Stĺpcové databázy

Zástupca, s ktorým budeme pracovať: Cassandra

EURÓPSKA ÚNIA
Európsky sociálny fond
Európsky fond regionálneho rozvoja

OPERAČNÝ PROGRAM
ĽUDSKÉ ZDROJE

MINISTERSTVO ŠKOLSTVA, VEDY, VÝSKUMU A ŠPORTU SLOVENSKEJ REPUBLIKY

Tento projekt sa realizuje vďaka podpore z Európskeho sociálneho fondu a Európskeho fondu regionálneho rozvoja v rámci Operačného programu Ľudské zdroje

www.minedu.sk www.employment.gov.sk/sk/esf/ www.itakademia.sk

Stĺpcové databázy – podľa článku o BigTable (2008)

Not Only SQL

- Nazývané aj databázy rodín stĺpcov, alebo databázy širokých riadkov
- Riadky majú potenciálne veľa stĺpcov asociovaných cez spoločné kľúče (ID riadkov)
- Rodina stĺpcov – stĺpce databázových riadkov, ktoré sú pre daný riadok načítavané aj ukladané spolu
  - Napr. pre každého pracovníka: osobné informácie, pracovné informácie, prístupové práva,...

Ústav informatiky
Prírodovedecká fakulta
UPJŠ v Košiciach

Riadky, stĺpce, rodiny stĺpcov – podľa BigTable

Not Only SQL

- Dáta tabuľky sú uložené v obrovskej utriedenej mape, kde kľúče sú:
  - (kľúč riadka, názov stĺpca, timestamp) alebo
  - (kľúč riadka, názov rodiny stĺpcov, názov stĺpca, timestamp)
- DB sa stará, aby záznamy so zhodným kľúčom riadka a rodinou stĺpcov boli uložené spolu
- Klasická relačná tabuľka sa najviac podobá na rodinu stĺpcov
  - Akurát, že každý riadok môže mať úplne iné stĺpce – schéma je nepovinná
  - Centrálne neregistrujeme, aké stĺpce môžu v riadkoch byť
  - Každý riadok si uloží iba vyplnené stĺpce (žiadne null)

Ústav informatiky
Prírodovedecká fakulta
UPJŠ v Košiciach

Stĺpcové databázy

- Google BigTable (neverejná)
- HBase
- Hypertable
- Accumulo
- Cassandra

- aktuálny zoznam: http://db-engines.com/en/ranking/wide+column+store

Cassandra

- Vytvorená Facebookom v Jave
- Má vlastný dopytovací jazyk CQL
  - Výrazovo jednoduchší ako SQL, ale veľmi podobný
    - CREATE, ALTER, DROP, INSERT, DELETE, UPDATE, SELECT, ...
    - Nepodporuje JOIN-y
- Podporuje map-reduce

Cassandra: vlastná cesta

- Žiadna rodina stĺpcov: tabuľka
  - Rozdiel oproti relačnej tabuľke je v tom, ako je uložená: distribuovaná hash mapa
- Tabuľky majú pevnú schému
  - Ak chceme pridať stĺpce meníme schému
- Rozdiely oproti relačným tabuľkám:
  - Pridané kolekcie (mapy, množiny a listy) a používateľom definované typy ako nové typy stĺpcov
    - Mapa predstavuje stĺpce v rodine stĺpcov v pôvodnom návrhu
  - Primárny kľúč určuje, ako sa budú riadky deliť do partícií
  - Ukladajú sa iba vyplnené stĺpce
  - Každá hodnota v tabuľke má časovú pečiatku vloženia
  - Staršie hodnoty sa nemažú z disku hneď
    - až pri pravidelnom upratovaní: compaction

Ústav informatiky
Prírodovedecká fakulta
UPJŠ v Košiciach

Primárny kľúč

- Každá tabuľka musí mať definovaný unikátny primárny kľúč, ktorý sa skladá z:
  - Kľúč partície – záznamy s rovnakým kľúčom partície sú uložené spolu (určuje na ktorom stroji sú dáta)
    - Záznamy s rôznym kľúčom partície, sú v rôznej partícii - môžu aj nemusia byť na rovnakom stroji
  - Klastrovacie stĺpce (voliteľné) – ak máme riadky s rovnakým kľúčom partície dodáme do primárneho kľúča ďalšie stĺpce, aby bol kľúč unique
    - V rámci partície sú riadky usporiadané podľa klastrovacích stĺpcov
- Vnútorná reprezentácia zhruba:
  - Map<byte[], SortedMap<klastr_stĺpce, riadok>>

kľúč partície

Primárny kľúč

- PRIMARY KEY (a): stĺpec a je kľúč partície, nemáme žiadne klastrovacie stĺpce
- PRIMARY KEY (a, b, c) : stĺpec a je kľúč partície. Stĺpce b a c sú klastrovacie stĺpce
- PRIMARY KEY ((a, b), c) : stĺpce a a b tvoria kľúč partície. Stĺpec c je klastrovací.

Distrubúcia partícií

- Každý riadok je priradený niektorému uzlu, ktorý ho bude spravovať
  - Hash stratégia: identifikátor partície (token) pre delenie je hash (MurmurHash alebo MD5) kľúča partície
  - Intervalová stratégia: token je priamo hodnota kľúča partície
  - Môžete si aj sami implementovať Partitioner
- Rozsah hodnôt tokenov sa uzavrie do kruhu – za najväčšou hodnotou je najmenšia
- Tento kruh sa rozdelí rovnomerne na intervaly a každý interval je pridelený jednému uzlu a jeho replikám
- Hranice intervalov sa môžu posúvať pri pridávaní nových dát (dáta sa migrujú)

Repikácia

- Peer-to-peer replikácia
- Určenie, ktoré uzly budú replikovať ktoré intervaly sa určuje na základe lokality (podľa IP adries, alebo nastavení)
  - Pre rýchlosť je dobré mať repliky pokope
  - Pre bezpečnosť uchovania dát - ďaleko od seba

Úrovne konzistencie

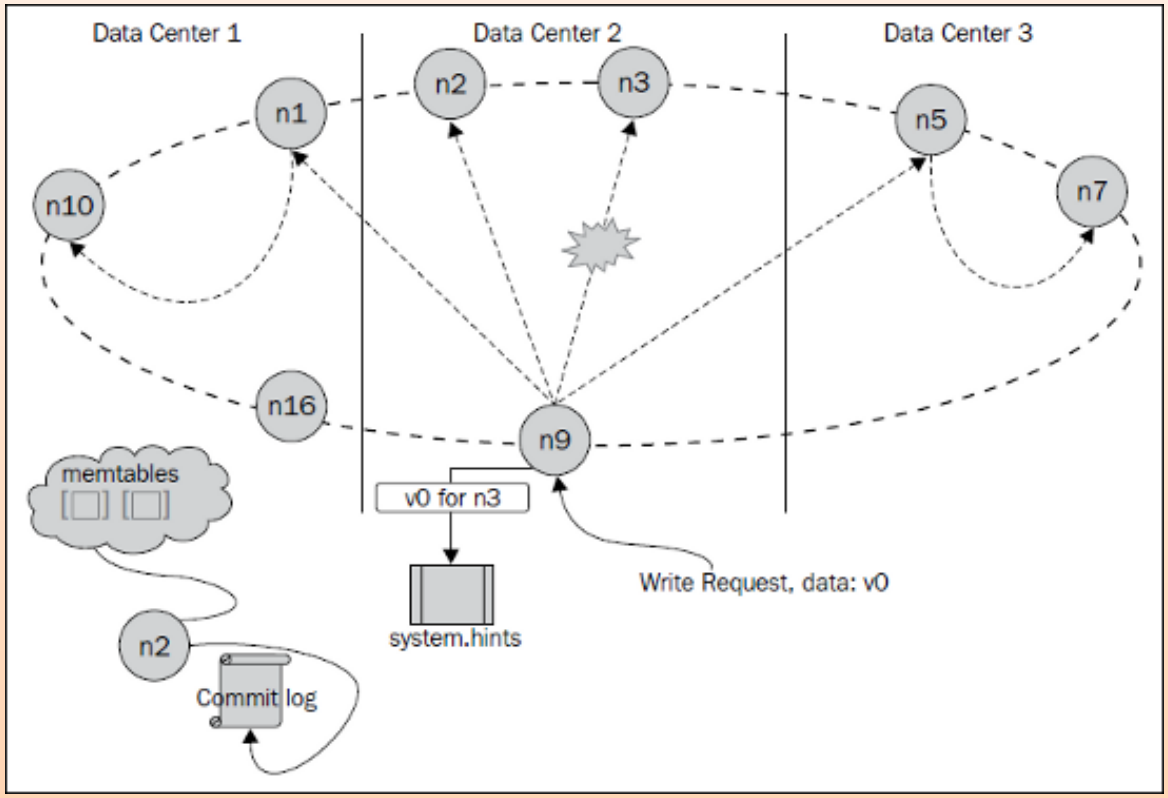
- Určenie koľko replík musí spolupracovať s koordinátorom pri zápise/čítaní
  - zapisuje sa vždy do všetkých kópií, ale do niektorých sa môže aj po ukončení operácie
- Možnosti:
  - ONE – replikácia je vypnutá,
  - TWO, THREE, ALL,
  - QUORUM – aspoň polovica
  - LOCAL_QUORUM (v rámci datacentra), EACH_QUORUM (v každom datacentre),
  - LOCAL_ONE, ANY – najnižšia konzistencia

Prídavné dáta k hodnotám

- Pridávajú sa ku každej hodnote atomického stĺpca a ku každej hodnote kolekcie
  - TTL (time-to-live) – po danom počte sekúnd sa hodnota zmaže
  - Timestamp – časová pečiatka poslednej zmeny
- Tieto hodnoty vieme získať selektom, žiaľ okrem kolekcií
  - SELECT nazov, **writetime(**nazov**)** from blogy WHERE autor='Anicka';

Spôsob uloženia

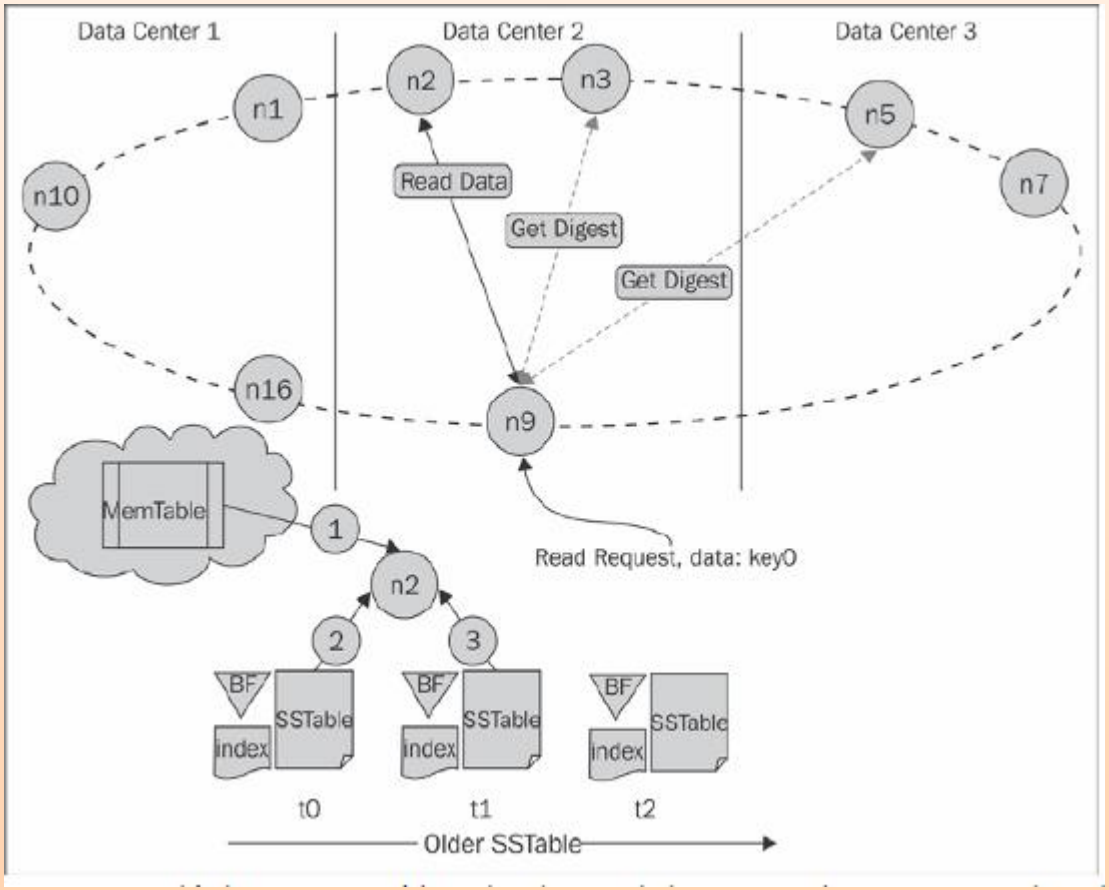
- Token identifikuje kde je partícia
- Súvislý podinterval tokenov je na jednom uzle
- Ak vieme kľúč partície, vieme, kde máme uložiť/hľadať riadok
- Dáta sa najprv zapíšu do Commit logu a potom uložia do memtable (utriedená mapa z kľúča na riadok)



- Ak je memtable pre danú tabuľku priveľká, zapíše sa na disk do read-only SSTable a vzniká nová memtable

Nájdenie riadku

- Nájdeme uzly z ktorých chceme čítať (závisí od úrovne konzistencie)
- Na uzle skúsime prehľadať príslušnú memtable
- Ak riadok nenájdeme, cez bloomfilter vylúčime tie SSTable, ktoré náš riadok určite nemajú
- Čítame zaradom z disku SSTable-y, z nich si prečítame index, či nenájdeme kľúč a ak áno vrátime riadok



Veľkosti záznamov

- Počet riadkov v partícii $2^{31}-1$ (2 miliardy) – počet záznamov v „lokálnom indexe“
- Počet stĺpcov v riadku maximálne 32768, odporúčaných je však iba 2 až 10 vyplnených
  - Bez chytrého uloženia na disku – neviem v konštantnom čase skočiť pre hodnotu i-teho stĺpca
- Hodnota klastrovacieho stĺpca max 64kB
- Hodnota obyčajného stĺpca v riadku max 2GB, odporúčaná veľkosť < 1MB
- List max $2^{31}-1$ záznamov, záznam max 64kB
- Set max $2^{31}-1$ záznamov, záznam max 64kB
- Mapa max $2^{31}-1$ záznamov, kľúč max 64kB, hodnota max 64kB

Ústav informatiky
Prírodovedecká fakulta
UPJŠ v Košiciach

Keyspace

- Súvisiace tabuľky (tá istá appka) ukladáme do spoločného priestoru kľúčov
- **CREATE KEYSPACE** moj_priestor **WITH** replication = {'class' : 'SimpleStrategy', 'replication_factor' : 2};
  - Každý riadok je uložený na 2 uzloch
- **USE** moj_priestor;

Definícia tabuľky a indexu

- CREATE TABLE blogy (
  autor text,
  zverejnene timestamp,
  blog text,
  komentare map<text, text>,
  PRIMARY KEY (autor, zverejnene)
  );
- CREATE INDEX ON blogy(keys(komentare));

Dátové typy: https://docs.datastax.com/en/cql/3.3/cql/cql_reference/cql_data_types_c.html

Insert a select

- Vkladanie záznamov:
  - INSERT INTO blogy (autor, zverejnene, blog) VALUES ('Jano',toTimestamp(now()),'Som múdry') USING TTL 900;
  - ...
- Získavanie záznamov:
  - SELECT * FROM blogy WHERE autor='Jano' AND zverejnene < '2018-02-28';
  - SELECT autor, count(*) AS pocet_blogov FROM blogy GROUP BY autor;

Zmaže sa po 15 minútach

Select

- Podporuje:
  - DISTINCT
  - GROUP BY iba podľa stĺpcov
  - ORDER BY iba podľa stĺpcov
  - LIMIT, aj PER PARTITION LIMIT
  - COUNT(*), WRITETIME(stĺpec), TTL(stĺpec)
- Obmedzenia:
  - Kľúče partície vieme filtrovať iba cez rovnosť s hodnotou (nie interval)
  - Where podmienka podporuje iba AND operátor
  - Čítame iba jednu tabuľku (žiadne joiny)

UPDATE

- UPDATE blogy SET komentare=
  {'Fero':'no urcite','Patrik':'neverime'}
  WHERE autor='Jano' AND
  zverejnene='2018-02-03T14:05:25.818+0000';
- UPDATE blogy SET
  komentare= komentare + {'Jana':'je to tak'}
  WHERE autor='Jano' AND
  zverejnene='2018-02-03T14:05:25.818+0000';

Použitie indexu

- Ak chceme získavať dáta nie podľa prefixu primárneho kľúča, musíme použiť prídavný index
- Nech primárny kľúč je (autor, zverejnene) a máme index na kľúče komentárov
- SELECT * FROM blogy WHERE komentare CONTAINS KEY ‘Fero' AND zverejnene > '2018-02-01' ALLOW FILTERING;
  - Musíme explicitne povoliť použitie kľúča na filtrovanie aj podľa ďalších stĺpcov

Podmienené operácie

- Garantovane atomické (odľahčená transakcia)
- Efektivita je nízka, mali by sa používať iba zriedka
- INSERT INTO … IF NOT EXISTS;
- UPDATE … SET … IF podmienka;
- UPDATE … SET … IF EXISTS;

Otázky?