성공적인 데이터 분석 및 활용을 위한 3가지 기술 전략

Data Federation, 데이터 모델링, In-DB 고급 분석

VMware Tanzu Data Leader, Korea 이상희 상무

VMware Korea

August 30, 2022

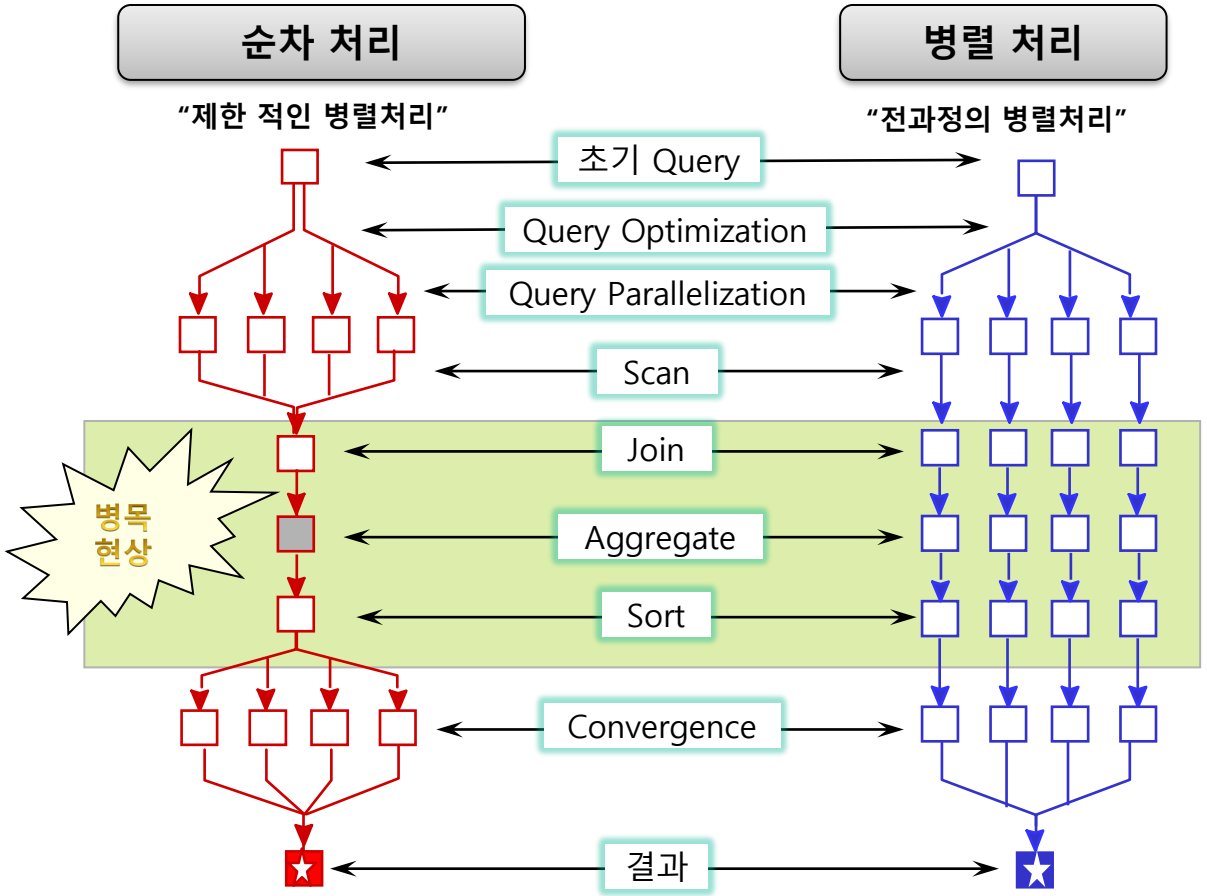
Agenda

1. 최근 분석 트렌드
2. 성공적인 데이터 분석을 위한 3가지 기술 전략
3. 적용 솔루션 소개 및 고객 사례
4. Private Cloud 분석 레퍼런스 아키텍처
5. 빅데이터 분석 플랫폼 고려 사항

최근 분석 트렌드

빅데이터 분석 트렌드 1

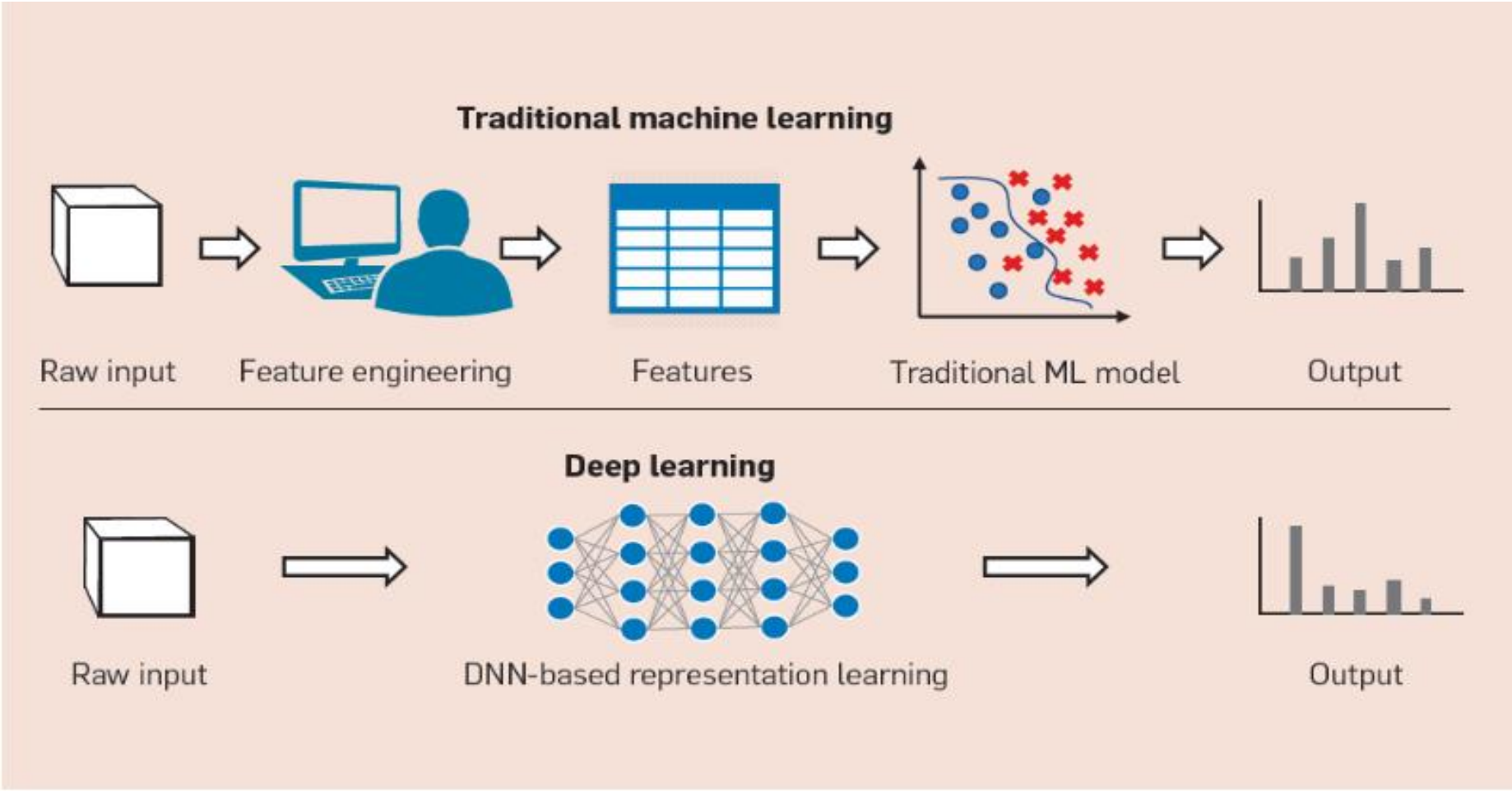
대용량 데이터 병렬 데이터 처리



Greenplum
VERTICA
APACHE hadoop
TERADATA
NETEZZA
hadoop HDFS
HIVE
Google Big Query
ORACLE Exadata
APACHE Spark
snowflake
cloudera IMPALA
AMAZON REDSHIFT

빅데이터 분석 트렌드 2

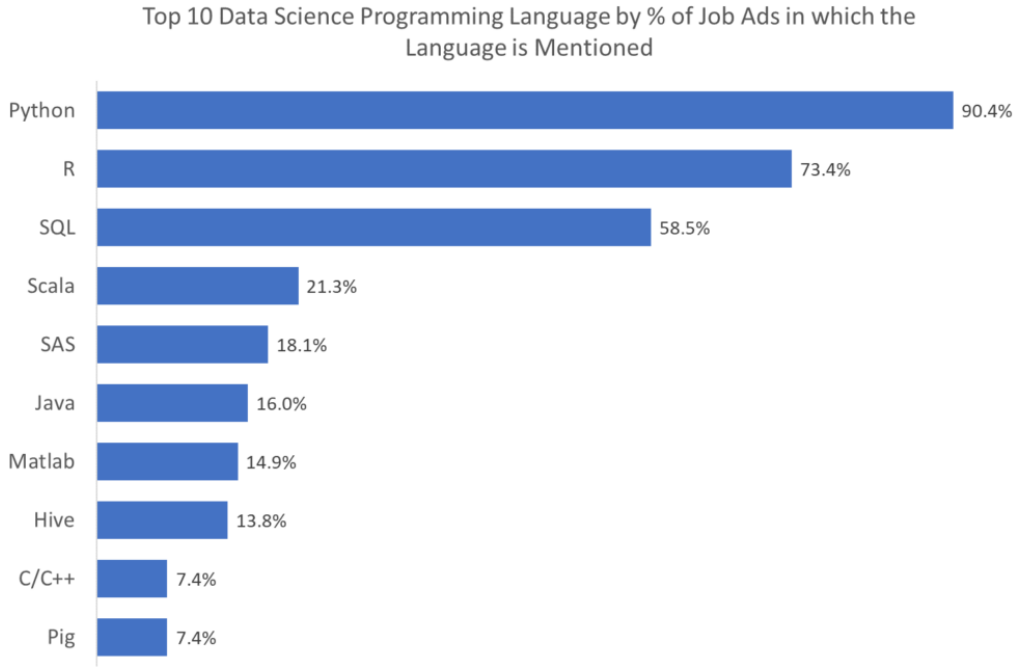
빅데이터를 기반으로 머신러닝/딥러닝 분석, 학습하여 결과를 예측



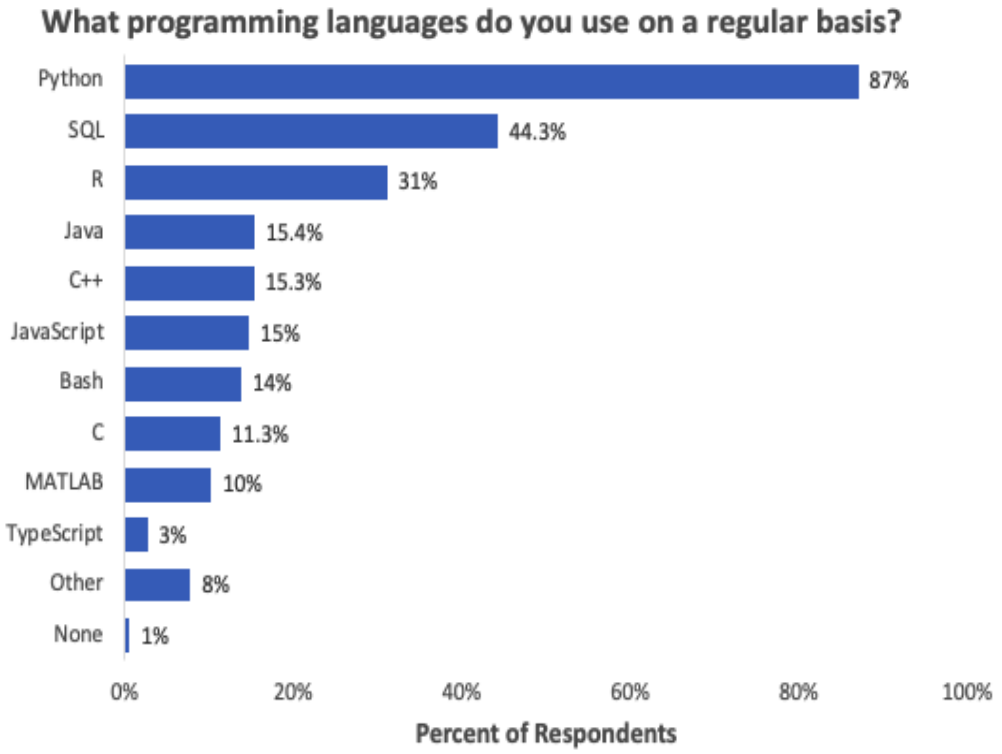
빅데이터 분석 트렌드 3

데이터 사이언티스트들이 사용하는 분석 언어

[데이터 사이언티스트 채용 시 필요한 랭귀지 순위[1]]



[데이터 사이언티스트 사용하는 랭귀지 순위[2]]



1) https://towardsdatascience.com/team-r-or-team-python-2f8cf04310e6
2) https://businessoverbroadway.com/2020/06/29/usage-of-programming-languages-by-data-scientists-python-grows-while-r-weakens/

고객사가 직면한 상황

이미 구축된 분석 시스템이 효율적으로 운영되고 있는가?

고급 분석 요건 증가

IT 전문가의 도움 없이도 손쉽게 통합된 최신의 데이터를 가지고 머신러닝과 같은 기법으로 빠르게 분석하고 싶다

실시간 분석 요구사항 증가

우리 기업의 분석 시스템은 분석가의 요구사항을 충족할 수 있도록 효율적으로 운영되고 있는가?

EDW 시스템의 사용 제약

- ✓ EDW 시스템의 높은 SLA 준수
- ✓ 대용량 데이터 분석을 위한 Ad-hoc 분석 제약

분석 플랫폼간의 연동 제한

- ✓ 다수의 에코시스템에 대한 운영 관리 복잡
- ✓ 전문가의 지원이 항상 필요
- ✓ 분석 시스템간 또는 서비스 시스템 연동 제한 (편의성, 성능)

ML/DL 분석 성능 한계

- ✓ 지속적인 데이터 증가
- ✓ ML/DL 분석으로 모델 개발시 장시간 소요
- ✓ 시간 부족에 따른 최적 모델 개발 제약

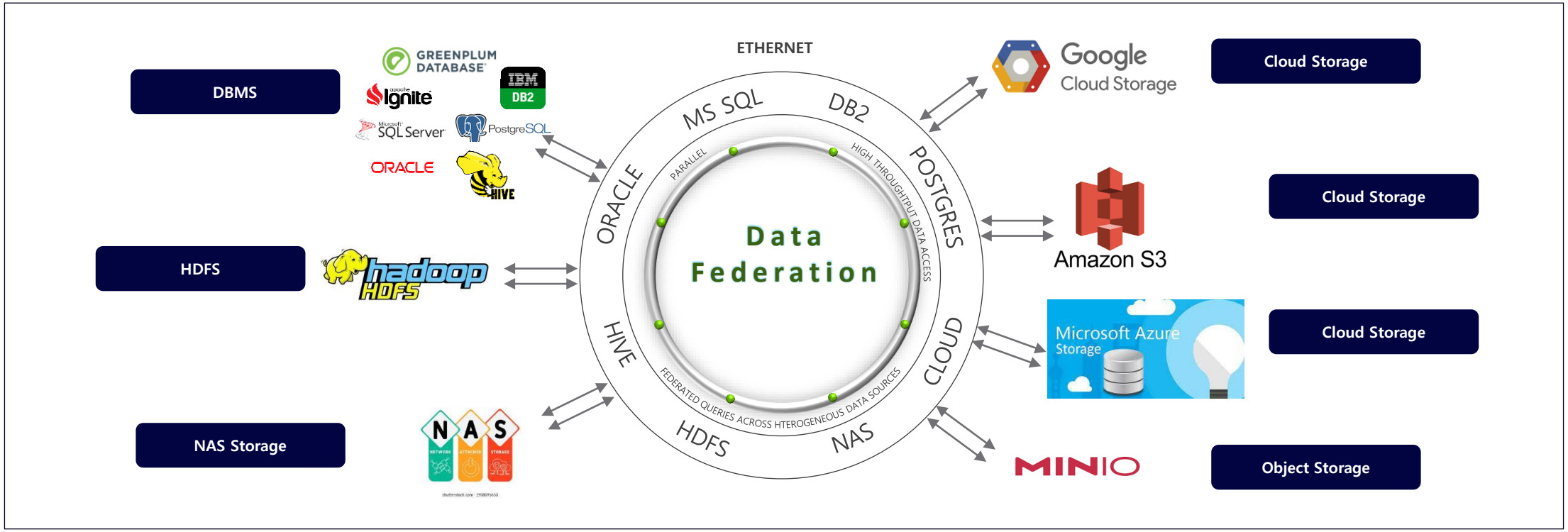
성공적인 데이터 분석을 위한 3가지 기술 전략

1. Data Federation

다양한 외부 데이터 소스를 ETL 없이 하나의 분석 시스템에서 연산 수행

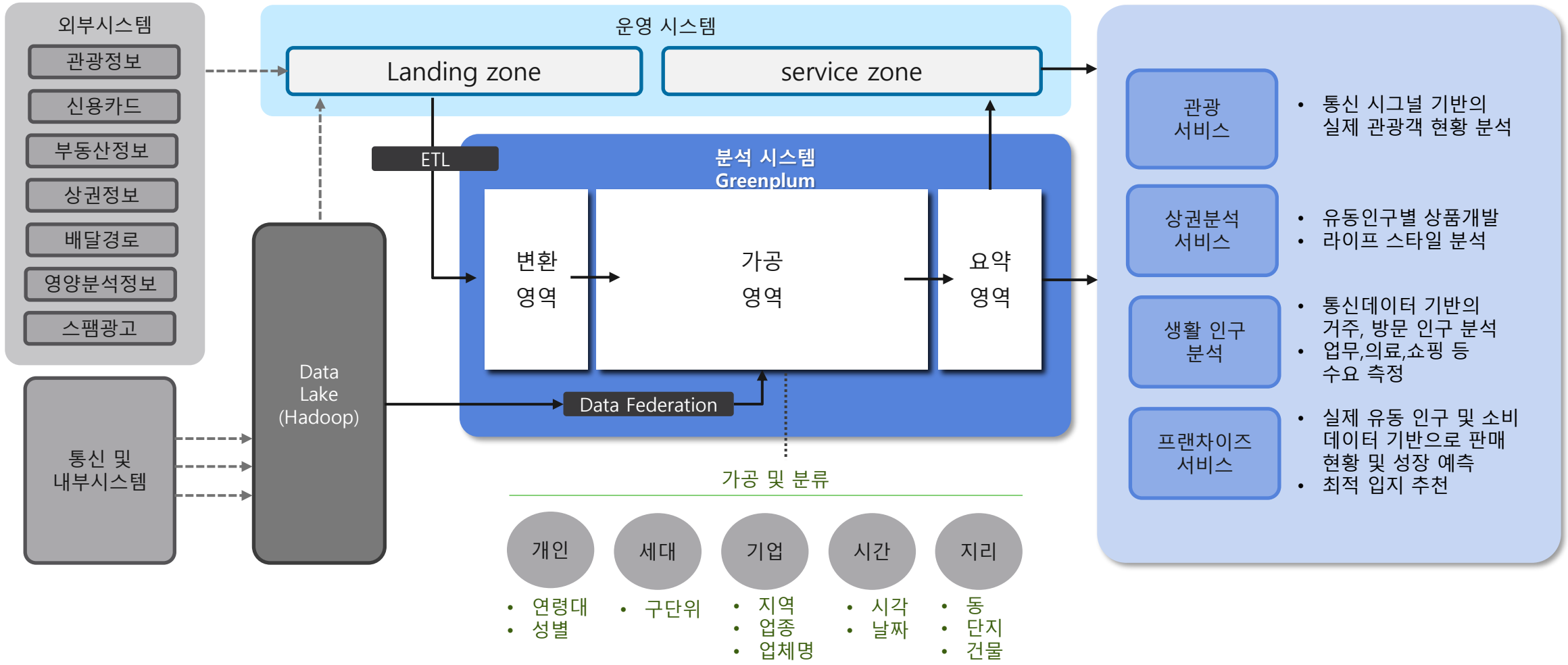
❑ **Data Federation 주요 고려 사항**

- 다양한 외부 시스템 데이터 소스 연계
- External Table 처럼 연동이 쉬워야 하며, 데이터 Read/Write 기능 제공
- 데이터 적재/추출 시 병렬 처리를 통한 고속 성능 제공



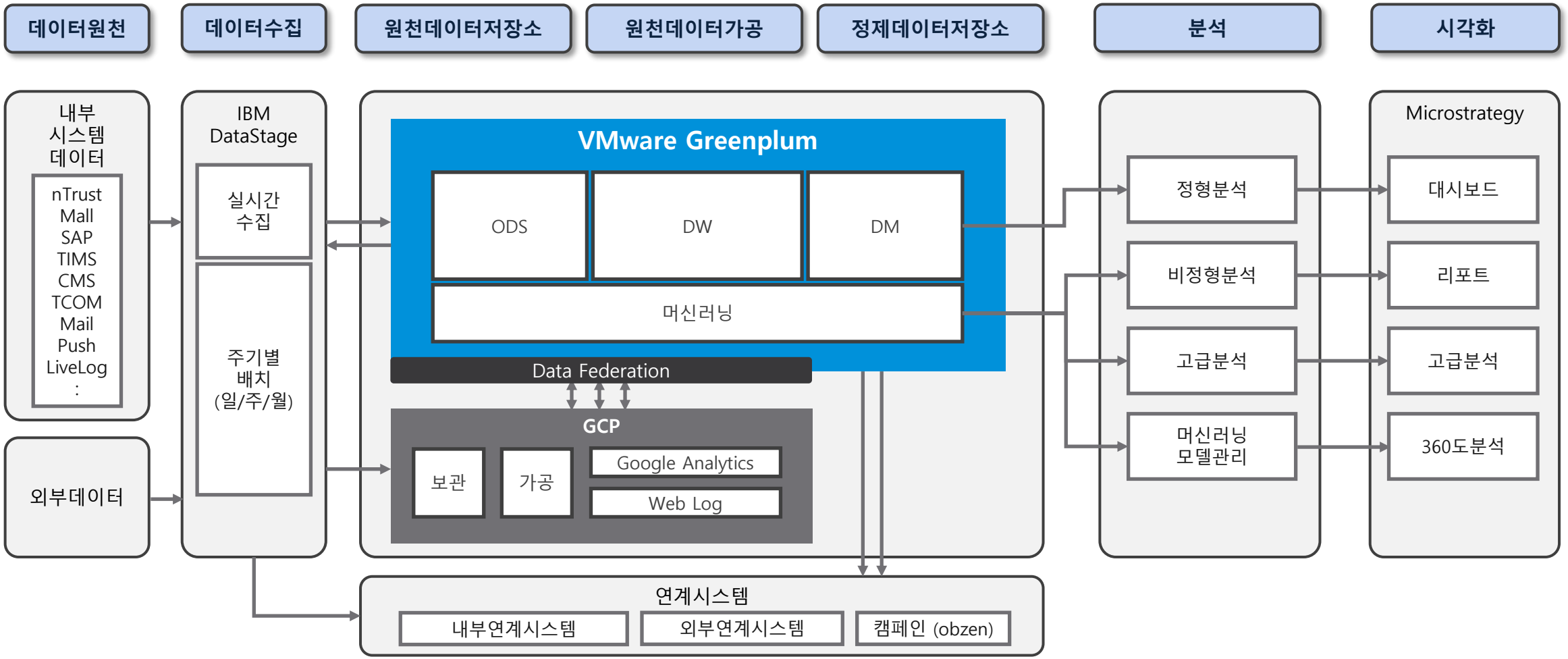
1. Data Federation

하둡과의 연동 분석한 국내 통신 사례



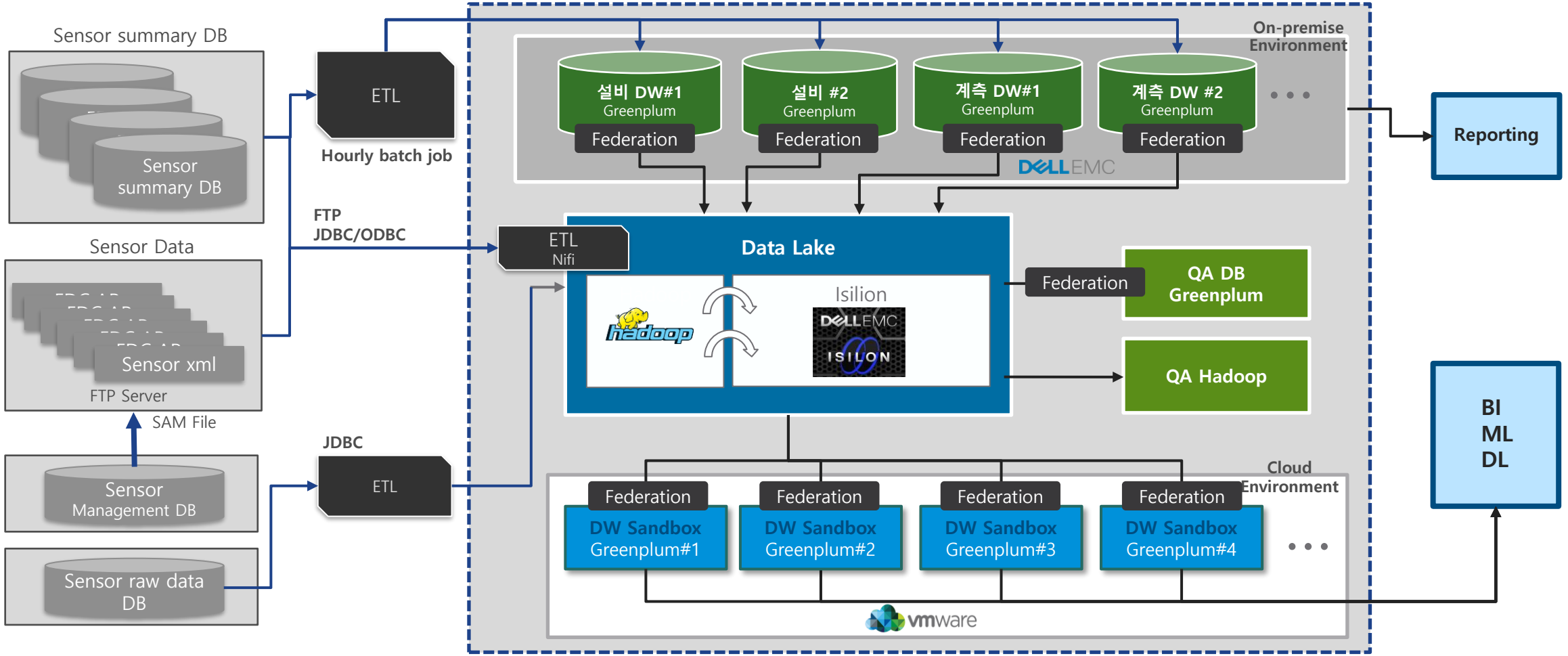
1. Data Federation

클라우드 오브젝트 스토리지와 연동 분석한 국내 유통 사례



1. Data Federation

하둡과의 연동 분석한 국내 제조 사례

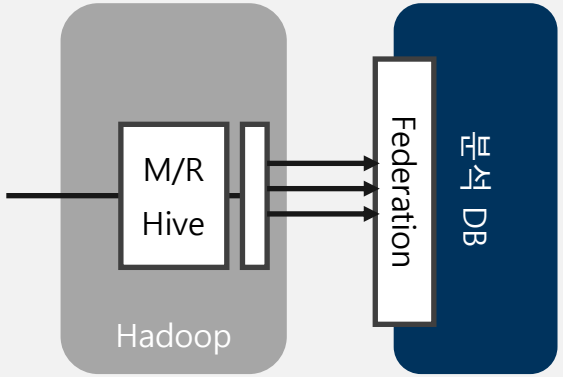


Data Lake 활용 방안

Data Federation 활용

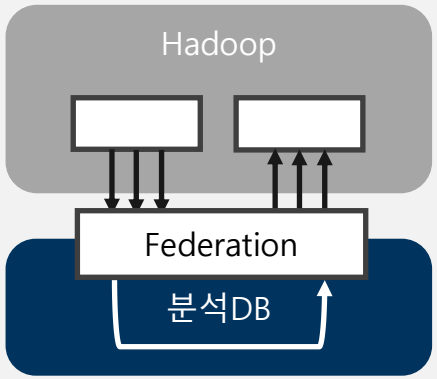
MapReduce/Hive를 사용하여 초기 데이터 가공 후 분석 DB로 적재

- 원천 데이터가 하둡 데이터 레이크로 수집
- MapReduce를 거쳐 1차적으로 데이터 가공
- HDFS의 파일 또는 Hive를 통해 분석 DB에서 직접 액세스하여 추출



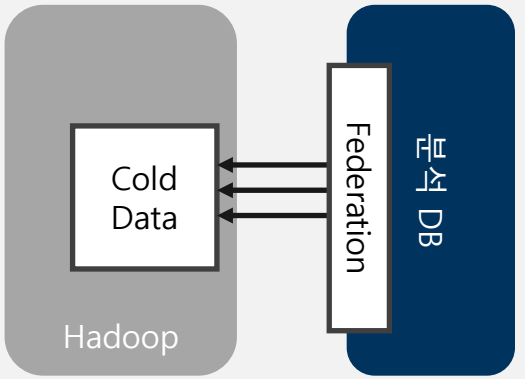
Hadoop의 파일 데이터를 분석 DB에서 쉽게 가공

- HDFS의 데이터 파일을 분석 DB에서 연산, 가공 처리하여 다시 HDFS로 저장
- 데이터에 대한 컴퓨팅 리소스를 분리하여 Hadoop시스템의 부하를 감소
- 하둡 파일에 대한 연산을 SQL로 쉽게 사용



분석 시스템의 데이터 수명 관리를 위한 아카이브 영역으로 활용

- 분석 시스템의 Old 데이터 또는 히스토리 데이터를 Hadoop 데이터 레이크에 보관
- 보관된 데이터는 필요시 external 테이블로 데이터 적재 필요 없이 직접 조회

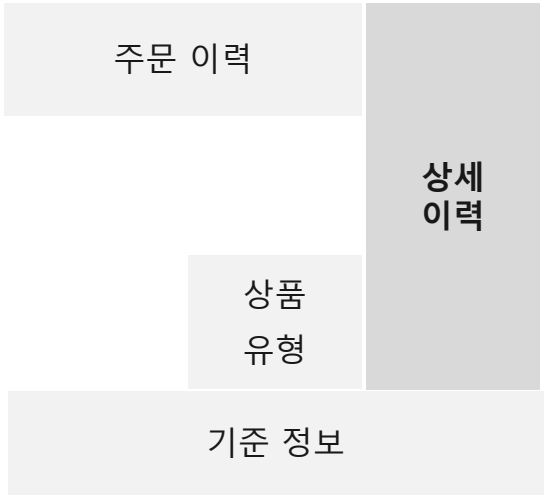


2. 빅데이터 처리를 위한 효과적인 데이터 모델링 적용

DW, DM 분석에 최적화된 데이터 모델

OLTP 최적화된 데이터 모델

개별 관계 테이블



- 이력성 정보, 상품 기준 정보, 상품 유형 관계에 따라 개별 테이블로 구성
- 데이터 조회 시 다수 Join 발생

변환

DW 분석에 최적화된 데이터 모델

DW 성능 최적화

통합 테이블

- 관계가 성립된 후 통합 테이블 구성
- N:M 구조로 row수가 많은 형태로 구성
- 데이터 조회 필요한 Join 연산을 최소화하여 조회 성능 향상

2. 빅데이터 처리를 위한 효과적인 데이터 모델링 적용

빅데이터 분석에 최적화된 데이터 모델

DW 분석에 최적화된 데이터 모델

통합 테이블

주문 이력	상품 유형	**상세 이력 (Row)**	기준정보

변환

빅데이터 분석에 최적화된 데이터 모델

빅데이터 성능 최적화

분석용 통합 테이블

주문 이력	상품 유형	**상세 이력 (Arr)**	기준정보

- 중복데이터가 많아 데이터 저장 공간 낭비
- 중복데이터가 많아 Index 사이즈 증가
- Row수가 많아서 데이터 검색 시 부하 발생

- 직렬로 연결하여 배열 구조로 변환
- 데이터 row수가 줄어들어 검색 성능 대폭 향상
- 데이터 Transpose가 되어 ML/DL 편리한 적용

2. 빅데이터 처리를 위한 효과적인 데이터 모델링 적용

대용량 데이터 처리를 위한 데이터 직렬화 예시

데이터 직렬화 변환

```
select  a.o_orderkey, a.o_custkey
        , array_agg(b.l_partkey) partkey_arr
        , array_agg(b.l_suppkey) suppkey_arr
        , array_agg(b.l_quantity) quantity_arr
        , array_agg(b.l_extendedprice) extendedprice_arr
        , array_agg(b.l_tax) tax_arr
        , array_agg(b.l_linestatus) linestatus_arr
        , array_agg(b.l_shipdate) shipdate_arr
from    edu_sch.orders a,
        edu_sch.lineitem b
where   a.o_orderkey =b.l_orderkey
group by a.o_orderkey
          , a.o_custkey
```

	o_orderkey	o_custkey	partkey_arr	suppkey_arr	quantity_arr
1	3	123,314	{4297,29380,62143,183095,19(	{1798,1883,9662,650,6540,3	{45.0,2.0,26.0,2
2	66	129,200	{115118,173489,115118,173489,'	{7630,3490,7630,3490,7630,	{31.0,41.0,31.0,
3	70	64,340	{37131,55655,45734,196156,17	{2138,3171,743,1195,7361,914	{37.0,19.0,11.0,1
4	97	21,061	{77699,77699,119477,119477,4!	{5221,5221,1989,1989,2073,2	{19.0,19.0,13.0,
5	99	88,910	{87114,108338,134082,123766,	{4639,849,1622,3767,1622,3:	{10.0,36.0,42.0
6	129	71,134	{31373,168569,2867,39444,28(	{8883,3602,5368,1948,5368	{24.0,1.0,46.0,3
7	130	36,964	{115635,128816,69130,1739,11!	{3169,8817,4143,4240,1861,8	{13.0,14.0,31.0,
8	131	92,749	{189021,44255,167505,167505,	{1540,9264,22,22,1540,9264	{4.0,50.0,45.0,4
9	193	79,061	{153954,92638,93878,92638,1	{1500,5148,6388,5148,1500,(	{15.0,9.0,23.0,9
10	224	2,476	{189967,93857,166377,50010,1	{7522,8876,1410,7526,1120,1'	{41.0,45.0,12.0,

직렬화에서 Row로 전환

```
SELECT o_orderkey, o_custkey
        , unnest(partkey_arr) partkey
        , unnest(suppkey_arr) suppkey
        , unnest(quantity_arr) quantity
        , unnest(extendedprice_arr) extendedprice
        , unnest(tax_arr) tax
        , unnest(linestatus_arr) linestatus
        , unnest(shipdate_arr)  shipdate
FROM edu_sch.orders_lineitem
```

	o_orderkey	o_custkey	partkey	suppkey	quantity	extendedprice	ta
1	3	123,314	183,095	650	28	32,986.52	
2	3	123,314	128,449	3,474	27	39,890.88	
3	3	123,314	62,143	9,662	26	28,733.64	
4	3	123,314	183,095	650	28	32,986.52	
5	3	123,314	19,036	6,540	49	46,796.47	
6	3	123,314	29,380	1,883	2	2,618.76	
7	3	123,314	62,143	9,662	26	28,733.64	
8	3	123,314	29,380	1,883	2	2,618.76	
9	3	123,314	19,036	6,540	49	46,796.47	
10	3	123,314	4,297	1,798	45	54,058.05	

2. 빅데이터 처리를 위한 효과적인 데이터 모델링 적용

대용량 데이터 처리를 위한 데이터 모델링 적용시 성능 개선 사항

건수	용량 (Table + Index)	인덱스	쿼리 성능
1/95 ↓	1/4 ↓	1/20 ↓	6 ↑

	테이블 구분	데이터 용량(MB)	인덱스 용량 (MB)	전체 용량(MB)	건수	비즈니스 쿼리
개별 관계 테이블 모델	주문 이력	90	60	150	123,787	
	상세 이력	938	4,079	5,017	86,996,677	
	합계	1,027	4,139	5,167	88,231,464	3.7 sec
역정규화/ 직렬화 모델	주문/상세 통합	944	203	1,147	933,061	0.6 sec

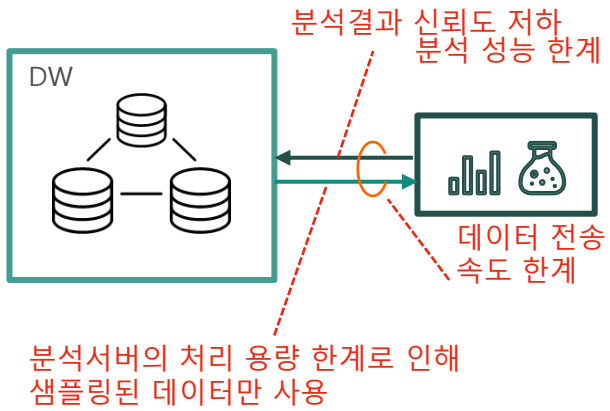
3. In-Database 분석

In-Database 개념

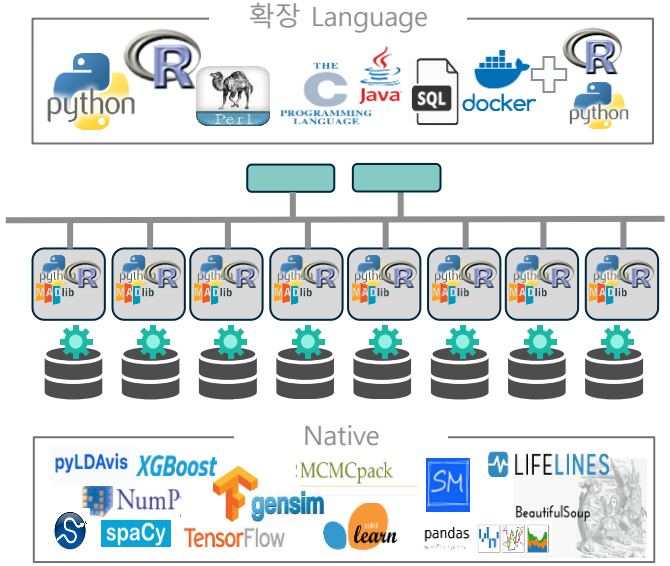
In-Database ML/DL 병렬 분석

ML/AI의 최신 고급 분석 라이브러리는 이용
데이터 이동 없이 병렬로 분석하여 시간 단축 및 대용량 데이터 분석으로 품질 향상

일반적인 싱글 노드 분석 방식



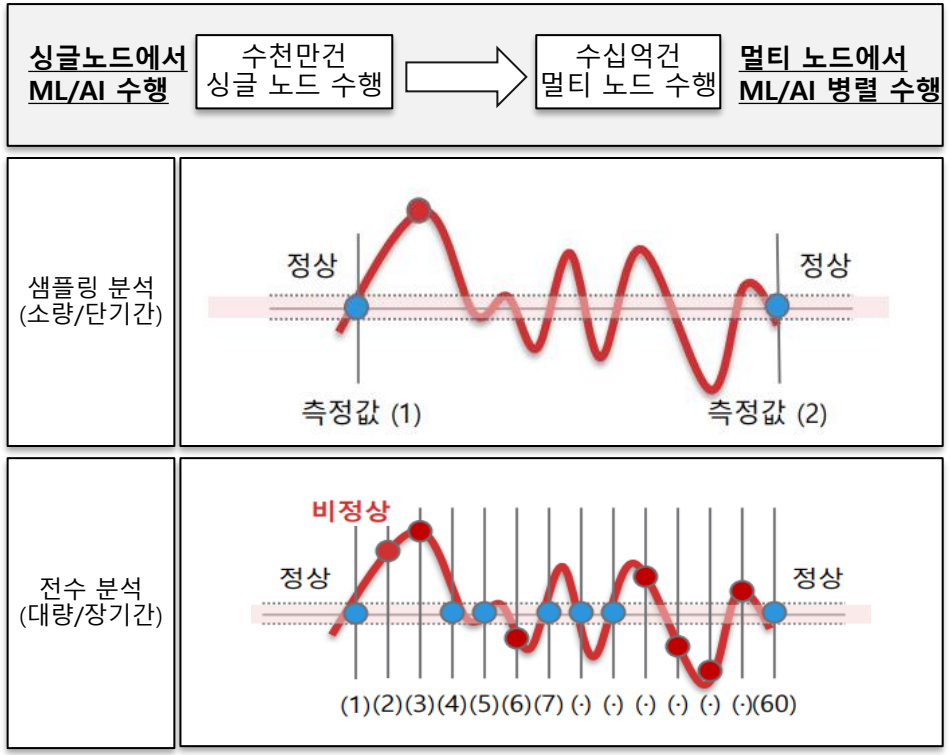
In-DB ML/DL 병렬 분석 방식



- 데이터 이동없이 전체 데이터를 분석의 범위로 확대하여 분석 결과의 신뢰도 향상
- 분석 알고리즘의 고성능 병렬 처리

In-Database ML/DL 병렬 분석 개선 효과

최신 ML/AI 분석 라이브러리를 In-DB에서 병렬 수행
샘플링에서 전수 데이터 처리 지원

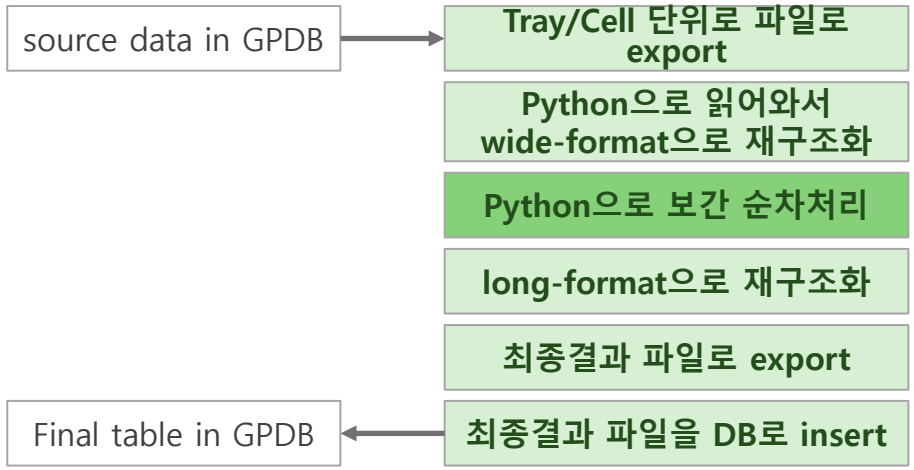


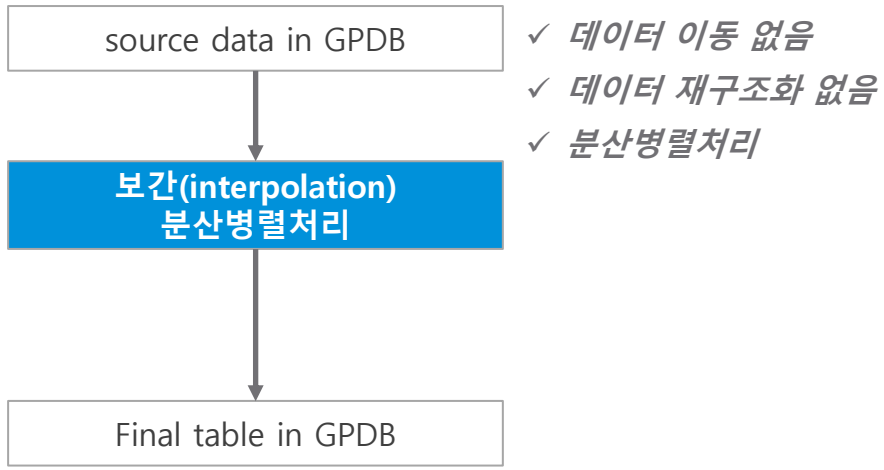
3. In-Database 분석

분석 소요 시간 단축 (20만개 Cell: 1 day → 20 minutes), Workflow 및 관리 Point 간소화

	AS-IS	결과
분석 환경 / 방법	• Single Local Machine, Python • 순차 처리 (sequential processing)	• Greenplum, PL/Container with Python • 대용량 데이터 병렬 처리 (massively parallel processing)
소요 시간	• 약 20 만개 Cell 처리에 **1 Day** 소요	• 20 만개 Cell 처리 시 약 **20분** 소요

Workflow 및 관리 Point





3. In-Database 분석

In-Database 내에서 ML/DL을 위한 pl/Python, pl/R 예시

병렬 ML/AI를 위한 PL/python 예시

Python 코드

```
CREATE OR REPLACE FUNCTION rf_predict_plpy
  (id_arr int[], y_arr float8[], x1_arr float8[], x2_arr float8[])
RETURNS rf_predict_type AS
$$
    import numpy as np
    from sklearn.ensemble import RandomForestRegressor
    id = np.array(id_arr).T
    y = np.array([y_arr]).T
    X = np.array([x1_arr, x2_arr]).T
    rf_regr = RandomForestRegressor(max_depth = 2,
                                    max_features = "auto",
                                    n_estimators = 200,
                                    random_state = 1004)
    rf_regr_model = rf_regr.fit(X, y)
    y_pred = rf_regr_model.predict(X)
    return {'id': id, 's_weight_predicted': y_pred}
$$
LANGUAGE 'plpythonu';

SELECT gender, UNNEST(id) AS id,
       UNNEST(s_weight_predicted) AS s_weight_predicted
FROM (
        SELECT  gender,
           (rf_predict_plpy(id_arr, y_arr, x1_arr, x2_arr)).*
        FROM abalone_array
        GROUP BY gender
     ) a
ORDER BY id;
```

Python 함수 호출

병렬 ML/AI를 위한 PL/R 예시

R 코드

```
CREATE OR REPLACE FUNCTION rf_predict_plr
  id int[], y float8[], x1 float8[], x2 float8[])
RETURNS SETOF rf_predict_type AS
$$
    library(randomForest)

    m1<- randomForest(y ~ x1 + x2)
    temp_m1<- data.frame(id, predict(m1))
    return(temp_m1)
$$
LANGUAGE 'plr';
```

Example

```
SELECT gender, UNNEST(id) AS id,
       UNNEST(s_weight_predicted) AS s_weight_predicted
FROM (
        SELECT  gender,
           (rf_predict_plr(id_arr, y_arr, x1_arr, x2_arr)).*
        FROM abalone_array
        GROUP BY gender
     ) a
ORDER BY id;
```

R 함수 호출

Greenplum에서 각 데이터 노드별로 병렬 처리

3. In-Database 분석

MADLib: SQL Interface의 ML/DL 수행

Train (예측 모델 생성)

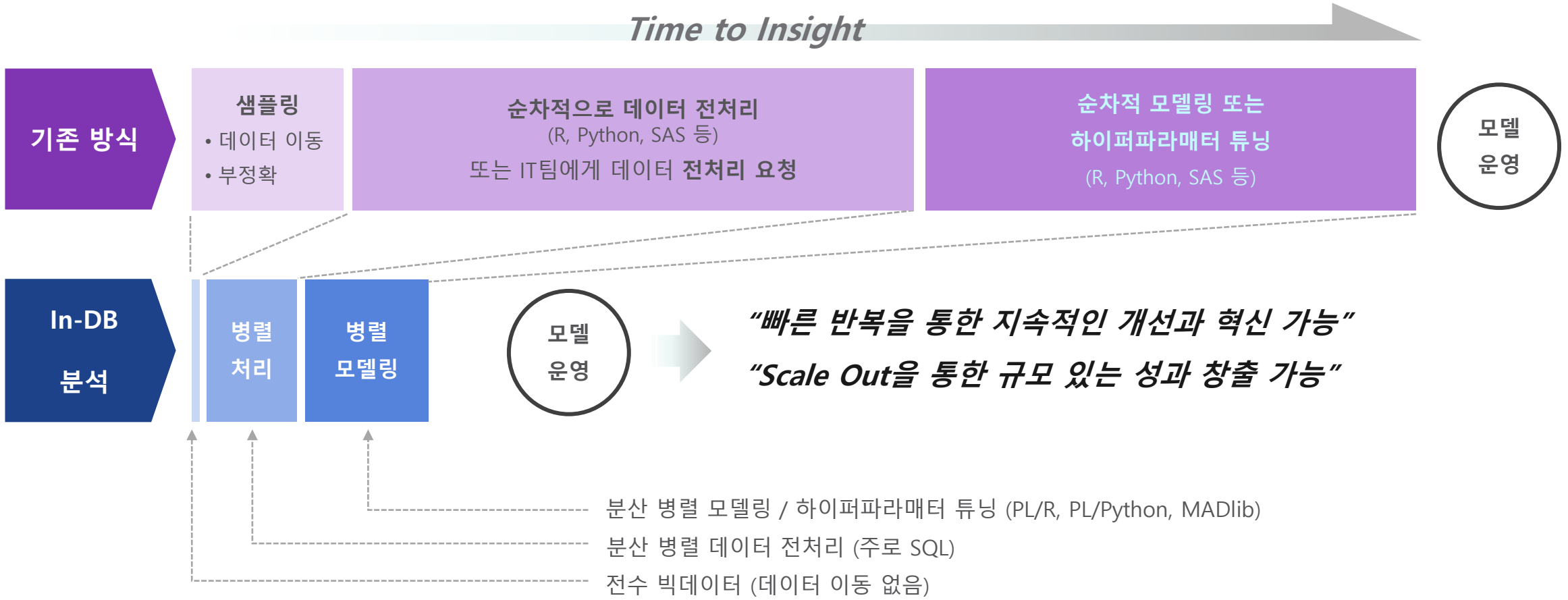
```
SELECT madlib.linregr_train( 'houses',                          -- Historical prices
                             'houses_linregr_bedroom',          -- Output model table
                             'price',                           -- Variable to predict
                             'ARRAY[1, tax, bath, size]',        -- Features
                             'bedroom'                          -- Diff models by #bedrooms
                           );
```

Predict (새로운 데이터로 예측 모델 적용)

```
SELECT houses_test.*,
       madlib.linregr_predict( model.coef,                  -- Trained model
                               ARRAY[1,tax,bath,size]       -- Features
                             ) as predicted_price
FROM houses_test, houses_linregr_bedroom as models
WHERE houses_test.bedroom = model.bedroom;
```

3. In-Database 분석

In-DB 분산 병렬 처리를 통해 기존 대비 분석 및 모델링 시간 대폭 감소



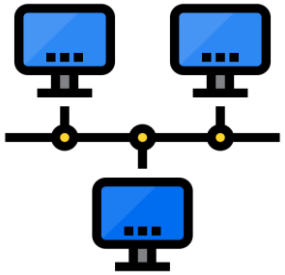
적용 솔루션 소개 및 고객 사례

VMware 분석 데이터베이스

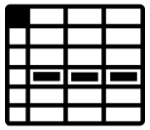
VMware Tanzu Greenplum

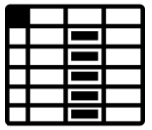
대용량 분석 작업

병렬처리

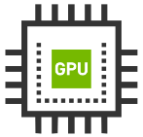


오픈소스 분석플랫폼





Transaction성능이 강화된
검증된 RDBMS



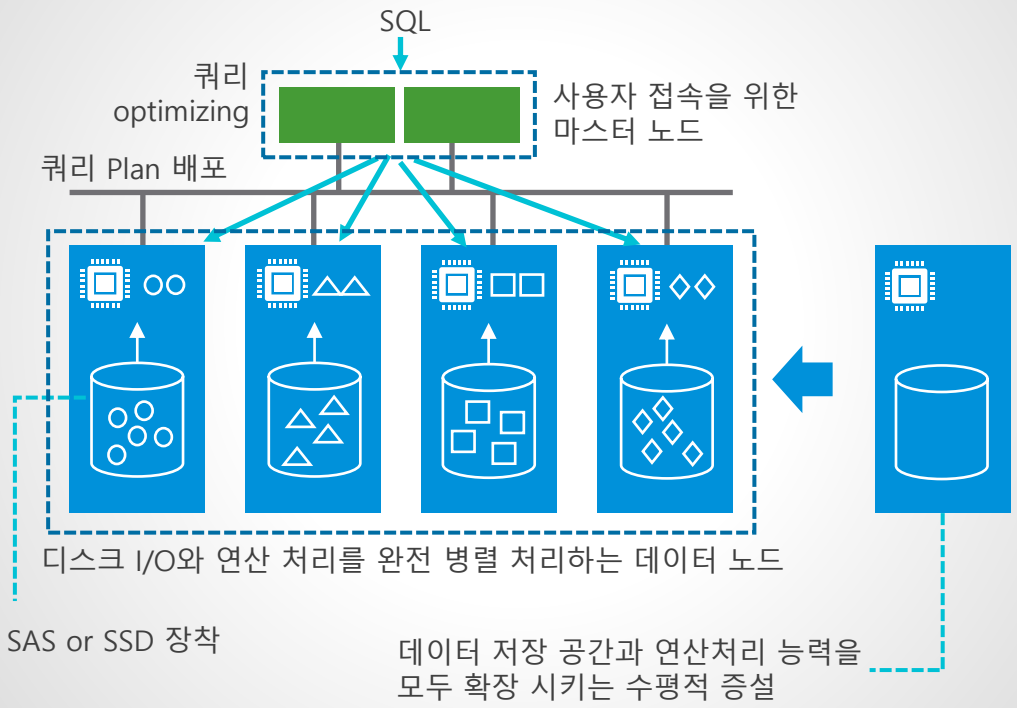
AI, ML을 위한
고급분석 엔진

VMware 분석 데이터베이스

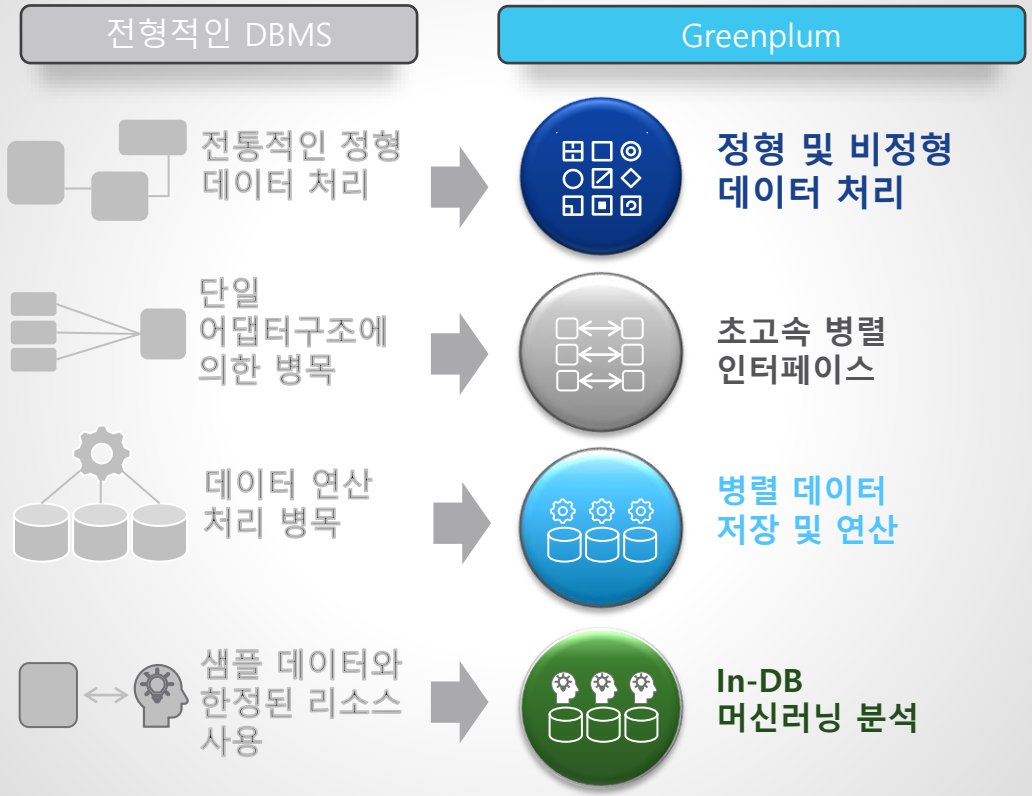
VMware Tanzu Greenplum

대용량 데이터 병렬 처리에 최적화된 아키텍처를 가진 SQL 엔진으로서 정형 및 비정형 데이터에 대한 분석력을 극대화하여 AI/ML을 위한 데이터 분석 플랫폼으로 진화

병렬 처리 아키텍처



특장점



Data Federation

Greenplum Platform Extension Framework(PXF) - Hadoop 및 외부 소스 연동

다양한 외부 시스템의 데이터 소스를 Greenplum으로 연동시 병렬 처리가 지원 되어 고속으로 데이터 전송/수신 지원

Greenplum PXF 연동 방법

- Readable / Writable External Table 생성 가능
- Profile 설정을 통한 Hive, HDFS, S3, Object Storage, JDBC 연결
- Parquet, Avro, Text, CSV, Image 등 다양한 파일 포맷 연결 지원

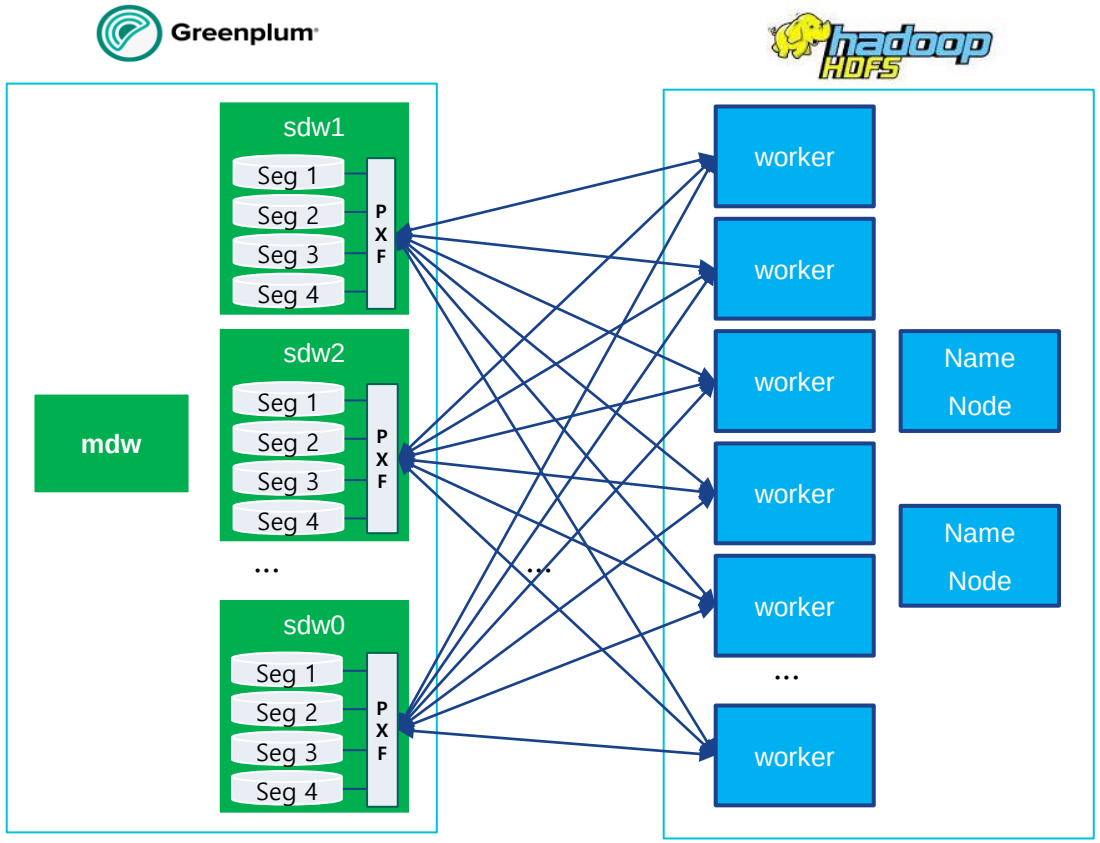
External Table 사용 예

```
CREATE EXTERNAL TABLE ext_hive_sample (
    -- column definitions)
LOCATION('pxf://gp_poc.CELL_RSDN?PROFILE=hive&Server=hadoop')
FORMAT 'CUSTOM' (formatter='pxfwritable_import');
```

```
CREATE EXTERNAL TABLE ext_hdfs_read_sample (
    -- column definitions )
LOCATION('pxf://user/gp/hdfs_cell_rsdn/?PROFILE=hdfs&server=hadoop')
FORMAT 'TEXT' (delimiter=',');
```

```
CREATE WRITABLE EXTERNAL TABLE ext_hdfs_write_sample (
    -- column definitions )
LOCATION('pxf://user/gp/hdfs_cell_rsdn/?PROFILE=hdfs&server=hadoop')
FORMAT 'TEXT' (delimiter=',');
```

Greenplum – Hadoop 데이터 연동 아키텍쳐



Data Federation

하둡 시스템으로부터 병렬 고속 데이터 전송/수신

Greenplum의 Platform Extension Framework(PXF)를 이용하여 하둡 시스템과의 병렬데이터 전송/수신 지원
Greenplum의 노드수가 증가하면 속도가 더 향상

Hadoop에서 Greenplum 적재 시간

데이터 소스	건수	Size (MB)	적재 시간	비고
Hive ORC	38,527,271	1,498	17 sec	2.2 백만건/sec
HDFS text	38,527,271	10,068	11 sec	3.5 백만건/sec

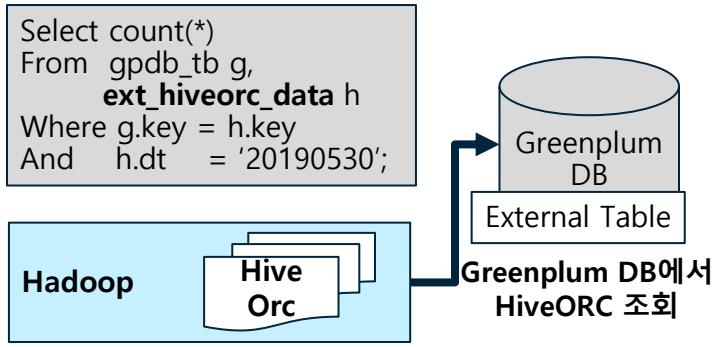
Hadoop HiveORC 데이터와 Greenplum Internal Table 조인 소요시간

데이터 소스 위치	테이블명	건수	Size	결과	비고
Hive ORC	hiveorc_data	47 억건	260 GB	소요 시간: 3 sec 결과 건수: 60 만건	Hive 3년 데이터 중 1일 필터링
Greenplum	gpdb_tb	4,000	1 MB		

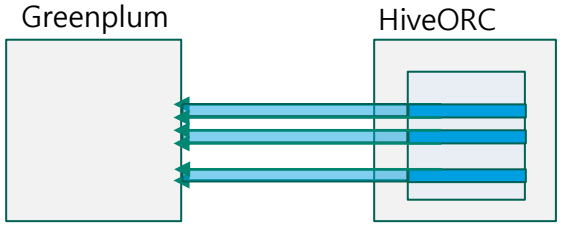
성능 테스트 환경

- **Greenplum 시스템**
  - Master Node: 1 EA, Data (Segment) Node: 4 EA
- **Hadoop 시스템**
  - Name Node: 2 EA, Data (Worker) Node: 8 EA
- **Greenplum – Hadoop Connector**
  - Greenplum Platform Extension Framework (PXF)
  - Greenplum의 확장 패키지, 병렬 Interface 제공

External Table & Internal Table 조인



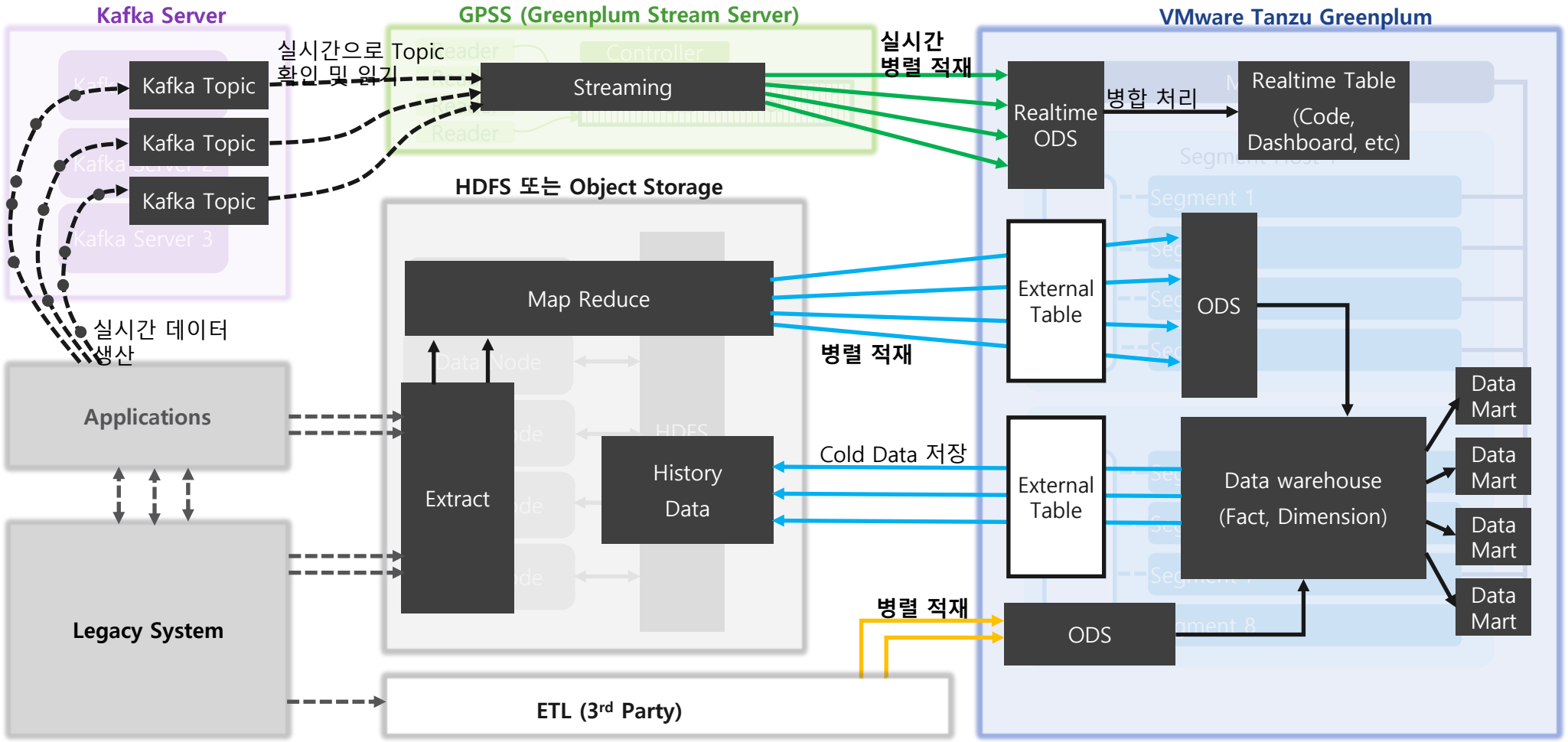
PXF Filter Pushdown



- PXF에서는 filter pushdown 기능 제공
- 외부 데이터소스의 데이터 전송 최소화
- 연동시 소스 시스템의 부하 감소

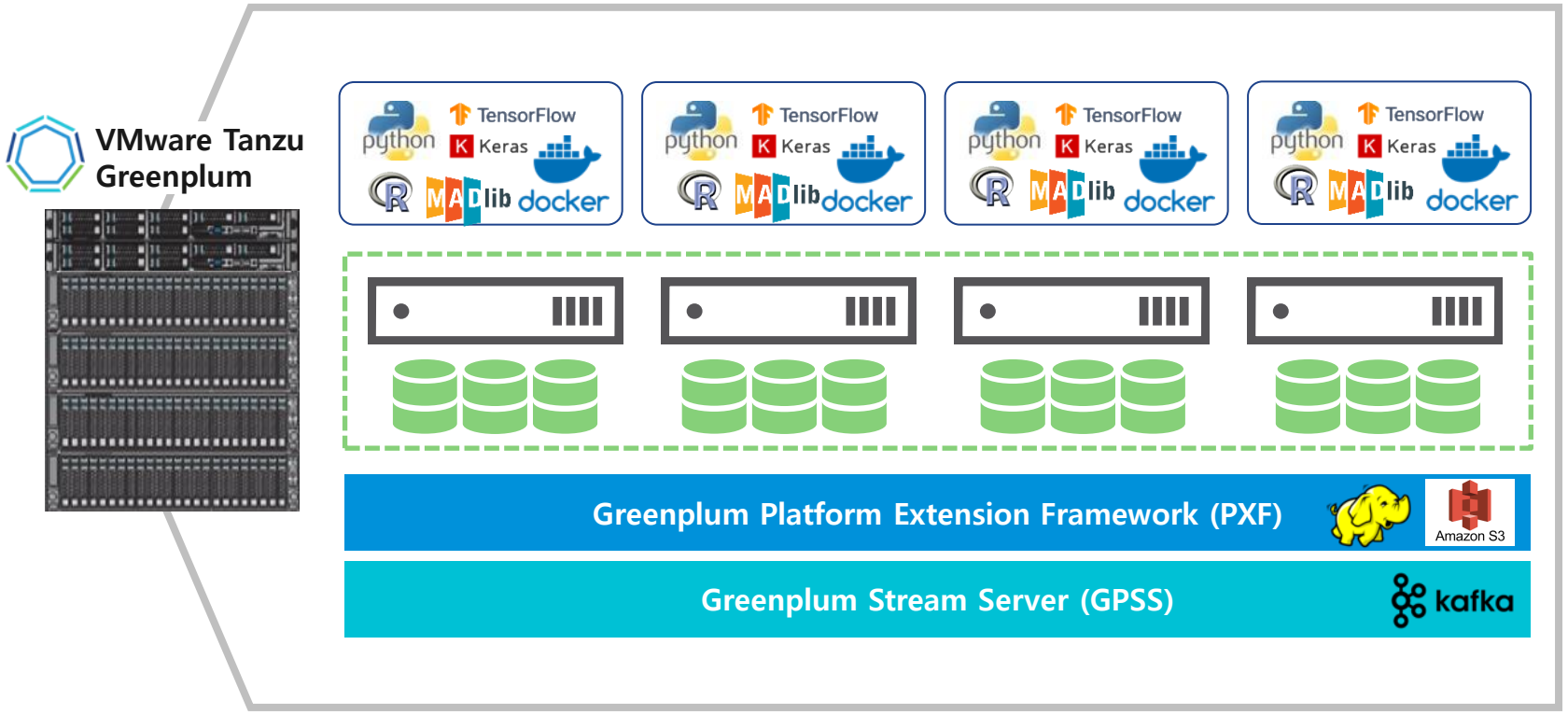
Data Federation

하둡/오브젝트 스토리지 등 모든 인터페이스 연동 시 병렬 데이터 처리 지원



머신러닝, 딥러닝을 위한 In-DB 분석

데이터 통합 연계, 머신러닝, 분산 병렬 처리



머신러닝 학습 및 스코어링을 위한 도커기반 데이터 사이언스 프로그래밍 환경

병렬 분산 데이터 저장, 연산, 조회

데이터 레이크와 고속 병렬 연동

Kafka 서버로부터 실시간 데이터 적재

머신러닝, 딥러닝을 위한 In-DB 분석

SQL ML/DL을 위한 MADlib 분석 알고리즘

- SQL 기반의 병렬처리 In-DB 머신러닝, 딥러닝 분석을 위한 오픈 소스 라이브러리
- Greenplum의 병렬 아키텍처에 최적화되어 전체 컴퓨팅 성능을 활용
- SQL 인터페이스에서 호출하므로 사용 방법이 간편
- Apache의 최상위 레벨의 프로젝트

SUPERVISED LEARNING
Neural Networks
Support Vector Machines (SVM)
Conditional Random Field (CRF)
Regression Models
- Clustered Variance
- Cox-Proportional Hazards Regression
- Elastic Net Regularization
- Generalized Linear Models
- Linear Regression
- Logistic Regression
- Marginal Effects
- Multinomial Regression
- Nai¨ve Bayes
- Ordinal Regression
- Robust Variance

Tree Methods
- Decision Tree
- Random Forest

UNSUPERVISED LEARNING
Association Rules (Apriori) Clustering (k-Means)
Principal Component Analysis (PCA) Topic Modelling (Latent Dirichlet Allocation)

DEEP LEARNING
Keras Fit/Evaluate/Predict Load Model Architectures Preprocessor for Images Parallel Image Loading

GRAPH
- All Pairs Shortest Path (APSP)
- Breadth-First Search
- Hyperlink-Induced Topic Search (HITS)
- Average Path Length
- Closeness Centrality
- Graph Diameter
- In-Out Degree
- PageRank
- Single Source Shortest Path (SSSP)
- Weakly Connected Component

DATA TYPES AND TRANSFORMATIONS
Array and Matrix Operations Matrix Factorization
- Low Rank
- Singular Value Decomposition (SVD)

Norms and Distance Functions
Sparse Vectors
Encoding Categorical Variables Path Functions
Pivot
Sessionize
Stemming

STATISTICS
Descriptive
- Cardinality Estimators
- Correlation and Covariance
- Summary

Inferential Statistics
- Hypothesis Tests

Probability Functions

TIME SERIES
- ARIMA

MODEL SELECTION
- Cross Validation
- Prediction Metrics
- Train-Test Split

NEAREST NEIGHBORS
- Nearest Neighbors

UTILITY FUNCTIONS
Columns to Vector Conjugate Gradient Linear Solvers
- Dense Linear Systems
- Sparse Linear Systems Mini-Batching

PMML Export
Term Frequency for Text Vector to Columns

분석 사례: synchrony FINANCIAL

장기간의 대용량 카드 거래 데이터 분석을 통한 1st Party (Synthetic ID) 사기 거래 방지

Fraud 유형

1st Party (Synthetic ID) Fraud

- **범죄자와 피해자가 동일인**
- 일정 기간 양호한 거래 유지
- 신용 한도를 지속적으로 올린 후 부도(no payment)
- 피해자는 신고할 동기가 없으며, 카드사에서 의심 행동 확인 전화 시 문제가 없다고 거짓 증언함

- **예측(Prediction)**
- **계좌 전체 거래 내역**이 사기가 발생할 조짐을 나타내고 있나?
- **배치/오프라인**

3rd Party Fraud

- 일반적으로 알고 있는 사기
- **범죄자와 피해자가 다름**
- 3 Parties: 고객, 기관, 도둑
- 예: 도난 카드, 정보 탈취
- 피해자는 카드 도난 시 신고할 동기 있음

- **탐지(Detection)**
- 이번 **개별 거래 건**이 정상인가 아닌가?
- **실시간**

신용카드 1st Party (Synthetic ID) Fraud 예측 모형 개발

활용 Data

- **3년 치 데이터**
- **1,700개 이상 변수, 100억 행(row) 이상**
- 거래 데이터
  : 고위험 업종 코드 (보석, 상품권, 카지노 등), Balance Transfer 등
- 거래 외 데이터
  : 지급 금액/회수, 계좌 개수, 신용 등급, IVR, 웹 로그 데이터 등

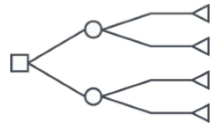
"기존 RDBMS, 통계 툴로는 처리/분석 불가"

분석 플랫폼/툴

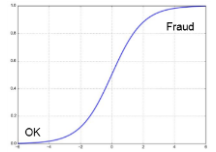
- **GPDB를 활용한 MPP** Greenplum (Massively Parallel Processing)
- **MADlib , PL/**을 활용한 대용량 데이터 분산 기계학습 MADlib

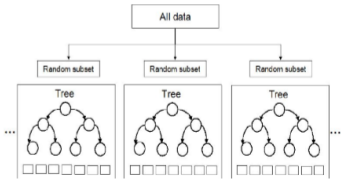
1st Party Fraud 예측 알고리즘

의사결정 나무 (설명력 높음)



로지스틱 회귀 (이진 분류, 확률)





랜덤 포레스트 (Tree 앙상블, 다수결 투표를 통한 예측)

예측 모형 적용 성과

- **$4MM Annual savings**
- **기존 대비 모델 정확도 9배 향상**
- **False Negative * 감소**

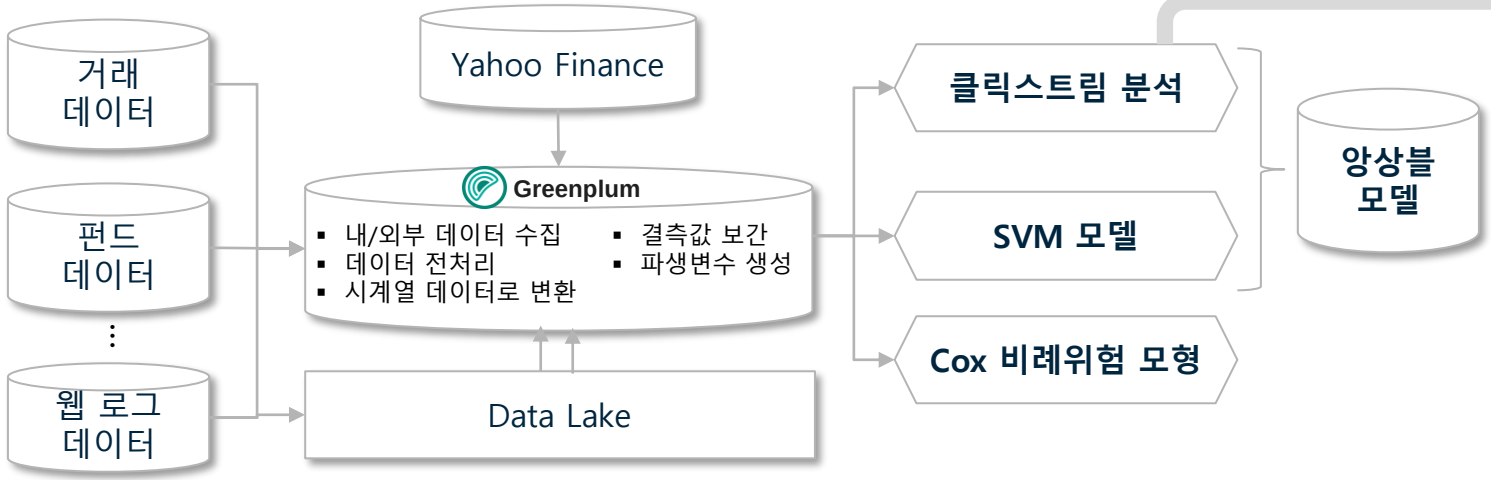
* False Negative: 정당한 거래에 대한 잘못된 승인 거절

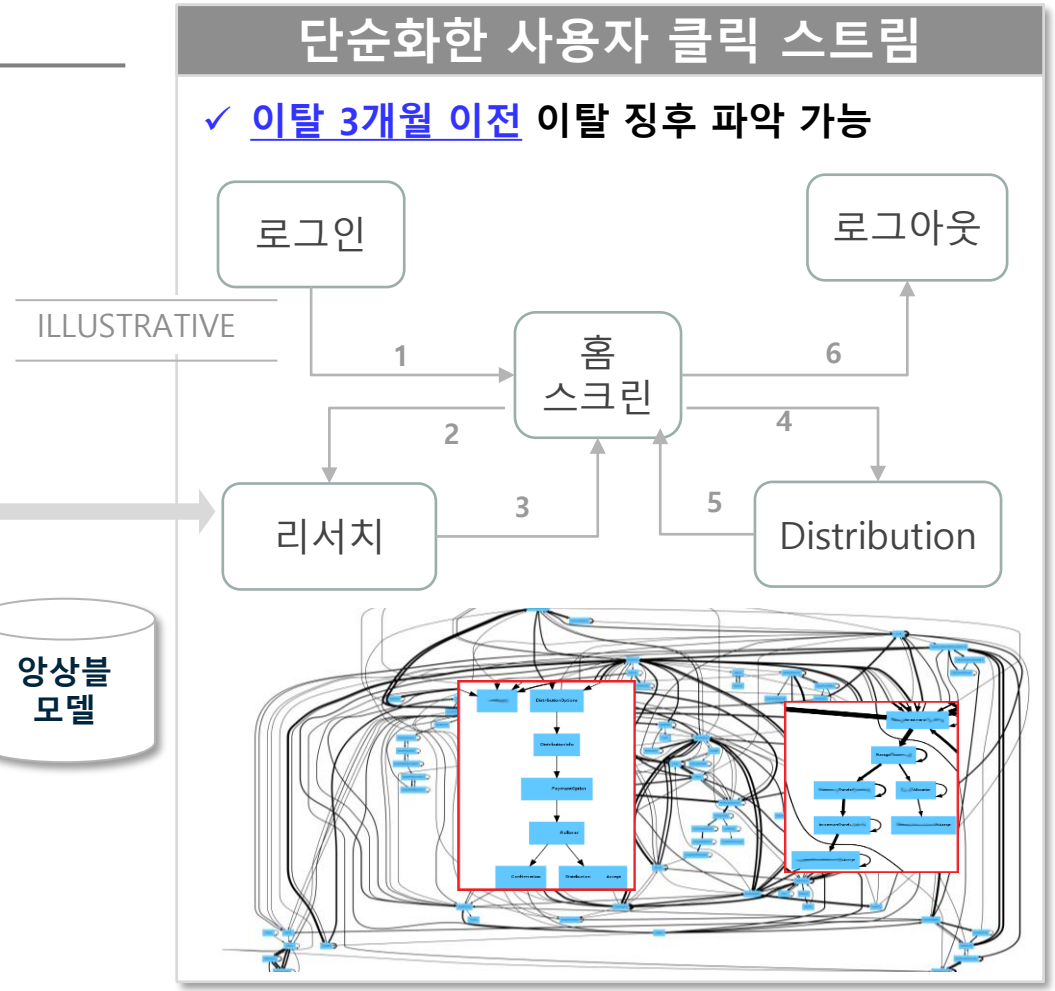
분석 사례: JPMorganChase

거래/펀드/웹로그 및 외부 데이터를 통합하여 기계학습 모델링을 통해 401K 이탈 예측, 방지

401K 이탈 방지 (Churn Management)

- **401K 퇴직연금 가입자의 경쟁사로의 이탈 방지를 위해 이탈 요인 규명**
  - 상품과 서비스 품질 만족도 저하
  - 경쟁사 대비 상품 제안과 가격 경쟁력 미흡
  - 실업으로 인한 401K 해지
- **잠재 자산 규모**와 **이탈 가능성(예측 모형)**을 기준으로 401K 가입자의 **관리 우선순위(Ranking)** 부여

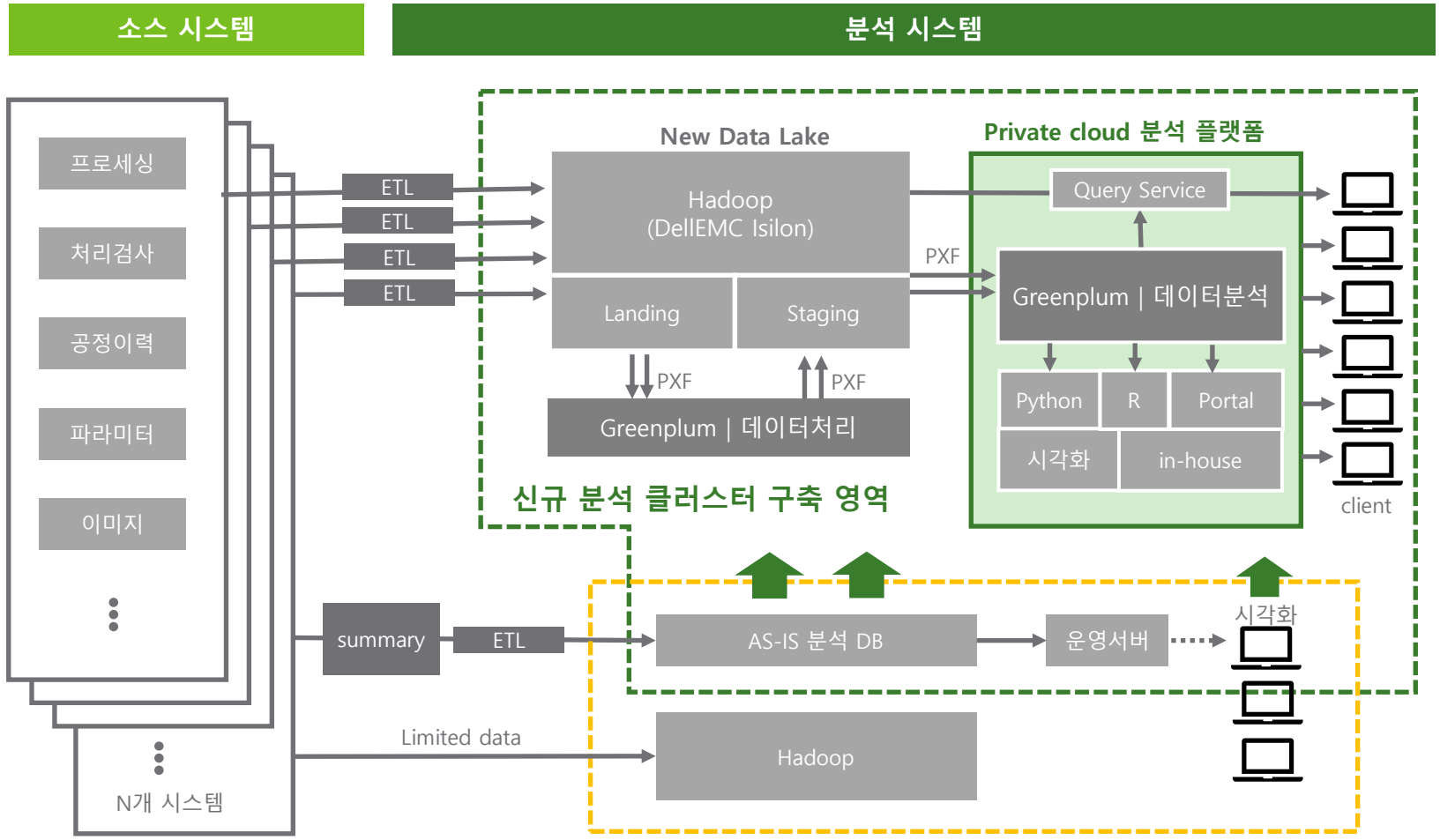




Private Cloud
분석 레퍼런스 아키텍처

Private Cloud 분석 플랫폼

개별적인 분석 시스템의 성능 개선 및 선진화된 클라우드 향 데이터 통합 분석 플랫폼 구축



도전과제 및 개선사항

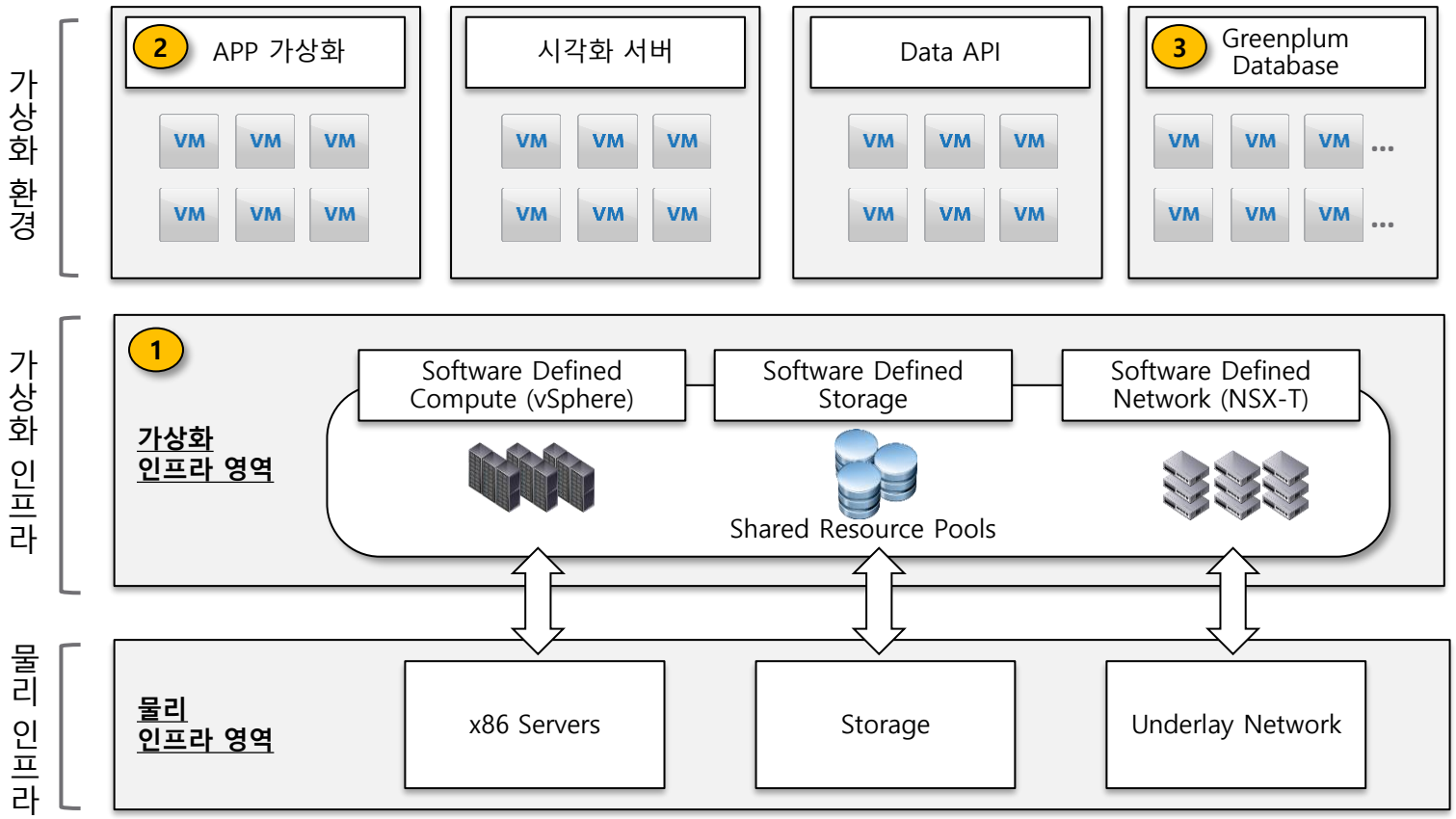
Challenges
- AS-IS 분석시스템의 성능/ 용량 한계
- 유연하지 않는 시스템 아키텍처 한계
- 전통적인 개발 방식의 한계 개선

Solution
- 고성능/대용량 병렬처리 DW 시스템 적용
- 클라우드 기반의 빅데이터 분석 플랫폼 구축
- Database를 분석 플랫폼으로 적용

Improvements
- 현업 요구 분석 속도 충족
- 사전 오류 방지로 생산 비용 절감
- 분석 생산성 향상 및 보안 강화
- 클라우드 통합 데이터 분석 플랫폼 구현으로 다양한 분석 환경 요구 충족

Private Cloud 분석 플랫폼

Private Cloud 분석 클러스터 개념도



영역별 개요

1 가상화 인프라 영역
- Software Defined 환경 구축
- ESXi/NSX-T 기반 인프라 설치
- NSX-T 기반의 Overlay network 환경 구현
- 패치 관리 자동화 도구 연동

2 APP 가상화
- Application이 VM에서 구동
- 시각화/Jupiter 노트북, Rstudio 등
- Greenplum database 와의 연동이 로컬 PC가 아닌 서버간 통신
- 네트워크 전송 개선 및 성능 향상
- 로컬 PC에서 수행되지 않기 때문에 데이터 보안 처리

3 Greenplum Database
- 데이터 저장/분석병렬 처리
- 데이터 전수신 병렬 처리
- ML/AI 병렬 처리
- 프로그램 랭귀지 지원 Python/Tensorflow, R, Java, C, C++ 등

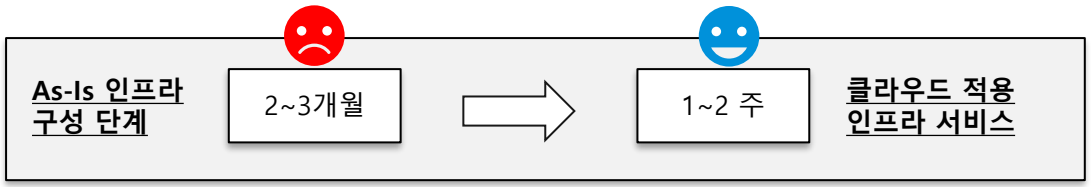
Big Data on Cloud 구축

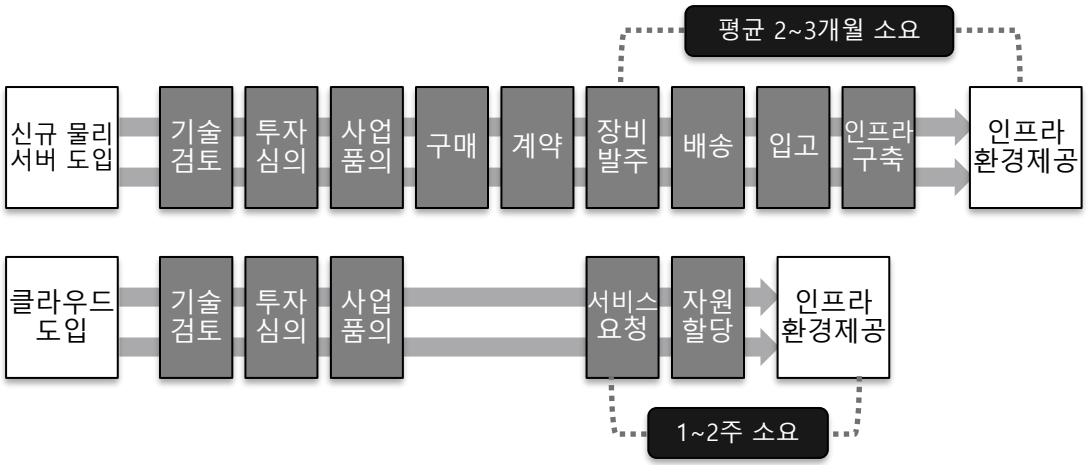
Analytics system on Private Cloud

신속한 분석 환경 구축, RDSH를 통한 앱 가상화로 데이터 추출 속도 개선 및 데이터 보안 강화

인프라 환경 구성

클라우드 환경에서 인프라 자원요청 부터 자원할당까지 수개월에서 1~2주 이내로 단축되어 비즈니스 민첩성이 향상 됨.

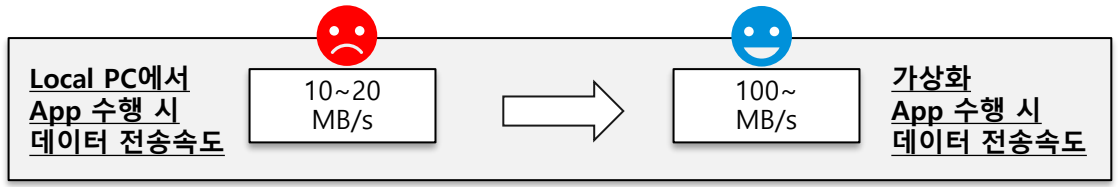


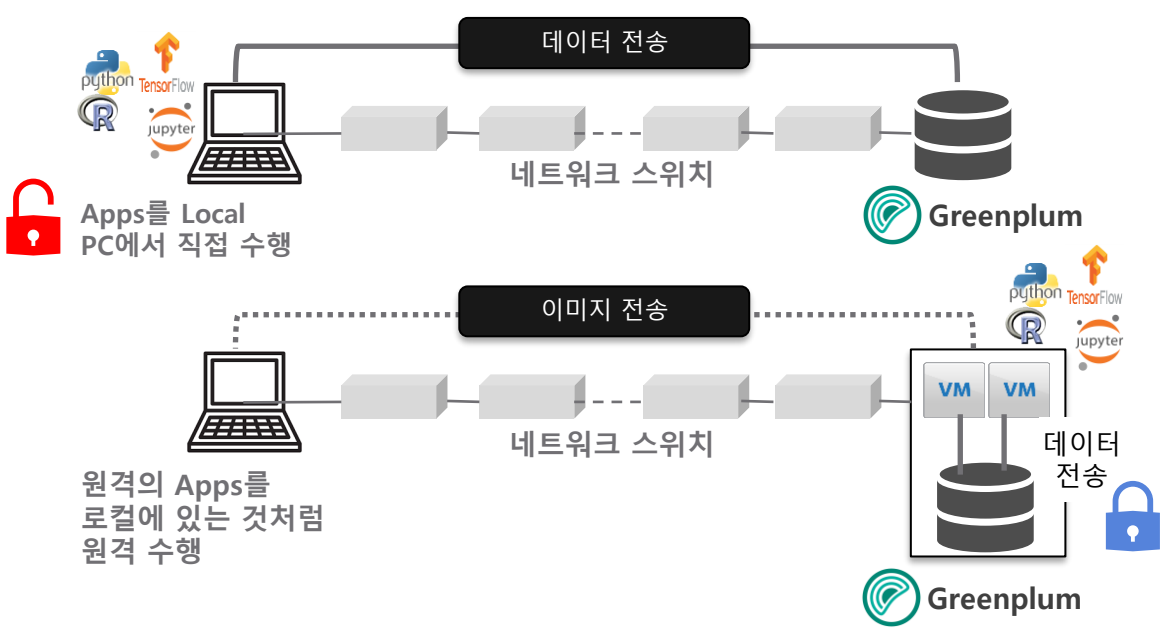


※ **VMware vRealize Automation 도입 시 10분~1시간 내외**로 인프라 환경제공

성능 및 보안

App 가상화로 대용량 데이터 전송 속도가 5~10배 개선 되어, 생산성 향상
데이터가 Private Cloud에 존재하기 때문에 데이터 보안 안정성 향상





빅데이터 분석 플랫폼 고려사항

빅데이터 분석 플랫폼 고려사항

고성능을 위하여 IO, 연산 분산 병렬 처리

ETL 없이 외부 멀티 소스 데이터를 빠른 병렬 연계

초대용량 데이터에 최상 성능을 위한 데이터 직렬화 모델링

머신러닝/딥러닝을 데이터 이동 없는 고속 병렬 처리

Bare Metal, Private cloud, Public Cloud 등의 구성이 유연한 클라우드형 아키텍처

Thank You