NoSQL baze podataka

Softversko iženjerstvo i iformacione tehnologije

V semestar 2+2

Istorija sistema za rukovanje podacima

- Poslednjih decenija relacioni sistemi za upravljanje bazama podataka su bili uspešni u rešavanju problema skladištenja i rukovanja podacima.
- Relacioni SUBP su prilagođeni za:
  - *On-line* transakcione obrade (*OLTP*)
  - *On-line* analitičke obrade (*OLAP*)
- Proizvođači kao što su Oracle, Microsoft, IBM su nudili i nude rešenja zasnovana na relacionom modelu podataka i *SQL* jeziku.

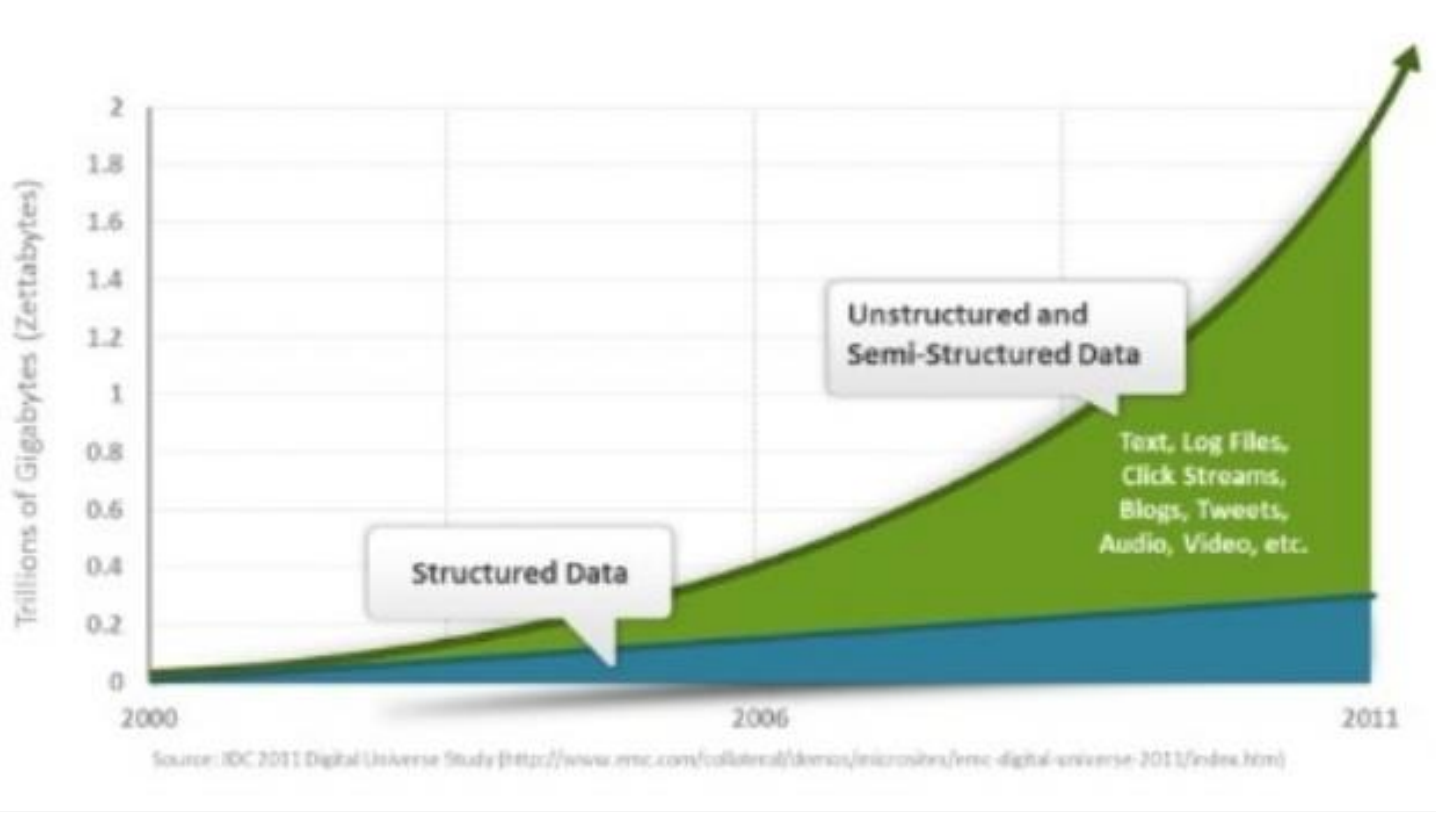
Nešto se promenilo

- Razvoj interneta doveo je do ekspanzije podataka.
- Došlo je do povećanja brzine generisanja podataka.
- Različiti varijeteti podataka.
- Može se reći da se svet informacionih tehnologija značajno promenio poslednjih nekoliko godina.
- Povećanje količine podataka posledica:
  - Povećanog broja korisnika;
  - Razne vrste mobilnih uređaja;
  - Aktivnosti na socijalnim mrežama (*Twitter, Facebook,...).*

Nešto se promenilo (nastavak)

- Prema pojedinim izvorima količina podataka se povećala 40 puta u poslednjih 10 godina.
- 80% podataka koji se generišu je nestruktuirano.
- Stopa rasta nestruktuiranih podataka petnaest puta veća od stope rasta struktuiranih podataka.
- Ova količina podataka ne skalira se uspešno u sistemima s tradicionalim relacionim bazama podataka.
- Pojavljuju se nova rešenja za distribuiranu obradu, ***NoSQL*** baze podataka.

Nešto se promenilo (nastavak)



Odnos struktuiranih i nestruktuiranih podataka

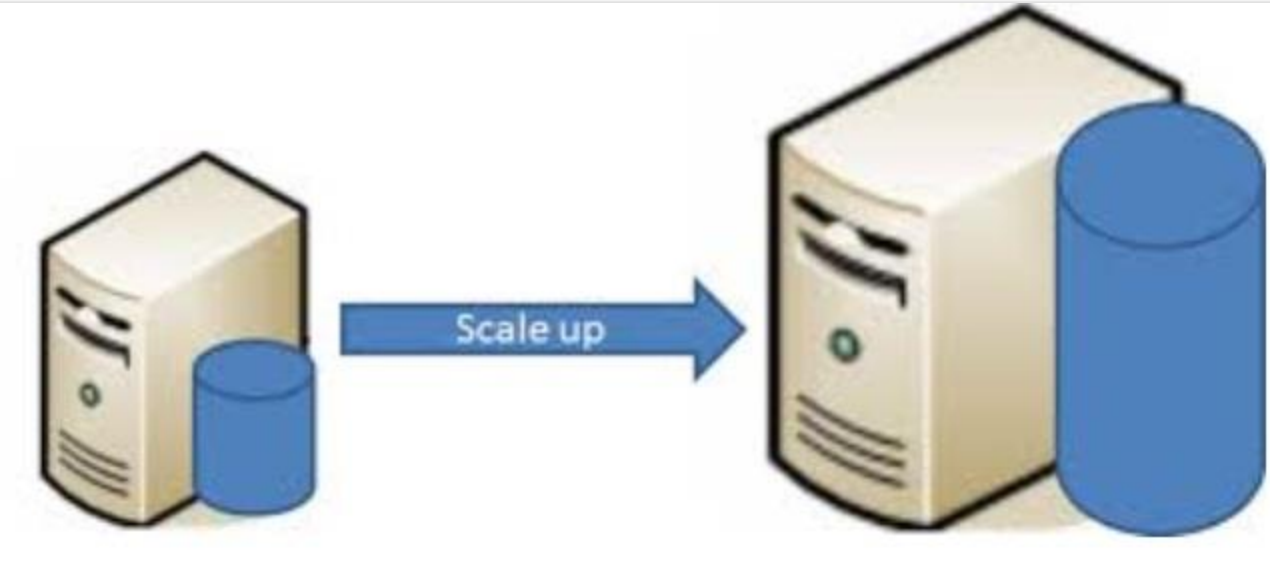
Nešto se promenilo (nastavak)

- Jedna od glavih karakteristika ***NoSQL*** baza podataka je njihova mogućnost da se izvršavaju na klasterima servera umesto na jednom računaru.
- Relacione baze podataka uglavnom projektovane da se izvršavaju centralizovano na jednom računaru.
- ***NoSQL*** baze podataka omogućavaju horizontalno skaliranje kada dođe do povećanja količine podataka i zahteva za dodatnim resursima.

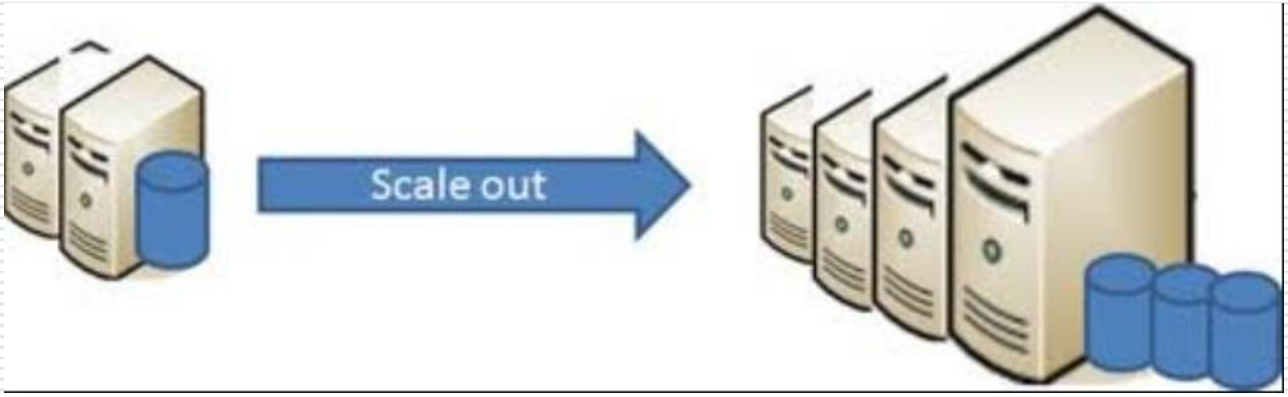
Nešto se promenilo (nastavak)

- U zavsnosti od načina korišćenja koji se planira u sistemu i potreba koje iz toga proističu određuje se model distribucije koji najviše odgovara u konkretnom slučaju.
- Pri korišćenju relacionih baza podataka vrši se vertikalno skaliranje kada dođe do zahteva za povećanjem resursa:
  - Instaliranjem SUBP-a na snažniji hardver koji ima:
    - Više operativne memorije;
    - Brži procesor;
    - Procesor s više jezgara.

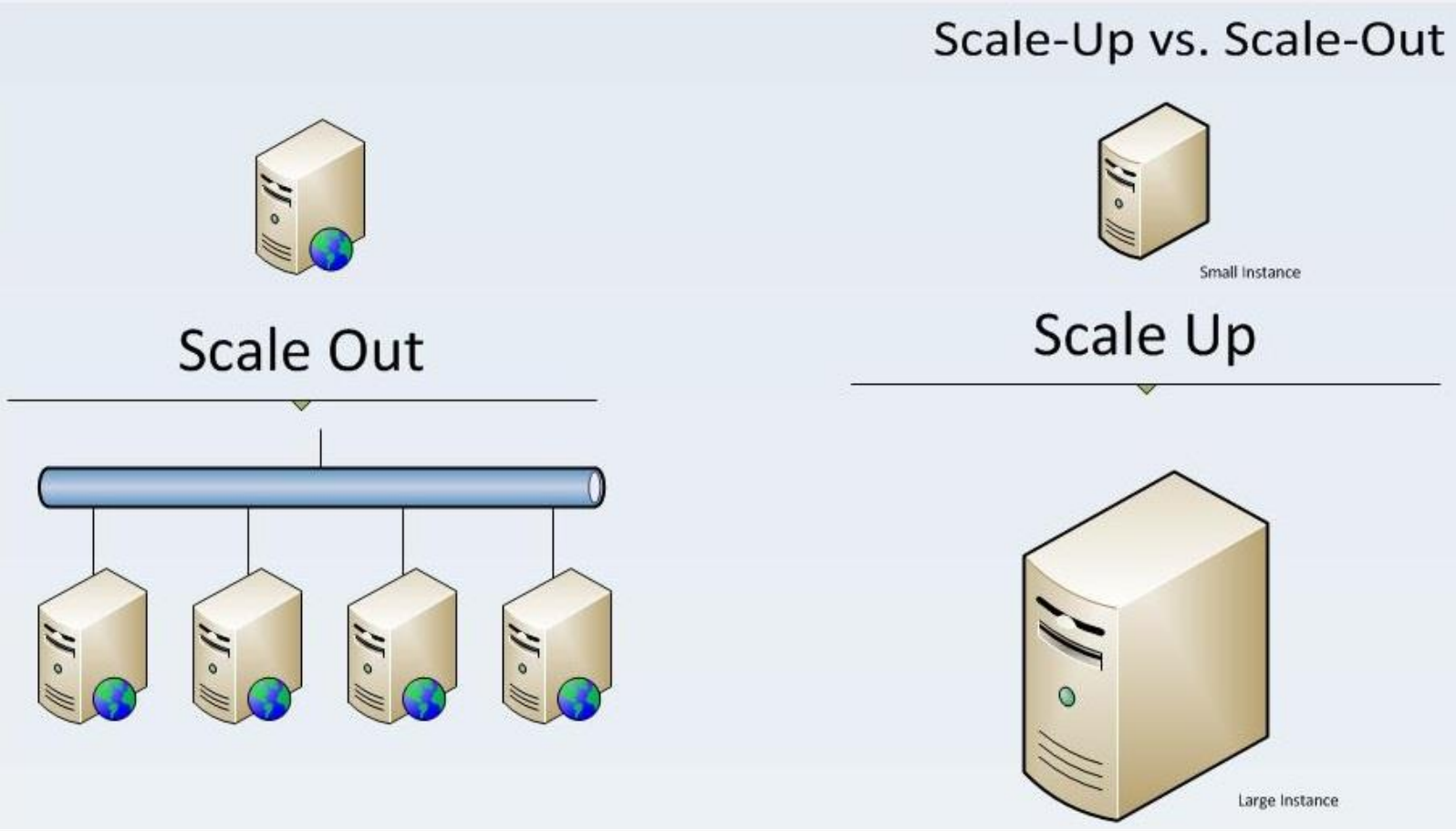
Vertikalno skaliranje



Horizontalno skaliranje



Horizontalno naspram vertikalnog skaliranja



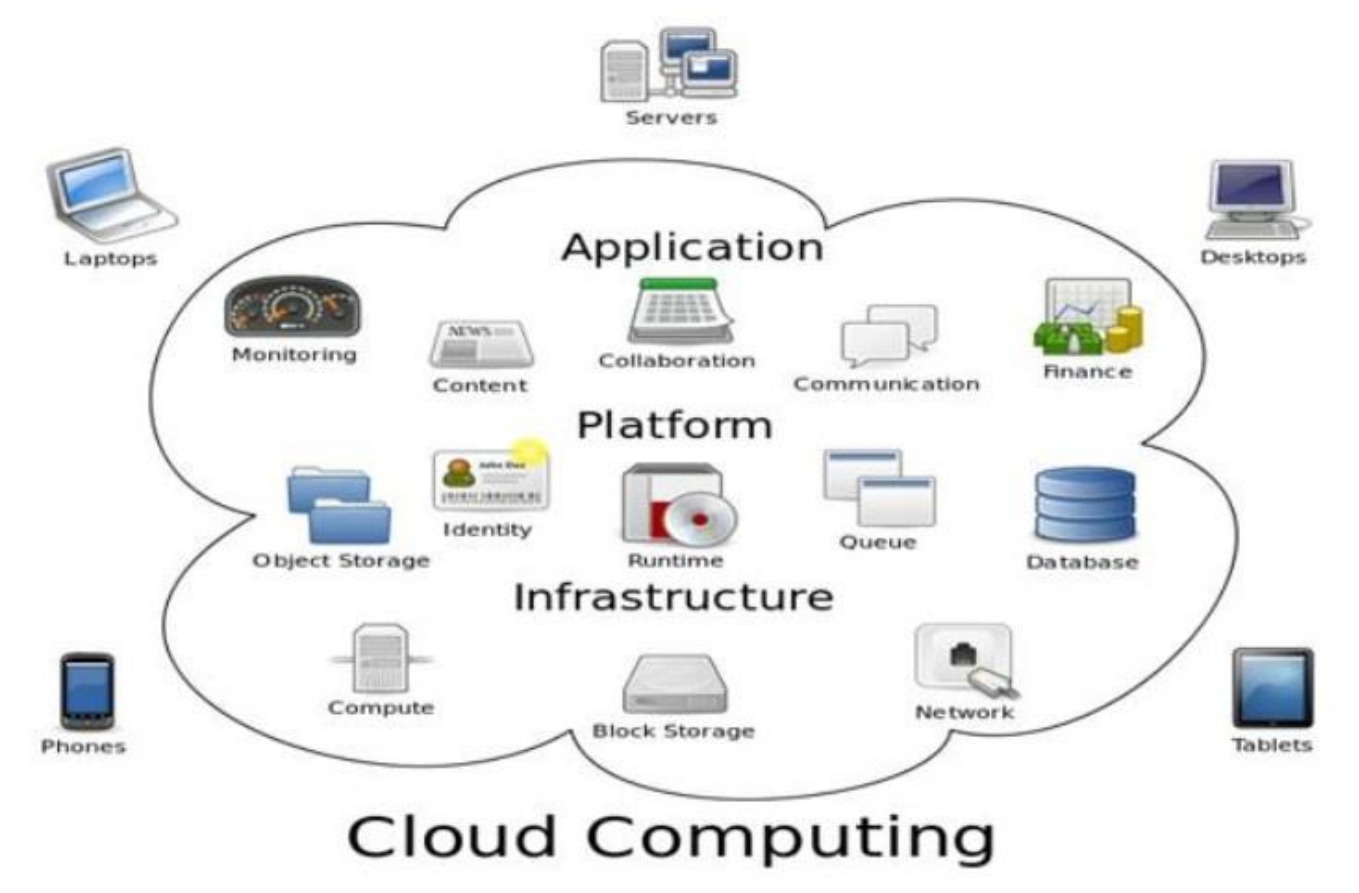
Data Centar

Data Centar

NoSQL <-> Cloud computing

- Pojavljuje se koncept Cloud computinga (računarstva u oblaku).
- Nude se različite vrste servisa u cloud computing rešenjima.
- Korišćenje relacionih SUBP za skaliranje u konfiguracijama gde ima na desetine servera je veoma komplikovan zadatak.
- Bolje je koristiti bazu podataka koja je od samog početka projektovana da relativno jednostavno podrži horizontalno skaliranje i replikaciju, kao što su to ***NoSQL*** baze podataka.

Cloud computing



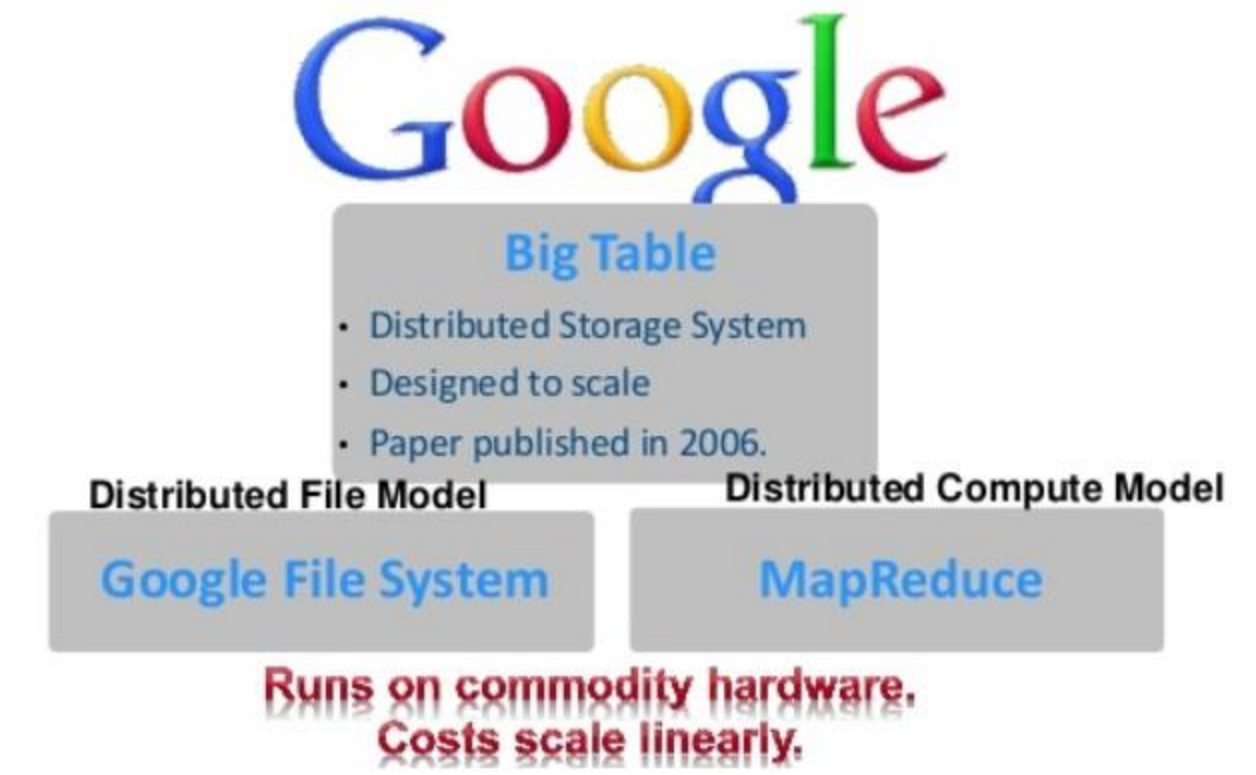
Nastanak *NoSQL* baza podataka

- Nastanak ***NoSQL*** baza podataka vezuje se za pojavu:
  - *Google BigTable*
  - *Amazon's Dynamo*
- Do danas se pojavilo na desetinu proizvoda ***NoSQL*** baza podataka zasnovanih na *Google*-ovim i *Amazon*-ovim konceptima koji se koriste u različitim oblastima primene.

Google BigTable

- *Google* objavio prvi rad o koncepciji i arhitekturi svog proizvoda 2006 godine.
- Zasnovan na *Google File System*-u.
- Dizajniran da se izvršava na stotinama i hiljadama računara uz mogućnost da se relativno lako dodaju novi računari u klastere.
- Sistem automatski počinje da koristi nove resurse bez rekonfigurisanja.
- Tabele optimizirane za Google File System, podelom na segmente oko 200MB veličine.

Google BigTable (nastavak)



Google BigTable (nastavak)

- Koristi se od strane velikog broja Google-ovih aplikacija
  - *Google Maps*
  - *Google Book Search*
  - *Google Earth*
  - *YouTube*
  - *Gmail*

Amazon's Dynamo

- Dynamo je Ključ-vrednost storage system.
- Ima osobine baza podataka i distribuiranih Hash tabela.
- Kreiran da bi podržao zahteve za skalabilnošću Amazon.com web sajta.
- Danas se koristi kod Amazonovih Web servisa kao što je S3.

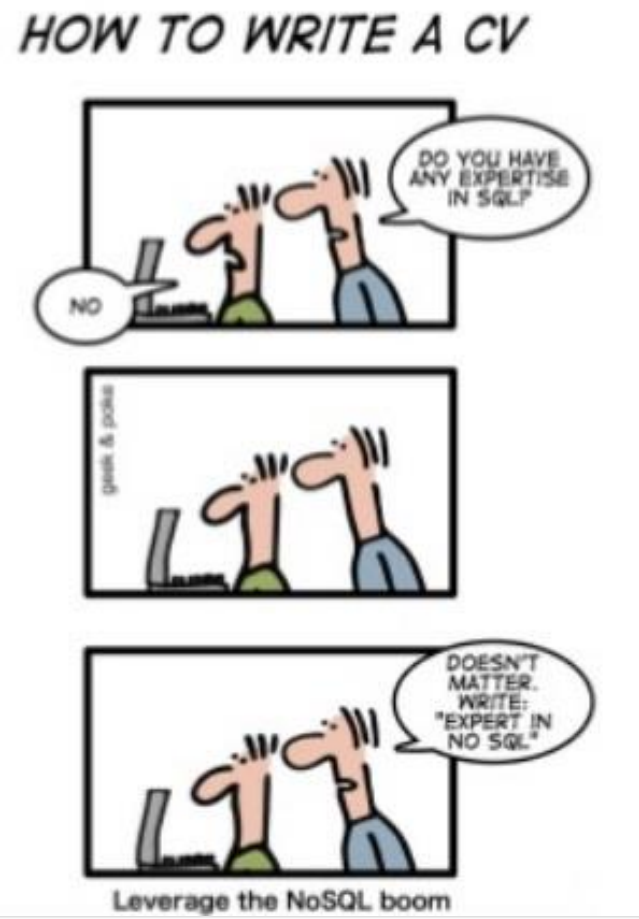
Amazon's Dynamo

- Podržava inkrementalnu skalabilnost.
- Od objavljivanja Amazonovog rada 2007. godine o DynamoDB bazi podataka nekoliko implementacija ***NoSQL*** baza podataka se pojavilo inspirisanih Amazonovim radom:
  - *Aerospike*
  - *Apache Casandra*
  - *Valdemort*
  - *Riak*

NoSQL baze podataka

- Termin ***NoSQL*** nastao 2009. godine
- Ne predstavlja jedinstven model podataka kao što ga imaju svi relacioni sistemi za upravljanje bazama podataka.
- Objedinjuje sve baze podataka koje ne slede principe relacionih baza podataka.
- *No SQL* ili bolje ***Not only SQL***.
- Ne predstavlja jedan proizvod ili tehnologiju.
- Predstavlja klasu proizvoda za rukovanje podacima kojima je zajedničko da su ne-relacioni.

NoSQL baze podataka



Zašto NoSQL?

- ***NoSQL*** baze podataka ne koriste šemu.
- Rukuju velikim količinama podataka.
- Jednostavnije rukovanje nestruktuiranim podacima.
- Jednostavniji interfes ka objektnom modelu.
- Podržava horizotalu skalabilnost.
- Određeni servise je jednostavnije implementirati nego kod relacionih SUBP.
- Bolje podržavaju mnoge “Web 2.0” servise.

Zašto ne NoSQL

- Nepostojanje standarda.
- Nepostojanje standardnog upitnog jezika.
- Dominanto otvorenog koda.
- Nedostatak dokumantacije.
- Nedostatak korisničke podrške.
- Relativno mali broj obučenih programera i administratora baza podataka.

RSUBP naspram *NoSQL*

- Stroga konzistentnost <-> Eventualna konzistentnost.
- Velike količine podataka <-> Ogromne količine podataka.
- Skaliraje moguće <-> Skaliranje jednostavno.
- SQL <-> Map-Reduce.
- Dobra raspoloživost <-> Vrlo velika raspoloživost.

NoSQL - Eventual Consistency

- Zbog distribuiranog modela, svaki server može odgovoriti na svaki upit.
- Serveri međusobno komuniciraju “iza scene”.
- Moguće je da server koji odgovara na upit nema najsvežije podatke.
- Kod nekih primena to ne mora biti problem. Na primer, da li se istog momenta vidi poslednji pristigli *tweet.*

NoSQL - Eventual Consistency

- Kod primena u radu s dobro struktuiraim podacima gde se zahteva stroga konzistentnost *eventual consistency je problem.*
- Postoje primene gde *eventual consistency ne predstavlja problem.*

NoSQL baze podataka

- **ACID** skup osobina koje garantuju pravilno izvršavanje transakcija (definisan krajem 1970-ih godina).
  - **A**tomicity (atomarnost)
  - **C**onsistency (konzistentnost)
  - **I**solation (izolovanost)
  - **D**urability (trajnost)

ACID osobine transakcija

- **A**tomicity (atomarnost) – Transakcija kao logička jedinica posla mora predstavljati atomarni skup aktvnosti koji se izvršava u celosti ili se poništavaju njena dejstva..
  - Podrazumeva skup aktivnosti nad bazom podataka takvih da će u slučaju uspešnog izvršavanja transakcije sve operacije koje su obuhvaćene transakcijom biti uspešno izvršene.
  - U slučaju neuspešnog izvršavanja transakcije neće doći do promene u bazi podataka.

ACID osobine transakcija

- **C**onsistency (kozistentnost) – obezbeđuje da baza podataka konzistentno promeni svoje stanje posle uspešnog potvrđenih (*comited*) promena.
  - Obezbeđuje da se sve operacije uspešno završe ili da se ponište sva dejstva transakcije do prethodnog konzistentnog stanja baze podataka.
  - Praktično obezbeđuje da se baza podataka prevede iz jednog konzistentnog stanja u drugo takođe kozistentno stanje baze podataka.

ACID osobine transakcija

- **I**solation (izolacija) – Transakcija se izvršava izolovano, ne utiče na rad ostalih transakcija nad bazom podataka koje se "paralelno" izvršavaju.
  - Transakcija svoje promene ne čini vidljivim drugim transakcijama pre nego što se završi, odnosno promene potvrdi u bazi podataka.
  - Obezbeđuje uslove kao da se transakcija sama izvršava nad bazom podataka.

ACID osobine transakcija

- **D**urability (trajnost) – obezbeđuje da potvrđeni (*comited*) rezultatai ili efekti transakcije budu trajni.
  - Da se ne mogu izgubiti čak i u slučaju ispada sistema.

CAP teorema

- Problemi sa kojima se susreću sistemi baza podataka dobijaju na složenosti uvođenjem distribuiranog okruženja.
- Pokazuje se da za neke primene nije neophodno primenjivati ACID usaglašenost.
- Primenom ACID kod nekih primena nepotrebno opterećuju sistem naročito ako se radi o Web aplikacijama kojima pristupa veliki broj korisnika.

CAP teorema

- Došlo je do pomeranja u razmišljanjima u smislu da da neki koncepti klasičnih relacionih baza podataka, kojih se ranije strogo pridržavalo, za neke primene nisu neophodni.
- Za ***NoSQL*** baze podataka potrebno je obezbediti:
  - Skalabilnost;
  - Dobro vreme odziva;
  - Mogućnost povećanja kapaciteta baze podataka;
  - Jednostavan programski model;

CAP teorema

- Većina relacionih SUBP inicijalno su bili projektovani da se izvršavaju na samostalnim serverima.
- Takav pristup se koristio u osamdesetim i devedesetim godinama ail se zadržao u projektovanju mnogih današnjih relacionih SUBP.
- S razvojem računarskih mreža dolazi do ubrzanog razvoja teorije i prakse distribuiranih sistema.

CAP teorema

- Danas je posebno aktuelan *Cloud computing* koncept.
  - Resursi se korisnicima isporučuju u vidu usluga preko mreže;
  - Najčešće preko interneta;
  - Sami fizički resursi su nezavisni od lokacije.

CAP teorema

- Termin distribuirani sistemi se orginalno, u početku, odnosio na računarske mreže u kojima su pojedini računari bili prostorno distribuirani.
- Danas se o distribuciji govori na mogo širi način;
- I kada se radi o autonomnim procesima koji se izvršavaju na jednom fizičkom računaru.
- U međusobnoj interakciji su razmenom poruka.

CAP teorema

- Iako ne postoji jedinstvena definicija distribuiranih sistema, može se reći da su to sistemi u kojima:
  - Postoji više računarskih sistema od kojih svaki ima svoju lokalnu memoriju;
  - Međusobna komunikacija se vrši razmenom poruka.

CAP teorema

- U teoriji distribuiranih sistema je razvijaena je CAP-teorema, poznata kao *Brewerova* teorema, nastala 2000. godine.
- Definiše problem sistema koji skladišti deljene podatke.

CAP teorema

- U teoriji distribuiranih sistema je razvijaena je CAP-teorema koja tvrdi da je za distribuirane sisteme nemoguće da istovremeno ispune sva tri od sledećih uslova:
  - Konzistentnost *(**C**osistency)* – svi čnorovi vide iste podatke u isto vreme;
  - Raspoloživost (***A**vailability)* – garancija da svaki zahtev dobija odgovor da li je uspešno izvršen ili je došlo do greške;
  - Tolerancija na razdvojenost (***P**artition tolerance)* – sistem nastavlja da radi uprkos tome što je došlo do gubljenja poruke ili je došlo do greške (ispada) dela sistema

CAP teorema

- Konzistentnost (***C****onsistency*)
  - Podrazumeva da sve akcije koje se vrše nad bazom podataka, bilo da su uspešno ili neuspešno izvršene, ostavljaju bazu podataka u konzistentnom stanju.
  - Takođe, svaka operacija čitanja iz baze podataka kao rezultat ima najnoviju verziju podataka.

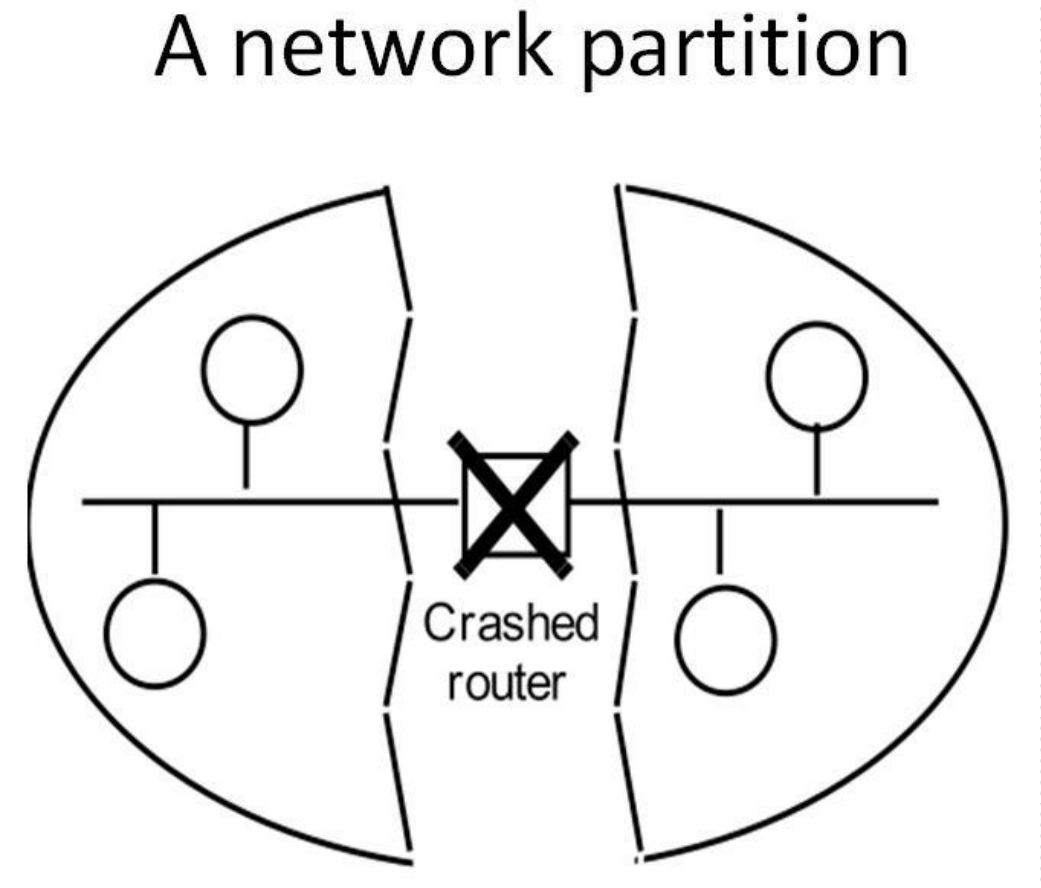
CAP teorema

- Dostupnost odnosno raspoloživost (***Availability***)
  - Podrazumeva prihvatljiv vremenski odziv sistema baza podataka.
  - Svakoj operaciji čitanja je dostupna bar jedna kopija traženog podatka bez obzira na celokupnu funkcionalnost sistema.
  - Dostupnost se povećava replikacijom i povećanjem broja serverskih sistema.

CAP teorema

- Tolerancija razdvojenosti (***Partition tolerance***) :
  - Sistem je u funkcionalnom stanju bez obzira na skup otkaza i
  - Situacije kada pojedini čvorovi u sistemu nisu u mogućnosti da međusobno komuniciraju i vrše razmenu podataka

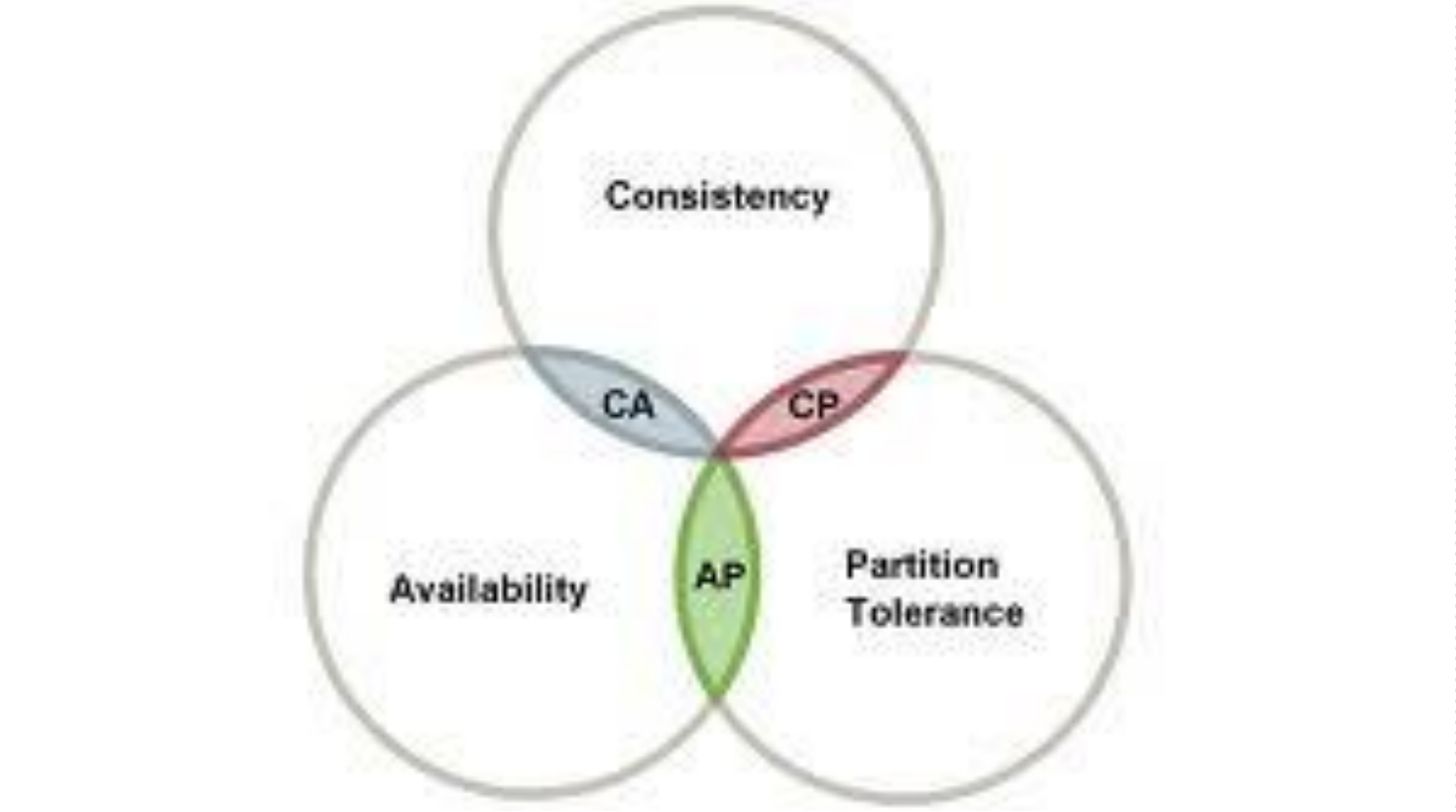
Razdvojenost (Partition)



CAP teorema

- Aktuelne *NoSQL* baze podataka baziraju se na kombinaciji prethodno opisana tri uslova tako da postoje sledeći izbori kombinacija:
  - **CA** (Konzistentnost/Dostupnost) – sistem ne podržava razdvojenost;
  - **CP** (Konzistentnost/Tolerancija razdvojenosti) – sistem ne garantuje postojanost svih podataka i pojedini podaci mogu biti nedostupni;
  - **AP** (Dostupnost/Tolerancija razdvojenosti) – sistem je dostupan u bilo kom trenutku posle podele na više čvorova ali konzistentnost podataka nije obezbeđena.

CAP teorema



CAP teorema i kombinacije

CAP teorema – relacioni SUBP

- Postavlja se pitanje kako se relacioni SUBP projektovani da se izvršavaju na samostalnim serverima uklapaju u koncepte distribuirane obrade.
- P iz CAP teoreme je bez značaja, jer je baza podataka na jednom serveru.
- Server je u funkciji i radi ili ne radi.
- Ne može se govoriti o situacijama da je server delimično raspoloživ.
- Kod relacionih SUBP pažnja se fokusira na konzistentnost.

CAP teorema – *NoSQL* baze podataka

- Kod pojedinih ***NoSQL*** baza podataka pažnja se fokusira na *C* i *A* deo *CAP* teoreme.
- Pri projektovanju ***NoSQL*** baza u startu se pošlo od činjenice da će se izvršavati na:
  - desetinama čvorova,
  - stotinama čvorova,
  - pa čak i na hiljadama čvorova nekog *Data Center*-a
- Tolerancija na otkaze postiže se tako što se baza podataka kompletno replicira na više *data center*-a.

CAP teorema – ***NoSQL*** baze podataka

- Prednost sistema baza podataka koji relativno jednostavno podržavaju replikaciju u odnosu na tradicionalne relacione sisteme je:
  - Zadatak koji treba obaviti raspoređuje se na sve računare u sistemu.
  - Postiže se veoma dobro vreme odziva čak i kod velikog broja čitanja i pisanja u bazu podataka.

CAP teorema – *NoSQL* baze podataka

- Kod nekih ***NoSQL*** baza podataka posebna pažnja se posvećuje A i P delu CAP teoreme
- Projektuju se tako da integrišu više *data center*-a.
- Kod ***NoSQL*** baza podataka prema CAP teoremi nemoguće je strogo poštovanje konzistentnosti.
- "slaba" konzistentnost je nezamisliv pojam kad su u pitanju relacioni SUBP.
- Kod ***NoSQL*** baza podataka se implementira takozvana eventualna (ili konvergentna) konzistentnost – ***evetual consistency.***

CAP teorema – *NoSQL* baze podataka

- Promene u bazi se ne repliciraju istovremeno – *replicated eventualy.*
- Pojedini čvorovi ili grupe čvorova nemaju u svakom trenutku poslednje stanje podataka.
- Kao i kod ***NoSQL*** baza podataka koje se fokusiraju na C i A deo CAP teoreme i kod ***NoSQL*** baza kod kojih je težište na A i P delu teoreme u fokus se stavlja:
  - Brz odziv i velika propusnost de bi se korisnicima obezbedili;
  - Što brži i što bogatiji odgovori na njihove upite.

CAP teorema – ***NoSQL*** baze podataka

- Ponekad se ide u pravcu popuštanja u konzistentnosti organizacije podataka;
- U korist skalabilnosti i postizanja boljeg vremena odziva celog sistema.
- Odbacivanje konzistentnosti:
  - Obezbeđuje se raspoloživost i tolerancija razdvojenosti;
  - Ne garantuje se čitanje poslednje verzije podataka u slučaju razdvojenosti;
  - U suprotnosti s ACID osobinama.

BASE skup osobina

- Umesto ***ACID*** osobina definiše se ***BASE*** skup osobina;
- ***B****asically* ***A****vailable,* ***S****oft state,* ***E****ventual Consistent.*
  - Definisao ih Eric Brewer tvorac CAP teoreme;
  - *BASE* skup osobina predstavlja suprotnost *ACID* skupu osobina u cilju postizanja kompromisa eliminacijom konzistentnosti;
  - Obezbeđuje se dostupnost podataka i tolerancija razdvojenosti.

BASE skup osobina

- ***Basically Available*** – Suštinski raspoloživ – sistem obezbeđuje dostupnost većini podataka.
  - Većina podataka je dostupna veći deo vremena
- ***Soft state*** - Nekonzistentno stanje - baza podataka ne mora biti konzistentna u svakom trenutku.
  - Stanje sistema menja se s vremenom, čak i kada nema unosa podataka.
- ***Eventually consistent*** – Konvergentna konzistentnost Sistem ne garantuje da će svi čvorovi sistema sadržati iste kopije podataka.
  - Teži se vremenskoj tački kada će svi čvorovi sadržati konzistentne podatke.

NoSQL baze podataka

- Sledeći način na koji se ***NoSQL*** baze podataka udaljavaju od tradicionalnog načina rada s podacima je napuštanje relacionog modela.
- Neki podaci prirodno ne odgovaraju korišćenju relacionog modela podataka iz različitih razloga. Zbog toga što:
  - Podaci često menjaju formu;
  - Dužinu podataka, ili
  - Zbog toga što su potpuno nestruktuirani.

NoSQL baze podataka

- Može se postaviti pitanje - **Da li je došao kraj relacionih baza podataka?**
- U velikoj oblasti primene rukovanja podacima u pogledu organizacije, skladištenja i pretraživanja ništa se dramatično nije promenilo.
- Relacione baze podataka i njihova *ACID* usaglašenost je jedini pravi i pouzdan odgovor na pitanje kako organizovati podatke i kako njima rukovati u tim oblastima primene.

Da li je došao kraj relacionih baza podataka?

- Gotovo je smešno govoriti o eventualnoj konzistentnosti podataka u nekom bankarskom informacionom sistemu.
- Veliki broj drugih oblasti primene gde se posebno insistira na konzistentnosti i integritetu podataka.
- S druge strane u nekim oblastima primene potrebe i zahtevi skladištenja i pretraživanja podataka u poslednjim godinama su se do te mere promenili da relacione baze podataka nisu mogle dati zadovoljavajuće odgovore.

Da li je došao kraj relacionih baza podataka?

- Zbog toga se i krenulo u pravcu rešavanja problema organizacije, skladištenja i pretraživanja ogromnih količina podataka;
- Kojima pristupa veliki broj korisnka i u kojima je dinamičan i obiman dnevni priliv podataka.
- Kao što je eksplodirala produkcija i potrošnja podataka isto tako su se veoma brzo počele pojavljivati različita ***NoSQL*** rešenja kao odgovor na taj problem.

Da li je došao kraj relacionih baza podataka?

- U tom smislu današnje stanje je takvo da nije realno govoriti o opciji ili/ili kad su u pitanju relacione i ***NoSQL*** baze podataka,
- Nnego o opciji i/i.
- Za neke oblasti primene su relacione baze podataka nezamenljive i kako stvari stoje još dugo će to biti.
- S druge strane postoji oblasti primene u kojima su ***NoSQL*** baze podataka u ovom trenutku jedino moguće rešenje.

Da li je došao kraj relacionih baza podataka?

- Prvi i najvažnije razlog zbog kojeg su relacioni SUBP i dalje ne samo aktuelni nego i nezamenljivi je neophodnost primene *ACID* transakcija u mnogim oblastima primene.
- Baze podataka koje se koriste u bankarskim, berzanskim i drugim poslovnim sistemima uvek moraju davati korektne podatke.
- Gde se radi o novcu aproksimacije jednostavno nisu dozvoljene.

Da li je došao kraj relacionih baza podataka?

- Za korisnike Twitera nije od presudne važnosti da li će do pojavljivanja poslednja twitovane poruke proći par minuta.
- To se ne može reći za Sisteme za prodaju karata ili Knjgovodstvene baze podataka.

Da li je došao kraj relacionih baza podataka?

- Sledeća stvar koja ide u korist relacionih baza podataka je njihovo korišćenje *SQL*-a.
- To je standardizovan jezik i ako je aplikaciju potrebno prebaciti sa jedne na drugu bazu podataka potrebno je vršiti minimalne promene da bi ona radila s novom bazom.
- S druge strane standardizovani ***NoSQL*** upitni jezik verovatno neće nikad ni postojati zbovg velikih razlika među pojedinim ***NoSQL*** bazama podataka.
- Moglo bi se reći da je jedina zajednička stvar za sve ***NoSQL*** baze podataka da ne postoji ništa ili malo toga zajedničkog.

Da li je došao kraj relacionih baza podataka?

- Svaka ***NoSQL*** baza podataka ima svoj skup API, biblioteke i najpoželjiniji programski jezik za interakciju s podacima u bazi.
- Kod relacionih baza podataka je sasvim nebitno koji programski jezik se koristi, uvek je relativno jednostavno doći do podataka iz baze u bilo kom formatu koji je neophodan.
- S druge strane kad su u pitanju ***NoSQL*** baze podataka moguća je situacija da izbor baze uslovljava izbor programskog jezika koji će se koristiti za rezvoj cele aplikacije.

Da li je došao kraj relacionih baza podataka?

- Sledeća stvar koja ide u prilog relacionih SUBP-uova je formalizam relacionog modela na kojem su zasnovani.
- Istorijat relacionih baza počinje rezultatima istraživanja E. F. Cod-a objavljenim sedamdesetih godina.
- Od tog perioda relacioni model je proširen i poboljšan.
- Osim toga relacioni model je popularan i zbog toga što je veoma pogodan način za organizaciju podataka.

Da li je došao kraj relacionih baza podataka?

- Relacioni model savršeno odgovara velikom broju realnih potreba skladištenja podata, a kad je normalizacija izvršena na odgovarajući način, tada je i brz i efikasan.
- ***NoSQL*** baze podataka imaju svoju oblast primene gde relacioni sistemi pokazuju odrđene slabosti, pa su i nastale kao rešenje problema u tom području.

Razlozi za korišćenje ***NoSQL*** baza podataka

- Paralelno procesiranje velike količine podataka u distribuiranim sistemima.
- Pretraga podataka u sistemima s velikom količinom podataka.
- Skladištenje velike količine podataka (struktuiranih, delimično struktuiranih, nestruktuiranih).
- Mogućnost horizontalnog skaliranja.
- Relativno jednostavna distribucija podataka.

Modeli distribucije

- Kao što je renije već bilo pomenuto, jedana od glavnih karakteristika ***NoSQL*** baza podataka je njihova mogućnost da se izvršavaju na klasterima servera umesto na samo jednom računara;
- Za razliku od relacionih baza podataka koje su uglavnom projektovane da se izvršavaju centralizovano na jednom serveru.
- Na taj način se otvara mogućnost skaliranja sistema kada dođe do povećanja količine podataka i zahteva za dodatnim resursima.
- U zavisnosti od načina korišćenja koje planiramo u našem sistemu i potreba koje iz toga proističu odrediće se koji model distribucije najviše odgovara u konkretnom slučaju.

Modeli distribucije

- Postoje dve osnovne tehnike distribucije,
  - *sharding* i
  - *replication*
- iz kojih se mogže izvesti nekoliko različitih modela distribucije podaaka.

Modeli distribucije

- *Sharding* tehnikom distribucije se podaci raspoređuju na različite čvorove, tako da ukpne podatke baze podataka čini unija podataka raspoređenih po različitim čvorovima.
- S druge strane, *replication* tehnikom se po čvorovima raspoređuju kopije kompletne baze podataka, tako da se isti podaci nalaze na više različitih računarskih čvorova.
- U nastavku će biti reči o različitim modelima distribucije podataka:
  - *single server*,
  - *imaster-slave* replikaciji,
  - *sharding* i
  - *peer-to-peer* replikaciji.

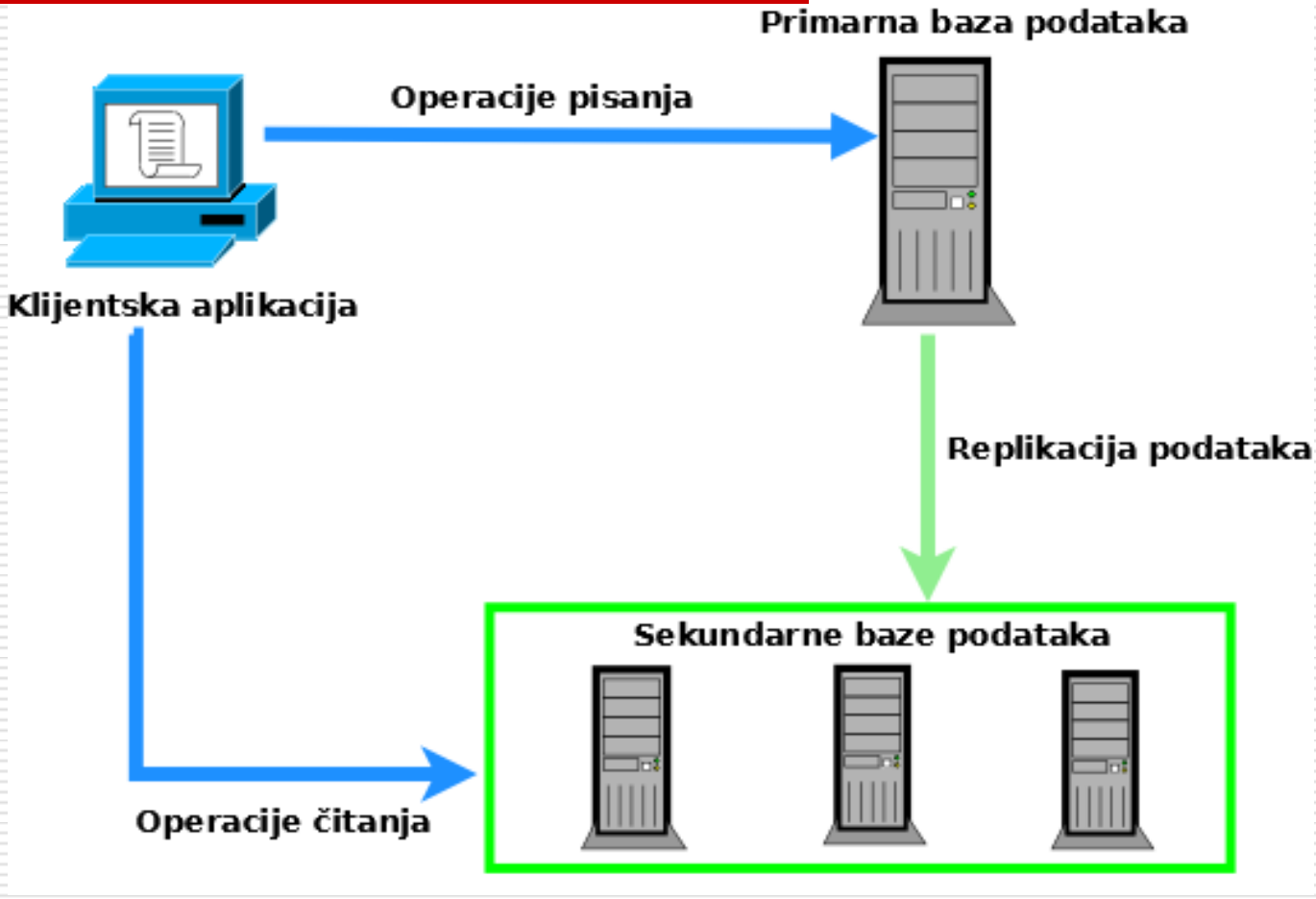
Modeli distribucije – *Single Server*

- Najjednostavniji pristup je uopšte ne vrtšiti distribuciju podataka, odnosno
  - da se sistem za upravljanje bazom podataka izvršava na jednom serveru i
  - da se na njega smesti kompletna baza podataka.
- Za mnoge oblasti primene gde se ne zahteva nikakva distribucija podataka ovakav pristup je odgovarajući i pojednostavljuje složenost sistema jer svim ažuriranjima i pretraživanjima rukuje isti server.

Modeli distribucije – *Single Server*

- Bez obzira što su *NoSQL* baze uglavnom projektovane da se izvršavaju u ambijentu računarskih klastera;
  - ponekad je opravdano *NoSQL* bazu koristiti na jednom serveru ako takav model *NoSQL* skladišta odgovara aplikaciji.
- U nastavku će biti prikazano nekoliko načina distrubucije, međutim;
  - ako ne postoje posebni razlozi za distribuciju, kad god je moguće treba izabrati arhitekturu s jednim serverom.

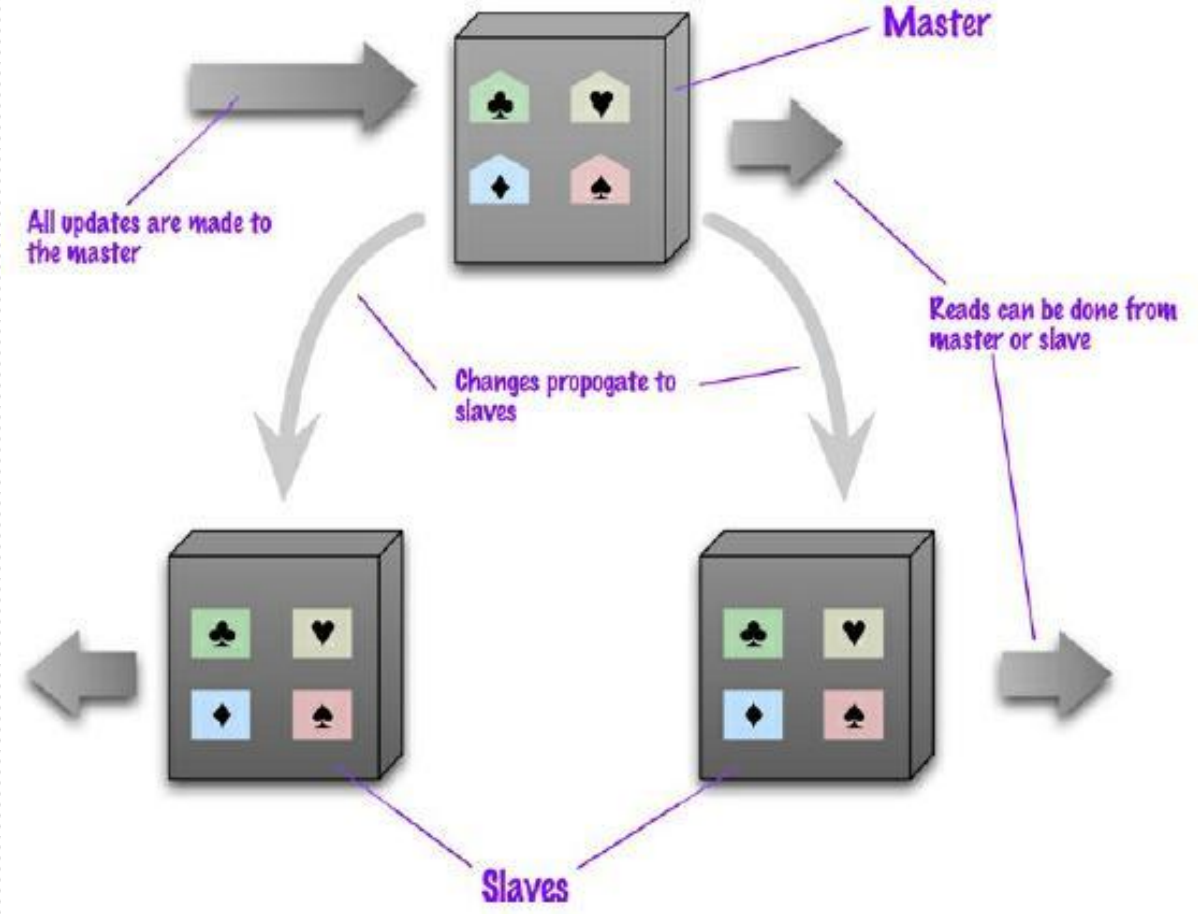
Modeli distribucije - Replikacija



Modeli distribucije – *Master-Slave replikacija*

- Kod *Master-Slave* modela distribucije baza podataka se replicira na više računarskih čvorova u klasteru, pri čemu je:
  - jedan od čvorova vodeći, *Master* čvor,
  - a ostali čvorovi su prateći ili *Slave* čvorovi.
- Za sve operacije ažuriranja podataka je odgovoran jedino *master* čvor, dok se
- pretraživanje može vršiti pristupom bilo *master* čvoru ili *slave* čvorovima.

Modeli distribucije – *Master-Slave replikacija*



Modeli distribucije – *Master-Slave replikacija*

- Proces odgovoran za replikaciju vrši sinhronizaciju podataka na *slave* čvorovima u odnosu na stanje podataka *master* čvora i
- Na taj način se obezbeđuje da stanje baze podataka *slave* čvorova u svakom trenutku odgovara stanju baze podataka *master* čvora.
- Na prethodnom slajdu je pojednostavljeno prikazan ovakav model distribucije.
- Ovakav model distribucije pogodan je za primenu u sistemima kod kojih je dominanto pretraživanje podataka u odnosu na ažuriranje.

Modeli distribucije – *Master-Slave replikacija*

- Prednosti koje donosi ovakav model distribucije mogu se svrstati u dve grupe.
  - Sistem se može horizontalno skalirati, kako bi mogao odgovoriti na veći broj zahteva za pretraživanjem podataka;
  - Dodavanjem *slave* čvorova i podešavanjem da se pretraživanje podataka usmeri ka *slave* čvorovima.
  - Osim toga, ovakavim modelom distribucije povećava se dostupnost podataka koji se pretražuju.

Modeli distribucije – *Master-Slave replikacija*

- U slučaju otkaza *master* čvora, *slave* čvorovi i dalje mogu prihvatiti zahteve za pretraživanjem podataka čime se;
- Povećava raspoloživost sistema, posebno sistema kod kojih je dominatno pretraživanje.
- S druge strane, postojanje kopije baze podataka na *slave* čvorovima omogućuje da neki od *slave* čvorova može veoma brzo preuzeti ulogu *master* čvora i obezbediti nesmetan nastavak rada sistema dok se ne izvrši oporavak *master* čvora.

Modeli distribucije – *Master-Slave replikacija*

- Pored prdnosti koje donosi, *Master-Slave* model distribucije pokazuje i određene nedostatke;
  - posebno u primeni kod sistema u kojima postoji veliki broj zahteva za ažuriranjem podataka.
  - Obzirom da se kod ovakvog modela distribucije sva ažuriranja vrše preko *master* čvora on može postati usko grlo u sistemu u slučaju intenzivnog ažuriranja podataka.
  - Iako se *master* čvor može rasteretiti preusmeravanjem upita ka *slave* čvorovima, tako da obrađuje samo zahteve za ažuriranjem, ipak se *master-slave* model distribucije ne predlaže za sisteme u kojima se vrši veliki broj ažuriranja podataka u jedinici vremena.

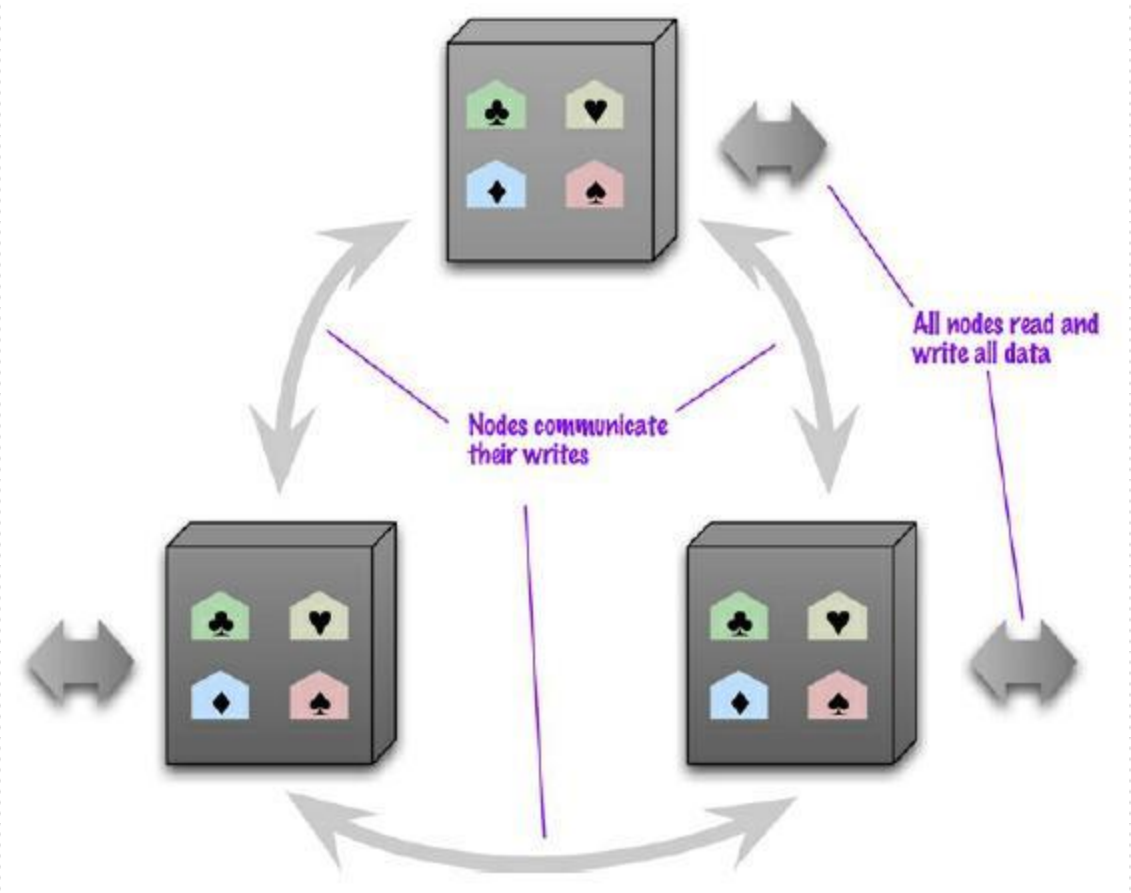
Modeli distribucije – *Peer-to-peer replikacija*

- *Master-Slave* replikacija pomaže kod skaliranja operacija čitanja baze podataka;
- Međutim ne pomaže u skaliranju operacija upisa, odnosno ažuriranja baze podataka.
- Njome se postiže otpornost na ispad *Slave* servera ali ne i *Master* servera baze podataka.
- Kod Master-Slave replikacije postoji tačka ranjivosti sistema (single point of failure) jer je Master server usko grlo.
- Peer-to-peer replikacijom se izbegava mogućnost ovog problema jer ne postoji Master server.

Modeli distribucije – *Peer-to-peer replikacija*

- Sve replikacije su ravnopravne i sve mogu prihvatiti operacije ažuriranja baze podataka.
- Ispadom jednog od servera ne ugrožava se pristup bazi podataka u celini.
- Sa *Peer-to-peer* replikacijom i u slučaju ispada nekog od čvorova ne ostaje se bez pristupa bazi podataka.
- Osim toga lako se mogu dodati dodatni čvorovi da bi se unapredile performanse

Modeli distribucije – *Peer-to-peer replikacija*



Modeli distribucije – *Peer-to-peer replikacija*

- Problem kod Peer-to-peer replikacije je obezbeđivanje konzistentnosti podataka.
- Kada je moguće ažuriranje podataka u bazi na dve različita mesta postoji rizik da dva korisnika ažuriraju iste podatke u isto vreme i da nastane konflikt upisa.
- Nekonzistentnost kod čitanja baze podataka je takođe problem, međutim ona je prolazna, dok je nekonzistentnost upisa trajna.
- Međutim, postoje načini da se problem nekonzistencije upisa eliminišu ili bar smanje na najmanju moguću meru.

Modeli distribucije – *Peer-to-peer replikacija*

- Potrebno je kod ažuriranja baze podataka obezbediti da prilikom ažuriranja podataka replikacije rade koordinirano kako bi se izbegao konflikt upisa.
- Na takav ačin se može postići isti rezultat kao u slučaju upisa kod *Master-Slave* replikacije, ali se povećava mrežni saobraćaj.
- S druge strane postoje situacije kada je moguće izboriti se s nekonzistentnošću upisa na taj način što će se vršiti spajanje nekonzistentnih upisa i na taj način obezbediti prednosti u performansama koje donosi *Peer-to-peer* replikacija s mogućnošću ažuriranja baze na bilo kojoj replikaciji.

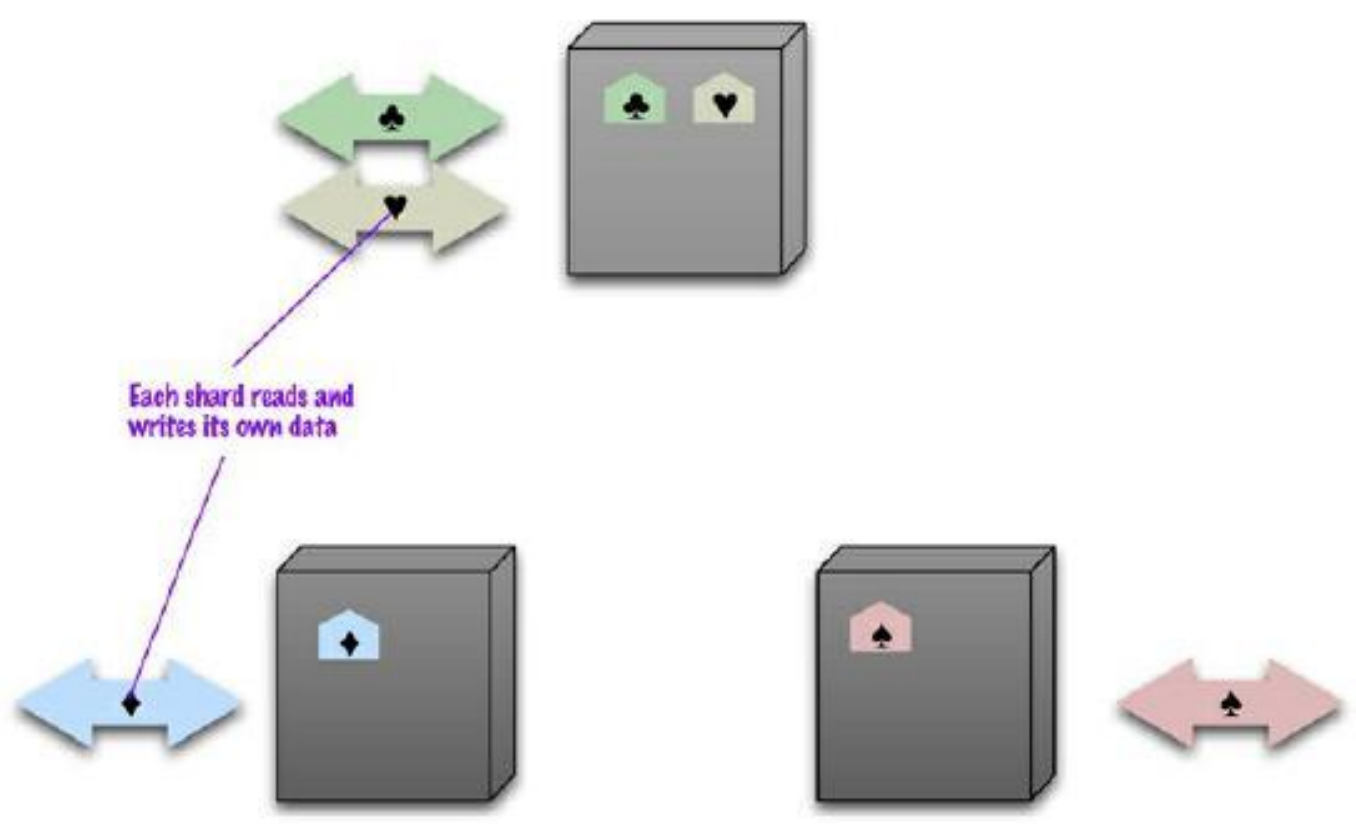
Modeli distribucije – *Sharding*

- Ponekad se velika konkurencija nad bazom podataka događa zbog toga što različiti korisnici baze podataka pristupaju potpuno različitim skupovima podataka.
- U takvim situacijama može se vršiti podrška horizontalnom skaliranju na način da se različiti skupovi podataka smeste na različite servere u sistemu.
- Ovakva tehnika distribucije podataka se naziva *Sharding*.

Modeli distribucije – *Sharding*

- U idealnoj situaciji imamo različite korisnike koji svoje potrebe za podacima iz baze podataka zadovoljavaju pristupajući različitim serverima.
- Jedan korisnik pristupa samo jednom serveru, tako da se na taj način postiže poboljšanje performansi i vreme odziva servera.
- Pristup bazi podataka se može izbalansirati, tako da se ravnomerno rasporedi po serverima.
- Na primer, ako postoji deset servera u idealnoj situaciji svaki bi obrađivao 10% ukupnih zahteva pristupa bazi podataka.

Modeli distribucije – *Sharding*



Modeli distribucije – *Sharding*

- Međutim, idealna situacija je veoma retka, tako da je potrebno pokušati obezbediti da se podaci kojima se zajedno pristupa budu raspoređeni na isti server i to na server koji obezbeđuje najbolji pristup podacima.
- Prvi deo ovog pitanja je kako grupisati podatke tako da jedan korisnik uglavnom dobija svoje podatke sa jednog servera.
- U ovoj situaciji agregiranje podataka postaje veoma korisno.
- Ceo smisao agregacija je da ih dizajniramo tako da kombinuju podatke kojima se cesto pristupa zajedno;
  - tako da se agregacije izdvajaju kao očigledane jedinice distribucije.

Modeli distribucije – *Sharding*

- Sto se tice rasporedjivanja po čvorovima, postoji vise faktora koji mogu da pomognu u poboljsavanju preformansi.
  - Ako znamo odakle se, u fizickom smislu, najcesce pristupa odredjenim agregacijama podataka, njih mozemo postaviti blizu lokacije iz koje se pristupa.
  - Ako imamo porudžbine od nekoga iz Bostona, te podatke mozemo rasporediti u data centar za istocni deo USA.

Modeli distribucije – *Sharding*

- U proslosti, sharding se uglavnom radio kao deo logike aplikacije.
- Mozemo sve korisnike sa prezimenima na slova izmedju A i D smestiti na jedan shard, a E i G na drugi.
- Ovo komplikuje model programiranja zato sto kod aplikacije mora obezbediti da su zahtevi rasporedjeni na vise *shard*-ova.

Modeli distribucije – *Sharding*

- Pored toga, rebalansiranje shardova znaci menjanje koda aplikacije i pomeranje podataka.
- Mnoge NoSQL baze podataka nude ***auto-sharding***, gde baza podataka preuzima odgovornost za alociranje podataka na shardove i osigurava da podaci odlaze na odgovarajuci *shard*.
- Ovo u mnogome olakšava korišćenje *sharding*-a u aplikaciji.

Modeli distribucije – *Sharding*

- Sharding je od posebnog znacaja za preformasne posto moze da poboljsa i performanse operacije citanja i pisanja u bazi podataka.
- Koriscenje repliciranja, posebno keširanjem, moze dosta poboljsati preformanse čitanja, ali nema puno znacaja za aplikacije koje imaju puno pisanja.
- *Sharding* obezbedjuje nacin za horizontalno skaliranje pisanja.

Modeli distribucije – *Sharding*

- Iako je sharding mnogo laksi sa agregiranjem podataka ne treba mu pribegavati olako.
- Neke baze podataka predvidjaju koriscenje shardinga od pocetka, tako da je dobro izvrsavati ih na klasteru od samog pocetka razvoja i svakako u fazi produkcije.
- U nekim situacijama sharding se koristi oprezno počinjući od single-server konfiguracije, a na sharding se prelaze tek u trenutku kada se pojavljuju problemi s opterećenjem.

Modeli distribucije – *Sharding*

- U svakom slucaju, iskorak od jednog severa ka *sharding*-u bice nezgodan.
- Postoje iskustva o timovima koji su se nasli u nezgodnoj situaciji zato sto su ostavljali sharding za sam kraj, tako da je proces distribucije shard-ova u potpunosti blokirao resurse.
- Pouka koju odavde izvlacimo jeste da sharding treba koristiti mnogo pre nego sto nam zaista zatreba, tako da imamo dovoljno prostora u smislu resursa da se distribucija shard-ova obavi bez problema.

Agregiranje podataka

- *Agregate* (agregacija) je skup objekata istog domena koji se mogu tretirati kao jedna celina.
- Primer u poslovnom informacionom sistemu može biti zaglavlje fakture i stavke, koje se tretiraju kao posebni objekti, ali je zgodno rukovati fakturom (zaglavlje + stavke) kao jednom celinom - agregacijom.

Agregiranje podataka

- Jedana od komponenti agregiranog objekta je koren agregacije.
- Referenciranje spolja se može se vršiti samo preko korena agregacije.
- Na taj način koren može obezbediti integritet agregacije kao celine.
- Agregacija je osnovni element prenosa između aplikacije i podsistema diskova, odnosno baze podataka.
- Pojam agregacija je čest i koristi se u različitim kontekstima, pri čemu se ne odnosi uvek na isti koncept kao kod *Domen-Driven Design Agregate.*

Agregiranje podataka

- Kada su u pitanju baze podataka, posebno pojavom *NoSQL* baza, osnovna ideja je bila da se podaci na efikasan način smeštaju na klastere većeg broja računara – ovakav način su prvi koristili *Google* i *Amazon*.
- Relacione baze od samog početka nisu bile projektovane za rad na klasterima, pa su se tražile alternative.
- Skladištenje agregiranih objekata u bazu podataka kao osnovne jedinice prenosa daje puni smisao ideji da se sistem baze poddataka izvršava na klasteru računara.

Agregiranje podataka

- Agregacije predstavljaju prirodne jedinice za različite strategije distribucije kao što je *Sharding,* jer postoji relativno veliki skup podataka kojem se pristupa kao celini.
- Osim toga, koncept agregiranja u pristupu podacima ima smisla i s programerskog aspekta.
- Kada se podaci preuzimaju s ekrana/forme aplikacije koja predstavlja jednu celinu i žele da se smeste u relacionu bazu podataka, pre smeštanja u bazu podaci se moraju dekomponovati na *n*-torke koje se smeštaju u skup različitih tabela.

Agregiranje podataka

- Kada se koriste agregirani koncepti, kao kod *NoSQL* baza podataka, preslikavanje je mnogo jednostavnije, jer se kompletan sadržaj koji se preuzima može smestiti u bazu podataka kao jedna celina.
- Zbog toga se često kao jedna od prednosti *NoSQL* baza podataka navodi da predstavljaju jednostavniji programski model.

Agregiranje podataka

- Programski model i modeli distribucije koje podržavaju *NoSQL* baze podataka omogućavaju da se kombinacijom mogućnosti koje nudi sama baza podataka i načinom na koji se u aplikativnim programima agregiraju podaci poboljšaju ukupne performanse sistema koji se izvršava na klesterima računara.

Agregiranje podataka

- S druge strane, postoje i nedostaci ovakvog pristupa.
- Sve radi veoma dobro kada se podacima pristupa preko agregacije kao celinie.
- Međutim, postavlja se pitanje - šta ako se želi drugačiji pogled na podatake i podacima je potrebno pristupiti na drugačiji način?
- Kad a se želi vršiti analiza podataka koji su deo agregacije, potrebno je iz sveke agregacije uzeti deo podataka koji se odnose na tu posebnu analizu.

Agregiranje podataka

- U tom smislu su u prednosti sistemi kod kojih se ne korite agregirane strukture u bazi podataka, kao što je slučaj kod relacionih baza podataka, jer omogućavaju da se pojedinačnim podacima jednostavno pristupa jer su već razdvojeni po posebnim tabelama.

Agregiranje podataka

- Zbog toga se kada su *NoSQL* baze podataka u pitanju koristi *Map-Reduce* programski model koji je pogodan za obradu podataka na klasterima računara.
- Kada se koristi *Map-Reduce,* podaci se, preko tzv. materijalizovanih pogleda, mogu organizovati u različite grupe za različite potrebe pretraživanja podataka.
- Međutim, ovakav pristup je složeniji i zahteva više rada nego kada se koriste relacione baze podataka.

Agregiranje podataka

- Iz svega se nameće zaključak da *NoSQL* baze podataka, gledajući na vremenskoj osi, nisu koncept koji dolazi kao zamena za relacione baze podataka;
- U smislu da ih prevazilazi, već kao dobra mogućnost u nekim oblastima obrade podataka u kojima relacione baze pokazuju slabosti.
- U pojedinim oblastima primene relacione baze jednostavno nemaju alternativu.

Agregiranje podataka

- Pojednostavljeno možemo reći da *NoSQL* baze treba koristiti kada se radi s podacima koji po prirodi stvri predstavljaju agregirane objekte, posebno kada se korisete klasteri računara;
- A relacione baze podataka kada je pojedinačnim podacima potrebno manipulisati na različite načine.

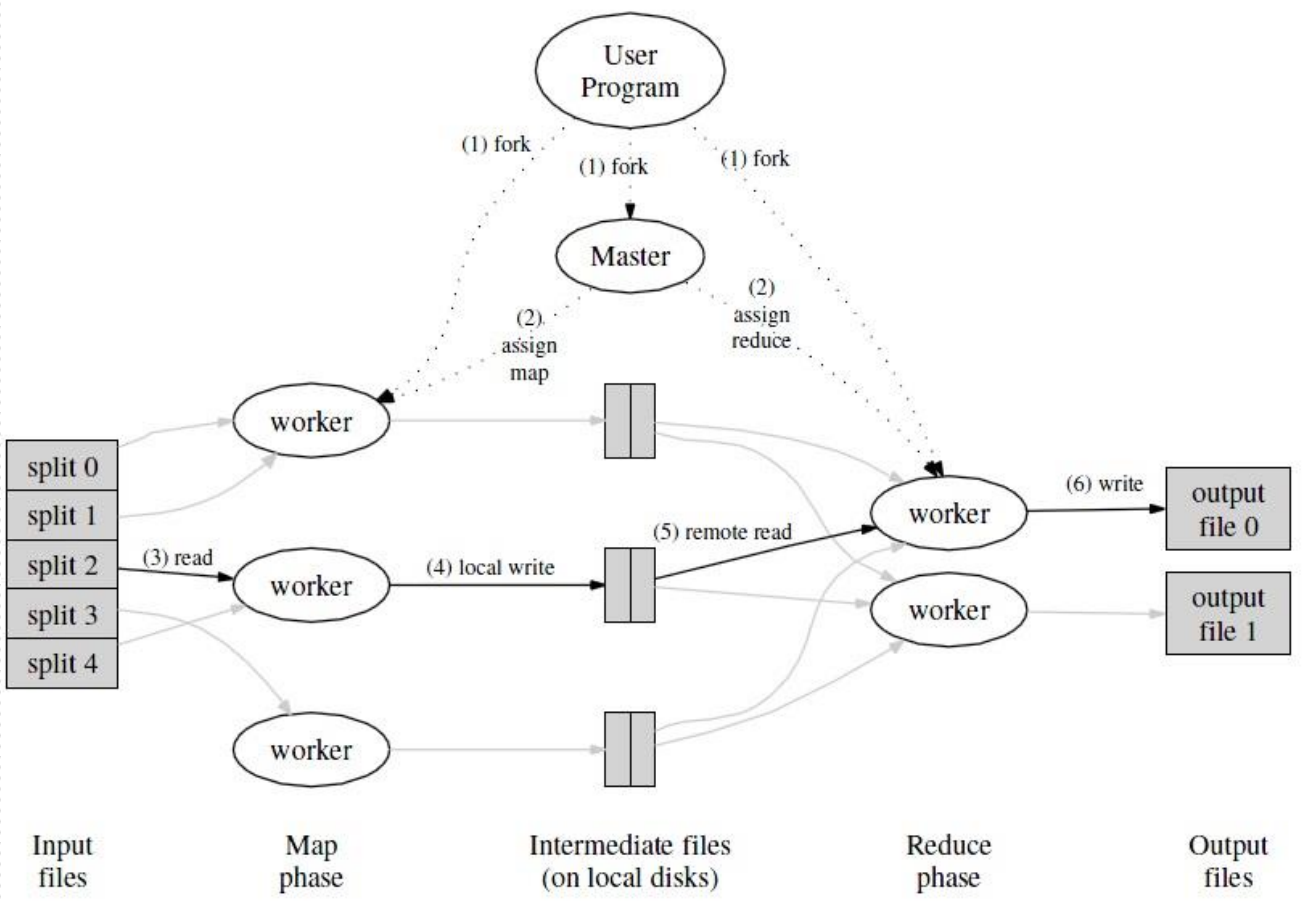
MapReduce algoritam

- Omogućava upravljanje velikum količinama podataka.
  - Posredstvom distribuirane mreže čvorova.
- Deli velike zadatke na manje delove;
  - Pogodne za paralelno procesiranje
- *MapReduce* okruženje
  - *Upravlja izvršenje MR* algoritma
  - *Obezbedjuje spregu s korisnikom*

MapReduce algoritam

- ***Map* korak**
  - **Glavni čvor**
    - Preuzima ulazne podatke
    - Deli ulazne podatke na manje delove
    - Distribuira radne delove u manje čvorove
  - **Radni čvor**
    - Procesiraju dodeljene zadatke
    - Vraćaju odgovor glavnom čvoru
- ***Reduce* korak**
  - **Glavni čvor**
    - Preuzima rezultate radnih čvorova
    - Kombinuje rezultate sa ciljem dobijanja traženog rezultata

MapReduce algoritam



Na kraju – da ponovimo

- Poslednjih nekoliko godina, došlo je do eksplozije količine podataka (*big data*) kojima je potrebno rukaovati,
- Međutim, postojeći resursi nisu bili dovoljni, pa su se intezivno tražila nova rešenja za skladištenje, ažuriranje i pretraživanje velikih količina podataka.
- Kao odgovor na pomenuti problem pojavljuju se NoSQL baze podataka koje su već u startu projektovane tako da podrže problem rukovanja velikim količinama podataka.

Na kraju – da ponovimo

- NoSQL baze, za razliku od do sada dominatnih relacionih baza podataka, projektovane tako da obezbede dobre performanse pri izvršavanju na klasterima računara jeftinog hardvera.
- Na taj način se obezbedilo jednostavnije i jeftinije horizontalno skaliranje sistema.
- Osim toga većinu priliva podataka predstavljaju nestruktuirani podaci kojima pogoduje nepostojanje šeme kod NoSQL baza, što omogućaja jednostavno i slobodno redefinisanje zapisa baze podataka, dodavanjem ili uklanjanjem polja/obeležja bez promene strukture.

Na kraju – da ponovimo

- Jedana od osnovnih prednosti koje donose *NoSQL* baze je mogućnost distribuirane obrade i izvršavanje na klasterima računara;
- Za razliku od relacionih baza podataka kod kojih se favorizuje centralizovana obrada na jednom serveru.
- Na taj način se omogućuje jednostavnije skaliranje sistema kada se pojavi veća količina podataka i potreba za dodatnim resursima

Na kraju – da ponovimo

- Primarni razlozi korišćenja *NoSQL* baza podataka su sledeći:
  - paralelno procesiranje velikih količina podataka u distribuiranim sistemima
  - pretraga podataka u sistemima sa velikom količinom podataka
  - skladištenje velikih količina podataka(struktuiranih, nesktruktuiranih, delimično struktuiranih).

Na kraju – da ponovimo

- Prednosti NoSQL sistema su:
  - ne poseduju formalno specificiranu šemu baze podataka- rukovanje fleksibilnim strukturama podataka,
  - nestrukturirani i polustrukturirani podaci,
  - ne oslanjaju se na relacioni model podataka - ne podržavaju operacije spajanja,
  - unapređene performanse u radu sa velikom količinom podataka,
  - horizontalna skalabilnost i elastičnost - povećanje kapaciteta i performansi u radnom režimu sistema,

Na kraju – da ponovimo

- dodavanjem čvorova u distribuiranu mrežu,
- arhitektura lokalizovanih resursa - svaki server se oslanja na lokalnu masovnu memoriju,
- particionisanje baze podataka - ukupan broj zapisa je distribuiran u particije,
- asinhrona replikacija - podaci nisu replicirani onog momenta kada su zapisani,
- izbegavanje objektno-relacionog mapiranja - podaci se čuvaju u posebnim strukturama podataka obično jednostavnim bliskim objektno orijentisanim programskim jezicima,
- tolerancija na otkaze.

Na kraju – da ponovimo

- Nedostaci NoSQL sistema su:
  - nepostojanje standarda,
  - nepostojanje standardnog upitnog jezika - UnQL (eng. *Unstructured Query Language*)
  - dominantno otvorenog koda, nedostatak dokumentacije i korisničke podrške.

Vrste *NoSQL* baza podataka

- Poslednjih godina, razvijena su mnoga NoSQL rešenja čiji je zadatak da zadovolje zahteve i potrebe različitih organizacija;
- U cilji skladištenja podataka.
- Skoro svaki sistem baza podataka koji ne podleže principima relacionih baza podataka može se svrstati u *NoSQL* sisteme.

Vrste *NoSQL* baza podataka

- Zbog toga je došlo do podela NoSQL baza podataka u nekoliko grupa i to:
  - skladišta podataka tipa ključ – vrednost,
  - kolonski orijentisana skladišta podataka,
  - baze podataka orijentisane ka dokumentima,
  - baze podataka orijentisane ka grafovima.

Različite vrste NoSQL baza podataka