BÁNKI KÖZLEMÉNYEK --- BÁNKI REPORTS
VIII. ÉVFOLYAM 1. SZÁM --- VOLUME VIII NO. 1

Big data feldolgozás teljesítményének vizsgálata hibrid felhőkörnyezetben: fájlformátumok és a sávszélesség hatása

Performance Analysis of Big Data Workloads in Hybrid Cloud Environments: The Role of File Formats and Bandwidth

Kantó Attila*, Marosi Attila Csaba**
*Cloudera Hungary Kft., Budapest, Magyarország
**HUN-REN Számítástechnikai és Automatizálási Kutatóintézet (HUN-REN SZTAKI), Magyar Kutatási Hálózat (HUN-REN), Budapest, Magyarország
akanto@cloudera.com, attila.marosi@sztaki.hun-ren.hu

***Összefoglalás* — A tanulmány azt vizsgálja, hogyan befolyásolja a fájlformátum-választás (CSV, Parquet, ORC) a lekérdezések teljesítményét hibrid felhőkörnyezetben, ahol az adatok hagyományos adatközpontban kerülnek tárolásra, a feldolgozás pedig felhőben történik. A TPC-DS benchmark alapján méréseink azt mutatják, hogy az oszlopalapú formátumok több mint 88%-kal csökkentik a végrehajtási időt, és akár 96%-kal az adatátvitelt a CSV-hez képest. Az eredmények rámutatnak arra, hogy a fájlformátum-választás nemcsak a tárolási hatékonyságot, hanem a hálózati terhelést is érdemben befolyásolja, különösen elosztott, hibrid környezetekben.**

***Kulcsszavak*:** oszlopos fájlformátum, data lake, adatlokalitás, hibrid cloud, forgalomszabályozás, TPC-DS

***Abstract* — This study examines how file format selection (CSV, Parquet, ORC) impacts query performance in a hybrid cloud environment where data remains on-premise while processing occurs in the cloud. Based on TPC-DS benchmarks, our measurements show that columnar formats reduce execution time by over 88% and data transfer by up to 96% compared to CSV. The results indicate that file format selection significantly impacts not only storage efficiency but also network load, especially in distributed hybrid environments.**

***Keywords*:** columnar storage, data lake, data locality, hybrid cloud, traffic shaping, TPC-DS

1 BEVEZETÉS

A nagyvállalati adatfeldolgozásban a hibrid felhőarchitektúrák olyan rugalmas megoldást kínálnak, amely lehetővé teszi az *on-premise* adatközpontok meglévő infrastruktúrájának kiegészítését felhőalapú számítási kapacitásokkal, így biztosítva a fokozatos átmenetet a teljes migráció helyett. A hibrid megközelítések bevezetését több szempont is motiválja.

Egyrészt gazdasági rugalmasságot kínálnak: a publikus felhők hozzáférést biztosítanak olyan speciális erőforrásokhoz, mint például GPU-k, FPGA-k vagy memóriaoptimalizált virtuális gépek. Ezek folyamatos használata azonban az *on-demand* árazás miatt jellemzően nem költséghatékony. A hibrid modell lehetőséget nyújt arra, hogy a már amortizált, helyszíni infrastruktúra szolgálja ki az állandó terhelést, míg az időszakosan megnövekedő kapacitásigények – például évzárás, kampányidőszak vagy előrejelezhető adatcsúcsok – a felhőbe kiszervezve kerüljenek feldolgozásra.

Másrészt szabályozási és geopolitikai kényszerek is meghatározzák a hibrid stratégiák kialakítását. Különösen a pénzügyi szektor szereplőinek kell szigorúan betartaniuk az adatrezidencia-szabályozásokat, amelyek előírják, hogy bizonyos típusú adatokat az adott ország határain belül kell tárolniuk és feldolgozniuk.

Végül nem elhanyagolható tényező a technikai inercia és alkalmazás-gravitáció sem. A meglévő, adatközponti környezetben futó analitikai rendszerek – mint például a tradicionális Hadoop-ökoszisztéma – gyakran nehezen migrálhatók felhőbe. Ennek oka többek között a *schema-on-read* szemantikán alapuló adattárolás, a beégetett és nem paraméterezhető elérési útvonalak, valamint a petabájtos nagyságrendű HDFS [1] klaszterek fizikai mozgatásának kihívásai. A hibrid modell lehetővé teszi, hogy ezek a rendszerek fokozatosan váljanak felhő-kompatibilissé, elkerülve a “migrációs szakadékot”, amely egy egyszeri, teljes átállás során jelentkezik.

A hibrid modell különösen hasznos lehet akkor, ha egy szervezet időszakosan jelentkező, nagy számítási igényű feladatokat futtat, és nem kíván állandóan túlméretezett saját infrastruktúrát fenntartani, ugyanakkor a hibrid felhőmodell hatékony működése számos technikai kihívást vet fel. Az egyik legkritikusabb tényező a hálózati sávszélesség, amely gyakran szűk keresztmetszetként jelenik meg a felhőből indított, de *on-premise* adattárolót használó lekérdezések esetében. Ezen a ponton különösen fontos szerepet kap az adattárolás formátuma. Jelen tanulmány célja annak vizsgálata, hogy különböző

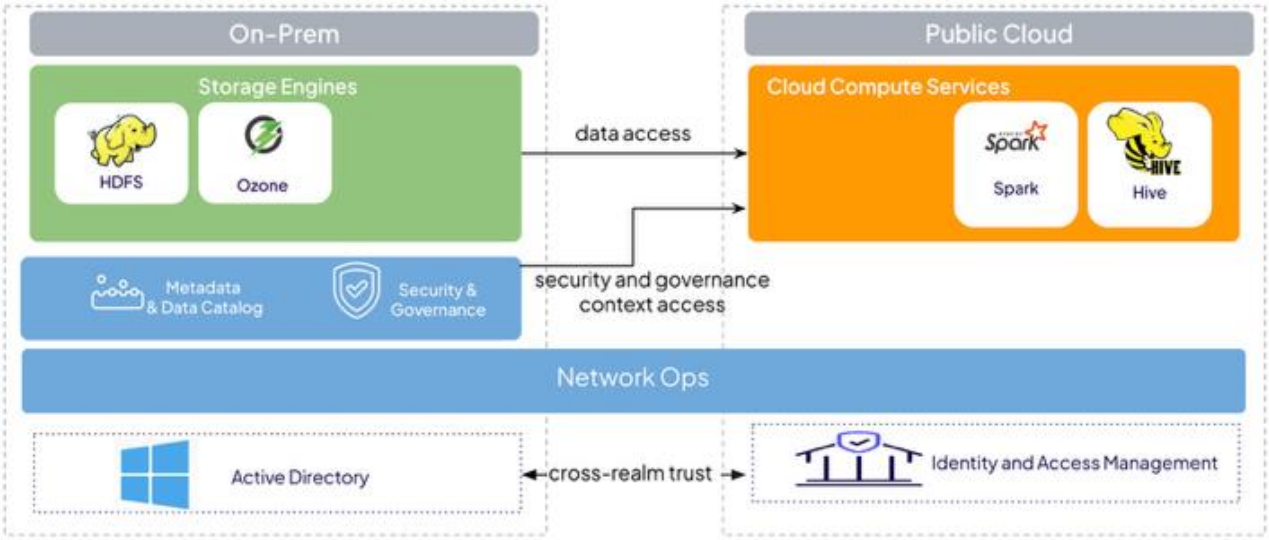
fájlformátumok hogyan befolyásolják a lekérdezések végrehajtási idejét és az átvitt adatmennyiséget.

A vizsgálat során az alábbi főbb következtetésekre jutottunk:

- A hibrid felhőarchitektúrákban alkalmazott fájlformátum-választás jelentős hatással van a lekérdezések végrehajtási idejére, a hálózaton átvitt adatmennyiségre és a CPU-kihasználtságra, még akkor is, ha a végrehajtási terv (Directed Acyclic Graph, DAG) lényegében változatlan marad.

2 A HIBRID FELHŐKÖRNYEZET ARCHITEKTURÁLIS MODELLJE

A nagyvállalati környezetekben az adatokhoz való hozzáférés nem csupán technikai kérdés, hanem biztonsági, megfelelőségi és üzleti szempontból is kritikus tényező. A hibrid architektúrák megvalósítása során kiemelt fontosságú az autentikáció és autorizáció, az adatbiztonság, valamint a *governance* következetes és egységes kezelése mind az *on-premise*, mind a felhős környezetekben. A hibrid felhőarchitektúrák két domináns megközelítése az adatok és a metaadatok – például tábladefiníciók, valamint



1. ábra. Távoli adathozzáférés on-premise tárolóból architektúra diagrammja

- Az oszlopalapú Parquet [2][4] és ORC [4] formátumok több mint 88%-kal csökkentették a végrehajtási időt a CSV-hez képest, miközben az átvitt adatmennyiség több mint 96%-kal volt alacsonyabb.
- A távoli adatelérés típusú hibrid modell sikeres alkalmazásához elengedhetetlen az adatátviteli mennyiség csökkentése. Ezt elsődlegesen a fájlformátum és az adatstruktúra megválasztásával lehet elérni.
- Bár jelen tanulmány nem tért ki a tömörítési algoritmusok részletes vizsgálatára, előzetes tapasztalataink alapján a formátumon belüli tömörítési technikák szintén jelentősen befolyásolják a hálózati átvitel mértékét és a feldolgozás teljesítményét.

A tanulmány a következőképpen épül fel. A második fejezet bemutatja a hibrid felhőarchitektúrák két domináns modelljét: a távoli adathozzáférésen és az adatreplikáción alapuló megközelítést. A harmadik fejezet áttekinti a technológiai háttérelemeket és korábbi kutatásokat. A negyedik fejezet részletesen ismerteti a kísérleti környezetet, különös tekintettel a hálózati sávszélesség szabályozására és a vizsgálati paraméterekre. Az ötödik fejezet bemutatja a TPC-DS [5] benchmark alapján elvégzett mérések eredményeit, valamint a fájlformátumok és hálózati korlátok teljesítményre gyakorolt hatását. A hatodik fejezet összefoglalja a főbb megállapításokat és kijelöli a jövőbeni kutatás lehetséges irányait.

a hozzájuk tartozó hozzáférési jogosultságok, vagyis annak meghatározása, hogy ki milyen szinten férhet hozzá az egyes táblákhoz, oszlopokhoz, sorokhoz vagy konkrét adatokhoz – fizikai elhelyezkedése és elérhetősége alapján: ezek a távoli adathozzáférésen, illetve az adatreplikáción alapuló megoldások. Ezt a két megközelítést a következő két alfejezetben részletesen bemutatjuk.

2.1 Távoli adathozzáférés on-premise tárolóból

Ebben a hibrid modellben az adatok fizikai tárolása az adatközponti infrastruktúrában történik (pl. HDFS vagy Ozone [6]), míg a számítási feladatokat a publikus felhőben futó szolgáltatások végzik (pl. Apache Spark, Hive). Az adatfeldolgozás tehát a felhőben fut, de az adatforrás továbbra is az *on-premise* környezet marad. Minden lekérdezés során hálózati adatátvitel történik a két környezet között (lásd 1. ábra). Az egyik legfontosabb előny ebben a modellben, hogy az analitikai alkalmazások kódja jellemzően változatlan maradhat, hiszen az adatok logikai elérési útjai és szerkezete nem változik. Nincs szükség az alkalmazások újra tervezésére vagy adatmigrációra, ami jelentős idő- és költségmegtakarítást eredményezhet. Ez különösen hatékony akkor, amikor ideiglenesen megnövekedett feldolgozási kapacitásra van szükség (*burst workload*). További előny, hogy a metaadat- és biztonsági kontrollréteg központosítva marad az *on-premise* környezetben, és ez a központosított biztonsági réteg lehetővé teszi, hogy a tábla-definíciók, hozzáférési jogosultságok, auditálási beállítások és *governance* szabályok egységesen érvényesüljenek – függetlenül attól, hogy a feldolgozás *on-premise* vagy publikus felhőben történik, például az adatbázis táblákhoz rendelt olvasási

vagy írási jogosultság mindkét környezetben ugyanúgy érvényesül.

Ugyanakkor a modell kihívásokkal is jár, különösen az identitáskezelési réteg integrációja terén. A nagyvállalati környezetekben az autentikációt jellemzően Active Directory biztosítja, míg a felhőben gyakran teljesen más identitáskezelési rendszerek működnek. Mivel a felhőben futó számítási motorok adatot érnek el az *on-premise* környezetből, biztosítani kell hogy azonosított és jogosult felhasználóként tudják ezt megtenni. Erre az egyik legelterjedtebb megoldás az ún. Cross-Realm Trust, amely lehetővé teszi az *on-premise* és felhőbeli tartományok közötti hitelesítési kapcsolatot, így az egyik rendszerben lévő felhasználók jogosultan férhetnek hozzá a másik rendszer erőforrásaihoz, legyen az adat, metainformáció, vagy akár rendszerkomponens.

2.2 Adatreplikáció a felhőbe

A második elterjedt hibrid megközelítés az adatreplikációra épül, ahol a vállalati adatok és metaadatok a helyszíni környezetből a felhőszolgáltató által biztosított tárhelyre kerülnek átmásolásra - jellemzően objektumtárolókba (pl. Amazon S3, Azure Blob Storage, Google Cloud Storage). A feldolgozás így teljes egészében a publikus felhőben történik, a replikált adatokkal dolgozó szolgáltatások közvetlenül felhős erőforrásokra támaszkodnak, ahogy a 2. ábra mutatja.

Ez a modell akkor lehet előnyös, ha az adatfeldolgozást teljesen el akarjuk választani az adatközponttól. A publikus és *on-premise* rendszerek fizikailag és logikailag is függetlenül működnek, így nincs szükség állandó hálózati kapcsolat fenntartására a két környezet között, valamint önállóan skálázhatók és üzemeltethetők. Az architektúra egy- vagy kétirányú replikációval is működhet, például a forrásadatok felhőbe történő másolása után előfordulhat, hogy a feldolgozás eredményeként keletkező adattermékek egy részét vissza kell másolni az adatközpontba.

Mivel az adatok fizikailag átkerülnek a felhőbe, az analitikai alkalmazásokat és lekérdezéseket módosítani kell, hogy az új adattárakat érjék el. A korábban említett, kódváltoztatás nélküli feldolgozás így lehetséges a logikát, útvonalakat újra kell konfigurálni.

Emellett a metaadatokat, a biztonsági beállításokat és a *governance* szabályokat is külön kell kezelni és replikálni, mivel az adatok fizikai helye, hozzáférési útvonala és jogosultsági kontextusa megváltozik. Ez különösen bonyolult lehet dinamikusan változó szabályok vagy érzékeny adatok esetén, mivel a biztonsági és *governance* réteg már nem központilag érvényesül, hanem külön kell kialakítani a felhőben és az adatközpontban is. További korlátozás, hogy mivel az alkalmazásokat, lekérdezéseket és adatfeldolgozó folyamatokat újra kell írni, és gyakran eltérő, felhőspecifikus komponensekhez kell igazítani, nem valósítható meg az a fajta dinamikus terhelésalapú irányítás, amely lehetővé tenné, hogy egy adott feldolgozási feladat valós időben kerüljön átirányításra a privát vagy publikus környezetbe a pillanatnyi erőforrás-igények szerint.

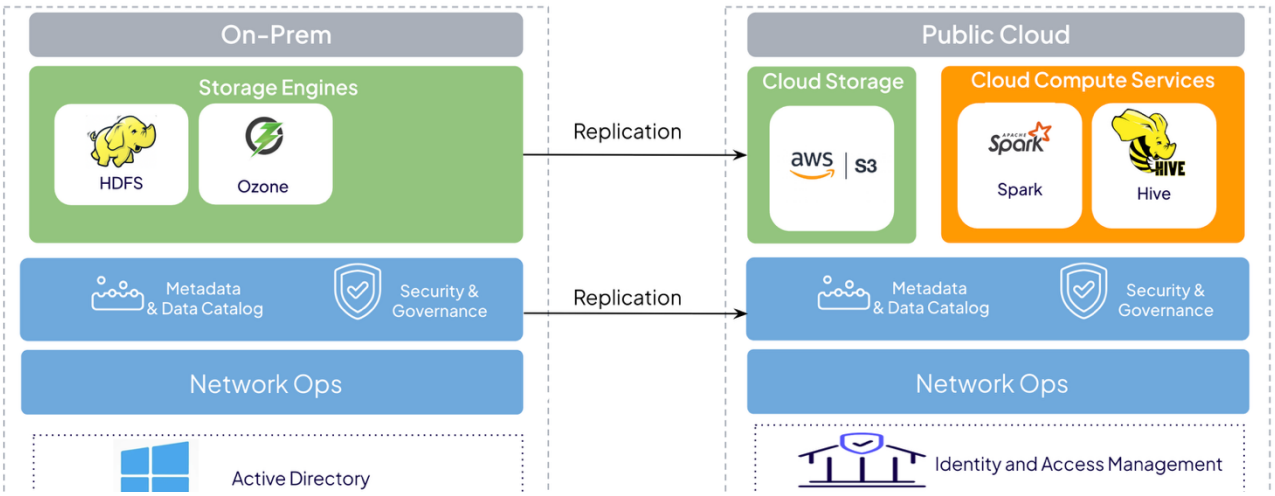
Ez az architektúra akkor lehet előnyös, ha a szervezet világosan el tudja különíteni a felhőben és az adatközpontban végzett feldolgozási feladatokat, és vállalja az ehhez szükséges alkalmazásmódosításokat és metaadat-replikációt. Ugyanakkor a környezetek közötti laza kapcsolódás, a független skálázhatóság és a hálózati szétválasztás magas fokú üzemeltetési rugalmasságot biztosít.

2.3 A vizsgálat fókusza: távoli adateléréses modell

A hibrid architektúrák közül jelen tanulmány az első megközelítést, a távoli adatelérés (*remote data access*) modelljét helyezi fókuszba. Ennek egyik legfontosabb indoka, hogy ez a megközelítés gyakorlatban a legelterjedtebb olyan szervezetek körében, amelyek már rendelkeznek jelentős méretű helyszíni (*on-premise*) adatplatformmal, de a feldolgozási kapacitásuk időszakosan kiegészítésre szorul.

A *burst* típusú munkaterhelések, amikor rövid ideig kiemelten nagy számítási igény lép fel, ideális jelöltek erre a modellre. Mivel az adatok logikai elérési útja nem változik, az analitikai alkalmazások kódja módosítás nélkül újrafelhasználható, és nincs szükség újraépített ETL *pipeline*-okra vagy felhőspecifikus rendszerkomponensek integrálására.

A modell hátránya, hogy minden lekérdezés adatot továbbít a hálózaton keresztül a helyszíni adatforrás és a felhőben futó végrehajtómotor között. Míg a felhőalapú számítási, memória- és tárolóerőforrások könnyen skálázhatók, a hálózati sávszélesség jellemzően merev korlát, amelynek bővítése költséges és technikailag is nehézkes lehet. A sávszélességet befolyásolja a fizikai infrastruktúra (pl. *switch*ek, *router*ek), a cloud interconnect



2. ábra. Adatreplikáció a felhőbe architektúra diagrammja

konfigurációja, ezért a távoli adatelérés esetén kulcskérdés, hogyan lehet csökkenteni a hálózaton továbbított adatmennyiséget.

3 Technológiai háttér és kapcsolódó kutatások

Ebben a fejezetben áttekintjük azokat az alapvető technológiákat és fogalmakat, amelyek a dolgozat későbbi részeiben szerepelnek.

3.1 Adattárolás

A Hadoop Distributed File System (HDFS) egy elosztott, nagy adatmennyiségek kezelésére tervezett fájlrendszer, amely replikáción alapul a megbízhatóság érdekében. A többszörösen tárolt blokkok lehetővé teszik a párhuzamos olvasást, ezáltal jelentősen növelhető az elérhető olvasási sávszélesség. Ez a modell akkor a leghatékonyabb, ha a számítás fizikailag közel történik az adatokhoz (*data locality*). A hibrid környezetek megjelenése azonban felbontja ezt a mintát: ha a feldolgozás felhőben történik, az on-premise HDFS eléréséhez minden esetben hálózaton keresztül kell adatot olvasni.

Bár újabb tárolási megoldások is megjelentek, a HDFS továbbra is széles körben használt az adatközponti rendszerekben, emiatt jelen tanulmány mérései során HDFS-alapú adattárolást alkalmaztunk.

3.2 Adatkatalógus

A Hive Metastore (HMS) egy központosított metaadat-szolgáltatás, amely sémákat, tábladefiníciókat és az adatok fizikai elérési útvonalát tartalmazó URI-kat tárolja. Emellett statisztikai információkat is nyilvántart sor-, tábla- és partíciószinten, amelyeket a lekérdező motorok optimalizálásra használnak. Jelenleg az *on-premise* környezetekben az HMS az iparági szabványként működik. A vizsgálat során mi is ezt használjuk. A jövőbeni kutatások során újabb generációs megoldásokat tervezünk megvizsgálni, mint például az Apache Iceberg [18], Hudi [19] vagy Delta Lake [20], amelyek táblakezelési és evolúciós képességek terén fejlettebbek, és a felhőalapú környezetekben már egyre elterjedtebbek.

3.3 Lekérdezésvégrehajtás Apache Spark SQL

A nagy adatmennyiségek lekérdezéséhez a strukturált lekérdezési nyelvek, elsősorban az SQL-alapú megközelítések, továbbra is a legelterjedtebbek. A Spark SQL integrálható az előző szekcióban bemutatott Hive Metastore-ral, így képes a meglévő sémákat és táblákat változtatás nélkül kezelni, támogatva az *on-premise* és hibrid környezetek közötti átjárhatóságot. Ezért a mérések során Spark SQL-t használtunk a lekérdezések végrehajtására.

3.4 Fájltárolási formátumok

A strukturált táblázatos adatok különböző formátumokban tárolhatók, amelyek hatással vannak az adatfeldolgozás teljesítményére, a tárolási költségekre, valamint a hálózati adatátvitel mértékére. Az alábbiakban bemutatjuk a három legfontosabb fájlformátumot, amelyek szerepeltek a mérések során: CSV, Apache Parquet és Apache ORC.

A CSV (*Comma-Separated Values*) az egyik legegyszerűbb fájlformátum, ahol az adatok egyszerű szövegként kerülnek tárolásra, soronként, egy meghatározott elválasztó karakterrel (általában vessző vagy pontosvessző). A mezők nem rendelkeznek típusinformációval, minden érték karakterláncként értelmeződik. A CSV soralapú elrendezése miatt a lekérdező motoroknak a teljes sort be kell olvasniuk, még akkor is, ha csak egyetlen oszlopra van szükség ez jelentős I/O terhelést eredményezhet nagy adathalmazoknál. Emellett a CSV nem támogatja a sémaváltozást (*schema evolution*), nem tartalmaz belső metaadatokat, és nem rendelkezik indexeléssel. Előnye ugyanakkor az egyszerűsége, könnyű generálhatósága és olvashatósága.

Az Apache Parquet egy modern, oszlopalapú fájlformátum, amelyet kifejezetten nagy adatmennyiségek hatékony tárolására és feldolgozására terveztek. A Parquet oszloponként tárolja az adatokat, így csak a lekérdezés által érintett oszlopok kerülnek beolvasásra; ez jelentősen csökkenti az I/O műveletek számát, és gyorsítja a feldolgozást. A formátum számos adattömörítési és kódolási [7] technikát alkalmaz, például:

- *Dictionary encoding*: gyakran előforduló értékek szótáralapú tárolása.
- *Run-Length Encoding* (RLE) és *Bit-Packing*: ismétlődő értékek és kis egész számok tömör tárolása.

Az Apache Parquet támogat többféle tömörítési algoritmust, köztük az Uncompressedet, Snappyt, Gzipet, LZ4-t és a Zstandardet (Zstd). A Parquet fájl logikai felépítése *row group*okra (sorcsoport) tagolódik. Egy *row group* több ezer sor adatát tartalmazza, oszloponként csoportosítva, valamint tárolja az adott csoport statisztikáit (pl. min/max, null-arány). Ezek lehetővé teszik a *predicate pushdown*-technikák alkalmazását, a lekérdező motor a statisztika alapján már a beolvasás előtt eldöntheti, hogy egy adott csoport releváns-e.

Az Apache ORC (Optimized Row Columnar) szintén oszlopalapú formátum, a Parquet-hoz hasonlóan ORC is tömbösíti az adatokat, de *stripe*-okba (adatsávokba) rendezi azokat, amelyek egy-egy vízszintes szeletként tartalmazzák az adatokat, indexeket, statisztikákat és opcionálisan Bloom-filtereket. Az ORC minden fájlban háromszintű indexelést biztosít: fájlszintű index, *stripe*-szintű index, és soronkénti statisztikák (minden 10 000 rekord után). Ez lehetővé teszi, hogy a lekérdezés-végrehajtó motor gyorsan kizárja azokat az adatblokkokat, amelyek biztosan nem tartalmaznak releváns értékeket az I/O műveletek minimalizálása érdekében.

3.5 TPC-DS benchmark

A teljesítménymérésekhez a TPC-DS (*Transaction Processing Performance Council - Decision Support*) [5][9] benchmarkot alkalmaztuk, amely a nagyvállalati döntéstámogató rendszerek terhelésének szabványos szimulációjára szolgál. A TPC-DS tartalmaz egy szintetikus, de üzletileg releváns adatkészletet, amely egy kiskereskedelmi vállalat működését szimulálja. Az adatkészlet 24 táblából áll: 17 ténytábla és 7 dimenziótábla. A ténytáblák lineárisan, míg a dimenziótáblák szublineáris arányban növekednek a konfigurált adatmennyiség függvényében. A vizsgálatok során 1000-es skálázási faktort használtunk, ami körülbelül 1000 GB nyers adatnak felel meg.

A TPC-DS benchmark része 99 SQL-lekérdezést tartalmaz, amelyek valós döntéstámogatási forgatókönyveket tükröznek. A lekérdezések között szerepelnek [8]:

- ad-hoc lekérdezések (egyszeri, gyors döntéshozatal támogatására);
- jelentéskészítés (pl. napi/heti összegzések);
- OLAP típusú elemzések (többdimenziós összehasonlítás, szezonális vizsgálatok);
- valamint adatbányászati mintázatkeresés.

Az adatok nem egyenletes eloszlásúak: a TPC-DS adatkészlet szándékosan olyan torzításokat (*skew*) tartalmaz, amelyek a valós üzleti környezetre jellemzőek. Ilyen például:

- A szezonális keresletváltozás, amely ünnepi időszakokra, kampányokra utal.
- Gyakran előforduló terméknevek és eladási csúcsok.
- Az időszakos visszaesések, amelyek természetes részei a kiskereskedelmi ciklusnak.

Ennek a felépítésének köszönhetően a TPC-DS különösen hasznos azokban a szcenáriókban, ahol időszakos terhelésnövekedés figyelhető meg, például promóciók, kampányidőszakok vagy szezonalitás miatt. Ezek a környezetek gyakran igénylik a rövid távú skálázást, amit a hibrid felhőmodellek *compute burst* képességei képesek kiszolgálni. Éppen ezért a TPC-DS ideális benchmark számunkra a hálózati korlátok és adattárolási formátumok összehasonlító vizsgálatához.

3.6 AWS Direct Connect

AWS Direct Connect [16] egy dedikált hálózati kapcsolatot biztosító szolgáltatás, amely lehetővé teszi, hogy egy szervezet saját belső hálózatát közvetlenül csatlakoztassa az AWS infrastruktúrájához egy szabványos Ethernet optikai kapcsolaton keresztül. A kapcsolat egyik vége a vállalati routerhez, a másik az AWS Direct Connect *router*éhez kapcsolódik. Ennek révén virtuális interfészek hozhatók létre, amelyek közvetlen hozzáférést biztosítanak AWS-szolgáltatásokhoz. Kutatásunk szempontjából az AWS Direct Connect kulcsfontosságú technológia, mivel egy olyan dedikált hálózati kapcsolatot képvisel, amely a vállalati gyakorlatban is széles körben használt megoldás a hibrid felhőarchitektúrák kialakítására.

3.7 Kapcsolódó kutatások

Ivanov és Pergolesi [10] munkájukban az ORC és Parquet fájlformátumok teljesítményét hasonlították össze Apache Hive és SparkSQL környezetekben a BigBench [11] benchmark segítségével. Céljuk annak vizsgálata volt, hogy a fájlformátum megváltoztatása, illetve a különböző konfigurációs paraméterek hogyan befolyásolják a feldolgozómotorok általános teljesítményét. Ezzel szemben jelen tanulmány célja annak feltárása, hogy egy hibrid felhőkörnyezetben, ahol az adat *on-premise* tárolóban marad, a feldolgozás pedig a felhőben történik, a fájlformátum miként hat a lekérdezések teljesítményére, különösen akkor, amikor a két környezet közötti hálózati sávszélesség szűk keresztmetszetet jelent.

4 KÍSÉRLETI KÖRNYEZET

A kísérleti környezet két elkülönített klaszterből állt:

- *On-premise* klaszter: Ez a környezet egy klasszikus *on-premise* telepítést szimulált, amelyben HDFS, AWS-ben futó, virtualizált gépeken került kialakításra.
- Felhőalapú klaszter: Ez a klaszter szintén AWS virtuális gépeken futott, és a feldolgozási feladatokat a SparkSQL motor segítségével hajtotta végre.

Mindkét környezet négy virtuális gépből állt: egy dedikált vezérlő gépből (*master*), amely az irányítási és koordinációs feladatokat látta el, valamint három munkavégző (*worker*) virtuális gépből, amelyek a benchmark futtatásáért feleltek. Bár minden gép az AWS infrastruktúráján futott, az *on-premise* klaszter szoftveres és hálózati konfigurációja a lehető legnagyobb mértékben követte egy valós HDFS-alapú adatközponti környezet felépítését. Célunk az volt, hogy egy hagyományos, *on-premise* infrastruktúra viselkedését minél pontosabban modellezzük. A használt virtuális gépek részletes hardverkonfigurációját az 1. táblázat tartalmazza [15].

1. táblázat. Hardver konfiguráció

Specifikáció	Master VM	Worker VM (x 3)
VM típus (AWS)	r5.4xlarge	m6i.4xlarge
vCPU	16	8
Memória (GB)	128	64
Adattároló típusa	–	GP3 EBS
Lemezek száma	–	4
Lemezkapacitás (TB)	–	1 / disk
Átviteli sebesség (MB/s)	–	250 / disk
Alap hálózati sávszélesség (Gb/s)	5.0	6.25
Burst hálózat (Gb/s)	10.0	12.5

4.1 Forgalomszabályozás (Traffic Shaping Gateway)

A távoli adathozzáférésen alapuló hibrid architektúrák esetében feltételezzük, hogy a hálózati sávszélesség korlátozottsága kritikus hatással van a lekérdezések végrehajtási idejére. E hipotézis vizsgálatára egy köztes infrastruktúraelem, az ún. forgalomszabályzó átjáró (*Traffic Shaping Gateway*, TSG) került beépítésre a helyszíni és a felhőalapú környezet közé. A TSG egy olyan központosított hálózati komponensként terveztük, amely lehetővé teszi a két klaszter közötti hálózati forgalom monitorozását és korlátozását, miközben az egyes klasztereken belüli kommunikáció zavartalanul és változatlan módon működhet. A teljes infrastruktúrában minden virtuális gépen statikus útvonalbeállításokat alkalmaztunk (*ip route* parancs segítségével) [12], így minden adat- és metaadatforgalom a TSG-n keresztül haladt át. A hálózati sebességkorlátozást a Linux tc [14] keretrendszerében elérhető *Hierarchical Token Bucket* (HTB) [13] algoritmus segítségével valósítottuk meg, amely lehetővé teszi sebességprofilok definiálását, késleltetések bevezetését, valamint a prioritások szabályozását.

Mivel az AWS-ben futó virtuális gépek hálózati viselkedését befolyásolhatja az ún. *network I/O credit* rendszer, mely ideiglenes, kreditalapú sávszélesség-növekedést (*burst*) tesz lehetővé, a mérések torzulásának elkerülése érdekében minden gépen rögzítettük a hálózati kapcsolatot az alap sávszélességi értéken, ezáltal egy valós

on-premise környezethez hasonló hálózati viselkedést biztosítva.

Az AWS Direct Connect [16] szolgáltatás 50 Mb/s és 25 Gb/s közötti sávszélességet támogat. Kísérleteink során kettő jellemző értéket, 5 Gb/s, és 25 Gb/s választottunk ki. Ezeket a sávszélesség-korlátokat egyenként alkalmaztuk a forgalomszabályzó átjárón keresztül. Ennek köszönhetően a vizsgálat nem csupán szintetikus tesztkörnyezetre épült, hanem iparági szempontból is valósághű kapcsolatmodelleket tükrözött.

5 MÉRÉSI EREDMÉNYEK

Ebben a fejezetben a TPC-DS benchmark lekérdezések végrehajtása során tapasztalt erőforrás-használati mintázatokat elemezzük. Méréseink során minden egyes lekérdezést öt alkalommal futtattunk le, és az eredmények közül a medián végrehajtási időt rögzítettük, hogy csökkentsük az esetleges zaj vagy infrastruktúra-ingadozás hatását. Méréseink során az alábbi metrikákat rögzítettük minden egyes lekérdezés esetében:

- Átlagos CPU-kihasználtság: a lekérdezés teljes futása alatt mért CPU-terhelés a munkavégző virtuális gépeken.
- Hálózati sávszélesség-használat: az *on-premise* és felhő között átvitt adatmennyiség, lekérdezésenként.
- Végrehajtási idő: a lekérdezés teljes időtartama.
- Adatkészlet mérete: a formátumtól és tömörítéstől függően változó tárolási méret.

5.1 Összehasonlító eredmények különböző fájlformátumok esetén

A benchmarkot három különböző fájlformátum esetén futtattuk le: tömörítetlen CSV, Parquet, valamint ORC. A 2. táblázat a 99 lekérdezés aggregált statisztikáit tartalmazza.

2. táblázat. TPC-DS lekérdezések aggregált eredménye különböző fájlformátumok esetén

Formátum	Adatkészletmérete (GB)	Végrehajtási idő (s)	Átvitt adat (GB)	CPU kihasználtság (%)
CSV	864,61	31018,39	22268,80	81,14
Parquet	326,23	3669,06	814,69	65,59
ORC	284,63	3876,30	704,19	70,19

Az eredmények alapján egyértelműen megállapítható, hogy az oszlopalapú adattárolási elvet követő fileformátumok jelentős teljesítménybeli előnyt biztosítanak a CSV formátumhoz képest, még akkor is, ha tömörítés nem került alkalmazásra. Vizsgálatunk kizárólag az oszlopalapú formátumok belső adatstruktúráira és kódolási mechanizmusaira koncentrált. A végrehajtási idő Parquet esetén 88,2%-kal, ORC esetén 87,5%-kal csökkent. Az *on-premise* és felhő közötti átvitt adatmennyiség Parquet esetén 96,3%-kal, ORC esetén 96,8%-kal volt alacsonyabb. Az adatkészlet mérete Parquet esetén 62,3%-kal, ORC esetén 67,1%-kal kisebb a CSV-hez viszonyítva. Az átlagos CPU-kihasználtság is számottevően csökkent: CSV esetén 87,14%, míg Parquet és ORC esetében 65,59% illetve 70,19%.

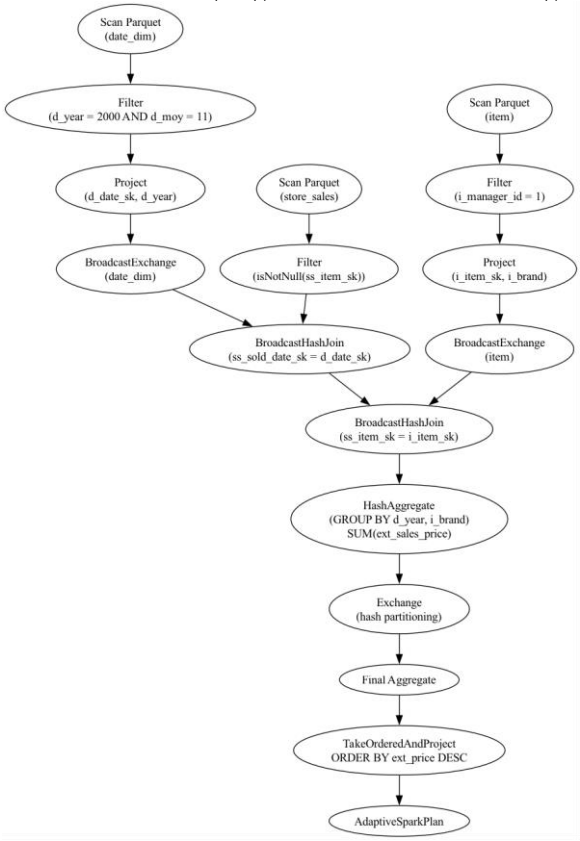
A teljesítménykülönbség elsősorban az oszlopalapú tárolás és a hozzá kapcsolódó belső kódolási technikák hatékonyságának köszönhető. Ezek közül kiemelendő:

- A *dictionary encoding*, amely a gyakran előforduló oszlopértékekhez szótárat rendel, ezáltal csökkenti a tárolandó értékek redundanciáját.
- Az RLE, amely a többször ismétlődő értékeket tömören reprezentálja, különösen hatékony alacsony kardinalitású oszlopok esetén.
- Az ORC formátum által támogatott Bloom--filterek, amelyek gyorsan képesek eldönteni, hogy egy adott érték szerepelhet-e az adott oszlopban, így hatékony előszűrést biztosítanak lekérdezés előtt.

Ezzel szemben a CSV formátum semmilyen ilyen típusú optimalizációt nem biztosít: sem kódolás, sem előszűrés, sem struktúra nincs beépítve. Ennek következtében minden lekérdezés során a fájl teljes egészében feldolgozásra kerül, ami jelentősen megnöveli a CPU- és hálózati terhelést, különösen hibrid környezetben, ahol a sávszélesség korlátos.

5.2 Esettanulmány: Query 52

Ebben a fejezetben egy konkrét példán keresztül mutatjuk be, hogyan befolyásolja a fájlformátum megválasztása egy lekérdezés végrehajtását. A TPC-DS 52-es [17] számú lekérdezése összetett *join*- és szűrési műveleteket tartalmaz (a végrehajtási tervet lásd a 3. ábrán), ezért jól reprezentálja azokat az analitikus lekérdezéseket, ahol a tárolási réteg és a formátum optimalizálása jelentős hatással lehet a végrehajtási időre, az adatátvitel mennyiségére és a CPU-kihasználtságra. Az



3. ábra. Lekérdezés-végrehajtási terv, TPC-DS lekérdezés 52

52-es lekérdezés egy összesített értékesítési kimutatást készít: kiszámítja a teljes eladási értéket (ext_sales_price) egy adott évben és hónapban értékesített, meghatározott márkájú termékekre. A végrehajtás során három táblát kapcsol össze: a date_dim, az item és a nagy méretű store_sales táblákat. A szűrés a date_dim táblán történik a d_moy és d_year oszlopok alapján, azaz időszakra szűr. Az ss_sold_date_sk kulcs alapján történő szűrés lehetővé teszi a store_sales partíciók hatékony beolvasását. Ennek köszönhetően, bár a tábla több mint 2,8 milliárd rekordot tartalmaz, a lekérdezés során csupán 88 millió rekord kerül tényleges beolvasásra, függetlenül a fájlformátumtól. A különbségek elsősorban a formátumok tárolási és dekódolási hatékonyságából fakadnak. CSV esetén a teljes date_dim tábla (73 049 sor) kerül beolvasásra, és csak utólag kerül alkalmazásra a szűrés. Ez nemcsak több adat beolvasását, hanem a teljes rekord minden mezőjének feldolgozását is jelenti (23 oszlop), még akkor is, ha csak 3 mező szükséges a lekérdezéshez. Parquet formátumban a Spark a *row group* statisztikák segítségével már a beolvasás előtt képes a szűrést részben alkalmazni, így csak 20 000 sort olvas be, amelyek közül végül 30 releváns sor marad. ORC esetén a formátum *stripe*-szintű statisztikái és Bloom-filterei révén a beolvasott rekordok száma 10.000-re csökken a szűrés előtt. A store_sales táblában is jelentős különbség tapasztalható. Míg CSV esetén minden oszlop beolvasásra kerül (majd a nem releváns 20 oszlop elvetésre), Parquet és ORC csak a lekérdezés által használt oszlopokat olvassa (ss_item_sk, ss_ext_sales_price, ss_sold_date_sk), így csökkentve az adatátviteli mennyiséget és a processzorterhelést. A következőkben a három formátum teljesítményét foglaljuk össze.

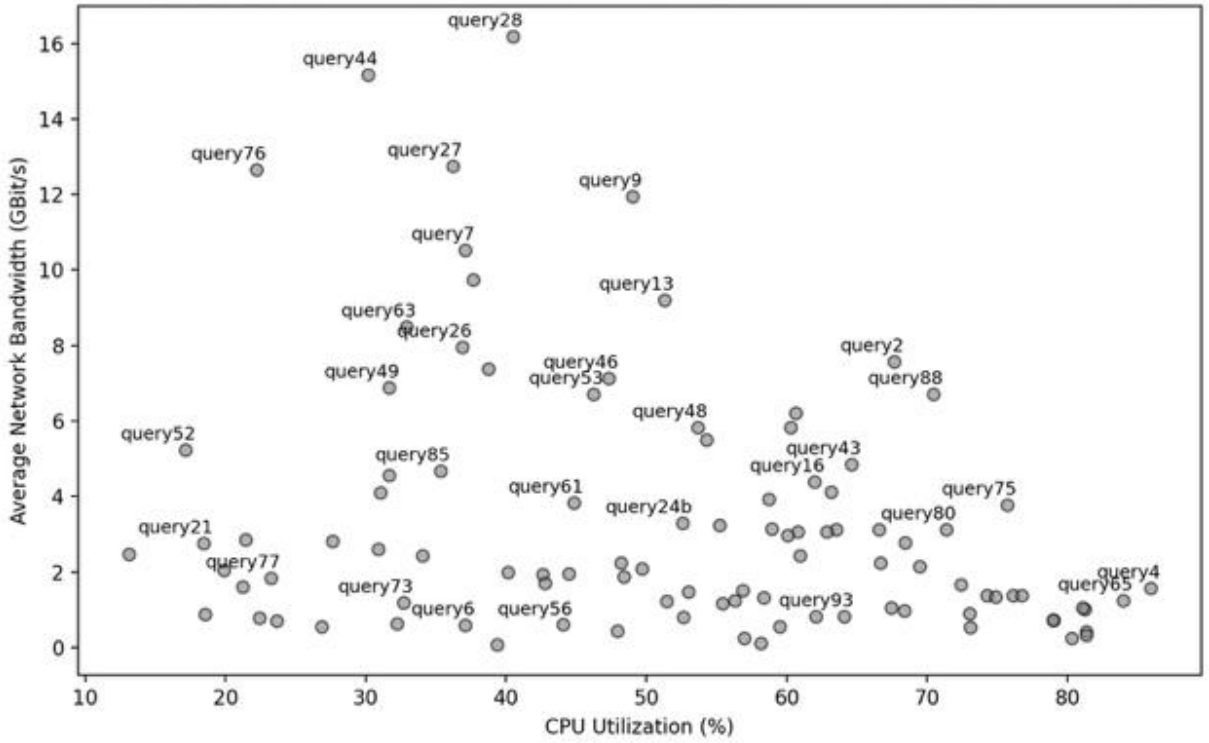
3. táblázat. TPC-DS 52-es lekérdezés teljesítményének összehasonlítása különböző fájlformátumok esetén

Formátum	Végrehajtási idő (s)	Átvitt adat (GB)	CPU kihasználtság (%)
CSV	17,41	11,05	69,52
Parquet	1,34	0,70	17,16
ORC	1,70	0,66	40,04

A 3. táblázatban szereplő mérések alapján jól látható, hogy a Parquet formátum alkalmazásával közel 13-szoros gyorsulás érhető el a CSV-hez képest, miközben az átvitt adatmennyiség több mint 15-szörös csökkenést mutat. Az ORC formátum szintén hasonló javulást eredményez, ugyanakkor valamivel magasabb CPU-kihasználás mellett, ami feltehetően a formátum dekódolási sajátosságaiból – például a Bloom-filterek alkalmazásából és a *stripe*-szintű indexelésből – adódik. Ezek az eredmények egyértelművé teszik, hogy bár a lekérdezés végrehajtási terve nem változik jelentősen, a fájlformátum megválasztása drasztikus hatással lehet a végrehajtási időre, az adatátvitel mennyiségére és a processzorterhelésre.

5.3 Lekérdezések erőforrásigényének vizsgálata scatter plot alapján

A következő két ábrán lekérdezésenként ábrázoljuk a CPU-kihasználtság és az átlagos hálózati sávszélesség értékeit, különböző sávszélességi korlátok mellett. Minden pont egy-egy TPC-DS lekérdezést jelöl, a címkéken a lekérdezés szám is feltüntetésre került.



4 ábra. TPC-DS lekérdezések CPU- és hálózati sávszélesség igényének eloszlása korlátlan sávszélesség mellett

A 4. ábra azt az esetet mutatja be, amikor a hálózati sávszélesség felső korlátját 25 Gbit/s értékre állítottuk be a forgalomszabályzó átjárón. Ez a kísérleti környezetben gyakorlatilag korlátlannak tekinthető, mivel a háttértárolók összesített I/O teljesítménye ennél alacsonyabb volt. Ezért látható az ábrán, hogy egyetlen lekérdezés sem éri el a 25 Gbit/s elméleti sávszélességet.

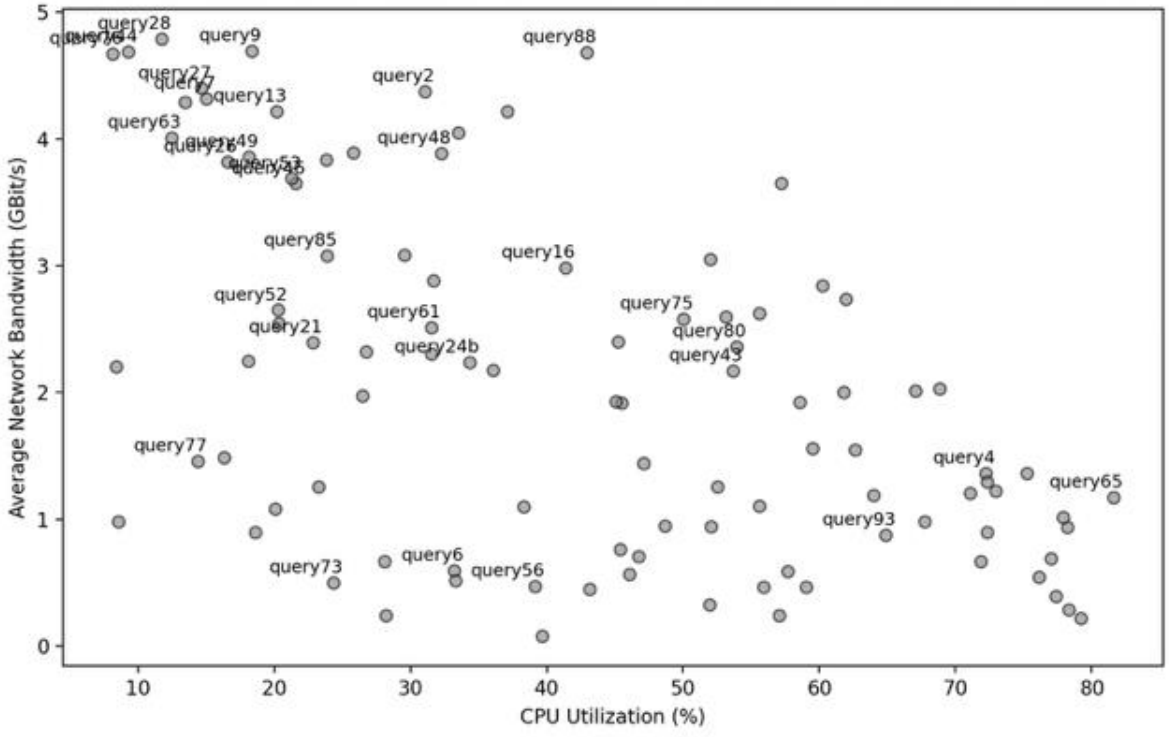
Ezzel szemben az 5. ábra azokat a méréseket szemlélteti, amelyek során a hálózati kapacitást 5 Gbit/s értékre korlátoztuk. Ez lehetővé tette, hogy megfigyeljük, hogyan viselkednek a hálózatintenzív és processzorigényes lekérdezések eltérő sávszélességi feltételek mellett.

A bal felső régióban elhelyezkedő lekérdezések, mint például a query28, query44, query76 és query9, jellemzően magas hálózati sávszélességet igényelnek, miközben viszonylag alacsony CPU-kihasználtsággal futnak. Ezek a lekérdezések gyakran nagyméretű adatmozgatást végeznek, és sávszélesség korlátozását követően ezek a lekérdezések egyértelműen lejjebb tolódtak a grafikonon, ami arra utal, hogy a rendelkezésre álló hálózati kapacitás csökkenése jelentősen befolyásolta az adatátvitel mértékét. Ennek hatására a végrehajtási idő is megnövekedett, mivel a csökkent hálózati áteresztőképesség miatt a feldolgozás hosszabb időt vett igénybe. Ezzel szemben a jobb alsó régióban található lekérdezések, például a query65, query4 és query93, kifejezetten magas CPU-terhelést mutatnak, ugyanakkor alacsony hálózati igénnyel rendelkeznek. Ezek esetében a sávszélesség korlátozása nem eredményezett érdemi elmozdulást a grafikonon, ami arra utal, hogy teljesítményüket elsősorban a processzorerőforrások határozzák meg, és nem érzékenyek a hálózati áteresztőképesség szűk keresztmetszeteire.

6 Következtetések

A tanulmány célja az volt, hogy megvizsgáljuk a hibrid felhőkörnyezetekben futó analitikus munkaterhelések teljesítményét különböző fájlformátumok alkalmazásával, valamint, hogy elemezzük a hálózati sávszélesség szűk keresztmetszetként való hatását. A vizsgálat során a *compute burst* típusú architektúrát alkalmaztuk, amelyben az adatok *on-premise* tárolókban maradtak, míg a feldolgozás felhőalapú erőforrásokkal történt. Az eredmények egyértelműen azt mutatják, hogy a fájlformátum kiválasztása kulcsfontosságú tényező, amely jelentős hatással van nemcsak a végrehajtási időre, hanem az adatátvitel mennyiségére és a CPU-kihasználtságra is. A Parquet és ORC formátumok – amelyek oszlopalapú tárolási szerkezetet, metaadat-alapú optimalizációkat (*predicate pushdown, column pruning*) és hatékony tárolási elvet alkalmaznak – több mint 88%-os gyorsulást eredményeztek CSV formátumhoz képest, miközben az átvitt adatmennyiség több mint 96%-kal csökkent.

A hálózati korlátozások hatását vizsgálva megállapítottuk, hogy egyes lekérdezések teljesítményét jelentősen befolyásolja a rendelkezésre álló sávszélesség. A hálózatintenzív lekérdezések – például query28 vagy query44 – érzékenyen reagáltak az 5 Gbit/s-ra való korlátozásra, míg a CPU-intenzív lekérdezések, mint a query65 vagy query93, gyakorlatilag változatlan teljesítményt mutattak. Ez megerősíti, hogy a hibrid felhőarchitektúrák teljesítménye nemcsak a számítási, hanem a tárolási és hálózati réteg tulajdonságaitól is függ. Fontos megjegyezni, hogy jelen munka nem tért ki a fájlformátumok által támogatott tömörítési algoritmusok (pl. Gzip, Snappy) teljesítményre gyakorolt hatásának részletes vizsgálatára. Ezt a kérdéskört egy különálló,



5 ábra. TPC-DS lekérdezések CPU- és hálózati sávszélesség igényének eloszlása 5 Gbit/s sávszélesség korlátozás mellett

jelenleg benyújtás alatt álló publikációban tárgyaljuk, amely kifejezetten azt elemzi, hogy a hálózatintenzív lekérdezések esetén különösen nagy adatmozgatást igénylő forgatókönyvekben mennyiben lehet javítani a teljesítményt a megfelelő tömörítési technika kiválasztásával. Jelen kutatás eredményei gyakorlati iránymutatást adhatnak a vállalati IT-stratégiák kialakításához, különösen olyan helyzetekben, amikor cél a meglévő adatközponti erőforrások felhőalapú számítási kapacitással történő kiegészítése. A jövőbeni munkák során olyan dinamikus döntéstámogató stratégiák kidolgozását tervezzük, amelyek képesek az alkalmazások vagy lekérdezések jellege alapján meghatározni, hogy az adott feladatot a felhőben célszerű végrehajtani, vagy inkább érdemes azt az adatközponti környezetben tartani.

KÖSZÖNETNYILVÁNÍTÁS

A jelen munkát részben a Technológiai és Ipari Minisztérium (korábban Innovációs és Technológiai Minisztérium) Nemzeti Kutatási, Fejlesztési és Innovációs Hivatala támogatta az Autonóm Rendszerek Nemzeti Laboratórium Program keretében; valamint a TKP2021-NVA-01 számú magyar projekt, amely a Technológiai és Ipari Minisztérium (korábban Innovációs és Technológiai Minisztérium) támogatásával, a Nemzeti Kutatási, Fejlesztési és Innovációs Alapból, a TKP2021-NVA finanszírozási program keretében valósult meg.

IRODALOMJEGYZÉK

[1] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system", 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), pp. 1–10, 2010. [Online]. Available: https://api.semanticscholar.org/CorpusID:13925042

[2] Apache Parquet, Apache Software Foundation. [Online]. Available: https://parquet.apache.org/ (accessed: 2025-05-18)

[3] D. Vohra, "Apache parquet", in Practical Hadoop Ecosystem: A Definitive Guide to Hadoop-Related Frameworks and Tools. Springer, 2016, pp. 325–335.

[4] Apache ORC, Apache Software Foundation. [Online]. Available: https://orc.apache.org/ (accessed: 2025-05-18)

[5] R. O. Nambiar and M. Pöss, "The making of tpc-ds," in Very Large Data Bases Conference, 2006. [Online]. Available: https://api.semanticscholar.org/CorpusID:5590663

[6] Apache Ozone, Apache Software Foundation. [Online]. Available: https://ozone.apache.org/ (accessed: 2025-05-18)

[7] Apache Parquet - Data Page Encodings, Apache Software Foundation. [Online]. Available: https://parquet.apache.org/docs/file-format/data-pages/encodings (accessed: 2025-05-18)

[8] M. Pöss, R. O. Nambiar, and D. Walrath, "Why you should run tpc-ds: A workload analysis," in Very Large Data Bases Conference, 2007. [Online]. Available: https://api.semanticscholar.org/CorpusID:2199276

[9] TPC Benchmark DS Standard Specification, Version 4.0.0, Transaction Processing Performance Council (TPC). [Online]. Available: https://www.tpc.org/TPC Documents Current Versions/pdf/TPC-DS v4.0.0.pdf (accessed: 2025-05-18)

[10] T. Ivanov and M. Pergolesi, "The impact of columnar file formats on sql-on-hadoop engine performance: A study on orc and parquet," Concurrency and Computation: Practice and Experience, vol. 32, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:203159206

[11] A. Ghazal, T. Rabl, M. Hu, F. Raab, M. Poess, A. Crolotte, and H.-A. Jacobsen, "Bigbench: towards an industry standard benchmark for big data analytics," in ACM SIGMOD Conference, 2013. [Online]. Available: https://api.semanticscholar.org/CorpusID:207202897

[12] Linux IP Routing - ip-route(8), man7.org. [Online]. Available: https://www.man7.org/linux/man-pages/man8/ip-route.8.html (accessed: 2025-05-18)

[13] M. Bosk, M. Gaji'c, S. Schwarzmann, S. Lange, and T. Zinner, "Htbqueue: A hierarchical token bucket implementation for the omnet++/inet framework," ArXiv, vol. abs/2109.12879, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:237940283

[14] Linux Traffic Control: Hierarchy Token Bucket (HTB), man7.org. [Online]. Available: https://man7.org/linux/man-pages/man8/tc-htb.8.html (accessed: 2025-05-18)

[15] Amazon EC2 Instance Types - General Purpose Instances, Amazon Web Services. [Online]. Available: https://docs.aws.amazon.com/ec2/latest/instancetypes/ (accessed: 2025-05-18)

[16] AWS Direct Connect, Amazon Web Services. [Online]. Available: https://aws.amazon.com/directconnect/ (accessed: 2025-05-18)

[17] TPC-DS Query 52, Cloudera. [Online]. Available: https://github.com/cloudera/impala-tpcds-kit/blob/master/query-templates/query52.tpl (accessed: 2025-05-18)

[18] Apache Iceberg, Apache Software Foundation. [Online]. Available: https://iceberg.apache.org/ (accessed: 2025-05-18)

[19] Apache Hudi, Apache Software Foundation. [Online]. Available: http://hudi.apache.org/ (accessed: 2025-05-18)

[20] M. Armbrust et al., "Delta Lake," Proceedings of the VLDB Endowment, vol. 13, pp. 3411–3424, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:221351108.