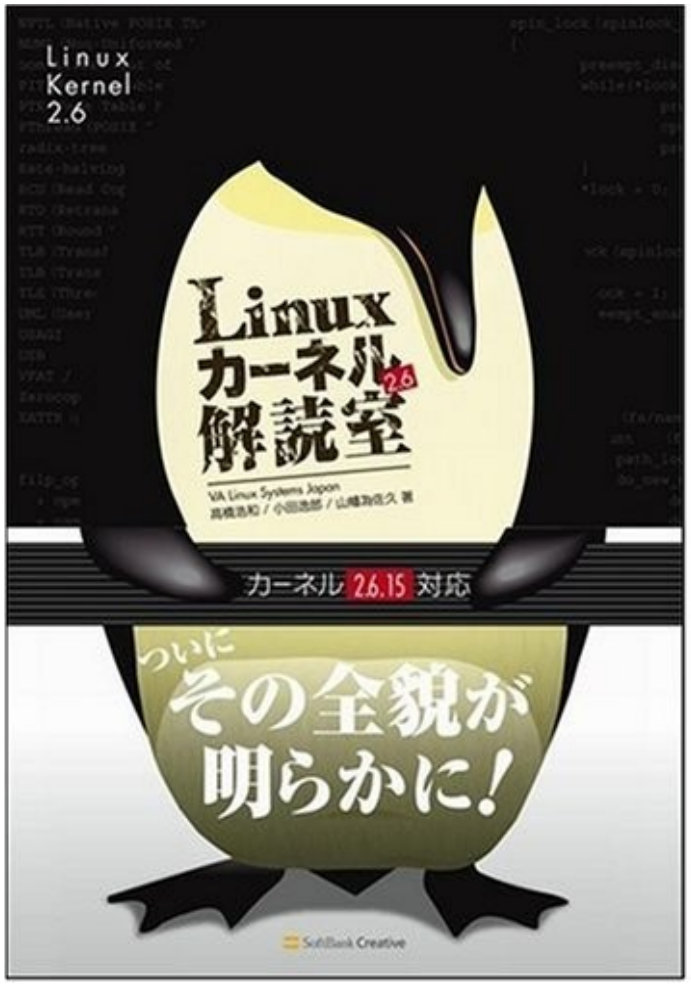
アジェンダ

- 自己紹介
- イントロダクション
- Xenの基本概念
- PV(paravirtualized) domain
- HVM(full virualized) domain
- paravirt_ops
- Xen/IA64
- 開発経験談

自己紹介

Linuxカーネル2.6解読室(共著)

高橋浩和，小田逸郎，山幡為佐久



Xenと私

2005年11月	Linux Kernel Conference 2005	「Xen 3.0のすべて: 内部実装詳解」
2005年12月	xen-ia64-devel初投稿	
2006年1月	Xen Summit初参加	第二回Austin
	2006年11月号-2007年1月号	Xen3.0解読室(第8,9,10回)
		OpenSourceMagazine休刊により打ち切り
2007年7月	Xen Conference 2007	「Xen/IA64の実装と最適化手法」
2008年5月	xen/ia64メンテナーに	
2008年6月	Xen Summit 2008 North America	「 Paravirt_Ops in Linux ia64」

- 2005年末頃から業務としてXenの開発を行なう
- 主にXen/IA64関係に関わる
  - 現在はlinux/ia64へのマージ作業が中心
    - paravirt_ops/ia64

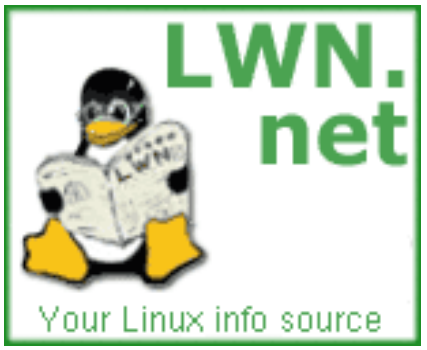
Xenへの貢献

Post-3.0.0 Rough Code Stats

	aliases	checkins	insertions
xensource.com	23	3000	550450
ibm.com	36	555	63967
intel.com	30	442	42437
valinux.co.jp	3	263	41299
novell.com	10	259	31794
hp.com	9	223	25102
fujitsu.com	20	290	13460
bull.net	1	157	14630
sgi.com	1	22	20569
redhat.com	12	179	10006
amd.com	7	85	6602
sun.com	7	80	8802
virtualiron.com	5	31	1586
ncsc.mil	3	26	6070
cam.ac.uk	3	23	3472
ubc.ca	1	14	493
unisys.com	4	8	872
other	29	302	55005

Stats since 3.0.0 Release

IA64 paravirt_ops/xen



The 2.6.28 merge window closes

[Kernel] Posted Oct 24, 2008 8:40 UTC (Fri) by corbet

Linus has released 2.6.28-rc1 and closed the merge window for this development cycle. Changes merged since the previous summary include IA64 Xen support, ultrawideband radio (UWB) support with wireless USB support on top of it, range timers, and some extensive changes to the block driver API. Also, this kernel has been named "Killer Bat of Doom."

Comments (22 posted)

イントロダクション
Xenの歴史

Xenって何ですか?

- Ian Prattらが始めたオープンソース仮想化技術
  - Xeno projectが母体
- Paravirtualization(準仮想化)を広めた
  - Guest OSに手を入れる
  - 仮想デバイス
  - 仮想化が難しいといわれるx86で良い性能を出した
  - Linux, FreeBSD, NetBSD, OpenSolaris, minix, plan9

Xenって何ですか?(cont.)

- hypervisorと管理OS(dom0)の両方が必要
  - hypervisorには最低限のデバイスドライバ(VGA, serial)しか持たず、デバイス(ディスク、NIC等)の制御は管理OSに任せる
  - 管理ツールは管理OSに存在
- 今では完全仮想化もサポート
  - 各種仮想化対応デバイスをサポート
    - VT-x, AMD-V, VT-d, AMD-IOMMU
    - 積極的に新フィーチャーに対応
  - X86以外のアーキテクチャもサポート

Xenの歩み

日時	出来事	
2002年位?	xeno project開始	
2003年1月	Xen2002(Technical Report)	
2003年7月	SourceFourgeにxen project作成	
2003年10月	Xen and the Art of Virtualization(SOSP)	これでparavirtualizationが有名に
2003年10月	Xen 1.0(stable) release	SourceFourge
2004年7月	Xen and the Art of Open Source Virtualzation	LinuxSymposium
2004年8月	Xen 2.0 beta	
2004年11月	Xen 2.0 official Release	SourceFourge
2005年1月?	XenSource社設立	
2005年7月	Xen 3.0 and the Art of Virtualization	LinuxSymposium
2005年11月	Xen 3.0	
2006年7月	The Ongoing Evolution of Xen	LinuxSymposium
	X86-64 XenLinux: Architecture, Implementation, and Optimizations	LinuxSymposium
2007年6月	LinuxSymposium XenBOFキャンセル	
2007年8月	Citrix買収発表	
2007年11月頃	Xen AB(Advisory Board)設立	
2007年12月	ホストがxen.orgに移行	

バージョン	リリース日
1	01 Oct 2003
1.1	28 Oct 2003
2.0.0	05 Nov 2004
2.0.1	17 Nov 2004
2.0.2	21 Dec 2004
2.0.3	11 Jan 2005
2.0.4	03 Feb 2005
2.0.5	10 Mar 2005
2.0.6	22 May 2005
3.0.0	05 Dec 2005
3.0.1	01 Feb 2006
3.0.2	14 Apr 2006
3.0.3	22 Oct 2006
3.0.4	28 Dec 2006
3.1.0	18 May 2007
3.1.1	11 Oct 2007
3.1.2	14 Nov 2007
3.2.0	16 Jan 2008
3.2.1	25 Apr 2008
3.2.2	17 Sep 2008
3.3.0	22 Aug 2008

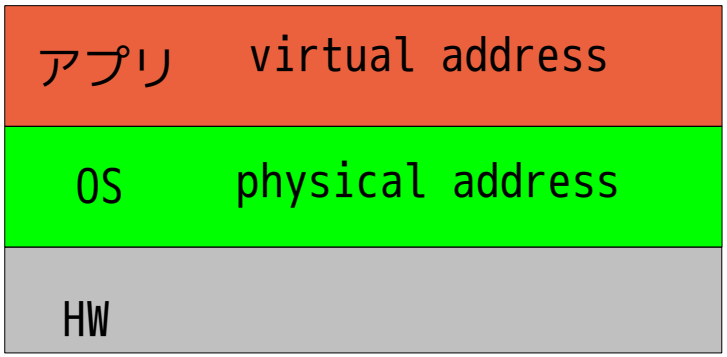
Xenの基本概念

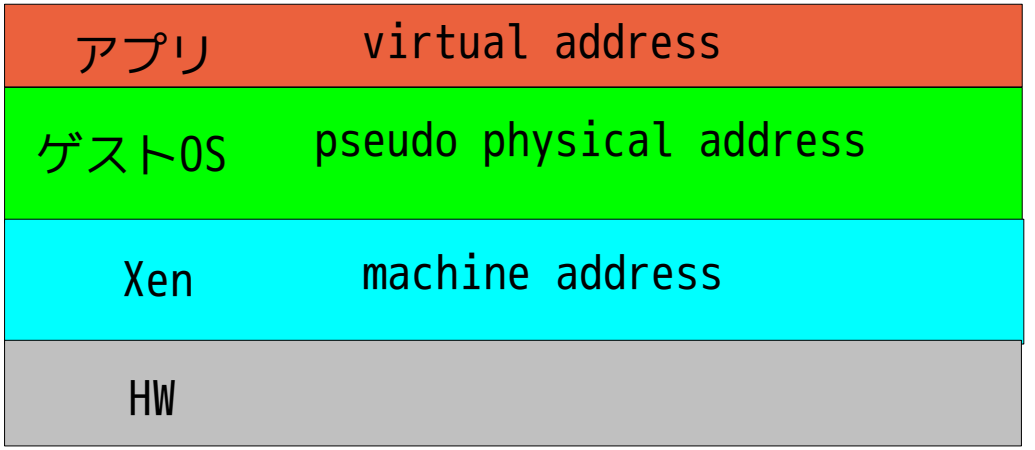
基本用語(ドメイン=ゲストVM)

- ドメイン
  - XenではVMをドメインと呼ぶ
    - 特権あり、特権なしのものがある
  - Domain0, dom0
    - 特権あり
      - ここからhost OSと呼ばれる事も
    - 起動時に必ず起動(domain id = 0になる)
    - 管理ツールが動作
    - 実デバイスの制御
  - DomainU, domU
    - 特権なし
    - ユーザが必要に応じて作成/破壊する
    - Xen仮想デバイスを使用

基本用語(アドレス)

- Machine physical address(machine address)
  - 物理マシンの物理アドレス
  - Host physical address
- Pseudo physical address
  - ゲストが物理アドレスと考えている仮想化された物理アドレス
  - Guest phsycial address
  - Metaphysical address(xen/ia64用語)





基本用語(アドレス)(cont.)

- M2P
  - machine => pseudo physical
  - この変換用の配列の事をM2P table/mptと呼ぶ
- P2M
  - Pseudo physical => machine
  - この変換用の配列の事をP2M table/pmtと呼ぶ

基本用語(アドレス)(cont.)

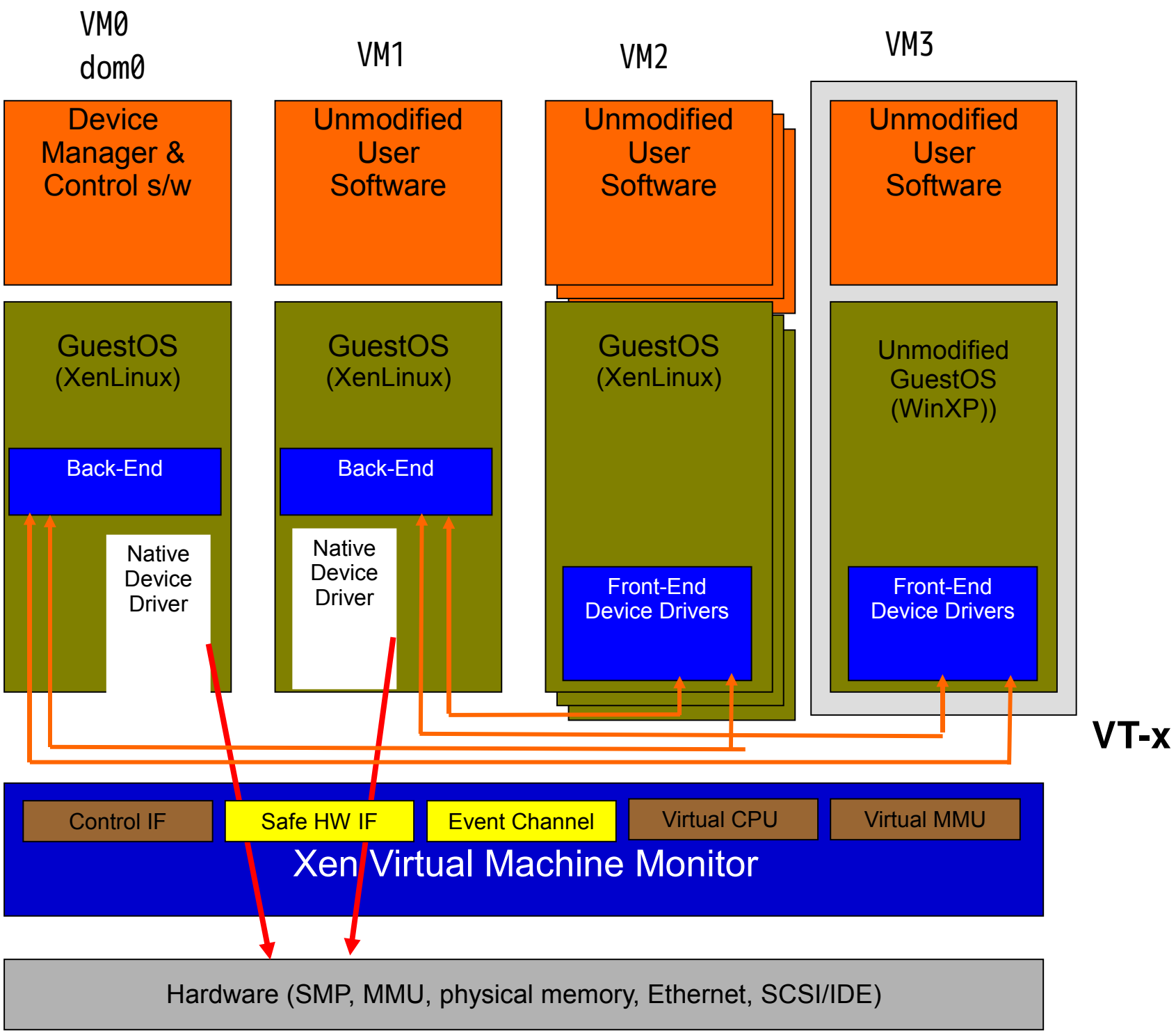
- コンベンション
  - xen/include/xen/mm.h参照
  - gpfn/gpaddr
    - Guest pseudo physical address
  - gmfn/gmaddr
    - Guest machine address
  - mfn/maddr
    - Real machine address
  - pfn/paddr
- コード内は割といい加減な気が、、、
- Shadow modeではまた違ったconventionが、、、

Xen paravirtualization

- CPU paravirtualization
  - De-privileged execution
    - Guest kernelを特権を下げて実行
  - 特権命令をハイパーコールで置き換え
    - カーネルソースコードの書き換え
    - 命令書き換えを避ける(VMWare特許)
    - Trap-and-emulateを避ける
    - インターフェースもh/wをそのままマップするよりも、都合のいいものを定義
      - 書き換え量が少なくなるように
      - 性能がでるように

Xen paravirtualization(cont.)

- MMU
  - direct-mode
  - shadow-mode
- IO paravirtualization
  - XenBus
  - 仮想デバイス(backtend/frontend device)
  - IO ring



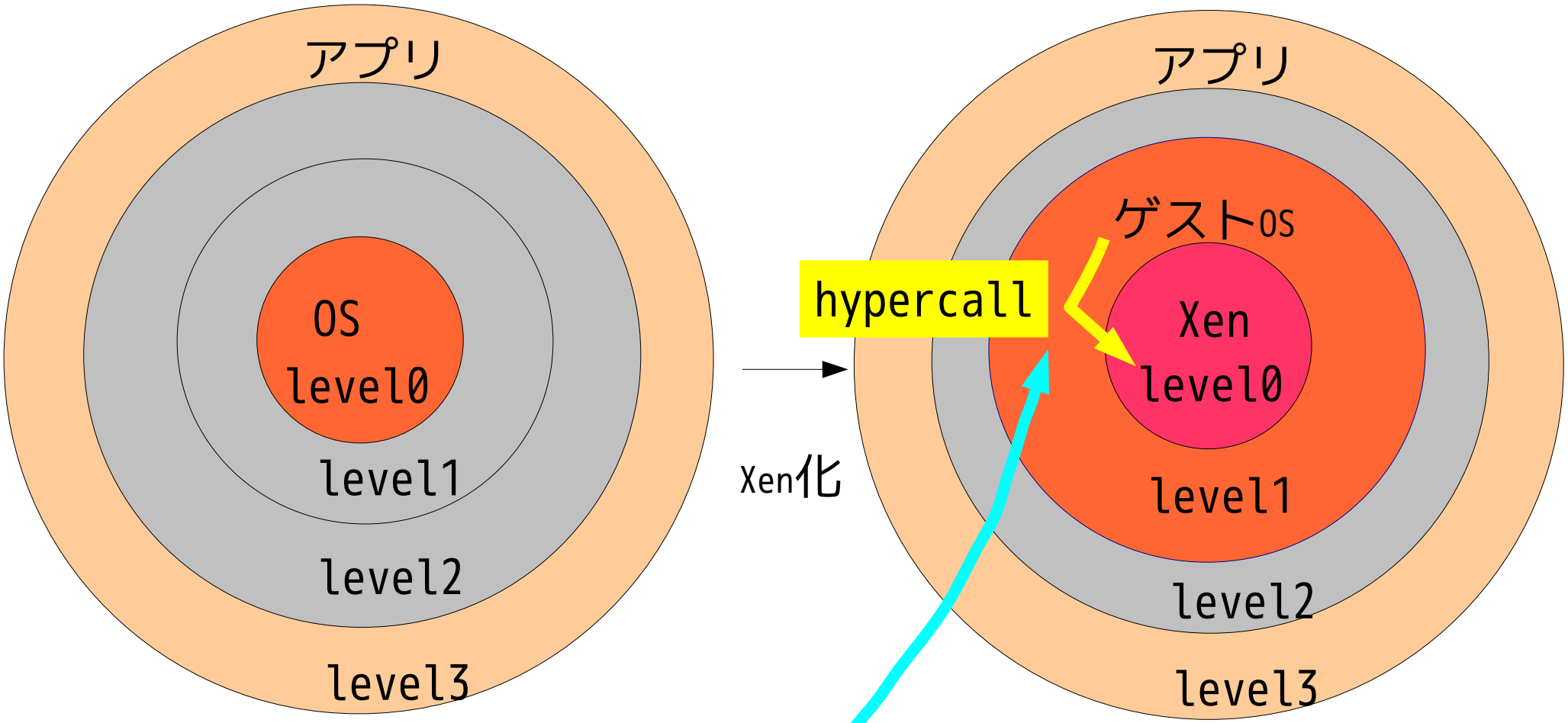
Cited from Xen Status Report: Ian Pratt at Xen Summit 2005

PV(paravirtualized) domain

CPU paravirtualization

De-privileged execution
32bit case

ゲストOSからXenを保護は
セグメント機能を使用



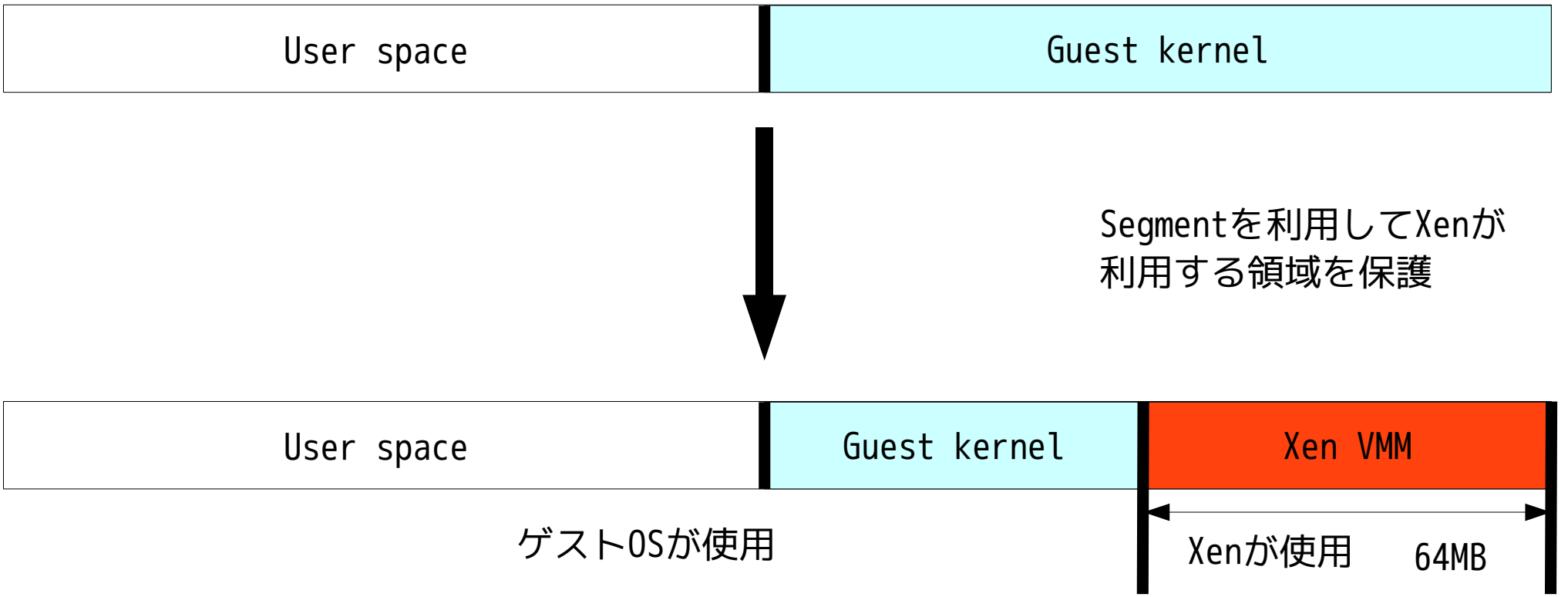
通常OSは特権レベル0で動作

Xen:特権レベル0
OS:特権レベル1
アプリ:特権レベル3

特権命令はhypercallで置き換える。

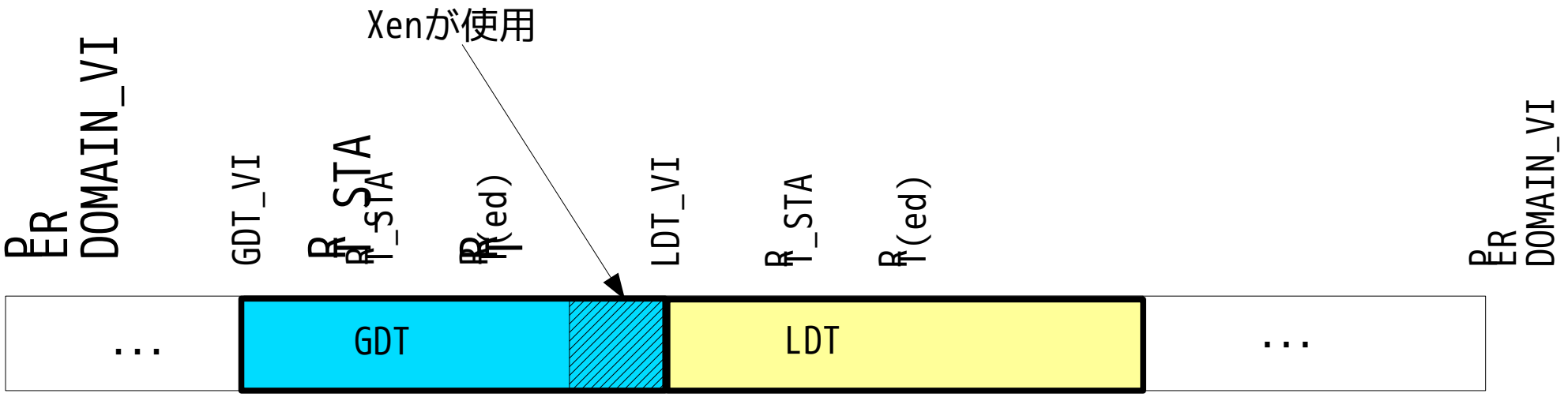
Address layout:32bit case

- Xenは仮想アドレスの高位64MBを使用する
  - ゲストOSは上位64MBを使用しないように修正される
  - Segmentを利用してXen VMMを保護



Segment

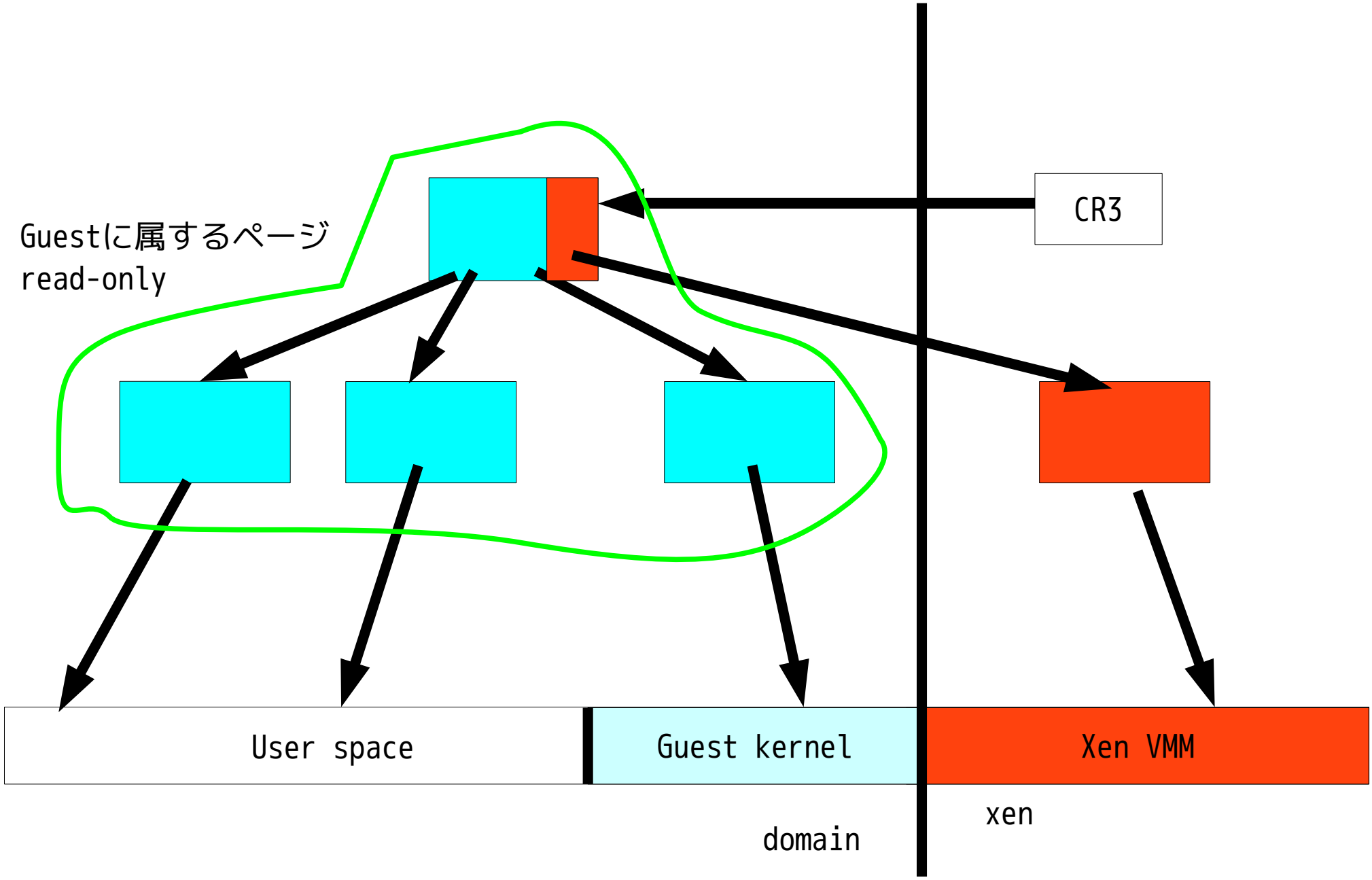
- GDTの後部をxenが使用
  - XenがGDT, LDT
- ゲストOSはhypercallを使用して書き換える
  - Xenが妥当性をcheck
  - Guestからは読み書き不可



Page table(direct-mode)

- Guest domain内のページを直接ページテーブルとして使用
  - エントリにはmachine addressが設定される
  - Guestがpseudo physical address=>machine addressの変換を行なう。
    - 最初にp2m変換テーブルがguestに与えられるので、guestが自前で追跡する
- Guest OSが管理するページテーブルはguestに対しread onlyでmapされる。
  - Xen VMMが妥当性をcheck
  - 一旦page table pageとして使用するとread-onlyに
  - hypercallを利用して変更

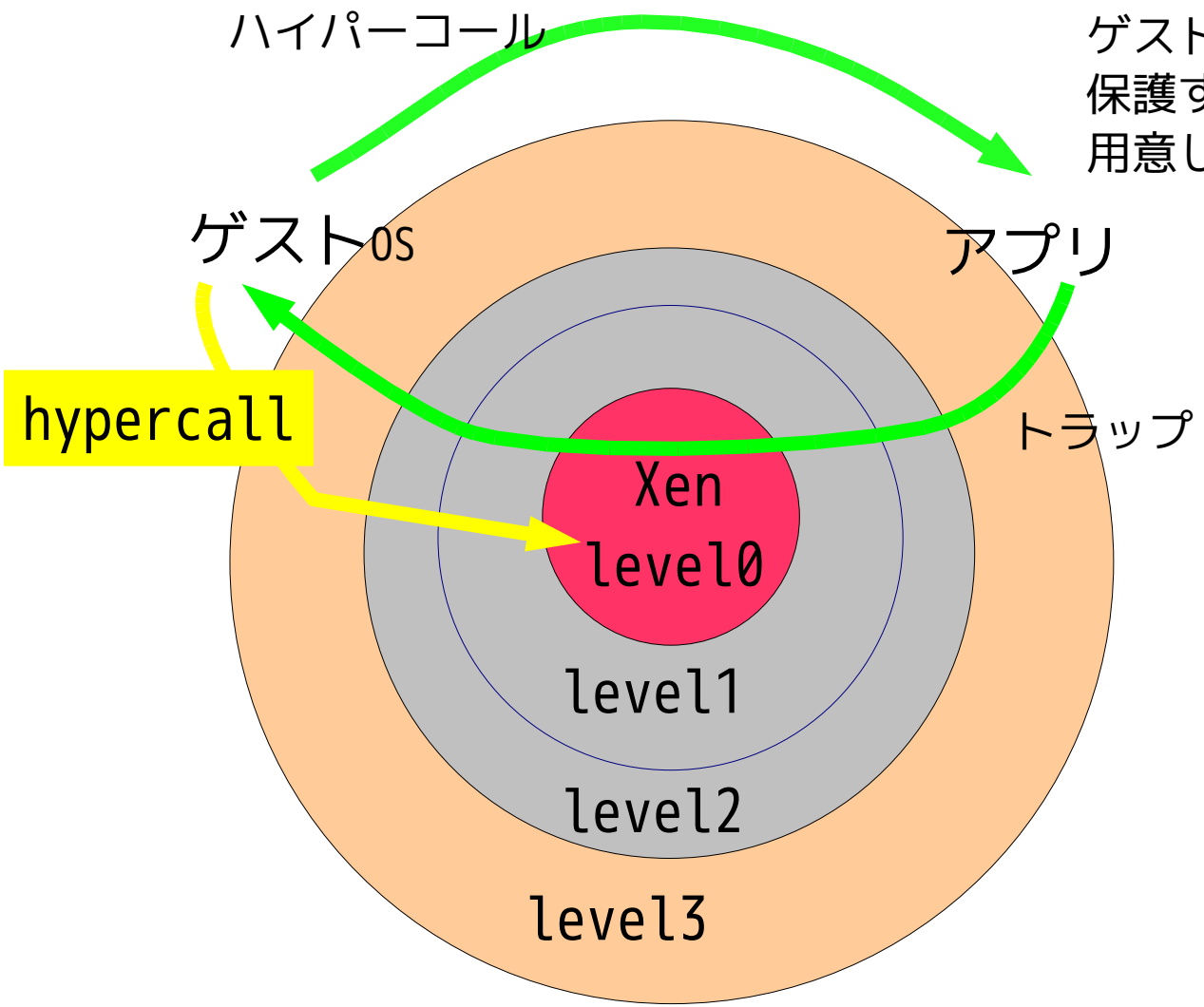
Direct Mode


Guestに属するページ
read-only
CR3
User space
Guest kernel
Xen VMM
domain
xen

64bit case

- Segmentが利用できない。
  - Xenが使用する領域はPage tableによる保護しかできない
- Guest kernelも特権レベル3で動作させるしかない。

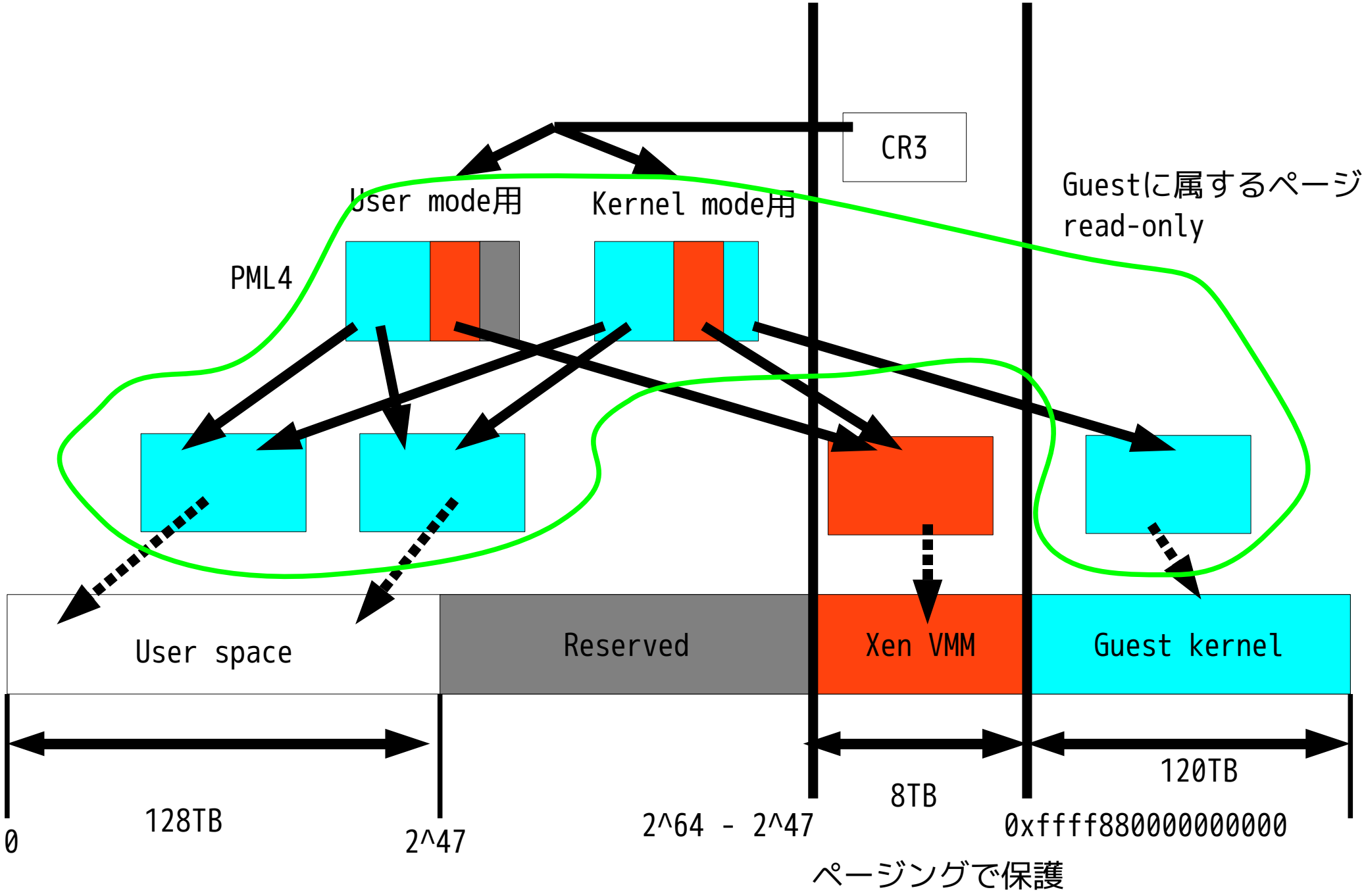
De-privileged execution
64bit case



ゲストカーネルをアプリケーションから
保護する為、二種類のページテーブルを
用意してハイパーコールで切替える。

Xen:特権レベル0
OS:特権レベル3
アプリ:特権レベル3

Address Layout:64bit case

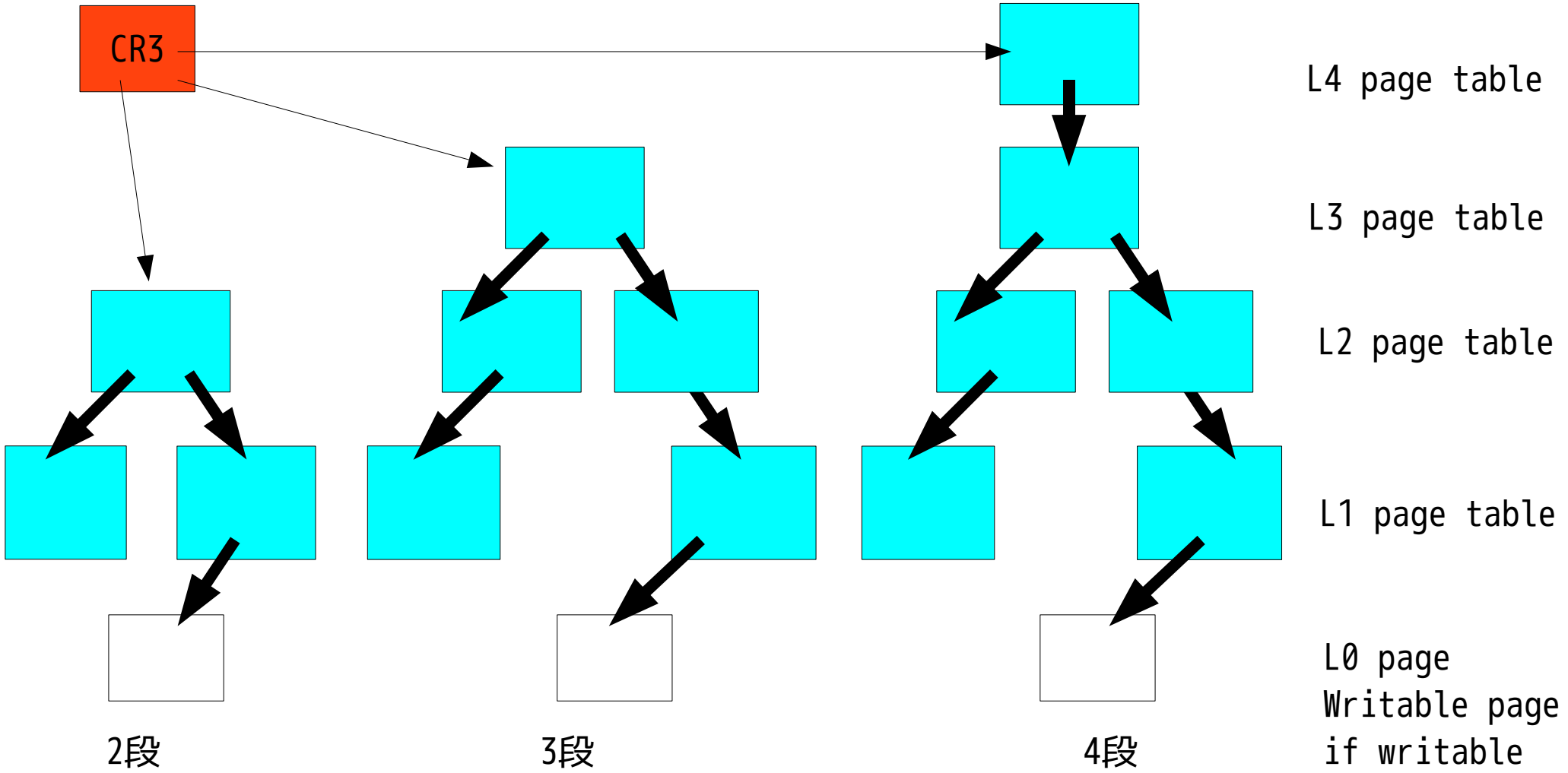


struct page_info

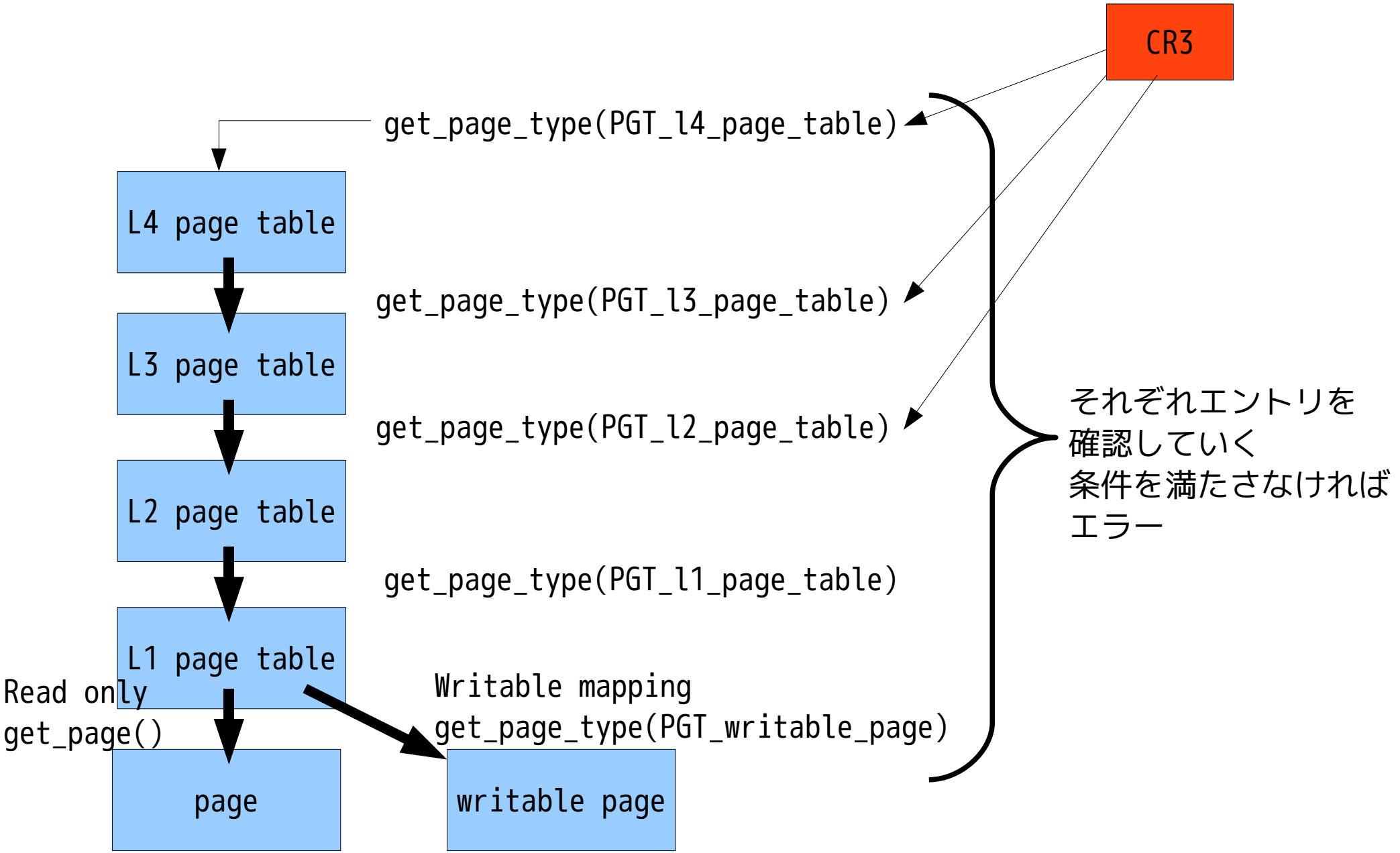
- 各pageを追跡
  - Owner: 属するドメイン
  - Reference count
  - Type count
    - 使用目的に応じて分類
    - タイプは排他的
    - L1, L2, L3, L4: ページテーブルページ
    - seg_desc:セグメントディスクリプタ
    - Writable:読み書き可能ページ
    - Read onlyでmapするにはタイプは不要。どのタイプのページもread only mapできる

Page type

- リーフに近い側からL0 page, L1, L2, L3, L4 page table
- ページテーブルの段数によらず統一的に扱う為



Type reference counting



Type reference counting(cont.)

- Linear page tableはどうする?
  - Linuxは使わないから考慮が後づけ
    - タイプが排他的
    - 同時に複数のタイプにはなれない
  - get_l{4,3,2}_linear_pagetable()
- guestがより複雑なmappingをしている事も考えられるが、実用上linear page tableの考慮のみで十分

Type reference counting(cont.)

- 当初は2 level page tableのみサポート
- 後に3 level/4 levelが追加された
  - 確認に時間がかかり過ぎる
  - Preemptionが追加された
    - get_page_type()/put_page_type()が中断/再開できる様に
    - コードは複雑になった

Writable page table

- 以前はOut of syncの一種が実装されていた
  1. L1 page tableへの書き込み時L2 page tableからのマップを解除
  2. guestに書き込みを許可させる
  3. Page faultの延長でL2 page tableからのmapを戻す
- 同一page table pageに対する大量の書き込みの性能向上が目的
  - fork/exec/exitに対する性能向上が目的
- 現在ではpteに対する書き込みをVMMでemulationするだけになっている

Event channel/Callback

- Event channel
  - interruptをeventとして抽象化
    - 外部interrupt
    - IPI
    - domain間通信
  - イベントが起きた時のentry pointを登録
- Callback
  - handlerの登録
  - Event channel, nmi, system call...

IO virtualization

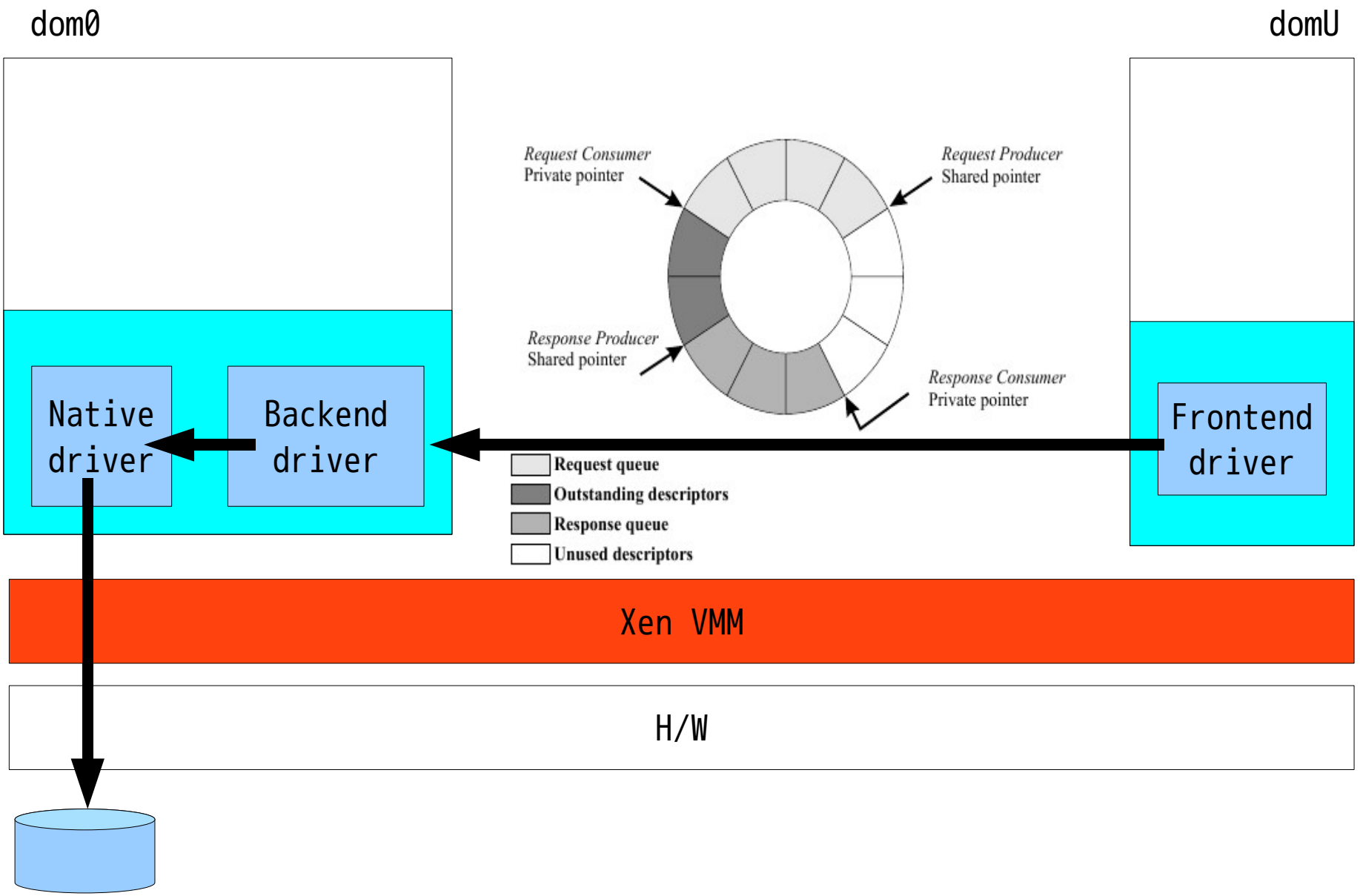
Virtual device

- Backend/frontend driver
- frontend(domU)の要求をbackend(dom0)に転送
  - I/O ringを通して要求
  - Event channelでnotification
- Dom0が実際の要求を処理
  - 必要ならdom0実デバイスにI/Oを行なう
  - 当初はbackendはXen内にあったがdevice driverがVMMからdom0に移されたのでdom0に
- Block, Netowrk, tpm, scsi, usb, pci

IO ring

- I/O要求と結果がlocklessで受け渡される
- Frontend:要求をI/O ringにqueue
- Backend:要求をI/O ringから取り出し処理
- backend:結果をI/O ringに格納
- Frontend: 結果をI/O ringから取り出し

Cited from Xen and the Art of Virtualization



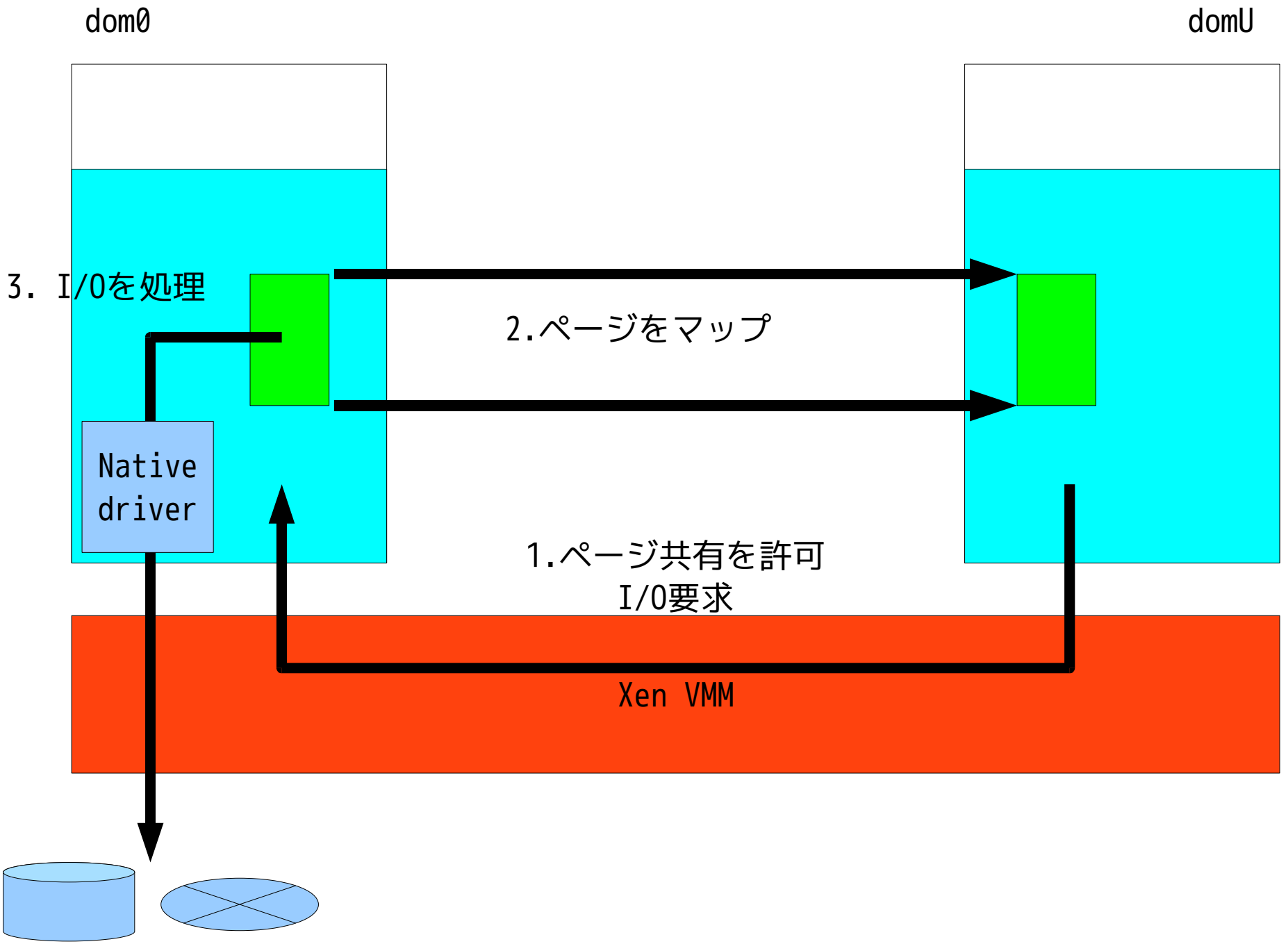
Grant Table

- Dom0がdomUに代わってI/Oする為に、domUのページを扱う必要がある
  - ページの読み書き
  - デバイスへのページに対するDMA要求
- ページ共有の仕組み

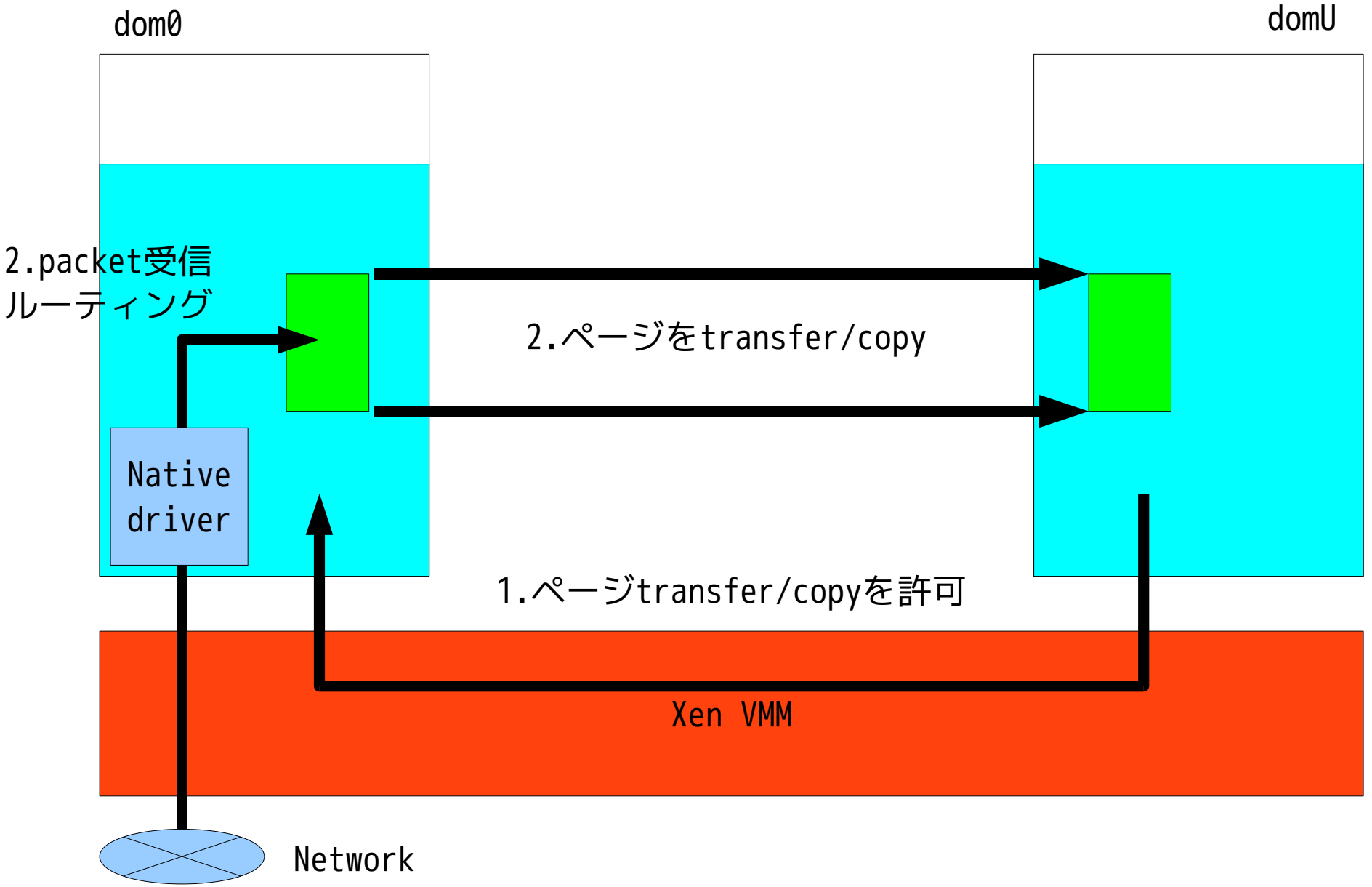
1. domU:dom0に対してページの読み書き許可
2. dom0:ドメイン内にページをマップ
3. dom0:I/O処理
4. dom0:ページアンマップ
5. domU:許可を終了

Grant Table(cont.)

- マップさせるだけでなくページ譲渡する事もできる
  - Page transferと呼ばれる
  - いわゆるpage flipping
- Page copyにて置き換えられた
  - 以前は使われていたが、現在は使われていない
  - 互換性維持の為、hypercallはサポートはされている


dom0
domU
3. I/Oを処理
2.ページをマップ
Native
driver
1.ページ共有を許可
I/O要求
Xen VMM

grant page transfer/grant copy



grant page transfer/grant copy

- 当初はpage transfer(page flipping)が使用されたいた
- Tlb shoot downが遅いのでhypervisor内でのmemory copyに変更された

blktap

- Block device backendの処理をuser landにもってきたもの
- 複雑な処理をする為にはuser landの方がやりやすい
  - File backed
  - COW
  - network越しにデータを送ったりなど

Timer

- 実デバイス(PIT, HPET等)を使わない。
- Periodic timerとone shoot timerを提供
- Guest OSは実デバイスでなくXenの提供するtimerを使用する様に書き換え

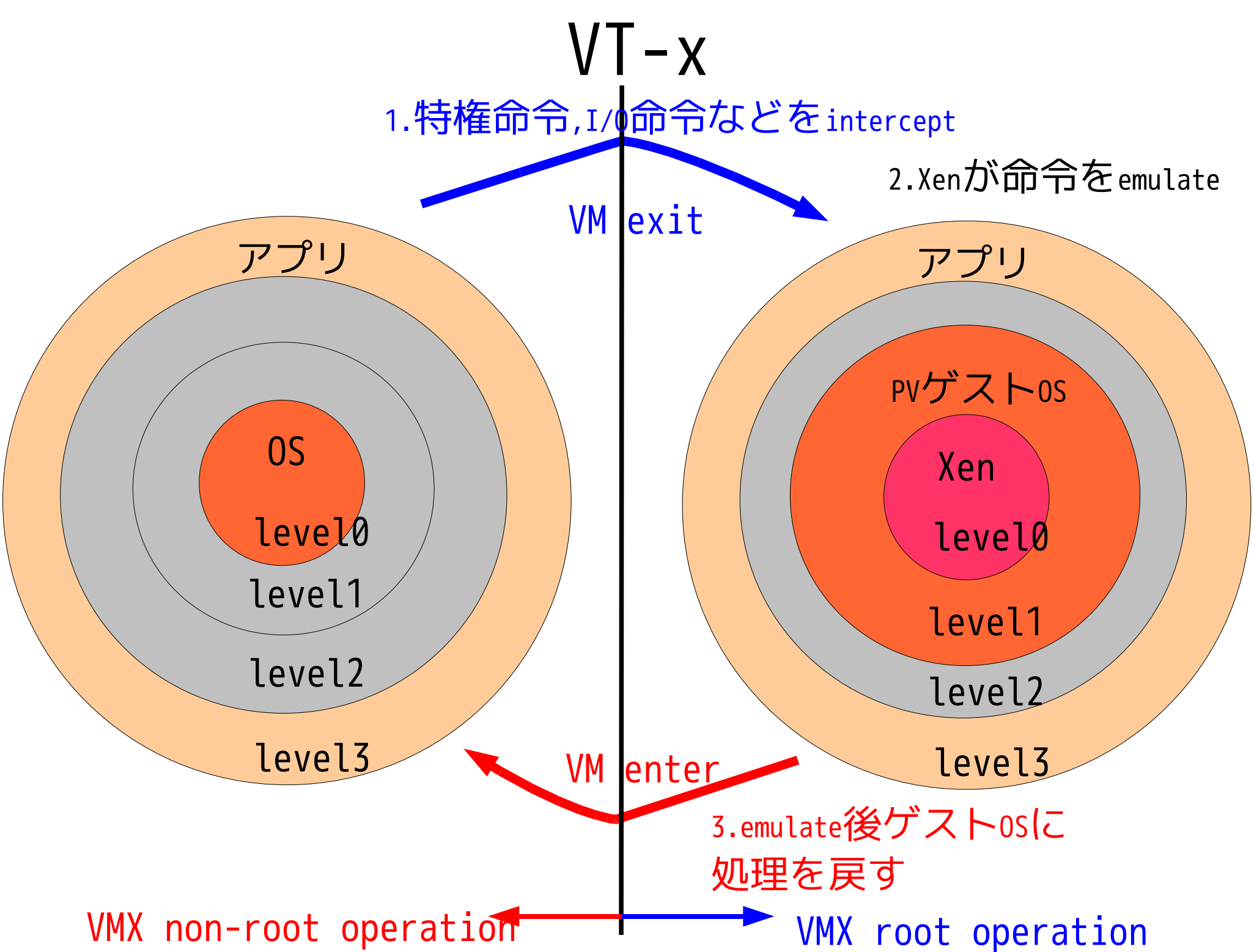
balloon

- カーネル内でpage allocateして、xenにページを返す
- OOMを引き起こし易くなる
  - Page hot plug/remove
- Vmwareとの競争上もっと高度な事も考えられている
  - 複数domainでのpage sharing
  - 同じ内容のpageの探索

HVM(full virtualized) domain

HVM(Hardware Virtual Machine)

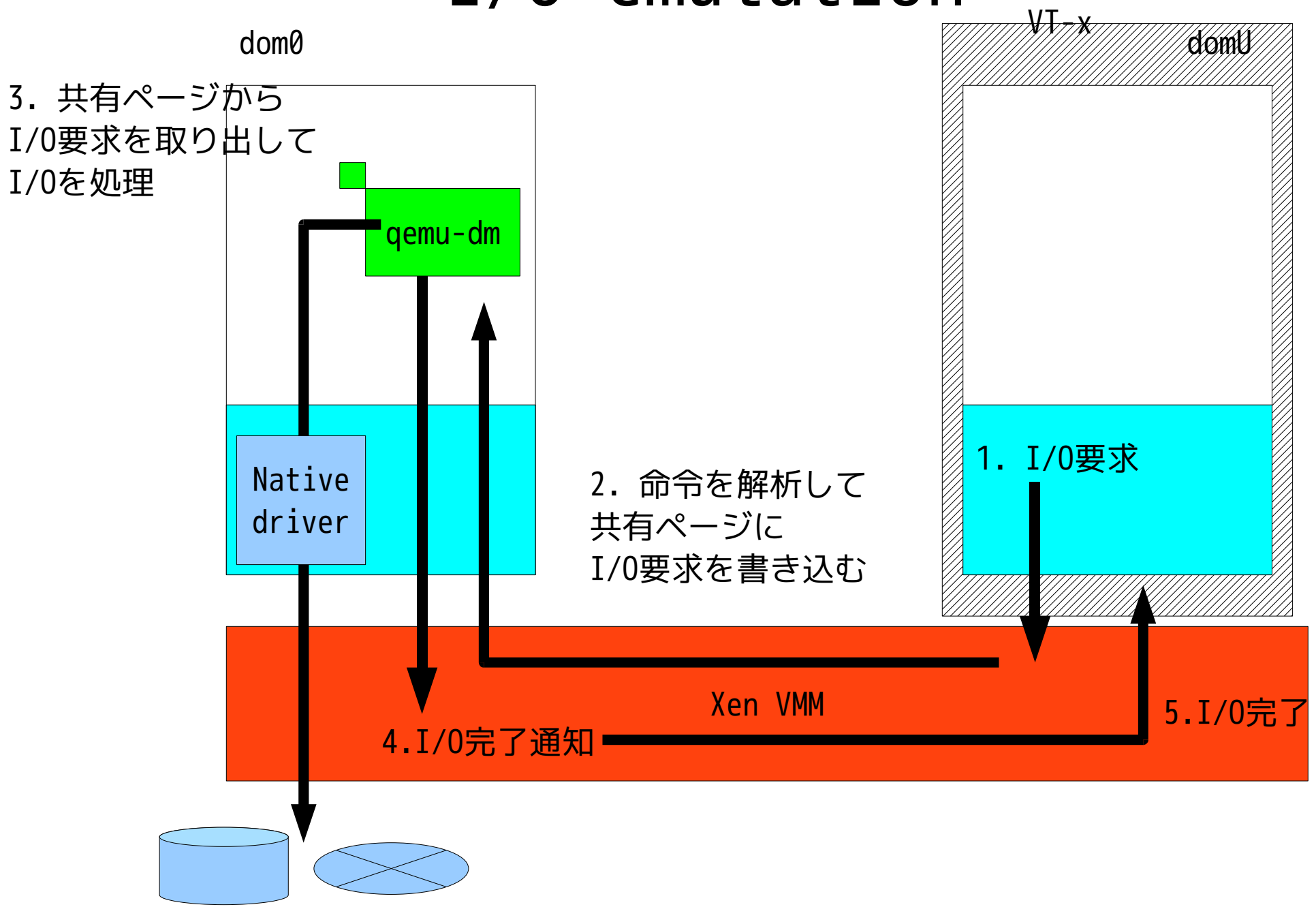
- hardware機能(Intel VT-x, AMD-V)を利用した完全仮想化ドメイン
- 当初は別コードだったが共通化するレイヤーが開発された
  - HVM abstraction layerはIBMが主体で開発


VT-x
1.特権命令,I/O命令などをintercept
VM exit
2.Xenが命令をemulate
アプリ
OS
level0
level1
level2
level3
アプリ
PVゲストOS
Xen
level0
level1
level2
level3
VM enter
3.emulate後ゲストOSに
処理を戻す
VMX non-root operation
VMX root operation

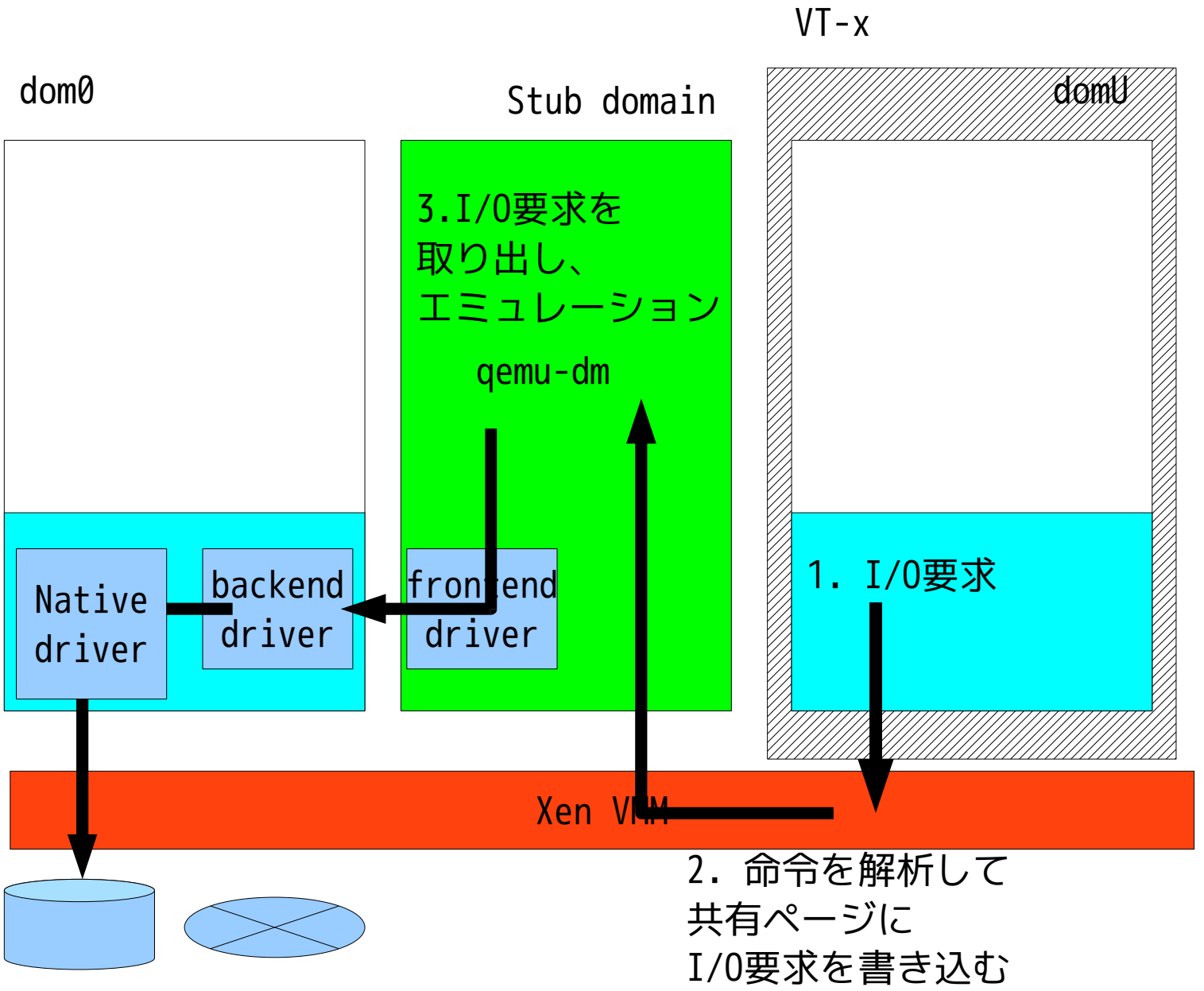
qemu-dm

- dm = device model
- デバイスエミュレーションにはqemuを流用
  - 性能の為いくつかのデバイスはVMM内でエミュレートされる
    - vioapic, vlapic, vpic, vpit
    - stdvga
- Guest firmware
  - bochs由来のfirmwareをベース

I/O emulation



Stub Domain



- Dom0内でプロセスとしてqemu-dmが動作してると
  - dom0のリソースを取り合う
  - アカウンティングが不正確に
  - レイテンシー大
- => 別ドメインとして独立させる

X86 instruction emulator

- 命令のdecode/emulationが必要な場所がいくつもある
  - IO、MMIOの命令decode
  - ページテーブル操作
    - pv writable page table
    - HVM shadow paging
  - Real mode emulation (Intel VT-x)
- 当初それぞれ必要な所で実装されコードが重複していたが、何度かの書き直しの末統合された

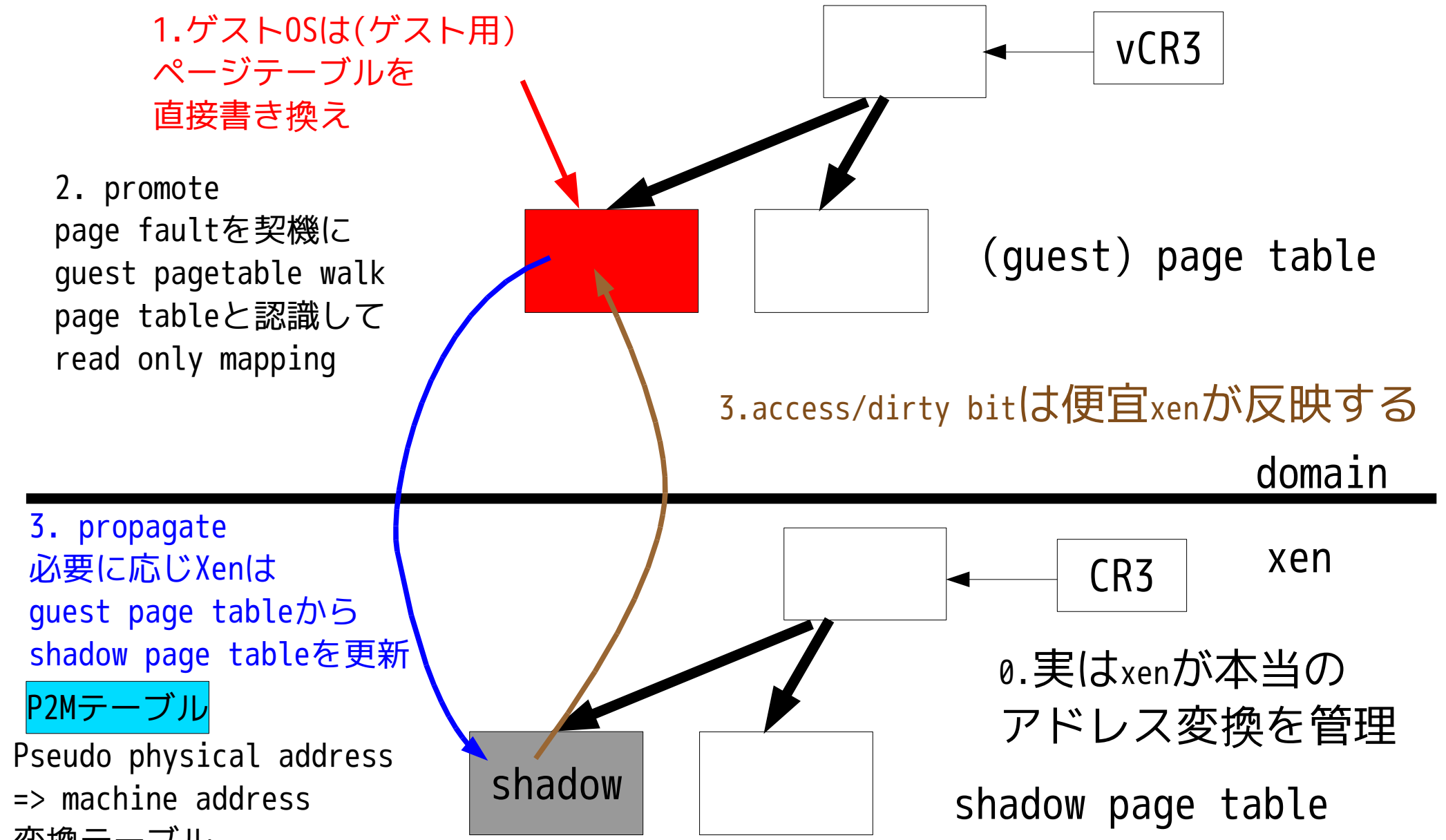
X86 instruction emulator(cont.)

- struct x86_emulate_ctxt
  - 実行コンテキスト
- struct x86_emulate_ops
  - コールバック
- x86_emulate()
  - 実行エンジン

Shadow paging

- MMUエミュレーション
- ゲストページテーブルをread onlyにするなどしてページテーブルに対する変更をトラップ
- シャドウページテーブルを必要に応じて更新
  - アドレスはP2Mテーブルを使用して変換
- ゲストページテーブルも更新の必要
  - Access bit, dirty bit

Shadow paging(cont.)



Xen Shadow Paging

- struct page_infoを乱用して情報管理
- Shadow pate tableに使用するメモリプリアロケート
- Reverse mappingなし
  - shadow/unshadow
    - 親page table entryはstruct page_infoに格納できる1つのみ追跡
  - promote/demote
    - promote
      - guest pageをpage tableとして管理read only mapping
    - demote:

Shadow pagingの歴史

- Shadow1
  - Ian Prattらによる最初の実装
  - Out Of Sync(OOS)も実装していた
  - windowsで性能出ず

<guest>-on-<VMM>
0: real mode
2: 32-bit page table
3: 32bit PAE page table
4: 64bit page table

- Shadow2
  - Shadow1の制限を取り除く為に再実装された
    - 0-on-2/3/4, 2-on-2/3/4, 3-on-3/4, 4-on-4
  - Correctness優先でOOSは実装されなかった
- Shadow3 = Shadow2 + OOS
- 他にもvirtual ironによるXIと呼ばれる実装があった:64bitのみ、reverse mappingあり

Heuristic

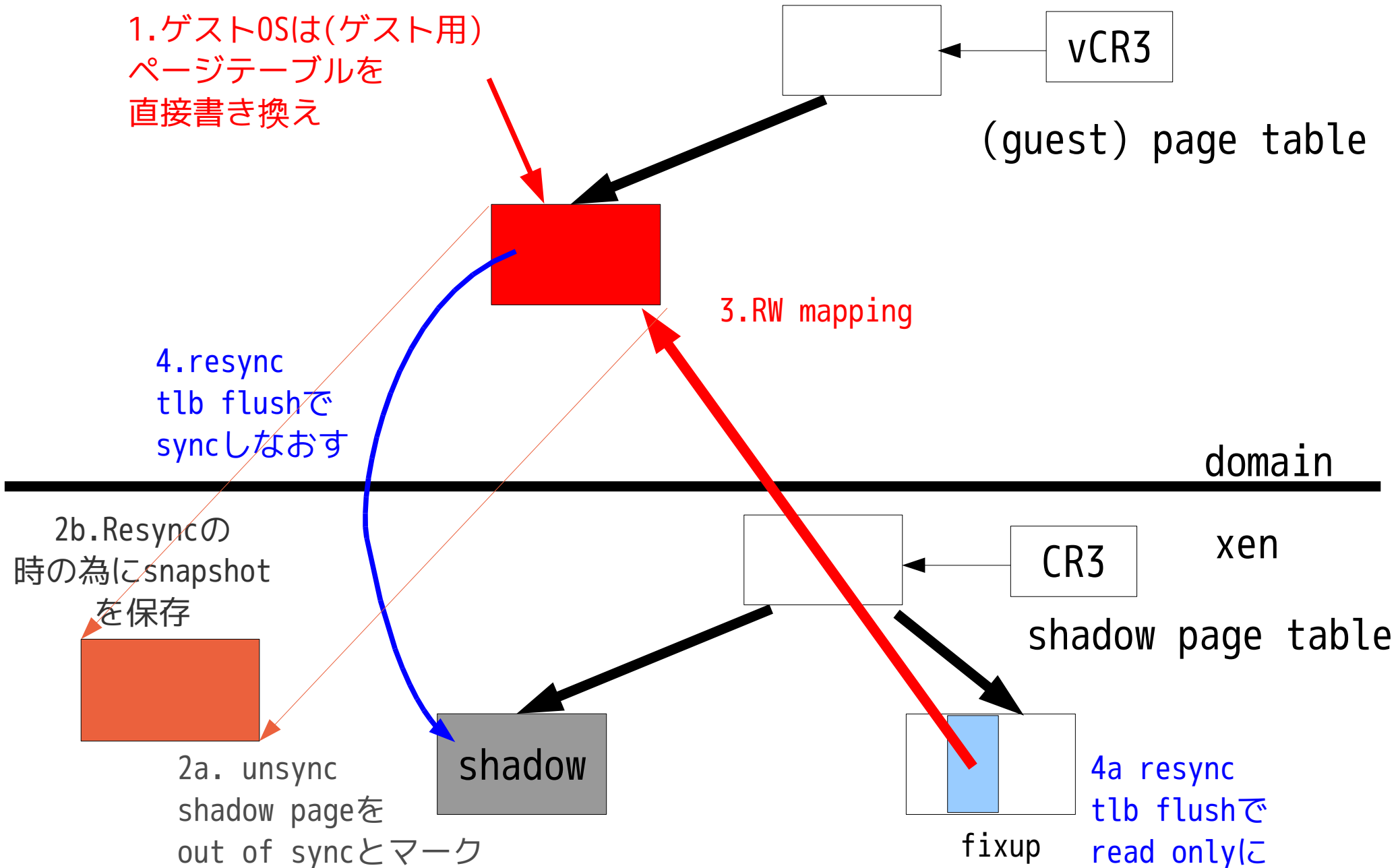
- Promote時にGuest pageをread onlyにする為に使用
- 謎アドレスが沢山
- Guest OSのアドレスレイアウトが変わると、追加の必要あり
  - 実際Linux 2.6.27で追加の必要があった

```
/* 64bit w2k3: linear map at 0xfffff68000000000 */
 switch ( level )
 {
 case 1: GUESS(0xfffff68000000000UL
               + ((fault_addr & VADDR_MASK) >> 9), 3); break;
 case 2: GUESS(0xfffff6fb40000000UL
               + ((fault_addr & VADDR_MASK) >> 18), 3); break;
 case 3: GUESS(0xfffff6fb7da00000UL
               + ((fault_addr & VADDR_MASK) >> 27), 3); break;
 }

 /* 64bit Linux direct map at 0xffff880000000000; older kernel
  * had it at 0xffff810000000000, and older kernels yet had it
  * at 0x0000010000000000UL */
 gfn = mfn_to_gfn(v->domain, gmfn);
 GUESS(0xffff880000000000UL + (gfn << PAGE_SHIFT), 4);
 GUESS(0xffff810000000000UL + (gfn << PAGE_SHIFT), 4);
 GUESS(0x0000010000000000UL + (gfn << PAGE_SHIFT), 4);

 /*
  * 64bit Solaris kernel page map at
  * kpm_vbase; 0xfffffe0000000000UL
  */
 GUESS(0xfffffe0000000000UL + (gfn << PAGE_SHIFT), 4);

  /* FreeBSD 64bit: linear map 0xffff800000000000 */
  switch ( level )
```

OOS(Out Of Sync)

- Guest page table pageとshadow page table pageを同期させない(out of sync)
  - Shadow3ではL1 page tableのみ
- L1 page tableへの書き込みを契機にout of sync状態へ
  - page fault
    - 必要なentryだけshadowへ反映
  - tlb flush(mov cr3, invlpg)
    - Resync: shadowへ変更を反映してout of syncを解消
  - Guestがtlb flushを発行しない場合、shadowのエントリが古いままで古いページを見る可能性がある
    - Native hardwareと同様

OOS(cont.)


1.ゲストOSは(ゲスト用)
ページテーブルを
直接書き換え
vCR3
(guest) page table
3.RW mapping
4.resync
tlb flushで
syncしなおす
domain
xen
2b.Resyncの
時の為にsnapshot
を保存
CR3
shadow page table
shadow
2a. unsync
shadow pageを
out of syncとマーク
fixup
4a resync
tlb flushで
read only(に

HAP(hardware assisted paging)

- Intel EPT(Extended Page Table)
- AMD NPT(Nested Page Table)
- Shadow pagingなどと共存する為、abstraction layerが追加された
  - struct paging_mode

Hyper-V(Viridian)

- Hyper-V対応hypercallを実装
  - VMMの実装を換骨奪胎
- Novelがパッチを出したが採用されず
  - Wrapperがいっぱい。
  - Coding styleがwindows風
- Citrix実装のものが採用されている
  - 今の所、最小限の実装のみ

paravirt_ops

paravirt_ops

- ソースレベルAPI
  - Linuxの標準スタイル
- 単一バイナリで複数hypervisorをサポート
  - Runtime switch
  - 含Native machine
    - VMM配下ではoverheadが問題にならなくてもnativeでは問題になりうる
  - ディストリビュータからの要求
    - 複数バイナリはテストが大変

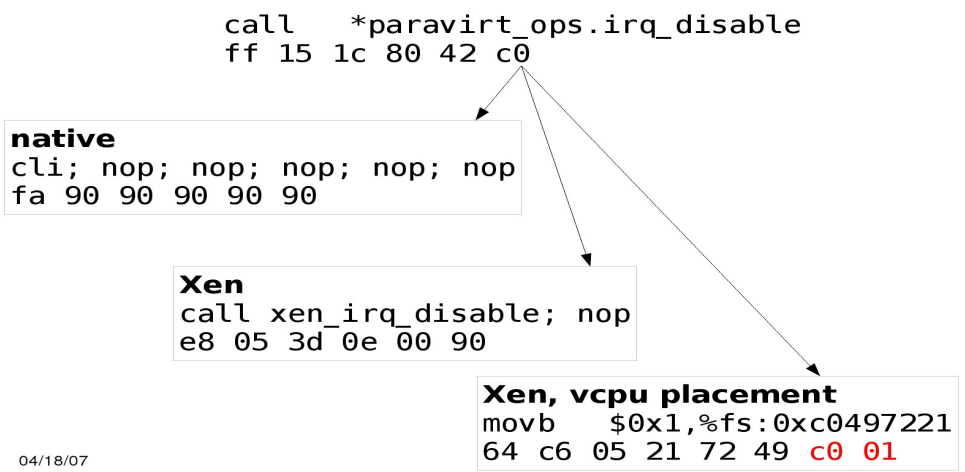
paravirt_ops history

	arch/xenアプローチ ｉｖｍｌ sub arch
2006年5月	VMI投稿
2006年7月	pv_ops提案(Kernel Summit)
2007年4月	pv_ops/VMIマージ
2007年6月	lguestマージ
2006年7月	最初のpv_ops/xen domUマージ

- どの様にvirtualization対応していくか合意をえる為に時間がかかった
- VMWareもVMIで必死
- その隙にKVMがマージされる
- paravirt_opsはRusty Russel(IBM)が提案

Binary patching

- 性能の為にバイナリパッチを活用
  - Native machine上でも性能低下を最小限に
  - Inline execution
  - Indirect call -> direct call
  - 何もしない場合はnopで埋める



実際は呼出元をannotateする為に
gcc inline assemblyで関数コールが
書いてある

Cited from
Xen/paravirt_ops upstreaming
Jeremy Fitzhardinge
Xen Summit April 2007

最近の開発動向

- IO関連
- Power management
  - Cpufreq
  - suspend/resume
- Direct divice attaching
  - VT-d, AMD IOMMU
- Network
  - Smart device

他architecture

- IA64
- POWER(hypervisor拡張が必要)
  - 最近リポジトリから消去された
- ARM(実装が複数ある)
  - SamSung
  - MontaVista
  - The University of British Columbia by Daniel Ferstay
  - The University of Illinois at Urbana-Champaign by Phillip Reames

Xen/IA64

architectureの違い

- Page tableが無い
  - tlb miss fault
  - Fault handler内でtlb insert
- PV domainでもMMUは完全仮想化されている
  - Virtual TLB
- DMAはparavirtualizeする必要あり

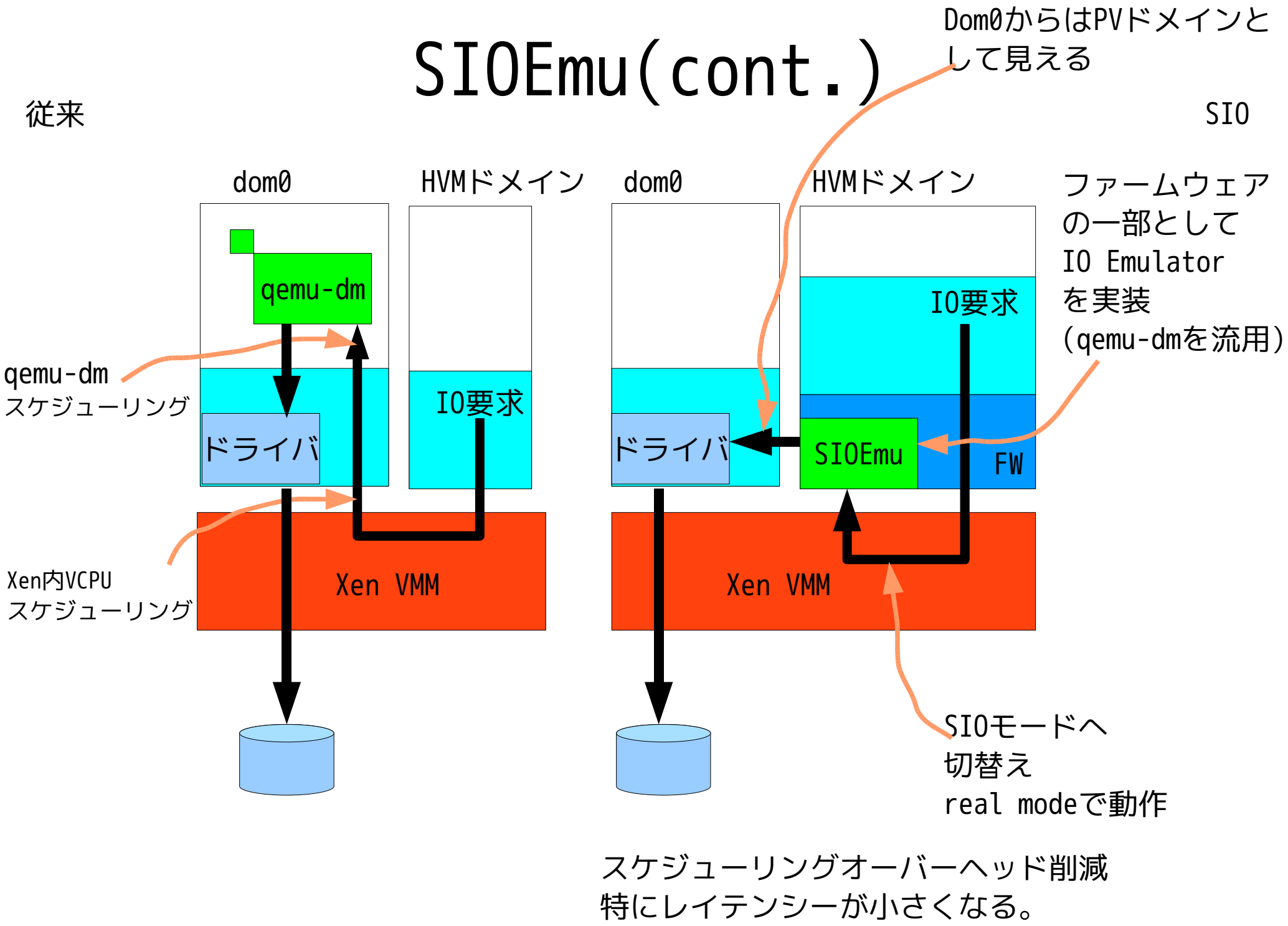
xencomm

- Guest OS <-> VMMのやりとりにpseudo physical addressを使用
- 始めはPOWERサポートの為に開発された
- IA64も同じ問題が発覚した為xencommを使用するように書き直された
  - 当初は仮想アドレスを使ってコピーしていた
  - linuxからコードを沢山借りてきていてcopy_from/to_user()がなんとなく動いていた
  - VMM内でTLB flushするhypercallが実装され出して問題が発覚

SIOEmu

- Self IO EMUlation
  - Qemu-dmを対象ドメイン内で動作させる仕組み
  - Tristan Gingoldが開発
  - IA64 MMUのモード切替えを生かしている。
  - X86で同様の事をやろうとするとMMUの切替のオーバーヘッドが大きいと思われる。
    - => stub domain

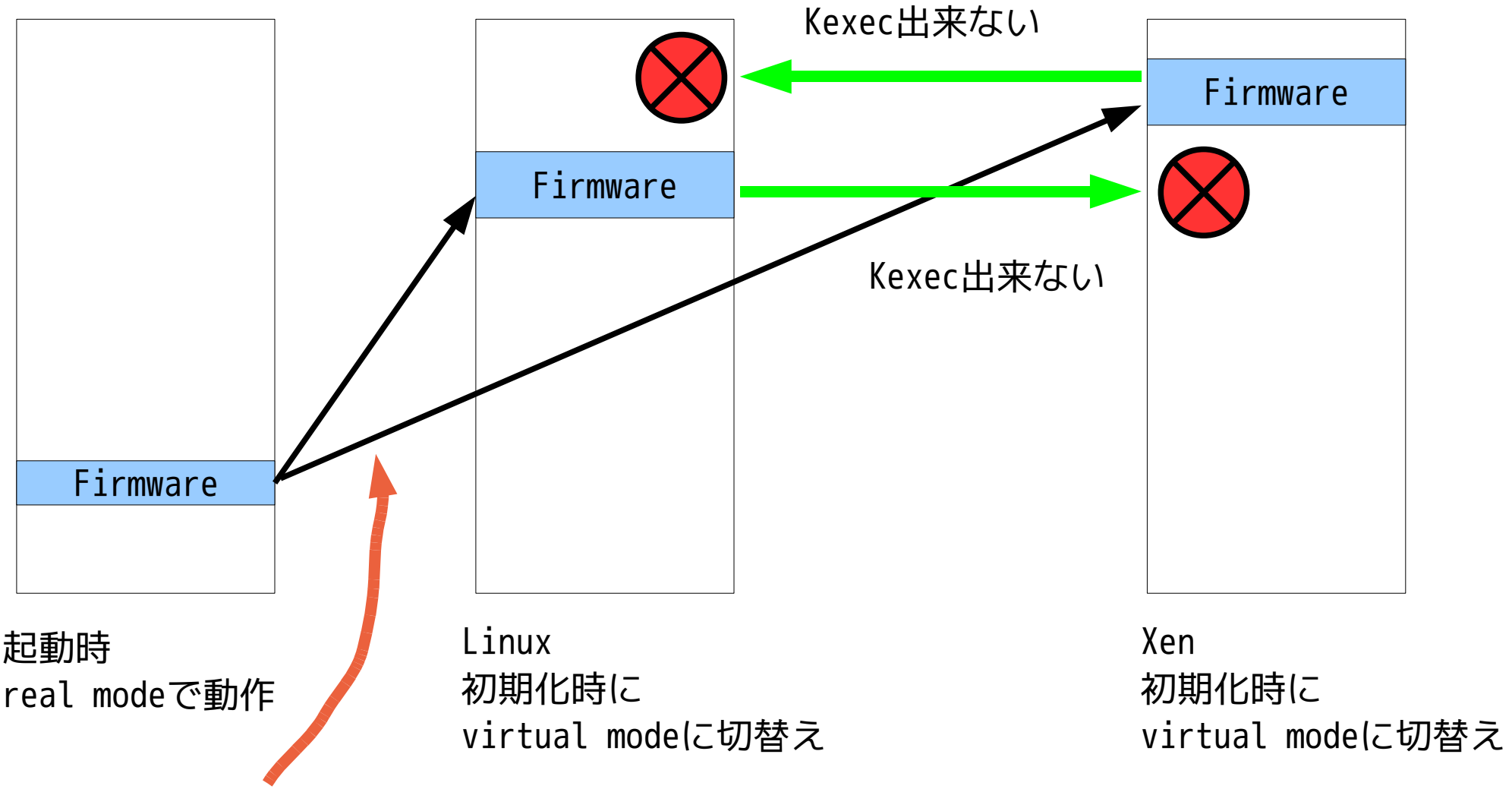
SIOEmu(cont.)


Dom0からはPVドメインとして見える
従来
SIO
dom0
HVMドメイン
dom0
HVMドメイン
ファームウェアの一部としてIO Emulatorを実装(qemu-dmを流用)
qemu-dm
qemu-dm
スケジューリング
IO要求
IO要求
ドライバ
ドライバ
SIOEmu
FW
Xen内VCPU
スケジューリング
Xen VMM
Xen VMM
SIOモードへ切替えreal modeで動作
スケジューリングオーバーヘッド削減
特にレイテンシーが小さくなる。

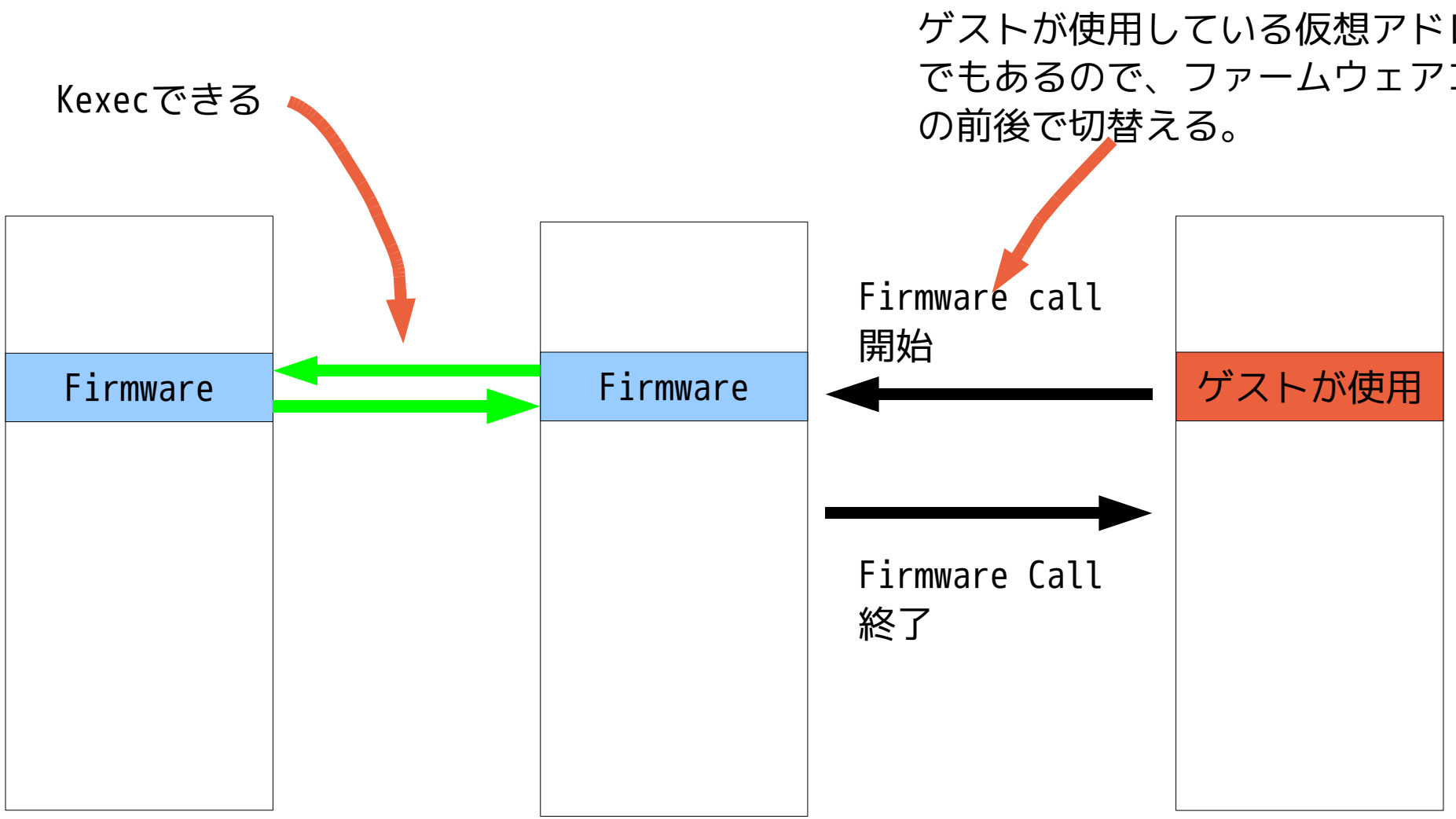
Kexec/Kdump

- EFI mapping
  - EFI
    - 起動時real modeで動作、一度だけvirtual modeに変更可能
    - LinuxとXenではマップするアドレスが違っているのでXen->Linux, Linux->Xenのkexecが出来ない。
    - Linuxと同じアドレスにマップ
    - ファームウェアコールの前後で空間切替え
  - X86でもUEFIをサポートすれば問題になるはず。

Kexec/Kdump(cont.)



Kexec/Kdump(cont.)



Linux
Firmwareは常に
マップされている

Xen Firmware実行時
Linuxが使用しているアドレスと
同じアドレスにマップ。

Xen
Firmwareはマップしない。
ゲストが使用している。

paravirt_ops/ia64

- 現在開発中
- pv_ops/x86とは実装は全く別もの

Binary Patch

- 特権命令準仮想化方式にparavirt_opsを使用する為必要となる。
  - 関数呼出オーバーヘッド削減
    - 特にnativeの場合の性能低下を防ぐ為に必要
    - nativeの場合は(ほぼ)1命令に対応
  - IA64の場合、gccインラインアセンブリでは関数呼出が記述できない
  - =>関数呼出オーバーヘッドが問題になる特権命令準仮想化(pv_cpu_ops)のみ対応させる。
  - =>呼び出される関数はアセンブリで記述

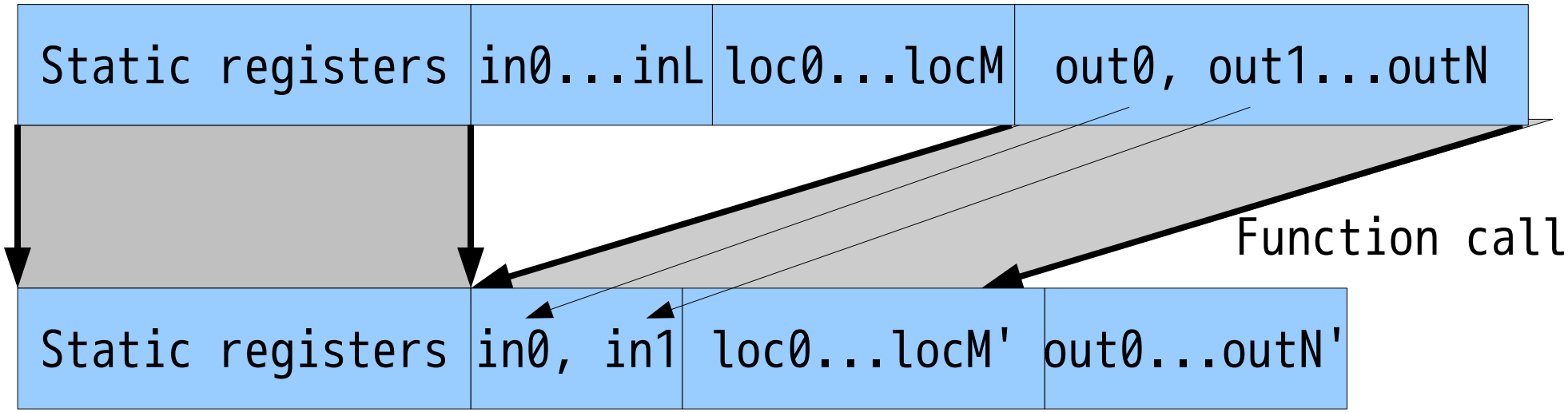
```
caller_func(in0, ... inL)
    loc0, ... locM
    out0, ... outN

    bp_func(out0, out1)
    other_func(out0, out1, ... outN)
```

asm(“...”::: “out0”, “out1”)では破壊されるレジスタが全て記述できない。

func(in0,..., inL)

r0-r31　　M, N are determined by GCC

Clobbered registers



r0-r31　　M', N' are determined by GCC

bp_func(in0, in1)

開発経験談

開発内容

- ダンプ機能の開発
  - kexec/kdump方式
    - 上流のとマージを考えるとkexec/kdump方式しかありえない
  - 解析ツール(crash utility)
- Xen/IA64の強化
  - 実用に耐えるものにする
    - ネットワークが使えなかった

開発環境

- Xen/IA64の担当になったけが、、、
  - 実機がなかなか来ない
- Ski simulator(IA64 simulator)が使えそうなのでcross compileして動かそうとした
  - Cross compleできなかった
    - それまでcrossで開発していた人はいなかったらしい
    - 皆さん贅沢
  - コードを見るとski simulatorに対応してそうなそ確だったけど、そのままでは動作せず
    - どうやら秘密(?)のsimulatorで開発していたらしい

Working with others

- Xen/IA64開発者はいろんなtime zoneの人がいた
  - US西海岸、東海岸、上海、フランス、日本
- コメントもらうのも一日待たないといけない
- 名前も通称の場合が
  - 西欧風の名前なので西海岸と思っていたら、何故か上海
  - Chineseの名前は発音できないので通称を使ってるそうだ

Debugging Xen/IA64

- 当時は何かあると無限ループでダンマリになるコードが多かった(それも割り込み禁止になってたりする)
  - 当然register dump/stack traceも出ない
  - 一体どうやってdebugしてるのか
    - simulatorかh/w debuggerだろう
- どうしようもないのでLinuxからもってくる事に
  - 今ではVMM debug用のgdbstubもありますけど誰も使ってない様子

Debugging Xen/IA64(cont.)

- Login promptがでればOkな事が多い
- Memory ordering大変
  - X86だと問題無い場合が
  - CPU core数が多くないと起きなかったり
  - 大体動いていて、たまにおかしくなる
  - Assemblyで書き直すと動かなくなる事も
    - Memory barrierを入れると直るけど、それまで発覚してない
    - gccの吐くコードによる
  - コードレビューで足りないmemory barrierを追加

Debugging Xen/IA64(cont.)

- xen/ia64 vtlbはlock lessなコードがあるがテストは怪しい
- OSはお行儀が良いので大丈夫
  - ダンプが走るとか特殊な場合に発覚
- VMMそのものに負荷をかけるのは難しい
  - User processによる負荷テストではOSに負荷はかかるけどVMMの負荷テストにはなりにくい

regression

- 人数が少ないのでenbugしても発覚するまで数カ月というのも珍しくなかった
- シリアルコンソール以外の機能は見過ごされがちだった
  - VGA、キーボード
  - X
  - RedHat installで発覚

Xen Merge Effort

- マージは大変
  - 共通部（≒xen/x86用)を修正するのは面倒
    - X86, ia64共に意味のあるコードにしないと入らない
    - 結局x86とia64の両方を解っていないといけない
  - kexec/kdumpはia64版が欲しかったのですが、まずはx86版を作る事に
  - Xen/IA64は開発者が少ないので、それほどでもない
    - 単にpatchの要求品質を低くせざる得ないだけでですが

kexec/kdump Merge Effort

- あまり関心を引かないパッチを入れるのは大変
- kexec/kdumpは反応を得るのが大変だった
  - 毎週パッチを投稿
    - コメントもらえず
    - ディストリビューターとか関心は高い筈なのですが実際に応援してくれない
  - Xen summit 2006 Sepで発表
    - その時の発表されたroad mapにも入っていたけどマージされたのは随分後
  - kexec-toolsのメンテナーも引き継いだ

paravirt_ops/ia64 merge

- これも関心低い
  - 最初のcommitが大変
  - なんとか押し込んでいく必要がある

paravirt_ops/ia64 merge(cont.)

- 都合最初のマージまで半年以上
  - 途中２～3カ月ぐらいの放置含む
- Trivialなパッチも放置された
  - メンテナにもよるが
  - 粘り強くお願いするしかない

trivialパッチの例

```
--- a/drivers/xen/cpu_hotplug.c
+++ b/drivers/xen/cpu_hotplug.c
@@ -2,7 +2,7 @@

 #include <xen/xenbus.h>

-#include <asm-x86/xen/hypervisor.h>
+#include <asm/xen/hypervisor.h>
 #include <asm/cpu.h>

 static void enable_hotplug_cpu(int cpu)
```

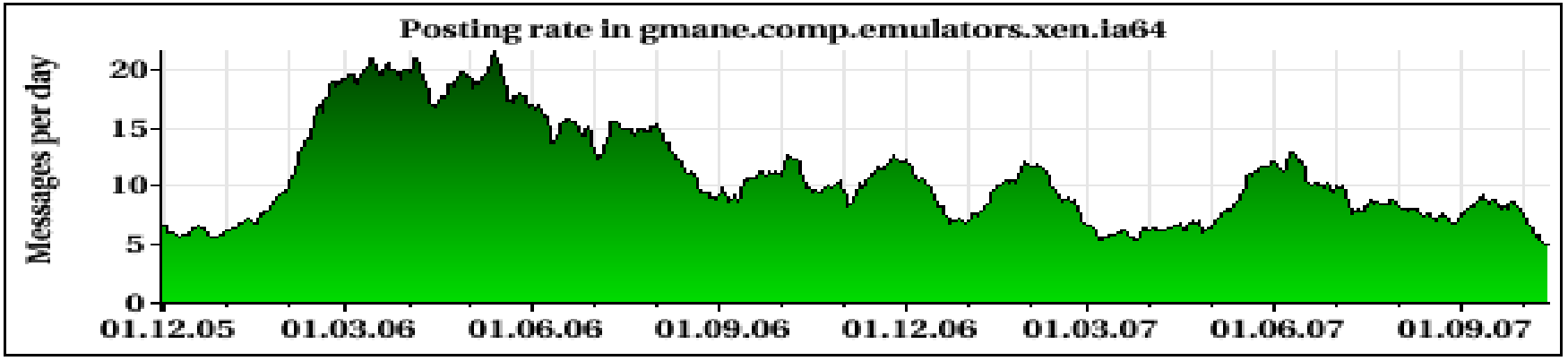
失敗例

- レビューし易い用にパッチを分割したけど、
- さすがに誰もレビューしてくれなかった

```
[PATCH 00/50] ia64/xen take 3: ia64/
 ├──>[PATCH 01/50] xen: add missing
 ├──>[PATCH 02/50] xen: add missing
 ├──>[PATCH 03/50] xen: add missing
 ├──>[PATCH 04/50] xen: add missing
 ├──>[PATCH 05/50] xen: move feature
 ├──>[PATCH 06/50] xen: move arch/x8
 ├──>[PATCH 07/50] xen: make include
 ├──>[PATCH 08/50] xen: replace call
 ├──>[PATCH 09/50] xen: make grant t
 ├──>[PATCH 10/50] xen: import inclu
 ├──>[PATCH 11/50] xen: import arch
 ├──>[PATCH 12/50] ia64/pv_ops: intr
 ├──>[PATCH 13/50] ia64/pv_ops: intr
 ├──>[PATCH 14/50] ia64/pv_ops: spli
 ├──>[PATCH 15/50] ia64/pv_ops: prep
 ├──>[PATCH 16/50] ia64/pv_ops: hook
 ├──>[PATCH 17/50] ia64/pv_ops: intr
...
```

Xen/ia64メンテナー

- Alex Williamson(HP)より引き継いだ
  - 逃げ遅れたとも
  - バグでても直す人が他にいないと直すはめに
- 開発は一段落していたのでパッチ投稿は少なかった
  - paravirt_ops/ia64と並行作業



Xen/ia64メンテナー(cont.)

- パッチのレビューは時間がかかる
  - 見たくないパッチも全部見ることになる
    - あんまりなパッチは放置したいけど
    - Intelの人とかは相手せざる得ない
  - Alex Williamsonはすごく大変だった筈
    - 当時はあまり気にしてなかったが、メンテナーになって解った
- パッチの品質は書いた人を見れば大体想像がつく

宣伝

Xen Conference

- http://www.valinux.co.jp/xen2008va/
- 2008年11月19日 （水） 13:30～17:30
  - Ian Pratt “Xen Open Source Project Overview”
  - Andrew Warfield “Services in the Virtualization Plane”

Xen Summit Asisa 2008(Tokyo)

- http://www.xen.org/community/xensummit.htm
- http://blog.xen.org/index.php/2008/07/16/xen-summit-tokyo-asia-call-for-speakers/
- November 20-21, 2008
- Fujitsu's Makuhari Lab located in Makuhari, Japan.

Xen Summit™

東京

November 20-21, 2008

Sponsored by FUJITSU

質疑応答

42

ご静聴ありがとうございました。

- So long, thanks for all the fish.