11.5 Disk scheduling

Tuesday, April 28, 2020 11:08 PM

Som følge av Moores lov har hastigheten til prosessorer (og hovedminnet) for lengst overgått hastigheten for diskaksess. Derfor er det enda viktigere å ha en effektiv tidsstyring av disk.

Aksessformen på disk er å enten søke frem til sporet, rotere fram til blokken, eller overføre hele blokken. Overføringstiden fra disk er avhenger av rotasjonshastigheten til disken på følgende måte:

$$T = \frac{b}{rN}$$

where

T = transfer time,
b = number of bytes to be transferred,
N = number of bytes on a track, and
r = rotation speed, in revolutions per second.

Dermed er den totale gjennomsnittlige aksesstiden:

$$T_a = T_s + \frac{1}{2r} + \frac{b}{rN}$$

where T_s is the average seek time.

Tidsstyringsdisipliner for disk

First-In-First-Out – FIFO
Den enkleste formen for tidsstyring som har fordelen at den er rettferdig. FIFO har tilnærmet samme ytelse som tilfeldig tidsstyring, som ikke er veldig bra.

Prioritet – PRI
Et system som benytter prioritet er ikke designet for å utnytte disk, men heller for å imøtekomme andre krav fra OS-et. Ofte prioriterer man kortere jobber, for at de skal kunne gå gjennom systemet raskt og gi god interaktiv responstid.

Last-In-First-Out – LIFO
LIFO viser seg å være bedre til disk, ettersom det har tendenser til mindre søking etter riktig spor (utnytter lokalitet). Problemet er potensiell starvation.

Shortest-Service-Time-First – SSTF
Valget i SSTF faller på disken som krever minst bevegelse av diskarmen fra den nåværende posisjonen. Dette gir god ressursutnyttelse. Gir altså et lav snitt men også en høy varians på responstid i forbindelse med diskaksesser.

SCAN
SCAN-algoritmen fungerer ganske likt som en heis. Diskarmen beveger seg kun én vei inntil den har tilfredsstilt alle forespørsler i den retningen. Deretter snus tjeneste-retningen til motsatt retning og scanningen gjentas. Denne oppfører seg nesten likt som SSTF, men utnytter ikke lokalitet like bra.

C-SCAN
For C-SCAN er scanningen kun rettet i en retning. Etter en fullført scan returnerer den dermed til andre side av disken for å gjennomføre scanningen på nytt. Dette reduserer maksimal forsinkelse for nye forespørsler. Gir lavere varians enn SSTF men har også høyere snitt på responstiden.

N-step-SCAN og FSCAN
N-step-SCAN segmenterer diskforespørsel-køen inn i subsekvenser med lengde N. Subsekvensene prosesseres én av gangen ved bruk av SCAN. Når en kø prosesseres legges nye forespørsler i en annen kø. Dersom mindre enn N forespørsler er tilgjengelig på slutten av scannen, så prosesseres alle ved neste Scan. Denne har samme ytelse som SCAN for store verdier av N og garanterer tjeneste. FSCAN er en rutine som bruker to subsekvenser. Når en scan begynner er alle forespørslene i den ene køen, mens den andre er tom. Under scanningen legges alle nye forespørsler i den tomme køen. Dermed må de gamle forespørslene vente på at de gamle prosesseres før de kan betjenes.