PDV 11 2024/2025s

Konsensus

Michal Jakob
michal.jakob@fel.cvut.cz
Centrum umělé inteligence, katedra počítačů, FEL ČVUT

Co mají tyto příklady společného?

Skupina procesů usilujících o :

- udržování jejich lokálních seznamů aktivních procesů aktuálních [detekce selhání]
- zvolení lídra a zajištění, že každý ví, kdo je lídrem [volba lídra]
- zajistit vzájemně exkluzivního přístupu ke sdílenému prostředku (např. souboru) [vyloučení procesů]
- usilujících o doručení stejných aktualizací ve stejném pořadí [uspořádaný multicast]

Souvisí s konsensem

Ve všech těchto případech se procesy snaží vzájemně koordinovat, aby se shodly na nějaké hodnotě

- na stavu každého procesu (aktivní/neaktivní)
- kdo je lídrem
- kdo má přístup k sdílenému prostředku
- pořadí zpráv

Všechny tyto problémy souvisejí s problémem **konsensu**

Problém konsensu

N procesů

Každý proces P má

- vstupní proměnou x_P (výchozí návrh): zpočátku buď 0 nebo 1
- výstupní proměnou y_p: může být změněna pouze jednou

Cílem je **shodnout na hodnotě** výstupní proměnné.

Algoritmu pro řešení konsensu

Algoritmus pro řešení konsensu musí splňovat následující vlastnosti:

- **Shoda**: všechny *nehavarované* procesy se shodnou na výstupní proměnné
- **Validita**: mají-li všechny *nehavarované* procesy stejnou hodnotu vstupní proměnné, musí se na této hodnotě shodnout i jako na hodnotě výstupní proměnné
- **Ukončení:** každý *nehavarovaný* proces musí v konečném čase nastavit hodnotu výstupní proměnné

Pozor: Shoda říká, že výsledek je *stejný* pro všechny. Validita omezuje, *jaký* ten výsledek může být (musí být jedním z navržených, a pokud byl navržen jen jeden, musí to být on).

Proč je konsensus důležitý?

Mnoho problémů v DS je **ekvivalentních** konsensu

- perfektní detekce selhání
- volba lídra
- vyloučení procesů
- spolehlivý nebo totálně uspořádaný multicast
- …

Vyřešení konsensu by tedy bylo velmi užitečné.

Konsensus v systémech bez selhání

Problém konsensu v systémech **bez selhání je řešitelný**.
(v synchronním i asynchronním případě).

1. Jednotlivé procesy si vzájemně pošlou hodnoty svých vstupních proměnných.
2. Následně provedou stejný výpočet nad posbíranými hodnotami (až budou mít všechny): např. max, min, majority apod.
3. Výstupní proměnnou nastaví na výsledek výpočtu.

Konsensus v synchronním systému se selháními

Problém konsensu v **synchronních** systémech **se selháními procesů je řešitelný**.

Uvažujeme maximálně *f* selhání procesů

Algoritmus pro proces P_i

$z_i \leftarrow x_i$ // vstupní proměnná

for $round$ **from** 1 **to** $f + 1$

  if aktuální hodnota z_i **nebyla rozeslána**

    broadcast(z_i)

  $t_{j,i} \leftarrow$ hodnota obdržená z procesu *j* v tomto kole (pokud nějaká byla)

  $z_i \leftarrow \min_j(x, t_{j,i})$

$y_i \leftarrow z_i$ // nastav výstupní proměnnou

Konsensus v asynchronních systémech se selháními

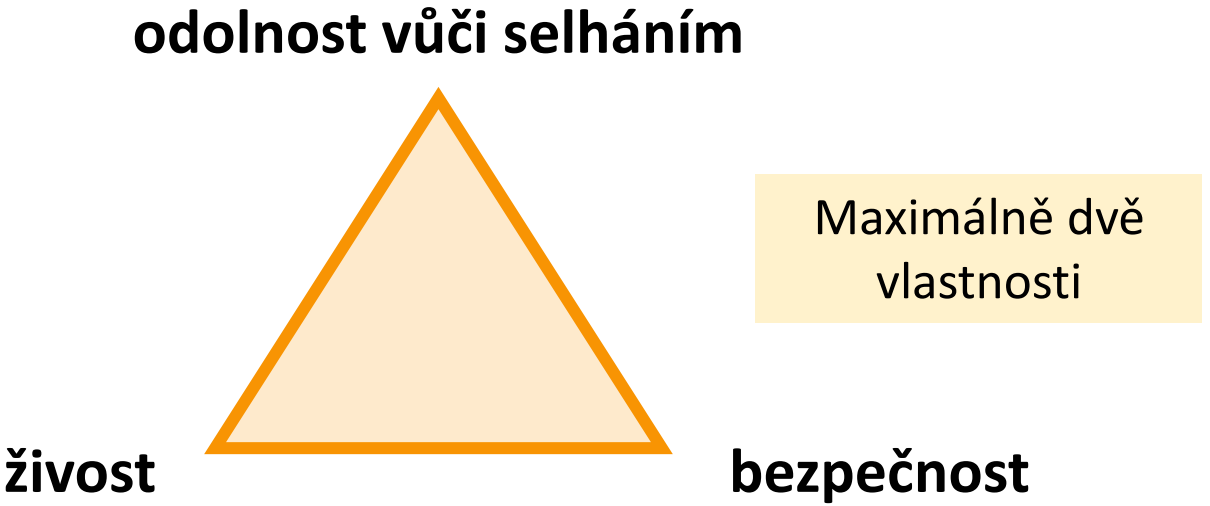
Problém konsensu v **asynchronních** systémech **se selháními není řešitelný**.

- nemůžeme využívat *time-outů* k rozlišení mezi selháním a zpožděním
- pro jakýkoliv algoritmus *existuje* nejhorší možný průběh, který *zabrání* dosažení konsensu
- vyplývá v tzv. **FLP teorému**

FLP teorém

FLP teorém

V asynchronním distribuovaném systému **nelze** dosáhnout **současně bezpečnosti a živosti** distribuovaného výpočtu, pokud v něm může docházet k **selháním** (byť i jediného procesu).



FLP teorém

Hlavní myšlenka důkazu:

- Představte si distribuovaný výpočet, ve kterém proces P hraje klíčovou roli
- Výpočet je odolný vůči chybám: když P havaruje, tak se výpočet adaptuje a pokračuje dále.
- V důkazu pomoci zpožďování zpráv „zmatem“ výpočet tak, aby si myslel, že proces P havaroval, ale následně se proces P vzpamatuje a znovu se „zapojí“ do výpočtu.
- Výše uvedené zabere čas, ale výpočet se nikam neposune. Toto lze libovolně krát zopakovat.

Výše uvedené „zmatení“ dosáhneme pouze skrze zpožďování zpráv (tj. *bez* selhání). **Tj. FLP „útočí“ na výpočty odolné proti selháním pomoci běhů, které jsou bez selhání!**

Řešitelnost vs. neřešitelnost

Problém je řešitelný, pokud pro něj existuje **úplný** a **korektní** algoritmus, tj. algoritmus, který produkuje správný výsledek a vždy doběhne **v konečném čase**.

→ Z tohoto pohledu problém konsensu **řešitelný není**.

Ale: tyto nikdy nekončící běhy algoritmu pro konsensus jsou velmi **nepravděpodobné**.

- Matematicky jsou sítě asynchronní (teoreticky může paket letět den), ale **statisticky** se chovají synchronně.

Z praktického hlediska je tedy lépe FLP teorém interpretovat tak, že **nelze garantovat vyřešení konsensus**

→ *občasná živost* (*eventual liveness*): algoritmus jednou doběhne –pokud nebude docházet k selháním a nadměrnému zpožďování zpráv,

Algoritmus Paxos

Nejstarší a nejznámější algoritmus pro distribuovaný konsensus (Lesslie Lamport)

Pracuje v kolech; každé kolo má unikátní číslo volebních zpráv

Kola jsou asynchronní

- Je-li proces v kole j a dorazí-li mu zpráva z kola $j + 1$: přeruší činnost v rámci kola j a posuň se do kola $j + 1$
- využívá časová time-outy (může být pesimistický)

Každé kolo rozděleno do třech fázi (které jsou taky asynchronní)

1. Fáze ELECTION: Je zvolen lídr
2. Fáze BILL: zvolený lídr navrhne hodnotu, ostatní procesy potvrzují
3. Fáze LAW: Lídr rozešle všem ostatním procesům finální (potvrzenou) hodnotu

Souhrn

Problém konsensu je v jádru mnoha problémů v DS.

V asynchronním DS **nelze** při přítomností selhání **konsensus vyřešit** ve smyslu bezpečnosti a živosti.

Praktická řešení většinou garantují **bezpečnost**.