

Pihi CouchDB-vel és RelaxDB- vel

2009.08.27 NoSQL budapest.rb
meetup

Érdi Bálint

twitter: @baaz

blog: <http://bucionrails.com>

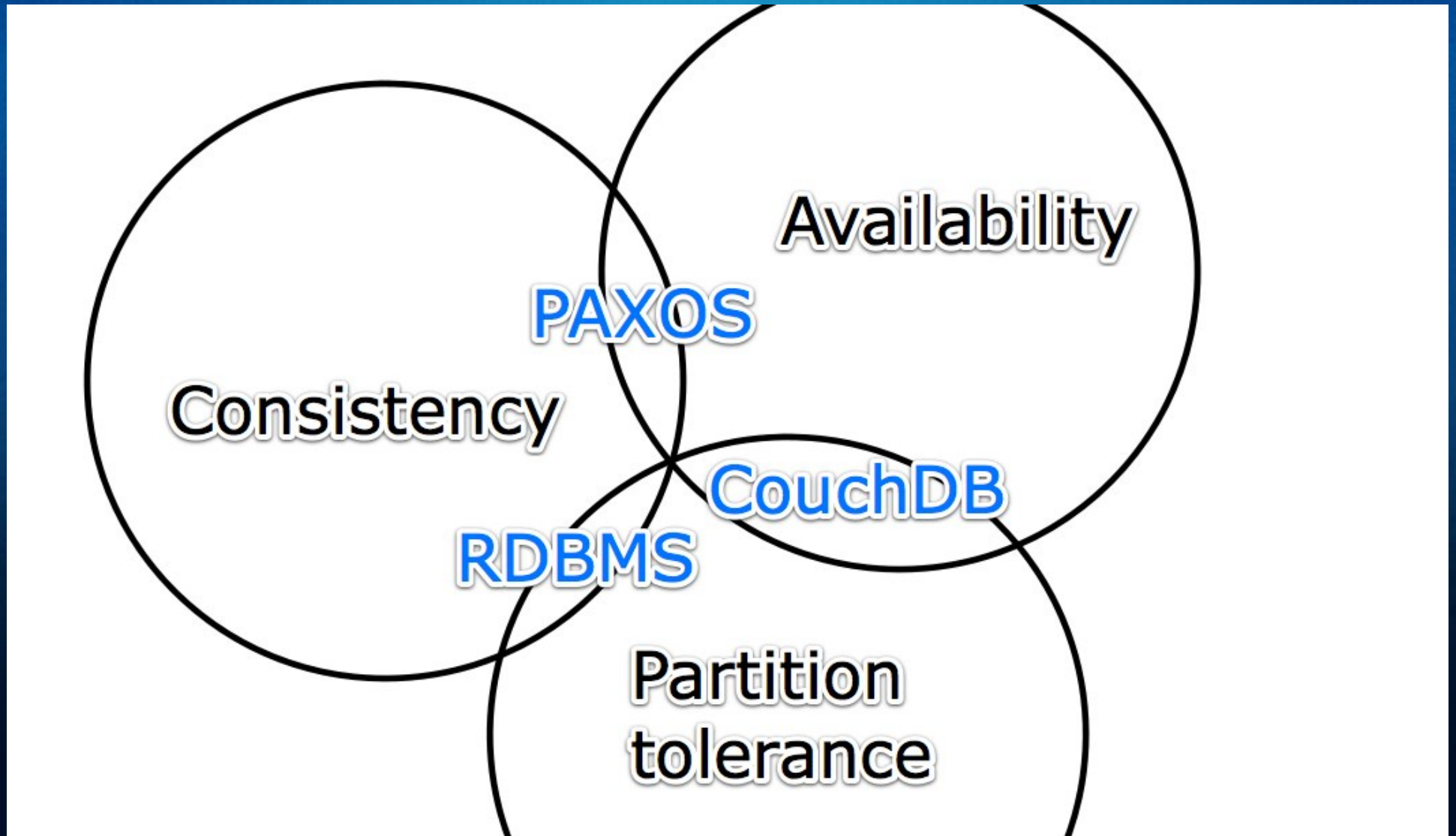
Miről lesz szó?

- CAP elmélet
- DBMS megfontolások
- CouchDB alapok
- RelaxDB
- Példaprogram
- Kérdések

CAP elmélet

- Consistency: mindenki ugyanazt az adatot látja
- Availability: mindenki hozzá tud férni az adat valamilyen verziójához
- Partition tolerance: az adatbázis elosztható több szerverre

CAP elmélet



Consistency vs. Availability

Webes trendek

- min. 335435139 reqs/sec
- skálázhatóság fontos
- egy nagy szerver drága, nem hibatűrő (SOP) és nem bővíthető akármeddig
- clustering to the rescue!

Consistency vs. Availability

Ha viszont clustering, akkor:

- hogyan biztosítjuk, hogy szinkronban legyenek?
- akár pár ms különbség lehet egy írás és olvasás között, ami ugyanazt a rekordot (v. dokumentumot) érinti
- agreement protokollok
- a hálózat méretével nő a probléma mérete is
- ha az availability fontosabb, mint a globális consistency: eventual consistency!

Locking vs. MVCC

(Pessimistic) Locking:

- egy write lockolja a rekordot, a beérkező readek várnak
- a beérkező kérések sorba rendezése miatt értékes idő vész el
- nem jól skálázható ill. clusterezhető

Locking vs. MVCC

MVCC (Multi Version Concurrency Control):

- nincs locking
- minden read a dokumentum aktuális verzióját kapja meg
- minden write új verziót hoz létre a dokumentumból
- csak a legfrissebb verziójú dok. írható
- az ezután beérkező write-ok az új verziót látják

Mi az a CouchDB?

"CouchDB is a distributed, fault-tolerant and schema-free document-oriented database accessible via a RESTful HTTP/JSON API."

MVCC: hogy csinálja a CouchDB?

Eventual consistency:

- minden szervernek local copy-ja van
- inkrementális replikáció a szerverek között
- minden adatváltoztatáskor teljesen új verzió keletkezik, új `_rev` mezővel
- a dokumentum írása akkor sikeres, ha a kérésben megadott `_rev` a legfrissebb

Adattárolás

Minden JSON-be szerializálódik

Minden B-treeben tárolódik:

- adatok
- design dokumentumok (bennük view-k)
- view-k eredményei

Query key-jel v. key range-dzsel:

- Pontosan megfelel a B-tree belső tárolási módjának -> nagyon gyors
- $O(\log N)$

Sémaság vs. sémanélküliség

A sematikus adatbázisok:

- előre kell tervezni
- viszonylag nehéz változtatni

A valóság sokszor nem sematikus

CouchDB:

- self-contained data
- real-world documents
- csak amire szükség van
- incremental id helyett automatikus uuid

HTTP API: "built of the web"

- query nyelv: HTTP
- megfelelő HTTP igék használata:
 - SELECT: GET
 - CREATE, UPDATE: PUT
 - DELETE: DELETE
- resource URL-ek
- javascript view-k

```
GET http://127.0.0.1:5984/_all_dbs
```

```
PUT http://127.0.0.1:5984/budapest_rb
```

```
DELETE http://127.0.0.1:5984/budapest_rb
```

View-k

- MapReduce funkció pár
- a DB design_documentjében tárolódik
- Javascript-ben írva
- eredménye B-treeben tárolódik

Map

- minden dokumentumon végigmegy először
- ha változik egy dokumentum, frissül a view
- egy doksihoz $\{0..n\}$ db key-value párt köp ki
- key szerint van rendezve a view

Reduce

- nem kötelező
- aggregáló funkciót lát el
- inputja lehet
 - a Map által generált sor
 - előző Reduce által előzőleg generált sor
 -

Egyéb query paraméterek

- key
- startkey, endkey
- limit
- group
 - true: unique key-enként összegez (~GROUP_BY)
 - false: egyetlen érték a Reduce funkció eredménye
- group_level
- reduce
 - true : default
 - false : nem futtaja a Reduce funkciót
- descending

RelaxDB: pozi- és negatívumok

- + "simple Ruby interface to CouchDB": tényleg!
- + nem akar SQL lenni
- + a fontos dolgokban segít
- + jó tanuló eszköz

- kicsit lanyha a fejlesztése
- CouchDB ≥ 0.9

- + könnyen fejlesztheted te is! ;)

RelaxDB: feature-ök

RelaxDB::Document

- property :title, :validator => :required
- references :from, :validation_msg => "-t nem adtad meg"
- has_n :items
- view_by :title
- life cycle hook-ok (before_save, after_save, etc.)
- néhány validáció típus

Egyéb Ruby library-k CouchDB-hez

- CouchRest
- CouchPotato
- CouchFoo
- ActiveCouch
- CouchObject
- CouchApp

CouchDB és Rails, sitting in a tree?

Egyelőre felejtse el

- a findereket
- a sok-sok kényelmes validátort
- a (form) helperek egy jó részét
- a kedvenc pluginedet (ó, authlogic...)

Jövő:

- DataMapperhez van CouchDB adapter
- ha a Rails majd megy DM-mel...

Demo app

?Kérdések?

Referenciák

Apache CouchDB wiki: <http://wiki.apache.org/couchdb>

CouchDB : The definitive guide: <http://books.couchdb.org/relax/>

NoSQL debrief: <http://blog.oskarsson.nu/2009/06/nosql-debrief.html>

RelaxDB: <http://github.com/paulcarey/relaxdb/tree/master>

CouchDB joins: <http://www.cmlenz.net/archives/2007/10/couchdb-joins>

Something blue (a példa app): <http://github.com/balinterdi/something-blue>

Ruby on the Couch eating potatoes (videó): <http://www.engineyard.com/blog/community/scotland-on-rails/>