Tolerance chyb a shoda v DS

Paralelní a distribuované systémy, přednáška 9

Tomáš Urbanec

Katedra informatiky PřF UPOL

2.12.2025

Co nás čeká?

1. Shoda v DS s chybami
   - Základní pohled
   - Raft
   - Paxos
   - Byzantské chyby podruhé
   - Některá omezení
2. Konzistence a replikace
   - Konzistence a její typy
   - Metody replikace
   - Některá omezení

Shoda v distribuovaném systému

Shoda v DS

Základy

- Systém se musí shodnout na výsledku, stavu, další akci, ...
- Bez chyb $\rightarrow$ triviální.
- S chybami? Záleží na okolnostech.
  - Typy chyb,
  - Model detekce,
  - ...

Skupiny

Shoda v DS

- Zavedení redundance.

→ Úlohu uzlu převezme skupina uzlů.

  - Protokol primární-záloha
    - Hierarchická skupina
    - Pracuje primární (zápisy)
    - Výpadek primárního → záloha převezeme roli
  - Protokol replikovaného zápisu
    - 'Plochá' skupina,
    - Všichni stejná role,
    - Nemají kritický bod,
    - . . . ale nutná koordinace.

- Skupina je odolná vůči chybám, pokud všechny bezchybné procesy vykonávají stejné operace ve stejném pořadí.

Algoritmus Flooding consensus

Shoda v DS

- Základní algoritmus pro shodu.
- Omezení na fail-stop model.
- Shoda se hledá v navazujících kolech.
  - V každém kole si uzly mění návrhy.
  - Každý uzel z návrhu vybere volbu deterministicky a všichni stejně
  - Pokud nikdo nedostane všechny odpovědi, začne nové kolo.
  - Všichni dostanou všechny odpovědi $\rightarrow$ volba $\rightarrow$ konec.
  - Někdo dostane všechny odpovědi a rozešle volbu $\rightarrow$ čekající přeberou volbu $\rightarrow$ konec.
  - (tabule)
- Existuje více variant. Základní myšlenka obdobná.

Byzantské chyby podruhé

Shoda v DS

- Co s byzantskými chybami?
- Idea problému
  - n-generálu se musí domluvit
  - m z nich je zrádných
  - Co s tím?
- Základní řešení
  - (tabule 2, 3, 4, . . . generálové)
  - $n > 3 \cdot m$
  - Exponenciální počet zpráv
- Jiná řešení
  - Mají další předpoklady a omezení.
  - HoneyBadgerBFT (dig. podpisy) $\rightarrow n > 2 \cdot m$
  - Blockchain (cena) $\rightarrow n > 2 \cdot m$
  - MinBFT (speciální HW) $\rightarrow n > m + 1$
  - Kryptografické protokoly (důvěra) $\rightarrow n > m + 1$

Raft algoritmus

Shoda v DS

- Ongaro, Osterhout, 2014
- Fail-noisy model.
- Shoda v DS ve formě replikovaného logu operací.

$\rightarrow$ uzly ve stejném stavu.

- Moderní, často používaný.
- 'Náhrada Paxosu' (dále).
- Skupina má lídra.
- Stavy uzlů: Lídr, následovník, kandidát.
- Koncepty: volba lídra, replikace logu, provedení operace.
- Většinové kvórum ($n > 2 \cdot m$).
- (tabule)
- Vizualizace: https://raft.github.io

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 ($n > 2 \cdot m$).
- Předpoklady:
  - I asynchronní DS.
  - I nespolehlivé spoje.
  - Chybné zprávy detekovatelné.
  - Operace deterministické.
  - Nemáme náhodné chyby.
- (tabule - zjednodušená verze)

Shoda v DS

Omezení

- 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.
  - Pořadí stejné 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 konflikty.
  - 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 klientovi potvrzuje ocas
- $\rightarrow$ silná konzistence
- Nutná správa řetězce (tolerance chyb, centrální uzel)
- Selhání (fail-stop model):
  - 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 nepříjemný...
- ...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 vlastnosti systému, ale na vlastnosti programu.
- Kdy lze (distribuovaný) program implementovat bez synchronizace 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 synchronizace, právě tehdy když je monotónní.

Changelog