Wirtualizacja sprzętowa

(czyli tej prezentacji – nie ma!)

Dominika Pawlik, Michał Gołębiowski, Kamil Kawka

20 listopada 2008

Co to jest?

Jest to sposób ukrycia części związanych z architekturą zasobów sprzętowych, aby można było wygodniej z nich korzystać.

Wirtualizuje się praktycznie wszystkie warstwy w komputerze:

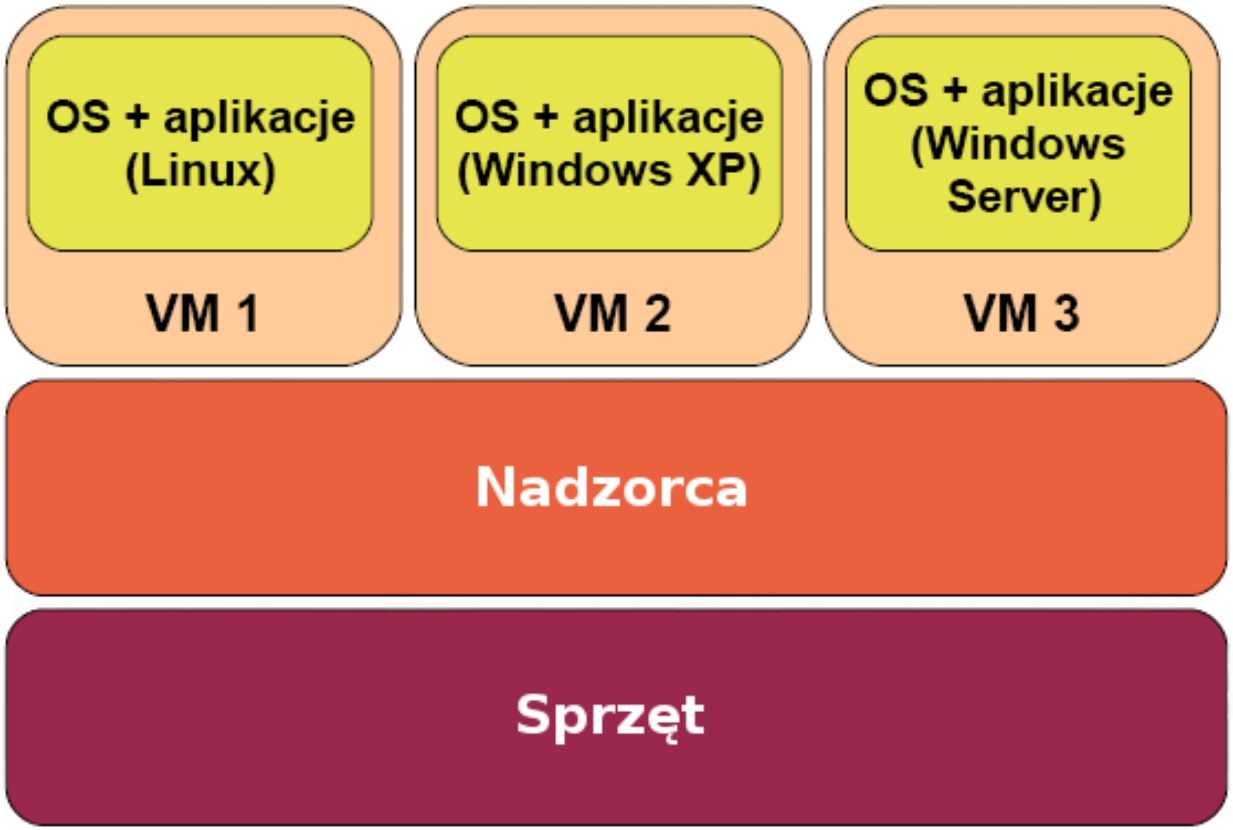
1. wirtualne sieci
2. wirtualne urządzenia
3. **wirtualna maszyna**
4. wirtalna pamięć
5. wirtualny system plików

Co to jest?

Jest to sposób ukrycia części związanych z architekturą zasobów sprzętowych, aby można było wygodniej z nich korzystać. Wirtualizuje się praktycznie wszystkie warstwy w komputerze:

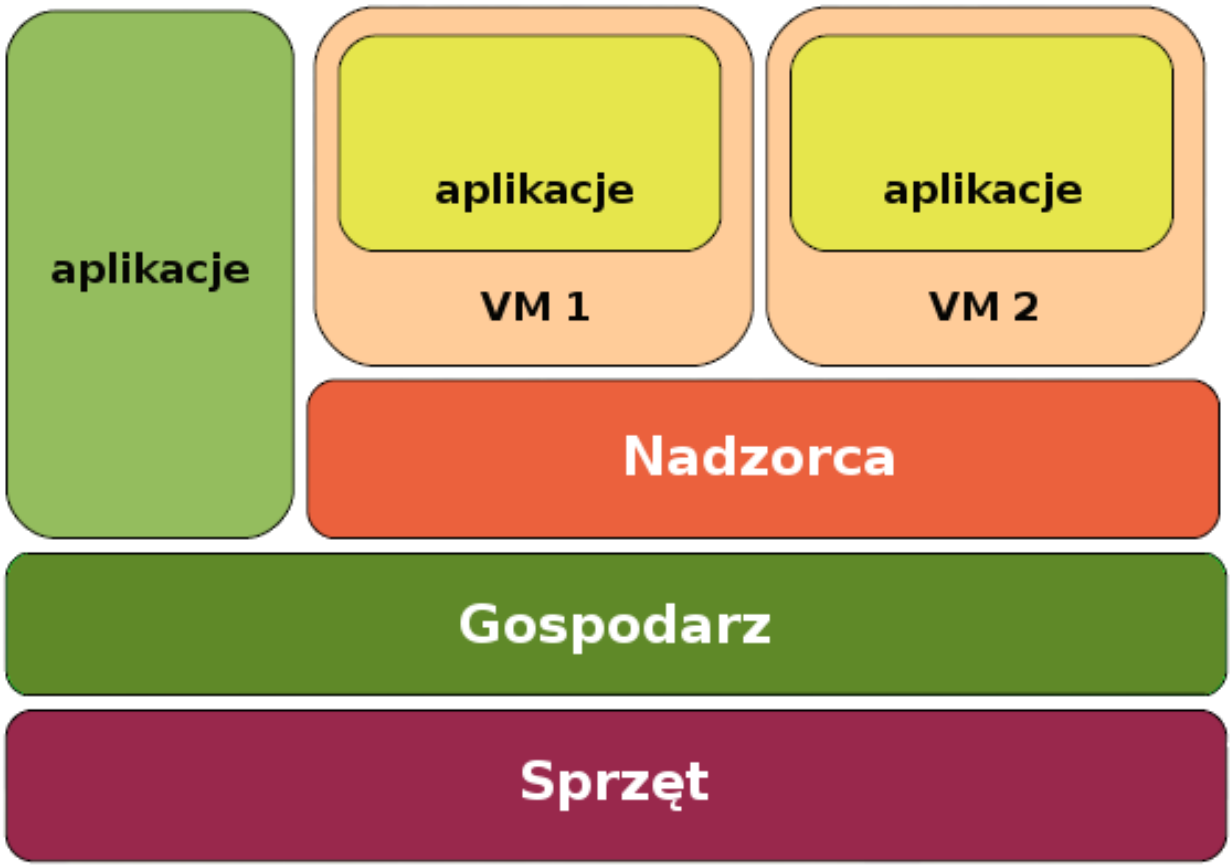
1. wirtualne sieci
2. wirtualne urządzenia
3. **wirtualna maszyna**
4. wirtalna pamięć
5. wirtualny system plików

Jak to działa („Niektórzy mówią, że nas oko łudzi”)



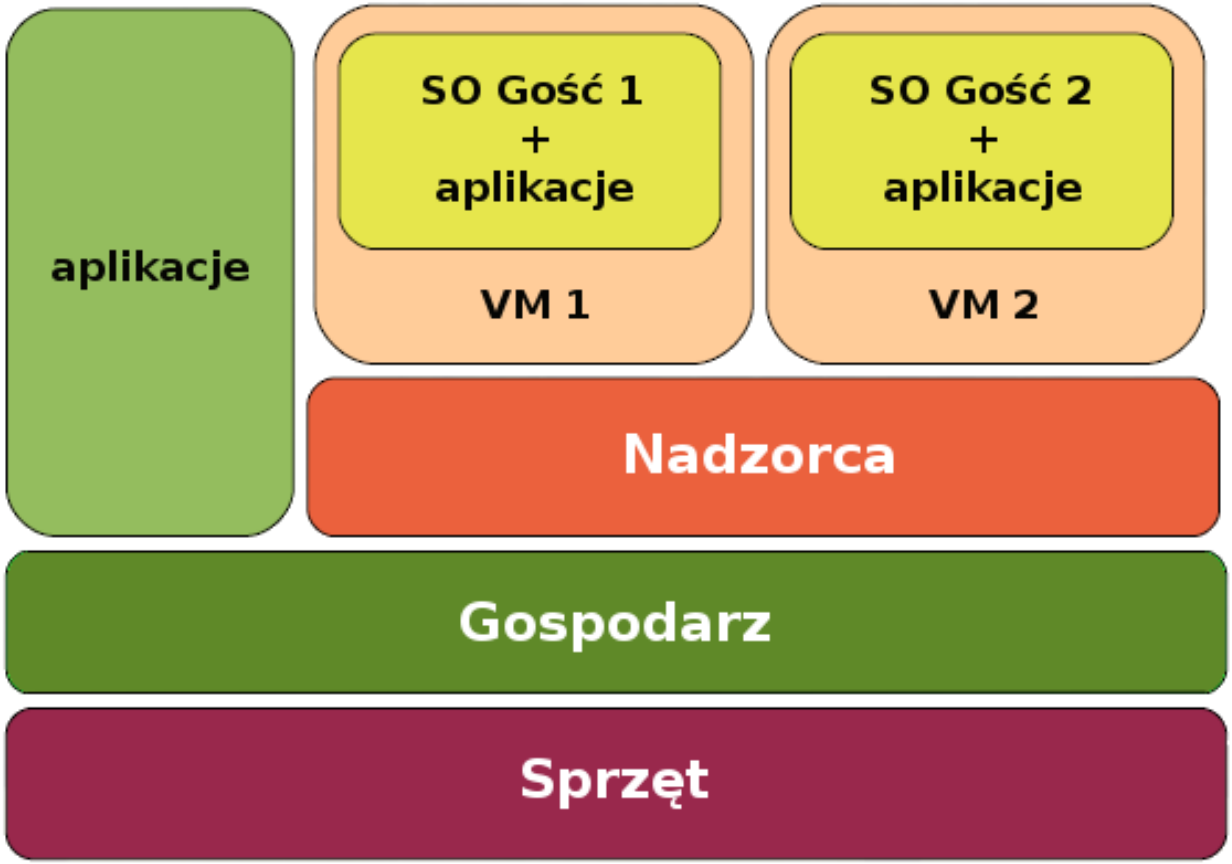
Jak to działa

(„Niektórzy mówią, że nas oko łudzi
I że nic nie ma ...”)



Jak to działa

(„Niektórzy mówią, że nas oko łudzi
I że nic nie ma, tylko się wydaje”)



A do czego służy?

1. serwery – zwykle obciążenie serwera to mniej niż 15%, więc czemu na jednej maszynie nie postawić wielu serwerów?
2. uruchamanie programów na inną architekturę
3. korzystanie z paru systemów operacyjnych
4. tworzenie snapshotów
5. testowanie niebezpiecznych programów
6. testowanie zmian w systemie operacyjnym

A do czego służy?

1. serwery – zwykle obciążenie serwera to mniej niż 15%, więc czemu na jednej maszynie nie postawić wielu serwerów?
2. uruchamanie programów na inną architekturę
3. korzystanie z paru systemów operacyjnych
4. tworzenie snapshotów
5. testowanie niebezpiecznych programów
6. testowanie zmian w systemie operacyjnym

A do czego służy?

1. serwery – zwykle obciążenie serwera to mniej niż 15%, więc czemu na jednej maszynie nie postawić wielu serwerów?
2. uruchamanie programów na inną architekturę
3. korzystanie z paru systemów operacyjnych
4. tworzenie snapshotów
5. testowanie niebezpiecznych programów
6. testowanie zmian w systemie operacyjnym

A do czego służy?

1. serwery – zwykle obciążenie serwera to mniej niż 15%, więc czemu na jednej maszynie nie postawić wielu serwerów?
2. uruchamanie programów na inną architekturę
3. korzystanie z paru systemów operacyjnych
4. tworzenie snapshotów
5. testowanie niebezpiecznych programów
6. testowanie zmian w systemie operacyjnym

A do czego służy?

1. serwery – zwykle obciążenie serwera to mniej niż 15%, więc czemu na jednej maszynie nie postawić wielu serwerów?
2. uruchamanie programów na inną architekturę
3. korzystanie z paru systemów operacyjnych
4. tworzenie snapshotów
5. testowanie niebezpiecznych programów
6. testowanie zmian w systemie operacyjnym

A do czego służy?

1. serwery – zwykle obciążenie serwera to mniej niż 15%, więc czemu na jednej maszynie nie postawić wielu serwerów?
2. uruchamanie programów na inną architekturę
3. korzystanie z paru systemów operacyjnych
4. tworzenie snapshotów
5. testowanie niebezpiecznych programów
6. testowanie zmian w systemie operacyjnym

Wady

1. większe zużycie zasobów
2. spadek wydajności
3. mniejsza wygoda
4. problemy z licencjami na oprogramowanie (np. Microsoft)

Wady

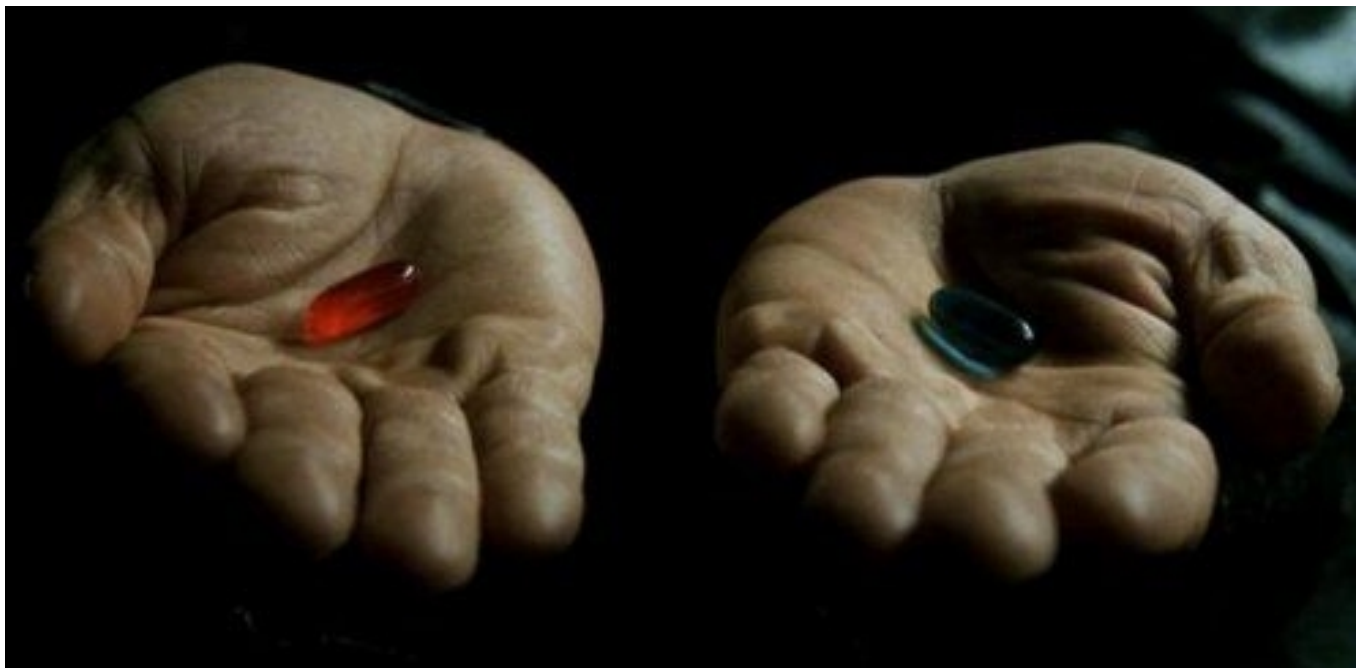
1. większe zużycie zasobów
2. spadek wydajności
3. mniejsza wygoda
4. problemy z licencjami na oprogramowanie (np. Microsoft)

Wady

1. większe zużycie zasobów
2. spadek wydajności
3. mniejsza wygoda
4. problemy z licencjami na oprogramowanie (np. Microsoft)

Zagrożenia

- SubVirt, Samuel T. King et al, University of Michigan and Microsoft Research
- BluePill, Joanna Rutkowska, COSEINC 3 września 2006



BluePill

- zaprojektowany przez Joannę Rutkowską w 2006
- teoretycznie umożliwia całkowite i niewykrywalne przejęcie kontroli nad systemem ofiary
- kiedy system połknie niebieską tabletkę:
  Blue Pill przechwytuje system w locie i ładuje go do swojej VM. Użytkownik ofiary pracuje bez straty wydajności i bez możliwości wykrycia Blue Pill
- prototypowa implementacja z opublikowanymi źródłami

Jak groźne są takie wirusy?

Zagrożenie ze strony rootkitów wykorzystujących VM jest na tyle realne, że jeden z autorów SubVirta przez jakiś czas wykorzystywał komputer, o którym nie wiedział, że jest zarażony prototypowym rootkitem.

I kolejne zyski

Honeypot – pułapka na wirusy:
`http://www.citi.umich.edu/u/provos/papers/honeyd.pdf`

Emulacja Pełna

Emulacja – program komputerowy emuluje działanie jednego systemu operacyjnego.
Symuluje: pamięć, procesor, urządzenie I/O, zegar, ...

Emulacja API – emulujemy tylko bibliotekę systemową.

Emulacja Pełna

Emulacja – program komputerowy emuluje działanie jednego systemu operacyjnego.
Symuluje: pamięć, procesor, urządzenie I/O, zegar, ...

Emulacja API – emulujemy tylko bibliotekę systemową.

Porównanie

:)	Zalety	Wady
Emulacja	• nie trzeba instalować SO • emuluje dowolną architekturę • przenośna	• wolna – 20% wydajności natywnej
Emulacja API	• nie trzeba instalować SO • średnio szybkie (60%) :)	• tylko aplikacje na architekturę gospodarza • średnio szybkie (60%) :)

Parawirtualizacja

- system operacyjny gościa wie, że jest wirtualizowany
- dostosowuje swój sposób korzystania z zasobów systemowych, np. z pamięci; na przykład zamiast `syscall` wywołuje `hypercall`
- szybka: ponad 95% wydajności gospodarza
- trzeba jednak zmienić system operacyjny, zatem nie można jej wykorzystywać do wprowadzania zmian w jądrze

Parawirtualizacja

- system operacyjny gościa wie, że jest wirtualizowany
- dostosowuje swój sposób korzystania z zasobów systemowych, np. z pamięci; na przykład zamiast `syscall` wywołuje `hypercall`
- szybka: ponad 95% wydajności gospodarza
- trzeba jednak zmienić system operacyjny, zatem nie można jej wykorzystywać do wprowadzania zmian w jądrze

Parawirtualizacja

- system operacyjny gościa wie, że jest wirtualizowany
- dostosowuje swój sposób korzystania z zasobów systemowych, np. z pamięci; na przykład zamiast `syscall` wywołuje `hypercall`
- szybka: ponad 95% wydajności gospodarza
- trzeba jednak zmienić system operacyjny, zatem nie można jej wykorzystywać do wprowadzania zmian w jądrze

Parawirtualizacja

- system operacyjny gościa wie, że jest wirtualizowany
- dostosowuje swój sposób korzystania z zasobów systemowych, np. z pamięci; na przykład zamiast `syscall` wywołuje `hypercall`
- szybka: ponad 95% wydajności gospodarza
- trzeba jednak zmienić system operacyjny, zatem nie można jej wykorzystywać do wprowadzania zmian w jądrze

Parawirtualizacja

- system operacyjny gościa wie, że jest wirtualizowany
- dostosowuje swój sposób korzystania z zasobów systemowych, np. z pamięci; na przykład zamiast `syscall` wywołuje `hypercall`
- szybka: ponad 95% wydajności gospodarza
- trzeba jednak zmienić system operacyjny, zatem nie można jej wykorzystywać do wprowadzania zmian w jądrze

Wirtualizacja OS

Uruchamiamy system operacyjny gościa pod systemem operacyjnym gospodarza. Chcielibyśmy, żeby działało:

1. szybko
2. identycznie
3. i aby gość niczego nie wiedział (w szczególności by kontrolował zasoby)

Wirtualizacja OS

Uruchamiamy system operacyjny gościa pod systemem operacyjnym gospodarza. Chcielibyśmy, żeby działało:

1. szybko
2. identycznie
3. i aby gość niczego nie wiedział (w szczególności by kontrolował zasoby)

Wirtualizacja OS

Uruchamiamy system operacyjny gościa pod systemem operacyjnym gospodarza. Chcielibyśmy, żeby działało:

1. szybko
2. identycznie
3. i aby gość niczego nie wiedział (w szczególności by kontrolował zasoby)

Wirtualizacja OS

Uruchamiamy system operacyjny gościa pod systemem operacyjnym gospodarza. Chcielibyśmy, żeby działało:

1. szybko
2. identycznie
3. i aby gość niczego nie wiedział (w szczególności by kontrolował zasoby)

W idealnym-wirtualnym świecie („W koszyczku jabłuszko ...”)

- każda instrukcja zmieniająca zasoby systemowe lub zależna od konfiguracji sprzętu (wrażliwie) wywołuje przerwanie sprzętowe
- **kryterium Popka-Golberga** mówi, że jeśli tak jest, to można skonstruować wirtualną maszynę
- w x86 tak nie jest – i to jest problem

W idealnym-wirtualnym świecie („W koszyczku jabłuszko ...”)

- każda instrukcja zmieniająca zasoby systemowe lub zależna od konfiguracji sprzętu (wrażliwie) wywołuje przerwanie sprzętowe
- **kryterium Popka-Golberga** mówi, że jeśli tak jest, to można skonstruować wirtualną maszynę
- w x86 tak nie jest – i to jest problem

W idealnym-wirtualnym świecie („W koszyczku jabłuszko ...”)

- każda instrukcja zmieniająca zasoby systemowe lub zależna od konfiguracji sprzętu (wrażliwie) wywołuje przerwanie sprzętowe
- **kryterium Popka-Golberga** mówi, że jeśli tak jest, to można skonstruować wirtualną maszynę
- w x86 tak nie jest – i to jest problem

Opis problemu („W jabłuszku robaczek ...”)

Instrukcje, których nie lubimy

W architekturze x86 jest ich aż 17. Na przykład:

- `SGDT`, `SIDT`, `SLDT` – czytanie rejestrów
- `SMSW` – korzystanie ze stanu procesora
- `PUSH`, `POP` – zależne od trybu pracy (poziomu ochrony)
- `POPF` – a nawet zmienianie flag w `EFLAGS`
- ...

Rozwiązanie problemu

- zarządca przechwytuje instrukcje wrażliwe
- trzeba rekompilować kod, ale to trwa i trwa , ...
- a utrudniają nam to jeszcze instrukcje skoku. I zostaje jeszcze zmieniający się kod
- Zatem stosujemy flagi (i wcale nas to nie bawi)

Rozwiązanie problemu

- zarządca przechwytuje instrukcje wrażliwe
- trzeba rekompilować kod, ale to trwa i trwa , ...
- a utrudniają nam to jeszcze instrukcje skoku. I zostaje jeszcze zmieniający się kod
- Zatem stosujemy flagi (i wcale nas to nie bawi)

Rozwiązanie problemu

- zarządca przechwytuje instrukcje wrażliwe
- trzeba rekompilować kod, ale to trwa i trwa , ...
- a utrudniają nam to jeszcze instrukcje skoku. I zostaje jeszcze zmieniający się kod
- Zatem stosujemy flagi (i wcale nas to nie bawi)

Rozwiązanie problemu

- zarządca przechwytuje instrukcje wrażliwe
- trzeba rekompilować kod, ale to trwa i trwa , ...
- a utrudniają nam to jeszcze instrukcje skoku. I zostaje jeszcze zmieniający się kod
- Zatem stosujemy flagi (i wcale nas to nie bawi)

Rozwiązanie problemu

- zarządca przechwytuje instrukcje wrażliwe
- trzeba rekompilować kod, ale to trwa i trwa , ...
- a utrudniają nam to jeszcze instrukcje skoku. I zostaje jeszcze zmieniający się kod
- Zatem stosujemy flagi (i wcale nas to nie bawi)

Rozwiązanie problemu

- zarządca przechwytuje instrukcje wrażliwe
- trzeba rekompilować kod, ale to trwa i trwa , ...
- a utrudniają nam to jeszcze instrukcje skoku. I zostaje jeszcze zmieniający się kod
- Zatem stosujemy flagi (i wcale nas to nie bawi)

Rozwiązanie problemu

- zarządca przechwytuje instrukcje wrażliwe
- trzeba rekompilować kod, ale to trwa i trwa , ...
- a utrudniają nam to jeszcze instrukcje skoku. I zostaje jeszcze zmieniający się kod
- Zatem stosujemy flagi (i wcale nas to nie bawi)

Rozwiązanie problemu

- zarządca przechwytuje instrukcje wrażliwe
- trzeba rekompilować kod, ale to trwa i trwa , ...
- a utrudniają nam to jeszcze instrukcje skoku. I zostaje jeszcze samozmieniający się kod
- Zatem stosujemy flagi (i wcale nas to nie bawi)

Rozwiązanie problemu

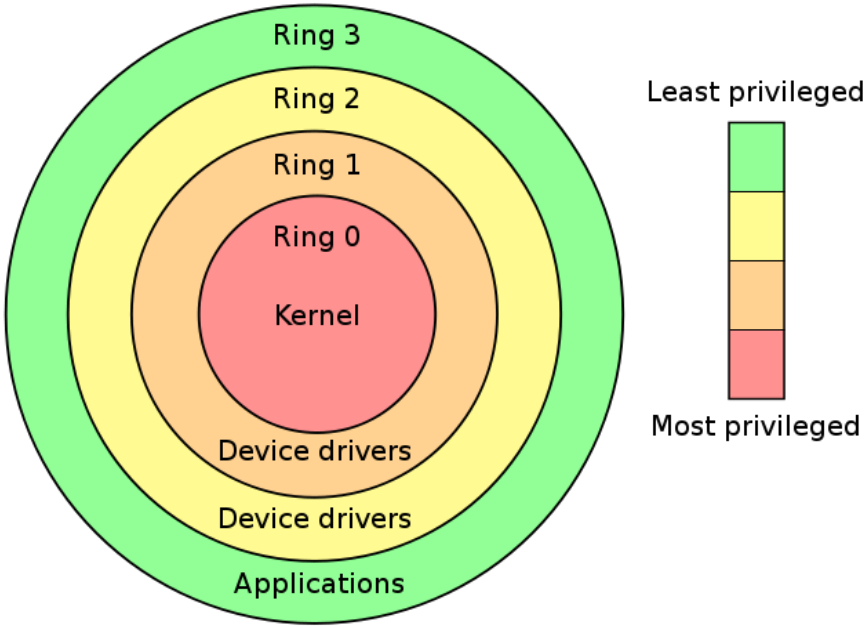
- zarządca przechwytuje instrukcje wrażliwe
- trzeba rekompilować kod, ale to trwa i trwa , ...
- a utrudniają nam to jeszcze instrukcje skoku. I zostaje jeszcze samozmieniający się kod
- Zatem stosujemy flagi (i wcale nas to nie bawi)

Rozwiązanie problemu

- zarządca przechwytuje instrukcje wrażliwe
- trzeba rekompilować kod, ale to trwa i trwa , ...
- a utrudniają nam to jeszcze instrukcje skoku. I zostaje jeszcze samozmieniający się kod
- Zatem stosujemy flagi (i wcale nas to nie bawi)

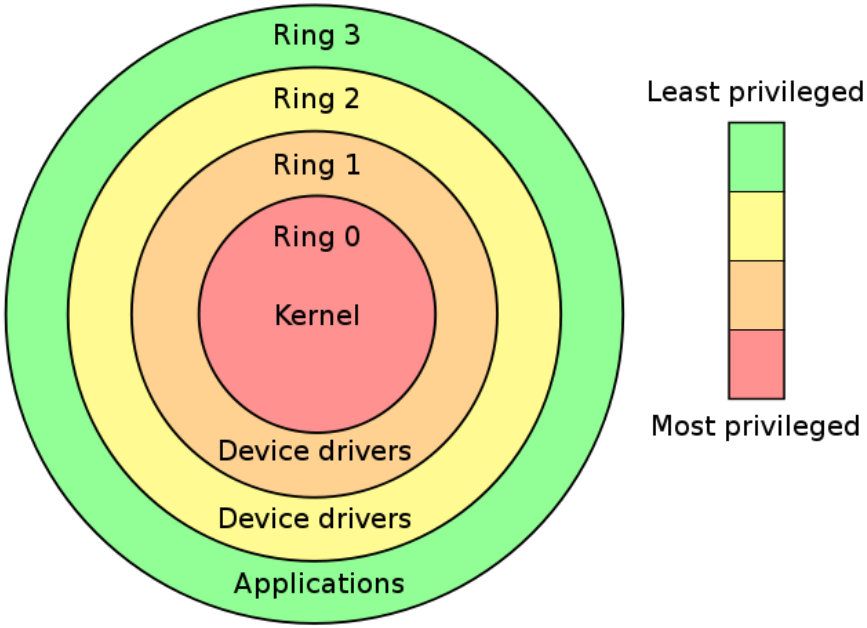
Ring – co to takiego?

1. *Ringi* ochronne – hierarchiczny system ochronny
2. Ringi określają uprawnienia dostępu do poszczególnych zasobów. Im niższy numer Ringu, tym większe uprawnienia
3. np. Ring zerowy: jądro systemu



Ring – co to takiego?

1. *Ringi* ochronne – hierarchiczny system ochronny
2. Ringi określają uprawnienia dostępu do poszczególnych zasobów. Im niższy numer Ringu, tym większe uprawnienia
3. np. Ring zerowy: jądro systemu



Ring – co to takiego?

Ringi zapewniają separację zasobów, pozwalającą na ochronę przed:

- błędami krytycznymi – niewiele procesów ma uprawnienia pozwalające na uszkodzenie podstawowej funkcjonalności OS-u
- złośliwym oprogramowaniem – program spyware uruchomiony w Ringu 3 nie będzie miał dostępu do kamery internetowej, wymagającej Ringu 1

Ring – co to takiego?

Ringi zapewniają separację zasobów, pozwalającą na ochronę przed:

- błędami krytycznymi – niewiele procesów ma uprawnienia pozwalające na uszkodzenie podstawowej funkcjonalności OS-u
- złośliwym oprogramowaniem – program spyware uruchomiony w Ringu 3 nie będzie miał dostępu do kamery internetowej, wymagającej Ringu 1

Ring – co to takiego?

Ringi zapewniają separację zasobów, pozwalającą na ochronę przed:

- błędami krytycznymi – niewiele procesów ma uprawnienia pozwalające na uszkodzenie podstawowej funkcjonalności OS-u
- złośliwym oprogramowaniem – program spyware uruchomiony w Ringu 3 nie będzie miał dostępu do kamery internetowej, wymagającej Ringu 1

Co z wirtualizacją? Ring -1

1. jądro systemu wymaga do działania Ringu 0
2. nie chcemy, by systemy odpalane na wirtualnej maszynie miały pełny dostęp do systemu-hosta
3. stworzenie Ringu -1 – superuprzywilejowanego trybu, który może być używany wyłącznie przez maszynę wirtualną
4. Ring -1 zapewnia obsługę instrukcji zmieniających zasoby, które nie wywołują przerwań sprzętowych (była o nich mowa wcześniej)

Co z wirtualizacją? Ring -1

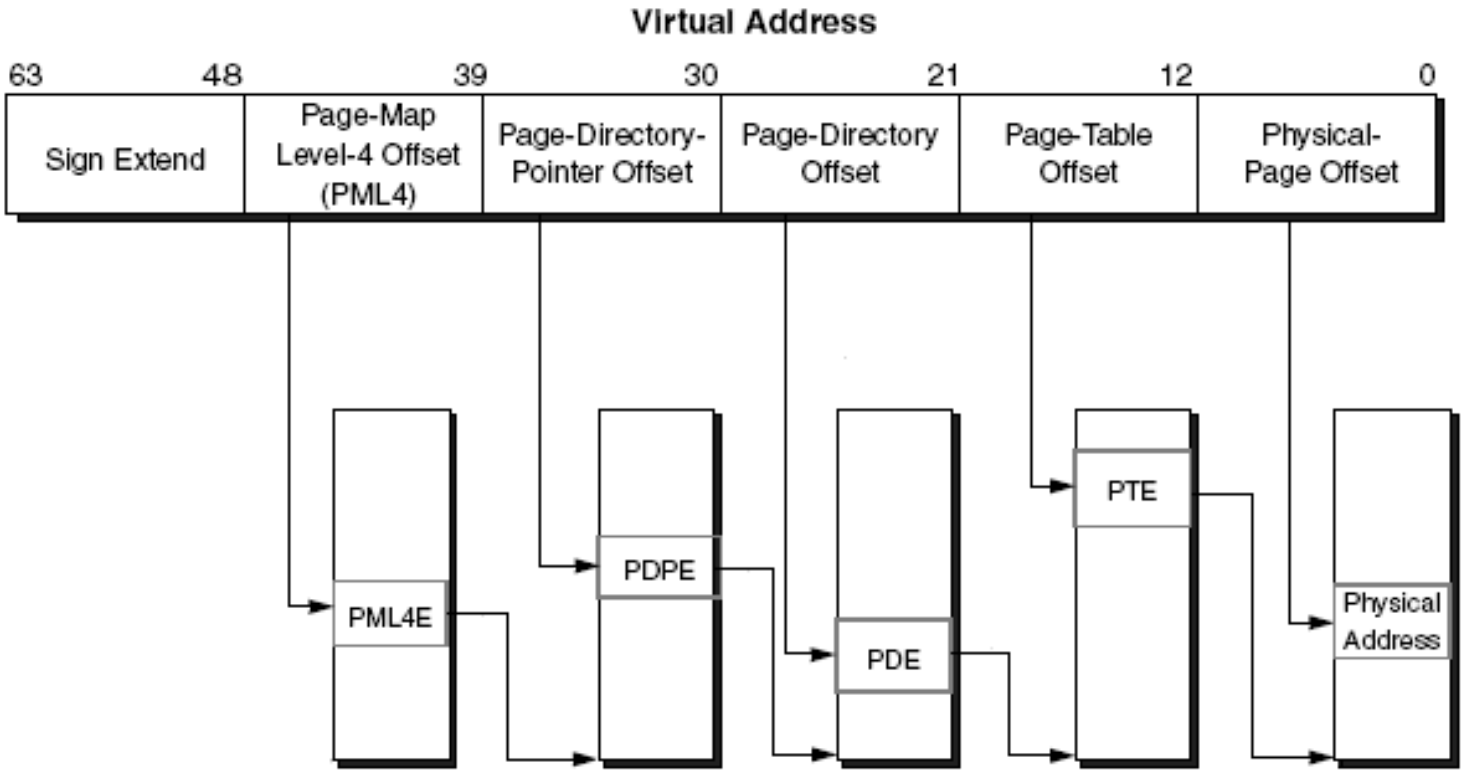
1. jądro systemu wymaga do działania Ringu 0
2. nie chcemy, by systemy odpalane na wirtualnej maszynie miały pełny dostęp do systemu-hosta
3. stworzenie Ringu -1 – superuprzywilejowanego trybu, który może być używany wyłącznie przez maszynę wirtualną
4. Ring -1 zapewnia obsługę instrukcji zmieniających zasoby, które nie wywołują przerwań sprzętowych (była o nich mowa wcześniej)

Co z wirtualizacją? Ring -1

1. jądro systemu wymaga do działania Ringu 0
2. nie chcemy, by systemy odpalane na wirtualnej maszynie miały pełny dostęp do systemu-hosta
3. stworzenie Ringu -1 – superuprzywilejowanego trybu, który może być używany wyłącznie przez maszynę wirtualną
4. Ring -1 zapewnia obsługę instrukcji zmieniających zasoby, które nie wywołują przerwań sprzętowych (była o nich mowa wcześniej)

Stronicowanie na komputerze niewirtualizowanym

Zwykły system operacyjny, translacja adresu liniowego do fizycznego z użyciem *page walkera*:



TLB – translation look-aside buffer

Translacja adresu jest bardzo obciążającą operacją, gdyż *page walker* musi odczytywać pamięć wiele razy

- rozwiązanie: tablica ostatnich translacji (TLB – translation look-aside buffer)
- procesor najpierw sprawdza, czy żądana translacja nie znajduje się w TLB; dopiero w przeciwnym wypadku przeprowadza ją od początku

TLB – translation look-aside buffer

Translacja adresu jest bardzo obciążającą operacją, gdyż *page walker* musi odczytywać pamięć wiele razy

- rozwiązanie: tablica ostatnich translacji (TLB – translation look-aside buffer)
- procesor najpierw sprawdza, czy żądana translacja nie znajduje się w TLB; dopiero w przeciwnym wypadku przeprowadza ją od początku

TLB – translation look-aside buffer

System operacyjny musi współpracować z procesorem w celu utrzymania spójności TLB, np.:

- po przełączeniu kontekstu możemy stracić uprawnienia dotyczące jakiegoś fragmentu pamięci – system operacyjny musi zażądać od CPU usunięcia odpowiedniego wpisu w TLB
- dodatkowy narzut przy użyciu wielu procesorów (SMP – symmetric multiprocessing – współdzielenie pamięci przez wiele procesorów) – jeśli tablica jest współdzielona, CPU musi usunąć odpowiedni wpis z TLB wszystkich procesorów

TLB – translation look-aside buffer

System operacyjny musi współpracować z procesorem w celu utrzymania spójności TLB, np.:

- po przełączeniu kontekstu możemy stracić uprawnienia dotyczące jakiegoś fragmentu pamięci – system operacyjny musi zażądać od CPU usunięcia odpowiedniego wpisu w TLB
- dodatkowy narzut przy użyciu wielu procesorów (SMP – symmetric multiprocessing – współdzielenie pamięci przez wiele procesorów) – jeśli tablica jest współdzielona, CPU musi usunąć odpowiedni wpis z TLB wszystkich procesorów

TLB – translation look-aside buffer

System operacyjny musi współpracować z procesorem w celu utrzymania spójności TLB, np.:

- po przełączeniu kontekstu możemy stracić uprawnienia dotyczące jakiegoś fragmentu pamięci – system operacyjny musi zażądać od CPU usunięcia odpowiedniego wpisu w TLB
- dodatkowy narzut przy użyciu wielu procesorów (SMP – symmetric multiprocessing – współdzielenie pamięci przez wiele procesorów) – jeśli tablica jest współdzielona, CPU musi usunąć odpowiedni wpis z TLB wszystkich procesorów

Obsługa translacji adresów gościa

Zakładamy, że stosujemy pełną wirtualizację (gość nie wie, że jest wirtualizowany).

- nadzorca musi kontrolować translacje adresów wykonywane przez gościa
- dodatkowy poziom translacji (translacja adresu „fizycznego” gościa do adresu fizycznego systemu)
- powoduje to dodatkowe obciążenie nadzorcy

Obsługa translacji adresów gościa

Zakładamy, że stosujemy pełną wirtualizację (gość nie wie, że jest wirtualizowany).

- nadzorca musi kontrolować translacje adresów wykonywane przez gościa
- dodatkowy poziom translacji (translacja adresu „fizycznego” gościa do adresu fizycznego systemu)
- powoduje to dodatkowe obciążenie nadzorcy

Obsługa translacji adresów gościa

Zakładamy, że stosujemy pełną wirtualizację (gość nie wie, że jest wirtualizowany).

- nadzorca musi kontrolować translacje adresów wykonywane przez gościa
- dodatkowy poziom translacji (translacja adresu „fizycznego” gościa do adresu fizycznego systemu)
- powoduje to dodatkowe obciążenie nadzorcy

Shadow page tables (sPT)

1. nadzorca żąda od CPU używania *shadow page table* (sPT) do przeprowadzania translacji adresów
2. sPT tłumaczy liniowy adres gościa na fizyczny adres systemu
3. nadzorca musi utrzymywać sPT w stanie spójnym
4. jedna z metod – nałożenie ochrony przed zapisem na całą pamięć gościa
   - każda próba dostępu gościa do jego tablicy stron (gPT) wywołuje błąd ochrony pamięci, przechwytywany przez CPU i przekazujący obsługę do nadzorcy
   - nadzorca wykonuje żądanie gościa i poprawia sPT
   - analogiczna obsługa w przypadku usuwania translacji przez gościa

Shadow page tables (sPT)

1. nadzorca żąda od CPU używania *shadow page table* (sPT) do przeprowadzania translacji adresów
2. sPT tłumaczy liniowy adres gościa na fizyczny adres systemu
3. nadzorca musi utrzymywać sPT w stanie spójnym
4. jedna z metod – nałożenie ochrony przed zapisem na całą pamięć gościa
   - każda próba dostępu gościa do jego tablicy stron (gPT) wywołuje błąd ochrony pamięci, przechwytywany przez CPU i przekazujący obsługę do nadzorcy
   - nadzorca wykonuje żądanie gościa i poprawia sPT
   - analogiczna obsługa w przypadku usuwania translacji przez gościa

Shadow page tables (sPT)

1. nadzorca żąda od CPU używania *shadow page table* (sPT) do przeprowadzania translacji adresów
2. sPT tłumaczy liniowy adres gościa na fizyczny adres systemu
3. nadzorca musi utrzymywać sPT w stanie spójnym
4. jedna z metod – nałożenie ochrony przed zapisem na całą pamięć gościa
   - każda próba dostępu gościa do jego tablicy stron (gPT) wywołuje błąd ochrony pamięci, przechwytywany przez CPU i przekazujący obsługę do nadzorcy
   - nadzorca wykonuje żądanie gościa i poprawia sPT
   - analogiczna obsługa w przypadku usuwania translacji przez gościa

Shadow page tables (sPT)

1. nadzorca żąda od CPU używania *shadow page table* (sPT) do przeprowadzania translacji adresów
2. sPT tłumaczy liniowy adres gościa na fizyczny adres systemu
3. nadzorca musi utrzymywać sPT w stanie spójnym
4. jedna z metod – nałożenie ochrony przed zapisem na całą pamięć gościa
   - każda próba dostępu gościa do jego tablicy stron (gPT) wywołuje błąd ochrony pamięci, przechwytywany przez CPU i przekazujący obsługę do nadzorcy
   - nadzorca wykonuje żądanie gościa i poprawia sPT
   - analogiczna obsługa w przypadku usuwania translacji przez gościa

Inne rozwiązanie (Virtual TLB)

- nie uaktualniamy sPT po każdej modyfikacji gPT przez gościa – zezwalamy na chwilową niespójność
- jeśli gość odwoła się do adresu, którego obsługa nie znajduje się w sPT, zostaje wywołany błąd ochrony pamięci
- nadzorca czyta całą tablicę gPT i wstawia brakujące translacje do sPT, przywracając spójność; następnie obsługuje żądanie gościa
- analogicznie dla usuwania

Inne rozwiązanie (Virtual TLB)

- nie uaktualniamy sPT po każdej modyfikacji gPT przez gościa – zezwalamy na chwilową niespójność
- jeśli gość odwoła się do adresu, którego obsługa nie znajduje się w sPT, zostaje wywołany błąd ochrony pamięci
- nadzorca czyta całą tablicę gPT i wstawia brakujące translacje do sPT, przywracając spójność; następnie obsługuje żądanie gościa
- analogicznie dla usuwania

Inne rozwiązanie (Virtual TLB)

- nie uaktualniamy sPT po każdej modyfikacji gPT przez gościa – zezwalamy na chwilową niespójność
- jeśli gość odwoła się do adresu, którego obsługa nie znajduje się w sPT, zostaje wywołany błąd ochrony pamięci
- nadzorca czyta całą tablicę gPT i wstawia brakujące translacje do sPT, przywracając spójność; następnie obsługuje żądanie gościa
- analogicznie dla usuwania

Wady takich rozwiązań

1. duża ilość błędów ochrony pamięci, których obsługa jest bardzo kosztowna
2. trudno odróżnić faktyczne błędy ochrony pamięci gościa od spowodowanych przez zaprezentowane rozwiązania wirtualizacyjne
3. każde przełączenie kontekstu wymaga poprawienia sPT – spory narzut
4. dodatkowe problemy dla gości z SMP – gPT może być współdzielona przez wiele procesorów. Nadzorca może:
   - dla każdego procesora utrzymywać osobną instancję sPT – spory dodatkowy narzut przy poprawianiu sPT
   - utrzymywać współdzieloną sPT dla wielu wirtualnych procesorów – duże koszty synchronizacji

Wady takich rozwiązań

1. duża ilość błędów ochrony pamięci, których obsługa jest bardzo kosztowna
2. trudno odróżnić faktyczne błędy ochrony pamięci gościa od spowodowanych przez zaprezentowane rozwiązania wirtualizacyjne
3. każde przełączenie kontekstu wymaga poprawienia sPT – spory narzut
4. dodatkowe problemy dla gości z SMP – gPT może być współdzielona przez wiele procesorów. Nadzorca może:
   - dla każdego procesora utrzymywać osobną instancję sPT – spory dodatkowy narzut przy poprawianiu sPT
   - utrzymywać współdzieloną sPT dla wielu wirtualnych procesorów – duże koszty synchronizacji

Wady takich rozwiązań

1. duża ilość błędów ochrony pamięci, których obsługa jest bardzo kosztowna
2. trudno odróżnić faktyczne błędy ochrony pamięci gościa od spowodowanych przez zaprezentowane rozwiązania wirtualizacyjne
3. każde przełączenie kontekstu wymaga poprawienia sPT – spory narzut
4. dodatkowe problemy dla gości z SMP – gPT może być współdzielona przez wiele procesorów. Nadzorca może:
   - dla każdego procesora utrzymywać osobną instancję sPT – spory dodatkowy narzut przy poprawianiu sPT
   - utrzymywać współdzieloną sPT dla wielu wirtualnych procesorów – duże koszty synchronizacji

Wady takich rozwiązań

1. duża ilość błędów ochrony pamięci, których obsługa jest bardzo kosztowna
2. trudno odróżnić faktyczne błędy ochrony pamięci gościa od spowodowanych przez zaprezentowane rozwiązania wirtualizacyjne
3. każde przełączenie kontekstu wymaga poprawienia sPT – spory narzut
4. dodatkowe problemy dla gości z SMP – gPT może być współdzielona przez wiele procesorów. Nadzorca może:
   - dla każdego procesora utrzymywać osobną instancję sPT – spory dodatkowy narzut przy poprawianiu sPT
   - utrzymywać współdzieloną sPT dla wielu wirtualnych procesorów – duże koszty synchronizacji

Wady takich rozwiązań

1. duża ilość błędów ochrony pamięci, których obsługa jest bardzo kosztowna
2. trudno odróżnić faktyczne błędy ochrony pamięci gościa od spowodowanych przez zaprezentowane rozwiązania wirtualizacyjne
3. każde przełączenie kontekstu wymaga poprawienia sPT – spory narzut
4. dodatkowe problemy dla gości z SMP – gPT może być współdzielona przez wiele procesorów. Nadzorca może:
   - dla każdego procesora utrzymywać osobną instancję sPT – spory dodatkowy narzut przy poprawianiu sPT
   - utrzymywać współdzieloną sPT dla wielu wirtualnych procesorów – duże koszty synchronizacji

Wady takich rozwiązań

1. duża ilość błędów ochrony pamięci, których obsługa jest bardzo kosztowna
2. trudno odróżnić faktyczne błędy ochrony pamięci gościa od spowodowanych przez zaprezentowane rozwiązania wirtualizacyjne
3. każde przełączenie kontekstu wymaga poprawienia sPT – spory narzut
4. dodatkowe problemy dla gości z SMP – gPT może być współdzielona przez wiele procesorów. Nadzorca może:
   - dla każdego procesora utrzymywać osobną instancję sPT – spory dodatkowy narzut przy poprawianiu sPT
   - utrzymywać współdzieloną sPT dla wielu wirtualnych procesorów – duże koszty synchronizacji

Wady takich rozwiązań

Szacuje się, że obsługa sPT może pochłaniać aż do 75% całego obciążenia nadzorcy.

Nested page tables

1. dodatkowa tablica nPT (nested page table), obsługiwana sprzętowo
2. nPT tłumaczy „fizyczne” adresy gościa na fizyczne adresy systemu, gPT mapuje liniowy adres gościa do „fizycznego” adresu gościa
3. gość zachowuje całkowitą kontrolę nad swoimi tablicami stron

Nested page tables

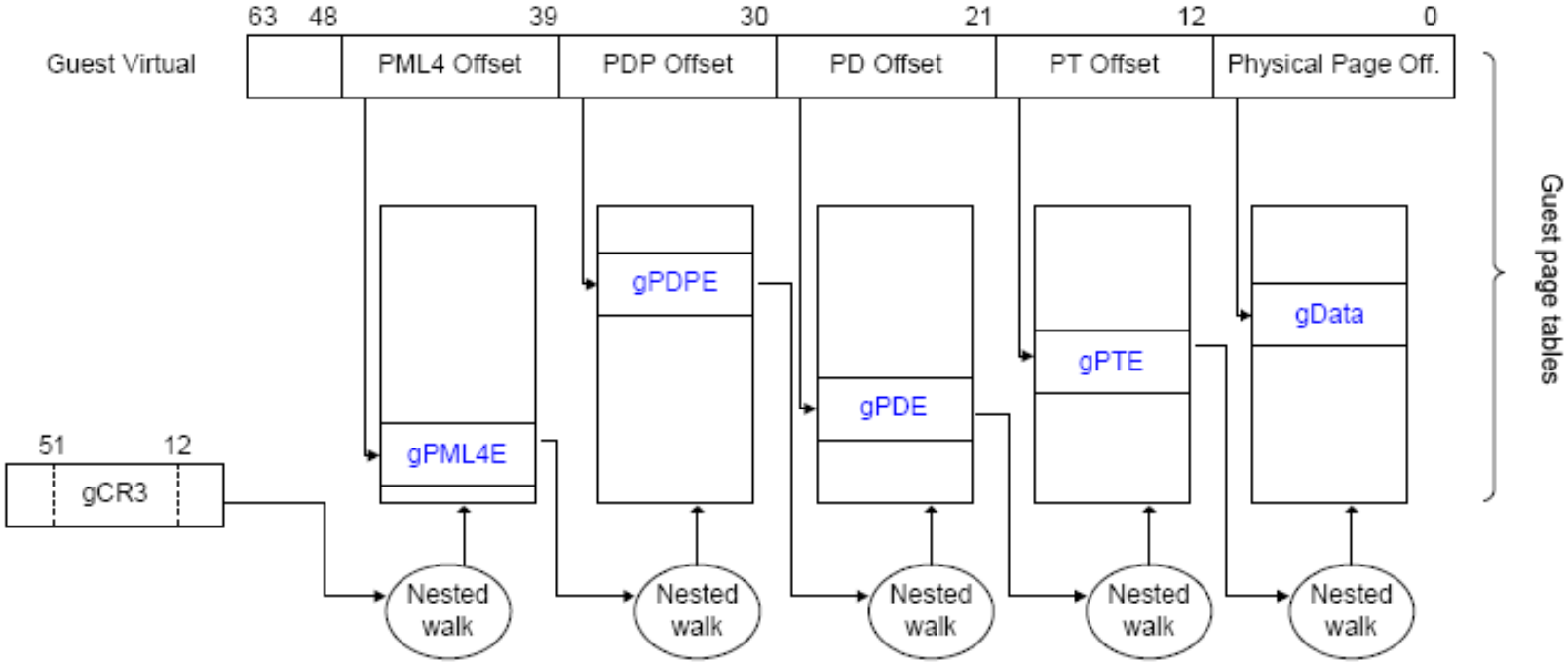
1. dodatkowa tablica nPT (nested page table), obsługiwana sprzętowo
2. nPT tłumaczy „fizyczne” adresy gościa na fizyczne adresy systemu, gPT mapuje liniowy adres gościa do „fizycznego” adresu gościa
3. gość zachowuje całkowitą kontrolę nad swoimi tablicami stron

Nested page tables

1. dodatkowa tablica nPT (nested page table), obsługiwana sprzętowo
2. nPT tłumaczy „fizyczne” adresy gościa na fizyczne adresy systemu, gPT mapuje liniowy adres gościa do „fizycznego” adresu gościa
3. gość zachowuje całkowitą kontrolę nad swoimi tablicami stron

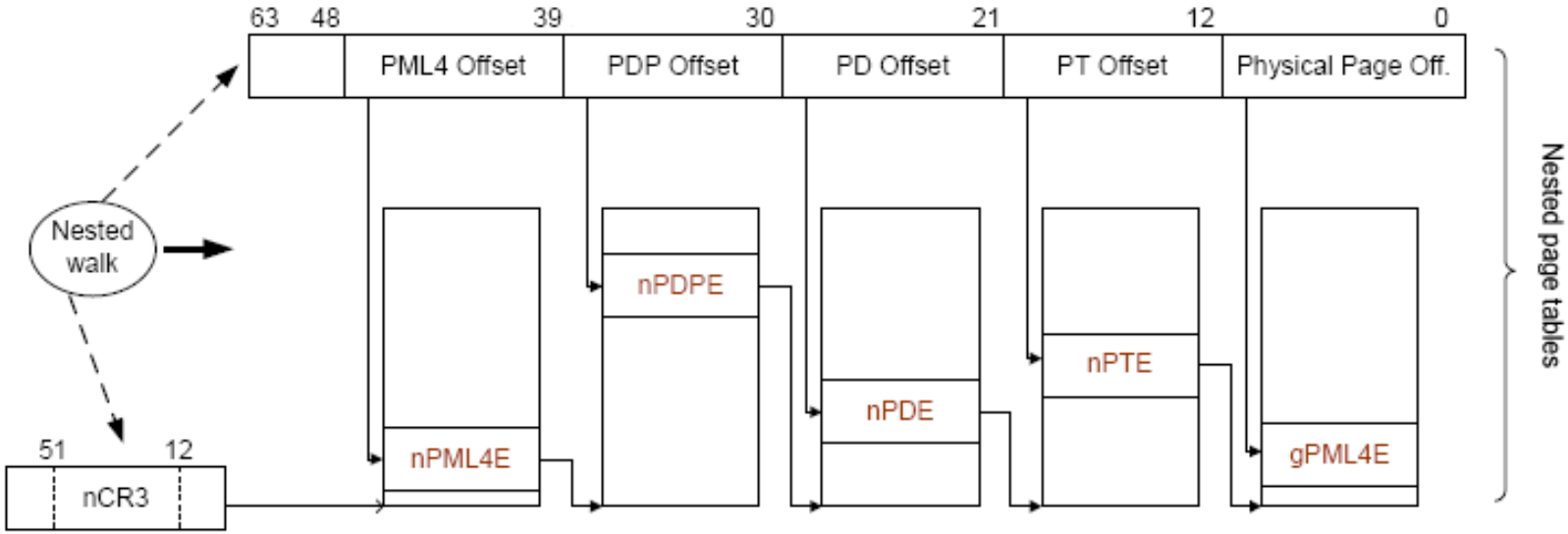
Praca *page walkera* przy użyciu nPT

Page walker wędruje jednocześnie po gPT i nPT, tłumacząc w ten sposób liniowy adres gościa na fizyczny adres systemu.



Praca *page walkera* przy użyciu nPT

Page walker wędruje jednocześnie po gPT i nPT, tłumacząc w ten sposób liniowy adres gościa na fizyczny adres systemu.



nPT a tablice TLB

1. Tryb gościa i tryb hosta:
   - w trybie gościa TLB mapuje liniowe adresy gościa na fizyczne adresy systemu
   - w trybie hosta TLB mapuje liniowe adresy hosta na fizyczne adresy systemu
2. Nested TLB – mapuje „fizyczne” adresy gościa na fizyczne adresy systemu
   - **WAŻNE!**

nPT a tablice TLB

1. Tryb gościa i tryb hosta:
   - w trybie gościa TLB mapuje liniowe adresy gościa na fizyczne adresy systemu
   - w trybie hosta TLB mapuje liniowe adresy hosta na fizyczne adresy systemu
2. Nested TLB – mapuje „fizyczne” adresy gościa na fizyczne adresy systemu
   - **WAŻNE!**

nPT a tablice TLB

1. Tryb gościa i tryb hosta:
   - w trybie gościa TLB mapuje liniowe adresy gościa na fizyczne adresy systemu
   - w trybie hosta TLB mapuje liniowe adresy hosta na fizyczne adresy systemu
2. Nested TLB – mapuje „fizyczne” adresy gościa na fizyczne adresy systemu
   - **WAŻNE!**

nPT a tablice TLB

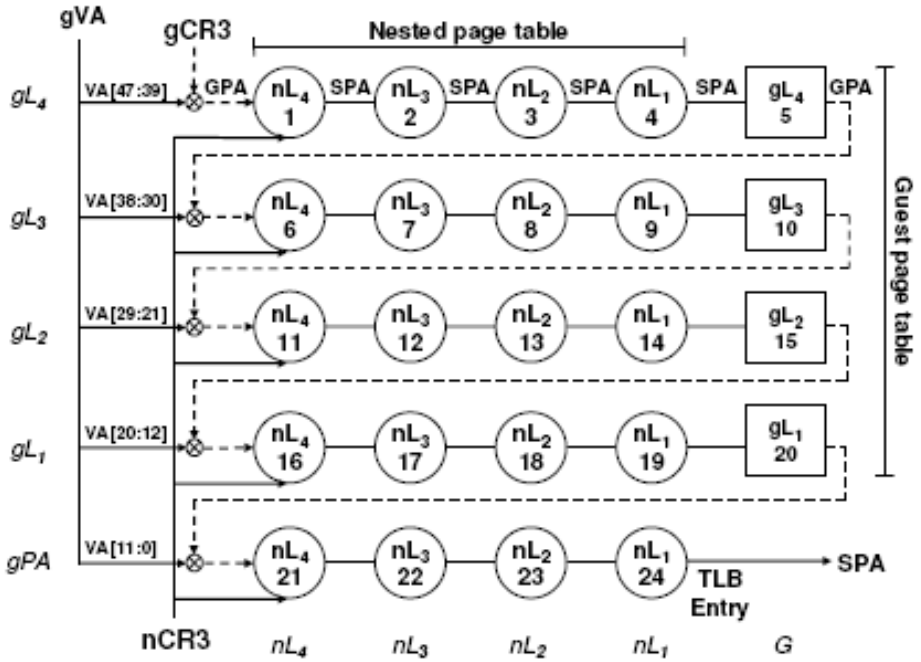
1. Tryb gościa i tryb hosta:
   - w trybie gościa TLB mapuje liniowe adresy gościa na fizyczne adresy systemu
   - w trybie hosta TLB mapuje liniowe adresy hosta na fizyczne adresy systemu
2. Nested TLB – mapuje „fizyczne” adresy gościa na fizyczne adresy systemu
   - WAŻNE!

nPT a tablice TLB

1. Tryb gościa i tryb hosta:
   - w trybie gościa TLB mapuje liniowe adresy gościa na fizyczne adresy systemu
   - w trybie hosta TLB mapuje liniowe adresy hosta na fizyczne adresy systemu
2. Nested TLB – mapuje „fizyczne” adresy gościa na fizyczne adresy systemu
   - **WAŻNE!**

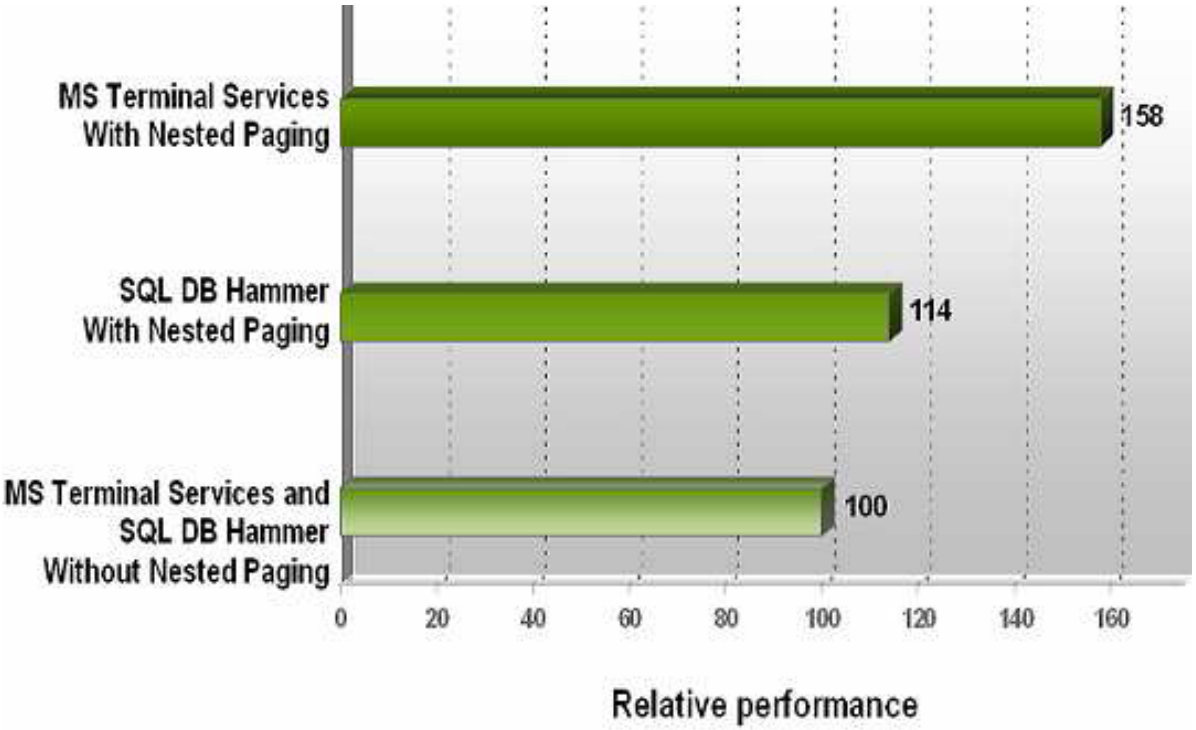
Po co Nested TLB?

Przykład – 4-poziomowa tablica stron gościa. Każda kolejna iteracja wymaga przejścia przez *page walkera* tablicy nPT. W tym przypadku bez nPT mamy 4 odwołania do pamięci; jeśli zaś mamy nPT – do 24:



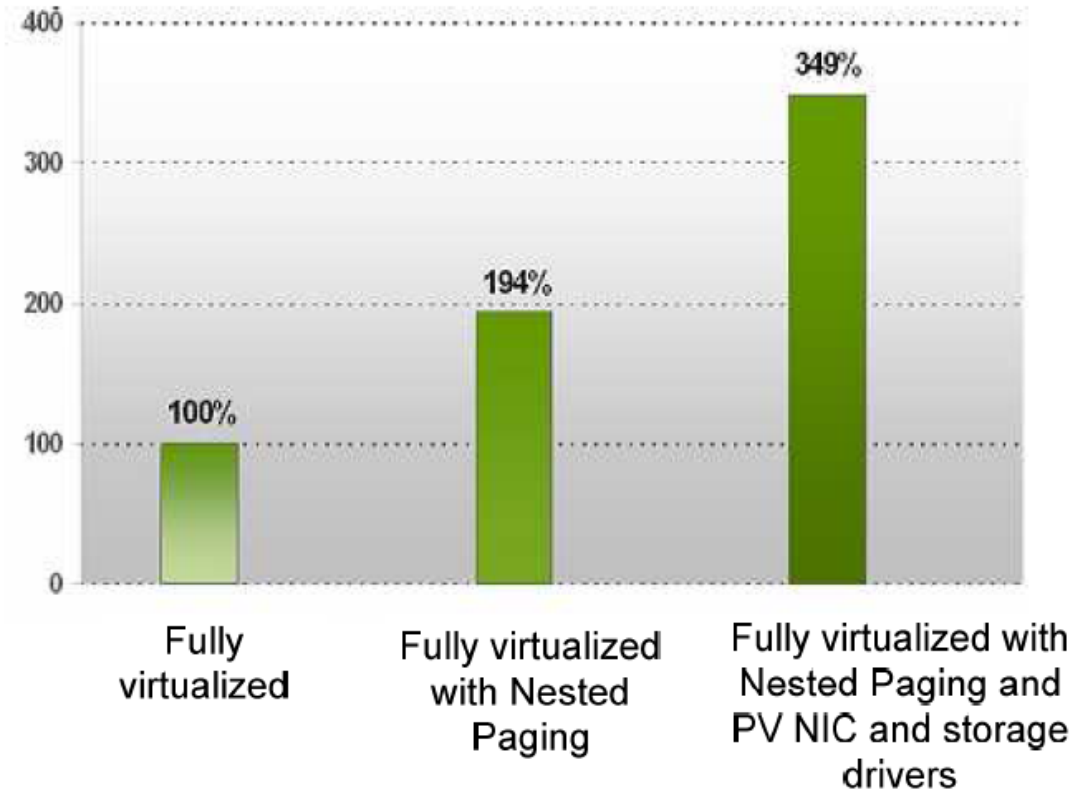
Parę testów

Wydajność z i bez użycia nested paging dla MS Terminal Services oraz SQL DB Hammer, uruchomione pod eksperymentalnym buildem VMware ESX:

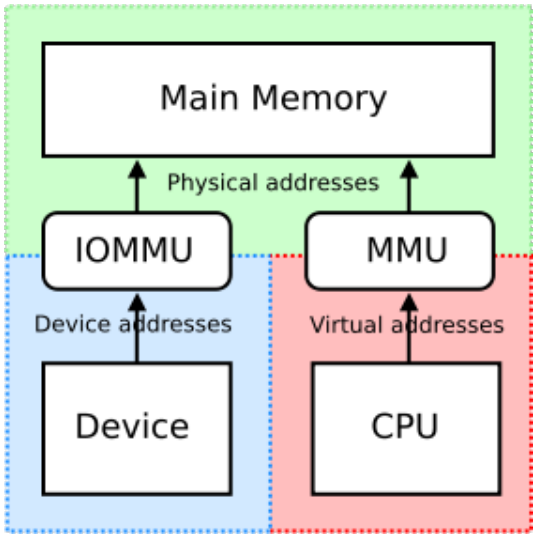


Parę testów

Oracle 10G OLTP z i bez użycia nested paging, pod Red Hat Enterprise Linux uruchomionym pod Xen 3.1:



IOMMU



- IOMMU jest jednostką zarządzania pamięcią
- podobny do zwykłego MMU, który tłumaczy widoczne dla procesora adresy wirtualne na adresy fizyczne
- IOMMU zajmuje się tłumaczeniem widocznych dla urządzeń adresów wirtualnych na adresy fizyczne

Gdzie można znaleźć IOMMU?

- AMD opublikował specyfikację dla IOMMU w architekturze HyperTransport
- Intel – IOMMU jako Virtualization Technology for Directed I/O, oznaczana jako VT-d
- Sun IOMMU – Device Virtual Memory Access (DVMA)
- IBM – The IBM Translation Control Entry (TCE) – Logical Partition Security in the IBM eServer pSeries 690

IOMMU – zalety

- można przydzielać duże obszary pamięci bez konieczności, aby były one ciągłe w pamięci fizycznej; pozwala to czasem uniknąć używania wektorowego I/O
- urządzenia, które nie są w stanie stworzyć dostatecznie dużych adresów, aby pokryć nimi całą pamięć fizyczną dzięki IOMMU mogą odwoływać się do całej pamięci
- ochrona pamięci: urządzenie nie może czytać lub pisać do pamięci, która nie została mu przydzielona
- w niektórych architekturach IOMMU może także przekierowywać sprzętowe przerwania w sposób podobny do standardowego przekierowywania adresów

IOMMU – wady

Wady IOMMU, w porównaniu do bezpośredniego fizycznego adresowania pamięci:

- pewne straty w wydajności z powodu translacji adresów i konieczności zarządzania nimi
- większe zużycie pamięci fizycznej z powodu dodania tablicy stron dla I/O

Zastosowania IOMMU poza wirtualizacją

The Graphics Address Remapping Table (GART) jest to zarządca pamięci wejścia/wyjścia (IOMMU) wykorzystywany przez karty graficzne używające złącz AGP i PCI Express. Dzięki niemu tekstury, wielokąty i inne potrzebne dane wczytywane są bezpośrednio z pamięci fizycznej komputera z wykorzystaniem DMA.

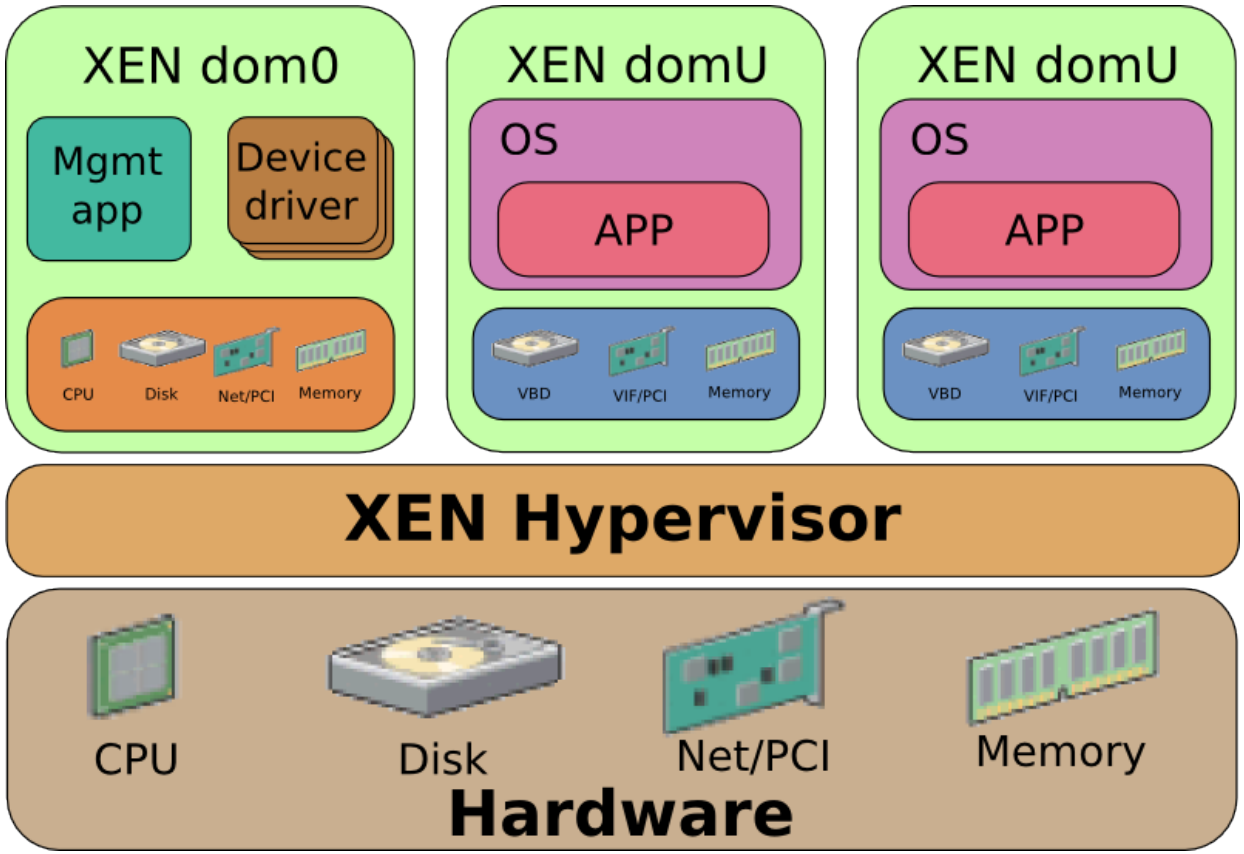
XEN

- monitor maszyn wirtualnych dla IA-32, x86, x86-64, IA-64 oraz PowerPC 97
- zarządza współpracą maszyn wirtualnych uruchomionych w jego środowisku
- Xen działa na zasadzie parawirtualizacji. – nie emuluje dokładnie istniejącego sprzętu, lecz pewne jego podobieństwo

XEN – trochę historii

- Xen rozwijany był przez grupę z Uniwersytetu Cambridge jako część projektu XenoServers
- z czasem Xen uzyskał status samodzielnego projektu
- dzięki bardzo dużej efektywność w porównaniu do innych programów służących do wirtualizacji powstał komercyjny projekt XenEnterprise

XEN – schemat działania

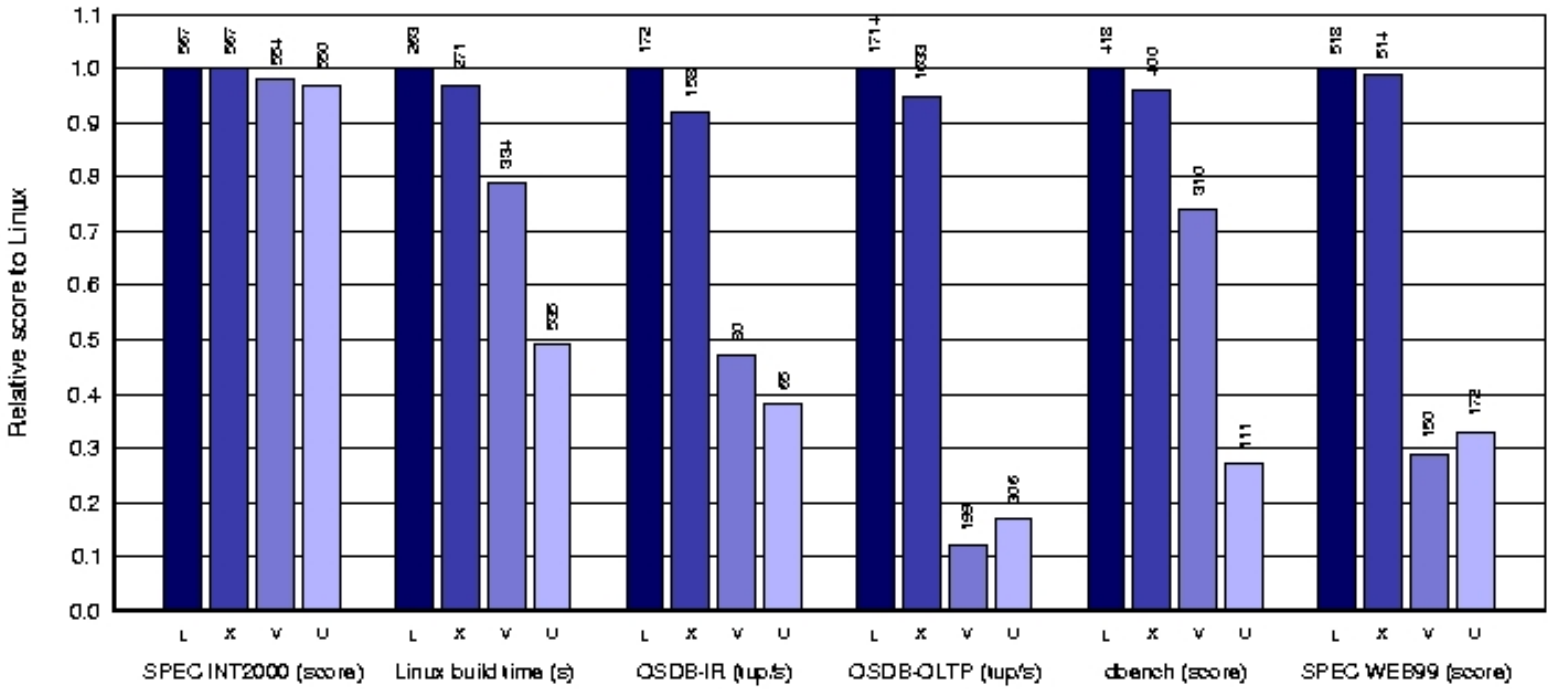


XEN – zastosowania

- konsolidacja serwerów: Xen umożliwia uruchomienie wielu serwerów równocześnie na jednej maszynie, zachowując efektywność oraz zapewniając pełną izolację

- jednoczesne uruchamianie wielu systemów operacyjnych
- testowanie jądra

Jak szybki jest XEN?



Linux (L), Xen/Linux (X), VMware (V), UML (U)

KVM – Kernel-based Virtual Machine

- komponent KVM pojawił się w jądrze Linuksa wraz z wersją 2.6.20 (luty 2007)
- wykorzystuje wirtualizację wspomaganą sprzętowo
- w pełni wirtualizowane rozwiązanie dla systemów Linux
- mały i względnie prosty (ok. 10000 linii kodu)

KVM

Podobieństwa i różnice w stosunku do:

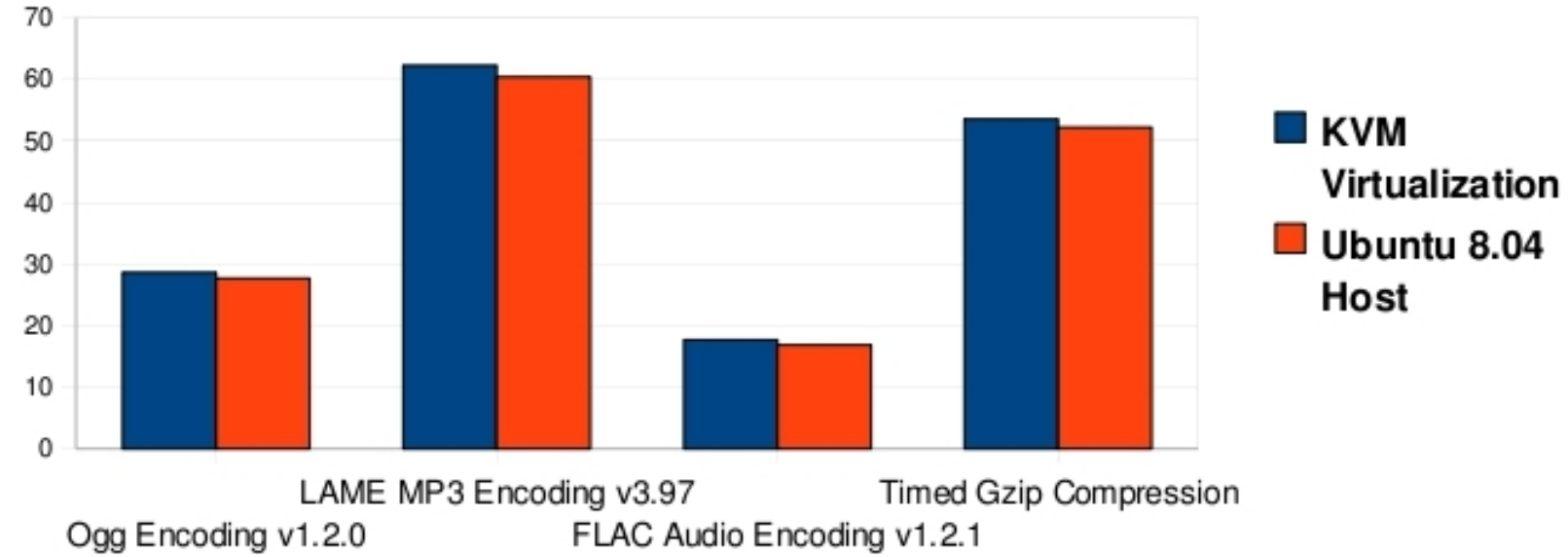
1. VMware – bardzo złożony, komercyjny
2. Xen – korzysta z Linuxa tylko do obsługi I/O. Ma własne: scheduler, moduł zarządzania pamięcią. Działa bez sprzętowego wsparcia dla wirtualizacji

KVM

1. **Ograniczenia**
   - wymaga procesora x86, który wspiera wirtualizację
   - nie wykorzystuje maszyn wieloprocesorowych
2. **Plany na przyszłość**
   - wsparcie dla parawirtualizacji
   - obsługa Power PC
   - obsługa IA-64
   - obsługa starszych platform takich jak Windows 95 i 98 SE
   - obsługa maszyn wieloprocesorowych

KVM – wydajność

	Ogg Encoding v1.2.0	LAME MP3 Encoding v3.97	FLAC Audio Encoding v1.2.1	Timed Gzip Compression
KVM Virtualization	28.61	62.2	17.71	53.5
Ubuntu 8.04 Host	27.66	60.34	16.86	52.17



http://www.phoronix.com

Co mierzyliśmy

Na systemie gospodarza, pod KVM oraz VMware mierzyliśmy czasy działania następujących programów:

- jednego procesu znajdującego liczby pierwsze
- dwóch procesów znajdujących liczby pierwsze
- jednego procesu liczącego silnię, korzystającego z architektury 64 bitowej
- dla wielu procesów czekających na siebie
- popularnego programu użytkowego: `make bzImage`

Co mierzyliśmy

Na systemie gospodarza, pod KVM oraz VMware mierzyliśmy czasy działania następujących programów:

- jednego procesu znajdującego liczby pierwsze
- dwóch procesów znajdujących liczby pierwsze
- jednego procesu liczącego silnię, korzystającego z architektury 64 bitowej
- dla wielu procesów czekających na siebie
- popularnego programu użytkowego: `make bzImage`

Co mierzyliśmy

Na systemie gospodarza, pod KVM oraz VMware mierzyliśmy czasy działania następujących programów:

- jednego procesu znajdującego liczby pierwsze
- dwóch procesów znajdujących liczby pierwsze
- jednego procesu liczącego silnię, korzystającego z architektury 64 bitowej
- dla wielu procesów czekających na siebie
- popularnego programu użytkowego: `make bzImage`

Co mierzyliśmy

Na systemie gospodarza, pod KVM oraz VMware mierzyliśmy czasy działania następujących programów:

- jednego procesu znajdującego liczby pierwsze
- dwóch procesów znajdujących liczby pierwsze
- jednego procesu liczącego silnię, korzystającego z architektury 64 bitowej
- dla wielu procesów czekających na siebie
- popularnego programu użytkowego: `make bzImage`

Co mierzyliśmy

Na systemie gospodarza, pod KVM oraz VMware mierzyliśmy czasy działania następujących programów:

- jednego procesu znajdującego liczby pierwsze
- dwóch procesów znajdujących liczby pierwsze
- jednego procesu liczącego silnię, korzystającego z architektury 64 bitowej
- dla wielu procesów czekających na siebie
- popularnego programu użytkowego: `make bzImage`

Metodologia

- test były powtarzane 20 razy
- obliczaliśmy czas minimalny, maksymalny, średnią, wariancję
- w systemie w tym czasie nie działały inne programy użytkownika
- każda z wirtualnych maszyn miała do dyspozycji 10 GB przestrzeni dyskowej oraz 1 GB pamięci operacyjnej, zaś gospodarz pracował na procesorze 2-rdzeniowym AMD64
- wyniki zależne od liczby procesów obrazujemy na wykresach o skali logarytmicznej

Metodologia

- test były powtarzane 20 razy
- obliczaliśmy czas minimalny, maksymalny, średnią, wariancję
- w systemie w tym czasie nie działały inne programy użytkownika
- każda z wirtualnych maszyn miała do dyspozycji 10 GB przestrzeni dyskowej oraz 1 GB pamięci operacyjnej, zaś gospodarz pracował na procesorze 2-rdzeniowym AMD64
- wyniki zależne od liczby procesów obrazujemy na wykresach o skali logarytmicznej

Metodologia

- test były powtarzane 20 razy
- obliczaliśmy czas minimalny, maksymalny, średnią, wariancję
- w systemie w tym czasie nie działały inne programy użytkownika
- każda z wirtualnych maszyn miała do dyspozycji 10 GB przestrzeni dyskowej oraz 1 GB pamięci operacyjnej, zaś gospodarz pracował na procesorze 2-rdzeniowym AMD64
- wyniki zależne od liczby procesów obrazujemy na wykresach o skali logarytmicznej

Metodologia

- test były powtarzane 20 razy
- obliczaliśmy czas minimalny, maksymalny, średnią, wariancję
- w systemie w tym czasie nie działały inne programy użytkownika
- każda z wirtualnych maszyn miała do dyspozycji 10 GB przestrzeni dyskowej oraz 1 GB pamięci operacyjnej, zaś gospodarz pracował na procesorze 2-rdzeniowym AMD64
- wyniki zależne od liczby procesów obrazujemy na wykresach o skali logarytmicznej

Metodologia

- test były powtarzane 20 razy
- obliczaliśmy czas minimalny, maksymalny, średnią, wariancję
- w systemie w tym czasie nie działały inne programy użytkownika
- każda z wirtualnych maszyn miała do dyspozycji 10 GB przestrzeni dyskowej oraz 1 GB pamięci operacyjnej, zaś gospodarz pracował na procesorze 2-rdzeniowym AMD64
- wyniki zależne od liczby procesów obrazujemy na wykresach o skali logarytmicznej

VMware i zegarek

(„Niektórzy mówią, że nas oko łudzi
I że nic nie ma, tylko się wydaje”)

- VMware nie umie mierzyć czasu
- zatem czas mierzyliśmy stoperem ;)
- niestety, nie ma wyników dla testów o krótkich czasach, a czasy maksymalne i minimalne nie są wiarygodne

VMware i zegarek

(„Niektórzy mówią, że nas oko łudzi
I że nic nie ma, tylko się wydaje”)

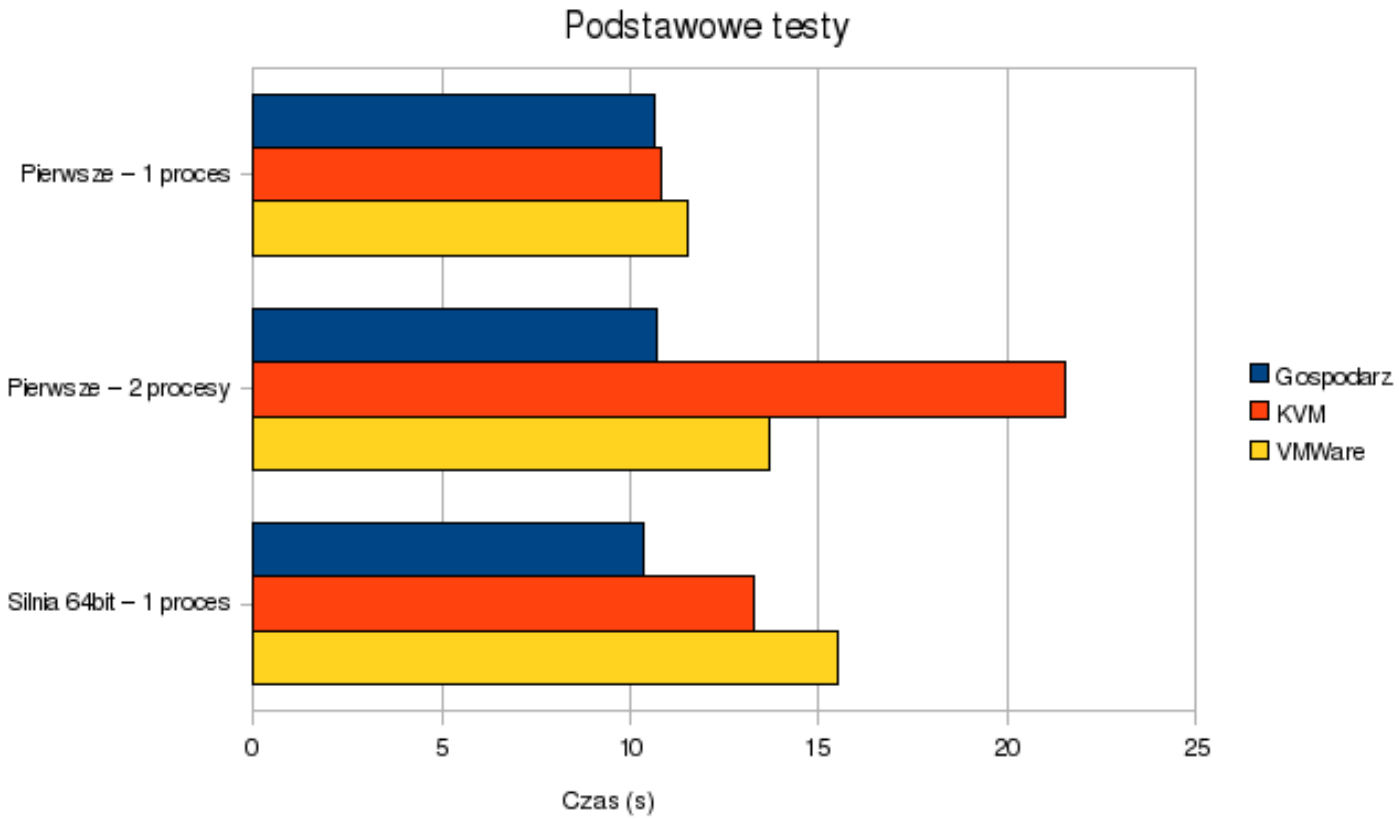
- VMware nie umie mierzyć czasu
- zatem czas mierzyliśmy stoperem ;)
- niestety, nie ma wyników dla testów o krótkich czasach, a czasy maksymalne i minimalne nie są wiarygodne

VMware i zegarek

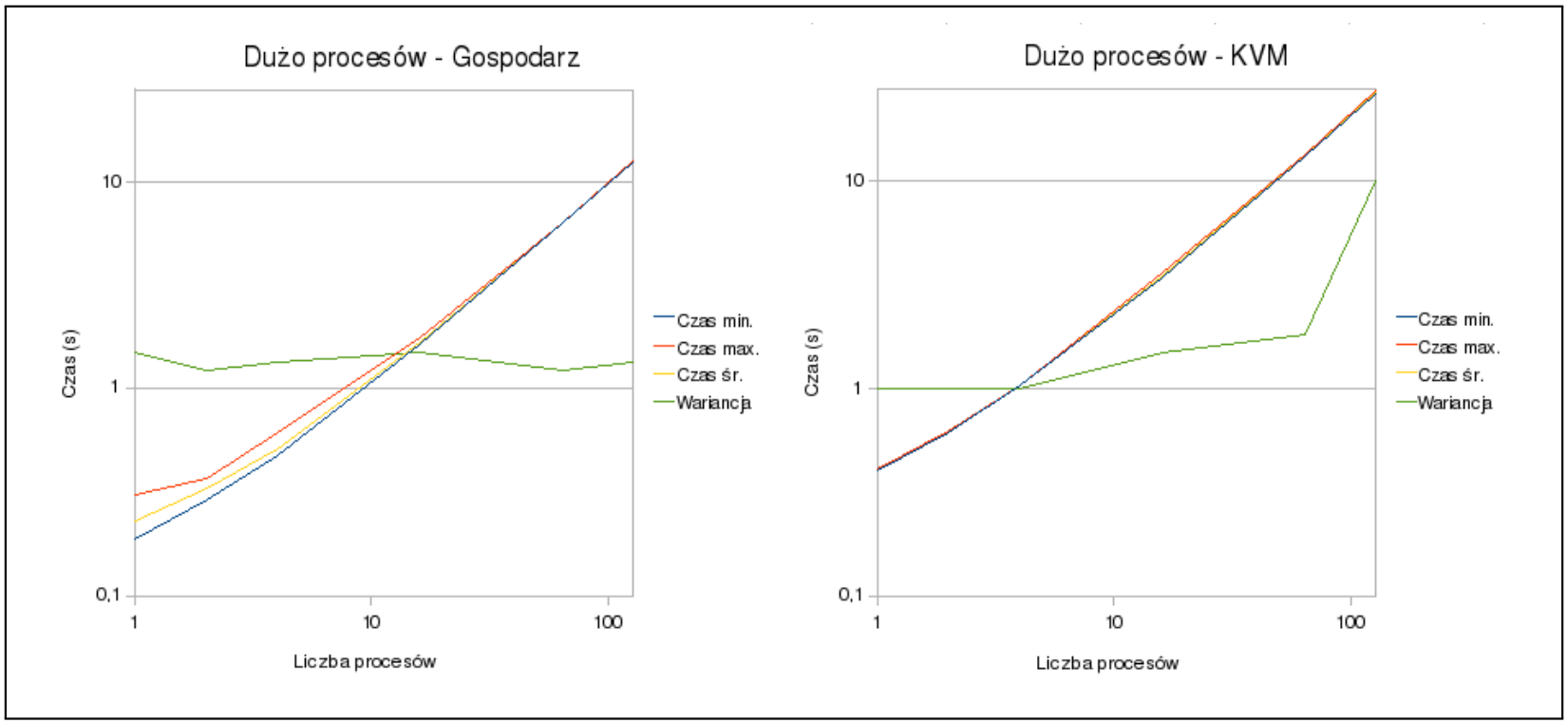
(„Niektórzy mówią, że nas oko łudzi
I że nic nie ma, tylko się wydaje”)

- VMware nie umie mierzyć czasu
- zatem czas mierzyliśmy stoperem ;)
- niestety, nie ma wyników dla testów o krótkich czasach, a czasy maksymalne i minimalne nie są wiarygodne

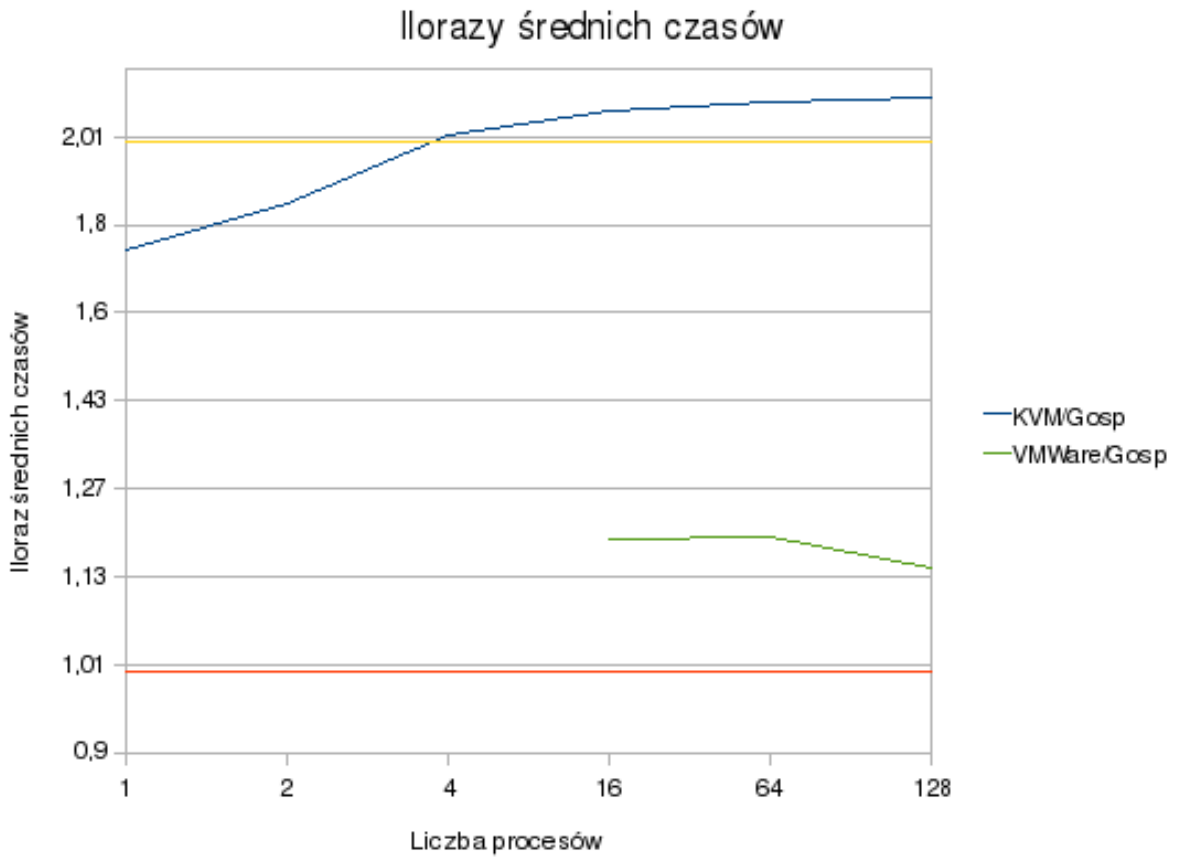
Liczby pierwsze i silnia



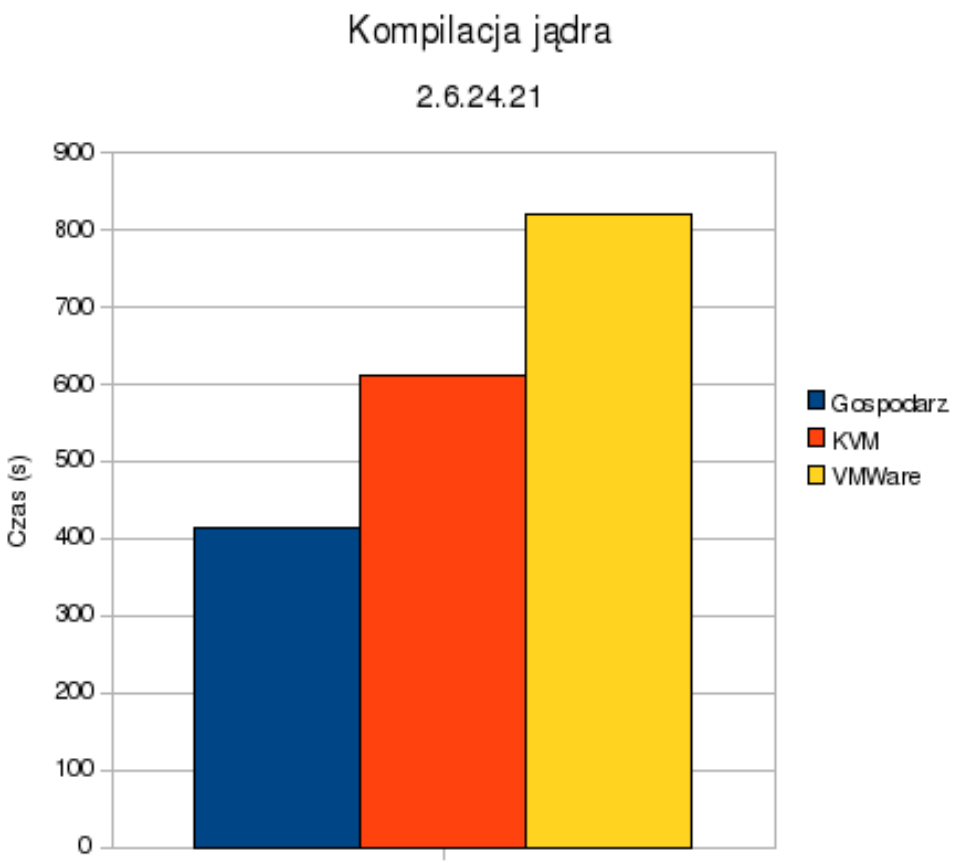
Dużo procesów – KVM



Dużo procesów – ilorazy



Deser


Kompilacja jądra
2.6.24.21
Czas (s)
900
800
700
600
500
400
300
200
100
0
Gospodarz
KVM
VMWare

Gdzie szukać wirtualizatora?

Kilka przydatnych linków:

- Virtualbox – `http://www.virtualbox.org/`
- Virtual PC – `http://www.microsoft.com/windows/products/winfamily/virtualpc/default.mspx`
- Qemu – `http://bellard.org/qemu/`
- KVM – `http://kvm.qumranet.com/kvmwiki`
- Lguest – `http://lguest.ozlabs.org/`
- Xen – `http://www.xen.org/`
- VMware – `http://www.vmware.com/`