Algebra w służbie replikacji

Karol Marcjan, Zielona Góra, styczeń 2019

Systemy rozproszone

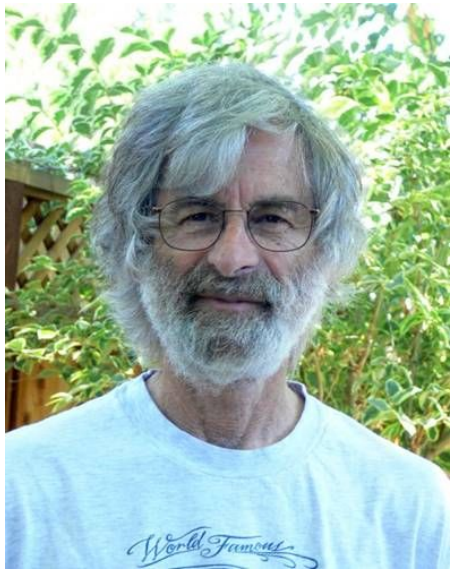
- wiele maszyn
- daleko od siebie
- wymieniają wiadomości
  - przesył rozciągnięty w czasie
  - nie każda zostanie dostarczona: at-most-once vs at-least-once

Systemy rozproszone

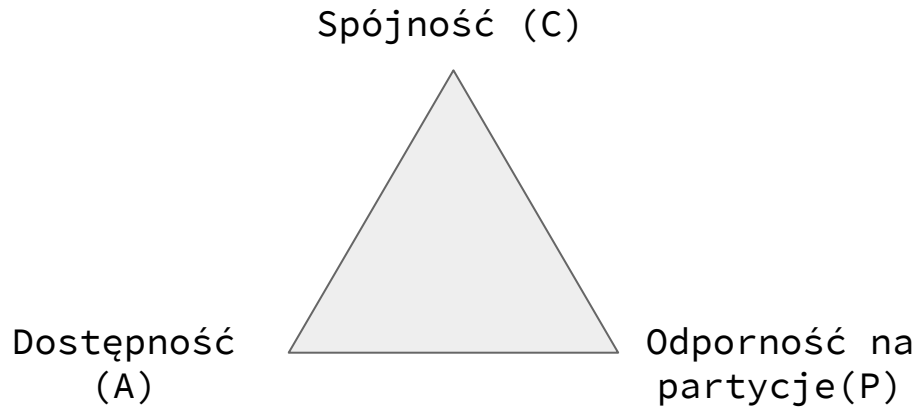
Systemy rozproszone

“System rozproszony to taki w którym problem z komputerem o istnieniu którego nie wiedziałeś może spowodować, że czegoś nie można zrobić na Twoim.”

- Leslie Lamport

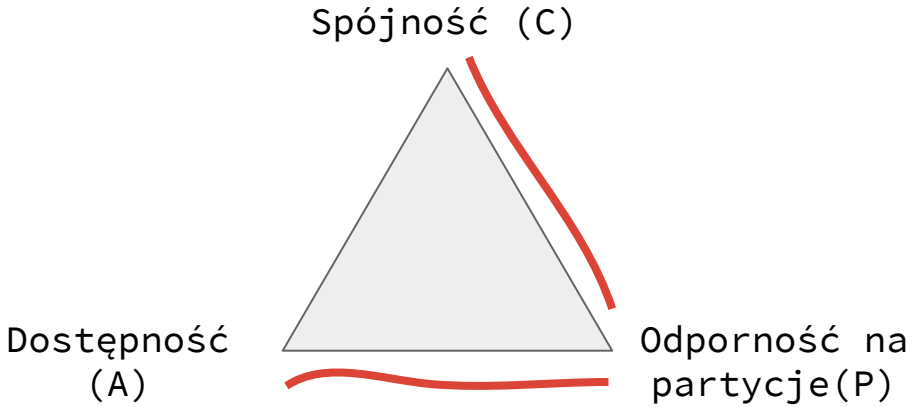


Twierdzenie CAP



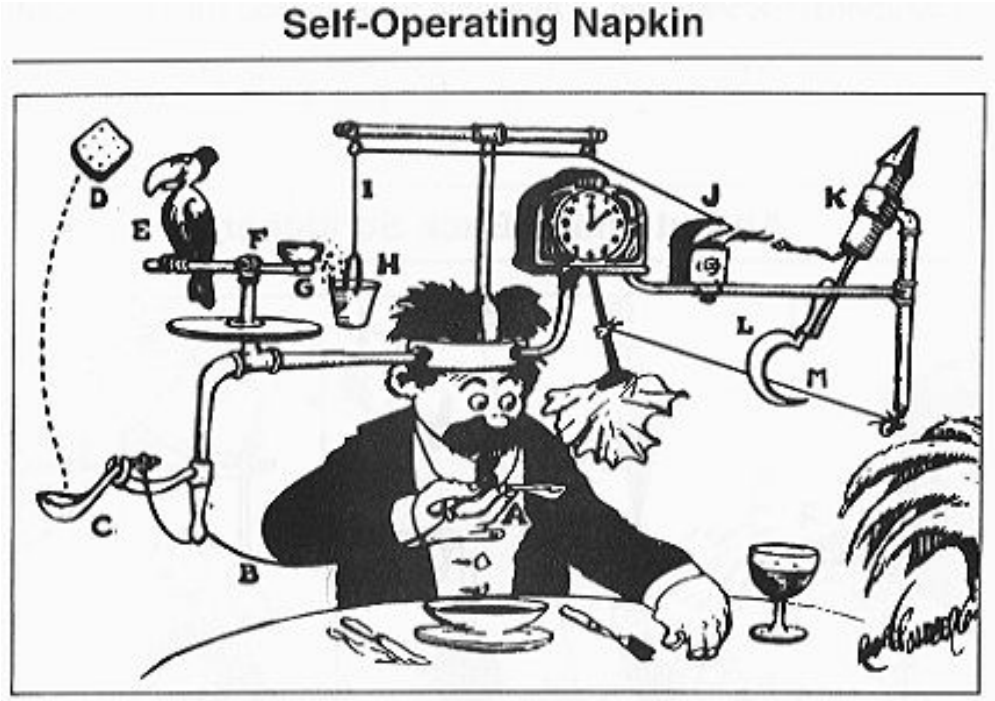
Gilbert, Lynch “Brewer’s Conjecture and the Feasibility of Consistent, Available, Partition-tolerant Web Services” ACM SIGACT News, Volume 33 Issue 2 (2002), pg. 51-59. doi:10.1145/564585.564601

Twierdzenie CAP



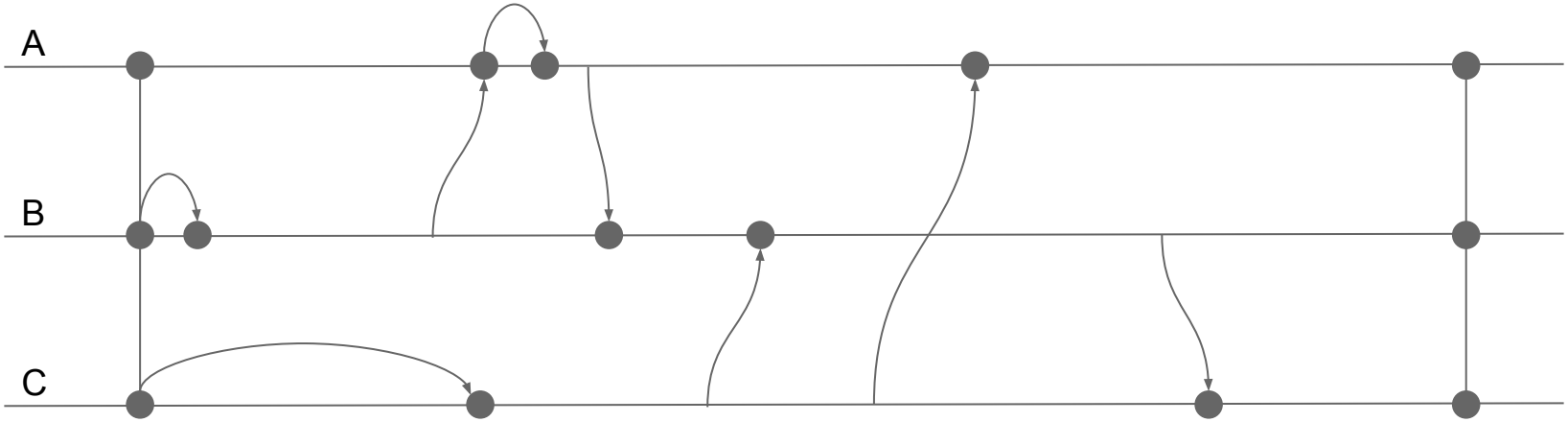
Gilbert, Lynch “Brewer’s Conjecture and the Feasibility of Consistent, Available, Partition-tolerant Web Services” ACM SIGACT News, Volume 33 Issue 2 (2002), pg. 51-59. doi:10.1145/564585.564601

Kepp it coordinated, stupid!



Keep it coordinated, stupid!

Spójność ewentualna



Spójność ewentualna

Wersja “zwykła”:

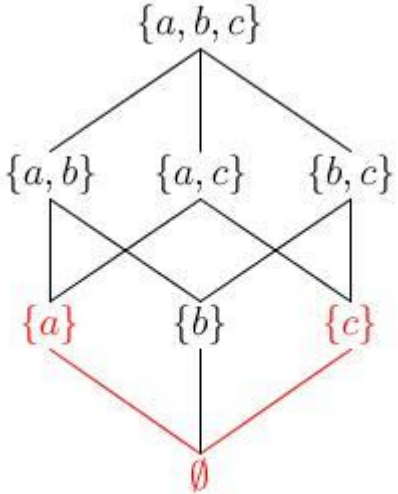
1. Klient wykonuje modyfikacje, na różnych maszynach, w różnych stanach.
2. Klient przestaje wykonywać modyfikacje.
3. Maszyny wymieniają się informacjami o zmianach.
4. Mija trochę czasu.
5. Wszystkie maszyny, zapytane, zwracają ten sam stan.

Wersja “silna”:

1. Klient wykonuje modyfikacje, na różnych maszynach, w różnych stanach.
2. Klient przestaje wykonywać modyfikacje.
3. Maszyny wymieniają się informacjami o zmianach.
4. ~~Mija trochę czasu.~~
5. Wszystkie maszyny, zapytane, zwracają ten sam stan.

Półkrata górna (join semilattice) - diagram

Półkrata górna (join semilattice) - diagram



$\{a\} \wedge \{c\}$

Półkrata górna (join semilattice) - algebra

zbiór S

operacja ^

a ^ (b ^ c) = (a ^ b) ^ c *(łączność)*

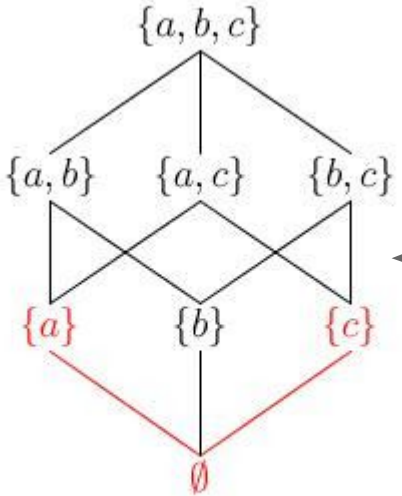
a ^ b = b ^ a *(przemienność)*

a ^ a = a *(idempotentność)*

Półkrata górna (join semilattice) - w Javie

```
interface JoinSemilattice<T> {

    T join(T remote);

}

assertThat(a.join(b.join(c)).isEqualTo(a.join(b).join(c)); // łączność

assertThat(a.join(a)).isEqualTo(a); // przemienność

assertThat(a.join(b)).isEqualTo(b.join(a)); // idempotentność
```

Półkrata górna (join semilattice) - operacje



$\{a\} \wedge \{c\}$

Kolejność!

Bezkonfliktowe zreplikowane typy danych (CRDT)

- Rozsyłamy stany:
  - Zbieżne Zreplikowane Typy Danych (Convergent Replicated Data Type)

- Rozsyłamy operacje:
  - Przemienne Zreplikowane Typy Danych (Commutative Replicated Data Type)

Menażeria typów

- Liczniki
- Rejestry
- Zbiory
- Słowniki
- Sekwencje
- Tekst
- JSON

Liczniki

- Rosnący (P-Counter)
  - Może rosnąć, zaczyna jako 0

- Rosnąco-malejący (PN-Counter)
  - Może rosnąć i maleć, zaczyna jako 0
  - Nie da się ograniczyć do określonego zakresu

Zbiory

- G-Set
  - Element można tylko dodać
- 2P-Set / U-Set
  - Element można dodać i usunąć, po usunięciu nie można dodać go ponownie
- PN-Set
  - Liczba zaobserwowanych dodań i usunięć decyduje
- OR-Set
  - Elementy można dodawać i usuwać
  - Współbieżne “dodanie” wygrywa z usunięciem

Rejestry

- Wielowartościowy (MV-Register)
  - Może przechowywać kilka współbieżnie ustawionych wartości
  - Następna ustawiona wartość “wygrywa” z wszystkimi widocznymi

- Najnowszy-wygrywa (LWW-Register)
  - Decyduje czas
  - Równe czasy => wybiera tę samą wartość na każdej maszynie

Use case'y

- League of Legends - chat https://engineering.riotgames.com/news/chat-service-architecture-persistence
- Amazonie - koszyk trzymany w rejestrze MV
- Nawigacje TomTom - ulubione lokalizacje użytkownika
  https://speakerdeck.com/ajantis/practical-demystification-of-crdts
- Git - graf commitów
- Teletype - zdalny pair programming w Atomie
  https://blog.atom.io/2017/11/15/code-together-in-real-time-with-teletype-for-atom.html
- Trellis - Trello bez serwera https://github.com/automerge/trellis
- Pixelpusher - edytor pixel artu dla wielu użytkowników, bez serwera
  https://github.com/automerge/pixelpusher

Twierdzenie CALM

Consistency As Logical Monotonicity

https://arxiv.org/abs/1901.01930

Question: What is the family of problems that can be *consistently computed in a distributed fashion without coordination*, and what problems lie outside that family?

Answer: A program P is monotonic if for any input sets S, T where S ⊆ T, P(S) ⊆ P(T).

A program has a consistent, coordination-free distributed implementation if and only if it is monotonic.

Klocki do zabawy

- Riak (baza danych; wspiera kilka typów CRDT)
- Automerge (JS)
- CosmosDB (baza danych od Microsoftu, tylko w Azure)
- Akka Distributed Data (Java/Scala, nie działa poza Akką)
- Phoenix (framework webowy dla Elixira)
- AntidoteDB
  (baza danych, rozwój finansowany m. in. z grantów UE, Erlang, inne klienty niestabilne)

Wyzwania

(Bo PM ma alergię na problemy.)

Wyzwania

- Keeping CALM: Nie każdy program spełnia warunki twierdzenia CALM.
  - Licznik rosnąco-malejący >= 0
  - Możliwe != łatwe
- Testowanie
  - Testy generatywne - stosunkowa nowość.
- Undo/redo
  - Przykład: Rejest wielowartościowy i wiele wartości przed/po
- Częściowa replikacja
  - Czasem nie każda maszyna potrzebuje całego stanu
  - Komplikuje implemetację
- Odśmiecanie
  - Niektóre CRDT muszą pamiętać całą swoją przeszłość żeby działać

Dzięki!

Pytania?