Andrzej Stasiak, Zbigniew Skowroński
UNIWERSYTET ZIELONOGÓRSKI, INSTYTUT INFORMATYKI I ELEKTRONIKI

Wirtualizacja - kierunek rozwoju platform n-procesorowych

dr inż. Andrzej STASIAK

Absolwent Uniwersytetu Zielonogórskiego. Adiunkt w Instytucie Informatyki i Elektroniki Uniwersytetu Zielonogórskiego. Zakres tematyczny prowadzonych badań obejmuje zagadnienia zintegrowanego projektowania sprzętu i oprogramowania mikrosystemów cyfrowych. Zainteresowania skupiają się w szczególności na projektowaniu i implementacji systemów osadzonych w układach reprogramowalnych klasy SOPC, z wykorzystaniem języków opisu sprzętu (VHDL, Verilog HDL).

e-mail: A.Stasiak@iie.uz.zgora.pl

dr inż. Zbigniew SKOWROŃSKI

Adiunkt w Instytucie Informatyki i Elektroniki Uniwersytetu Zielonogórskiego. Zainteresowania badawcze obejmują zagadnienia zintegrowanego projektowania systemów sprzętowo-programowych, ze szczególnym uwzględnieniem języków opisu sprzętu (VHDL, Verilog HDL) oraz systemów osadzonych, zawierających reprogramowalne układy logiczne.

e-mail: Z.Skowronski@iie.uz.zgora.pl

Streszczenie

Artykuł stanowi wprowadzenie do zagadnień wirtualizacji systemów operacyjnych osadzonych na platformie x86. Przedstawia w zarysie rozwiązania sprzętowe zaimplementowane w dzisiejszych procesorach ogólnego przeznaczenia, mające na celu wsparcie rozwiązań programowych. Kierunek rozwoju platform wieloprocesorowych, w tym procesorów wielordzeniowych, wyznacza rynek, nie tylko rozumiany przez pryzmat dedykowanych centrów obliczeniowych, lecz również coraz częściej jako domowe czy biurowe centra danych cyfrowych.

Abstract

This paper presents a concept and existing approaches to the virtualization technology related to today's' x86 architectures. It describes hardware extensions that have been developed to support software to run and manage virtualization machine monitor as well as to accelerate most critical paging and transfer operations. Market drives today's trends/direction pushing processor vendors and motherboard providers to develop multi- and many core processors/systems that may be use as basis in data centre labs as well as home/office digital centres.

Słowa kluczowe: wirtualizacja, system operacyjny, maszyna wirtualna, procesor.

Keywords: virtualization, operating system, virtual machine, processor.

1. Wprowadzenie

Prawo Gordon E. Moore-a [3], sformalizowane w roku 1965, znajduje potwierdzenie w ponad 30-letniej historii prac prowadzonych nad architekturą x86. Co więcej, po raz pierwszy w historii (w roku 2008) przełamana została „magiczna" bariera 24 miesięcy podwojenia liczby tranzystorów w zintegrowanym układzie cyfrowym. W przeciągu kilkunastu tygodni od ogłoszenia produkcji układów cyfrowych w technologii 45nm, prekursor i lider w produkcji procesorów ogólnego przeznaczenia - Intel Corporation [5] - rozpoczął produkcję pamięci RAM w technologii 32nm [4]. Czynnikiem wymiernym, motywującym do prowadzenia tak intensywnych prac nad coraz to „mniejszymi" układami cyfrowymi jest ograniczenie poboru mocy przy jednoczesnym zachowaniu wydajności obliczeniowej lub zwiększenie częstotliwości taktowania procesora. Na rynek dostarczane są procesory o czterech rdzeniach CPU. Firma Intel pracuje nad architekturą n-rdzeniową procesorów. Do współpracy dołączyła miedzy innymi firma Microsoft oraz Uniwersytet w Berkley. Platforma Tera-Scale [1] już dziś udostępnia zasób składający się z 80 rdzeni CPU. Okazuje się, że technologia produkcji procesorów wielordzeniowych została opracowana i zweryfikowana. Można postawić pytanie: *Jak można wykorzystać dostarczaną moc obliczeniową, skoro oprogramowanie (w tym kompilatory dla x86) nie wspiera współbieżności?*

Jednym z intensywnie rozwijających się rynków na świecie, wymagający dużej wydajności obliczeniowej i niskiego poboru mocy (przy zachowaniu niskich kosztów oraz standaryzacji dostarczanych rozwiązań) są centra przetwarzania danych oraz centra świadczące usługi wejścia/wyjścia (serwisy: http, pop3, smtp, VoIP, VoD i inne). Dodatkowo zaobserwowano, że tylko 10% całkowitej mocy obliczeniowej serwerów jest zabsorbowane przez 90% czasu ich pracy. W dalszym ciągu wymagana jest moc obliczeniowa systemu komputerowego gotowa sprostać zapotrzebowaniom chwilowym w różnych segmentach rynku. Dla istniejących i przyszłych modeli centrów przetwarzania oraz usług kluczowymi aspektami w budowie systemu komputerowego są między innymi: wydajność, skalowalność, koszty, bezpieczeństwo, jakość i dostępność.

Jedną z technologii intensywnie dziś rozwijanych przez głównych dostawców procesorów klasy x86 (w tym AMD i Intel), a która posiada wystarczający potencjał by sprostać dzisiejszym i przyszłym wymaganiom architektur systemów komputerowych jest wirtualizacja.

2. Koncepcja wirtualizacji, rozwiązania i charakterystyka

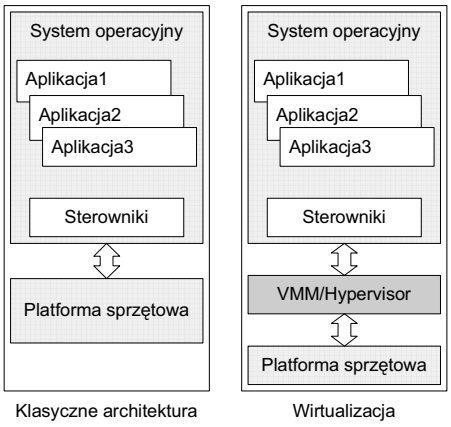
Koncepcja wirtualizacji jest znana w informatyce w wielu formach, począwszy od lat 60-tych. Prekursorem była firma IBM, która wprowadziła architekturę mainframe (komputery System/360). Wirtualizacja charakteryzuje się abstrakcyjnym postrzeganiem zasobów sprzętowych, w taki sposób, że zasób może być aktywowany do pracy na różne sposoby niezależnie od jego fizycznej formy lub lokalizacji.

Wirtualizacja to trwałe rozdzielenie dwu warstw każdego systemu komputerowego: warstwy sprzętu i warstwy systemu operacyjnego - rezydujących i pracujących w obrębie maszyny cyfrowej [7, 8] oraz dodanie dodatkowej warstwy programowej, implementującej funkcjonalność sprzętu - komputer w komputerze (rysunek 2.1). Oznacza to, że programowo emulowane są wszystkie podstawowe typy urządzeń, takie jak: karta dźwiękowa, CPU, pamięć, dyski twarde, karty sieciowe i inne. Instancja systemu operacyjnego pracującego w emulowanym - wirtualnym środowisku sprzętu nazywana jest maszyną wirtualną. Technologia wirtualizacji pozwala na jednoczesną pracę na tej samej platformie sprzętowej wielu heterogenicznych systemów operacyjnych, pozostających względem siebie w izolacji programowej.

Poprzez emulację kompletnego systemu sprzętowego, począwszy od procesora do np. karty sieciowej, każda maszyna wirtualna może współdzielić dostępny zbiór zasobów sprzętowych bez obawy na jednoczesny dostęp do tych zasobów przez inne maszyny wirtualne. System operacyjny pracujący w trybie maszyny wirtualnej „widzi" jednolity, znormalizowany zbiór zasobów sprzętowych niezależnie od aktualnie dostępnych komponentów sprzętowych. Technologia, taka jak Intel ®Virtualization Technology [6] (Intel® VT), która zostanie omówiona w następnym rozdziale, znacząco udoskonala i wspiera wirtualizację.

Za przykład wirtualizacji, którą wykorzystuje dziś większość komputerów można uznać pamięć wirtualną. Dzięki tej technice

program może zaadresować znacznie większy obszar pamięci niż w rzeczywistości jest dostępny. Aby to osiągnąć, zazwyczaj aktualnie niewykorzystywany obszar pamięci fizycznej składowany jest na dysku twardym, by w późniejszym czasie ponownie zostać skopiowanym do pamięci operacyjnej. W rezultacie, każdy program ma dostęp do 4GB pamięci operacyjnej (zakładając brak rozszerzenia PEA), pomimo że na platformie sprzętowej dostępnych jest np. tylko 256MB pamięci RAM. W przypadku dostępu do pamięci masowej (dyski twarde) wirtualizacja może polegać na przekazaniu do wirtualizowanego systemu operacyjnego jednego dysku twardego ATA, np. o pojemności 20GB, gdzie w rzeczywistości konfiguracja sprzętu zrealizowana jest poprzez interfejs Fiber Channel z czterema dyskami SCSI, każdy o pojemności 5GB.



Rys.2.1. Klasyczna architektura komputera a wirtualizacja

W technologii wirtualizacji oprócz maszyny wirtualnej występuje także monitor maszyn wirtualnych VMM (ang. Virtual Machine Monitor). VMM jest to oprogramowanie, które implementuje funkcje wirtualizacji zintegrowane z dedykowanym nadrzędnym systemem operacyjnym (ang. Host-OS). Podstawowym zadaniem VMM jest wirtualizacja wybranych zasobów sprzętowych (CPU, pamięć, dyski twarde i inne) oraz emulacja sterowników do pozostałych urządzeń systemu, każdemu „gościowi OS" (ang. Guest-OS). Oznacza to, że VMM pełni rolę zarządcy platformy sprzętowej dla wirtualizowanych systemów operacyjnych guest-OS, udostępniając lub ukrywając wybrane komponenty systemu.

Znane są trzy różne architektury programowe wirtualizacji VMM:

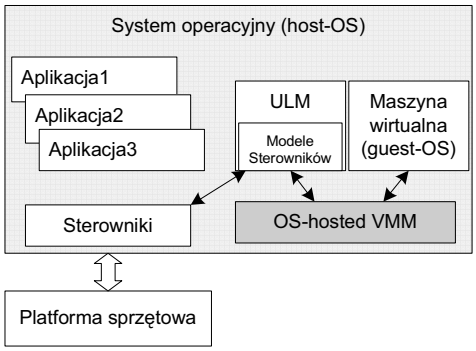
- OS-hosted VMM,
- samodzielny VMM,
- hybrydowy VMM.

Każdy z wymienionych stylów realizacji VMM posiada pewne zalety i wady. Decyzja o wyborze właściwego rozwiązania podyktowana jest najczęściej wymaganiami stawianymi modelowi użycia systemu komputerowego w wybranym segmencie rynku, gdzie dane rozwiązanie będzie miało swoje zastosowanie.

OS-hosted VMM

Pierwszym sposobem jest zbudowanie architektury programowej VMM w istniejącej infrastrukturze systemu operacyjnego. W rozwiązaniu tym VMM uruchamiany jest przez nadrzędny system operacyjny (rysunek 2.2) w obszarze ring0 - trybie pracy procesora o najwyższym poziomie przywilejów. Do końca swojej pracy pozostaje podrzędnym zadaniem w stosunku do głównego OS, który dostarcza do VMM kompletny interfejs platformy sprzętowej. Jądro VMM przełącza kontekst pomiędzy stanami guest-OS a host-OS okresowo według „planisty" host-OS lub żądań zewnętrznych (np. przerwań sprzętowych).

Mimo, że guest-OS jest upoważniony do bezpośredniego wykonania kodu na fizycznym procesorze oraz dostępu do pamięci operacyjnej, to każdy dostęp/akcja do urządzeń WE/WY jest przechwytywany przez jądro VMM i przekierowywany do poziomu drugiego VMM - do poziomu użytkownika ULM (ang. User-Level Monitor). Proces ULM uruchamiany jest jako zwykły proces nadrzędnego systemu operacyjnego host-OS, który zawiera i udostępnia modele wirtualnych urządzeń WE/WY, obsługujące żądania guest-OS. Model wirtualnego urządzenia ULM komunikuje się w dalszej kolejności z systemem plików i sterowników host-OS w celu realizacji żądań guest-OS.

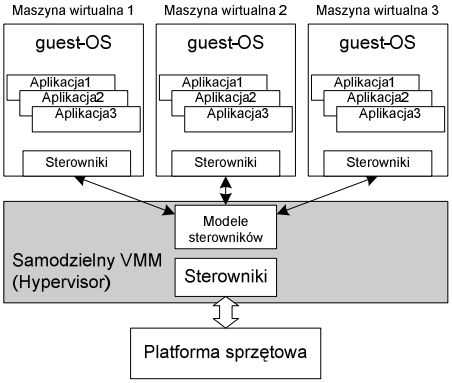


Rys.2.2. Rozwiązanie osadzonego VMM w nadrzędnym systemie operacyjnym

Zaletami architektury OS-hosted VMM są: a) VMM używa sterowników dostarczanych z host-OS; b) łatwa przenaszalność VMM w zakresie innych systemów host-OS. Natomiast główną wadą jest pełna zależność VMM, pod względem bezpieczeństwa, stabilności i niezawodności od nadrzędnego systemu operacyjnego host-OS. Wybierając tę architekturę, „planista" host-OS przydziela jednostki czasu dostępu do procesora zarządcy VMM na równi z innymi aplikacjami i usługami systemu host-OS.

Samodzielny VMM

Alternatywnym podejściem jest realizacja VMM jako samodzielnego monitora VMM hypervisor, który nie jest zależny od nadrzędnego systemu operacyjnego (rysunek 2.3).



Rys.2.3. VMM jako samodzielny zarządca

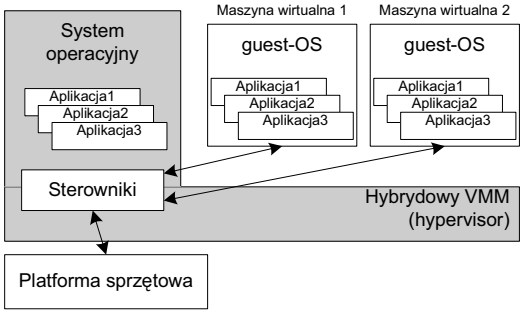
VMM hypervisor dostarcza niezależny zestaw sterowników, modeli urządzeń oraz system „planisty". Tego typu architektura zapewnia pełną kontrolę przydziału/dostępu do platformy sprzętowej wszystkim guest-OS, skracają jednocześnie ścieżkę przepływu żądania guest-OS → urządzenie WE/WY. Przez minimalizację kodu oraz samodzielność pracy VMM możliwe jest dostarczenie bezpiecznego pod względem stabilności i gwarancji czasu rzeczywistego, niezawodnego rozwiązania. Zalety przedstawionej architektury VMM kontrastują z jej główną wadą: zależnością od platformy sprzętowej zestawu sterowników dla każdej nowej platformy - koniecznością dostarczenia tego zestawu razem z VMM.

Hybrydowy VMM

W podejściu tym podjęto wysiłek połączenia dwóch poprzednich rozwiązań. Cel osiągnięto dostarczając hybrydowy VMM, łączący cechy bezpieczeństwa, stabilności, niezawodności, niezależności a przede wszystkim zastosowania sterowników urządzeń WE/WY, dostarczanych z bazowym systemem operacyjnym. W omawianym rozwiązaniu VMM (rysunek 2.4) - małe jądro „hypervisor" - kontroluje wszystkie aspekty CPU oraz dostęp i zasoby pamięci operacyjnej. Natomiast urządzenia WE/WY kontrolowane są bezpośrednio przez sterowniki „usługującego" systemu operacyjnego, pracującego w trybie ograniczonym (w pełni sterowanym przez VMM hypervisor).

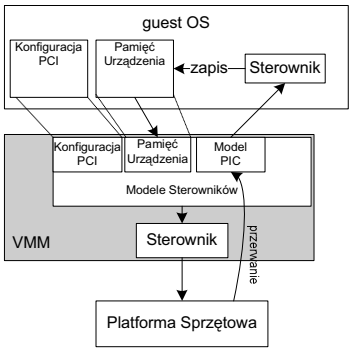
Przedstawiona architektura zapewnia najlepsze charakterystyki pracy oraz zarządzania usługowego i hybrydowego VMM. Próba zastosowania tej architektury VMM z użyciem procesorów x86, bez dedykowanego wsparcia w sprzęcie (związanego z implementacją nowych instrukcji w procesorach i układach chipset) zazwy-

czaj kończy się niepowodzeniem. Niezbędne jest rozszerzenie konstrukcji x86 o nowe instrukcje, specjalny układ ADMA oraz dedykowany obszar (ang. Ring) uprawnień procesora, czyli dołączenie do procesorów ogólnego przeznaczenia technologii Intel VT-x, VT-d, VT-d2; czy AMD-V [2].



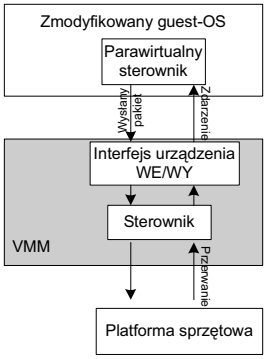
Rys.2.4. Hybrydowy VMM

Kluczowym czynnikiem wpływającym na wydajność pracy VMM jest sposób wirtualizacji sterowników urządzeń platformy sprzętowej. Wyróżnia się dwa podejścia. Pierwszym jest emulacja (rysunek 2.5) związana z pełnym odwzorowaniem funkcjonalności urządzenia w programie, tzn. emulowane urządzenie „udaje" rzeczywistą pracę komponentu sprzętowego. System operacyjny typu guest-OS „nie wie", że komunikuje się z programowym odpowiednikiem urządzenia. VMM jest odpowiedzialny za dostarczenie modeli sterowników w taki sposób, by były one „widoczne" dla maszyny wirtualnej (guest-OS) oraz za obsługę przychodzących rozkazów, jak i wystawianie/przekierowywanie przerwań do systemu operacyjnego.



Rys.2.5. Emulacja sterowników urządzeń platformy sprzętowej

Inną techniką wirtualizacji urządzeń WE/WY - noszącą nazwą parawirtualizacji (rysunek 2.6) - jest modyfikacja sterowników w obszarze guest-OS. W rozwiązaniu tym maszyna wirtualna „wie" o tym, że pracuje w systemie wirtualnym a sterowniki guest-OS dla platformy sprzętowej komunikują się bezpośrednio z urządzeniami, ale w dalszym ciągu pod kontrolą VMM.



Rys.2.6. Parawirtualizacja sterowników platformy sprzętowej

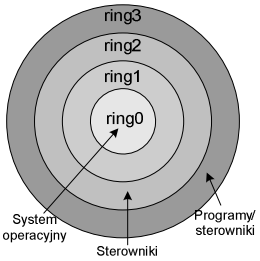
Zaletą parawirtualizacji jest znaczące zwiększenie wydajności obsługi urządzeń w stosunku do emulacji poprzez między innymi zmniejszenie liczby interakcji pomiędzy guest-OS a VMM oraz bezpośrednim dostępem do urządzeń. Ponadto, parawirtualizacja używa zdarzeń lub bezpośrednich wywołań zamiast mechanizmu emulacji przerwań. Eliminuje to nadmiarowe instrukcje do układu PIC (ang. Programmable Interrupt Controller) oraz dodatkowe opóźnienia wprowadzane w związku z tym przez system operacyjny.

3. Rozwiązania Intel VT-x, VT-d, VT-d2

Procesory z rodziny IA-32 nie zostały opracowane pod kątem realizacji funkcji/instrukcji wirtualizacji. Zostały zaprojektowane przede wszystkim do uruchomienia pojedynczej instancji systemu operacyjnego. W systemach używających architektury IA-32 bez wsparcia wirtualizacji możliwa jest tylko wirtualizacja programowa, czyli podejście OS-hosted VMM. Dostępne są natomiast rozszerzenia instrukcji procesorów wspierające wirtualizację, znane jako: VT-x, VT-d i Vt-d2.

Rozwiązania VT-x

Procesory Intel dostarczają mechanizmu zabezpieczenia wykonania kodu programu poprzez zdefiniowanie obszarów - poziomów przywilejów (rysunek 3.1) dla wykonywanego kodu: ring0, ring1, ring2, ring3. Poziomy przywilejów determinują, które akcje/instrukcje może wykonać dany proces - program. Na przykład, odwzorowywanie (mapowanie) pamięci może być wykonane tylko na poziomie 0. Kontrastując, kod użytkownika wykonywany jest na poziomie 3, będącym poziomem o najmniejszych prawach. Oprogramowanie pracujące na niżej numerowanym poziomie przywilejów może kontrolować programy uruchomione na poziomach o wyższych numerach.



Rys.3.1. Zakres poziomów przywilejów w procesorach x86

Wiele systemów operacyjnych musi zostać uruchomionych na zerowym 0 poziomie przywilejów, co wynika z żądania nieograniczonego dostępu do wszystkich funkcji/instrukcji procesora. Podobnie VMM, musi zostać uruchomiony w obszarze 0, by w pełni kontrolować dostęp maszyn wirtualnych do zasobów sprzętowych.

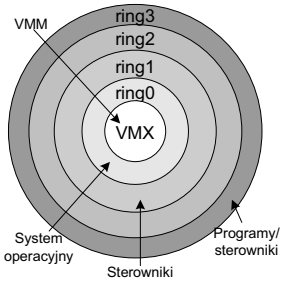
W konsekwencji powstaje konflikt zawłaszczenia obszaru 0 przywilejów procesora, który próbowano rozwiązać programowo w implementacji VMM. Otóż, możliwa jest relokacja maszyny wirtualnej (guest-OS) do innego obszaru - technika znana jako pozbawienie przywilejów (ang. Ring Deprivileging), polegająca na tym, że system operacyjny maszyny wirtualnej uruchomiony zostaje, np. w obszarze 1 lub 3. Niestety zmiana poziomów przywilejów pociąga za sobą dodatkowe wyzwania dla VMM. Musi on stale monitorować aktywność guest-OS w celu przechwycenia prób dostępu do sprzętu oraz wielu wywołań systemowych. VMM musi emulować wszystkie wywołania systemowe guest-OS (przechwycenie, wykonanie, zwrócenie wyniku), co w wielu przypadkach nie jest możliwe.

Jeśli oprogramowanie (system operacyjny) zostało przygotowane do pracy na innym poziomie przywilejów niż aktualnie pracuje, to wówczas narasta kolejny problem związany z powiązaniem obszaru przywilejów (ang. Ring Aliasing). System operacyjny wywołuje szereg instrukcji autoryzowanych do wykonania poza obszarem 0 w celu zweryfikowania swoich przywilejów. Jeśli guest-OS wykryje, że został uruchomiony poza obszarem 0, to wówczas jego zachowanie jest trudne do przewidzenia (niespecyfikowane w dostępnej dokumentacji).

VMM wykonuje instrukcje dotyczące przełączenia kontekstu procesu maszyny wirtualnej w celu przydzielenia platformy sprzętowej dla kolejnego guest-OS. Jednak system operacyjny nie jest napisany by wspierać zmiany kontekstu. Potrafi zapisać dane w ukrytych lokacjach. Przełączenie kontekstu może zniszczyć te dane, w związku z czym wirtualizowany system operacyjny przestanie funkcjonować.

Powyżej przytoczono dwa scenariusze, w których VMM nie potrafi programowo rozwiązać przedstawionych problemów. Niezbędne jest zatem wsparcie ze strony procesora i układu chipset.

Intel ®Virtualization Technology, poprzez szereg sprzętowych innowacji dostarczanych jako rozszerzenia standartowych instrukcji x86 procesora i układu chipset, zapewnia rozwiązanie wielu problemów dotyczących technologii wirtualizacji dla rodzin IA32/64 (wspomnianych również w tym artykule). Pozwala między innymi na uruchomienie VMM poza strefą standardowych poziomów przywilejów, pozwalając systemowi operacyjnemu i programom na pracę (bez zmian) w domyślnych obszarach przywilejów.



Rys.3.2. Nowe rozwiązania - dodatkowy obszar przywilejów dla celów wirtualizacji

Technologie VT-x i VT-i zapewniają ten sam zakres rozszerzeń wirtualizacji odpowiednio dla rodzin IA-32 i Italium2. VT-x dodaje do znanej architektury IA-32 nowy tryb pracy procesora: VMX - dodatkowy obszar przywilejów (rysunek 3.2). Monitor wirtualnej maszyny (VMM) uruchamiany jest w głównym trybie operacyjnym „root VMX" o pełnych przywilejach. Wirtualne maszyny (guest-OS) pracują w drugorzędnym trybie operacyjnym „non-root VMX". Należy zwrócić uwagę, że guest-OS uruchamiany jest w obszarze 0, który stanowi jego naturalny poziome przywilejów. Wprowadzono dwie nowe instrukcje związane z VMX, które przekazują kontrolę pomiędzy VMM i guest-OS:

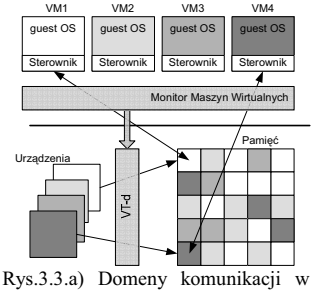
- VM ENTRY - tranzycja VMM-to-guestOS, przejście w tryb "non-root VMX";
- VM EXIT - tranzycja guestOS-to-VMM, przejście w tryb „root VMX".

Komenda VM ENTRY pozwala na rozpoczęcie pracy guest-OS w roboczym trybie operacyjnym „non-root VMX". Następnie, wykonanie instrukcji VM EXIT przywraca kontrolę dla VMM, który kontynuuje realizacje swoich zadań w głównym trybie operacyjnym „root VMX".

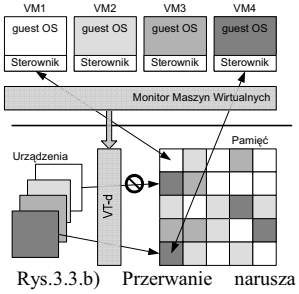
Technologie AMD-V dostarczają podobnych rozwiązań programowych problemów wirtualizacji. Widoczne są drobne różnice implementacyjne na poziomie konstrukcji procesora, jednak w ogólnym podejściu zakres i funkcjonalność są takie same.

Rozwiązania VT-d

VMM implementuje wirtualizację żądań IO zgłaszanych przez maszynę wirtualną poprzez techniki emulacji lub parawirtualizacji. Wymagania bezpieczeństwa i niezawodności, stawiane przytoczonym wyżej modelom wirtualizacji, dotyczą izolacji dostępu urządzeń IO w zakresie zasobów/VM (i odwrotnie), które zostały przydzielone przez VMM.



Rys.3.3.a) Domeny komunikacji w VT-d



Rys.3.3.b) Przerwanie narusza domenę komunikacji – VT-d wykrywa przewinienie

Wykorzystując technologię VT-x, która dostarcza wsparcia dla wirtualizacji procesora i pamięci operacyjnej, oprogramowanie VMM jest znacząco ograniczone w poprawnej interpretacji zgłaszanych przerwań sprzętowych IO oraz transferów DMA. Intel VT-d (ang. Virtualization Technology for Directed IO) rozszerza wsparcie dla programowego monitora maszyn wirtualnych o funkcję sprzętową realizującą prze-mapowanie (ang. remapping) transferów DMA i przerwań IO. Nowe rozszerzenie VT-d pozwala na definiowanie domen powiązanych przestrzeni adresowych pamięci i urządzeń IO, rysunek 3.3, zapewniając bezpieczeństwo komunikacji, niezawodność oraz znaczace zwiększenie wydajność transmisji IO--pamięć (poprzez sprzętową funkcję tłumaczenia adresów i przerwań).

4. Obszary zastosowań wirtualizacji

Redukcja sprzętu i oprogramowania, zwiększenie wydajności i skalowalności oraz redukcja czasu przestoju są czynnikami zdecydowanie wpływającymi na zarządzanie kosztami w dzisiejszych centrach komputerowych. Maszyny wirtualne (wirtualizacja) pozwalają osiągnąć wyżej wskazane cele. Można wymienić wiele korzyści płynących ze stosowania wirtualizacji:

a) Maszyny wirtualne pozwalają na efektywniejsze wykorzystanie zasobów poprzez konsolidację wielu środowisk operacyjnych w mniejszą liczbę serwerów wirtualnych.
b) Maszyny wirtualne upraszczają proces zarządzania systemami operacyjnymi.
c) Środowisko maszyny wirtualnej jest kompletnie izolowane od komputera oraz innych środowisk maszyn wirtualnych, przez co możliwe jest budowanie bardzo bezpiecznych środowisk programowych dostosowanych do potrzeb rynku.
d) Można dokonywać migracji przestarzałych systemów operacyjnych do maszyn wirtualnych pracujących na nowoczesnych platformach sprzętowych, które nie są wspierane przez dany system operacyjny.
e) Można uruchomić jednocześnie wiele, różnych systemów operacyjnych (od różnych dostawców) na jednym komputerze.
f) Ponieważ maszyny wirtualne zawierają się w obrębie plików i katalogów, to możliwa jest łatwa migracja maszyn pomiędzy platformami sprzętowymi, na których działa VMM.
g) VMM zarządzając zasobami sprzętowymi w wielordzeniowych architekturach komputerowych umożliwia przydzielenie każdej maszynie wirtualnej jednego lub więcej fizycznych rdzeni CPU.

Literatura

[1] http://techresearch.intel.com/articles/Tera-Scale/1421.htm#Vision
[2] http://www.amd.com/us-en/Processors/ProductInformation/0,,30_118_15331_15332,00.html
[3] http://www.intel.com/pressroom/kits/bios/moore.htm?iid=search
[4] http://www.intel.com/technology/architecture-silicon/32nm/index.htm?iid=tech_arch+body_32nm
[5] http://www.intel.com/technology/architecture-silicon/45nm-core2/index.htm?iid=tech_arch+body_45nm_hi-k
[6] http://www.intel.com/technology/virtualization/index.htm
[7] J. Sugarman, G.Venkitachalam, B. Lim: Virtualizing I/O Devices on VMware Workstation's Hosted Virtual Machine Monitor, in Proceedings of 2002 USENIX Annual Technical Conference, pp. 1-14, June 2001
[8] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, A. Warfield: Xen and the Art of Virtualization, in Proceedings of the 19th ACM Symposium on Operating Systems Principles, pp.164-177, October 2003

Title: Virtualization - the way to utilized many-cores' platforms.