

Višeprosorsko i višejezgreno planiranje



Klasifikacije višeprocorskih sustava

Slabo spojeni ili distribuirani višeprocorski sustav

- Sastoji se od zbirke relativno autonomnih sustava, pri čemu svaki procesor ima svoju glavnu memoriju i U/I kanale

Funkcionalno specijalizirani procesori

- Postoji glavni procesor opće namjene
- Specijalizirani procesori su pod kontrolom glavnog procesora

Čvrsto spojeni višeprocorski sustavi

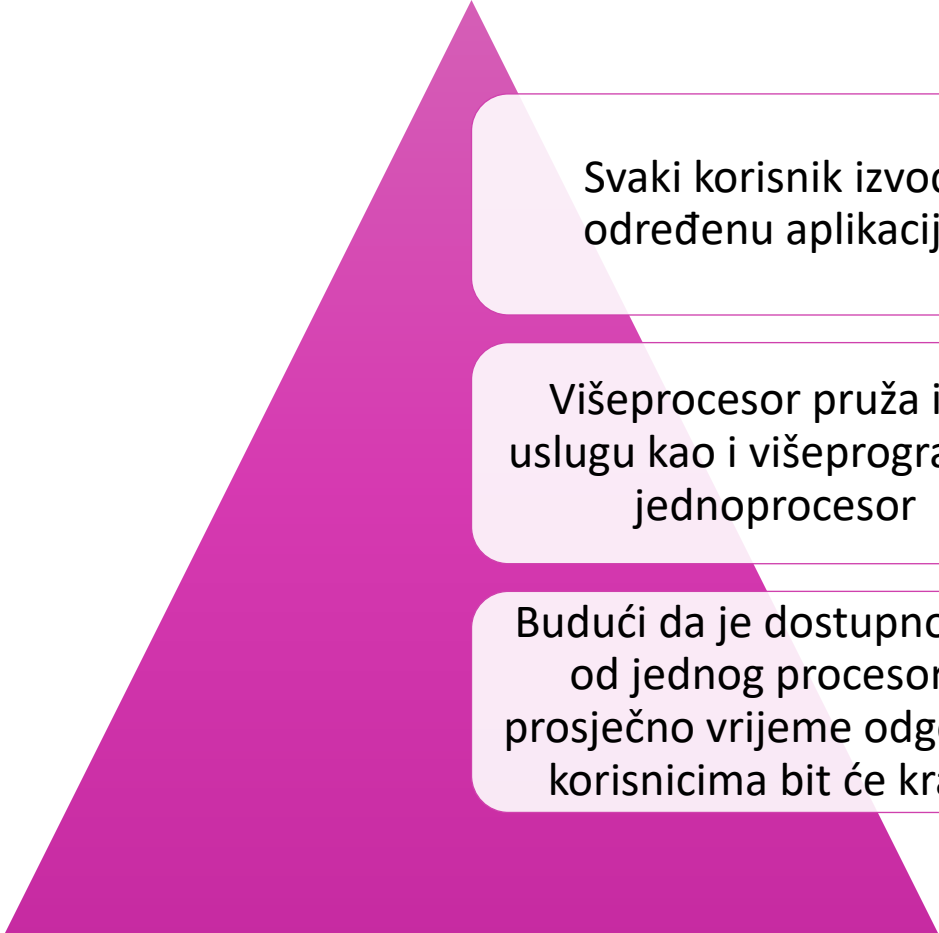
- Sastoji se od skupa procesora koji dijele zajedničku glavnu memoriju i koji su pod integriranom kontrolom operativnog sustava

Sinkronizacija granularnost

Veličina	Opis	Interval sinkronizacije
Fine	Paralelizam svojstven jednom toku instrukcija	6-20
Medium	Paralelna obrada ili multitasking unutar jedne aplikacije	20-200
Coarse	Višestruka obrada istodobnih procesa u višeprogramskom okruženju	200-2000
Very Coarse	Distribuirana obrada po mrežnim čvorovima kako bi se formiralo jedinstveno računalno okruženje	2000-1000000
Independent	Više nepovezanih procesa	na

Nezavisni paralelizam

- Nema eksplicitne sinkronizacije među procesima
 - Svaki predstavlja zasebni, neovisni posao
- Tipična upotreba je u sustavu dijeljenja vremena



Svaki korisnik izvodi određenu aplikaciju

Višeprosesor pruža istu uslugu kao i višeprogramski jednoprocesor

Budući da je dostupno više od jednog procesora, prosječno vrijeme odgovora korisnicima bit će kraće

Grubi i vrlo grubi paralelizam

- Postoji sinkronizacija među procesima, ali na vrlo gruboj razini
- Lako se obrađuje kao skup istodobnih procesa koji se izvode na višeprogramskom jednoprosesoru
- Može se podržati na višeprosesoru s malo ili bez promjene korisničkog softvera

Srednji paralelizam

- Jedna aplikacija može se učinkovito implementirati kao zbirka dretve unutar jednog procesa
 - Programer mora eksplicitno specificirati potencijalni paralelizam aplikacije
 - Mora postojati visok stupanj koordinacije i interakcije među dretvema aplikacije, što dovodi do srednje razine sinkronizacije
- Budući da različite dretve aplikacije međusobno djeluju tako često, odluke o rasporedu koje se odnose na jednu dretvu mogu utjecati na izvedbu cijele aplikacije

Fini paralelizam

- Predstavlja mnogo složeniju upotrebu paralelizma nego što se nalazi u upotrebi dretvi
- To je specijalizirano područje s mnogo različitih pristupa

Izazovi dizajna

- Poduzeti pristup ovisit će o stupnju granularnosti aplikacija i broju dostupnih procesora



Dodjela procesa procesorima

- Nedostatak statičke dodjele je da jedan procesor može biti u stanju mirovanja dok drugi procesor ima zaostatake
 - Kako bi se spriječila ova situacija, može se koristiti uobičajeni red čekanja
 - Druga opcija je dinamičko balansiranje opterećenja



Dodjela procesa procesorima

- I dinamičke i statičke metode zahtijevaju neki način dodjeljivanja procesa procesoru
- pristupi:
 - Master/Slave
 - Peer

Master/Slave arhitektura

- Ključne funkcije kernela uvijek rade na određenom procesoru
- Master je odgovoran za dodjeljivanje procesorskog vremena
- Slave šalje zahtjev za uslugu masteru
- Jednostavan je i zahtijeva malo poboljšanja jednoprocesorskog operacijskog sustava za više programa
- Rješavanje sukoba je pojednostavljeno jer jedan procesor ima kontrolu nad svim memorijskim i I/O resursima

Nedostaci:

- Neuspjeh Mastera ruši cijeli sustav
- Master može postati usko grlo u izvedbi

Peer arhitektura

- Kernel se može izvršiti na bilo kojem procesoru
- Svaki procesor vrši samododjeljivanje iz skupa dostupnih procesa

Komplicira operaciski sustav

- Operaciski sustav mora osigurati da dva procesora ne odaberu isti proces i da se procesi na neki način ne izgube iz reda čekanja

Planiranje procesa

- U većini tradicionalnih višeprocorskih sustava procesi nisu dodjeljeni procesorima
- Za sve procesore koristi se jedan red čekanja
 - Ako se koristi neka vrsta prioritetne sheme, postoji više redova koji se temelje na prioritetu, a svi se unose u zajednički skup procesora
- Sustav se promatra kao arhitektura čekanja na više poslužitelja

Planiranje dretvi

- Izvršenje dretvi je odvojeno od ostatka definicije procesa
- Aplikacija može biti skup dretvi koje surađuju i izvršavaju se istodobno u istom adresnom prostoru
- Na jednoprocesoru, dretve se mogu koristiti kao pomoć pri strukturiranju programa i za preklapanje U/I s obradom
- U višeprocesorskom sustavu dretve se mogu koristiti za iskorištavanje istinskog paralelizma u aplikaciji
- Dramatični dobici u performansama mogući su u višeprocesorskim sustavima
- Male razlike u upravljanju dretvama i planiranju mogu imati utjecaj na aplikacije koje zahtijevaju značajnu interakciju među dretvama

Pristupi planiranju dretvi

Dijeljenje opterećenja

Procesi nisu dodijeljeni određenom procesoru

Grupno planiranje

Skup povezanih dretvi planiranih za pokretanje na skupu procesora u isto vrijeme, na bazi jedan na jedan

Četiri pristupa za planiranje višeprocorskih dretvi i dodjelu procesora su:

Dodjela namjenskog procesora

Omogućuje implicitno raspoređivanje definirano dodjeljivanjem dretvi procesorima

Dinamičko planiranje

Broj dretvi na procesu može se mijenjati tijekom izvršavanja

Dijeljenje opterećenja

- Najjednostavniji pristup i onaj koji se najviše prenosi iz jednoprocesorskog okruženja

Prednosti:

- Opterećenje se ravnomjerno raspoređuje na procesore, osiguravajući da nijedan procesor nije u stanju mirovanja dok ima zadataka
- Nije potreban centralizirani planer

- Verzije dijeljenja opterećenja:
 - First-come-first-served (FCFS)
 - Najprije najmanji broj dretvi
 - Prekidno prvi najmanji broj dretvi

Nedostaci dijeljenja opterećenja

- Središnji red čekanja zauzima područje memorije kojem se mora pristupiti na način koji provodi međusobno isključivanje
 - Može dovesti do uskih grla
- Malo je vjerojatno da će preventivne dretve nastaviti s izvršavanjem na istom procesoru
 - Keširanje može postati manje učinkovito
- Ako se sve dretve tretiraju kao zajednički skup dretvi, malo je vjerojatno da će sve dretve programa dobiti pristup procesorima u isto vrijeme
 - Promjene procesa mogu ugroziti performanse

Grupno planiranje

- Istovremeno raspoređivanje dretvi koje čine jedan proces

Prednosti:

- Blokiranje sinkronizacije može biti smanjeno, manje prebacivanja procesa može biti potrebno, a izvedba će se povećati.
- Dodatni troškovi planiranja mogu se smanjiti

- Korisno za srednje i fino paralelne aplikacije čija se izvedba ozbiljno pogoršava kada bilo koji dio aplikacije nije pokrenut dok su drugi dijelovi spremni za rad
- Također je korisno za svaku paralelnu primjenu

Dodjela namjenskog procesora

- Kada je aplikacija zakazana, svaka njena dretva dodjeljuje se procesoru koji ostaje posvećen toj dretvi dok se aplikacija ne završi
- Ako je dretva aplikacije blokirana čekajući U/I ili sinkronizaciju s drugom dretvom, tada procesor te dretve ostaje neaktivan
 - Nema multiprogramiranja procesora
- Prednosti strategije:
 - U vrlo paralelnom sustavu, s desecima ili stotinama procesora, iskorištenost procesora više nije toliko važna kao metrika učinkovitosti ili performansi
 - Potpuno izbjegavanje promjene procesa tijekom životnog vijeka programa trebalo bi rezultirati značajnim ubrzanjem tog programa

Dinamičko planiranje

- Za neke aplikacije moguće je osigurati jezične i systemske alate koji dopuštaju da se broj dretvi u procesu dinamički mijenja
 - To bi omogućilo operaciskom sustavu da prilagodi opterećenje kako bi se poboljšala iskorištenost
- I operaciski sustav i aplikacija sudjeluju u donošenju odluka o rasporedu
- Odgovornost za raspoređivanje operacijskog sustava prvenstveno je ograničena na dodjelu procesora
- Ovaj pristup je superiorniji od grupnog raspoređivanja ili dodjele procesora za aplikacije koje ga znaju iskoristiti

Dijeljenje predmemorije

Kooperativno dijeljenje resursa

- Više dretvi pristupa istom skupu lokacija glavne memorije
- Primjeri:
 - Aplikacije koje su višedretvene
 - Interakcija dretvi

Natjecanje za resurse

- Dretve, ako rade na susjednim jezgrama, natječu se za mjesto u predmemoriji
- Ako je više predmemorije dinamički dodijeljeno jednoj dretvi, konkurentska dretva nužno ima manje dostupnog prostora predmemorije i stoga trpi degradaciju performansi
- Cilj planiranja sukoba je dodijeliti dretve jezgrama kako bi se povećala učinkovitost dijeljene predmemorije i minimizirala potreba za pristupima vanjskoj memoriji

Planiranje na Linux-u

- Tri primarne Linux raspoređivanja su:
 - SCHED_FIFO: First-in-first-out planiranje – real-time
 - SCHED_RR: Round-robin planiranje – real-time
 - SCHED_NORMAL: Druge dretve – non real-time
- Unutar svake klase može se koristiti više prioriteta, s prioritetima u klasama u stvarnom vremenu višim od prioriteta za klasu SCHED_NORMAL

Non-Real-Time planiranje

- Planer za Linux 2.4 za klasu SCHED_OTHER nije se dobro skalirao s povećanjem broja procesora i povećanjem broja procesa
- Nedostaci ovog planera uključuju:
 - Planer za Linux 2.4 koristi jedan red za izvođenje za sve procesore u simetričnom višeprocensnom sustavu (SMP)
 - To znači da se zadatak može dodijeliti na bilo koji procesor, što može biti dobro za balansiranje opterećenja, ali loše za korištenje predmemorije
 - Planer za Linux 2.4 koristi zaključavanje u redu čekanja
 - Dakle, u SMP sustavu, čin odabira zadatka za izvršenje blokira bilo koji drugi procesor od manipuliranja redovima za izvođenje, što rezultira neaktivnim procesorima koji čekaju otpuštanje zaključavanja čekanja i smanjenom učinkovitosti
 - Preuzimanje nije moguće u Linux 2.4 planeru
 - To znači da se zadatak nižeg prioriteta može izvršiti dok je zadatak višeg prioriteta čekao da se dovrši

Non-Real-Time planiranje

- Linux 2.6 koristi potpuno novi planer prioriteta poznat kao planer $O(1)$.
- Planer je dizajniran tako da je vrijeme za odabir odgovarajućeg procesa i dodjeljivanje procesoru konstantno bez obzira na opterećenje sustava ili broj procesora
- Pokazalo se da je planer $O(1)$ glomazan u kernelu jer je količina koda velika, a algoritmi složeni

Potpuno pošten raspored

Completely Fair Scheduler (CFS)

- Koristi se kao rezultat nedostataka planera $O(1)$.
- Modelira idealan višezadačni procesor na stvarnom hardveru koji pruža pošten pristup svim zadacima
- Kako bi se postigao ovaj cilj, CFS održava virtualno vrijeme izvođenja za svaki zadatak
 - Virtualno vrijeme izvođenja je količina vremena provedenog u izvršavanju, normalizirano brojem pokretanih procesa
 - Što je manje virtualno vrijeme izvođenja zadatka, to je veća potreba za procesorom
- Uključuje koncept pravednosti spavanja kako bi se osiguralo da zadaci koji se trenutno ne mogu izvoditi dobiju usporediv udio procesora kada im to na kraju zatreba
- Implementirano klasom rasporeda *fair_sched_class*

Crveno crno drvo - Red Black Tree

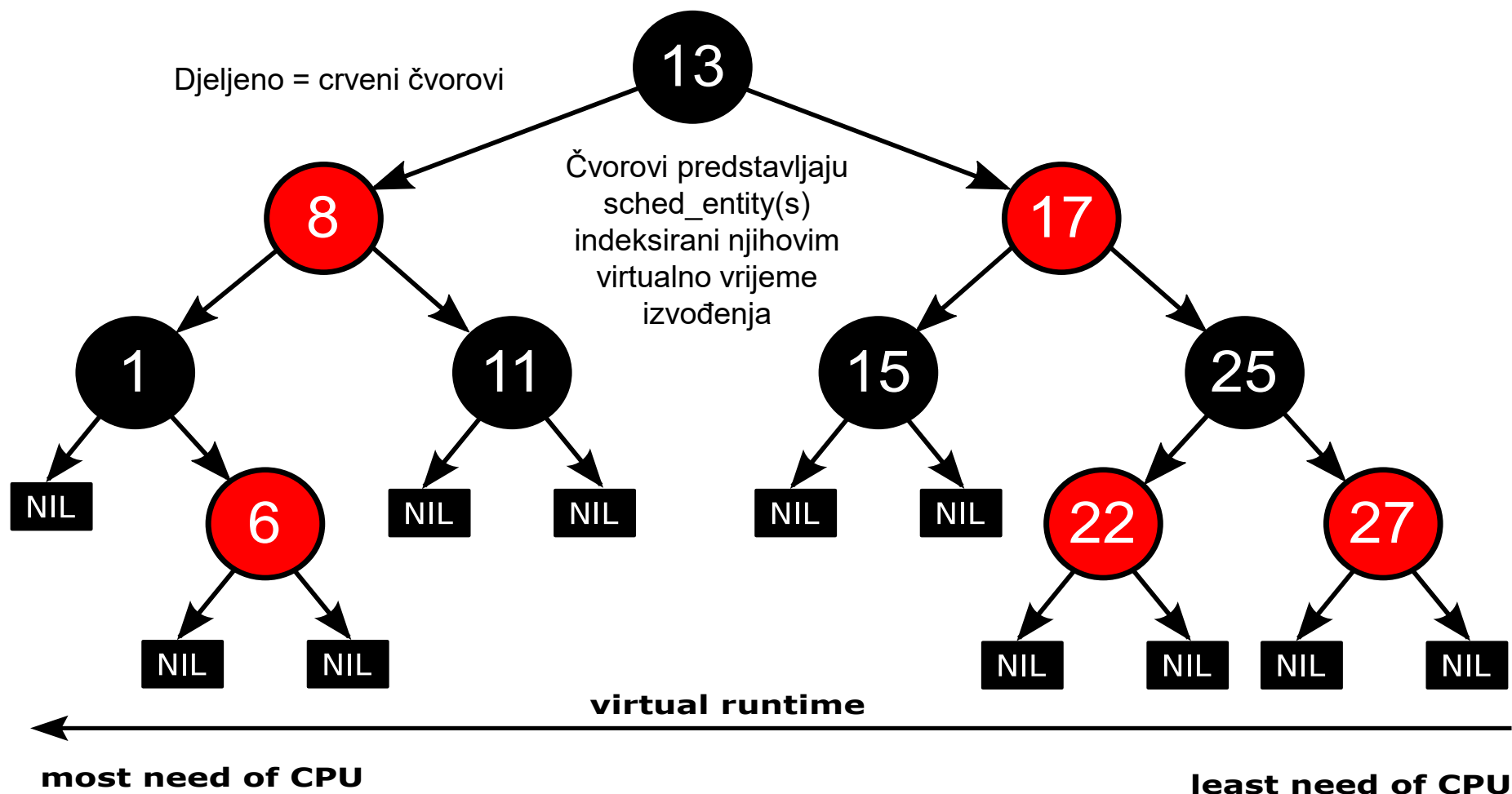
CFS planer se temelji na korištenju crveno-crnog stabla, za razliku od drugih planera, koji se obično temelje na redovima za izvođenje

- Ova shema pruža visoku učinkovitost u dodavanju, brisanju i traženju zadataka, zbog svoje $O(\log N)$ složenosti

Crveno crno stablo je vrsta samobalansirajućeg binarnog stabla pretraživanja koje poštuje sljedeća pravila:

- Čvor je crven ili crn
- Korijen je crn
- Svi listovi (NIL) su crni
- Ako je čvor crven, tada su oba njegova djeteta crna
- Svaki put od zadanog čvora do bilo kojeg od njegovih NIL čvorova potomaka sadrži isti broj crnih čvorova

Primjer Red Black Tree za CFS



UNIX SVR4 planiranje

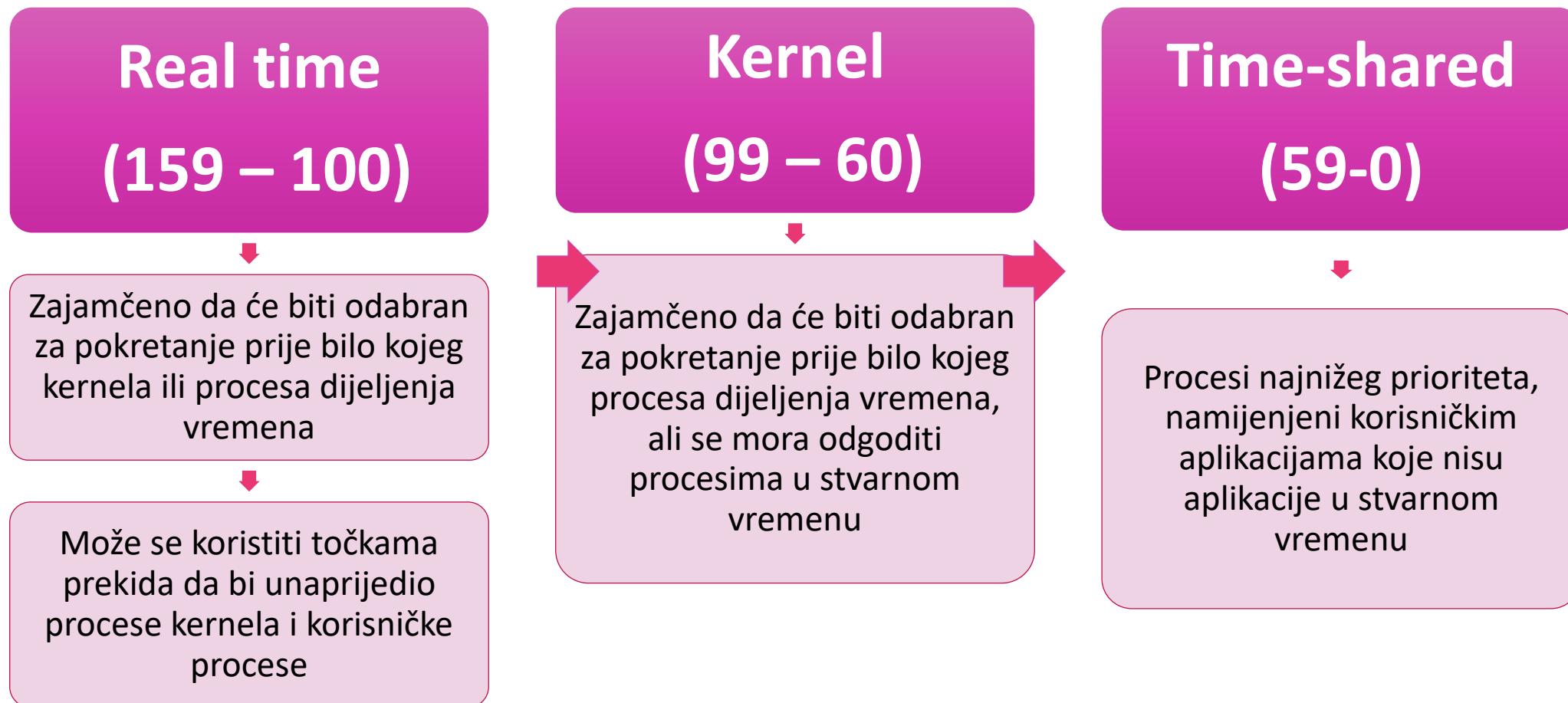
- Potpuna revizija algoritma raspoređivanja korištenog u ranijim UNIX sustavima

Novi algoritam je osmišljen tako da daje:

- Najveća prednost procesima u stvarnom vremenu
- Sljedeća najveća prednost za procese u kernel načinu rada
- Najniža prednost u korisničkom načinu rada

- Glavne izmjene:
 - Dodavanje statičkog rasporeda prioriteta i uvođenje skupa od 160 razina prioriteta podijeljenih u tri klase prioriteta
 - Umetanje točaka prednosti

Klase prioriteta kod UNIX SVR



Klase planiranja na FreeBSD sustavu

Prioritet	Tip dretve	Opis
0–63	Niska razina jezgre	Planirano s prekidima. Može biti blokiran radi čekanja resursa
64–127	Visoka razina jezgre	Radi dok se ne blokira ili ne završi. Može biti blokiran radi čekanja resursa
128–159	Korisničke aplikacije u stvarnom vremenu	Dopušteno izvođenje dok se ne blokira ili dok dretve višeg prioriteta ne zatraže pristup. Preventivno planiranje
160–223	Korisničke aplikacije	Prilagođava prioritete na temelju korištenja procesora
224–255	Neaktivni korisnik	Pokreće se samo kada nema drugih dretvi
Napomena: Niži broj odgovara višem prioritetu.		

Podrška za SMP i više jezgri

- FreeBSD planer je dizajniran da osigura učinkovito planiranje za SMP ili višejezgreni sustav
- Ciljevi dizajna:
 - Koristi afinitet procesora u SMP i višejezgrenim sustavima
 - Afinitet procesora – planer koji migrira dretvu samo kada je potrebno kako bi se izbjegao neaktivan procesor
 - Omogućite bolju podršku za višedretvenost na višejezgrenim sustavima
 - Poboljšava izvedbu algoritma za planiranje tako da više nije funkcija broja dretvi u sustavu

Interaktivnost

- Dretve se smatra interaktivnom ako je omjer njenog vremena mirovanja u odnosu na vrijeme izvođenja ispod određenog praga
- Prag interaktivnosti definiran je u kodu planera i ne može se konfigurirati

Migracija dretvi

- Afinitet procesora omogućuje dretvi koja je spremna da se izvrši na istom procesoru na kojem je radila
 - Značajno zbog predmemorija procesora



Planiranje - Windows

- Prioriteti u sustavu Windows organizirani su u dva pojasa:

Prioritetna klasa u stvarnom vremenu

- Sve dretve imaju fiksni prioritet koji se nikada ne mijenja
- Sve aktivne dretve na danoj razini prioriteta nalaze se u round-robin redu

Varijabilna klasa prioriteta

- Prioritet dretve počinje s početnom vrijednošću i može privremeno povećavati tijekom životnog ciklusa dretve

- Svaki pojas se sastoji od 16 razina prioriteta
- Dretve koje zahtijevaju trenutnu pozornost su u Real-time klasi
 - Uključite funkcije kao što su komunikacija i zadaci u stvarnom vremenu

Višeprocesorsko planiranje

- Windows podržava višeprocesorske i višejezgrene hardverske konfiguracije
- Dretve bilo kojeg procesa mogu se izvoditi na bilo kojem procesoru
- U nedostatku ograničenja afiniteta, planer dodjeljuje spremnu dretvu sljedećem dostupnom procesoru
- Više dretvi iz istog procesa može se izvršavati istovremeno na više procesora
- Meki afinitet
 - Koristi se kao zadana postavka
 - Planer pokušava dodijeliti spremnu dretvu istom procesoru na kojem je zadnji put radio
- Teški afinitet
 - Aplikacija ograničava svoje izvođenje dretvi samo na određene procesore
 - Ako je dretva spremna za izvršavanje, ali jedini dostupni procesori nisu u njezinom skupu afiniteta procesora, tada je dretva prisiljena čekati, a kernel planira sljedeću dostupnu dretvu

Sažetak

- Višeprosesorsko i višejezgreno raspoređivanje
 - Granularnost
 - Problemi dizajna
 - Planiranje procesa
 - Raspored dretvi
 - Raspored višejezgrenih dretvi
- Planiranje - Linux
 - Real-time i Non-real-time
- Planiranje UNIX FreeBSD
 - Klase prioriteta
 - SMP i višejezgrenost
- Planiranje Windows
 - Prioriteti

An abstract graphic on the left side of the slide, composed of several thick, curved lines. The lines are colored in a gradient from orange at the bottom to pink at the top. They form a shape that resembles a stylized letter 'A' or a large, open 'C' that is tilted to the right.

**Thank you for
your attention!**