8. Logički satovi

Na ovim vježbama ćemo se baviti logičkim satovima. Odgovorit ćemo na pitanja zašto su nam potrebni, kakve imamo te ćemo ih implementirati uz pomoć Akka.NET-a.

8.1 Stanja i događaji

Konfiguracija raspodijeljenog sustava se sastoji od kompozicije lokalnih stanja svakog procesa i poruka koje su u postupku slanja, tj. nisu još stigle na svoje odredište. Tranzicija od jedne do druge konfiguracije distribuiranog sustava je vezana uz događaj (*event*) u nekom procesu ili u slučaju sinkrone komunikacije - dva događaja u dva procesa. Proces može izvršiti tri tipa događaja:

- interni, engl. *internal*
- slanje poruke, engl. *send*
- primanje poruke, engl. *receive*

Interni događaj je vezan uz mijenjanje lokalnog stanja procesa, a tipični primjeri su čitanje ili pisanje u varijablu.

Proces nazivamo inicijatorom ako je prvi događaj interni događaj ili slanje poruke. Drugim riječima, inicijator je proces koji može započeti izvršavati događaje bez utjecaja drugih procesa.

8.2 Causal order

Događaji koji se u distribuiranom sustavu pojavljuju u različitim procesima su neovisni jedan o drugome, što znači da se mogu pojaviti u bilo kojem redosljedu. *Causal order* je odnos između događaja prilikom izvršavanja distribuiranog sustava. Označuje se binarnom relacijom $\prec$ s time da a $\prec$ b označuje da se a dogodio prije događaja b.

Casual order za izvršavanje je najmanja relacija takva da vrijedi:

- ako su a i b događaji na istom procesu i a se dogodio prije b onda vrijedi $a \prec b$;
- ako je a događaj slanja poruke, a b pripadajući odgovor primanja poruke, onda vrijedi $a \prec b$;
- ako je $a \prec b$ i $b \prec c$ onda $a \prec c$.

Istovremeni događaji (*concurrent events*) se ne mogu urediti na ovakav način.

8.3 Logički satovi

Tradicionalni pojam vremena ne funkcionira kod distribuiranih sustava. Problem je održavanja zajedničkog globalnog sata u različitim procesima distribuiranog sustava. Na svu sreću, kod dosta primjena nas ne zanima točno vrijeme, već samo redosljed kojim su se događaji pojavili. Logički sat C mapira pojavljivanje događaja u parcijalno uređen skup na način da

$$a \prec b \implies \mathrm{C}(a) < \mathrm{C}(b)$$

8.3.1 Lamportov logički sat

Lamportov sat je jednostavan algoritam za određivanje redosljeda događaja u distribuiranom sustavu. Svakom događaju a dodjeljuje duljinu k najduljeg uzročnog lanca događaja. Vrijednosti koje Lamportov sat dodjeljuje se računaju tokom izvršavanja sustava.

Algoritam:

- Ako je a interni događaj ili događaj slanja, onda je potrebno vrijednosti lamportovog sata uvećati za 1, $LC(a) = k+1$ te se ta vrijednost šalje skupa s porukom;
- U slučaju da se događaj a primanje poruke, a b odgovarajući događaj slanja s drugog procesa, onda $LC(a) = \max\{k, LC(a)\} + 1$.

Ranije smo naveli da je interni događaj itanje iz varijable ili upisivanje u istu. Po izvoru [2] interni događaj može biti i poziv procedure. Po izvoru [3] lamportov sat se inkrementira svaki put kada proces izvrši nekakav posao.

Dakle, lamportov sat je predstavljen običnim brojačem kojeg posjeduje svaki proces.

Problem s lamportovim satom je taj što istovremeni događaji mogu imati istu vrijednost lamportov sata. Pogledaj sliku 8.2.

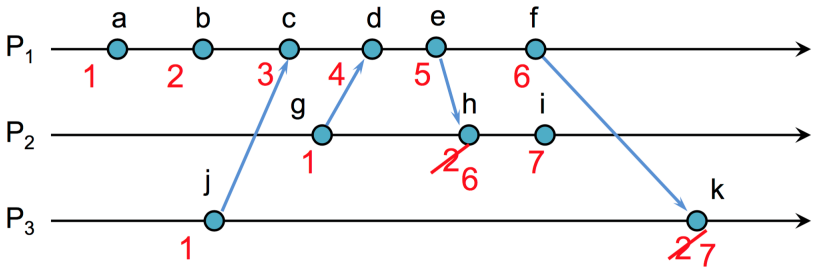

Figure 8.1: Događaji označeni lamportovim satom

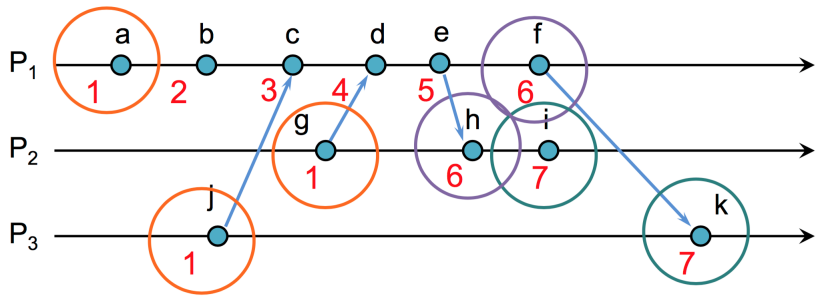

Figure 8.2: Događaji na različitim procesima imaju istu vrijednost lamportovog sata

Jedan od načina kojim se može pristupiti ovome problemu jest da se koristi uređeni par koji se sastoji od vrijednosti lamportovog sata i identifikatora procesa, (T_i, i).

U tom slučaju vrijedi $(T_i, i) < (T_j, j)$ ako i samo ako:

- $T_i < T_j$ ili
- $T_i = T_j$ i $i < j$.

Problem s ovim pristupom je taj što ne možemo detektirati povezane događaje. Ako imamo $L(e) < L(e')$ ne možemo zaključiti $e \to e'$.

8.4 Vektorski sat

Vektorski sat je algoritam za generiranje parcijalnog uređenja među događajima u distribuiranom sustavu. Vektorski sat je predstavljen nizom od N elemenata gdje svaki element niza predstavlja vrijednost logičkog sata za odgovarajući proces. Naravno, u implementaciji ovo nije nužno niz. Ovaj niz od N elemenata nazivamo vektor.

Algoritam vektorskog sata:

- U početku za svaki element vektora elemenata inicijaliziramo sat na početnu vrijednost koja je minimalna moguća, najčešće je to 0.
- Kada se dogodi interni događaj, proces i, uveća $V[i] = V[i] + 1$.
- Kada god proces šalje poruku, sa njom uključi cijeli vektorski sat.
- Proces koji prima vektorski sat postavi sve elemnte vektorskog sata na: $\max\{local, received\}$. Zatim proces uveća svoju poziciju u vektoru za jedan.

Vektorska vremena uspoređujemo na sljedeći način:

- $V = V'$ ako i samo ako $V[i] = V'[i]$ za svaki $i = 1...N$
- $V \leq V'$ ako i samo ako $V[i] \leq V'[i]$ za svaki $i = 1...N$

Uz pomoć vektorskog sata za bilo koja dva događaja možemo reći:

- ako $e \rightarrow e'$ onda $V(e) < V(e')$ - ovo je isto kao kod Lamportov sata;
- ako $V(e) < V(e')$ onda $e \rightarrow e'$

Za dva događaja kažemo da se izvšaravju istovremeno ako ne možemo odrediti ni $V(e) \leq V(e')$ ni $V(e') \leq V(e)$.

8.4.1 Poopćenje vektorskog sata

Već smo spomenuli da se vector ne mora odnositi na niz, već može biti bilo koja struktura podataka koja može zapamtiti veći broj elemenata.

Vector u tom slučaju može izgledati kao lista uređenih parova (ili Dictionary, ili HashMap) $< P_1, T_{P1} >, < P_2, T_{P2} >, < T_3, T_{P3} > ...$, gdje su $P_1, P_2, P_3 ...$ identifikatori procesa, a T_{P1}, T_{P2}, T_{P3} vrijednosti logičkih satova procesa.

Svaki proces može imati samo parcijalno znanje o ostalim procesima. U tom slučaju vrijede sljedeća svojstva:

- Nove oznake za primljenu poruku:
  - Usporedi sve skupove ID-eva procesa: postavi najveću vrijednost;
  - Ako je dobiven par čiji ID procesa se ne nalazi u strukturi, onda se isti dodaje u strukturu.
- Za odnos "dogodilo se prije":
  - Barem jedan skup ID-eva procesa mora mora biti zajednički u dvije oznake vremena;
  - Pridruži sve odgovarajuće skupove $< P, T >$: A: $< P_i, T_a >$, B: $< P_i, T_b >$;
  - Ako $T_a \leq T_b$ za sve zajedničke procese P, onda $A \rightarrow B$.

8.5 Zadatci

1. Napravite jednostavnu aplikaciju Klijent - Server u kojoj će server klijentu nuditi matematičke operacije zbrajanja, oduzimanja, djeljenja, množenja i potenciranja. Implementirajte lamportov logički sat. Testirajte aplikaciju pokretanjem više klijenata i ispisom vrijednosti logičkih satova.
2. Izmijenite aplikaciju iz prethodnog zadatka tako da koristi vektorski sat.
3. Dovršite LamportServer i LamportKlijent aplikacije. Aplikacije su trenutno nedovršene i imaju nekoliko nedostataka: Lamportov sat se ne ispisuje, klijent ne može primiti rezultat, server podržava samo zbrajanje. Pretvorite lamport klijent u windows forms aplikaciju te ju dovršite. NB: po spojenom klijentu je jedan worker koji pita distributora koji kod distributora provjerava vrijednost lamportovog sata.
4. Izmijenite prethodni zadatak tako da koristi vektorski sat.
5. Napravite jednostavnu chat aplikaciju klijent - server na način da lamportovim satom kontrolirate da se poruke klijentima ispisuju redosljedom kojim bi trebale. Simulirajte zastoj na način da za neki slučajan broj pauzirate nit prije slanja poruke.
6. Izmijenite Loggy program na način da je implementiran vektorski sat.

8.6 Izvori

1. Distributed Algorithms - An intuitive approach - Wan Fokkink
2. http://www.cs.fsu.edu/ xyuan/cop5611/lecture5.html, 11.04.2016., 21:38
3. http://book.mixu.net/distsys/single-page.html#time, 12.04.2016., 18:25 (Preporuka, Poglavlje 3)
4. https://www.cs.rutgers.edu/ pxk/417/notes/content/06-logical-clocks-slides.pdf , 12.04.2016, 22:37