차세대 DBMS 연산 스토리지 엔진 기술 개발

4세부

KETI 한국전자기술연구원

연구 목표

클라우드 환경에서 다수의 연산 스토리지 장치들을 고속 네트워크로 상호 연결하여 방대한 데이터 분석/추론 성능 및 에너지 절감을 이루기 위한 차세대 DBMS 연산 스토리지 기술 개발

✔비전: 세계 수준의 차세대 DBMS 스토리지 SW 플랫폼 오픈소스 化

Open CSD
Computational Storage Device

01 이종 DBMS 엔진 지원을 위한 표준 SQL Pushdown 인터페이스 기술
※ Predicate Pushdown ➔ 이종 DBMS에 대한 표준 SQL 지원

02 고신뢰성 저지연 고성능 DBMS 스토리지 서비스 SW 기술
※ RDMA + SPDK ➔ 호스트 CPU 부하 감소, I/O 성능 향상

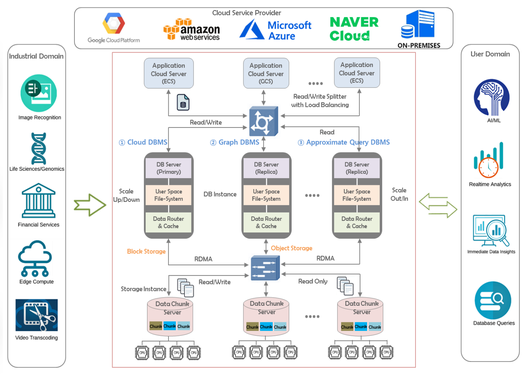
03 클라우드향 DBMS 태스크 연산 스토리지 오프로딩 기술
※ SIMD, FPU 등 가속기 처리➔ 대규모 데이터 처리, 호스트 부하 감소

04 연산 스토리지 오프로딩을 통한 자원 절감 및 분석 모델 기술
※ 쿼리 연산자별 대역폭, 소비전력 평가 ➔ 성능 향상 및 소비전력 절감

05 다수의 연산 스토리지를 지원하는 DBMS 연산 스토리지 엔진 기술
※ 동시 다발 Snippet 스케쥴링 ➔ 연산 스토리지 병렬처리로 성능향상

06 차세대 DBMS 스토리지 SW 통합 관리 플랫폼 化
※ On-Premise & Public Cloud 연산 스토리지 플랫폼 ➔ 사업화 연계

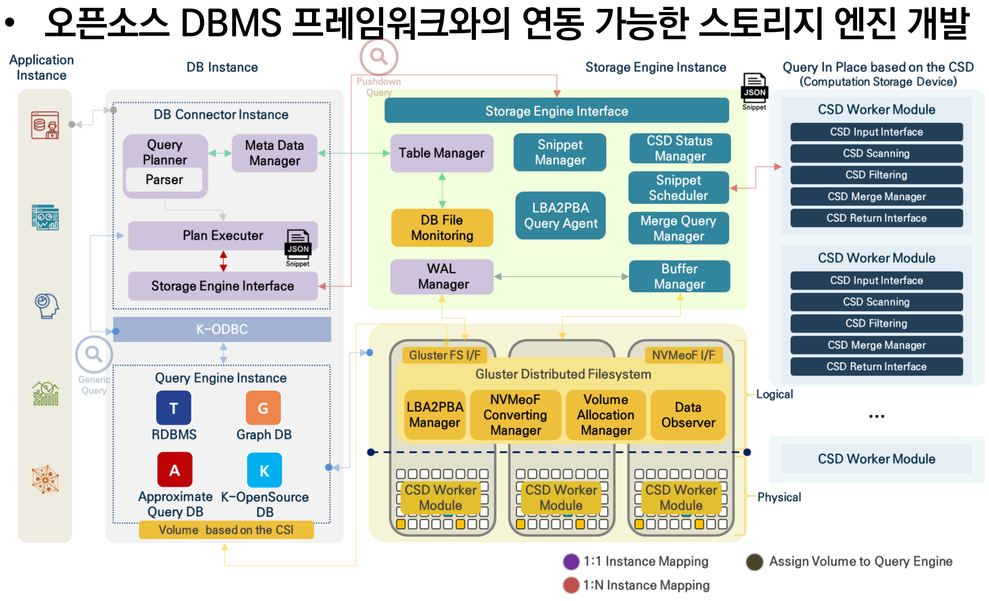
07 차세대 DBMS 연산 스토리지 SW 시험 및 검증, 안정화
※ 1, 2, 3 세부 통합 시험 및 검증 ➔ Reference Site 확보



연구 내용

클라우드 향(向) DBMS 연산 스토리지 엔진 아키텍처

- DBMS와 스토리지 장치가 물리적으로 분리된 경우의 연산 스토리지 처리 기술 개발
- DBMS 태스크 연산 스토리지 오프로드 기반 자원절감 효과 분석 모델 개발
- 다수의 연산 스토리지를 지원하는 DBMS 스토리지 엔진 개발
- 오픈소스 DBMS 프레임워크와의 연동 가능한 스토리지 엔진 개발



〈자체 성능 평가〉

1. 기존 SSD 대비 데이터 처리 속도

➢ TPC-H 쿼리를 기준으로 자체 테스트 진행

```
start TPCH Query With CSD...
Query Done... Time -> 0 hour 1 minute 21.72 second
CSD performed:
    read:                          7504.56 (MB)
        avg:                       91.8 (MB/s)

start TPCH Query WithOut CSD...
Query Done... Time -> 0 hour 2 minute 7.84 second
SSD performed:
    read:                          7504.56 (MB)
        avg:                       58.68 (MB/s)
```

처리 속도: 58.68MB/s –› 91.8MB/S (56.44% 증가)

2. Aggregation Query〈JOIN〉의 성능 향상 비율

➢ TPC-H 쿼리를 기준으로 자체 테스트 진행

```
start TPCH Query With CSD...
current CPU Total Usage -> 1 %
Query Done... Time -> 0 hour 8 minute 39.77 second

start TPCH Query WithOut CSD...
current CPU Total Usage -> 1 %
Query Done... Time -> 0 hour 13 minute 43.71 second
```

성능 향상 비율:
13 min 43.71 sec –› 8 min 39.77 sec (기존 DB 대비158.47% 향상)

3. 기존 빅데이터 시스템 대비 호스트 CPU 절감효과

➢ TPC-H 쿼리를 기준으로 자체 테스트 진행

```
start TPCH Query With CSD...
current CPU Total Usage -> 1 %
Query Done... Time -> 0 hour 8 minute 39.77 second
Average During Query CPU Usage -> 4.92 %
Average During Query Usage Watts -> 71.3 W

start TPCH Query WithOut CSD...
current CPU Total Usage -> 1 %
Query Done... Time -> 0 hour 13 minute 43.71 second
Average During Query CPU Usage -> 5.53 %
Average During Query Usage Watts -> 112.9 W
```

기존 시스템 대비 CPU 사용량: 5.53 % –› 4.92 % (11.04% 감소)

4. 기존 빅데이터 시스템 대비 에너지 절감 비율

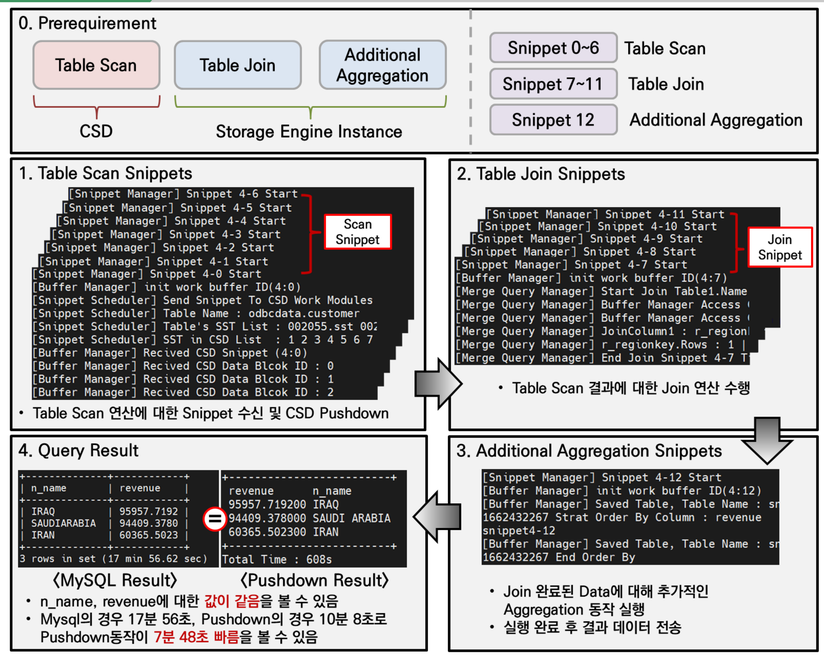
➢ TPC-H 쿼리를 기준으로 자체 테스트 진행

```
start TPCH Query With CSD...
current CPU Total Usage -> 1 %
Query Done... Time -> 0 hour 8 minute 39.77 second
Average During Query CPU Usage -> 4.92 %
Average During Query Usage Watts -> 71.3 W

start TPCH Query WithOut CSD...
current CPU Total Usage -> 1 %
Query Done... Time -> 0 hour 13 minute 43.71 second
Average During Query CPU Usage -> 5.53 %
Average During Query Usage Watts -> 112.9 W
```

에너지 절감 비율 : 112.9 W–› 71.3 W (기존 DB 대비158.34% 향상)

연구 내용



〈오픈소스 DBMS 기반 Pushdown 적용 검증 (TPC-H 쿼리)〉

DBMS와 스토리지 장치가 물리적으로 분리된 경우의 연산 스토리지 처리 기술 개발

- DBMS와 스토리지 장치가 물리적으로 분리되어 기존 결합형 DBMS에 비해 스토리지의 유연한 확장이 가능함
- 확장된 스토리지에 사용자의 쿼리정보가 도달하기 위한 스토리지 상태기반 스케줄링을 통한 쿼리정보 할당

〈스케줄러 성능 검증〉

스토리지 데이터 접근을 위한 논리 ↔ 물리 Block 주소 변환 검증

```
"filename": "000051.sst",
"Chunks": [
    {
        "Offset": 0,
        "Length": 4000
    },
```

논리 ↔ 물리 Block 주소 변환

```
"000051.sst": [
    {
        "SEQ ID": 1,
        "CSD ID": 5,
        "Offset": 143708393472,
        "Length": 4000
    },
```

신속한 데이터 처리를 위한 Best CSD 선정 검증

```
CSD Primary CSD Replica
    2           5
    3           5
    1           4
CSD Map :
CSD Name CSD IP
    4      10.0.5.120+10.1.1.2
    2      10.0.5.119+10.1.2.2
    5      10.0.5.119+10.1.3.2
    3      10.0.5.120+10.1.2.2
    1      10.0.5.119+10.1.1.2
```

연산 스토리지 상태 정보 조회

```
CSD Working Block Map :
CSD 4 CSD 2 CSD 5 CSD 3 CSD 1
  0     0     0     0     0
```

연산 스토리지 Working Status 조회

```
Primary CSD Working Block Num : 0
Replica CSD Working Block Num : 0
--> Best CSD : CSD 5
```

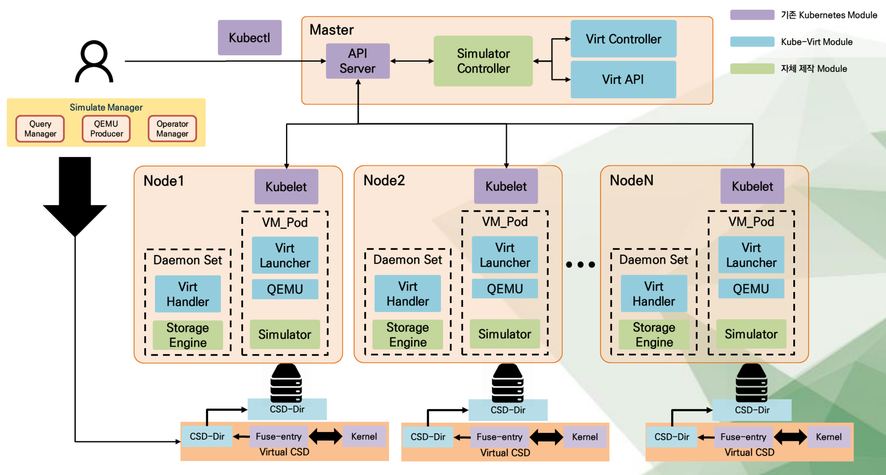
Best CSD 선정

연산 스토리지로의 Snippet Push Down 검증

```
******************Snippet JSON****************
{
    "WorkID": 1,
    "CSD Name" : "2",
    "table_name": "small_line.lineitem111_Table1",
    "table_col" : ["small_line.lineitem111_Table1.l_orderkey",
    ...
    "table_offset" : [2, 6, 10, 14, 18, 25, 32, 39, 46, 47, 48, 51, 54, 57, 82, 92],
    "table_offlen" : [4, 4, 4, 4, 7, 7, 7, 7, 1, 1, 3, 3, 3, 25, 10, 44],
    "table_datatype" : [3, 3, 3, 3, 246, 246, 246, 246, 254, 254, 14, 14, 14, 254, 254, 15],
    ...
```

스토리지 내부에서 Snippet 수신 Log

DBMS 태스크 연산 스토리지 오프로드 기반 자원 절감 효과 분석 모델 개발



차세대 DBMS 연산 스토리지 엔진 기술 개발

4세부

KETI 한국전자기술연구원

연구 내용

- 자원 절감 효과 분석 모듈을 통해 연산 스토리지 기반 쿼리 별 CPU 절감 비율, 연산 스토리지 장치의 대역폭 절감 비율, 에너지 효율성에 대한 검증이 가능
- 시뮬레이터는 기존 결합형 DBMS를 시뮬레이션한 SSD Simulator와 연산 스토리지기반 분리형 DBMS를 시뮬레이션 한 CSD Simulator가 존재
- 각 Simulator를 실행하여 쿼리 별 CPU 사용량, 스토리지 대역폭, 에너지 사용량을 측정하여 검증 진행

〈분석 모델 자원 사용률 측정 검증〉

CSD Simulator 쿼리 종료 및 Metric 측정 검증

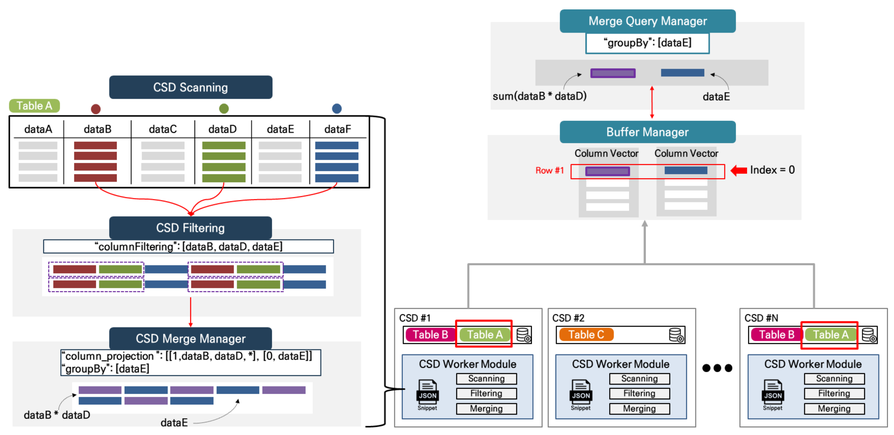
```
start TPCH Query WithOut CSD...
current CPU Total Usage -> 3 %
current MEM Total Usage -> 2 G

Query Done... Time -> 0 hour 15 minute 6.222 second
Average During Query CPU Usage -> 73 %
Average During Query MEM Usage -> 30 G
```

SSD Simulator 쿼리 종료 및 Metric 측정 검증

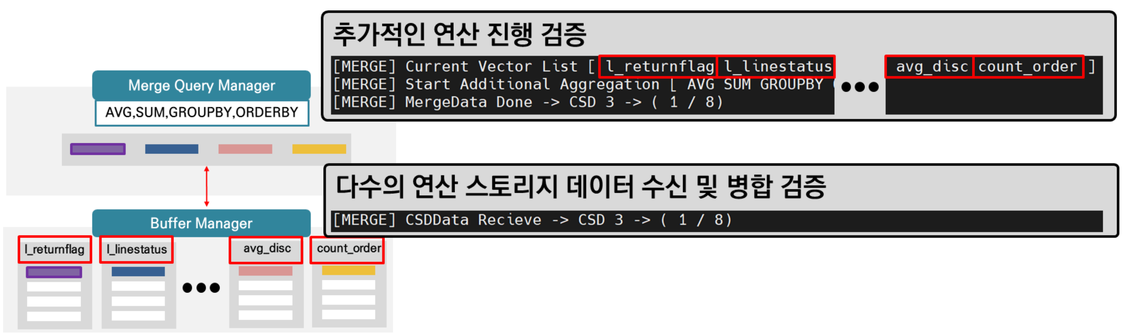
```
Query Done... Time -> 0 hour 14 minute 34.532 second
Average During Query CPU Usage -> 60 %
Average During Query MEM Usage -> 17 G
```

다수의 연산 스토리지를 지원하는 DBMS 스토리지 엔진 개발



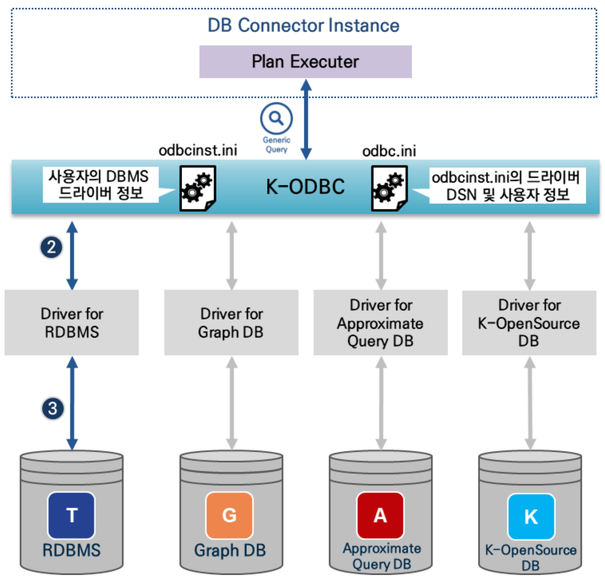
- DBMS와 스토리지 장치의 분리로 인한 다수의 연산 스토리지의 결과를 수집/병합 및 추가적인 연산을 진행 하여 쿼리에 대한 결과 제작이 가능한 스토리지 엔진 개발
- 쿼리 결과에 대한 Column 정보를 미리 Column Vector 형태로 만든 후 각 스토리지의 결과를 해당 Column Vector에 저장
- 저장 완료된 Column Vector로 부터 추가적인 연산 진행
- 각각의 연산 스토리지 내에서는 전체 테이블 데이터 접근이 불가능 하기 때문에 추가적인 연산이 필요하며 추가연산에는 아래와 같은 것들이 있음
  - ✓ AVG
  - ✓ SUM
  - ✓ GROUPBY
  - ✓ ORDERBY
  - ✓ SUBQUERY
  - ✓ JOIN

〈다수 연산 스토리지에 대한 병합 및 추가 연산 검증〉



연구 내용

오픈소스 DBMS 프레임 워크 연동을 위한 스토리지 엔진 구조 개발



- Storage Engine Instance로 Push Down 되지 않는 쿼리들은 사용자가 지정한 DBMS Driver를 통해 해당 DBMS에서 쿼리를 수행
- 이종의 DBMS로의 유연한 연결을 위해, K-ODBC를 개발
- Push Down되지 않는 쿼리에는 대표적으로 아래와 같은 DDL 문이 있음
  - ✓ UPDATE
  - ✓ DELETE
  - ✓ INSERT

〈K-ODBC 쿼리 수행 검증 및 연결 가능 DBMS 확인〉

Create Table 문에 대한 K-ODBC 실행 검증(lineitem Table 생성)

```
DB Connected!
Using DBMS :
mysql  Ver 14.14 Distrib 5.6.35, for Linux
DSN : myodbc5w
User ID : root
Using DataBase : tpch
SQLRowCount returns 0
```

```
+-----------------+
| Tables_in_tpch  |
+-----------------+
| customer        |
| lineitem        |
| nation          |
| orders          |
| part            |
```

Table 생성 확인

K-ODBC 연결 Driver 확인

```
├── mariadb
│   ├── odbc.ini
│   └── odbcinst.ini
├── mysql
│   ├── odbc.ini
│   └── odbcinst.ini
└── postgresql
    ├── odbc.ini
    └── odbcinst.ini
```

3종 DB 연결 가능

향후 계획

- 연산 스토리지 – Storage Engine Instance 간 전송 데이터 압축 정책 개발
- Column Vector 병합의 병목현상을 방지하기 위한 연산 스토리지 – 지원 모듈 간 RDMA 통신 인스턴스 개발
- Storage Engine Instance의 클라우드 화 진행
- 시뮬레이터 및 전체 시스템의 측정 Metric 통계를 위한 Metric UI/UX 지원 모듈 개발