Konzistence, replikace a globální stav DS

Paralelní a distribuované systémy, přednáška 10

Tomáš Urbanec

Katedra informatiky PřF UPOL

4.12.2024

Co nás čeká?

1. Shoda (dodělávky z minula)
   - Paxos
   - Některá omezení
2. Konzistence a replikace
   - Konzistence a její typy
   - Metody replikace
   - Některá omezení
3. Globální stav
   - Snapshoty a řezy
   - Chandy-Lamport algoritmus
   - Dijkstra-Scholten algoritmus

Shoda

Paxos algoritmus

Shoda v DS

- Lamport, 1989. Mnoho variant.
- Fail-noisy model.
- Často používaný, ale komplikovaný ($\rightarrow$ raft).
- Více typů procesů (client, proposer, leader, learner, acceptor).
- Vnitřní dělení uzlů (proposer + acceptor + learner).
- Většinové kvórum.
- Předpoklady:
  - I asynchronní DS.
  - I nespolehlivé spoje.
  - Chybné zprávy detekovatelné.
  - Operace deterministické.
  - Nemáme náhodné chyby.
- (tabule - zjednodušená verze)

Omezení

Shoda v DS

- Shoda není vždy možná.
- Potřebujeme:
  - synchronní systém,
  - nebo omezený čas na zprávu a pořadí zpráv,
  - nebo multicast.
- A co když máme v DS více menších DS?
- A každý má jiné chyby?
- Třeba byzantské?
- ...

Replikace a konzistence

Redundance, replikace a konzistence

Základy

- Redundance – již známe (více vzájemně nahraditelných uzlů)
- Replikace – již tušíme (raft jako replikace stavu)
- Konzistence – repliky musí být shodné (dále)

Replikace:

- Důvody
  - zvýšení spolehlivosti (redundance),
  - zvýšení výkonu.
- Zahrnuje redundanci (replika přežije pád 'originálu'),
  ale nabízí více (geografická dostupnost, rozložení zátěže, . . .).
- Problémy s konzistencí (úprava jedné repliky? propagace?).
  - Problém výkonu $\rightarrow$ replikace $\rightarrow$ synchronizace replik
    $\rightarrow$ problém výkonu?
- Př. Kešování, CDN (Content Deliver Network), . . .
- Př. DNS

Konzistence

Základy

- ≈ jednotlivé uzly DS reagují (dostatečně) stejně.
- Typicky nás zajímá výsledek proložených write a read operací.
- Více typů: silná, eventuální, sekvenční, kauzální, ...
- Silná (striktní) konzistence
  - Jako kdyby byly změny propagovány okamžitě.
  - Všichni reagují stejně.
- Sekvenční konzistence
  - Výsledný stav odpovídá nějakému sekvenčnímu pořadí operací.
  - Pořádí správně v rámci jednotlivých procesů.

Konzistence

Základy

- Kauzální konzistence
  - Pokud a potenciálně předchází b ($a \rightarrow b$), pak všichni vidí nejprve výsledek a a pak výsledek b.
  - U konkurenčních (nekauzálních) párů operací je to jedno.
  - (tabule)
- Eventuální konzistence
  - Jednou dostaneme aktuální data, ale ne nutně hned.
  - Typické kešování.
  - Tj. povolujeme read-write konlikty.
  - Co s write-write konflikty?
    - slabá: neřešíme,
    - silná: skončí ve stejném stavu (CRDT, spec. datové struktury $\rightarrow$ nepotřebujeme shodu)

Raft

Metody replikace

- Už známe.
- Bavili jsme se o shodě na operaci...
- ...přitom se nám replikoval log leadera (jeho stav).

Řetězová replikace

Metody replikace

- Řetězec uzlů: hlava, . . . , ocas
- Zápisy na hlavu, čtení na ocase
- Zápis se propagují od hlavy k ocasu
- Zápis se potvrzuje od ocasu k hlavě (až pak uživateli)
- $\rightarrow$ silná konzistence
- Nutná správa řetězce (tolerance chyb, centrální uzel)
- Selhání:
  - Hlavy: náhrada následníkem
  - Ocasu: náhrada předchůdcem
  - Uzel uprostřed: řeší centrální uzel
  - Centrální uzel: nutná replikace!
- Koncepčně jiné:
  - Nemáme leadera.
  - Při čtení reaguje ocas rovnou (vše je potvrzeno).
  - Jeden pomalý uzel degraduje celý systém $\rightarrow$ latence.

CAP teorém

Omezení

- Uvažujme DS s replikací dat (DB, obecně data store)
- Pozorování: síťovým problémům se nelze vyhnout.

→ Tj. rozpadu sítě (partitioning) se nelze vyhnout.

- Když nastane rozpad (P) můžeme zachovat:
  - dostupnost (Availability),
  - nebo konzistenci (Consistency),
  - (ale ne obojí).
- (tabule)

PACELC teorém

Omezení

- CAP teorém je nepřijemný. . .
- . . . ale k rozpadu nemusí dojít.
- Tj. můžeme mít C i A!
- . . . ale . . .
- I když nemáme P, tak (Else) musíme volit mezi
  - Latencí (Latency)
  - Konzistencí (C)
- (tabule)
- PA/EL, PA/EC, PC/EL, nebo PC/EC?
- PostgreSQL (distr.) PC/EC, NoSQL (distr.) barvité

CALM teorém

Omezení

- „consistency as logical monotonicity“
- Nedívejme se na vlastnost systému, ale na vlastnsoti programu.
- Kdy lze (distribuovaný) program implememntovat bez synchornizace a zachovat konzistenci?
- Když je monotónní:
  - Pokud přidáme data na vstup, tak dostaneme stejný nebo rozšířený výstup.
  - Tj. žádné přepisy nebo ubírání.
  - Ano: přidávání do seznamu, sjednocení, ...
  - Ne: Maximum/minimum, množinový rozdíl, ...
  - (tabule)
- Program má konzistentní implementaci bez distribuované synchonizace, právě tehdy když je monotónní.

Globální stav

Globální stav

Základy

- Lokální stav procesu
  - hodnoty proměnných,
  - alkovoané zdroje,
  - stav vykonávání programu.
- Globální stav DS
  - lokální stav všech procesů DS,
  - stavy všech kanálů mezi nimi.
- Řez: rozdělení událostí v systému na dvě skupiny
  - již se staly,
  - ještě se nestaly.
- Konzistence řezu C: $(a \rightarrow b \wedge b \in C) \Rightarrow a \in C$
- (tabule)
- Snapshot
  - $\approx$ globální stav DS v daném okamžiku
  - $=$ stav (konzistentního) řezu
- Posloupnost konzistentních řezů $\approx$ průběh výpočtu v DS

Chandy-Lamport algoritmus

Globální stav

- Výpočet snapshotu (zjištění stavu DS)
  1. Iniciátor uloží svůj stav pošle zprávu s markerem (snapshot request).
  2. Proces obrdží zprávu s markerem
     - 2.1 Poprvé: uloží svůj stav, pošle jej iniciátorovi a pošle snapshot request všem ostatním.
     - 2.2 Jindy: uloží stav na kanálu
  3. Proces s markerem dostane zprávu bez markeru
     - musí být z okamžiku před snaphostem,
     - přepošle ji iniciátorovi, aby ji přidal.
- FIFO kanály (jinak problém)
- (tabule)

Dijkstra-Scholten algoritmus

Globální stav

- Detekce ukončení výpočtu.
- V obecné síti buduje kostru ($\approx$ volba leadera v ad-hoc síti).
- Uzel je aktivní, nebo skončil.
- Zprávy
  - SIGNAL: Výzva počítáš/nepočítáš?
  - ACK: Odpověď už nepočítám, nebo se už hlásím jinam.
- Modelováno pomocí deficitů kanálů (#SIGNAL - #ACK zpráv na vstupu/výstupu).
- (tabule)

Changelog