Virtuální paměť

Virtuální paměť (též **virtualizace paměti**) je v informatice způsob správy operační paměti počítače, který umožňuje předložit běžícímu procesu adresní prostor paměti, který je uspořádán jinak nebo je dokonce větší, než je fyzicky připojená operační paměť RAM. Z tohoto důvodu procesor rozlišuje mezi *virtuálními* adresami (pracují s nimi strojové instrukce, resp. běžící proces) a *fyzickými* adresami paměti (odkazují na konkrétní adresové buňky paměti RAM). Převod mezi virtuální a fyzickou adresou je zajišťován samotným procesorem (je nutná hardwarová podpora) nebo samostatným obvodem.

V současných běžných operačních systémech je virtuální paměť implementována pomocí stránkování paměti spolu se stránkováním na disk, které rozšiřuje operační paměť o prostor na pevném disku (stránkování na disk je nesprávně označováno jako *swapování*).

Historie

Princip virtuální paměti byl rozpracován na přelomu 50. a 60. let 20. století. Z pozorování využití paměti RAM vyplynulo, že program nepracuje s celou pamětí najednou, ale využívá její omezené části, které se spolu s během programu postupně mění. Koncem 60. let se začala podpora virtuální paměti stávat standardem operačních systémů určených pro sálové počítače.

Operační paměť RAM je rozšířena o místo na pevném disku, které je sice výrazně pomalejší, ale také výrazně lacinější. Vyšší režie i nároky na systémové a hardwarové zdroje (zvláště na diskový subsystém) jsou zanedbatelné v porovnání s výhodami, které použití virtuální paměti přináší. Systém virtuální paměti umožňuje efektivně využít menší operační paměť za cenu akceptovatelné ztráty výkonu a přináší výhody dokonce i v případě, že k odložení paměti na disk nedojde (např. mmap).

Virtuální paměť je efektivní, neboť mnohdy není třeba do operační paměti zavádět celé kódy programů. To je především z těchto důvodů:

- Programy mají tendenci zůstávat po jistou dobu v rámci několika procedur.
- Datové struktury jsou často lineární, další záznam je „sousedem“ právě zpracovávaného.
- Mimo volání procedur a skoků je program vykonáván sekvenčně.

Výhody virtualizace

- Paměť, kterou má běžící proces k dispozici, není omezena fyzickou velikostí instalované paměti.
- Je omezeno plýtvání pamětí, kterou proces ve skutečnosti nevyužije nebo ji začne využívat až později.
- Každý běžící proces má k dispozici svou vlastní paměťovou oblast, ke které má přístup pouze on sám a nikdo jiný.
- Paměť jednotlivým procesům lze tak organizovat, že se paměť z hlediska procesu jeví jako lineární, přestože ve skutečnosti může být umístěna na různých místech vnitřní paměti i odkládacího prostoru.

Nevýhody virtualizace

Při nedostatečné kapacitě fyzické operační paměti může dojít ke ztrátě výpočetního výkonu (thrashing). Pokud proces nemá v paměti dost stránek, často generuje přerušení **výpadek stránky**. To zaměstnává procesor a zpomaluje běh dalších procesů, ale hlavně tohoto procesu, neboť je v době načítání stránky pozastaven a nevykonává žádnou činnost. Dále jsou do paměti zaváděny další procesy, neboť procesor často čeká na ukončení vstupně-výstupní operace (právě zavádění a případné odkládání stránek). To vede k dalšímu zhoršení výkonu. (Je ovšem nutné zmínit, že bez virtuální paměti by v takovém případě výpočet vůbec nemohl proběhnout.)

Principy virtualizace

Všechny adresy, které proces používá, jsou spravovány pouze jako virtuální – transformaci na fyzické adresy provádí správa virtuální paměti.

Každý proces po spuštění obdrží od operačního systému oblast pro uložení programu a pro data. Pokud potřebuje více operační paměti, může operační systém požádat o přidělení další paměti. Pokud se jí nedostává, operační systém odsune části obsahu operační paměti na disk a provede příslušné úpravy ve stránkovacích registrech procesu.

Procesům je buď přidělen pevný počet rámců (ne nutně stejná hodnota pro všechny procesy, přidělený počet rámců muže být úměrný velikosti programu), nebo je použito tzv. prioritní přidělování, kdy procesům s větší prioritou je dáno více rámců.

Existují dvě základní metody implementace virtuální paměti – **stránkování** a **segmentace**.

- Při stránkování je paměť rozdělena na větší úseky stejné velikosti, které se nazývají stránky. Správa virtuální paměti rozhoduje samostatně o tom, která paměťová stránka bude zavedena do vnitřní paměti a která bude odložena do odkládacího prostoru (*swapu*).
- Při segmentaci je paměť rozdělena na úseky různé velikosti nazývané segmenty.

Používají se dvě základní politiky:

- **Stránkování (resp. segmentace) na žádost**, kdy se stránka zavádí pouze jako důsledek přerušení typu výpadek stránky.
- **Předstránkování**, kdy se počítá, že proces bude brzy pravděpodobně odkazovat na sousední stránku na sekundární paměti (používá např. Windows XP).

Odkládání lze provádět:

- do souboru proměnné délky (nejpomalejší varianta),
- do souboru pevné délky (o něco rychlejší, neboť nedochází k fragmentaci v souborovém systému, ale dochází k plýtvání místem; na druhé straně může dojít k nedostatku virtuální paměti),
- přímo do vyhrazeného oddílu na pevném disku (nejvýkonnější varianta, diskový oddíl neobsahuje souborový systém, stránkovací registry obsahují přímé adresy na disku),
- kombinovaně (vyhrazený oddíl na pevném disku lze v případě potřeby doplnit swapovacím souborem pevné, resp. proměnné délky).

Odkaz na stránku (či segment) mimo operační paměť způsobí přerušení výpadkem stránky a následuje přibližně toto:

- OS pozastaví proces, kterému chybí stránka.
- OS spustí modul pro zavedení chybějící stránky do operační paměti a v případě nutnosti (je-li operační paměť plná) odstraní z fyzické paměti stránku podle jedné z uvedených strategií.
- Během přenosu požadované stránky je proces, kterému chybí stránka nadále pozastaven a běží jiné procesy.
- Po zavedení stránky je původní proces označen jako připravený a čeká na přidělení procesoru.

Některé stránky, např. vstupně-výstupní vyrovnávací paměť nebo vyhrazené stránky OS, nelze odložit na disk.

Určení optimální velikosti stránek

Volba malé velikosti stránky vede ke zmenšení vnitřní fragmentace (méně plýtvání místem uvnitř alokovaných oblastí) a zmenšení počtu výpadků stránek. Nevýhodou je ale veliká tabulka stránek (zabírá více operační paměti) a náročnější prohledávání tabulky stránek (při odstraňování stránky). U velkých stránek nastávají opačné problémy.

Strategie výběru oběti

Požadujeme minimální frekvenci výpadku stránek. Stránka, která je vybrána k odstranění z paměti je nazývána také jako **oběť**.

OPT

Optimal – nahraď tu stránku (segment, položku cache), která bude nejpozději znovu zapotřebí. S výjimkou experimentálních systémů se jedná o čistě teoretickou strategii (protože nelze předvídat chování programu) používanou pro teoretický výzkum účinnosti ostatních strategií.

FIFO

First in, first out – nahraď nejstarší stránku. Ta ovšem může být stále používána, tato strategie proto není příliš efektivní. Je ale jednoduchá na implementaci.

FIFO s druhou šancí

Stránka při použití získává „život k dobru“ (max. jeden, nesčítají se). Je-li vybrána za oběť, tento druhý život ztrácí. Staré, ale často používané stránky přežívají, efektivita strategie se blíží LRU.

LRU

Least Recently Used – nahraď nejdéle nevyužívanou stránku. Při přesné implementaci vyžaduje náročné udržování informací o používaných stránkách. Varianty:

- LRU řízen hodinami (vyhazuje se stránka s nejstarším záznamem o použití), SW i HW náročné, prohledávání v poli.
- Zásobníková implementace. Při použití přesune stránku na vrchol zásobníku. Oběť je na dně zásobníku. SW příliš náročné, proto pouze HW implementace.

LFU

Least Frequently Used – nahraď nejméně využívanou stránku. Při přesné implementaci vyžaduje náročné udržování informací o používaných stránkách. Pro uchovávání počtu přístupů implementován tzv. čítač přístupů.

Pseudo-LRU

Označení pro algoritmus podobný LRU s omezenou (často drasticky) přesností. V praxi se často používá právě Pseudo-LRU, případně se stopami LFU.

MFU

Most Frequently Used – obětí je stránka s největší hodnotou čítače. Úvaha je taková, že stránka s nejmenší hodnotou čítače přístupů byla právě zavedena do paměti a ještě nestihla být použita.

Zdroje článků a přispěvatelé

Virtuální paměť *Zdroj*: https://cs.wikipedia.org/w/index.php?oldid=11230848 *Přispěvatelé*: Ben Skála, Beren, Bruce Shorty, Chiak, Fraxinus, Hkmaly, Hugo, Jvs, Ludek, Mercy, Miaow Miaow, Milan Keršláger, Onovy, Pastorius, Porthos, Postrach, Singularita, ToOb, Twardowski, Unknown entity, Zirland, 11 anonymní úpravy

Licence