BIG DATA 4

Technológie spracovania veľkých dát

Peter Bednár, Martin Sarnovský

Distribuované súborové systémy a databázy

- Distribuované súborové systémy
- Distribuované databázy
- Konzistentnosť, dostupnosť a tolerancia voči výpadkom
  - Mechanizmy na zabezpečenie konzistentnosti dát
  - CAP teoréma

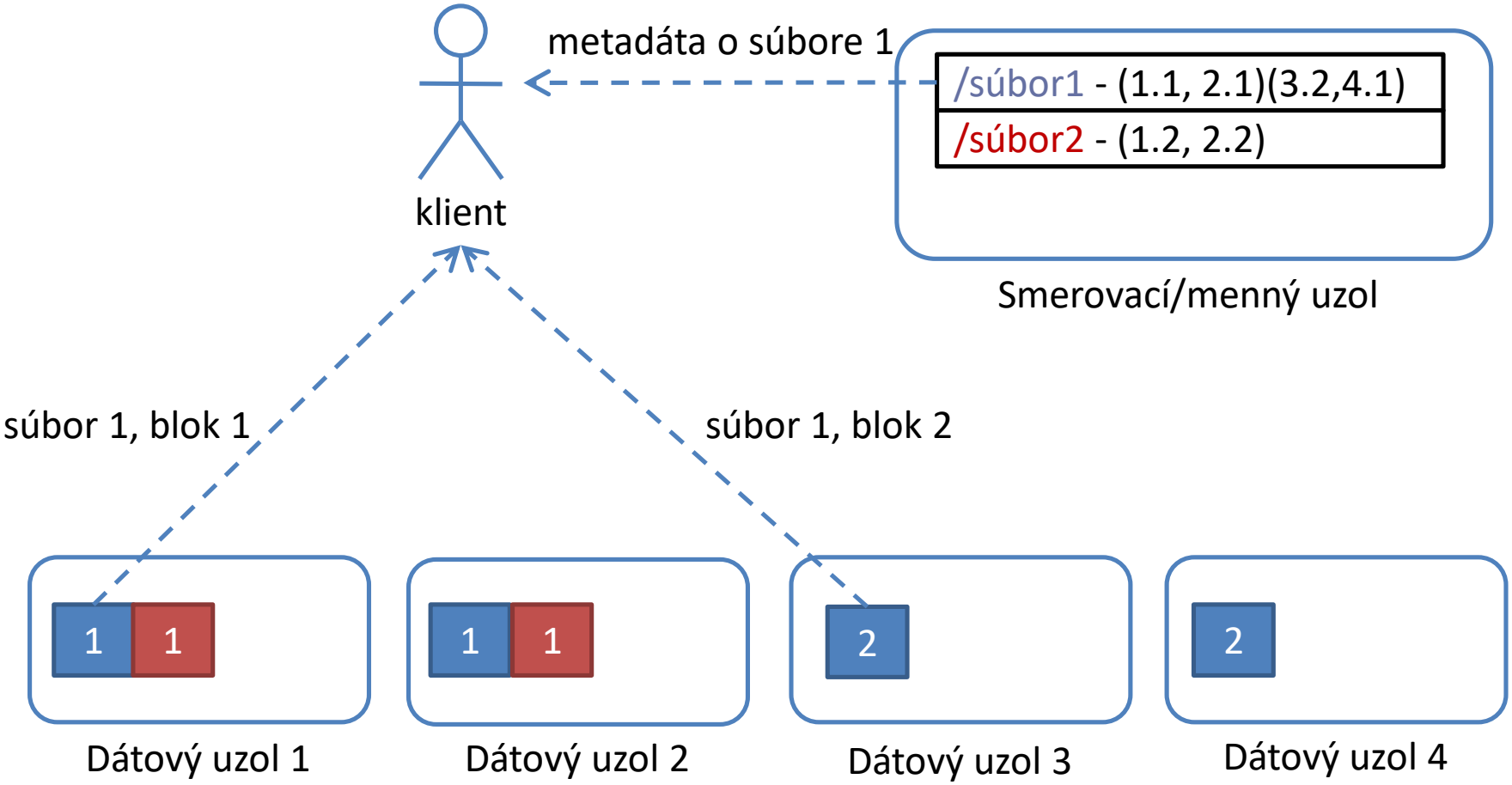
Distribuované súborové systémy

- Veľké súbory (nad niekoľko TB) už nie je možné uložiť na jeden fyzický disk
- Distribuovanie dát – súbory sa rozdelenia na menšie bloky, ktoré sa uložia na samostatných serveroch – dátových uzloch
- Replikácia dát - kópie tých istých dátových blokov sú uložené na viacerých uzloch
  - Zabezpečenie dát – ak zlyhá jeden uzol, dáta sú ešte dostupné na záložných uzloch
  - Rýchlejší prístup k dátam – pri čítaní dát sa môže klient pripojiť k viacerým uzlom, ktoré uchovávajú kópie požadovaných dát a môže si vybrať uzol s ktorým má najrýchlejšie sieťové spojenie
- Bloky majú zvyčajne rovnakú, pevne danú max. veľkosť

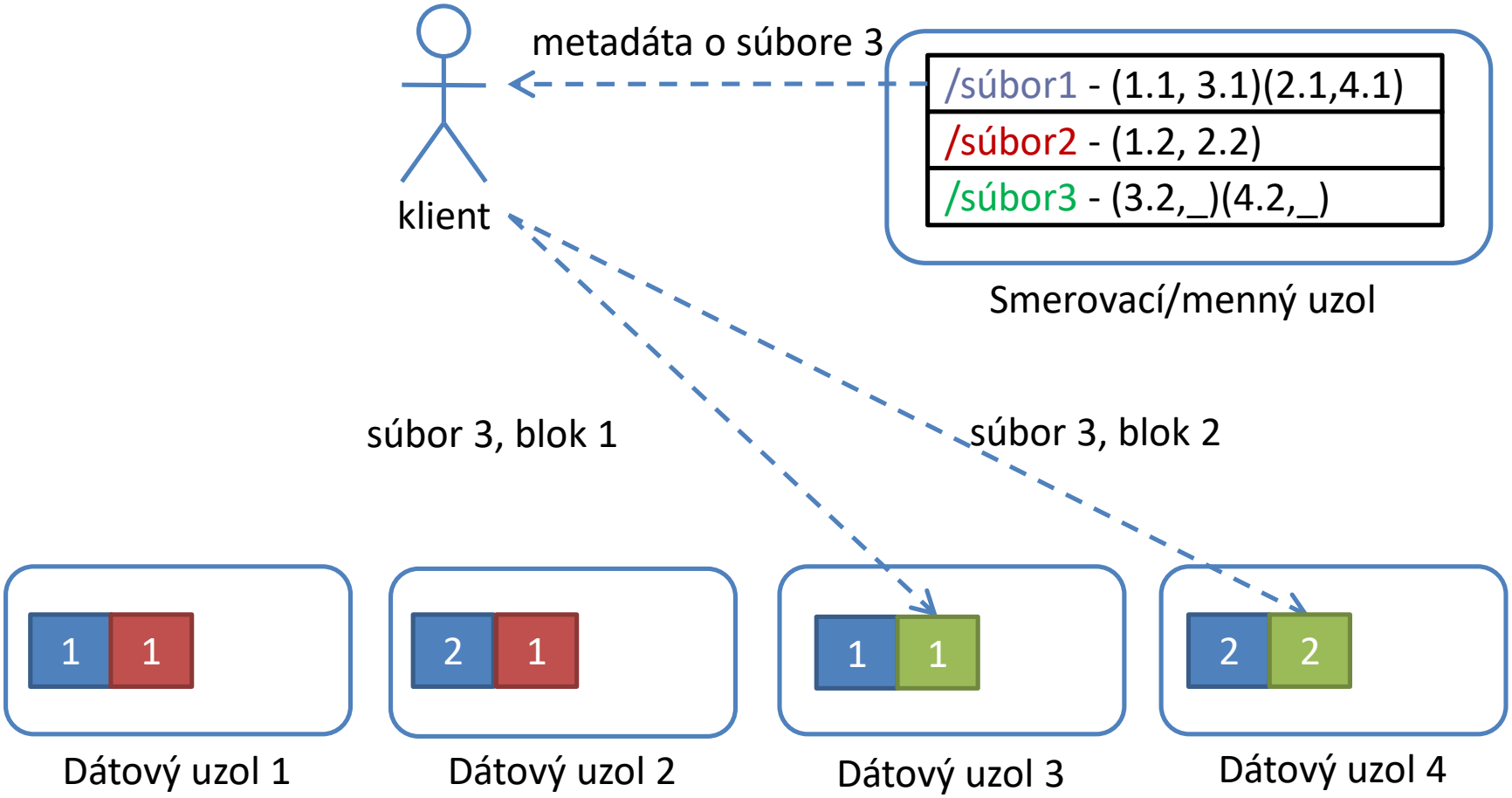
Architektúra distribuovaného súborového systému

- Dva typy serverov - uzlov:
- Dátový uzol
  - Uchováva dátové bloky
- Smerovací/menný uzol
  - Rozdeľuje súbory na bloky a rozhoduje, na ktorých dátových uzloch budú bloky uložené/replikované
  - Uchováva metadáta o rozdelení súborov a umiestnení blokov
- Pri zápise/čítaní klient komunikuje najprv so smerovacím/menným uzlom, ten určí kam sa majú uložiť zapisované dátové bloky, alebo kde sa nachádzajú čítané dáta, potom klient komunikuje priamo z vybraným dátovým uzlom

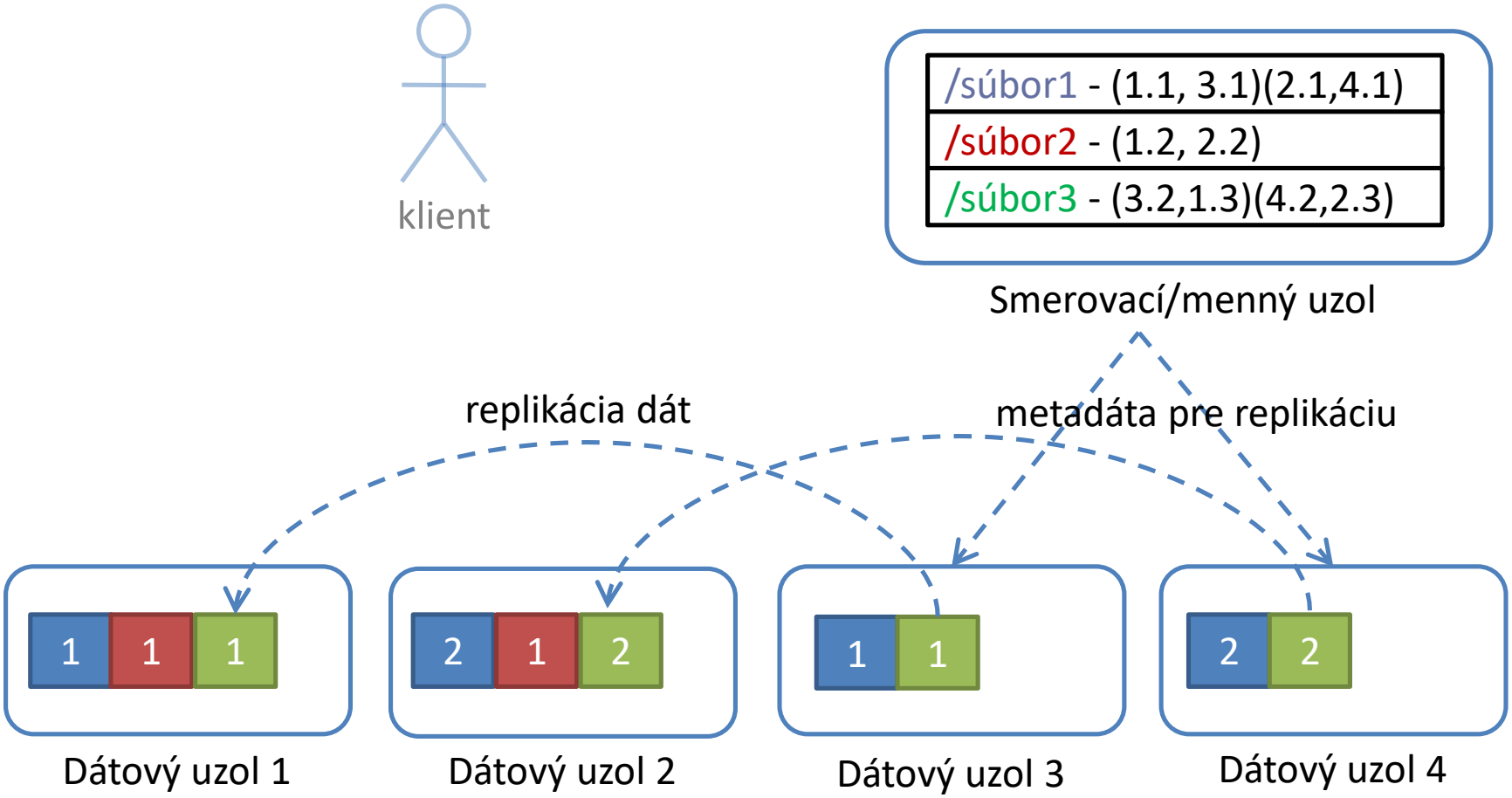
Distribuovaný súborový systém – čítanie dát



Distribuovaný súborový systém – zápis dát



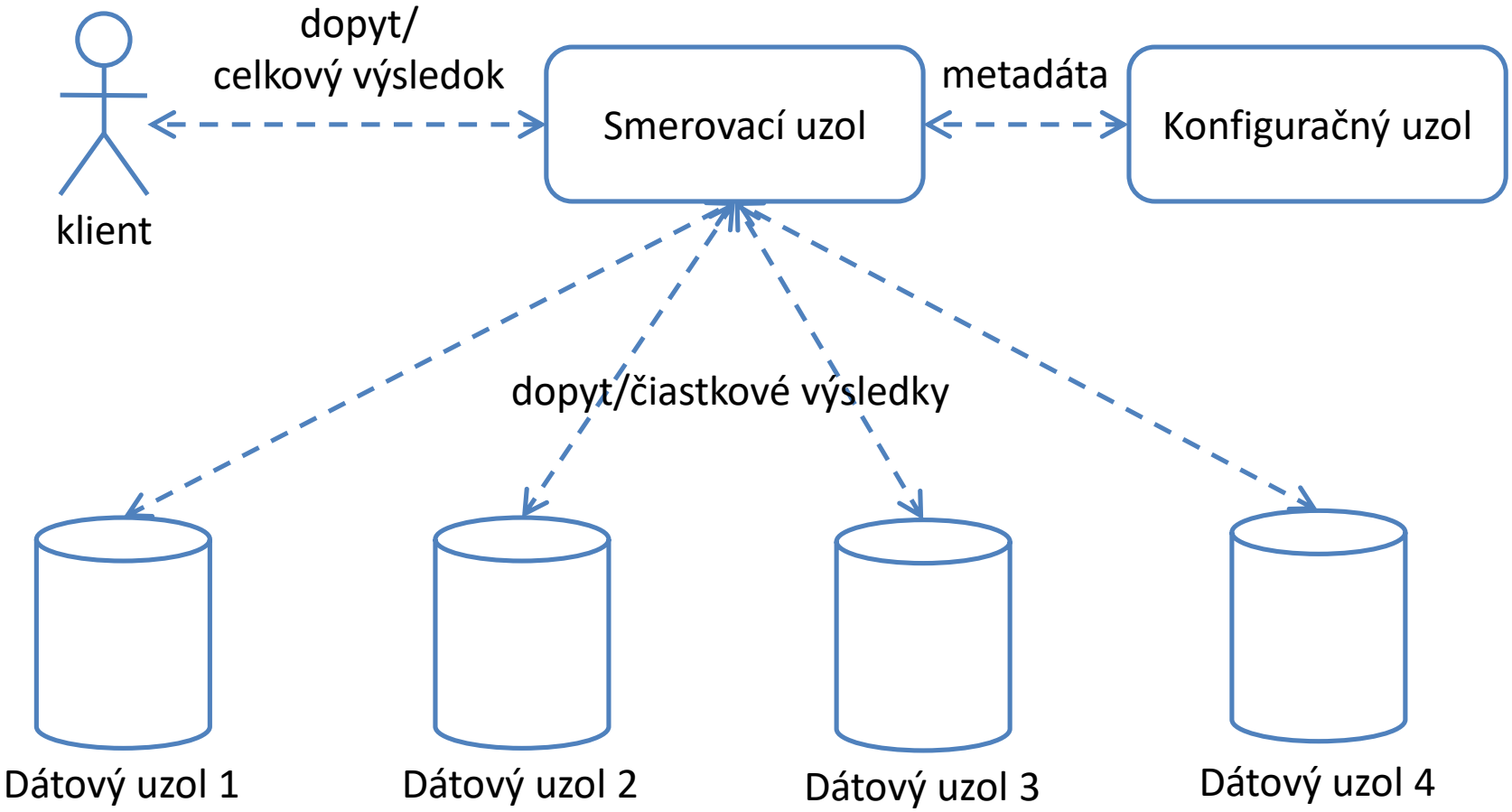
Distribuovaný súborový systém – replikácia



Distribuované databázy

- Podobná architektúra ako pri distribuovaných súborových systémoch:
- Dátový uzol - uchováva dátové záznamy
- Smerovací uzol
  - Pri pridávaní nových záznamov rozhoduje na ktorom dátovom uzly budú dáta uložené/replikované
  - Pri dopytovaní rozosiela dopyt na jednotlivé dátové uzly a spája čiastkové výsledky z dátových uzlov do výsledného výsledku
- Konfiguračný uzol - uchováva metadáta o rozdelení záznamov na jednotlivé dátové uzly

Architektúra distribuovanej databázy



Distribuovanie dát v databázach - *sharding*

- Dátová tabuľka/kolekcia sa rozdelí na dátové uzly po riadkoch/záznamoch (horizontálne delenie dát)
  - Dáta je možné distribuovať aj po stĺpcoch – vertikálne delenie dát - bežné pre stĺpcové databázy
- Programátor môže zvyčajne určiť jeden kľúč, podľa ktorého sa dáta rozdelia:
  - Rozdelenie by malo byť rovnomerné aby bola záťaž na jednotlivých dátových uzloch vyvážená
  - Ak sa pri dopytovaní dáta často filtrujú podľa niektorého atribútu (napr. podľa poštového čísla) je výhodné tento atribút zvoliť ako kľúč pre distribuovanie dát – smerovací uzol potom môže dopredu zistiť, ktoré uzly priamo obsahujú filtrované dáta a nemusí kontaktovať všetky dátové uzly – aplikačne závislé

Sharding - rozdelenie dát, príklad (1)

id	položka	počet
0	položka 1	10
1	položka 1	5
2	položka 2	15
3	položka 2	10
4	položka 1	20
5	položka 2	30
6	položka 3	25
7	položka 2	10
8	položka 1	5
9	položka 3	20

0	položka 1	10
3	položka 2	10
6	položka 3	25
9	položka 3	20

uzol 0
kľúč 0

1	položka 1	5
4	položka 1	20
7	položka 2	10

uzol 1
kľúč 1

2	položka 2	15
5	položka 2	30
8	položka 1	5

uzol 2
kľúč 2

Sharding - rozdelenie dát, príklad (2)

kľúč = id % 3, max. veľkosť 4, počet uzlov 3

0/0	položka 1	10
1/1	položka 1	5
2/2	položka 2	15
3/0	položka 2	10

uzol 0 kľúč [0-2]

4/1	položka 1	20

uzol 1

Kým sa nedosiahne max. veľkosť, záznamy sú ukladané na jeden uzol.

uzol 2

Sharding - rozdelenie dát, príklad (3)

kľúč = id % 3, max. veľkosť 4, počet uzlov 3

0/0	položka 1	10
3/0	položka 2	10

uzol 0
kľúč 0

1/1	položka 1	5
2/2	položka 2	15
4/1	položka 1	20

uzol 1
kľúč [1,2]

Keď sa prekročí max. veľkosť, záznamy sú rovnomerne rozdelené na viacero uzlov.

uzol 2

Sharding - rozdelenie dát, príklad (4)

kľúč = id % 3, max. veľkosť 4, počet uzlov 3

0/0	položka 1	10
3/0	položka 2	10
6/0	položka 3	25

uzol 0
kľúč 0

7/1	položka 2	10

1/1	položka 1	5
2/2	položka 2	15
4/1	položka 1	20
5/2	položka 2	30

uzol 1
kľúč [1,2]

uzol 2

Sharding - rozdelenie dát, príklad (5)

kľúč = id % 3, max. veľkosť 4, počet uzlov 3

0/0	položka 1	10
3/0	položka 2	10
6/0	položka 3	25

uzol 0
kľúč 0

1/1	položka 1	5
4/1	položka 1	20
7/1	položka 2	10

uzol 1
kľúč 1

2/2	položka 2	15
5/2	položka 2	30

uzol 2
kľúč 2

Sharding – zvolený kľúč

Snažíme sa zvoliť kľúč, ktorý sa často používa pre filtrovanie dát pri dopytoch.

Napr. ak aplikácia potrebuje často zobraziť iba záznamy pre položku 1, smerovací uzol získa výsledok priamo z uzla 0.

Dáta však už nemusia byť rozdelené rovnomerne.

0	položka 1	10
1	položka 1	5
4	položka 1	20
8	položka 1	5

uzol 0
položka 1

2	položka 2	15
3	položka 2	10
5	položka 2	30
7	položka 2	10

uzol 1
položka 2

6	položka 3	5
9	položka 3	20

uzol 2
položka 3

Vertikálne delenie dát pre stĺpcové databázy

id	položka	počet
0	položka 1	10
1	položka 1	5
2	položka 2	15
3	položka 2	10
4	položka 1	20
5	položka 2	30
6	položka 3	25
7	položka 2	10
8	položka 1	5
9	položka 3	20

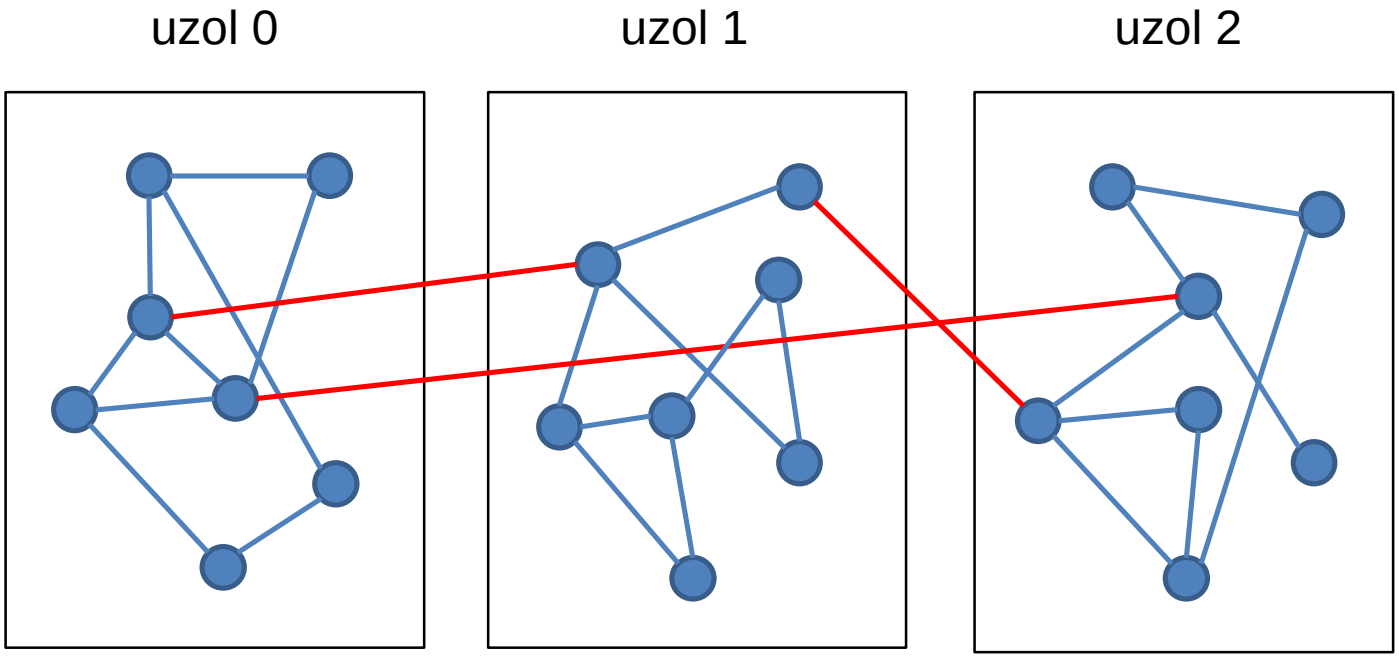
položka 1	0,1,4,8
položka 2	2,3,5,7
položka 3	6,9

uzol 0
položka

5	1,8
10	0,3,7
15	2
20	4,9
25	6
30	5

uzol 1
počet

Distribuovanie dát v grafových databázach



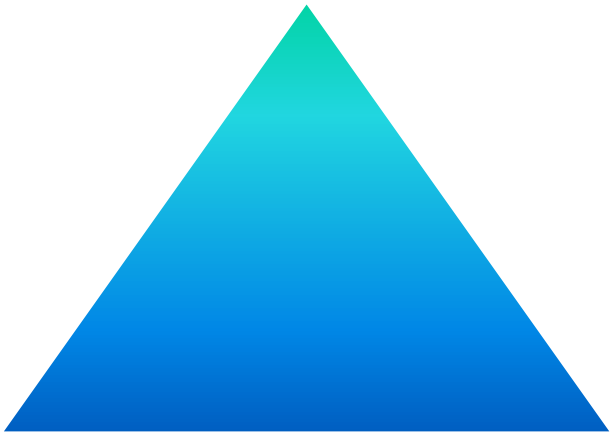
Pri grafových databázach sa snažíme rozdeliť vrcholy grafu tak, aby čo najmenej hrán prechádzalo medzi vrcholmi uloženými na rôznych dátových uzloch

Základné vlastnosti distribuovaných systémov (1)

- Distribuovaný systém je zložený z viacerých uzloch, ktoré uchovávajú dáta a navzájom komunikujú cez sieť.
- Klienti sa pripájajú ku niektorým z uzlov, pričom naraz sa môže pripojiť paralelne viacero klientov s rôznymi požiadavkami
- Komunikácia cez sieť medzi klientom a uzlami, alebo medzi uzlami navzájom nie je spoľahlivá. Odosielané správy:
  - Nemusia byť vôbec doručené
  - Môžu byť doručené v odlišnom poradí než boli odoslané
  - Môžu byť doručené viac krát
- Jednotlivé uzly môžu zlyhať

Základné vlastnosti distribuovaných systémov (2)

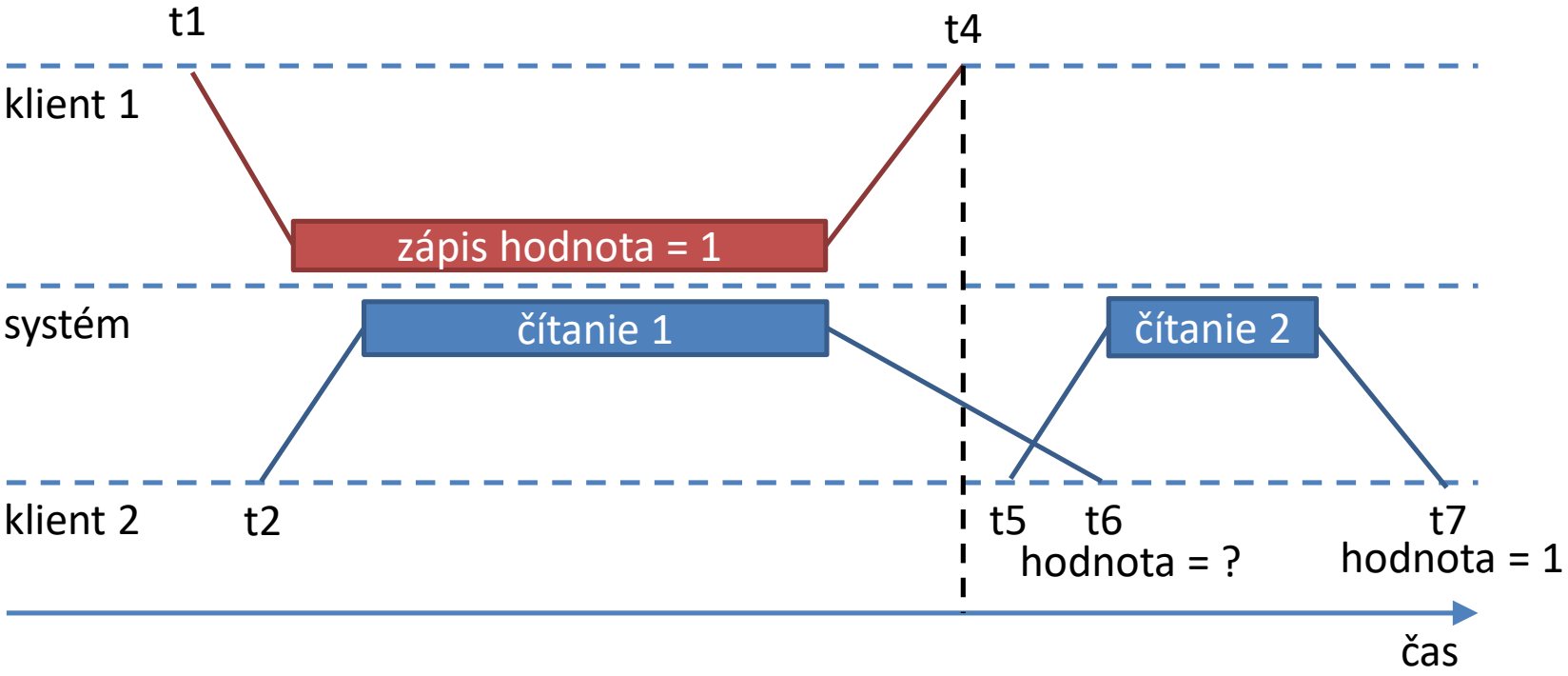
Konzistentnosť – **Consistency**
všetci klienti majú „správny“
pohľad na dáta



Dostupnosť – **Availability**
všetci klienti môžu vždy
zapisovať a čítať dáta

Spoľahlivosť – **Partition**
systém pracuje správne aj
pri výpadkoch uzlov/
komunikácie

Konzistentnosť



- Ak čítanie začne po potvrdení zápisu, každý klient musí prečítať aktualizovanú hodnotu

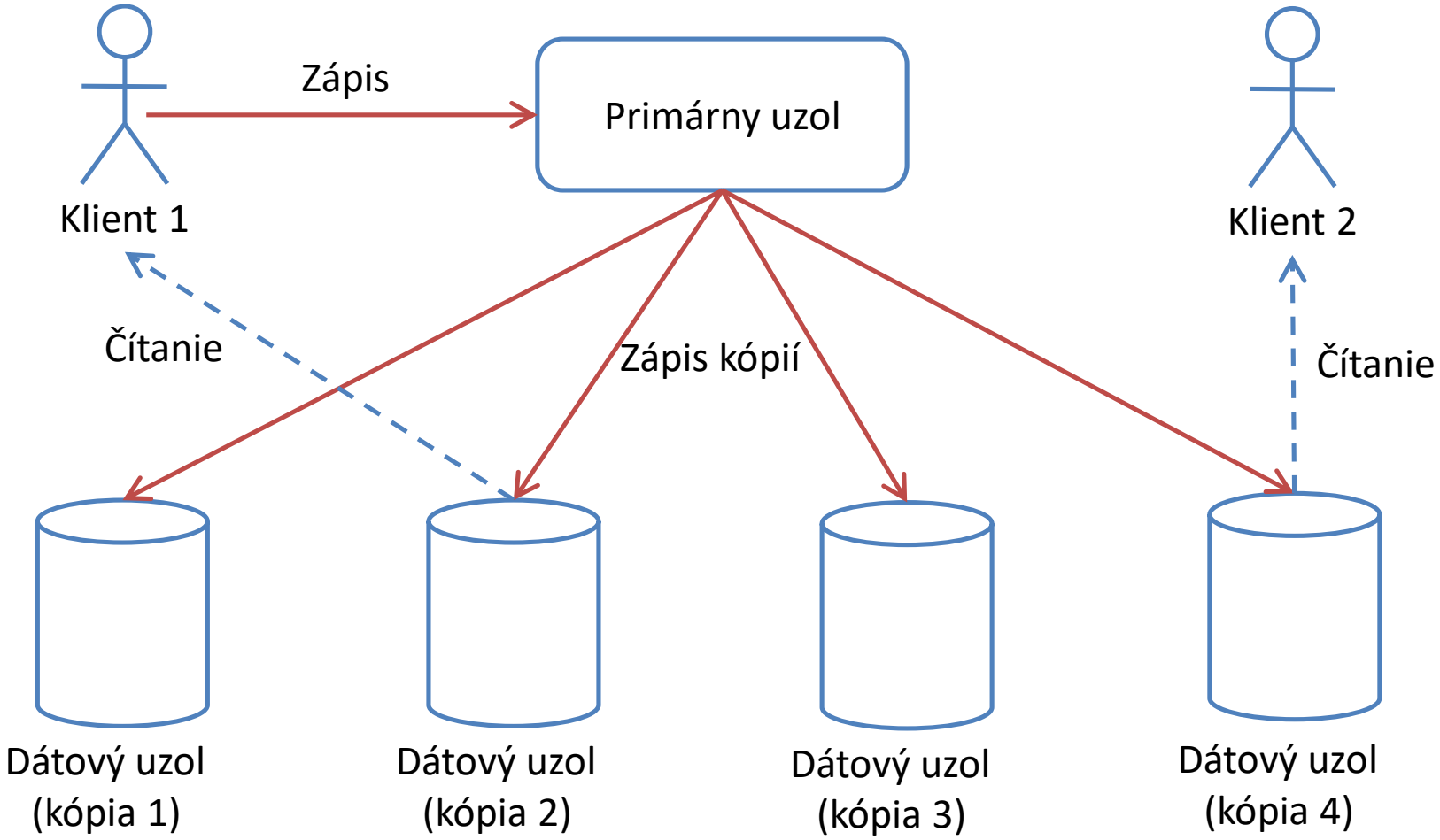
Konzistentnosť a replikácia

- Pri replikácii sú kópie dát uložené na viacerých dátových uzloch
- Klient môže dáta čítať z ľubovoľného uzla na ktorom je uložená kópia požadovaných dát
  - Inak by sa obmedzila škálovateľnosť
- Silná konzistentnosť
  - Systém pri potvrdení zápisu musí zabezpečiť, že každá kópia dát bude aktualizovaná. Všetci klienti pri nasledujúcom čítaní získajú aktuálnu hodnotu
- Slabá konzistentnosť
  - Iba niektoré kópie sú aktualizované v čase potvrdenia zápisu. Niektoré uzly sú aktualizované asynchrónne, tzn. niektorí klienti môžu dočasne prečítať neaktuálne hodnoty. Po čase musí systém skonvergovať do konzistentného stavu

Replikácia – jeden primárny uzol (1)

- Pre čítanie dát sa klienti môžu pripojiť na ľubovoľný uzol na ktorom je kópia požadovaných dát
- Pri zápise sa všetci klienti pripoja na jeden určený primárny uzol – ten musí zabezpečiť atomickú aktualizáciu všetkých kópií (tzn. buď každý z uzlov aktualizuje svoju kópiu, alebo žiadny)
  - Vyžaduje si to komunikáciu medzi uzlami a tzv. protokol dvojfázového potvrdzovania

Replikácia – jeden primárny uzol (2)



Protokol dvojfázového potvrdzovania

1. Primárny uzol odošle zapisované dáta a požiadavku na hlasovanie o zápise všetkým dátovým uzlom
2. Každý dátový uzol prevezme zapisované dáta do dočasného logu, ak nenastane chyba odošle hlasovanie za zápis, pri chybe odošle hlasovanie za zrušenie zápisu
3. Ak primárny uzol získa hlasovanie od všetkých uzloch, tak:
   a. Ak všetky uzly hlasovali za zápis, odošle uzlom potvrdenie zápisu
   b. Ak aspoň jeden uzol hlasoval za zrušenie, odošle uzlom príkaz pre zrušenie zápisu
4. Ak dátový uzol získa potvrdenie pre zápis, trvalo zapíše dáta z logu, inak zápis zruší a vymaže dáta z logu

Jeden primárny uzol - zhrnutie

- Vieme zabezpečiť silnú konzistentnosť
- Dobrá škálovateľnosť pri čítaní dát – klient môže komunikovať s najbližším dátovým uzlom

- Slabá škálovateľnosť pri zapisovaní
  - Všetci klienti komunikujú s jedným primárnym uzlom
  - Primárny uzol musí čakať na potvrdenie od všetkých dátových uzlov (ak len jeden má zlé spojenie s primárnym uzlom, zápis sa spomalí)
- Jeden bod zlyhania (slabá spoľahlivosť)
  - Ak vypadne primárny uzol, žiaden klient nemôže zapisovať dáta

Replikácia – väčšinové hlasovanie (1)

- Primárny uzol nie je pevne daný
- Pri zápise môže klient komunikovať s ľubovoľným dátovým uzlom, ktorý sa pokúsi získať súhlas na to aby sa stal primárnym uzlom od väčšiny ostatných uzlov, ktoré uchovávajú tú istú kópiu
- Ak máme uložiť n kópií, definujeme:
  - QW – kvórum pre zápis
  - QR – kvórum pre čítanie dát

  pričom platí $QW > n/2$ a $QW + QR > n$
  - Napr. ak máme 7 kópií, tak QW = 5, QR = 3

Replikácia – väčšinové hlasovanie (2)

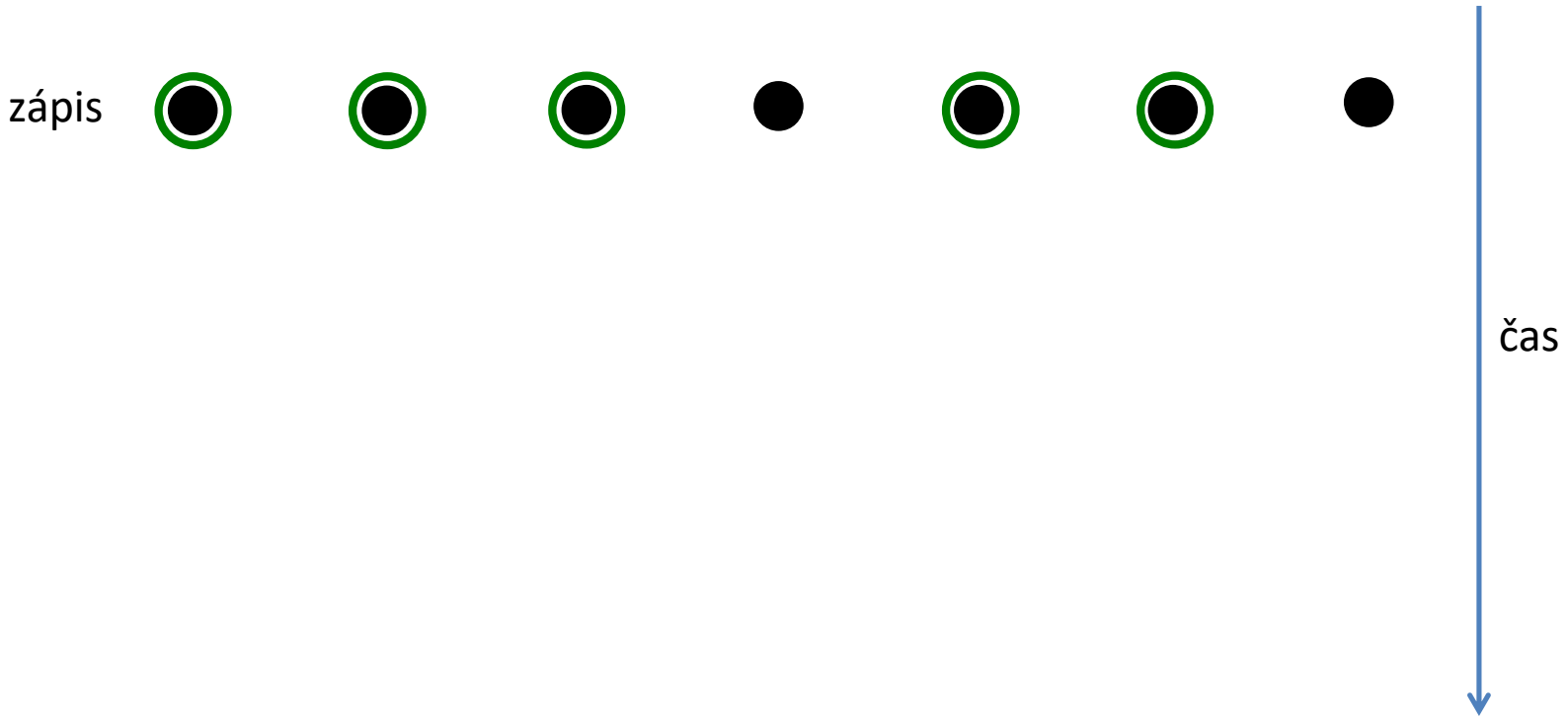
- Klient začne komunikovať s ľubovoľným uzlom – zapisujúci uzol
- Zapisujúci uzol sa postupne pokúsi získať súhlas od ostatných uzlov ktoré uchovávajú tú istú kópiu
- Každý uzol ktorý dostane požiadavku, odošle súhlas na zápis, alebo odošle zamietnutie ak už prebieha iný zápis (tzn. ak je sám zapisujúcim uzlom, alebo už odoslal súhlas inému zapisujúcemu uzlu – naraz sa môže pokúšať o zápis viacero klientov)
- Ak zapisujúci uzol získa súhlas od QW uzlov, atomicky zapíše dáta na všetky kópie od ktorých dostal súhlas a odošle potvrdenie klientovi
- Ak sa mu nepodarí získať súhlas, zapisujúci uzol počká stanovený čas a pokúsi sa získať súhlas znovu
- Zvyšné kópie sú aktualizované asynchrónne zapisujúcim uzlom

Replikácia – väčšinové hlasovanie (3)

- Klient čaká iba na zápis QW kópií, zvyšné sa aktualizujú asynchrónne
- Aby získal klient vždy aktuálnu hodnotu, musí prečítať dáta aspoň od QR ľubovoľných uzlov, ktoré uchovávajú tú istú kópiu
- Z definície kvóra vyplýva, že:
  - Iba jeden uzol môže získať povolenie na zápis v danom čase
  - Ak si ľubovoľne vyberieme QR uzlov, aspoň jeden bude vždy obsahovať aktuálne dáta

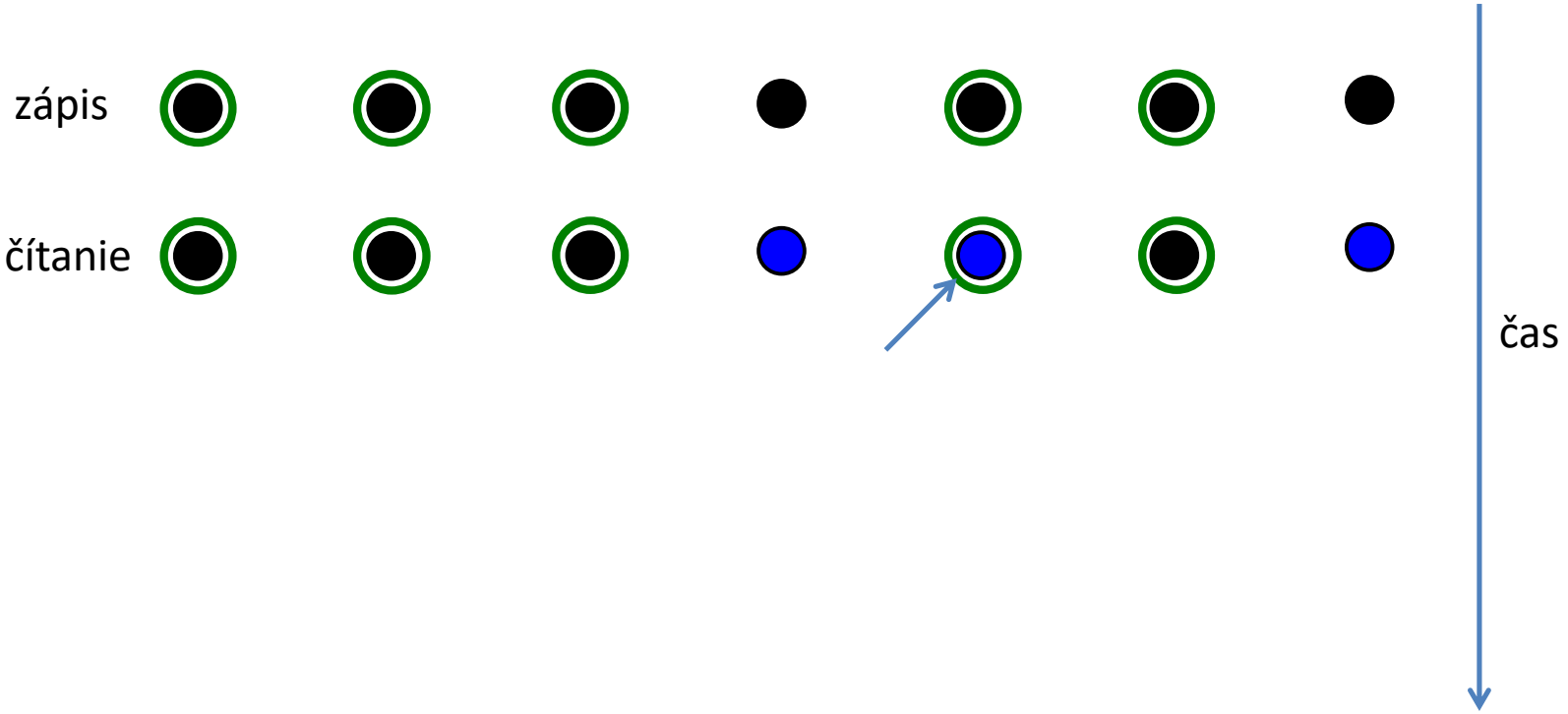
Väčšinové hlasovanie – príklad (1)

- príklad pre n = 7, QW = 5, QR = 3



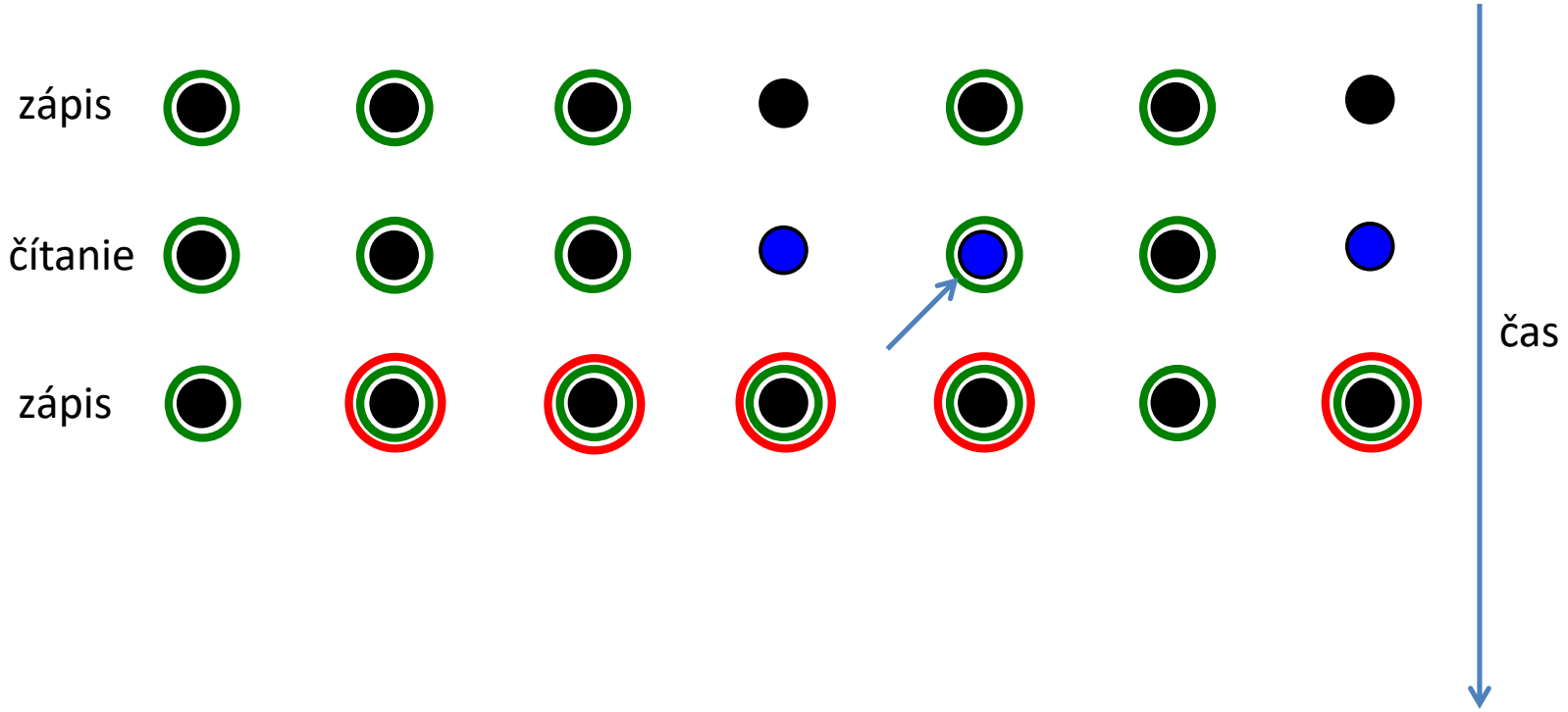
Väčšinové hlasovanie – príklad (2)

- príklad pre n = 7, QW = 5, QR = 3



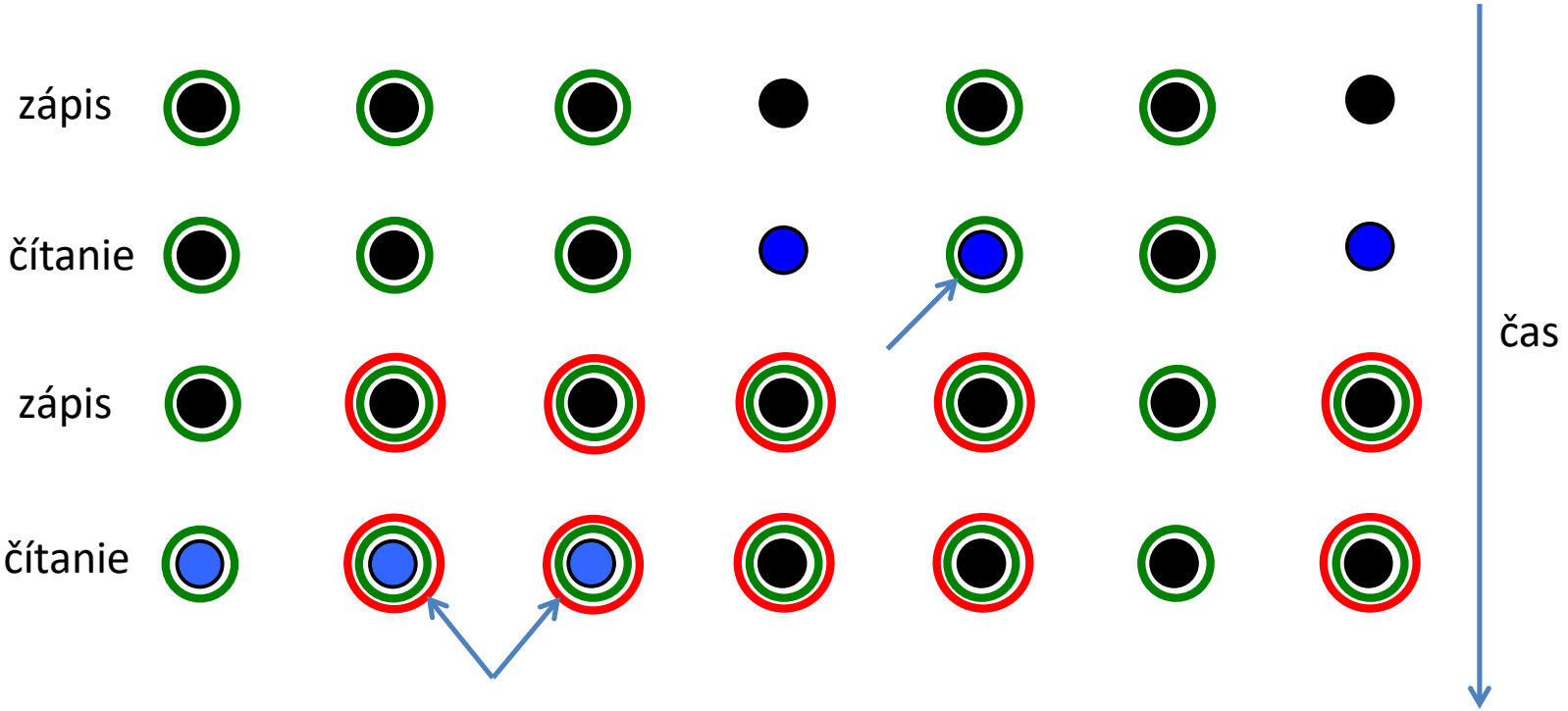
Väčšinové hlasovanie – príklad (3)

- príklad pre n = 7, QW = 5, QR = 3



Väčšinové hlasovanie – príklad (4)

- príklad pre n = 7, QW = 5, QR = 3



Väčšinové hlasovanie - zhrnutie

- Vieme zabezpečiť silnú konzistentnosť
- Vieme zabezpečiť dobrú spoľahlivosť – ak vypadne pri zápise/alebo čítaní niektorý z uzlov, klient môže kontaktovať ďalší

- Silná konzistentnosť si vyžaduje viac komunikácie (hlasovanie, čítanie z viacerých uzloch ak chceme získať vždy aktuálnu hodnotu)
- Kompromis – zlepšili sme spoľahlivosť na úkor škálovateľnosti

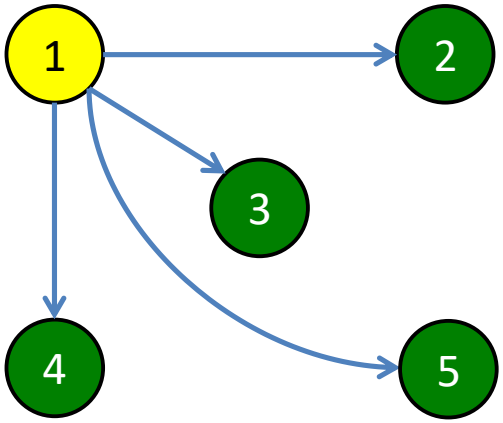
Distribuované transakcie

- Pri replikácii ten istý záznam kopírujeme na viacero uzlov
- Pri transakciách je potrebné atomicky modifikovať viacero záznamov na rôznych uzloch – podobné problémy s konzistentnosťou
- Používa sa protokol dvojfázového uzamykania - podobný ako dvojfázové potvrdzovanie
  - Najprv sa potvrdí začiatok transakcie na všetkých uzloch, ktoré dočasne uzamknú modifikované záznamy
  - Ak každý z uzlov zahrnutých do transakcie potvrdí vykonanie operácie, rozošle sa potvrdenie o prijatí celej transakcie a záznamy sa odomknú

Spoľahlivosť

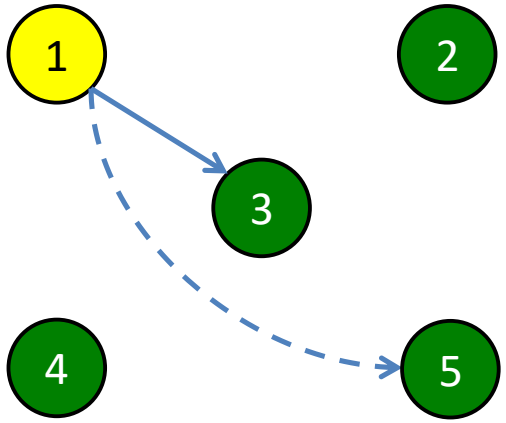
- Je potrebné uvažovať rôzne scenáre zlyhania
- Ak uzol/uzly neodpovedajú na správy
  - Ide o výpadok siete?
    - Úplná strata pripojenia
    - Čiastočná strata pripojenia, rozdelenie siete
  - Zlyhal primárny/zapisujúci/dátový uzol?
    - Ide o trvalé zlyhanie?
    - Ide o dočasné zlyhanie a uzol sa reštartuje a znovu pripojí?
  - Uzol má dočasne, alebo trvalo vyčerpané výpočtové zdroje (zaťažený procesor, vyčerpaná operačná pamäť, atď.)

Výpadok systému – príklad rozdelenia siete (1)



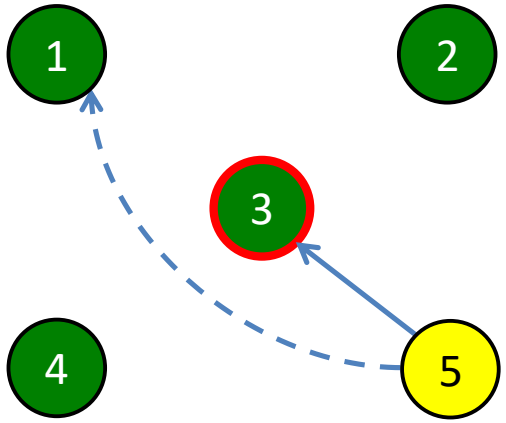
Na začiatku môže každý uzol komunikovať s každým. Uzol 1 je zapisujúci uzol ktorý replikuje dáta od klienta na ostatné uzly. Predpokladajme, že klient ukladá dáta ktoré sú uložené v 3 kópiách na uzloch 1,3,5.

Výpadok systému – príklad rozdelenia siete (2)



Ak došlo k výpadku siete, uzly 1 a 5 nemôžu komunikovať medzi sebou, ale stále vedia komunikovať s uzlami 2,3,4. Uzol 1 stále môže získať kvórum pre zápis a zapíše dáta do uzla 1,3. Uzol 5 má neaktuálne dáta.

Výpadok systému – príklad rozdelenia siete (3)



Ak ďalší klient číta a zapisuje dáta cez uzol 5, prečíta neaktuálnu hodnotu a prepíše dáta na uzly 3.

CAP teoréma

- Vieme zabezpečiť naraz silnú konzistentnosť (Consistency), dostupnosť (Availability) a spoľahlivosť (Partition tolerance)?

- CAP teoréma tvrdí, že nie je možné navrhnúť systém ktorý by splnil všetky 3 požiadavky naraz
- Pri rozsiahlych distribuovaných systémoch je spoľahlivosť nutnosť, tzn. pri návrhu systému sa volí kompromis medzi konzistentnosťou a dostupnosťou
  - Napr. pri hlasovaní musia klienti čakať na zapisujúci uzol kým získa kvórum – znížili sme dostupnosť na úkor konzistentnosti
- Pri spracovaní veľkých dát sa však často uprednostňuje dostupnosť, ktorá zabezpečí lepšiu škálovateľnosť pred konzistentnosťou

Distribuované databázy - zhrnutie

- Nie je možné dosiahnuť úplnú dostupnosť, konzistentnosť a spoľahlivosť naraz
  - Kompromis medzi konzistentnosťou a dostupnosťou (škálovateľnosťou)
  - Ak vie aplikácia tolerovať dočasne nekonzistentné dáta, uprednostníme dostupnosť
  - Pre dôležité dáta zvolíme databázu zabezpečujúcu konzistentnosť
- Spoľahlivosť je nutnou podmienkou pre nasadenie rozsiahlych aplikácií - je dôležité vedieť:
  - Ktoré výpadky vie zvolená databáza detegovať
  - Ktoré vie automaticky tolerovať bez narušenia dát
  - Pri ktorých je potrebný zásah administrátora