

머신 러닝을 위한
엔터프라이즈 DW 데모
IIAS with SparkML

강성희 실장
Korea Client Technical Professional
Hybrid Data Management Platform
Data & AI
IBM Korea

Data and AI Forum by IBM

IBM

Modernizing Data Warehouse Infrastructure

(Checklist Report 2018. TDWI 인용)

- ✓ Diversify your portfolio of data platform to satisfy the data requirements of the modern warehouse
- ✓ Modernize your data warehouse with cloud and hybrid platform strategies
- ✓ Modernize data warehouse hardware for greater speed and scale, lower costs, and innovative best practices
- ✓ Coordinate data warehouse modernization with business modernization and analytics modernization
- ✓ Adjust data management best practice to fit modern data warehouse and analytics
- ✓ Leverage multivendor partnership

Analyzing with Spark : Apache Spark Cluster 를 이용한 Modernization

Relational Engine

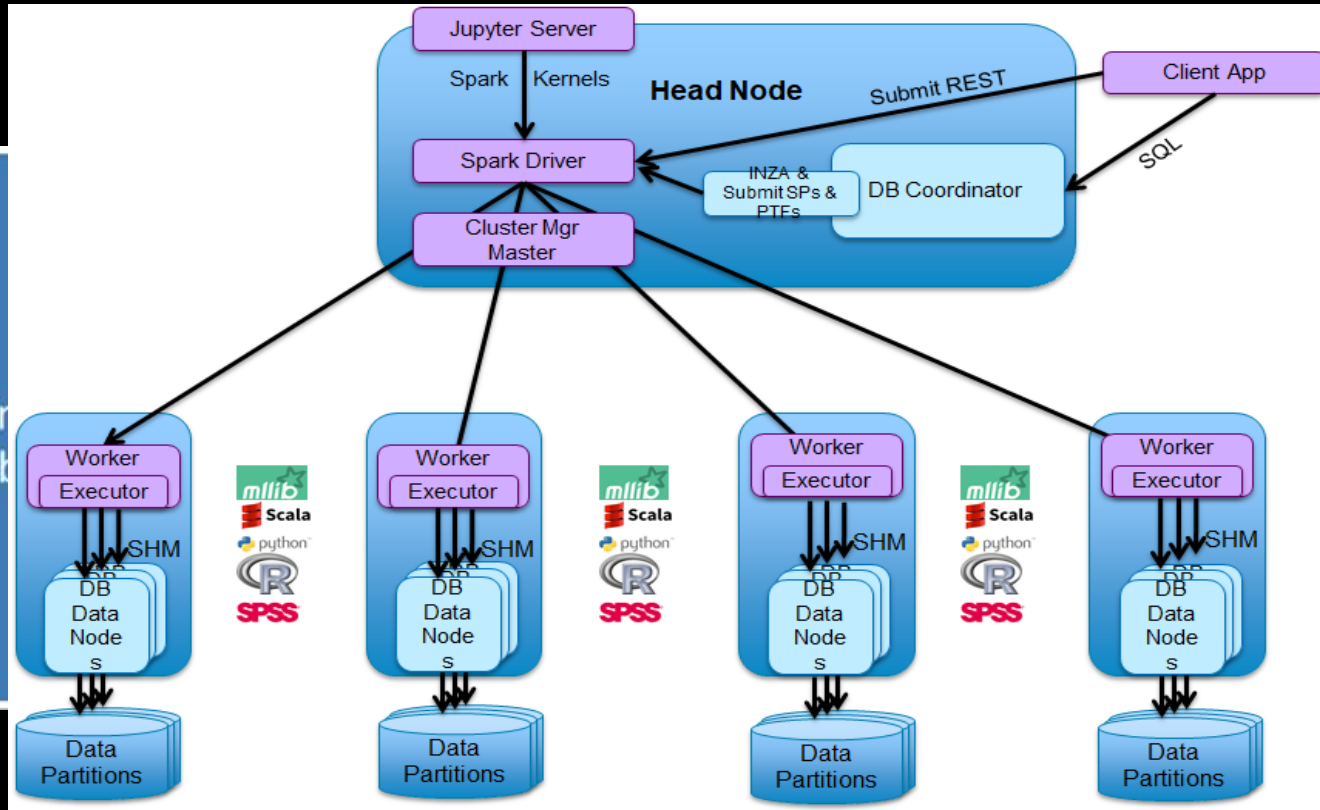
In-memory, columnar database engine
(Db2 BLU), part of IBM Common SQL
Engine



Analytics Engine

Optimization co-location of the
Spark engine

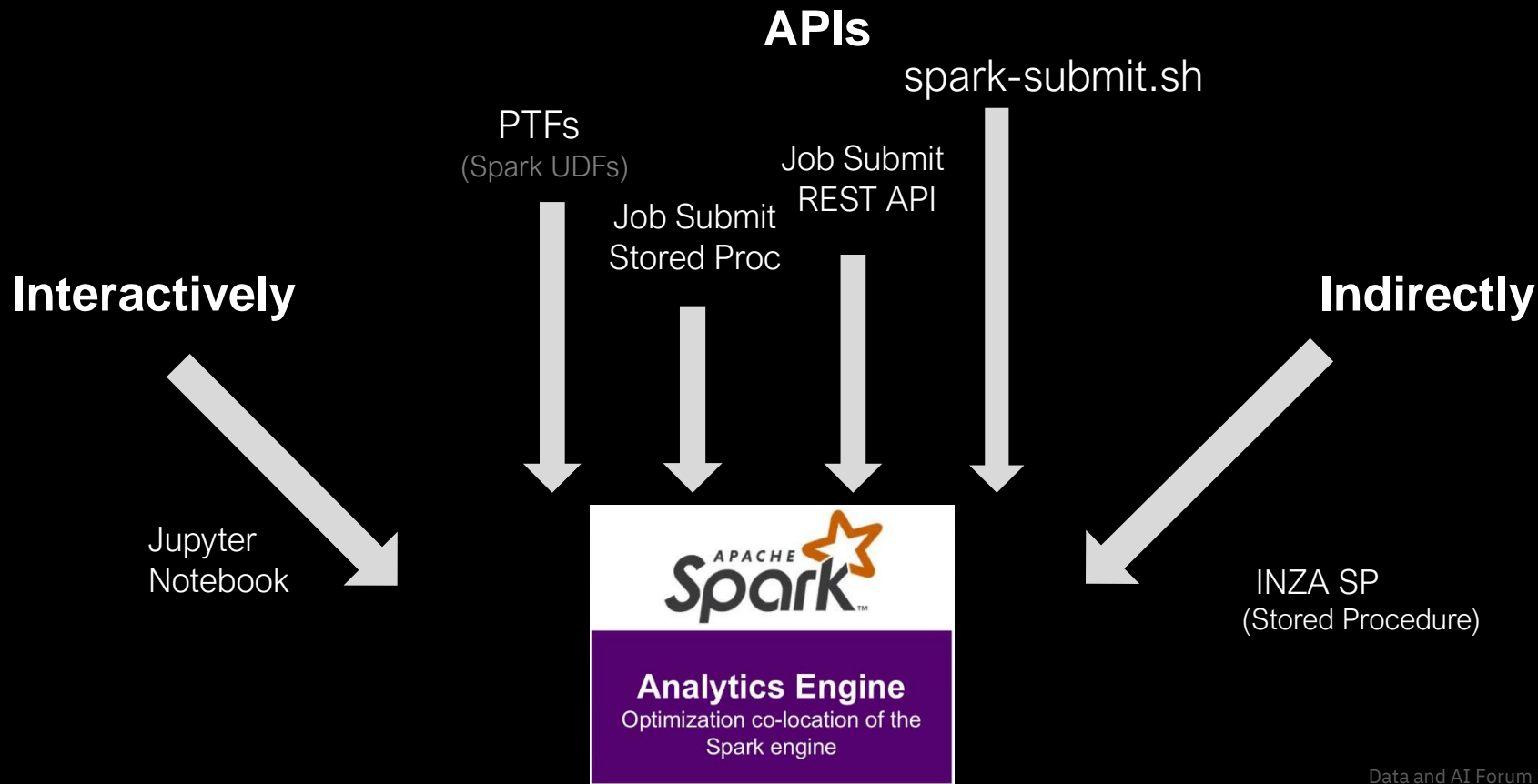
Analyzing with Spark : Apache Spark Cluster 를 이용한 Modernization



IIAS/Db2 Warehouse 에서 Spark 사용의 개요

- R/Python/Java/Scala with Spark 소스를 spark job submit tool, Stored Procedure, Rest API 를 사용하여 실행
- Spark 소스는 데이터베이스 내에서 데이터를 분석하고 결과를 다시 Table 로 저장함
- Spark Cluster 와 DBMS MPP 엔진이 각 노드별로 colocation 되어 있어 application 이 partition 별로 병렬처리하기에 이상적임

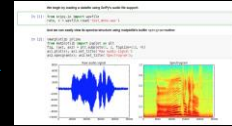
IIAS/Db2 Warehouse 내에서 Spark Application 접근 방법



IIAS/Db2 Warehouse Spark 연동의 개요

- 데이터베이스 서버내에 Spark 연동 모듈과 엔진을 내장. 최근 많이 사용되는 오픈 소스 기반의 분석을 수행

- Database 서버 내에서 별도 구성 없이 Spark 기반의 데이터에 예측모델, 머신 러닝 등을 수행 할 수 있다



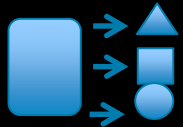
- 알고리즘을 SQL 인터페이스를 통해 Parallel/Scalable 한 환경에서 수행할 수 있다



- SQL 인터페이스에서 SP를 통한 머신러닝 Job 구동



- ELT에 활용하여 복잡한 데이터 추출, 변환, 저장 수행



- IoT 데이터 같은 스트리밍 소스 실시간 변경 / 데이터베이스 저장



- 최근 Object Storage에 저장하는 JSON, CSV, or Parquet 같은 데이터를 변환 처리, 저장



- Rest API를 통한 파이썬 데이터 처리
작업 수행 processing via REST



데모시나리오 요약

Spark ML 수행

- 1) spark_submit.sh command line 호출
- 2) Stored Procedure 를 통한 호출
- 3) REST API 를 이용한 호출

모니터링

결과 확인

Job 수행 예>

1. spark-submit.sh 를 이용한 Job 호출

```
$ spark-submit.sh idax_example.jar --class com.ibm.idax.spark.examples.ReadWriteExampleKMeans
```

```
[bluadmin@node07 - Db2wh bin]$ ./spark-submit.sh ../examples/jars/idax_examples.jar --class com.ibm.idax.spark.examples.ReadWriteExampleKMeans  
--- dashDB spark-submit.sh ---  
Status: submitted  
Description: The application was submitted.  
Submission ID: 20190902010418365000  
ExitCode:  
Message:
```

Job 모니터링 예>

spark-submit.sh --list-app

```
$ spark-submit.sh --list-apps
```

```
[bluadmin@node07 - Db2wh bin]$ ./spark-submit.sh --list-apps  
--- dashDB spark-submit.sh ---
```

SubmissionID	ApplicationID	Status
20190905033959200000	app-20190905034003-0000	running
20190903013806777000	app-20190903013810-0000	ended
20190903013951069000	app-20190903013954-0001	ended
20190904134209367000	app-20190904134213-0002	ended

```
[bluadmin@node07 - Db2wh bin]$ ./spark-submit.sh --list-apps  
--- dashDB spark-submit.sh ---
```

SubmissionID	ApplicationID	Status
20190903013806777000	app-20190903013810-0000	ended
20190903013951069000	app-20190903013954-0001	ended
20190904134209367000	app-20190904134213-0002	ended
20190905033959200000	app-20190905034003-0000	ended

Job 모니터링 예> Web console 모니터링

The screenshot shows the Spark Master web console interface. At the top, the browser address bar shows the URL `:8443/sparkui/bluadmin/`. The page title is "Spark Master for bluadmin" with the Apache Spark logo and version "2.3.0".

Key system information is displayed:

- URL: `spark://node07.ibm.com:25000`
- Alive Workers: 1
- Cores in use: 160 Total, 32 Used
- Memory in use: 18.5 GB Total, 3.7 GB Used
- Applications: 1 Running, 6 Completed
- Drivers: 0 Running, 0 Completed
- Status: ALIVE

Workers (1)

Worker Id	Address	State	Cores	Memory
worker-20190830154305-node07.ibm.com-25002	node07.ibm.com:25002	ALIVE	160 (32 Used)	18.5 GB (3.7 GB Used)

Running Applications (1)

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
app-20190902012757-0006 (kill)	com.ibm.idax.spark.examples.ReadWriteExampleKMeans	32	3.7 GB	2019/09/02 01:27:57	bluadmin	RUNNING	3 s

Completed Applications (6)

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
app-20190902010421-0005	com.ibm.idax.spark.examples.ReadWriteExampleKMeans	32	3.7 GB	2019/09/02 01:04:21	bluadmin	FINISHED	11 s
app-20190902004442-0004	Apache Toree	32	3.7 GB	2019/09/02 00:44:42	bluadmin	FINISHED	3.4 min
app-20190831235505-0003	Apache Toree	32	3.7 GB	2019/08/31 23:55:05	bluadmin	FINISHED	58 min
app-20190831111630-0001	Apache Toree	32	3.7 GB	2019/08/31 11:16:30	bluadmin	FINISHED	12.6 h
app-20190830172046-0000	Apache Toree	32	3.7 GB	2019/08/30 17:20:46	bluadmin	FINISHED	30.5 h
app-20190831233909-0002	Apache Toree	32	3.7 GB	2019/08/31 23:39:09	bluadmin	FINISHED	14 min

데모시나리오

2. SP 이용

```
db2 => CALL IDAX.SPARK_SUBMIT(?, 'appResource=idax_examples.jar | mainClass=com.ibm.idax.spark.examples.ReadWriteExampleKMeans');
```

```
db2 => CALL IDAX.SPARK_SUBMIT(?, 'appResource=idax_examples.jar | mainClass=com.ibm.idax.spark.examples.ReadWriteExampleKMeans');
```

```
Value of output parameters
```

```
-----  
Parameter Name : SUBMISSION_ID
```

```
Parameter Value : 20190905224617142000
```

```
Return Status = 0
```

```
db2 => quit
```

데모시나리오

3. REST API 이용

```
$ curl -k --user "bluadmin":"bluadmin" -X POST "https://10.12.13.97:8443/dashdb-api/analytics/public/apps/submit" --header "Content-Type:application/json;charset=UTF-8" -d '{"appResource":"idax_examples.jar","mainClass":"com.ibm.idax.spark.examples.ReadExample"}
```

```
[bluadmin@node07 ~]$ curl -k --user "bluadmin":"bluadmin" -X POST "https://10.12.13.97:8443/dashdb-api/analytics/public/apps/submit" --header "Content-Type:application/json;charset=UTF-8" -d '{"appResource":"idax_examples.jar","mainClass":"com.ibm.idax.spark.examples.ReadExample"}
```

```
{"statusDesc":"The application was submitted.","submissionId":"20190905230600480000", "exitInfo":{"code":"","details":[],"message":"","messageDB":""},"resultCode":200,"applicationId":"app-20190905230604-0006","username":"bluadmin","status":"submitted"}
```

```
[bluadmin@node07 ~]$
```

```
[bluadmin@node07 ~]$
```

```
[bluadmin@node07 ~]$
```

```
[bluadmin@node07 ~]$
```

```
[bluadmin@node07 ~]$ ./spark-submit.sh --list-apps
```

```
--- dashDB spark-submit.sh ---
```

SubmissionID	ApplicationID	Status
20190905033959200000	app-20190905034003-0000	ended
20190905034452659000	app-20190905034456-0001	ended
20190905150225236000	app-20190905150229-0002	ended
20190905224153221000	app-20190905224157-0000	ended
20190905224617142000	app-20190905224621-0001	ended
20190905230143818000	app-20190905230147-0002	ended
20190905230249799000	app-20190905230253-0003	ended
20190905230409565000	app-20190905230413-0004	ended
20190905230544802000	app-20190905230548-0005	ended
20190905230600480000	app-20190905230604-0006	ended

데모시나리오

3. REST API 이용

```
$ curl -k --user "bluadmin":"bluadmin" -X POST "https://10.12.13.97:8443/dashdb-api/analytics/public/apps/submit" --header "Content-Type:application/json;charset=UTF-8" -d '{"appResource":"idax_examples.jar","mainClass":"com.ibm.idax.spark.examples.ReadExample"}'
```

```
[bluadmin@node07 ~]$ curl -k --user "bluadmin":"bluadmin" -X POST "https://10.12.13.97:8443/dashdb-api/analytics/public/apps/submit" --header "Content-Type:application/json;charset=UTF-8" -d '{"appResource":"idax_examples.jar","mainClass":"com.ibm.idax.spark.examples.ReadExample"}'
```

```
{"statusDesc":"The application was submitted.,"submissionId":"20190905230600480000", "exitInfo":{"code":"","details":[],"message":"","messageDB":"","resultCode":200,"applicationId":"app-20190905230604-0006","username":"bluadmin","status":"submitted"}
```

```
[bluadmin@node07 ~]$
```

```
[bluadmin@node07 ~]$
```

```
[bluadmin@node07 ~]$
```

```
[bluadmin@node07 ~]$
```

```
[bluadmin@node07 ~]$ ./spark-submit.sh --list-apps
```

```
--- dashDB spark-submit.sh ---
```

SubmissionID	ApplicationID	Status
20190905033959200000	app-20190905034003-0000	ended
20190905034452659000	app-20190905034456-0001	ended
20190905150225236000	app-20190905150229-0002	ended
20190905224153221000	app-20190905224157-0000	ended
20190905224617142000	app-20190905224621-0001	ended
20190905230143818000	app-20190905230147-0002	ended
20190905230249799000	app-20190905230253-0003	ended
20190905230409565000	app-20190905230413-0004	ended
20190905230544802000	app-20190905230548-0005	ended
20190905230600480000	app-20190905230604-0006	ended

IBM Integrated Analytics System

Netezza Heritage + New Hardware + Data Science + Cloud Ready

기존 모델의 장점은 더 강화

Simplicity (간편성)

- 인덱스 관리 불필요 : Storage Index / Zonemap / 자동압축 / No Admin
- 물리적 모델링 없이 바로 사용 : Load & Go / 성능개선용 물리모델설계 불필요

Performance (성능)

- 고성능 하드웨어 : 더 빠른 CPU, 더 많은 Cache, 더 큰 Memory
- Micro latency Flash Storage** : I/O 지연 없는 데이터 전송
- In-Memory** 기술 : 컬럼기반의 인메모리 기술로 N3001대비 수배 성능향상
- 신뢰성 높은 스토리지 : 2-Dimensional flash RAID (99.999% 신뢰 있는 하드웨어 컴포넌트)

In-Place Expansion (확장)

- 모듈 방식 확장 : Server + Storage 증설
- 스토리지 확장 지원 : Storage 증설



M4002

최신 기술 적용

In-Memory (인메모리 기술)

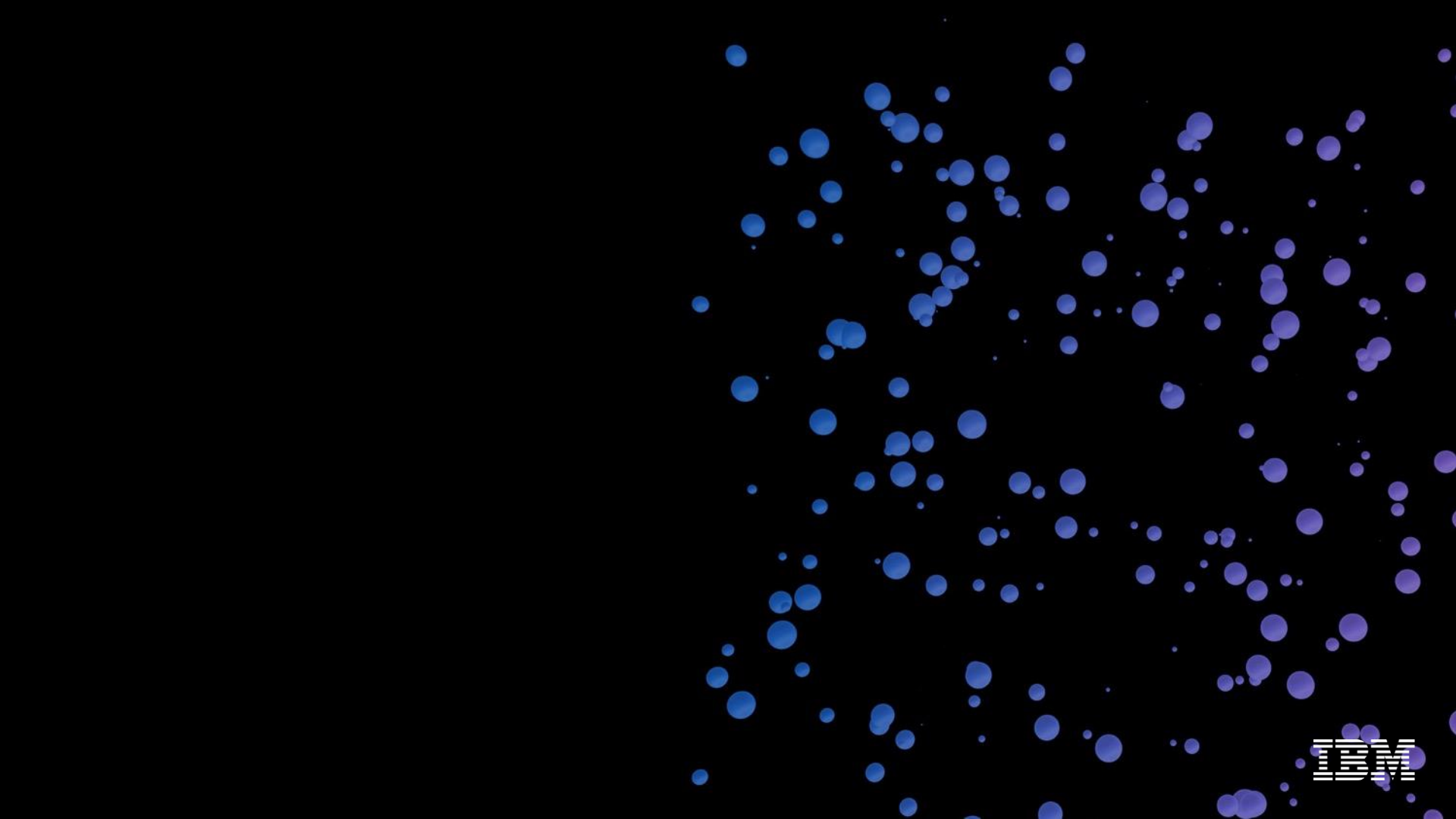
- 집계 및 분석에 용이한 컬럼기반 데이터베이스 기술 : IBM의 BLU기술을 이용하여 성능 및 기술 안정성 확보
- 속도 향상 : Analytic Query PDA N3001대비 수배 성능향상
- 리소스 최적화 : I/O를 최소화 하고 CPU사용률을 절감

Analytic Tool (분석을 위한 틀)

- 데이터 분석가를 위한 환경 : 분석에 필요한 머신러닝 Lib, Jupyter Notebook, Rstudio등을 기본 탑재하여 Analytic Sandbox 분석환경 구축 지원
- 외부 **Spark** 과의 연계

Cloud Ready

- SW Defined Analytic DW** : 동일한 구성으로 Public Cloud, Private Cloud, On-prem DW 구성가능
- 공통 **SQL** 엔진 : 공통 엔진 / 공통 SQL 사용으로 상호 이관이 빠르고 용이함



IBM