Alen Prodan

Utjecaj upravljanja virtualnom memorijom na performanse Oracle sustava

Agenda

- Uvod u memorijske strukture Oracle RDBMS sustava
- Upravljanje memorijom sustava

  Adresiranje memorije (segmentacija, paging, translacija linearnih u fizičke adrese, adresni prostor procesa i upravljanje page frameovima)
- Utjecaj upravljanja virtualnom memorijom na performanse Oracle sustava

  Case study: utjecaj paging mehanizma na performanse Oracle sustava
- Zaključna riječ
- Pitanja i odgovori

Uvod u memorijske strukture Oracle RDBMS

Uvod

- osnovna namjena RDBMS sustava je omogućiti sigurnu i brzu pohranu i pretraživanje korisničkih podataka
- u idealnom svijetu takav bi sustav bio temeljen na brzim CPU jedinicama i vrlo brzom memorijom male latencije i velikog kapaciteta pohrane
- zbog razloga ekonomske isplativosti potreban je kompromis između cijene koštanja i osnovnih svojstava memorije: latencije pristupa podacima, kapaciteta pohrane podataka i propusnosti
- u današnjim modernim računalima prisutne su dvije glavne vrste memorije: brži i skuplji RAM, te sporiji i jeftiniji diskovni sustav većeg kapaciteta
- arhitektura memorijskih struktura Oracle poslužitelja prilagođena je navedenih specifičnostima

Uvod u memorijske strukture Oracle RDBMS

Pregled memorijskih struktura Oracle sustava

- kako bi zaobišli glavni nedostatak diskovnog sustava - brzinu pristupa - operativni sustav i Oracle koriste različite cache strukture pohranjene u RAM
- arhitektura memorijskih struktura Oracle poslužitelja temelji se na modelu dijeljenih (shared) struktura, te privatnih (private) struktura
- u osnovne memorijske strukture Oracle sustava ubrajamo područje programskog koda (software code area), System Global Area (SGA), te Private Global Area (PGA)

Uvod u memorijske strukture Oracle RDBMS

Pregled memorijskih struktura Oracle sustava: SGA

- System Global Area (SGA) - predstavlja područje iz kojega Oracle poslužitelj dodjeljuje sve dijeljene memorijske strukture, te kontrolne strukture za jednu Oracle instancu.
- podaci unutar SGA dostupni su svim korisnicima na sustavu
- najvažnije komponente SGA: db buffer cache, shared pool, java pool, large pool, itd.
- SGA može biti veliki potrošač memorije na velikim sustavima iz razloga što djeluje kao cache i IPC mehanizam, a pristup podacima iz memorije je nekoliko redova veličine brži nego pristup sa diska

Uvod u memorijske strukture Oracle RDBMS

Pregled memorijskih struktura Oracle sustava: PGA

- Private Global Area (SGA) - privatno memorijsko područje
- postoji jedno PGA memorijsko područje za svaki fizički proces instance
- PGA sadrži kontrolne podatke i informacije za taj pojedini proces, ali ipak najveći dio memorije otpada na run-time područja koja se dodjeljuju prilikom izvođenja SQL cursora

Upravljanje memorijom sustava

Uvod u Virtualnu Memoriju

- podsustav za upravljanje memorijom jedan je od najznačajnijih dijelova operativnog sustava
- još od najranijih dana razvoja računalnih sustava prisutna je potreba za više memorije nego što je stvarno raspoloživo
- od raznih razvijenih strategija za upravljanje memorijom najuspješnijom se je pokazala Virtualna memorija
- mehanizam VM stvara privid da na sustavu ima više memorije nego što je ima u stvarnosti, jer se fizička memorija dodjeljuje procesima u momentu kada je zatraže

Upravljanje memorijom sustava

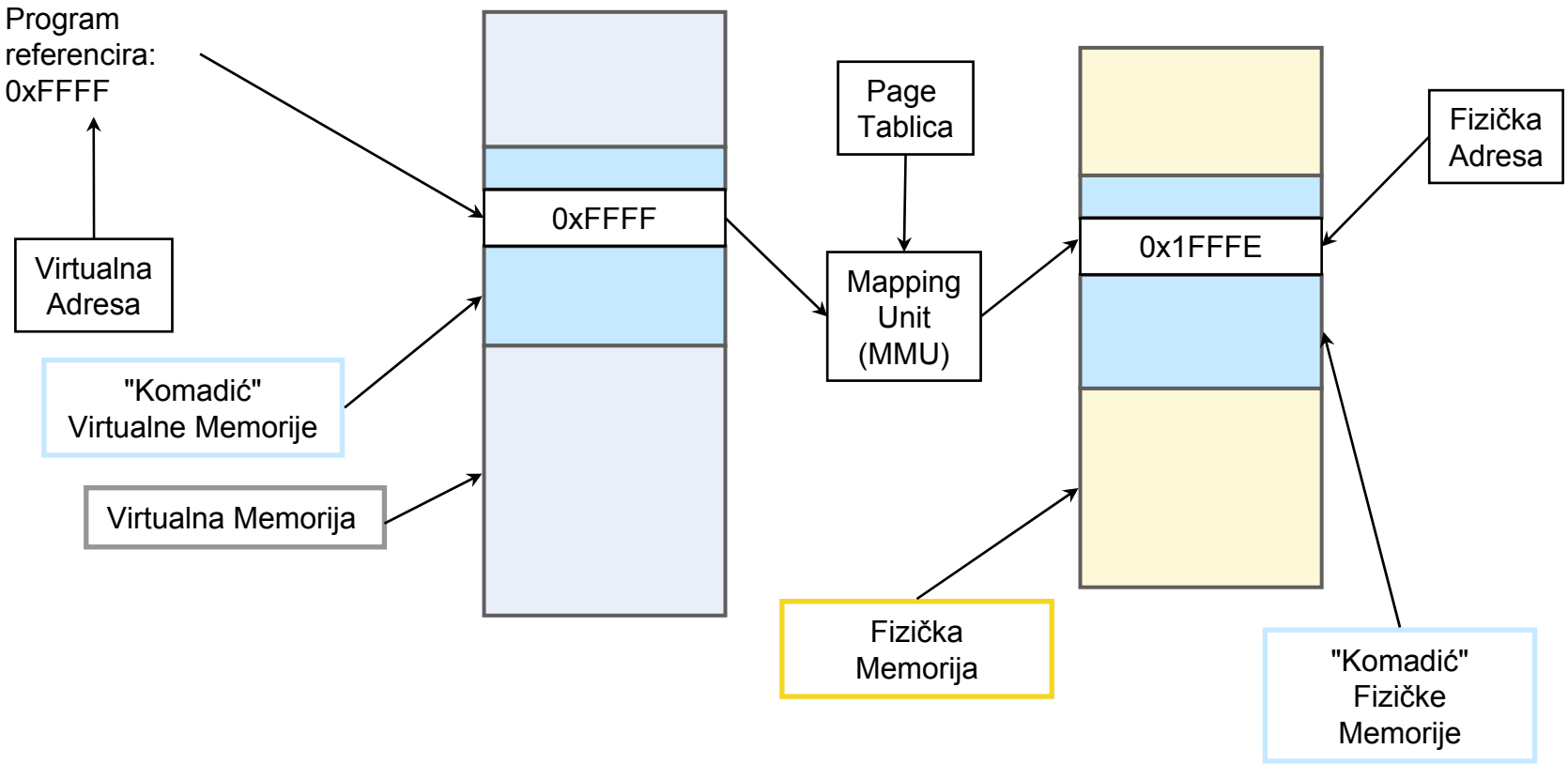
Prednosti Virtualne Memorije

VM omogućuje sljedeće prednosti:

- veliki adresni prostor - stvara privid da na računalu ima više memorije nego što je stvarno ima
- zaštita - svaki proces ima svoj virtualni adresni prostor potpuno odvojen od ostalih procesa, pa procesi ne mogu utjecati jedan na drugoga
- mapiranje memorije (memory mapping) - mapira image i datoteke direktno u virtualni adresni prostor
- poštena raspodijela memorije - ravnomjerna raspodjela fizičke memorije procesima
- zajednička virtualna memorija - dijeljenje istih fizičkih lokacija među više procesa (izvršne verzije programa, DLL biblioteke, IPC shared memory)

Upravljanje memorijom sustava

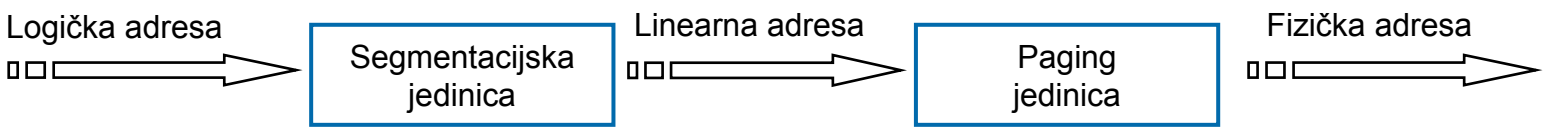
Prikaz VM mehanizma



Upravljanje memorijom sustava

Adresiranje memorije

- u x86 arhitekturi postoje tri vrste adresa: logičke adrese, linearne adrese i fizičke adrese
- kako bi mogli adresirati neku lokaciju u fizičkoj memoriji, potrebno je prvo logičku adresu pretvoriti u linearnu adresu, a zatim linearnu adresu u fizičku adresu
- kod IA-32 arhitekture resursi za upravljanje memorijom podijeljenu su u dva dijela: segmentaciju i paging



Upravljanje memorijom sustava

Segmentacija

- segmentacija je mehanizam pomoću kojega je moguće izolirati pojedinačne module koji sadrže kod, podatke ili stack, na način da se više programa ili taskova mogu izvršavati na istom procesoru bez međusobnih interferencija
- osnovna zamisao segmentacije je upravljati memorijom koristeći skup segmenata
- svaki segment nalazi se u vlastitom adresnom prostoru
- logička adresa sastoji se od dva dijela: segment identifikatora i offseta koji određuje relativnu adresu unutar segmenta

Upravljanje memorijom sustava

Segmentacija u Linuxu

- Linux kernel ima ograničenu podršku za segmentaciju zbog želje za portabilnošću na RISC platforme i preklapanjem uloga segmentacije i paginga
- u upotrebi je flat model segmentacije u kojem OS i aplikacije imaju pristup cjelokupnom, nesegmentiranom adresnom prostoru
- Linux koristi 4 segmenta: user code, user data, kernel code i kernel data segment
- svi segmenti počinju od adrese 0 (0x00000000) i završavaju na adresi 2^32-1 = 4 GB (0xFFFFFFFF)
- jedina razlika među segmentima odnosi se na razinu privilegija i tip segmenta

Upravljanje memorijom sustava

Paging mehanizam

- paging mehanizam implementira memorijski sustav temeljen na demand pagingu, izolirajući procese mapiranjem istog linearnog adresnog prostora u različiti fizički adresni prostor
- linearne adrese grupirane su u intervale fiksne veličine koje nazivamo page
- paging jedinica particionizira cijeli RAM u page frameove fiksne veličine, koje još nazivamo fizičkim pagevima
- page frame je konstituent glavne memorije i područje u koje se pohranjuju podaci
- potrebno je razlikovati page i page frame: page označava samo blok podataka koji se može nalaziti u page frameu (memoriji) ili na disku (swap)

Upravljanje memorijom sustava

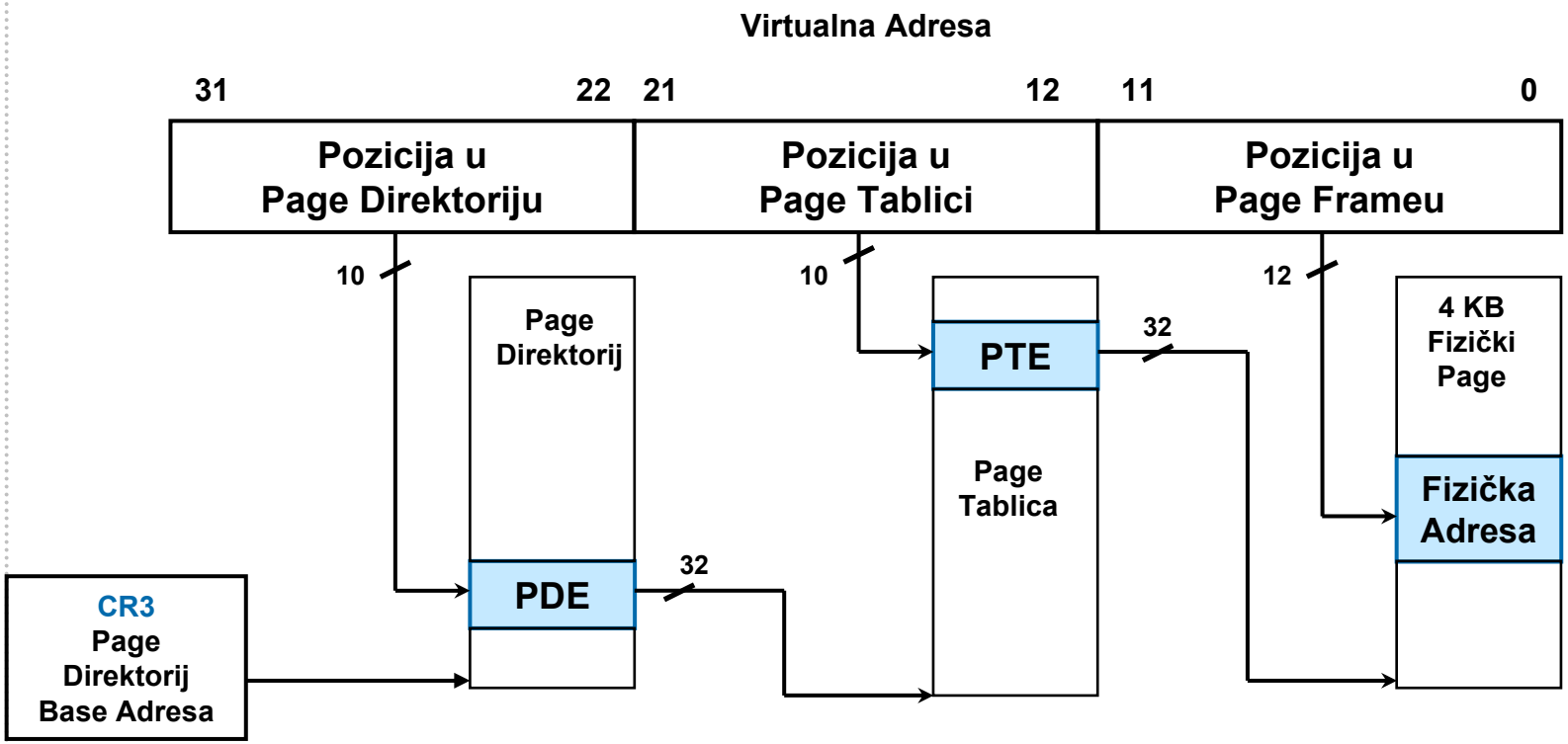
Paging mehanizam

- strukture u koje se pohranjuju mapirane linearne u fizičke adrese nazivamo page tablicama
- page tablice pohranjene su u glavnoj memoriji računala
- svaki proces ima svoju privatnu page tablicu
- veličine pageva na x86 platformi mogu biti: 4KB, 2MB i 4MB
- kod 32 bitne arhitekture moguće je da svaki proces koristi 4.294.967.296 / 4096 = 1.048.576 pageva po procesu !!!
- ideja o formiranju jednog velikog arraya sa svim pagevima za svaki proces nije praktična i potrebno je primjeniti drugi princip !!!

Upravljanje memorijom sustava

Paging mehanizam na Intel x86 32-bit arhitekturi

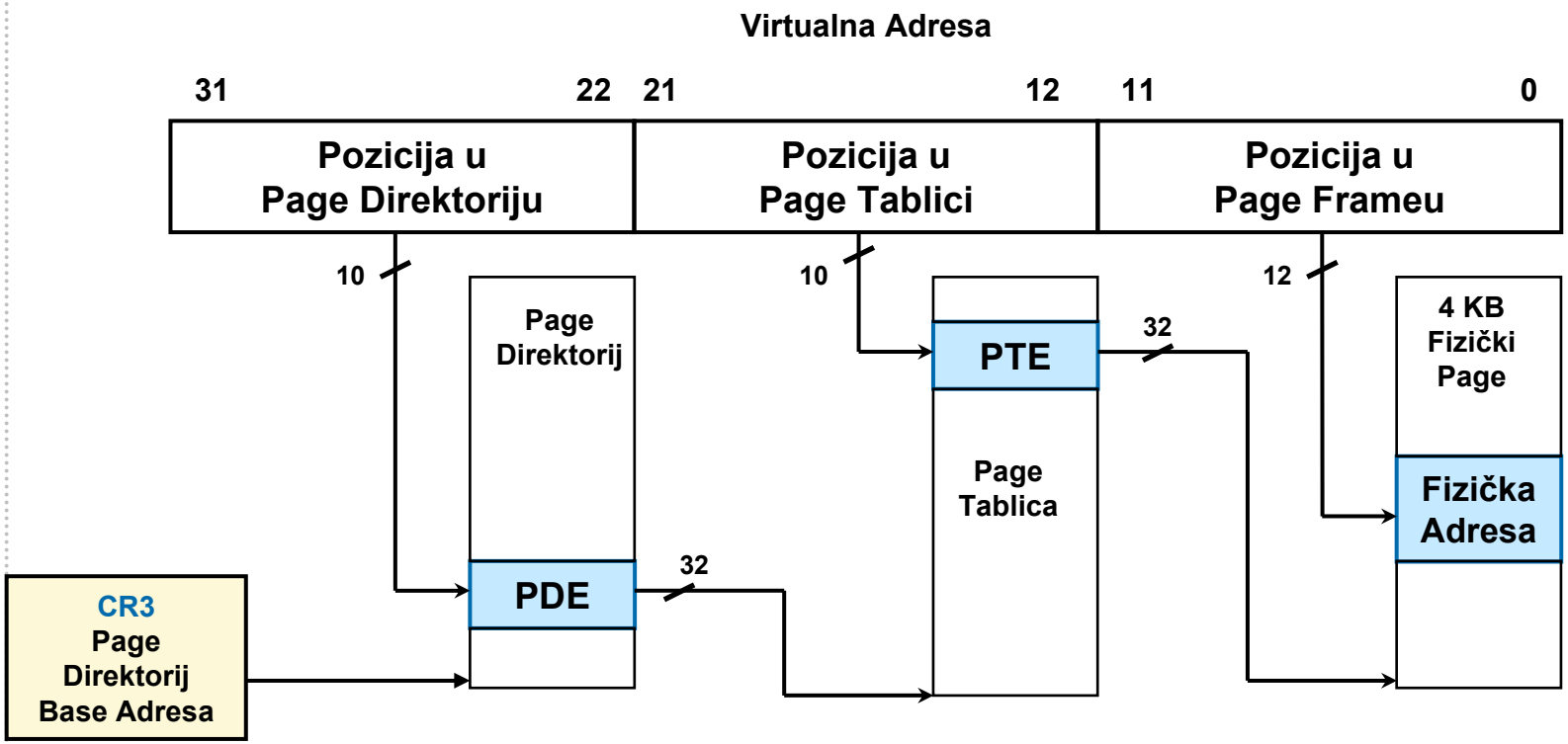
- kod Intel x86 arhitekture, paging mehanizam implementiran je na dvije razine, što znači da su linearne adrese veličine 32 bita podijeljene u 3 dijela:



Upravljanje memorijom sustava

Paging mehanizam na Intel x86 32-bit arhitekturi

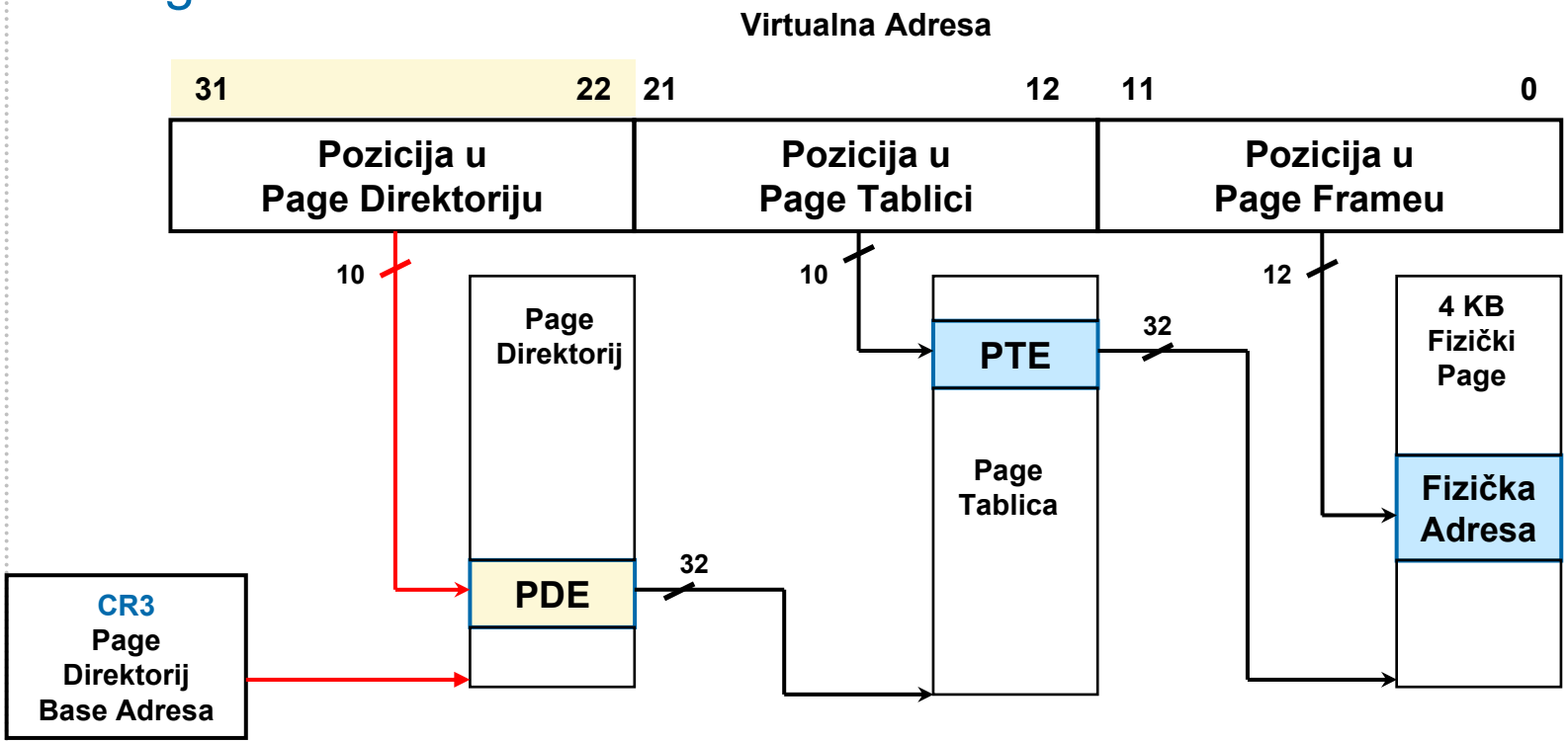
- translacija linearne u fizičku adresu vrši se na načina da se fizička adresa Page Direktorija učita u CR3 registar na procesoru



Upravljanje memorijom sustava

Paging mehanizam na Intel x86 32-bit arhitekturi

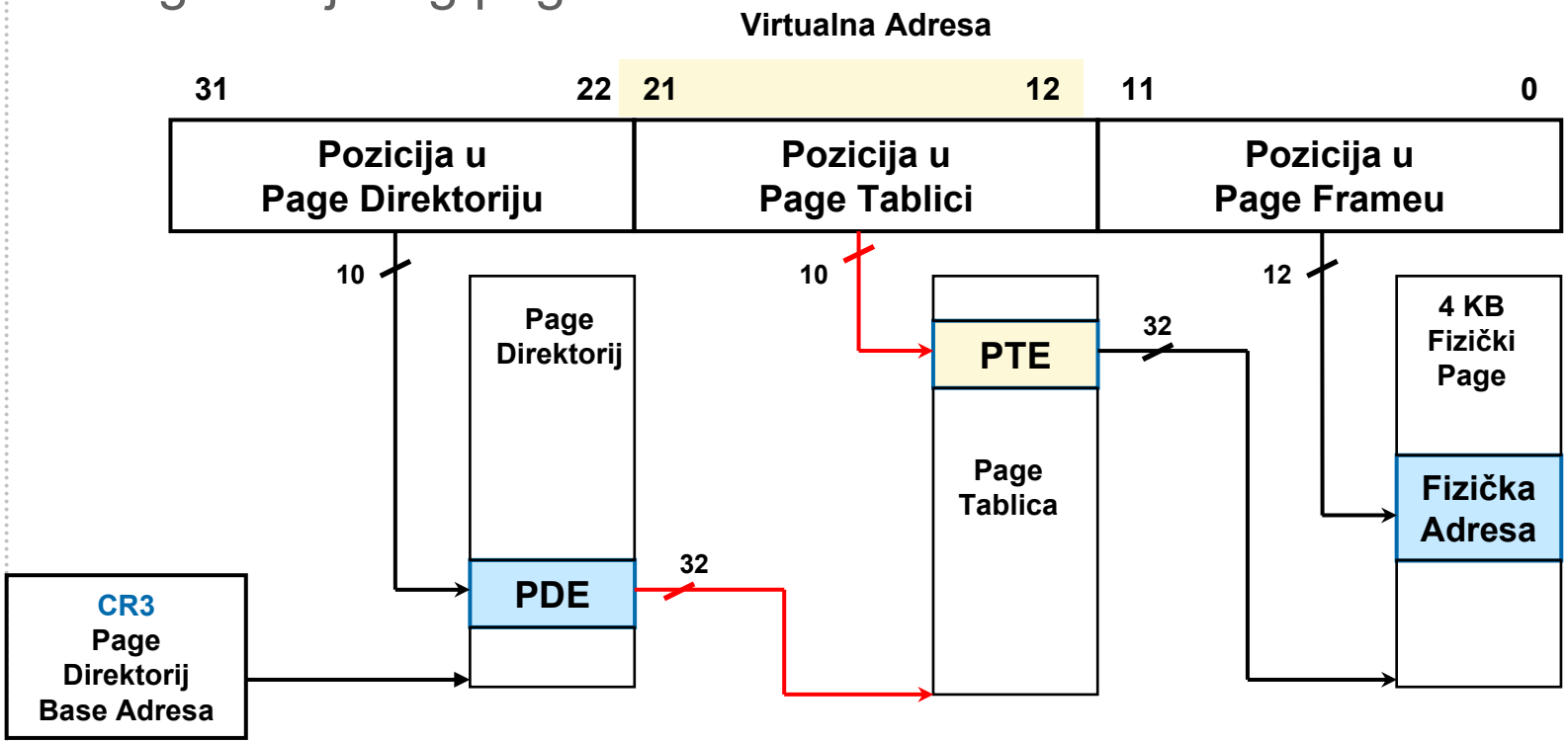
- 10 najznačanijih bitova (bitovi 22-31) referenciraju element u Page Direktoriju. Na temelju očitane vrijednosti iz Page Direktorija dobivamo baznu adresu druge tablice koju nazivamo Page Tablicom



Upravljanje memorijom sustava

Paging mehanizam na Intel x86 32-bit arhitekturi

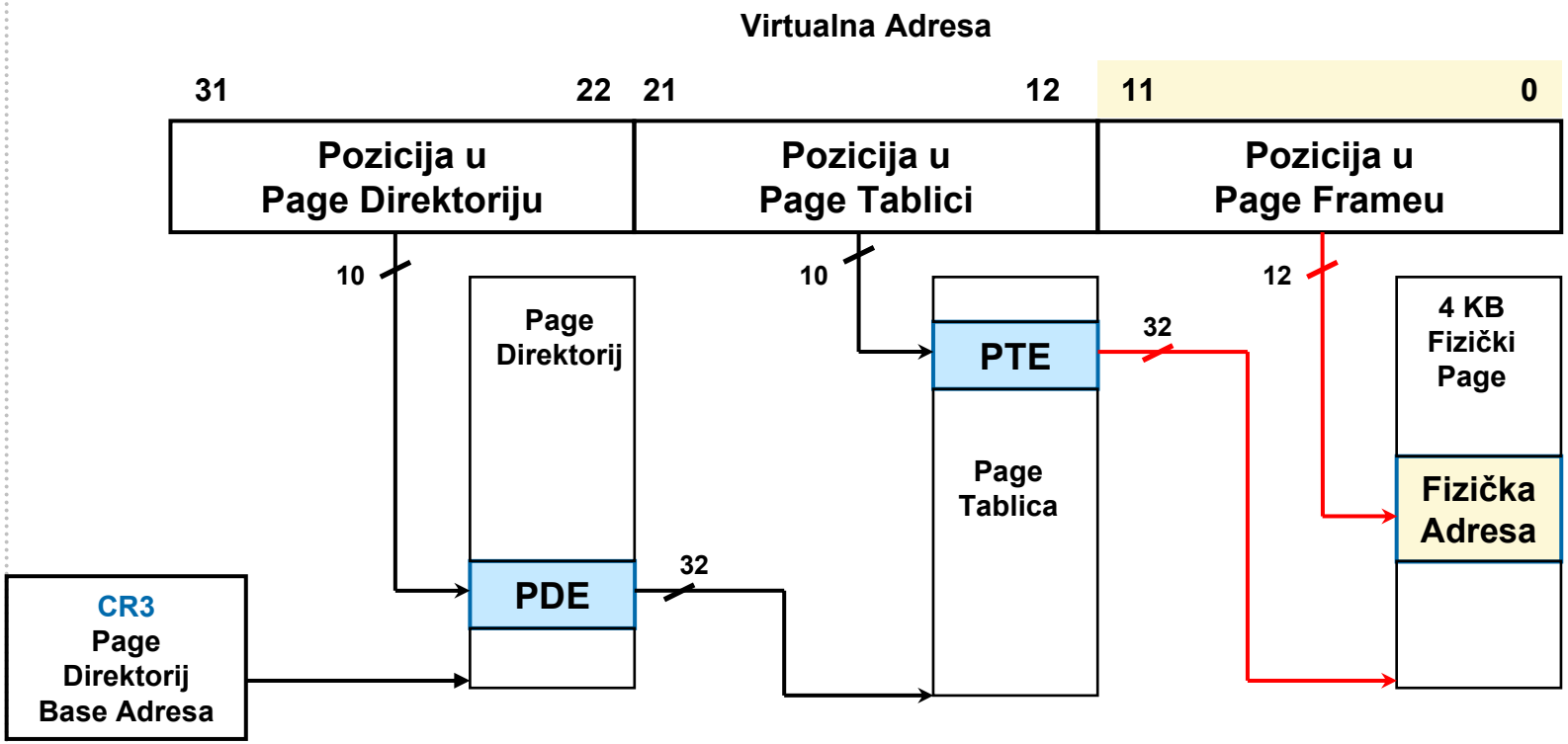
- srednjih 10 bitova (bitovi 12-21) predstavljaju adresu odgovarajućeg elementa unutar Page Tablice. Vrijednost dobivena iz Page Tablice predstavlja fizičku adresu odgovarajućeg pagea



Upravljanje memorijom sustava

Paging mehanizam na Intel x86 32-bit arhitekturi

- zadnjih 12 bitova predstavlja fizičku adresu (offset) unutar pagea koju referencira linearna adresa



Upravljanje memorijom sustava

Paging mehanizam na Intel x86 32-bit arhitekturi

za skup linearnih adresa postoji:

- jedan Page Direktorij veličine 2^10 = 1024 elemenata
- svaki od tih elemenata sadrži baznu adresu Page Tablice koja je također veličine 2^10 = 1024 elemenata
- svaki od tih elemenata iz Page Tablice sadrži fizičku adresu pagea
- svakom aktivnom procesu mora biti dodijeljen Page Direktorij, no nije potrebno dodijeliti RAM za sve elemente Page Tablice odjednom, već kada to procesu zaista zatreba

Upravljanje memorijom sustava

PAE mehanizam na x86-32 bit platformi

- veliki sustavi imaju potrebu za više od 4 GB RAM-a
- Intel odgovara povećanjem broja adresnih pinova na CPU sa 32 na 36, čime CPU može podržati 2^{36} = 64 GB RAM-a
- potrebno je uvesti novi mehanizam koji prevađa 32-bitne linearne na 36-bitne fizičke adrese, a naziv mehanizma je PAE
- 64 GB RAM-a podijeljeno je u 2^{24} = 16.777.216 pageva
- polja fizičke adrese povećana su sa 20 na 24 bita, a zajedno sa 12 bitova offseta čine 36 bitova

Upravljanje memorijom sustava

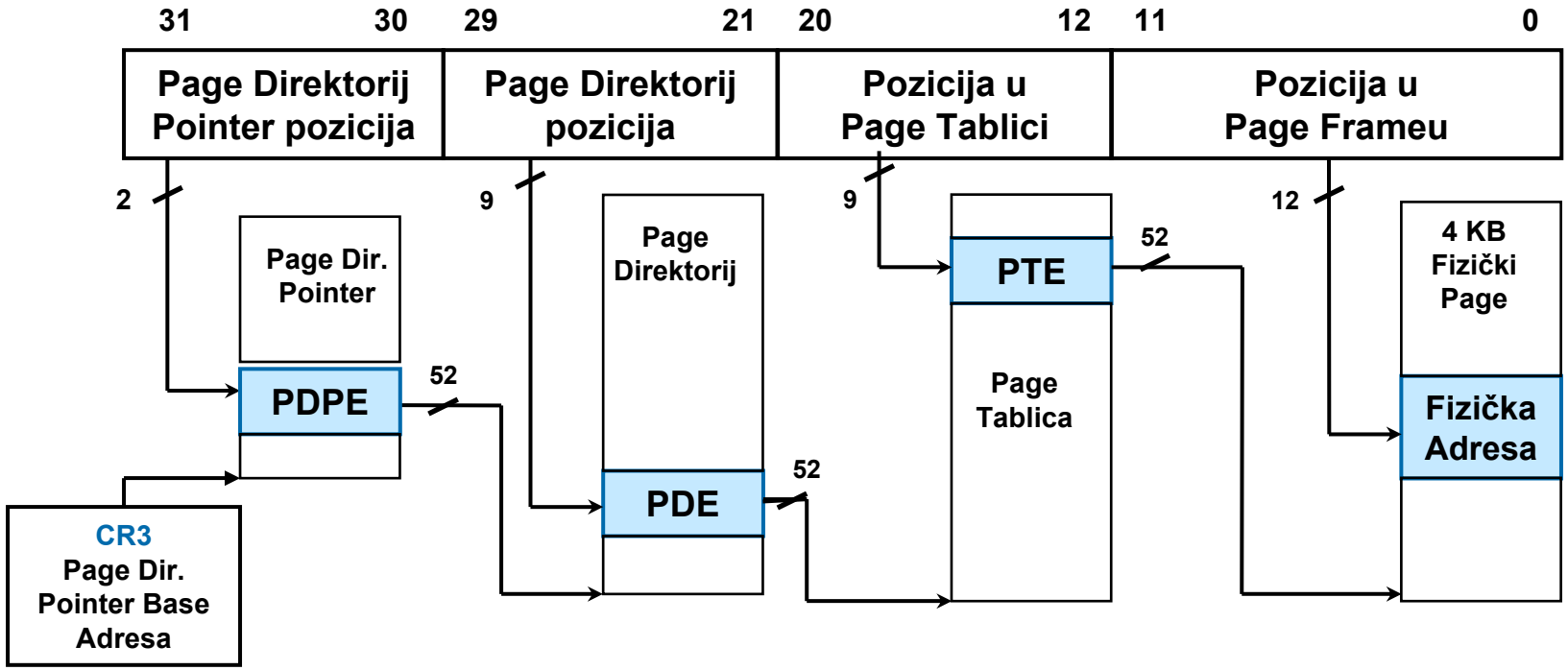
PAE mehanizam na x86-32 bit platformi

- podatkovne strukture Page Direktorija i Page Tablice povećane su sa 32 na 64 bita, kako bi mogle udomiti 36 bitne fizičke adrese
- zbog povećanja podatkovnih struktura sa 32 na 64 bita (4 na 8 byteova), u jednoj 4 KB Page Tablici stane samo 512 elemenata, umjesto prijašnjih 1024
- Novost: 4 elementa Page Direktorij Pointer tablice gledaju na Page Direktorij, a nakon toga translacija radi kao i obično
- Važno: adresni prostor procesa je i dalje ograničen na 32 bita, ali sustav sa više RAM-a može udomiti veći broj procesa

Upravljanje memorijom sustava

PAE mehanizam na x86-32 bit platformi

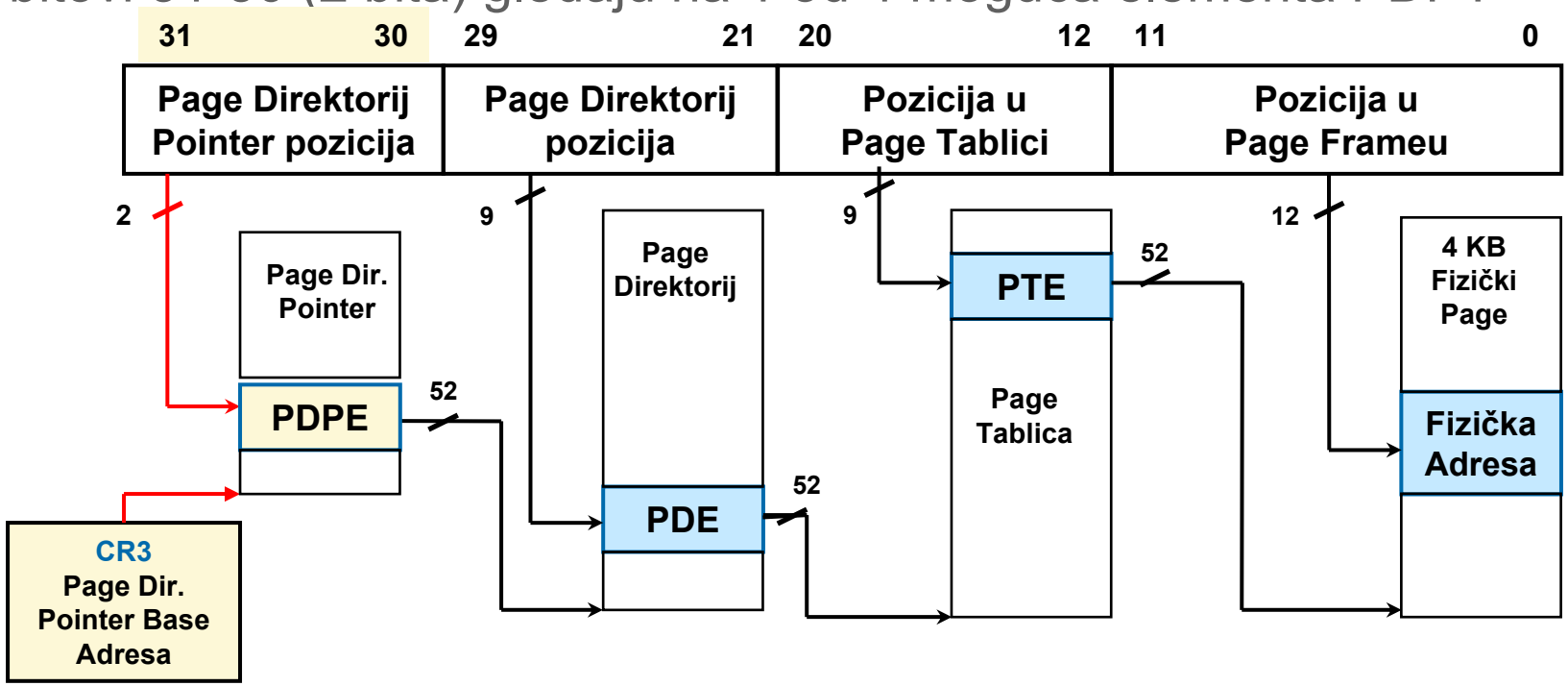
- translacija linearnih adresa na x86 koristeći PAE i 4KB pageve



Upravljanje memorijom sustava

PAE mehanizam na x86-32 bit platformi

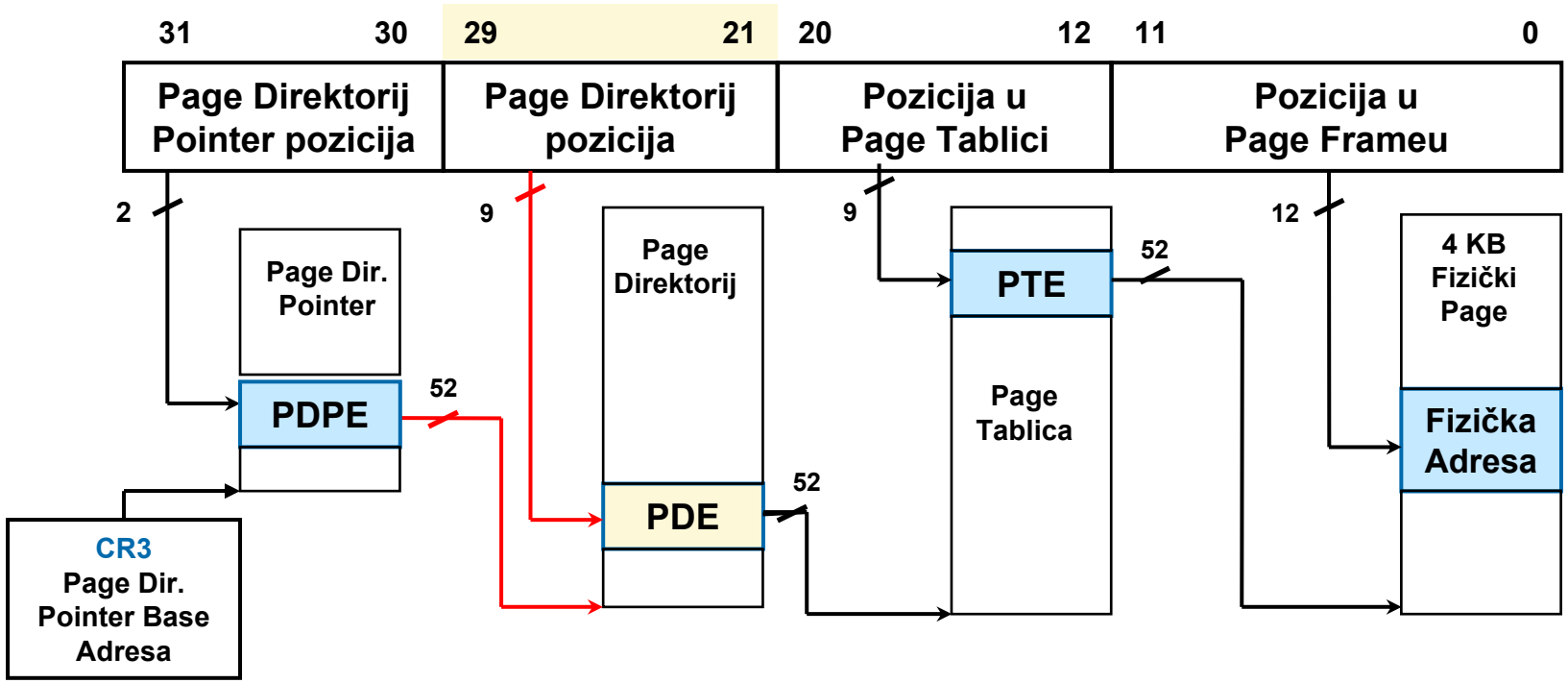
- CR3 gleda na Page Direktorij Pointer Tablicu
- bitovi 31-30 (2 bita) gledaju na 1 od 4 moguća elementa PDPT



Upravljanje memorijom sustava

PAE mehanizam na x86-32 bit platformi

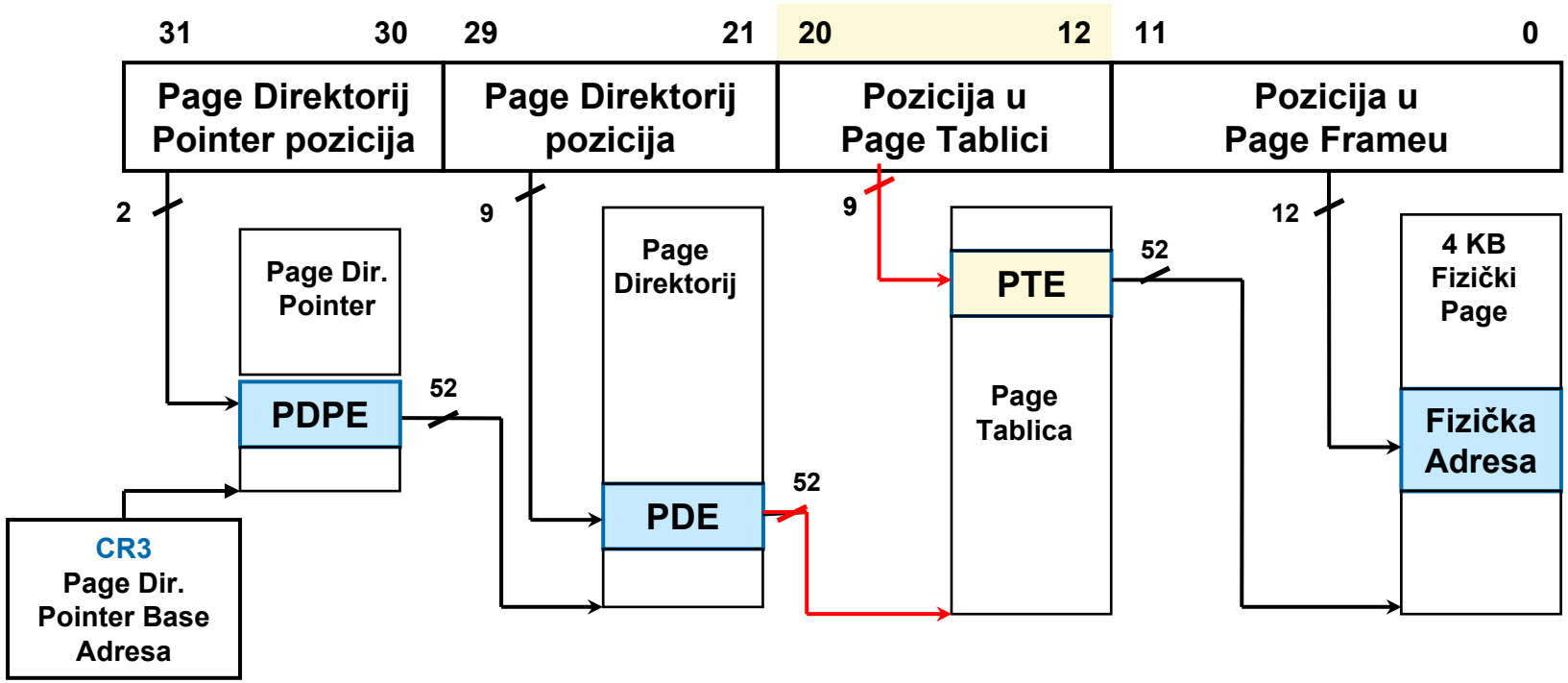
- bitovi 29-21 (9 bitova) gledaju na 1 od 512 mogućih elemenata u Page Direktoriju



Upravljanje memorijom sustava

PAE mehanizam na x86-32 bit platformi

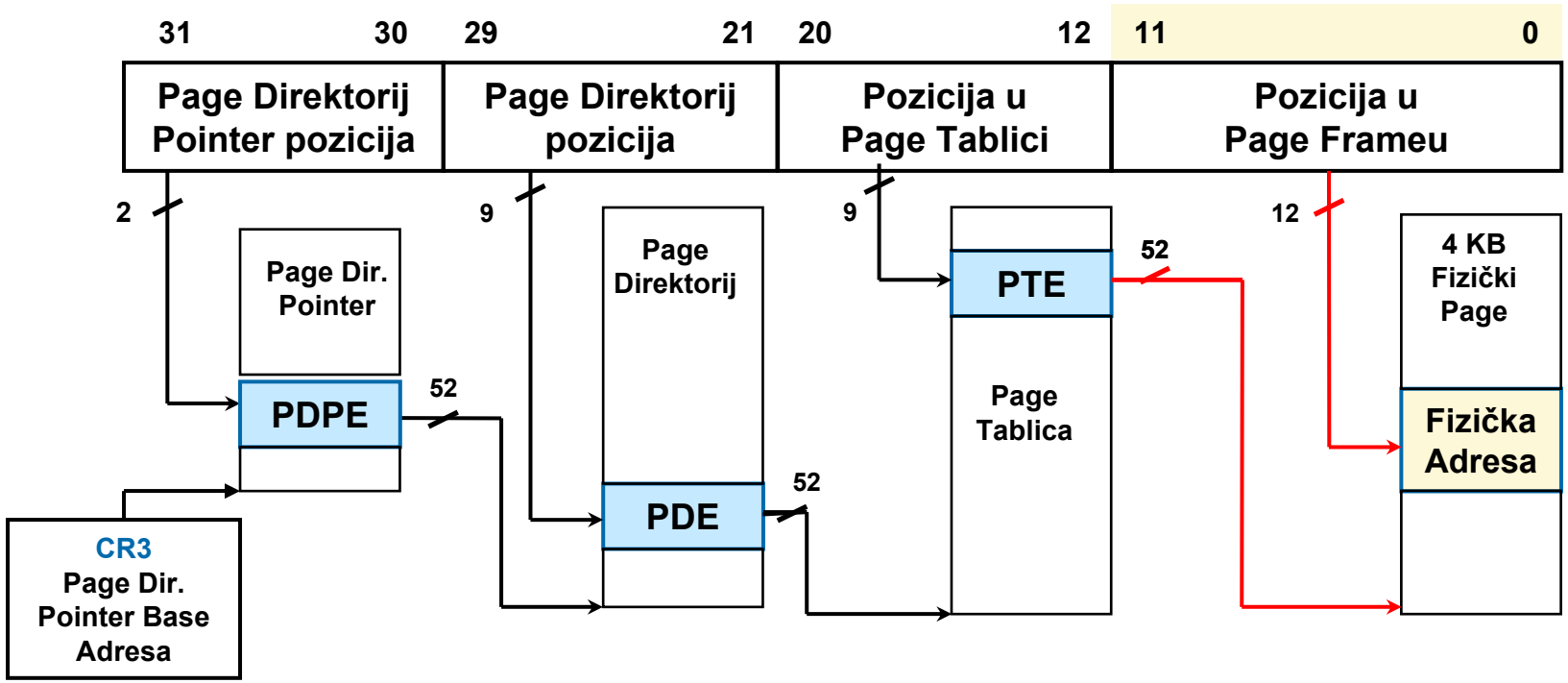
- translacija linearnih adresa na x86 koristeći PAE i 4KB pageve



Upravljanje memorijom sustava

PAE mehanizam na x86-32 bit platformi

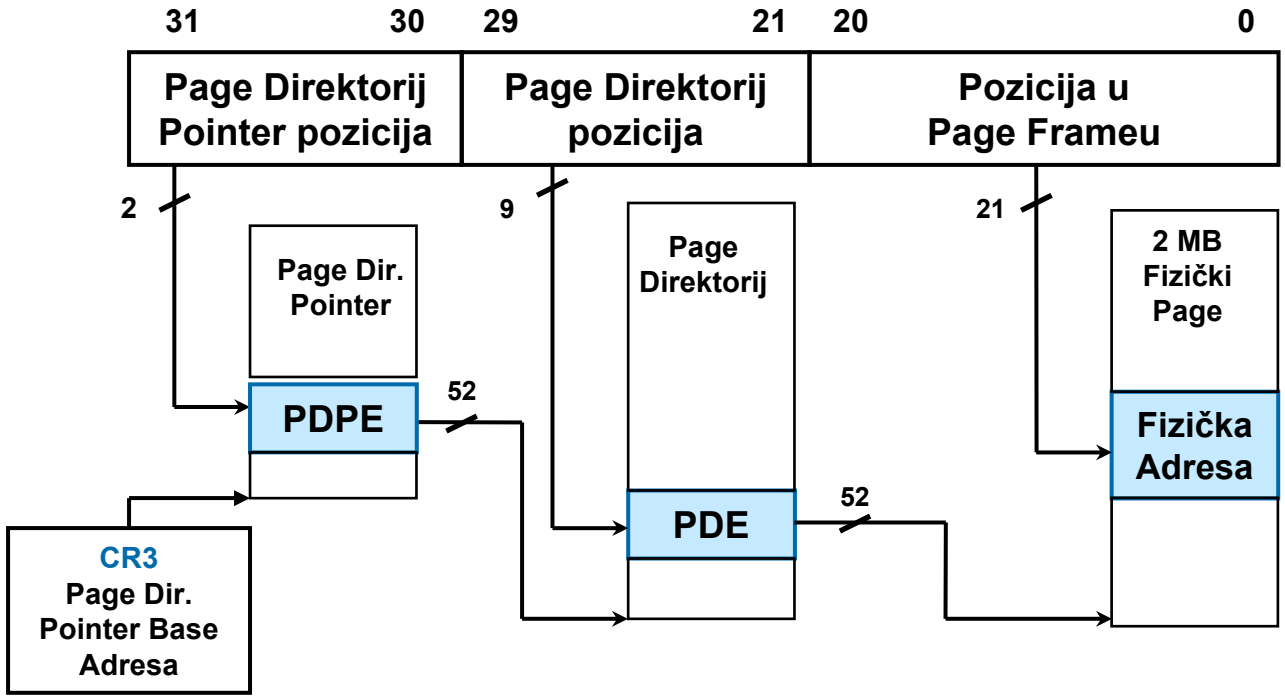
- bitovi 11-0 (12 bitova) predstavljaju offset u 4 KB page frame



Upravljanje memorijom sustava

PAE mehanizam na x86-32 bit platformi

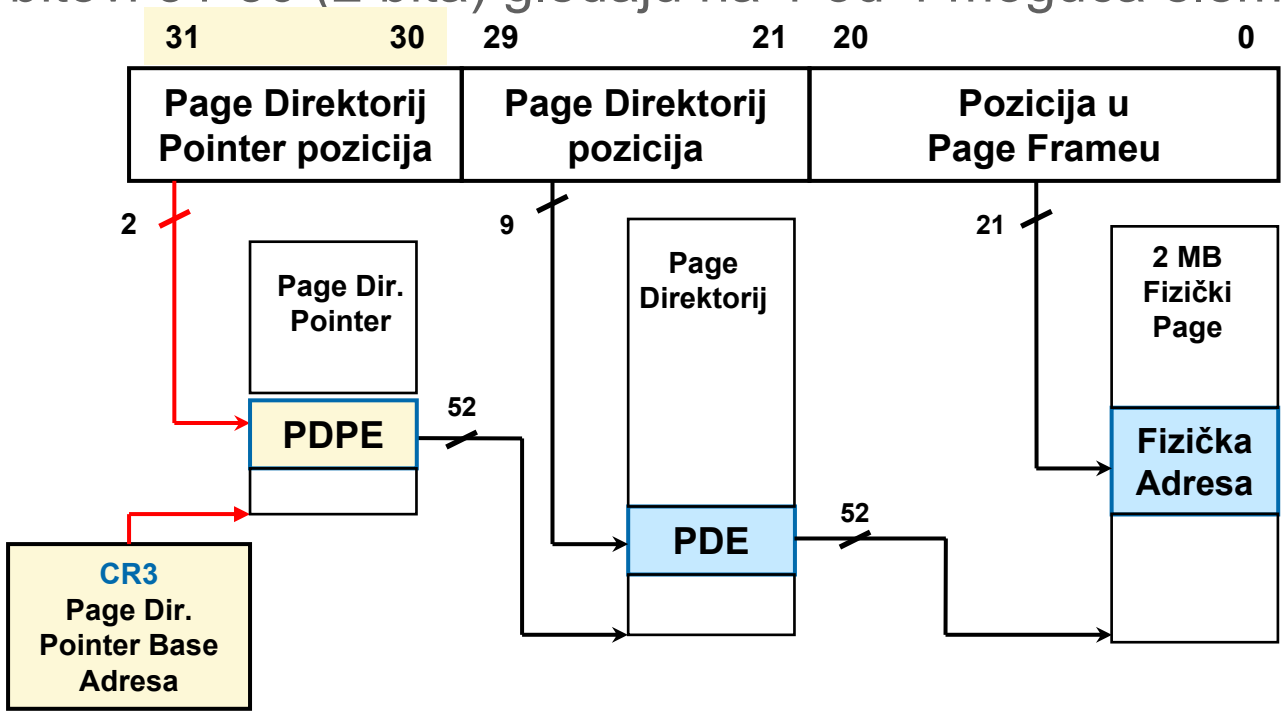
- translacija linearnih adresa na x86 koristeći PAE i 2MB pageve



Upravljanje memorijom sustava

PAE mehanizam na x86-32 bit platformi

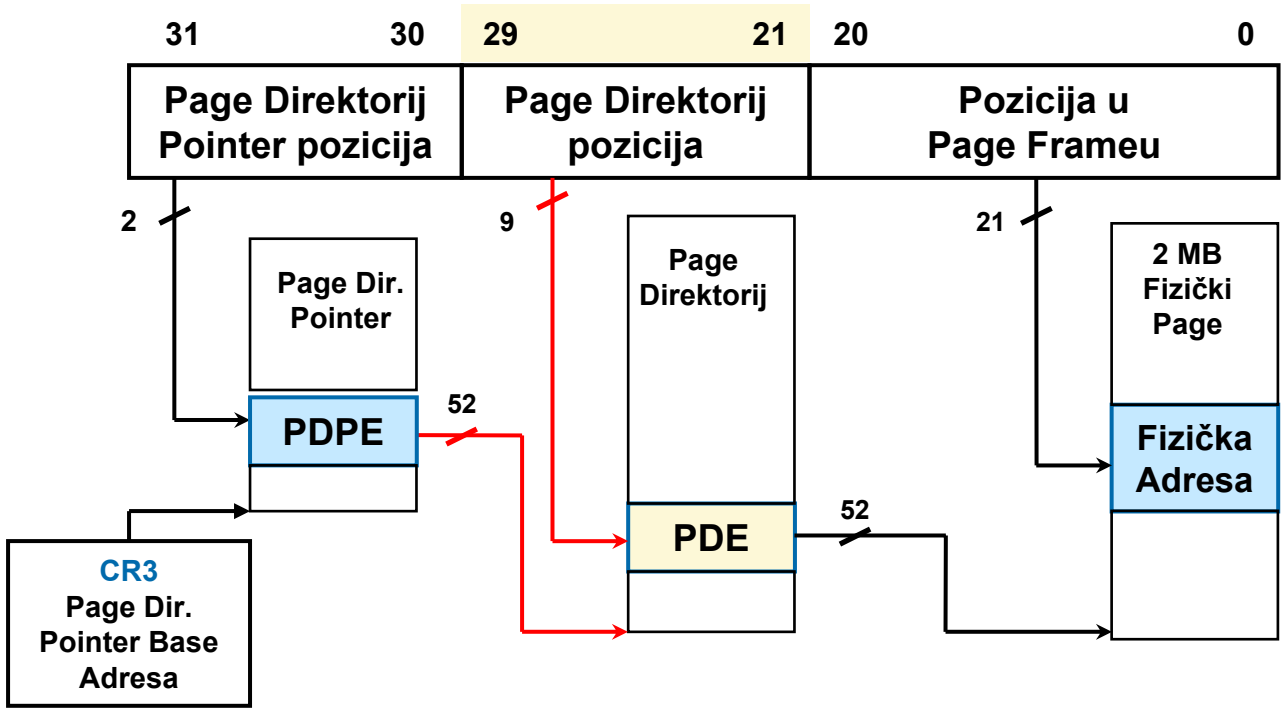
- CR3 gleda na Page Direktorij Pointer Tablicu
- bitovi 31-30 (2 bita) gledaju na 1 od 4 moguća elementa u PDPT



Upravljanje memorijom sustava

PAE mehanizam na x86-32 bit platformi

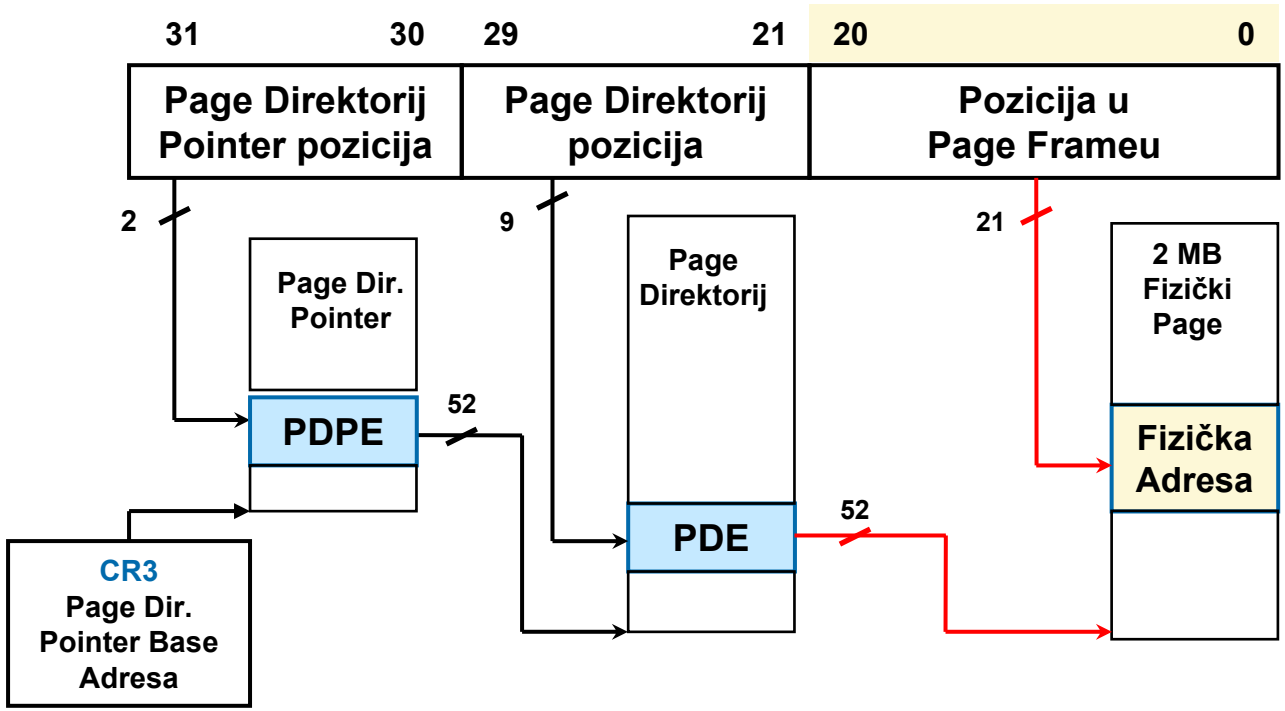
- bitovi 29-21 (9 bita) gledaju na 1 od 512 mogućih elemenata u page direktoriju



Upravljanje memorijom sustava

PAE mehanizam na x86-32 bit platformi

- bitovi 20-0 (21 bit) predstavljaju offset u 2 MB pageu



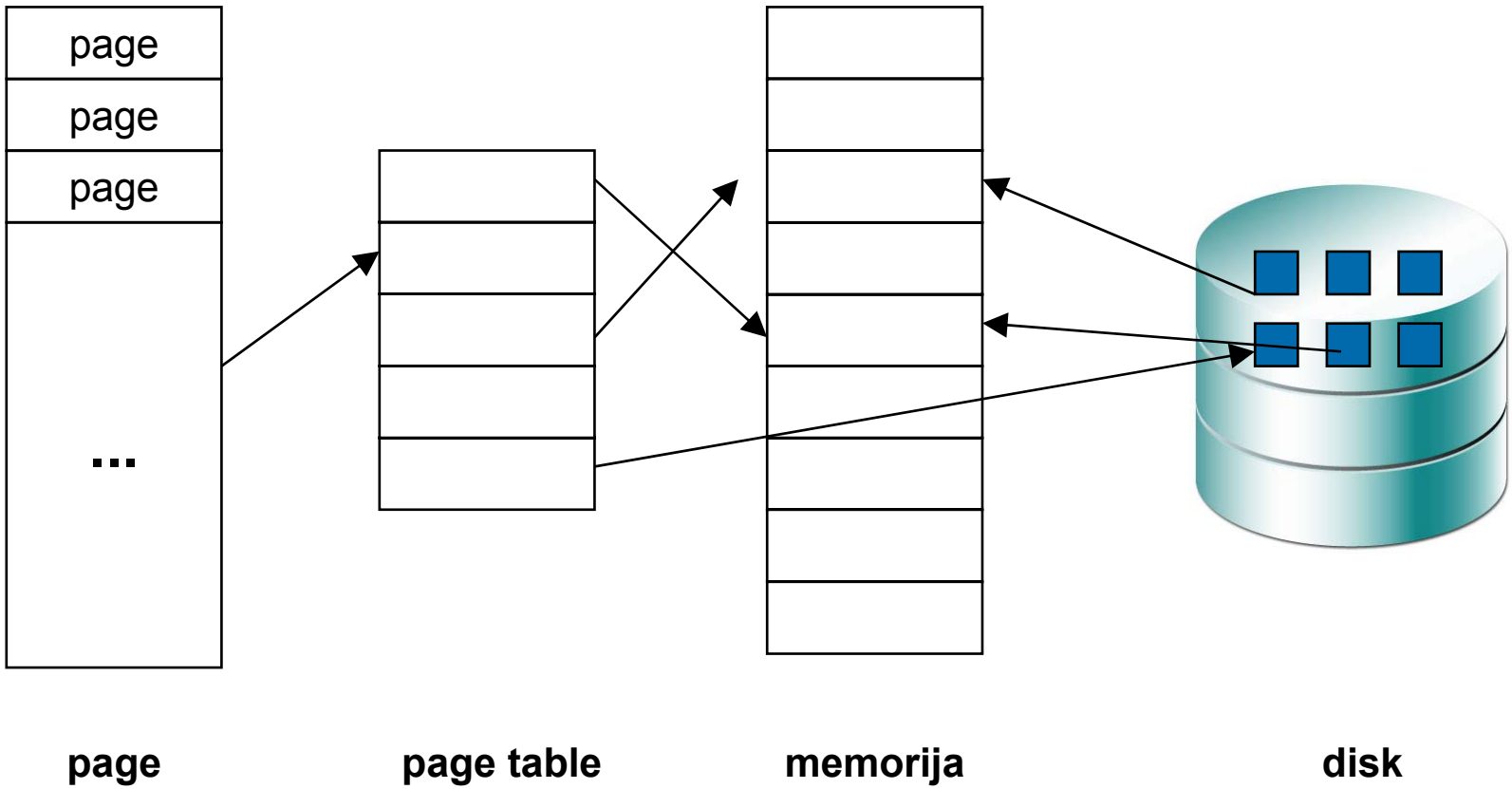
Upravljanje memorijom sustava

Demand paging

- kada CPU pristupi virtualnoj adresi koja nije u memoriji, tada obavještava operativni sustav da je nastao *page fault*
- jedan od načina štednje fizičke memorije je učitavanje samo onih virtualnih pageva koji se trenutno koriste
- CPU mora učitati odgovarajući page u memoriju sa diska, a po završetku, proces restarta operaciju pristupa virtualnoj adresi koja je pokrenula page fault
- taj mehanizam bolje je poznat kao demand paging i njegov je utjecaj vidljiv kada je sustav pod opterećenjem, ali i kada se image neke datoteke prvi puta učitava u memoriju
- drugim riječima to znači da proces može izvršavati image binarne izvršne datoteke koja se djelomično nalazi učitana u memoriju u nekom određenom trenutku

Upravljanje memorijom sustava

Demand paging



Upravljanje memorijom sustava

Swapping

- kada fizička memorija na sustavu ponestane a proces mora učitati page u fizičku memoriju tada operativni sustav mora odlučiti što učiniti
- OS mora ravnomjerno dijeliti fizičke pageve među procesima, a ponekad mora osloboditi jedan ili više pageva kako bi osigurao mjesta u memoriji za nove pageve
- način na koji se vrši izbor pageva koji moraju napustiti fizičku memoriju utječe na performanse sustava
- ukoliko je page stari a nije modificiran ne mora biti pohranjen, dok se tzv. "dirty", odnosno prljavi pagevi spremaju u swap datoteku
- ukoliko swapping mehanizam nije efikasan nastaje fenomen pod nazivom trashing, gdje OS neprestano zapisuje i čita pageve iz swap datoteke oduzimajući resurse user mod procesima

Upravljanje memorijom sustava

System V IPC Shared Memory

- set mehanizama koji omogućuju User Mode procesima sinhronizaciju sa drugim procesima
- svaki proces mora u svoj adresni prostor dodati novu memorijsku regiju koja će mapirati page frameove koji pripadaju tom SHM segmentu
- shmat() dodavanjem memorijske regije ne modificira elemente page tablice koji pripadaju tom procesu
- IPC pagevi mogu biti swapani na disk, a kernel mora paziti jer su ti pagave perzistentni i moraju biti sačuvani čak i kada nisu mapirani sa strane niti jednog procesa

Utjecaj upravljanja memorijom na Oracle

Osnovna konfiguracija kernel parametara

Parametar	Opis	Datoteka
shmall	maksimalni broj pageva u jednom shared memory segmentu	/proc/sys/kernel/shmall
shmmax	maksimalna veličina shared memory segmenta	/proc/sys/kernel/shmmax
shmmni	maksimalni broj shared memory segmenata koje je moguće kreirati na operativnom sustavu	/proc/sys/kernel/shmmni
nr_hugepages	podrška za HugePages mehanizam	/proc/sys/vm/nr_hugepages

Utjecaj upravljanja memorijom na Oracle

Veličina elemenata PTE u fizičkoj memoriji

- kako proces počinje referencirati razne dijelove SGA, tako operativni sustav počinje računati translaciju linearnih u fizičke adrese i spremati ih u PTE
- koliko memorije zauzmu PTE za SGA od 2 GB ?
- na x86_64 za instancu sa 1000 procesa to iznosi 4MB * 1000 = 4000MB

Platforma	Veličina SGA (KB)	Veličina Pagea (KB)	Broj Pageva	Veličina PTE	Ukupno memorije za PTE
x86, bez PAE	**2.097.152**	**4**	**2.097.152 / 4 = 524.288**	**32 bita (4 Bytea)**	**524.288 * 4 = 2.097.152**
x86_64	**2.097.152**	**4**	**2.097.152 / 4 = 524.288**	**64 bita (8 Bytea)**	**524.288 *8 = 4.194.304**

Utjecaj upravljanja memorijom na Oracle

Case study - karakteristike sustava

- ozbiljnost problema najbolje je prikazati primjerom iz stvarnog života na sustavu koji ima sljedeće karakteristike:
- Hardver platforma: 4 x Intel Xeon CPU 3.00 GHz; RAM 12 GB; FC Disk Array
- OS: RedHat Enterprise Linux AS 4
- Oracle RDBMS: Oracle 9.2.0.8 EE za Linux EM64T/AMD64
- Veličina SGA: 6 GB

Utjecaj upravljanja memorijom na Oracle

Case study - prikaz TOP komande

```
top - 09:29:58 up 12 days, 15:18,  2 users,  load average: 49.10, 43.69, 28.51
Tasks: 711 total,  51 running, 660 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.0% us, 100.0% sy,  0.0% ni,  0.0% id,  0.0% wa,  0.0% hi,  0.0% si
Mem:  12301728k total, 12276956k used,    24772k free,     1656k buffers
Swap: 10241396k total,   441368k used,  9800028k free,  5854872k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
24375 oracle    25   0 23508 1704  760 R 17.7  0.0   0:52.33 httpd
 7360 root      25   0     0    0    0 R 17.3  0.0  30:35.66 kjournald
13649 oracle    25   0 6251m 452m 449m R 17.3  3.8   1:20.60 oracle
24356 oracle    25   0 74084 4552 4052 R 17.3  0.0   0:54.70 oracle
24366 oracle    25   0 74080 4244 3800 R 17.3  0.0   0:49.09 oracle
  300 oracle    25   0 6246m 572m 568m R 17.0  4.8   1:37.26 oracle
 2850 root      25   0     0    0    0 R 17.0  0.0  36:16.80 kjournald
 7347 root      25   0     0    0    0 R 17.0  0.0  33:08.82 kjournald
10726 oracle    25   0 6245m 284m 279m R 17.0  2.4   1:38.65 oracle
11263 oracle    25   0 6242m 192m 191m R 17.0  1.6   1:20.20 oracle
13609 oracle    25   0 6288m 693m 661m R 17.0  5.8   2:50.59 oracle
19872 oracle    25   0 6384m 1.5g 1.3g R 17.0 12.5   3:49.07 oracle
24210 oracle    25   0 6242m  45m  43m R 17.0  0.4   0:55.75 oracle
27661 oracle    25   0 6261m 1.1g 1.1g R 17.0  9.2   2:44.09 oracle
  102 root      25   0     0    0    0 R 16.7  0.0 297:03.04 kswapd0
 1562 oracle    25   0 6272m 2.0g 1.9g R 16.7 16.7   4:16.20 oracle
 7816 oracle    25   0 6242m  46m  45m R 16.7  0.4  62:23.05 oracle
 7820 oracle    25   0 6248m  29m  29m R 16.7  0.2  99:17.00 oracle
10341 oracle    25   0 20400 1256  624 R 16.7  0.0  56:09.24 httpd
16888 oracle    25   0 6246m 1.9g 1.9g R 16.7 16.3   2:43.21 oracle
```

Utjecaj upravljanja memorijom na Oracle
Case study - Statspack report

```
STATSPACK report for
DB Name         DB Id    Instance     Inst Num Release     Cluster Host
------------ ---------- ------------ -------- ----------- ------- ------------
xxxx          2994866433 xxxx                1 9.2.0.8.0   NO      xxx
              Snap Id     Snap Time      Sessions Curs/Sess Comment
            --------- ------------------ -------- --------- -------------------
Begin Snap:         7 05-Pro-07 10:47:01      879      19.6
  End Snap:         8 05-Pro-07 10:54:02      850      19.2
   Elapsed:               7.02 (mins)
Cache Sizes (end)
~~~~~~~~~~~~~~~~~
               Buffer Cache:     5,120M      Std Block Size:          8K
           Shared Pool Size:       512M          Log Buffer:     20,480K
Instance Efficiency Percentages (Target 100%)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
            Buffer Nowait %:  100.00       Redo NoWait %:    100.00
            Buffer  Hit   %:   98.20    In-memory Sort %:    100.00
            Library Hit   %:   99.95        Soft Parse %:     99.82
         Execute to Parse %:   21.84         Latch Hit %:     99.93
Parse CPU to Parse Elapsd %:    1.46     % Non-Parse CPU:     89.14
 Shared Pool Statistics        Begin   End
                               ------  ------
             Memory Usage %:   97.69   97.55
    % SQL with executions>1:   70.75   70.52
  % Memory for SQL w/exec>1:   42.47   42.00
Top 5 Timed Events
~~~~~~~~~~~~~~~~~~                                                 % Total
Event                                             Waits    Time (s) Ela Time
--------------------------------------------- ------------ ----------- --------
latch free                                         5,682      40,768    86.39
db file sequential read                           73,604       2,786     5.90
```

Utjecaj upravljanja memorijom na Oracle

Case study - vmstat komanda

```
zzz ***Mon Dec 10 11:50:36 UTC 2007
procs -----------memory---------- ---swap-- -----io---- --system-- ----cpu----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in    cs us sy id wa
28 19 740200 119860   5332 5830520    9    6   764   227    2     3  7  4 74 14
17  8 739820 125092   6508 5846268 1060   20 15120  2972 9400 16352 61 31  0  7
53  9 754464  26644   7008 5849104  592 5476 13444  7280 77696 35660  1 99  0  0
zzz ***Mon Dec 10 11:51:47 UTC 2007
procs -----------memory---------- ---swap-- -----io---- --system-- ----cpu----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in    cs us sy id wa
42 16 771824  24788   6844 5837332    9    6   764   227    2     3  7  4 74 14
41  9 781360  24852   7144 5842672  996 10452  6324 11524 2954  4102 17 83  0  0
55  3 784924  24868   7008 5843952  496 6824  6984  7256 2719  3476 14 86  0  0
zzz ***Mon Dec 10 11:59:23 UTC 2007
procs -----------memory---------- ---swap-- -----io---- --system-- ----cpu----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in    cs us sy id wa
37 25 975696 236444   7980 5719820    9    6   764   227    2     3  7  4 74 14
20 19 971736 205716   8988 5740068 2504    0 17152  2716 8522 14899 76 24  0  0
29 17 968792 156492   9644 5758996 1708    0 13256  5532 6298 11472 79 21  0  0
```

Utjecaj upravljanja memorijom na Oracle

Case study - PGA: mogući memory leak bug

```
select * from v$pgastat;
NAME                                                  VALUE UNIT
---------------------------------------- ---------------- ------------
aggregate PGA target parameter              2,097,152,000 bytes
aggregate PGA auto target                   1,576,074,240 bytes
global memory bound                           104,857,600 bytes
total PGA inuse                               425,730,048 bytes
total PGA allocated                           786,807,808 bytes
maximum PGA allocated                       1,407,452,160 bytes
total freeable PGA memory                     178,061,312 bytes
PGA memory freed back to OS               243,791,167,488 bytes
total PGA used for auto workareas              79,711,232 bytes
maximum PGA used for auto workareas           519,530,496 bytes
total PGA used for manual workareas                     0 bytes
maximum PGA used for manual workareas           1,060,864 bytes
over allocation count                                   0
bytes processed                           401,277,350,912 bytes
extra bytes read/written                   27,127,411,712 bytes
cache hit percentage                                   94 percent
```

Utjecaj upravljanja memorijom na Oracle

Case study - /proc/meminfo

```
MemTotal:     12301728 kB
MemFree:         90988 kB
Buffers:         12332 kB
Cached:        5106004 kB
SwapCached:     604524 kB
Active:        6266772 kB
Inactive:       644832 kB
HighTotal:           0 kB
HighFree:            0 kB
LowTotal:     12301728 kB
LowFree:         90988 kB
SwapTotal:    10241396 kB
SwapFree:      7986728 kB
Dirty:             412 kB
Writeback:           0 kB
Mapped:        6143188 kB
Slab:           115652 kB
CommitLimit:  16392260 kB
Committed_AS: 11811264 kB
PageTables:    5.098.956 <= PTE zauzimaju više od 40% ukupnog RAMa!!!
VmallocTotal: 536870911 kB
VmallocUsed:      7696 kB
VmallocChunk: 536863111 kB
HugePages_Total:     0
HugePages_Free:      0
Hugepagesize:     2048 kB
```

Utjecaj upravljanja memorijom na Oracle

Case study - rješenje problema

- rješenje je u implementaciji i obaveznom korištenju HugePages mehanizma
- Huge Pages je alternativa 4KB pagevima
- na x86_64 sa PAE mehanizmom, HugePagevi su veličine 2 MB

```
$ grep "Hugepagesize" /proc/meminfo
Hugepagesize:     2048 kB
```

- Osnovne karakteristike HugePageva: nisu swapabilni, smanjenje veličine page tablice, smanjeno opterećenje kod pretraživanja page tablice, efikasnije korištenje TLB mehanizma

Utjecaj upravljanja memorijom na Oracle

Case study - rješenje problema

- drugim riječima, to znači da cijeli SGA veličine 6 GB, proces može referencirati sa:
- 6.291.456 KB / 2048 KB = **3072 PTE * 8 = 25 KB** za razliku od:
- 6.291.456 KB / 4 KB = **1.572.864 PTE * 8 = 12.5 MB**

Utjecaj upravljanja memorijom na Oracle

Case study - rješenje problema

- sve što je potrebno napraviti kako bi omogućili Oracle instanci da koristi HugePages mehanizam na Linux 2.6 verziji, npr. za SGA veličine 2 GB:
- u /etc/sysctl.conf dodati: **`vm.nr_hugepages = 1024`**
- te u /etc/security/limits.conf dodati:

```
oracle  soft  memlock      2097152
oracle  hard  memlock      2097152
```

- Oracle instanca prilikom sljedećeg pokretanja u shmget() pozivu postavi SHM_HUGETLB flag i time automatski koristi HugePages mehanizam, a to je moguće provjeriti sa:

```
$ grep "HugePages_" /proc/meminfo
HugePages_Total: 4096
HugePages_Free: 1181
```

Utjecaj upravljanja memorijom na Oracle

Case study - rješenje problema

- na kraju zanimljivo je usporediti promjenu koja je nastala uvođenjem HugePages mehanizma na sustav iz slučaja kojega smo opisali, što možemo provjeriti iz sadržaja /proc/meminfo datoteke:

Stanje sa 4 KB pagevima:

```
$ grep "PageTables" /proc/meminfo
PageTables: 5098956 kB
```

Stanje nakon aktiviranja 2MB HugePageva:

```
$ grep "PageTables" /proc/meminfo
PageTables: 55852 kB
```

Utjecaj upravljanja memorijom na Oracle

Oracle 11g Automatic Memory Management i podrška za Linux HugePages mehanizam

- novo svojstvo baze podataka automatski upravlja cijelom SGA+PGA memorijom, automatski prebacujući memorijske resurse instance na one komponente kojima je to potrebno (MEMORY_TARGET i MEMORY_MAX_TARGET parametri)
- postavlja se pitanje: kako je moguće oduzimati dio SGA memorije da bi ga se dodijelilo PGA dijelu, kada znamo da kernel ne smije eliminirati shared memory pageve, odnosno može ih eventulano swapati na disk?

Oracle 11g Automatic Memory Management i podrška za Linux HugePages mehanizam

- Oracle 11g koristi /dev/shm za implementaciju dijeljene memorije
- AMM memorijski segmenti su datoteke mapirane u VM na /dev/shm
- u osnovi to je datotečni sustav koji sprema sve datoteke u VM i stoga uopće ne koristi diskovni sustav za pohranu
- po prvi puta Oracle je odustao od konvencionalno dugogodišnjeg korištenja SysV IPC Shared Memory mehanizma
- OPREZ: trenutna Linux implementacija shm/tmpfs ne podržava HugePages

Zaključak

- upravljanje memorijskim resursima jedna je od najvažnijih zadaća svakog operativnog sustava
- za performanse i stabilnost rada Oracle RDBMS izuzetno je važno da podsustav za upravljanje memorijom bude efikasan i stabilan
- paging mehanizam može uzrokovati probleme sa performansama
- korištenjem HugePages mehanizma ti se problemi mogu ublažiti ili otkloniti
- 11g Automatic Memory Management ne podržava HugePages mehanizam, stoga je potreban oprez

Pitanja i Odgovori