FMS the Future of Memory and Storage

SAMSUNG

# Near Data Processing using Samsung Zero-ETL

## Reference solution

**Pramod Peethambaran**

Director of Engineering, Data Fabric Solution, MSL
Samsung Semiconductor, Inc.

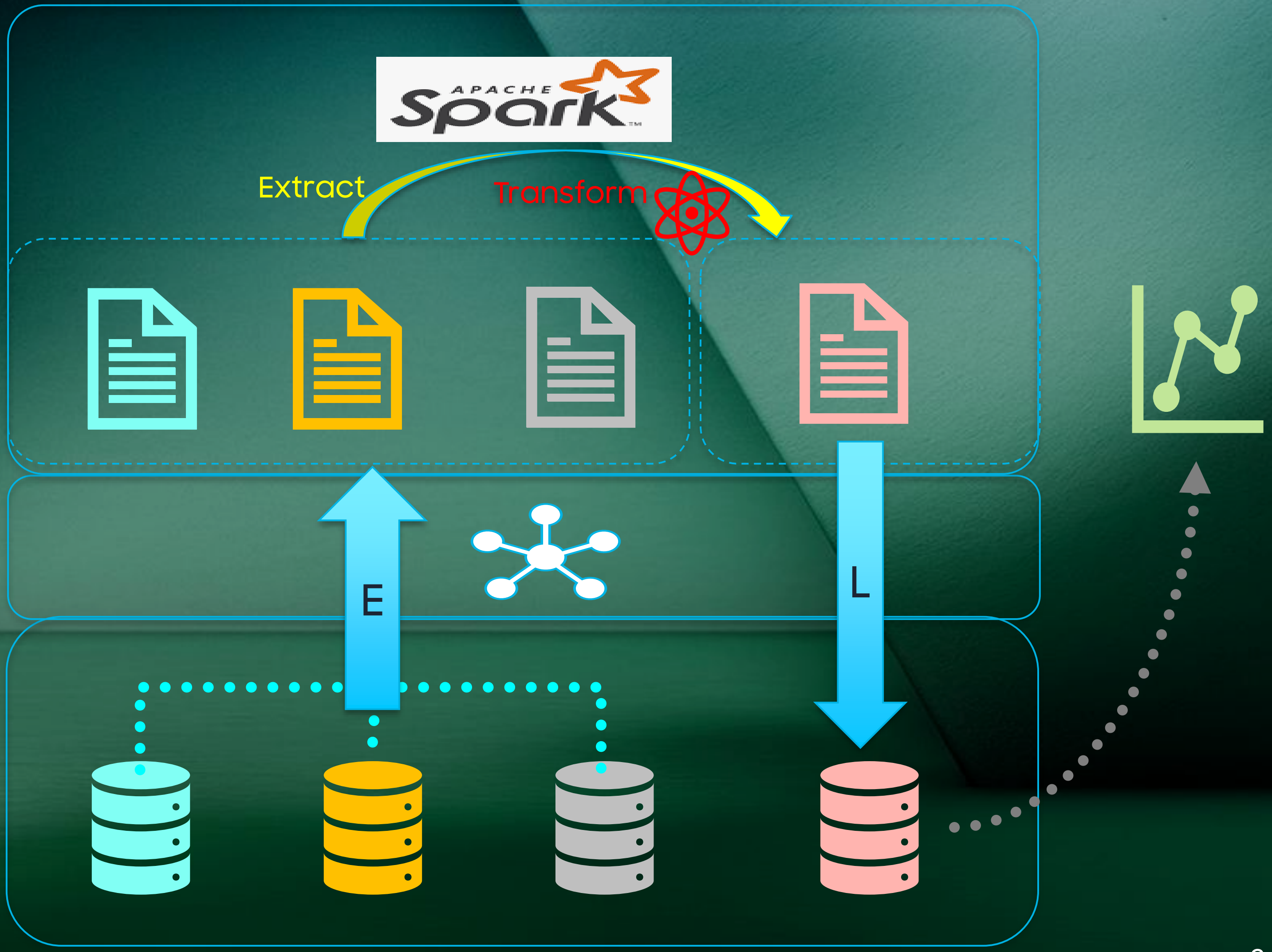
https://bit.ly/SamsungMSL_DFS

# Contents

- Introduction
- High level Data flow comparison
- Overall architecture & deployment model
- How does it help the ETL users
- A FinTech use-case
- Test results
- Conclusion

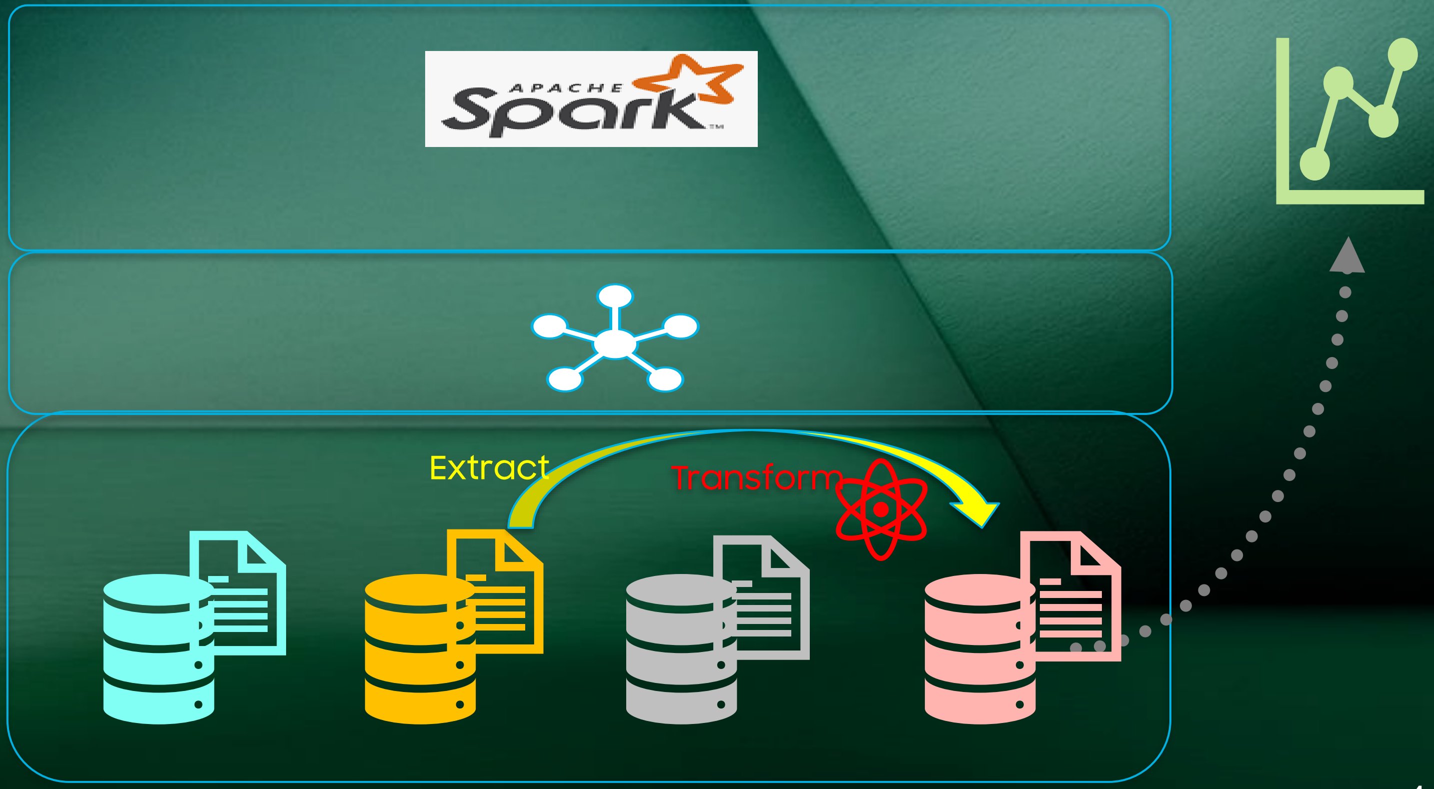
# Typical (high level) Data flow in a ETL pipeline

- Extract
  - Retrieve data from multiple heterogenous sources and format.
- Transform
  - extract data before it can be fed for processing
- Load
  - Load data for further processing, typically for analytics



# Data flow (high level) in a ETL pipeline with Samsung Zero-ETL

- Extract
  - Happens near data (Storage)
- Transform
  - Happens near data (Storage)
- Load
  - Same as traditional ETL pipeline



## Framework for ETL developers

- Framework for creating and loading offloaded compute units to Storage
- Object Storage optimized

## REST APIs based offload compute units

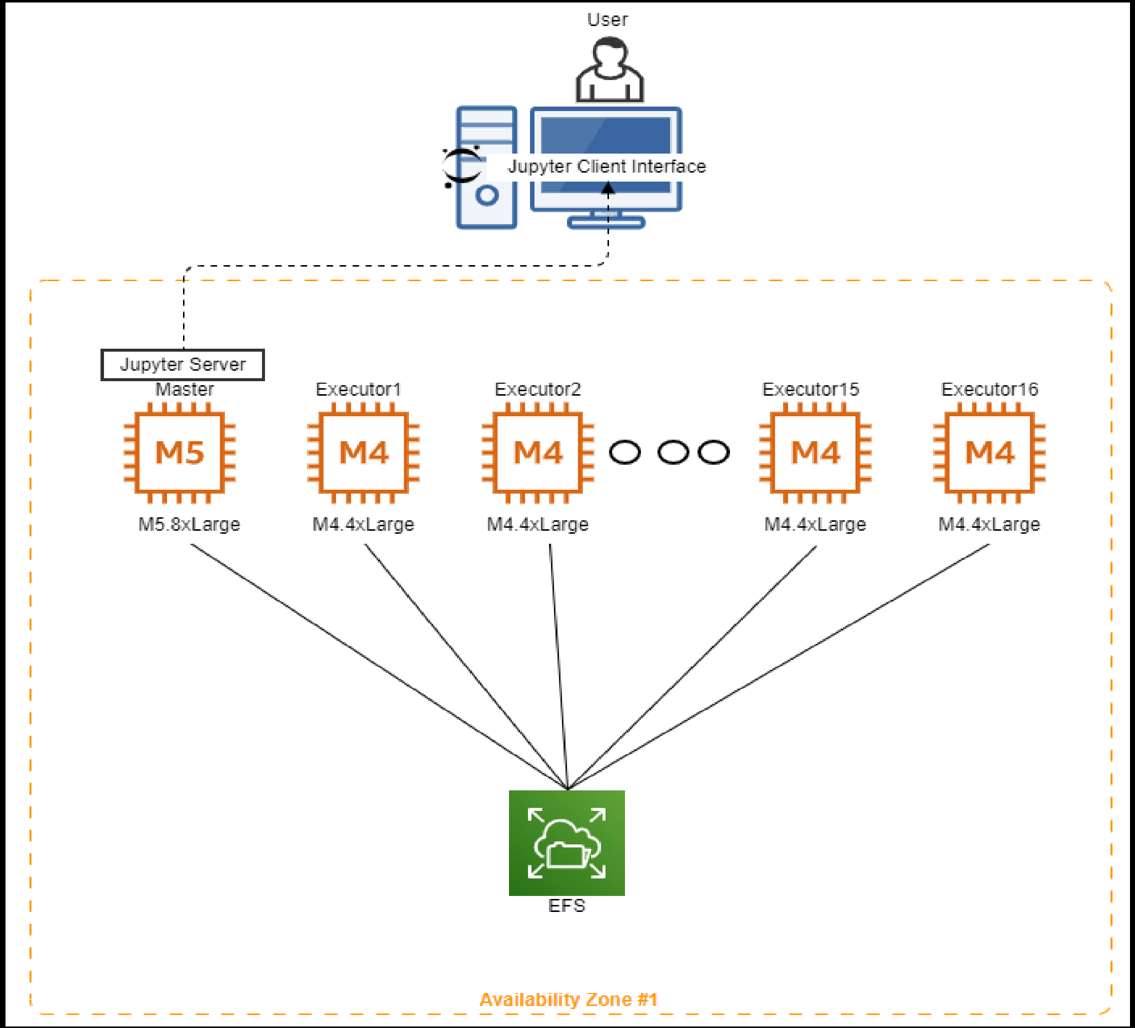
- APIs for Data flow orchestration and error management
- Actual offloaded compute units is user defined via REST APIs

## TCO Savings through Near Data Processing

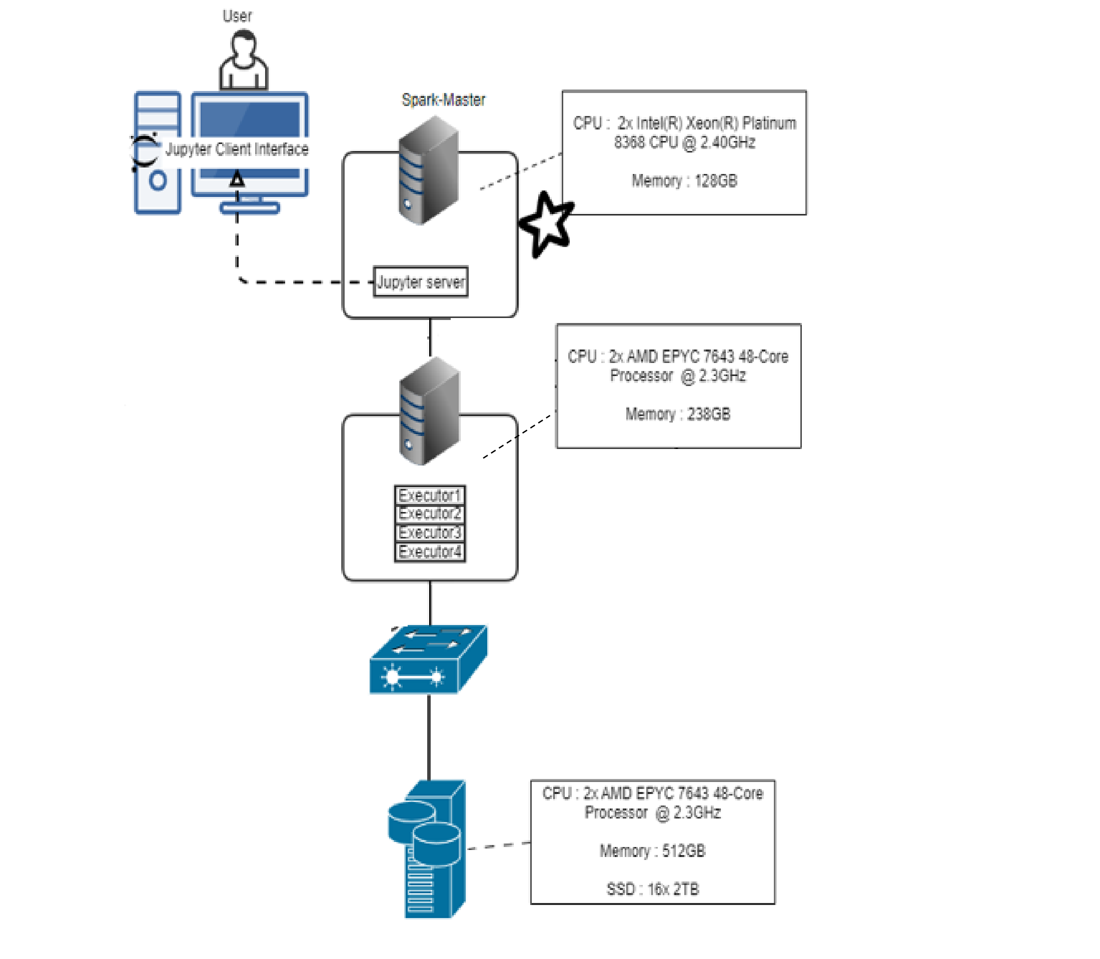
- Built on the foundational concept of Near Data processing (NDP) – reduces data transfer
- Reduces need for expensive compute clients

# Topology: Baseline (EMR based) vs Samsung Zero-ETL* based

## AWS EMR Topology



## Samsung Zero-ETL

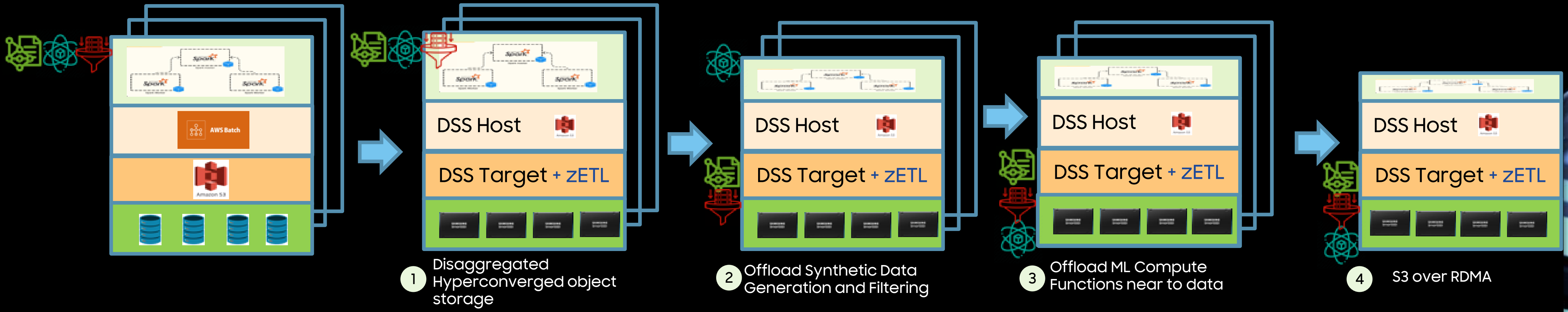

User
Jupyter Client Interface
Spark-Master
Jupyter server
CPU : 2x Intel(R) Xeon(R) Platinum 8368 CPU @ 2.40GHz
Memory : 128GB
CPU : 2x AMD EPYC 7643 48-Core Processor @ 2.3GHz
Memory : 238GB
Executor1
Executor2
Executor3
Executor4
CPU : 2x AMD EPYC 7643 48-Core Processor @ 2.3GHz
Memory : 512GB
SSD : 16x 2TB

* Research Reference Solution

# Zero-ETL

- Data intensive (PBs) ML pipeline but spending 35% Opex in data transfer to run a model
- Needed solution with higher Efficiency/Capacity with near data processing

Phase-wise Adoption



1 Disaggregated Hyperconverged object storage

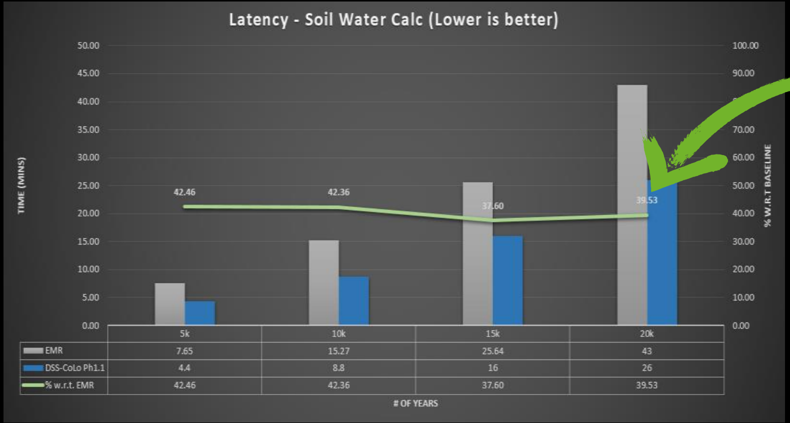
2 Offload Synthetic Data Generation and Filtering

3 Offload ML Compute Functions near to data

4 S3 over RDMA

Spark Conf

- 4 Clients (H)
- 16 Executors (S)
- 1 Driver/Master node
- 16 cores per node
- 64 GB per node



| | 5k | 10k | 15k | 20k |
|---|---|---|---|---|
| EMR | 7.65 | 15.27 | 25.64 | 43 |
| DSS-CoLo Ph1.1 | 4.4 | 8.8 | 16 | 26 |
| % w.r.t. EMR | 42.46 | 42.36 | 37.60 | 39.53 |

40% faster, >30% TCO

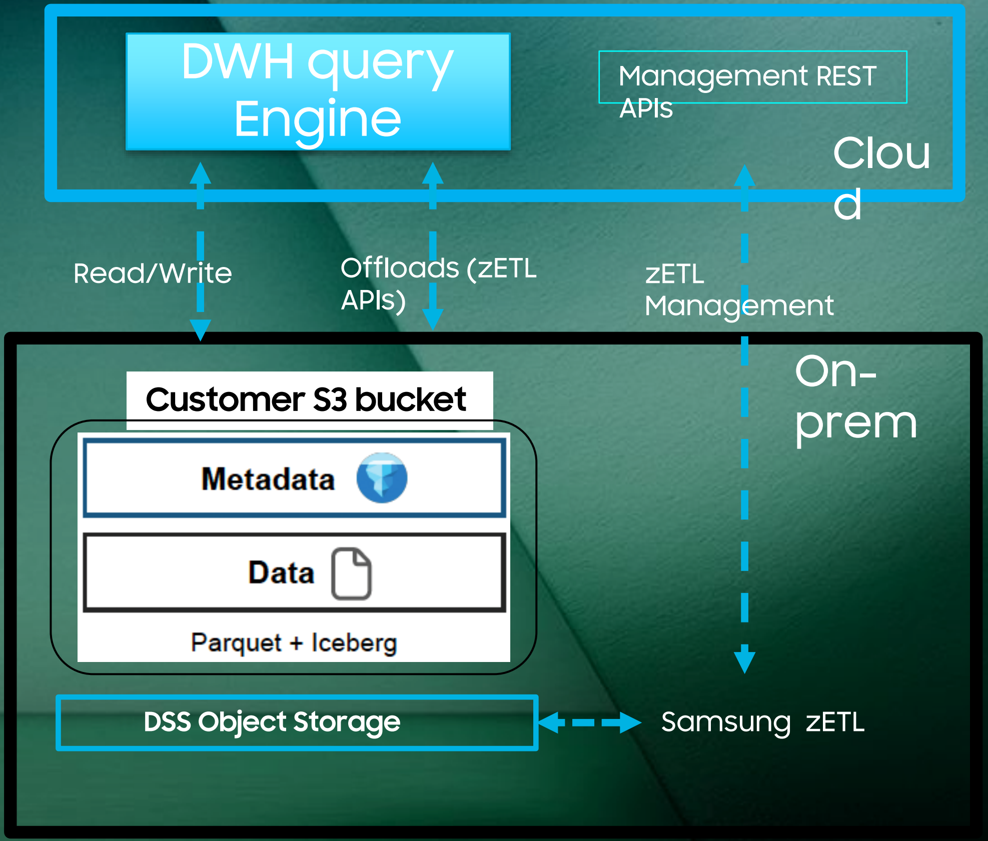
Synthetic Data

ML Compute units

Data Filtering

# Case Study: Deployment model for Datawarehouse(DWH) use-case

- DWH compute engine connected to Customer S3 bucket located in private data center over direct connect
- DSS – Disaggregated Storage Solution, a Open Source ultra high bandwidth object storage: https://github.com/OpenMPDK/DSS
- Samsung zETL* installed alongside of DSS, expose zETL APIs to DWH Connector on-prem/cloud
- DWH can offload compute/ML binaries to DSS without exposing IP
- Developer friendly Samsung zETL APIs
- DWH integration with Management REST APIs to configure zETL



* Research Reference Solution

# Conclusion

- Samsung Zero-ETL*, built on Near Data processing, reduces the data transfer between the compute and the Data Storage.
- Because of reduction of data transfer, customer can reduce # of nodes for processing same data size, hence reducing the TCO
- Developer friendly API for offloading the compute to Data Storage
- Easy way to integrate to the existing Datawarehouse

* Research Reference Solution

# Thank You

SAMSUNG