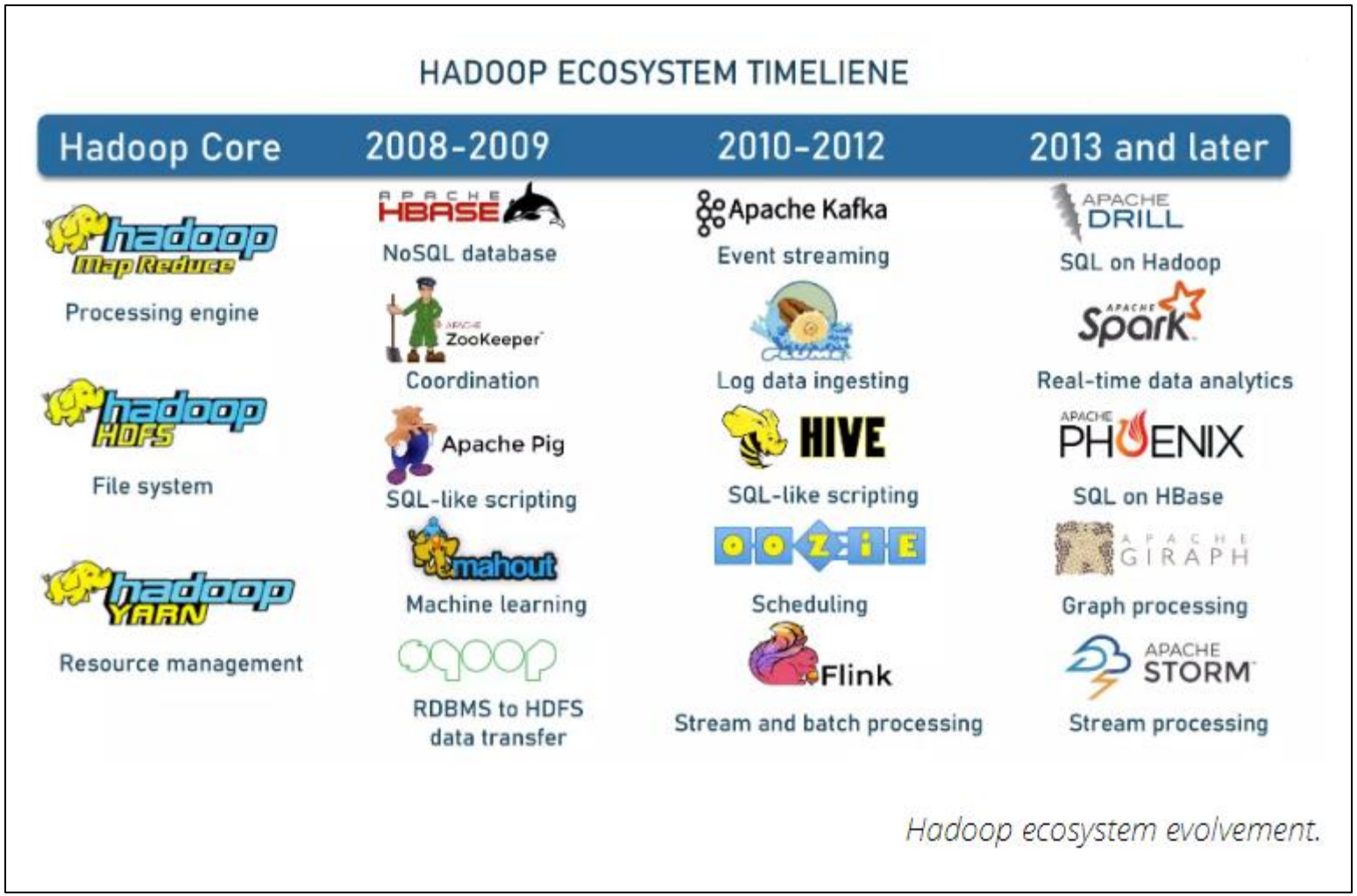
We Do Technology | SK hynix

Hadoop과 헤어지고 On premise에서 Modern Data Architecture로 빅데이터 플랫폼 구축하기

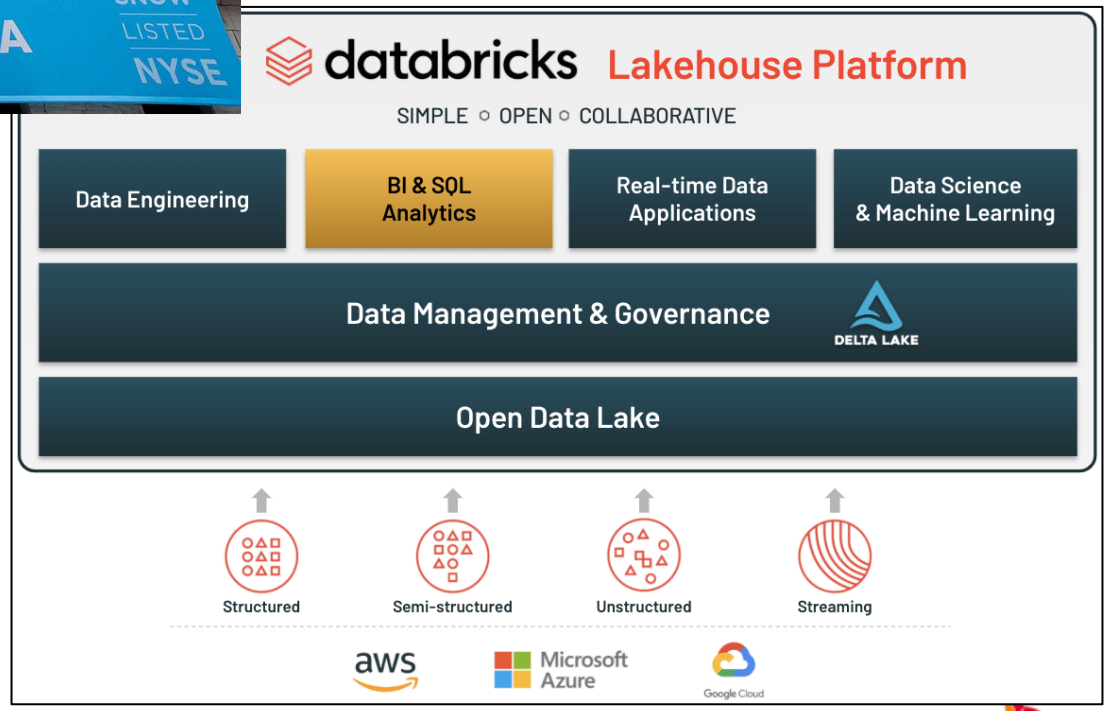
SK하이닉스 신호승
2023년 07월 03일

1. Hadoop 권불십년 화무십일홍 (權不十年 花無十日紅)



https://www.altexsoft.com/blog/hadoop-pros-cons/

2. Modern Data Platform의 등장



3. What is the Modern Data Architecture?

Lakehouse: A New Generation of Open Platforms that Unify Data Warehousing and Advanced Analytics

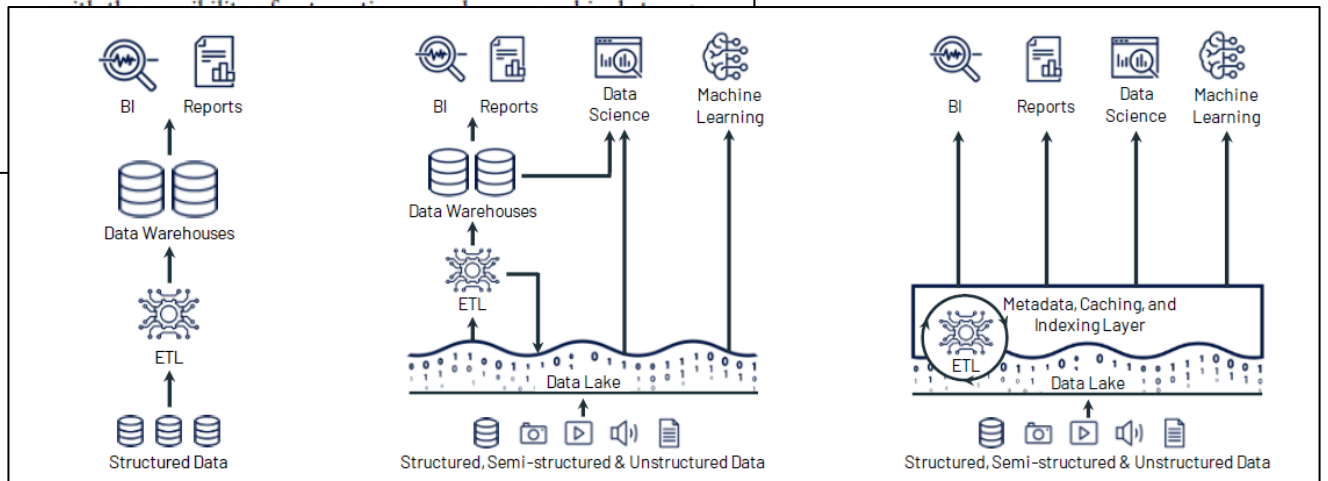
Michael Armbrust[1], Ali Ghodsi[1,2], Reynold Xin[1], Matei Zaharia[1,3]
[1]Databricks, [2]UC Berkeley, [3]Stanford University

Abstract

This paper argues that the data warehouse architecture as we know it today will wither in the coming years and be replaced by a new architectural pattern, the Lakehouse, which will (i) be based on open direct-access data formats, such as Apache Parquet, (ii) have first-class support for machine learning and data science, and (iii) offer state-of-the-art performance. Lakehouses can help address several major challenges with data warehouses, including data staleness, reliability, total cost of ownership, data lock-in, and limited use-case support. We discuss how the industry is already moving toward Lakehouses and how this shift may affect work in data management. We also report results from a Lakehouse system using Parquet that is competitive with popular cloud data warehouses on TPC-DS.

quality and governance downstream. In this architecture, a small subset of data in the lake would later be ETLed to a downstream data warehouse (such as Teradata) for the most important decision support and BI applications. The use of open formats also made data lake data directly accessible to a wide range of other analytics engines, such as machine learning systems [30, 37, 42].

From 2015 onwards, cloud data lakes, such as S3, ADLS and GCS, started replacing HDFS. They have superior durability (often >10 nines), geo-replication, and most importantly, extremely low cost



(a) First-generation platforms. (b) Current two-tier architectures. (c) Lakehouse platforms.

Figure 1: Evolution of data platform architectures to today's two-tier model (a-b) and the new Lakehouse model (c).

https://www.cidrdb.org/cidr2021/papers/cidr2021_paper17.pdf

4. Lakehouse Review(1) – 분석 플랫폼의 발전 역사

1세대 DW (1980s)		❍ 운영 DB -> 중앙 집중식 DW로 Data 수집 -> 분석적 통찰력 확보 ❍ Downstream용 BI에서 사용되기 최적화되도록 Schema-on-write를 보장
1세대 DW 문제점		❍ Appliance 장비 기반 컴퓨팅과 스토리지 결합 ❍ 기업들은 최대 사용자의 부하와 데이터 기준으로 비용 지불하고 Data가 증가하면 초고비용 발생 ❍ 데이터의 빠른 증가와 DW는 처리할 수 없는 비정형 데이터의 중요성 증가
2세대 Data Platform 등장		❍ 모든 raw data를 저장하는 저비용 구조, 개방형 파일 포맷(parquet/orc등)의 스토리지 시스템 – Hadoop 기반 Data Lake 등장 ❍ Data Lake는 모든 데이터를 저비용 구조로 빠르게 저장할 수 있는 schema-on-read 구조 ❍ 하지만 Data의 품질과 거버넌스 관리에 대한 취약점 발생 ❍ 사용자의 분석 서비스에 사용되는 BI application을 위해 Data Lake의 데이터를 ETL로 downstream DW로 전처리된 일부 데이터를 전달하는 구조 -> 2계층 구조 ❍ 개방형 파일 형식을 사용하여 ML과 같은 다른 분석 엔진에서 직접 Data Lake의 데이터에 접근
2세대 2계층 아키텍처의 문제	신뢰성	❍ Data Lake와 DW의 일관성을 유지하는 것은 매우 어렵고 많은 비용 발생 ❍ 두 시스템 간 데이터를 ETL하고, 빠른 의사결정 지원, BI에서 활용하기 위해서는 지속적인 엔지니어링 필수 ❍ ETL 단계에서는 오류가 발생할 수 있고, Data Lake와 DW 엔진간의 차이로 인해 발생하는 데이터 품질 저하
	정합성	❍ DW는 Data Lake의 비해 오래된 데이터인 경우가 많음. 새 데이터를 즉각적으로 반영하기에는 많은 비용 발생 ❍ 운영 시스템에서 DW로 직접 데이터를 적재해 쿼리에 사용할 수 있던 1세대 분석 시스템에 비해 단점 ❍ 지속적인 ETL 필요, 데이터 복제로 인한 스토리지 비용 증가, metadata 정합성 불일치, 일반적으로 DW는 매우 비싼 구조
	고급 분석 제한	❍ DW는 Data Lake의 비해 오래된 데이터인 경우가 많음. 새 데이터를 즉각적으로 반영하기에는 많은 비용 발생 ❍ 운영 시스템에서 DW로 직접 데이터를 적재해 쿼리에 사용할 수 있던 1세대 분석 시스템에 비해 단점 ❍ 지속적인 ETL 필요, 데이터 복제로 인한 스토리지 비용 증가, metadata 정합성 불일치, 일반적으로 DW는 매우 비싼 구조

4. Lakehouse Review(2) – Implementing a Lakehouse System

핵심 IDEA

- Apache parquet와 같은 표준 파일 형식을 사용하여 저비용 object storage(S3)에 데이터 저장
- Object Storage의 최상단 계층에는 트랜잭션 metadata를 위한 구현체를 두어 테이블의 버전 등을 정의
- Metadata 계층에서 ACID 트랜잭션과 버전 관리 같은 DBMS 관리 기능 구현
- 대용량의 데이터를 사용자가 직접 접근할 수 있도록 표준 파일 형식 사용
- Delta Lake와 Iceberg를 통한 구현

Good SQL 성능

- Metadata 계층은 데이터 관리 기능을 확보할 수 있으나, 좋은 SQL 성능을 보장하지 않음
- DW에서 사용한 다양한 기술을 적용
  - SSD를 통한 Hot data 저장
  - 통계 정보 관리
  - Index와 같은 효율적인 접근 방식 구현
  - Data 형식과 컴퓨팅 엔진간의 최적화
  - Caching과 보조적인 데이터 구조를 포함한 변경되지 않은 파일에 대한 Index와 통계
  - 데이터 구조 최적화

고급 분석용 데이터 관리 기능

- TensorFlow나 Spark MLlib와 같은 라이브러리들은 선언적 DataFrame API를 사용하고 데이터 관리 기능을 제공
- API는 Parquet와 같은 Data Lake의 파일을 직접 접근 가능
- Parquet의 데이터를 추출하기 위해 metadata에서 추출한 데이터를 ML 라이브러리에 전달만 하면 통합 가능
- 즉 API를 사용하여 Parquet의 데이터를 DF로 바로 제공
- 이 과정에서 Optimizer가 연산 계획을 자동으로 최적화하고 캐싱 등을 활용

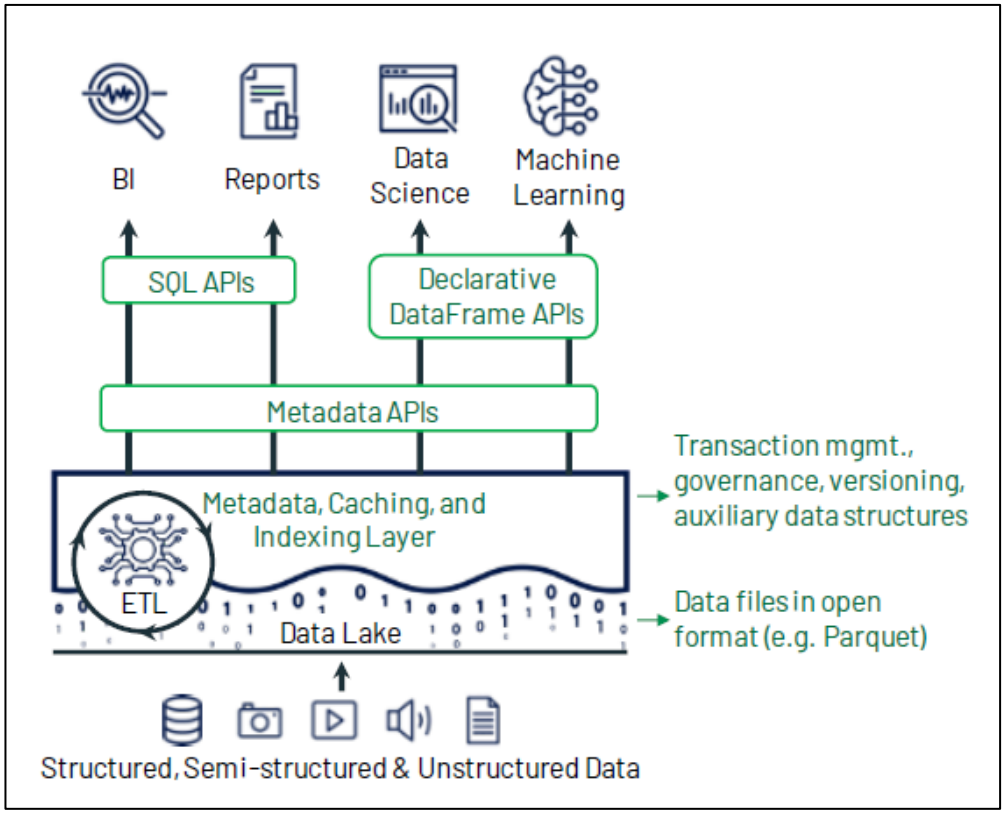
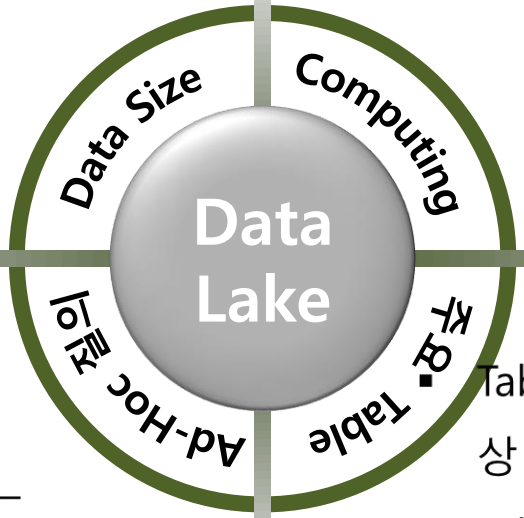


Figure 2: Example Lakehouse system design, with key components shown in green. The system centers around a metadata layer such as Delta Lake that adds transactions, versioning, and auxiliary data structures over files in an open format, and can be queried with diverse APIs and engines.

5. SK하이닉스의 빅데이터 – only Data Lake

- Data Size – 8PB
- ETL Table – 13,000개
- DS/분석가 – 2000명
- AI Product - 100여개



- CPU – 15000 cores
- Memory – 200 TB
- 스토리지 – ALL Flash Disk/NAS/S3
- 네트워크 – Bigdata 클러스터 전용망

- Memory – 3PB/day
- ad-hoc Data API 요청 건수 – 700,000건/day
- 평균 HDFS Scan size – 5PB/day
- 평균 HDFS Write size – 5TB/day(user mart DW)

- Table 1 - 1일 130억건, 분석가들은 이상 증상 탐지를 위해 주로 6개월치 조회
- Table 2 - 1일 2500억건, 거의 모든 데이터는 기본 6개월 단위로 분석
- Table 3 - 1일 파티션 1만개 + 200억건 -> 변경 데이터 처리 필수(최근 30일치)
- 과거에는 100만개 넘어가는 hive 파티션 테이블을 관리했으나 모두 최적화

SK hynix

6. Hadoop은 Legacy 기술의 대표 병목점

What are the challenges with Hadoop architectures?

- **Complexity** - Hadoop is a low-level, Java-based framework that can be overly complex and difficult for end-users to work with. Hadoop architectures can also require significant expertise and resources to set up, maintain, and upgrade.
- **Performance** - Hadoop uses frequent reads and writes to disk to perform computations, which is time-consuming and inefficient compared to frameworks that aim to store and process data in memory as much as possible, like Apache Spark.
- **Long-term viability** - In 2019, the world saw a massive unraveling within the Hadoop sphere. Google, whose seminal 2004 paper on MapReduce underpinned the creation of Apache Hadoop, stopped using MapReduce altogether, as tweeted by Google SVP of Technical Infrastructure, Urs Hölzle. There were also some very high-profile mergers and acquisitions in the world of Hadoop. Furthermore, in 2020, a leading Hadoop provider shifted its product set away from being Hadoop-centric, as Hadoop is now thought of as "more of a philosophy than a technology." Lastly, 2021 has been a year of interesting changes. In April 2021, the Apache Software Foundation announced the retirement of ten projects from the Hadoop ecosystem. Then in June 2021, Cloudera agrees to private. The impact of this decision on Hadoop users is still to be seen. This growing collection of concerns paired with the accelerated need to digitize has encouraged many companies to re-evaluate their relationship with Hadoop.

- Hadoop Eco의 dependency 복잡성과 의존성 종속 문제
- Mission Critical을 지원할 수 없는 기술 구조
- 유지보수에 대한 어려움
- 하지만 대규모 데이터의 안정적인 처리 관점에서는 대안이 거의 없음
- 대안을 찾기 위한 다양한 시도 필요

SK hynix

7. 진화하는 데이터 서비스 요구사항

비즈니스 요구사항

- 현업 시스템과 Data Lake 사이의 정합성 확보
- 정형/비정형/실시간 데이터 통합 분석
- 변경 데이터 처리 가능한 빅데이터
- Data 적재 주기 단축
- Mission Critical 분석 시스템 지원
- Data Lake와 DW 데이터의 통합
- Table Metadata 통합 Map과 검색

플랫폼 요구사항

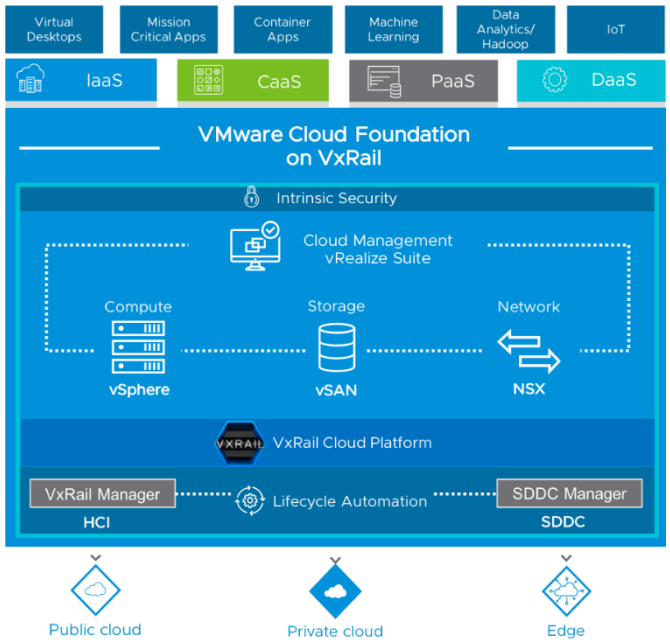
- Snowflake와 Databricks 같은 Managed 플랫폼 서비스
- 현업 시스템 – Data Lake – DW 정합성 일치
- 개별 분석 시스템 전용 워크로드
- 자유로운 사용자 Mart DB 생성과 시스템향으로 사용
- 단일 SQL 인터페이스 사용
- 쉽게 접근 가능한 공개 스토리지 기술
- 워크플로우 기반 빅데이터 분석 환경

8. Hadoop을 버리기 위한 노력과 경험

초기 Data Lake Hadoop on VM

Hadoop 아키텍처의 Risk를 모두 경험

- VMware 가상 환경 기반의 Hadoop 컴퓨팅
- Shuffle data는 remote vSAN 스토리지
- NSX 가상 네트워크 환경
- Isilon remote HDFS(secondary Hadoop 설정)



모든 서비스를 Remote로 구성

Hadoop 구성 서비스를 완전히 분리 검증

- Bare Metal 노드 기반 Hadoop Eco 구성
- Node Manager와 Datanode를 분리
- 컴퓨팅과 데이터 완전 분리
- Data Lake의 모든 Table은 remote location 사용
- Cache 아키텍처와 nvme shuffle 적극 활용
- 기존 VMWare 클러스터 대비 성능 비교
- Node Manager와 datanode를 통합했을때와 비교

성능 검증

- **특정 Data의 Skew가 무조건 발생하는 multi join과 Grouping 조건이 많은 질의 환경에서는 shuffle 영역의 성능이 더욱 중요**
- **컴퓨팅과 데이터를 분리할 경우 유지 보수 관점에서 매우 큰 장점 획득**
- **기존 Hadoop 개발자의 오해의 범위 확인 – 데이터와 가까운 노드에서 컴퓨팅?**

Hadoop 없는 세상

Hadoop은 데이터 저장소 역할만 부여

- YARN의 스케줄링 기능만 사용
- 모든 Table은 remote location 사용
- 기존 순수 Hadoop 기반 Hive 연산을 모두 제거
- Hadoop 기반 전사 ETL Pipeline 대체
- Tez/Spark의 DAG로 RM 기능 대체
- HDFS 저장소를 S3로 대체하여 성능 확인

- **Hadoop 없이도 제조기반 빅데이터 환경 개발 검증 완료**
- **Hive Metastore는 대안 없음**
- **Remote의 모든 data locatio을 하나의 통합 HMS Catalog로 통합하여 Governance 관리**

SK hynix

9. 데이터 플랫폼 현대화를 위한 품질 속성 고민

1. 제조 데이터 규모와 복잡성을 처리할 수 있는 Cloud Native 아키텍처 구성이 가능할 것인가?

가용성

- Hadoop 기반의 빅데이터 분산 시스템과 K8S 기반 Cloud Native 분산 시스템의 결합 안정성
- 특히 MPP와 Hadoop은 Cloud Native 아키텍처 기술이 아니기 때문에 K8S의 네트워크 부하 가중 발생
- Mission Critical 서비스를 위한 가용성 정보 부족

2. Cloud Native 아키텍처에서 빅데이터 서비스의 결함허용 범위와 장애 발생시 복구 방안

결함허용 장애복구

- 결함 허용 범위와 대응 전략 준비
- 빅데이터 컴포넌트의 회복 탄력성과 관련된 개발 범위
- Mission Critical 서비스 장애 발생시 복구에 걸리는 시간 측정 방안

3. 데이터 정합성 준수를 위한 서비스 구현

정합성

- Full ACID를 지원하는 빅데이터 서비스를 위한 Lakehouse 아키텍처 적용
- Data Governance의 일관성 확보를 위해 통합 catalog와 metadata 자동화
- 권한/인증 체계 따른 데이터의 데이터 보호

4. Cloud Native 기반 Big Data 구축

변경 용이성

- 무중단 업그레이드
- 서비스 컴포넌트 대체 난이도
- 클러스터 Provisioning 도구의 편의성

3. Hadoop 대안 서비스 성능 확보

성능

- **HDFS를 대체하는 S3 스토리지의 분산 처리 성능** -> S3는 느리다는 편견을 바꾸는 과정 필수
- Hadoop 컴퓨팅을 대체하는 **Serverless Spark와 Impala의 성능**
- YARN 스케줄러를 대체하는 K8S 스케줄러의 성능

10. 분산 아키텍처 관점의 핵심 Quality Attribute

성능 최적화와 Mission Critical Business 지원을 위한 아키텍처 고려

	주요 속성	아키텍처 검증 요소
인프라 아키텍처	리눅스 System Call	• read(), write(), open(), seek(), fsync(), fflush(), mmap(), mlock(), mtdop(), 페이지 캐시, short-circuit read, zero-copy read
	컴퓨팅용 스토리지	• On disk cache, 서비스별 disk 레이아웃과 quarter limit, block size과 IO 튜닝
	Big Data Network	• East-West 트래픽 분산용 회복성 있는 Spine-Leaf Network, 100G망, sNAT 대응 스위치, LB, NTP 동기화
스토리지 아키텍처	결함허용	• 물리 Node, 디스크, Pod, Control Plane/Worker Node 장애와 K8S 전체 유실 상황에 대한 fail over 검증
	고가용성	• 서비스별 HA, PV 스토리지, 수평적/수직적/systemic 고가용성 항목 상세 검증
Modem Data 아키텍처 통합	통합 Catalog	• 통합 catalog schema 관리, Lakehouse 아키텍처 적용, 파티셔닝 정보 최적화,
	Governance	• 권한/인증 통합, metadata 자동화
성능	자원 효율화	• Serverless 아키텍처, non native doud 컴포넌트의 custom화,
	분산 성능	• Nvme S3 Remote Shuffle, scratch disk, spilled to disk, multi thread DOP, remote cache, Rank/Top 알고리즘 최적화

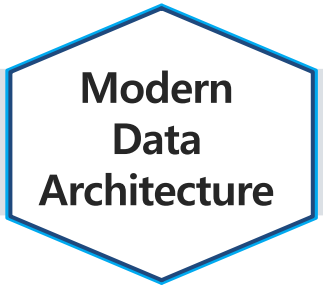
SK hynix

11. Open Source 기반 참조 아키텍처

01
아키텍처
목표

02
제약 사항

03
참조 모델
기술 도입
(Open Source)



CLOUD NATIVE
COMPUTING FOUNDATION

개발자 3명으로 모든 개발

- 가장 검증된 기술
- Open Source Community의 활성화
- 성능 관점의 아키텍처 우월성
- 기존 Legacy Hadoop 플랫폼도 공존 필수
- CI/CD Pipeline 자동화를 위한 git-sync 수정
- 빅데이터 성능과 자원 효율화를 위한 최적화

✓ **Lakehouse 기술**
- Apache Iceberg

✓ **HDFS -> S3**
- minIO S3 with directPV

✓ **Hadoop Computing 대안**
- Apache Spark
- Apache Impala
- Multi Thread DOP

✓ **Serverless 아키텍처**
- Knative

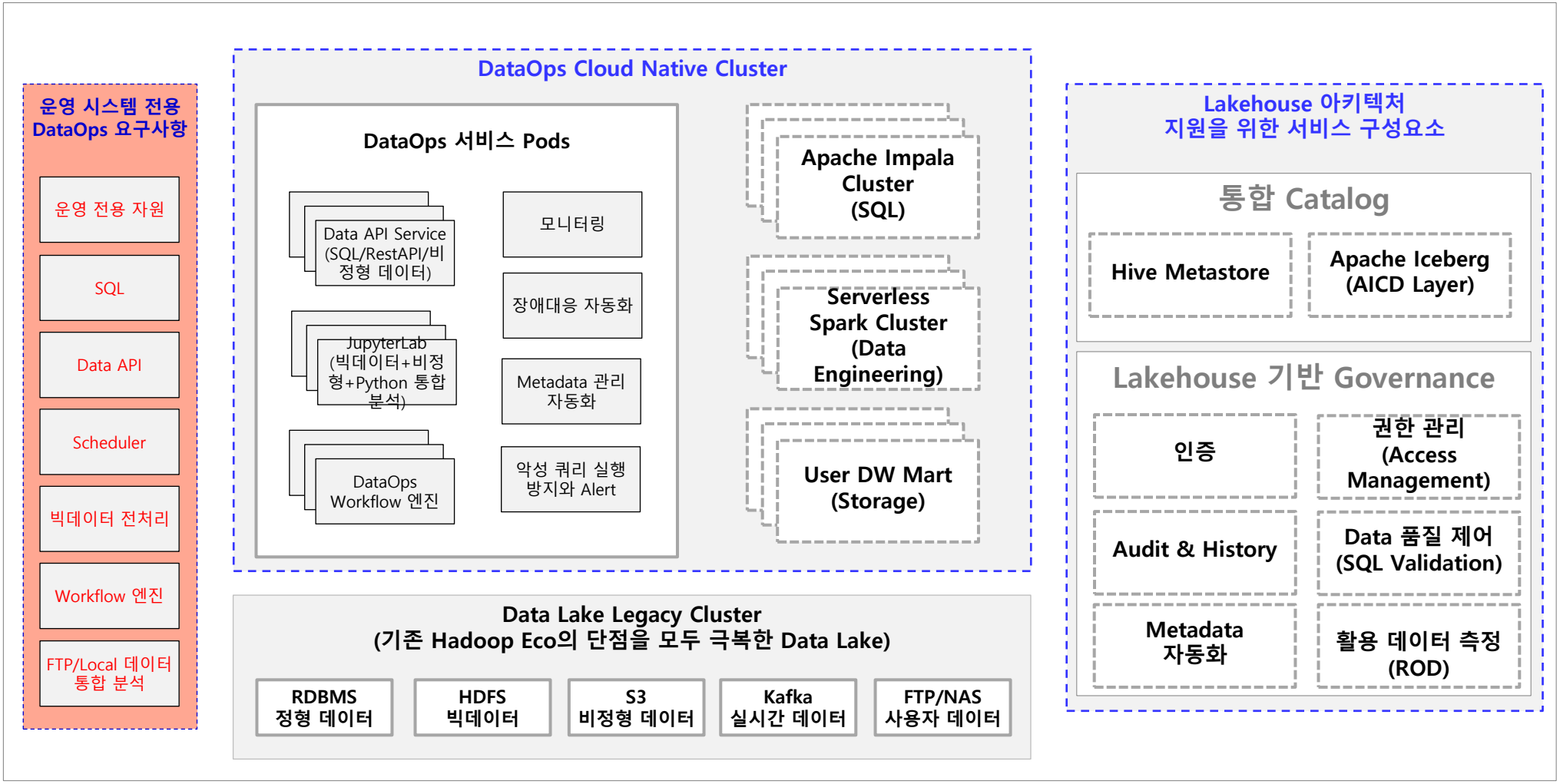
✓ **성능 우위의 네트워크 구조**
- Calico IPVS mode vxlan
- 빅데이터 Block File IO 최적화

✓ **Spark 성능 최적화**
- nvme 기반 S3 shuffle storage
- uber Remote Shuffle Service

SK hynix

12. Modern Data Platform Architecture overview

기존 Data Lake + Modern Data Platform 통합 환경



13. Modern Data Architecture 기반 Best Practice

실시간 anomaly score 탐지 AI Product을 위한 Data Pipeline