Operációs rendszerek

Folyamatok ütemezése

Alapok

- Az ütemezés, az események sorrendjének a meghatározása.
- Az ütemezés használata OPR-ekben:
  - az azonos erőforrásra igényt tartó folyamatok közül történő választás, az erőforrás kiosztása, allokálása.
- Multiprogramozás foka:
  - az adott pillanatban a memóriában jelenlévő folyamatok száma.

Alapok

- A két alapvető erőforrás (CPU, Memória) kiosztása versengés útján történik.
- Az a folyamat versenghet, amelyiknek az éppen végrehajtandó utasítása a memóriában van.
- Tehát a processzort egyidejűleg egy folyamat használja, a többiek egy várakozási sorban tartózkodnak.

Alapok

- A processzor felszabadulása után az OPR választja ki, valamilyen algoritmussal a következő CPU-ra váró folyamatot.
- Ez az ún. CPU-ütemezés (rövid távú ütemező).
- A multiprogramozott OPR alapjának a CPU--ütemezés tekinthető.
- Látszólagos párhuzamos működés egy nagyobb időléptékben nézve.

Processzor ütemezés

- Rövid távú ütemezés, amely gyakran fut. Így gyorsnak kell lennie, hogy a CPU-idő ne az ütemezéssel teljék. Ezért ez az ütemező a kernel része, és állandóan a memóriában van.
- Fajtái:
  - Preemptív: ha az OPR elveheti a futás jogát az éppen futó folyamattól, és futásra kész állapotúvá teheti.
  - Nem preemptív: nem veheti el a futási jogot. Így a folyamat addig fut, amíg egy általa kiadott utasítás hatására állapotot nem vált. Azaz csak maga a folyamat válthat ki ütemezést.

Processzor ütemezés

- Ütemezést kiváltó események:
  - a futó folyamat befejeződik,
  - a futó folyamat várakozni kényszerül,
  - a futó folyamat önként lemond a futási jogáról, vagy elveszik tőle,
  - egy folyamat felébred és futásra késszé válik.

Ütemezési algoritmusok összehasonlítása

- CPU kihasználtság:
  - azt mutatja meg, hogy a CPU-idő hány százaléka fordítódik a folyamatok utasításainak a végrehajtására,
  - (ΣCPU-idő – Σ(henyélés+admin)) / ΣCPU-idő x 100 = általában 40-90 [%].
- Átbocsátó képesség:
  - időegység alatt elvégzett munkák száma,
  - elvégzett munkák száma / időegység = a feladat jellegétől függ, pl.: 1/h, 10/sec.

Ütemezési algoritmusok összehasonlítása

- Körülfordulási idő:
  - egy munkára fordított teljes idő,
  - Σ(végrehajtási + várakozási) idő,
  - csak a várakozási idő függ az ütemezéstől.
- Várakozási idő:
  - egy munka teljes várakozási ideje,
  - Σ(hosszú távú ütemezésig eltelt + futásra kész + + várakozó + felfüggesztett) idő.
- Válaszidő:
  - a felhasználó(k) felé történő reagálás ideje.

Ütemezési algoritmusokkal szembeni követelmények

- Valamelyik paraméter (az előbb említettek közül) szempontjából optimális.
- Korrekt, azaz minden folyamat kapjon bizonyos CPU-időt.
- Biztosítson prioritásokat egyes folyamatoknak.
- Kerülje a folyamatok kiéheztetését.
- Legyen megjósolható viselkedésű, azaz terheléstől függetlenül becsülhetőek legyenek a rendszer paraméterei (pl. körülfordulási idő).

Ütemezési algoritmusokkal szembeni követelmények

- Részesítse előnyben a kihasználatlan erőforrást igénylő folyamatokat.
- Részesítse előnyben a fontos erőforrásokat használó folyamatokat.
- Legyen maximális az átbocsátó képessége.
- Növekvő terhelés hatására a rendszer teljesítőképessége fokozatosan csökkenjen ne omoljon össze.

Megoldás keresés

Az előbbiek mindegyike teljesíthetetlen, sőt ellentmondásos, ezért fontos a cél-orientált rendszer-tervezés.

Egyszerű ütemezési algoritmusok

- Legrégebben várakozó (FCFS):
  - nem preemptív,
  - a várakozási sor legelső folyamatát futtatja,
  - nagy az átlagos várakozási idő (konvoj hatás).
- Körbeforgó (RR):
  - preemptív,
  - az időosztásos rendszerek alapja,
  - minden folyamat amikor futni kezd kap egy időszeletet,

Egyszerű ütemezési algoritmusok

- ha a folyamat CPU-lökete hosszabb ennél, akkor az időszelet végén az ütemező elveszi a folyamattól a CPU-t, és a futásra kész sor végére állítja,
- ha rövidebb, akkor a löket végén újraütemezés történik, és a futó folyamat időszelete újraindul,
- ha az időszelet hosszú: FCFS,
- ha rövid: sok környezetváltás,
- ezért fontos az időszelet-hossznak a helyes megválasztása, így: a CPU-löketek kb. 80%-a legyen rövidebb az időszeletnél.

Prioritásos ütemezési algoritmusok

- A futásra kész folyamatok mindegyike kap egy számot, ami a futás kívánatos sorrendjére utal.
- Ez az ún. prioritás.
- Ütemezéskor, a legnagyobb prioritású kap futási engedélyt.
- Az ütemezés lehet preemptív és nem preemptív is.
- Hátrányuk a kiéheztetés veszélye.

Prioritásos ütemezési algoritmusok

- Prioritás típusok:
  - külső: a folyamat kéri, ill. az operátor állítja be,
  - belső: az OPR állítja be,
  - statikus: azaz időben állandó,
  - dinamikus: az OPR módosíthatja.

A prioritás meghatározása a CPU--löketidő alapján

- Löketidő meghatározása:
  - átlagos löketidő "bevallása” (büntetés bevezetése),
  - löketidő becslése (exponenciális) átlagolással,
  - előző futásból ismert.
- Algoritmusok:
  - legrövidebb löketidejű (SJF),
  - legrövidebb hátralevő idejű (SRTF),
  - legjobb válaszarány (HRR).

Legrövidebb löketidejű (SJF)

- Nem preemptív.
- A legrövidebb becsült löketidejű folyamatot választja.
- Nincs konvoj hatás.
- Optimális a körülfordulási és így az átlagos várakozási idő is.

Legrövidebb hátralévő idejű (SRTF)

- Az előző (SJF) algoritmus preemptív változata.
- Ha a következő futásra kész folyamat löketideje rövidebb, mint az éppen futó hátralévője, akkor folyamatot cserél.
- Ilyenkor a környezetváltással is számolni kell!

Legjobb válaszarány (HRR)

- Az SJF algoritmus kiéheztetést elkerülő változata.
- A kicsi prioritású, régóta várakozó folyamatok prioritását fokozatosan növeli, ez az ún. öregítés.
- A folyamat kiválasztásnál a löketidő mellett a várakozási időt is figyelembe veszi.
- A prioritás alapjául a következő képlet szolgál:
  - (löketidő + k * várakozási idő) / löketidő,
  - a ”k” egy tapasztalati konstans.

Többszintű ütemezési algoritmusok

- A futásra kész folyamatok több sorban várakoznak.
- Minden sor prioritásos.
- Az ütemező a kisebb prioritású sorból csak akkor választ ki folyamatot, ha a magasabb prioritású sor(ok) már üresek.
- A sorokon belül eltérő kiválasztó algoritmusok tényked(het)nek.

Többszintű ütemezési algoritmusok

- Statikus többszintű sorok (SFQ):
  - a folyamatok az elinduláskor egy állandó prioritást kapnak - ez alapján sorolódnak sorokba,
  - egy lehetséges besorolás:
    - rendszerfolyamatok (rendszer működtetés),
    - interaktív folyamatok (elfogadható válaszidők biztosítása),
    - interaktív szövegszerkesztők (kevésbé kritikus válaszidők),
    - kötegelt feldolgozás (ha ”van idő” akkor fut),
    - rendszerstatisztikák (nincs kihatás a pillanatnyi működésre),
  - hátrányuk a kiéheztetés (pl. 1973-ban kikapcsolt IBM gépen volt egy 1967-ben elindított és még várakozó folyamat!).

Többszintű ütemezési algoritmusok

- Visszacsatolt többszintű sorok (MFQ):
  - cél a kiéheztetés elkerülése (különböző módszerek),
  - változó - dinamikus - prioritású folyamatok, ezáltal sorváltás valósítható meg,
  - rövid CPU-löketidejű folyamatok kapnak magasabb prioritást,
  - a magasabb prioritású sorokon belül általában preemptív algoritmusok (pl. RR) dolgoznak,
  - míg a legalsóban egy egyszerű FCFS,
  - induláskor a folyamatok a legmagasabb besorolást kapják,

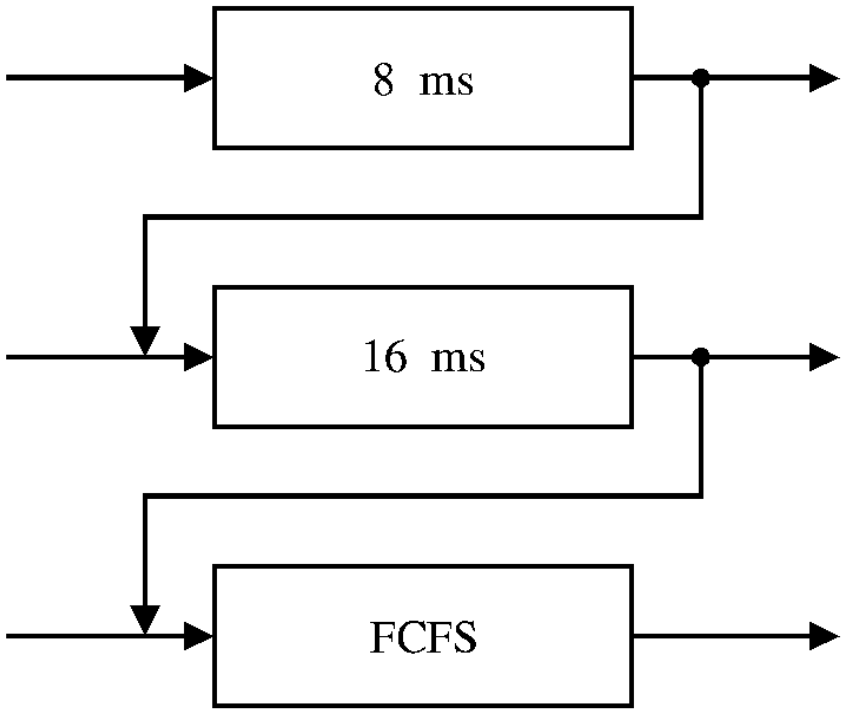
Többszintű ütemezési algoritmusok

– ha túllépik az időkorlátot eggyel alacsonyabb prioritású, de nagyobb időszeletű sorba kerülnek át,
– időről időre felül kell vizsgálni a besorolásokat (pl. átlagos löketidő mérése alapján), és ha kell, a prioritást növelni,
– így elkerülhető, hogy egy folyamat egy egyszeri időszelet túllépés miatt alacsonyabb sorban vesztegeljen,
– továbbá, a régóta várakozó folyamatok prioritását az OPR megnövelheti.

Többszintű ütemezési algoritmusok

– osztályokba sorolás a következő szempontok szerint:
  - hány sorban várakoznak futásra kész folyamatok,
  - milyen algoritmusokat használ a sorokon belül,
  - hogyan kaphat nagyobb prioritást egy folyamat,
  - mikor csökken egy folyamat prioritása,
  - melyik sorba lépnek be az elindított folyamatok.

Visszacsatolt többszintű sorok ütemezési sémája



Ütemezési algoritmusok ”jóságának” értékelése

- Analitikus vizsgálat:
  - determinisztikus (ok-okozati) modell, inkább csak elemzésre alkalmas (a rendszerek terhelése számos véletlenszerű paraméterrel jellemezhető),
  - sorban-állás elmélet és sztochasztikus (statisztikai valószínűség) modell, előre összeállított adatokból dolgozva, csak szűk körben lesz igaz.
- Szimuláció:
  - korábbi mérések alapján, meghatározott eloszlású véletlen-számokkal és konkrét mért értékekkel való leírás,

Ütemezési algoritmusok ”jóságának” értékelése

  - az eredmény nagyban függ, az adatok helyességétől és a szimulációk számától.
- Implementáció:
  - valós környezetben való teljesítmény mérés,
  - legmegbízhatóbb módszer, csak igen költséges, ezért nem igazán használják.

Többprocesszoros ütemezés

- Heterogén rendszerek esetén a folyamatok, csak a nekik megfelelő utasításkészletű processzoron futhatnak.
- Szorosan csatolt (közös óra és memória) homogén rendszerek esetén bármelyik, éppen szabad processzoron elindulhatnak, a közös várakozási sorban lévő folyamatok.

Többprocesszoros ütemezés

- Közös várakozási sor(ok) esetén a következő ütemezési megoldások alkalmazhatók:
  - szimmetrikus:
    - minden CPU külön ütemezőt futtat,
    - a sor(ok) hibamentes, megosztott használata érdekében, biztosítani kell a hozzáférésre vonatkozó kölcsönös kizárást!,

Többprocesszoros ütemezés

– aszimmetrikus:
  - egyetlen ütemező fut, egy kiválasztott processzoron,
  - ez osztja szét a feladatokat a szabad CPU-k között,
  - ez a megoldás egyszerűbb adathozzáférést eredményez.