

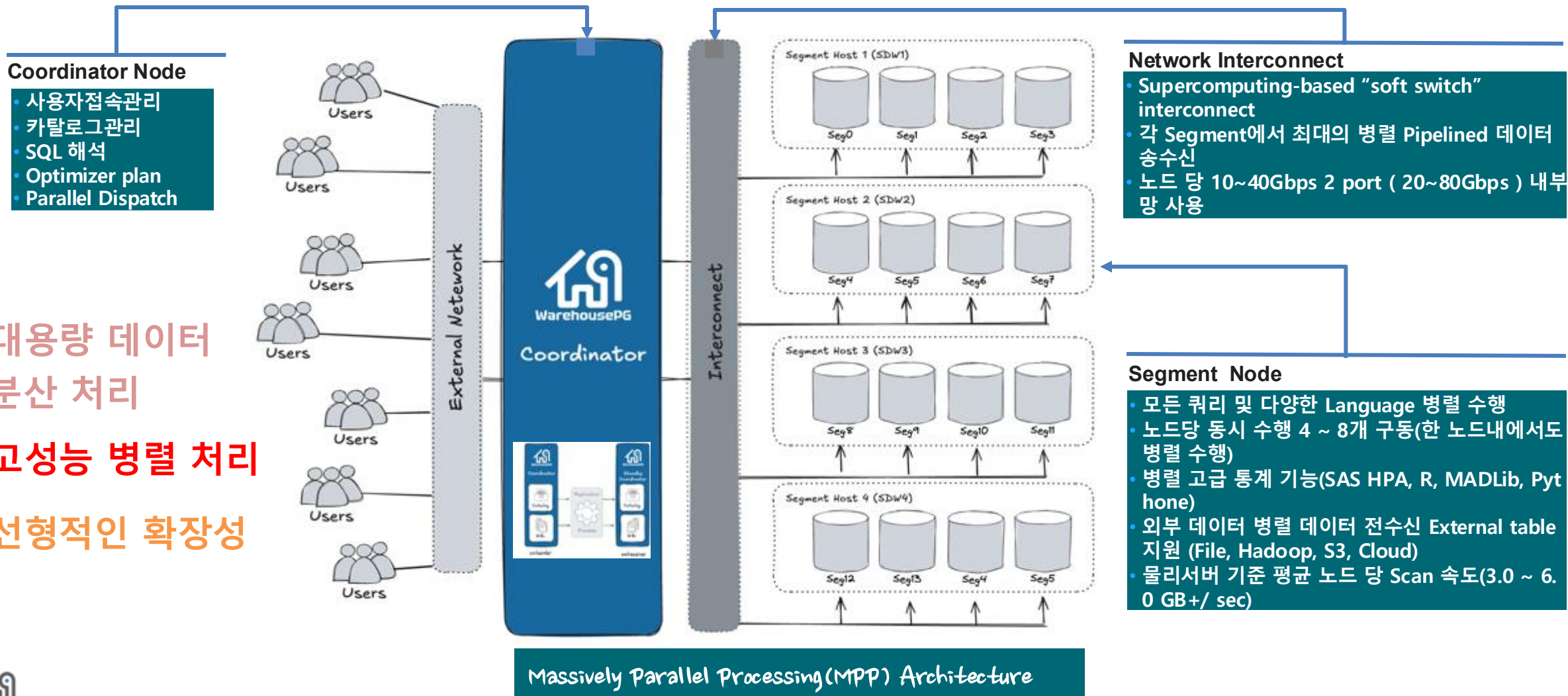
WarehousePG Architecture

Aug 2025

EDB Senior Sales Engineer / 손 광 락

Analytic Accelerator > Support Greenplum Workload > WarehousePG

- 병렬처리 (MPP, Massive Parallel Processing) 구조로 초대용량의 데이터를 더욱 빠르게 보관 및 연산 처리할 뿐만 아니라 더욱 많은 고급 분석을 병렬로 처리합니다



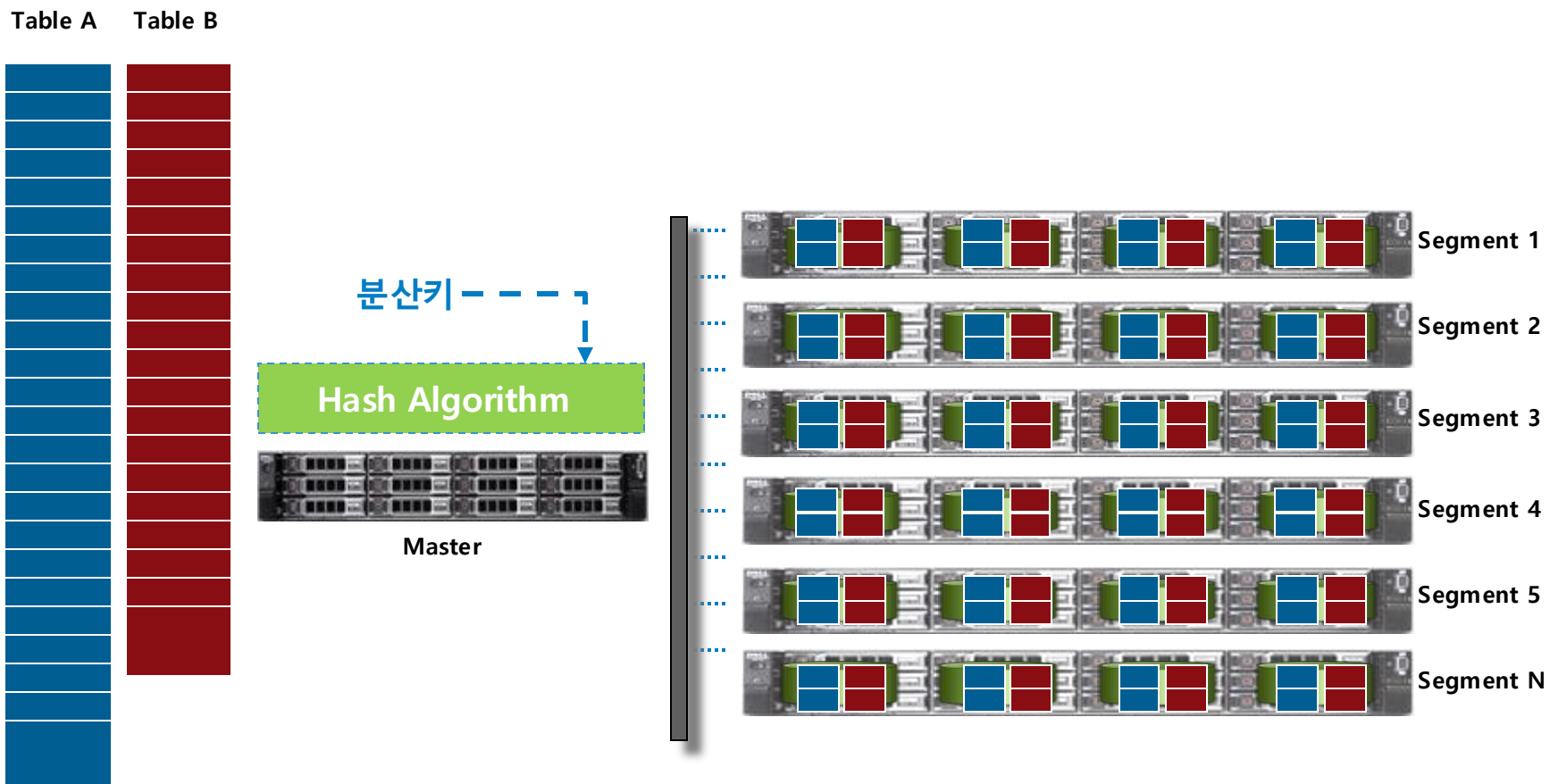
대용량 데이터 처리 필수 기술이란 ?

```
CREATE TABLE financial_news (  
  id bigserial,  
  date date,  
  topic text,  
  headline text,  
  content text,  
  writer text,  
  like_num integer,  
  read_num integer,  
  embedding vector(1536),  
  location geography(Point, 4326)  
)  
DISTRIBUTED BY (id)  
PARTITION BY RANGE (date) (  
  -- Monthly partitions from 2023-01-01 to 2025-01-01  
  PARTITION p_monthly_2023_01_2025_01  
  START ('2023-01-01') END ('2026-03-01') EVERY ('1 month'::INTERVAL)  
  WITH (appendonly=true, orientation=column, compressstype=zstd),  
  -- Yearly partitions from 2015-01-01 to 2023-01-01  
  PARTITION p_yearly_2015_01_2023_01  
  START ('2015-01-01') END ('2023-01-01') EVERY ('1 year'::INTERVAL)  
  WITH (appendonly=true, orientation=column, compressstype=zlib));
```

- ① 병렬 처리 → 분산 저장
- ② 처리할 데이터만 → 파티션, 컬럼나
- ③ 물리 데이터 최소화 → 압축
- ④ 벡터화 → 분석 기반 블럭화
 - 데이터 분산저장(Sharding) 및 병렬처리
 - 파티션(Partitioning)
 - 컬럼 저장 테이블(Columnar oriented table)
 - 압축 (zlib, zstd, rle_type)
 - Polymorphic Storage (파티션 레벨에 따른 다른 압축 방식 사용)

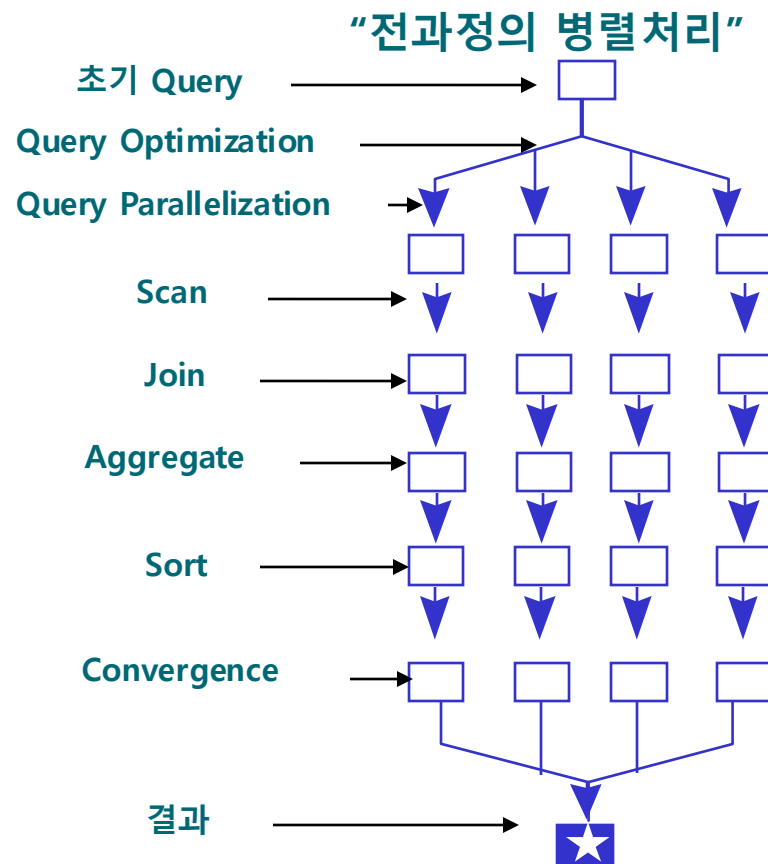
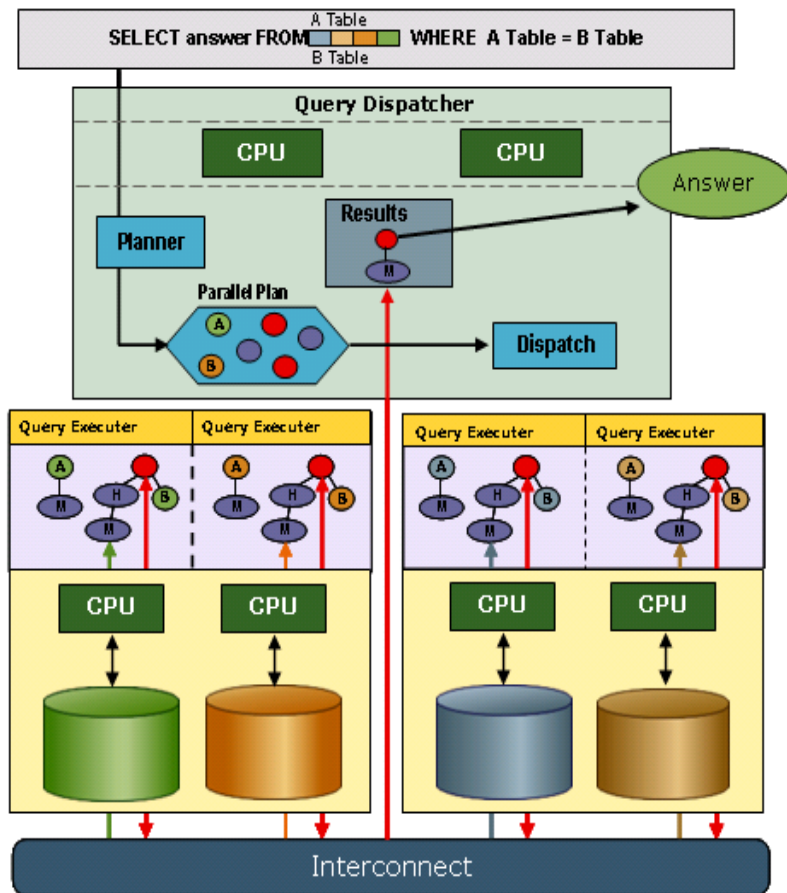
분산키 기반 데이터 분산(샤딩) 저장

- Segment에 있는 모든 Disk에 데이터를 고르게 분산하기 위한 Hash Algorithm을 사용 함.
- Row들은 Hash Algorithm에 의하여 전 Segment에 있는 Disk에 고르게 분산되어 쿼리에 대한 병렬 처리를 최적화 할 수 있는 기반 제공.
- Greenplum의 분산정책은 Hash Distribution과 Random Distribution, Replicated Distribution 3가지 방식이 있음.



병렬 쿼리 실행

- 쿼리 수행 시 전 과정에 걸친 병렬처리로 대용량 데이터 조회 성능이 뛰어남.

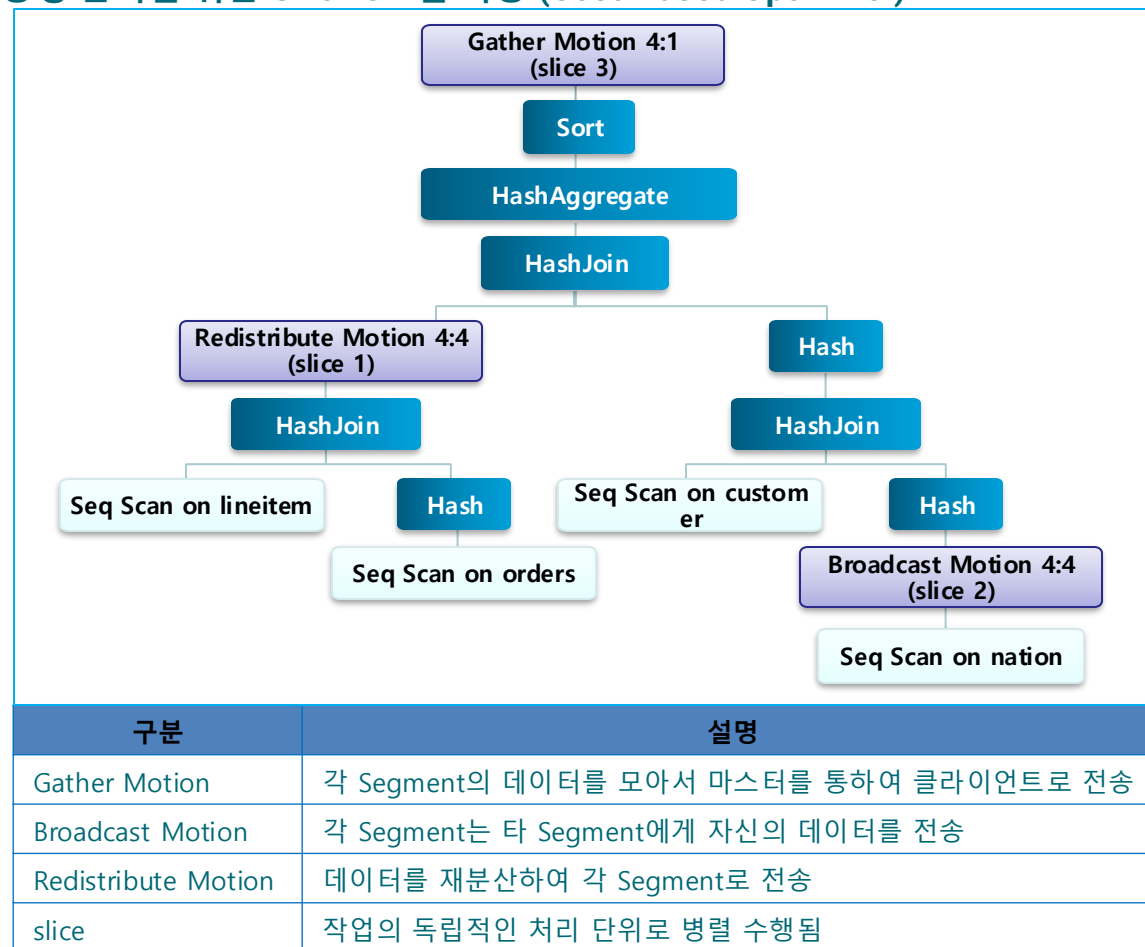


병렬 쿼리 실행 (계속)

- Master는 테이블 스캔, 조인, 집계 또는 정렬 등과 같은 연산의 집합 단위로 병렬 수행되고, 필요 시 각 Segment 사이에 데이터가 이동되어 처리되도록 쿼리실행계획을 수립함
- 트랜잭션 워크로드를 위한 Postgres-Based Planner 와 대용량 분석을 위한 GPORCA 를 사용 (Cost-Based Optimizer)

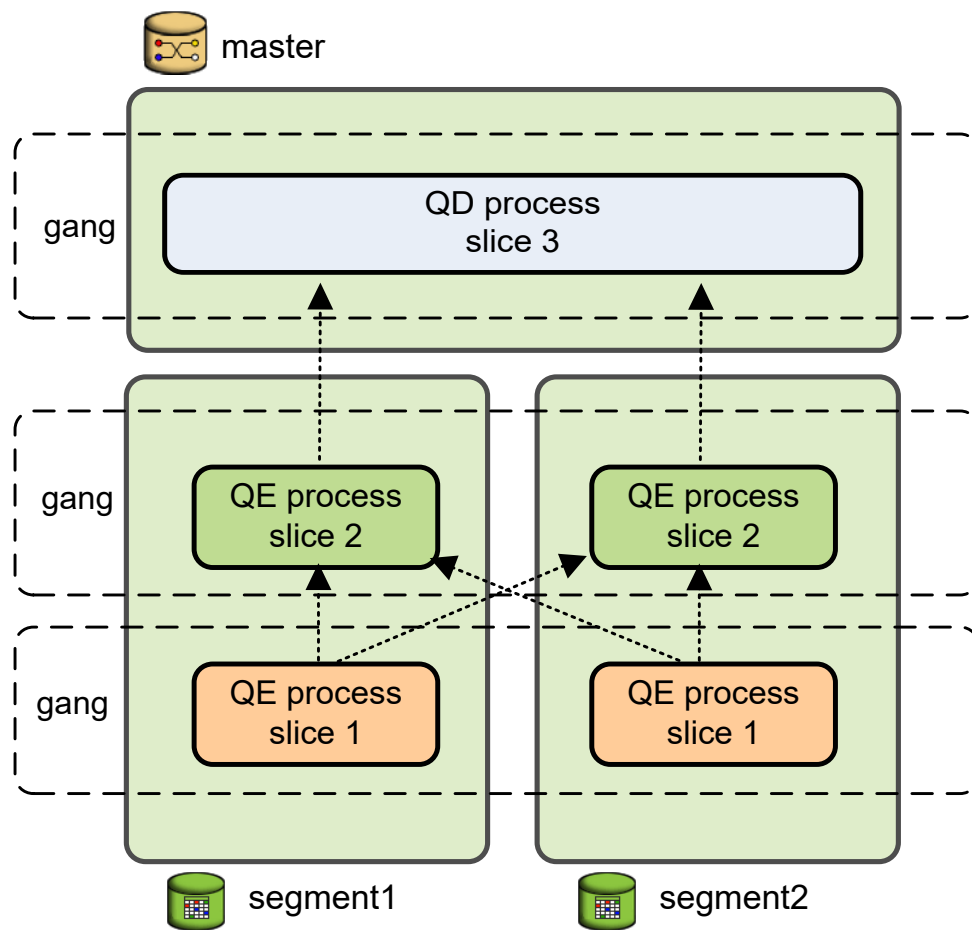
explain

```
select
  c_custkey, c_name,
  sum(l_extendedprice * (1 - l_discount)) as revenue,
  c_acctbal, n_name, c_address, c_phone
from
  customer, orders, lineitem, nation
where
  c_custkey = o_custkey
  and l_orderkey = o_orderkey
  and o_orderdate >= date '1994-08-01'
  and o_orderdate < date '1994-08-01'
    + interval '3 month'
  and l_returnflag = 'R'
  and c_nationkey = n_nationkey
group by
  c_custkey, c_name, c_acctbal,
  c_phone, n_name, c_address, c_comment
order by
  revenue desc
```



병렬 쿼리 실행 (계속)

- 쿼리 작업 시 Query Dispatcher가 실행계획을 생성하여 segment에 전달하고, 각 Segment node에서 Query Executor에 의해 작업을 실행한 후 결과를 Query Dispatcher가 받아 최종적으로 클라이언트에 전달함.



구분	설명
QD	<ul style="list-style-type: none"> Query dispatcher Master에서 수행되는 프로세스 쿼리실행계획 생성하여 각 Segment에 전달함 처리된 최종 결과를 모아서 클라이언트에 전송함
QE	<ul style="list-style-type: none"> Query executor 각 Segment에서 수행되는 프로세스 쿼리실행계획의 slice 단위로 할당된 작업을 수행하고, 다른 프로세스와 결과를 공유하여 협업함
gang	<ul style="list-style-type: none"> 쿼리실행계획에서 동일한 부분의 작업에 관련된 프로세스들의 집합 gang 프로세스는 부분 작업이 완료되면 다음 gang 프로세스로 데이터가 이동됨 프로세스간의 통신은 interconnect를 통하여 수행됨

Parallel Data Federation : 이기종 고속 SQL Interface

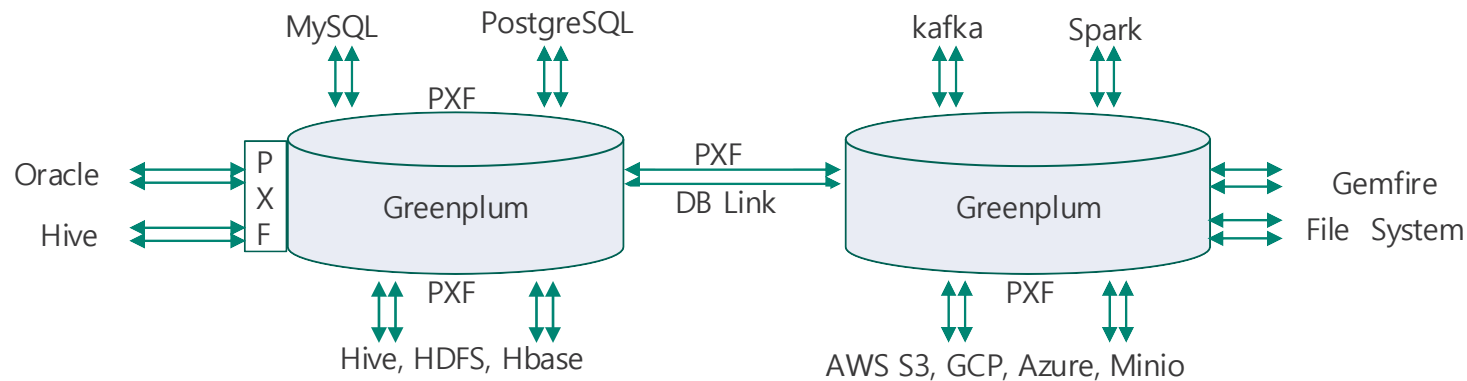
- 다양한 외부 시스템의 데이터 소스를 Greenplum에서 one SQL으로 연계 시스템 분석 기능 제공
- Delta / Iceberg 지원 예정

□ Greenplum Data Federation

- 데이터는 원천 시스템에 저장, 연산은 Greenplum에서 수행
- 이기종 Data (database, hadoop, IMDG 등)를 ETL과 같이 데이터 이관 없이 Greenplum에서 연산 수행
- Greenplum에서 one SQL으로 연계 시스템 분석 기능 제공

□ Greenplum PXF

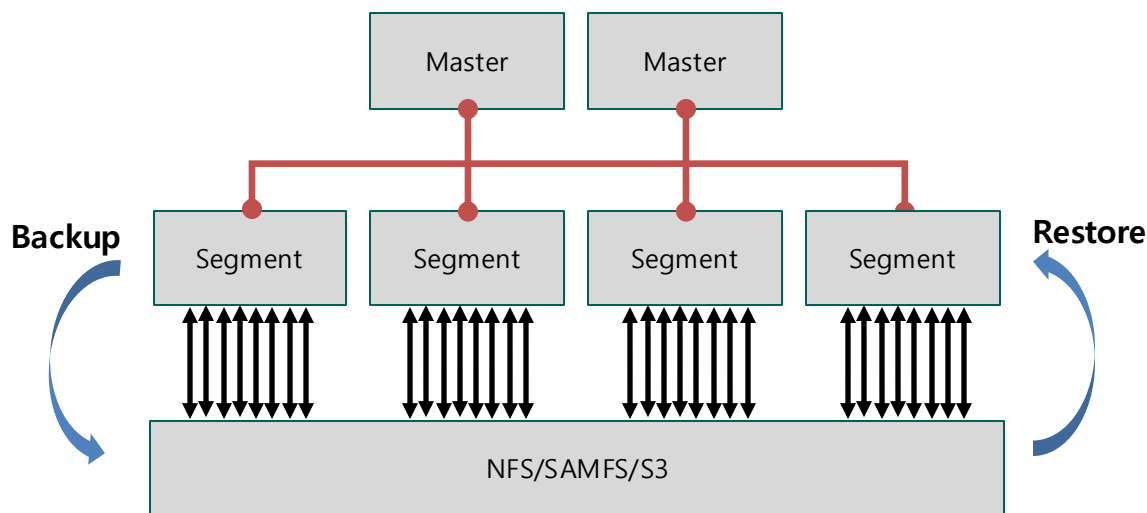
- 연동 편의성 제공: External Table의 기능으로 쉬운 Interface 제공
- 최상의 성능 제공: 고속의 병렬 Interface 기능 제공
- Avro, JSON, ORC, Parquet, CSV 등 지원



Parallel Backup/Restore

- Greenplum 백업/복구 프로그램이 자체적으로 데이터 정합성을 보장하며 병렬 백업 및 복구를 수행하므로 작업시간이 단축 됩니다. Object storage 및 NAS Storage에 백업 데이터를 보관하여 클러스터 서버 장애 시에도 안정적으로 복구할 수 있습니다.

Greenplum 백업/복구



- 백업 수행 시 병렬 수행
 - 서버 별로 병렬 백업
 - 각 서버에서도 병렬 프로세스 수행

1 백업 / 복구 메커니즘

- 마스터 노드에서 백업/복구 커맨드 실행
- 모든 세그먼트 노드에서 병렬로 백업/복구 프로세스 실행
- 모든 세그먼트 노드에서 병렬로 덤프 생성

2 백업 / 복구 기능

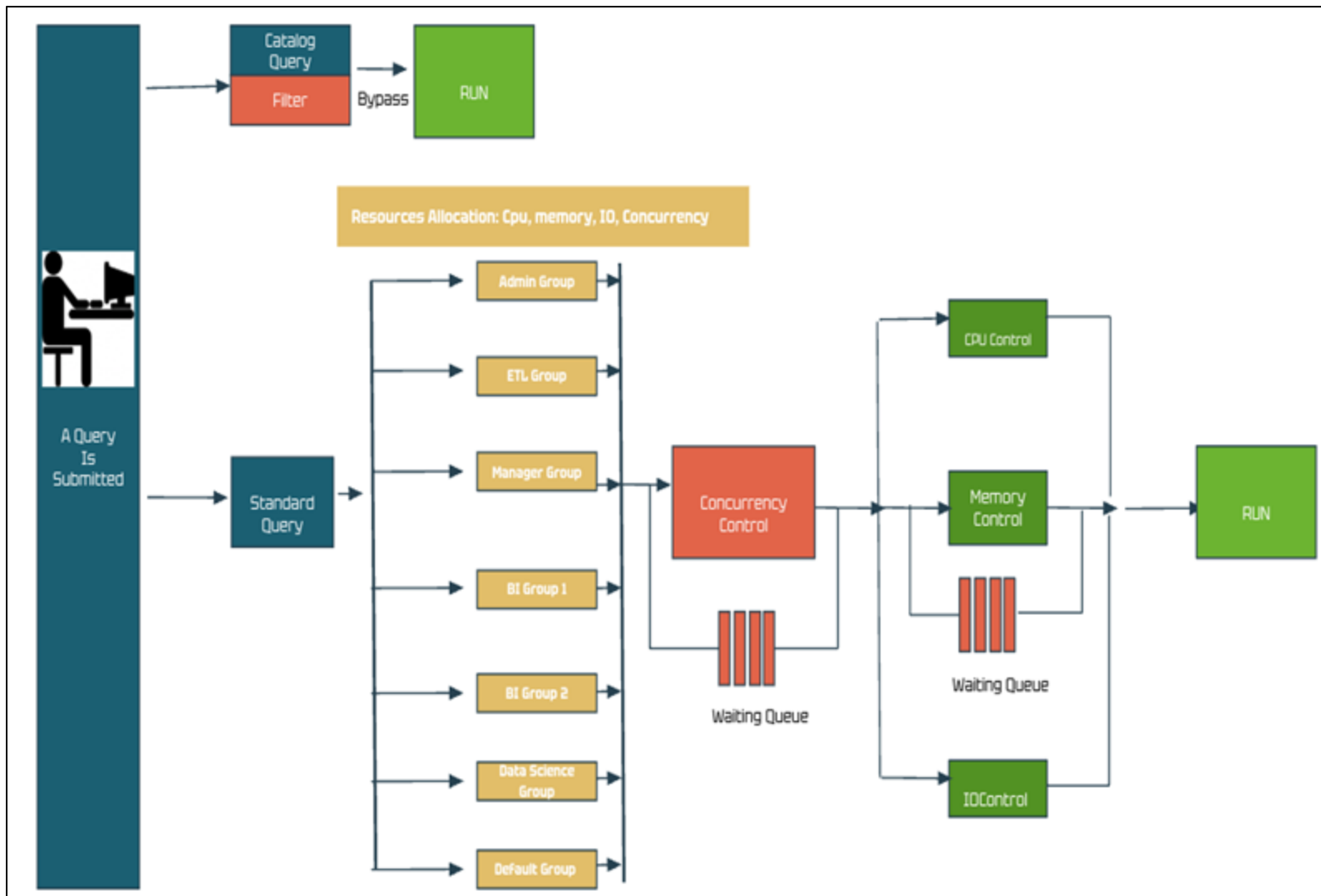
- Online 백업 / 복구 지원
- Incremental / Full 백업 복구 지원
- 백업 프로세스 동시 개수 조정
- DB 백업 후 특정 테이블 복원 기능 제공
- Object Storage 및 NAS 스토리지 지원

3 백업 성능

- 시간당 ?? TB 이상 지원
- $$x.x \text{ GB/sec} / \text{Node} * x \text{ Node} * 3600 \text{ sec/hr} = xx,x00 \text{ GB/hr} = xx \text{ TB/hr}$$

워크로드 관리 - Based on Linux cGroups V1 & V2

- WarehousePG의 리소스 그룹은 멀티 테넌트 리소스 관리 및 워크로드 격리에 필수적입니다.



각 그룹은 다음과 같은 정책을 적용합니다.:

- ✓ CPU 와 Memory 쿼터 설정
- ✓ 쿼리 동시 처리 수 제한
- ✓ 워크로드 우선순위 설정
- ✓ I/O 관리
- ✓ 논리적 워크로드 분리처리

```
CREATE RESOURCE GROUP BGroup1 WITH  
(CONCURRENCY=20,  
CPU_MAX_PERCENT=20, MEMORY_QUOTA=250,  
CPU_WEIGHT=500, MIN_COST=50,  
IO_LIMIT='pg_default: wbps=1000,  
rbps=1000, wiops=100, riops=100');
```

```
CREATE ROLE robert RESOURCE GROUP BGroup1;
```

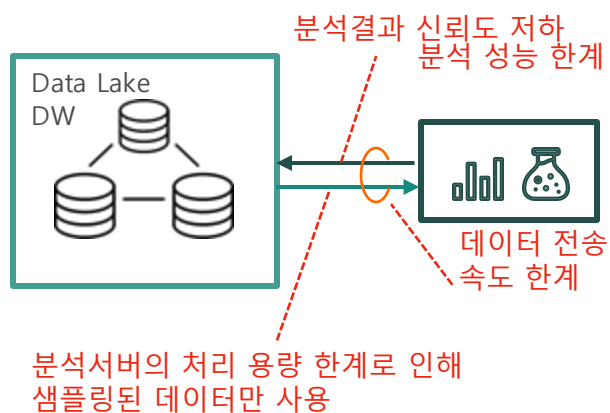
Parallel ML/AI Analytics

최신 R, Python 라이브러리로 병렬 ML/AI 고급 분석

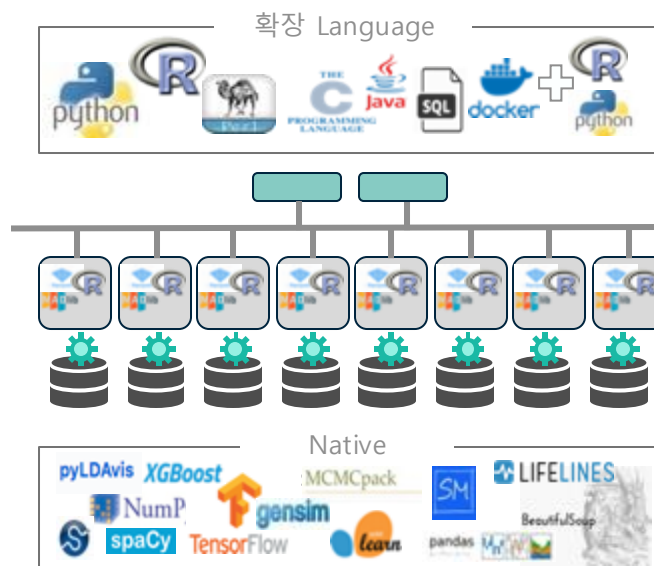
ML/AI 분석

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



Greenplum의 In-DB 분석 방식



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

병렬 ML/AI 지원 라이브러리

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

지원 언어	지원 라이브러리
MADLib	<ul style="list-style-type: none"> • SQL 기반의 Open Source 머신러닝/AI 라이브러리 • SUPERVISED LEARNING: Neural Network, Regression, Tree etc. • UNSUPERVISED LEARNING: Association Rules, Clustering, Topic etc. • Time Series Analysis: ARIMA • GRAPH: Page Rank, Single Source Shortest Path etc.
PL/R	<ul style="list-style-type: none"> • Open Source R 3.6+ • 지원 라이브러리: R cran에서 제공되는 모든 Library
PL/Python	<ul style="list-style-type: none"> • Open Source Python 3.8+ • 지원 라이브러리: Numpy, TensorFlow, Pandas, spacy 등의 OSS 모든 라이브러리
기타 지원 언어	<ul style="list-style-type: none"> • PL/PGSQL : DB의 함수, SQL 기반의 프로시저 • PL/Java : Java 기반의 함수 지원 • PL/C, C++ : C 기반의 User 함수 지원 • PL/Perl : Perl 기반의 User 함수 지원

Parallel ML/AI Analytics (예시)

PL/R과 PL/Python 예시

병렬 ML/AI를 위한 PL/R 예시

```
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';
```

R 코드

```
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
) a
ORDER BY id;
```

R 함수 호출

병렬 ML/AI를 위한 PL/python 예시

```
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';
```

Python 코드

```
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
) a
ORDER BY id;
```

Python 함수 호출





















Greenplum에서 각 데이터 노드별로 병렬 처리

Example

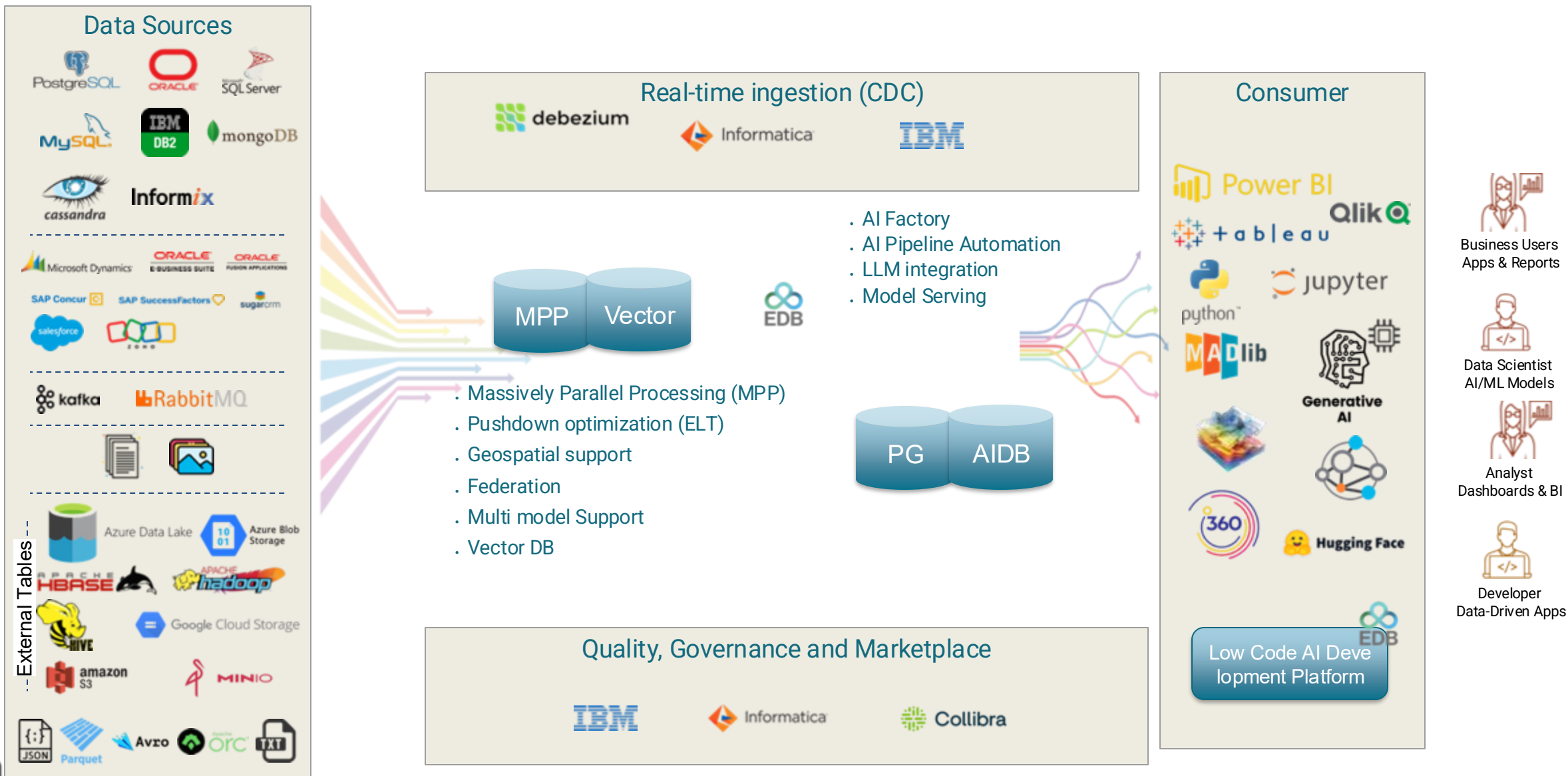
AI 분석을 위한 MPP 플랫폼

The AI-Native MPP Platform for Hybrid Analytics

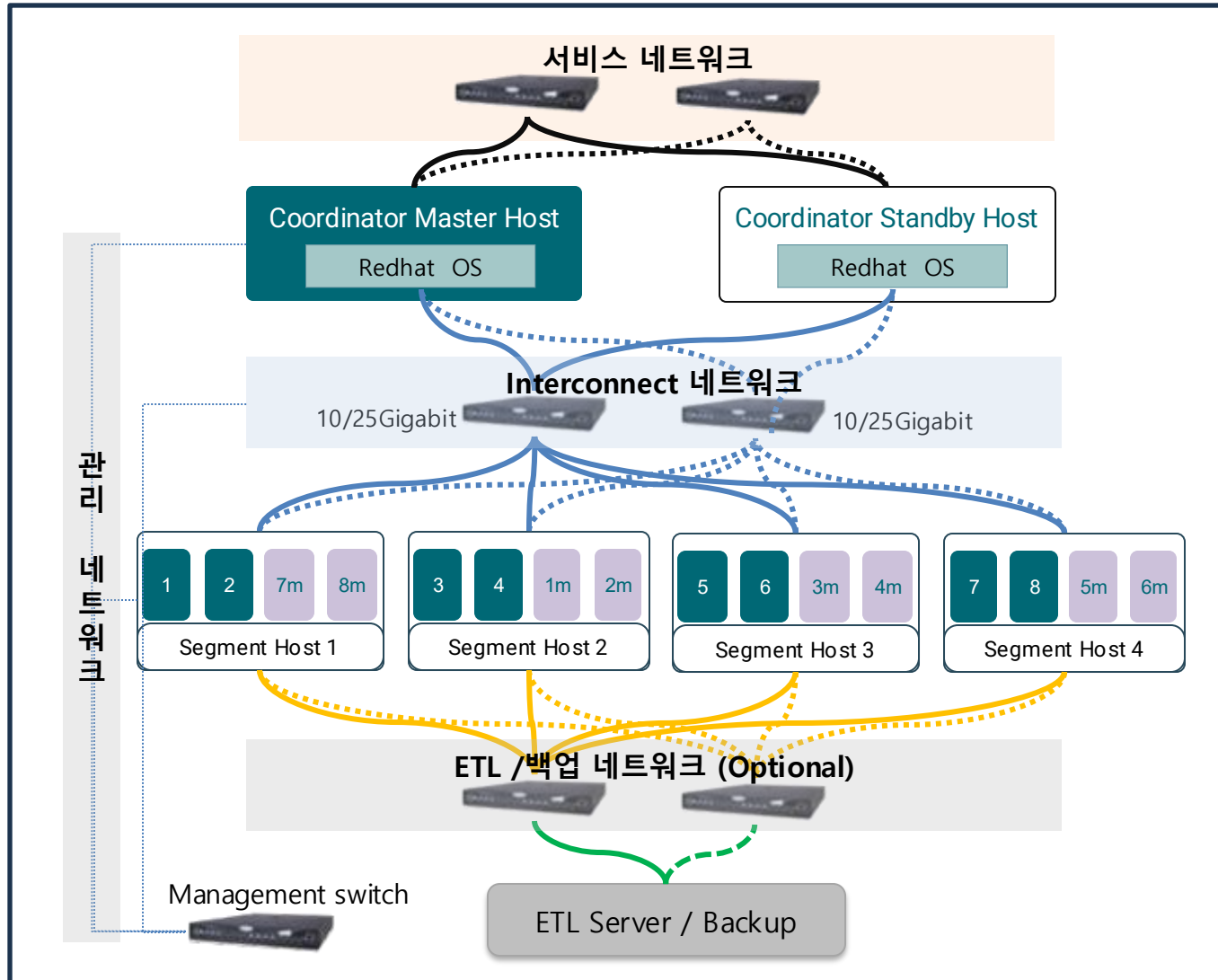


 Distributed MPP SQL Engine	 Full ACID Compliance	 Petabyte-Scale Performance	 Data Federation Layer	 Hybrid Storage
 Unified AI & Analytics Engine	 Vector Database Built-In	 In-Database Machine Learning	 Geospatial Intelligence	 Time Series & Forecasting
 Searchable Database btree, bitmap, Gin, Gist, HSNW, Hash, Brin	 Analytical Optimizer "Orca"	 PI-Container Python, R	 Image & Video Recognition LLM Integration	 GPU Acceleration
 Workload Management CPU, Memory, IO, Concurrency	 Object Storage Integration	 HTAP Architecture	 PostgreSQL Extensible	 Enterprise Grade Security

EDB WarehousePG & AI Reference Architecture



지원 환경, Deployment



지원 OS

- Redhat 7(제한적), 8, 9, 10
- Redhat Compatible OS
- 지속적 지원 예정

Bare-Metal 환경

- 인프라 스트럭처 구성 권장 방안 제공 (서버, 네트워크, 스토리지 등)
- OS 설정, WarehousePG 설치 및 구성

Vmware 등 가상 환경

- 인프라 스트럭처 구성 권장 방안 제공
- OS 설정, WarehousePG 설치 및 구성
- 자동 배포 (Ansible, Terraform)

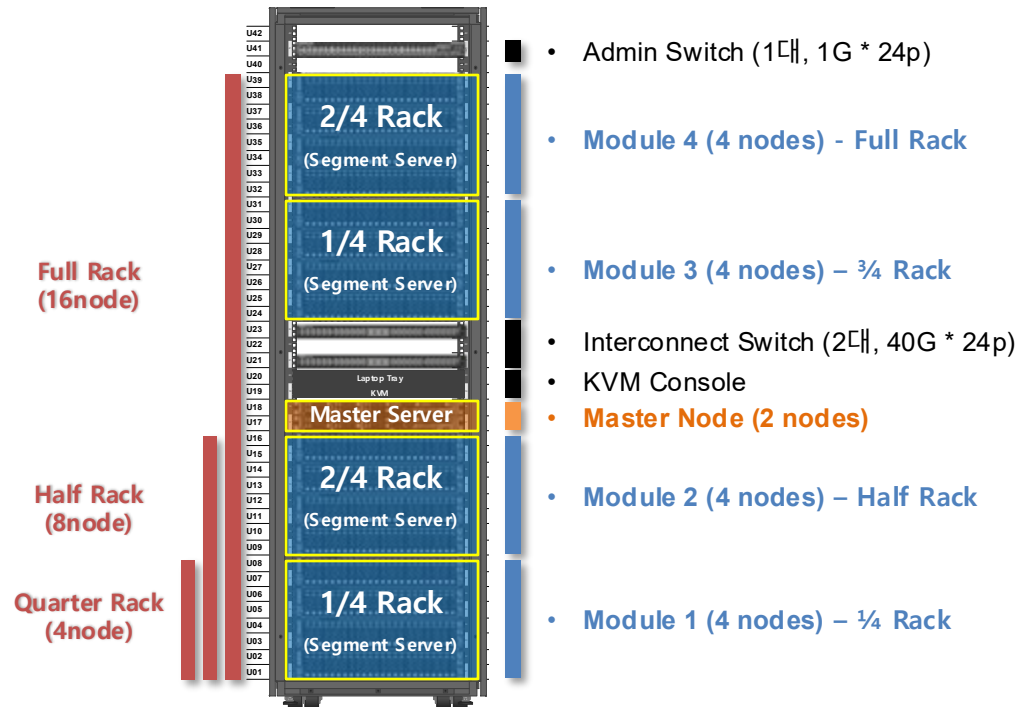
Private/Public Cloud 환경

- VM 구성 가능한 모든 클라우드에 설치 가능
- OS 설정, WarehousePG 설치 및 구성
- 자동 배포 (Ansible, Terraform, CloudFormation)

하드웨어 표준 구성

아래 구성은 표준 구성이며, DW, ADW 등 사용 용도 및 필요 용량에 따라 CPU/Memory/Disk/Network 구성은 변경될 수 있습니다. (HW Vendor도 변경 가능)

▪ Rack 구성도 (Full Rack)



▪ Master Server : 2EA

Model	Dell PowerFlex Appliance R650
CPU cores	Intel Gold 6342 @2.80GHz 24C/48T * 2EA
Memory	512 GB
Disk	6 x 1.92TB SAS SSD 2.5" drives (RAID 5) RAID card : PERC H755 * 1ea
Network	Dual port 40G * 2장

▪ Segment Server : 4Module * 4EA = 16EA

Model	Dell PowerFlex Appliance R750
CPU cores	Intel Gold 6342 @2.80GHz 24C/48T * 2EA
Memory	1024 GB
Disk	24 x 3.84TB SAS SSD 2.5" drives (RAID 6) RAID card : PERC H755 * 2ea
Network	Dual port 40G * 2장

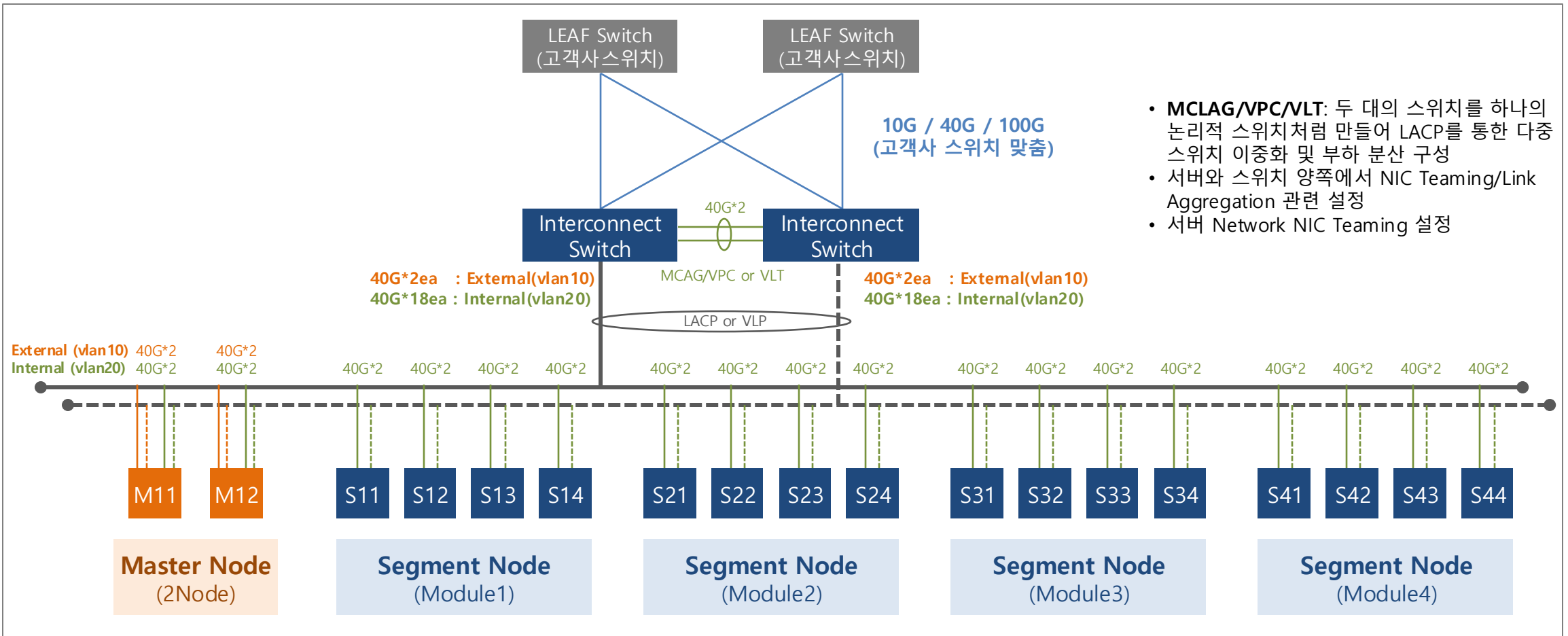
▪ Rack 구성별 사양 및 스토리지 용량

Rack 종류	Master Node		Segment Node			
	노드 수	구성 스펙	노드 수	CPU/Memory	물리 용량	가용량(RAID6 구성후)
Quarter Rack	2	CPU : 2.8GHz 48C, Memory : 512GB, 물리용량 : 11.52TB	4	CPU : 2.8GHz 192C, Memory : 4,096GB	368.53	307.2
Half Rack	2	가용량 : 7.68TB	8	CPU : 2.8GHz 384C, Memory : 8,192GB	737.28	614.4
Full Rack	2		16	CPU : 2.8GHz 768C, Memory : 16,384GB	1474.56	1227.8

기본 구성 > Network 구성 (1안)

■ Network 구성도 (Full Rack)

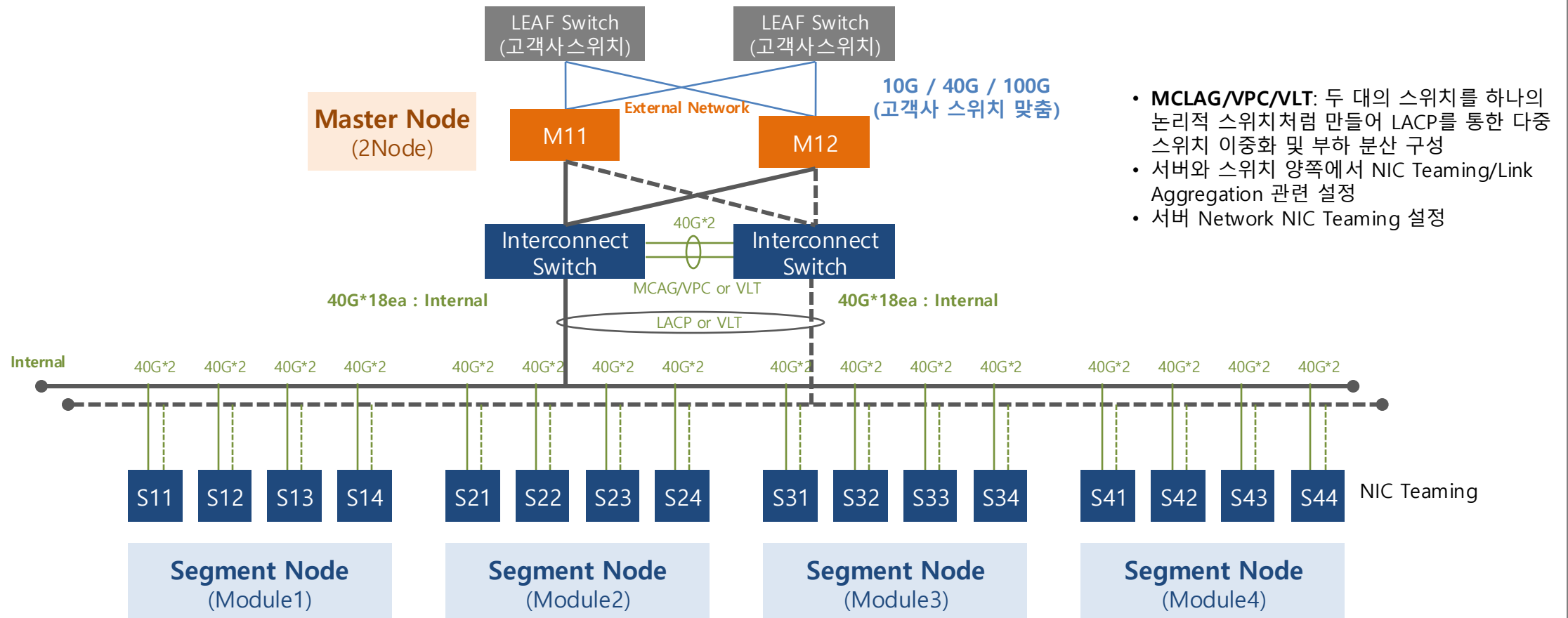
- 시스템(스위치, 서버)은 40G/100G 지원 가능 (SFP Type 변경만으로 40G, 100G 선택하여 사용가능)
- SAS SSD 사용 시 40Gbps 네트워크 구성 권고, NVMe SSD 사용 시 100Gbps 네트워크 구성 권고
- 네트워크 구성은 동일하며, SFP만 40G/100G Type 선택



기본 구성 > Network 구성 (2안)

- **Network 구성도 (Full Rack)**

- 시스템(스위치,서버)은 40G/100G 지원 가능 (SFP Type 변경만으로 40G, 100G 선택하여 사용가능)
- SAS SSD 사용 시 40Gbps 네트워크 구성 권고, NVMe SSD 사용 시 100Gbps 네트워크 구성 권고
- 네트워크 구성은 동일하며, SFP만 40G/100G Type 선택



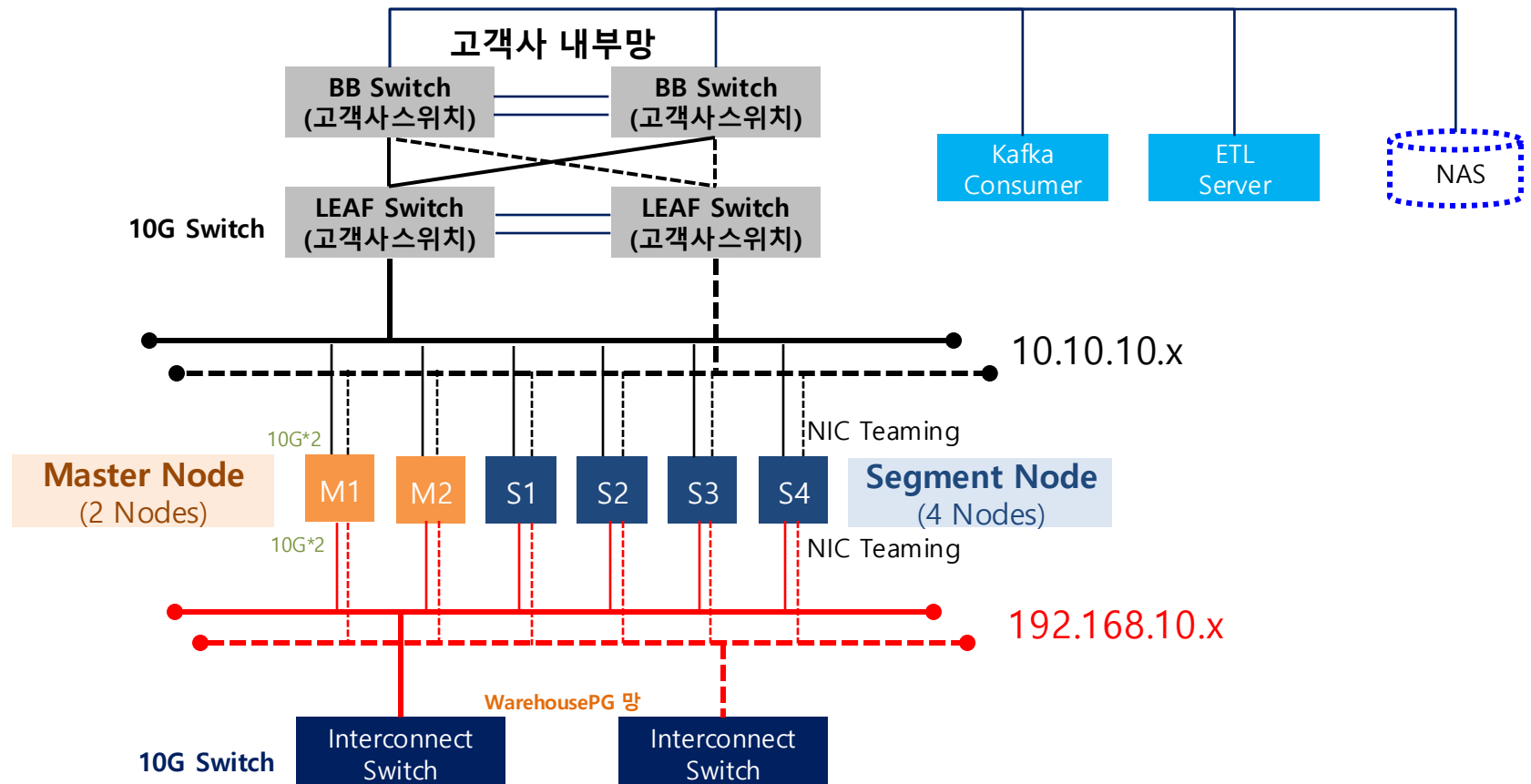
기본 구성 > Network 구성 (3안)

■ Network 구성도 (Quarter Rack)

- 백본시스템(스위치,서버)에 Leaf 스위치 연결 후 Leaf 스위치에서 WarehousePG 서버 연결
- WarehousePG용 사설망 구성(내부 트래픽 처리)

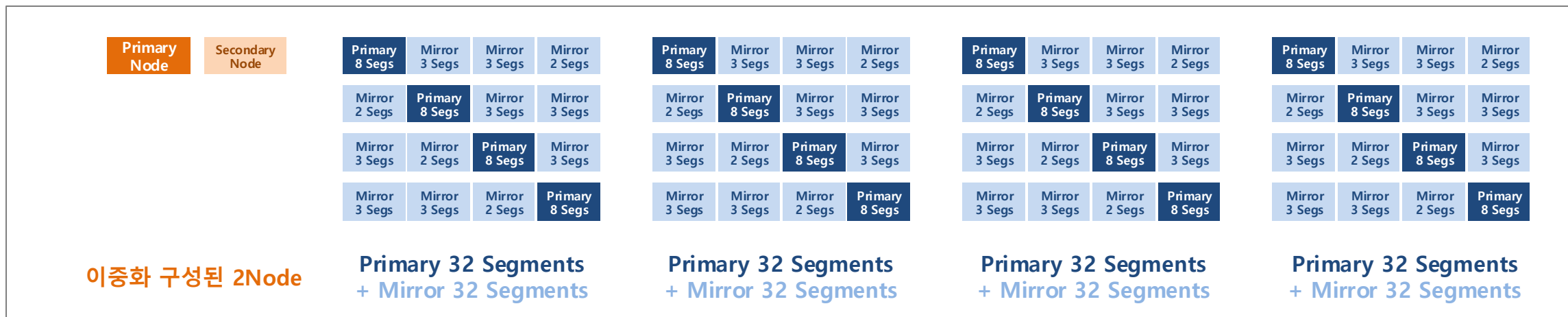
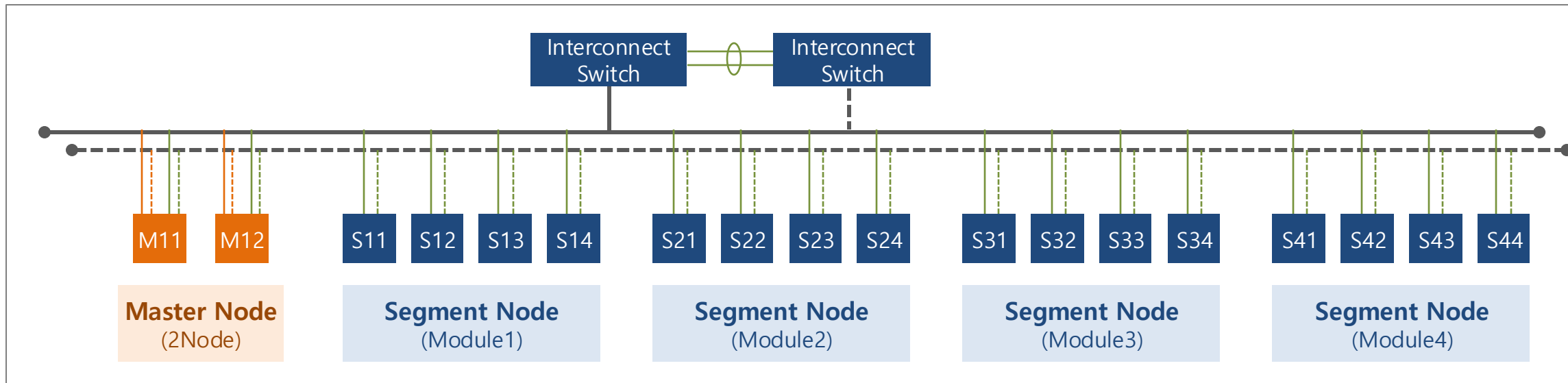
mdw-ext 10.10.10.11
smdw-ext 10.10.10.12
seg01-ext 10.10.10.111
seg02-ext 10.10.10.112
seg03-ext 10.10.10.113
seg04-ext 10.10.10.114

mdw 192.168.10.11
smdw 192.168.10.12
seg01 192.168.10.111
seg02 192.168.10.112
seg03 192.168.10.113
seg04 192.168.10.114



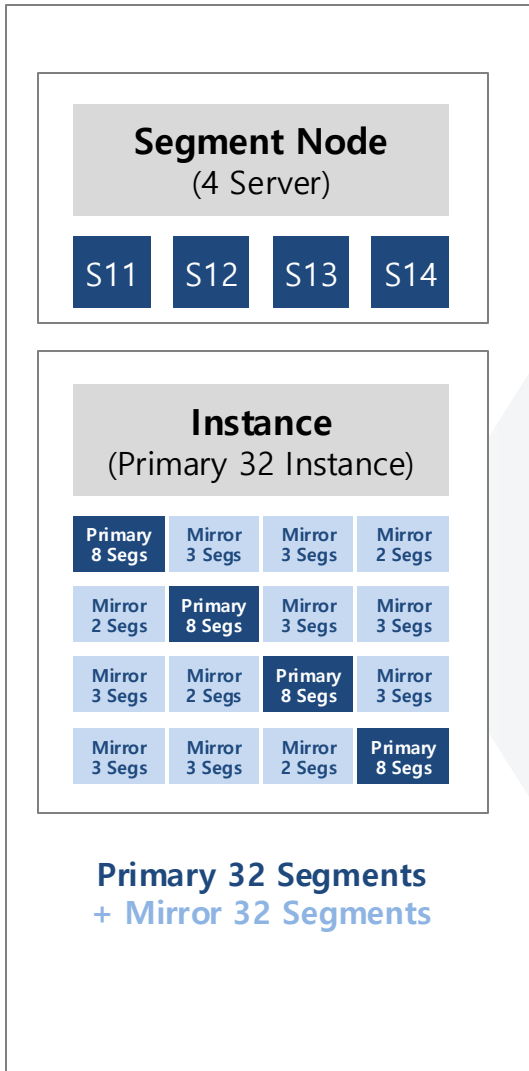
기본 구성 > Instance 구성

Instance 구성 (Full Rack)

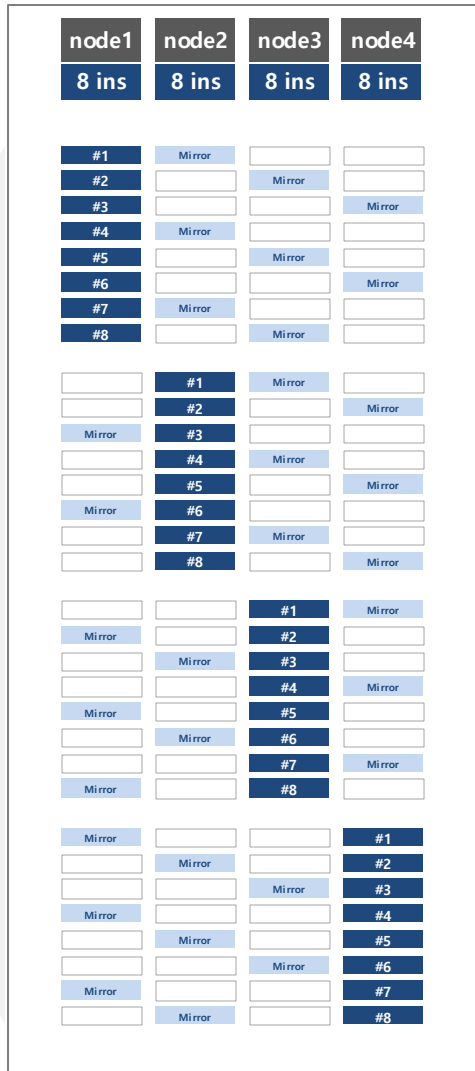


기본 구성 > Instance 구성

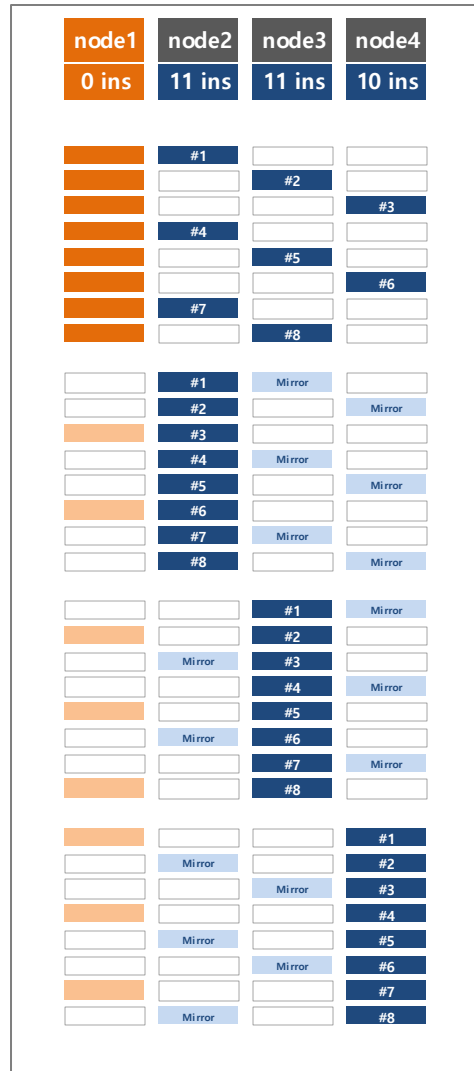
- **Instance Config (1Module)**



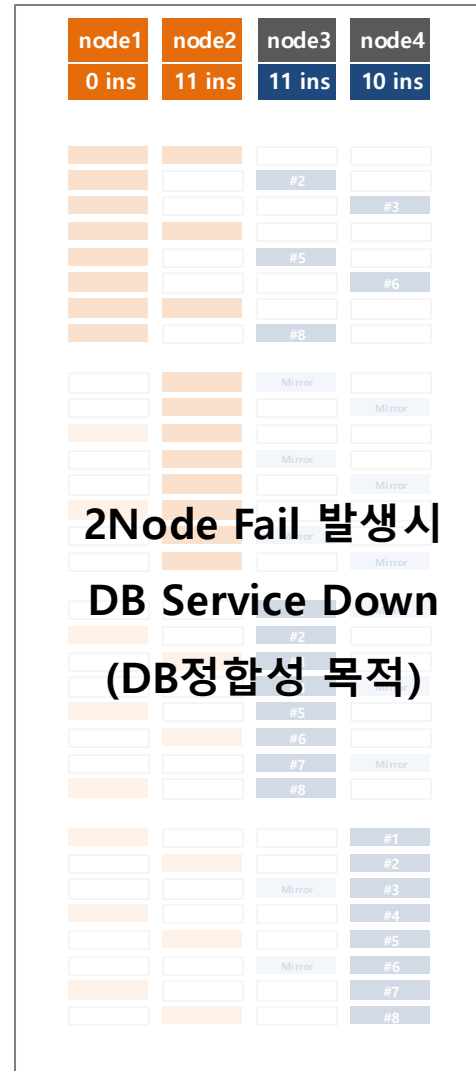
- **Detail Instance Config**



- **1Node Fail Case**



- **2Node Fail Case**



* Round Robin Mirror Config

THANK YOU

고비용을 줄이고 효율을 높이는 것이 불황기의 핵심 전략

