# Spark Streaming and Cloud

feldolgozzák a nagy mennyiségű adatot. A NoSQL célja közel valós időben válaszokat nyújtani, míg a Big Data esetében a fő szempont az, hogy bármilyen adatra és bármekkora adathalmazra képesek legyünk lekérdezéseket végrehajtani.

A batch processing azt jelenti, hogy hosszú idő alatt összegyűjtött adatokat feldolgozzuk. Ez a Big Data nyelvekben használatos, mivel nem valós időben futnak, hanem várakozniuk kell a következő adatbatch-re. A streaming viszont azt jelenti, hogy az adatokat azok létrehozása után azonnal feldolgozzuk. A streamelésnek az a problémája, hogy nemcsak tudnunk kell, milyen adatot kapunk, hanem azt is, hogy eddig milyen adatokkal rendelkezünk, ami nem triviális feladat. Egy másik probléma az, hogy az esemény bekövetkezésének ideje nem feltétlenül azonos a feldolgozás sebességével, így késleltetve kaphatunk információkat, amelyek már a múltban történtek. Éppen ezek a tényezők miatt a streamelést hosszú ideig megbízhatatlannak tartották.sokáig.

A lambda architektúra arra törekszik, hogy egyensúlyt teremtsen a batch processing és a streaming előnyei között. Ennek érdekében a batch feldolgozást használja az adatok tárolására, hogy azok átláthatóak legyenek, miközben valós időben feldolgozza a streamből érkező adatokat.

A kappa architektúra egy szoftverarchitektúra, amely az adatokat a streamből dolgozza fel. Az alapelve az, hogy a stream adatait feldolgozza és kombinálja a batch processinggel egy verem (stack) segítségével. Az összes adatot streamként kezeli, és egy analízis motort alkalmaz, ami egyszerűbbé teszi a kódolást és karbantartást. Az architektúra alapelve az "Egyszerűség a kulcs" (Keep it short and simple - KISS), és nem alkalmaz változtatható adattípusokat. Ez azt jelenti, hogy az adatok állandóak maradnak, és az igényeink szerint változtathatjuk az adatokat.

A kérdés az, hogy hogyan dolgozzuk fel az eseményeket (events), erre a célra segítséget nyújt az Apache Kafka, amely egy szétosztott adattároló rendszer, és valós időben történő feldolgozást tesz lehetővé. A Kafka csatornákból áll, ahol különböző témákhoz kapcsolódó üzeneteket lehet cserélni, és ezeket az üzeneteket a felhasználók megkaphatják. Minden csatorna és fogyasztó külön kezelést kap. Az egyes üzenetek tárolási időtartamát be lehet állítani, és később további fogyasztókat is hozzá lehet adni. Nem jelent problémát, ha az egyik fogyasztó gyorsabban dolgozza fel az üzeneteket, mint a másik, mert mindegyik a saját tempójában haladhat. A Kafka klusztereket brókereknek nevezzük, és képesek a témákat partíciókra osztani, hasonlóan a NoSQL rendszerekhez. A fogyasztókkal is számolni kell, és létre lehet hozni fogyasztócsoportokat, amelyek megosztják az üzeneteket a fogyasztók között.

A másik kérdés az, hogy hogyan dolgozzuk fel a nagy mennyiségű adatot. Ebben segítséget nyújt nekünk a Spark streaming API-ja. A streaming folyamat során a cél az, hogy átalakítsuk a bemeneti adatokat - például fájlok, Kafka, socket - kimeneti adatokká, például fájlok, Kafka, Foreach vagy konzol. Egy ilyen átalakítást triggerek indítanak be, amelyek lehetnek: mikroszakaszok alapján (kisebb adatmennyiségek), időintervallum alapján vagy egyszeri feldolgozás, ami pontosan ugyanaz, mint a batch processing. A kimenetnél vagy hozzáadhatunk a már létrehozott adatokhoz, vagy felülírhatjuk azokat, amelyek már léteznek. A Spark esetén az adatfolyamot úgy képzelhetjük el, mint egy táblát, amelynek nincs meghatározott mérete, vagyis folyamatosan hozzáadhatunk sorokat, és minden új adat egy új sort eredményez. Valójában egy végtelen nagy adattáblaként lehet elképzelni. A Spark képes az adatfolyam olvasására és írására is. A legtöbb transzformáció elérhető a streaming folyamat során, például aggregációk és join műveletek.

Az események időbeli eloszlását általában megtaláljuk az adatokban, és időintervallumok (time window) alapján csoportosítjuk őket. Különböző típusú időablakokat különböztetünk meg:

- Tumble window: Ez az ablaktípus adatainkat rendszeresen, például 5 perces időközönként vizsgálja. Minden 5 percben új ablak indul, és az abban az időintervallumban eső eseményeket kezeli.

- Sliding window: Ebben az esetben az ablakot például 10 perces időintervallumokban tesszük. Az ablakok azonban egymás mellett mozognak, így egy adott pillanatban több ablak is aktív lehet. Például az átlagolási műveleteknél gyakran alkalmazzák.

- Session window: Ez az ablaktípus nem rendelkezik előre meghatározott időtartammal. Az ablak akkor kezdődik, amikor érkezik egy esemény, és ha egy ideig nem érkezik újabb esemény, az ablak megszűnik. Amikor új esemény érkezik, egy új ablak indul. A későbbi események feldolgozását a vízjel (watermark) határozza meg, ami nem technikai részlet, hanem inkább üzleti megközelítést igényel.

Ezek az időablakok lehetővé teszik az események csoportosítását és feldolgozását az időbeli szempontok alapján.

Felhőalapú számítástechnika

A felhőalapú számítástechnika olyan számítási szolgáltatásokat jelent, amelyeket az interneten keresztül érhetünk el, és gyakorlatilag bérelhetünk. Különböző típusai vannak:

- Infrastruktúra mint szolgáltatás (IaaS): Ez a típus lehetővé teszi a szerverek, hálózatok és adatközpontok bérlését. Az IaaS segítségével rugalmasan skálázható infrastruktúrát alakíthatunk ki az üzleti igényeinkhez igazodva.

- Szoftver mint szolgáltatás (SaaS): Ez a típus lehetővé teszi a szoftverek bérlését. A SaaS modellben a szoftvereket a felhőben futtatják, és a felhasználók távolról érhetik el őket böngészőn vagy kliensalkalmazásokon keresztül.

- Szervertelen (Serverless): Ez a típus lehetővé teszi olyan alkalmazások tervezését, amelyeknél nem kell aggódni az infrastruktúra miatt. A szervertelen architektúrában az alkalmazásokat részletekbe tagolva, különböző funkciókként (function) hozzuk létre, amelyeket a felhőhöz kötve futtatnak.

A felhőalapú számítástechnika lehetővé teszi az erőforrások rugalmas kihasználását és a költségek optimalizálását, valamint könnyű hozzáférést biztosít a szolgáltatásokhoz és alkalmazásokhoz az interneten keresztül.

A felhőalapú számítástechnika előnyei közé tartozik, hogy rendkívül skálázható, könnyen bővíthető, és világszerte hozzáférhető. Emellett megbízható és erős védelemmel rendelkezik. Amikor a felhőre gondolunk, gyakran először a Hadoopra gondolunk, amely rendelkezésre áll Hadoop mint szolgáltatásként. Ennek előnye, hogy nem kell a fenntartásával foglalkozni, könnyen létrehozhatunk több klusztert, és csak annyit kell fizetnünk, amennyit valóban használunk. Ha szükséges, a rendszer automatikusan növelhető. Fontos megjegyezni azonban, hogy bizonyos korlátokkal kell számolni a konfiguráció terén, és néhány országban különböző szabályozások érvényesek lehetnek. Emellett hosszú távú használat esetén figyelembe kell venni az árakat is.

A felhőben lehetőségünk van adatok tárolására, amely egy fájlrendszerként jelenik meg, hasonlóan a Hadoophoz. Ez azt jelenti, hogy az adatok szét vannak osztva és többszörösen replikálva, és rendkívül költséghatékony és 99,999999999%-ban megbízható. Ezért gyakran használják ezt a felhő tárolót a HDFS helyett, mivel az adatok nem kötődnek egyetlen számítógéphez, és több kluszter tudja ugyanazzal az adattal dolgozni. Az adatlokalitástól való elmozdulásra törekszünk, a hálózatok egyre gyorsabbá válásának köszönhetően, és bár a HDFS még mindig gyorsabb, a felhő előnyei tagadhatatlanok és megérnek egy kis sebességcsökkenést. Már úgy tekintünk rá, mint egy olyan kódrészletre, amely átveszi az infrastruktúra szerepét: klusztereket hoz létre, szoftvereket telepít, és a kluszter feladatainak befejezése után mentés után leállítja azokat. Pénzt is megtakaríthatunk azáltal, hogy "spot instance"-okat használunk, ami azt jelenti, hogy akkor futtatjuk a lekérdezéseket, amikor a szolgáltatások alig vannak kihasználva, vagy ha nincs sürgős szükség a befejezésükre.