

Konsenzus v distribuovaném systému

B4B36PDV – Paralelní a distribuované výpočty

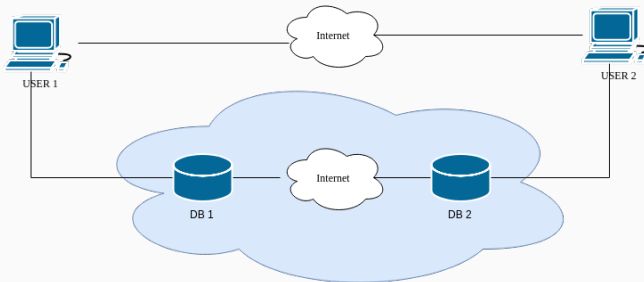
- Opakování z minulého cvičení
- Proč bychom mohli chtít řešit konsenzus v DS?
- RAFT algoritmus
- Zadání semestrální úlohy

Opakování z minulého cvičení

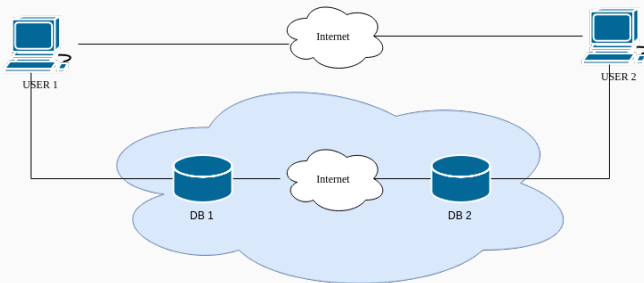
<http://goo.gl/a6BEMb>

Konsenzus v distribuovaném světě

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



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



Co se stane, když server zodpovědný za replikaci selže?

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

- ⚠ 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 uložena

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

 Části databáze si proto raději držíme v paměti

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?

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?

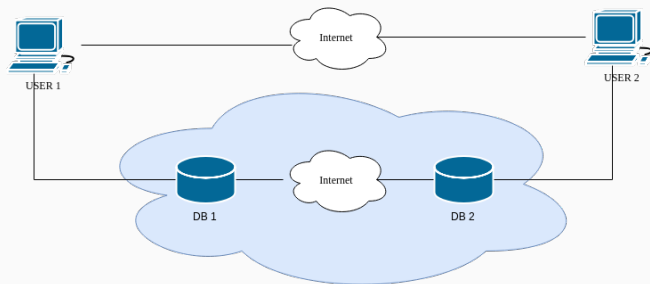
Jak bychom mohli použít journaling v distribuované DB?

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

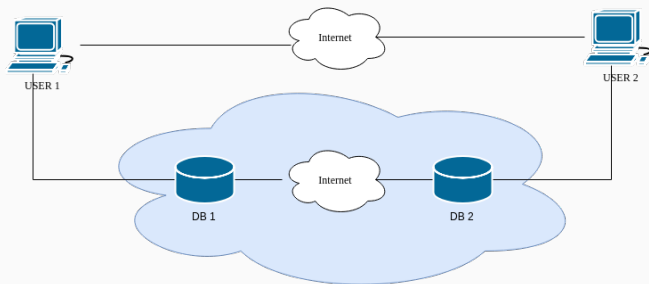
Jak řešit konkurenční požadavky?



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

Možnosti: Dohodnout se na globálním uspořádání

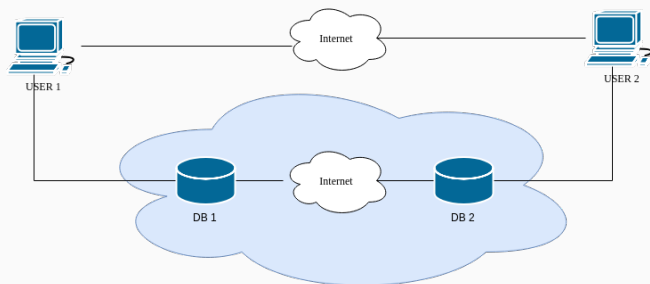
Jak řešit konkurentní požadavky?



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*)

Co když pak požadavek přijme ne-leader?

 Leader musí mít v DS autoritu!

Co to znamená?

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?

⚠ 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**

⚠ 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

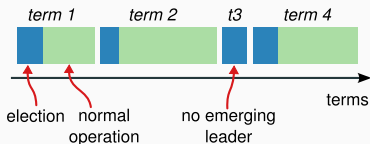
Přišel Vám RequestVote...

Kdy budete kandidáta respektovat?

Přišel Vám RequestVote...

Kdy budete kandidáta respektovat?

- Jeho požadavek musí být aktuální podle „logického času“

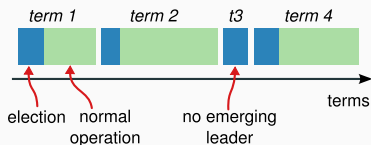


Aktuální **term** posíláme v každé zprávě.
`currTerm = max(currTerm, msg.term)`

Přišel Vám RequestVote...

Kdy budete kandidáta respektovat?

- Jeho požadavek musí být aktuální podle „logického času“



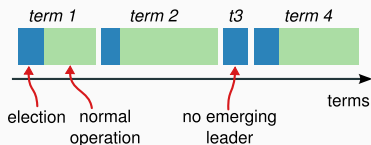
Aktuální **term** posíláme v každé zprávě.
$$\text{currTerm} = \max(\text{currTerm}, \text{msg.term})$$

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

Přišel Vám RequestVote...

Kdy budete kandidáta respektovat?

- Jeho požadavek musí být aktuální podle „logického času“

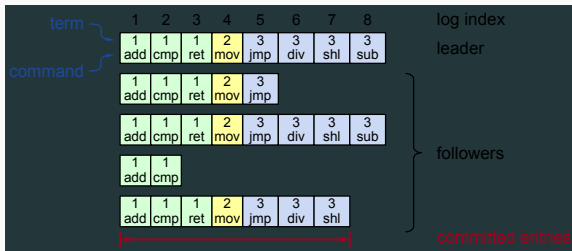


Aktuální **term** posíláme v každé zprávě.
`currTerm = max(currTerm, msg.term)`

- Jeho log obsahuje všechny potvrzené (*committed*) požadavky
To ale nedokážeme zjistit :-)

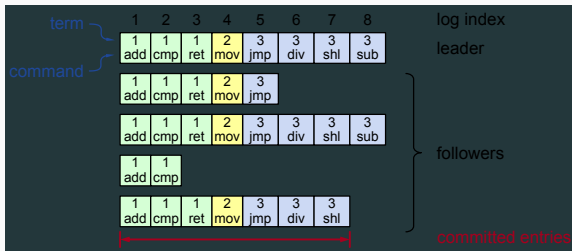
Jediné, co dokážeme zjistit je, jestli je daný log „aktuálnější“

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



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

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

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?

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

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

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

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 followeri?

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

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...

- V každém **termu** zvolíme maximálně jednoho leadera.

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

- V každém **termu** 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.

- V každém **termu** 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 popsán na:

<https://www.cs.princeton.edu/courses/archive/fall16/cos418/papers/raft.pdf>

- V každém **termu** 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 popsán 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

Naimplementujte algoritmus Raft pro replikaci key-value storu

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

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>