Virtualisasi Sumber Daya Cloud

Lanjutan

Pendahuluan

- Ada banyak realisasi fisik dari abstraksi mendasar yang diperlukan untuk menggambarkan pengoperasian sistem komputasi.
  - Penerjemah.
  - Memori.
  - link komunikasi.
- Virtualisasi adalah prinsip dasar cloud computing, menyederhanakan pengelolaan sumber daya fisik untuk tiga abstraksi tersebut.
- Keadaan mesin virtual (VM) yang berjalan di bawah monitor mesin virtual (VMM) dapat disimpan dan dipindahkan ke server lain untuk menyeimbangkan beban.
- Virtualisasi memungkinkan pengguna untuk beroperasi di lingkungan yang mereka kenal, daripada memaksa mereka ke lingkungan yang istimewa.

Pendahuluan Lanjutan

- Virtualisasi sumber daya cloud penting untuk:
  - Keamanan sistem, karena memungkinkan isolasi layanan yang berjalan pada perangkat keras yang sama.
  - Performa dan keandalan, karena memungkinkan aplikasi bermigrasi dari satu platform ke platform lainnya.
  - Pengembangan dan pengelolaan layanan yang ditawarkan oleh penyedia.Isolasi kinerja.

Virtualisasi

Mensimulasikan antarmuka ke objek fisik dengan:

- Multiplexing: membuat beberapa objek virtual dari satu instance objek fisik. Contoh - prosesor dimultipleks di antara sejumlah proses atau threads.
- Agregasi: membuat satu objek virtual dari beberapa objek fisik. Contoh - sejumlah disk fisik digabungkan ke dalam disk RAID.
- Emulasi: membangun objek virtual dari jenis objek fisik yang berbeda. Contoh - disk fisik mengemulasi Random Access Memory (RAM).
- Multiplexing dan emulasi. Contoh - memori virtual dengan multipleks paging memori dan disk nyata; alamat virtual mengemulasi alamat asli.

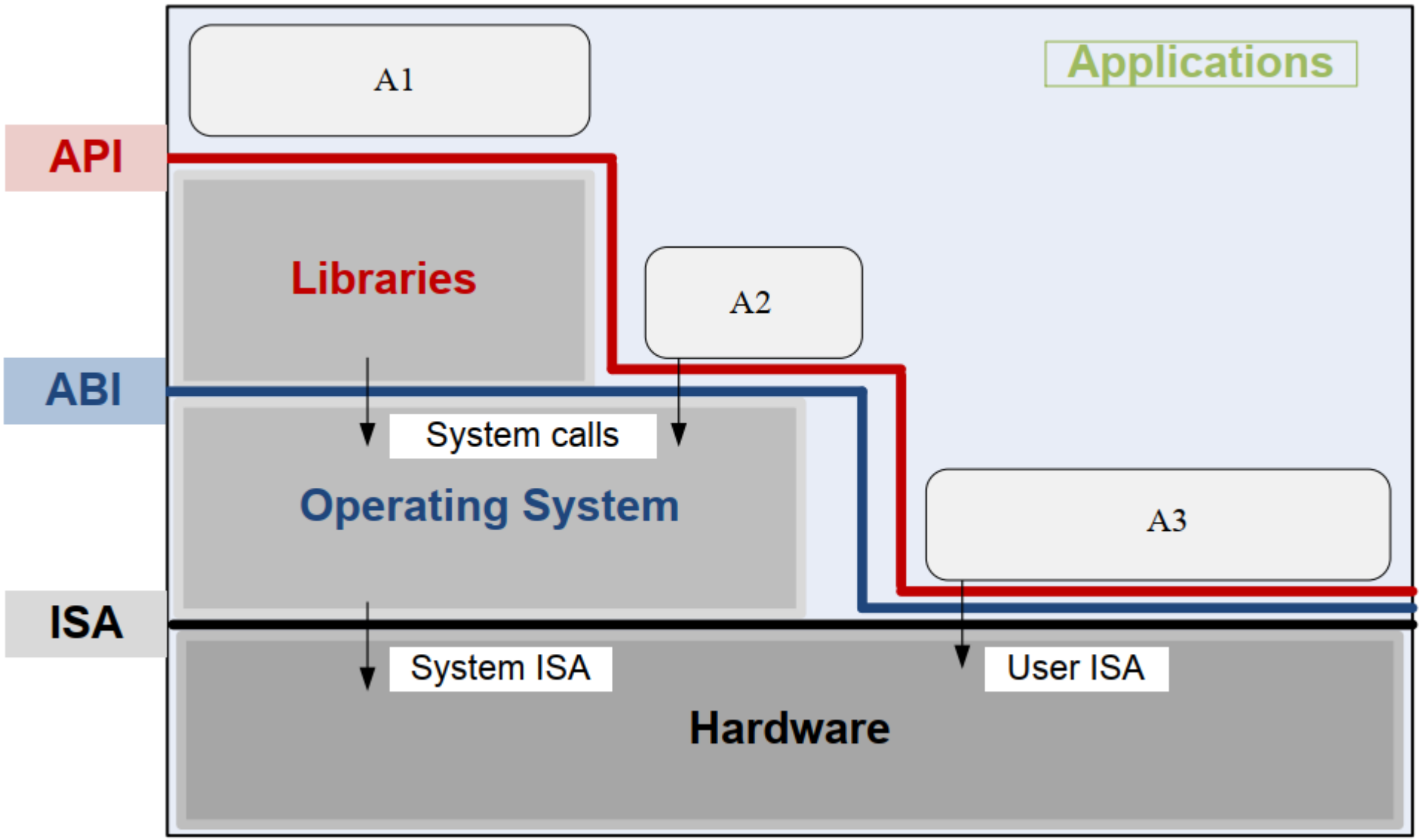
Layering (berlapis-lapis)

- Layering – pendekatan umum untuk mengelola kompleksitas sistem.
  - Meminimalkan interaksi antara subsistem dari sistem yang kompleks.
  - Menyederhanakan deskripsi subsistem; setiap subsistem diabstraksikan melalui antarmukanya dengan subsistem lainnya.
  - Kita dapat merancang, mengimplementasikan, dan memodifikasi subsistem individu secara mandiri.
- Layering dalam sistem komputer.
  - Perangkat Keras.
  - Perangkat Lunak.
    - ✓ Sistem operasi.
    - ✓ Libraries.
    - ✓ Aplikasi.

Interfaces

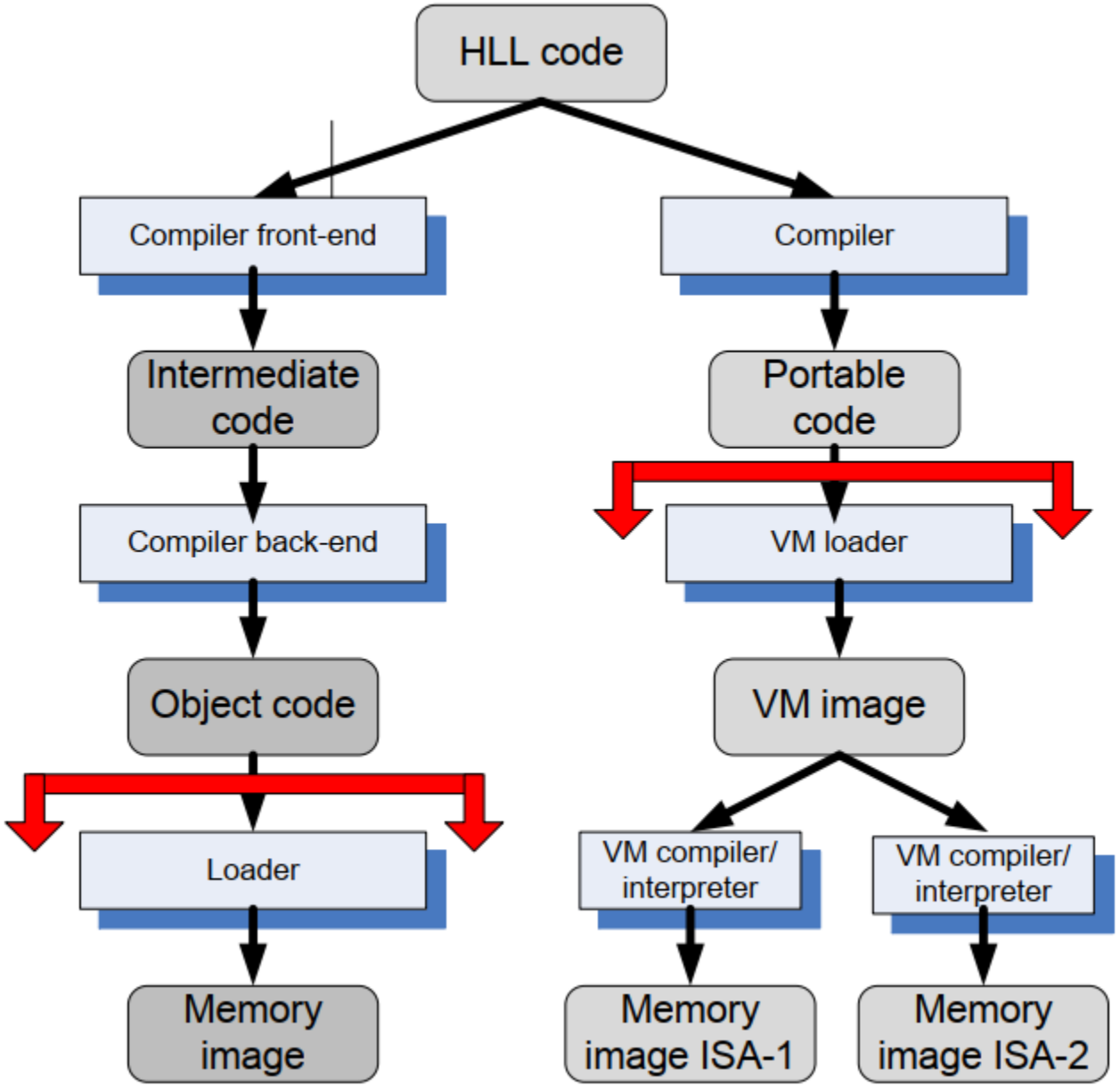
- Instruction Set Architecture (ISA) – pada batas antara perangkat keras dan perangkat lunak.
- Application Binary Interface (ABI) – memungkinkan pengabungan yang terdiri dari aplikasi dan modul library untuk mengakses perangkat keras; ABI tidak menyertakan instruksi sistem yang diistimewakan, melainkan memanggil panggilan sistem.
- Application Program Interface (API) - mendefinisikan set instruksi perangkat keras dirancang untuk mengeksekusi dan memberikan akses aplikasi ke ISA; itu termasuk panggilan HLL (high-level language) library yang sering dipergunakan untuk pemanggilan sistem.

Aplikasi menggunakan fungsi library (A1), membuat panggilan sistem (A2), dan mengeksekusi instruksi mesin (A3)



Portabilitas kode

- Binari yang dibuat oleh kompiler untuk ISA tertentu dan sistem operasi tertentu tidak portabel.
- Namun, dimungkinkan untuk mengkompilasi program HLL untuk lingkungan mesin virtual (VM) di mana kode portabel diproduksi dan didistribusikan dan kemudian diubah oleh penerjemah biner ke ISA dari sistem host. Terjemahan biner dinamis mengubah blok instruksi tamu dari kode portabel ke instruksi host dan menghasilkan peningkatan kinerja yang signifikan, karena blok tersebut di-cache dan digunakan kembali


HLL code
Compiler front-end
Compiler
Intermediate code
Portable code
Compiler back-end
VM loader
Object code
VM image
Loader
VM compiler/ interpreter
VM compiler/ interpreter
Memory image
Memory image ISA-1
Memory image ISA-2

Virtual machine monitor (VMM / hypervisor)

- Mempartisi sumber daya sistem komputer menjadi satu atau lebih mesin virtual (VM). Memungkinkan beberapa sistem operasi berjalan secara bersamaan pada satu platform perangkat keras.
- VMM memungkinkan:
  - Beberapa layanan untuk berbagi platform yang sama.
  - Migrasi langsung - perpindahan server dari satu platform ke platform lainnya.
  - Modifikasi sistem dengan tetap menjaga kompatibilitas dengan sistem asli.
  - Menerapkan isolasi di antara system untuk keamanan.

VMM memvirtualisasikan CPU dan memori

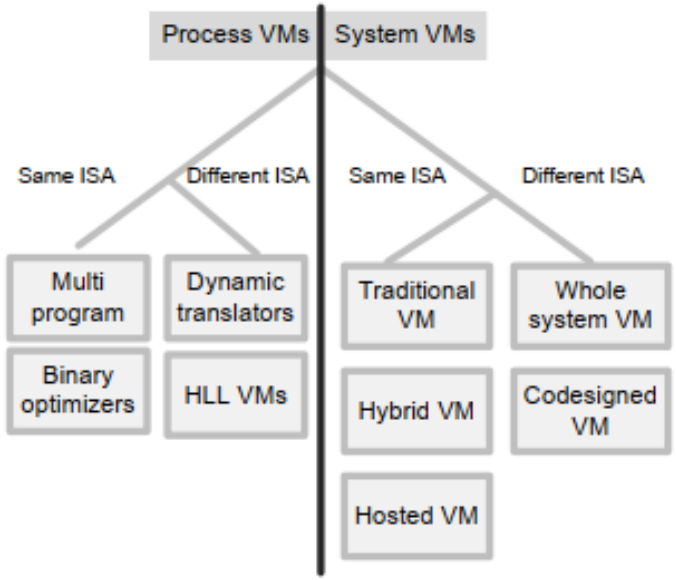
Sebuah VMM memunginkan untukj:

- menjalankan instruksi tertentu yang dijalankan hanya oleh OS tamu dan menjaga correctness (kebenaran) dan keamanan (Safety) operasi.
- Mengintrrupt dan mengirimkannya ke OS tamu.
- Mengontrol manajemen memori virtual.
- Mempertahankan shadowing table untuk setiap OS tamu dan mereplikasi setiap modifikasi yang dibuat oleh OS tamu dalam tabel halaman shadowing sendiri.
- shadowing table ini menunjuk ke bingkai halaman yang sebenarnya dan digunakan oleh Unit Manajemen Memori (MMU) untuk terjemahan alamat dinamis.Memantau kinerja sistem dan mengambil tindakan korektif untuk menghindari penurunan kinerja. Misalnya, VMM dapat menukar Mesin Virtual untuk menghindari thrashing.

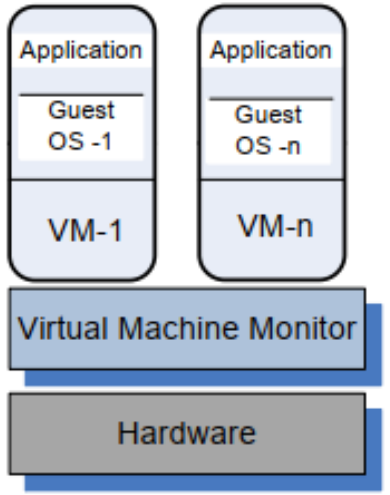
Virtual machines (VMs)

- VM - lingkungan terisolasi yang tampak seperti seluruh komputer,tetapi sebenarnya hanya memiliki akses ke sebagian dari sumber daya komputer.
- Proses VM - platform virtual yang dibuat untuk proses individual dan dihancurkan setelah proses berakhir.
- Sistem VM - mendukung sistem operasi bersama dengan banyak proses pengguna.
- VM Tradisional - mendukung beberapa mesin virtual dan berjalan langsung di perangkat keras.
- Hybrid VM - berbagi perangkat keras dengan sistem operasi host dan mendukung beberapa mesin virtual.
- Hosted VM - berjalan di bawah sistem operasi host.

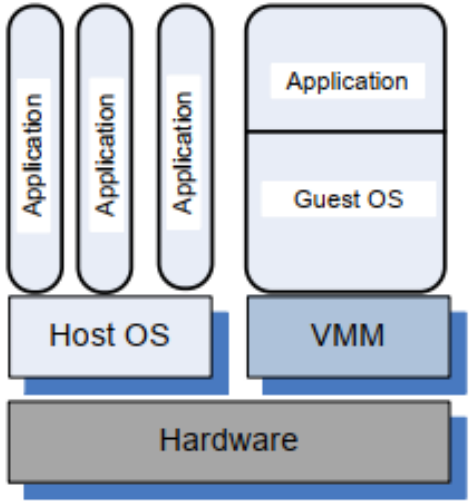
Traditional, hybrid, and hosted VMs



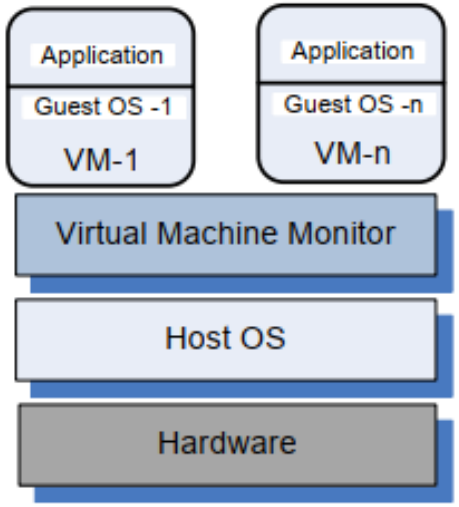
(a)



(b)



(c)



(d)

Name	Host ISA	Guest ISA	Host OS	guest OS	Company
Integrity VM	*x86*-64	*x86*-64	HP-Unix	Linux,Windows HP Unix	HP
Power VM	Power	Power	No host OS	Linux, AIX	IBM
z/VM	z-ISA	z-ISA	No host OS	Linux on z-ISA	IBM
Lynx Secure	*x86*	*x86*	No host OS	Linux, Windows	LinuxWorks
Hyper-V Server	*x86*-64	*x86*-64	Windows	Windows	Microsoft
Oracle VM	*x86*, *x86*-64	*x86*, *x86*-64	No host OS	Linux, Windows	Oracle
RTS Hypervisor	*x86*	*x86*	No host OS	Linux, Windows	Real Time Systems
SUN xVM	*x86*, SPARC	same as host	No host OS	Linux, Windows	SUN
VMware EX Server	*x86*, *x86*-64	*x86*, *x86*-64	No host OS	Linux, Windows Solaris, FreeBSD	VMware
VMware Fusion	*x86*, *x86*-64	*x86*, *x86*-64	MAC OS *x86*	Linux, Windows Solaris, FreeBSD	VMware
VMware Server	*x86*, *x86*-64	*x86*, *x86*-64	Linux, Windows	Linux, Windows Solaris, FreeBSD	VMware
VMware Workstation	*x86*, *x86*-64	*x86*, *x86*-64	Linux, Windows	Linux, Windows Solaris, FreeBSD	VMware
VMware Player	*x86*, *x86*-64	*x86*, *x86*-64	Linux Windows	Linux, Windows Solaris, FreeBSD	VMware
Denali	*x86*	*x86*	Denali	ILVACO, NetBSD	University of Washington
Xen	*x86*, *x86*-64	*x86*, *x86*-64	Linux Solaris	Linux, Solaris NetBSD	University of Cambridge

Isolasi kinerja dan keamanan

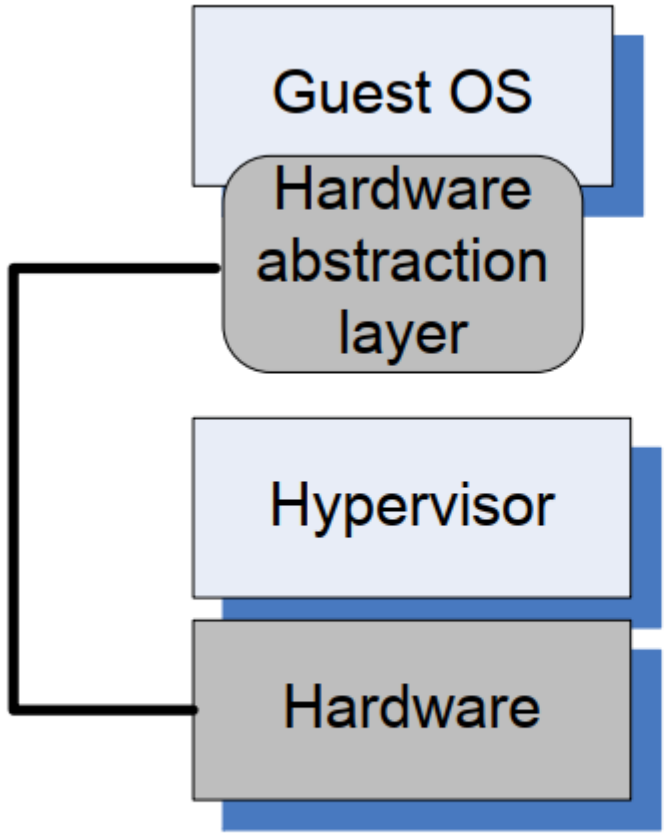
- Perilaku run-time aplikasi dipengaruhi oleh aplikasi lain yang berjalan secara bersamaan pada platform yang sama dan bersaing untuk siklus CPU, cache, memori utama, disk, dan akses jaringan. Dengan demikian, sulit untuk memprediksi waktu penyelesaian!
- Isolasi kinerja - kondisi kritis untuk jaminan QoS di lingkungan komputasi bersama.VMM adalah sistem spesifikasi yang jauh lebih sederhana dan lebih baik daripada sistem operasi tradisional. Contoh - Xen memiliki sekitar 60.000 baris kode; Denali hanya memiliki sekitar setengahnya, 30.000.
- Kerentanan keamanan VMM sangat berkurang karena sistem mengekspos sejumlah kecil fungsi yang diperlukan.

Arsitektur komputer dan virtualisasi

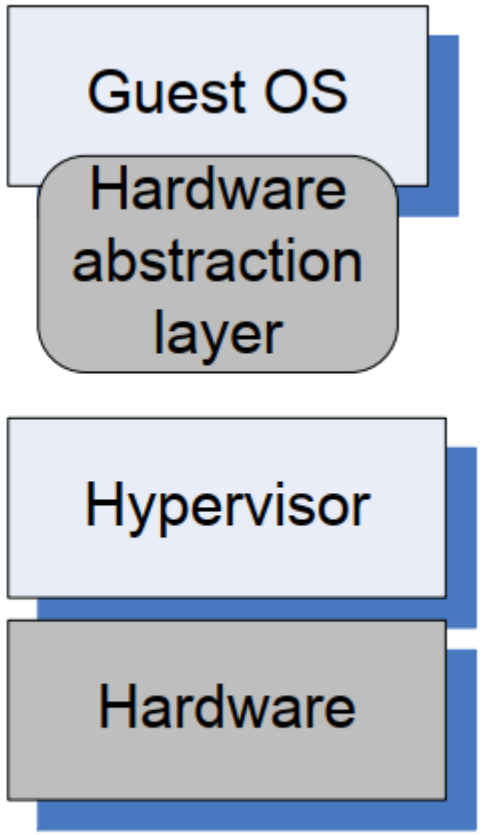
- Kondisi untuk virtualisasi yang efisien:
  - Program yang berjalan di bawah VMM harus menunjukkan perilaku yang pada dasarnya identik dengan yang ditunjukkan saat dijalankan pada mesin yang setara secara langsung.
  - VMM harus berada dalam kendali penuh atas sumber daya virtual.
  - Secara statistik dari instruksi mesin harus dieksekusi tanpa intervensi dari VMM.
- Dua kelas instruksi mesin:
  - Sensitif - memerlukan tindakan pencegahan khusus pada waktu eksekusi:
    - Kontrol sensitif - instruksi yang mencoba mengubah alokasi memori atau mode istimewa.
    - Mode sensitif - instruksi yang perilakunya berbeda dalam mode istimewa.
  - Tidak berbahaya - tidak sensitif.

Virtualisasi penuh dan paravirtualisasi

- Virtualisasi penuh – OS tamu dapat berjalan tidak berubah di bawah VMM seolah-olah dijalankan langsung pada platform perangkat keras.
  - Memerlukan arsitektur yang dapat divirtualisasikan.
  - Contoh: Vmware.
- Paravirtualization - sistem operasi tamu dimodifikasi untuk hanya menggunakan instruksi yang dapat divirtualisasikan. Alasan paravirtualisasi:
  - Beberapa aspek perangkat keras tidak dapat divirtualisasikan.
  - Peningkatan kinerja.
  - Hadirkan antarmuka yang lebih sederhana.
  - Contoh: Xen, Denaly



(a) Full virtualization



(b) Paravirtualization

Virtualisasi arsitektur x86

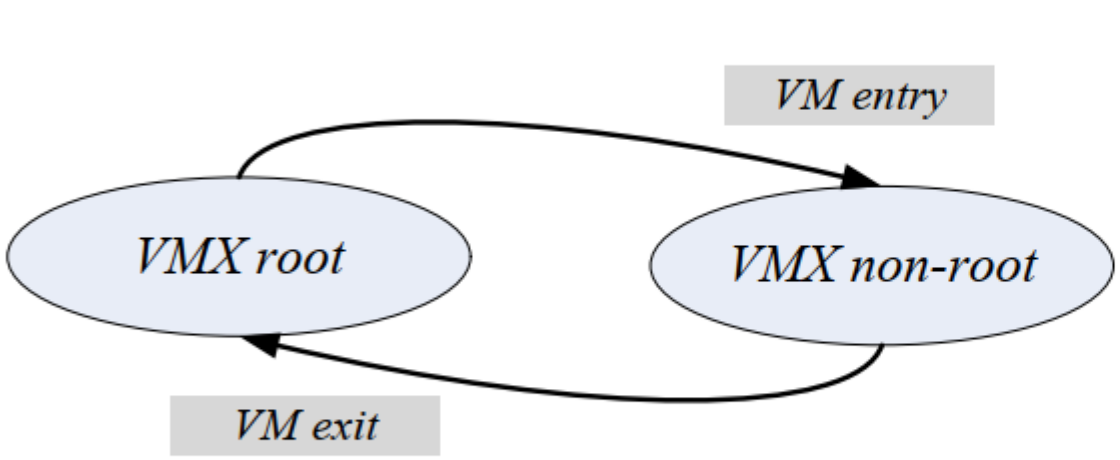
- Ring de-privileging - VMM memaksa sistem operasi dan aplikasi untuk berjalan pada tingkat hak istimewa yang lebih besar dari 0.
- Ring aliasing - OS tamu dipaksa untuk berjalan pada tingkat hak istimewa selain yang awalnya dirancang untuk itu.Kompresi ruang alamat - VMM menggunakan bagian dari ruang alamat tamu untuk menyimpan beberapa struktur data sistem.
- Akses non-faulting ke status istimewa - beberapa instruksi penyimpanan hanya dapat dieksekusi pada level istimewa 0 karena mereka beroperasi pada struktur data yang mengontrol operasi CPU.
- Mereka gagal secara diam-diam ketika dieksekusi pada tingkat hak istimewa selain 0.Panggilan sistem tamu yang menyebabkan transisi ke/dari tingkat hak istimewa 0 harus ditiru oleh VMM.
- Virtualisasi interupsi - sebagai respons terhadap interupsi fisik, VMM menghasilkan “interupsi virtual” dan mengirimkannya nanti ke OS tamu target yang dapat menutupi interupsi.

Virtualisasi arsitektur x86 Lanjutan

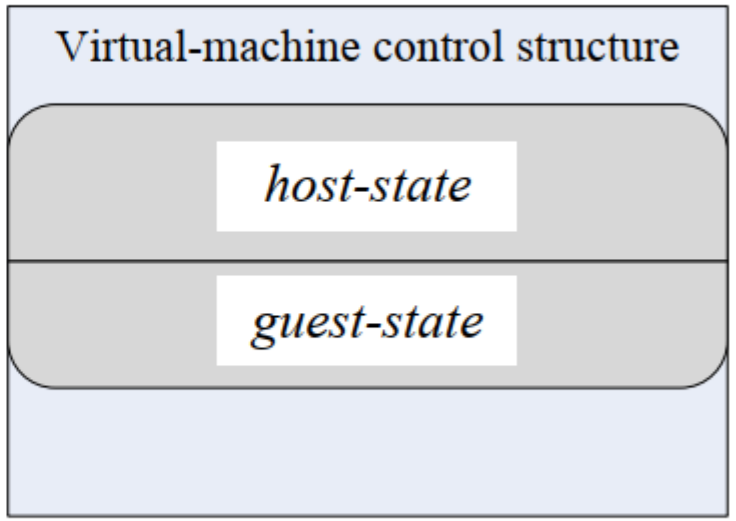
- Akses ke status tersembunyi - elemen status sistem, misalnya, cache deskriptor untuk register segmen, disembunyikan; tidak ada mekanisme untuk menyimpan dan memulihkan komponen tersembunyi ketika ada peralihan konteks dari satu VM ke VM lainnya.
- Ring Compression - paging dan segmentasi melindungi kode VMM agar tidak ditimpa oleh OS dan aplikasi tamu. Sistem yang berjalan dalam mode 64-bit hanya dapat menggunakan paging, tetapi paging tidak membedakan antara tingkat hak istimewa 0, 1, dan 2, sehingga OS tamu harus berjalan pada tingkat hak istimewa 3, yang disebut mode (0/3/3). Privilege level 1 dan 2 tidak dapat digunakan, namanya kompresi ring.
- Register prioritas tugas sering digunakan oleh OS tamu; VMM harus melindungi akses ke register ini dan menjebak semua upaya untuk mengaksesnya. Ini dapat menyebabkan penurunan kinerja yang signifikan.

VT-x, peningkatan arsitektur utama

- Mendukung dua mode operasi:
  - VMX root - untuk operasi VMM.
  - VMX non-root - mendukung VM.
- Struktur Kontrol Mesin Virtual termasuk host-state dan guest-state area.
  - Entri VM - status prosesor dimuat dari status tamu VM dijadwalkan untuk dijalankan; kemudian kontrol ditransfer dari VMM ke VM.
  - VM exit - menyimpan status prosesor di area status tamu menjalankan VM; kemudian memuat status prosesor dari area host-state, akhirnya mentransfer kontrol ke VMM.


VM entry
VMX root
VMX non-root
VM exit

(a)


Virtual-machine control structure
host-state
guest-state

(b)

VT-d, arsitektur virtualisasi baru

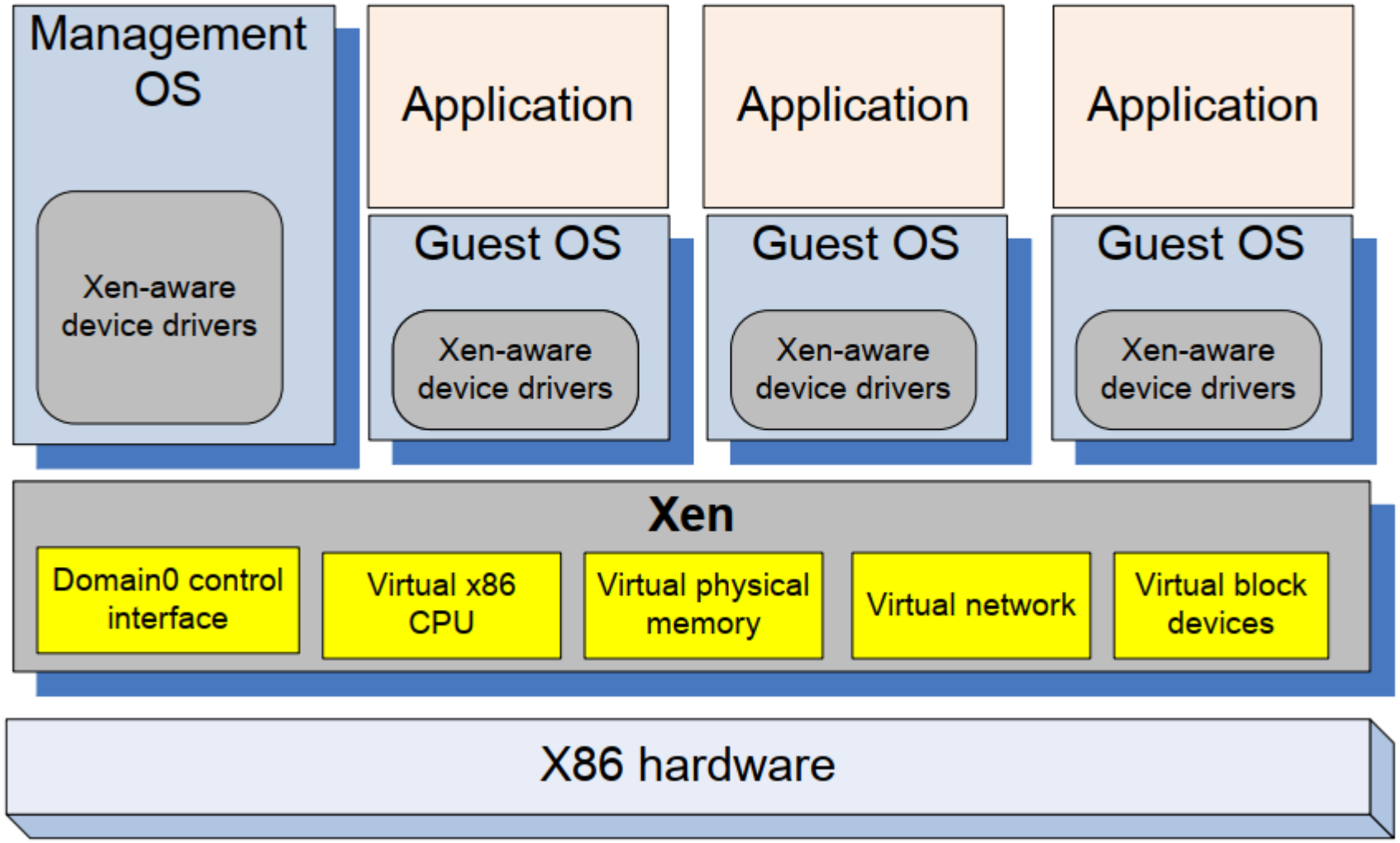
Virtualisasi I/O MMU memberi VM akses langsung ke perangkat periferal. VT-d mendukung:

- pemetaan ulang alamat DMA, terjemahan alamat untuk transfer DMA perangkat.
- Interupsi pemetaan ulang, isolasi interupsi perangkat, dan perutean VM.
- Penetapan perangkat I/O, perangkat dapat ditetapkan oleh administrator ke VM dalam konfigurasi apa pun.
- Fitur keandalan, ia melaporkan dan mencatat DMA dan menginterupsi kesalahan yang merusak memori dan memengaruhi isolasi VM.

Xen - VMM berdasarkan paravirtualization

- Tujuan dari grup Cambridge - merancang VMM yang mampu menskalakan hingga sekitar 100 VM yang menjalankan aplikasi dan layanan standar tanpa modifikasi apa pun pada Application Binary Interface (ABI).
- Linux, Minix, NetBSD, FreeBSD, NetWare, dan OZONE dapat beroperasi sebagai OS tamu Xen paravirtual yang berjalan pada arsitektur x86, x86-64, Itanium, dan ARM.
- Domain Xen - kumpulan ruang alamat yang menghosting OS tamu dan aplikasi yang berjalan di bawah OS tamu. Berjalan pada CPU virtual.
  - Dom0 - didedikasikan untuk pelaksanaan fungsi kontrol Xen dan instruksi istimewa.
  - DomU - domain pengguna.
- Aplikasi melakukan panggilan sistem menggunakan hypercall yang diproses oleh Xen; instruksi istimewa yang dikeluarkan oleh OS tamu diparavirtualisasikan dan harus divalidasi oleh Xen.

Xen


Management OS
Xen-aware device drivers
Application
Guest OS
Xen-aware device drivers
Application
Guest OS
Xen-aware device drivers
Application
Guest OS
Xen-aware device drivers
Xen
Domain0 control interface
Virtual x86 CPU
Virtual physical memory
Virtual network
Virtual block devices
X86 hardware

Implementasi Xen pada arsitektur x86

- Xen berjalan pada hak istimewa Level 0, OS tamu di Level 1, dan aplikasi di Level 3.
- Arsitektur x86 tidak mendukung penandaan entri TLB atau manajemen perangkat lunak TLB. Dengan demikian, perpindahan ruang alamat, ketika VMM mengaktifkan OS yang berbeda, membutuhkan TLB flush yang lengkap; ini memiliki dampak negatif pada kinerja.
- Solusi - muat Xen dalam segmen 64 MB di bagian atas setiap ruang alamat dan delegasikan pengelolaan tabel halaman perangkat keras ke OS tamu dengan intervensi minimal dari Xen. Wilayah ini tidak dapat diakses atau dipetakan ulang oleh OS tamu.
- Xen menjadwalkan domain individu menggunakan algoritma penjadwalan Borrowed Virtual Time (BVT).
- OS tamu harus mendaftarkan tabel deskripsi ke Xen dengan alamat penangan pengecualian untuk validasi.

Dom0 components

- XenStore – proses Dom0.
  - Mendukung registri dan layanan penamaan di seluruh sistem.
  - Diimplementasikan sebagai penyimpanan nilai kunci hierarkis (key-value storage).
  - Fungsi watch memberi tahu pendengar tentang perubahan kunci dalam penyimpanan yang telah mereka subscribe.
  - Berkomunikasi dengan VM tamu melalui memori bersama menggunakan hak istimewa Dom0.
- Toolstack - bertanggung jawab untuk membuat, menghancurkan, dan mengelola sumber daya dan hak istimewa VM.
  - Untuk membuat VM baru, pengguna menyediakan file konfigurasi yang menjelaskan alokasi memori dan CPU serta konfigurasi perangkat.
  - Toolstack mem-parsing file ini dan menulis informasi ini di XenStore.
  - Memanfaatkan hak istimewa Dom0 untuk memetakan memori tamu, memuat kernel dan BIOS virtual, dan mengatur saluran komunikasi awal dengan XenStore dan dengan konsol virtual saat VM baru dibuat.

Strategi untuk manajemen memori virtual, multiplexing CPU, dan Perangkat I/O

Function	Strategy
Paging	A domain may be allocated discontinuous pages. A guest OS has direct access to page tables and handles pages faults directly for efficiency; page table updates are batched for performance and validated by *Xen* for safety.
Memory	Memory is statically partitioned between domains to provide strong isolation. `XenoLinux` implements a *balloon driver* to adjust domain memory.
Protection	A guest OS runs at a lower priority level, in ring 1, while *Xen* runs in ring 0.
Exceptions	A guest OS must register with *Xen* a description table with the addresses of exception handlers previously validated; exception handlers other than the page fault handler are identical with *x86* native exception handlers.
System calls	To increase efficiency, a guest OS must install a "fast" handler to allow system calls from an application to the guest OS and avoid indirection through *Xen*.
Interrupts	A lightweight event system replaces hardware interrupts; synchronous system calls from a domain to *Xen* use *hypercalls* and notifications are delivered using the asynchronous event system.
Multiplexing	A guest OS may run multiple applications.
Time	Each guest OS has a timer interface and is aware of "real" and "virtual" time.
Network and I/O devices	Data is transferred using asynchronous I/O rings; a ring is a circular queue of descriptors allocated by a domain and accessible within *Xen*.
Disk access	Only *Dom0* has direct access to IDE and SCSI disks; all other domains access persistent storage through the Virtual Block Device (VBD) abstraction.

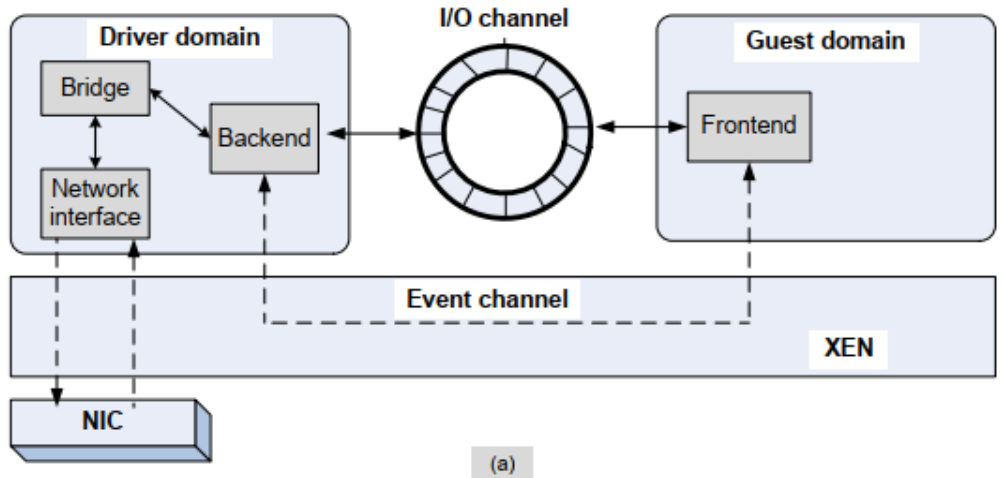
Abstraksi Xen untuk jaringan dan I/O

- Setiap domain memiliki satu atau lebih Antarmuka Jaringan Virtual (VIF) yang mendukung fungsionalitas kartu antarmuka jaringan. VIF dilampirkan ke Virtual Firewall-Router (VFR).
- Driver split memiliki front-end di DomU dan back-end di Dom0; keduanya berkomunikasi melalui cincin di memori bersama.
- Ring - antrian melingkar dari deskriptor yang dialokasikan oleh domain dan dapat diakses dalam Xen. Deskriptor tidak berisi data, buffer data dialokasikan secara off-band oleh OS tamu.
- Dua ring deskriptor buffer, satu untuk pengiriman paket dan satu lagi untuk penerimaan paket.
- Untuk mengirimkan paket:
  - OS tamu mengantrekan deskriptor buffer ke ring kirim,
  - kemudian Xen menyalin deskriptor dan memeriksa keamanan,
  - hanya menyalin header paket, bukan payload, dan
  - mengeksekusi aturan yang cocok.

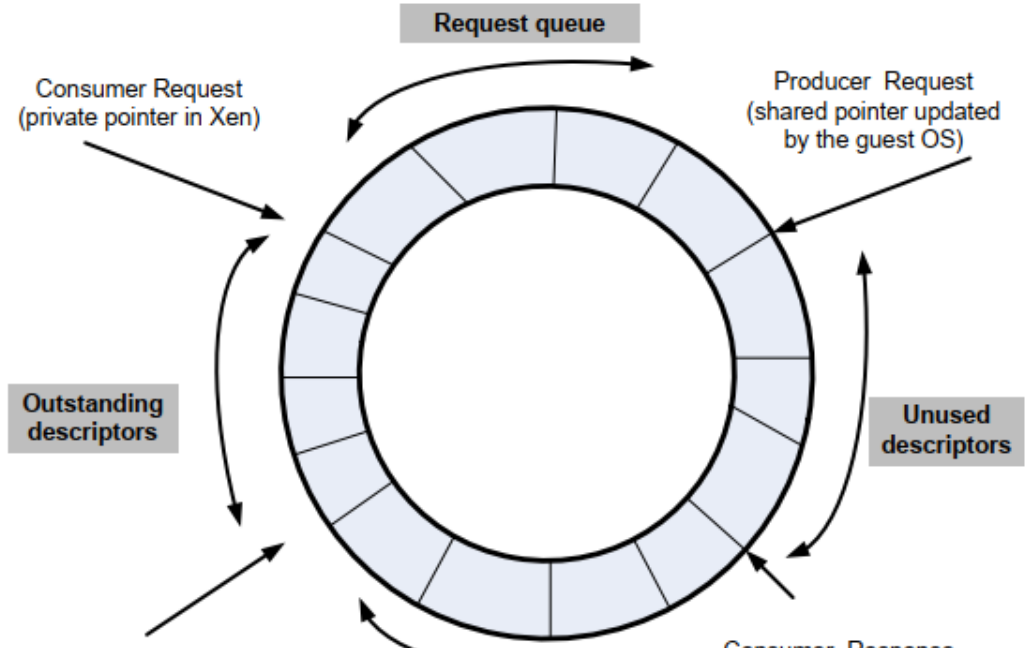
Semantik nol-salinan Xen untuk transfer data menggunakan cincin I/O.

(a) Komunikasi antara domain tamu dan domain driver melalui I/O dan saluran acara; NIC adalah Pengontrol Antarmuka Jaringan.

(b) (b) cincin buffer yang melingkar.



(a)



(b)

Xen 2.0

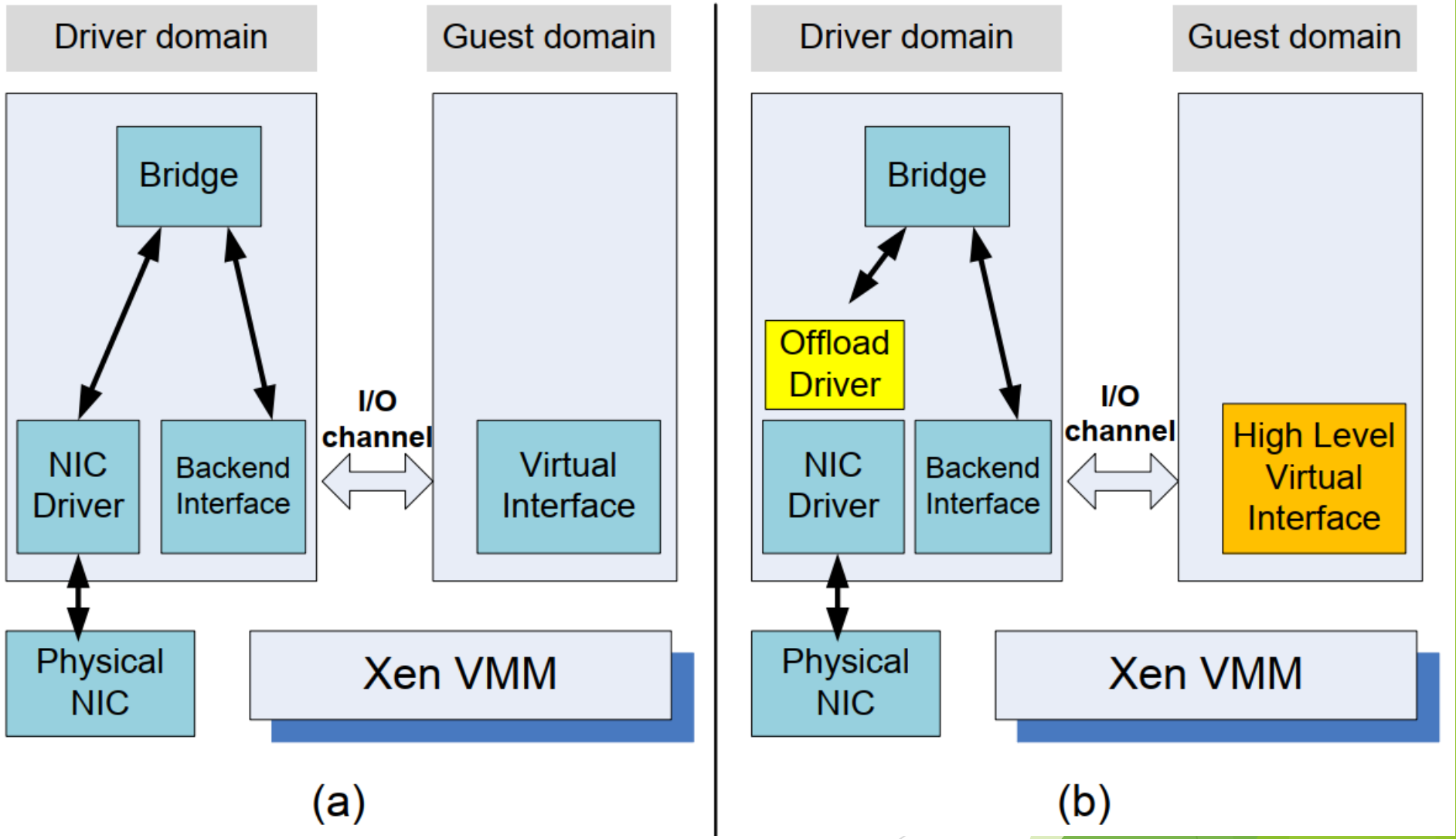
Melakukan optimasi pada:

- Antarmuka virtual - memanfaatkan kemampuan beberapa NIC fisik, seperti checksum offload.
- Saluran I/O - daripada menyalin buffer data yang menyimpan paket, setiap paket dialokasikan di halaman baru dan kemudian halaman fisik yang berisi paket dipetakan ulang ke domain target.
- Memori virtual - memanfaatkan superpage dan perangkat keras pemetaan halaman global pada prosesor. Entri superpage mencakup 1.024 halaman memori fisik dan mekanisme terjemahan alamat memetakan satu set halaman yang berdekatan ke satu set halaman fisik yang berdekatan. Ini membantu mengurangi jumlah kesalahan.

Arsitektur jaringan Xen .

(a) Arsitektur asli;

(b) (b) Arsitektur yang dioptimalkan



Perbandingan Data optimasi Xen 2.0

System	Receive data rate (Mbps)	Send data rate (Mbps)
Linux	2 508	3 760
Xen driver	1 728	3 760
Xen guest	820	750
optimized *Xen* guest	970	3 310

Perbandingan kecepatan pengiriman dan penerimaan data untuk sistem Linux asli, domain driver Xen, domain tamu Xen asli, dan domain tamu Xen yang dioptimalkan.

Perbandingan kinerja mesin virtual

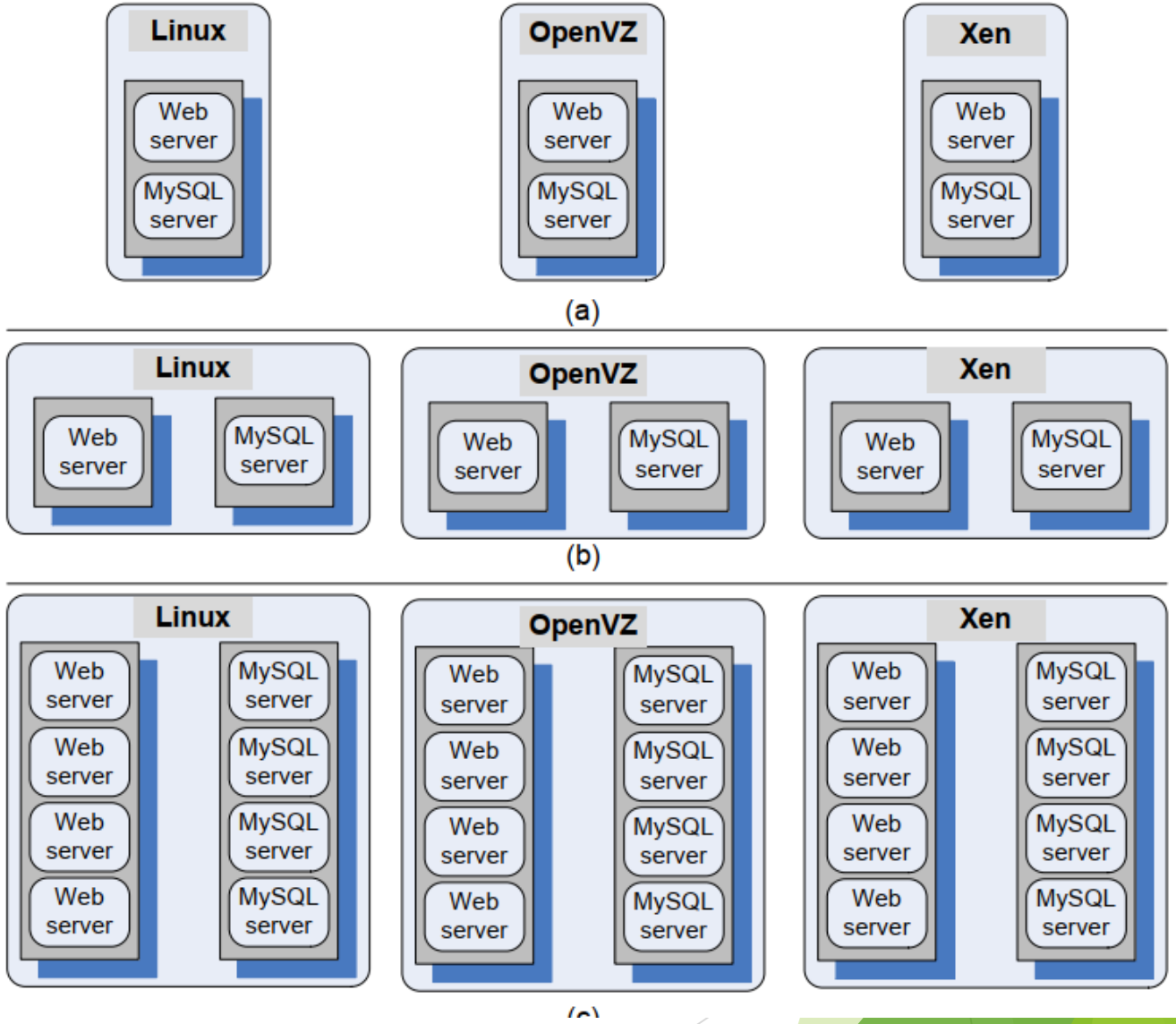
- Bandingkan kinerja Xen dan OpenVZ dengan, sistem operasi standar, Linux biasa.
- Hal-hal yang dibandingan adalah:
  - Bagaimana peningkatan kinerja dengan beban?
  - Apa dampak dari campuran aplikasi?
  - Apa implikasi dari penetapan beban pada server individual?
- Kesimpulan utama:
  - Overhead virtualisasi Xen jauh lebih tinggi daripada OpenVZ dan ini terutama disebabkan oleh misses L2-cache.
  - Penurunan kinerja saat beban kerja meningkat juga terlihat pada Xen.
  - Hosting beberapa tingkatan dari aplikasi yang sama di server yang sama bukanlah solusi yang optimal.

Pengaturan untuk perbandingan kinerja sistem Linux asli dengan OpenVZ, dan sistem Xen. Aplikasi tersebut adalah web server dan database server MySQL.

(a) Eksperimen pertama, web dan DB, berbagi satu sistem;

(b) (b) Eksperimen kedua, web dan DB, dijalankan pada dua sistem yang berbeda;

(c) (c) Eksperimen ketiga, web dan DB, dijalankan pada dua sistem yang berbeda dan masing-masing memiliki empat instance.



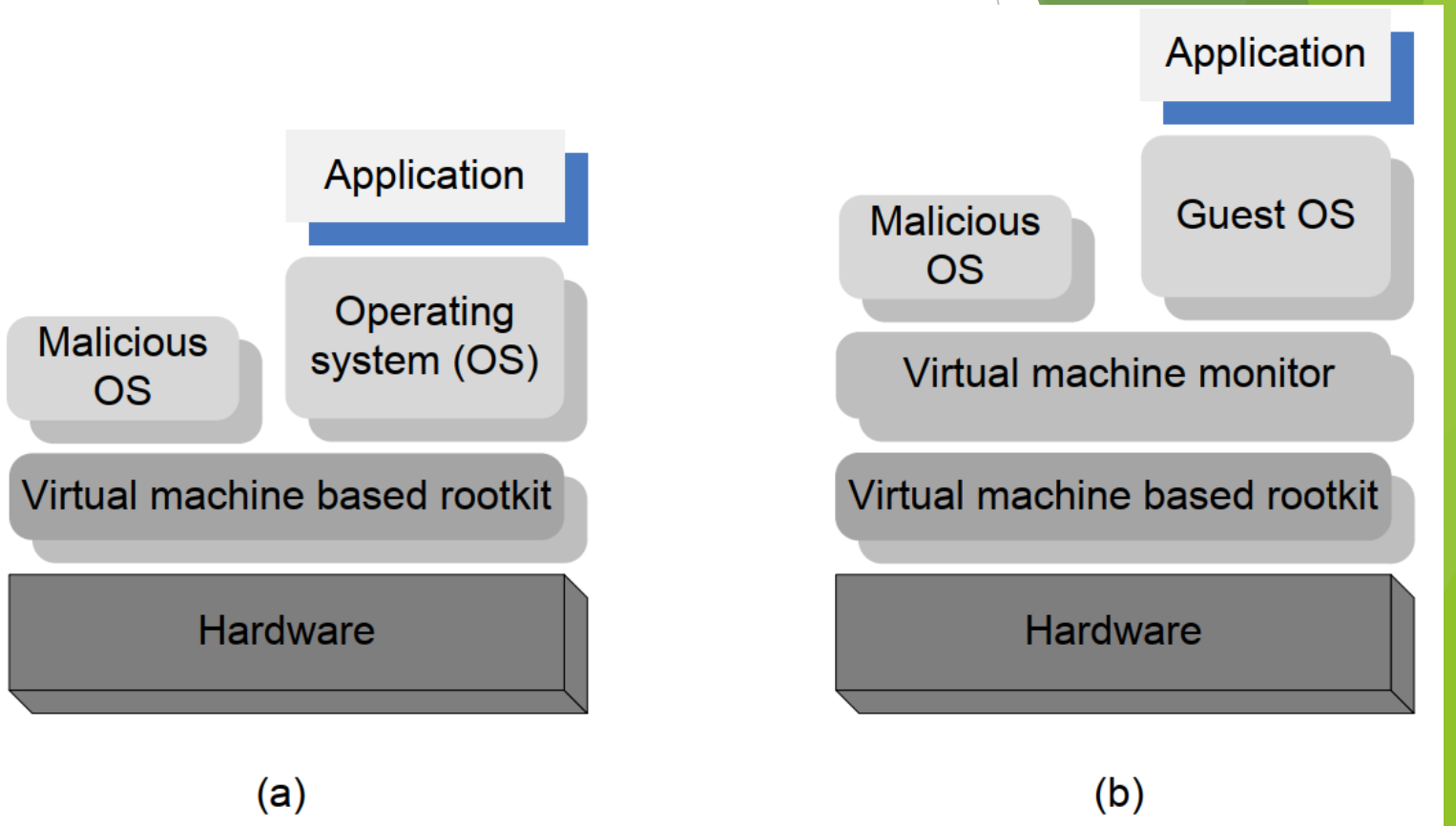
vulnerabilities dari virtualisasi

- Dalam struktur berlapis, mekanisme pertahanan di beberapa lapisan dapat dinonaktifkan oleh malware yang berjalan pada lapisan di bawahnya.
- Adalah layak untuk menyisipkan VMM jahat, Rootkit Berbasis Mesin Virtual (VMBR) antara perangkat keras fisik dan sistem operasi.
- Rootkit - malware dengan akses istimewa ke sistem.
- VMBR dapat mengaktifkan OS berbahaya yang terpisah untuk berjalan secara diam-diam dan membuat OS berbahaya ini tidak terlihat oleh OS tamu dan aplikasi yang berjalan di bawahnya.
- Di bawah perlindungan VMBR, OS berbahaya dapat:
  - mengamati data, peristiwa, atau keadaan sistem target.
  - menjalankan layanan, seperti relai spam atau serangan penolakan layanan terdistribusi.
  - mengganggu aplikasi.

Penyisipan Rootkit Berbasis Mesin Virtual (VMBR) sebagai lapisan terendah dari tumpukan perangkat lunak yang berjalan pada perangkat keras fisik;

(a) di bawah sistem operasi;

(b) (b) di bawah monitor mesin virtual yang sah. VMBR memungkinkan OS berbahaya untuk berjalan diam-diam dan membuatnya tidak terlihat oleh OS asli atau tamu dan aplikasi.



Fitur SFI untuk Native Client pada x86-32, x86-64, dan ARM.

Feature /Architecture	*x86*-32	*x86*-64	*ARM*
Addressable memory	1 GB	4 GB	1 GB
Virtual base address	any	44GB	0
Data model	ILP32	ILP32	ILP 32
Reserved registers	0 of 8	1 of 16	0 of 16
Data address mask	None	Implicit in result width	Explicit instruction
Control address mask	Explicit instruction	Explicit instruction	Explicit instruction
Bundle size (bytes)	32	32	16
Data in text segment	forbidden	forbidden	allowed
Safe address registers	all	rsp, rbp	sp
Out-of-sandbox store	trap	wraps mod 4 GB	No effect
Out-of-sandbox jump	trap	wraps mod 4 GB	wraps mod 1 GB

Link Video penjelasan

https://drive.google.com/file/d/1ENN3ntC-9jot5j8B5zZu-wjiTYX-PbiD/view?usp=sharing

Terimakasih