Systemy Operacyjne — Pamięć wirtualna cz. 2

Arkadiusz Chrobot

Zakład Informatyki, Politechnika Świętokrzyska w Kielcach

Kielce, 8 stycznia 2020

Plan wykładu

Plan wykładu

Plan wykładu

Plan wykładu

Plan wykładu

Plan wykładu

Plan wykładu

Plan wykładu

Plan wykładu

Plan wykładu

Plan wykładu

Plan wykładu

Plan wykładu

Plan wykładu

Plan wykładu

Plan wykładu

Plan wykładu

Plan wykładu

Plan wykładu

Wstęp

W poprzedniej części wykładu ustaliliśmy, że wpływ na efektywność stronicowania na żądanie, oprócz algorytmu wymiany stron, może mieć również metoda przydziału wolnych ramek procesom. Kiedy w systemie pracują tylko dwa procesy, system operacyjny i proces użytkownika, to przydział ramek jest prosty. Podział zbioru wolnych ramek jest dokonywany tak, aby proporcja była korzystna dla procesu użytkownika (w końcu systemy komputerowe są tworzone po to by wykonywać procesy użytkownika, a system operacyjny ma tylko nimi zarządzać i ułatwiać ich pracę). Jeśli się skończy pula wolnych ramek, to strony procesu użytkownika lub systemu operacyjnego podlegają wymianie. Możliwe są również scenariusze, w których system przekazuje część swoich ramek na rzecz procesu użytkownika (np. ramek przeznaczonych na bufory, które aktualnie nie są używane) lub w których system utrzymuje zawsze pewną liczbę wolnych ramek, aby usprawnić proces wymiany stron. Problem przydziału ramek komplikuje się, gdy mamy do czynienia z systemem wielozadaniowym.

Minimalna liczba ramek

Aby proces mógł wykonać choć jeden rozkaz, to w pamięci komputera muszą się znajdować jednocześnie wszystkie strony, których ten rozkaz może dotyczyć. Ponieważ nie można określić jaki to będzie rozkaz przed załadowaniem programu do pamięci, to system operacyjny zakłada najgorszy scenariusz i przydziela tyle ramek, aby mogły się w nich zmieścić wszystkie strony konieczne do wykonania najbardziej złożonego rozkazu na liście rozkazów procesora. Przez złożoność rozumiemy tu złożoność jego trybu adresowania. Najbardziej korzystnie pod tym względem wypadają procesory RISC, w których istnieją zazwyczaj tylko proste tryby adresowania, najgorzej procesory, które pozwalają na wielopoziomowe adresowanie pośrednie (należy ograniczyć liczbę poziomów do pewnej liczby, po której przekroczeniu generowany jest wyjątek).

Algorytmy przydziału ramek

W systemach wielozadaniowych możliwych jest wiele strategii przydziału programom użytkownika wolnych ramek. Najprostszy jest *przydział równy* w którym każdemu procesowi przydziela się taką samą liczbę ramek. Liczba ta jest określona wzorem m/n, gdzie m jest całkowitą liczbą wolnych ramek, a n jest liczbą procesów. Ponieważ każdy z nich ma inne zapotrzebowanie na pamięć, taki przydział nie jest optymalny. Bardziej sprawiedliwy jest *przydział proporcjonalny*, który uwzględnia rozmiar każdego z procesów. Załóżmy, że s_i jest wielkością pamięci wirtualnej procesu p_i. Sumaryczny rozmiar pamięci wirtualnej wszystkich procesów jest zatem równy $S = \sum_i s_i$ Liczba ramek przydzielona programowi p_i jest określona wzorem $a_i \approx \frac{s_i}{S} \cdot m$. Ta wartość musi zostać zaokrąglona do najbliższej wartości całkowitej, większej od minimalnej liczby ramek, jaką należy procesowi przydzielić. Metody wywodzące się z przydziału proporcjonalnego oprócz rozmiaru procesu mogą uwzględniać też inne jego właściwości, np. priorytet.

Szamotanie - definicja i przyczyna

W systemie, w którym procesom są przyporządkowane priorytety można pozwolić, aby proces o wysokim priorytecie odbierał w razie potrzeby ramki przydzielone procesom o niższym priorytecie. Przy takim rozwiązaniu, jeśli liczba ramek, którą dysponuje proces niskopriorytetowy spadnie poniżej określonej wartości, to proces ten zaczyna intensywnie wymieniać swoje strony. Jeśli proces zużywa więcej czasu na wymianę stron niż na wykonanie to mówimy o zjawisku *szamotania* (ang. trashing). Szczególnie intensywnie to zjawisko przebiegało w pierwszych systemach, które stosowały równocześnie planowanie długoterminowe i stronicowanie na żądanie. Szamotanie choć jednego z procesów powoduje zwiększenie obciążenia urządzenia wymiany, a zmniejszenie obciążenia procesora. Ponieważ zadaniem planisty długoterminowego jest dbanie o to, aby procesor był maksymalnie wykorzystany, to żeby zwiększyć jego obciążanie wprowadzał on do pamięci nowy proces. Ten nowy proces otrzymywał do dyspozycji ramki, które zabierane były procesom już rezydującym w pamięci. To z kolei powodowało zwiększenie intensywności szamotania procesów, które obejmowało do tej pory to zjawisko i doprowadzenie do niego innych procesów. W efekcie planista długoterminowy uzyskiwał efekt odwrotny do zamierzonego: małe obciążenie procesora, a duże jednostki stronicującej.

Zbiór roboczy

Aby wyeliminować szamotanie należy zadbać o to, by proces zawsze dysponował wystarczającą liczbą ramek, aby pomieścić jednocześnie w pamięci wszystkie strony niezbędne w danej chwili do jego działania. Wykonanie procesu podlega *modelowi strefowemu*. *Strefą* nazywamy zbiór stron, które pozostają we wspólnym użyciu (muszą razem być obecne w pamięci operacyjnej). Taką strefę mogą tworzyć np. strony na których umieszczony jest jakiś podprogram lub jakaś struktura danych. Program składa się wielu stref, niekoniecznie rozłącznych. Podczas wykonania procesu sterowanie przechodzi od bieżącej strefy do następnej. Model zbioru roboczego jest ściśle powiązany z modelem stref. Przyjmuje się pewien parametr Δ, który określa szerokość *okna zbioru roboczego*, czyli liczbę ostatnich odniesień procesu do stron, które powinien system operacyjny zapamiętać. *Zbiorem roboczym* (ang. working set) nazywamy wszystkie strony do których nastąpiło ostatnich Δ odniesień. Należy zadbać o to by każdy proces miał tyle ramek, aby mógł utrzymać w pamięci swój zbiór roboczy. Problemem jest dobranie takiego parametru Δ aby model zbioru roboczego działał. Do szamotania nie dojdzie, jeśli rozmiar sumy zbiorów roboczych WSS_i wszystkich procesów będzie zawsze mniejszy od liczby dostępnych ramek D, co można zapisać nierównością $D > \sum_i WSS_i$.

Częstotliwość błędów strony

Prostszym sposobem niedopuszczania do zaistnienia szamotania jest monitorowanie częstotliwości występowania błędów stron we wszystkich procesach. Zakładamy przy tym pewną maksymalną i minimalną wartość graniczną. Jeśli któryś z procesów wykazuje za mało błędów stron, to część ramek jest mu odbierana i przekazywana procesowi, który wykazuje zbyt dużą liczbę błędów stron.

Przydział lokalny i globalny

Istnieją dwie strategie przeprowadzania wymiany strony dla pojedynczego procesu wykonywanego w systemie wielozadaniowym. Pierwsza, nazywana *wymianą globalną*, polega na objęciu algorytmem wymiany wszystkich ramek w pamięci fizycznej. Oznacza to, że proces (a właściwie system operacyjny w imieniu procesu) może odebrać ramkę innemu procesowi i umieścić w niej swoją stronę. Druga strategia jest nazywana *wymianą lokalną*. W tym przypadku system operacyjny wymienia strony należące jedynie do procesu, który wykazał błąd strony. Dodatkowo, liczba ramek przydzielonych procesowi nie ulega zmianie, jeśli w systemie nie ma wolnych ramek. Wymiana lokalna ogranicza nasilanie zjawiska szamotania, ale z kolei wymiana globalna daje lepszą przepustowość systemu i pozwala dostosować liczbę ramek procesu do jego zbioru roboczego.

Stronicowanie wstępne

Jeśli proces ulega szamotaniu, to może być całkowicie wycofany z pamięci operacyjnej do przestrzeni wymiany, do momentu aż w pamięci głównej pojawi się odpowiednia liczba ramek pozwalająca na jego prawidłowe wykonanie. Aby program nie był sprowadzany do pamięci strona po stronie, tak jak ma to miejsce na początku jego wykonywania, to można zastosować *stronicowanie wstępne*, czyli wprowadzić do pamięci wszystkie jego strony, które zostały wycofane. Skuteczność tej techniki zależy od tego ile z tych stron będzie przydatnych podczas dalszej pracy procesu. Jeśli zbyt mało, to nie opłaca się tej techniki stosować.

Rozmiar strony

Chociaż w większości platform sprzętowych wpływ programistów systemowych na wielkość strony jest ograniczony do wyboru między kilkoma opcjami zaoferowanymi przez twórcę procesora (najczęściej są one dwie), to warto rozważyć zalety stosowania małych i dużych stron. Za stosowaniem małych stron przemawiają następujące czynniki:

- mniejsza fragmentacja wewnętrzna,
- mniejszy czas przesyłania strony z pamięci pomocniczej do operacyjnej lub odwrotnie,
- lepsza *rozdzielczość* tzn. stosunek ilości informacji potrzebnej w danej chwili do ilości informacji niepotrzebnej, zawartych w obrębie strony.

Za stosowaniem stron dużych przemawiają z kolei:

- mniejsza liczba operacji wejścia-wyjścia wykonywanych podczas działania procesu,
- mniejszy rozmiar tablicy stron,
- mniejsza liczba błędów stron.

Obecnie stosuje się najczęściej duże strony. Procesory firmy Intel i pokrewne (32-bitowe i 64-bitowe) pozwalaja na wybór między 4KiB, a 4MiB. Procesory DEC Alpha (64--bitowe) stosują strony o wielkości 8KiB. Procesor Motorola 68030 (32-bitowy) pozwala programiście systemowemu na wybór rozmiaru strony w zakresie od 256 B do 32 KiB.

Rozmiar strony

Chociaż w większości platform sprzętowych wpływ programistów systemowych na wielkość strony jest ograniczony do wyboru między kilkoma opcjami zaoferowanymi przez twórcę procesora (najczęściej są one dwie), to warto rozważyć zalety stosowania małych i dużych stron. Za stosowaniem małych stron przemawiają następujące czynniki:

- mniejsza fragmentacja wewnętrzna,
- mniejszy czas przesyłania strony z pamięci pomocniczej do operacyjnej lub odwrotnie,
- lepsza *rozdzielczość* tzn. stosunek ilości informacji potrzebnej w danej chwili do ilości informacji niepotrzebnej, zawartych w obrębie strony.

Za stosowaniem stron dużych przemawiają z kolei:

- mniejsza liczba operacji wejścia-wyjścia wykonywanych podczas działania procesu,
- mniejszy rozmiar tablicy stron,
- mniejsza liczba błędów stron.

Obecnie stosuje się najczęściej duże strony. Procesory firmy Intel i pokrewne (32-bitowe i 64-bitowe) pozwalaja na wybór między 4KiB, a 4MiB. Procesory DEC Alpha (64--bitowe) stosują strony o wielkości 8KiB. Procesor Motorola 68030 (32-bitowy) pozwala programiście systemowemu na wybór rozmiaru strony w zakresie od 256 B do 32 KiB.

Rozmiar strony

Chociaż w większości platform sprzętowych wpływ programistów systemowych na wielkość strony jest ograniczony do wyboru między kilkoma opcjami zaoferowanymi przez twórcę procesora (najczęściej są one dwie), to warto rozważyć zalety stosowania małych i dużych stron. Za stosowaniem małych stron przemawiają następujące czynniki:

- mniejsza fragmentacja wewnętrzna,
- mniejszy czas przesyłania strony z pamięci pomocniczej do operacyjnej lub odwrotnie,
- lepsza *rozdzielczość* tzn. stosunek ilości informacji potrzebnej w danej chwili do ilości informacji niepotrzebnej, zawartych w obrębie strony.

Za stosowaniem stron dużych przemawiają z kolei:

- mniejsza liczba operacji wejścia-wyjścia wykonywanych podczas działania procesu,
- mniejszy rozmiar tablicy stron,
- mniejsza liczba błędów stron.

Obecnie stosuje się najczęściej duże strony. Procesory firmy Intel i pokrewne (32-bitowe i 64-bitowe) pozwalaja na wybór między 4KiB, a 4MiB. Procesory DEC Alpha (64--bitowe) stosują strony o wielkości 8KiB. Procesor Motorola 68030 (32-bitowy) pozwala programiście systemowemu na wybór rozmiaru strony w zakresie od 256 B do 32 KiB.

Rozmiar strony

Chociaż w większości platform sprzętowych wpływ programistów systemowych na wielkość strony jest ograniczony do wyboru między kilkoma opcjami zaoferowanymi przez twórcę procesora (najczęściej są one dwie), to warto rozważyć zalety stosowania małych i dużych stron. Za stosowaniem małych stron przemawiają następujące czynniki:

- mniejsza fragmentacja wewnętrzna,
- mniejszy czas przesyłania strony z pamięci pomocniczej do operacyjnej lub odwrotnie,
- lepsza *rozdzielczość* tzn. stosunek ilości informacji potrzebnej w danej chwili do ilości informacji niepotrzebnej, zawartych w obrębie strony.

Za stosowaniem stron dużych przemawiają z kolei:

- mniejsza liczba operacji wejścia-wyjścia wykonywanych podczas działania procesu,
- mniejszy rozmiar tablicy stron,
- mniejsza liczba błędów stron.

Obecnie stosuje się najczęściej duże strony. Procesory firmy Intel i pokrewne (32-bitowe i 64-bitowe) pozwalaja na wybór między 4KiB, a 4MiB. Procesory DEC Alpha (64--bitowe) stosują strony o wielkości 8KiB. Procesor Motorola 68030 (32-bitowy) pozwala programiście systemowemu na wybór rozmiaru strony w zakresie od 256 B do 32 KiB.

Rozmiar strony

Chociaż w większości platform sprzętowych wpływ programistów systemowych na wielkość strony jest ograniczony do wyboru między kilkoma opcjami zaoferowanymi przez twórcę procesora (najczęściej są one dwie), to warto rozważyć zalety stosowania małych i dużych stron. Za stosowaniem małych stron przemawiają następujące czynniki:

- mniejsza fragmentacja wewnętrzna,
- mniejszy czas przesyłania strony z pamięci pomocniczej do operacyjnej lub odwrotnie,
- lepsza *rozdzielczość* tzn. stosunek ilości informacji potrzebnej w danej chwili do ilości informacji niepotrzebnej, zawartych w obrębie strony.

Za stosowaniem stron dużych przemawiają z kolei:

- mniejsza liczba operacji wejścia-wyjścia wykonywanych podczas działania procesu,
- mniejszy rozmiar tablicy stron,
- mniejsza liczba błędów stron.

Obecnie stosuje się najczęściej duże strony. Procesory firmy Intel i pokrewne (32-bitowe i 64-bitowe) pozwalaja na wybór między 4KiB, a 4MiB. Procesory DEC Alpha (64--bitowe) stosują strony o wielkości 8KiB. Procesor Motorola 68030 (32-bitowy) pozwala programiście systemowemu na wybór rozmiaru strony w zakresie od 256 B do 32 KiB.

Rozmiar strony

Chociaż w większości platform sprzętowych wpływ programistów systemowych na wielkość strony jest ograniczony do wyboru między kilkoma opcjami zaoferowanymi przez twórcę procesora (najczęściej są one dwie), to warto rozważyć zalety stosowania małych i dużych stron. Za stosowaniem małych stron przemawiają następujące czynniki:

- mniejsza fragmentacja wewnętrzna,
- mniejszy czas przesyłania strony z pamięci pomocniczej do operacyjnej lub odwrotnie,
- lepsza *rozdzielczość* tzn. stosunek ilości informacji potrzebnej w danej chwili do ilości informacji niepotrzebnej, zawartych w obrębie strony.

Za stosowaniem stron dużych przemawiają z kolei:

- mniejsza liczba operacji wejścia-wyjścia wykonywanych podczas działania procesu,
- mniejszy rozmiar tablicy stron,
- mniejsza liczba błędów stron.

Obecnie stosuje się najczęściej duże strony. Procesory firmy Intel i pokrewne (32-bitowe i 64-bitowe) pozwalaja na wybór między 4KiB, a 4MiB. Procesory DEC Alpha (64--bitowe) stosują strony o wielkości 8KiB. Procesor Motorola 68030 (32-bitowy) pozwala programiście systemowemu na wybór rozmiaru strony w zakresie od 256 B do 32 KiB.

Rozmiar strony

Chociaż w większości platform sprzętowych wpływ programistów systemowych na wielkość strony jest ograniczony do wyboru między kilkoma opcjami zaoferowanymi przez twórcę procesora (najczęściej są one dwie), to warto rozważyć zalety stosowania małych i dużych stron. Za stosowaniem małych stron przemawiają następujące czynniki:

- mniejsza fragmentacja wewnętrzna,
- mniejszy czas przesyłania strony z pamięci pomocniczej do operacyjnej lub odwrotnie,
- lepsza *rozdzielczość* tzn. stosunek ilości informacji potrzebnej w danej chwili do ilości informacji niepotrzebnej, zawartych w obrębie strony.

Za stosowaniem stron dużych przemawiają z kolei:

- mniejsza liczba operacji wejścia-wyjścia wykonywanych podczas działania procesu,
- mniejszy rozmiar tablicy stron,
- mniejsza liczba błędów stron.

Obecnie stosuje się najczęściej duże strony. Procesory firmy Intel i pokrewne (32-bitowe i 64-bitowe) pozwalaja na wybór między 4KiB, a 4MiB. Procesory DEC Alpha (64--bitowe) stosują strony o wielkości 8KiB. Procesor Motorola 68030 (32-bitowy) pozwala programiście systemowemu na wybór rozmiaru strony w zakresie od 256 B do 32 KiB.

Wpływ stronicowania na żądanie na wykonanie procesu

Choć stronicowanie na żądanie jest przezroczyste dla programisty piszącego aplikacje dla użytkownika, to dokonany przez niego dobór struktur danych oraz sposobu odwoływania do nich może mieć wpływ na częstotliwość błędów stron generowanych przez jego program. Zalecane jest stosowanie dużej liczby struktur odznaczających się dobrą lokalnością, a małej struktur odznaczających się złą lokalnością. Do pierwszej kategorii zalicza się między innymi stos, do drugiej tablica z adresowaniem mieszającym (ang. hash table). Również etapy kompilacji i ładowania mogą mieć znaczenie dla częstości błędów stron generowanych przez proces. Można ją zmniejszyć, jeśli kompilator będzie oddzielał kod od danych, a program ładujący wyrównywał rozmieszczenie spójnych elementów programu (jak podprogramy i duże struktury danych) do granicy stron. Stronicowanie na żądanie ma wpływ na efektywność aplikacji wieloprocesorowych (wielordzeniowych).

Blokowanie stron

Jeśli system operacyjnych pozwala, żeby proces użytkownika umieszczał bufory dla operacji wejścia-wyjścia w obrębie swojej przestrzeni adresowej, to musi także zapewnić, że strona je zawierająca pozostanie w ramce do czasu zrealizowania tej operacji i że ramka ta nie zostanie przydzielona innemu procesowi. Działanie to wymaga wsparcia sprzętowego, w postaci odpowiedniego mechanizmu kontrolującego *bit blokady* (ang. lock) w tablicy stron. Jeśli jest on ustawiony dla którejś ze stron, to oznacza, że tej strony nie można wymienić, nawet jeśli jest idealną kandydatką do wymiany i że musi ona pozostać w ramce. Ramka ta nie może także zmienić właściciela. Blokowanie stron może również być użyte do ograniczenia szamotania procesów o niskim priorytecie. Zablokowanie stron takiego procesu uniemożliwia odebranie ich przez proces o wysokim priorytecie.

Implementacje tablicy stron

Ponieważ tablice stron mogą być bardzo dużych rozmiarów, to stosowane są różne ich implementacje, celem zmniejszenia ich wielości. Jedną z nich jest *odwrócona tablica stron* (ang. inverted page table). W takiej tablicy indeksami są fizyczne adresy stron (adresy bazowe ramek), a wartościami elementów numery stron i identyfikatory procesów do których one należą. Ponieważ pamięć fizyczna jest zazwyczaj mniejsza od pamięci wirtualnej, to tym samym rozmiar odwróconej tablicy stron jest mniejszy od „zwykłej" tablicy stron. Aby przyspieszyć wyszukiwanie informacji w tej tablicy stosowane jest haszowanie, zazwyczaj realizowane sprzętowo. Wadą tego rozwiązania jest niemożność szybkiego określenia, czy błąd strony spowodowany jest jej brakiem w pamięci operacyjnej, czy też tym, że ta strona nie istnieje. Inna wada, to brak lokalności odwołań, co może stanowić problem dla rejestrów TLB. Odwrócone tablice stron stosowane są w 64-bitowych platformach sprzętowych. Innym popularnym rozwiązaniem jest *wielopoziomowa tablica stron* (ang. multilevel page table). Polega ono na podziale tablicy na np.: trzy części nazywane *katalogiem głównym*, *katalogiem pośrednim* i tablicą stron. Pozycje w katalogu głównym wskazują na pozycje w katalogu pośrednim, a te na pozycje w tablicy stron. Rozwiązanie to zmniejsza ilość pamięci operacyjnej wymaganej do przechowania tablicy stron (niepotrzebne elementy mogą być trzymane w przestrzeni wymiany) oraz pozwala na lepszą współpracę tablicy stron z buforami TLB. System operacyjny utrzymuje osobne tablice zawierające numery bloków pamięci pomocniczej, w których umieszczone są wymienione strony procesu. Korzysta z nich procedura obsługi błędu strony.

Segmentacja na żądanie

Segmentacja na żądanie jest alternatywnym w stosunku do stronicowania na żądanie sposobem realizacji koncepcji pamięci wirtualnej. Nie jest ona tak wydajna jak stronicowanie, ale również nie wymaga takiego jak ono wsparcia sprzętowego. Stosował ją system OS/2. Tablica segmentów nazywana tutaj *tablicą deskryptorów segmentów* zawiera *deskryptory segmentów*, w których są umieszczone wszelkie informacje związane z poszczególnymi segmentami, takie jak ich wielkość, tryb ochrony i umiejscowienie. W każdym z nich umieszczony jest również *bit poprawności* określający, czy dany segment znajduje się w pamięci operacyjnej, czy też w przestrzeni wymiany. Jeśli proces odwołuje się do segmentu, którego nie ma w pamięci, to generowany jest *błąd segmentu* i w wyniku jego obsługi żądany segment jest sprowadzany do pamięci. Do określenia, który segment ma być wymieniony, w przypadku takiej konieczności służy zawarty w deskryptorze *bit udostępnienia*. Istnieją również wywołania systemowe, które pozwalają procesowi użytkownika wskazać systemowi, które z jego segmentów można usunąć z pamięci, a które powinny w niej pozostać.

Segmentacja stronicowana na żądanie

Obie metody realizacji pamięci wirtualnej można połączyć w jeden schemat nazywany *segmentacją stronicowaną na żądanie*. Ten schemat zarządzania pamięcią stosuje system OS/2 Warp. Również system Linux korzysta na platformach sprzętowych udostępniających segmentację z segmentów, choć podstawę jego systemu zarządzania pamięcią dla wszystkich platform stanowi stronicowanie na żądanie. Segmentacja jest wykorzystywana przez niego w platformach x86, jako uzupełnienie mechanizmu ochrony stron. Korzysta on z segmentów kodu i danych zarówno dla jądra, jak i procesów użytkownika, z tym, że obejmują one zasięgiem całą pamięć wirtualną, a wymianie podlegają strony wewnątrz segmentów procesu użytkownika. W bardzo ograniczonym zakresie wykorzystywany jest również segment TSS (ang. Task State Segment). W segmentacji stronicowanej na żądanie mogą być jednocześnie używanie strony o różnych rozmiarach.

Pytania

?

Koniec

Dziękuję Państwu za uwagę!