Konkurentno programiranje - koncept napredovanja

Stanić Ivan
mr16016@matf.bg.ac.rs

19. novembar 2018.

1 Uvod

Koncept napredovanja govori da ukoliko neki događaj treba da se desi (npr. završetak rada programa) da će se on i desiti u nekom trenutku, odnosno da se stalno pravi nekakav progres, tj. napredak [4].

2 Uzroci prestanka napredovanja

Uzajamno blokiranje (deadlock) je najčešći uzrok prestanka progresa. Dešava se kada dva ili više zadataka čekaju jedni druge da završe, pa nijedan ne napreduje. Uslovi neophodni za pojavu uzajamnog blokiranja su sledeći:

- Mutex: Makar jedan resurs mora se nalaziti u nedeljivom prostoru. Dva procesa ne mogu pristupiti resursu u istom vremenskom intervalu.
- Zadržavanje resursa: Proces trenutno drži makar jedan resurs i neophodni su mu drugi resursi koje drže drugi procesi.
- Nemogućnost prisvajanja: Resurs može biti oslobođen samo dobrovoljno od strane resursa koji ga koristi.
- Kružno čekanje: Procesu A je neophodan resurs koji koristi proces B, dok resurs B čeka da A oslobodi svoj resurs. Opštije, može da postoje niz procesa P1,P2,....,Pn gde Pi-ti proces čeka resurs zadržan od strane procesa Pi+1 i Pn-ti proces čeka resurs koji drži proces P1.

Ove uslove je prvi put zapisao Edvard Kofman u svom članku 1971. pa su poznati kao Kofmanovi uslovi [3].

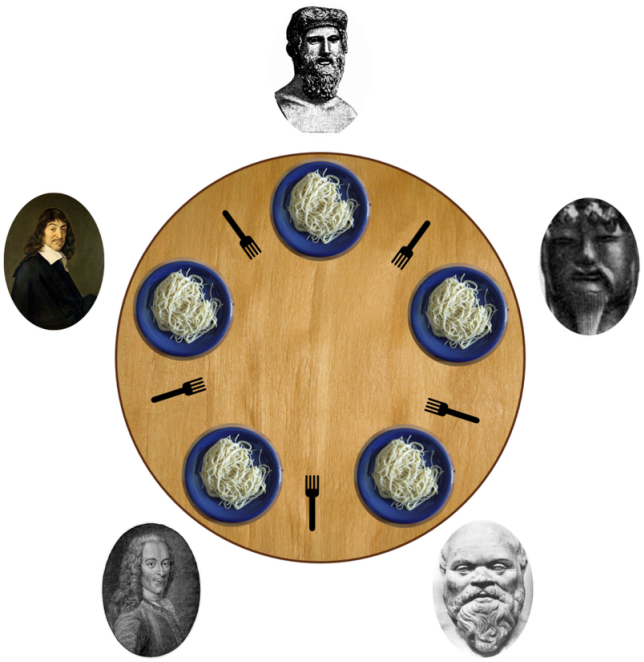
Problemi koji takođe mogu nastati su izgladnjavanje resursa (resource starvation) i živo blokiranje (livelock).

Izgladnjavanje predstavlja situaciju kada zadatak ne može da pristupi deljenim resursima, pa samim tim ne može da napreduje. To se dešava kada su deljeni resursi nedostupni u dužim intervalima jer ih koriste "pohlepni"(greedy) zadaci. Na primer, ako imamo metod koji u datom trenutku može da koristi samo jedna nit, a kojem treba dugo vremena da se izvrši, nastaće problem ukoliko jedna nit često koristi taj metod, a on je neophodan i drugim nitima [1].

Ponašanje zadatka često može da bude uslovljeno ponašanjem drugog zadatka. Ako i drugi zadatak zavisi od ponašanja prvog, može doći do pojave živog blokiranja. Kao i kod uzajamnog blokiranja, zadaci ne napreduju. Međutim, ovde zadaci nisu zapravo blokirani, već ih ta međusobna uslovljenost sprečava da napreduju. Primer koji se često koristi za ilustraciju ovog problema je situacija u kojoj dva čoveka pokušavaju da mimoiđu jedan drugog u hodniku. Videvši osobu B kako joj prilazi, osoba A se pomera u levu stranu da je propusti. Istovremeno, osoba B se pomera desno, takođe u želji da propusti osobu A. U tom trenutku shvate da se blokiraju, pa se obe osobe pomere u suprotne strane i opet dođu u nepovoljan položaj [1].

3 Problem filozofa koji večeraju

Problem filozofa koji večeraju je poznati problem u oblasti konkurentnog programiranja. Na njemu se lepo mogu ilustrovati uzajamno i živo blokiranje.



Slika 1: Filozofi sa stolom

Kao što vidimo na slici 1, pet filozofa sede za kružnim stolom i ispred svakog se nalaze po dve viljuške. Svakom filozofu su neophodne dve viljuške da bi mogao da večera. Problem je kako postići da svi filozofi večeraju [2]?

Uzajamno blokiranje predstavlja situaciju u kojoj svaki od filozofa uzme viljušku u istom trenutku i nijedan ne planira da ostavi svoju već čeka da mu neko drugi da njegovu viljušku. U želji da razrešimo ovaj problem dajemo filozofima vremenski limit za zadržavanje viljuški, kao i minimalno vreme koje je potrebno da prođe da ponovo uzmu viljušku. Problem koji sada može da se desi je da svi filozofi istovremeno uzmu viljuške, drže ih sve dok mogu, a

zatim ih ponovo uzmu čim se prilika uspostavi, opet istovremeno. Takav ciklus bi takođe mogao da se ponavlja u nedogled. Ovakav scenario predstavlja živo blokiranje.

Jedan od načina da se izbegnu prethodni problemi je numerisanje viljuški, tj. hijerarhija resursa. Ako numerišemo viljuške brojevima od 1 do 5 i kažemo filozofima da moraju prvo uzeti viljušku sa manjim pa tek onda sa većim brojem, u početnom trenutku četiri filozofa će uzeti viljuške dok će onaj koji bi u slučaju deadlocka uzeo viljušku sa brojem 5 čekati da se oslobodi viljuška sa manjim rednim brojem. Sada će tu viljušku uzeti najbliži filozof i deadlock će se izbeći. Loše kod ovakvog pristupa je što je neefikasan. Npr, ako imate viljuške sa rednim brojevima 3 i 5, da bi ste uzeli viljušku sa rednim brojem 4 trebaće vam 5 koraka.

Koncept rada semafora se takođe lepo ilustruje ovim problemom. Ako imamo 5 viljuški za stolom, ne želimo da u bilo kom trenutku za istim bude više od 4 filozofa. To se može obezbediti postavljanjem konobara (semafor) koji će se starati da filozof ne može da se priključi stolu gde već sede 4 filozofa. Rad semafora može sprečiti uzajamno, ali nije dobra odbrana od pojave živog blokiranja.

Literatura

[1] Starvation and Livelock. link: https://docs.oracle.com/javase/tutorial/essential/concurrency/starvelive.html.

[2] The Dining Philosophers Problem With Ron Swanson. link: http://adit.io/posts/2013-05-11-The-Dining-Philosophers-Problem-With-Ron-Swanson.html.

[3] Jr. Coffman, Edward G. *System Deadlocks.*

[4] Slajdovi profesorke Milene Vujoševic Janičić. Uvod. Kurs dizajn programskih jezika, link: http://www.programskijezici.matf.bg.ac.rs/dpj/2017/predavanja/03/konkurentno_programiranje.pdf.