Műszaki Tudományos Közlemények vol. 9. (2018) 83–86.
https://doi.org/10.33895/mtk-2018.09.16,
https://eda.eme.ro/handle/10598/30297
Angol nyelven: https://doi.org/10.33894/mtk-2018.09.16

MODERN, NOSQL ADATBÁZISOK MŰKÖDÉSÉNEK AZ ÁTTEKINTÉSE. ESETTANULMÁNY: APACHE CASSANDRA

MODERN, NOSQL DATABASE OF THE OPERATION OVERVIEW. CASE STUDY: APACHE CASSANDRA

Ferencz Katalin

Sapientia EMTE Marosvásárhelyi Kar, Villamosmérnöki tanszék, Számítástechnika szak, 540485, Románia, Marosvásárhely/Koronka, 1/C szám; ferenczkatalin@yahoo.com

Abstract

The spread of IoT devices make possible to collect an enormous amount of data. Traditional SQL (structured query language) database management systems are not suitable for storing this type of data. For this task, distributed database management systems are the most appropriate. Apache Cassandra is an open source, distributed database server software that stores large amounts of data on low-coast servers, providing high availability. The Cassandra uses the gossip protocol to exchange information between the distributed servers. The query language used is the CQL.
In this paper I present an alternative solution to traditional SQL-based database management systems, the so called NoSQL type database management systems, I summarize the main types of these systems and I provide a detailed description of the Apache Cassandra open source distributed database server installation, configuration and operation.

***Keywords**: database, NoSQL, distributed, Cassandra, CQL.*

Összefoglalás

Az IoT eszközök elterjedése hatalmas mennyiségű adatbegyűjtésre ad lehetőséget. A hagyományos SQL (Structured Query Language) alapú adatbáziskezelő rendszerek nem megfelelőek az ilyen típusú adatok tárolására. Ehhez a feladathoz az osztott adatbáziskezelő rendszerek a legmegfelelőbbek. Az Apache Cassandra egy nyílt forráskódú elosztott adatbázisszerver szoftver, amelyet arra terveztek, hogy nagy mennyiségű adatot tároljon alacsony költségű szervereken, magas rendelkezésre állást szolgáltatva. A Cassandra gossip protokollt használ arra, hogy az osztott rendszert alkotó szerverek információt osszanak meg egymással. Lekérdezőnyelve a CQL.
A jelen dolgozatban bemutatom a hagyományos, SQL alapú adatbáziskezelő rendszerek egy alternatíváját, a NoSQL típusú adatbáziskezelő rendszereket, összefoglalom ezeknek a rendszereknek a fő típusait illetve részletesen kitérek az Apache Cassandra nyílt forráskódú elosztott adatbázisszerver telepítésére, konfigurálására és működésére.

***Kulcsszavak:** adatbázis, NoSQL, elosztott, Cassandra, CQL.*

1. Bevezetés

A mai gyorsan fejlődő világunk legfontosabb jellemzője, hogy a minket körülvevő környezetet folyamatosan megfigyeljük, információkat gyűjtünk, tárolunk el és dolgozunk fel. A környezetünk feltérképezésére különböző intelligens szenzorokat és

Internetre csatlakoztatott eszközöket használunk. Ezek folyamatosan küldenek adatokat, melyeket a felhasználók szemszögéből "valahol" a "felhőben" tárolunk. Az ilyen alkalmazásokat fejlesztő szakemberek számára azonban jelentős kihívást jelent a "felhőben" való tárolás hatékony megtervezése és gyakorlati megvalósítása.

A kutatás során fő célunk beüzemelni egy több számítógépből álló, osztott adatbázis rendszert és naplózni különféle IoT eszközök szenzoradatait. Gyakorlatban is igazolni szeretnénk, hogy egy olyan informatikai rendszer fejleszthető, amely alkalmas a nagy mennyiségű adatok tárolására és megfelelő feldolgozására.

2. Adatbáziskezelő rendszerek

Az IoT gyors fejlődési szakaszának indulása előtt a legelterjedtebb adatbáziskezelő rendszerek SQL alapú rendszerek voltak. Viszont ez a fejlődés magával hozta az adatbázisok továbbfejlesztését is. Így a 2000-es évek elején kezdték fejleszteni a NoSQL (Not only SQL) lekérdező nyelvet, mely az SQL alapú adatbázisoktól az adatok tárolásában és lekérdezésében különbözik. A NoSQL főként osztott adatbázisnak nevezhető, és megoldja az SQL adatbázisok azon problémáját, hogy az adatokat csak egy számítógépen lehet tárolni, így lehetőséget ad a felhőben való tárolásra. Nagyon fontos tulajdonsága, hogy dinamikus sémája van a struktúrálatlan adatok számára és horizontálisan skálázható, ami azt jelenti, hogy újabb gépek és szerverek hozzákapcsolásával nagyobb forgalmat is akadálymentesen tudnak kezelni.

A NoSQL adatbázisok esetében az adatok tárolására négy különböző módot lehet alkalmazni:

- **kulcs/érték tárolók**, melyek a kulcsokat és a hozzájuk rendelt értékeket tárolják;
- **dokumentumtárolók**, melyekben félig strukturált adatokat tudunk tárolni;
- **oszloptárolók**, melyek a táblán belül egymás után tárolják a sorokat;
- **gráf tárolók**, melyek esetében az adatok gráfként jól modellezhetőek és az adatok határozatlan számú kapcsolattal vannak összekötve.

Az internet általános elterjedésével, a szenzorok megjelenésével a Big Data korát éljük, vagyis nagyszámú felhasználó hozhat létre és férhet hozzá hatalmas nagy mennyiségű információhoz. Emiatt a Big Data alkalmazásoknak három kihívása van, melyre a NoSQL próbálja megadni a megoldást. Ezt a probléma hármast 3V-nek is szokták nevezni, vagyis: volume – velocity – variety.

1. táblázat. *A legelterjedtebb NoSQL típusú adatbáziskezelő rendszerek [1]*

Név	Adatmodell	Használja
Cassandra	Kulcs-érték és oszlopos adatmodell hibrid; Cassandra lekérdez-őnyelv;	CERN, eBay, Netflix, GitHub
Redis	Kulcs-érték pár, összetett típusok	Twitter, GitHub, Flickr, Stack Overflow
Voldemort	Összetett kulcs-érték objektumok	LinkedIn
DynamoDB	Dokumentum és kulcs-érték modell	Amazon, BMW
Allegro Graph	RDF-Resource Description Framework, gráf adatbázis	Stanford, IBM, Ford, Siemens, NASA
Memcached	Nincs replikáció és perzisztencia	Wikipédia

A NoSQL típusú rendszerek lemondanak az azonnali konzisztenciáról (következetesség, ellentmondás-mentesség), mert az a fő koncepciójuk, hogy több számítógép legyen egy adatbázishoz hozzárendelve. Ha

meghibásodik valamelyik szerver, akkor átkapcsolnak egy másik szerverre, így maximális rendelkezésre állás van biztosítva. A CAP teóriával próbálják leírni a NoSQL tulajdonságait: consistency – availability – partition tolerance.[2]

A napjainkban legelterjedtebb NoSQL típusú adatbáziskezelő rendszereket az **1. táblázat**ban van összefoglalva.

A táblázatba foglalt adatbáziskezelő rendszerek közül a kutatásom során az Apache Cassandra rendszert használjuk, mivel széles körben elterjedt, nyílt forráskódú, jól dokumentált és számos API van már hozzá fejlesztve.

3. Apache Cassandra

Az Apache Cassandra egy C/C++ nyelven fejlesztett, nyílt forráskódú elosztott adatbázisszerver szoftver, melynek az a célja, hogy nagy mennyiségű adatot tároljon alacsony költségű szervereken, magas rendelkezésre állást biztosítva. A Cassandra fejlesztését az Amazon DynamoDB fejlesztőjeként ismert Avinash Lakshman és Prashant Malik kezdte a Facebook-nál, így nem meglepő, hogy főleg az Amazon DynamoDB architektúráját és elképzelését követi. 2009. márciustól az Apache inkubátor tagja. [3]

A Cassandra támogatja a több adatközponton működő fürtök (klaszter) működését. A struktúrájában használt adatmodell a kulcs-érték és az oszlopos adatmodell hibridje. Két kulcs pontja van egy Cassandra rendszernek: az adat partíció és az adat modell.

Az adatbiztonság miatt az adatok több gépen is tárolásra kerülnek, így az egyik gépről a másikra át kell szinkronizálni az adatokat.

Az Apache Cassandra fő tulajdonságai:

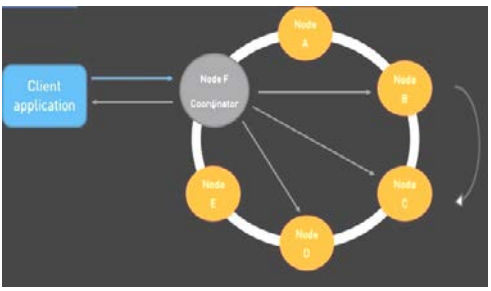
- Elosztott rendszer: több számítógép van a hálózatban, és ezek össze vannak kötve.
- Decentralizált: nagyon fontos tulajdonsága a Cassandrának, mivel ez különbözteti meg a többi adatbáziskezelőtől. Ennek értelmében nem master-slave alapú, nincs Single Point Failure, vagyis ha egy része megsérül, a teljes rendszer nem omlik össze és elérhetőek lesznek az adatok. Ez annak köszönhető, hogy gyűrű topológia jellemzi, adatközpontok vannak, és egy adatközponton belül minden csomópont (node) azonos értékű.
- Hibatűrő: az adatok legalább 3 számítógépen tárolva vannak, így nincs adatvesztés.
- Magas rendelkezésre állás.
- Rugalmasan skálázható: ha megnövekszik a terhelés egy rendszeren és bővíteni kell, ezt észrevétlenül meg tudjuk oldani, úgy, hogy új node-ot adunk hozzá.
- Lineárisan skálázható: ha a node-ok számát megduplázzuk, az adatbáziskezelő áteresztőképessége is megduplázódik.
- Hangolható konzisztencia: a rendelkezésre állás rovására lehet erősíteni a konzisztenciát.

Egy Cassandra klaszter kialakítható több fizikai számítógépből vagy több virtuális gépből, melyeket hálózatba kapcsolunk. Mivel a hálózatban más számítógépek is lehetnek, a klaszter tagjai a rajtuk megtalálható konfigurációs állomány beállításai alapján fogják tudni, hogy melyik node tartozik a klaszterbe. A klaszter tagjai Gossip protokollt használ az egymással való kommunikáció lebonyolítására. A klaszter konfigurációjának a kialakítására egy másik protokollt is használ: ez a Snitch melynek segítségével megadható, hogy egy adott node a klaszterből melyik adatközponthoz és rack-hez tartozik.

Miután a Gossip és a Snitch protokoll beállításainak segítségével felépül a klaszter, annak érdekében, hogy a rendszer hibatűrő és magas rendelkezésre álló legyen, definiálni kell az adatok elhelyezkedését, vagyis meg kell határozzuk, hogy egy adatok melyik node-on vagy node-okon legyenek eltárolva. A magas rendelkezésre állás

érdekében az szükséges, hogy egy adat több node-on is el legyen tárolva.

A hibatűrés és a magas rendelkezésre állás abból következik, hogy replikáljuk az adatokat, általában minimum három replika kell legyen. Ez azt jelenti, hogy kiszámoltunk egy adott token-t az adott partíciós kulcshoz, ezzel meghatároztuk, hogy melyik node fogja azt a partíciót tárolni, viszont a gyűrűben az óramutató járásával megegyező következő két (ha a replication_factor: 3) node is fogja tárolni ezt a partíciót.



1. ábra. *Cassandra replikáció*

Az ábrán látható, hogy a kliens ír egy adatot az adatközpontba és mivel az adott adatért a B node a felelős, így a mellette lévő 2 node is tárolni fogja. Megfigyelhetjük, hogy a kliens alkalmazás az F node-al kommunikál, így ő lesz most a koordinátor, de bármelyik másik node-al kommunikálhatna, abban az esetben az a node válna koordinátorrá, mivel a node-ok egyenrangúak.

Egy másik fontos tulajdonsága a Cassandrának, hogy tudjuk szabályozni a konzisztencia szintet, mely segítségével azt határozzuk meg, hogy a replikák közül hány kell sikeres legyen írásnál vagy olvasásnál. Alapértelmezetten három értéket vehet fel:

- ONE: elegendő, ha egy node visszajelez;
- ALL: minden node-nak választ kell adni;
- QUORUM: a többség választ kell adjon.

Azonnali konzisztencia abban az esetben érhető el, ha az írás és olvasás konzisztencia szintek összege nagyobb, mint a replikációs faktor.

A Cassandra Query Language (CQL) az elsődleges nyelv, amely segítségével kommunikálni tudunk az Apache Cassandra adatbázissal. A Cassandra-val való interakció legegyszerűbb módja a CQL shell, a **cqlsh**. Ennek használatával kulcstereket és táblákat hozhatunk létre, beszúrhatunk új információkat és lekérdezhetünk táblákat, valamint még sok más műveletet végrehajtható.

A CQL számos beépített adattípust kínál, beleértve a gyűjteménytípusokat is. Ezeken kívül lehetőséget ad a felhasználónak saját, egyedi adattípusok létrehozására is.

4. Összegzés

Munkánk eredményeként egy olyan ismeretanyagot állítottunk össze, amely segítségével könnyen meg lehet tanulni a CQL lekérdezőnyelvet, meg lehet érteni Cassandra NoSQL adatbázis működését, illetve ki lehet alakítani egy néhány node-ból álló Cassandra klasztert. Ezt az oktatási célokra kidolgozott ismeretanyagot felhasználva könnyedén elvégezhetőek a szokások adatbázis műveletek: létrehozás, beírás, lekérdezés, módosítás, törlés.

Szakirodalmi hivatkozások

[1] N.Q.Mehmood, R.Culmone, L. Mostarda: *Modeling temporal aspects of sensor data for MongoDB NoSQL database*, J Big Data (2017) 4:8
[2] https://en.wikipedia.org/wiki/NoSQL
[3] http://cassandra.apache.org