Wirtualizacja

wspomagana sprzętowo - zalety, wady, zagrożenia

Szymon Doroz & Bartosz Janiak & Przemysław Zych

Agenda

- Czym jest wirtualizacja
- Krótka historia
- Wirtualizacja wspomagana sprzętowo
- Prezentacje wybranych maszyn wirtualnych

Czym jest wirtualizacja?

Technika ukrywania charakterystyki zasobów sprzętowych, pozwalająca na bardziej swobodne korzystanie z tych zasobów.

Przykłady

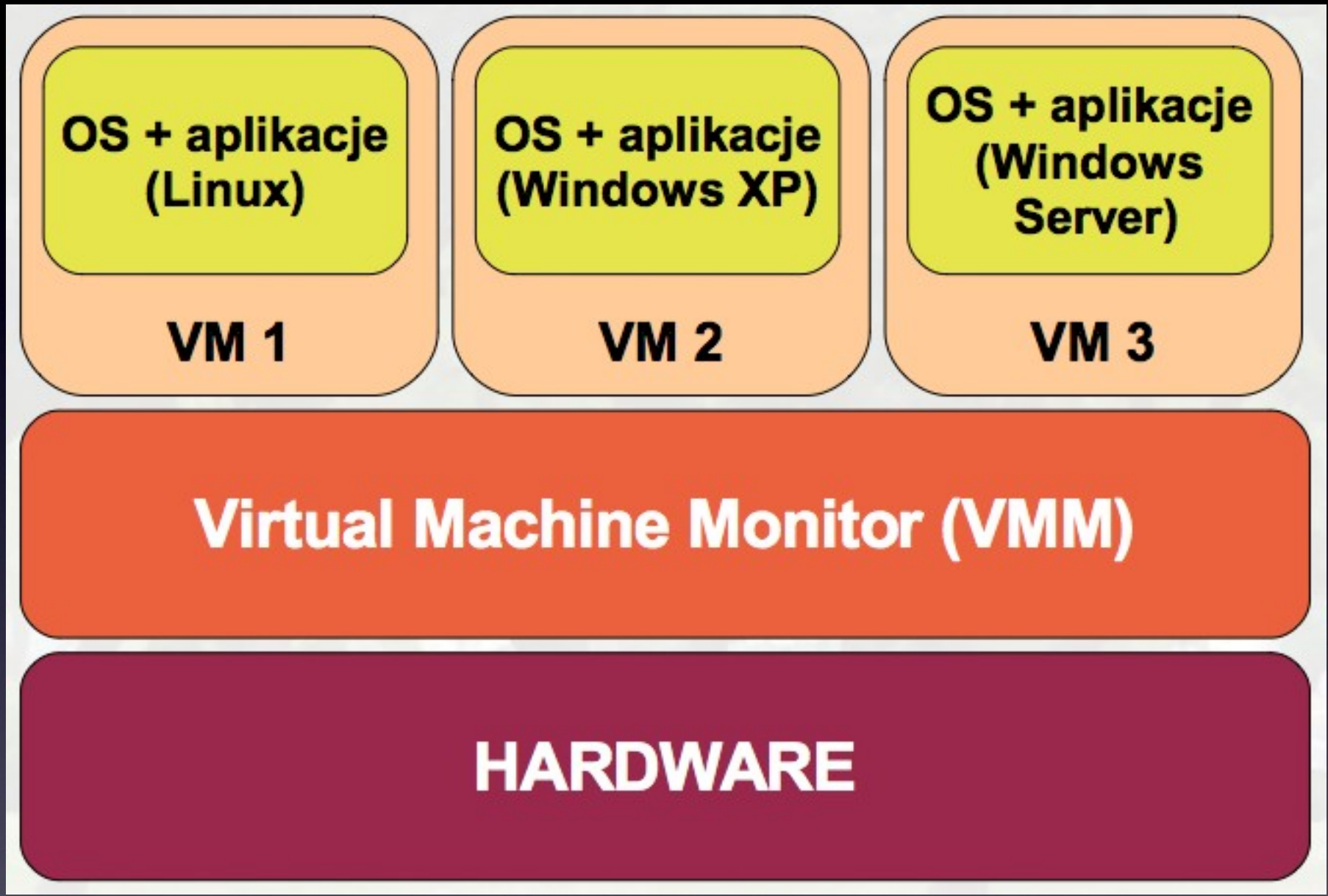
- Pamięć wirtualna
- Partycje dysku twardego
- Maszyna wirtualna Javy
- Wirtualna infrastruktura
- VPN

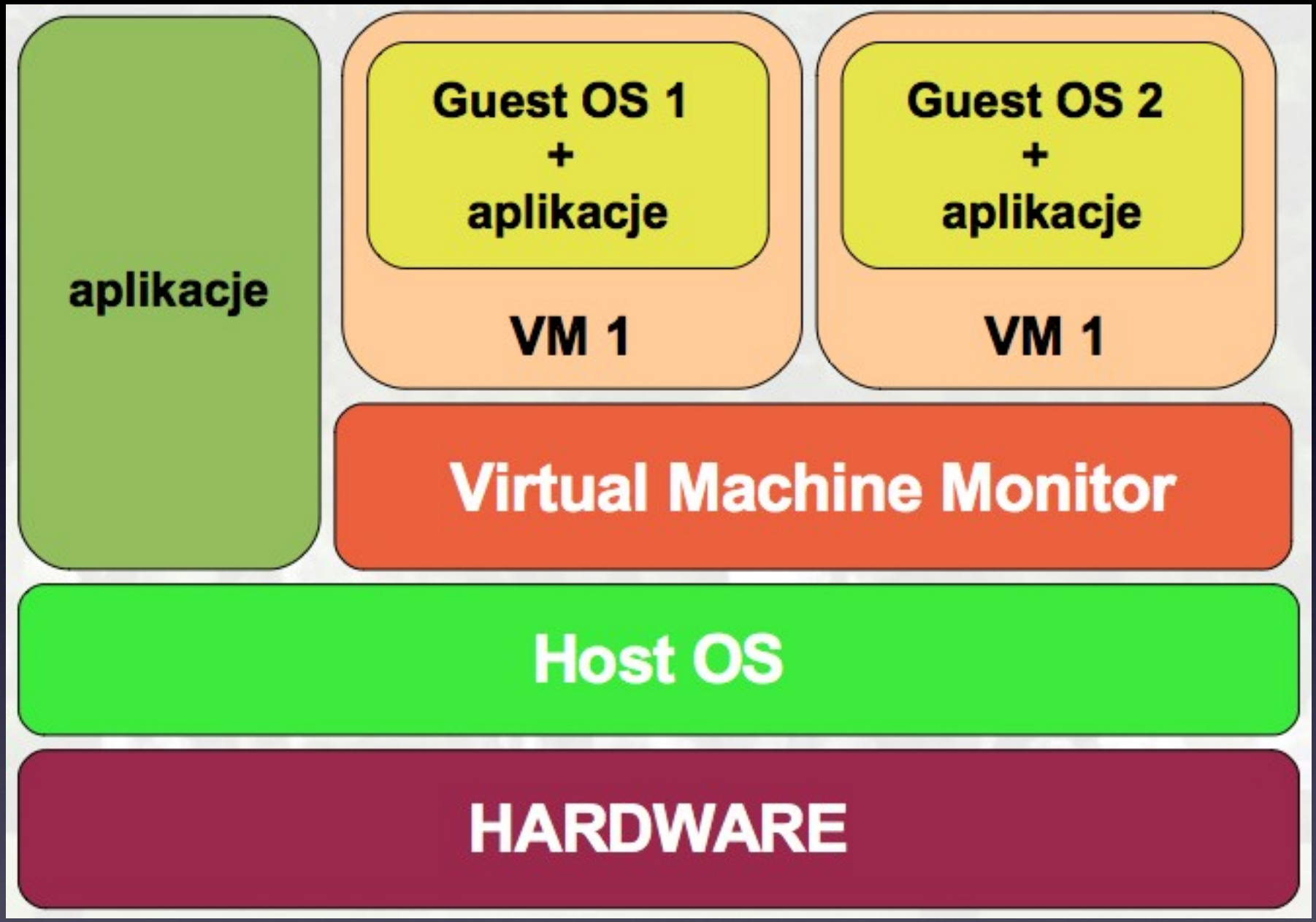
Podstawowe definicje

Virtual Machine Monitor (VMM), inaczej **Hypervisor**. Pośredniczy w dostępie do CPU, pamięci oraz I/O.

Rozdzaje VMM

- Type 1 – native – VMM musi trochę przypominać system operacyjny, maszyny wirtualne są bliżej sprzętu
- Type 2 – hosted – jako aplikacja systemu operacyjnego, VMM jest prostszy, ale wirtualizacja bardziej kosztowna


OS + aplikacje (Linux)
VM 1
OS + aplikacje (Windows XP)
VM 2
OS + aplikacje (Windows Server)
VM 3
Virtual Machine Monitor (VMM)
HARDWARE


aplikacje
Guest OS 1
+
aplikacje
VM 1
Guest OS 2
+
aplikacje
VM 1
Virtual Machine Monitor
Host OS
HARDWARE

Rodzaje wirtualizacji

- Translacja binarna – kod wykonywany jest skanowany, instrukcje wymagające uprzywilejowania są zamieniane na ciągi instrukcji, np. VMWare Player (na maszynie bez wsparcia sprzętowego)

Rodzaje wirtualizacji

- Parawirtualizacja – źródła hosta i gościa są tak modyfikowane, żebym VMM mógł wykonywać kod gościa natywnie bez translacji, np. Xen

Rodzaje wirtualizacji

- Wspomagana sprzętowo – architektura zawiera wsparcie dla wirtualizacji, np. udostępniany jest specjalny tryb pracy procesora dla gościa

Krótka historia

- Prace rozpoczęły się około 50 lat temu
- Obecne pomysły są niewiele nowsze (sprzed 45 lat)
- Time Sharing
- VM/370 na System/370
- Zapotrzebowanie na wirtualizację na x86

Time Sharing

- Wprowadzony przez IBM
- Inicjalna idea wirtualizacji
- Wielokrotne „kopie” hardware'u uruchamiane jako sesje wirtualne; jedynie monitor maszyny wirtualnej (VMM) operował bezpośrednio na hardware'rze

VM/370

- System operacyjny zaprojektowany specjalnie z myślą o wirtualizacji
- Jeden z bardziej udanych produktów IBM

Motywacje dla wirtualizacji x86

- Stopniowe odchodzenie od mainframe'ów
- Popularność systemów rozproszonych
- Wykorzystanie architektury klient – serwer
- Konieczność obniżenia kosztów zarządzania dużą ilością serwerów
- Niskie wykorzystanie pojedynczych maszyn

Kryteria Popka-Goldberga

Oczekiwania wobec wirtualizacji:

- Równoważność
- Kontrola zasobów
- Wydajność

Skuteczność wg Popka-Goldberga

W każdej architekturze można wyodrębnić grupy instrukcji:

- Wrażliwe – w trakcie wykonywania mogą zmieniać konfigurację zasobów systemowych lub działanie jest zależne od konfiguracji systemu
- Uprzywilejowane – ich efektem jest przerwanie lub wywołanie systemowe w trybie użytkownka lub ich brak w trybie jądra

Twierdzenie Popka-Goldberga

Jeśli zbiór instrukcji wrażliwych jest podzbiorem zbioru instrukcji uprzywilejowanych, to wirtualna maszyna może zostać skonstruowana.

Problemy z architekturą x86

- Pierwotna architektura x86 nie spełniała kryteriów Popka-Goldberga
- Bardzo trudno było stworzyć maszynę wirtualną
- Stronicowanie, mechanizm zabezpieczania, segmentacja, w założeniu miały być zarządzane tylko przez jeden system operacyjny
- Jedynym rozwiązaniem było programowe ominięcie różnych problemów powstających przy wirtualizacji

Problemy z architekturą x86

- Instrukcje, które pozwalają na odczytanie rejestrów systemowych w trybie użytkownika
- Nie ma sprzętowych mechanizmów wykrywania wszystkich instrukcji, które nie działają poprawnie w wirtualnym środowisku
- Należy tak przerywać wykonanie kodu (tworząc breakpointy), żeby nadzorca mógł odpowiednio reagować
- Modyfikacje jakich użyjemy nie mogą zostać wykryte przez gościa

Problemy z architekturą x86

- Konieczność ustawiania breakpointów, żeby nigdy nie dopuścić do wykonania kodu, który nie był jeszcze sprawdzony
- Kod może zaglądać do już przeskanowanej części programu
- Kod może modyfikować przeskanowaną część
- Istnieją sztuczki, dzięki którym tworzy się stronę z kodem wykonywalnym, którego nie można odczytać ani zapisać

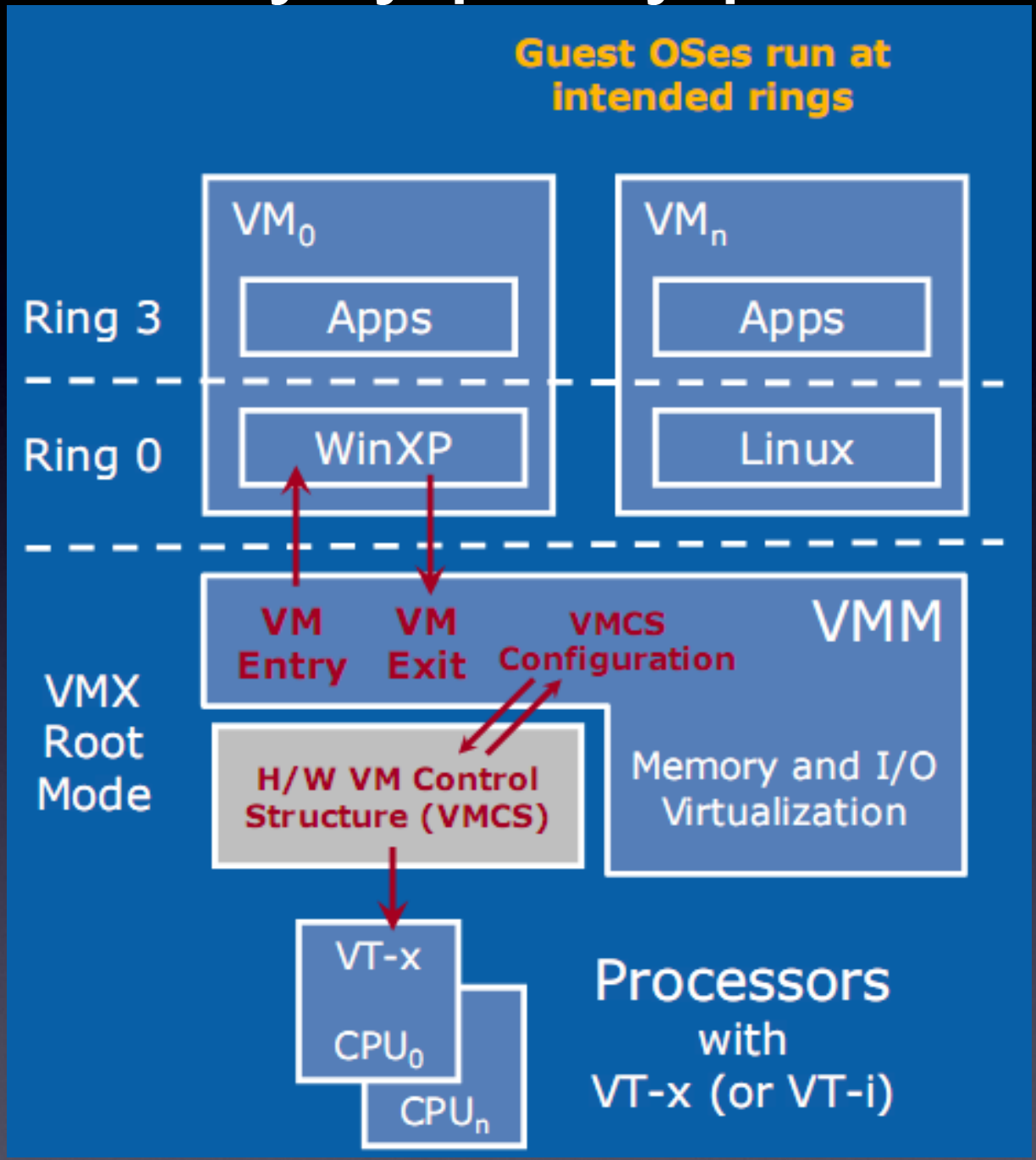
Wirtualizacja sprzętowa na architekturze x86

- Intel VT
- VT – x („Vanderpool”)
- VT – d
  - Bezpośredni dostęp do urządzeń peryferyjnych
- VT – c
  - Akceleracja wejścia / wyjścia
- AMD-V („Pacifica”)
- IOMMU

VT-x – główne cechy

- Nowe rozkazy procesora
- VMPTRLD, VMPTRST, VMCLEAR, VMREAD, VMWRITE, VMCALL, VMLAUNCH, VMRESUME, VMXOFF, VMXON
- Dwa tryby pracy CPU
- VMX root mode
- VMX non-root mode
- Dodatkowa struktura w pamięci - VMCS
- Cel
- Ułatwienie napisania lekkiego VMM

VT-x – tryby pracy procesora



Ułatwienia związane z użyciem VT-x

- Kompresja przestrzeni adresowej
- VT-x pozwala na zmianę liniowej przestrzeni adresowej przy każdym przejściu pomiędzy root i non-root mode
- Ring aliasing
- VT-x umożliwia pracę systemu gościa w środowisku nieodróżnialnym od uprzywilejowanego Ring 0, nawet gdy VM uruchamiana jest na poziomie Ring 3.
- SYSENTER i SYSEXIT nie muszą być emulowane.

Ułatwienia związane z użyciem VT-x

- Nonfaulting Access to Privileged State
- Architektura x86 zawiera zestaw instrukcji, które mogą zostać wykonane poprawnie jedynie w Ring 0, lecz nie powodują błędu w Ring 3.
- VT-x pozwala VMM zdefiniować wywołania których z tych instrukcji mają prowadzić do przekazania kontroli do VMM (tj. przejścia do root-mode)

Ułatwienia związane z użyciem VT-x

- Wirtualizacja przerwań
- System gościa w non-root mode może swobodnie modyfikować maskowanie przerwań, bez wpływu na maskowanie hosta.
- Gdy VMM ma do dostarczenia przerwanie do gościa, ustawia odpowiednią flagę. Gdy gość będzie gotowy do odebrania przerwania (np. zmieni maskowanie) procesor automatycznie opuści non-root mode.

Ułatwienia związane z użyciem VT-x

- Dostęp do ukrytych elementów stanu procesora
- Przykładowo: cache dla deskryptorów rejestrów segmentowych nie jest dostępny dla programów.
- VMM musi mieć kontrolę nad tym cache z uwagi na zmianę przestrzeni adresowej dla gościa.
- Proste rozwiązanie w VT-x: ukryte elementy stanu są przechowywane w VMCS i procesor odtwarza je przy każdym przejściu pomiędzy root i non-root mode.

Ułatwienia związane z użyciem VT-x

- Częsty dostęp do uprzywilejowanych zasobów
- Wymaga wymuszenia błędu i jego przechwycenia przez klasycznego hypervisora – to wpływa na wydajność
- Przykład: TPR (Task Priority Register)
- VT-x wybudza VMM tylko wtedy, gdy jest to niezbędne: odpowiednie pole w VMCS określa poniżej jakiego poziomu musi spaść TPR, by doszło do wybudzenia.

VT-x - wsparcie sprzętowe

- Pentium 4 662 oraz 672
- Pentium Extreme Edition 955 oraz 965
- Pentium D 920-960 poza 945, 925, 915
- Core Solo U1000
- Core Duo T2300, T2400, T2500, T2600, T2700, L2000, U2000
- Core 2 Solo
- Core 2 Duo (wszystkie poza E8190, E7xxx poza 7300), E4xxx, T5200-T5550, T5750
- Core 2 Quad (wszystkie poza Q8200)
- Core 2 Extreme Duo oraz Quad
- Xeon 3000, 5000, 7000

Software korzystający z VT-x

- Projekty open source
- KVM
- VirtualBox
- Xen od wersji 3.0.2
- Projekty komercyjne
- VMware Workstation, Server, Fusion
- Parallels Workstation, Desktop
- Microsoft Hyper-V
- ...

VT-x, AMD-V : wydajność

- Brak niezależnych, dokładnych benchmarków
- Aplikacje nieuprzywilejowane na poziomie 90% do 97% wydajności hosta (dane VMware)
- Gorzej z Ring 0
- Ładowanie Ubuntu na VirtualBox
- Z VT-x: 36s
- Bez: 28s [za tombuntu.com]
- Twórcy niektórych VM zalecają wyłączenie VT-x, jeśli zależy nam na wydajności

Dalszy rozwój wsparcia sprzętowego dla wirtualizacji

- EPT/NPT (VT-x drugiej generacji)
- VT-d
- IOMMU

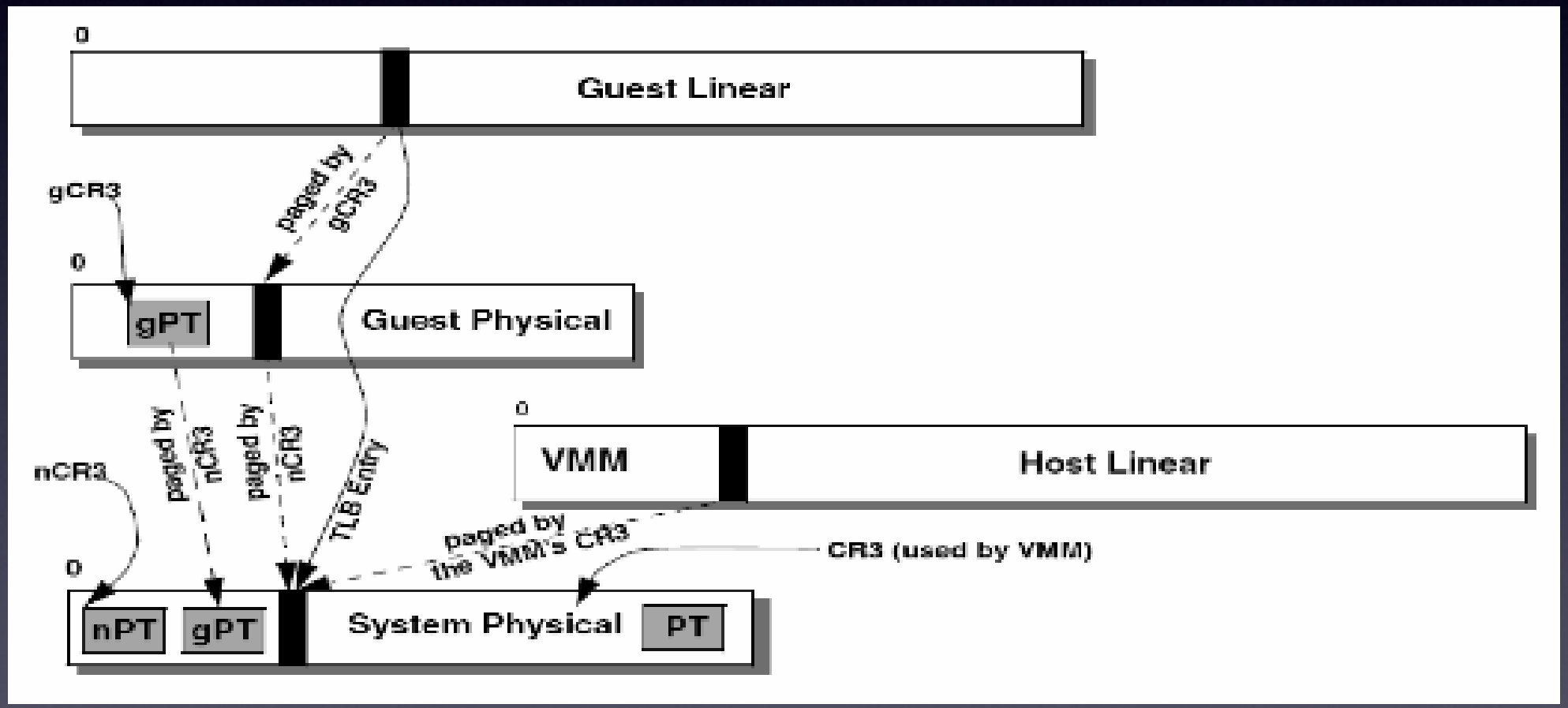
Software Page Table Virtualization

- Shadow Paging
- Tablica stron uaktualniana przy modyfikowaniu tablicy stron gościa

- Virtual TLB
- Tablica stron uaktualniana przy odwoływaniu się do nieistniejącego adresu oraz kasowaniu wpisów w tablicy stron gościa

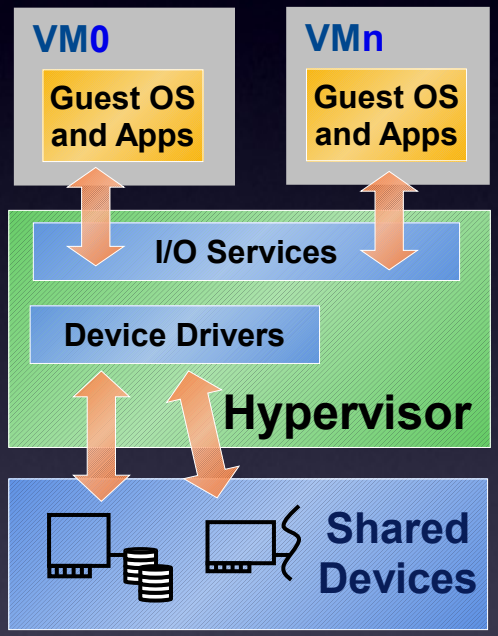
Hardware Page Table Virtualization

- Intel Extended Page Tables (EPT)
- AMD Nested Page Tables (NPT)



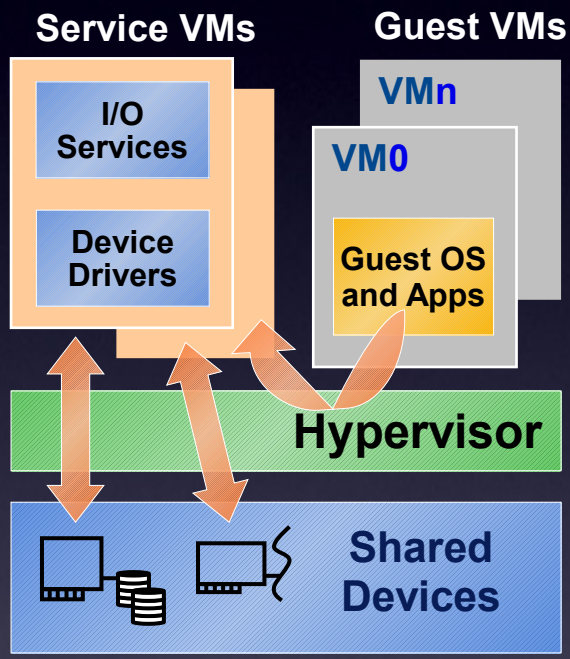
VM a urządzenia peryferyjne

Monolithic Model



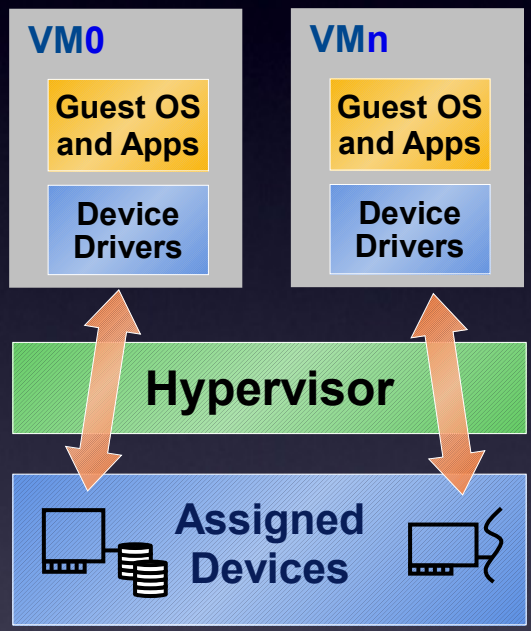
- Pro: Higher Performance
- Pro: I/O Device Sharing
- Pro: VM Migration
- Con: Larger Hypervisor

Service VM Model



- Pro: High Security
- Pro: I/O Device Sharing
- Pro: VM Migration
- Con: Lower Performance

Pass-through Model



- Pro: Highest Performance
- Pro: Smaller Hypervisor
- Pro: Device assisted sharing
- Con: Migration Challenges

Direct I/O

- Direct Assigment
- VT-d Interrupt Remapping
- VT-d DMA Remapping
- AMD-V
- IO MMU – I/O Memory Managment Unit)

Blue Pill

- Rootkit dla Windows Vista stworzony przez Joannę Rutkowską
- Implementacja bazuje na AMD-V
- Założenie, że idealna wirtualizacja jest niewykrywalna przez system goszczony
- W założeniu całkowite i niewykrywalne przejęcie kontroli nad zainfekowaną maszyną
- Wirtualizacja w locie

Blue Pill – możliwości wykrycia

- Pomiar czasu trwania instrukcji
- Pomiar ilości dostępnych zasobów
- Red Pill
- Instrukcja SIDT
- Dostęp do adresu tabeli deskryptorów przerywań z poziomu ring3

SubVirt

- Stworzony przez Microsoft i Uniwersytet w Michigan
- Implementacja pozwala na:
- Logowanie klawiatury
- Przeglądanie dysku w poszukiwaniu informacji o użytkowniku
- Unieszkodliwianie programów antywirusowych

Zapobieganie

- Stworzenie sprzętowego wykrywania VM rootkitów
- Bootowanie z niezainfekowanego źródła np. CD-ROM, pamięci flash, przez sieć
- Wyłączenie sprzętowego wspomagania wirtualizacji w BIOSie

Pararells Desktop

VirtualBox

- Open Source (GPL)
- Potrafi uruchomić się z poziomu Windows, Linux, Macintosh oraz OpenSolaris
- Wsparcie dla Windows (NT 4.0, 2000, XP, Server 2003, Vista), DOS/Windows 3.x, Linux (2.4 and 2.6), Solaris and OpenSolaris, and OpenBSD

VirtualBox - możliwości

- Snapshoty
- Symulacja sieci, portów USB, portów szeregowych
- Przekierowywanie portów
- Współdzielone foldery
- Zdalny pulpit
- Obrazy dysków twardych (stałego i zmiennego rozmiaru, różnicowe oraz fizyczne)

Dziękujemy za uwagę

Pytania?

Wirtualizacja

wspomagana sprzętowo - zalety, wady, zagrożenia

Szymon Doroz & Bartosz Janiak & Przemysław Zych

Agenda

- Czym jest wirtualizacja
- Krótka historia
- Wirtualizacja wspomagana sprzętowo
- Prezentacje wybranych maszyn wirtualnych

Czym jest wirtualizacja?

Technika ukrywania charakterystyki zasobów sprzętowych, pozwalająca na bardziej swobodne korzystanie z tych zasobów.

Przykłady

- Pamięć wirtualna
- Partycje dysku twardego
- Maszyna wirtualna Javy
- Wirtualna infrastruktura
- VPN

Podstawowe definicje

Virtual Machine Monitor (VMM), inaczej **Hypervisor**. Pośredniczy w dostępie do CPU, pamięci oraz I/O.

Rozdzaje VMM

- Type 1 – native – VMM musi trochę przypominać system operacyjny, maszyny wirtualne są bliżej sprzętu
- Type 2 – hosted – jako aplikacja systemu operacyjnego, VMM jest prostszy, ale wirtualizacja bardziej kosztowna

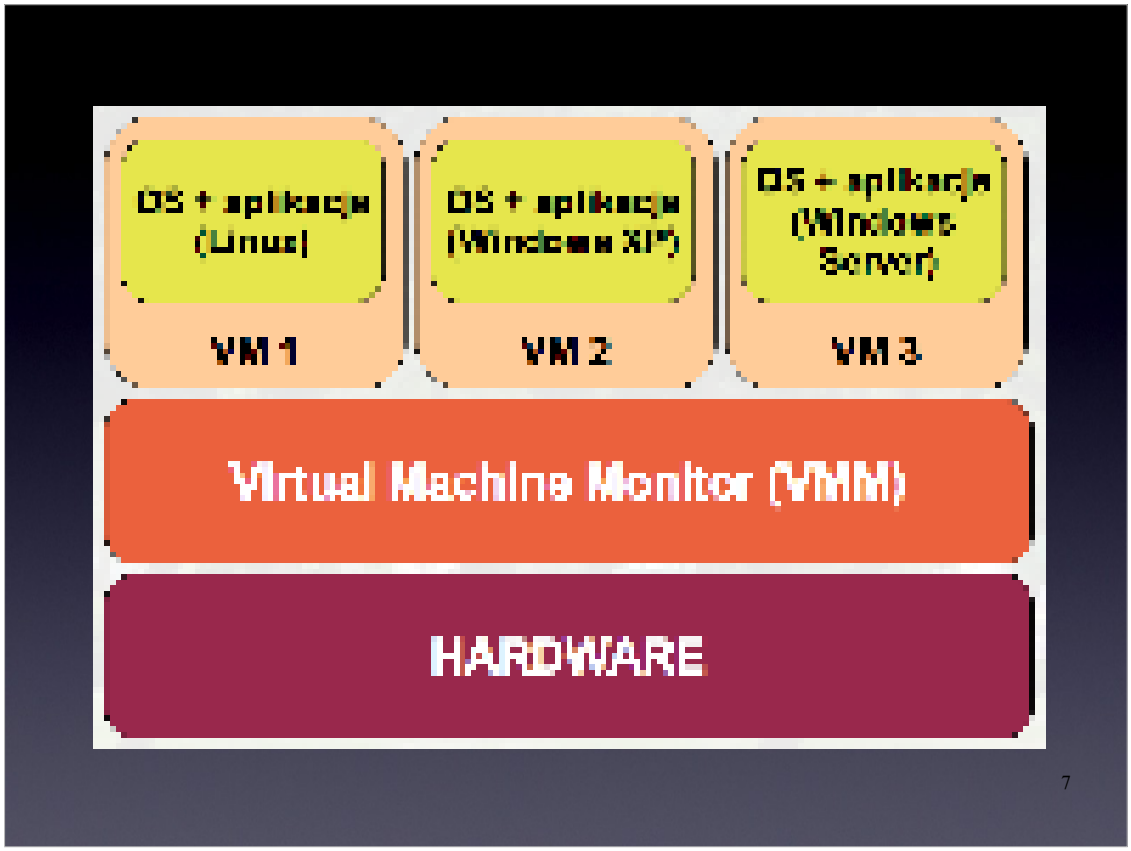


Diagram dla typu pierwszego (native)

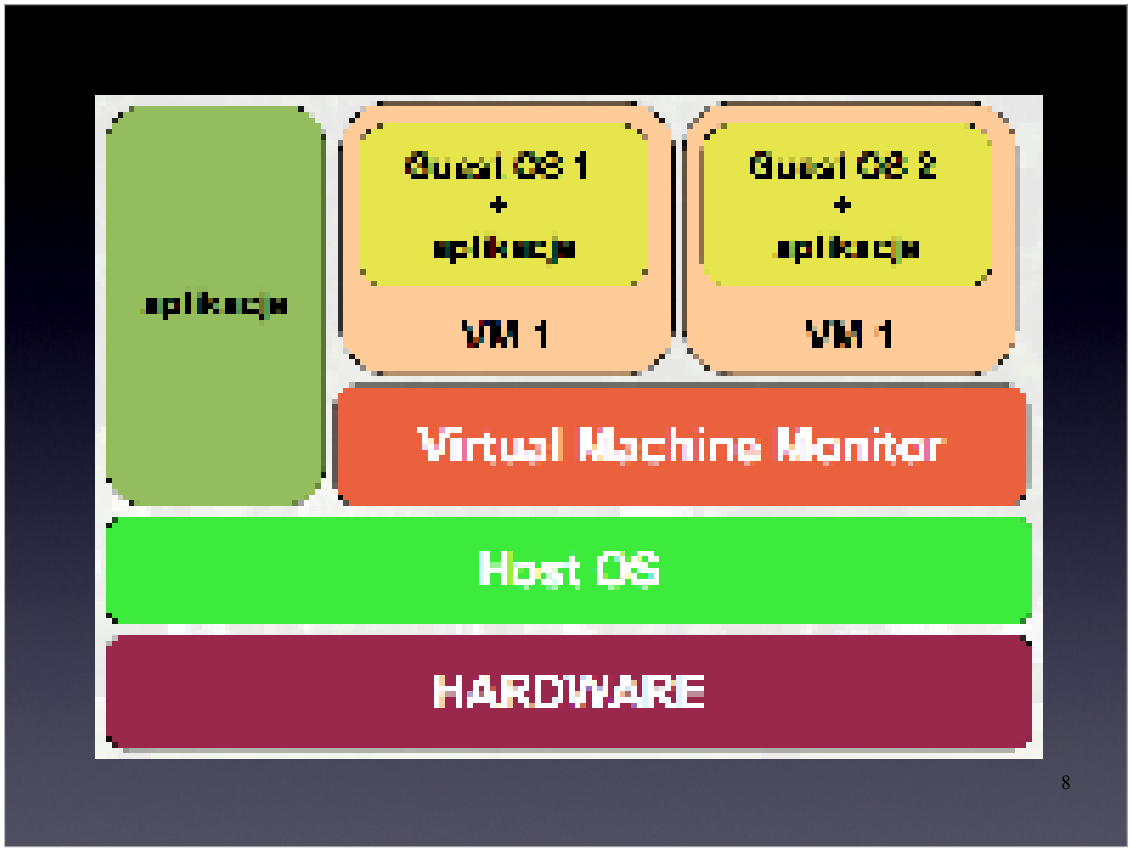


Diagram dla typu drugiego (hosted)

Rodzaje wirtualizacji

Translacja binarna – kod wykonywany jest skanowany, instrukcje wymagające uprzywilejowania są zamieniane na ciągi instrukcji, np. VMWare Player (na maszynie bez wsparcia sprzętowego)

Taka translacja wiąże się często ze sporym narzutem

Rodzaje wirtualizacji

Parawirtualizacja – źródła hosta i gościa są tak modyfikowane, żebym VMM mógł wykonywać kod gościa natywnie bez translacji, np. Xen

Dość inwazyjna metoda wirtualizacji, oparta na uprzednim dostosowywaniu systemów

Rodzaje wirtualizacji

Wspomagana sprzętowo – architektura zawiera wsparcie dla wirtualizacji, np. udostępniany jest specjalny tryb pracy procesora dla gościa

Krótka historia

Prace rozpoczęły się około 50 lat temu

Obecne pomysły są niewiele nowsze (sprzed 45 lat)

- Time Sharing
- VM/370 na System/370
- Zapotrzebowanie na wirtualizację na x86

12

Idea wirtualizacji nie jest wcale nowa. Przeciwnie – istnieje od około 50 lat, zaś obecnie używane techniki pojawiły się około 45 lat temu. Wirtualizacja jest dojrzałą technologią w świecie IT, mającą akceptację zarówno na rynku mainframe'ów jak i ostatnio małych serwerów.

Historycznie IBM zaczal badac techniki wirtualizacji w latach 60, czego efektem był time-sharing.

Badania doprowadziły do wyprodukowania VM/370, który obsługiwał maszyny System/370.

Time Sharing

- Wprowadzony przez IBM
- Inicjalna idea wirtualizacji
- Wielokrotne „kopie” hardware'u uruchamiane jako sesje wirtualne; jedynie monitor maszyny wirtualnej (VMM) operował bezpośrednio na hardware'rze

Time sharing – ponieważ pierwsze mainframe'my i minikomputery były nieprzyzwoicie drogie, nie było możliwości żeby pojedynczy użytkownik miał wyłączny dostęp do maszyny w trybie interaktywnym . Ponieważ komputery w trybie interaktywnym marnują zasoby w oczekiwaniu na input użytkownika, wymyślono, żeby wiele użytkowników mogło dzielić maszynę poprzez alokowanie jałowego czasu jednego użytkownika do obsługiwania innych.

VM/370

System operacyjny zaprojektowany specjalnie z myślą o wirtualizacji

- Jeden z bardziej udanych produktów IBM

Motywacje dla wirtualizacji x86

- Stopniowe odchodzenie od mainframe'ów
- Popularność systemów rozproszonych
- Wykorzystanie architektury klient – serwer
- Konieczność obniżenia kosztów zarządzania dużą ilością serwerów
- Niskie wykorzystanie pojedynczych maszyn

15

Ze względu na taniejący sprzęt i odchodzenie od mainframe'ów, wirtualizacja przestała być topowym tematem w latach 80-90. Systemy rozproszone i architektura klient serwer była wówczas modna, wiele firm zaoszczędziło spore sumy pieniędzy na przejściu na niskokosztowne systemy oparte na architekturze x86.

Kryteria Popka-Goldberga

Oczekiwania wobec wirtualizacji:

- Równoważność
- Kontrola zasobów
- Wydajność

Skuteczność wg Popka-Goldberga

W każdej architekturze można wyodrębnić grupy instrukcji:

- Wrażliwe – w trakcie wykonywania mogą zmieniać konfigurację zasobów systemowych lub działanie jest zależne od konfiguracji systemu
- Uprzywilejowane – ich efektem jest przerwanie lub wywołanie systemowe w trybie użytkownka lub ich brak w trybie jądra

Twierdzenie Popka-Goldberga

Jeśli zbiór instrukcji wrażliwych jest podzbiorem zbioru instrukcji uprzywilejowanych, to wirtualna maszyna może zostać skonstruowana.

Problemy z architekturą x86

Pierwotna architektura x86 nie spełniała kryteriów Popka-Goldberga

- Bardzo trudno było stworzyć maszynę wirtualną
- Stronicowanie, mechanizm zabezpieczania, segmentacja, w założeniu miały być zarządzane tylko przez jeden system operacyjny
- Jedynym rozwiązaniem było programowe ominięcie różnych problemów powstających przy wirtualizacji

Problemy z architekturą x86

Instrukcje, które pozwalają na odczytanie rejestrów systemowych w trybie użytkownika

- Nie ma sprzętowych mechanizmów wykrywania wszystkich instrukcji, które nie działają poprawnie w wirtualnym środowisku
- Należy tak przerywać wykonanie kodu (tworząc breakpointy), żeby nadzorca mógł odpowiednio reagować
- Modyfikacje jakich użyjemy nie mogą zostać wykryte przez gościa

Problemy z architekturą x86

Konieczność ustawiania breakpointów, żeby nigdy nie dopuścić do wykonania kodu, który nie był jeszcze sprawdzony

- Kod może zaglądać do już przeskanowanej części programu
- Kod może modyfikować przeskanowaną część
- Istnieją sztuczki, dzięki którym tworzy się stronę z kodem wykonywalnym, którego nie można odczytać ani zapisać

Wirtualizacja sprzętowa na architekturze x86

- Intel VT
  - VT – x („Vanderpool”)
  - VT – d
    - Bezpośredni dostęp do urządzeń peryferyjnych
  - VT – c
    - Akceleracja wejścia / wyjścia
- AMD-V („Pacifica”)
- IOMMU

VT – x: podstawowa funkcjonalność, wsparcie dla wirtualizacji sprzętowej. Dostępny w procesorach Intela od 2005 roku.
VT – d: prezentowany na konferencjach, jeszcze niedostępny w produkowanych procesorach
VT – c: w fazie projektowania

VT – i: odpowiednik VT – x na architekturę Itanium

AMD-V: odpowiednik technologii Intela, podobne możliwości ale niekompatybilna implementacja

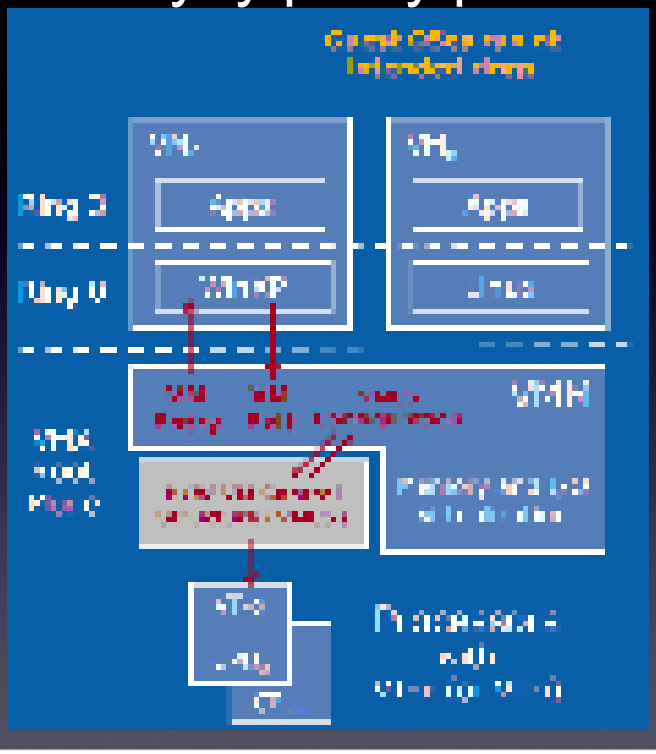
VT-x – główne cechy

- Nowe rozkazy procesora

VMPTRLD, VMPTRST, VMCLEAR, VMREAD, VMWRITE, VMCALL, VMLAUNCH, VMRESUME, VMXOFF, VMXON

- Dwa tryby pracy CPU
- VMX root mode
- VMX non-root mode
- Dodatkowa struktura w pamięci - VMCS
- Cel
- Ułatwienie napisania lekkiego VMM

VT-x – tryby pracy procesora



W ramach trybu non-root procesor pozwala na wykonywanie wielu operacji jądra systemu bez potrzeby przechodzenie do warstwy VMM.
W szczególności non-root mode zawiera niezależne od „prawdziwych” (tj. z root mode) tryby uprzywilejowania ring 3..0.

Ułatwienia związane z użyciem VT-x

- Kompresja przestrzeni adresowej
- VT-x pozwala na zmianę liniowej przestrzeni adresowej przy każdym przejściu pomiędzy root i non-root mode
- Ring aliasing
- VT-x umożliwia pracę systemu gościa w środowisku nieodróżnialnym od uprzywilejowanego Ring 0, nawet gdy VM uruchamiana jest na poziomie Ring 3.
- SYSENTER i SYSEXIT nie muszą być emulowane.

25

Kompresja przestrzeni adresowej

Software'owa VM musi sama chronić swoją własną pamięć od dostępu z gościa – potrzebna translacja dostępów do pamięci.

Ring aliasing

Aktualny poziom uprzywilejowania jest dostępny w rejestrze CS, co nie pozwala na proste ukrycie faktu uruchamiania systemu gościa w Ring 3 bez użycia wirtualizacji sprzętowej.

Ułatwienia związane z użyciem VT-x

Nonfaulting Access to Privileged State

- Architektura x86 zawiera zestaw instrukcji, które mogą zostać wykonane poprawnie jedynie w Ring 0, lecz nie powodują błędu w Ring 3.
- VT-x pozwala VMM zdefiniować wywołania których z tych instrukcji mają prowadzić do przekazania kontroli do VMM (tj. przejścia do root-mode)

26

Przykładowe instrukcje spełniające ten warunek: dostęp do rejestrów GDTR, IDTR, LDTR i TR.

Ułatwienia związane z użyciem VT-x

Wirtualizacja przerwań

- System gościa w non-root mode może swobodnie modyfikować maskowanie przerwań, bez wpływu na maskowanie hosta.
- Gdy VMM ma do dostarczenia przerwanie do gościa, ustawia odpowiednią flagę. Gdy gość będzie gotowy do odebrania przerwania (np. zmieni maskowanie) procesor automatycznie opuści non-root mode.

27

Nie można pozwolić, by gość faktycznie modyfikował maskowanie przerwań procesora. Bez wsparcie sprzętowego trzeba więc wirtualizować zmiany tych ustawień (np. translacja binarna). Wiele systemów operacyjnych wykonuje tą operację bardzo często, to może drastycznie obniżyć wydajność.

Ułatwienia związane z użyciem VT-x

Dostęp do ukrytych elementów stanu procesora

- Przykładowo: cache dla deskryptorów rejestrów segmentowych nie jest dostępny dla programów.
- VMM musi mieć kontrolę nad tym cache z uwagi na zmianę przestrzeni adresowej dla gościa.
- Proste rozwiązanie w VT-x: ukryte elementy stanu są przechowywane w VMCS i procesor odtwarza je przy każdym przejściu pomiędzy root i non-root mode.

Ułatwienia związane z użyciem VT-x

Częsty dostęp do uprzywilejowanych zasobów

Wymaga wymuszenia błędu i jego przechwycenia przez klasycznego hypervisora – to wpływa na wydajność

- Przykład: TPR (Task Priority Register)
- VT-x wybudza VMM tylko wtedy, gdy jest to niezbędne: odpowiednie pole w VMCS określa poniżej jakiego poziomu musi spaść TPR, by doszło do wybudzenia.

VT-x - wsparcie sprzętowe

- Pentium 4 662 oraz 672
- Pentium Extreme Edition 955 oraz 965
- Pentium D 920-960 poza 945, 925, 915
- Core Solo U1000
- Core Duo T2300, T2400, T2500, T2600, T2700, L2000, U2000
- Core 2 Solo
- Core 2 Duo (wszystkie poza E8190, E7xxx poza 7300), E4xxx, T5200-T5550, T5750
- Core 2 Quad (wszystkie poza Q8200)
- Core 2 Extreme Duo oraz Quad
- Xeon 3000, 5000, 7000

Znaczący rozwój techologii pomiędzy pierwszymi, a aktualnymi implementacjami: ilość cykli procesora potrzebna na przejście pomiędzy root a non-root mode w Core 2 kilkakrotnie niższa od Pentium 4.

Software korzystający z VT-x

Projekty open source

KVM

- VirtualBox
- Xen od wersji 3.0.2
- Projekty komercyjne
- VMware Workstation, Server, Fusion
- Parallels Workstation, Desktop
- Microsoft Hyper-V
- ...

KVM jest bazowany na software'owym QEMU. Nie jest jeszcze uznawany za dostatecznie stabilny, by być włączonym do oficjalnej dystrybucji QEMU.

VT-x, AMD-V : wydajność

- Brak niezależnych, dokładnych benchmarków
- Aplikacje nieuprzywilejowane na poziomie 90% do 97% wydajności hosta (dane VMware)
- Gorzej z Ring 0
- Ładowanie Ubuntu na VirtualBox
- Z VT-x: 36s
- Bez: 28s [za tombuntu.com]
- Twórcy niektórych VM zalecają wyłączenie VT-x, jeśli zależy nam na wydajności

http://tombuntu.com/index.php/2007/10/01/should-you

Dalszy rozwój wsparcia sprzętowego dla wirtualizacji

- EPT/NPT (VT-x drugiej generacji)
- VT-d
- IOMMU

33

EPT – Extented Page Tables – technologia Intela
NPT – Nested Page Tables – techonolgia AMD

Software Page Table Virtualization

- Shadow Paging
  - Tablica stron uaktualniana przy modyfikowaniu tablicy stron gościa
- Virtual TLB
  - Tablica stron uaktualniana przy odwoływaniu się do nieistniejącego adresu oraz kasowaniu wpisów w tablicy stron gościa

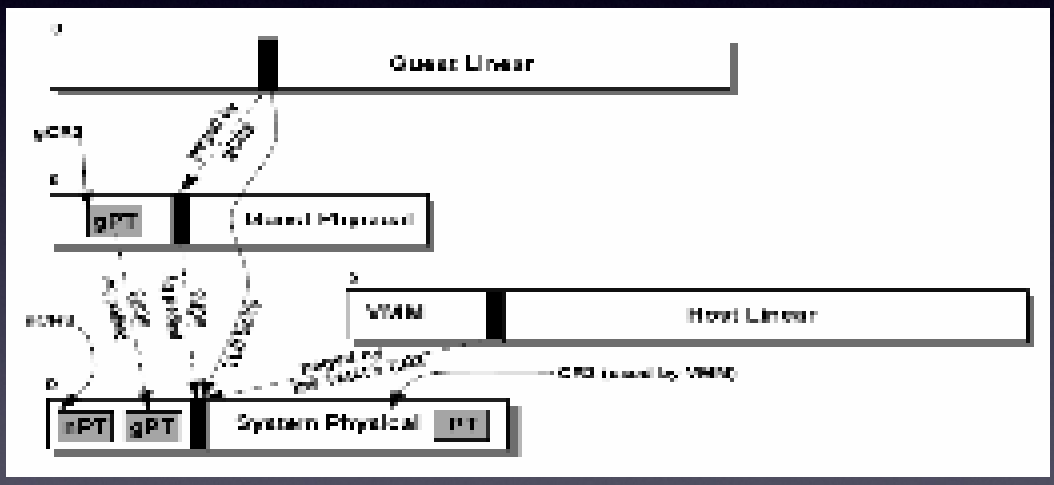
34

Shadow Pages – Jedną z możliwych technik to zabezpieczenie przed zapisem gPT. Hypervisor zabezpiecza przed zapisem wszystkie fizyczne strony z których składa się gPT. Każda próba modyfikacji przez system goszczony tablic translacji skutkuje podniesieniem wyjątku Page Fault. Przy każdym takim wyjątku sterowanie jest przenoszone do hypervisora który odpowiednio modyfikuje tablicę gPT (składa swoją tablicę stron z gPT, tworząc sPT - shadow Page Table).

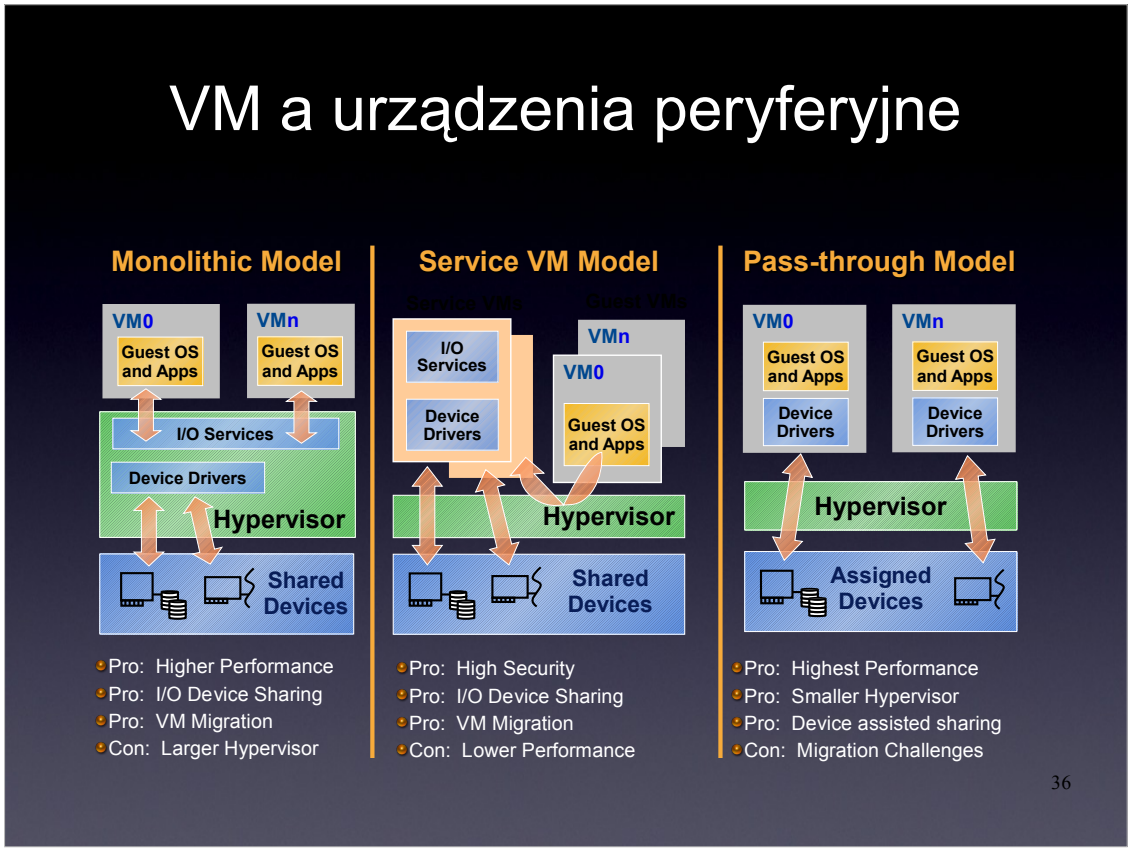
Virtual TLB – hypervisor pozwala systemowi goszczonemu na dodawanie wpisów do gPT bez przechwytywania sterowania. Jednak kiedy później gość wykona instrukcje która będzie używała dodanej translacji poskutkuje to podniesieniem wyjątku Page Fault i przechwyceniem sterowania przez hypervisora.

Hardware Page Table Virtualization

- Intel Extended Page Tables (EPT)
- AMD Nested Page Tables (NPT)



Sprzętowa wirtualizacja polega na dodaniu nowych jednostek translacji do procesora. Dzięki nim, system goszczony może bezpośrednio aktualizować swoje gPT bez udziału hypervisora, a tym bardziej przełączania kontekstu pomiędzy systemem goszczonym a hypervisorem.
Taka translacja pomimo, że jest dużo szybsza od softwarowej i tak jest dwukrotnie wolniejsza od translacji w systemie nadrzędnym. Teraz każdy adres musi przejść po dwa razy wyższym drzewie translacji.



Możliwe są różne podejścia:
- Urządzenia peryferyjne emulowane w całości.
- Specjalnie spreparowane sterowniki (parawirtualizacja) które nie komunikują się bezpośrednio z fizycznym urządzeniem, tylko z usługą uruchomioną przez hypervisora która bezpośrednio komunikuje się z urządzeniem
- Bezpośrednie przypisanie fizycznego urządzenia do maszyny wirtualnej i komunikacja bez jakiejkolwiek ingerencji hypervisora

Direct I/O

Direct Assigment

- VT-d Interrupt Remapping
- VT-d DMA Remapping
- AMD-V
- IO MMU – I/O Memory Managment Unit)

37

Dzięki technice Direct Assignment, system goszczony może pracować bezpośrednio na fizycznym sprzęcie. Dzięki wsparciu VT-d Interrupt Remapping przerywania zawierają dodatkową informację, do której wirtualnej maszyny powinny trafić. Dzięki czemu sprzęt może rozmawiać bezpośrednio z wirtualizowanymi systemami. VT-d DMA Remapping dostarcza dodatkową warstwę translacji adresów z przestrzeni urządzenia do przestrzeni systemu goszczonego.

Takie podejście posiada także wady:
- dzisiejsze urządzenia peryferyjne nie są przystosowane do bycia obsługiwanym przez wiele niezależnych systemów operacyjnych przez co jedno urządzenie może być na raz przypisane do jednego systemu.
- pojawiają się problemy przy przenoszeniu wirtualizowanego systemu w locie na inną maszynę. Przy takim przenoszeniu należałoby także odwzorować po drugiej stronie wewnętrzny stan urządzeń peryferyjnych co nie zawsze jest możliwe.

Blue Pill

Rootkit dla Windows Vista stworzony przez Joannę Rutkowską

- Implementacja bazuje na AMD-V
- Założenie, że idealna wirtualizacja jest niewykrywalna przez system goszczony
- W założeniu całkowite i niewykrywalne przejęcie kontroli nad zainfekowaną maszyną
- Wirtualizacja w locie

Blue Pill – możliwości wykrycia

- Pomiar czasu trwania instrukcji
- Pomiar ilości dostępnych zasobów
- Red Pill
- Instrukcja SIDT
- Dostęp do adresu tabeli deskryptorów przerywań z poziomu ring3

SubVirt

Stworzony przez Microsoft i Uniwersytet w Michigan

- Implementacja pozwala na:
- Logowanie klawiatury
- Przeglądanie dysku w poszukiwaniu informacji o użytkowniku
- Unieszkodliwianie programów antywirusowych

Zapobieganie

- Stworzenie sprzętowego wykrywania VM rootkitów
- Bootowanie z niezainfekowanego źródła np. CD-ROM, pamięci flash, przez sieć
- Wyłączenie sprzętowego wspomagania wirtualizacji w BIOSie

Pararells Desktop

VirtualBox

- Open Source (GPL)
- Potrafi uruchomić się z poziomu Windows, Linux, Macintosh oraz OpenSolaris
- Wsparcie dla Windows (NT 4.0, 2000, XP, Server 2003, Vista), DOS/Windows 3.x, Linux (2.4 and 2.6), Solaris and OpenSolaris, and OpenBSD

VirtualBox - możliwości

- Snapshoty
- Symulacja sieci, portów USB, portów szeregowych
- Przekierowywanie portów
- Współdzielone foldery
- Zdalny pulpit
- Obrazy dysków twardych (stałego i zmiennego rozmiaru, różnicowe oraz fizyczne)

Dziękujemy za uwagę

Pytania?