Mr Ratimir Kvaternik
Fakultet organizacije i informatike
V a r a ž d i n

UDK 681.142.2
Prethodno saopćenje

O D R E D J I V A N J E R A D N O G S K U P A S T R A N I C A U O P E R A T I V N I M S I S T E M I M A S A L O K A C I J O M M E M O R I J E P O M O Ć U "S T R A N I C A N A Z A H T J E V"

U članku se nastoji na jednostavan i sažet način bez ulaženja u egzaktne i formalizirane dokaze postići slijedeće:

1. *pokazati kako se odvija alokacija memorije pomoću stranica na zahtjev,*
2. *objasniti i definirati što je radni skup stranica,*
3. *sugerirati kao ideju, a ne dokaz ili razradjeni primjer, kako bi se mogao radni skup stranica odrediti u stvarnoj situaciji.*

Na kraju izvlači se zaključak da teorija radnog skupa povezuje alokaciju memorije i alokaciju procesora i zbog toga predstavlja globalnu strategiju i korak naprijed u teoretskom tretiranju kompjuterskog sistema, odnosno situacija koje se u sistemu javljaju u vezi s alokacijom njegovih resursa procesu. Kao drugi bitni zaključak može se spomenuti još uvijek nenadoknadivost realne centralne memorije s virtuelnom bez obzira na uspješnost realizacije modela radnog skupa stranica.

1. UVOD

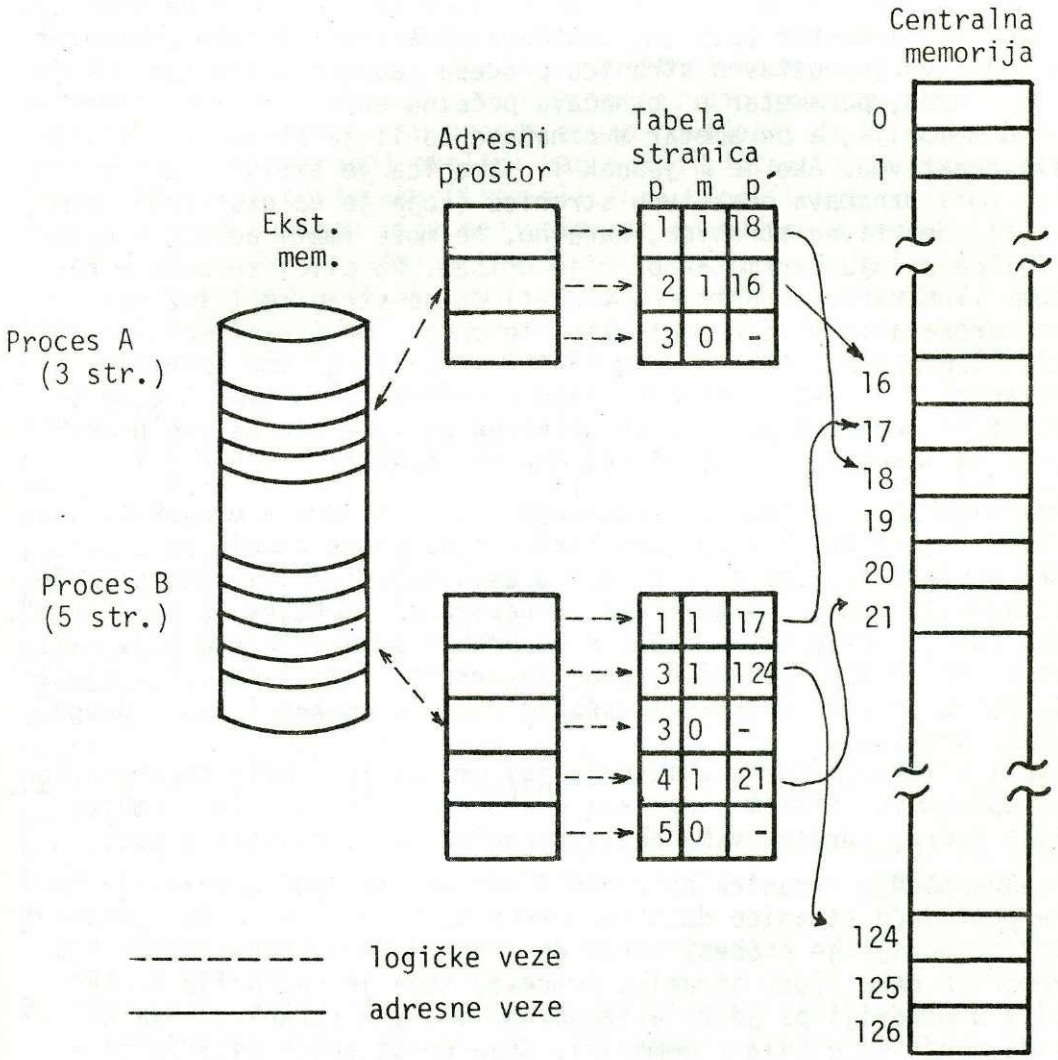
Centralna memorija je pored centralnog procesora osnovni i najvažniji resurs kompjuterskog sistema o čijem korištenju treba voditi računa. U multiprogramiranoj okolini ona vrlo često postaje usko grlo sistemskih resursa pa je posebno u takvim sistemima potreban što je moguće djelotvorniji način dodjele ili alokacije memorije pojedinim procesima. Pod procesom možemo smatrati program u stanju izvodjenja, a pod alokacijom memorije podrazumijevamo niz hardwarskih dijelova i razvijenih softwarskih tehnika unutar operativnog sistema koje se brinu i provode smještaj procesa u memoriju. Analogno tome pod alokacijom procesora smatramo niz softwarskih tehnika ugradjenih u operativni sistem koje provode i odlučuju o dodjeli procesora svakom pojedinom procesu. Alokacija memorije i alokacija procesora medjusobno su povezani jer se procesor ne može dodijeliti procesu kojem nije prethodno dodijeljena memorija. Sve do pojave sistema s podjelom memorije na "stranice" ova je povezanost predstavljala ozbiljnu prepreku za uspješno korištenje kompjuterskog sistema. Nakon prvog sistema na stranicama (kompjuter ATLAS 1960. konstruiran na univerzitetu u

Manchasteru u Engleskoj) pojavilo se nekoliko različitih sistema koji su koristili stranice. Jedan od najuspješnijih sistema sa stranicama je takozvani sistem alokacije memorije pomoću "stranica na zahtjev". Ovim načinom alokacije memorije ostvarena je vrlo djelotvorna virtuelna memorija a time i uspješno korištenje čitavog kompjuterskog sistema u cjelini.

2. ALOKACIJA MEMORIJE POMOĆU "STRANICA NA ZAHTJEV"

Ova se metoda alokacije temelji na slijedećoj jednostavnoj ideji: proces u manjem vremenskom intervalu koristi samo odredjeni broj stranica, a ne sve stranice, pa bi trebalo biti dovoljno za izvo djenje procesa da se samo te stranice nalaze u memoriji, a ostale se mogu nalaziti na eksternoj memoriji i po potrebi bi se on da pozivale u centralnu memoriju. Te stranice memorije nazivamo aktivnim stranicama; one se "upravo izvode", odnosno one sadrže dio procesa koji se upravo izvodi. Stranice s eksterne memorije unose se u memoriju tek onda kada su stvarno potrebne, njih se na odredjeni način zahtijeva ili traži, pa se zbog toga čitav sistem ovakvog "straničenja" naziva alokacija memorije pomoću stranica na zahtjev ili "straničenje na zahtjev" (demand paging system). Generirani zahtjev za unošenje stranice procesa s eksterne memorije u centralnu ima za posljedicu brisanje, odnosno, kako se to često naziva, "izbacivanje" stranice koju više ne trebamo i smještaj zahtijevane stranice na njezino mjesto. Medjutim, budući da u memoriji ima više aktivnih stranica procesa a u momentu zahtjeva ne koristi se više ni jedna od njih jer se izvodjenje prebacuje upravo u zahtijevanu stranicu, postavlja se pitanje koju od tih aktivnih stranica valja izbaciti i na temelju čega zaključiti koja od tih aktivnih stranica neće više biti potrebna. Postoji niz strategija i na bazi tih strategija razvijenih algoritama koji odredjuju koju stranicu treba izbaciti.

Za provodjenje čitavog tog sistema potrebno je čuvati u memoriji neke osnovne informacije o čitavom procesu, odnosno o svim stranicama. To se provodi pomoću tzv. "tabele stranica" Page Map Table) kao što je to ilustrirano na slici 1.


Centralna
memorija
Adresni
prostor
Tabela
stranica
p m p'
Ekst.
mem.
Proces A
(3 str.)
Proces B
(5 str.)
1 1 18
2 1 16
3 0 -
1 1 17
3 1 124
3 0 -
4 1 21
5 0 -
0
1
16
17
18
19
20
21
124
125
126
logičke veze
adresne veze

Slika 1.

Na eksternoj memoriji nalaze se sve stranice procesa A, dakle čitav adresni prostor procesa A. Isto je prikazano i za proces B. Proces A ima ukupno tri stranice, a proces B ukupno pet stranica. Stranica 3 procesa A je neaktivna, dok su stranice 2 i 1 aktivne stranice. U tabeli stranica vode se sve stranice procesa. Pojedini parametar (p,p',m) označava stranicu, pa tako parametar p označava jednostavno stranicu procesa (adresnog prostora) koja je po redu, parametar p' označava početnu adresu dotične stranice u memoriji, a parametar m označava da li je stranica aktivna ili neaktivna. Ako je m jednak 1, stranica je aktivna, dok m jednak nuli označava neaktivnu stranicu (koja je na eksternoj memoriji). Neaktivna stranica, naravno, ne može imati adrese u memoriji pa za nju parametar p' nije upisan. Na slici se može u našem ilustrativnom primjeru vidjeti da se stranice 1 i 2 nalaze na adresama 18 i 16 u centralnoj memoriji. Drugim riječima,stranica 1 procesa A nalazi se u 18-toj stranici glavne memorije, a stranica 2 u 16-toj stranici glavne memorije. Adekvatno tome vidi se za proces B da se aktivne stranice 1, 2 i 4 nalaze u memoriji na adresama 17, 124 i 21, dok su stranice 3 i 5 neaktivne.

Adresiranje, odnosno izračunavanje stvarnih adresa u ovakvom sistemu izvodi posebni adresni hardware uz pomoć tabele stranica. Ako se tražena adresa ne nalazi u aktivnoj stranici, što se može ustanoviti pomoću parametra m, generira se "zahtjev za stranicom". Taj zahtjev nije ništa drugo nego prekid posebne vrste koji nazivamo prekid zbog stranice (page interrupt) i koji se dalje rješava posebnim hardwarsko-softwarskim mehanizmom kao i svaka druga vrsta prekida.
Svaki proces (program u izvodjenju) ima svoju tabelu stranica koju operativni sistem pronalazi preko centralne tabele stranica koja sadrži adrese svih tabela stranica svih aktivnih procesa.

Svako unošenje stranice na zahtjev ima za posljedicu brisanje neke postojeće stranice dotičnog procesa. Kako se time ne rješava trajno odvijanje proces, ubrzo će trebati ponovno napuniti u memoriju neku drugu stranicu procesa, koja je već prije mogla biti u memoriji pa je bila izbačena, ili pak stranicu koja do tada uopće nije bila u memoriji. Koja će stranica biti "zahtijevana", to ovisi o ponašanju procesa, dakle programa u svakom pojedinom slučaju. Takodjer je moguće zamisliti da će neki proces izmijeniti sadržaj stranice koju je koristio, što znači da će upisati nešto u tu stranicu, pa će prilikom izbacivanja takve stranice biti potrebno stvarno fizički prenijeti stranicu na eksternu memoriju na njezino mjesto i tako "prebrisati" staru

stranicu koja se na eksternoj memoriji nalazi. Ovo se dogadja doduše rjedje i taj problem neće ovdje biti razmatran. U svakom,medjutim, slučaju dolazi do intenzivnog prenošenja stranica s eksterne memorije u centralnu a ponekad i obratno. Takvo prenošenje stranica naziva se promet stranicama (page traffic).

Očigledno ovakvim načinom rada postižu se kod izvodjenja svakog pojedinog procesa znatne uštede u korištenju memorijskog prostora centralne memorije,jer se samo manji dio procesa mora smjestiti u memoriju da bi se proces mogao odvijati. To je vrlo važno u multiprogramiranoj okolini kao i kod sistema koji rade na bazi vremenskih odsječaka (time-slicing). Korištenje resursa u kompjuterskom sistemu nije medjutim izolirano, nego medjusobno povezano tako da uspješnije korištenje memorije, koje se očituje u smještaju mnogo više aktivnih procesa u memoriju, imati će za posljedicu pojačano korištenje centralnog procesora, a ovo će konačno rezultirati u uspješnijem korištenju kanalskih procesora, odnosno ulazno-izlaznih jedinica. Prema tome, uspješnost korištenja čitavog kompjuterskog sistema znatno se povećava. Ovdje se pod uspješnošću korištenja kompjuterskog sistema, kao što se može zaključiti, smatra dužina korištenja sistema unutar nekog vremenskog perioda, odnosno preciznije gledano, koliko dugo se neki resurs koristi u nekom promatranom vremenskom periodu. To je ispravno gledajući s internog stanovišta kompjuterskog sistema,ali ne mora biti ispravno sa stanovišta vanjskog korisnika koji uspješnost korištenja kompjuterskog sistema mora kompleksnije definirati.Medjutim, u svakom slučaju ovako definirana uspješnost upravno će biti proporcionalna s uspješnošću definiranom od strane korisnika jer će veće iskorištenje resursa u istom vremenskom periodu nužno imati za posljedicu i više izvršenog rada za korisnika.

U ovakvom simplificiranom prikazu možemo već uočiti nekoliko problema koji se kod realizacije moraju riješiti. Ti se problemi mogu izraziti slijedećim pitanjima:

1. Kako ćemo znati gdje se na eksternoj memoriji nalazi stranica koju treba ubaciti u memoriju?
2. Kako ćemo znati kuda u memoriju valja smjestiti "zahtijevanu" stranicu, odnosno koju stranicu u memoriji valja prekriti(izbrisati)?
3. Kako ćemo znati da stranicu koju treba izbrisati (prekriti)ni je prethodno trebalo prebaciti natrag na eksternu memoriju jer je u stranicu bilo upisivano nešto tokom izvodjenja procesa?

Takvih problema možemo uočiti i više. Standardno rješenje može biti za pitanje 1 i 3 jednostavno u proširenju tabele stranica s dodatnim upisima koji će sadržavati potrebnu informaciju za

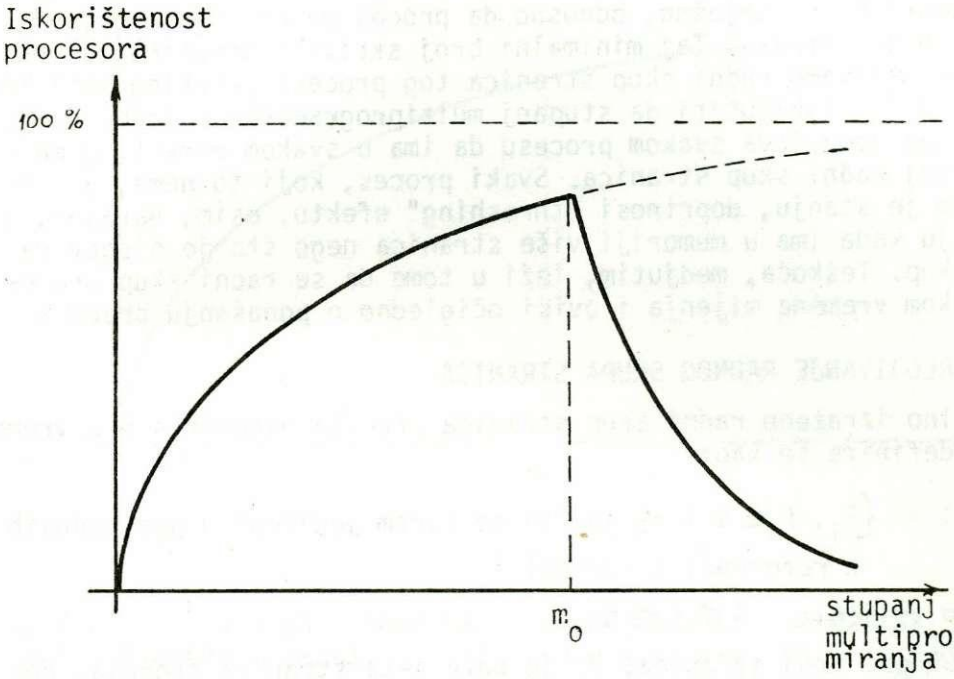
rješenje problema, a to su adresa stranice na eksternoj memoriji ili pak pokazivač (pointer) adrese ako se radi o problemu 1. Problem 2 je znatno složeniji. Treba naime odlučiti na temelju podataka ili mjerenja o ponašanju procesa koju stranicu valja izbaciti. Algoritam bi u idealnom slučaju morao pronaći stranicu koju neće proces više uopće trebati, odnosno koju neće najduže trebati. Takav algoritam bio bi optimalan, medjutim, to bi značilo poznavanje ponašanja procesa u budućnosti. Budući da na osnovu ponašanja procesa u prošlosti u općenitom slučaju nije moguće predvidjeti ponašanje u budućnosti, ne može se takav algoritam ni realizirati, nego se odredjeno ponašanje procesa unaprijed pretpostavlja i na bazi toga konstruira algoritam. Naravno, za neke procese takav će algoritam biti zadovoljavajući, dok će za druge prouzrokovati izbacivanje stranica koje ćemo ubrzo zatim morati ponovno unositi u memoriju.

Postoji niz algoritama, a svaki se bazira na odredjenom ponašanju, odnosno pretpostavlja unaprijed odredjeno ponašanje procesa i na temelju toga pronalazi stranicu koja će se izbaciti.Uspješnost dakle cijelog sistema alokacije memorije pomoću stranica "na zahtjev" ovisit će, izmedju ostalog, i o ponašanju samog procesa. Naime, samo prenošenje stranica može u nepovoljnom slučaju jako narasti i sistem može biti znatnim dijelom zaposlen na tom prometu stranicama. Takav nekoristan posao, koji možemo nazvati i izgubljeni rad na interne potrebe sistema, naziva se"overhead". Svaki operativni sistem nužno generira "overhead", i to ne samo na poslovima upravljanja memorijom nego i na raznim drugim postupcima i procesima koji se odvijaju interno u kompjuterskom sistemu, a za korisnika su nevidljivi. Neke od ovih operacija su softwarske, kao što je na primjer prenošenje i odlučivanje koju stranicu valja prenijeti, dok su druge hardwarske, kao što je na primjer izračunavanje adresa, što znači da postoje posebni hardwarski dijelovi koji služe isključivo za tu svrhu.

3. POJAM RADNOG SKUPA STRANICA

Kod "straničenja na zahtjev" moguće je, kao što je navedeno i objašnjeno, postići znatno uspješnije korištenje kompjuterskog sistema nego kod drugih načina alokacije memorije. Na prvi pogled moglo bi se čak zaključiti da "straničenje na zahtjev" može dovesti korištenje centralnog procesora do teoretskog maksimuma, tj. stopostotnog iskorištenja ukoliko samo smjestimo u memoriju dovoljno velik broj procesa tako da procesor može u svakom momentu pronaći proces u stanju spremnom za izvodjenje (READY stanje).

Praktično eksperimentiranje pokazuje medjutim sasvim suprotan efekt. Naime, raste li stupanj multiprogramiranja, gdje pod stupnjem multiprogramiranja smatramo broj aktivnih procesa u centralnoj memoriji, rasti će i iskorištenje centralnog procesora kao što se to vidi na slici 2.



Slika 2 .

Medjutim, neočekivano stupanj iskorištenja procesora dostiže maksimum kod odredjenog stupnja multiprogramiranja m_o, a nakon toga porastom stupnja multiprogramiranja iskorištenje procesora opada. Efekt je naizgled začudjujući, a poznat je pod nazivom "Thrashing", što bi se u slobodnom prijevodu moglo nazvati "prazni rad sistema". Efekt se može objasniti na temelju medjuzavisnosti rada centralnog procesora i memorije koje je Denning (2) uočio.

Na temelju toga objašnjenje za "thrashing" efekt je slijedeće: Nakon prekoračenja kritičnog stupnja multiprogramiranja m_o procesi dobivaju sve manji broj stranica jer ih ima sve više, a centralna memorija je ograničena, zbog toga se generira sve više prekida zbog stranica jer procesi traže učestalo unošenje novih stranica.

To traženje novih stranica zapošljava kanale u tolikoj mjeri da oni postaju zasićeni prijenosom, tj. ne uspijevaju poslužiti toliko zahtjeva i većina procesa biva blokirana te ne može nastaviti s izvodjenjem pa se zbog toga korištenje procesora znatno smanjuje. Denning je zaključio na temelju toga da svaki proces mora posjedovati minimalni broj stranica u memoriji kako bi korištenje procesora bilo uspješno, odnosno da proces ne doprinosi pojavi "thrashing" efekta. Taj minimalni broj aktivnih stranica za neki proces nazivamo radni skup stranica tog procesa (working set).Može se dalje zaključiti da stupanj multiprogramiranja treba da je takav da omogućava svakom procesu da ima u svakom momentu u memoriji svoj radni skup stranica. Svaki proces, koji to nema, a u aktivnom je stanju, doprinosi "thrashing" efektu, osim, naravno, u slučaju kada ima u memoriji više stranica nego što je njegov radni skup. Teškoća, medjutim, leži u tome da se radni skup procesa tokom vremena mijenja i ovisi očigledno o ponašanju procesa.

4. ODREDJIVANJE RADNOG SKUPA STRANICA

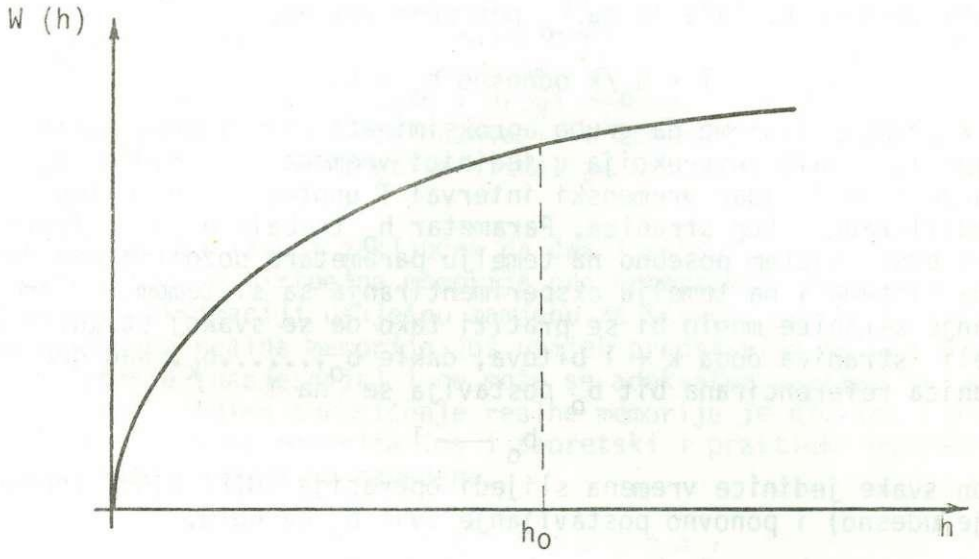
Formalno izraženo radni skup stranica procesa označen s W u vremenu t definira se kao:

$$W(t,h) = \{P_i,\ P_i \in N \text{ i } P_i \text{ javlja se barem jedanput u posljednjih } h \text{ referenci stranica}\}$$

pritom vrijedi: $0 < i \leq N$

N je ukupni broj stranica, P_i je neka i-ta stranica procesa, dok je h parametar koji definira koliko daleko u prošlost gledamo odnosno koliko veliki vremenski interval uzimamo u kojem želimo izdvojiti radni skup stranica procesa. Radni skup stranica ovisi o h kao što je prikazano na slici 3, tj. W(h) je monotono rastuća funkcija.

Funkcija pokazuje da što se dalje u prošlost gleda, to će manji broj dodatnih stranica biti u radnom skupu. Odaberemo li h dovoljno velik ($h=h_0$), dobit ćemo dovoljno velik radni skup da bude dovoljno velika vjerojatnost da ne dodje do prekida zbog stranica. Drugim riječima, veličina radnog skupa ovisi o tome koliko želimo dati vremena procesoru da ne dodje do prekida zbog stranica. Denning i Schwartz (3) formalno matematički su razradili teoriju radnog skupa stranica i dokazali egzaktno ovdje iznijete postavke.
Kako praktički odrediti koje stranice spadaju u radni skup? Postoji sigurno mogućnost da se problem riješi hardwarskim putem,



Slika 3.

koji bi se sastojao u tome da se ugrade posebni hardwarski elementi, mjerači vremena, i to za svaku stranicu, koji bi ujedno detektirali da li je stranica bila referencirana (korištena)ili ne pa bi se na temelju toga moglo odrediti koje stranice u odredjenom vremenskom periodu spadaju u radni skup, jer prema navedenoj definiciji;stranica koja nije u posljednjih h referenci odnosno u odredjenom proteklom vremenu referencirana, ne spada u radni skup. Mogao bi se upotrijebiti i poseban procesor koji bi isključivo služio za odredjivanje radnog skupa stranica.Hardwarsko rješenje bilo jednog ili drugog oblika, iako, mora se napomenuti, to nije eksperimentalno provjereno, moglo bi biti uspješno kod manjih i srednjih sistema te sistema sa sporijim centralnim procesorom jer bi softwarsko rješavanje moglo dodatno opteretiti procesor (overhead) a isto tako i memoriju.Medjutim, ovakvo rješenje predstavlja i odredjeni poremećaj u sistemu i ni je ispitano i provjereno kako bi se takav poremećaj u radu kompjuterskog sistema manifestirao. Nadalje, hardwarsko rješenje sigurno znači i poskupljenje samog kompjuterskog sistema,no vjerojatno je da bi se odnos cijena/performanse popravio.

Kod većih i velikih sistema s većim memorijama i brzim centralnim procesorima vjerojatno je da bi se mogao primijeniti softwarski pristup problemu. Softwarsko bi se rješenje moglo izraditi na temelju slijedeće ideje: Pretpostavimo da je broj referenci u jedinici vremena k. Tada je za h_0 potrebno vrijeme T

$$T = h_0/k \text{ odnosno } h_0 = k:T$$

Ako k poznamo (možemo ga grubo aproksimirati npr. s prosječnim brojem izvedenih instrukcija u jedinici vremena) i odredimo h_0, onda je time i zadan vremenski interval T unutar kojeg želimo odrediti radni skup stranica. Parametar h_0 trebalo bi procijeniti za svaki sistem posebno na temelju parametara poznatih kod dizajna sistema i na temelju eksperimentiranja sa sistemom. Referenciranje stranice moglo bi se pratiti tako da se svakoj stranici u tabeli stranica doda k + 1 bitova, dakle $b_0,.....b_k$. Kad god je stranica referencirana bit b_0 postavlja se na 1

$$b_0 \longrightarrow 1$$

Nakon svake jedinice vremena slijedi operacija SHIFT RIGHT (pomicanje udesno) i ponovno postavljanje svih b_0 na nulu.

$$b_0 \longrightarrow 0,\ b_{i-1} \longrightarrow b_i \quad \text{za } i = 0,1 \ldots k$$

Time se dobiva "bit slika" referenciranja stranica. Nakon takvih operacija, odnosno nakon T vremena, može se izbaciti ona stranica za koju vrijedi

$$b_i = 0 \quad \text{za } i = 0,1 \ldots .k$$

jer takva stranica nije referencirana u vremenu T odnosno posljednjih h_0 referenti. Na taj način mogao bi se odrediti radni skup stranica u konkretnom slučaju, tj. kod svakog kompjuterskog sistema koji ima alokaciju memorije u stranicama "na zahtjev".

5. ZAKLJUČAK

Odredjivanje radnog skupa stranica na opisani način temeljilo bi se u stvari na principu lokalnosti procesa koji kaže da je vjerojatno da će proces unutar kratkog vremenskog intervala imati isti radni skup. Zbog toga je moguće zaključivati na temelju bliske prošlosti o ponašanju u bliskoj budućnosti procesa. Princip lokalnosti ne vrijedi medjutim općenito. Radni skup stranica može nadalje utjecati na strategiju zamjene, punjenja procesa u memoriju i dodjelu procesora te medjusobno povezivanje upravljanja procesorom i upravljanja memorijom. Naime, stranice koje

pripadaju radnom skupu očigledno se ne mogu izbaciti iz memorije. Isto tako proces se ne može "napuniti" u memoriju ako ne stane njegov radni skup u memoriju, a ne može ni ući u stanje izvodjenja (RUN stanje). Tu se dakle vodi računa o svim procesima i povezuje se upravljanje procesorom i upravljanje memorijom. Zbog toga teorija radnog skupa stranica predstavlja u stvari globalnu strategiju alokacije memorije, a ne lokalnu, kao što su to druge strategije. Ipak, postoje i drugi resursi sistema koje ni teorija radnog skupa ne može obuhvatiti, što znači nadalje da ni teorija radnog skupa ne tretira situaciju u kompjuterskom sistemu dovoljno općenito.

Nadalje, treba izvući zaključak da čak i u slučaju uspješne realizacije ovakve virtuelne memorije ona predstavlja samo u vrlo gruboj aproksimaciji uspješnu zamjenu za stvarnu memoriju, drugim riječima realna memorija još uvijek predstavlja daleko bolje rješenje (uspješnije) i ne može se adekvatno zamijeniti s virtuelnom. Jedino ograničenje realne memorije je njezina cijena, dok virtuelna memorija ima i teoretski i praktički ograničenja kao što je to bilo navedeno.

L I T E R A T U R A :

1. *Denning P.J.: "Virtual Memory". ACM Computing Survays, Vol.2, No.3, str.153-159, 1970.*

2. *Denning P.J.: "Thrashing: Its Causes and Prevention", Proc.AFIPS, 1968, Vol.33.*

3. *Denning P.J., Schwartz S.C.: "Properties of the Working Set Model" CACM, Vol.15, No.3, str.191-198, 1972.*

4. *Madnick S.E., Donovan J.J.: Operating Systems McGraw-Hill, New York, 1974.*

Primljeno: 1979-9-29.

Kvaternik R. Operativni sistemi i alokacija memorije

Zbornik radova(1979),2-3

Kvaternik R. Determination of Working Set in Operating Systems with Demand Paging Memory Allocation.

SUMMARY

The article briefly describes demand paging memory allocation and suggests one of a few possible ways for determination of the working set. In the introduction and the second part the principles of the demand paging memory allocation have been outlined. Part 3 explains the effect of thrashing without going too deep into formal definitions. It follows from this explanation that the working set is a certain minimal set of pages that are required by the process to be held in the main memory. If less than this number of pages are present, then the process is continually interrupted by page faults which contribute towards thrashing.

The problem of how to determine which pages belong to the working set at a certain time interval could be solved by adding a string of k bits to each page entry in the Page Map Table. Whenever the page is referenced, bit b_o is set to 1 and at each sampling period all bits are shifted right and b_o is set again to 0. The bits are examined every T seconds, where $T=h_0 k$ and k is an integer constant.

If any of b_i bits (i=0,1...k) equals 1, the page belongs to the working set. If, on the contrary, all b_i bits are equal to 0, then the page does not belong to the working set and could be removed from the main memory.

The working set theory is based on the principle of process locality, which however does not apply for any process. Thus although the working set model allows efficient virtual memory and correlates memory and processor allocation, there is still no substitute for the real memory.