SODOBNE NERELACIJSKE PODATKOVNE BAZE (NOSQL)

UNIVERZITETNI ŠTUDIJ RI IN UI, 3. LETNIK
MAGISTRSKI ŠTUDIJ RI

Matjaž Kukar, 2022/23

Vsebina

- problemi relacijskih podatkovnih baz (teoretični in praktični)
- nerelacijske podatkovne baze, NoSQL

- MongoDB (dokumentni SUPB)
  - namestitev
  - delo z dokumenti
  - iskanje
  - posodabljanje
  - vgnezdeni dokumenti
  - kurzorji
  - indeksiranje
- Neo4j (grafni SUPB)
  - namestitev
  - delo z grafi
  - iskanje

Uvod

"Facebook now has more than 1.8 billion monthly active users. Its data storage is more than 300 petabytes. Every 60 seconds on Facebook:

- *510 comments are posted*
- *293,000 statuses are updated*
- *136,000 photos are uploaded"*

"Twitter now has more than 320 million active users, sending > 500 million tweets every day"

"Webscale" aplikacije

- veliko podatkov
- veliko istočasnih uporabnikov
- veliko zahtevanih obdelav
- časovno pogojene obremenitve (Facebook torek in četrtek +20%)

So relacijske baze primerne za takšna bremena?

Problemi z resursi - možne rešitve

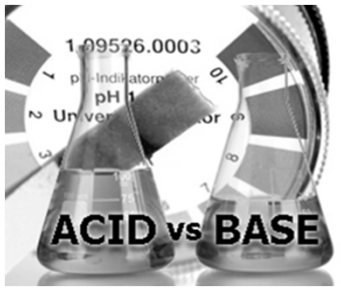
- **vertikalno skaliranje (scaling up):** nadgradnja virov z zmogljivejšimi
  - kje je meja? cena?
- **horizontalno skaliranje (scaling out):** porazdelitev podatkov po več strežnikih
  - master-slave: pisanje se izvaja na glavnem strežniku, branja se lahko izvajajo iz sekundarnih baz (problem - konsistentnost)
  - deljenje podatkov (partitioning, sharding): razdelitev podatkov v podmnožice, boljše skaliranje, vendar izguba možnosti izvedbe stikov med podatki in referenčne integritete
- **druge možne rešitve**
  - replikacija cele baze v več "master" baz,
  - izvedba brez stikov – denormalizacija (stiki so eno od ozkih grl),
  - hranjenje majhnih baz v primarnem spominu,
  - nerelacijske rešitve?

NoSql

Not Only SQL

- **relacijske baze**: *one size fits all*
- **ozka grla**
  - Stične operacije
  - Konsistentnost podatkov v PB pri sočasni uporabi - ACID

- **nerelacijske baze**: *prilagojene zbirke podatkov za specifične aplikacije*
  - večinoma žrtvujejo shemo ACID
  - shema BASE (basically available, soft state, eventually consistent)
  - prednosti: cena, zmogljivost
  - slabosti: pomanjkanje standardov, potrebna specifična priučitev, omejena prenosljivost, nezrelost tehnologije
  - ne uporabljajo fiksne podatkovne sheme in stikov!



BASE vs ACID

Basic Availability

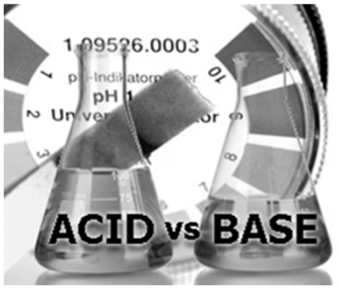
- Podatki v PB so dostopni „večino časa“

Soft-state

- Konsistentnost pisanja ni zagotovljena, ravno tako ne konsistentnost med replikami podatkov (v vsakem trenutku)

Eventual consistency

- Sčasoma se podatek „stabilizira“ v konsistentno stanje



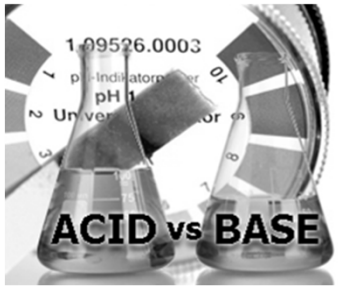
CAP – zaželene lastnosti porazdeljenih sistemov (tudi PB)

NoSQL

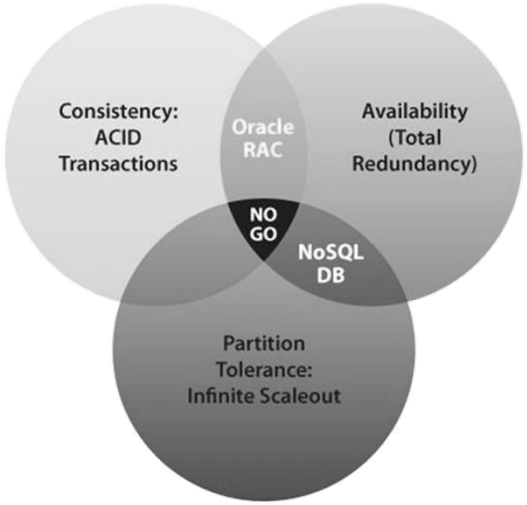
- Consistency - konsistentnost: (ACID)
  Vsako branje prebere zadnjo veljavno vrednost podatka (ali pa branje ni dovoljeno)

- Availability – razpoložljivost:
  Vsaka bralna zahteva dobi vrnjeno vrednost (v principu brez garancije, da gre za zadnjo verzijo podatka).

- Partition tolerance – zmožnost delitve podatkov:
  Sistem deluje kljub težavam (zakasnitve ali izgube podatkov) na poljubno velikem delu (<100%) povezav med vozlišči.



Teorem CAP



- Eric Brewer, 2000; dokaz na MIT 2002
- Vsak sistem za delo s podatki ima tri lastnosti:
  - konsistentnost (Consistency),
  - razpoložljivost (Availability) in
  - zmožnost delitve podatkov (Partitions)
- Teorem pravi: pri deljenju podatkov
  lahko zagotovimo **največ dve** od teh treh lastnosti

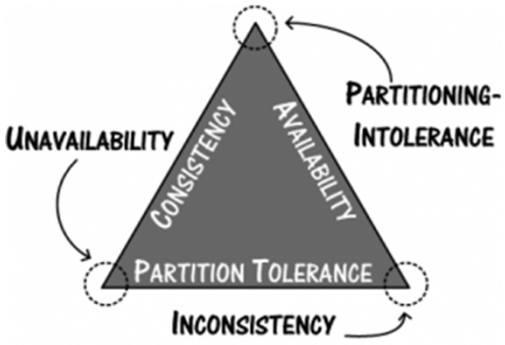
Primer: podatke razdelimo na več računalnikov. Kasnejša posodobitev podatka zahteva posodobitev vseh kopij. Scenarija:

- zaklenemo vse kopije, da zagotovimo konsistentnost (zmanjšana razpoložljivost), ali
- žrtvujemo konsistentnost na račun večje razpoložljivosti (sčasoma podatki postanejo konsistentni)

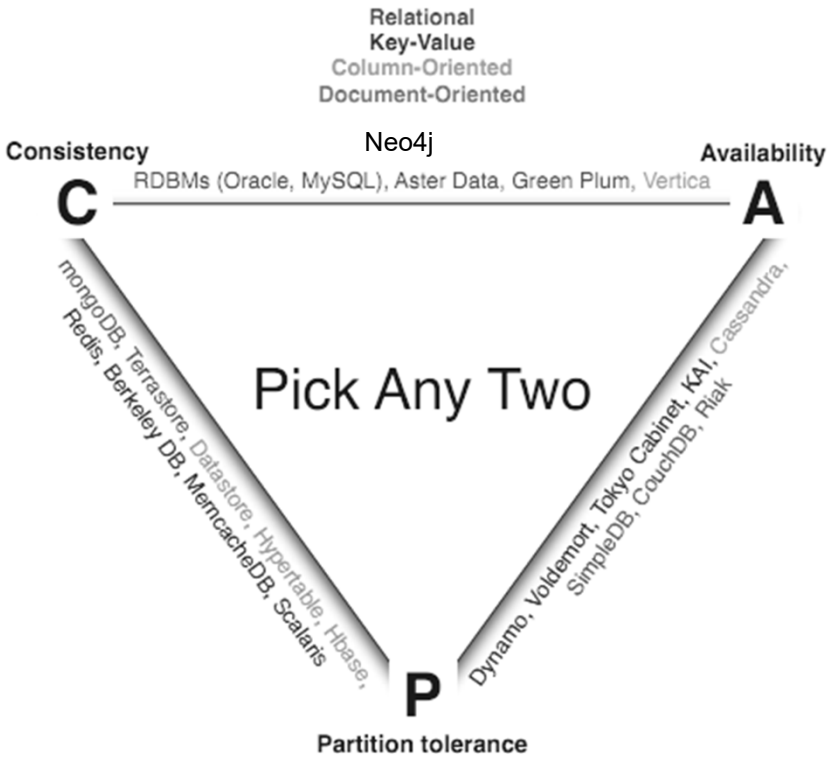
Teorem CAP

Torej: žrtvovati je potrebno enega od kriterijev: C, A ali P

- **konsistentni in razpoložljivi (CA)** sistemi imajo težavo z deljenjem podatkov, težave običajno obvladujejo z replikacijo,
- **konsistentni, porazdeljeni sistemi (CP)** imajo težavo z razpoložljivostjo podatkov ob naporu zagotavljanja njihove konsistentnosti,
- **razpoložljivi, porazdeljeni sistemi (AP)** dosegajo sčasno konsistentnost (eventual consistentcy) z replikacijo podatkov in občasnim preverjanjem konsistentnosti v podatkovnih vozliščih.



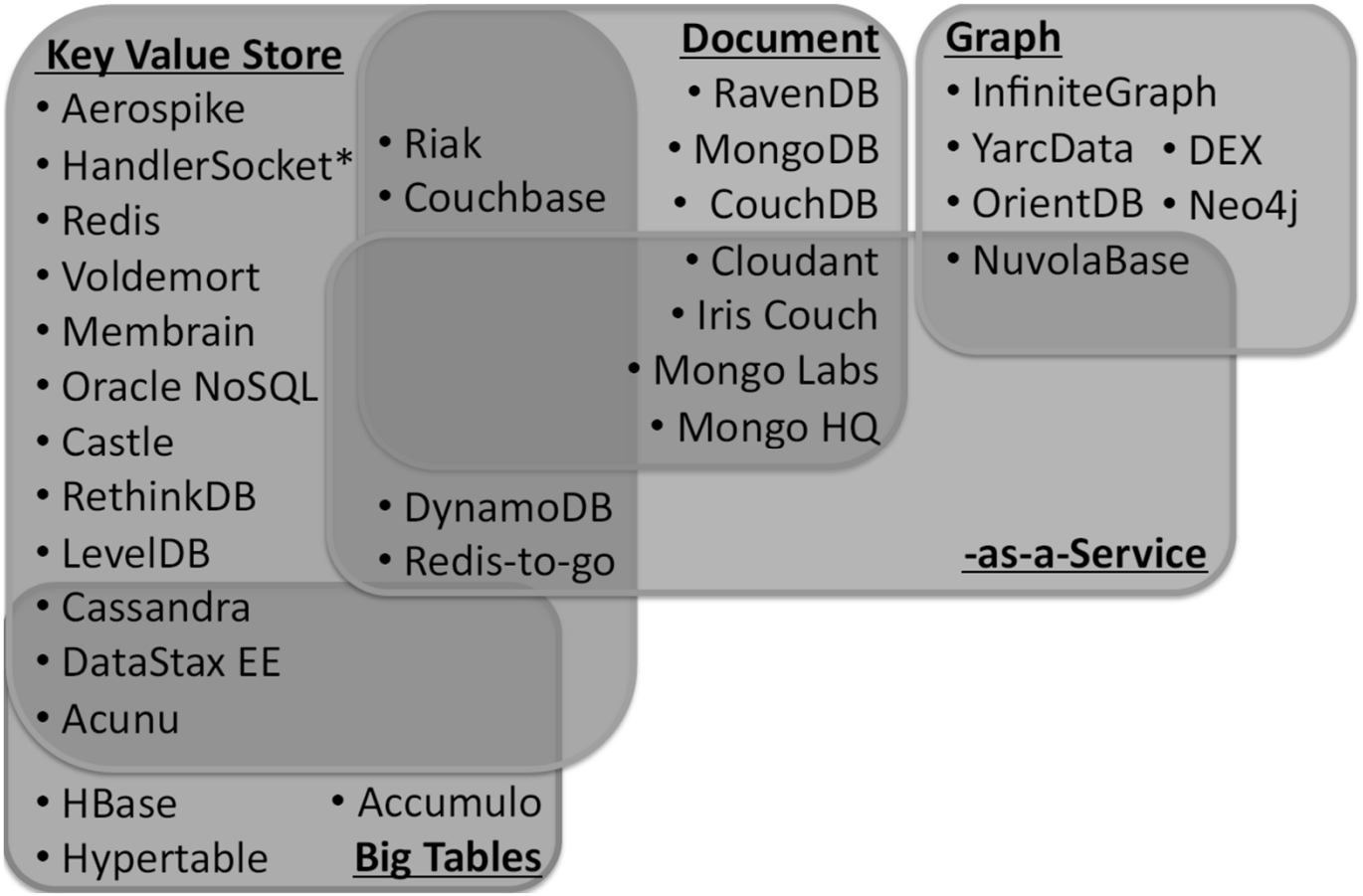
Teorem CAP


Relational
Key-Value
Column-Oriented
Document-Oriented
Neo4j
Consistency
Availability
C
RDBMs (Oracle, MySQL), Aster Data, Green Plum, Vertica
A
mongoDB, Terrastore, Datastore, Hypertable, Hbase,
Redis, Berkeley DB, MemcacheDB, Scalaris
Pick Any Two
Dynamo, Voldemort, Tokyo Cabinet, KAI, Cassandra,
SimpleDB, CouchDB, Riak
P
Partition tolerance

Izvedbe NoSQL

- Prilagojene specifičnim načinom uporabe in vrstam aplikacij
  - **shrambe s ključi (key-value pairs):** BerkleyDB, Keyspace, Dynomite, Voldemort, MemcachedDB in Tokyo Cabinet.
  - **dokumentne zbirke**: MongoDB (Foursquare, SourceForge, Fotopedia, Joomla Ads), CouchDB (CERN, BBC), Redis
  - **zbirke grafov:** Neo4j, AllegroGraph, InfoGrid, Sones, graphDB in FlockDB
  - **stolpično usmerjene zbirke (column-oriented databases):** Hypertable in Cassandra (Digg, Facebook, Twitter, Rackspace), dodatki rel. bazam
  - **objektne shrambe-v zadnjem času dvig popularnosti (podatkovna jezera):** Oracle Coherence, db4o, ObjectStore, GemStone, Polar

NoSQL SUPB


Key Value Store
• Aerospike
• HandlerSocket*
• Redis
• Voldemort
• Membrain
• Oracle NoSQL
• Castle
• RethinkDB
• LevelDB
• Riak
• Couchbase
Document
• RavenDB
• MongoDB
• CouchDB
• Cloudant
• Iris Couch
• Mongo Labs
• Mongo HQ
Graph
• InfiniteGraph
• YarcData
• DEX
• OrientDB
• Neo4j
• NuvolaBase
• DynamoDB
• Redis-to-go
-as-a-Service
• Cassandra
• DataStax EE
• Acunu
• HBase
• Hypertable
• Accumulo
Big Tables

Težave NoSQL

- Ni standardnega povpraševalnega jezika
  - Sami delamo, kar sicel dela SQL prevajalnik (nizkonivojske operacije)
- Zavržemo > 40 let izkušenj relacijskih SUPB
  - Težko smo boljši od prevajalnika/optimizatorja SQL poizvedb
  - Visokonivojski jeziki so boljši po več kriterijih (neodvisnost od podatkov, manj kode)
- Ni shranjenih podprogramov
  - samo ena interakcija med aplikacijo in SUPB (namesto ena za vsak zapis, kot pri NoSQL)
  - premaknemo kodo k podatkom in ne obratno
- Če ACID danes ne potrebujemo, ali lahko zagotovimo to za jutri?
  - premiki podatkov, nekomutativne posodobitve, večzapisna stanja
  - če potrebujemo integritetne omejitve
  - eventuelna konsistentnost $\rightarrow$ zmeda v podatkih

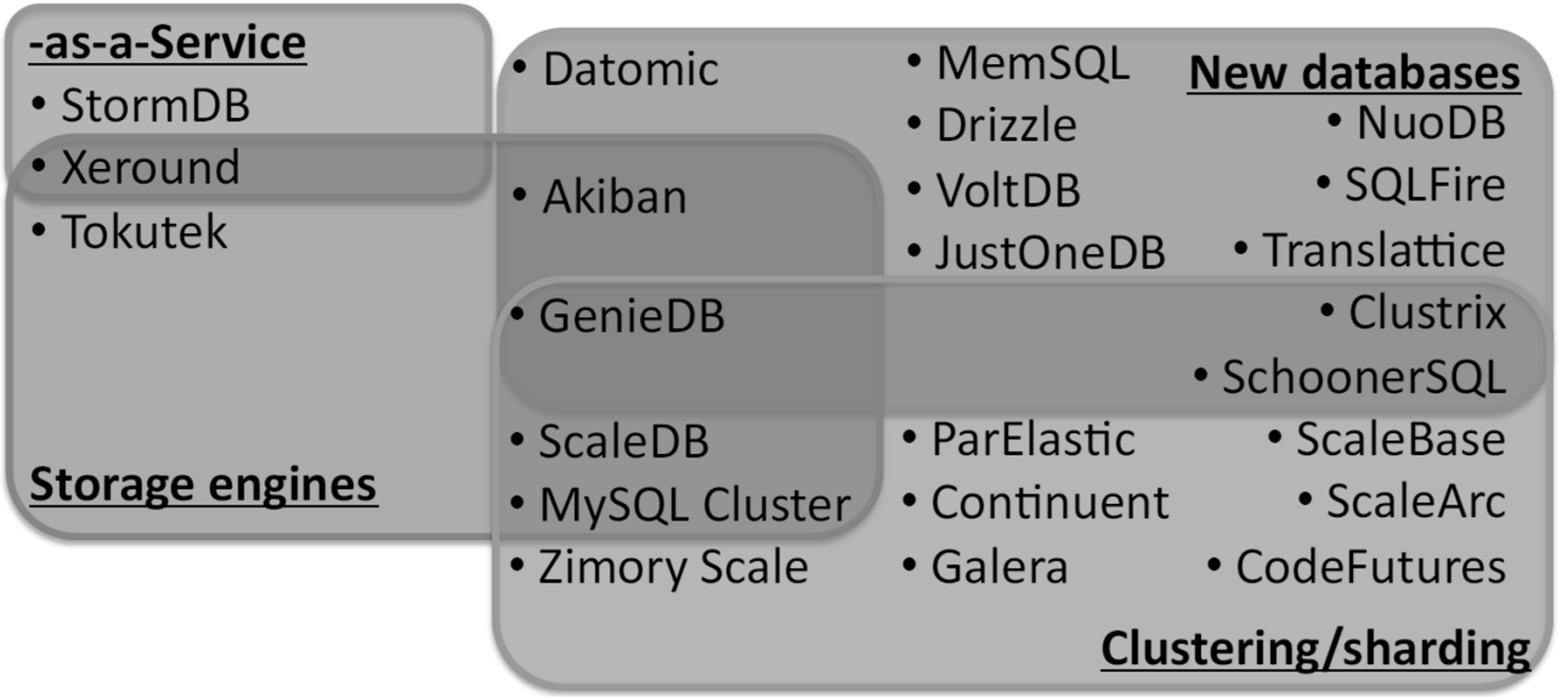
Za kaj oz. kdaj je torej primeren NoSQL

- Netransakcijski sistemi
- Enozapisne, komutativne transakcije
- Neprimerno za sodobne OLTP sisteme
- Ne potrebujemo enovitega povpraševalnega jezika
  - programska koda
  - CQL, UnQL (po zgledu SQL, nestandardno, nekompatibilno)

SQL + NoSQL = NewSQL?

- NoSQL:
  - Nova, moderna zvrst nerelacijskih SUPB
  - Zavračanje pojmov fiksnih shem tabel in stičnih operacij
  - Načrtovan z namenom omogočanja zahtev po masivnem horizontalnem skaliranju
  - Omogoča upravljanje podatkov brez definirane sheme
  - Bye, bye, SQL
- NewSQL
  - Zadnji trend na področju relacijskih SUPB
  - Ohranja SQL in ACID
  - Načrtovan z namenom omogočanja zahtev po masivnem horizontalnem skaliranju ali
  - Tako velik performančni napredek, da horizontalno skaliranje ni več potrebno (VoltDB)

NewSQL SUPB


-as-a-Service
• StormDB
• Xeround
• Tokutek
Storage engines
• Datomic
• Akiban
• GenieDB
• ScaleDB
• MySQL Cluster
• Zimory Scale
• MemSQL
• Drizzle
• VoltDB
• JustOneDB
• ParElastic
• Continuent
• Galera
New databases
• NuoDB
• SQLFire
• Translattice
• Clustrix
• SchoonerSQL
• ScaleBase
• ScaleArc
• CodeFutures
Clustering/sharding

Splošen zaključek

- Zelo specializirana namembnost večine NoSQL SUPB
- Žrtvovanje konsistentnosti za boljše porazdeljeno delovanje
- Vendar v praksi:
  - Lastnost P je nepogrešljiva
  - Konsistentnost se vedno bolj kaže kot nepogrešljiva
- Moderni pristopi:
  - Teorem CAP v resnici prepoveduje le perfektno razpoložljivost (A) in konsistentnost (C) ob prisotnosti razpada/particioniranja omrežja (kar je zelo redek dogodek)
  - Kompromis zagotoviti perfektno A ali C je v resnici dinamičen - NoSQL SUPB samo nekaj časa vztraja v konkretnem (CP ali AP) načinu: CAP -> PACELC (kompromisi)
- Zanimivo branje
  - *http://www.infoq.com/articles/cap-twelve-years-later-how-the-rules-have-changed*
  - Abadi, Daniel J. "Consistency tradeoffs in modern distributed database system design", *IEEE Computer Magazine* 45.2 (2012): 37

An airline reservation system:

- When most of seats are available: it is ok to rely on somewhat out-of-date data, availability is more critical
- When the plane is close to be filled: it needs more accurate data to ensure the plane is not overbooked, consistency is more critical