

O počítači IBM i (System i, iSeries, AS/400)

Vladimír Župka

2009-04-21

Není třeba ani statistiky, abychom ve veřejnosti vypozerovali jistý jev vztahující se k počítačům. Panuje totiž dost pevné přesvědčení, že počítač rovná se osobní počítač, PC. Mluví-li se o počítačových sítích, je v první řadě míněna síť osobních počítačů. A když někdy přijde řeč na servery, má se ovšem na mysli PC server. Informovanější část veřejnosti zná jistě ještě jiné počítače, zejména ty, na kterých lze provozovat operační systém Unix či Linux; starší také vědí něco o počítačích zvaných střediskové či sálové (mainframe).

Počítačový systém s názvem *IBM i* patří mezi ty, které jsou u nás známé jen málo, i když v roce 1998 proběhla reklamní kampaň, která situaci mírně zlepšila. Podrobnější informace o tomto systému jsou však odborné veřejnosti stále málo dostupné. Vynikající kniha F. G. Soltise "AS/400 zevnitř" sice podává velmi podrobné a ucelené informace o architektuře tohoto systému, je však poměrně náročná (ovšem kromě pasáží věnovaných historii).

Pro poslední dobu je charakteristická marketingová snaha počítačové systémy přejmenovávat, aby se vzbudilo zdání, že jde vždy o něco, co je o stupeň lepší než to předchozí, zastaralé. Slovo rebranding, u nás používaný s jakýmkoliv produktem, situaci vystihuje. Proto i u firmy IBM se systém, o němž mluvíme, několikrát přejmenovával, zatímco jeho základní architektura zůstává stejná: System/38, AS/400, eServer iSeries, System i s operačním systémem i5/OS, pak "Power Systems IBM i" s operačním systémem zvaným "i".

V tomto článku bychom chtěli populárnější a stručnější formou podat právě ty informace, podle nichž si čtenář může učinit představu, jak vlastně tento funguje, jaké možnosti poskytuje, a porovnat jej s jinými počítačovými systémy, o nichž již něco ví.

Systém má některé vlastnosti, které jej dost podstatně odlišují od ostatních a předurčují mu dobrou budoucnost.

Nejprve si řekneme něco o jeho historii a pak se podíváme na technické informace.

Historie

Systém 38

Začíná rokem 1978, kdy v laboratořích firmy IBM ve městě Rochester (Minnesota, USA) vznikl kancelářský minipočítač *System/38*. Byl to první počítač, který byl navržen s důrazem na co nejsnadnější vytváření a provozování komerčních aplikací, z čehož vyplynul postup vývoje. Při tom požadavky na software a aplikace byly prvotní a určovaly hardwarové řešení. U jiných počítačů byl a stále ještě je běžný opačný postup, kde návrh hardware je prvotní, a od něj se teprve odvíjí software.

Operační systém tohoto stroje se skrýval pod zkratkou CPF (Control Program Facility - řídicí program). Autoři se snažili vyhnout pojmu operační systém, protože v té době si mnoho uživatelů myslelo, že operační systém je složitá věc vyžadující ke své obsluze tým odborníků. Přestože i tento operační systém byl skutečně složitý, nebylo to navenek vidět,

protože většina obtížných detailů byla před uživateli ukryta. System/38 měl již tehdy adresovací schopnost 48 bitů, což se může zdát překvapivé, když v té době se pracovalo s 16, 24, 32 nebo nejvýše 36 bity. A to byl ještě u Systému 38 kompromis, protože z hlediska návrhu by byla bývala ideální 64bitová adresa. Cena operační paměti, na jejíž kapacitu by 64bitová adresace kladla značné nároky, však v té době byla tak vysoká, že tato varianta byla zamítnuta.

Systém AS/400

Systém 38 vydržel 10 let - do roku 1988, kdy byl ohlášen systém *AS/400* (Application System/400). Ten představoval zdokonalenou verzi Systému 38, stále ještě se 48bitovou adresou. Řídicí program však již vystoupil z anonymity a nesl název Operating System/400, zkráceně *OS/400*. Později byl přejmenován na *i5/OS* a ještě později na "i", ale v dokumentaci z pochopitelných důvodů někdy vystupuje pod svým původním názvem.

Transakční zpracování dat

Systém AS/400 byl v této době určen zejména pro *víceuživatelské transakční a dávkové zpracování dat* s využitím obrazovkových *terminálů* a *integrované relační databáze*. Databázový řídicí systém neměl ještě žádné jméno, takže si mnohdy uživatelé ani neuvědomovali, že nějakou databází mají. Záměrně se totiž (už v Systému 38) využívalo pojmů z tradičního zpracování datových souborů (děroštitkových, magnetopáskových, diskových), tak aby uživatelé nebyli odstrašováni novou terminologií relačních databází, která byla pojmově obtížnější.

Databázový jazyk SQL byl sice také k dispozici, ale protože nebyl integrován do operačního systému a muselo se za něj zvlášť platit, nedoznal mezi uživateli systému AS/400 velkého rozšíření. Uživatelé raději používali tradiční přístup k jednotlivým datovým záznamům (unit record access), převážně za pomoci programovacího jazyka RPG nebo Cobol.

Jazyk RPG vznikl původně jako generátor tiskových sestav (Report Program Generator) a postupem doby se vyvinul v mohutný prostředek k vytváření komerčních aplikací. Již na počítači System/38 (dokonce ještě dříve na počítači System/3 - nejstarším z kancelářských počítačů rochesterské laboratoře) byl jazyk RPG oblíben natolik, že se používá i nyní.

Systém AS/400 byl od počátku orientován především na *transakční i dávkové zpracování* komerčních aplikací (business applications), které kladou velké nároky na vstup a výstup dat. Proto obsahoval alespoň jeden speciální procesor pro *vstup a výstup*, který odebírá podstatnou porci práce hlavnímu procesoru. Hlavní procesor pak nemusel podávat takový výkon, jako kdyby vše musel zvládnout sám. Hlavní procesory rochesterských počítačů nebyly tudíž z nejrychlejších, což umožnilo stanovit dostupnou cenu.

Stojí za povšimnutí, že *víceuživatelské transakční zpracování dat* s použitím alfanumerických terminálů je v systému AS/400 zcela transparentní, protože je začleněno do operačního systému. Naproti tomu jsou transakční systémy pro unixové počítače samostatné produkty (např. Tuxedo) prodávané jako nadstavba nad operačním systémem. Podobná situace je u systému Windows, kde transakční zpracování přichází o mnoho později, a to opět v podobě dodatečných produktů jako je MS Transaction Server, Tuxedo, CICS aj.

Klient - server

Postupem doby však terminálové transakční zpracování přestávalo stačit, protože se objevil fenomén *klient - server*. Do systému byla zařazena podpora pro využití osobních

počítačů jakožto klientů, nejprve jednodušší pod názvem *PC Support/400* (pro prostředí DOS a OS/2), později bohatší pod názvem *Client Access/400* (pro prostředí Windows a Apple Macintosh). Roli víceúčelového serveru (tj. databázového, aplikačního, tiskového) zvládal systém AS/400 velmi dobře. Jako souborový server (pro textové editory apod.) však nevykazoval zrovna nejlepší výkon. Protože architektura AS/400 dovoluje přidávat další specializované procesory, byl problém souborového serveru vyřešen začleněním (integrací) procesoru Intel s dostatečnou vlastní operační pamětí. Jeho název byl nejprve FSIOF (File Server I/O Processor), později IPCS (Integrated PC Server). Druhý název odrážel skutečnost, že PC procesor mohl sloužit jako libovolný *PC server*, nejen souborový. Takových procesorů může být u vyšších modelů Systému i zabudováno několik a každý z nich může provozovat jiný operační systém s určitou aplikací. Původně to byl OS/2 s aplikací File Server nebo Lotus Domino, či Firewall nebo také Novell NetWare.

Nověji se v PC serveru provozuje systém *Windows Server*. PC procesor s výhodou využívá spolehlivého prostředí IBM i (např. diskový prostor, zabezpečení dat, komunikační linky, prostředky pro ukládání a obnovu dat aj.). Systém Windows je dokonce v tomto prostředí spolehlivější než ve svém přirozeném prostředí PC, protože některé jeho chybové stavy napravuje systém. Propojení PC a IBM i se průběžně zdokonaluje. Postupem doby se název PC serveru mění, dnes se nazývá *Integrated xSeries Server* a dovoluje provozovat systémy Windows a Linux.

Začlenění unixových funkcí

V roce 1994 přikročila firma IBM ke změně vnějšího vzhledu počítače a nahradila dosavadní větší béžové skříně menšími černými. Ale nejen to. Do operačního systému přibýly funkce, které přispěly podstatnou měrou k jeho otevřenosti. Šlo o soustavu funkcí umožňující aplikačním programům provádět operace běžné v unixových systémech. Tím byl položen základ k převodu aplikací z *prostředí Unix* do prostředí AS/400. K převodům aplikací také skutečně došlo. Za všechny lze uvést aplikační systémy SAP/R3 nebo Baan. Traduje se historka, že programátoři, kteří se podíleli na takových převodech, se zprvu systému AS/400 štítili, ale v průběhu práce si jej natolik oblíbili, že se stali dokonce jeho fanoušky. V té době systém (AS/400) neprovozoval přímo některý z unixových systémů, ale jen převedené aplikace. Funkce API (Application Programming Interface) byly totiž voleny tak, aby splňovaly specifikaci "Single Unix" (dříve Spec 1170) a specifikaci POSIX. Také nevyčerpávaly veškeré definice těchto specifikací, ale jen ty, které byly vhodné pro komerční výpočty. Od doby, kdy byly zavedeny tzv. *logické oddíly (logical partitions)* a sjednocen hardware s řadou RS 6000, provozují se na systému i pravé operační systémy *AIX* a *Linux*.

Přechod na 64bitovou adresaci

Časem stále rostly nároky uživatelů na výkonnost počítačových systémů, s nimiž se různé firmy vyrovnávaly různě. Firma IBM v roce 1995 u systému AS/400 zvolila cestu náhrady dosavadního vlastního procesoru typu CISC (Complex Instruction Set Computer) sériově vyráběným procesorem PowerPC typu RISC (Reduced Instruction Set Computer) rozšířeným na *64bitovou adresaci* a o několik speciálních funkcí. Dalo by se očekávat, že tak dalekosáhlá změna způsobí uživatelům velké starosti s náhradou dosavadních aplikací novými, nebo vývojářům starosti s modifikací a rekompilací programů. Nic takového se nestalo. Všechny změny spojené s novým procesorem provedla firma IBM uvnitř hardwarově závislé části operačního systému zvané SLIC (System Licensed Internal Code). Přechod aplikací ze 48bitové na 64bitovou architekturu se u uživatelů projevil jen malým zpožděním při prvním spuštění programů, kdy proběhne překlad do nových

instrukcí a pak aplikace pracuje jako dříve, ovšem podstatně rychleji. Jak je to možné, uvidíme dále, až se podíváme na některé rysy architektury tohoto systému.

Objevily se další požadavky na výpočetní systémy. Jde zejména o datové sklady (data warehouses) s vysokými nároky na objem skladovaných dat a na výkonnost procesoru. Podobné požadavky kladou internetové servery s vysokými nároky na zabezpečení. Proto v Systému i figurují stále rychlejší procesory, v závislosti na modelu sdružené po čtyřech až dvaatřiceti v režimu symetrického multiprocesingu (SMP); stále se rozšiřuje operační paměť (desítky terabajtů) i disková paměť prakticky neomezeně (v počtu stovek terabajtů). Navíc lze spojovat systémy do shluků (clusters) pomocí optických nebo tzv. High Speed Link spojů.

Firma IBM propaguje pro vývoj aplikací systém *Java*, o kterém jsou čtenáři odborných publikací poměrně dobře informováni. Nakolik se *Java* prosadí v soutěži s tradičními prostředky používanými v Systému i (jazyky RPG, Cobol, C, C++), ukáže teprve čas.

V poslední době se dostává do popředí také jazyk PHP v souvislosti s tzv. modernizací. Umožňuje totiž spolupráci grafického uživatelského rozhraní na webu s ostatními prostředky operačního systému "i", zvláště s databází.

Architektura systému

Systém "IBM i" se odlišuje od běžných výpočetních systémů několika specifickými vlastnostmi. Ty jsou takové, že umožňují systému téměř neomezený vývoj. Jde zejména o tyto rysy:

- technologicky nezávislé rozhraní mezi softwarem a hardwarem,
- objektová orientace,
- jednoúrovňová paměť v jediném virtuálním adresním prostoru,
- transparentní hierarchická soustava procesorů.
- integrovaný operační systém zahrnující i databázi, komunikace, zabezpečení aj.

Na každý z těchto rysů se podíváme podrobněji.

Technologicky nezávislé rozhraní

Strojové instrukce

Nejprve si připomeňme některé zdánlivé samozřejmosti, které nám umožní lépe pochopit princip nezávislosti na hardwaru. Každý počítač je vybaven procesorem (či několika procesory), který umí zpracovávat příkazy zvané strojové instrukce. Soustava všech instrukcí a datových formátů daného procesoru tvoří rozhraní mezi hardwarem a softwarem v tom smyslu, že software je sestaven s použitím těchto strojových instrukcí a dat. Jak známo, software čili souhrn programů se nevytváří tak, že by programátor zapisoval přímo strojové instrukce. Tento způsob programování se sice kdysi skutečně používal, ale zanikl již v samém začátku počítačové historie.

Kompilátory

Vývoj programovacích prostředků šel cestou, které se říkalo automatizace programování, kdy se nejprve číselné adresy nahradily symbolickými adresami a číselné kódy operací se nahradily mnemotechnickými zkratkami. O převedení takových symbolických instrukcí se postaral speciální překládací program, který byl původně napsaný ve strojových instrukcích, později v symbolických instrukcích. Překládacímu programu se začalo říkat překladač, translátor, autokód, *assembler* apod. a vstupnímu textu *zdrojový kód* (source code). Teprve výsledný produkt překladače byl schopný spuštění, tj. mohl sloužit přímo

jako sled povelů pro procesor. Později vznikly programovací jazyky jako je FORTRAN, Cobol, a mnoho dalších. Překladače z těchto jazyků se nazývají *kompilátory* a procesu překladač z programovacího jazyka do strojového kódu se říká *kompilace*.

Dnes se většina operačních systémů vytváří pomocí některého z programovacích jazyků, zejména C nebo C++. Zároveň však se některé citlivé části operačních systémů píšou v jazyku assembleru pro příslušný procesor, protože kompilátor vyššího jazyka by nevytvořil optimální (tj. nejrychleji fungující) sled strojových instrukcí. Sledu strojových instrukcí a dat se často říká prostě *kód*. Je zcela zřejmé, že kód vzniklý kompilací je zcela závislý na hardwaru.

Virtualizace strojového kódu v systému 38

Už při návrhu počítače IBM System/38 učinili návrháři jedno velmi důležité rozhodnutí. Aby snížili závislost programovací práce na použitém procesoru, vymysleli hypotetický, můžeme říci i *virtuální procesor* se soustavou instrukcí připomínajících obvyklé strojové instrukce, které však neprováděl žádný existující procesor. Je jasné, že takové instrukce jsou zcela nezávislé na jakémkoliv hardwaru.

Instrukční soustava Machine Interface (MI)

Tato pomyslná instrukční soustava byla nazvána *TIMI (Technology Independent Machine Interface)*, kratěji *MI*. Co však s takovými instrukcemi, když je žádný procesor nemůže provádět? Odpověď je tato: Vytvoříme *překladač*, který takové "strojové instrukce" přeloží do pravých instrukcí *pro právě používaný procesor*. Skutečných výsledných instrukcí bude v programu zpravidla více než "fiktivních" MI instrukcí. Kompilátory z programovacích jazyků přeloží zdrojový kód nejprve do kódu MI, který pak v koncové fázi kompilace převezme překladač a vytvoří z něj opravdový strojový kód. Proces kompilace je tedy v podstatě dvoustupňový (i když první stupeň kompilačního procesu sám probíhá v několika stupních).

Z definice rozhraní MI vyplývá, že nepracuje s žádnými hardwarovými registry, ani s adresami paměti. Instrukce MI obsahují sice operační kód a operandy, podobně jako jiné strojové instrukce, ale ty jsou pouze symbolické. Operandů představují pouze odkazy (relativní adresy) na místa v MI programu, kde jsou umístěny jejich definice (např. délka a typ). Rozhraní MI nezná žádnou paměť, tu operandům přidělí na základě definic až překladač (poslední fáze kompilace), který teprve s pamětí pracuje. Strojové instrukce vzniklé překladem pak již pochopitelně obsahují příslušné adresy operandů ve strojové paměti, popřípadě zásobníkovou paměť. MI program si tedy můžeme představit jako poněkud zvláštní formu binárního zdrojového kódu napsaného v poněkud nezvyklém programovacím jazyku. Rozhraní MI bylo zvoleno na optimální úrovni, ne příliš nízko, aby kopírovalo některou konkrétní soustavu strojových instrukcí, ani příliš vysoko, aby připomínalo některý z vyšších programovacích jazyků.

Překlad z MI do strojového kódu

Proces překladač z MI do strojového kódu připomíná provádění strojových instrukcí v některých procesorech pomocí tzv. mikroprogramů neboli *firmware*. Tam každá strojová instrukce také podléhá transformaci, ale ne při překládání programu, nýbrž až při jeho provádění. Říkáme, že strojová instrukce je *interpretována* nebo *emulována* mikroprogramem. *Mikroprogram* je tvořen (mikro)instrukcemi ještě těsněji spjatými s hardwarem. Zatímco však *mikroinstrukce* jsou považovány za součást hardware (používá je jen firma vyrábějící hardware), instrukce jsou k dispozici programátorům (dnes už spíše kompilátorům). Za příklad mohou sloužit novější procesory Intel, kde instrukce x86 jsou za běhu interpretovány (emulovány) speciálními mikroprogramy pro RISC

procesor obsažený ve stejném čipu. Analogie mezi mikroprogramovanými strojovými instrukcemi na jedné straně a instrukcemi MI na druhé straně je poměrně přesná. Zatímco však strojové instrukce jsou prováděny stabilními, předem připravenými mikroprogramy teprve při spuštění programu, z instrukce MI je nejprve při překladu vygenerován "in-line" kód, tedy sled strojových instrukcí. Těch sledů (sekvencí) je tolik, kolik instrukcí MI je v původním programu.

Přeložený program jako objekt

Celý přeložený kód se po překladu uskladí do paměti jakožto objekt, který je schopen spuštění, a zároveň s ním se uloží i původní MI program jako tzv. *šablona – template*. Později, při spuštění takto vytvořeného objektu (programu), již procesor provádí jen opravdové strojové instrukce, aniž se zabývá instrukcemi MI.

Implementace virtuálního procesoru

Rozhraní MI je ve skutečnosti libovolně rozšiřitelné. Nemusí obsahovat jen výpočetní instrukce podobné tradičním (např. aritmetické, přesunové, skokové apod.), ale obsahuje také instrukce *volání systémových funkcí*, např. pro vstup a výstup dat. Tyto instrukce se už nepřekládají do in-line sekvencí, ale jsou vlastně interpretovány (emulovány) složitými programy uvnitř vrstvy *SLIC (System Licensed Internal Code)*, což je hardwarově závislá vrstva operačního systému (jádro). Tato skutečnost byla také bohatě využita v době, kdy do MI byly přidány funkce pro Unix. Těmto složitějším MI funkcím se říká *API (Application Program Interface)*. Ve vrstvě SLIC jsou také obsaženy všechny funkce operačního systému, které závisí na hardwaru, a ty jsou velmi početné. Ještě nověji byla do vrstvy MI zařazena soustava binárních instrukcí *JVM (Java Virtual Machine)* též implementovaná ve vrstvě SLIC. Je vidět, že vrstva SLIC je velmi tlustá, tj. obsahuje obrovské množství programů momentálně složených z instrukcí 64bitového RISC procesoru. Při přechodu na architekturu RISC bylo nutné dřívější hardwarově závislou vrstvu přeprogramovat. K tomu byl použit hlavně jazyk C++, který poskytuje všechny výhody objektově orientovaného programování, zejména zvýšenou produktivitu programátorských prací.

Šablona virtuálního programu (MI template)

Překladač z MI do strojového kódu hrál také podstatnou roli v převodu aplikací z instrukcí CISC do instrukcí RISC. Existuje-li MI program ve formě šablony jako součást převáděného programového objektu (o objektech bude řeč dále), provede se při prvním spuštění tohoto programu překlad ze šablony do nového strojového kódu a starý kód se zahodí. Překlad provede překladač obsažený v novém operačním systému. Při dalším používání programu se již žádný překlad neprovádí. Na otázku, jak nový překladač ví, zda má či nemá překlad provést, je odpověď taková, že v programovém objektu je kromě MI programu uložena také informace o tom, jakou verzi strojového kódu programový objekt obsahuje, a tu operační systém při každém vyvolání programu zjišťuje. Existence programu ve formě MI (šablony - template) se nazývá *pozorovatelnost (observability)*. Protože MI předloha zabírá poměrně velký objem paměti, existuje možnost, jak ji z programového objektu odstranit, a mnozí dodavatelé aplikací to také dělají. Program, který není pozorovatelný, do nového systému pochopitelně převést nelze. Dodavatelé aplikací jsou však na tento případ připraveni a buď zabezpečí pozorovatelnost programů, nebo dodají novou, již převedenou verzi.

Objektová orientace

Všechny údaje jsou v Systému i uloženy jako *objekty*. Objekt je určitá vyhrazená část paměti, která je předepsaným způsobem strukturována. Obsahuje *záhlaví a funkční část*.

V záhlaví jsou uvedeny obecné informace o objektu, které musí obsahovat každý objekt (jméno objektu, typ objektu, popisný text, datum a čas vytvoření, odkaz na zdroj, z něhož byl objekt vytvořen atd.). Funkční část objektu obsahuje informace závislé na tom, o jaký typ objektu jde.

V systému existuje značné množství typů objektů, které reprezentují programy, databázové tabulky, příkazy řídicího jazyka, fronty zpráv, definice úloh (jobs), uživatelské profily, konfigurace komunikačních zařízení a mnohé další. Každý *typ objektu* lze zpracovávat jen jemu určenými příkazy. Např. obsah řádků databázové tabulky lze zobrazit určitými příkazy, zatímco strojové instrukce programového objektu lze jen spustit, ale ne zobrazit. Typy objektů v IBM i připomínají typy objektového programování (např. *class* v jazyku Java), uživatel si však nemůže vytvářet své vlastní typy, ani si odvozovat další typy z existujících typů. Typy systémových objektů jsou tedy pevně stanoveny a nové může přidávat jen firma IBM. To pochopitelně neznamená, že bychom při používání jazyka C++ nebo Java nemohli vytvářet a odvozovat nové typy (třídy) objektů; nejsou to však systémové typy objektů, ale uživatelské typy příslušného jazyka.

Objekty mohou být buď *trvalé* (permanent) nebo *dočasné* (temporary). Trvalé objekty jsou např. programy, databázové tabulky aj., dočasné objekty jsou např. informační struktury potřebné pro provoz úlohy (proces, job), obsahující otevřené přístupové cesty k datům, statické a automatické proměnné pro spuštěný program apod. Zatímco trvalé objekty přetrvávají tak dlouho, dokud je oprávněný uživatel výslovně nezničí příslušným příkazem, jsou dočasné objekty rušeny pokaždé, když se systém spouští. Spuštění systému se nazývá IPL (Initial Program Load), což znamená počáteční zavedení programu (u jiných počítačů se používá pojem *bootstrap* či *boot*). Bylo by sice možné rušit dočasné objekty hned při ukončení úlohy, ale to by mohlo zdržovat ostatní úlohy; proto bylo zvoleno hromadné rušení dočasných objektů v době IPL.

Knihovny objektů

Organizace objektů spočívá v tom, že se sdružují do tzv. knihoven (libraries). Knihovny lze přirovnat k adresářům (directories) v jiných systémech. Rozdíl je v tom, že hierarchie knihoven je jen jednoúrovňová. Nejvyšší knihovna (kořenová) je jen jedna a jmenuje se QSYS. V ní jsou zapsány všechny ostatní knihovny jako objekty typu *LIB. Ostatní knihovny již nemohou obsahovat další knihovny. Pro každou běžící úlohu (proces) je k dispozici seznam knihoven (library list), v němž se objekty postupně vyhledávají směrem od první knihovny k poslední. Seznam lze pro každou úlohu nastavit individuálně příslušnými příkazy. Seznam knihoven lze přirovnat k sestavě cest příkazu SET PATH v systému DOS pro PC, až na to, že seznam knihoven je prostý, bez hierarchické struktury.

Integrovaný souborový systém IFS

Požadavek otevřenosti si vynutil vytvoření dalšího způsobu organizace objektů, a to tak, aby zahrnoval většinu adresářových systémů používaných ve známých operačních systémech. Výsledný produkt byl nazván IFS (Integrated File System - integrovaný souborový systém). Název vyplývá z toho, že jde o hierarchickou adresářovou strukturu, která zastřešuje dosavadní struktury organizace objektů a zahrnuje nové. Základní, kořenový adresář (root) je analogie adresářové struktury systému OS/2 nebo Windows. Jiné adresáře jsou určeny pro systémy Unix aj. Konvence pro zápis jmen adresářů a objektů v nich je vždy shodná s konvencí pro příslušný operační systém. Unixovské cesty pochopitelně rozlišují malá a velká písmena ve jménech adresářů. Objekty na koncích adresářové struktury se nazývají *proudový soubor* (stream file) a jsou uloženy jako spojitá posloupnost bajtů, jejichž strukturu zná jen aplikace, pro kterou je objekt

určen, ne však operační systém. V rámci operačního systému je možné jimi manipulovat jako s celkem (přemisťovat, kopírovat, rušit apod.) a v programech lze používat velké množství funkcí API k jejich zpracování. Specializované adresáře a jejich objekty (soubory) jsou určeny také pro aplikace provozované na integrovaném PC procesoru.

Jednourovňová virtuální paměť

Operační a disková paměť tvoří základ pro tzv. virtuální paměť. *Virtuální paměť* je vlastně prostor obsahující tolik bajtů, kolik dovoluje adresovací schopnost procesoru. Máme-li tedy 64bitovou adresu, obsahuje virtuální paměť 2^{64} bajtů, což je obrovské číslo, vyjádřené dekadicky jako 18,446.744,073.709,551.606. Toto číslo představuje zhruba počet milimetrů ve dvou světelných letech nebo známý počet obilných zrn kladených vždy ve dvojnásobném počtu na políčka šachovnice. Takový počet jistě dostačí k adresování velkokapacitních pamětí ještě velmi dlouho. Všimněme si, že virtuální prostor je jen jediný a souvislý, adresy se pohybují v rozmezí 0 až $2^{64} - 1$ (od 64 binárních nul do 64 binárních jedniček).

V počítačích s 32bitovou adresou se také pracuje s virtuální pamětí a virtuálním prostorem. Většinou však je virtuální prostor omezen na 4 GB, tj. 2^{32} , což dnes nestačí pokrýt ani dostupnou diskovou kapacitu. Proto se každému procesu (úloze) přiděluje samostatný virtuální prostor, kde data vznikají v době existence procesu a zanikají zároveň s procesem. Současně probíhající procesy nemohou sdílet objekty v cizí virtuální paměti. Aby různé procesy mohly sdílet data, je nejprve nutné je uložit ve formě souborů do souborového systému. To v systému IBM i není nutné, protože i souborový systém je zahrnut ve virtuální paměti a procesy jej tedy mohou sdílet.

Virtuální paměť a stránkování

Žádný počítač v dnešní době nemá dost fyzické kapacity, aby obsáhla celou virtuální paměť o velikosti 2^{64} bajtů. Je tedy nevyhnutelně třeba mít nějaký mechanismus (algoritmus), který by libovolně velkou virtuální adresu převedl (mapoval) na adresy operační a vnější (dnes ještě diskové) paměti. Tomuto mechanismu se říká stránkování (paging). Operační paměť se rozdělí na stejně dlouhé úseky (4096 bajtů), které se nazývají *paměťové stránky*. Je jasné, že stránek v operační paměti je tak málo, že se do ní všechny objekty nevejdou. Stále je značný nepoměr mezi velikostí operační paměti a velikostí vnější paměti (vnější je přibližně o dva řády větší). Proto se momentálně nepotřebné paměťové stránky ukládají na vnější paměť (diskovou), kde se nazývají *paměťové rámy* (page frames). Když program potřebuje stránku, která není právě v operační paměti, nastane tzv. výpadek stránky (page fault) a stránkovací mechanismus přečte potřebný stránkový rám do příslušné paměťové stránky. To je jen velmi částečně a zhruba nastíněná činnost stránkování. Ve skutečnosti jde o velmi propracovaný algoritmus, který obsahuje řadu optimalizačních prvků pro zajištění co nejvyšší výkonnosti systému při mapování virtuálního prostoru do fyzické paměti. Znovu je vhodné připomenout, že veškerá disková paměť je k dispozici pro stránkovací mechanismus, a tedy pro virtuální paměť. Pochopitelně, že na disky se vejde jen vybraná část virtuální paměti, která však nemusí být adresována souvisle.

Připomeňme, že virtuální paměť a stránkování jsou pojmy, o nichž programy nad rozhraním MI (operační systém a aplikace) nevědí nic. Jde o mechanismus, s nímž pracuje výhradně vrstva SLIC, která je závislá na hardwaru.

Uložení a sdílení objektů

Všechny objekty včetně datových souborů jsou uloženy ve virtuální paměti, která umožňuje *ukládat data natrvalo*. Na rozdíl od jiných systémů nemá systém IBM i žádný souborový systém mimo virtuální paměť. Virtuální paměť je realizována celou operační i vnější pamětí. Disky nejsou považovány za zvláštní zařízení (diskové jednotky), ale za organickou součást souvislé virtuální paměti. Znamená to, že všechny *objekty* ve virtuální paměti *mohou být sdíleny* více uživateli nebo procesy. V jiných počítačích mohou být sdíleny jen datové soubory v souborovém systému, nikoliv však ve virtuální paměti.

Kdyby v budoucnosti byly např. diskové paměti nahrazeny nějakou zcela jinou paměťovou technologií, aplikační programy by to vůbec nepoznaly. Firma IBM by ovšem měla práci se začleněním nového hardwaru do vrstvy SLIC a uživatelé by museli překopírovat své objekty ze staré paměti do nové.

Multiprocesorová hierarchie

Již od první chvíle obsahuje systém 38 (AS/400, iSeries, System i, IBM i) kromě hlavního procesoru také alespoň jeden separátní procesor pro vstup a výstup (I/O procesor). U větších modelů bývá takových I/O procesorů více, např. zvláštní procesor pro diskové paměti, zvláštní procesor pro páskové jednotky atd., ale také několik integrovaných PC procesorů. V poslední době přibylo také hlavních procesorů (4 až 32), takže ve velkých modelech existuje soustava několika rovnocenných hlavních procesorů a několika vedlejších procesorů. Vedlejší procesory mají tu dobrou vlastnost, že hlavním procesorům odebírají zátěž spojenou se vstupem a výstupem dat, a tak podstatně zvyšují celkovou průchodnost systému. Když si uvědomíme, že komerční aplikace využívají vstup a výstup dat velmi intenzívně, pochopíme zásadní důležitost procesorové hierarchie pro výkonnost počítače. Poznamenejme, že samotný systém IBM i nepoužívá grafické procesory (akcelerátory), protože nemá vlastní grafické zařízení jako třeba PC. Grafické uživatelské rozhraní se většinou realizuje na připojených osobních počítačích (PC).

I/O procesory v systému AS/400 byly specializované procesory, které hostily operační systémy reálného času (real time operating systems) vytvořené speciálně pro příslušná I/O zařízení. V poslední době se v zájmu snížení ceny používají stejné procesory jako hlavní procesor, který je již dostatečně levný. V budoucnu není vyloučeno, že funkci I/O procesorů převezmou hlavní procesory, když budou dostatečně rychlé a bude jich dost.

Integrované PC servery však jsou nadále založeny na bázi procesorů Intel umožňujících provoz systému Windows nebo Linux.

Integrovaný operační systém

Operační systém se skládá ze dvou částí. Jedna část je "i" čili i5/OS čili OS/400 a je naprogramovaná s použitím instrukcí MI, tedy *nezávislá* na hardwaru. Druhá část je vrstva SLIC (System Licensed Internal Code) a představuje operační systém v užším slova smyslu. Vrstva SLIC je sestavena z instrukcí skutečného procesoru, a je tudíž *závislá* na hardwaru. OS/400 leží nad rozhraním MI, zatímco SLIC leží pod ním. Obě části se doplňují a všechny systémové funkce jsou vhodným způsobem mezi ně rozděleny. Ve vrstvě SLIC jsou zařazeny ty součásti systému, které nemohou být nezávislé na hardwaru (např. správa virtuální paměti nebo komunikační funkce), dále ty, které vyžadují optimalizaci a musí tedy přímo využívat vlastnosti hardwaru (např. vyhledávání

údajů v databázi nebo fyzický vstup a výstup dat) a ty, které musí být chráněny před neoprávněným přístupem (např. evidence hesel nebo oprávnění k různým objektům).

Integrovaná databáze

Co je však na systému IBM i snad to nejdůležitější, je jeho funkční celistvost - integrita. Markantním příkladem je systém řízení databáze. Databázový systém se nejprve nejmenoval vůbec nijak, pak byl pojmenován DB2/400, pak DB2 for i5/OS, dnes se nazývá DB2 for i, ale stále je organickou součástí celého systému. Nejde tedy o zvláštní produkt, který by bylo nutné přikoupit, ale o souhrn funkcí, kterými je prostoupen celý systém. Například vyhledávání údajů podle indexu se používá jak pro relační databázové tabulky, tak pro nejrůznější objekty systému, např. vyhledání objektu v knihovně nebo uživatelského profilu v seznamu profilů. Dokonce i zdrojové texty programů jsou umístěny v databázovém souboru jako samostatně pojmenované členy (members) a jejich editory vlastně pracují s řádky textu jako s databázovými řádky (záznamy).

Integrované komunikace

Podobně jsou všechny hardwarově nebo protokolově závislé komunikační programy soustředěny do vrstvy SLIC, zatímco konfigurační objekty (popisy) se vytvářejí v rámci MI tak, aby uživatel měl s komunikacemi co nejméně starostí. Zde má integrace komunikačních protokolů (SNA, TCP/IP, IPX aj.) do operačního systému tu výhodu, že používají společné prostředky (např. přenosové programy nižší úrovně, zabezpečovací funkce apod.) a jsou prezentovány uživateli vhodnou stabilní formou.

Důsledky integrace

Obecně lze říci, že integrace pokročilých komponent, které se obvykle nepočítají k operačnímu systému (databáze, komunikace, zabezpečení) zaručuje jejich hladkou a spolehlivou součinnost se zbytkem operačního systému.

Jestliže se kupříkladu databázový soubor (tabulka) během provozu zcela zaplní, vydá o tom systém zprávu operátorovi. Operátor pak může odpovědět tak, aby systém přidal k souboru další prostor a pokračoval v práci. Uživatel může tedy během zpracování příkazem zvětšovat prostor v souboru, aniž by musel úlohu zastavit. Tato zdánlivá drobnost, ale ve skutečnosti velmi důležitá vlastnost, není u jiných systémů vůbec známá; tam v takových případech bývá zpravidla nutné zastavit provoz, překopírovat data, rozšířit prostor na disku, vrátit data, a pak teprve obnovit provoz.

Podobně příznivé vlastnosti vykazuje integrovaný systém i v případě evidence uživatelů, hesel, oprávnění, připojování přídavných zařízení za provozu a v dalších případech. Proslulá stabilita systému IBM i je dána z velké části právě integrací zmíněných funkcí, za něž ručí výrobce systému.