Nem-relációs adatbáziskezelés

Gajdos Sándor

2018. május 8.

Motiváció

- A feladat pontosabb értelmezése: Hogyan lehet adatokat minél nagyobb hatékonysággal kezelni?
- Mit jelent az „adatkezelés”?
- Meddig lehet a rendszer funkcionalitását egyszerűsíteni/csökkenteni?
- Mitől lesz ez adatbáziskezelés?

Történelmi előzmények

- Hierarchikus adatbáziskezelés (IBM, 1960-)
- Hálós adatbáziskezelés (~1970-85)
- Ami mindkettőben volt:
  - Imperatív lekérdezések
    - A programozó dolgozik, nem a számítógép
    - Deklaratív lekérdezések hiánya, nem kell lekérdezés optimalizálás
  - Kapcsolatok megvalósítása direkt linkekkel
    - V.ö: relációsnál adatok értékegyezése alapján – keresés
- Következmény: eleve lényegesen gyorsabbak lehetnek.

A legnagyobb weboldalak (2011.)

Weboldal	Terhelés
Google	24500 tablet szerver, 1.2 millió adatbázis lekérdezés másodpercenként, 16Gb/s lekérdezés
Facebook	Inbox: 100 TB, 150 gépes kaszter Adattárház: 15 Petabyte adat, 1400 gép, 11200 CPU
Youtube	
Microsoft Live, Bing	
Yahoo!	92 Petabyte adat, a legnagyobb lekérdezés 10ezer gépen fut párhuzamosan 73 óráig
Twitter	Adatbázis növekedés 7 TB naponta, 2+ PB évente
Wikipedia	
BBC	
Myspace	
Amazon	

A legnagyobb weboldalak adatbázisai

Weboldal	Főbb adatbázismotor	Adatbázis típusa
Google	GFS, Googe BigTable	Columnar NoSQL
Facebook	Cassandra, Hadoop/HIVE	Columnar NoSQL
Youtube	Memcached	Key-Value
Microsoft Live, Bing	Azure	Tuple store, RDBMS
Yahoo!	Hadoop, PNUTS	Columnar NoSQL
Twitter	FlockDB, Cassandra, Hadoop/Hbase	Graph, Columnar NoSQL
Wikipedia	Memcached, Flatfile, MySQL	Key-Value, Flat file, RDBMS
BBC	CouchDB	Document
Myspace	Aster Data nCuster	MPP RDBMS + MapReduce
Amazon	Amazon Dynamo	Columnar NoSQL

Mi van a relációson túl?

- Hierarchikus DB
- Multidimenziós DB
- Document store
- Gráf DB
- Key/value store (on disk, in RAM)
- Object DB
- ...

Összefoglaló nevük: NoSQL adatbáziskezelők

NoSQL

- Relációs rendszerek gyengéi
  - Sok dokumentum indexelése
  - Nagyforgalmú weboldalak kiszolgálása
  - Adatstream-ek szolgáltatása
- Általános jellemzők
  - Gyenge konzisztenciagaranciák (eventual consistency vagy egyadatelemes tranzakciók)
  - Elosztott architektúra

Hierarchikus adatbázisok

- Az adatok faszerű struktúrákban
- Rekordorientált szemlélet
- 1:N kapcsolatok natív leképezése
- Imperatív lekérdezhetőség
- IMS (IBM), Windows registry (Microsoft)

Multidimenziós adatbázisok

- N-dimenziós tömbök
- Cellák: tényadatok, melyeket a dimenziók koordinátáival címezhetünk
- Segédstruktúrák (indexek)
- Nem tévesztendő össze a relációs alapú dimenzióssal
- Hyperion Assbase, Cognos, Oracle Express, Microsoft,...

Dokumentum adatbázisok

- Natív OODB (vagy lehet réteg egy relációs felett)
- Minden rekord egy dokumentum, aminek tetszőleges számú, nevű és méretű mezője lehet
- Nincs üres mező, a mezők többszörös adatelemeket is tartalmazhatnak
- hasonló „eredmény” érhető el pl. XML-lel -> minden XML adatbázis dokumentum DB is
- Egyszerű használat, könnyű programozhatóság
- CouchDB, IBM Lotus Notes, MongoDB, RavenDB, XML databases: MarkLogic Server, eXist

Gráf adatbázisok (ld. korábbi ea)

- Az adattárolást gráfstruktúrák (csomópontok, élek + attribútumok) valósítják meg
- A gráf lehet tetszőleges vagy spec. (ld. hálós adatbázisok)
- OO alkalmazások esetén (is) hatékonyabb
- Gyakran változó séma esetén
- IDMS (hálós), Neo4j, AllegroGraph, Core Data, FlockDB

Key/value adatbázisok

- A legegyszerűbb NoSQL DB
- Kulcs+adat (akár blob, a tartalma közömbös)
- Az alkalmazás felelőssége az adat értelmezése
- Nincs hagyományos séma
- Hozzáférés gyakran csak a kulcsértéken kereszül, igen jól optimalizálható
- Természetesen az adat is indexálható
- A tárolás lehet diszken, memóriában, rendezve vagy anélkül, módosítást kizárva (ld. CDB),...
- BigTable, CDB, Memcachedb, Redis, SimpleDB, Tokyo Cabinet, Tuple space, NMDB, Memcachedb, Berkeley DB,...

Mit lehet feladni a sebességért?

- Elvileg mindent, ami egy DBMS-t jellemez:
  - Atomicity
  - Consistency
  - Isolation
  - Durability
- A relációs modell előnyeit is:
  - Kényelem
  - Változatos lekérdezések hatékonyan

Mindegyik jelentősen megkönnyíti az alkalmazás-fejlesztők munkáját is.

A sörfőző sapka-elmélete

- Eric Brewer's CAP theory (2000):
  - **Consistency:** valójában atomicity.
  - **Availability:** minden művelet a tervezett eredménnyel fejeződik be (nem pl. hibaüzenettel).
  - **Partition tolerance:** Semmilyen részleges hiba nem okozhatja, hogy a rendszer helytelen választ ad

közül egyszerre csak kettő teljesülhet elosztott környezetben.

Az elméletből formális bizonyítás után tétel lett (2002)

- Következmény: horizontálisan skálázott rendszerek vagy konzisztensen működnek, vagy a rendelkezésreállásuk biztosított.
- Alkalmazás: pl. Amazon, EBay, Twitter

Értelmezés

- Példa:
  - Két-node-os adatbázis klaszter
  - 2PC
  - Mindkét node szükséges egy tranzakcióhoz
- Konzisztencia OK, de hogyan értinti a rendelkezésreállást?
- Rendelkezésreállás:
  - A szükséges komponensek rendelkezésreállásának szorzata
  - A használható, de nem használt komponensek nem csökkentik a rendelkezésreállást
- Ha mindkét node-nak 99% a rendelkezésreállása, akkor a tranzakcióé már csak 98%

Igen nagy tranzakciós terhelések

- Új gondolkodásmód kell az erőforrások kezeléséről
- ACID problémás, ha a terhelést több node között osztjuk meg
- Műveletek szétcsatolása javítja a rendelkezésreállást és skálázhatóságot a konzisztencia rovására

Egy ACID alternatíva: BASE

- BASE=Basically Available, Soft state, Eventually consistent (2008)
- ACID: pessszimista, BASE: optimista, elfogadja a DB „képlékeny” konzisztenciáját a tranzakció végén
- Következmény:
  - Kezelhető
  - Magas szintű horizontális skálázhatóság
  - Rendelkezésreállás: részleges hibák megtűrése

Példa:

- Sajátos séma tranzakciókhoz

```
Begin transaction
  Insert into transaction(xid, seller_id, buyer_id, amount);
  Update user set amt_sold=amt_sold+$amount where id=$seller_id;
  Update user set amt_bought=amount_bought+$amount where id=$buyer_id;
End transaction
```

- ACID-stílusú megoldás

```
Begin transaction
  Insert into transaction(id, seller_id, buyer_id, amount);
End transaction
Begin transaction
  Update user set amt_sold=amt_sold+$amount where id=$seller_id;
  Update user set amt_bought=amount_bought+$amount
        where id=$buyer_id;
End transaction
```

- Konzisztencia lazítása

Példa 1: Directory „databases” and LDAP I.

- Directory:
  - Bejegyzések (entry) hierarchikus struktúrában
  - A bejegyzéseknek attribútumaik lehetnek, melyeknek nevük van és értékük
  - Minden bejegyzésnek van egyedi neve (DN), ami tip. szülő azonosító + alkalmas attrib. értékek
- LDAP: a kezelését biztosító protokoll.
- Kezdetben „Lightweight Directory Browsing Protocol”, ma:
  - Search — search for and/or retrieve directory entries
  - Compare — test if a named entry contains a given attribute value
  - Add a new entry
  - Delete an entry
  - Modify an entry
  - Modify Distinguished Name (DN) — move or rename an entry
  - Abandon — abort a previous request
  - Extended Operation — generic operation used to define other operations

Példa 1: Directory „databases” and LDAP II.

- Adattárolás módja nem specifikált:
  - Flat file
  - Adatbázis
  - Gateway másik szerverhez
- Általános access protokoll szolgáltatásokhoz (ld. SQL adatbázishoz!)
- Jellegzetes felhasználás: authentikáció
- Jólismert hierarchikus adatbázis: MS Windows Registry

Példa 2: MapReduce és Google File System

- A MapReduce igényeihez nem volt megfelelő fájlrendszer.
- Fő probléma: az adatokat igen nagy fájlokban (~100 GB) kell tárolni, fürtözötten
- nagyon ritkán kell csak
  - törölni,
  - felülírni
  - csökkenteni a méretüket,
- leginkább csak hozzáírnak (append).
- Google kifejlesztett egy sajátot...

BigTable DBMS

- A GFS nagy file-jainak kezelésére kihegyezett
- Redundáns, tömörített, nem relációs
- Elosztott, igen jól skálázható
- Új gépek hozzáadása automatikus, újrakonfig nélkül
- „multidimensional sorted map”
- Ötvözi a sor-orientált és oszloporientált tárolást
- Keresés:
  - Volt-e nemrég ilyen lekérdezés?
  - Ha nem, akkor keresés az indexben
  - A robotok által meglátogatott oldalakat 20 MapReduce folyamat dolgozza fel (húszféleképpen), naponta 20-25 PB-ot (2011).

Irodalom:

- Pritchett: BASE: An Acid Alternative, ACM Queue, May/June 2008.
- Brewer’s keynote speech: http://www.cs.berkeley.edu/~brewer/cs262b-2004/PODC-keynote.pdf