4.2 Types of threads

Wednesday, April 8, 2020 3:39 PM

Det er to hovedkategorier av trådimplementasjon: bruker-nivå tråding (ULT) og kjerne-nivå tråder (KLT, *kernel-level threads*).

Brukernivå-tråder

I et strengt ULT-miljø, så gjøres alt arbeid av trådhåndtering av applikasjonen, og kjernen er ikke klar over eksistensen av tråder. Enhver applikasjon kan programmeres til å være flertrådet ved bruk av et tråd-bibliotek. I utgangspunktet starter applikasjonen med en enkel tråd og starter kjøringen i den tråden. Denne applikasjonen og dens tråd er tildelt til en enkel prosess som håndteres av kernel. Tråd-biblioteket muliggjør for at ny tråder kan opprettes i programmet, og alt av håndtering tilknyttet trådene gjøres ved bruk av dette biblioteket. Alle disse aktivitetene gjøres I brukerområdet og I en enkel prosess. Dermed er kernel ikke klar over aktivitetene.

Fordeler ved bruk av ULT istedenfor KLT:

1. Trådbytting krever ikke kjerne-modus pribileger fordi alt av tråd-håndteringsdatastrukturer befinner seg i brukers adresserom i en enkel prosess. Dermed bytter ikke prosessen til kjerne-modus for trådhåndtering. Man unngår altså overhead med å bytte frem og tilbake fra bruker-modus til kjerne-modus.
2. Planlegging kan være spesifikk for applikasjoner. En applikasjon kan dra fordel av en enkel *round-robin* planleggingsalgoritme, mens en annen kan dra fordel av av en prioritets-basert planleggingsalgoritme. Planleggingsalgoritmen kan sammensettes til applikasjonen uten å forstyrre den underliggende OS-planleggeren.
3. ULT-er kan kjøre på ethvert OS. Ingen endringer er påkrevd til den underliggende kernel for å støtte ULT-er. Tråd-biblioteket er et sett med applikasjonsnivå-funksjoner delt av alle applikasjoner.

Ulemper ved ULT istedenfor KLT:

1. I et typisk OS er det mange system-kall som blokkerer. Men når et ULT executer et system-kall, er ikke bare den tråden blokkert, men alle trådene i den samme prosessen blir også blokkert.
2. I en streng ULT-strategi, så kan ikke en flertrådet applikasjon dra fordel av multiprosessering. En kernel tildeler en prosess til kun en prosessor om gangen. Derfor kan bare en enkel-tråd i en prosess executes om gangen. Selv om denne multiprogrammeringen kan resultere i en betydelig raskere applikasjon, så er det applikasjoner som heller skulle hatt muligheten til å execute deler av koden samtidig.

Det er løsninger rundt disse problemene, for eksempel å implementere applikasjonen ved bruk av flere prosesser istedenfor flertråding. Men denne tilnærmingen fjerner hovedfordelen ved tråding: Hvert skifte blir et prosess-skifte istedenfor et tråd-skifte, som gir mye mer overhead.

En annen løsningen for å håndtere blokking av tråder er såkalt *jacketing*. Konseptet bak jacketing er å konvertere et blokkeringssystem-kall til et ikke-blokkeringssystem-kall. Istedenfor å direkte kalle en system I/O-rutine, kan en tråd istedenfor kalle en applikasjonsnivå I/O-jacket-rutine. I denne jacket-rutinen er det kode for å sjekke om I/O-enheten er opptatt eller ikke. Hvis den er opptatt, så går tråden inn i blokkert tilstand og gir kontroll til andre tråder.

Kjernenivå-tråder (Kernel-level threads)

I et strengt KLT-miljø blir alt av trådhåndtering utført av kernel. Det er ikke noe trådhåndterings-kode i applikasjonsnivået, kun et API til kjernetrådfasilitetene. Windows er et eksempel på dette.
Kjernen håndterer kontekstinformasjon for prosessen som en helhet og for individuele tråder i den prosessen. Planleggingen gjort av kjernen gjøres på et tråd-basis. Denne tilnærmingen unngår da de to ulempene ved ULT. Kernelen kan samtidig planlegge flere tråder fra samme prosess på flere prosessor. Hvis en tråd i en prosess er blokkert, så kan kernelen planlegge en annen tråd av samme prosess. En annen fordel ved bruk av KLT er at kernel-rutiner selv kan bli flertrådet. Hovedulempen ved KLT kontra ULT er at overføringen av kontroll fra en tråd til en annen i samme prosess krever et modusskifte til kjerne-modus.

Som vi ser er det altså en betydelig hastighetsøkning ved å bruke KLT multitrpding sammenlignet med enkel-trådede prosesser, men det er også en betydelig hastighetsøkning ved å bruke ULT. Om denne potensielle hastighetsøkningen realiseres avhenger av applikasjonene involvert. Hvis mange av tråd-byttene i en applikasjon krever aksess til kjerne-modus så kan et ULT-basert skjema bli utilstrekkelig i henhold til ytelse.

Kombinerte tilnærminger

Det finnes noen operativsystemer som kombinerer ULT/KLT. I et kombinert system skjer trådopprettelse i brukerrommet, det samme gjelder planleggingen og synkroniseringen av trådene i applikasjonene. Flere av ULT-ene fra en enkel applikasjon er så overført til et antall KLT-er.

I den kombinerte versjonen kan flere tråder i samme applikasjon kjøre i parallell på flere prosessorer, og et blokkeringssystem-kall trenger ikke å blokkere hele prosessen. Hvis denne tilnærmingen implementeres godt minimeres ulempene ved ULT og KLT samtidig som man får utnyttet de kombinerte fordelene.