Techniky virtualizace počítačů (2)

Luděk Matyska, ÚVT MU

V předchozím článku jsme si pověděli něco o konceptu virtualizace počítačů a důvodech, pro které se tento koncept dostal do popředí zájmu výzkumu i průmyslu. V tomto článku se soustředíme více na technickou stránku věci a způsoby, kterými je možné architekturu IA-32[1] virtualizovat. Cílem samozřejmě není vybavit čtenáře hlubokými technickými znalostmi, které by mu umožnily napsat vlastní virtuální monitor, jako spíše podat přehled o používaných technikách a ukázat jejich sílu i slabé stránky.

Běžné počítače, odpovídající v podstatě modelu von Neumannovy architektury, se skládají z několika základních komponent. Na základní úrovni se jedná o procesor, paměť a periferie. Ty zpravidla dále členíme a explicitně pracujeme s diskem, klávesnicí a myší, grafickým subsystémem, USB a síťovým rozhraním atd. Pak si namísto fyzických komponent představíme jejich abstraktní variantu v podobě virtuálních komponent, můžeme jejich složením vytvořit požadovaný virtuální počítač. Na něm pak můžeme spustit operační systém a vytvoříme tak virtualizované prostředí.

1 Plná virtualizace

Pokud postupujeme tímto způsobem, tj. virtualizujeme důsledně všechny součásti počítače, hovoříme o tzv. *plné virtualizaci* (full virtualization). V takovémto případě nabízíme prostředí, v němž běžící operační systém nemůže žádným způsobem poznat, že nemá přístup k fyzickému technickému vybavení (hardware). Operační systém ani aplikační programy nepotřebují žádné modifikace. Jedná se v podstatě o ideální stav, kdy dochází k plnému oddělení fyzické vrstvy, veškeré programy běží pouze na virtuálním hardware a přístup k fyzickému vybavení je vždy zprostředkován. To má samozřejmě řadu výhod – můžeme virtuální prostředí navrhnout tak, aby nám vyhovovalo (velikost paměti, typ procesoru, typ a kapacitu disku apod.). Programy jsou rovněž nezávislé na konkrétním technickém vybavení, jeho změna nemá na virtuální prostředí vliv (samozřejmě kromě výkonnostních charakteristik, tj. náš virtuální počítač může běžet rychleji nebo pomaleji, ale v každém případě poběží).

U plné virtualizace nemusí existovat žádná jednoduchá vazba mezi virtuálním prostředím a konkrétním hardware, na němž je virtuální počítač provozován. To umožňuje plnou přenositelnost – operační systém a aplikace běžící na procesoru Intel s architekturou IA-32 můžeme spouštět třeba na počítačích, vybavených procesory PowerPC. A následně je můžeme přenést na počítače vybavené jiným procesorem, aniž bychom provedli jedinou úpravu na úrovni virtuálního počítače. Podobně můžeme vytvořit virtuální počítač vybavený procesorem, který je teprve ve vývoji – návrh a ladění operačního systému a aplikací tak může probíhat paralelně s vývojem vlastního hardware.

Jako poněkud úsměvný příklad je možné uvést virtualizaci oblíbeného počítače ZX Spectrum v podobě Java Appletu (`http://www.spectrum.lovely.net/`), v němž je možné spouštět hry původně vyvinuté pro tento počítač. Mezi profesionální systémy, které nabízí plnou virtualizaci počítačů s procesorem Intel patří Microsoft Virtual Server a VMWare ESX Server™.

2 Paravirtualizace

Samozřejmě plná virtualizace má svou cenu. Protože dochází k úplnému oddělení fyzické a programové vrstvy, je při plné virtualizaci prakticky nemožné dosáhnout plného výkonu i v tom případě, že virtuální počítač je víceméně přesným obrazem hardware, na kterém běží (především nabízí identický procesor a další periferie). Virtuální monitor totiž musí kompletně odstínit virtuální počítač od jakékoliv možné změny hardware. Toho dosáhne tak, že *emuluje* fyzické vybavení a většinu operací (včetně řady instrukcí procesoru, práce s pamětí, operace přístupu na disk a další) provádí ve vlastním software namísto aby je přímo vykonával hardware. Nemá-li dojít k výraznému zpomalení virtuálního počítače, je virtualizace omezena pouze na virtuální prostředí, které se maximálně podobá tomu fyzickému.

[1] IA-32 je zkratka pro Intel Architecture s 32-bitovým adresovým prostorem.

Pokud však předpokládáme, že se alespoň některé komponenty virtuálního a fyzického počítače shodují (např. virtuální počítač bude vždy nabízet stejný procesor, nanejvýš s poněkud nižším výkonem), pak můžeme odstoupit od principu plné virtualizace a pracovat s tzv. *paravirtualizací*. Ta se vyznačuje tím, že provádí jen částečnou abstrakci na úrovni virtuálního počítače, tj. nabízí virtuální prostředí, které je *podobné* tomu fyzickému, na kterém virtuální počítač provozujeme. Virtualizace v tomto případě není úplná, některé vlastnosti např. procesoru mohou být omezeny a operační systém může rozpoznat, že běží ve virtuálním prostředí. Na druhou stranu skutečnost, že virtuální a fyzický hardware se příliš neliší, umožňuje, aby virtuální počítač v maximální míře využíval vlastnosti základního fyzického prostředí (nemusíme emulovat všechny komponenty virtuálního počítače).

Paravirtualizace je široce využívána při tvorbě virtuálních prostředí nad procesory Intel (a AMD). VMWare workstation a Xen patří mezi neznámější systémy, které jsou postaveny na paravirtualizaci. Základní principy paravirtualizace si představíme na (zjednodušeném) modelu, který používá právě prostředí *Xen*.

Prvním problémem, který je třeba vyřešit, je virtualizace procesoru. Každý procesor pracuje alespoň ve dvou různých režimech – *privilegovaném*, který je přístupný pouze jádru operačního systému, a *uživatelském*, ve kterém běží všechny programy. Úkolem privilegovaného režimu je zajistit, že uživatelé mají kontrolovaný přístup k hardware a nemohou přímo provádět operace, které by mohly ohrozit jiné programy či integritu dat (např. přímý přístup na disk či složitější operace s virtuální pamětí). Pokud ale počítač virtualizujeme, potřebujeme ještě jednu úroveň, na které poběží virtuální monitor. V případě plné virtualizace to není problém, při tomto přístupu emulujeme celý procesor se všemi úrovněmi ochrany, v případě paravirtualizace je to však mnohem složitější.

Virtuální monitor musí běžet na nejvyšším stupni ochrany. Na stejné úrovni však nemůže automaticky běžet operační systém, protože by mohl ovlivnit stav virtuálního monitoru. Jednou z možností je pozměnit kód operačního systému tak, že nebude provádět žádnou operaci, pro jejíž provedení je třeba oprávnění té nejvyšší úrovně. Provedení instrukce se změní ve volání příslušné funkce virtuálního monitoru, který nejprve zkontroluje, zda je operace povolena a následně ji provede tak, aby změnila stav virtuálního, nikoliv fyzického počítače. Nemalý problém nám však budou v tomto přístupu dělat instrukce čtení paměti. Jádro operačního systému předpokládá, že má přímý přístup k libovolné části fyzické paměti, to však samozřejmě v případě virtuálního počítače není možné. Protože nelze předem poznat, zda konkrétní operace čtení z paměti bude přistupovat k privilegovaným údajům, museli bychom nahradit v operačním systému všechny instrukce čtení – tím se ale začneme velmi nepříjemně přibližovat k plné virtualizaci. Další problém spočívá v ochraně operačního systému před běžícími uživatelskými programy. Pokud bychom měli jen dvě úrovně ochrany (privilegované a neprivilegované), musel by operační systém virtuálního počítače pracovat neprivilegovaně, tím by však byl vystaven ohrožení ze strany aplikací.

Paravirtualizace je tak možná jen díky tomu, že konkrétní procesory podporují více úrovní ochrany. Procesory Intel mají definovány 4 úrovně ochrany, tzv. *okruhy* (rings). Na nejvyšším stupni ochrany (ring 0) běží operační systém, uživatelské programy běží s nejnižším stupněm ochrany (ring 3). Ostatní stupně se běžně nevyužívají. Pokud použijeme paravirtualizaci, pak virtuální monitor pracuje na nejvyšším stupni ochrany, tj. v okruhu 0. Operační systém virtuálního počítače se posune o jeden stupeň (do okruhu 1), aplikační programy běží stále s nejmenší ochranou. Operační systém má tak stále vyšší úroveň ochrany než aplikační programy, na druhé straně už nemůže provádět operace, které vyžadují plně privilegovaný přístup. Úrovně ochrany však můžeme využít i místo výše zvýšené modifikace privilegovaných instrukcí – necháme operační systém ve virtuálním počítači provádět všechny instrukce, pokud však bude chtít provést „zakázanou" operaci (tj. takovou, na kterou teď nemá dostatečná oprávnění), pak dojde k přerušení a řízení převezme virtuální

monitor. Ten operaci zkontroluje a provede ji tak, aby správně změnila stav virtuálního počítače. Není v principu třeba měnit operační systém, většina instrukcí běží přímo, pouze privilegované instrukce jsou výrazně pomalejší, protože je musí provést virtuální monitor. Operační systém však může zjistit, že běží ve virtuálním prostředí, protože může mít i na úrovni 1 možnost číst některé části paměti, které jsou ve virtuálním počítači jiné něž ve fyzickém. Pro paravirtualizaci je proto třeba modifikovat některé součásti operačního systému, změny jsou však malé a dobře lokalizovatelné (zvlášť dobře je pak možné provést tyto změny u operačních systémů, k nimž jsou k dispozici zdrojové kódy; i proto začala být tak oblíbená (para)virtualizace v prostředí Linuxu).

Přístup k hardware je v prostředí Xen zajišťován vrstvou virtuálního monitoru (Virtual Machine Monitor, VMM). Nad touto vrstvou jsou pak vytvářeny virtuální počítače (Virtual Machines, VM). Jeden z těchto virtuálních počítačů má speciální postavení – v terminologii Xenu se nazývá Doménou 0 (Dom 0). Operační systém, který běží v tomto virtuálním počítači, má přímý přístup k rozhraní virtuálního monitoru, může tedy definovaným způsobem měnit jeho stav a může vytvářet a rušit ostatní virtuální počítače běžící nad VMM. Další zajímavou vlastností Xenu (opět související s paravirtualizací) je to, že může konkrétnímu virtuálnímu počítači přímo zpřístupnit konkrétní rozhraní. Představme si, že v jednom z virtuálních počítačů běží uživatelský program, který intenzivně komunikuje s jiným počítačem prostřednictvím počítačové sítě. Pokud používá virtuální síťovou kartu, pak její propustnost je omezena a velmi zatěžuje procesor. Pokud ale příslušnému virtuálnímu počítači po dobu běhu tohoto uživatelského programu přímo exportujeme rozhraní na fyzickou kartu, pak může síťová komunikace probíhat plnou rychlostí, kterou podporuje příslušný hardware. Samozřejmě v takovém případě kartu může používat pouze tento virtuální počítač, to ale nemusí být na závadu (fyzický počítač může mít více síťových rozhraní, ostatní virtuální počítače pak sdílí ta ostatní).

Přestože má paravirtualizace řadu výhod proti plné virtualizaci, potřebuje určité modifikace operačních systémů, což komplikuje její nasazení (zejména u proprietárních operačních systémů) a vede k určité neefektivnosti. Intel proto v poslední době zavedl další systém podpory virtualizace v podobě tzv. *Intel Virtualization Technology* (IVT). Jedná se o rozšíření možností procesorů tak, že přibývá další úroveň ochrany (*ring -1*) pro VMM a přibývají speciální instrukce na této úrovni. Virtuální monitor tak může obsluhovat několik virtuálních počítačů, které již pracují v prostředí, které se neliší od toho, které je k dispozici ve standardních procesorech bez podpory virtualizace. Operační systémy ve virtuálních počítačích není třeba modifikovat, přitom zůstávají základní výhody paravirtualizace, tj. přímé vykonávání instrukcí virtuálního počítače fyzickým procesorem.

3 VServer

I paravirtualizace má však ještě značnou režii, která se projevuje zejména v těch případech, kdy všechny virtuální počítače sdílející hardware pracují se stejným operačním systémem. Pro takový případ byl navržen koncept *Virtuálních privátních serverů* (Virtual Private Servers), který implementuje např. systém *VServer*. V tomto případě je virtualizace realizována až na úrovni aplikačních programů. Namísto virtuálního monitoru běží na počítači jádro standardního operačního systému, v něm jsou pak spouštěny uživatelské virtuální servery, které toto jádro sdílí. Každý z virtuálních serverů tak nabízí pouze uživatelské prostředí (v němž běží uživatelské programy), vzájemná ochrana programů i virtuálních serverů je pak řešena standardními prostředky jádra operačního systému. Koncept VServeru využívá řadu standardních technik, které jsou běžně dostupné v Linuxu a podobných operačních systémech, jejich kombinací pak dosahuje virtualizačního efektu. Pomocí systému „způsobilostí" (capabilities) chrání jednotlivé procesy mezi virtuálními servery (proces je oprávněn provést určité operace jen v prostředí jeho virtuálního serveru, nikoliv v prostředí základního operačního systému). Omezení přidělených zdrojů (resource limits) zase umožňuje za-

jistit, že i při chybě aplikace si žádný virtuální server nebude uzurpovat větší část výkonu či paměti, než mu bylo přiděleno. Prostředí `chroot` společně s důsledným využitím atributů souborů pak zajišťuje bezpečný přístup k přidělené části systému souborů a v něm uložených souborech.

Zatímco v případě paravirtualizace je nutné modifikovat operační systém, v případě VServeru je třeba modifikovat aplikace, zejména pokud používají některé z vlastností (např. způsobilosti), na nichž je tento koncept postaven. Je třeba rovněž upravit celou řadu systémových programů, které poskytují informace o stavu celého systému. Např. systémové volání `uptime` udává, jak dlouho je operační systém aktivní. V případě VServeru by však volání `uptime` nemělo vracet čas běhu základního operačního systému, ale pouze virtuálního privátního serveru, v němž byl `uptime` volán. Jakmile je ale prostředí VServeru vytvořeno, má ze všech virtualizačních technik nejmenší režii a garantuje tak nejlepší využití hardware.

4 Závěr

V současné době je k dispozici celá řada virtualizačních systémů pro procesory Intel, využívajících principů virtualizace, paravirtualizace či konceptů typu VServer. Virtuální počítače se proto začínají ve stále větší míře nasazovat jako nástroje řešení problémů, které jsou bez nich těžko řešitelné nebo pro něž nemáme žádné řešení. Příští pokračování proto bude věnováno podrobnější diskusi nasazení virtuálních počítačů a výhod, které to přináší. □