Napredni modeli i baze podataka

predavanja
siječanj 2016.

NoSQL
dodatak

Konzistencija: detekcija konflikta

Oznake verzija (*version stamps*)

- Oznaka verzije: polje koje se mijenja kod svake promjene odgovarajućih podataka, npr. HTTP ETag
- Opcije za generiranje oznake verzije:
  1. **Brojač**
     - ✓ Zna se koja je novija
     - ✗ Single master za generiranje
  2. **GUID**
     - ✓ Može ga ga generirati svatko, nema duplikata
     - ✗ Ne zna se tko je noviji
  3. **Hash**
     - ✓ Može ga generirati svatko; ako je dovoljno velik ~ GUID
     - ✓ Deterministički - isti hash za isti sadržaj (u različitim čvorovima)
     - ✗ Ne zna se tko je noviji
  4. **Vremenska oznaka (*timestamp*)**
     - ✓ Malen, zna se tko je noviji
     - ✗ Sinkronizacija satova, granularnost (npr. jesu li milisekunde dovoljne)?
  5. **Neka kombinacija prethodnih** (npr. CouchDB koristi (1) brojač + (3) hash)

Poredak i uzročna ovisnost događaja

- U distribuiranom sustavu, želimo znati:
  - Poredak događaja
  - Potencijalne konflikte
- Označiti sve događaje s vremenskom oznakom (fizički sat)?
  - Problem sinkronizacije satova (npr. kvarcni sat griješi ~ 1s/11.6 dana)
  - Ne možemo odrediti uzročnu ovisnost (*eng. causality*)
- Poredak (*happened-before*, *Lamport 1978.*), možemo znati samo na temelju:

  (a) redoslijed **internih** događaja istog procesa

  (b) redoslijed **slanja** i **primanja** iste poruke
- Dakle, e ε {internal, send, receive}
- **Definicija**:

  Uzročna relacija *happened-before* (označava se s →) definira se kao:

  i. Ako postoji takav proces p_i : e $\rightarrow_i$ e' , tada e → e'

  ii. Za svaku poruku m, *send(m)* → *receive(m)*

  iii. (tranzitivnost) Ako su e, e' i e'' takvi događaji takvi da je e→e' i e'→e'' tada je i e→e''.

Logički sat

- Logički satovi:
  - Služe za određivanje redoslijeda događaja
  - Događaju e dodjeljuje broj C(e) (~ fizičkom vremenu)
- Konzistencija satova:
  - Slaba (kronologija): $e_1 \rightarrow e_2 \Rightarrow C(e_1) < C(e_2)$
  - Jaka/stroga (uzročna ovisnost): $C(e_1) < C(e_2) \Rightarrow e_1 \rightarrow e_2$

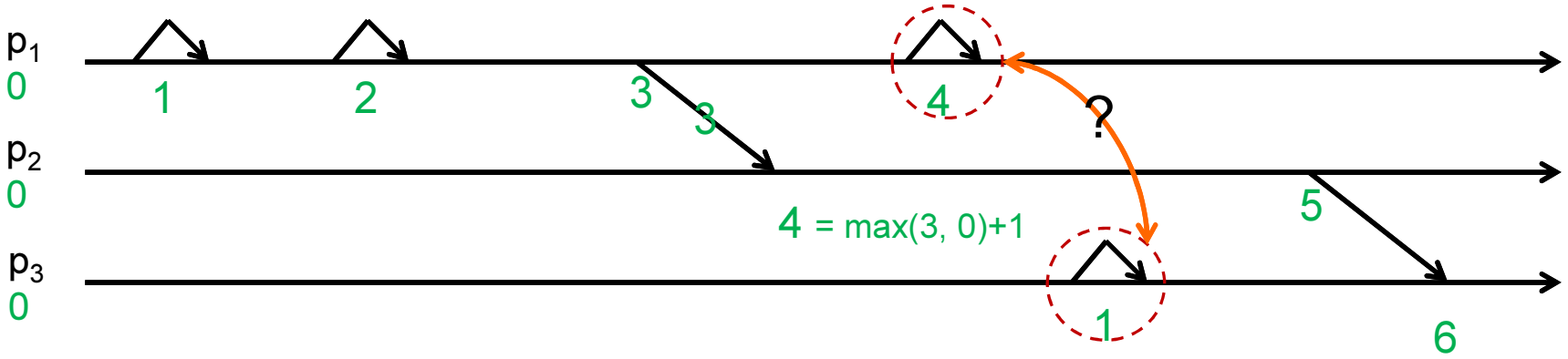
Lamportov sat

- Svaki proces ima svoj interni logički sat c
- Algoritam:
  - Kod internog događaja: ++c
  - Kod slanja poruke ++c i uključiti c u poruku
  - Kod primanja poruke s vrijednošću sata c':

$$c = 1 + \max(c, c')$$

```
public class LamportClock {
   int c;
   public LamportClock() {
      c = 1;
   }
   public int getValue() {
      return c;
   }
   public int tick() { //on internal events
      return ++c;
   }
   public void sendAction() {
      c++;
      //include c in message
   }
   public void receiveAction(int sentValue){
      c = Util.max(c, sentValue) + 1;
   }
}
```

Lamportov sat - primjer



- Nedostatak:
  - Slaba (kronologija): $e_1 \rightarrow e_2 \Rightarrow C(e_1) < C(e_2)$
  - ~~Jaka (uzročna ovisnost): $C(e_1) < C(e_2) \Rightarrow e_1 \rightarrow e_2$~~
  - Razlog zašto ne vrijedi obrat je to što logički sat stanjima pridružuje brojeve, a brojevi se uvijek mogu usporediti. Na taj način će dva neusporediva stanja izgledat usporediva.
- Ako negiramo: $C(e_1) >= C(e_2) \Rightarrow e_1 \not\rightarrow e_2$
  ili drugim rječima:
  $C(e_1) >= C(e_2)$ => e_1 se možda dogodio prije e_2,
  ili nisu usporedivi po →,
  ali svakako se e_1 nije dogodio prije e_2

Vektorski sat (1)

- Lamportov sat -> nedovoljno dobro za distribuirane (paralelne) procese
- Mattern [1989] i Fidge [1991], želi se ostvariti:
  - Slaba: $e_1 \rightarrow e_2 \Rightarrow V(e_1) < V(e_2)$
  - Jaka/stroga: $V(e_1) < V(e_2) \Rightarrow e_1 \rightarrow e_2$
- Opisuje **uzročno posljedične** veze
- Ideja: svaki proces ima svoj brojač (sat)
  - $V_p[p]$ broj događaja koje je ostvario proces **p**
  - $V_p[m]$ broj događaja za koje proces **p** smatra da ih je ostvario proces **m**

Primjer

- P_1 je zbilja obavio 7 događaja, P_2 smatra da je obavio 5, P_3 smatra da je obavio 6.
- P_2 je zbilja obavio 8 događaja, P_1 smatra da je 8, P_3 smatra da je 4
- P_3 je zbilja obavio 12 događaja, P_1 i P_2 smatraju da je 9

	1	2	3
P_1	7	8	9
P_2	5	8	9
P_3	6	4	12

Vektorski sat (2)

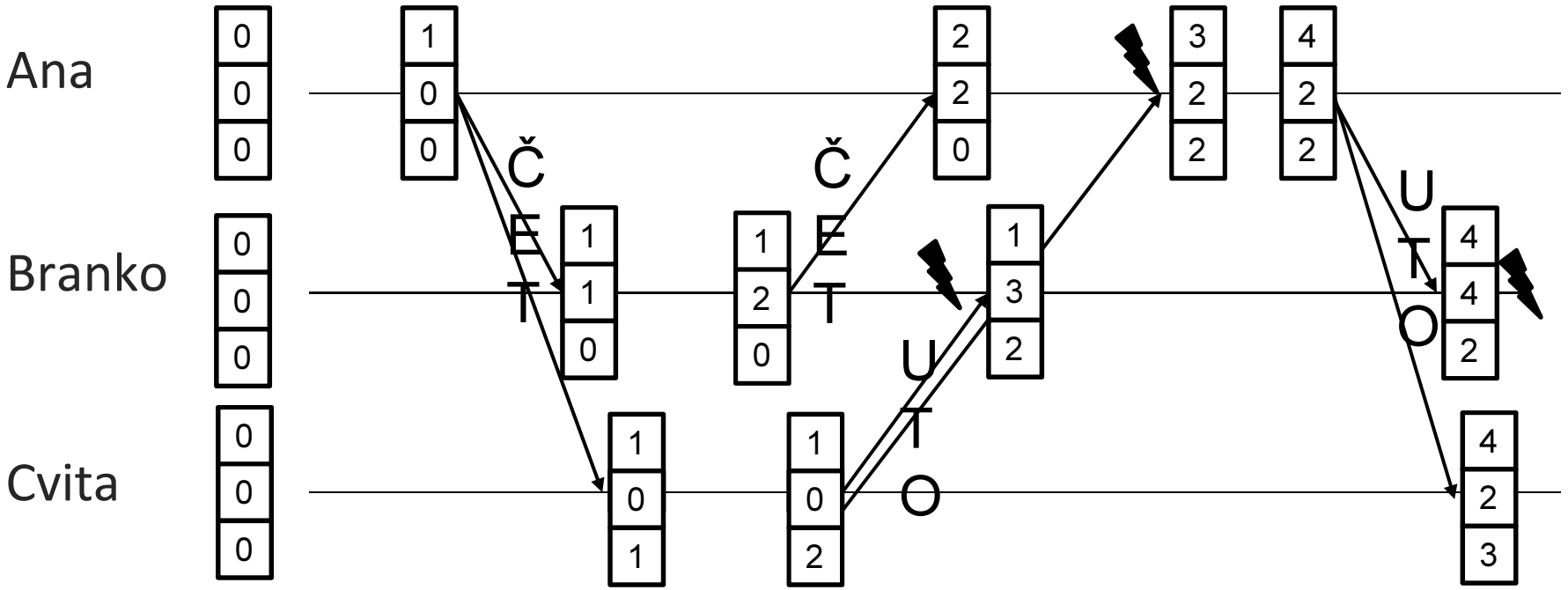
- Parcijalni uređaj:
  - $V=V' \Leftrightarrow V[i] = V'[i], \quad i=1,\ldots,N$
  - $V \leq V' \Leftrightarrow V[i] \leq V'[i], \quad i=1,\ldots,N$
  - $V<V' \Leftrightarrow V \leq V'$ i $V \neq V'$
- $e_1 || e_2$ akko nisu usporedivi odnosno ne vrijedi niti $e_1 \leq e_2$ niti $e_2 \leq e_1$ (parcijalni uređaj)

Usporedimo vektore:

V_1	7	7	7
V_2	6	7	6
V_3	6	7	8

- Pravila:
  - VC1: Inicijalno $V_i[j] = 0$, za i,j = 1,2,..N
  - VC2: U procesu p_i, neposredno prije (bilo kojeg) događaja:
    $V_i[i] = V_i[i] + 1$
  - VC3: Proces p_i prilaže svoj V_i svakoj poruci koju šalje
  - VC4: Kad p_i primi poruku s vektorom W, postavlja:
    $V_i[j] = \max(V_i[j], W[j])$ - *primijetiti da se obavlja i VC2*

Primjer - dogovor za večeru



Podsjetnik:

VC1: Inicijalno $V_i[j] = 0$, za i,j = 1,2,..N

VC2: U procesu p_i, neposredno prije (bilo kojeg) događaja:

$V_i[i] = V_i[i] + 1$

VC3: Proces p_i prilaže svoj V_i svakoj poruci koju šalje

VC4: Kad p_i primi poruku s vektorom W, postavlja:

$V_i[j] = \max(V_i[j], W[j])$ - *primijetiti da se obavlja i VC2*

Razrješavanje konflikata NIJE zadaća vektorskih satova!

Primjer: Riak vektorski sat (1)

- Riak koristi vektorske satove (http://basho.com/why-vector-clocks-are-easy/)
- U svaki zahtjev uključiti:
  - `X-Riak-ClientId` za identifikaciju "actora" tj. procesa
  - `X-Riak-Vclock`
- Ako dođe do konflikta, Riak odlučuje na temelju `allow_mult` svojstva:
  - `True`: zadržava (i vraća) obje vrijednosti (`"siblings"`), svaka vrijednost ima svoj `"vtag"`
  - `False`: zadnji "pobjeđuje"

Primjer: Riak vektorski sat (2)

- Napravimo novi bucket s postavkom allow_mult = true (default je false) čime dopuštamo stvaranje siblinga (*sibling* = brat ili sestra) zapisa.

```
$ curl -v -X PUT
http://192.168.56.12:10018/buckets/dogovor/props
-H "Content-Type: application/json„
 -d '{"props":{"allow_mult":true}}'
```

- Pretpostavimo da se Ljilja, Krešo i Jasna hoće dogovoriti koju ocjenu će dati studentu Igoru iz domaće zadaće.
- Krešo predlaže jedan:

```
$ curl -v -X PUT
http://192.168.56.12:10018/buckets/dogovor/keys/igor -H
"Content-Type: application/json" -H "X-Riak-ClientId:
kreso" -d '{"ocjena":"1"}'
```

Primjer: Riak vektorski sat (3)

- Dohvatimo vektorski sat:

```
$ curl -v
http://192.168.56.12:10018/buckets/dogovor/keys/igor

npr. X-Riak-Vclock:
a85hYGBgzGDKBVIcKlYHQoIWzJiQwZTImMfKsCZA6gxfFgA=
```

- Ljilja i Jasna čitaju, te Ljilja predlaže dovoljan i uključuje Vclock u zahtjev:

```
$ curl -i -X PUT
http://192.168.56.12:10018/buckets/dogovor/keys/igor \
        -H "X-Riak-ClientId: ljilja" \
        -H "X-Riak-Vclock:
a85hYGBgzGDKBVIcKlYHQoIWzJiQwZTImMfKsCZA6gxfFgA=" \
        -H "Content-Type: application/json" \
        -d '{"ocjena" : 2}'
```

Primjer: Riak vektorski sat (4)

- Jasna se slaže s Krešom, predlaže nedovoljan i uključuje Vclock u zahtjev:

```
$ curl -i -X PUT
http://192.168.56.12:10018/buckets/dogovor/keys/igor \
        -H "X-Riak-ClientId: jasna" \
        -H "X-Riak-Vclock:
a85hYGBgzGDKBVIcKlYHQoIWzJiQwZTImMfKsCZA6gxfFgA=" \
        -H "Content-Type: application/json" \
        -d '{"ocjena" : 1}'
```

- Što Riak sad treba uraditi*?
- Obje verzije (siblings, "brat i sestra") možete dohvatiti ako dodate -H "Accept: multipart/mixed":

```
$ curl -v
http://192.168.56.12:10018/buckets/dogovor/keys/igor -H
"Accept: multipart/mixed"
```

**Primjer preuzet (i skraćen) iz (starih) uputa za Riak projekt, pogledati*

Vektorski sat (4)

- Uočavanje a ne razrješavanje konflikata
- Dodatna literatura (posebice za one koji nisu bili na predavanju):
  - Wikipedia:
    - http://en.wikipedia.org/wiki/Lamport_timestamps
    - http://en.wikipedia.org/wiki/Vector_clocks
  - Lecture 2. Unit 4. Logical clocks and vector clocks, ID2203: http://www.youtube.com/watch?v=TH1dA7dPB2c
  - I. Podnar Žarko, K. Pripužić, I. Lovrek, M. Kušek, Raspodijeljeni sustavi, radna inačica udžbenika v.1.0, 2012: http://www.fer.unizg.hr/_download/repository/Rassus-2013_knjiga_v_1_0.pdf

- Za "one koji žele znati više":
  - Primjena Lamportovog sata: http://www.mcs.csueastbay.edu/~cclee/cs6580/logicalclockspart1.ppt
  - Dodatna potrošnja memorije odnosno povećanje veličine poruke:
    - An efficient implementation of vector clocks: http://www.sciencedirect.com/science/article/pii/002001909290028T
    - Riak vector clock prunning: http://docs.basho.com/riak/latest/references/appendices/concepts/Vector-Clocks/#Vector-Clock-Pruning
  - Interval Tree Clocks: generalizacija za nepoznat broj procesa: https://github.com/sinabz/itc4j

Bloomov filter

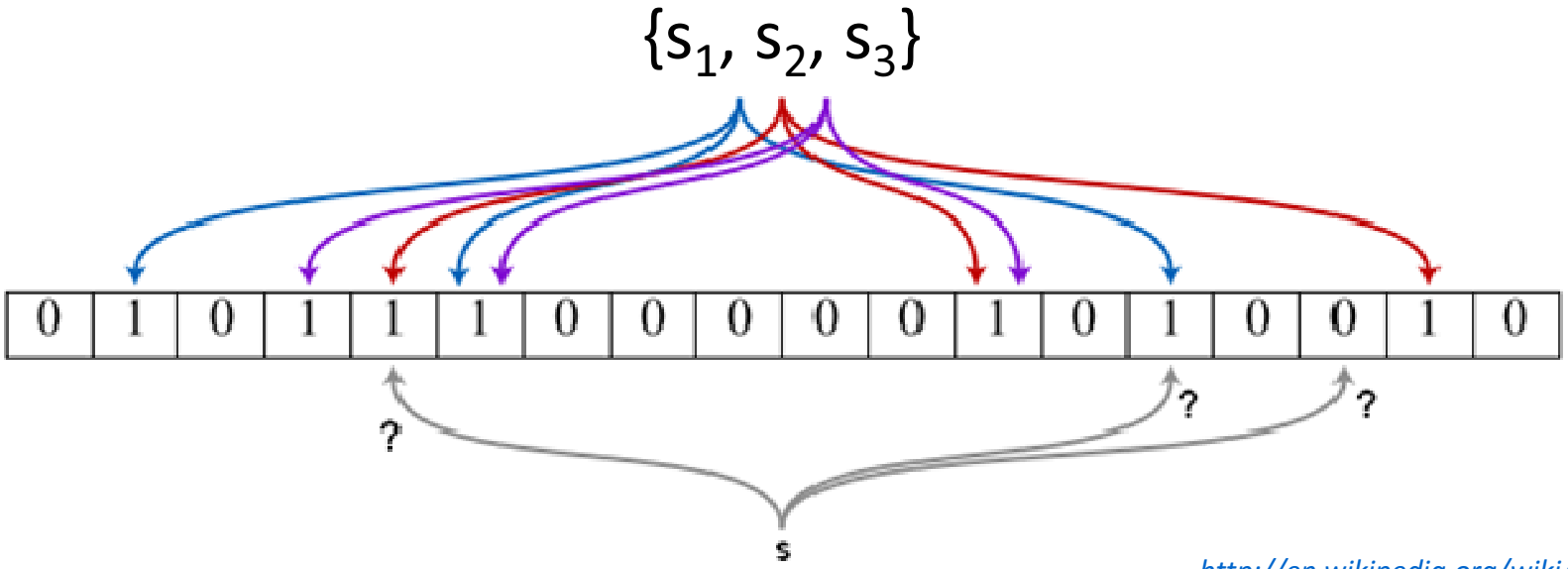
Bloomov filter (1)

- Prostorno učinkovita **probabilistička podatkovna struktura** koja se koristi za ispitivanje članstva elemenata u skupu
- Razvio B. H. Bloom 1970.* za potrebe programa koji provjeravaju pravopis
- Učinkovitost se ostvaruje nauštrb male vjerojatnosti krivog odgovora, naime:
- Bloomov filter može dati **lažni pozitivan odgovor:**
  - moguće je da se za neki element tvrdi da **jest** član skupa iako on to nije
  - ali ne vrijedi obrat: nije moguće da se za neki element tvrdi da nije član skupa a da on to jest.

**Bloom, B. Space/time tradeoffs in hash coding with allowable errors, Communications of the ACM, 13(7):422-426, Srpanj 1970.*

Bloomov filter (2)

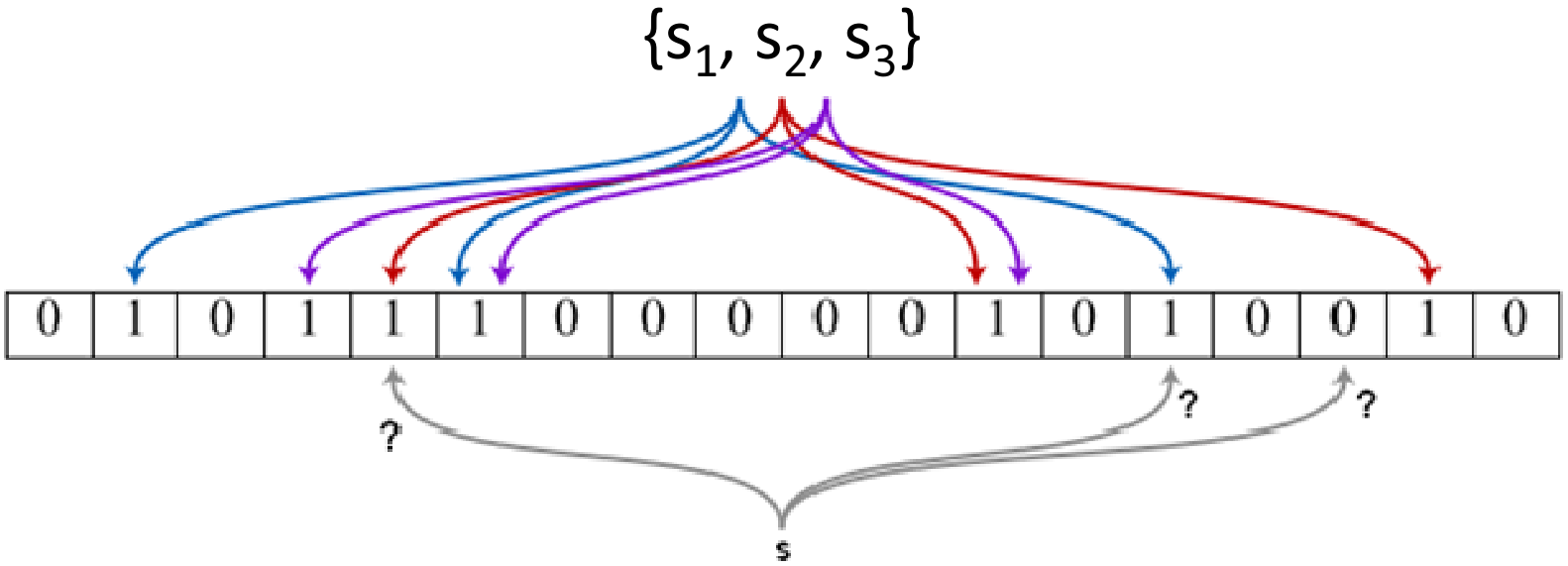
- $S = \{s_1, s_2, \ldots, s_n\}$
- S predstavljamo poljem od **m** bitova (inicijalno 0)
- **k** nezavisnih funkcija raspršenog (uniformnog) adresiranja:
  $h_1, h_2, \ldots h_k$ s rasponom {0, m-1}
- Elementi se mogu dodavati u skup (ali se ne mogu izbacivati)



http://en.wikipedia.org/wiki/Bloom_filter

Bloomov filter (3)

- Dodavanje s u skup:
  - postaviti $h_1(s)$, $h_2(s)$, ... $h_k(s)$ bitove na 1
- Provjera članstva za element s:
  - (vjerojatno) j**est član** akko ∀ $h_i(s)$==1, i=1..k
  - inače:(sigurno) **nije član**



Bloomov filter (4) - vjerojatnost lažnog pozitivnog odgovora

- Jedan element, vjerojatnost da bit **nije** postavljen na 1 (za jednu hash funkciju):

$$1-\frac{1}{m}$$

- …odnosno, za k hash funkcija:

$$\left(1-\frac{1}{m}\right)^{k}$$

- Za n elemenata:

$$\left(1-\frac{1}{m}\right)^{nk}$$

- Obratno, vjerojatnost da bit **je** postavljen na 1:

$$1-\left(1-\frac{1}{m}\right)^{nk}$$

- Konačno, kod testiranja provjeravamo k bitova:

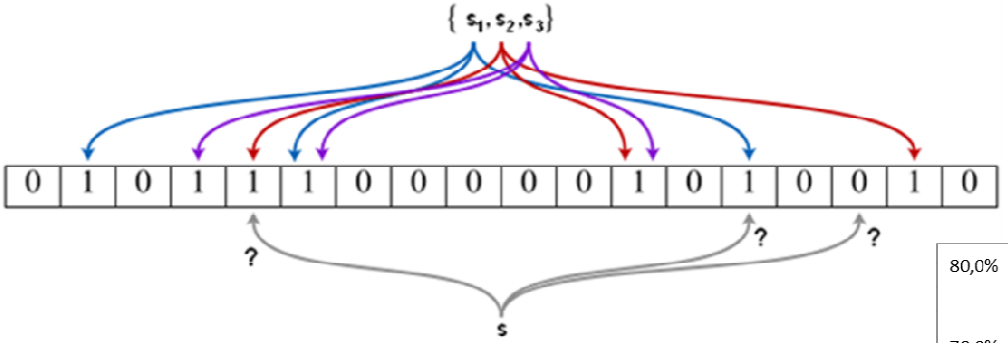
$$\left(1-\left(1-\frac{1}{m}\right)^{nk}\right)^{k} \approx \left(1-e^{-nk/m}\right)^{k}$$

- Za zadani n i m, k_{min} je:

$$k_{min} = \frac{m}{n}\ln 2 \approx 0.7\frac{m}{n}$$

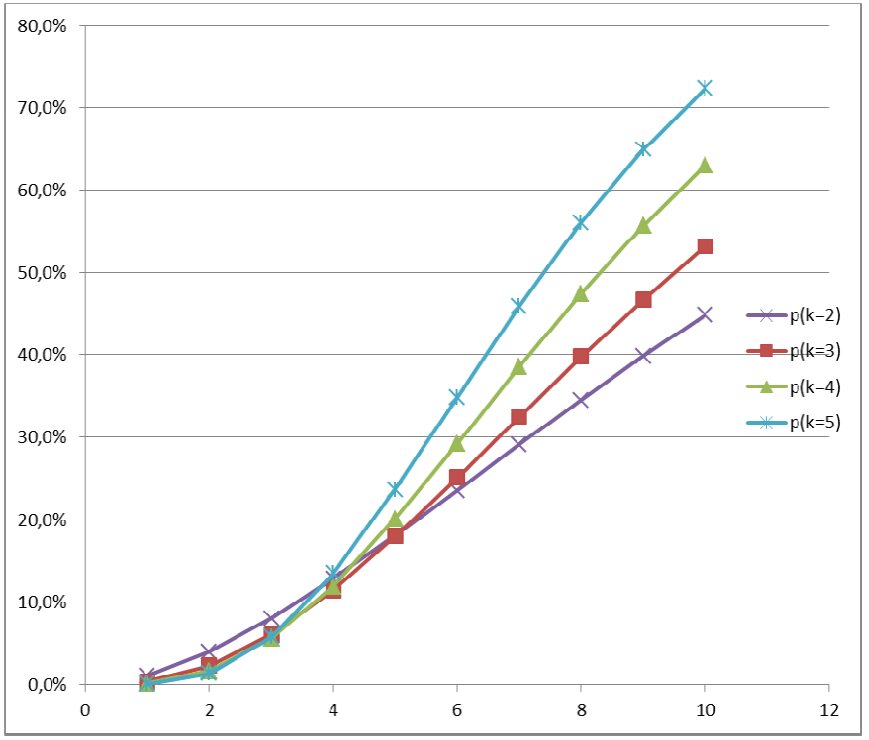
Bloomov filter (5) - vjerojatnost lažnog pozitivnog odgovora

- Za naš primjer (m=18, n=3, k=3) provjeravamo je li s u skupu



$$k_{min} = \frac{m}{n} \ln 2 = \frac{18}{3} \ln 2 \approx 4{,}16$$

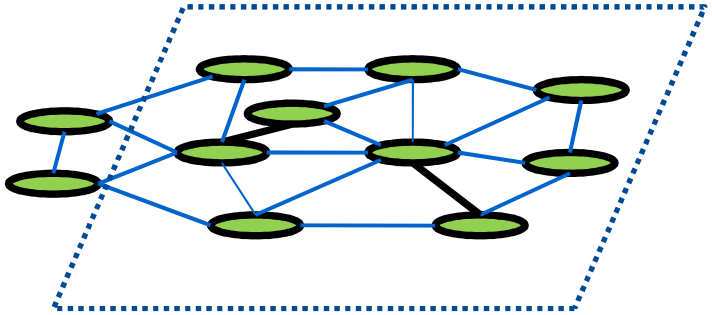
n	p(k=2)	p(k=3)	p(k=4)	p(k=5)
1	1,1%	0,4%	0,2%	0,1%
2	3,9%	2,3%	1,6%	1,4%
3	**8,0%**	**6,0%**	**5,6%**	**5,7%**
4	12,8%	11,4%	11,9%	13,5%
5	18,1%	18,0%	20,1%	23,7%
6	23,6%	25,1%	29,2%	34,9%
7	29,1%	32,5%	38,5%	46,0%
8	34,5%	39,8%	47,5%	56,1%
9	39,8%	46,7%	55,7%	64,9%
10	44,8%	53,2%	63,0%	72,3%



Distribuirane tablice raspršenog adresiranja

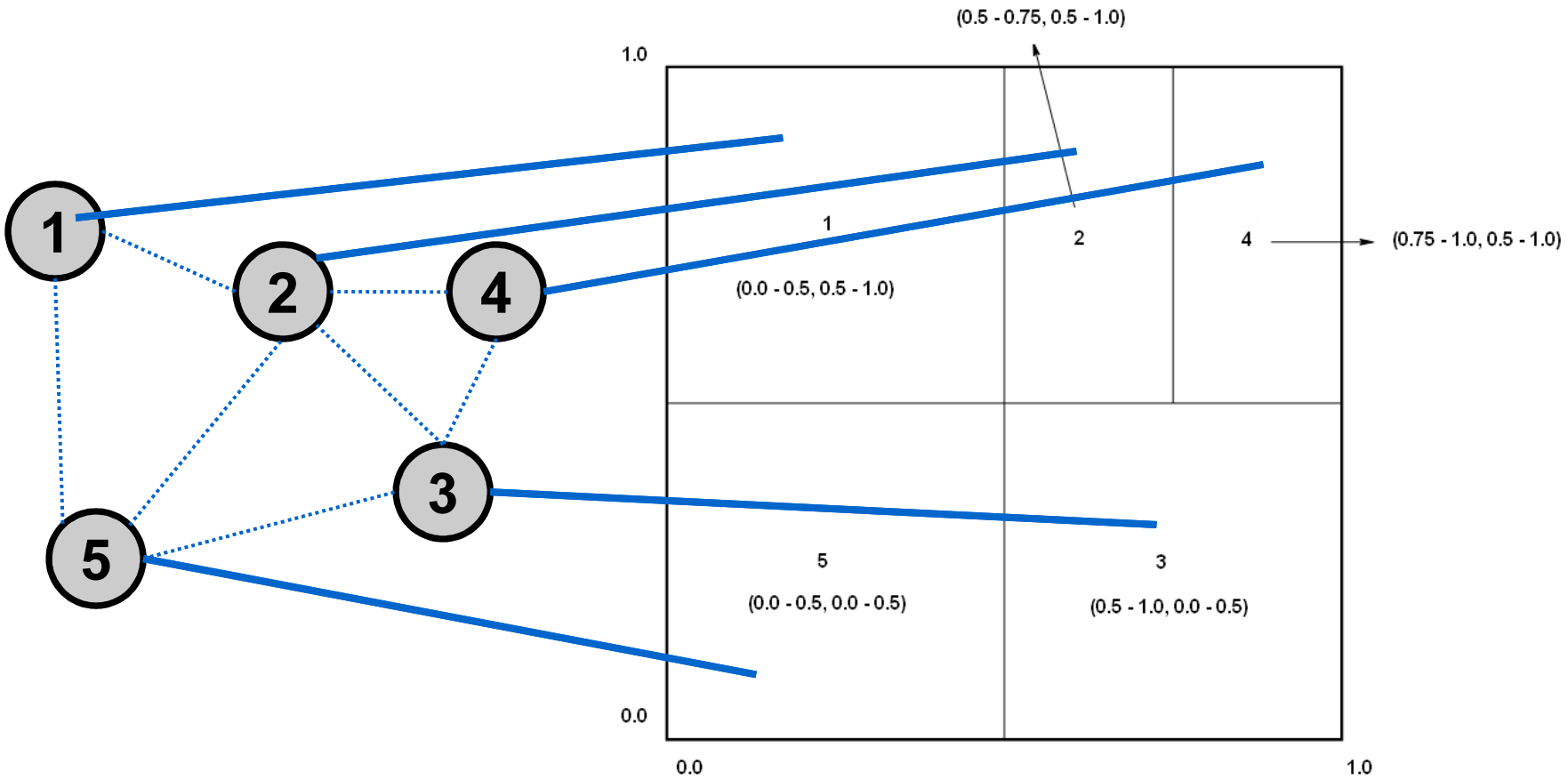
Distribuirane tablice raspršenog adresiranja

- Distribuirane tablice raspršenog adresiranja (*DHT - Distributed Hash Table*) su **decentralizirani distribuirani** sustavi koji omogućuju funkcionalnost tablica raspršenog adresiranja odnosno implementiraju dvije funkcije:
  - `stavi(ključ, vrijednost)`
  - `dohvati(ključ)`



- Svaki čvor koji sudjeluje u sustavu može pozivati te dvije funkcije, odnosno zadužen je za obavljanje tih funkcija.
- Odgovornost za preslikavanje ključa u vrijednost je raspodijeljena među sudionicima u sustavu na takav način da promjene u sastavu sudionika **ne izazivaju velike promjene** u sustavu.
- Pružaju infrastrukturu za razvoj mnogih složenijih aplikacija kao što su raspodijeljeni datotečni sustavi, P2P sustavi za razmjenu podataka, sustavi za distribuciju podataka, sustavi za razmjenu poruka (engl. *instant messaging*), itd.

Distribuirane tablice raspršenog adresiranja



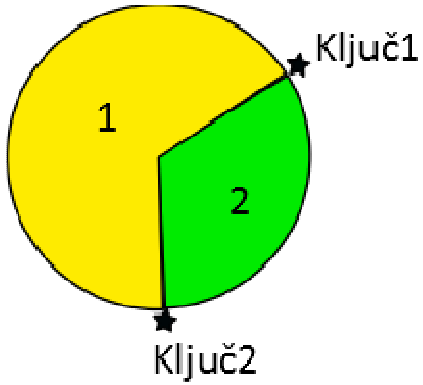
Distribuirane tablice raspršenog adresiranja

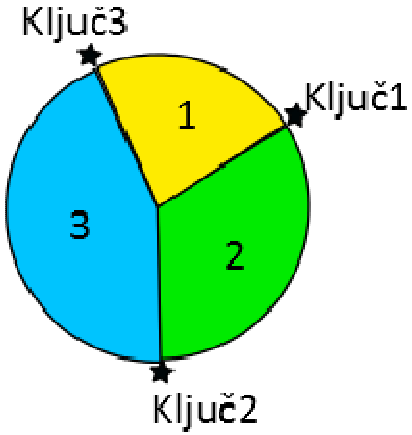
- P2P: Gnutella, Napster, … BitTorrent
- Svojstva:
  - **decentralizacija** – sudionici tvore sustav koji nema centralni autoritet
  - **skalabilnost** – sustav funkcionira učinkovito i s jako velikim brojem sudionika
  - **otpornost na pogreške** – sustav je stabilan i dovoljno pouzdan usprkos kontinuiranom dolasku, odlasku i kvaru sudionika
- Kako bi se mogla ostvariti ova dobra svojstva svaki sudionik održava veze samo s malim brojem sudionika u sustavu, tipično s O(log(n)). Na taj način je ograničen posao koji je potrebno obaviti uslijed promjena u sastavu.
- Svaki sudionik u sustavu biva zadužen za određeni dio prostora ključeva.
- Većina distribuiranih tablica raspršenog adresiranja, prilikom preslikavanja ključeva u čvorove, koristi neku vrstu **konzistentnog raspršenog adresiranja**

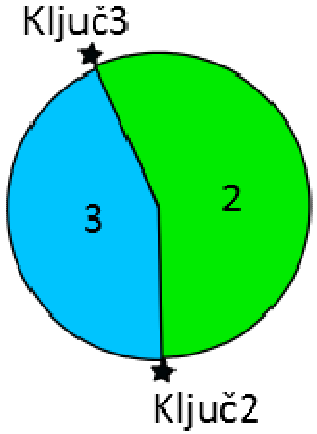
Konzistentno raspršeno adresiranje (*consistent hashing*)

- *David Karger, Eric Lehman, Tom Leighton, Matthew Levine, Daniel Lewin, and Rina Panigrahy.* ***Consistent Hashing and Random Trees: Tools for Relieving Hot Spots on the World Wide Web****. STOC 1997*
- Zadatak: raspršiti objekte na n pretinaca (čvorova, računala)
- uobičajeno: **`hash(o) % n`**
- Što ako se **n** promijeni?
- Ideja konzistentnog raspršenja jest da:
  - ako se **doda novi** spremnik on preuzme samo mali ("pravedan") dio opterećenja
  - ako se **isključi** neki spremnik njegovo se opterećenje razdijeli ostalima

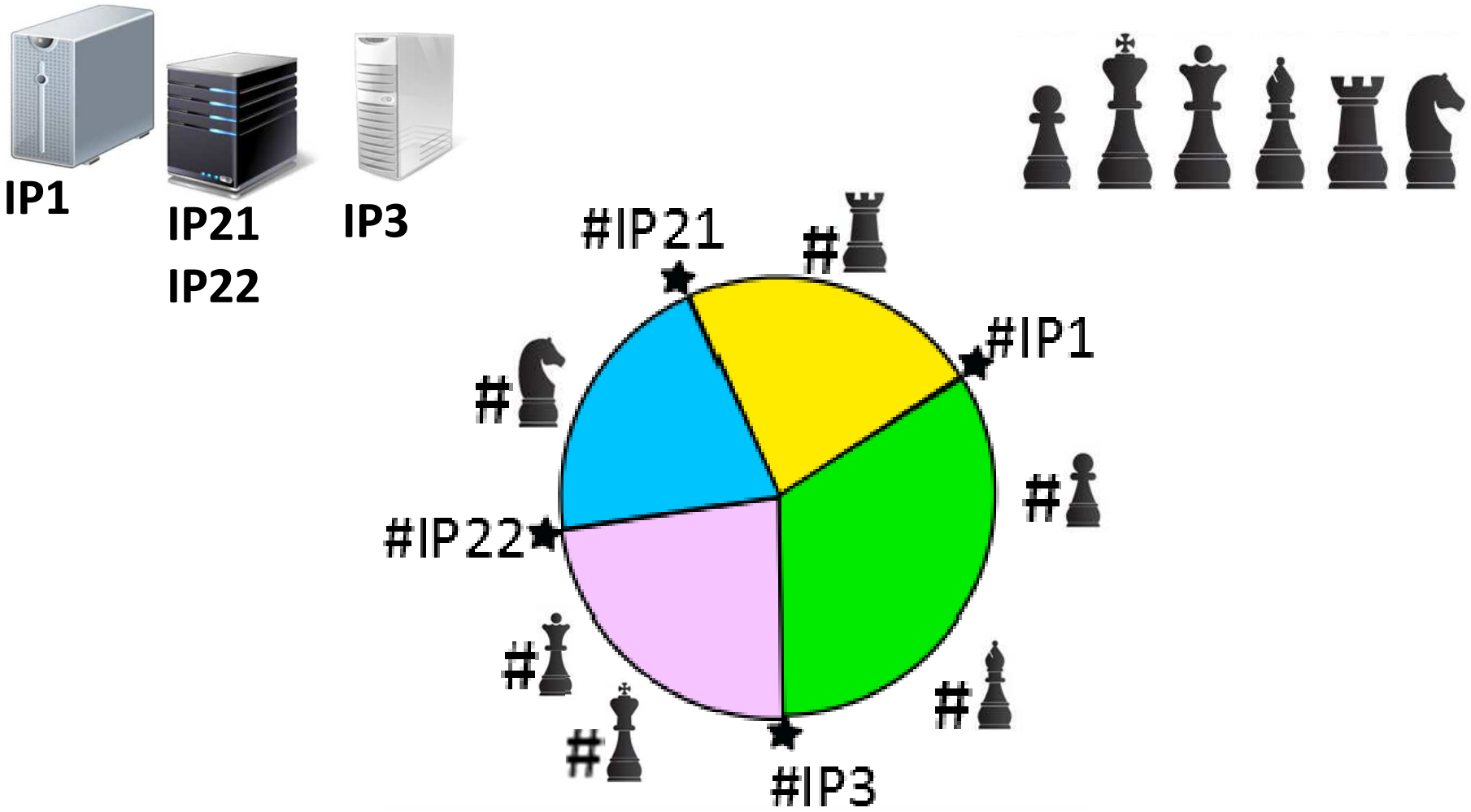
Kako postići što manji broj promjena nakon što se promijeni broj pretinaca?







Jedno računalo može biti više virtualnih servera



http://www.tomkleinpeter.com/2008/03/17/programmers-toolbox-part-3-consistent-hashing/
http://en.wikipedia.org/wiki/Consistent_hashing
http://blogs.msdn.com/b/csliu/archive/2009/09/17/consistent-hashing-theory-implementation.aspx

Primjeri

U nastavku se ponavljaju brojni slajdovi iz prethodna dva predavanja koji su korišteni kao primjeri implementacije nekog od koncepata.

Ovdje su sad ponovljeni radi kompletnosti, grupirano po bazi podataka.


riak

Riak

- Inspiriran s *Amazon Dynamo*
- Distribuirana **P2P** KV baza podataka
- V - bilo što (*plain text*, JSON, slika, video, …)
- Jednostavno HTTP sučelje (*„Riak speaks web”*)
- Fault tolerant
- Skalabilnost
  - Lako je dodati novi čvor u klaster
  - Automatska distribucija podataka
  - *„a near-linear performance increase as you add capacity”*
- Nedostatci:
  - Ad-hoc upiti
  - Povezivanje zapisa (ref.int.)

Instalacija i pokretanje

- Install Erlang, install (from source) Riak
- Pokrećem tri servera:

```
$ dev/dev1/bin/riak start
$ dev/dev2/bin/riak start
$ dev/dev3/bin/riak start
```

- Povezujem ih u prsten (klaster):

```
$ dev/dev2/bin/riak-admin join -f dev1@127.0.0.1
$ dev/dev3/bin/riak-admin join -f dev2@127.0.0.1

$ dev1/bin/riak-admin cluster plan
$ dev2/bin/riak-admin cluster commit
```

Key Value repozitorij

- REST:
  - `/riak/bucket/`**`key`** `# Old format`
  - `/buckets/bucket/keys/`**`key`** `# New format`
- Put (pohranjivanje)

```
$ curl -v -X PUT http://localhost:8091/riak/test/helloworld \
-H "Content-Type: text/html" \
-d "<html><body>hello world</body></html>"
```

- Get (dohvat):

```
$ curl http://localhost:8091/riak/test/helloworld
<html><body>hello world</body></html>
$ curl http://localhost:8092/riak/test/helloworld
<html><body>hello world</body></html>
$ curl http://localhost:8093/riak/test/helloworld
<html><body>hello world</body></html>
```

Links

- Jednostrane poveznice (linkovi) na druge zapise
- `-H "Link: </riak/slike/igor>; riaktag=\"slika\""`
- "Link walking" - moguće je slijediti niz poveznica
- Npr., stavimo sliku:

```
$ curl -X PUT http://localhost:8091/riak/slike/igor.png \
-H "Content-type: image/png" \
--data-binary @homer.png
```

Links (2)

- Povežimo nastavnika i sliku:

```
$ curl -v -X PUT http://localhost:8091/riak/nastavnici/igor?returnbody=true \
-H "Content-Type: application/json" \
-H "Link: </riak/slike/igor>; riaktag=\"slika\"" \
-d '{"ime" : "Igor Mekterović", "zavod" : "ZPR"}'
```

- Dodajemo još i studenta:

```
$ curl -v -X PUT http://localhost:8091/riak/studenti/0036342145?returnbody=true \
-H "Content-Type: application/json" \
-H "Link: </riak/nastavnici/igor>; riaktag=\"mentor\"" \
-d '{"ime" : "Pero Perić", "modul" : "PIIS"}'
```

- *Link-walking*: na url se dodaje: `/_,_,_`
  tj. `/bucket/tag/keep(=0/1)`, npr.

```
$ curl http://localhost:8091/riak/studenti/0036342145/_,_,_
```

  - Vraća sve povezane objekte, odnosno ovdje samo mentora

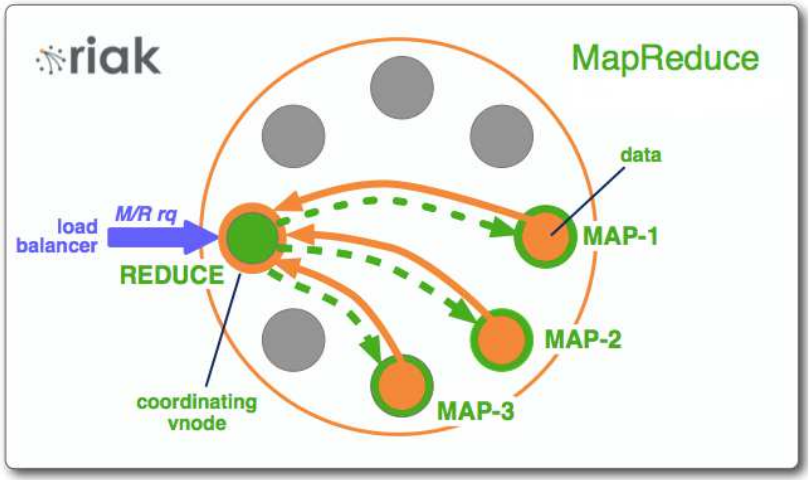
-

```
$ curl http://localhost:8091/riak/studenti/0036342145/_,_,1/_,slika,_
```

  - Vraća mentora i mentorovu sliku (1 zadaje da se vraća i sve objekte „po putu”)

Riak MapReduce

- *„In practice, your MapReduce job code is likely less than 10 kilobytes, it is more efficient to send the code to the gigs of data being processed, than to stream gigabytes of data to your 10k of code.”* riakdocs
- Upit + MR koraci
- Upit definira izvore podataka za prvotni map korak
- Dvije vrste koraka (phases):
  - **Map** – šalju se zahtjevi svim čvorovima zaduženima za te podatake da obave map funkciju na njima.



  - **Reduce** je CR. Prima listu podataka kao ulaz i vraća listu podataka kao izlaz. Može biti pozvana više puta!

Zadavanje M/R upita

HTTP POST na /mapred

```
$ curl -X POST -H "content-type:application/json" \
http://localhost:8091/mapred --data @-
{
        "inputs":[...],
        "query":[
                {"link" : ...},
                {"map":{
                        ...
                        }
                },
                {"reduce":{
                        ...
                        }
                }
        ],
        "timeout": 90000
}
(Ctrl+D)
```

Može biti popis:
- bucket/key vrijednosti, ali
- može se zadati i samo bucket pa će biti uzete sve vrijednosti iz njega

Postoji i koncept „filtera”.

Zadaju se funkcije (js ili erlang) ili direktno ili u vidu „pohranjene procedure”.

keep=false (default)

- **M i R faze se mogu izostavljati i/ili više puta kombinirati**!
  - Npr. Map+Reduce+Reduce

Primjer – samo map

```
$ curl -XPUT http://localhost:8098/buckets/training/keys/1 \
    -H 'Content-Type: text/plain' -d 'pizza data goes here'
$ curl -XPUT http://localhost:8098/buckets/training/keys/2 \
    -H 'Content-Type: text/plain' -d 'pizza pizza pizza pizza'
$ curl -XPUT http://localhost:8098/buckets/training/keys/3 \
    -H 'Content-Type: text/plain' -d 'nothing to see here'
$ curl -XPUT http://localhost:8098/buckets/training/keys/4
    -H 'Content-Type: text/plain' -d 'pizza pizza pizza'
```

```
$ curl -XPOST http://localhost:8098/mapred \
    -H 'Content-Type: application/json' \
    -d '{   "inputs":"training",
            "query":[
                {"map":{"language":"javascript",
                        "source": "
               function(riakObject) {
                  var val = riakObject.values[0].data.match(/pizza/g);
                  return [[riakObject.key, (val ? val.length : 0 )]];
               }
             "}}]
        }'
```

```
["1",1],["2",4],["3",0],["4",3]]
```

Primjer Map+Reduce (1)

- Klasični primjer prebrojavanja riječi
- Savjet: isprobajte ih prvo u npr. js konzoli
- Pohranimo prvo map u riak (nalik pohranjenoj proceduri u SQL-u):

```
// map.js
function(v) {
   var words = v.values[0].data.toLowerCase().match(/\w*/g);
   var counts = [];
   for(var i in words)
      if (words[i] != '') {
         var count = {};
         count[words[i]] = 1;
         counts.push(count);
      }
    return counts;
}
```

```
curl -v -X PUT http://localhost:8091/riak/nmbp/nmbp_map.js \
   -H "Content-Type: text/javascript" \
   --data-binary @map.js
```

Primjer Map+Reduce (2)

- Zatim reduce:

```
// reduce.js
function(values) {
  var result = {};
  for (var value in values) {
    for(var word in values[value]) {
      if (word in result)
        result[word] += values[value][word];
      else
        result[word] = values[value][word];
    }
  }
  return [result];
}
```

```
curl -v -X PUT http://localhost:8091/riak/nmbp/nmbp_reduce.js \
   -H "Content-Type: text/javascript" \
   --data-binary @reduce.js
```

Primjer Map+Reduce (3)

- Konačno, pokrenemo:

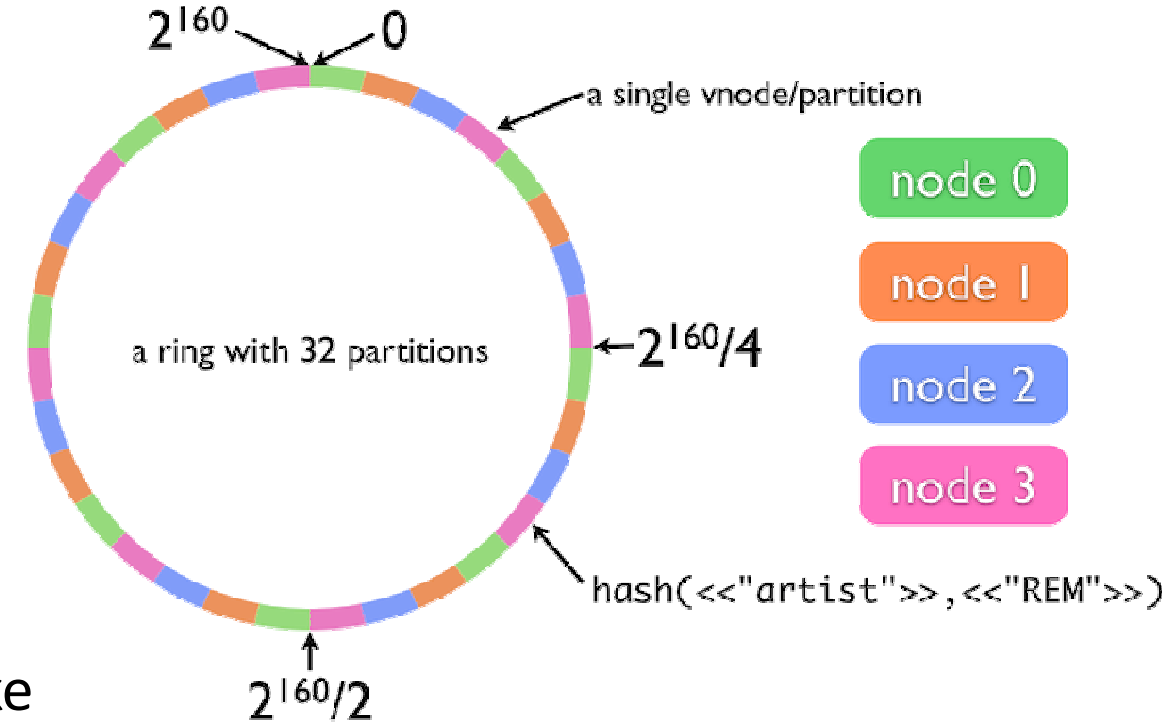
```
$ curl -X POST -H "content-type:application/json" \
http://localhost:8091/mapred --data @-
{
    "inputs":[...],
        "query":[
            {"map":{
                    "language":"javascript",
                    "bucket":"nmbp",
                    "key":"nmbp_map.js"
                }
                },
            {"reduce":{
                    "language":"javascript",
                    "bucket":"nmbp",
                    "key":"nmbp_reduce.js"
                }
            }
    ]
}
```

Napomena

- Objavljeni su zadatci za vježbu koji vam omogućavaju da isprobate i pobliže se upoznate s Riakom
- Zadatci nisu obavezni („za one koji žele znati više”), tj. neće se ispitivati Riak na razini implementacijskih detalja

Riak prsten

- 160-bitni integer
- Q particija (def=64)
- Na slici je Q=32, te postoji 32 vnodea
- Svaki vnode je zadužen za raspon ključeva
- N = 4 fizička stroja
- Svaki zadužen za otprilike Q/N vnodeova




```
$ curl -H "Accept: text/plain" http://localhost:8091/stats
…
"ring_ownership":
[{'dev3@127.0.0.1',21},{'dev2@127.0.0.1',21},{'dev1@127.0.0.1',22}]"
```

Consistency & Durability

- n_val - broj replika vrijednosti (default = 3)
- r - broj čitanja nakon kojeg se čitanje smatra uspješnim
- w - broj pisanja nakon kojeg se pisanje smatra uspješnim
- Npr. postavljanje parametara bucketa:

```
curl -X PUT http://localhost:8091/riak/animals \
       -H "Content-Type: application/json" \
       -d '{"props":{"w":2}}'
```

- Mogu se postaviti kod svakog zahtjeva!
  Npr.

```
curl http://localhost:8091/riak/animals/floki
       -H "Content-Type: application/json" \
       -d '{"props":{"r":3}}'
```

- Ili:

```
curl http://localhost:8091/riak/animals/floki?r=3
```

Consistency & Durability , kratice

- U Riaku su uveli znakovne kratice za uobičajene vrijednosti:

Kratica	Definicija
`One`	`1`
`All`	`n_val`
`Quorum`	`n_val/2+1`
`Default`	Ono što je postavljeno za taj *bucket*

- Npr.

```
curl http://localhost:8091/riak/animals/floki?r=quorum
```

Durability

- Kod pisanja:
  i. Objekt se prvo piše u memoriju (*buffer*)
  ii. Potvrđuje se uspješno pisanje
  iii. Objekt se piše na disk
- Potencijalni gubitak podataka između (ii) i (iii).
- Ipak, može se eksplicitno zadati da se (iii) obavi prije (ii) na proizvoljnom broju čvorova, npr. na jednom (dw = *durable write*):

```
$ curl -X PUT http://localhost:8091/riak/animals \
       -H "Content-Type: application/json" \
       -d '{"props":{"dw":"one"}}'
```

Razrješenje konflikata

- Riak koristi vektorske satove (http://basho.com/why-vector-clocks-are-easy/)
- U svaki zahtjev uključiti:
  - `X-Riak-ClientId` za identifikaciju "actora" tj. procesa
  - `X-Riak-Vclock`
- Ako dođe do konflikta, Riak odlučuje na temelju `allow_mult` svojstva:
  - `True`: zadržava (i vraća) obje vrijednosti (`"siblings"`), svaka vrijednost ima svoj `"vtag"`
  - `False`: zadnji "pobjeđuje"

Napredne mogućnosti

- Hooks ~ okidači
  - Pre-commit hook (js ili erlang)
  - Post-commit hook (erlang)
- Pretraživanje (po defaultu isključeno, koristi *hooks* za održavanja GIN-a)
- Indeksiranje (sekundarno)
  - (po defaultu isključeno) uključiti novi storage (LevelDB)
  - Dodatni indeksi se zadaju u zaglavlju (nisu dio vrijednosti)
- Za one koji žele znati više pogledati:
  http://docs.basho.com/riak/latest/tutorials/querying/

Riak zaključno (1)

- Za:
  - Visoka dostupnost
  - Nema single point of failure
  - Koristi HTTP, ima i mnogobrojne *client library (PHP, Java, C#, Python, Ruby, ...)*
  - (moguće koristiti i Protobuf)
  - Nema shemu
- Protiv:
  - Ad hoc upiti, mogućnosti pretraživanja
  - Nema shemu, nema ACID, …

Riak zaključno (2)

- Koristiti za:
  - Session info
  - User profiles, preferences
  - Shopping cart podaci
- Ne korisiti kad:
  - Odnosi među podacima (relationships)
  - Potrebne transakcije koje uključuju više agregata
  - Upiti na temelju sadržaja
  - Skupovske operacije


mongoDB

mongoDB

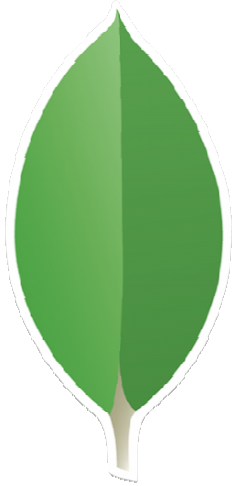
- Document baza podataka (JSON dokumenti pohranjeni kao BSON)
- Master - Slave
- Svojstva:
  - Ad-hoc upiti
  - Indeksiranje
  - MS replikacija
  - Sharding + load balancing
  - File storage
  - Aggregation
  - ServerSide Javascript
  - Language binding
  - FLOSS

http://db-engines.com/

The most popular database management systems

December 2015		Score
1.	Oracle	1498
2.	MySQL	1299
3.	Microsoft SQL Server	1123
4.	MongoDB	301
5.	PostgreSQL	280

» more



Dohvat podataka

1. Dohvat se obavlja pomoću **find funkcije** koja je oblika:

   db.collection.find(query, projection)

   - Upit se zadaje pomoću js/json objekata pri čemu su na raspolaganju razni mongo operatori (logički, usporedbe, …), npr.

```
db.inventory.find(
   {
    $or: [ { qty: { $gt: 100 } }, { price: { $lt: 9.95 } } ]
   }
)
```

   - Više u **Uputama za projekt** i na: https://docs.mongodb.org/manual/tutorial/query-documents/

2. **Aggregation pipeline** – agregacija pomoću koncept cjevovoda
3. **Map/Reduce** – agregacija pomoću M/R koncepta, funkcije se pišu u JS-u

Indeksi (1)

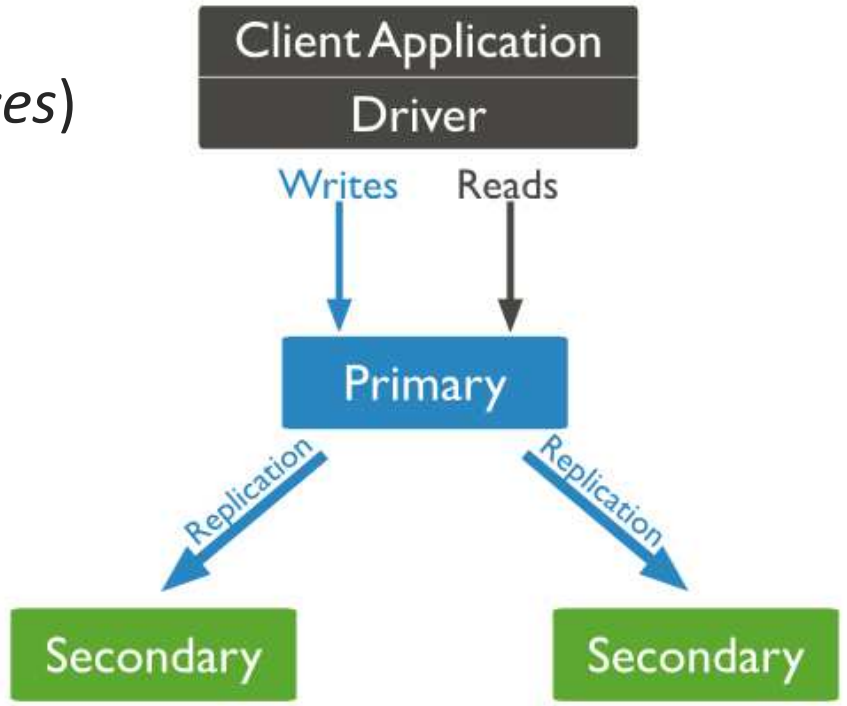
- *Index vs collection scan*
- Definiraju se na razini kolekcije, može se indeksirati bilo koji (pod)atribut dokumenta
- Vrste indeksa:
  - Default _id
  - Jednostavni (jedan atribut)
  - Kompozitni (više atributa)
  - Multikey indeks (indeksiraju se elementi polja!) – ako je atribut polje Mongo to radi automatski
  - Geoprostorni indeksi (2d indexes, 2sphere indexes)
  - Tekst indeksi (ukljanjaju stop riječi, svode na korijen)
  - Hash indeksi (*„to support hash based sharding”*) – ravnomjernija distribucija podataka

Indeksi (2)

- Svojstva:
  - Unique
  - Partial – samo dokumenti u kolekciji koji zadovoljavaju zadane uvjete
  - Sparse – samo dokumenti koji imaju atribut, može se kombinirati s unique
  - TTL indeks – mogu se koristiti za uklanjanje „starih” dokumenata iz baze

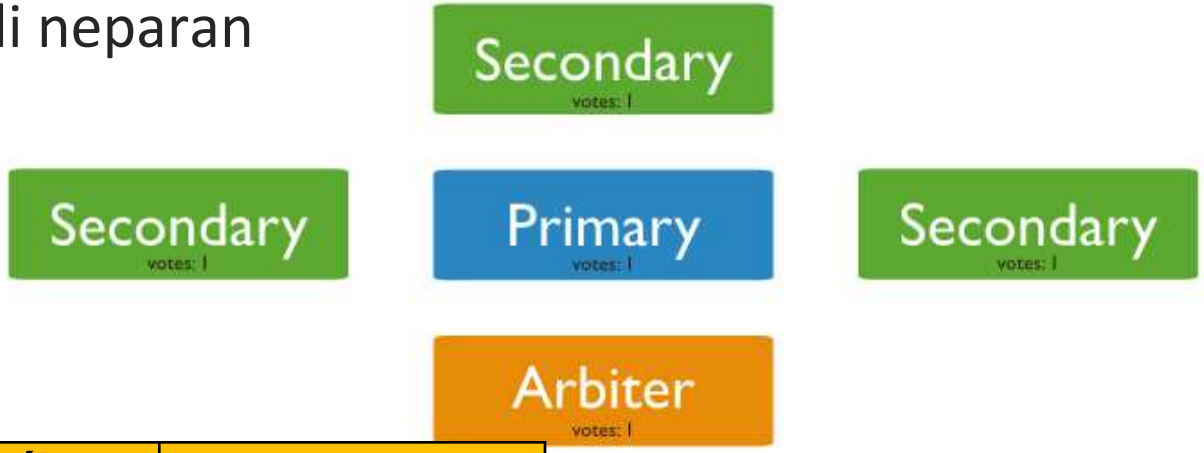
Replikacija

- Standardno (u produkciji) RS (*replica sets*) od **tri** člana. RS-ovi omogućuju redundanciju i *fault tolerance*.
- Ako dođe do kvara mastera, ostali čvorovi organiziraju izbore i izabiru novog mastera (*failover proces*)
- Do 50 članova, ali samo 7 s pravom glasa



Izbori, arbiter

- Moguće je dodati i „arbiter” čvorove koji nemaju podatke i čija jedina uloga je omogućit kvorum u RS
- Dodaju se kad imamo paran broj čvorova, kako bi dobili neparan
- Npr.



Br. Čvorova	Potrebna većina za izbore	Fault tolerance
3	2	1
4	3	1
5	3	2
6	4	2

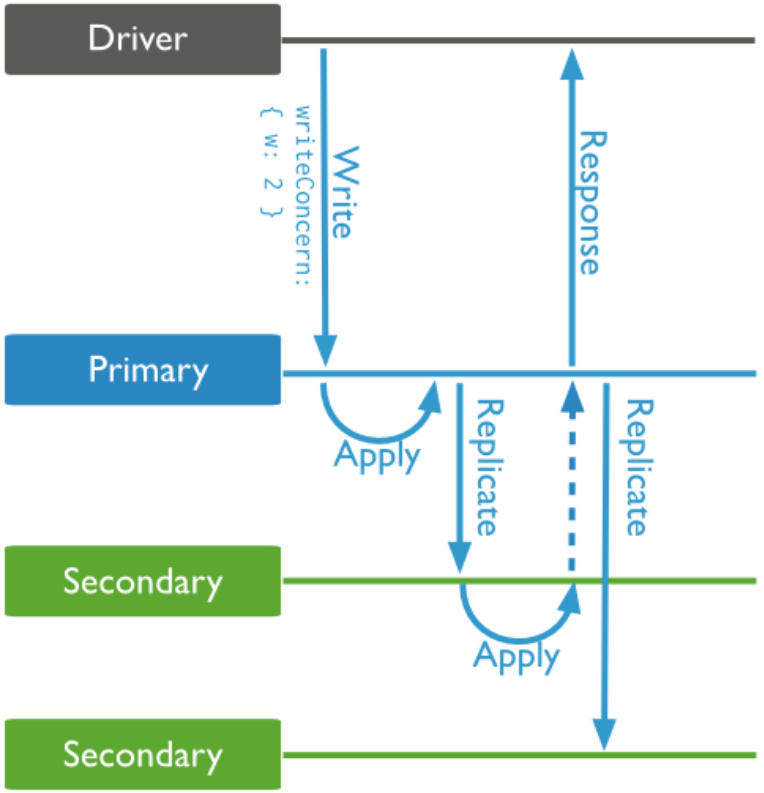
Mongo Write/Read concern

- Opisuje razinu potvrde servera kod obavljanja zadane operacije
- { w: <value>, j: <boolean>, wtimeout: <number> }
  - w – broj instanci (0, 1, „majority”, <tag set>)
  - j: zapisano u dnevnik
  - wtimeout: vremenski limit
- Npr.:

```
db.products.insert(
   { item: „čokolada", qty : 10 },
   { writeConcern:
        { w: 2, wtimeout: 5000 } }
)
```

- readConcern:

```
   { level: <majority|local> }
```

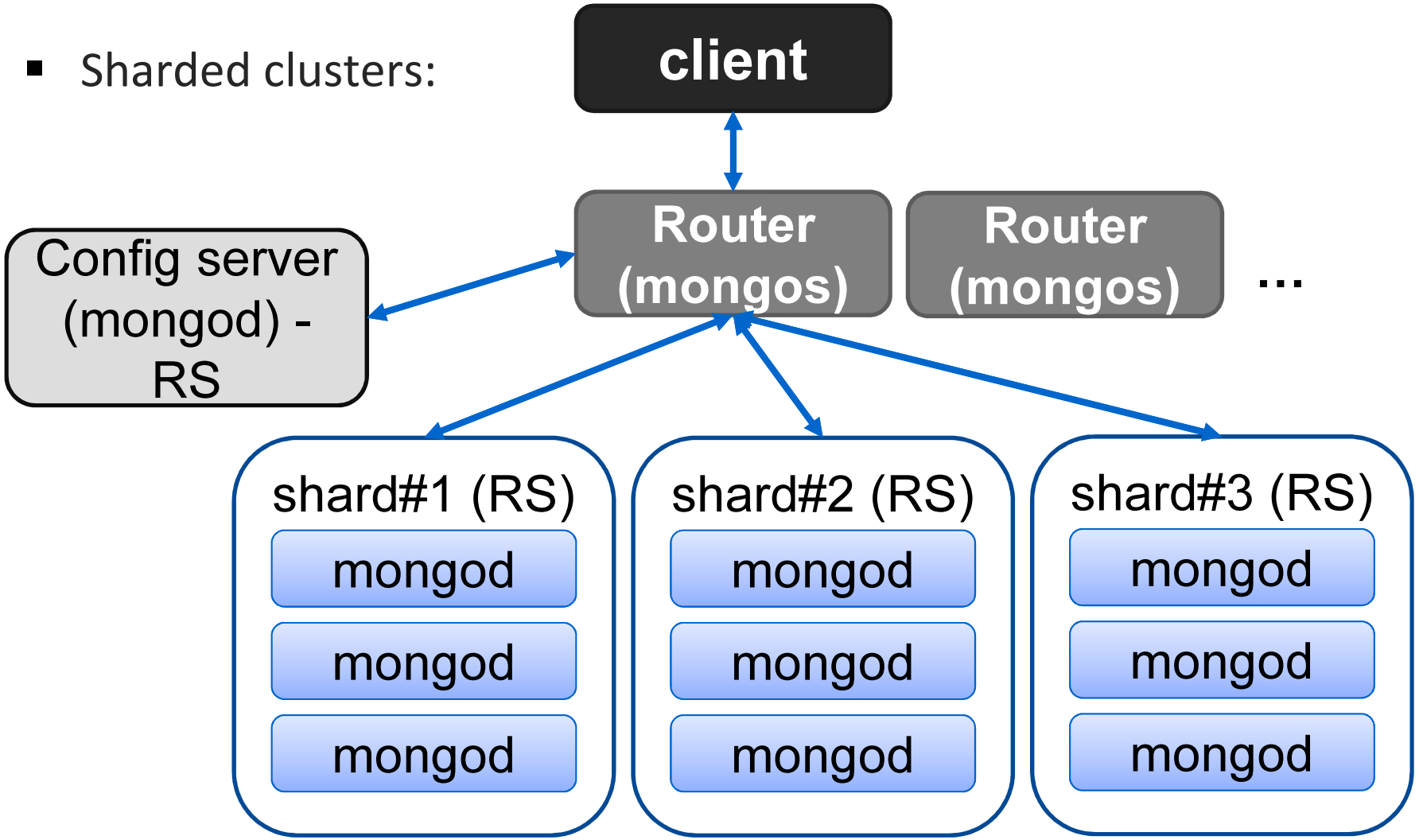


Atomarnost i „transakcije”

- Operacije su atomarne **na razini dokumenta** (može imati ugniježđene dokumente)
- Operacija koje mijenja više dokumenata nije, kao transakcija, atomarna
- $isolated operator – simulira transakciju tako što postavlja exclusive lock na cijelu kolekciju
  - Ne radi na sharded clusteru
  - Ne omogućuje *all-or-nothing atomicity*, npr. kod greške nema rollbacka
- Two-Phase Commits:
  - primjer na: https://docs.mongodb.org/manual/tutorial/perform-two-phase-commits/
  - „The example transaction above is intentionally simple …
    … Production implementations would likely be more complex. ”

Sharding (1)

- Sharded clusters:

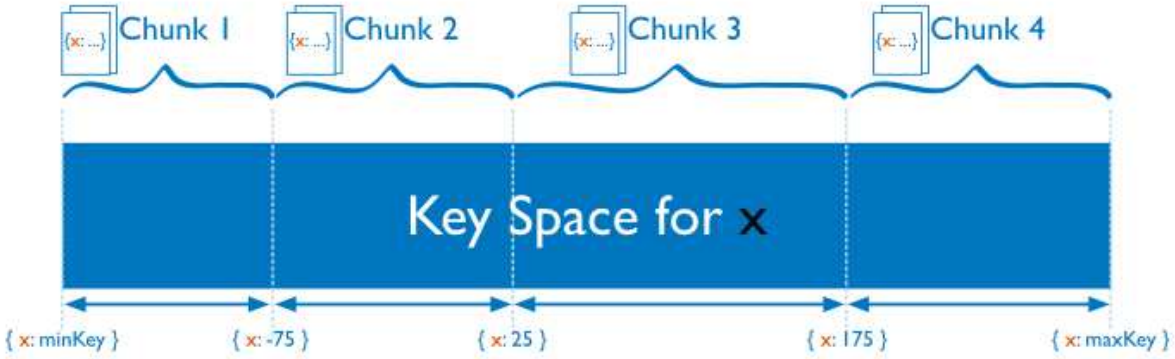


Sharding (2)

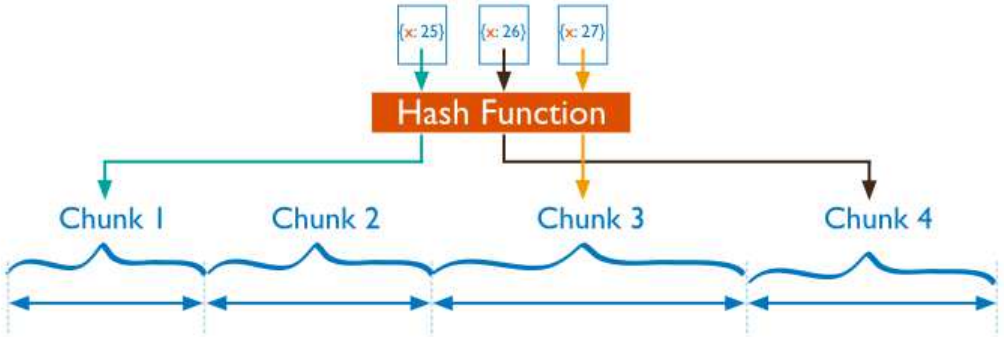
- *Shard key* – indeksirani atribut ili skup atributa koji postoji u svakom dokumentu
- Ključevi (*shard key*) se raspoređuju u grumene (*chunk*) koji se ravnomjerno (sljedeći slajd) raspoređuju po čvorovima

Sharding (3)

- *Range based sharding*
  - range based queries brži, ali mogu narušiti distribuciju čime se ugrožava skaliranje



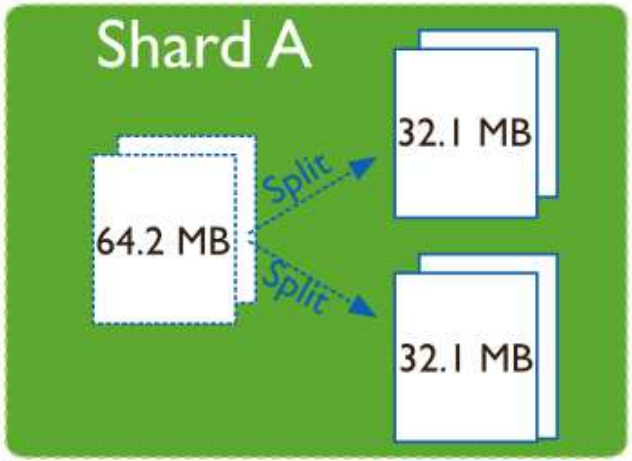
- *Hash based sharding*



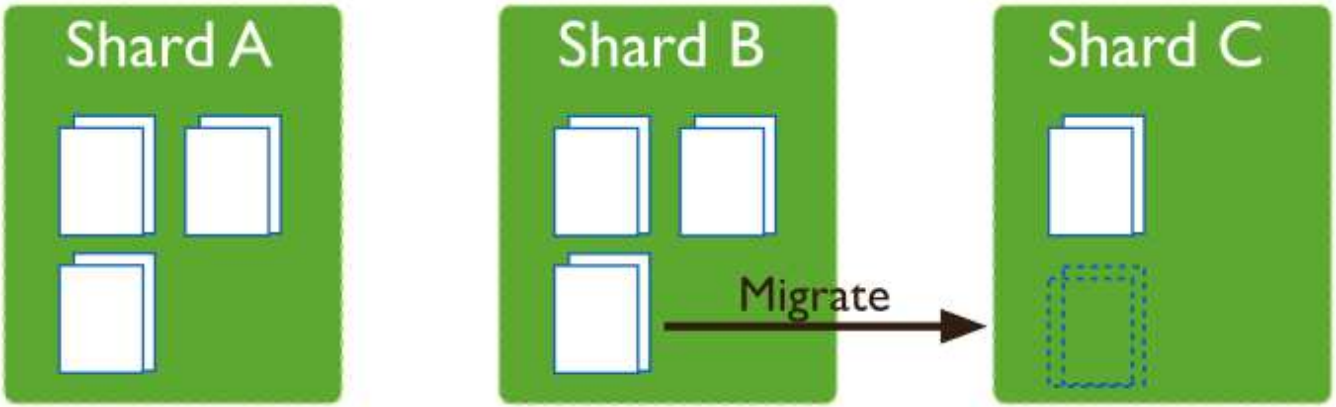
- *Tag aware sharding*

Sharding (4) – balansiranje opterećenja

- Splitting



- Balancing



Sharding (5)

- Npr. zanimljiva prezentacija:

  MongoDB for Time Series Data Part 3: Sharding

  https://www.mongodb.com/presentations/mongodb-time-series-data-part-3-sharding

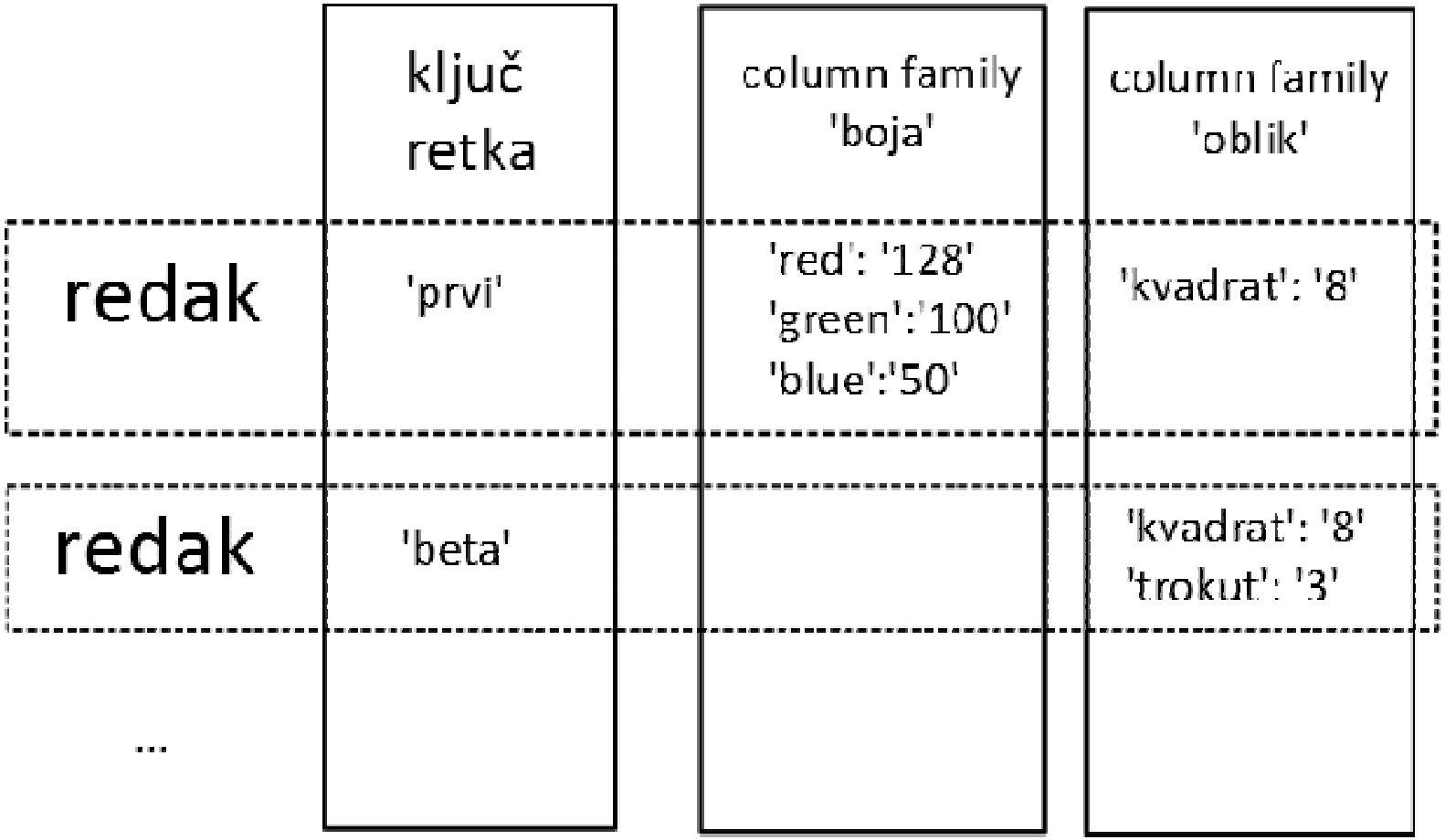

APACHE
HBASE

HBase

- Stupčasta baza podataka, inspirirana s Google Big Table*
- *“Sparse, distributed, persistent multidimensional sorted map.”*
- Koristi Hadoop
- Samo ako se podaci broje u GB+:
  "This project's goal is the hosting of very large tables -- billions of rows X millions of columns -- atop clusters of commodity hardware."
- Dobra svojstva:
  - Skalabilnost
  - Održavanje verzija
  - Kompresija
  - Memorijski rezidentne tablice
  - Fault tolerant

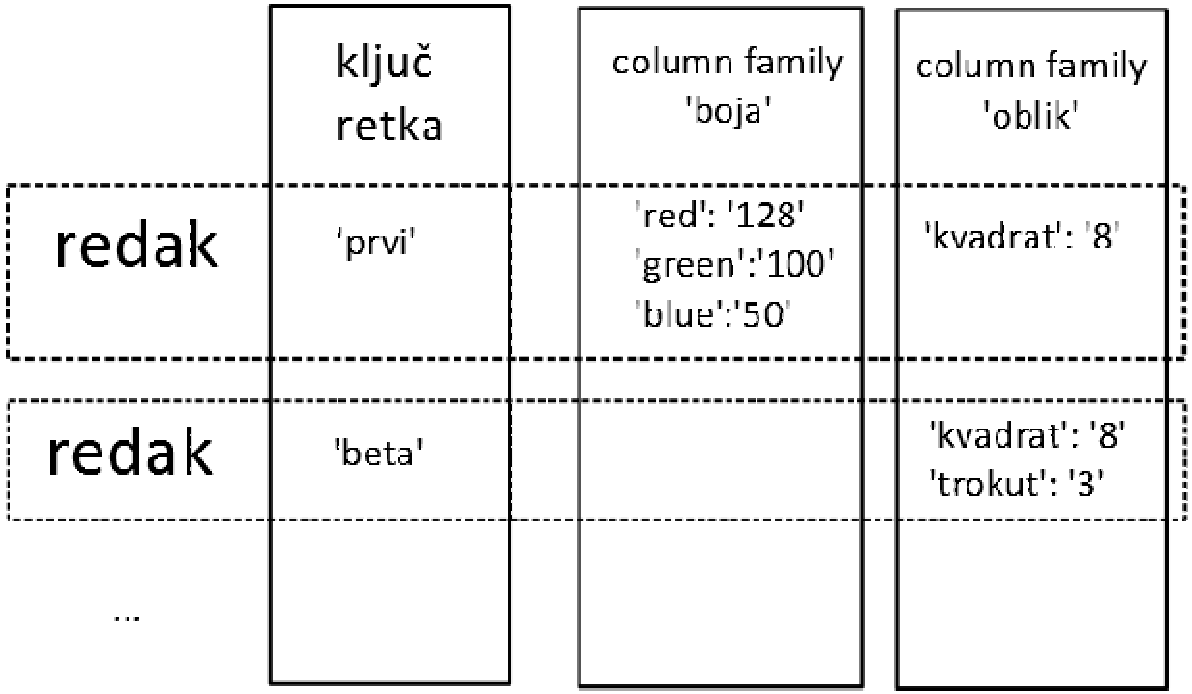
**Chang et al. [2006], Bigtable: A Distributed Storage System for Structured Data*

Struktura (podsjetnik)



Struktura

- "table"* - ključ vrijednost - mapa mapa (*map of maps*)
- "row" (sortirano)
- "column family"
- "column"
- "value", npr.
  prvi/boja:green
  je '100'

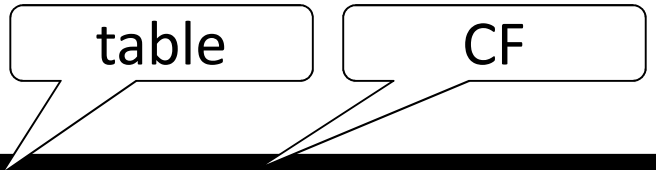


Terminologija podsjeća na relacijske baze podataka, ali su pojmovi suštinski vrlo različiti. Kako bi se to naglasilo, nazivi su ovdje pisani u navodnicima.

Column families

- Sve Hbase operacije su atomarne na razini retka - konzistentni retci
- Zašto ne staviti sve atribute u istu obitelj?
  - Zasebna konfiguracija svake CF

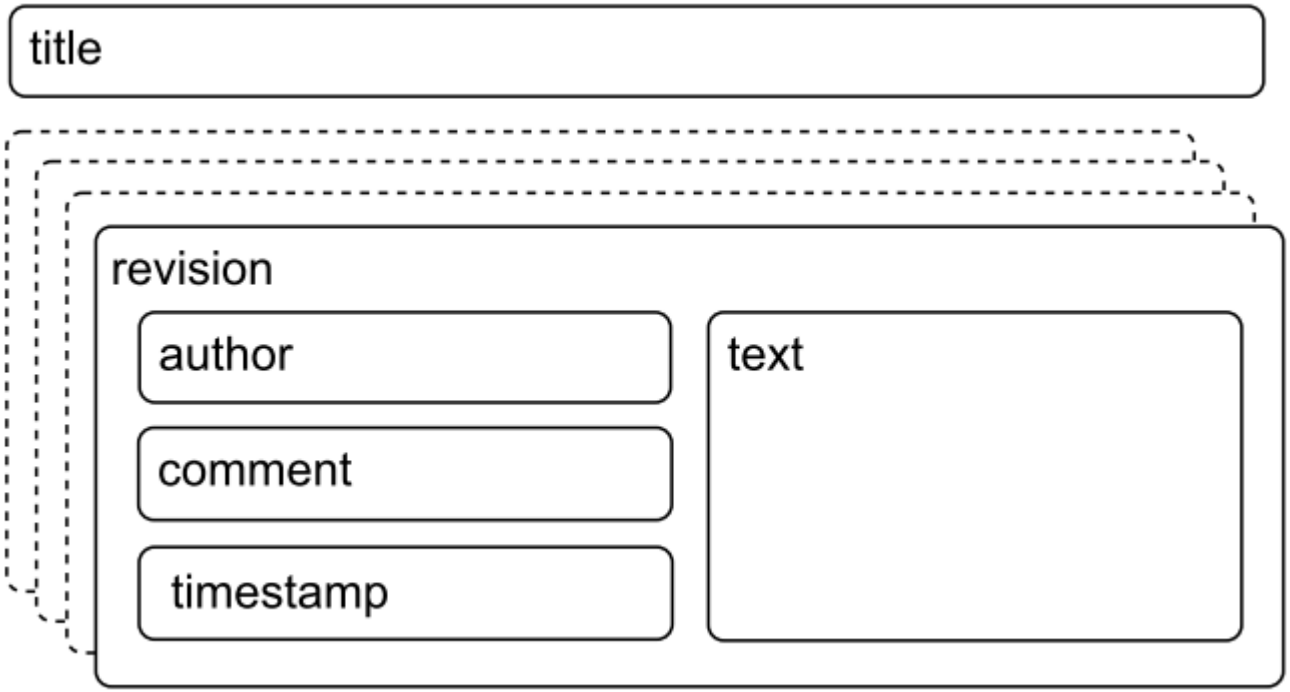
Verzije




```
hbase(main):002:0> create 'test', 'semafor'

hbase(main):004:0> put 'test', '1', 'semafor:', 'crvena'

hbase(main):005:0> put 'test', '1', 'semafor:', 'zuta'

hbase(main):006:0> put 'test', '1', 'semafor:', 'zelena'
0 row(s) in 0.0040 seconds

hbase(main):007:0> get 'test', '1'
COLUMN                          CELL
 semafor:                       timestamp=1372341842883, value=zelena

hbase(main):010:0> get 'test', '1', {COLUMN => 'semafor:', VERSIONS => 4}
COLUMN                          CELL
 semafor:                       timestamp=1372341842883, value=zelena
 semafor:                       timestamp=1372341838768, value=zuta
 semafor:                       timestamp=1372341829568, value=crvena
```

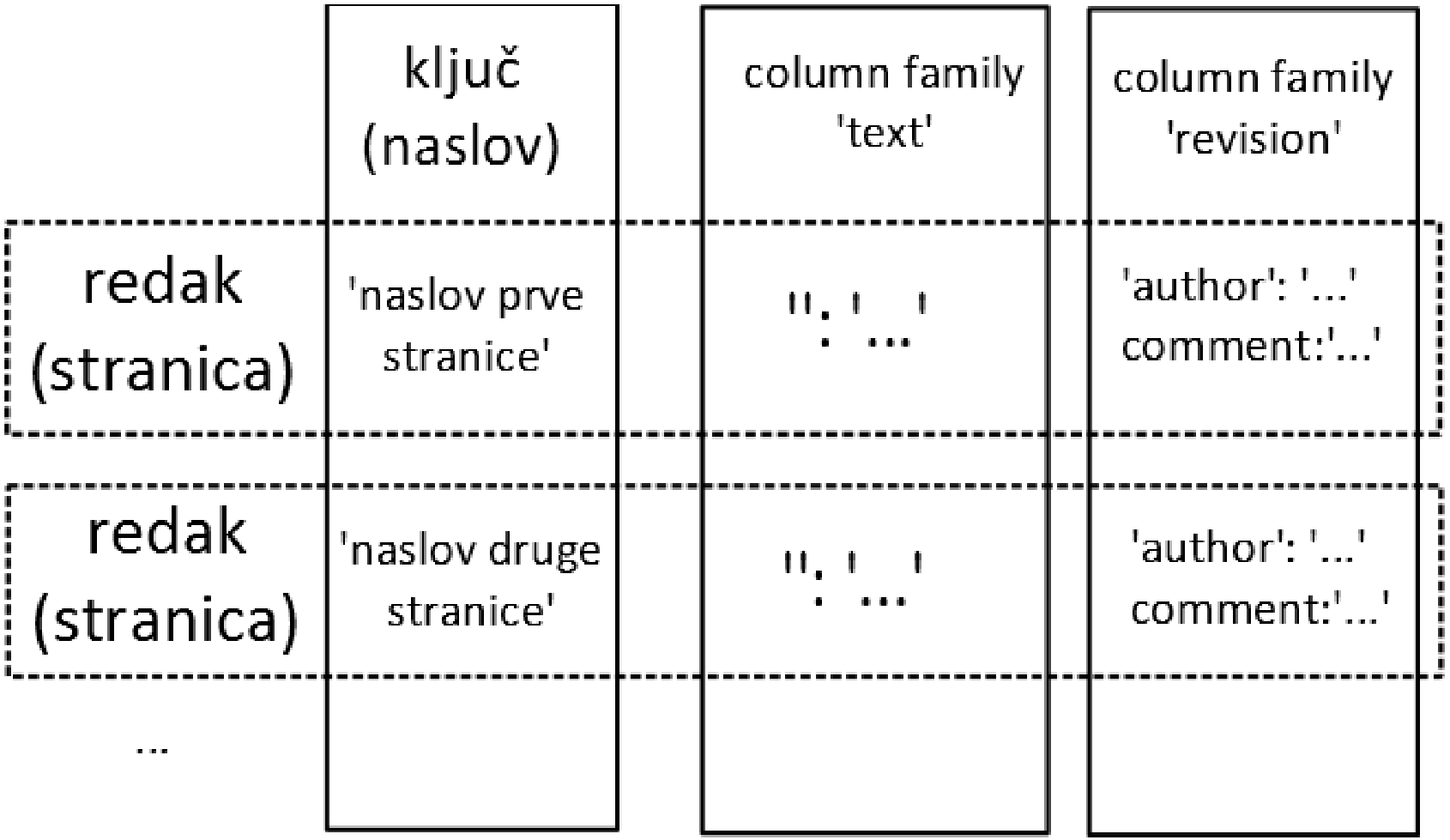
Primjer - wiki*

```
hbase(main):007:0> create 'wiki',
{NAME => 'text', VERSIONS => org.apache.hadoop.hbase.HConstants::ALL_VERSIONS },
{NAME => 'revision', VERSIONS => org.apache.hadoop.hbase.HConstants::ALL_VERSIONS}
0 row(s) in 0.2040 seconds
```



**Preuzet iz knjige Seven databases in seven weeks*

Primjer - wiki (2)



Primjer - wiki (3) – unos stranice

Unesimo prvi redak:

```
hbase(main):002:0> put 'wiki', 'Pocetna', 'text:', 'Dobrodosli'
0 row(s) in 0.6750 seconds

hbase(main):003:0> put 'wiki', 'Pocetna', 'revision:author', 'igor'
0 row(s) in 0.0050 seconds

hbase(main):004:0> put 'wiki', 'Pocetna', 'revision:comment', 'Prvi zapis u mom
wikiju'
0 row(s) in 0.0130 seconds

hbase(main):005:0> get 'wiki', 'Pocetna'
COLUMN                          CELL
 revision:author                timestamp=1372332177394, value=igor
 revision:comment               timestamp=1372332182860, value=Prvi zapis u mom
wikiju
 text:                          timestamp=1372332177312, value=Dobrodosli
3 row(s) in 0.0340 seconds
```

Nije idealno - ts je različit. Nažalost, shell ne dopušta unos više od jedne vrijednosti u jednoj naredbi, pa je za to potrebno korisititi API.

Primjer - wiki (4) - alter table

- Da bi se izmijenila, tablica se mora "isključiti"

```
hbase(main):004:0> disable 'wiki'
0 row(s) in 2.0530 seconds

hbase(main):005:0> alter 'wiki',
{ NAME => 'text', COMPRESSION => 'GZ', BLOOMFILTER => 'ROW'}

Updating all regions with the new schema...
1/1 regions updated.
Done.
0 row(s) in 1.0970 seconds

hbase(main):006:0> enable 'wiki'
0 row(s) in 1.1080 seconds
```

Primjer - wiki (5) - import wikipedije

- S Interneta dohvaćamo "dump" wikipedije i odmah punimo hbase tablicu (koristeći ruby skriptu koja parsira xml)*

```
hbase(main):004:0> curl
http://dumps.wikimedia.org/enwikibooks/20130626/enwikibooks-20130626-pages-
articles-multistream.xml.bz2 | bzcat | ./bin/hbase shell
./ruby/import_from_wikipedia.rb
```

```
…
 95  116M   95  110M    0     0   765k      0  0:02:35  0:02:28  0:00:07  800k 100500 records
inserted (Ba Zi/Date Selection in 1927)
101000 records inserted (Development Cooperation Handbook/The video resources linked to this
handbook/The Documentary Story/Searching for the questions to ask)
 97  116M   97  113M    0     0   756k      0  0:02:37  0:02:33  0:00:04  489k 101500 records
inserted (Template:DPL)
 97  116M   97  113M    0     0   753k      0  0:02:38  0:02:34  0:00:04  380k 102000 records
inserted (Ba Zi/HP H6)
 98  116M   98  114M    0     0   748k      0  0:02:39  0:02:36  0:00:03  380k 102500 records
inserted (Managing your business with anyPiece/Inventory system for recycle business/Storage
Packaging/Storage packaging - History)
103000 records inserted (How to Ace FYLSE/October 2012 Exam)
 99  116M   99  115M    0     0   750k      0  0:02:38  0:02:37  0:00:01  480k 103500 records
inserted (ElementarySpanish/Meeting people/Lesson 1)
100  116M  100  116M    0     0   752k      0  0:02:38  0:02:38 --:--:--  712k
```

*Detaljnije opisano u knjizi *Seven databases in seven weeks*

Primjer - wiki (6) - import wikipedije

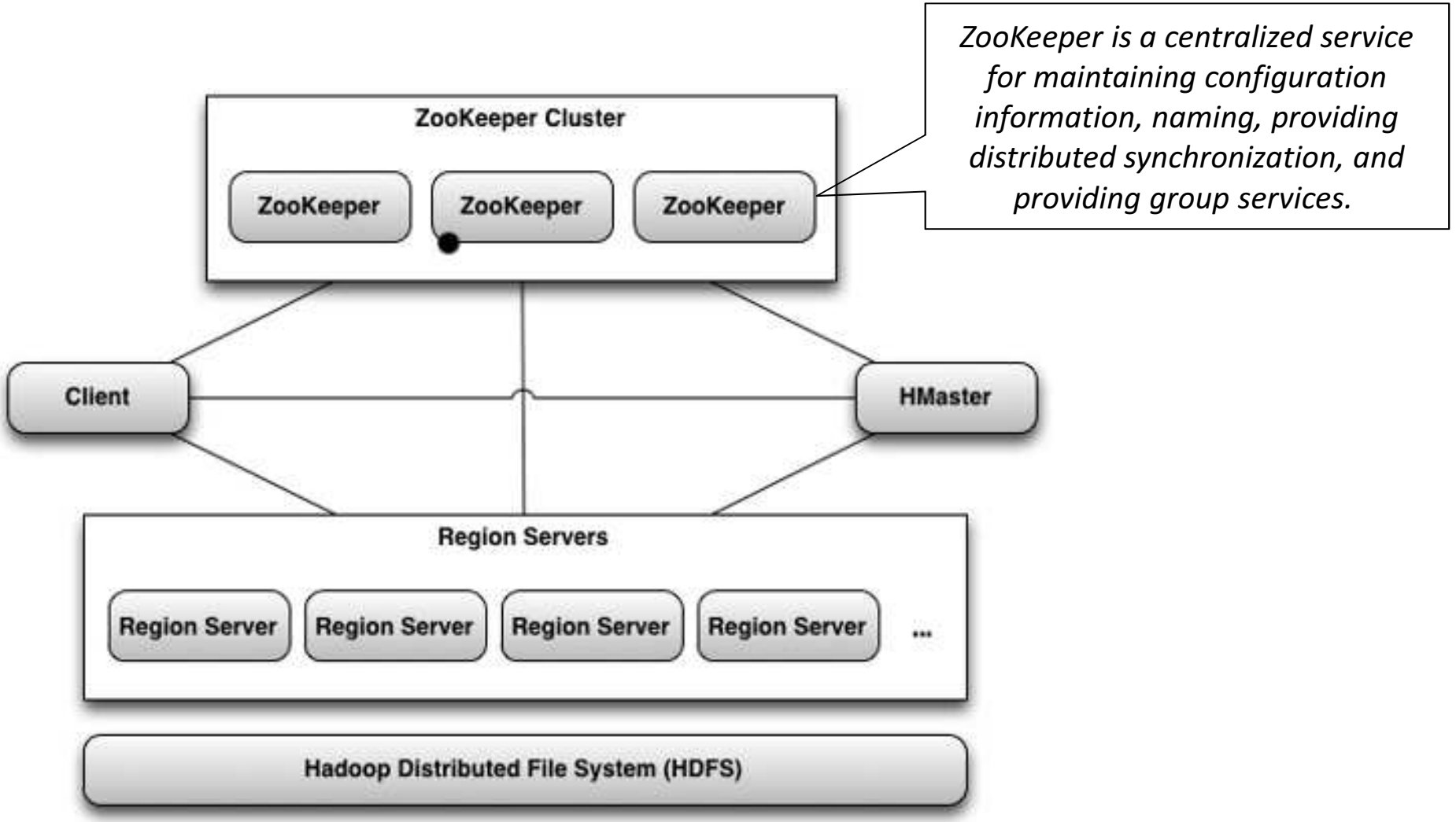
```
hbase(main):004:0> count 'wiki', INTERVAL => 10000
Current count: 10000, row: Blender 3D: Noob to Pro/Basic Animation/Rendering
Current count: 20000, row: Chemical engineering
Current count: 30000, row: Development Cooperation Handbook/The video resources linked to
this handbook/The Documentary
Story/The KFI story
Current count: 40000, row: File:NotesCornell.png
Current count: 50000, row: Hebrew Roots/The Law and the Covenants/Covenants:The Covenant
of Israeli Sovereignty
Current count: 60000, row: Mandarin Chinese/Sentences/He is a student
Current count: 70000, row: Programming:Game Maker/Intro
Current count: 80000, row: Structural Biochemistry/Protein function/Heme
group/Hemoglobin/Affinity Constant
Current count: 90000, row: Template:User language/sah
Current count: 100000, row: Wikijunior Languages/Finnish
103927 row(s) in 8.5280 seconds

hbase(main):005:0> get 'wiki', 'Chemical engineering'
COLUMN                          CELL
 revision:author                timestamp=1277299531, value=Adrignola
 revision:comment               timestamp=1277299531, value=Redirected page to
[[Subject:Chemical engineering]]
 text:                          timestamp=1277299531, value=#REDIRECT [[Subject:Chemical
engineering]]
3 row(s) in 0.0470 seconds
```

Wiki - smještaj podataka

```
[root@minerva hbase]# du -h
4.0K    ./.oldlogs
4.0K    ./.corrupt
4.0K    ./.logs/minerva.zpr.fer.hr,45246,1371804716705
8.0K    ./.logs
4.0K    ./.archive
4.0K    ./.tmp
4.0K    ./-ROOT-/.tmp
12K     ./-ROOT-/70236052/.oldlogs
4.0K    ./-ROOT-/70236052/.tmp
12K     ./-ROOT-/70236052/info
12K     ./-ROOT-/70236052/recovered.edits
52K     ./-ROOT-/70236052
68K     ./-ROOT-
12K     ./.META./1028785192/.oldlogs
4.0K    ./.META./1028785192/.tmp
12K     ./.META./1028785192/info
40K     ./.META./1028785192
44K     ./.META.
12M     ./wiki/c089c61a331c1404a0d47f7f2b118efe/revision
4.0K    ./wiki/c089c61a331c1404a0d47f7f2b118efe/.tmp
68M     ./wiki/c089c61a331c1404a0d47f7f2b118efe/text
79M     ./wiki/c089c61a331c1404a0d47f7f2b118efe
4.0K    ./wiki/.tmp
10M     ./wiki/2d11659bf990bcf13a9d8fa1d62025e2/revision
4.0K    ./wiki/2d11659bf990bcf13a9d8fa1d62025e2/.tmp
65M     ./wiki/2d11659bf990bcf13a9d8fa1d62025e2/text
75M     ./wiki/2d11659bf990bcf13a9d8fa1d62025e2
153M    ./wiki
154M    .
```

Dvije regije

Tipična arhitektura Hbase klastera



Preuzeto s: http://www.packtpub.com/sites/default/files/Article-Images/7140_08_01.png

Hbase, zaključno (1)

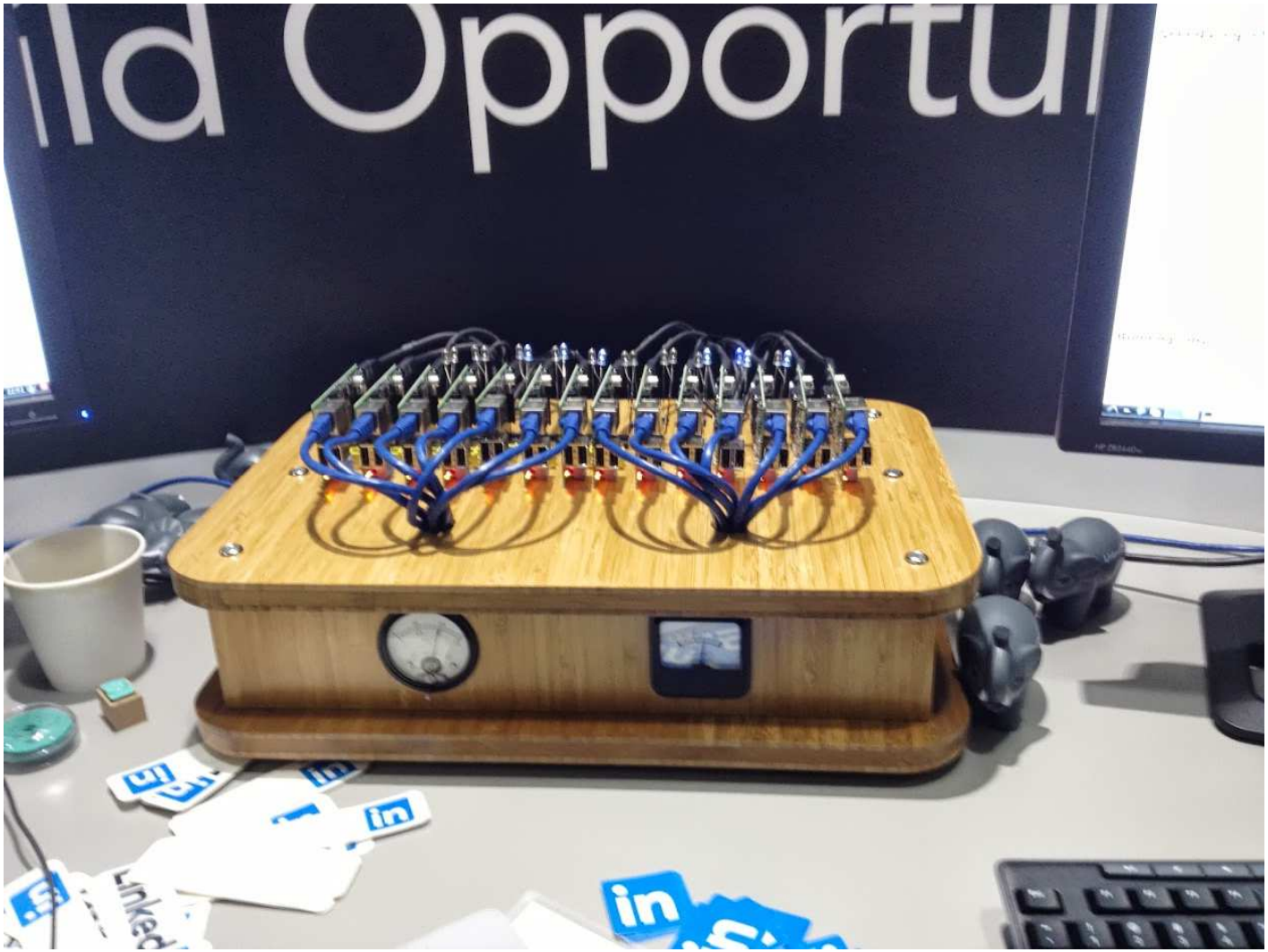
- Prednosti:
  - Robustan
  - Skalabilan (GB->TB)
  - Verzije
- Nedostatci:
  - Minimalno pet čvorova
  - Administracija
  - Prateći "ekosustav" (+/-)
  - Nema sortiranja (osim ključeva redaka) ni indeksiranja
  - Nema tipova podataka (neinterpretirani bajtovi)

Hbase zaključno (2)

- Koristiti za:
  - *Event logging*
  - *Content management*

- Ne korisiti kad:
  - „Malo” podataka
  - ACID
  - Upiti koji agregiraju podatake (morate to napraviti client-side)
  - Rana faza razvoja (dok nisu poznati query patterns)

Što je ovo?




Neo4j

Neo4j

- “whiteboard friendly,”
- All about relationships
- U raznim „veličinama”:
  - Emedded
  - Cluster support, MS replikacija
- Može pohraniti desetke milijardi čvorova i bridova

- Uptini jezici:
  - Gremlin
  - Cypher
- Language binding
- Moguće je indeksirati svojstva čvora i/ili brida (za start upita)
  (koristi *Lucene* za indeksiranje)

Cypher

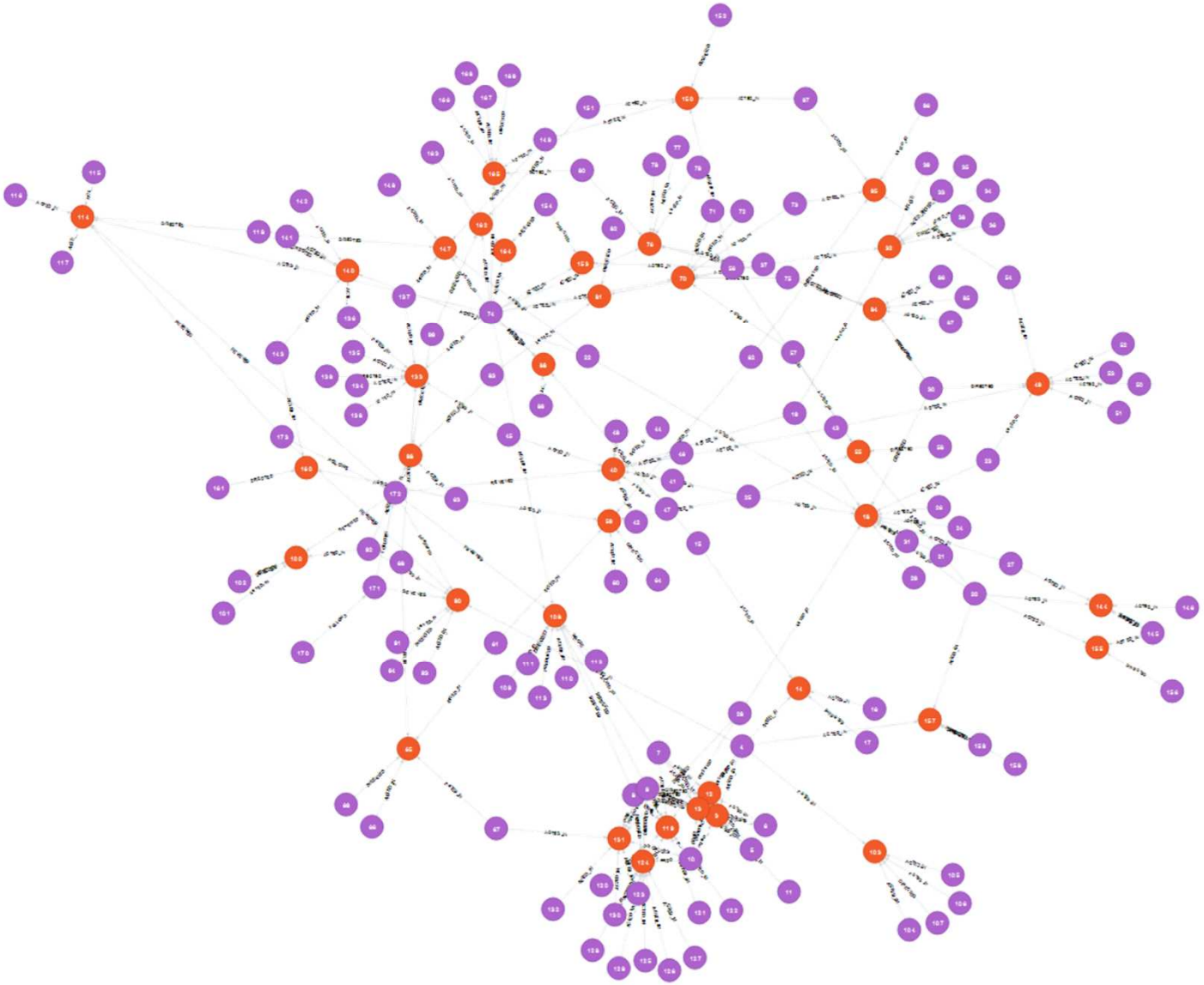
- Pogledati http://docs.neo4j.org/refcard/1.9/
- Na ispitu vas možemo pitati jednostavniji Cypher upit, pri čemu smijete koristiti gore navedeni podsjetnik

```
START
[MATCH]
[WHERE]
RETURN [ORDER BY] [SKIP] [LIMIT]
```

Primjer: unos zapisa

```
CREATE (TheMatrix:Movie {title:'The Matrix', released:1999, tagline:'Welcome to the
Real World'})
CREATE (Keanu:Person {name:'Keanu Reeves', born:1964})
CREATE (Carrie:Person {name:'Carrie-Anne Moss', born:1967})
CREATE (Laurence:Person {name:'Laurence Fishburne', born:1961})
CREATE (Hugo:Person {name:'Hugo Weaving', born:1960})
CREATE (AndyW:Person {name:'Andy Wachowski', born:1967})
CREATE (LanaW:Person {name:'Lana Wachowski', born:1965})
CREATE (JoelS:Person {name:'Joel Silver', born:1952})
CREATE
  (Keanu)-[:ACTED_IN {roles:['Neo']}]->(TheMatrix),
  (Carrie)-[:ACTED_IN {roles:['Trinity']}]->(TheMatrix),
  (Laurence)-[:ACTED_IN {roles:['Morpheus']}]->(TheMatrix),
  (Hugo)-[:ACTED_IN {roles:['Agent Smith']}]->(TheMatrix),
  (AndyW)-[:DIRECTED]->(TheMatrix),
  (LanaW)-[:DIRECTED]->(TheMatrix),
  (JoelS)-[:PRODUCED]->(TheMatrix)

CREATE (Emil:Person {name:"Emil Eifrem", born:1978})
CREATE (Emil)-[:ACTED_IN {roles:["Emil"]}]->(TheMatrix)

CREATE (TheMatrixReloaded:Movie {title:'The Matrix Reloaded', released:2003,
tagline:'Free your mind'})
```

Primjer: filmovi i glumci



Cypher - primjeri

- Dohvati Toma Hanksa:

```
MATCH (tom {name: "Tom Hanks"})
RETURN tom
```

- Dohvati čvora naslova „Cloud Atlas”:

```
MATCH (cloudAtlas {title: "Cloud Atlas"})
RETURN cloudAtlas
```

- Dohvati imena deset osoba (čvorova tipa Person):

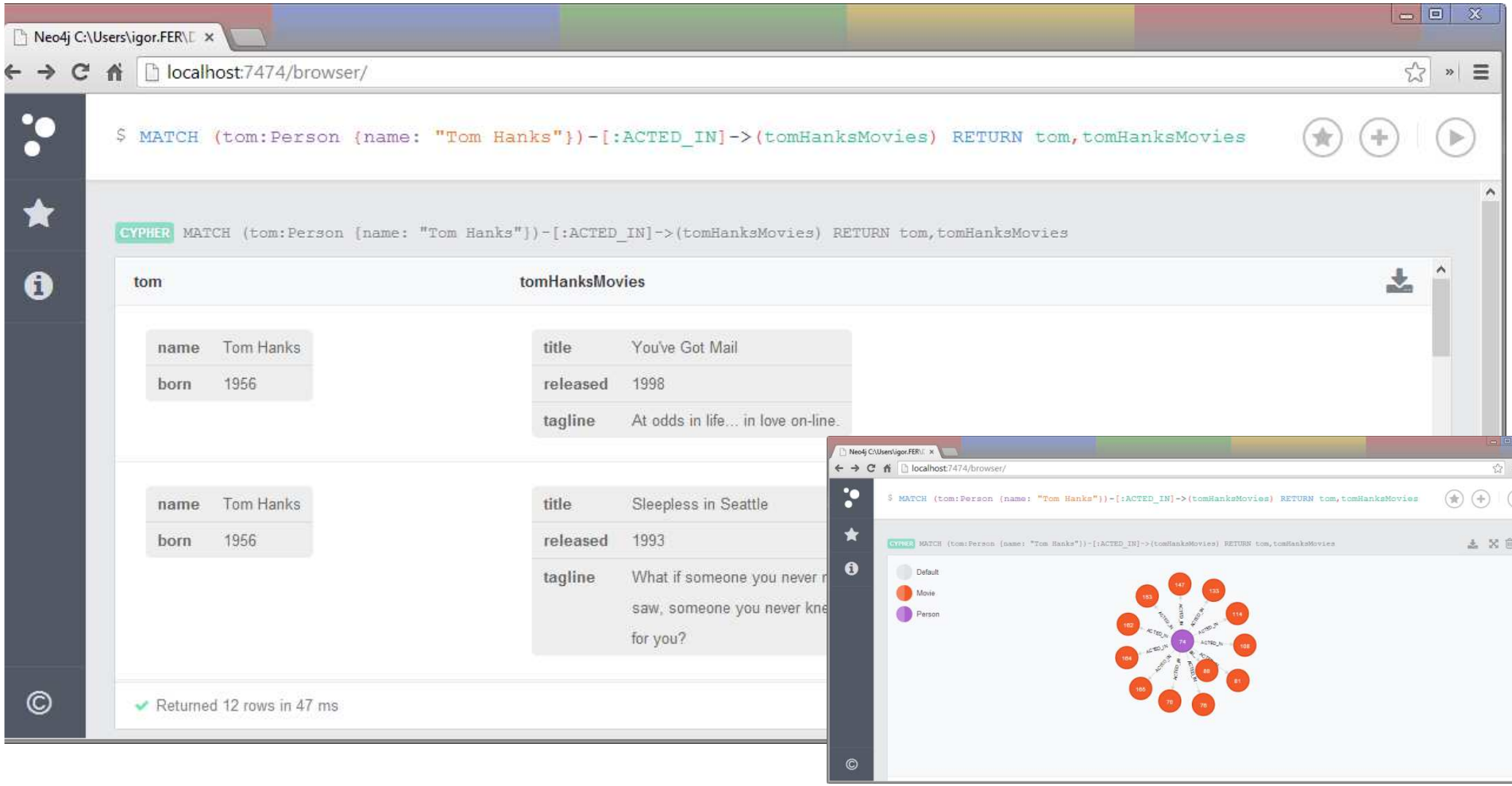
```
MATCH (people:Person)
RETURN people.name LIMIT 10
```

- Dohvati filmove iz devedesetih:

```
MATCH (nineties:Movie)
WHERE nineties.released > 1990 AND nineties.released < 2000
RETURN nineties.title
```

Primjer: filmovi u kojima je glumio TH

```
MATCH (tom:Person {name: "Tom Hanks"})-[:ACTED_IN]->(tomHanksMovies)
RETURN tom,tomHanksMovie
```



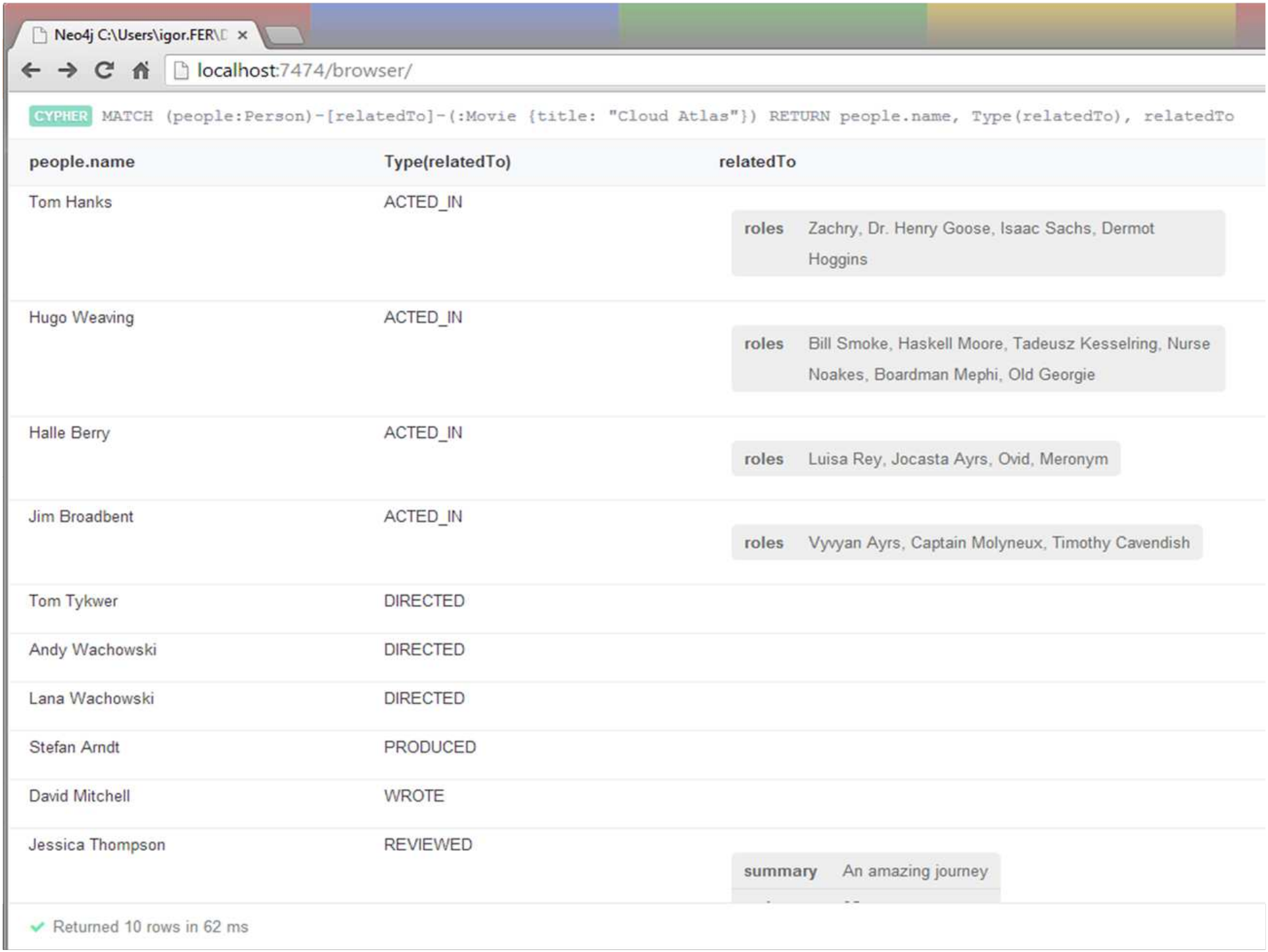
Primjer

- Glumci koji su glumili s TH-om:

```
MATCH (tom:Person {name:"Tom Hanks"})-[:ACTED_IN]->(m)<-[:ACTED_IN]-(coActors)
RETURN coActors.name
```

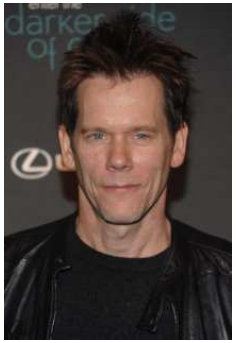
- Na koji način su ljudi povezani s „Cloud Atlasom”?

```
MATCH (people:Person)-[relatedTo]-(:Movie {title: "Cloud Atlas"})
RETURN people.name, Type(relatedTo), relatedTo
```


Neo4j C:\Users\igor.FER\
localhost:7474/browser/
CYPHER MATCH (people:Person)-[relatedTo]-(:Movie {title: "Cloud Atlas"}) RETURN people.name, Type(relatedTo), relatedTo
people.name
Type(relatedTo)
relatedTo
Tom Hanks
ACTED_IN
roles Zachry, Dr. Henry Goose, Isaac Sachs, Dermot Hoggins
Hugo Weaving
ACTED_IN
roles Bill Smoke, Haskell Moore, Tadeusz Kesselring, Nurse Noakes, Boardman Mephi, Old Georgie
Halle Berry
ACTED_IN
roles Luisa Rey, Jocasta Ayrs, Ovid, Meronym
Jim Broadbent
ACTED_IN
roles Vyvyan Ayrs, Captain Molyneux, Timothy Cavendish
Tom Tykwer
DIRECTED
Andy Wachowski
DIRECTED
Lana Wachowski
DIRECTED
Stefan Arndt
PRODUCED
David Mitchell
WROTE
Jessica Thompson
REVIEWED
summary An amazing journey
Returned 10 rows in 62 ms

Six Degrees of Kevin Bacon

*A small proportion of actors have an undefined **Bacon number**, meaning that they cannot be linked to Bacon in any number of connections at all. According to the Oracle of Baconwebsite, approximately 12% of all actors cannot be linked to Bacon using its criteria, but this number is difficult to verify.* http://en.wikipedia.org/wiki/Six_Degrees_of_Kevin_Bacon



```
MATCH (bacon:Person {name:"Kevin Bacon"})-[*1..4]-(hollywood)
RETURN DISTINCT hollywood
```

- Ugrađeni shortest path algoritam – najkraći put KB->MR

```
MATCH p=shortestPath(
  (bacon:Person {name:"Kevin Bacon"})-[*]-(meg:Person {name:"Meg Ryan"})
)
RETURN p
```

Brisanje

- Brisanje svih osoba, filmova i veza

```
MATCH (a:Person),(m:Movie)
OPTIONAL MATCH (a)-[r1]-(), (m)-[r2]-()
DELETE a,r1,m,r2
```

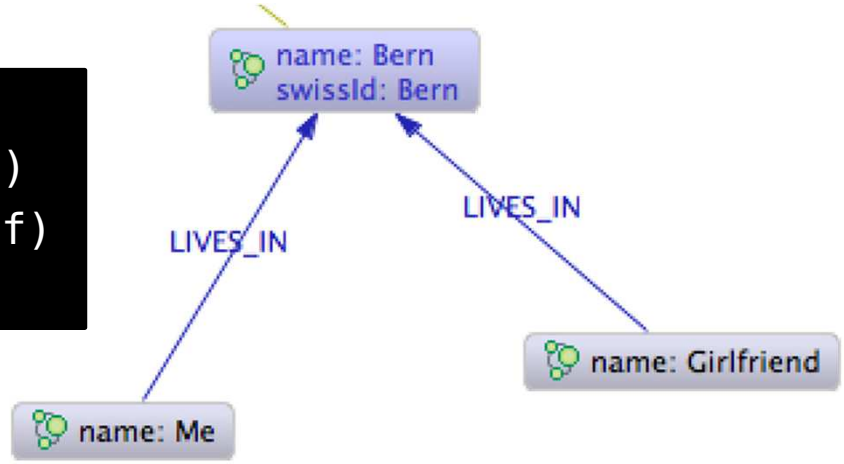
- Provjera, ispis svih čvorova (ne bi smjelo vratiti išta):

```
MATCH (n) RETURN n
```

Opaska: zašto nije uključen početni čvor?

- Pitanje: http://stackoverflow.com/questions/16748655/neo4j-cypher-why-is-starting-node-excluded

```
START me=node:node_auto_index(name="Me")
MATCH me-[:LIVES_IN]->()<-[:LIVES_IN]-(f)
RETURN f.name;
```



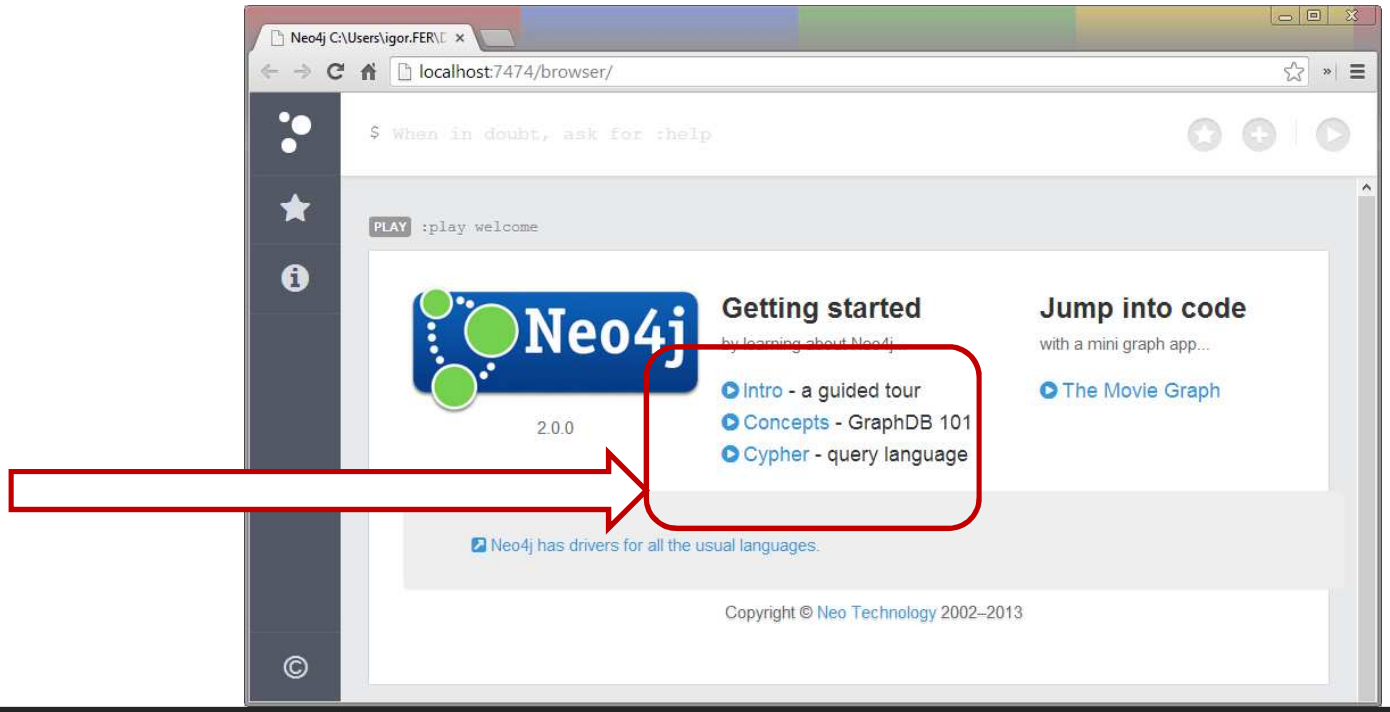
vraća:

```
Results
f.name  Girlfriend;
```

…zato jer put naveden u MATCH nikad neće dvaput sadržavati istu relaciju.

Zadaci za vježbu

- Instalirati Neo4j http://www.neo4j.org/download/other_versions
- Proći kroz Getting started tutoriale, čime ćete isprobati i osnove Cyphera, odnosno isprobati sve primjere s predavanja
- Postoje i online tutorial (http://www.neo4j.org/learn/cypher) koji je i opširniji od ovoga (ne morate ga proći)



Neo4j, zaključno (1)

- Prednosti:
  - Typless, schemaless, nema ograničenja na način povezivanja podataka
  - Indeksiranje
  - Upitni jezici
  - REST interface
  - Brz
  - Enterprise edition, High availability
- Nedostatci:
  - Community free, Enterpise nije
  - Može replicirati samo cijeli graf

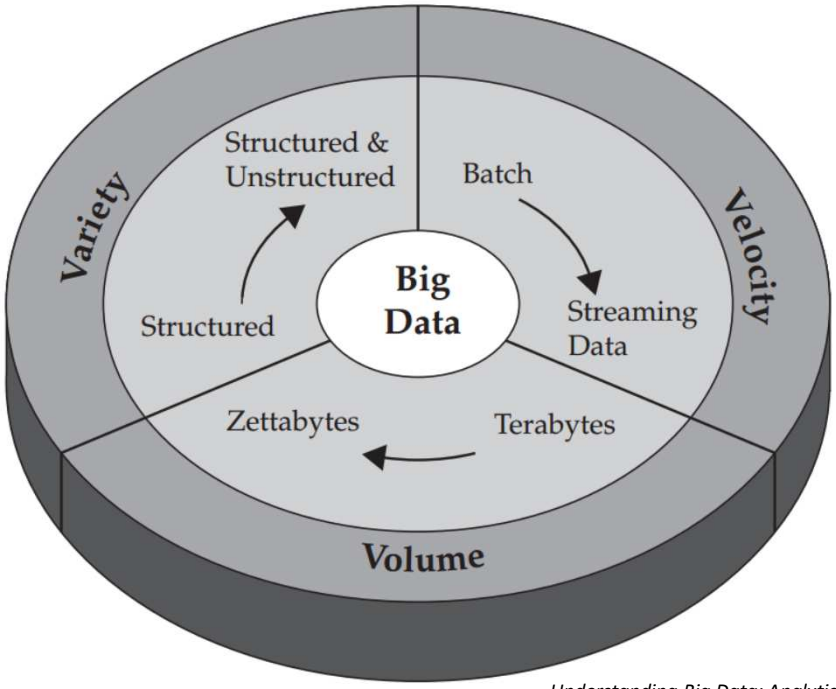
Neo4j, zaključno (2)

- Koristiti za:
  - *Connected data (npr. Social networks)*
  - *Routing, dispatch, location-based services*
  - *Recommendation engine*
- Ne korisiti kad:
  - Ažuriranje svih ili podskupa entiteta

Big Data

Big Data (1)

- Oni podaci kojima se ne može upravljati, obrađivati ili analizirati koristeći tradicionalne metode i alate.
- Problemi uključuju: snimanje, organizaciju, pohranu, pretraživanje, dijeljenje, prijenos, analizu i vizualizaciju.*
- 3Vs:
  - **Volume** = volumen, veličina
  - **Variety** = različitost, heterogenost
  - **Velocity** = brzina

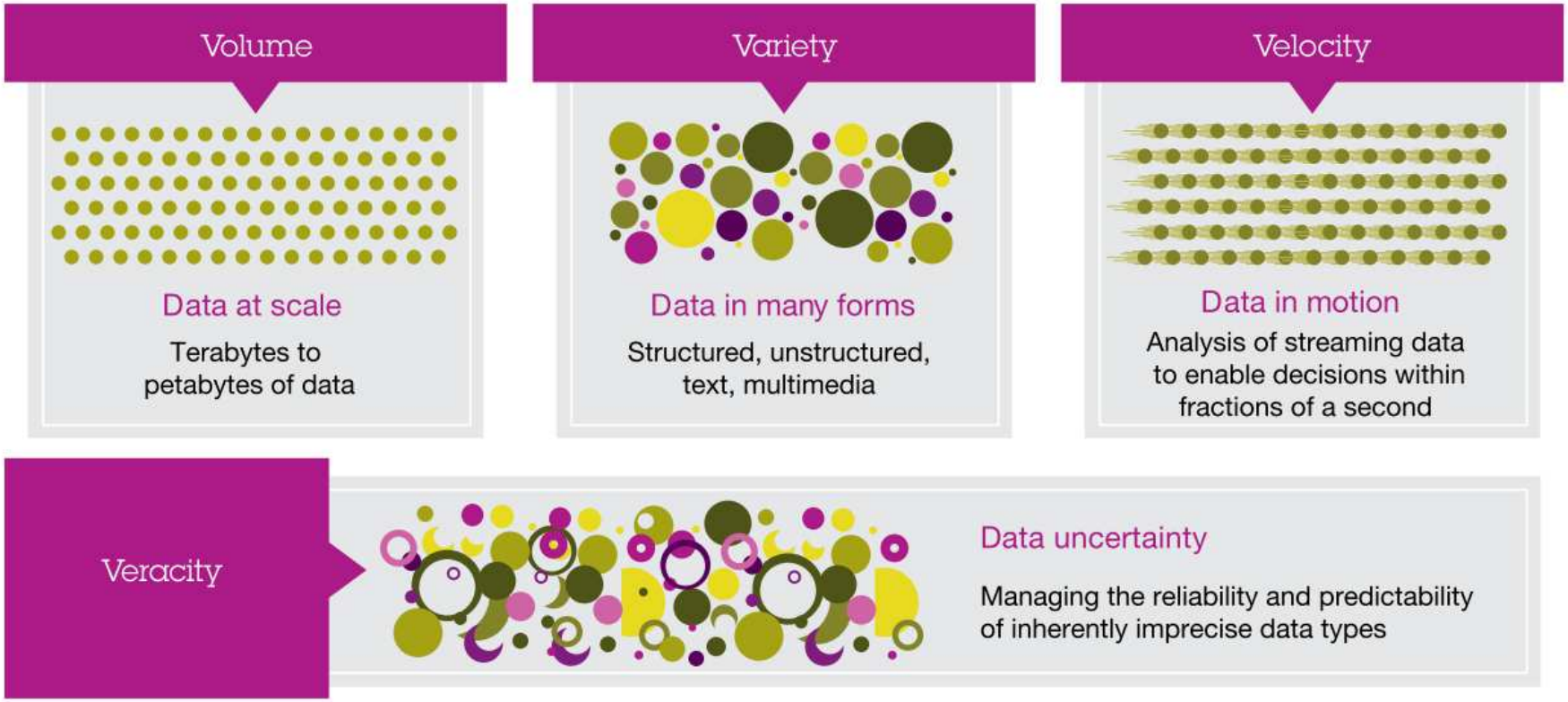


Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data, Zikopoulos, Eaton, DeRoos, Deutsch, Lapis

*http://en.wikipedia.org/wiki/Big_data

Big Data (2)

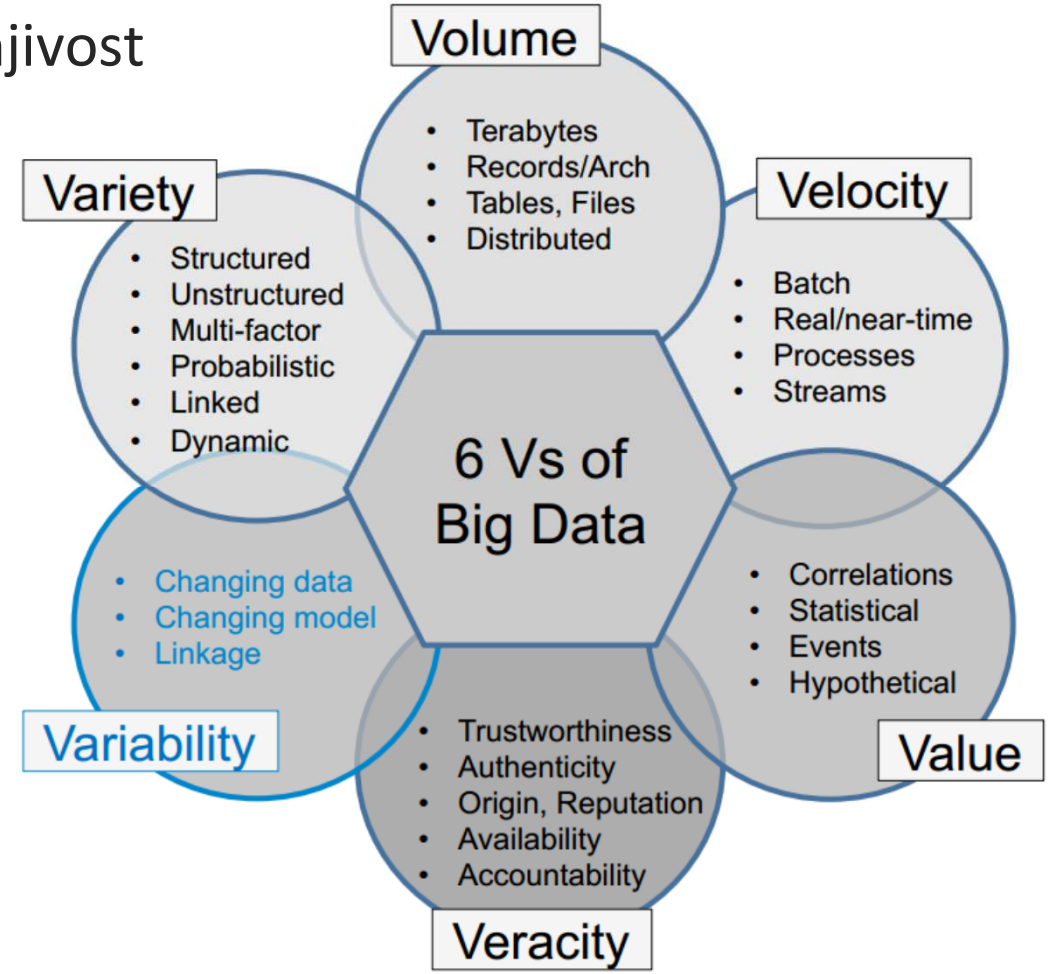
- **+Veracity** = vjerodostojnost (for example: sentiment and truthfulness in humans)



IBM Institute for Business Value: „Analytics: The real-world use of big data"

Big Data (3)

- **+Value** = vrijednost
- **+Variability** = promjenjivost

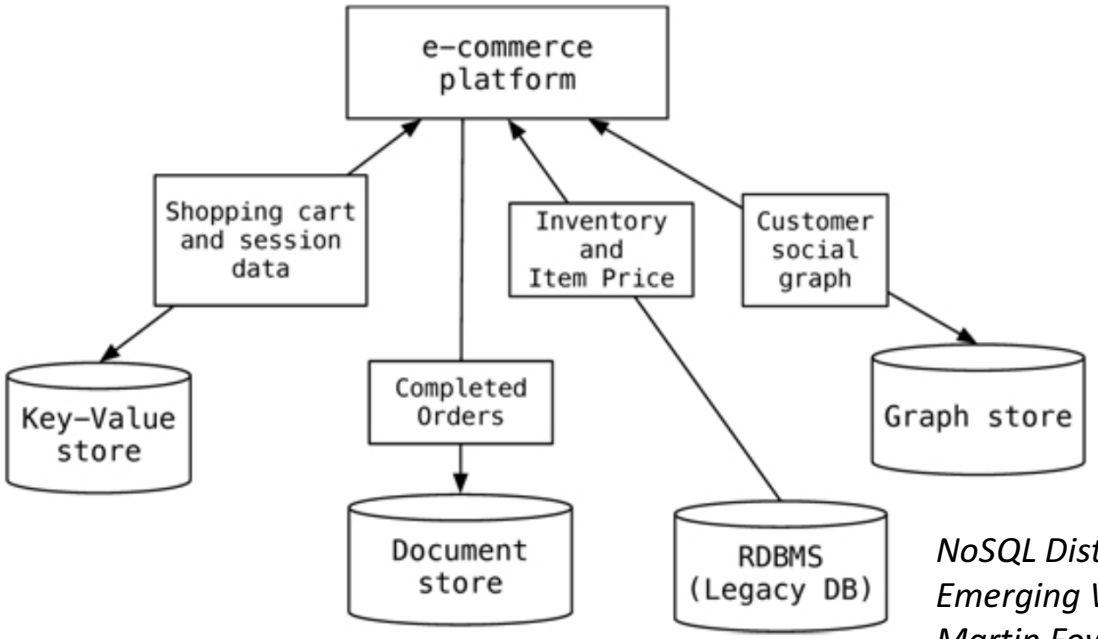


https://rd-alliance.org/sites/default/files/attachment/Breakout_reports.pdf

Polyglot persistance

Polyglot persistance (1)

- Različite baze podataka su namjenjene za rješavanje različitih problema
- Hibridni pristup perzistenciji podataka:
  kombiniranje tehnologija kako bi se najbolje zadovoljile različite klase potreba za perzistencijom podataka
- Npr.



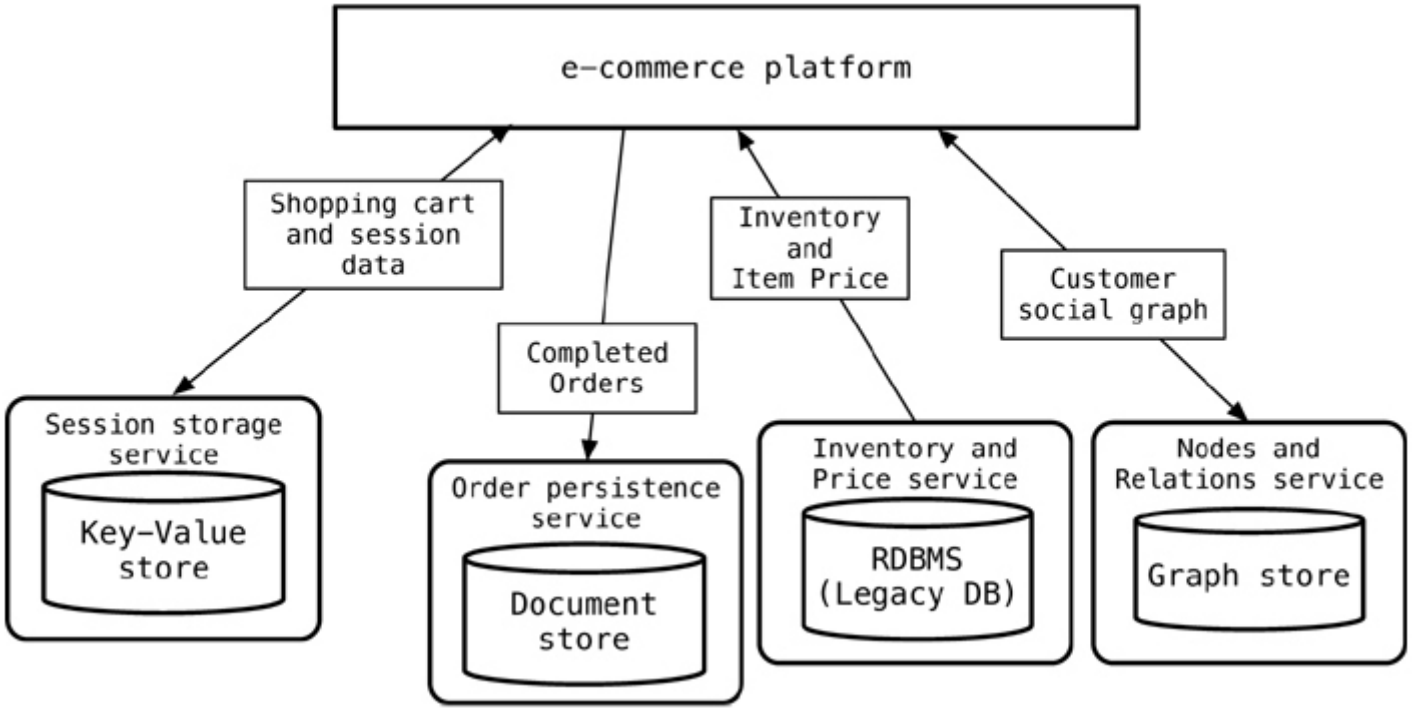
NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence, Martin Fowler

Ili primjer:

- *http://www.martinfowler.com/bliki/PolyglotPersistence.html*

Polyglot persistance (2)

- Ili, ako baze podataka izložimo putem servisa (da smanjimo utjecaj na ostatak sustava):



NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence, Martin Fowler

Polyglot persistance (3)

- Problemi:
  - *Poly-skilled DBA*: nadzor, *backup*, *restore*,...
  - Dozvole, podrška, alati, nadogradnja, driveri, itd.
  - Sigurnost, pravila pristupa
  - Složenost kod razvoja, održavanja, *deployment*, ...

NoSQL zaključno

NoSQL zaključno: 10=5+5

- **10 things: 5 advantages**
- http://www.techrepublic.com/blog/10-things/10-things-you-should-know-about-nosql-databases/

1. **Elastic scaling**
   - Horizontalno skalabilni
2. **Big Data**
   - Omogućuje rad s BD-om
3. **Goodbye DBAs?**
   - Automatski oporavak, podešavanje, distribucija
4. **Economics**
   - Uglavnom besplatan, FLOSS softver
   - Temeljeni na *commodity* računalima
5. **Flexible Data Models**
   - Schemaless

NoSQL zaključno: 10=5+5

- **10 things: 5 challenges**
- http://www.techrepublic.com/blog/10-things/10-things-you-should-know-about-nosql-databases/

1. **Maturity**
   - Još uvijek u aktivnom razvoju
2. **Support**
   - Uglavnom FLOSS; startups, ograničenih sredstava i kredibiliteta
3. **Administration**
   - Zahtijeva puno truda/znanja za instalaciju i održavanje
4. **Analytics & BI**
   - Fokusirani na web
   - Ograničeni ad-hoc upiti
5. **Expertise**
   - Malen broj poznavatelja na tržištu

NoSQL, zaključno:

- Nisu zamjena za relacijske sustave, već namjenski softver za određene probleme
- Najčešće: povećanje dostupnosti (*availabilty*) i učinkovitosti na račun dosljednosti (*consistency*)
- BASE a ne ACID
- Velike količine podataka
- Fleksibilna shema podataka
- Horizontalna skalabilnost
- Nije zrela tehnologija, nedostatak standarda
- Najčešće besplatno/otvorenog koda
- Relativno lako za uspostaviti (za testiranje)