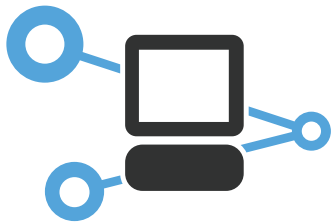
Bazy danych NoSQL

wprowadzenie

Szymon Francuzik
szymon.francuzik@cs.put.poznan.pl

Poznań, 16.05.2012



Distributed Systems Group

Plan prezentacji

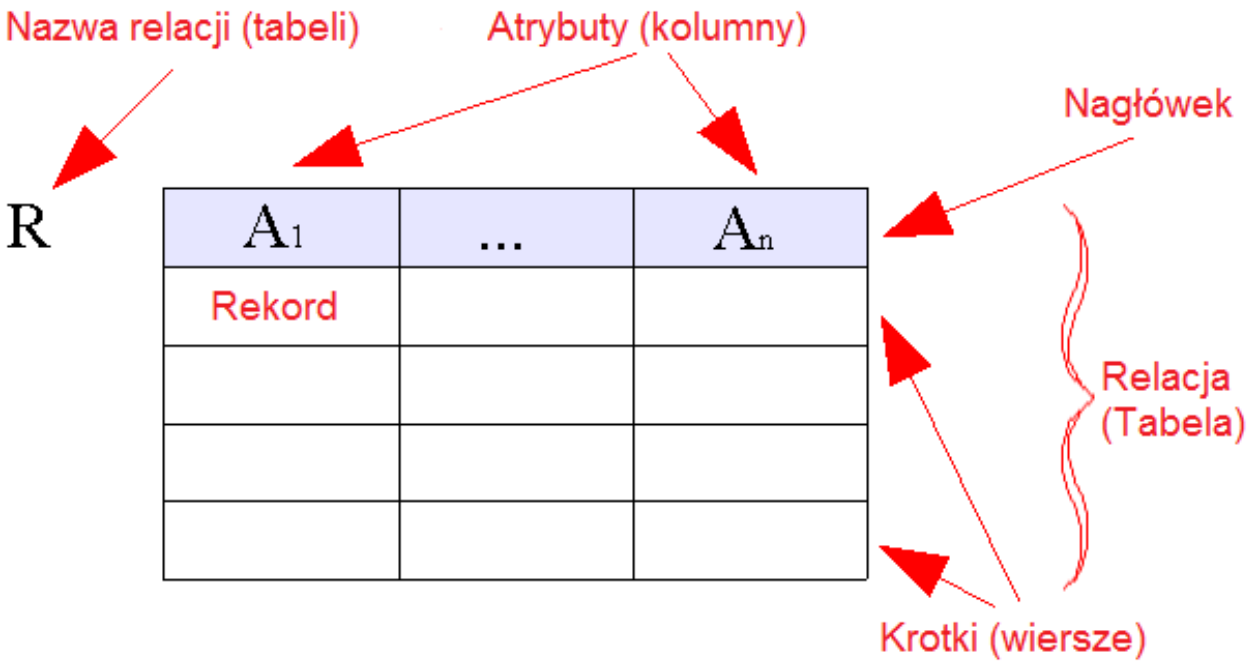
1. Motywacja
   - Relacyjne bazy danych
   - Problemy relacyjnych baz danych
2. NoSQL
   - Definicja
   - Typy baz NoSQL
3. Amazon Dynamo
   - Przeznaczenie
   - Realizacja
4. Cassandra
   - Przeznaczenie
   - “Rozpraszanie”
   - API

RDBMS

Relacyjne systemy zarządzania bazami danych:

- relacyjny model danych — Edgar Codd (IBM) — 1970
- język SQL
- transakcje
- ACID

Model relacyjny



ACID

- Atomicity (atomowość)
- Consistency (spójność)
- Isolation (izolacja)
- Durability (trwałość)

Wydajność

- atomowość — konieczność logowania operacji
- spójność — weryfikacja poprawności
- izolacja — blokowanie/wycofywanie
- trwałość — zapis do pamięci trwałej

Skalowalność

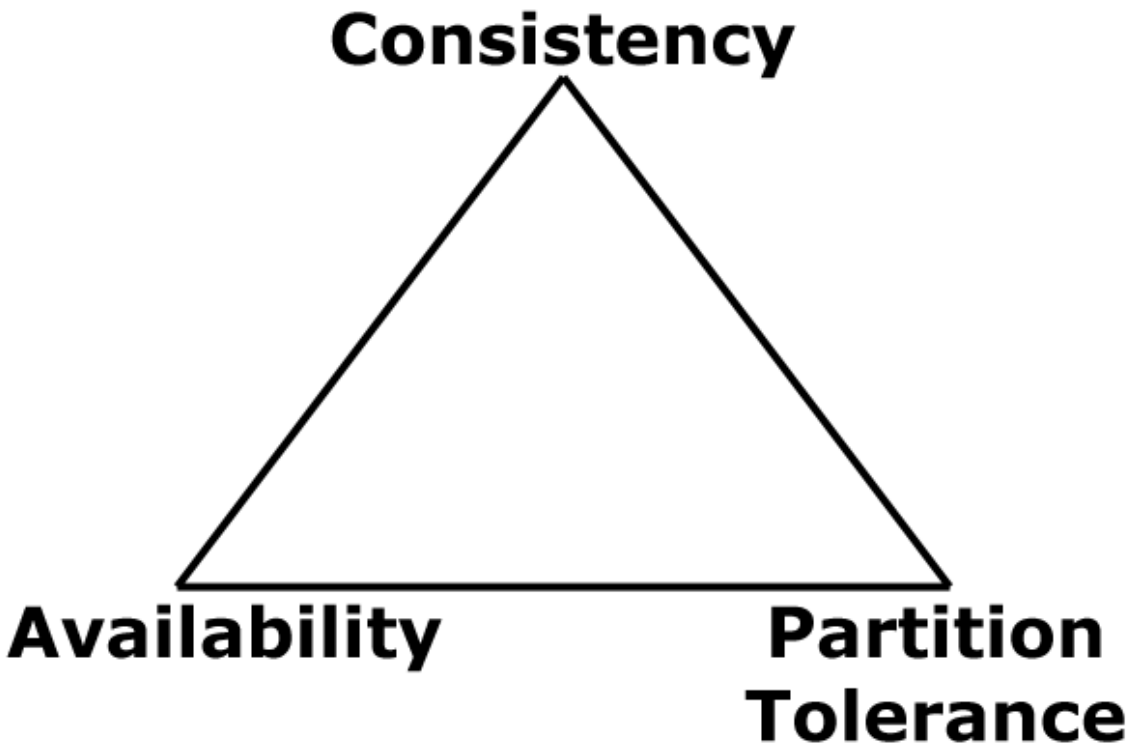
Możliwość zwiększania wydajności systemu wraz z rosnącym zapotrzebowaniem.

- skalowanie wertykalne (ang. scale up)
- skalowanie horyzontalne (ang. scale out)
  - sharding (partycjonowanie poziome)
  - partycjonowanie pionowe
  - repliki do odczytu

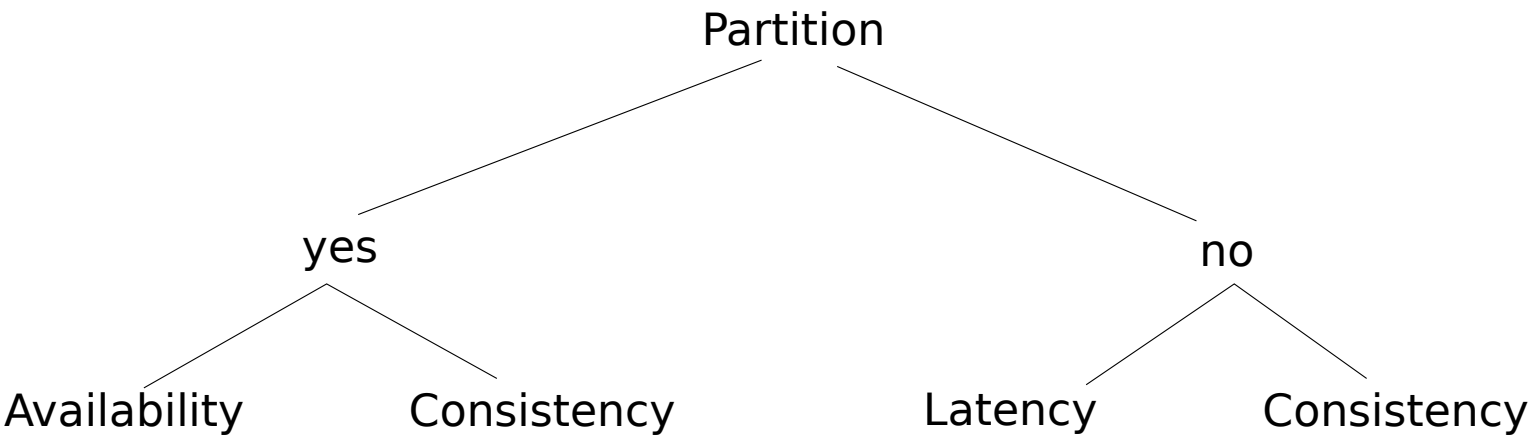
Model danych

- relacyjny model danych nie zawsze wygodny
  - normalizacja vs. efektywność
  - brakujące dane
- trudności przy zmianie schematu danych

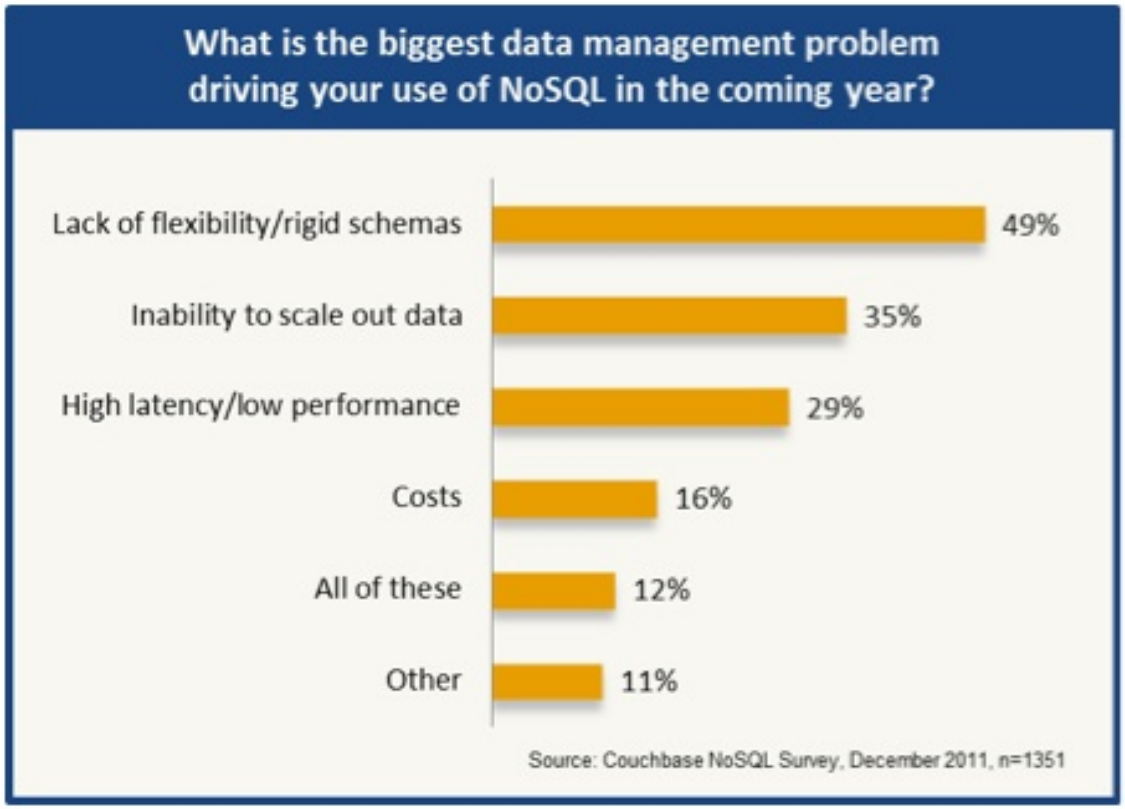
CAP Theorem (Brewer's conjecture)



PACELC


Partition
yes
no
Availability
Consistency
Latency
Consistency

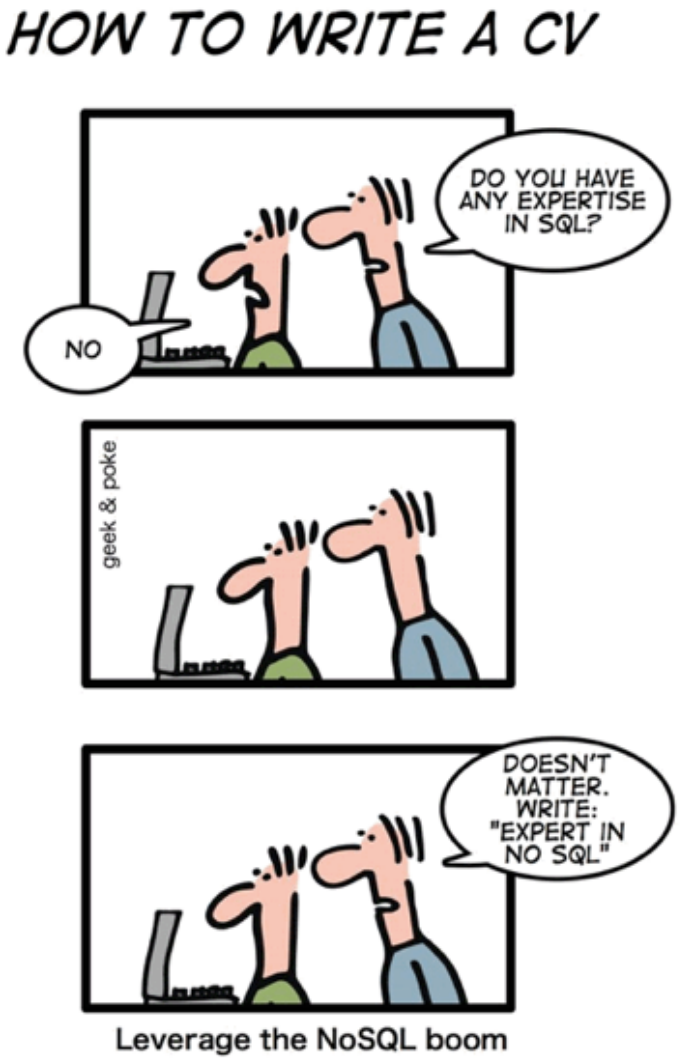
Problemy podsumowanie



Definicja NoSQL

Wikipedia:
NoSQL jest klasą systemów zarządzania bazą danych nie pasujących do powszechnie stosowanego modelu relacyjnych baz danych:

- brak języka SQL (w szczególności brak operacji JOIN)
- nierelacyjny model danych
- nie musi zapewniać ACID
- rozproszona, odporna na awarie architektura



Alternatywa dla ACID

BASE:

- **B**asic **A**vailability
- **S**oft-state
- **E**ventual consistency

Podział ze względu na reprezentację danych

- klucz-wartość
- hierarchiczna struktura klucz-wartość (“BigTable-like”)
- dokumentowe
- grafowe

Bazy klucz-wartość

- przechowują pary klucz-wartość
- dostęp do danych jedynie po kluczu
- przykłady:
  - Berkeley DB
  - Riak
  - Dynamo

Bazy z hierarchiczną strukturą klucz-wartość

- wzorowane na BigTable (Google)
- każdy wiersz może mieć przyporządkowany inny zestaw kolumn
- częściowo ustrukturalizowane
- przykłady:
  - HBase
  - Cassandra
  - SimpleDB

row-key	columnfamily1					columnfamily1	
	supercolumn1		supercolumn2			col21	col22
	col1	col2	col1	col3	...	val5	val6
	val1	val2	val3	val4	...		

Dokumentowe bazy danych

- przechowuje dokumenty zamiast wierszy/rekordów
- dokument: wpis w bazie składający się z pól (nazwa-wartość)
- możliwość odwoływania się po polach nie będących kluczem podstawowym
- przykłady:
  - CouchDB
  - MongoDB
  - ThruDB

```
{ imie:        "Jan",
  nazwisko:    "Kowalski",
  nr_indeksu: 98765,
  oceny:       [5, 4.5, 3, 4]
  dzienny:     true }

db.students.find({nazwisko:
                  "Kowalski"})
```

Grafowe bazy danych

- węzły, krawędzie (łuki), własności
- szybki dostęp do powiązanych danych
- przykłady:
  - HyperGraphDB
  - Neo4J
  - Trinity

Amazon Dynamo

- baza typu klucz-wartość
- stworzona do zarządzania stanem usług oferowanych przez Amazon:
  - koszyk zakupów: dziesiątki milionów zapytań, do 3 mln transakcji zakupu dziennie
  - sesje klientów: setki tysięcy równoległych sesji
  - inne usługi: katalog produktów, system rekomendacji
- rozproszona architektura — serwery rozproszone po całym świecie
- spójność ostateczna

Cele

- niezawodność (dostępność) — nawet kosztem spójności
- skalowalność pozioma — commodity hardware
- SLA — ograniczenia na czas odpowiedzi

Dlaczego nie RDBMS

- dostęp do danych poprzez klucz podstawowy — nie potrzeba modelu relacyjnego
- brak operacji odwołujących się do kilku elementów — nie potrzeba transakcji
- RDBMS wymaga specjalistów i drogiego sprzętu
- ograniczona dostępność w przypadku awarii
- słaba skalowalność pozioma — brak wsparcia dla automatycznego partycjonowania

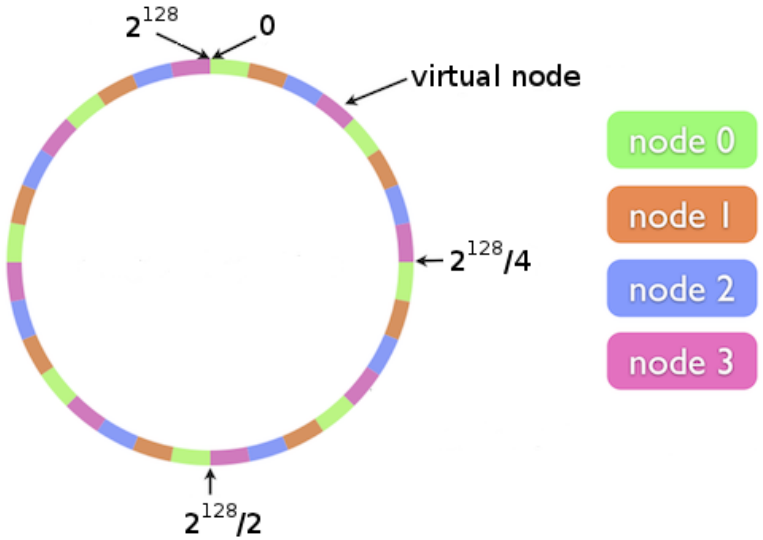
Interfejs

- komunikacja po HTTP
- operacje:
  - get(key) — odsyła wartość oraz kontekst
  - put(key, context, value)

Partycjonowanie danych

consistent hashing:

- na podstawie klucza generowany hash MD5 (128b)
- przestrzeń haszy podzielona między wirtualne serwery
- do każdego fizycznego serwera przypisana określona liczba serwerów wirtualnych

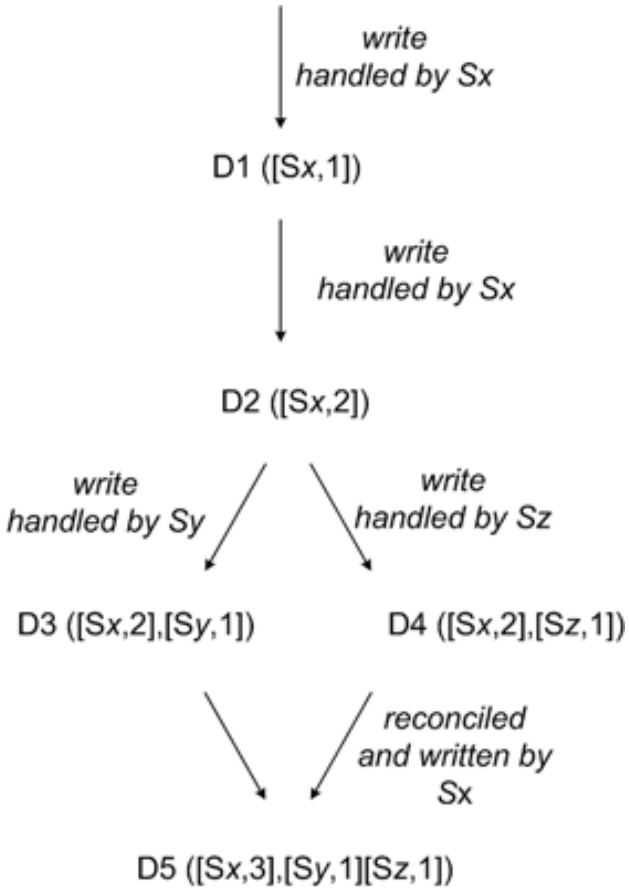


Replikacja

- możliwość zdefiniowania ilości replik
- replikacja wpisu na N kolejnych replik w pierścieniu
- lista preferencji replik
- możliwość ustawienia parametru R i W — sloppy quorum
- uspójnianie stanu:
  - uspójnianie przy odczycie
  - anti-entropy
  - hinted handoff

Wielowersyjność obiektów

- wersja obiektu opisywana przez zegar wektorowy:
  $TS_1 : \{(S_1,3), (S_2,5), (S_3,1)\}$
  $TS_2 : \{(S_1,3), (S_2,4), (S_3,2)\}$
  $TS_3 : \{(S_1,3), (S_2,6), (S_3,2)\}$
- przechowywanie wersji powstałych współbieżnie do czasu uspójnienia przez klienta
- przy odczycie klient otrzymuje wszystkie dostępne wersje obiektu



Cassandra

- hierarchiczna struktura klucz wartość
- Facebook — indeksowanie na potrzeby przeszukiwania skrzynki użytkownika
  - setki milionów użytkowników
  - 600+ rdzeni
  - rozmiar indeksu: 120+ TB
- open source

Struktura danych

- przestrzeń kluczy (keyspace)
- rodzina kolumn (column family) — zdefiniowane statycznie
  - super kolumny (super columns)
  - kolumny (columns)
- wiersz (row) — pojedynczy element z przypisanymi kolumnami/super-kolumnami

row-key	columnfamily1					columnfamily1	
	supercolumn1		supercolumn2			col21	col22
	col1	col2	col1	col3	...	val5	val6
	val1	val2	val3	val4	...		

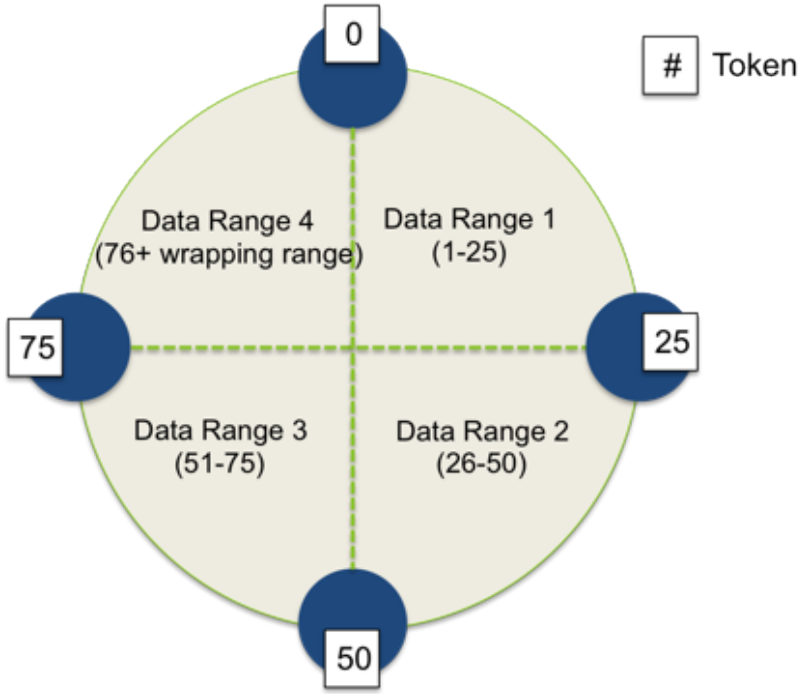
Reprezentacja danych

- klucz i wartość reprezentowana jako tablica bajtów
- kolumny i super-kolumny posortowane po nazwach lub czasie modyfikacji
- indeksowanie kluczy

Indeksowanie

Adam	index_by_users_interactions					
	Marek		Kasia			...
	m100	m252	m124	m511	m600	...
	-	-	-		-	...

Partycjonowanie danych



- consistent hashing
- dynamiczne równoważenie obciążenia
- dwie strategie podziału:
  - random partitioner
  - ordered preserving partitioner

Replikacja

- możliwość konfigurowania liczby replik
- strategia rozmieszczania replik:
  - przydział prosty
  - świadoma topologi sieci: rack aware, datacenter aware

Kontrola poziomu spójności

Zapisy:

- any
- one
- quorum
- local_quorum
- each_quorum
- all

Odczyty:

- one
- quorum
- local_quorum
- each_quorum
- all

Uspójnianie replik

- uspójnianie przy odczycie
- anti-entropy
- hinted handoff

Interfejs

- Thrift — RPC framework for cross-language service development
- wspierane języki:
  C++, Java, Python, PHP, Ruby, Erlang, Perl, Haskell, C#, Cocoa, JavaScript, Node.js, Smalltalk, OCaml
- gotowe biblioteki dla wybranych języków programowania

Bibliografia

http://nosql-database.org/

NoSQL Your Ultimate Guide to the Non - Relational Universe!

[the **best selected** nosql link Archive in the web]
...never miss a conceptual article again...
News Feed covering all changes here !

NoSQL DEFINITION: Next Generation Databases mostly addressing some of the points: being **non-relational, distributed, open-source** and **horizontally scalable**. The original intention has been **modern web-scale databases**. The movement began early 2009 and is growing rapidly. Often more characteristics apply such as: **schema-free, easy replication support, simple API, eventually consistent** / **BASE** (not ACID), a **huge amount of data** and more. So the misleading term *"nosql"* (the community now translates it mostly with "**not only sql**") should be seen as an alias to something like the definition above. [based on 5 sources, 11 constructive feedback emails (thanks!) and1 disliking comment. Agree / Disagree? Tell me so! By the way: this is a strong definition and it is out there since 2009!]

LIST OF NOSQL DATABASES [currently 122+]

Core NoSQL Systems: [Mostly originalted out of a Web 2.0 need]

Wide Column Store / Column Families

Hadoop / HBase: API: **Java / any writer**, Protocol: **any write call**, Query Method: **MapReduce Java / any exec**, Replication: **HDFS Replication**, Written in: **Java**, Concurrency: ?, Misc: **Links**: 3 Books [1, 2, 3]

Cassandra: API: **many Thrift » languages**, Protocol: ?, Query Method: **MapReduce**, Replicaton: , Written in: **Java**, Concurrency: **eventually consistent** , Misc: like "Big-Table on Amazon Dynamo alike", initiated by Facebook, Slides » , Clients », Installation »

Hypertable: API: **Thrift** (Java, PHP, Perl, Python, Ruby, etc.), Protocol: **Thrift**, Query Method: **HQL, native Thrift API**, Replication: **HDFS Replication**, Concurrency: **MVCC**, Consistency Model: **Fully consistent** Misc: High performance C++ implementation of Google's Bigtable. Commercial support »

Accumulo: Accumulo is based on **BigTable** and is built on top of Hadoop, Zookeeper, and Thrift. It features improvements on the BigTable design in the form of **cell-based access control**, improved **compression**, and a server-side programming mechanism that can modify key/value pairs at various points in the data management process.

NoSQL matters

EVENTS

21-23 Aug NoSQL Now! San Jose »
29+30May **NoSQL matters** Cologne »
22th May **GOTO** CPH »
9-11 May SkillsMatter Tutorials »
Mongo all around the world »
20th Apr NoSQL **RoadShow Zürich** »

All past NoSQL Conferences »
register your event here! »

NoSQL Summer in 27 cities »

Conceptual quality articles, blogs, links, research papers, etc.

NoSQL

ARCHIVE

NoSQL **User Stories** »

Announcing the availability of

Bibliografia

http://nosqltapes.com/

Bibliografia

http://nosqlsummer.org/