Konsenzus v distribuovaném systému

B4B36PDV – Paralelní a distribuované výpočty

Osnova

- Opakování z minulého cvičení
- Proč bychom mohli chtít řešit konsenzus v DS?
- RAFT algoritmus
- Zadání semestrální úlohy

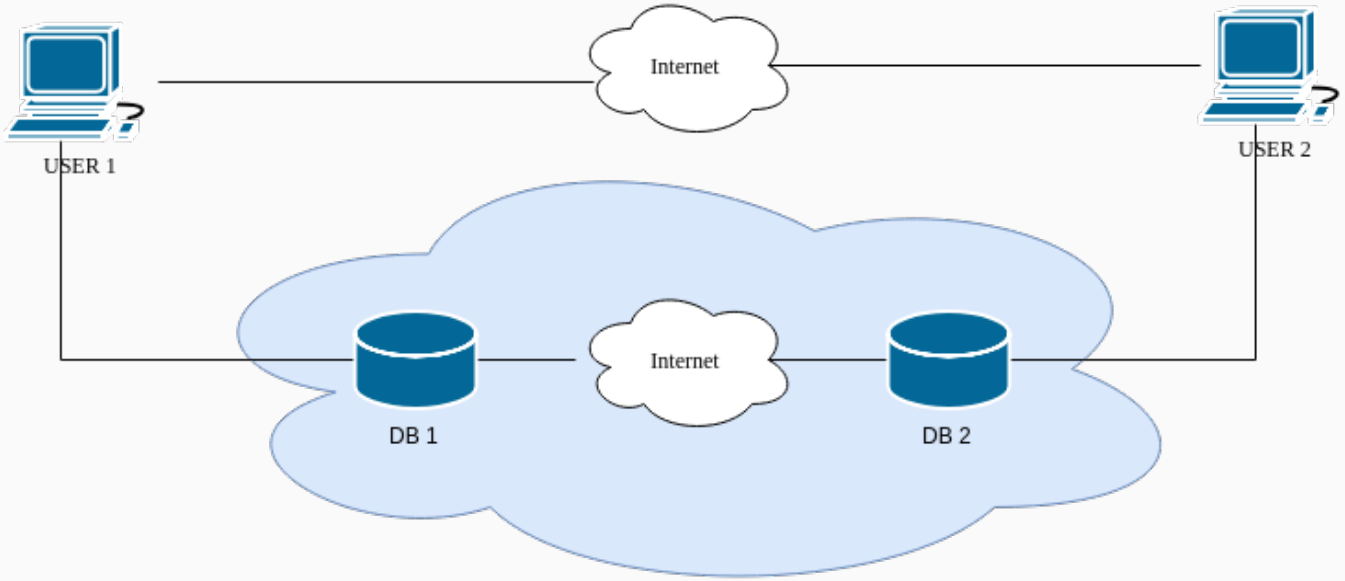
Opakování z minulého cvičení

http://goo.gl/a6BEMb

Konsenzus v distribuovaném světě

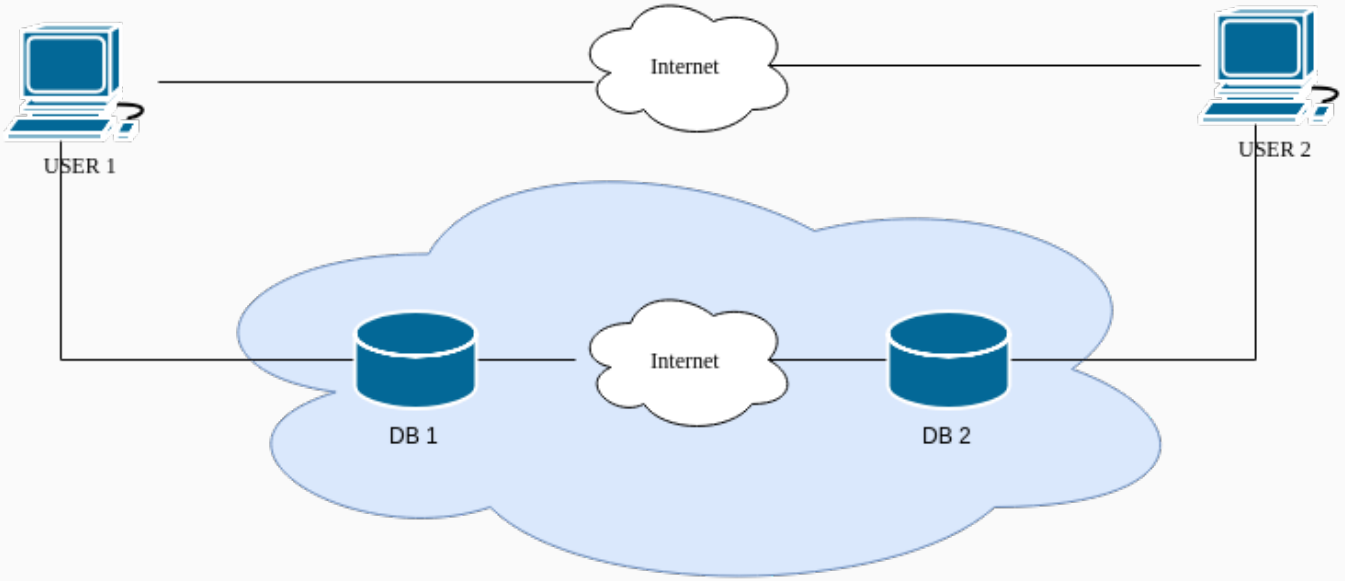
Distribuovaná databáze

Uvažujme upravený protokol, kdy klient čeká na potvrzení zápisu...



Distribuovaná databáze

Uvažujme upravený protokol, kdy klient čeká na potvrzení zápisu...



Co se stane, když server zodpovědný za replikaci selže?

Distribuovaná databáze

⚠ Některé servery mohou znát aktuální data a jiné nikoliv!

Potřebujeme se shodnout na společné „pravdě“

Distribuovaná databáze

⚠ Některé servery mohou znát aktuální data a jiné nikoliv!
Potřebujeme se shodnout na společné „pravdě“

⚠ Zároveň se nám nesmí ztrácet data, o kterých si klient myslí, že jsou uložená

Jak to řeší centralizované databáze?

Ukládání obsahu databáze na disk při každé operaci je drahé...

⚠ Části databáze si proto raději držíme v paměti

Jak to řeší centralizované databáze?

Ukládání obsahu databáze na disk při každé operaci je drahé…

⚠ Části databáze si proto raději držíme v paměti

Co když nám server po potvrzení zápisu spadne?

Jak to řeší centralizované databáze?

Ukládání obsahu databáze na disk při každé operaci je drahé…

⚠ Části databáze si proto raději držíme v paměti

Co když nám server po potvrzení zápisu spadne?

Řešení: Před potvrzením zápisu požadavek uložíme do logu (žurnálu)!

→ Pokud server spadne, z žurnálu obnovíme ztracená data

Tomu se říká *journaling* nebo *write-ahead logging* a je součástí většiny „rozumných“ databázových systémů.

Jak bychom mohli použít journaling v distribuované DB?

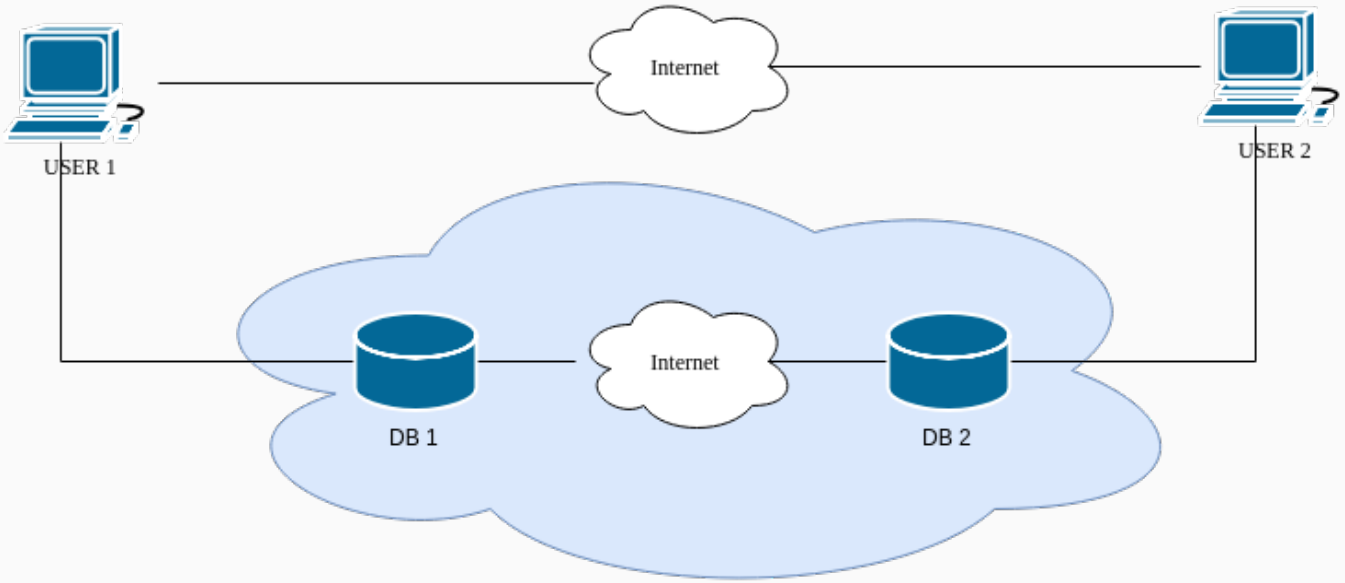
Jak bychom mohli použít journaling v distribuované DB?

Nemusíme se shodovat na konkrétním obsahu databáze
(ten může být potenciálně obrovský!)

Stačí, když se shodneme na obsahu žurnálu
(a doplníme případné chybějící požadavky/záznamy)

Raft algoritmus

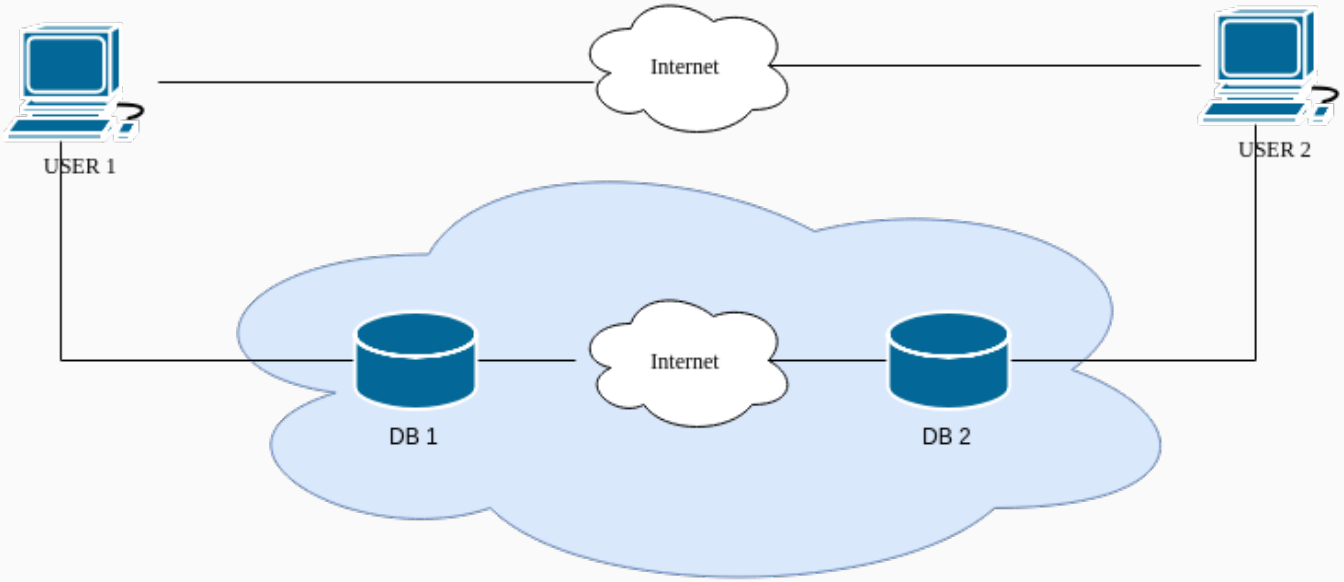
Jak řešit konkurentní požadavky?



Co když dva klienti budou provádět zápisy současně?
(potenciálně na různé repliky?)

Možnosti: Dohodnout se na globálním uspořádání

Jak řešit konkurentní požadavky?

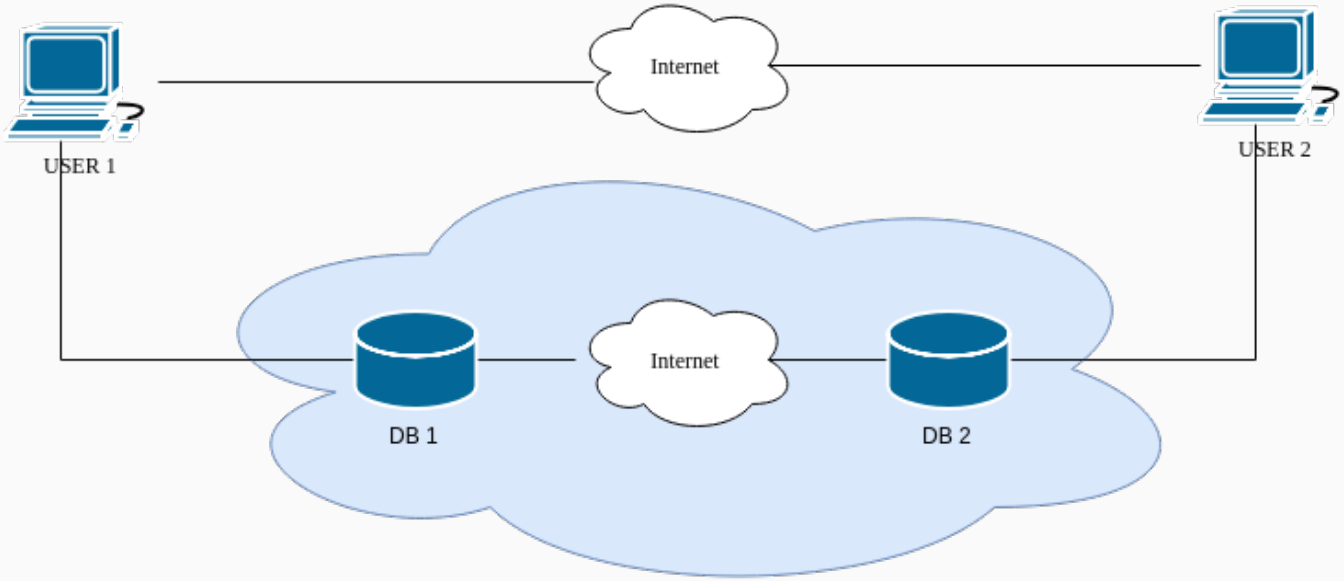


Co když dva klienti budou provádět zápisy současně?
(potenciálně na různé repliky?)

Možnosti: Dohodnout se na globálním uspořádání

Zvolit centrální autoritu, která požadavky seřadí (*leadera*)

Jak řešit konkurentní požadavky?



Co když dva klienti budou provádět zápisy současně?
(potenciálně na různé repliky?)

Možnosti: Dohodnout se na globálním uspořádání

Zvolit centrální autoritu, která požadavky seřadí (*leadera*)

Jak řešit konkurentní požadavky?

Co když pak požadavek přijme ne-leader?

Volba leadera

⚠ Leader musí mít v DS autoritu!

Co to znamená?

Volba leadera

⚠ **Leader musí mít v DS autoritu!** Co to znamená?

→ Musí mít důvěru nadpoloviční většiny serverů

Volba leadera:

1. Pokud si server myslí, že v systému není leader, chce se jím stát sám.
   Kdy si myslí, že v systému není leader?

Volba leadera

⚠ **Leader musí mít v DS autoritu!** Co to znamená?

→ Musí mít důvěru nadpoloviční většiny serverů

Volba leadera:

1. Pokud si server myslí, že v systému není leader, chce se jím stát sám.
   Kdy si myslí, že v systému není leader?
2. Pošle ostatním serverům žádost o to, aby ho respektovali
   (zpráva `RequestVote`)
3. Stane se `kandidátem`

Volba leadera

⚠ **Leader musí mít v DS autoritu!** Co to znamená?

→ Musí mít důvěru nadpoloviční většiny serverů

Volba leadera:

1. Pokud si server myslí, že v systému není leader, chce se jím stát sám.
   Kdy si myslí, že v systému není leader?
2. Pošle ostatním serverům žádost o to, aby ho respektovali
   (zpráva `RequestVote`)
3. Stane se `kandidátem`
4. Pokud s tím většina serverů souhlasí, stane se leaderem

Volba leadera

Přišel Vám `RequestVote`...

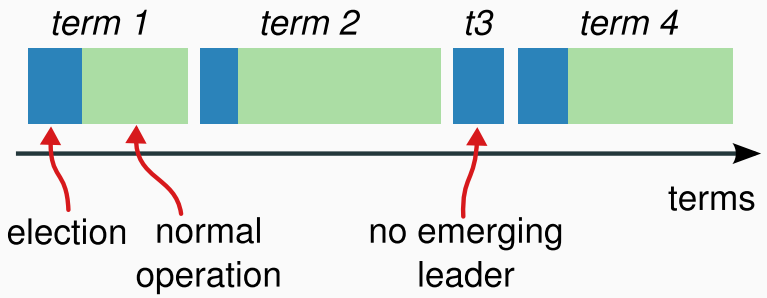
Kdy budete `kandidáta` respektovat?

Volba leadera

Přišel Vám `RequestVote`...

Kdy budete `kandidáta` respektovat?

- Jeho požadavek musí být aktuální podle „logického času“



Aktuální `term` posíláme v každé zprávě.

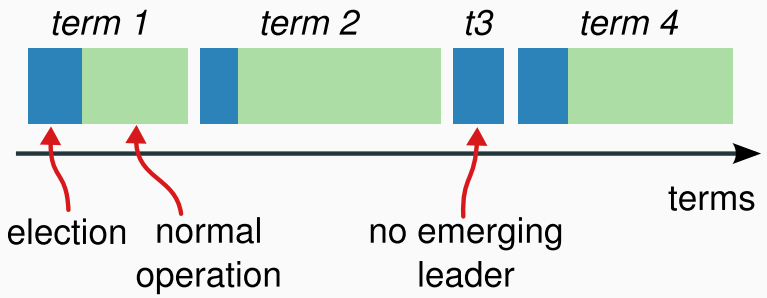
```
currTerm = max(currTerm, msg.term)
```

Volba leadera

Přišel Vám `RequestVote`...

Kdy budete `kandidáta` respektovat?

- Jeho požadavek musí být aktuální podle „logického času“



Aktuální `term` posíláme v každé zprávě.

```
currTerm = max(currTerm, msg.term)
```

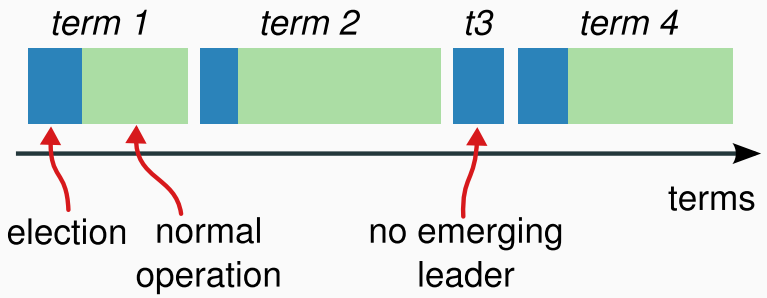
- Jeho log obsahuje všechny potvrzené (*committed*) požadavky

Volba leadera

Přišel Vám `RequestVote`...

Kdy budete `kandidáta` respektovat?

- Jeho požadavek musí být aktuální podle „logického času“



Aktuální `term` posíláme v každé zprávě.

```
currTerm = max(currTerm, msg.term)
```

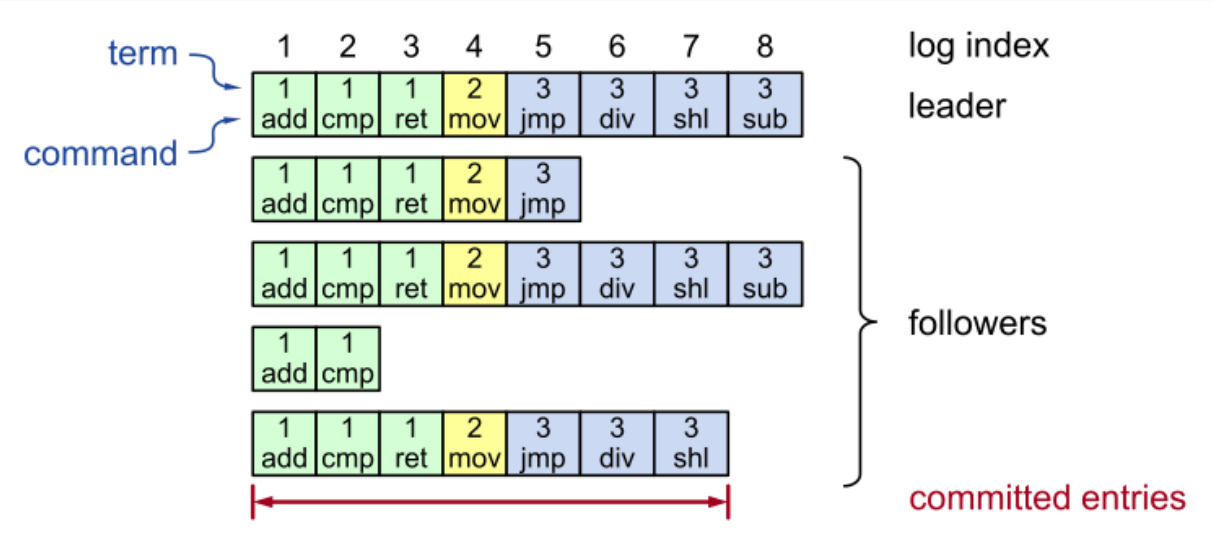
- Jeho log obsahuje všechny potvrzené (*committed*) požadavky

To ale nedokážeme zjistit :-(

Log („žurnál“)

Jediné, co dokážeme zjistit je, jestli je daný log „aktuálnější“

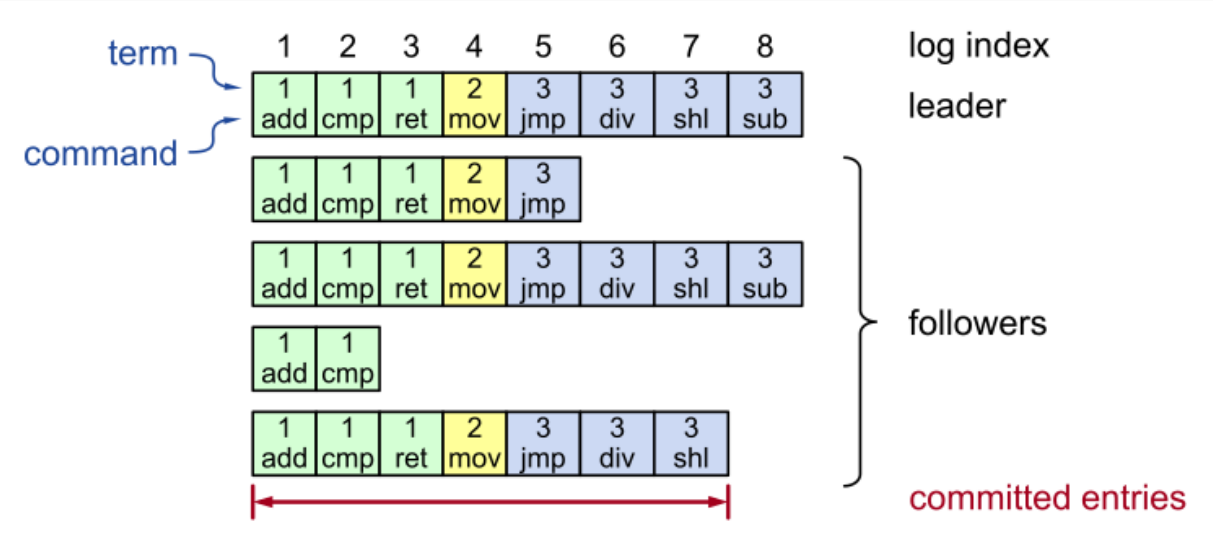
U každého záznamu si držíme `term`, ve kterém byl zapsaný a jeho pozici (`index`)



Log („žurnál“)

Jediné, co dokážeme zjistit je, jestli je daný log „aktuálnější“

U každého záznamu si držíme `term`, ve kterém byl zapsaný a jeho pozici (`index`)



Log **A** je aktuálnější než **B** pokud lexikograficky

```
(A[A.last].term, A[A.last].index) > (B[B.last].term, B[B.last].index)
```

Potvrzování zápisů

Potřebujeme ustanovit vztah mezi „aktualitou“ logu, hlasováním a potvrzením úspěšného požadavku…

Vzpomeňte si, že server, kterému chybí nějaký potvrzený záznam nesmí být zvolen leaderem…

Na kolika serverech musí být záznam zapsaný?

Potvrzování zápisů

Potřebujeme ustanovit vztah mezi „aktualitou“ logu, hlasováním a potvrzením úspěšného požadavku…

Vzpomeňte si, že server, kterému chybí nějaký potvrzený záznam nesmí být zvolen leaderem…

Na kolika serverech musí být záznam zapsaný?

⚠ **Stačí nám zapsat záznam na nadpoloviční většinu serverů :-)**

- Pokud by měl být zvolený leaderem server, kterému chybí potvrzený záznam, pak bude mít méně „aktuální“ log

Replikace logu

Pro zreplikování záznamu v logu zašle leader zprávu `AppendEntries` všem followerům

Replikace logu

Pro zreplikování záznamu v logu zašle leader zprávu `AppendEntries` všem followerům

Po obdržení potvrzení od většiny serverů považuje zápis za úspěšný

... a provede změnu v DB a potvrdí úspěch i klientovi

Replikace logu

Pro zreplikování záznamu v logu zašle leader zprávu `AppendEntries` všem followerům

Po obdržení potvrzení od většiny serverů považuje zápis za úspěšný

... a provede změnu v DB a potvrdí úspěch i klientovi

Kdy provedou změnu v DB i followeři?

Pro rekonstrukci stavu z logu, musí být log kompletní!

Nestačí nám tak shoda na zápisu jednoho prvku do logu

$\rightarrow$ Musíme se shodnout i na všech předcházejících prvcích

Pro rekonstrukci stavu z logu, musí být log kompletní!

Nestačí nám tak shoda na zápisu jednoho prvku do logu

→ Musíme se shodnout i na všech předcházejících prvcích

Co to znamená pro followera?

Ve zprávě `AppendEntries` posíláme

- Informace o replikovaném záznamu (`term`, `index` a obsah záznamu)

a navíc...

- Informace o záznamu předcházejícím (jeho `term` a `index`)

Ve zprávě `AppendEntries` posíláme

- Informace o replikovaném záznamu (`term`, `index` a obsah záznamu)

a navíc...

- Informace o záznamu předcházejícím (jeho `term` a `index`)

Follower zápis odmítne, pokud se na předchozím záznamu neshodne

Ve zprávě `AppendEntries` posíláme

- Informace o replikovaném záznamu (`term`, `index` a obsah záznamu)

a navíc...

- Informace o záznamu předcházejícím (jeho `term` a `index`)

Follower zápis odmítne, pokud se na předchozím záznamu neshodne

Ale co pak? Log musíme nějak doplnit...

Vlastnosti Raftu

- V každém `term`u zvolíme maximálně jednoho leadera.

Vlastnosti Raftu

- V každém `term`u zvolíme maximálně jednoho leadera.
- Pokud dva logy obsahují stejný záznam (stejný `term` a `index`), pak jsou až po `index` shodné.

Vlastnosti Raftu

- V každém `term`u zvolíme maximálně jednoho leadera.
- Pokud dva logy obsahují stejný záznam (stejný `term` a `index`), pak jsou až po `index` shodné.
- Pokud někdy klientovi potvrdíme úspěšný zápis, pak nikdy nebude leaderem server, kde tento zápis ještě neproběhl.

Vlastnosti Raftu

- V každém `term`u zvolíme maximálně jednoho leadera.
- Pokud dva logy obsahují stejný záznam (stejný `term` a `index`), pak jsou až po `index` shodné.
- Pokud někdy klientovi potvrdíme úspěšný zápis, pak nikdy nebude leaderem server, kde tento zápis ještě neproběhl.

Kompletní algoritmus je popsaný na:
`https://www.cs.princeton.edu/courses/archive/fall16/cos418/papers/raft.pdf`

Vlastnosti Raftu

- V každém `term`u zvolíme maximálně jednoho leadera.
- Pokud dva logy obsahují stejný záznam (stejný `term` a `index`), pak jsou až po `index` shodné.
- Pokud někdy klientovi potvrdíme úspěšný zápis, pak nikdy nebude leaderem server, kde tento zápis ještě neproběhl.

Kompletní algoritmus je popsaný na:
`https://www.cs.princeton.edu/courses/archive/fall16/cos418/papers/raft.pdf`

Užitečná je i vizualizace na:
`https://raft.github.io/`

Zadání semestrální práce

Konsenzus pomocí Raft algoritmu

Naimplementujte algoritmus Raft pro replikaci key-value storu

Zpracování musí být **distribuované**, procesy si nesahají vzájemně do paměti!

Konsenzus pomocí Raft algoritmu

Naimplementujte algoritmus Raft pro replikaci key-value storu

Zpracování musí být **distribuované**, procesy si nesahají vzájemně do paměti!

Termín odevzdání je **7. 6. 23:59 CET**
(příp. do termínu Vaší zkoušky, pokud byste na ni šli před 7. 6.)

Podrobnosti upřesníme.

Díky za pozornost!

Budeme rádi za Vaši zpětnou vazbu! →

http://bit.ly/2VCUuJc