Koordinace v distribuovaném systému: čas a vzájemné vyloučení

Paralelní a distribuované systémy, přednáška 7

Tomáš Urbanec

Katedra informatiky PřF UPOL

6.11.2024

Co nás čeká?

Koordinace v distribuovaném systému

1. Čas v distribuovaném systému
   - Skutečný čas
   - Logický čas
2. Vzájemné vyloučení v distribuovaním systému
   - Centralizované řešení
   - Distribuované řešení
   - Řešení tokenem
   - Decentralizované řešení

Čas v DS

Čas a koordinace DS

- Často koordinace na základě času (typicky timestamp).
- Absence globálních hodin.
- UTC (Coordinated Universal Time [sic])
  - radiostanice, satelity,
  - úrovně (vrstva),
  - přesnost ± 10 ms (lze až 50 ns).
- Různý čas na různých uzlech (*skew*).
- Různé důvody: technologie, mimoběžnost, *drift*, ...
- Skutečný čas vs logický čas.
- Omezení: čas nikdy nevracíme zpět.
  - Proč?
  - A co tedy děláme?

Skutečný čas

Základy

- Synchronizace = omezení rozdílu.
- Nelze synchronizovat plně.
- Naivní algoritmus
  - Klient K požaduje čas u serveru S s UTC.
  1. K požádá o čas S.
  2. S odpoví svým aktuálním časem t_s.
  3. K nastaví čas dle t_s.
  - Ale to nefunguje. . . Proč?

Cristianův algortimus

Skutečný čas

- Klient K požaduje čas od serveru S s UTC.

1. K požádá o čas S a uloží čas požadavku t_1.
2. S odpoví svým aktuálním časem t_s.
3. K uloží čas přijetí t_2.
4. Výsledný čas K: $t_s + (t_2 - t_1)/2$.

- Problém při různém trvání dotazu a odpovědi
  - Vytváření, přenos, ...
- Lze poslat více dotazů a udělat průměr.

Network Time Protocol

Skutečný čas – NTP

- Skutečně a často používané.
- Centrální autorita – UTC.
- Základní myšlenka podobná Cristianovu algoritmu.
- Klient K požaduje čas od serveru S s UTC.
  1. K požádá o čas S v čase t_1
  2. S přijme požadavek v čase t_2
  3. S pošle odpověď s aktuálním časem t_3
  4. K přijme odpověď v čase t_4
  5. Odhad offsetu K vůči S: $\Theta = t_3 + \frac{(t_2 - t_1) + (t_4 - t_3)}{2} - t_4$.
  6. Odhad zpoždění zpráv: $\delta = \frac{(t_4 - t_1) - (t_3 - t_2)}{2}$.
  7. K případně upraví rychlost času do srovnání.
- Výpočet Θ a δ se provede vícekrát.
- Použije výsledek s minimální δ.
- Vždy se ptá ten s více skoky od UTC serveru (na vyšší vrstvě).

Reference Broadcast Synchronization

Skutečný čas – RBS

- Distribuované řešení.
- Používané v bezdrátových/senzorových sítích.
- Cíl není skutečný čas (UTC), ale shodnout se na stejném čase.
- Předpoklady
  - Mezičasy generuje až síťové rozhraní – žádné ztráty výpočtem.
  - Předpokládáme stejnou rychlost přenosu mezi všemi páry uzlů.
- Uzel u pošle (broadcast) zprávu m (reference).
  1. Uzly p a q po obdržení m uloží časy přijetí t_p a t_q.
  2. Uzly p a q si vzájemně pošlou zprávu s časem přijetí m (tj t_p a t_q).
  3. Uzly p a q z dodaného času vypočítají offset proti druhému uzlu.
  4. Časem výsledky mohou být neaktuální kvůli driftu $\rightarrow$ průměry, lin. regrese.
- Uzly pouze ukládají offsety, neupravují svůj čas.

Logické hodiny

- Dosud jsme řešili skutečný čas.
- Stačilo by nám méně?
- Logický čas $\approx$ uspořádání operací.
- Tj. operace *a* nastala někdy po operaci *b*.
- Leslie Lamport, 1978

Lamportův algoritmus

Logické hodiny

- Lokální hodiny $\approx$ čítač.
- S každou zprávou se odesílá i aktuální hodnota čítače.
- Čítač se zvedne
  - Při vykonání lokální operace.
  - Před odesláním zprávy.
  - Při přijetí zprávy s vyšší hodnotou čítače.
- Koncept předcházení: a předchází b ($a \rightarrow b$), pokud
  - a a b jsou v jednom procesu a a se stalo před b
  - a je odeslání zprávy a b je přijetí této zprávy
  - tranzitivní uzávěr
- Při souběhu zpráv rozhodujeme dle id procesu.
- Relace $\rightarrow$ je tranzitivní, ale ne nutně úplná.
- (tabule)
- Lze použít pro úplné uspořádání událostí v DS.
- Lze použít pro vzájemné vyloučení.

Vektorové hodiny

Logické hodiny

- Lamportovy hodiny zaručují $a \rightarrow b \Rightarrow C(a) < C(b)$
- Nic ale nezaručuje $C(a) < C(b) \Rightarrow a \rightarrow b$

$\rightarrow$ Lamportovy hodiny nepopisují kauzalitu

- Vektorové hodiny
  - $=$ vektor $\overrightarrow{C_i}$ logických hodin v každém procesu
  - $\overrightarrow{C_i}[i]$ je lokální logický čas procesu i
  - $\overrightarrow{C_i}[j] = k$ znamená, že proces i ví, že proces j vykonal k operací.
  - Uspořádání
    - $\overrightarrow{C_1} \leq \overrightarrow{C_2}$, pokud $\forall i : \overrightarrow{C_1}[i] \leq \overrightarrow{C_2}[i]$.
    - $\overrightarrow{C_1} < \overrightarrow{C_2}$, pokud $\overrightarrow{C_1} \leq \overrightarrow{C_2}$ a $\overrightarrow{C_1} \neq \overrightarrow{C_2}$.
  - $\overrightarrow{C_i}(a) < \overrightarrow{C_j}(b)$ znamená $a \rightarrow b$ (ale kauzalita?)
  - Neporovnatelné hodiny znamenají konkurentní operace.
- (tabule)

Vzájemné vyloučení v DS

Centralizované řešení

Vzájemné vyloučení

- Velmi jednoduchá myšlenka i provedení.
- Jeden uzel (rozhodčí) rozhoduje o výhradním přístupu ke sdílenému zdroji.
  - Rozhodčí drží frontu žádostí.
  1. Zájemce *i* pošle zprávu *REQUEST*.
  2. Rozhodčí potvrdí *OK*, je-li zdroj volný, jinak dá *i* do fronty.
  3. Proces *i* po dokončení práce pošle *RELEASE*.
  4. Rozhodčí pošle *OK* dalšímu ve frontě, nebo čeká na žádost.
- Rozhodčí může selhat (ostatní nepoznají).
- Rozhodčí může být přetížený (bottleneck).

Distribuované řešení

Vzájemné vyloučení

- Ricart-Agrawala algoritmus, 1981
- Insipirován logickými hodinami.
- Potřebuje $2(N-1)$ zpráv.
- Nejsou nutné FIFO kanály.
- Dva typy zpráv: *REQUEST* a *ACK*
- Proces i chce výhradní přístup:
  1. Zašle REQUEST (i, t) ostatním.
  2. Proces j po přijetí odpoví ACK (nechce přístup nebo $t < C_j$), nebo přidá i do fronty žádostí.
  3. Pokud čekající i dostane $(N-1)$ ACK, získává přístup.
  4. Při ukončení práce pošle i ACK všem ve své frontě.
- Potřebuje multicast.
- Má N bodů selhání (opakované dotazy).

Řešení tokenem

Vzájemné vyloučení

- Vytvoříme kruhový overlay nad sítí.
- V sítí je právě jeden token.
- Výhradní práce se zdrojem = držení tokenu.
  1. Proces *i* obdrží token.
  2. Chce-li pracovat, pracuje. Jinak odesílá token dále.
  3. Po ukončení práce pošle *i* token dále (nesmí jej držet).
- Ztráta tokenu? Informace o ztrátě tokenu?
- Udržování kruhu (někdo vypadne $\rightarrow$ přeskočit? potvrzování?).

Decentralizované řešení

Vzájemné vyloučení

- Založeno na hlasování.
- Každý zdroj je v N replikách.
- Každá rpelika mákoordinátora.
- Výhradní práce se zdrojem = hlas více než $N/2$ koordinátorů.
- Kooridnátoři mohou selhat (resetovat se)
  - Vydání více povolení?
  - Vydání více povolení u více než $N/2$ koordianátorů?
  - Možné, ale extrémně nepravděpodobné.
  - Ve srovnání s jinými typy chyb zanedbatelné.

Srovnání přístupů

Vzájemné vyloučení

- Nejjednodušší je centralizovaný přístup.
- Př. ZooKeeper – koordinační systém
  - vzájemné vyloučení,
  - volba leadera.
  - monitoring,
  - zotavení z chyb.
  - ...
- token vs. vyžádání přístupu
- centralizované, decentralizované nebo distribuované

Changelog