Systemy operacyjne

Wykład 15: Wirtualizacja

Terminy podstawowe

maszyna wirtualna środowisko uruchomieniowe, w którym można wykonywać oprogramowanie działające w trybie jądra

hipernadzorca program nadzorujący działanie VM,
jest dla maszyn wirtualnych tym czym jądro SO dla procesów

system gospodarza wykonuje kod hipernadzorcy

system gościa jest wykonywany wewnątrz maszyny wirtualnej

emulator vs. **symulator** obydwa tworzą iluzję sprzętu,
pierwszy nakierowany na wydajność, drugi na dokładność

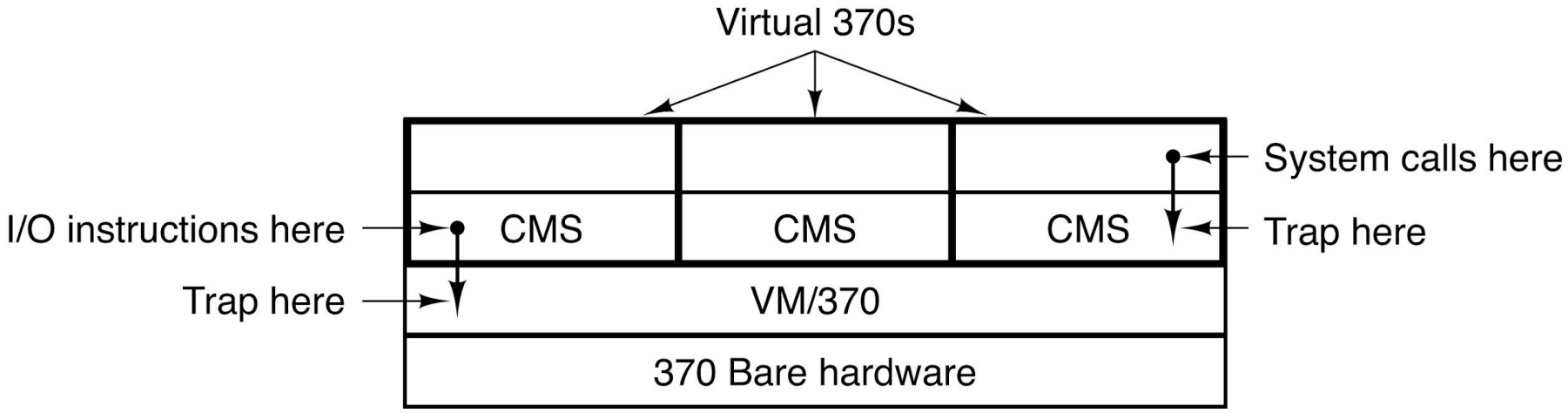
Zalety stosowania VM

Migawki (ang. *shapshot*) Zamrażamy stan VM i zrzucamy na dysk. Dalsze działanie powoduje uszkodzenie SO gościa → przywracamy migawkę. Wydajne tworzenie z użyciem techniki kopiowania przy zapisie. Przydatne przy testowaniu oprogramowania → sandboxing.

Migracja (ang. *migration*) Maszynę trzeba wyłączyć lub jest przeciążona? Przenosimy stan VM na inny serwer! Nie chcemy przerywać świadczenia usług (ang. *live migration*)? Synchronizacja stanu (dysk, RAM) z maszyną docelową. Próg brudnych strony pamięci, bloków dysku, itd. mały? Wstrzymujemy oryginał, kończymy synchronizację, uruchamiamy nową.

Konsolidacja (ang. *consolidation*) Kilka maszyn realizujących różne funkcje (backup, serwer Web, poczta, itd.) ze względu na izolację. Część niedociążona → konwersja do VM i wrzucenie na jedną mocną maszynę.

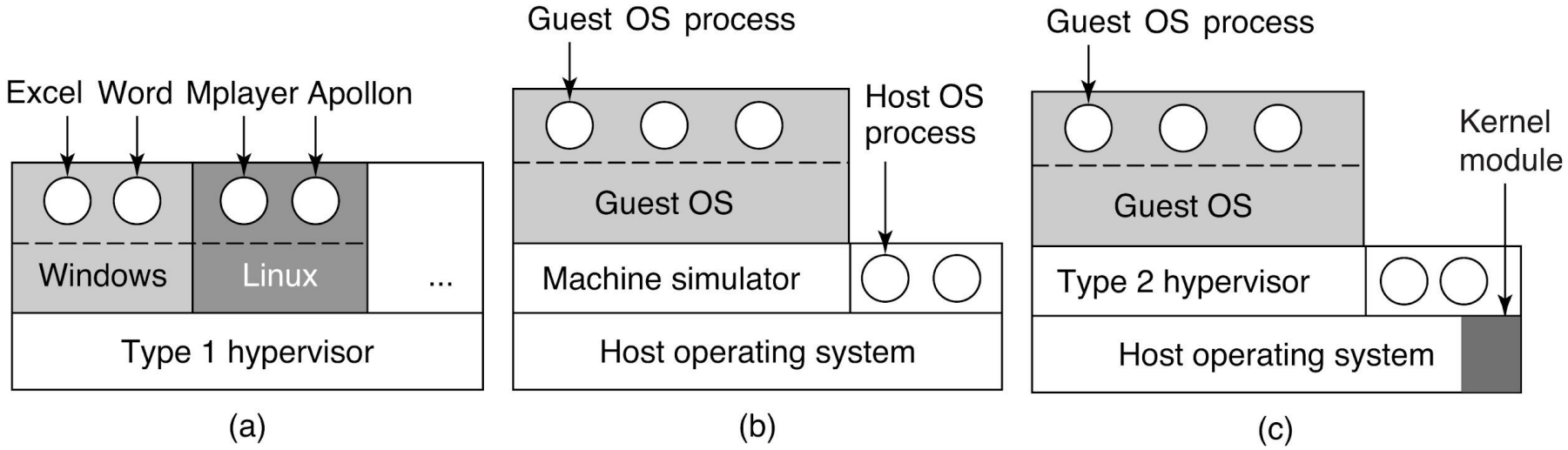
Historia maszyn wirtualnych



Pierwsze zastosowanie komercyjne w mainframe IBM (1972).
Jak uruchamiać oprogramowanie ze starszych maszyn?

Dodatkowy tryb pracy procesora (ang. *hypervisor*). Monitor maszyn wirtualnych współdzieli zasoby między systemy (czas procesora, pamięć, urządzenia wejścia-wyjścia). Przechwytywanie **instrukcji uprzywilejowanych** i **wrażliwych**.

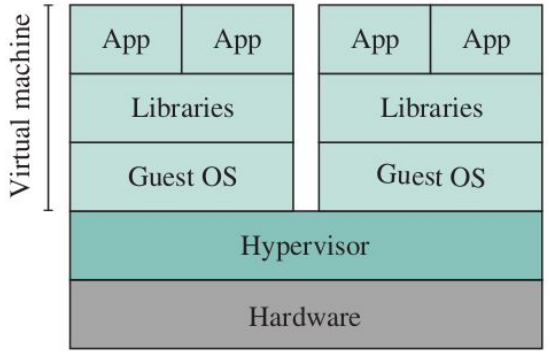
Rodzaje maszyn wirtualnych



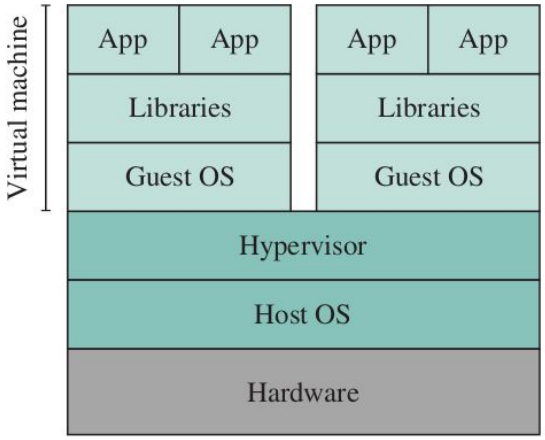
a. na gołym sprzęcie (`Hyper-V`, `Xen`, `VMware ESXi`)
b. symulator maszyny (`QEMU full system emulator`)
c. na systemie gospodarza (`VirtualBox`, `QEMU`, `VMWare Player`)

Czasami kategoryzacja jest ciężka → `KVM` i `byhve` stoją po środku.

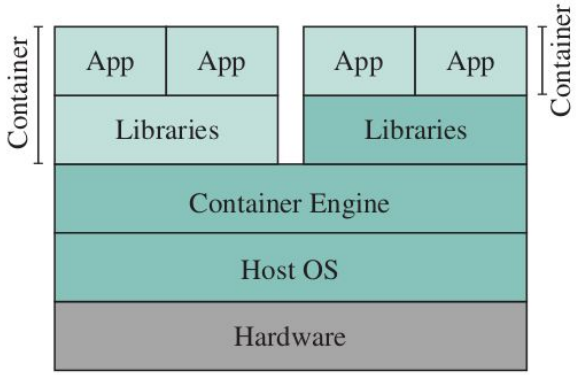
Różnica między kontenerem, a maszyną wirtualną



(a) Type 1 Hypervisor



(b) Type 2 Hypervisor



(c) Container

Wirtualizacja na poziomie systemu operacyjnego

Jądro SO tworzy kontener posiadający osobną przestrzeń nazw dla procesów, użytkowników, połączeń sieciowych, itd. Np. procesy wewnątrz kontenera widzą tylko te procesy, które zostały uruchomione w jego wnętrzu. Zasoby lokalnej przestrzeni nazw dla danego kontenera są odzwierciedlone w globalną przestrzeń nazw.

Przykłady: `Linux containers` (`Docker`), `FreeBSD jails`.

```
docker run -it debian:latest /bin/bash
```

Wymagania odnośnie wirtualizacji

Bezpieczeństwo Hipernadzorca musi mieć pełną kontrolę nad zwirtualizowanymi zasobami – tj. bezpieczny multipleksowany dostęp do zasobów, izolacja między VM, współdzielenie danych między VM dla wydajności.

Dokładność Zachowanie programów użytkownika na wirtualnej maszynie identyczne z zachowaniem na prawdziwym sprzęcie, pomijając drobne różnice w wydajności. Jądro SO może działać inaczej, jeśli jest świadome działania pod kontrolą VMM.

Wydajność Większość instrukcji programów działa wprost na procesorze. Niski narzut wirtualizacji – wykonywanie kodu w VM nie wymaga częstego angażowania hipernadzorcy.

Pełna wirtualizacja

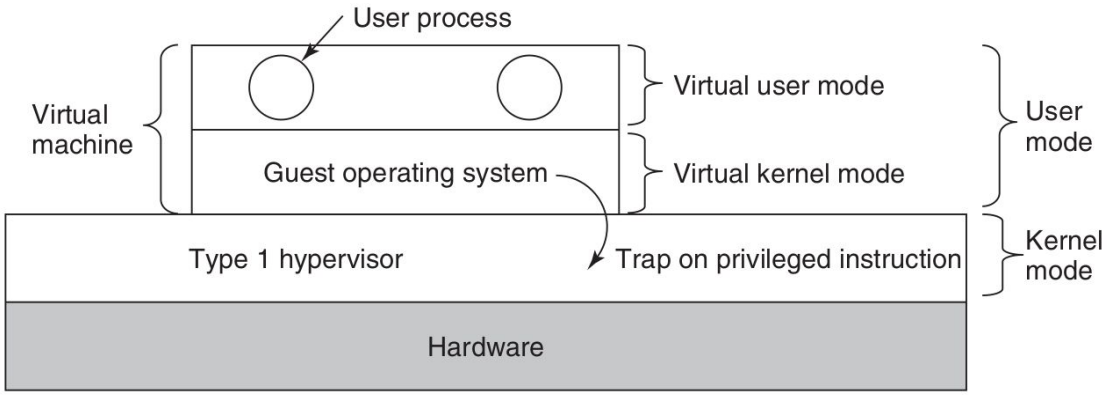
Architektura jest wirtualizowalna wtw. gdy instrukcje wrażliwe są podzbiorem uprzywilejowanych (Popek & Goldberg).

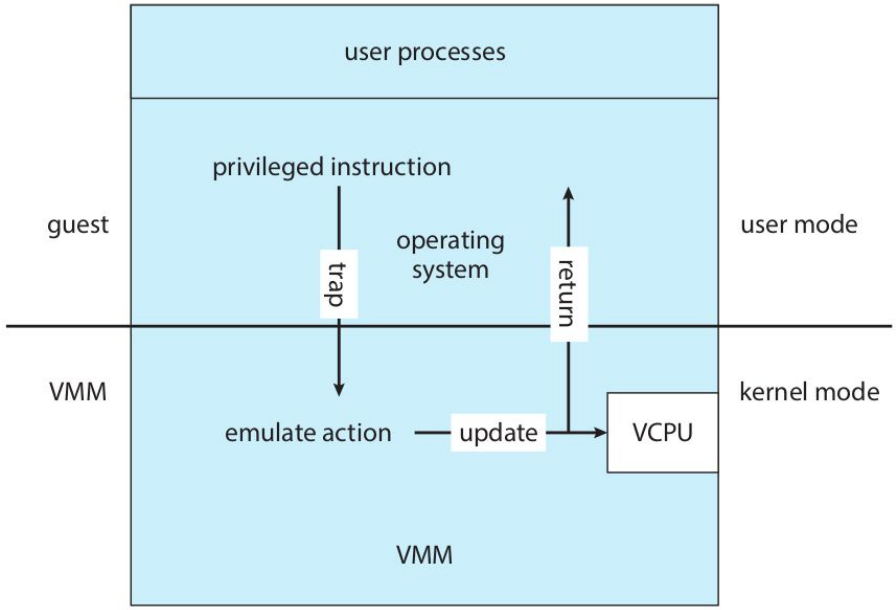
Tak → technika **przechwyć i emuluj** (ang. *trap and emulate*). Nadzorca przechwytuje wyjątki i modyfikuje odpowiedni stan VM.

Nie (x86 pre-VT) → **tłumaczenie kodu** (ang. binary translation). Nadzorca analizuje **bloki podstawowe** (ang. basic block) jądra gościa i przepisuje instrukcje wrażliwe na uprzywilejowane.

Obserwacja: VMM musi się domyślać o co chodzi jądru SO gościa na podstawie informacji skojarzonej z wyjątkiem. Jak wyrażać intencje bezpośrednio? Trzeba zmodyfikować jądro gościa…

Przechwyć i emuluj





Programowe tłumaczenie adresów

Tłumaczenie adresów: guest VA → (guest PA | host VA) → host PA.
Bez wsparcia sprzętowego nie ma rozróżnienia między guest PA i VA.

Jądro SO gościa zmienia odwzorowanie pamięci poprzez zapis do tablicy stron. Instrukcja wrażliwa, ale nie zgłasza wyjątku! Co może zrobić VMM?

VMM wie gdzie znajduje się tablica stron, bo modyfikacja wskaźnika na PT jest instrukcją uprzywilejowaną. Oznacza pamięć zajmowaną przez tablicę stron tylko do odczytu. Przy zapisie przechwytuje wyjątek **błędu strony wywołanego przez gościa** (ang. *guest-induced page fault*), modyfikuje oryginał i **cień tablicy stron** (ang. *shadow page table*).

Aktualizację SPT można robić leniwie → **błąd strony wywołany przez hipernadzorcę** (ang. *hypervisor-induced page fault*).

Przy parawirtualizacji prościej → po modyfikacji PTEs wołamy VMM.

Parawirtualizacja

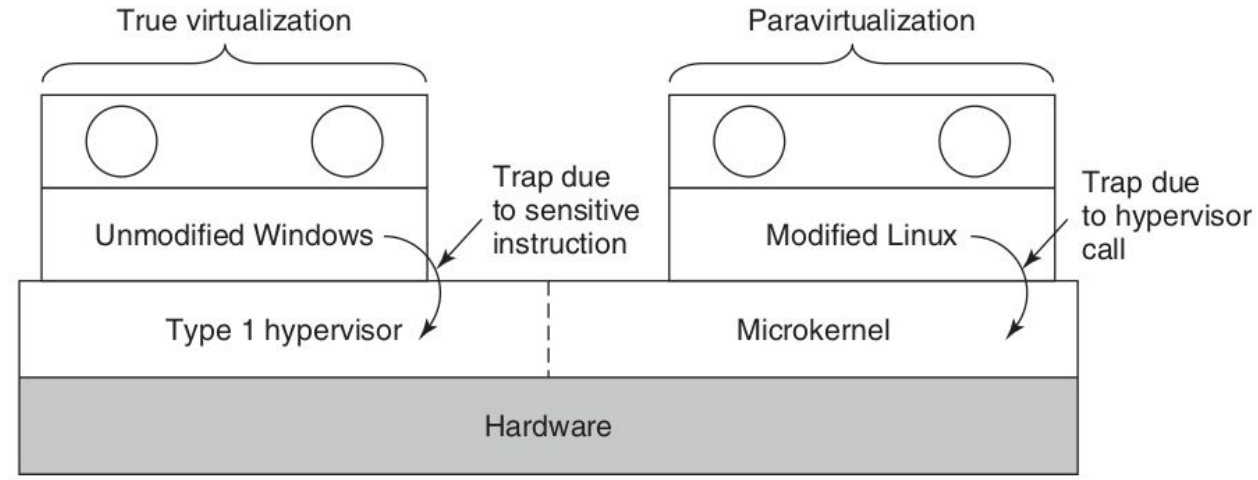
Pełna wirtualizacja → **emulacja** urządzeń I/O, tj. symulacja stanu wew. i buforów kontrolera karty sieciowej / dysku. Komunikacja sieciowa między VMs na jednej maszynie fizycznej realizowana poprzez symulowanie kart sieciowych zamiast z użyciem pamięci dzielonej.

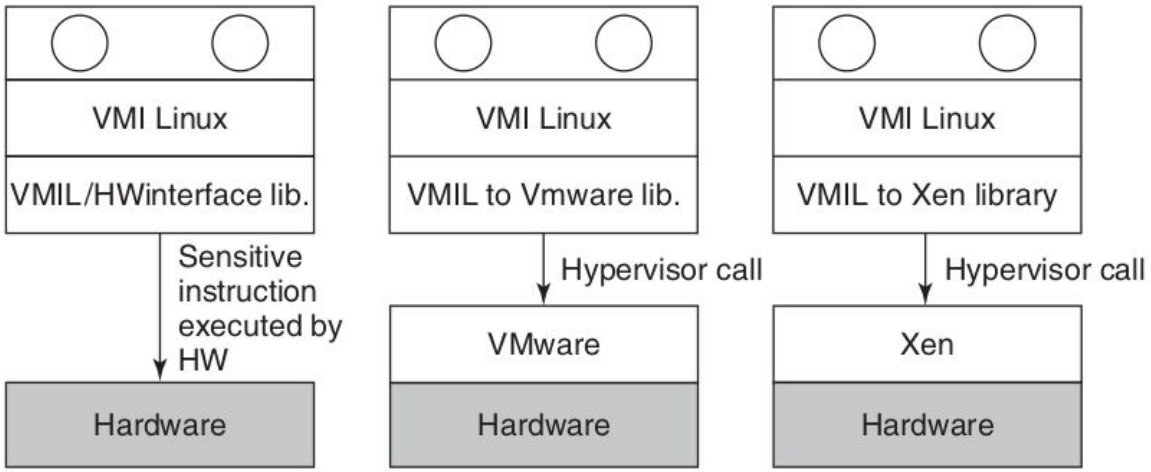
Parawirtualizacja → jądro gościa dostarcza abstrakcji przykrywającej instrukcje wrażliwe i specjalnych sterowników o małym narzucie obsługi. Jeśli jądro gościa wykryje uruchomienie w VM, to przełącza się na korzystanie z **hiperwywołań** i sterowników zwirtualizowanych urządzeń.

Wyjście z VM (ang. *VM exit*) Powrót do hipernadzorcy celem obsługi wyjątku lub przerwania (VMM musi poznać intencję VM). W trakcie obsługi stan wew. CPU zastępowany (tj. TLB, cache, predyktor skoków).

Parawirtualizacja znacząco zmniejsza liczbę wyjść do VMM!

Parawirtualizacja



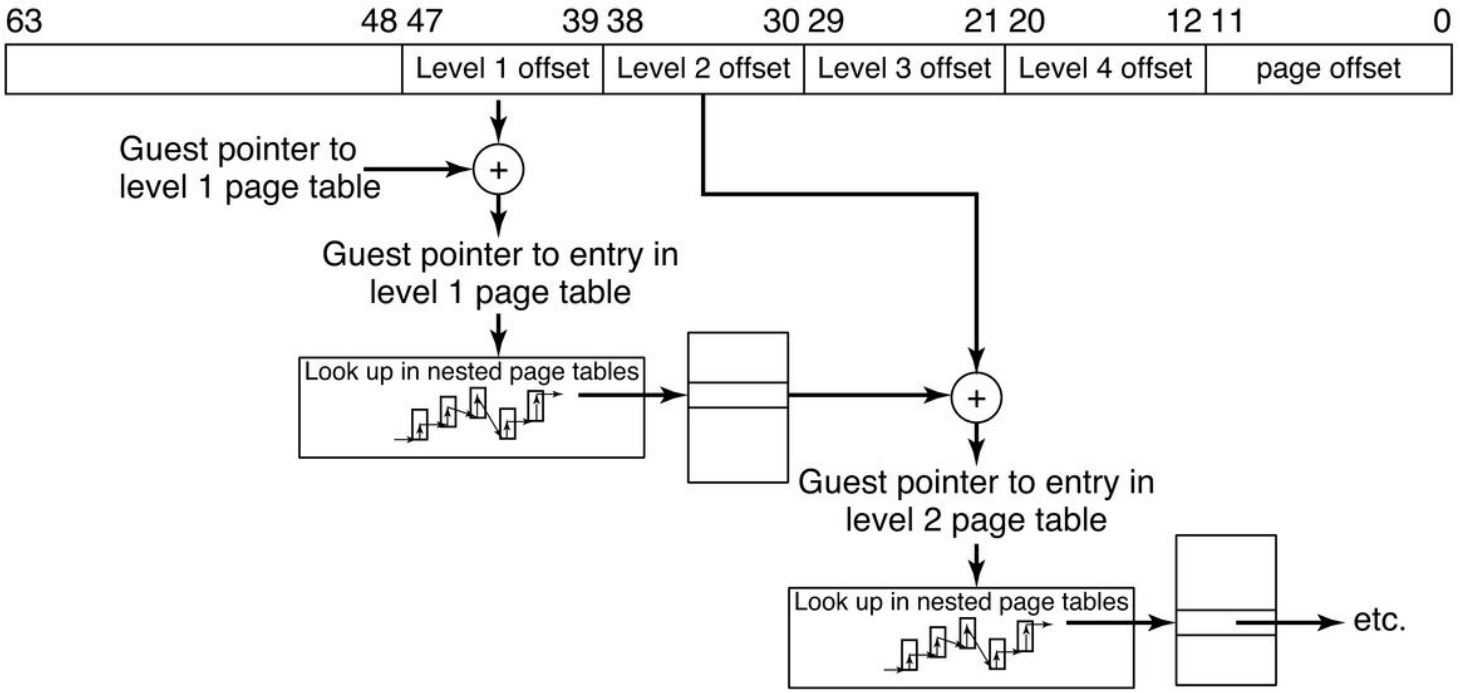


Hiperwywołania XEN: realizacja funkcji

Hypercall	Mode	Description
console_io	PV	I/O to and from Xen virtual console
fpu_taskswitch	PV	modify privileged cr0 register for FPU context switching
iret	PV	implement unvirtualizable interrupt return instruction
mmuext_op	PV	notify hypervisor of page-table updates; TLB flushes
set_gdt	PV	implement unvirtualizable global-descriptor table update
update_descriptor	PV	implement unvirtualizable update of descriptor registers
update_va_mapping	PV	map or unmap a page in virtual memory
event_channel_op	PV/HVM	allocate, bind, use, and close event channels
grant_table_op	PV/HVM	manipulate shared-memory grant tables
memory_op	PV/HVM	adjust physical page reservation; map page into guest
sched_op	PV/HVM	invoke hypervisor scheduler operations such as yield
vcpu_op	PV/HVM	control per-virtual CPU properties such as periodic timers
xen_version	PV/HVM	query Xen version
multicall	PV/HVM	batch a series of hypervisor requests via one hypercall

PV → paravirtualization, HVM → hardware virtual machine.

Sprzętowe tłumaczenie adresów



SLAT (ang. *second level address translation*) – zagnieżdżone tablice stron. Krok guest VA → guest PA robimy z użyciem tablicy stron gościa, analogicznie host VA → host PA z użyciem tablicy stron gospodarza. Tłumaczenie adresów guest VA → host PA automatycznie w sprzęcie!

Zarządzanie pamięcią wirtualną w hipernadzorcy

Obserwacja: wykorzystanie całej pamięci fizycznej VM jest rzadkością.

nadprzydział pamięci (ang. *memory overcommitment*) Dajmy VM więcej pamięci operacyjnej niż mamy. Braki pokryte przestrzenią wymiany VMM.

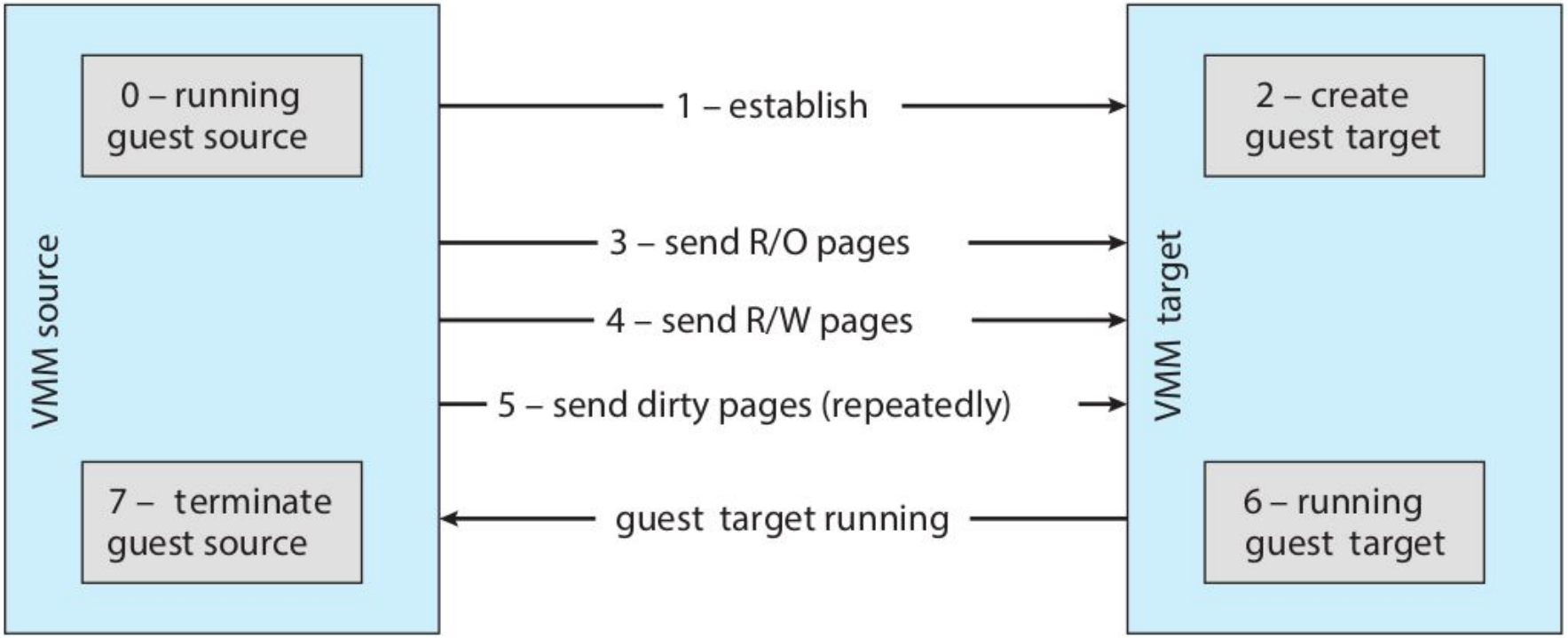
Obserwacja: uruchamiamy tą samą dystrybucję (jądro, narzędzia, itd.) w wielu VM. Możliwość współdzielenia kodu i danych między VM.

deduplikacja (ang. deduplication) Przy wsparciu jądra gościa można wyznaczyć zbiór stron, na których będziemy wyliczać wartość funkcji skrótu. Jeśli takie same, to można współdzielić z *copy-on-write*.

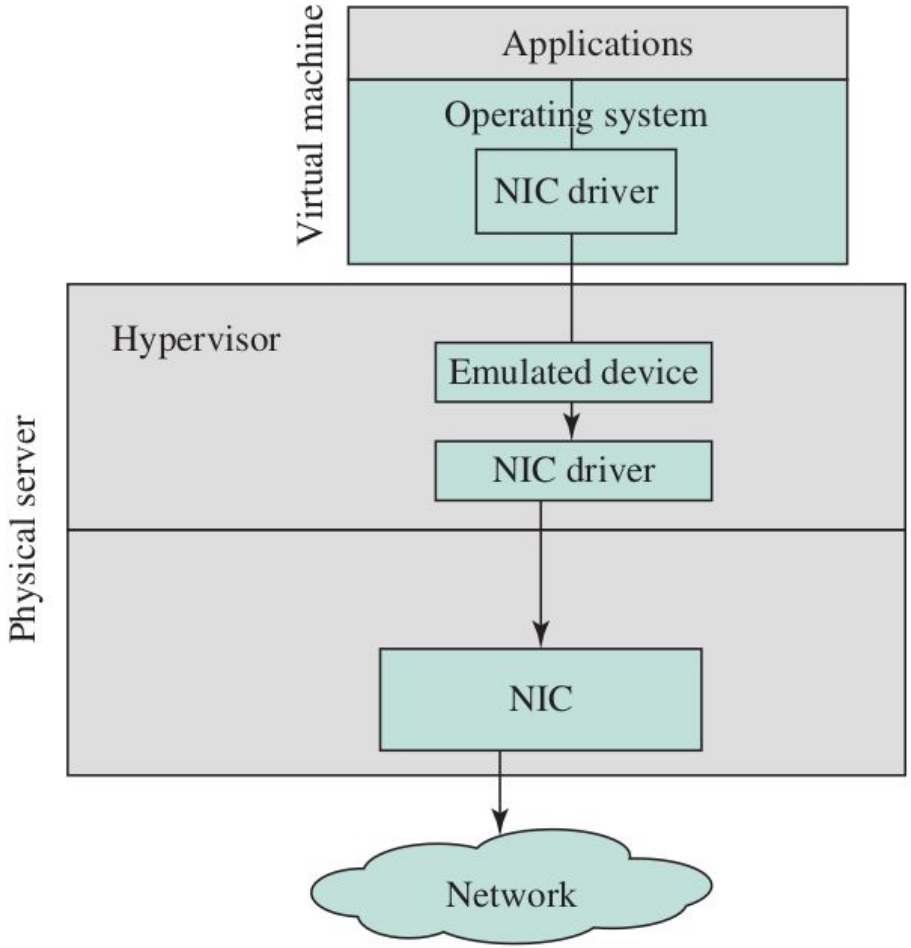
Obserwacja: przy wymianie stron na dysk hipernadzorca nie wie, które strony są cenne z punktu widzenia gościa → możliwe konkurowanie.

balonikowanie (ang. *balooning*) Zabieramy strony jądru gościa pompując “balon” (sterownik) → gość zwalnia / wymienia nieużywane strony.

Migracja maszyn wirtualnych



Nieefektywność I/O



Wirtualizacja urządzeń wejścia-wyjścia

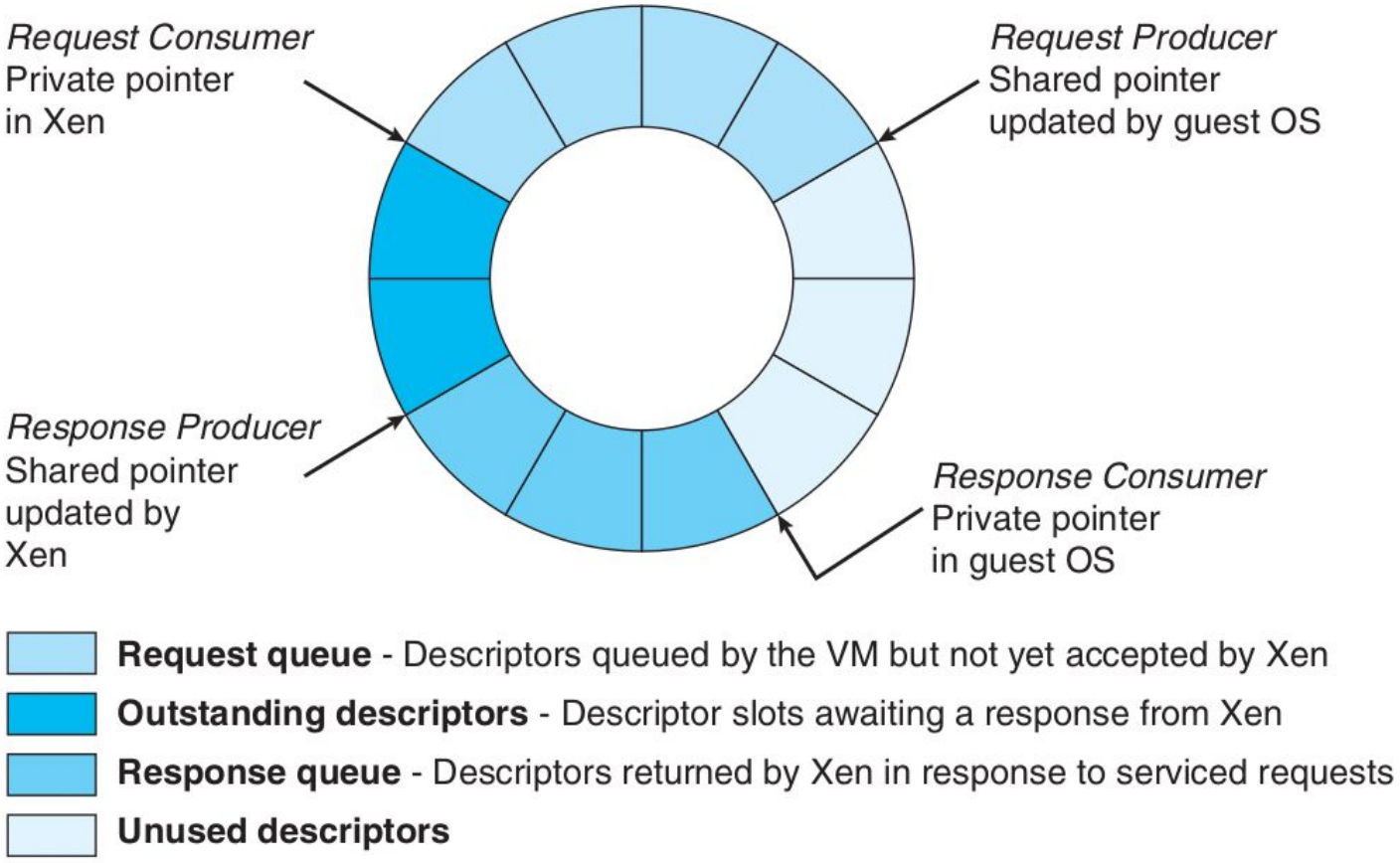
Translacja operacji wej.-wyj. dzięki emulacji jest użyteczne, ale powolne!

Chcemy wirtualizować zasoby sprzętowe: szyny PCI, pamięć / rejestry urządzeń, przerwania, kanały DMA.

I/O MMU Chcemy by DMA korzystało z *guest PA*. Urządzenie ma własne TLB wypełniane przez hipernadzorcę. Możliwość przypisania urządzenia do konkretnej VM (ang. *device pass through*). I/O MMU wprowadza izolację urządzeń → błędne żądanie DMA nie uszkodzi pamięci innej VM. Do którego vCPU trafi przerwanie → **remapowanie przerwań**!

SR-IOV (ang. *Single Root I/O Virtualization*) Sprzętowy mechanizm wbudowany w kartę `PCIe`. Udostępnianie wielu identycznych urządzeń z niezależnymi zasobami, ale tylko przez **VF** (ang. virtual functions). Gospodarz ma pełen dostęp przez **PF** (ang. physical functions).

`VirtIO`: efektywne wirtualizowanie urządzeń



Obsługę urządzenia przez rejestry i DMA zastępujemy współdzielonym fragmentem pamięci. Hiperwywołania do zgłaszania modyfikacji `vring`.

Pytania?