Distribuovaný broadcast, synchronizace v distribuovaných systémech

Z FITwiki

Obsah

Distribuovaný broadcast

- pojmy:
  - m - zpráva z množiny možných zpráv, každá zpráva zahrnuje odesílatele (sender(m)) a sekvenční číslo (seq(m))
  - funkce bcast(m) a deliver(m)
  - např. sender(m)=p a seq(m)=i znamená, že m je i-tá zpráva poslaná procesem p

Základní vlastnosti broadcastu

- Validity = Pokud korektní proces odešle broadcast, pak všechny korektní procesy tento broadcast obdrží (dříve či později)
- Agreement = Pokud korektní proces obdržel broadcast, potom všechny korektní procesy obdrží broadcast také (dříve či později)
- Integrity = Každý korektní procs obdrží broadcast maximálně jednou (nedojde k zacyklení)

Pokud platí všechny 3 vlastnosti, jde o **spolehlivý broadcast**.

Další vlastnosti broadcastu

- FIFO Order = pokud nějaký proces broadcastuje zprávu m před zprávou n, potom žádný korektní proces nepřijmne zprávu n, aniž by nejdříve přijal zprávu m (tj. zprávy jsou doručovány ve stejném pořadí jak byly odeslány)
- Causal Order = FIFO Order, ale v rámci všech procesů v systému (tj. Pokud máme 2 procesy a každý vysílá 2 broadcasty A (m,n), B(o,p) pak žádný proces nepřijme v pořadí (m,n,p,o) nebo (o,p,n,m) nebo (p,n,m,o)... Jediné správné je (m,n,o,p) nebo (o,p,m,n) nebo (m,o,p,n) nebo (m,o,n,p))
- Total Order = doručování přesně ve stejném pořadí, jak byly broadcasty odeslány

Klasifikace broadcastů

- Reliable (RBCAST) = Validity + Agreement + Integrity
- FIFO (FBCAST) = reliable + FIFO Order
- Causal (CBCAST) = reliable + Causal Order
- Atomic (ABCAST) = reliable + Total Order

Algoritmus pro spolehlivý broadcast (RBCAST)

```
Odesílatel o:
bcast(R,m): //R = reliable bcast, m = zprava
 pridej do m odesilatele(m) a sekvencni_cislo(m);
 send(m) vsem sousedum, vcetne sebe;
```

```
Prijemce p:
deliver(R,m):
 pri receive(m) do
  if p jeste neprijal tento broadcast then
   if sender(m) != p then send(m) vsem sousedum;
   deliver(R,m)
  endif;
 enddo;
```

Synchronizace v distribuovaných systémech

V distribuovaných systémech není údaj o globálním čase (i kdyby byl, nejsme schopní kontrolovat zpoždění mezi doručením zpráv). Tedy každý uzel má vlastní hodiny. Řešíme problém vzájemného vyloučení pří požadavku více klientů v distribuovaném prostředí na sdílený zdroj. 2 možnosti:

- Lamportův algoritmus + optimalizace Ricart-Agrawala algoritmus (založeno na časových razítkách)
- Raymondův algoritmus (založen na tokenech)

Lamportův algoritmus

Lamportův algoritmus pro přístup do kritické sekce používá Lamportovy logické hodiny.

Lamportovy logické hodiny

- Není zde žádná souvislost s fyzickými hodinami, ve skutečnosti jde o časová razítka.
- Založeno na relaci mezi dvěma událostmi stalo_se_dříve(a,b).

- Vychází z pravidel fyzické kauzality (než něco přijmeme, musíme to odeslat)

Synchronizace hodin mezi 2 procesy probíhá takto:

1. Každý proces má vlastní čítač běžící různou rychlostí
2. Proces 1 chce odeslat zprávu procesu 2, společně se zprávou vloží hodnotu svého čítače
3. Proces 2 přijme zprávu od procesu 1 a zjistí časové razítko
4. Proces 2 porovná hodnoto svého čítače s hodnotou extrahovanou z časového razítka procesu 1
5. Proces 2 si nastaví hodnotu svého čítače na vyšší z těchto dvou hodnot a inkrementuje hodnotu
6. V tuto chvíli jsou procesy 1 a 2 synchronizovány (ale jen na chvíli, protože jejich čítače běží obecně různě rychle)

Lamportov algoritmus pre pristup do kriticke sekcie

wiki (http://en.wikipedia.org/wiki/Lamport%27s_Distributed_Mutual_Exclusion_Algorithm)

Proces vyšle žádost, a čeká až dorazí odpovědi od všech ostatních, a všechny žádosti v jeho frontě mají vyšší časovou značku.

- p posílá žádost Mp se svým timestampem
- přijetí žádosti od i: zapamatuje si žádost, pošle ACK s vlastním timestampem
- když dostane od někoho ACK, přidá si ho ke svému požadavku
- do kritické sekce proces vstoupí, když
    1. od všech ostatních dostal ACK
    2. a zároveň neví o žádném starším požadavku
- když skončí s kritickou sekcí, pošle ostatním release
- po přijetí release si proces vymaže k němu patřící žádost (a někdo další pak na základě toho může vlézt do kritické sekce)

R-A rozsirenie: zlucenie release and reply odpovede do jednej (delayed reply)

Raymondův algoritmus

Uzly, představující procesy v distribuovaném systému jsou uspořádány do binárního stromu. V tomto stromě se pohybuje "token", udávající povolení ke vstupu do kritické sekce. Procesy bojují o token, pouze ten který ho má smí vstoupit do KS. Problémem je přerušení některé větve v systému - pak procesy v této větvi nemohou vstoupit do KS.

Více v PDI:

https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/PDI-IT/lectures/lecture_2.pdf

https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/PDI-IT/lectures/PDI-Group_Communication.pdf

https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/PDI-IT/lectures/lecture3-part1.pdf

Další zdroje

[zcu.arcao.com/kiv/ds/ds.pdf] kapitola 4.3

Kategorie: Státnice 2011 | Paralelní a distribuované algoritmy

---

- Stránka byla naposledy editována 20. 6. 2012 v 10:09.