Wirtualizacja

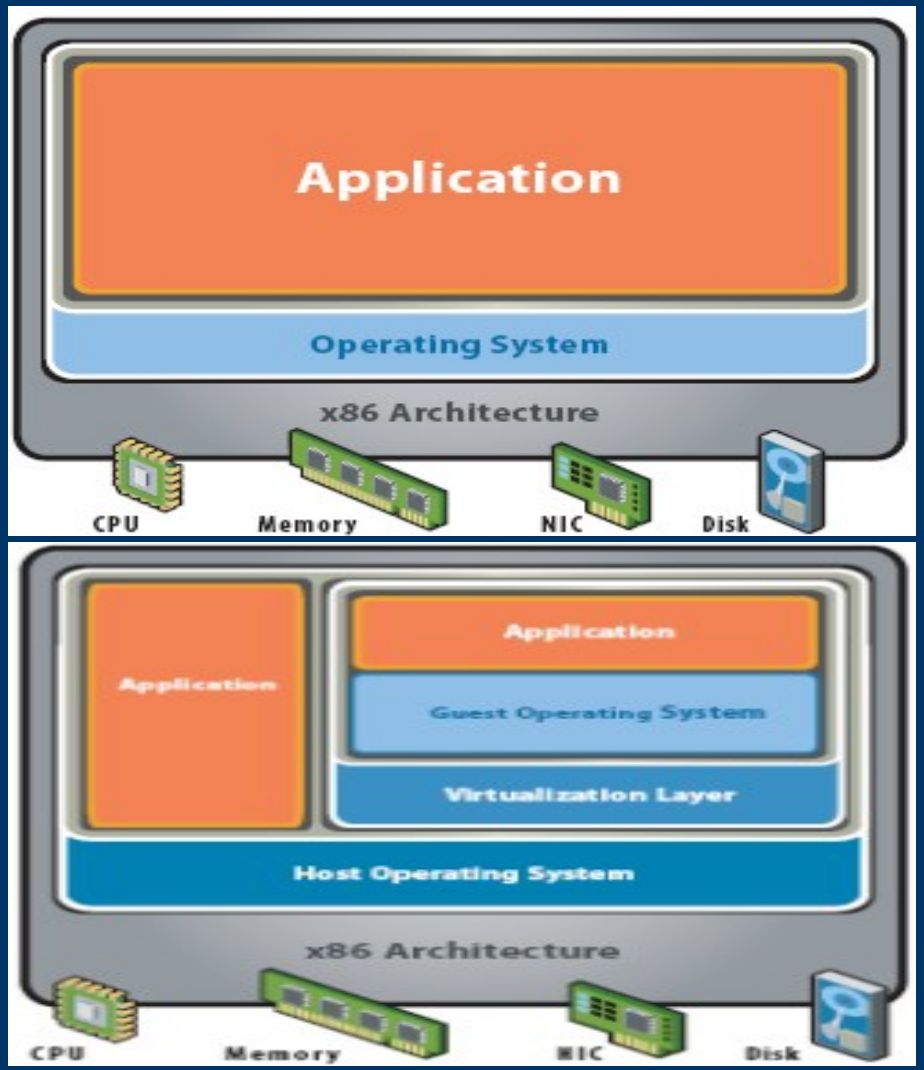
Maciej Pawlisz, Konrad Tomala, Paweł Łukasz

Plan Prezentacji

- Co to jest wirtualizacja?
- Zastosowania wirtualizacji
- Historia wirtualizacji
- Typy wirtualizacji programowej
  - Emulacja pełna
  - Emulacja API
  - Wirtualizacja właściwa
    - Parawirtualizacja
    - Pełna wirtualizacja
- Kryterium Popka-Goldberga
- Twierdzenie Popka-Goldberga
- Problemy w x86
- Intel VT-x (VT-i) i AMD-V
- Narzędzia wirtualizujące
- Pytania

Co to jest wirtualizacja?

- Wirtualizacją nazywamy użycie oprogramowania w celu stworzenia abstrakcji (iluzji) posiadanych zasobów.
- Wirtualizacja sprzętu (partycje, virtual CDROM)
- Wirtualizacja programowa



Zastosowania wirtualizacji

- Możliwość uruchamiania przestarzalego oprogramowania
  - Wielka prędkość rozwoju branży komputerowej
  - Firmy często korzystają ze specyficznego oprogramowania
  - Gry komputerowe przeznaczone na starsze platformy
- Uniezależnienie od ograniczeń sprzętu czy systemu operacyjnego
  - Korzystanie z aplikacji wielu systemów operacyjnych
  - Symulowanie różnych architektur i konfiguracji sprzętowych
  - Proste przenoszenie obrazu systemu pomiędzy maszynami

Zastosowania wirtualizacji

- Bezpieczeństwo uruchamiania podejrzanych aplikacji, debugowanie i monitorowanie
  - Nieprawidlowe dzialanie programu nie spowoduje destabilizacji systemu hosta
  - Możliwość uruchomienia kodu poziomu systemowego na poziomie użytkownika, obserwacja z „zewnątrz” skutków niepożądanego dzialania
  - Testowanie nowych aplikacji, systemów operacyjnych, zmian w systemch, ciekawych scenariuszy dzialania systemu, różne eksperymenty – przywrócenie poprzedniego stanu bez konieczności restartowania

Zastosowania wirtualizacji

- Równolegla praca na wielu systemach operacyjnych
  - Używane w help-deskach
  - Korzyść - szybkie zmiany OS bez konieczności rebootowania
- Konsolidacja serwerów
  - Wiele maszyn wirtualnych na jednej maszynie fizycznej, oszczędność sprzętu, serwisowania, energii
- Wykorzystanie podczas szkoleń
  - Bezpieczeństwo
  - Prosty powrót do poprzednich stanów systemu

Zastosowania wirtualizacji

- Tworzenie przenośnego oprogramowania (np. wirtualna maszyna Javy)
  - Kod wykonywany w ten sam sposób na każdym OS
- Honey spot
  - Przyciągnięcie uwagi hakera i analizowanie jego działań
- Tworzenie abstrakcji sieci komputerowych na jednej maszynie
  - Symulowanie działania sieci i sprawdzanie bezpieczeństwa
  - Oszczędność sprzętu i miejsca

Historia wirtualizacji

- Uniwersytet w Manchesterze- komputer Atlas
- Lata 60, laboratoria IBM, projekt M40/44X prowadzący do stworzenia całej klasy komputerów Mainframe
- Połowa lat 90 – Java, VMWare

Typy wirtualizacji programowej

- Emulacja pełna
- Emulacja API
- Wirtualizacja właściwa
  - Parawirtualizacja
  - Pełna wirtualizacja

Emulacja pełna

- Instrukcje maszynowe skompilowanych programów wykonywane programowo w pętli
- Symulacja pamięci, procesora, czasu, rejestrów
- Ogromny spadek wydajności (nawet kilkaset razy)
- Duża przenośność i bezpieczeństwo
- Pełna kontrola nad systemem emulowanym
- Przykłady
  - Bochs
  - QEMU

Emulacja API

- Zastąpienie funkcji systemowych i bibliotecznych emulowanego systemu przez ich odpowiedniki w naszym systemie
- Trzeba przepisać całe API
- Dobra wydajność
- Nieprzenośna
- Przykład – Wine

Wirtualizacja właściwa

- Wykorzystanie tej samej platformy sprzętowej do wirtualizacji różnych systemów operacyjnych
- Nie trzeba tłumaczyć wszystkich instrukcji
- Szybsza od emulacji pełnej z prawie taką samą funkcjonalnością
- Podział
  - Parawirtualizacja
  - Wirtualizacja pełna

Parawirtualizacja

- Hipernadzorca
- Nie jest tworzona iluzja sprzętu – dostarczane jest specjalne API do jego obsługi
- System gościa wie, że jest wirtualizowany
- Modyfikowane jest jego jądro (czyli ograniczone do systemów typu open source) aby korzystało z dostarczanego API
- Bardzo wydajna
- Przykład - XEN

Pełna wirtualizacja

- System gościa nie wie, że nie jest sam w systemie
- Zbiór instrukcji emulowanych ograniczony do minimum
- Potrzebna odpowiednia architektura
- Architektura x86 nie jest dobrze przystosowana
- Przykłady:
  - VirtualBox
  - VirtualPC
  - VMware

Kryterium Popka-Goldberga

- Odpowiedniość – program działający na maszynie wirtualnej musi zachowywać się w dokładnie taki sam sposób, jak na rzeczywistym sprzęcie
- Kontrola zasobów – wirtualna maszyna musi w pełni kontrolować wszystkie zasoby, które są wirtualizowane
- Wydajność – większa część instrukcji musi być wykonywana bez udziału maszyny wirtualnej

Twierdzenie Popka-Goldberga

- Instrukcje uprzywilejowane
- Instrukcja wrażliwe ze względu na kontrolę
- Instrukcje wrażliwe ze względu na wykonanie

Dla każdego standardowego komputera trzeciej generacji wirtualna maszyna zostać skonstrukowana, jeśli zbiór instrukcji wrażliwych jest podzbiorem zbioru instrukcji uprzywilejowanych

Problemy w x86

- Niektóre instrukcje wrażliwe (korzystające z rejestrów i ustawień systemowych) nie wywołują przerwań
- Samomodyfikujący się kod jest trudny do wykrycia
- Poziomy uprzywilejowania

Intel VT-x (VT-i) i AMD-V

- Specjalne struktury danych do przechowywania stanu procesora systemu gościa i gospodarza
- Dwa typy operacji: root operations i non-root operations
- VMM ma kontrole nad poziomami uprzywilejowania
- Więcej instrukcji wrażliwych jest uprzywilejowanych (wywołują przerwania)

VirtualBox

- Open source
- Snapshoty
- Zapisywanie stanu
- Sieciowość: NAT, HIF, wewnętrzna
- Guest Additions (np. seamless mouse)
- Możliwość współdzielenia obszarów dysku pomiędzy hostem a guestem

VirtualBox cd.

- Obrazy dysków twardych
  - O stałym rozmiarze, rozszerzalne, różnicowe, fizyczne
  - W trybie normalnym, niezmiennym i write-through
- Obsługuje też *.vmdk (na razie tylko write-through)
- iSCSI
- Interfejsy: VirtualBox, VBoxManage, VBoxSDL, zdalny VRDP

Virtual PC

- Stworzony przez firmę Connectix Corp., która została przejęta przez Microsoft w 2003 r. Jest też wersja Microsoft Virtual PC for Mac 7.

- w roli hosta może wystąpić:
- Windows XP Professional, Windows XP Tablet Edition, Windows 2000 Professional

- w roli guesta może wystąpić:
- Windows XP (home i prof.), Windows 2000 Professional
- Windows NT 4.0 Workstation (wymagany SP6a), Windows ME, 98, 95, MS-Dos 6.22, OS/2

- Pełna lista działających systemów: http://vpc.visualwin.com/

- Część sprzętu jest wirtualizowana (np. procesor, ale w wersji na Mac emulowany)
- część sprzętu emulowana (kontroler DMA, kontroler przerwań, IDE/ATA, kontroler pamięci, wyjścia/wyjścia
- Stałą konfiguracja maszyny wirtualnej
- płyta z chipsetem Intel 440BX
- karta graficzna S3 Trio
- karta sieciowa DEC/Intel 21140A
- karta muzyczna Sound Blaster 16 ISA

- Są 4 rodzaje dysków z których mogą korzystać maszyny wirtualne
- - pliki o zadanym maksymalnym rozmiarze, zwiększające się w miarę potrzeb
- - pliki o stałym rozmiarze
- - odnośniki do prawdziwych dysków – można uruchomić system z istniejącej instalacji
- - dyski różnicowe
- Możliwość współdzielenia dysków wirtualnych i katalogów dzielonych
- Pamięć operacyjna przydzielana od razu
- Adres IP przydzielany dynamicznie
- Możliwość stworzenia sieci gdzie maszyny wirtualne i host mogą komunikować się
- Można zmieniać ustawienia istniejącej maszyny

Virtual Machine Additions

- Instalowane w gueście (tylko DOS i windows)
- Współdzielenie katalogów
- Współdzielenie schowka
- Drag & drop
- Synchronizacja czasowa
- Płynne przechodzenie kursora między systemami

- Zalety:
- Łatwa obsługa
- Możliwość ustalania priorytetów
- Dyski Undo, umożliwiające łatwe cofnięcie zmian
- Szybszy niż VMware

- Wady
- Nie działa USB, FireWire
- Sieć tylko przez DHCP

UML

- Autorem jest Jeff Dike
- Obsługuje platwormy x86, ia64, PowerPC
- Strona *http://user-mode-linux.sourceforge.net*
- Pozwala uruchomić jądro linuxa pod innym linuxem
- Działa jak zwykły proces hosta
- Dostęp to sprzętu przez funkcje systemowe

Tryb TT (Treacing Thread)

- Każdy proces ma odpowiadający proces w systemie macierzystym
- Pamięć jądra dostępna dla uruchomionych procesów
- Specjalny proces przechwytuje sygnały za pomocą ptrace, anuluje je a proces pod ULM-em wchodzi w tryb jądra UML-a
- Przekazywanie obsługi nad wywołaniami systemowymi za pomocą sygnałów

Tryb SKAS(Separate Kernel Address Space)

- Niedostępna przestrzeń adresowa jądra
- Ograniczenie liczby procesów do 4
- - proces jądra UML-a, - wywołuje kod jądra i przechwytuje wywołania systemowe procesów działających pod UML-em
- - proces użytkownika - uruchamia wszystkie procesy działające pod UML-em i przełącza kontekst pomiędzy systemem hosta a UML-e
- - proces asynchronicznych operacji na wirtualnych urządzeniach blokowych
- - operacji wejścia-wyjścia
- Wyeliminowanie obsługi sygnałów
- Trudniejszy w debuggowaniu
- Wymaga modyfikacji jądra hosta

Uruchomienie UML

- Ściągnięcie jądra i systemu plików, wozpakowanie
- Uruchomienie UML:

  host% chmod 755 ./linux-2.6.23

  host% ./linux-2.6.23 ubda=FedoraCore5-x86-root_fs mem=128M
- Zaloguj się jako root (nie potrzeba hasła)

  localhost login: root

  [root@localhost ~]#
- Można zamknąć jądro pod UML

  [root@localhost ~]# halt

Budowanie ze źródeł

- Rozpakuj jądro

  host% bunzip2 linux-2.6.16.tar.bz2
  host% tar xf linux-2.6.16.tar
  host% cd linux-2.6.1

- Skonfiguruj jądro

  host% make defconfig ARCH=um
  host% make menuconfig ARCH=um
  host% make mrproper
  host% make mrproper ARCH=um

- zbuduj

  host% make ARCH=um

Wine (Wine is not Emulator)

- Powstał w 1993 r. (początkowo obsługiwał tylko 16-bitowe aplikacje), do rozwoju dołączył się m.in. Corel. W 2002 r. udostępniono program na licencji Lesser General Public License.
- umożliwia uruchamianie windosowych programów (Win32, Win 3.x, Dos) pod Linuxem

- Wykorzystuje podobieństwa instrukcji hosta i domyślnego systemu
- Konieczność emulowania tylko niektórych instrukcji
- Zawiera implementację wielu bibliotek DLL, chociaż nie wszystkie są kompletne
- Wspiera pliki COM, dobrze sobie radzi z zarządzaniem rejestrami, można drukować z aplikacji windosowych, dobre wsparcie dźwięku, obsługa TCP/IP, wsparcie zaawansowanych kontrolerów pod Win32
- Wspiera grafikę na X11, SDL, częściowe wsparcie DireX
- zawiera bardzo dobry debugger

Wady

- Wymagana wspólna architektura
- Konieczność podmontowywania urządzeń
- Instalacja z linii poleceń
- Dodatkowe kroki instalacji
- Urządzenia muszą być obsługiwane przez linuxa (nie działają windosowe sterowniki)

Zalety

- Możliwość uruchamiania windosowych programów w linuxowych skryptach
- Zdalne uruchamianie programów
- Zalety linuxa: bezpieczeństwo, stabilność
- Nie trzeba mieć licencji Windowsa
- Dostępny kod

- Instalacja ze źródeł

  tar zxvf Wine-YYYYMMDD.tar.gz

  ./configure && make depend && make

  su -c "make install" (jako root)
- Instalacja wersji binarnych

  RPM: rpm -iv wine-YYYYMMDD-i386.rpm

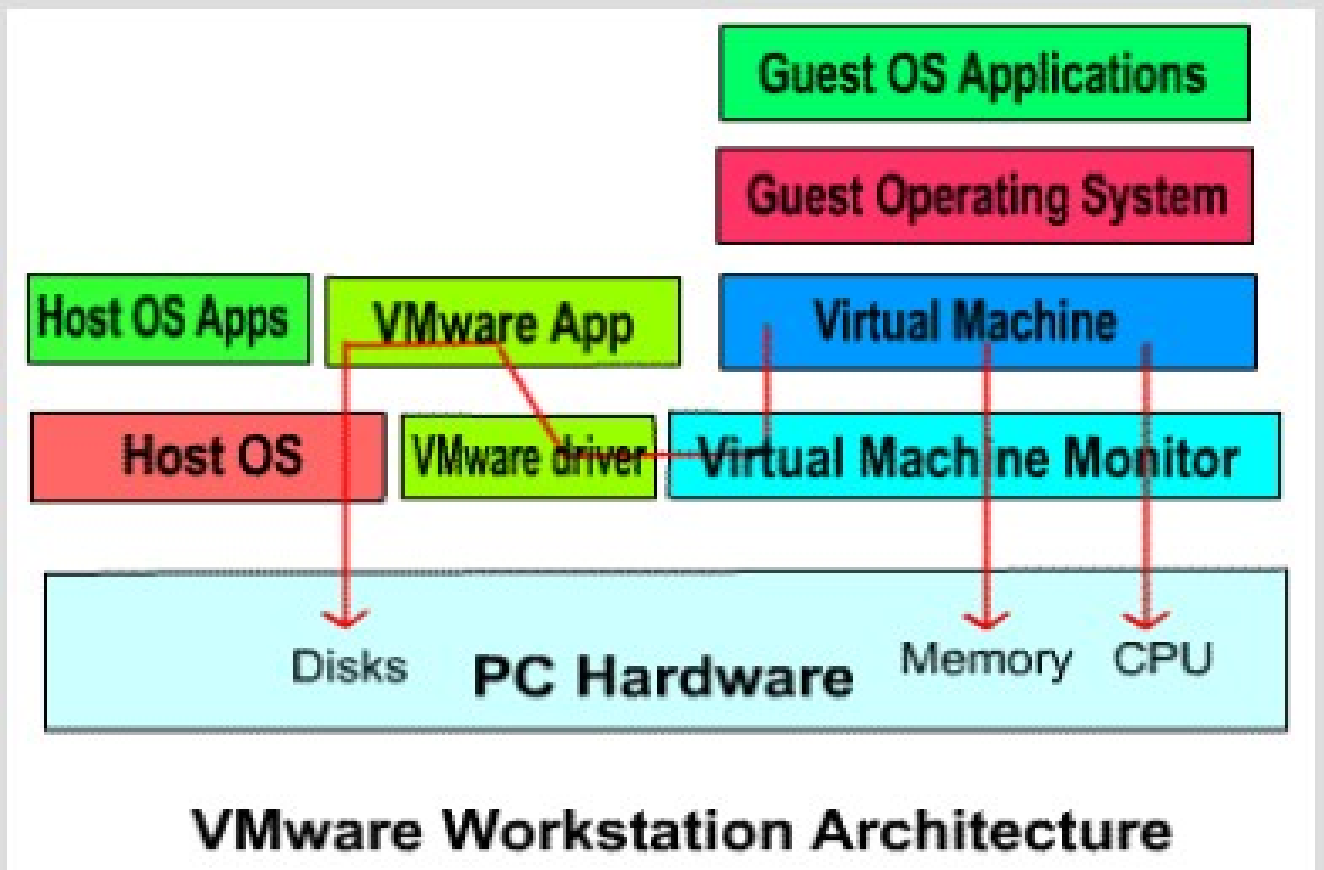
  Debian: apt-get install wine
- Uruchomienie aplikacji

  wine nazwa_programu

VMware Workstation

- Strona domowa projektu: www.vmware.com
- Projekt komercyjny, 30 dniowa wersja darmowa
- Przedstawiciel pelnej wirtualizacji, umożliwia uruchomienie kilku systemów operacyjnych na jednym komputerze rodziny x86, systemy te są zakapsulkowane w oddzielnych maszynach wirtualnych

Zasada dzialania VMware



Podzial VMware na komponenty

- Virtual Machine Monitor (VMM)
- VMX Driver
- VMware Application (Vmapp)

VMM – dziala na pozomie jądra hosta, zarządza wirtualizacją, filtr instrukcji procesora

VMX Driver – dziala na poziomie jądra hosta, , połączenie pomiędzy VMM a systemem operacyjnym hosta

Vmapp – dziala na poziomie aplikacji hosta, laduje VMM do pamięci jądra przy pomocy VMX Driver

Obslugiwane systemy operacyjne

- Host: Windows , Linux
- Guest: DOS, Linux, Windows, FreeBSD, Netware, Solaris i inne

Zastosowania VMware

- Uruchamianie wielu systemów operacyjnych na jednej maszynie i proste przelaczanie pomiedzy nimi
- Uniknięcie fizycznego partycjowania dysku
- Zasymulowanie dzialania sieci
- Testowanie nowej aplikacji na kilku systemach jednocześnie
- Szkolenia, niebezpieczne aplikacje – wykorzystanie snapshotów
- Możliwość użycia Network Address Translation przez maszynę hosta
- Możliwość grupowania maszyn wirtualnych, badanie zależności klient – serwer
- Wykonywanie klonów maszyn wirtualnych

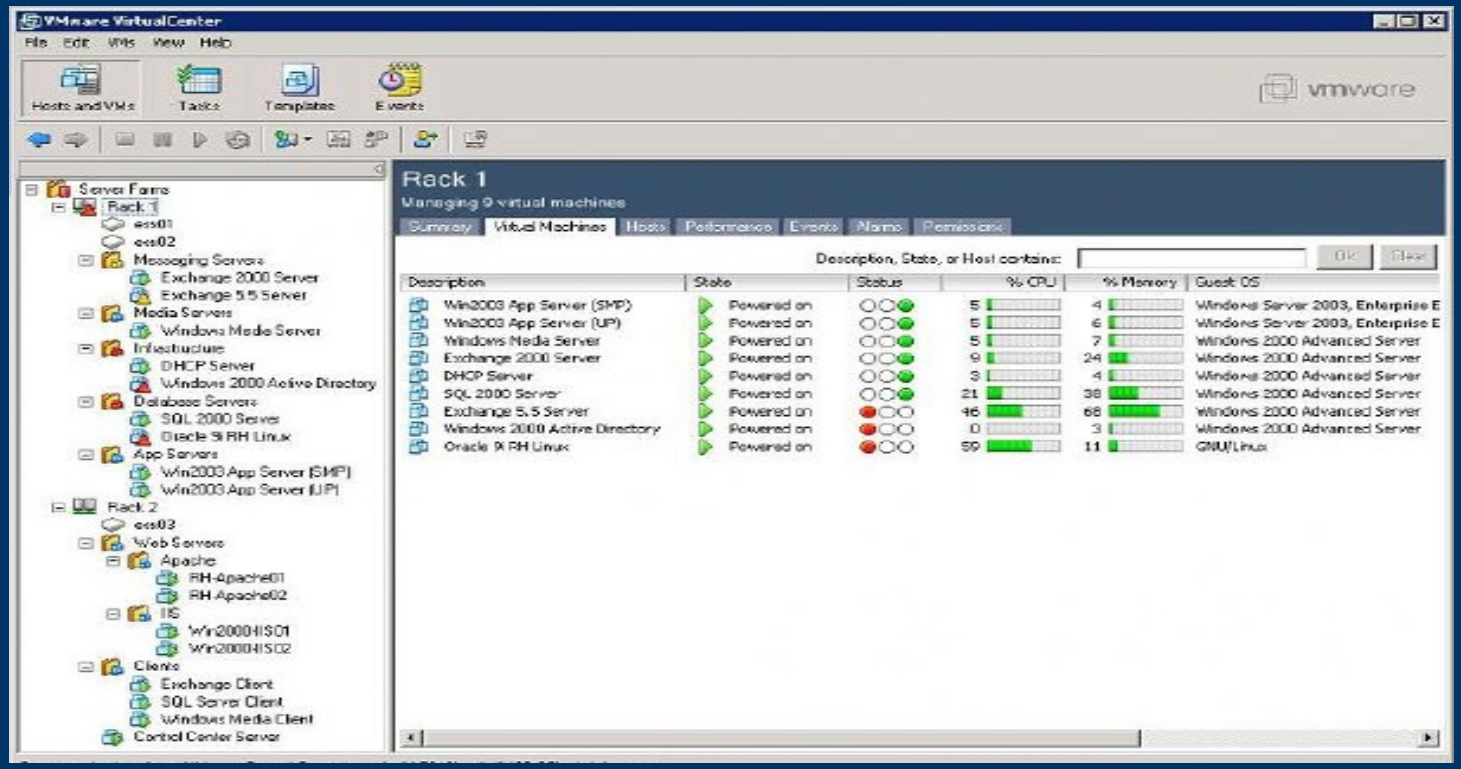
Narzędzia VMware Tools

- Sterownik SVGA poprawiający jakość grafiki
- Proste przenoszenie plików pomiędzy systemami guesta i hosta, dzielenie plików (shared folder)
- Synchronizacja czasu pomiędzy hostem a guestem
- Automatyczne przełączenie myszy

Inne produkty VMware

- VMware P2V – wirtualizowanie fizycznych maszyn
- VMware ACE – rozszerzenie Workstation o system autoryzacji
- VmwarePlayer – darmowe oprogramowanie, pozwala tylko na uruchomienie maszyny utworzonej przez któreś z narzędzi powyżej,
- VMware ESX Server – oprogramowanie do tworzenia wirtualnej infrastruktury
- VMware Server – darmowe oprogramowanie do tworzenia wirtualnej infrastruktury
- VMware Virtual Center – centrum zarządznia wirtualną infrastrukturą opartą na ESX Server i Server

Vmware Virtual Center



- Łatwe zarządzanie komputerami
- Ujednolicone środowisko
- Prosta duplikacja serwera

- Emulator procesora napisany przez Francice Bellarda
- Strona projektu : http://fabrice.bellard.free.fr/qemu/
- Dostępny jako darmowe oprogramowanie

Może działać w dwóch trybach

- Tryb pełnej emulacji systemu, QEMU emuluje pełny komputer zawierający procesor oraz rozmaite urządzenia peryferyjne. Zastosowanie do uruchomienia różnych systemów operacyjnych bez restartowania komputera oraz do debugowania kodu systemowego
- Emulacja trybu użytkownika (dla Linuxa jako hosta) uruchamianie procesów Linuxa skompilowanych na innym procesorze niż bieżący, sprawdzanie wyników cross-kompilacji i zastosowanie do cross-debugowania

Możliwości QEMU

Dziala na platformach:
Intel x86, IA-64, AMD64, PowerPC, Alpha, SPARC 32 i64, ARM, M68

Potrafi emulować:

Target CPU	User emulation	System emulation
x86	OK	OK
x86_64	Not supported	OK
ARM	OK	OK
SPARC	OK	OK
SPARC64	Dev only	Dev only
PowerPC	OK	OK
PowerPC64	Not supported	Dev only
MIPS	OK	OK
m68k (Coldfire)	OK	OK
SH-4	Dev only	Dev only
Alpha	Dev only	Dev only

Zalety QEMU

- Wyprzedza konkurencję (Bochs, PearPC) - większa szybkość dzialania, obsluga wielu architektur
- Nieograniczony dostęp do stanu maszyny
- Możliwość zapisywania (stan pamięci, rejestrów, dyski, stan zegara) i wznawiania stanu maszyny
- Możliwość emulacji karty sieciowej
- Obsluga USB
- Pelna otwartość kodu
- Obsluga SMP

Wady QEMU

- Niepelna obsluga systemu MS Windows w roli gospodarza
- Niekompletne wsparcie dla mniej popularnych platform sprzętowych
- Trudniejszy w użytkowaniu niż inne emulatory (interfejs niezbyt czytelny, używamy konsoli)
- Brak narzędzi dla emulowanych systemów zwiększających komfort pracy

Rozwiązania zwiększające szybkość

- Stosuje dynamiczną translację
- Technika mikrooperacji
- Może emulować wlasną jednostkę zarządzania pamięcią (tzw. MMU) za pomocą MMU gospodarza, jeśli dochodzi do prawdziwej emulacji używa cache tlumaczenia adresów

Qemu Accelerator - kqemu

- Darmowe narzędzie, źródla są jednak zamknięte
- Znaczna część instrukcji wykonywana na procesorze hosta, przypomina pelną wirtualizację
- Przeznaczony pod Linuxy 2.4 i 2.6 oraz Windowsy 2000 i XP
- Pozwala na znaczne zwiększenie szybkości emulacji
  - Qemu bez kqemu – ok. 10-20% prędkości symulowanej maszyny
  - Qemu z kqemu – ok. 50-100% prędkości w zależności od wykonywanych instrukcji

Instalacja Qemu pod Windows

- Pobieramy i rozpakowujemy archiwum ze strony: http://www.h7.dion.jp/~qemu-win
- Tworzymy pusty obraz dysku dla systemu
  qemu-img.exe create -f qcow hda.img 5G
- Instalacja nowego systemu z plyty CD bądź obrazu instalacyjnego systemu
  qemu.exe -L . -cdrom ['\\.\.D:' | nazwa plilu z obrazem] -hda hda.img -m 256 -boot d
- Zainstalowany system uruchamiamy przy pomocy komendy
  qemu -L . -hda hda.img -m 256

Qemu Accelerator - kqemu

- Możęmy pobrać z tej samej strony: http://www.h7.dion.jp/~qemu-win
- Bardzo prosta instalacja
- Przy uruchamianiu do podstawowej komendy możemy dodać:
  - -kernel-kqemu - uruchamia z akceleratorem
  - -no-kqemu – uruchamia bez akceleratora
- Wizualizacja używania QEMU na filmie

Pytania