CQuery : 우당탕탕 Trino와 썸타기

유재호
NAVER

NAVER DEVIEW 2023

CONTENTS

1. AIDA CQuery 입니다. (Who am I)

- AIDA CQuery
- AIDA CQuery History & Now

1.1 AIDA CQuery

NAVER
DEVIEW
2023

AIDA (Advanced Interface for Data and AI)

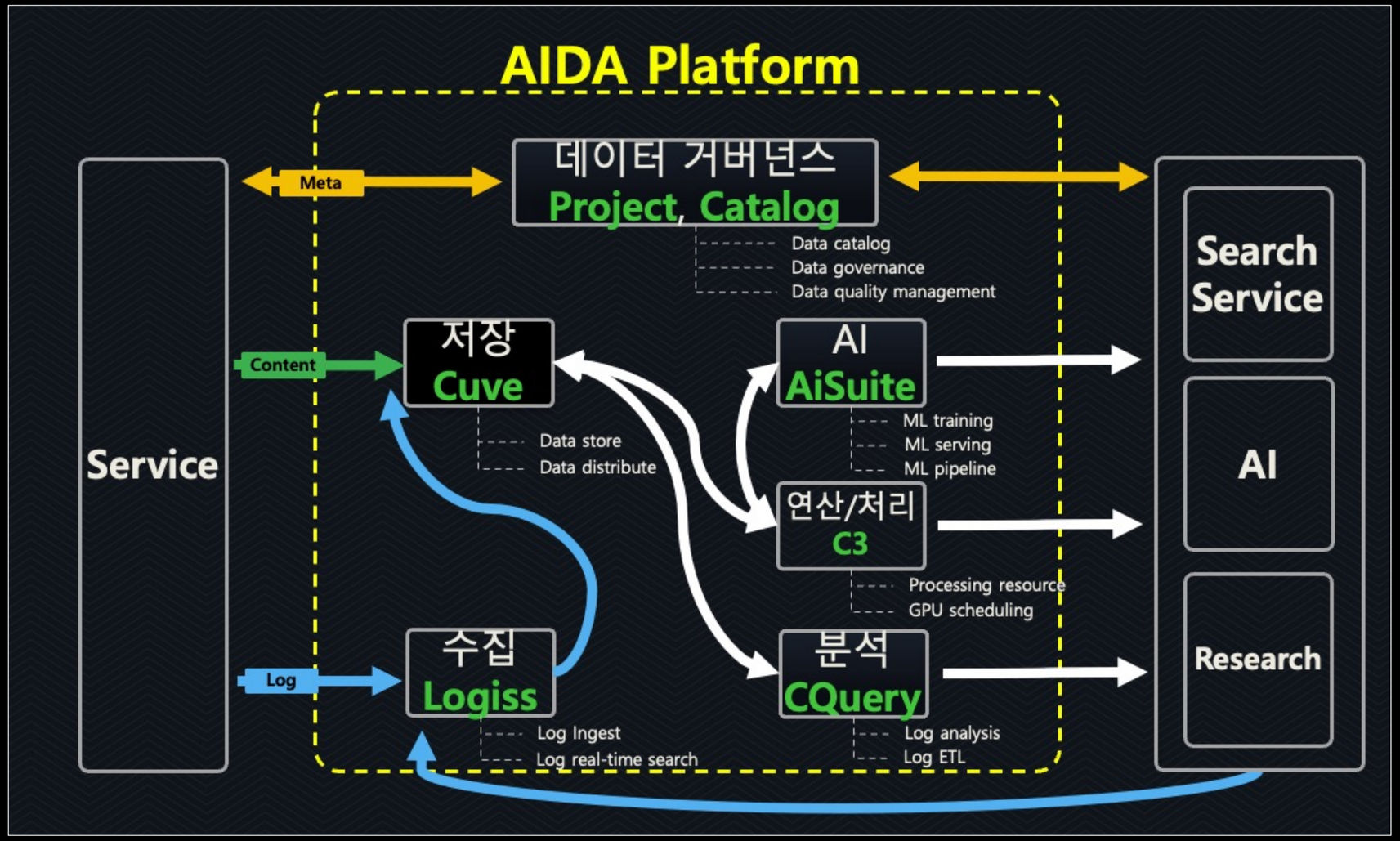


그림1 - https://naver-career.gitbook.io/kr/service/search/ai-and-data-platform

1.1 AIDA CQuery

NAVER
DEVIEW
2023

AIDA (Advanced Interface for Data and AI)

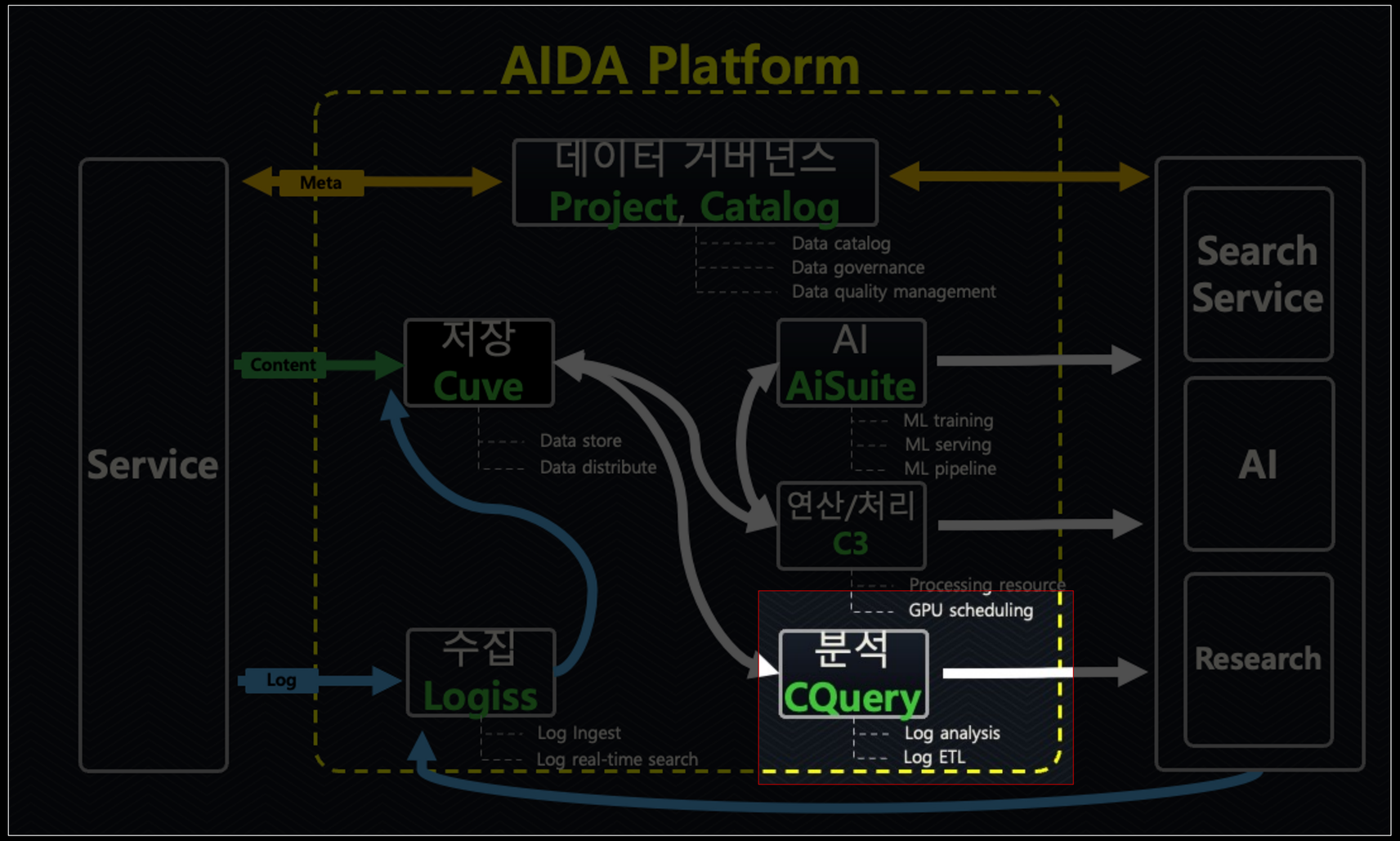


그림1 - https://naver-career.gitbook.io/kr/service/search/ai-and-data-platform

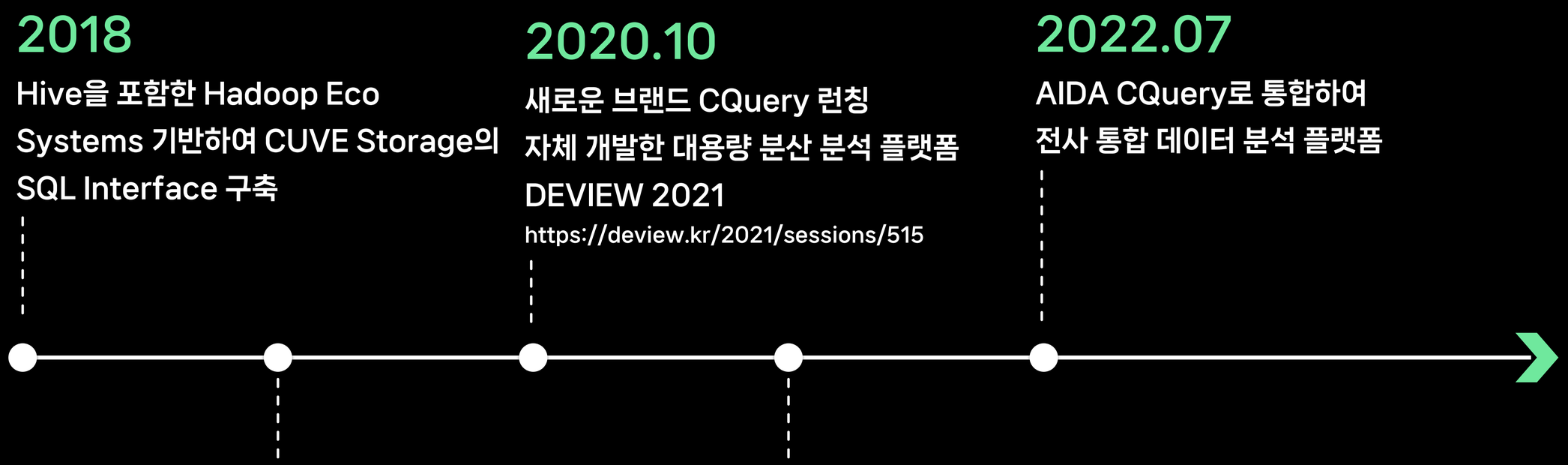
1.1 AIDA CQuery

NAVER
DEVIEW
2023

네이버 서비스에서 발생하는 PB 단위 대용량의 로그를
단 몇 분 안에 SQL로 조회 및 분석하는 환경을 제공하는
멀티테넌트 플랫폼 통합 분석 솔루션

1.2 AIDA CQuery History

NAVER
DEVIEW
2023

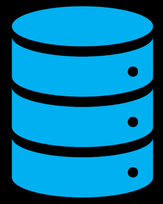


1.3 AIDA CQuery Now

NAVER DEVIEW 2023

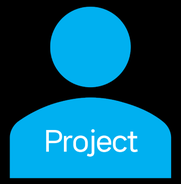
NAVER

- Avg : 8,000 / day
- Peak : 30,000 / day



- Input : 1.x TB / day
- Total Size : ~ 2.x PB

Business Analytics Warehouse



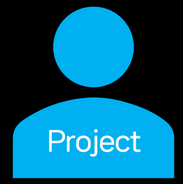
- Avg[1] : 130 / day

LINE

- Avg : 10,000 / day
- Peak : 14,000 / day

- Input : 1.x TB / day
- Total Size : ~ 1.x PB

Business Analytics Warehouse



- Avg : 50 / day

[1] 평균 이용하는 프로젝트 계정 수

2. 한 눈에 반하다. (Architecture)

- Hive (Tez) vs Trino
- Trino Architecture 및 동작 방식

2.1 Hive (Tez) vs Trino

NAVER DEVIEW 2023

2.3.1 Hive+Tez vs Trino 성능 비교

DEVIEW 2021

비교 대상 1 : 모바일 통합검색 클릭 로그

```
SELECT count(*)
FROM korea_naver.mobile_click_log
WHERE log_date = '2021-06-27'
```

기간	데이터 크기	Hive+Tez	Trino	소요 시간 비교
하루	수십 GB	1분(60초)	2초	98% 감소
일주일	수백 GB	2분(120초)	10초	91% 감소
한달	수 TB	6분(360초)	43초	88% 감소
6개월	수십 TB	23분(1380초)	2분(120초)	91% 감소
1년	수십 TB	104분(6240초)	4분(240초)	96% 감소

2.3.1 Hive+Tez vs Trino 성능 비교

DEVIEW 2021

비교 대상 2 : CQuery 사용자들이 가장 많이 쓰는 SQL Template

다수의 Inner Select, FULL OUTER JOIN, ORDER BY로 조합된 SQL

기간	데이터 크기	Hive+Tez	Trino	소요 시간 비교
하루	수 GB	1분(60초)	6초	90% 감소
일주일	수십 GB	5분(300초)	10초	96% 감소
한달	수백 GB	16분(960초)	18초	98% 감소
6개월	수 TB	75분(4500초)	1분(60초)	98% 감소
1년	수 TB	173분(10380초)	2분(120초)	98.8% 감소

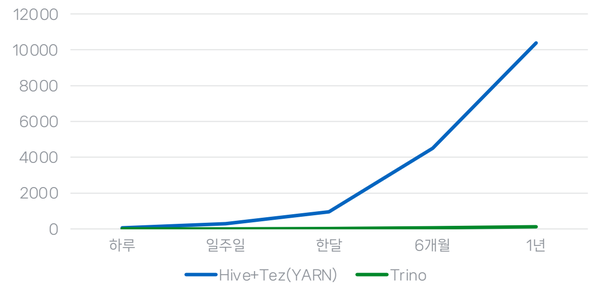
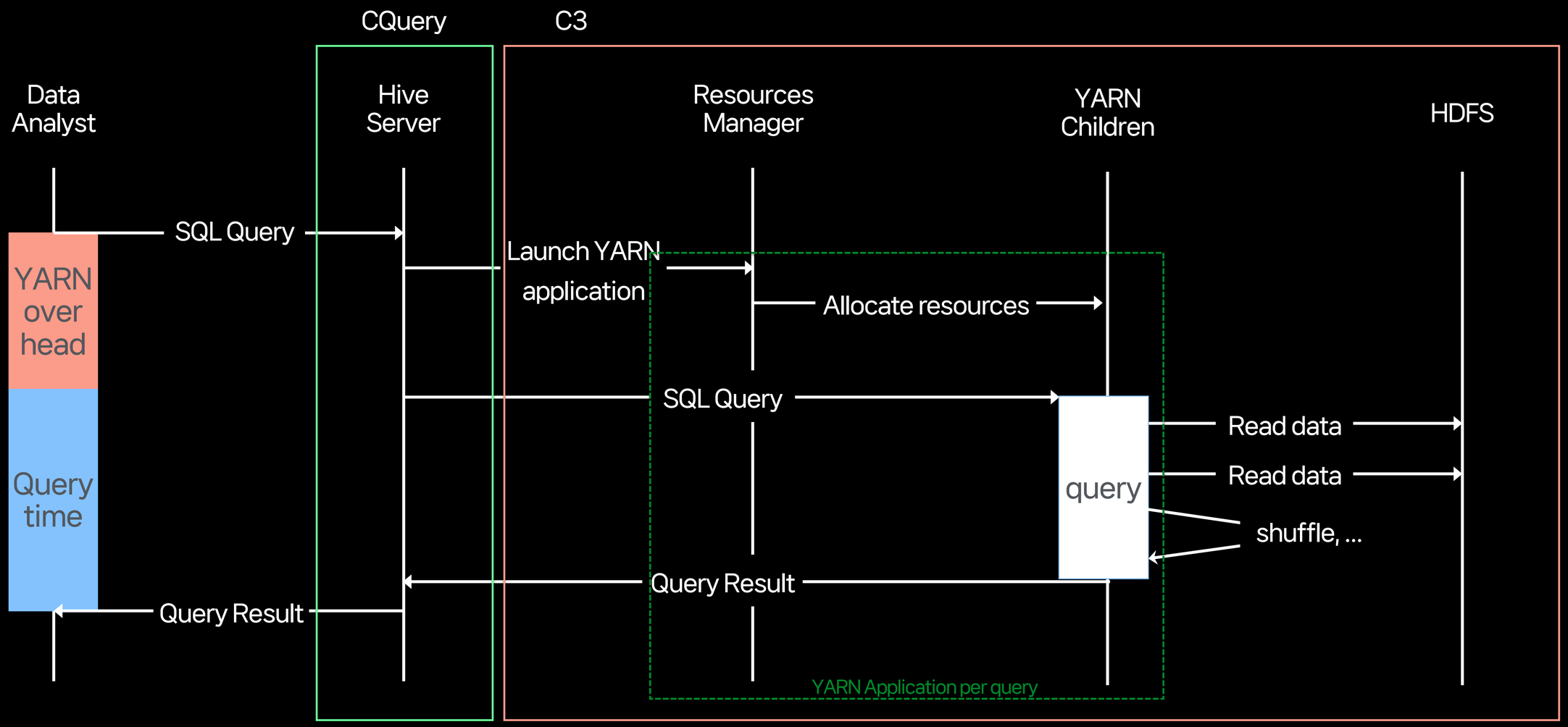


그림2 - https://deview.kr/2021/sessions/515

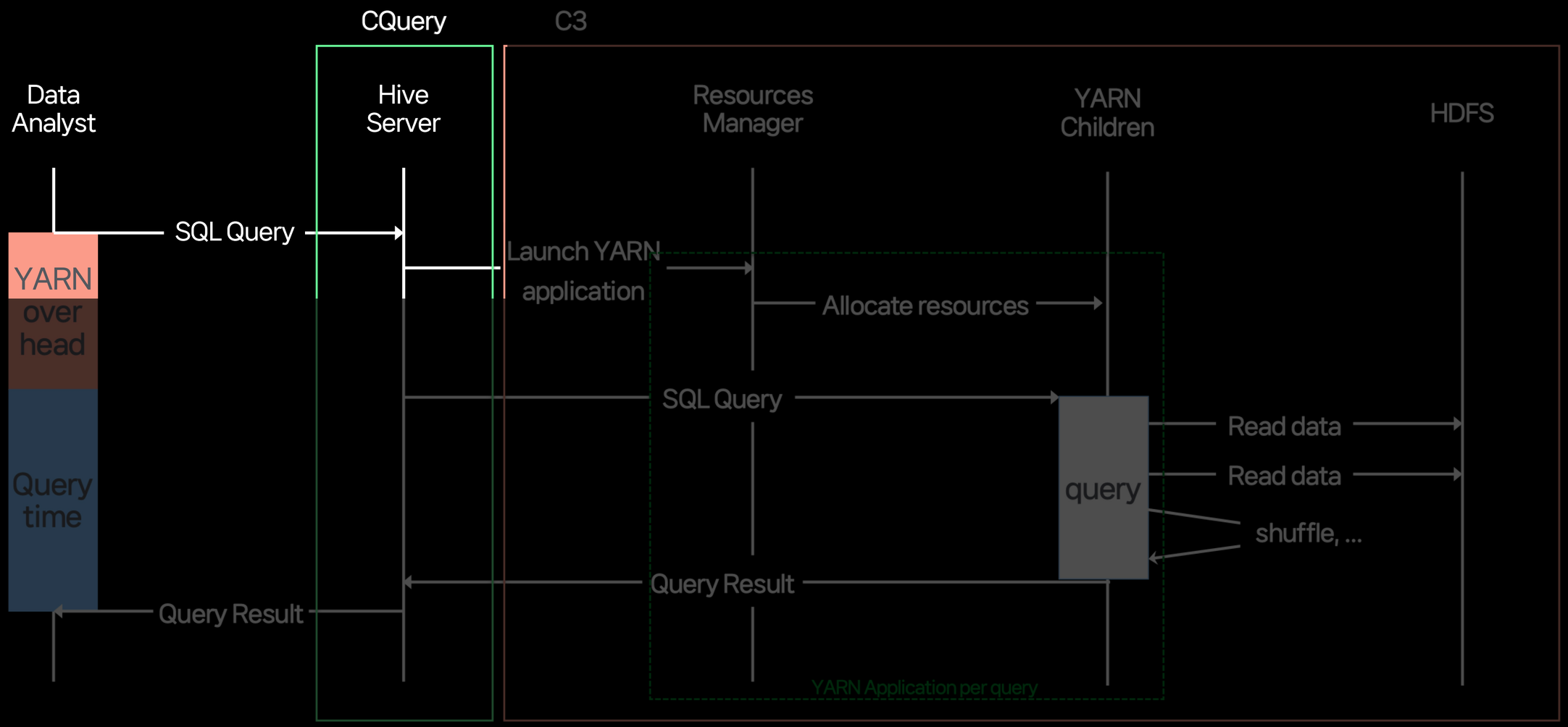
2.1 Hive (Tez) vs Trino

NAVER DEVIEW 2023



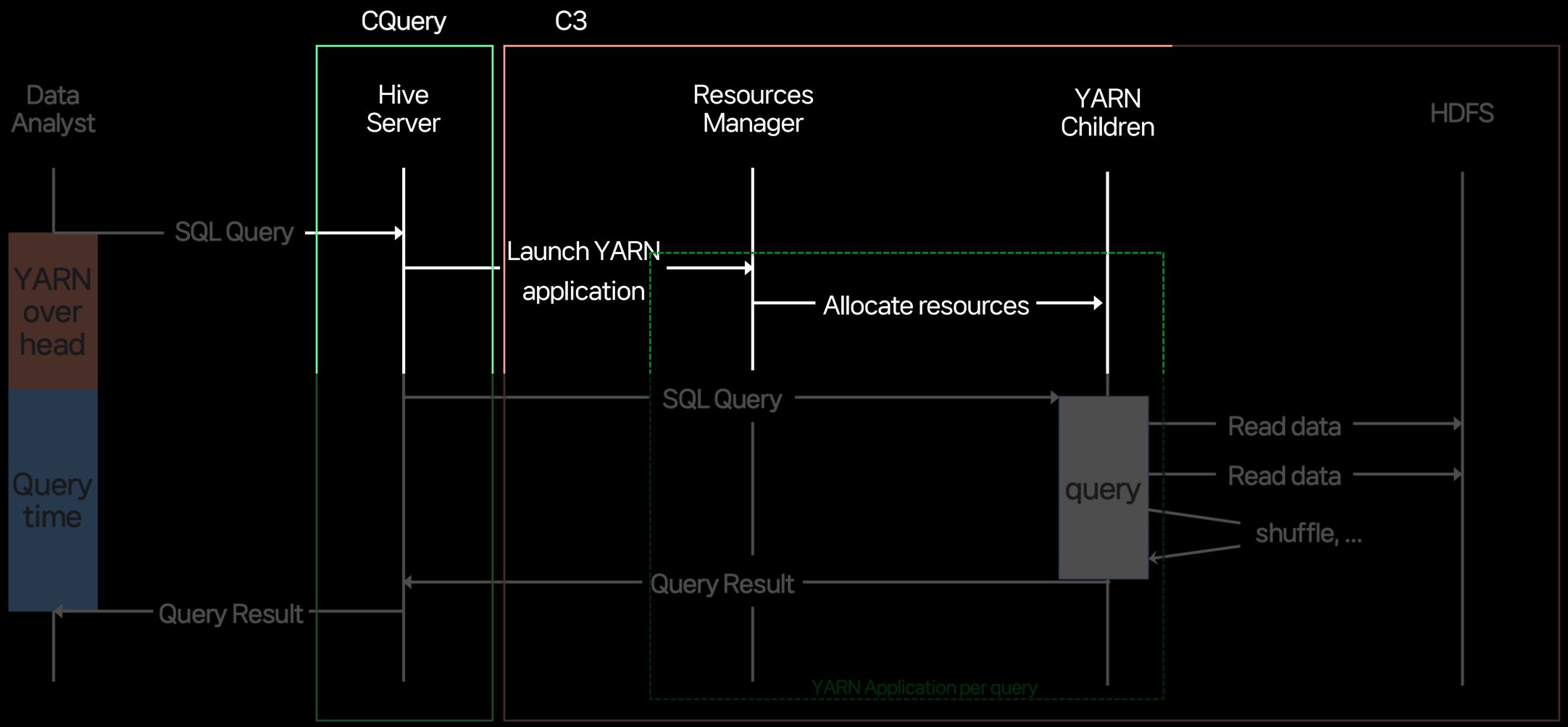
2.1 Hive (Tez) vs Trino

NAVER
DEVIEW
2023



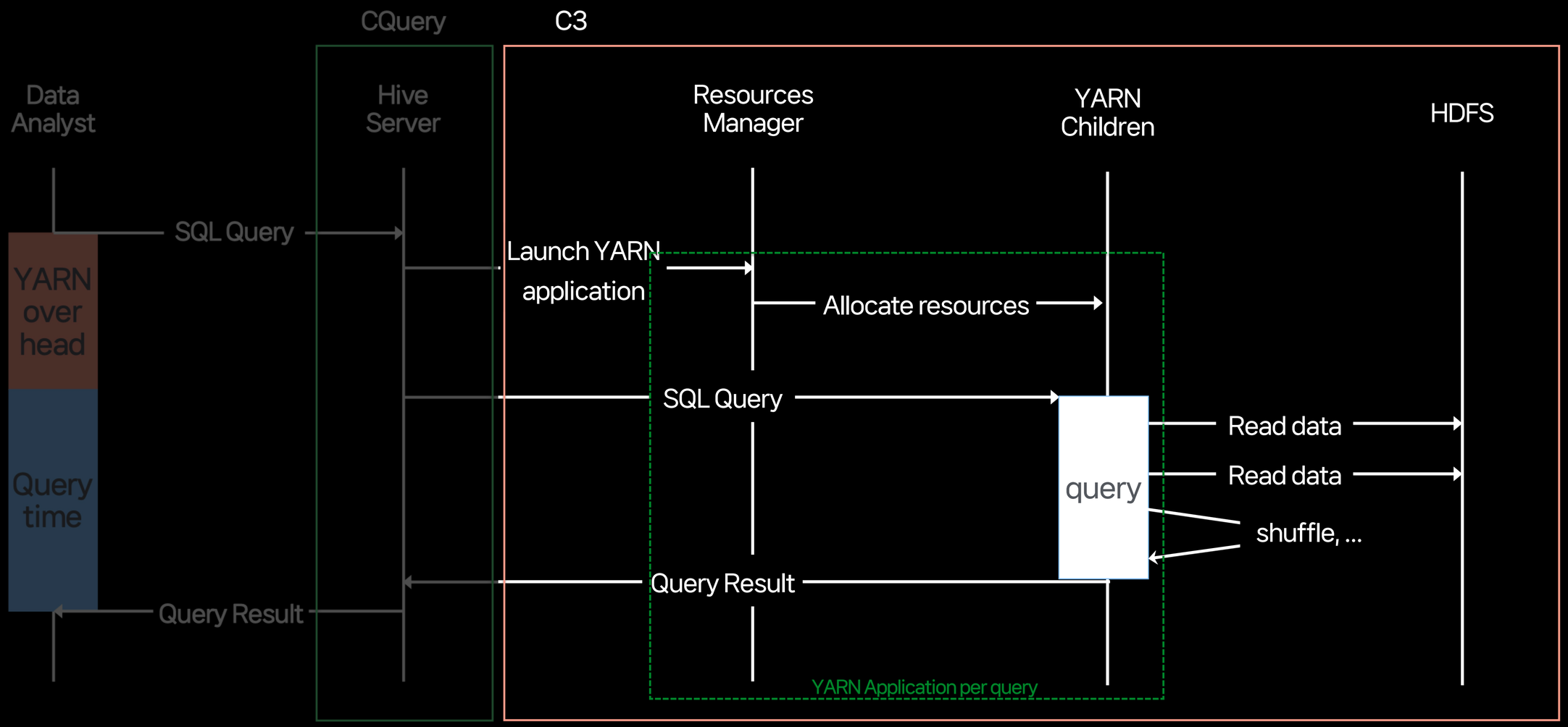
2.1 Hive (Tez) vs Trino

NAVER DEVIEW 2023



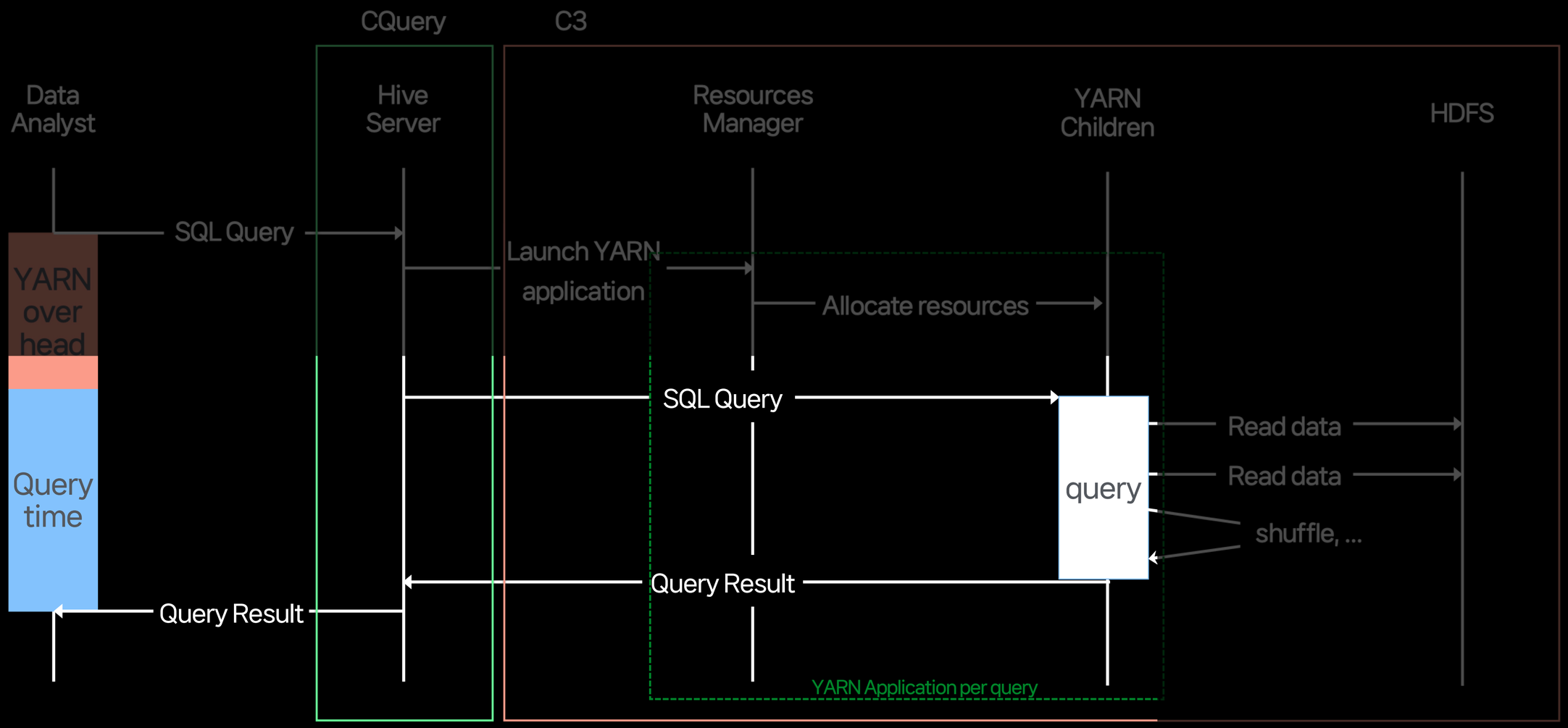
2.1 Hive (Tez) vs Trino

NAVER DEVIEW 2023



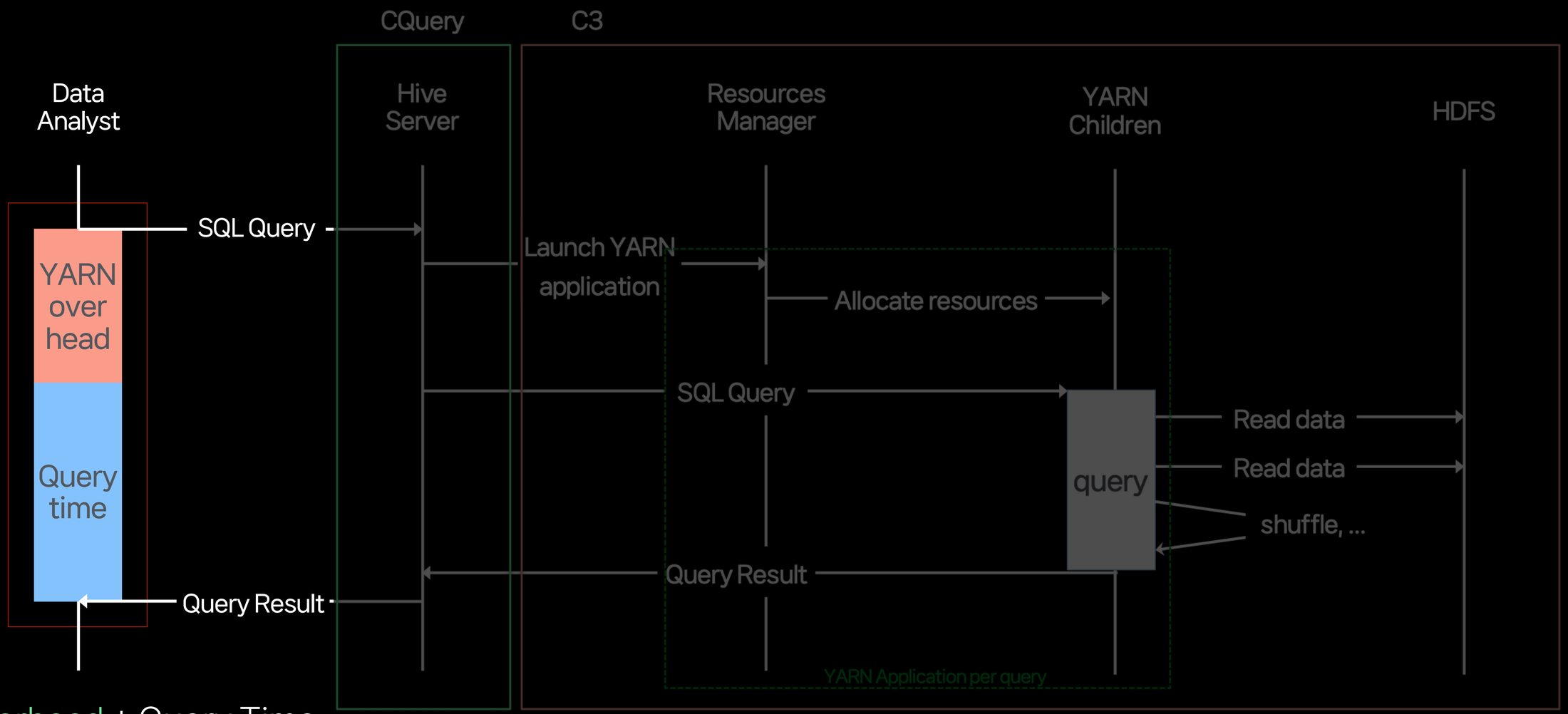
2.1 Hive (Tez) vs Trino

NAVER
DEVIEW
2023



2.1 Hive (Tez) vs Trino

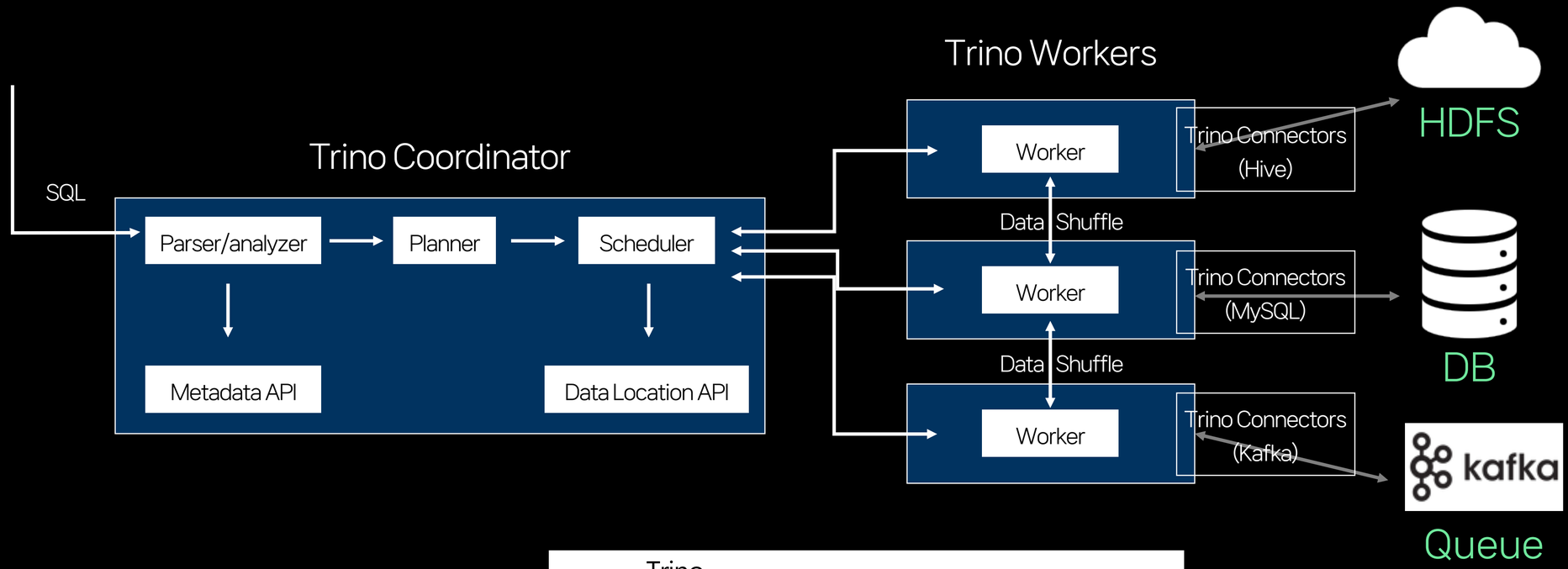
NAVER DEVIEW 2023

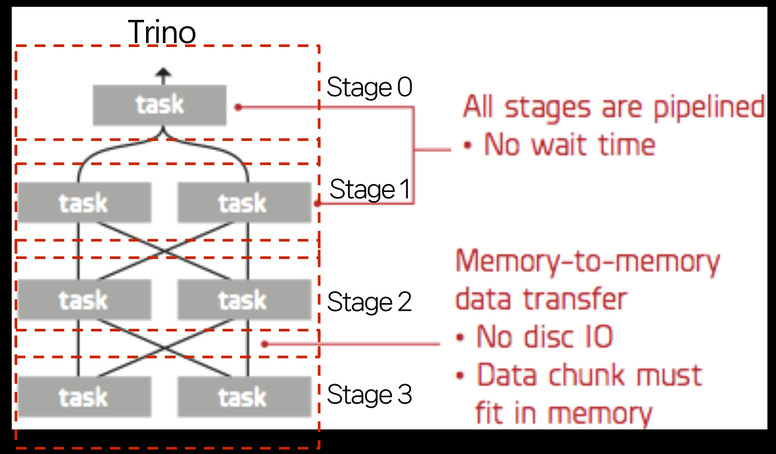


Yarn Overhead + Query Time

2.2 Trino Architecture 및 동작 방식

NAVER DEVIEW 2023



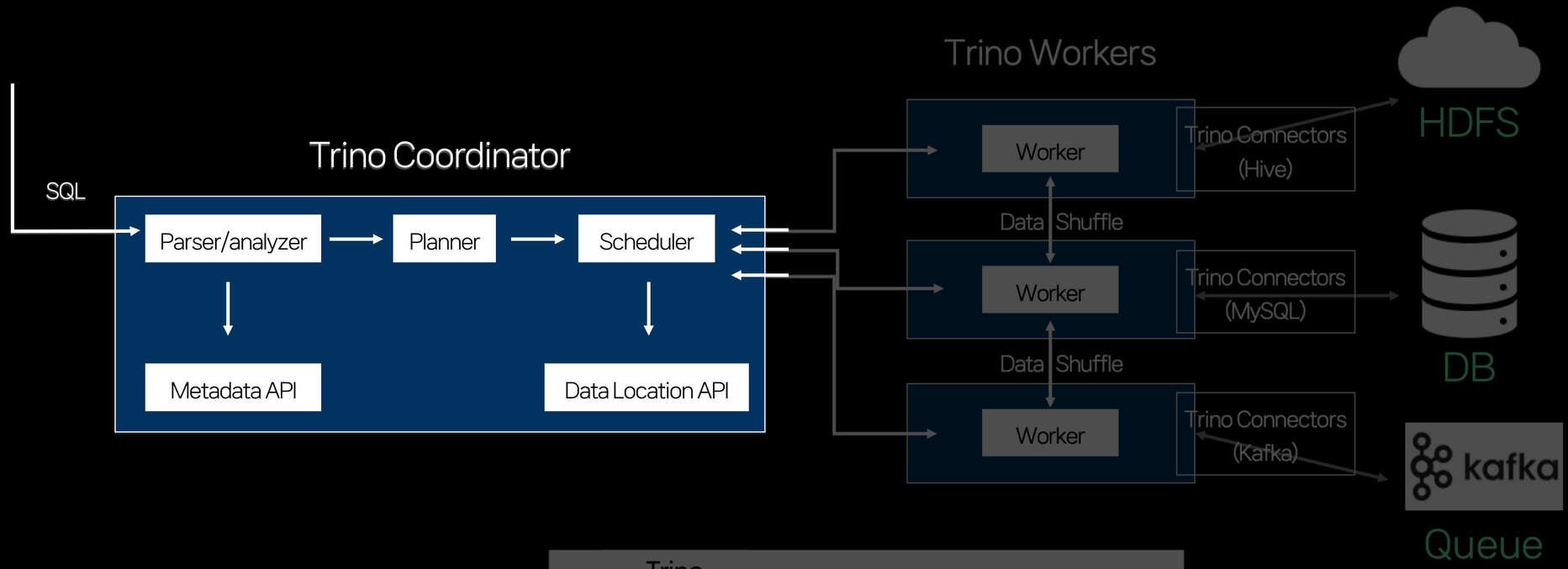


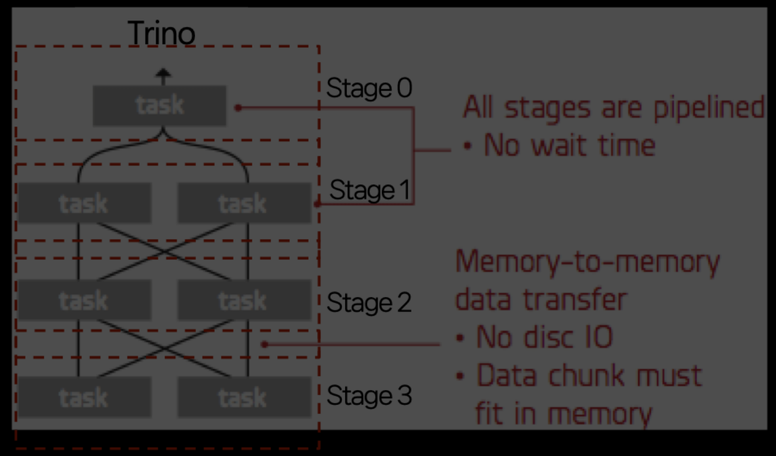
~~Yarn Overhead~~ + Query Time

그림3 - https://blog.treasuredata.com/blog/2015/03/20/presto-versus-hive/

2.2 Trino Architecture 및 동작 방식

NAVER
DEVIEW
2023



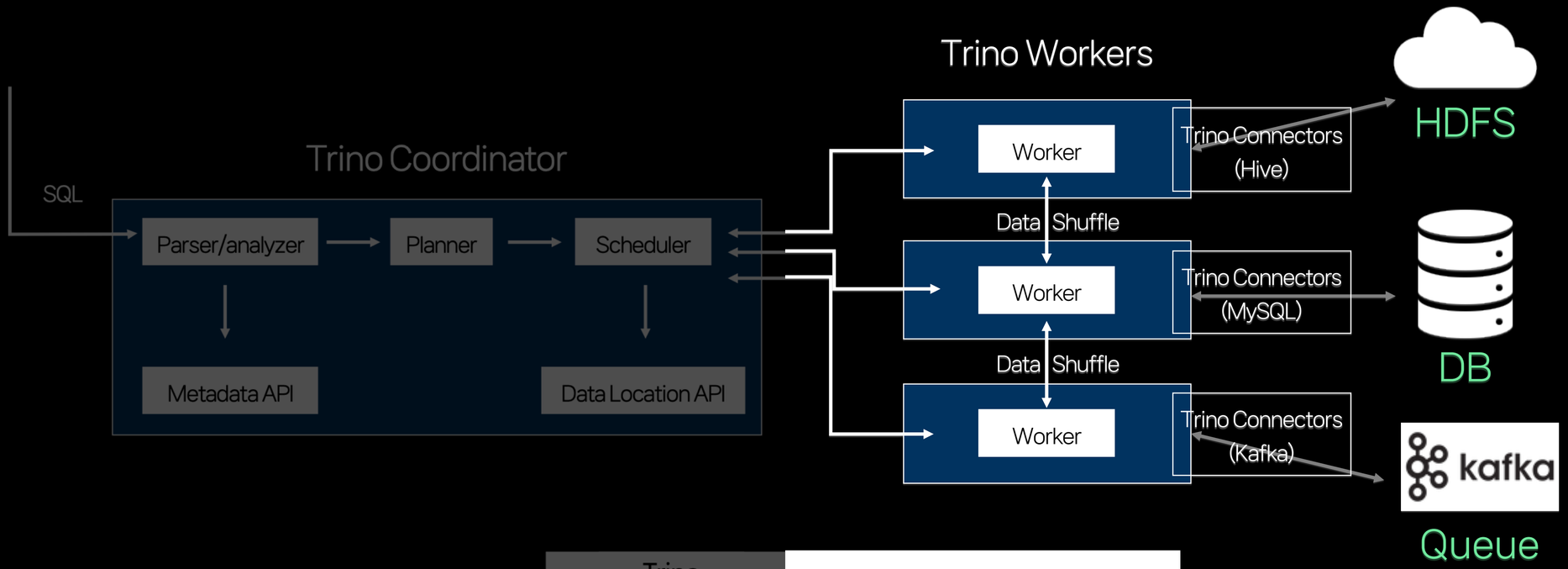


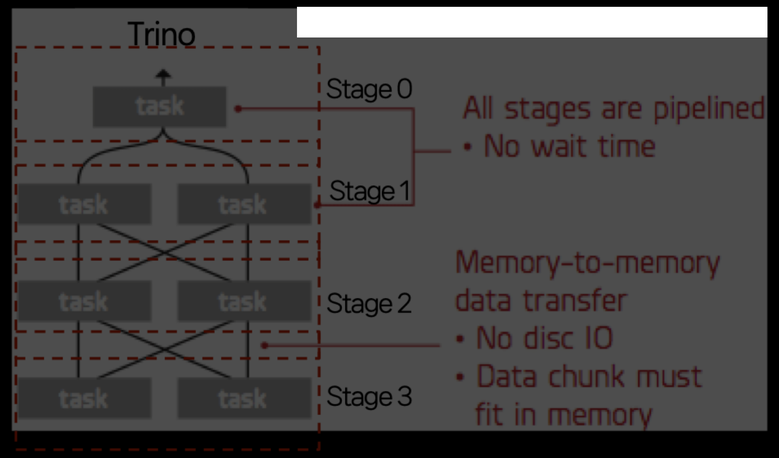
~~Yarn Overhead~~ + Query Time

그림3 - https://blog.treasuredata.com/blog/2015/03/20/presto-versus-hive/

2.2 Trino Architecture 및 동작 방식

NAVER
DEVIEW
2023



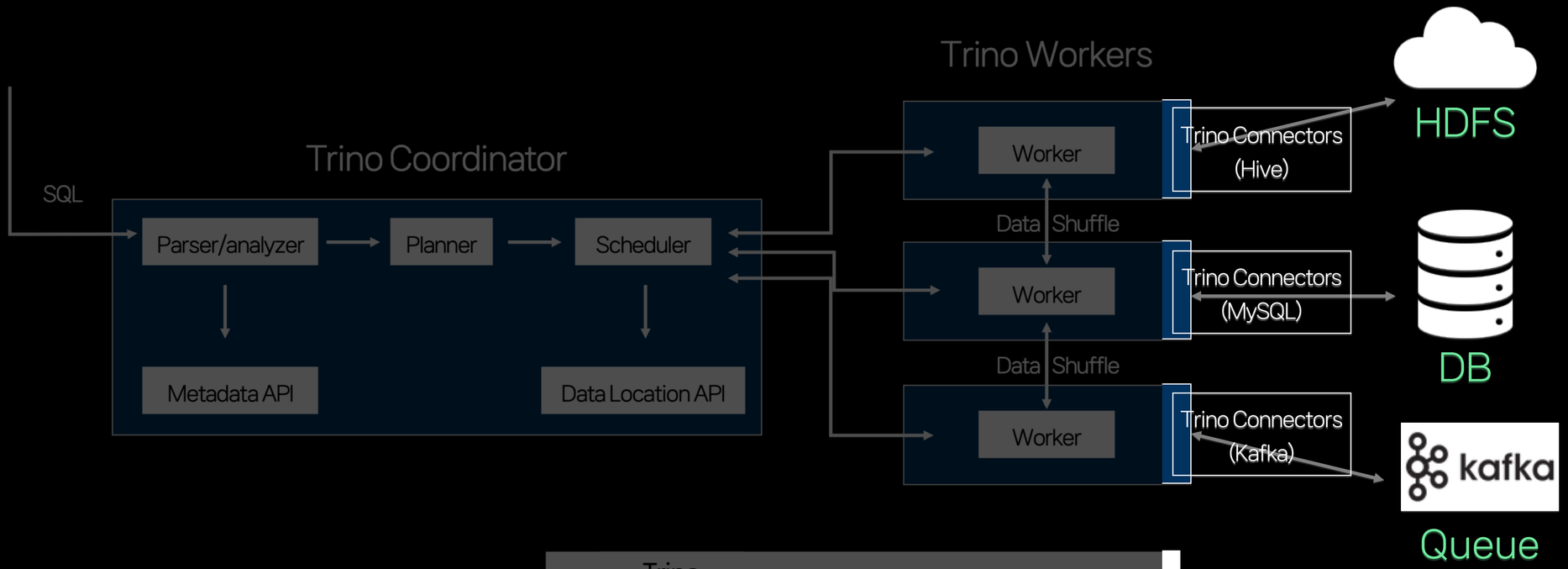


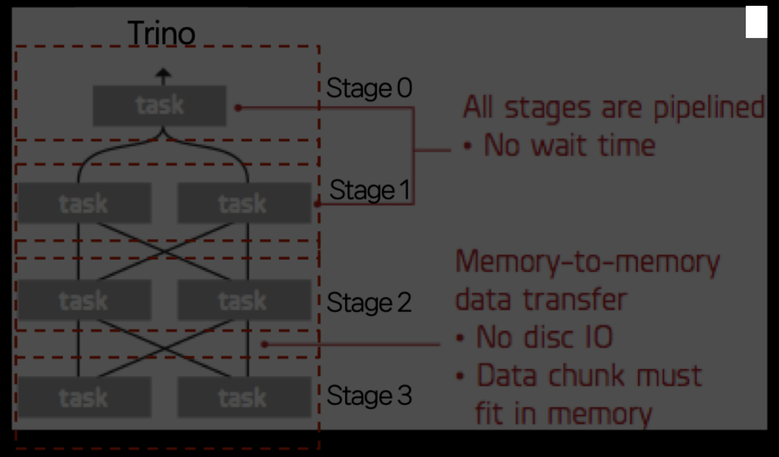
~~Yarn Overhead~~ + Query Time

그림3 - https://blog.treasuredata.com/blog/2015/03/20/presto-versus-hive/

2.2 Trino Architecture 및 동작 방식

NAVER
DEVIEW
2023



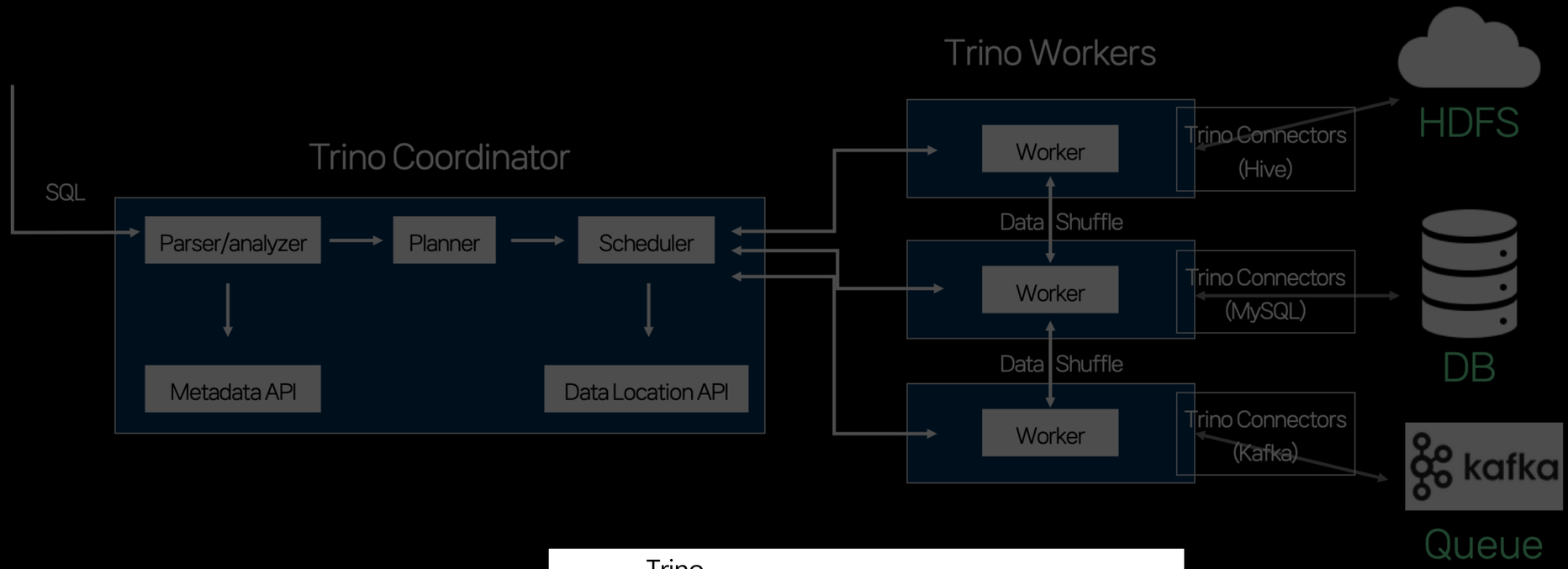


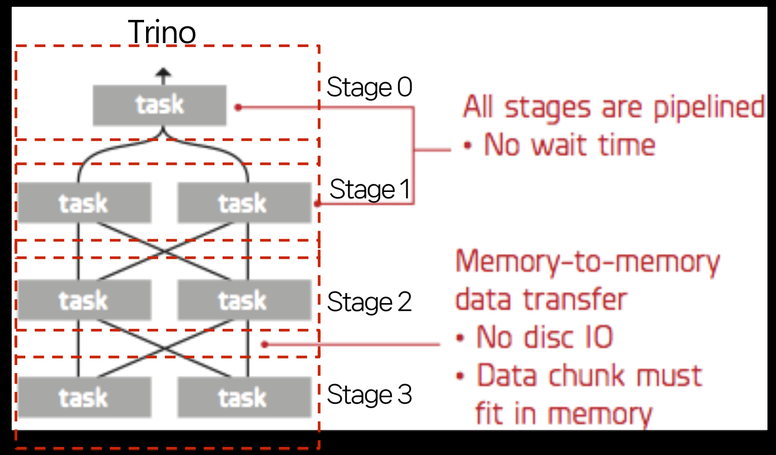
~~Yarn Overhead~~ + Query Time

그림3 - https://blog.treasuredata.com/blog/2015/03/20/presto-versus-hive/

2.2 Trino Architecture 및 동작 방식

NAVER
DEVIEW
2023





~~Yarn Overhead~~ + Query Time

그림3 - https://blog.treasuredata.com/blog/2015/03/20/presto-versus-hive/

3. 알아가는 단계입니다. (Features)

- Memory
- Data Structure
- Web UI

3.1 메모리

NAVER
DEVIEW
2023

- 커널, 디스크/네트워크 버퍼 등 20%
- Thread stacks, GC, off heap 메모리 등 30%
- 쿼리 실제 처리 가용 메모리는 서버의 56%
  - 100 GB 서버 → 80 GB JVM (-20%)
  - 80 GB JVM → 56 GB 쿼리 메모리 (-30%)
  - 총 44 GB (-44%) 손실
- 메모리 부족시 OOM, 서버 무응답 등 장애 발생

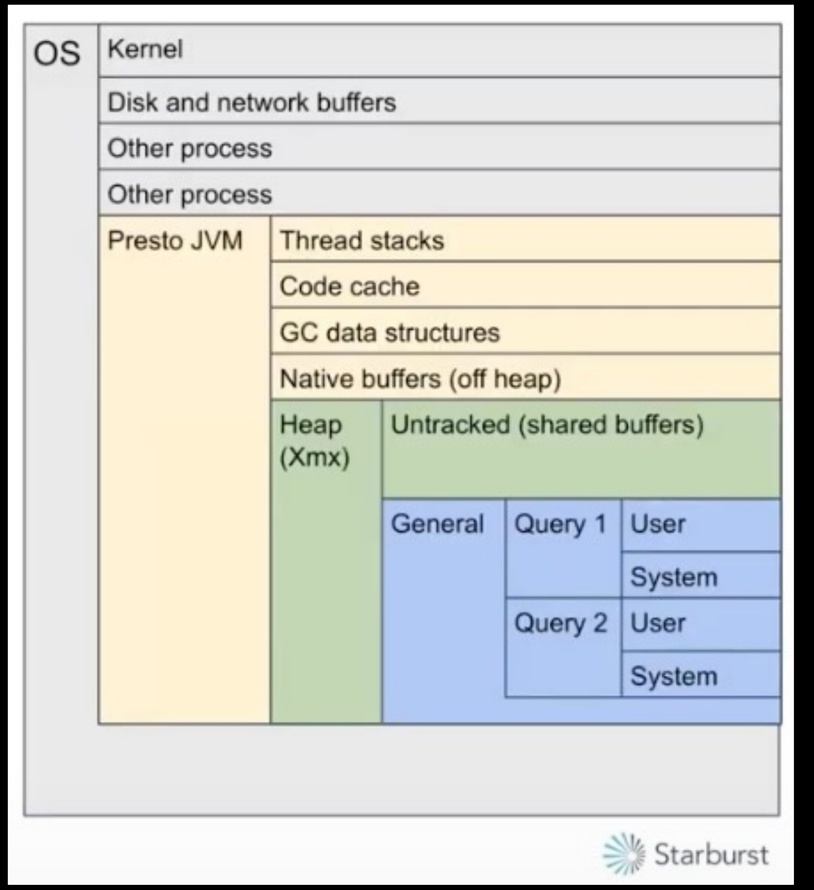
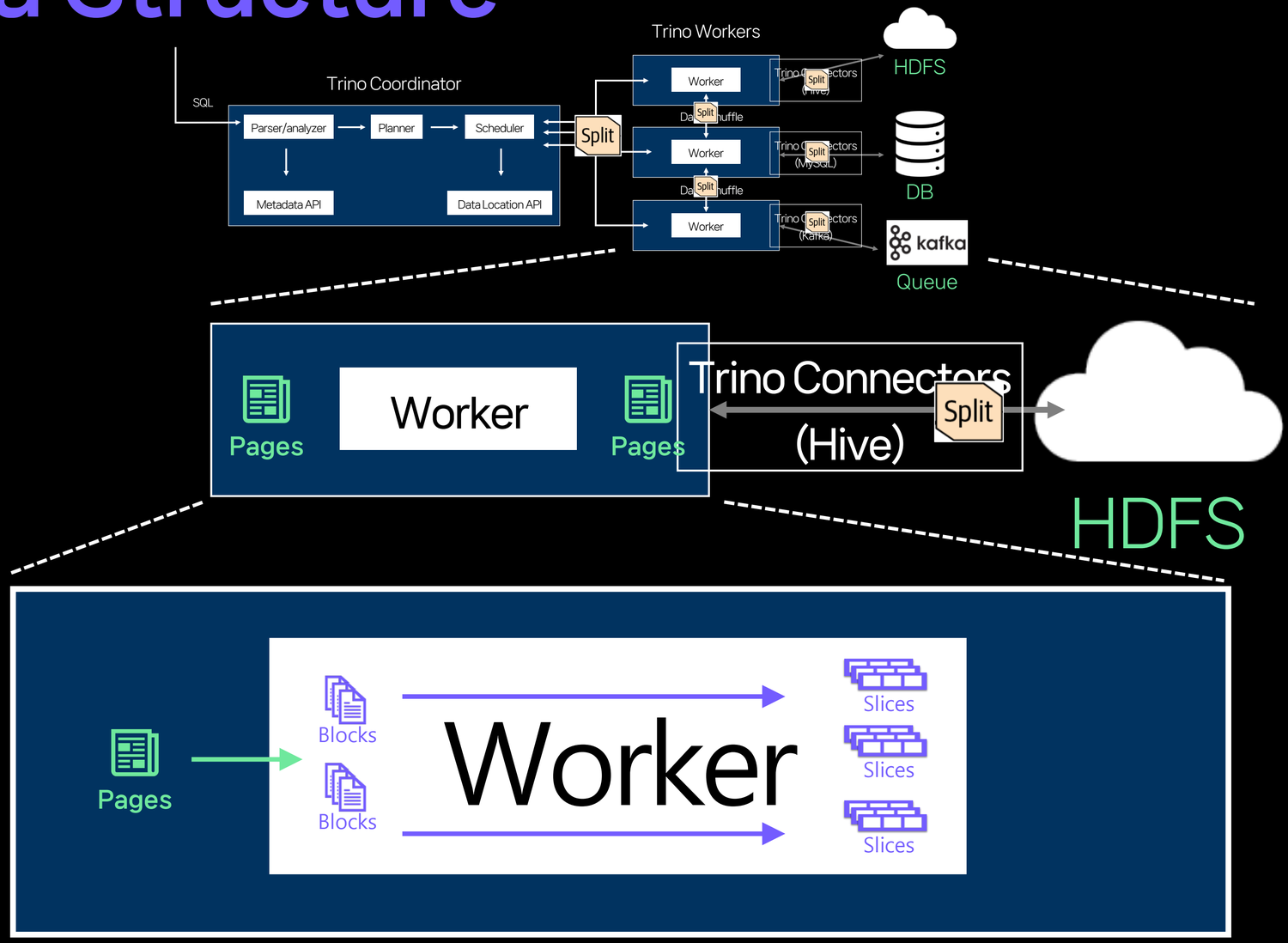


그림4 - https://www.youtube.com/watch?v=Pu80FkBRP-k&t=2086s

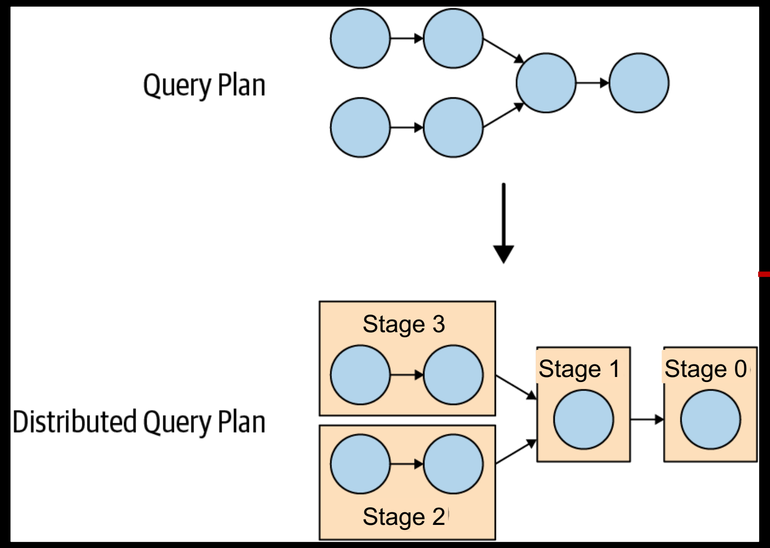
3.2 Data Structure



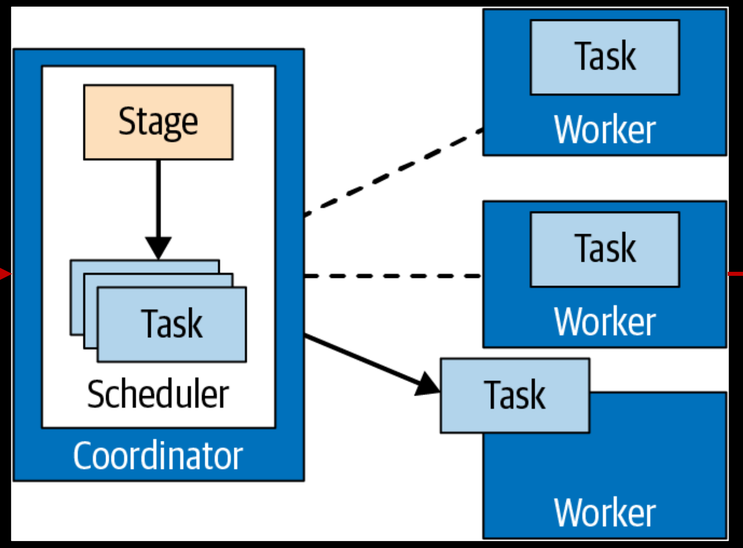
Slices ⊂ Blocks ⊂ Pages ⊂ Split

3.2.1 Data Structure – Splits

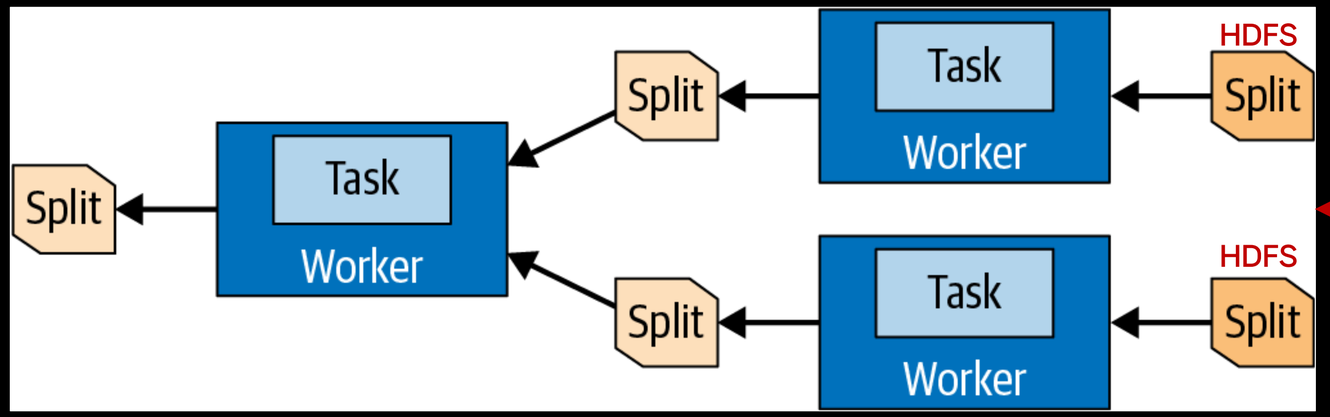
NAVER DEVIEW 2023



Transformation of the query plan to a distributed query plan[1]



Task management performed by the coordinator



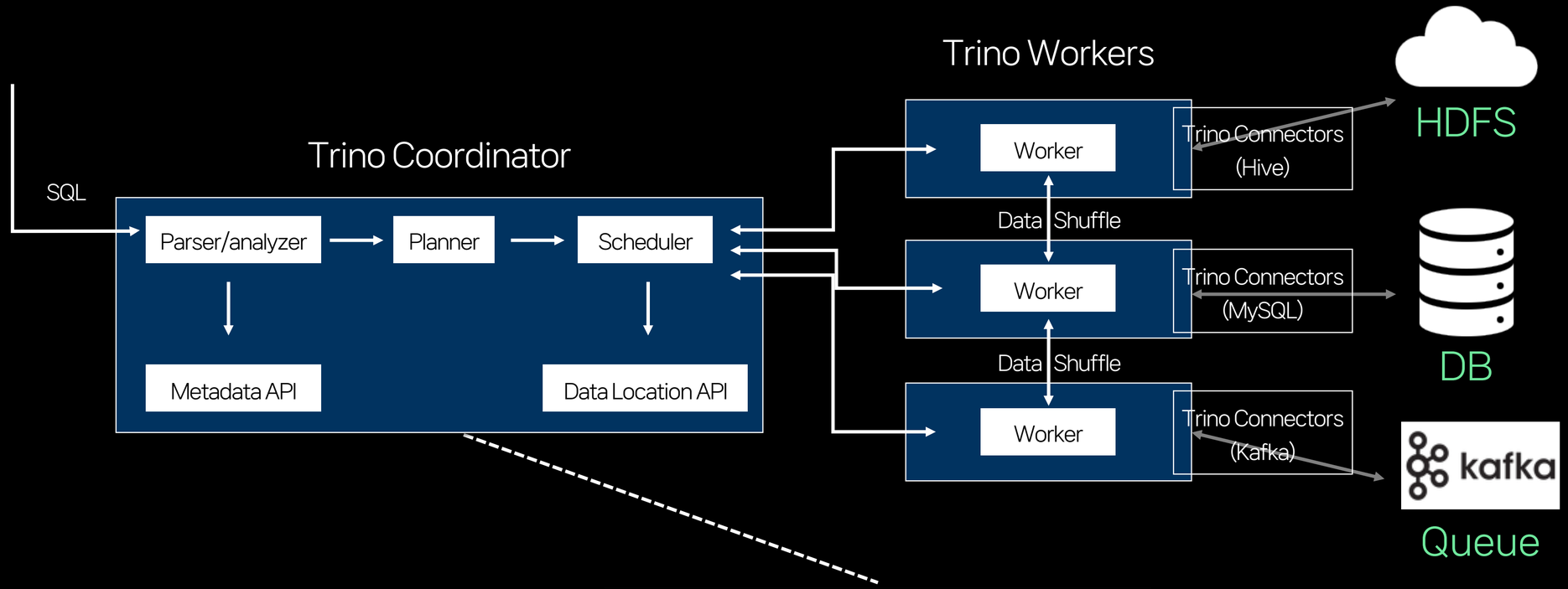
Data in splits is transferred between tasks and processed on different workers

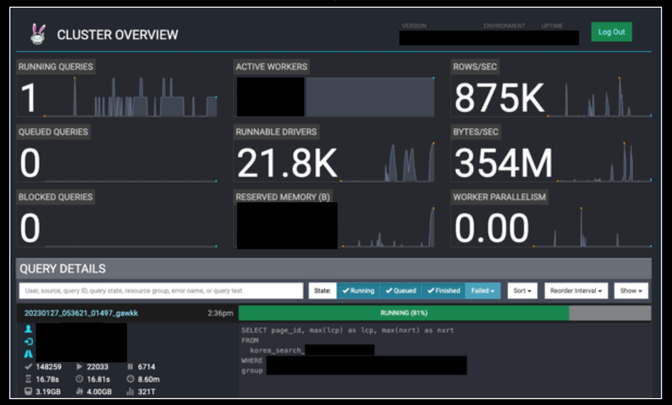
그림5 - https://www.oreilly.com/library/view/trino-the-definitive/9781098107703/ch04.html

[1] 이해하기 쉽게 Stage 번호 변경

3.3 Web UI

NAVER
DEVIEW
2023

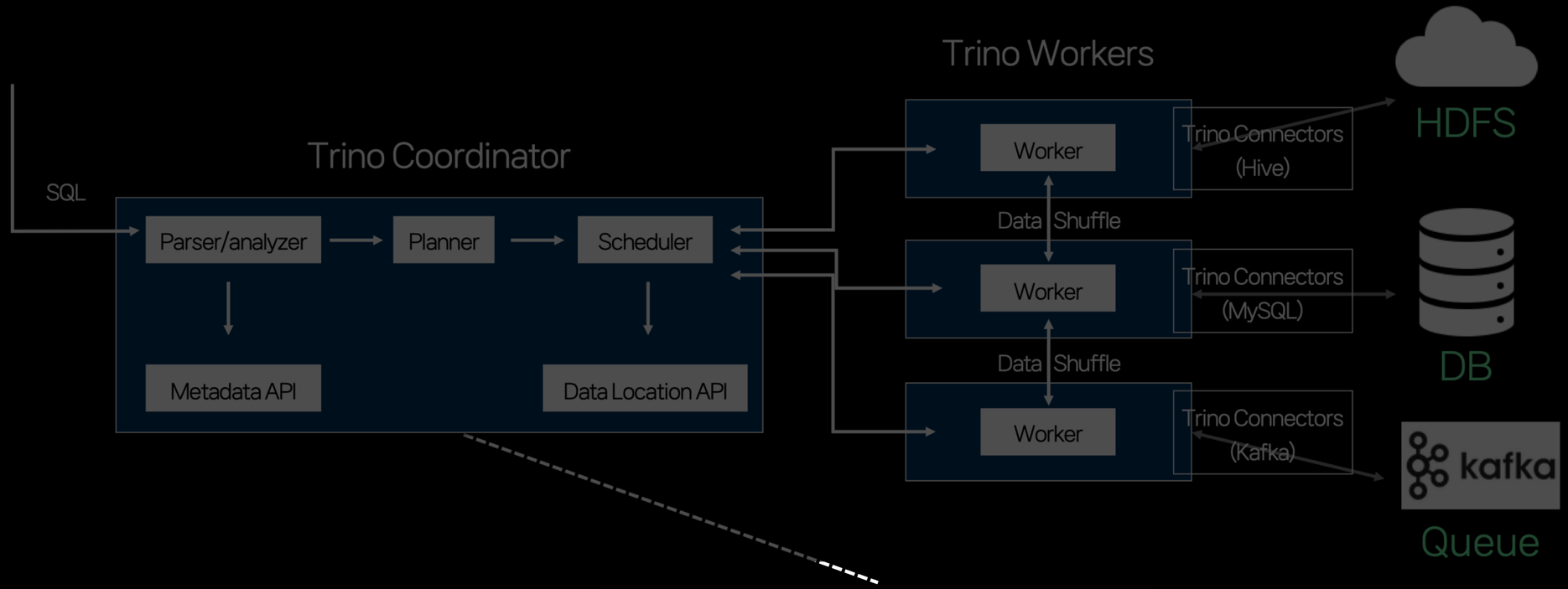


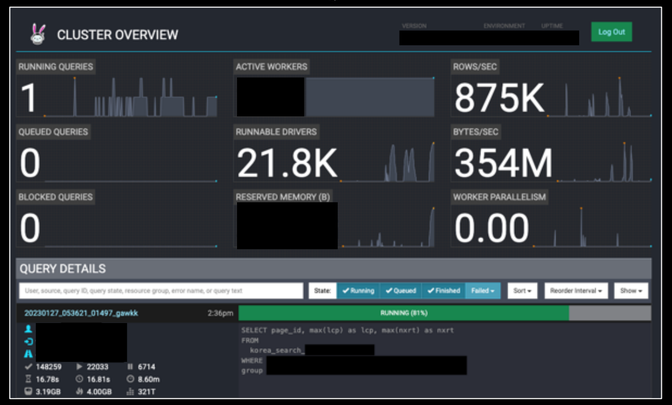


Trino Web UI (Coordinator)

3.3 Web UI

NAVER
DEVIEW
2023

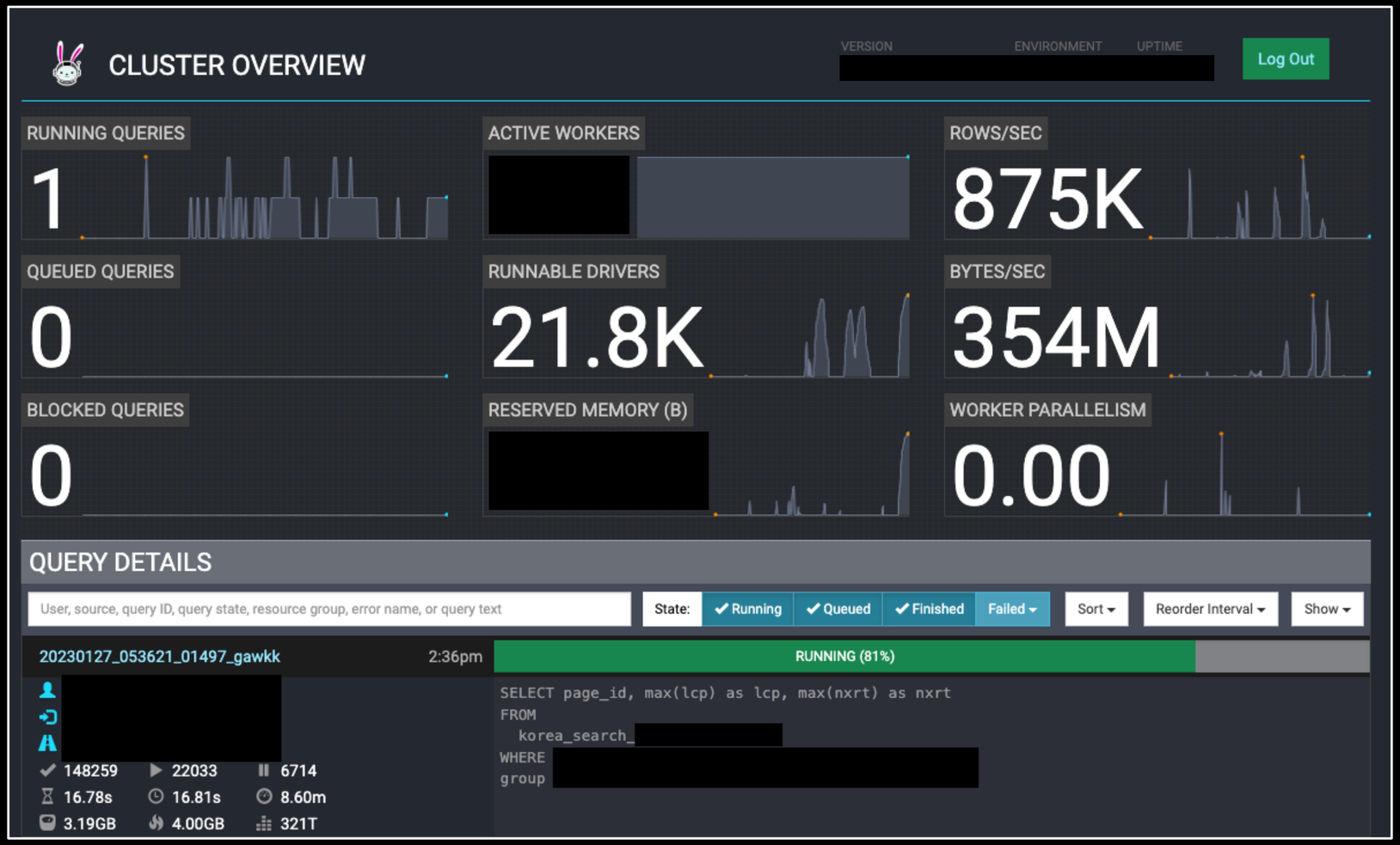




Trino Web UI (Coordinator)

3.3.1 Web UI – Cluster Overview

NAVER
DEVIEW
2023



Trino Web UI (Coordinator)

Web UI 에서 보여지는 모든 정보는 DB가 아닌 Coordinator 메모리에 임시 저장

3.3.1 Web UI – 접근 및 세션 제어

NAVER
DEVIEW
2023

파일 기반 접근 제어

```
{
    "queries": [
        {
            "user": "admin1|admin2|admin3",
            "allow": ["execute", "kill", "view"]
        },
        {
            "allow": ["execute", "kill"]
        }
    ],
    "system_information": [
        {
            "user": "admin1",
            "allow": ["read", "write"]
        }
    ],
    "impersonation": [
        {
            "original_user": "admin1|admin2|admin3",
            "new_user": ".*"
        }
    ],
```

access-rules.json

Queries: 사용자가 쿼리를 실행, 조회 또는 종료하는 기능 제어

- 일반 사용자가 Web UI 접근할 때 자기가 실행한 쿼리만 조회/종료가 가능하고 오직 관리자인 admin 계정만 모든 쿼리에 대해서 관리할 수 있도록 구분

System information: 시스템 정보 접근 권한 제어

- 운영에 필요한 시스템 정보가 불필요하게 공유되지 않도록 제한

Impersonation: 사용자가 다른 사용자를 가장하는 기능 제어

- 데이터 거버넌스 원칙을 지키기 위해 관리자 계정 admin만 impersonation 가능
- 쿼리 요청이 들어올 때마다 Kafka, HDFS, MS 등 각 컴포넌트 권한을 승인 받는 절차를 간편화하기 위해 관리자 계정으로 대행

3.3.1 Web UI – 접근 및 세션 제어

NAVER
DEVIEW
2023

파일 기반 세션 제어

```
23     "system_session_properties": [
24         {
25             "user": "admin2",
26             "allow": true
27         },
28         {
29             "property": "retry_policy",
30             "allow": true
31         }
32     ],
33     "catalog_session_properties": [
34         {
35             "catalog": "c3s",
36             "allow": true
37         }
38     ]
39 }
```

access-rules.json

System 및 Catalog Session: 사용자가 임의로 세션 설정 변경

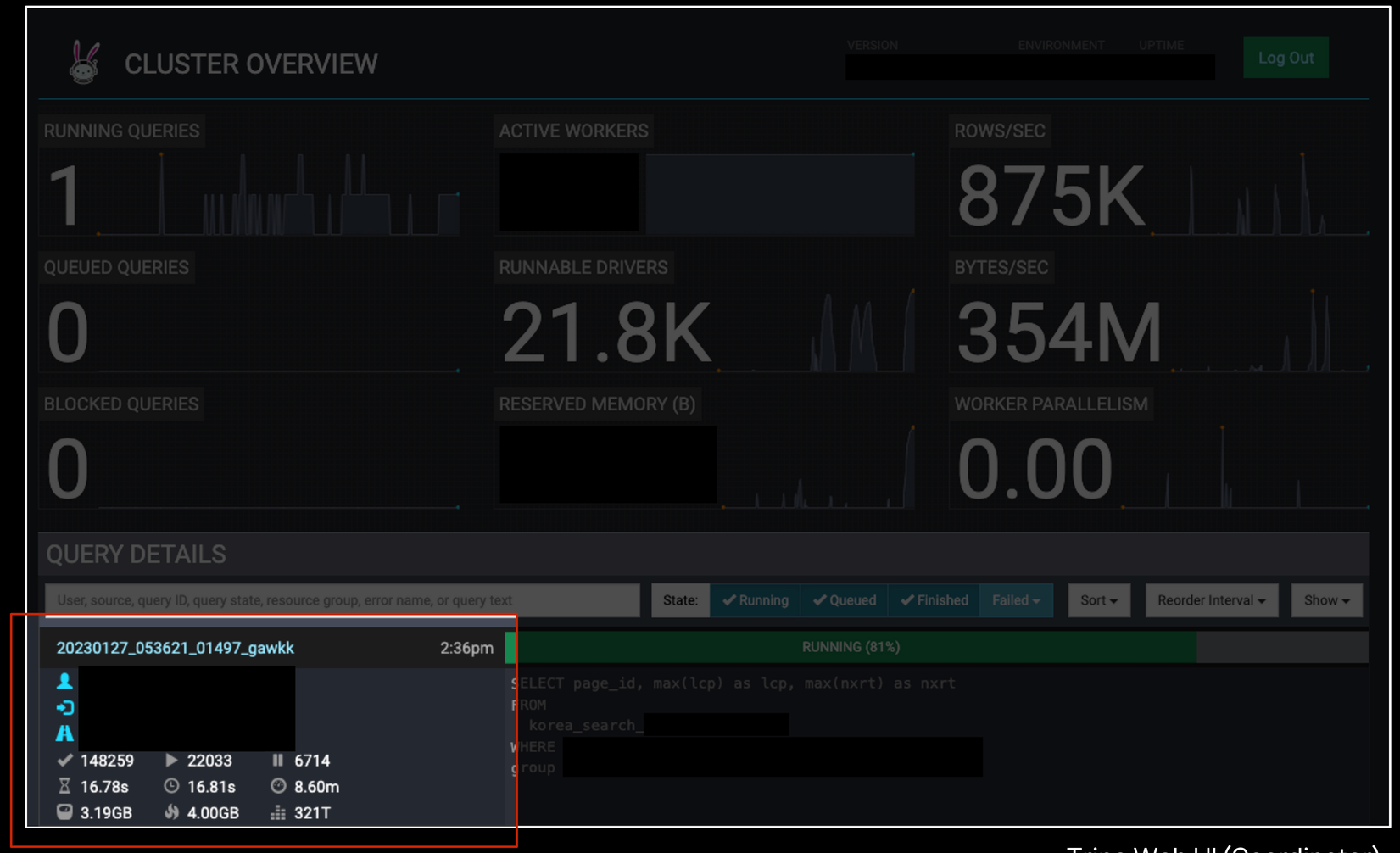
- 쓰레드 개수 등 클러스터 시스템 설정 변경은 안정상 관리자만 가능
- 저장 포맷 설정 등 Catalog 설정 변경 허용
  - Catalog란, 특정 데이터 소스에 대한 접근을 가능하게 하는 설정
  - 각 catalog 설정을 별도 파일로 관리할 수 있지만 편의상 JSON 하나로 클러스터에 연결된 모든 catalog 관리

Retry policy: 쿼리 작업 실패 시 자동으로 재시도

- Fault-tolerant execution 옵션 설정 켜기
- 특정 작업에 대해서 재시도가 불필요한 경우 사용자가 임의로 끌수 있도록 허용

3.3.1 Web UI – Cluster Overview

NAVER
DEVIEW
2023



Trino Web UI (Coordinator)

Web UI 에서 현재 실행중인 쿼리 정보

3.3.1 Web UI – Cluster Overview

NAVER
DEVIEW
2023

Source : **쿼리 실행** client (Python, GO, JDBC 등)

Resource Group : **속한 리소스 그룹** 4.2 Resource Group **참고**

Split : **데이터 처리하는 단위**

3.3.2 Web UI – Query Overview

NAVER DEVIEW 2023

Overview | Live Plan | Stage Performance | Splits | JSON

Query

```
SELECT

FROM k
WHERE
  AND
  AND
GROUP
```

Stages

Auto-Refresh: On

0

TIME		MEMORY		TASKS	
SCHEDULED	4.20s	CUMULATIVE	243K	PENDING	0
BLOCKED	1.92h	CURRENT	0B	RUNNING	0
CPU	1.10s	BUFFERS	0B	BLOCKED	0
FAILED	0.00ns	PEAK	310KB	FAILED	0
CPU FAILED	0.00ns	FAILED	0	TOTAL	100

SCHEDULED TIME SKEW

CPU TIME SKEW

1

TIME		MEMORY		TASKS	
SCHEDULED	29.26s	CUMULATIVE	6.27M	PENDING	0
BLOCKED	0.00ns	CURRENT	0B	RUNNING	0
CPU	10.66s	BUFFERS	0B	BLOCKED	0
FAILED	0.00ns	PEAK	55.7MB	FAILED	0
CPU FAILED	0.00ns	FAILED	0	TOTAL	197

SCHEDULED TIME SKEW

CPU TIME SKEW

Trino Web UI – Overview

Stage 0 : Coordinator에서만 실행되는 task으로 stage 1 작업의 결과 최종 집계

Stage N(1) : Distributed stage으로 각 worker에서 작업을 수행하는 단계

3.3.3 Web UI – Live Plan

NAVER DEVIEW 2023

Overview | Live Plan | Stage Performance | Splits | JSON

Tasks

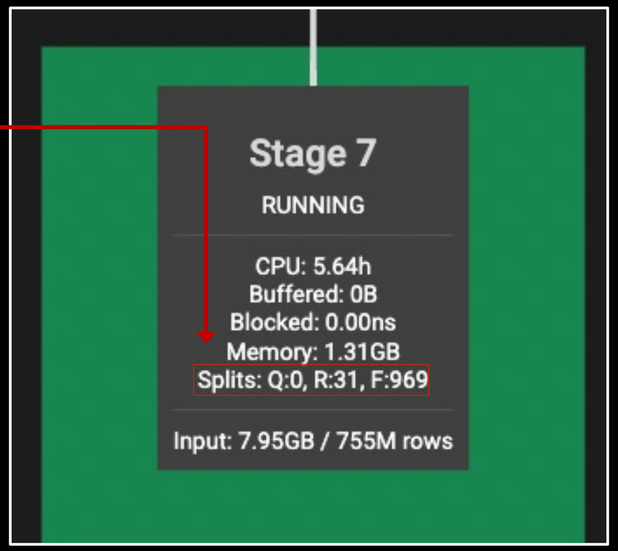
Show

ID	Host	State					Rows	Rows/s	Bytes	Bytes/s	Elapsed	CPU Time	Mem	Peak Mem
7.4.0	[redacted]	RUNNING	0	3	0	2	220K	4.30K	40.6M	812K	51.16s	2.42m	85.6MB	94.2MB
7.14.0		RUNNING	0	3	0	2	255K	4.97K	40.7M	813K	51.26s	2.45m	85.7MB	85.7MB
7.28.0		RUNNING	0	3	0	2	447K	8.68K	40.9M	812K	51.55s	2.48m	85.9MB	94.5MB

Trino Web UI – Overview

Overview | Live Plan | Stage Performance | Splits | JSON

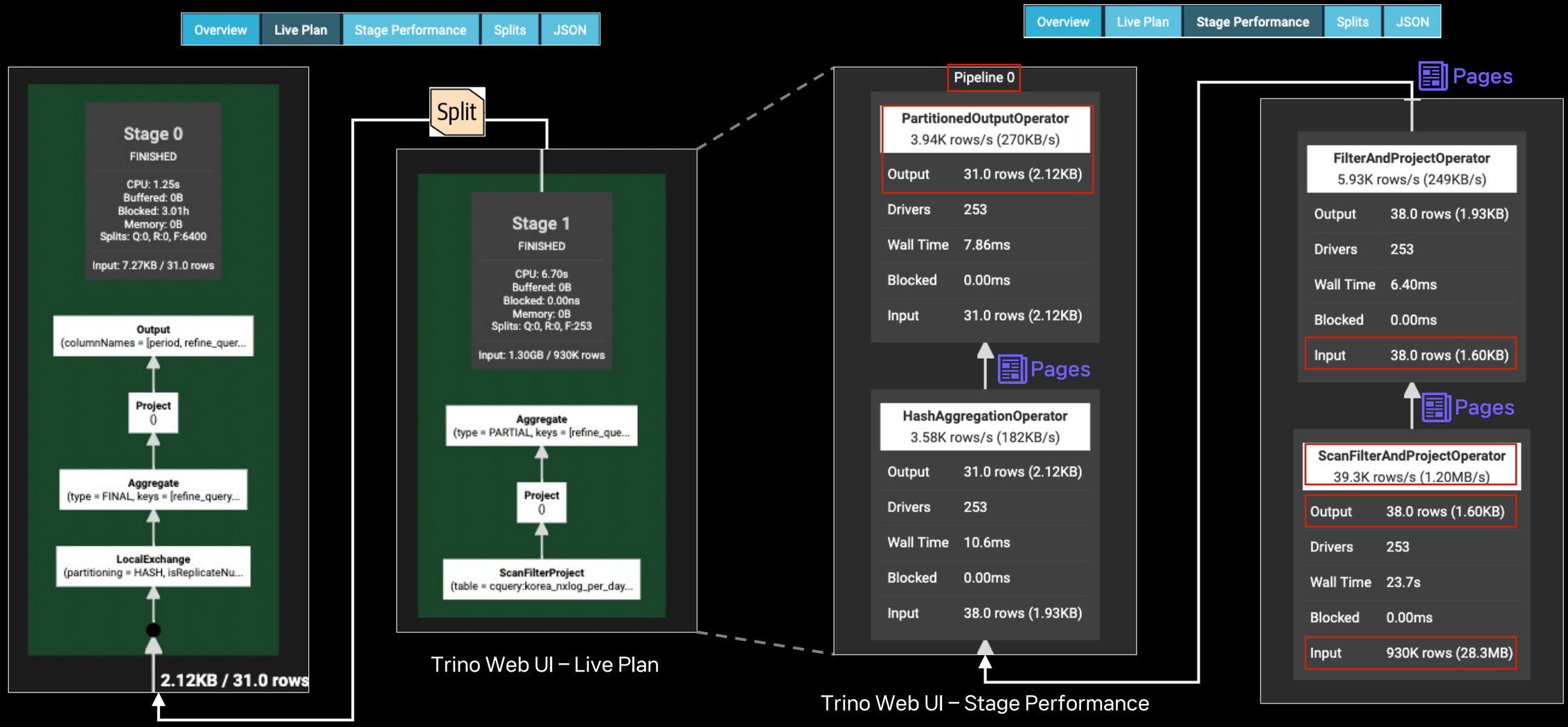
Q : Queued
R : Running
F : Finished



Trino Web UI – Live Plan

3.3.4 Web UI – Stage Performance

NAVER DEVIEW 2023

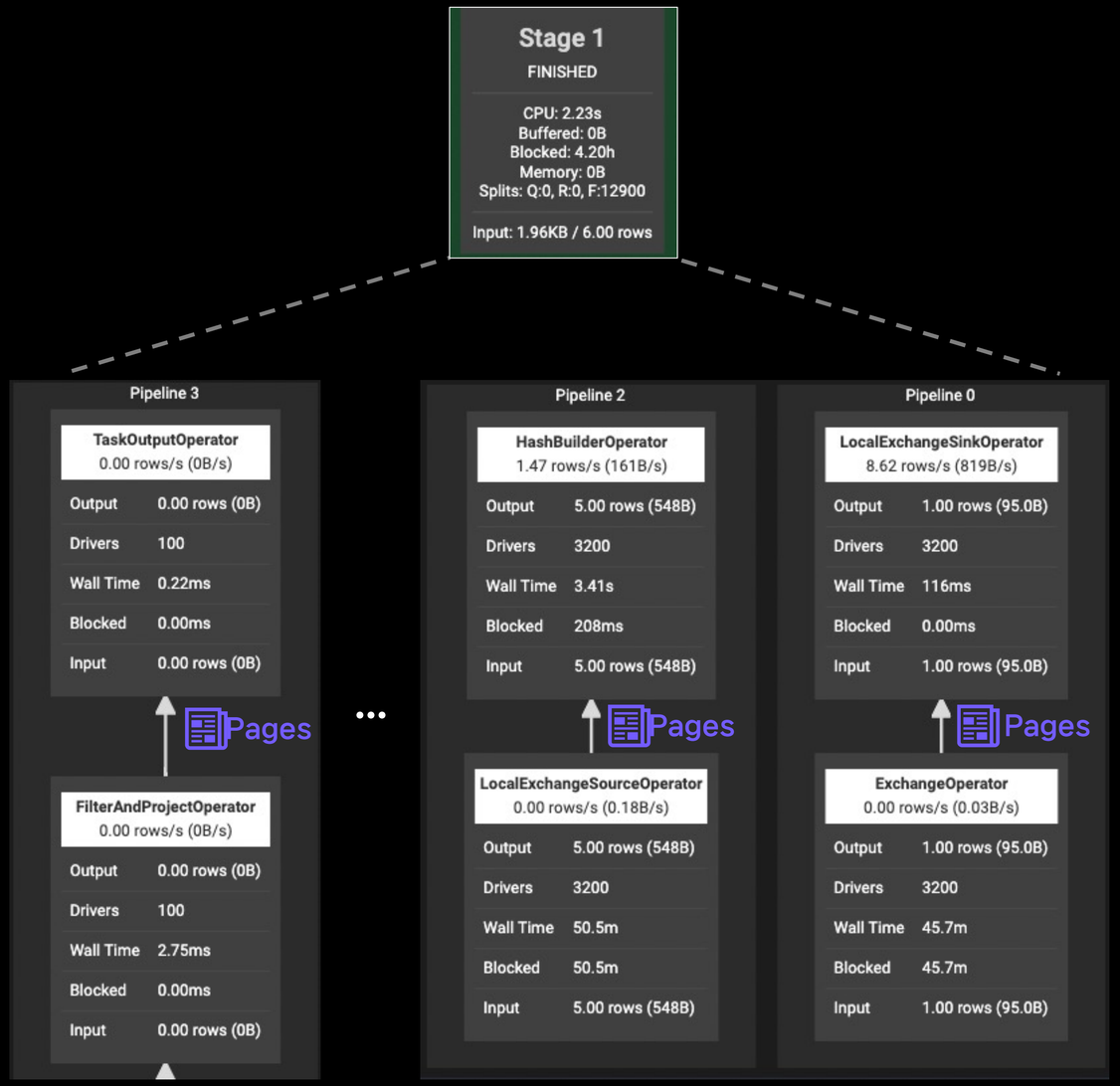


Pipeline : 데이터 처리 요소의 집합으로, 한 요소의 출력은 다음 요소의 입력

Operator : 데이터를 소비, 변환, 생성하는 역할

3.3.4 Web UI – Stage Performance

NAVER DEVIEW 2023



Trino Web UI – Stage Performance

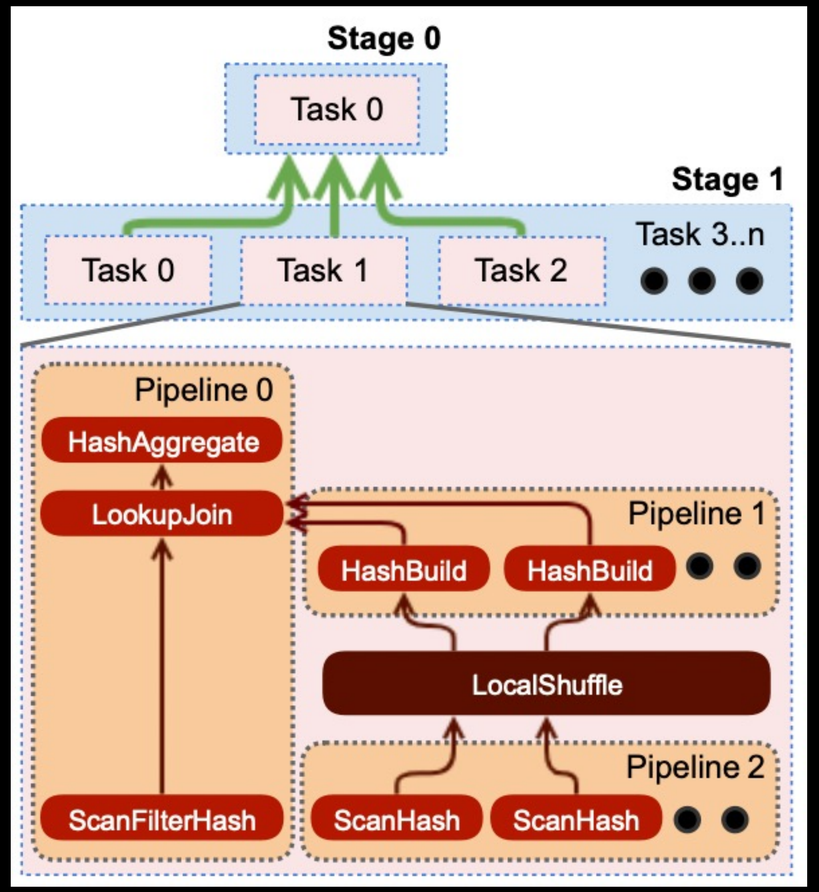


그림6 – Presto(Trino) SQL On Everything IEEE Paper

4. 우리 제법 잘 어울려요. (Ops)

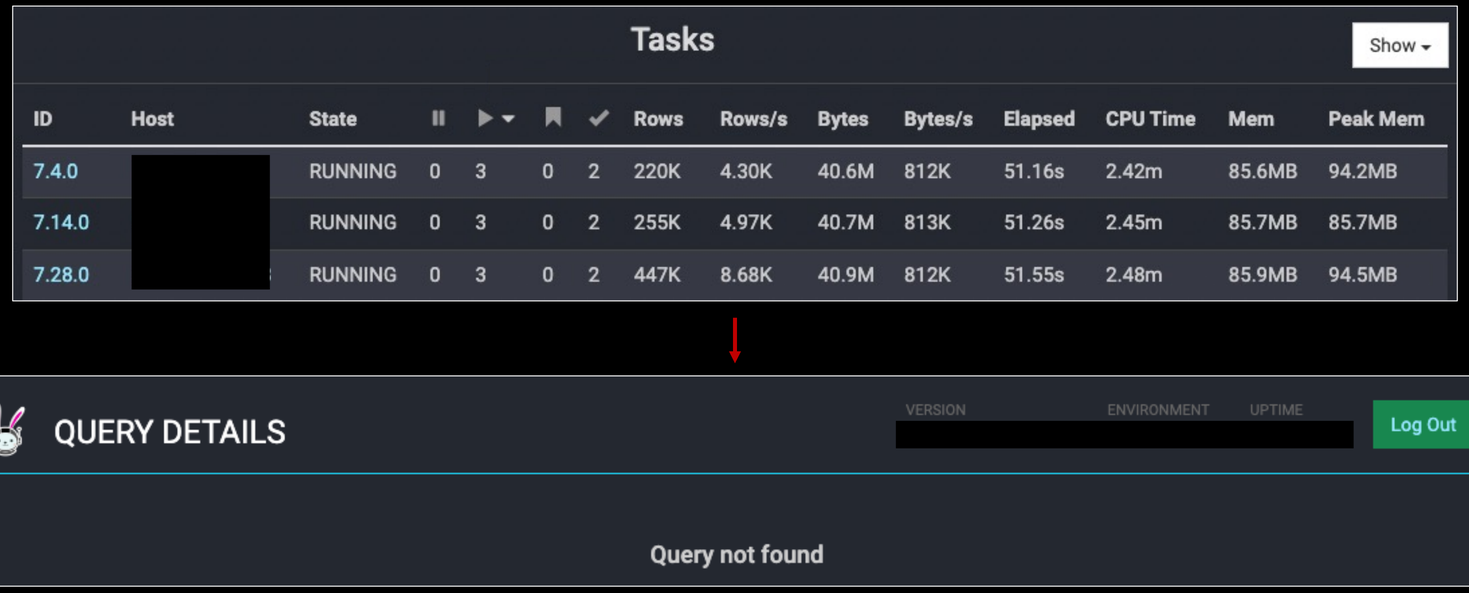
- Monitoring
- Resource
- Guard
- Stability

4.1 Monitoring 문제

NAVER
DEVIEW
2023

Web UI History 개수 이슈

- Web UI를 통해 Cluster 상태, 쿼리 정보 등을 실시간으로 볼 수 있어서 문제가 발생하면 상황 파악하기 용이
- 하지만 화면에 보이는 모든 정보는 Coordinator 메모리에서 관리
- 메모리 한계 때문에 UI에서 보여줄수 있는 실행 목록 제한 (CQuery 100개)
- History 개수만큼 GC가 되지 않기 때문에 개수를 너무 늘리면 OOM, 서버 무응답 등 장애 발생
  - https://trinodb.slack.com/archives/CGB0QHWSW/p1659118464926819



4.1 Monitoring 데이터 연동

NAVER
DEVIEW
2023

모니터링 기능 지원

- 특히 메모리 기반 SQL 엔진을 운영하는데 있어 모니터링이 중요
- Web UI에서 모든 지표 확인 불가능
- 실시간 지표와 실행된 쿼리의 정보를 쌓는게 중요

모니터링을 위해 활용한 기능

- 기본 지표 - JMX 로 제공하는 지표를 Prometheus와 연동
  - Active한 Worker 장비 수
  - 장비 별 CPU, 메모리 여유 상태

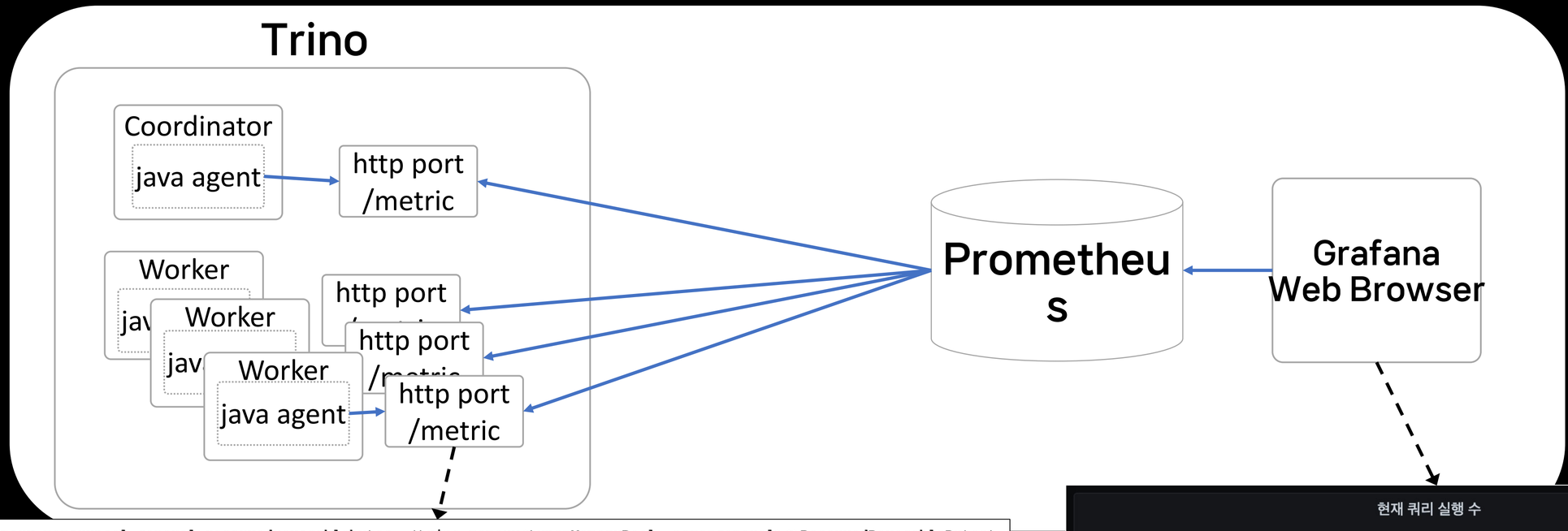
Custom 지표 - Event Listener를 통한 쿼리 별 실행 정보를 Elastic Search 와 연동

- 실행한 쿼리 정보
- 사용한 리소스 정보 (메모리, CPU, Input 파일 크기 등)
- 실패한 쿼리 exception 정보

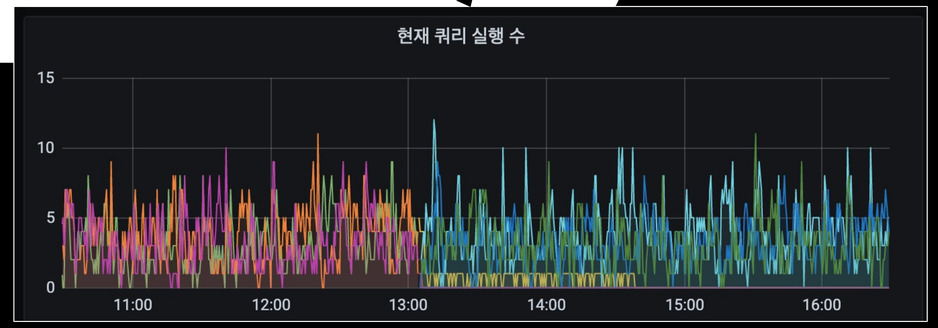
4.1.1 JMX, Prometheus 연동

NAVER
DEVIEW
2023

JMX 로 제공하는 지표를 Prometheus 와 연동 흐름




```
# HELP trino_memory_memorypool_general_reservedrevocablebytes  (trino.memory<type=MemoryPool, name=general><>ReservedRevocableBytes)
# TYPE trino_memory_memorypool_general_reservedrevocablebytes gauge
trino_memory_memorypool_general_reservedrevocablebytes 0.0
# HELP trino_memory_memorypool_general_freebytes  (trino.memory<type=MemoryPool, name=general><>FreeBytes)
# TYPE trino_memory_memorypool_general_freebytes gauge
trino_memory_memorypool_general_freebytes 8.8691074662E10
# HELP trino_execution_taskmanager_outputpositions_totalcount_total  (trino.execution<name=TaskManager><>OutputPositions.TotalCount)
# TYPE trino_execution_taskmanager_outputpositions_totalcount_total counter
trino_execution_taskmanager_outputpositions_totalcount_total 2.769115537E9
# HELP trino_java_runtime_start_time StartTime (java.lang<type=Runtime><>StartTime)
# TYPE trino_java_runtime_start_time gauge
```



4.1.1 JMX, Prometheus 연동

NAVER
DEVIEW
2023

- **JMX 로 제공하는 지표를 Prometheus와 연동 방법**
  - Prometheus에서 제공하는 Java Agent를 활용한 JMX Exporter 사용
  - JMX Exporter config 설정 방법

```
rules:
  # trino {{{
  - pattern: 'trino.execution[redacted]
    name: 'trino_execution_$1_$2'
    help: '$1 time in milliseconds for alltime'
    type: GAUGE
  - pattern: 'trino.execution[redacted]
    name: 'trino_execution_$1_$2_total'
    type: COUNTER
  # }}}

  # Execution and queued time histograms/summaries {{{
  - pattern: 'trino.execution[redacted]
    name: 'trino_execution_querymanager_$1time_fiveminutes_p$2'
    help: '$1 time in milliseconds for the $2 quantile'
    type: GAUGE
  # }}}
```

- **JMX 설정 방법**
  - jvm.config

```
...
-javaagent:/etc/trino/jmx_prometheus_javaagent-...jar=...:/etc/trino/prometheus_config.yml
```

4.1.2 Event Listener, Elastic Search 연동

NAVER
DEVIEW
2023

Event Listener를 통한 쿼리 별 실행 정보를 Elastic Search 와 연동 방법

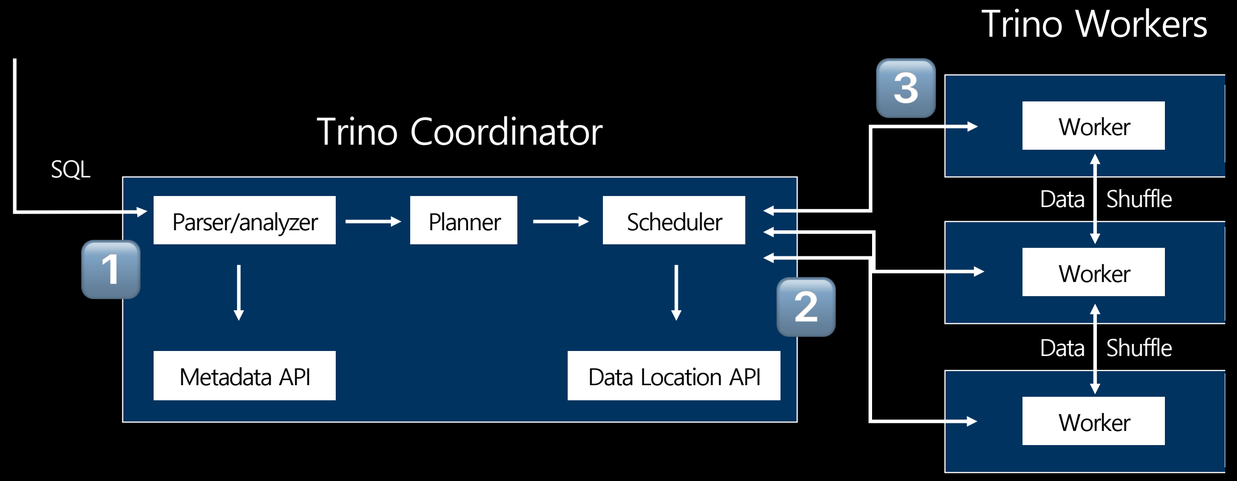
- Event Listener 는 Service Provider Interface(SPI) Plugin 모듈로 제공
- 개발자가 직접 custom 할 수 있게 제공되는 모듈
  - SPI 는 Event Listener 이외의 Access Control, User Defined Functions 등 다양한 Plugin을 제공
- 사용자 편의를 위해 Coordinator에서 Custom Event Listener 지원

4.1.2 Event Listener, Elastic Search 연동

NAVER DEVIEW 2023

Event Listener 란?

- 아래 이벤트가 발생할 때 Custom하게 제어할 수 있도록 지원하는 Listener
  1. Query Creation: 사용자 쿼리가 입력되고 실행되기 전 발생하는 이벤트
  2. Query Completion: 사용자 쿼리가 종료된 이후 발생하는 이벤트
  3. Split Completion: 쿼리 실행 중 각각의 Stage 마다 split 완료될 때마다 발생하는 이벤트



4.1.2 Event Listener, Elastic Search 연동

NAVER DEVIEW 2023

Event Listener 구현 방법

- SPI Plugin 에서 getEventListenerFactories에 EventListenerFactory 구현한 객체를 전달
- EventListenerFactory 는 개발자가 구현한 EventListener 객체 전달
- EventListener 객체는 3개 이벤트 함수를 제공하는 클래스

```java
import io.trino.spi.Plugin;
import io.trino.spi.eventlistener.EventListenerFactory;

public class QueryEventListenerPlugin implements Plugin {

    @Override
    public Iterable<EventListenerFactory> getEventListenerFactories() {
        EventListenerFactory listenerFactory = new QueryEventListenerFactory();
        return ImmutableList.of(listenerFactory);
    }
}
```

```java
public class QueryEventListenerFactory implements EventListenerFactory {

    @Override
    public String getName() {
        return [redacted]
    }

    @Override
    public EventListener create(Map<String, String
        return new QueryEventListener(config);
    }
}
```

```java
public class QueryEventListener implements EventListener {

    public QueryEventListener(Map<String, String> config) {
    }

    public void queryCreated(QueryCreatedEvent event) {
        LOG.debug("queryCreated");
    }

    public void queryCompleted(QueryCompletedEvent event) {
        LOG.debug("queryCompleted");
    }

    public void splitCompleted(SplitCompletedEvent event) {
        LOG.debug("splitCompleted");
    }
}
```

4.2 Resource Groups

NAVER
DEVIEW
2023

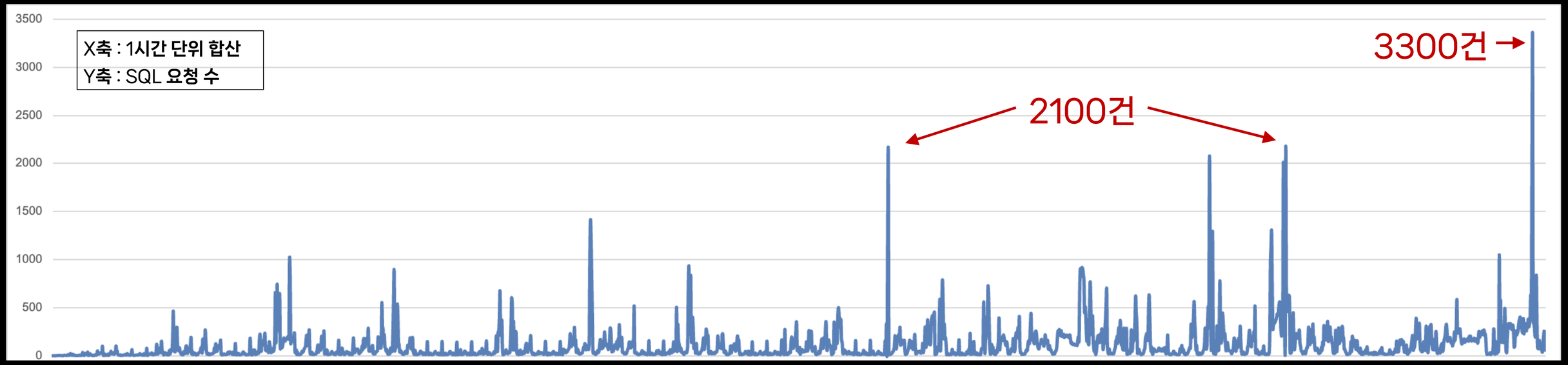
특정 사용자의 다수의 쿼리 어뷰징 문제 발생

- FOR 문을 이용해 기간만 바꿔서 동시에 대량 요청 → Coordinator 서버 응답 지연 및 서비스 장애
- 단기간 요청이 많을 경우 시스템 불안전

4.2 Resource Groups

NAVER
DEVIEW
2023

다수의 쿼리 어뷰징 패턴 파악



시간대별 쿼리 요청 그래프

4.2 Resource Groups

NAVER
DEVIEW
2023

데이터베이스 기반 Resource Group 설정

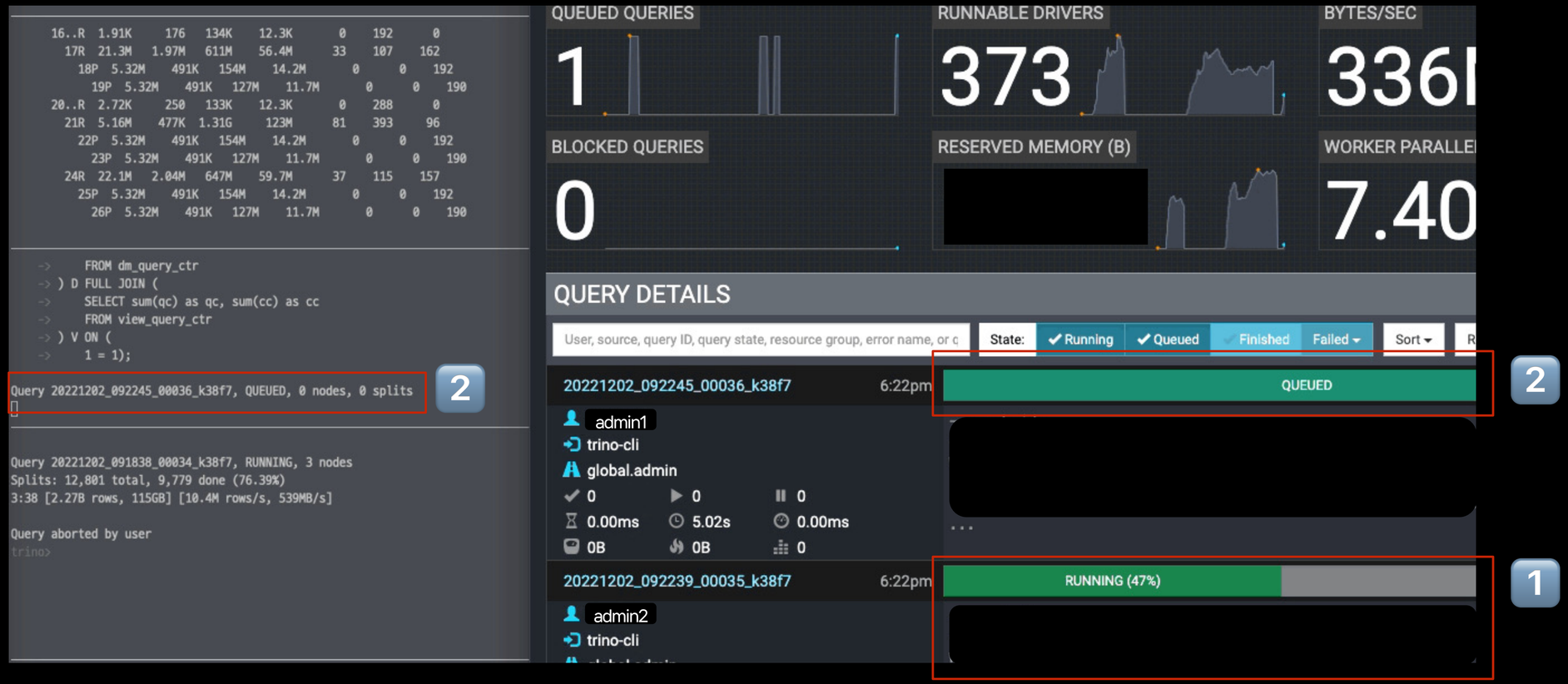
그룹 종류	Soft Memory Limit	Max Queue	Concurrency Limit	Scheduling Wait
${user}	50%	20	10	1
admin	80%	100	20	10

- **그룹 종류**
  - Admin: 관리자 그룹에 속한 관리자 계정들 모두에 대한 제한 설정
  - ${user}: 사용자 계정별로 각각 제한 적용
  - 예) 일반 계정 foo 및 bar 각각 10개씩 동시 실행, admin 계정 admin1이 15개 실행중이면 admin2는 최대 5개 실행 가능
- Soft Memory Limit: **클러스터 내 사용할 수 있는 최대 메모리 제한**
- Max Queue: **큐 쌓일수 있는 최대 개수**
- Concurrency Limit: **동시에 실행 가능한 쿼리 개수**
- Scheduling Weight: **숫자가 높을 수록 높은 우선 순위**

4.2 Resource Groups

NAVER DEVIEW 2023

Demo - Concurrency Limit (1)



4.2 Resource Groups

NAVER
DEVIEW
2023

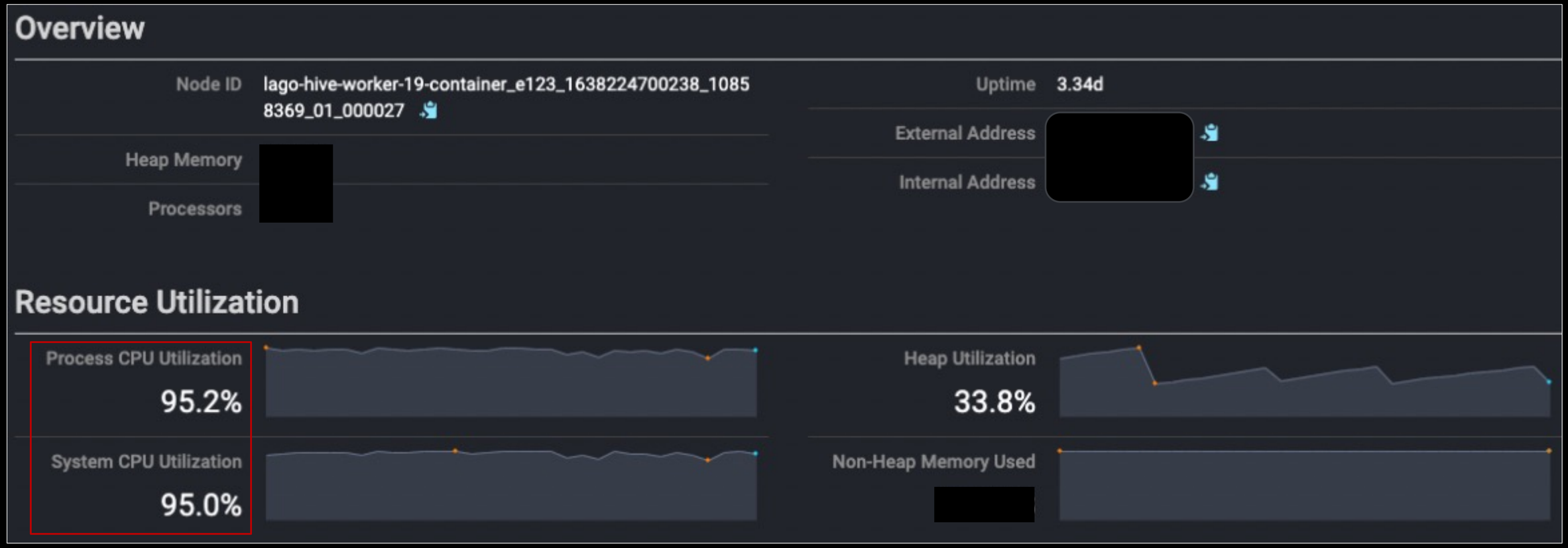
Demo - Max Queue (1)

4.3 Guard

NAVER
DEVIEW
2023

Abusing 쿼리로 인해 클러스터 리소스 전부 잠식하는 경우 발생

- 특정 worker 장비에서 장기간 CPU 로드 과부하로 인해 클러스터 데이터 처리 지연



4.3 Guard

NAVER
DEVIEW
2023

```
select count(*) as new
from (select refine_query
        from ...refine_query_2020
        union
      select refine_query
        from ...daily_refine_query
       where log_date <= '2021-12-07')
```

- **중복 제거하면서 합치는 UNION 문제로 CPU 리소스 과다 사용**
- **파티션 조건 (log_date)가 있어도 장기간(대량) 데이터 조회 문제**

```
SELECT count(distinct refine_query)
FROM ...ctr_query
WHERE log_date BETWEEN '2020-01-01' AND '2021-12-07'
AND ...
AND cc>0
```

- **고유한 refine_query를 찾는 count(distinct) 같이 무거운 작업 사용**
  - 문서에는 보다 안전한 approx_distinct() 함수 사용 권장
- 약 2년치 데이터로 TB 크기의 약 몇 천억 로그 대용량 데이터 조회

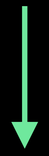
읽는 데이터량에 대한 제한을 둬서 과도한 리소스 사용 제한

4.3 Guard

NAVER
DEVIEW
2023

Query 제어 매커니즘 부재

- Trino EventListener는 Query 처리 시작/종료 이벤트만 받을 수 있음
- 읽는 데이터량을 파악하기 위해서 SQL 파싱 필요
- Query 시작 알림을 받고 중지하기에는 리소스를 이미 사용하고 있어서 원천 차단 방법 필요



- ASM 또는 BCEL 기반으로 Java runtime hooking 적용
  Trino Coordinator 소스를 분석하여 소스 레벨에서 SQL 처리 위치 파악
  Coordinator JVM 실행 시 JavaAgent 옵션을 통해 runtime hooking 로직 실행
- 조회 기간을 찾는 Trino 함수를 이용해서 데이터량 제한

4.3 Guard – Java Agent Hooking

NAVER
DEVIEW
2023

ByteBuddy library를 Java Runtime Hooking 과정에 이용하여 abusing 쿼리 제한

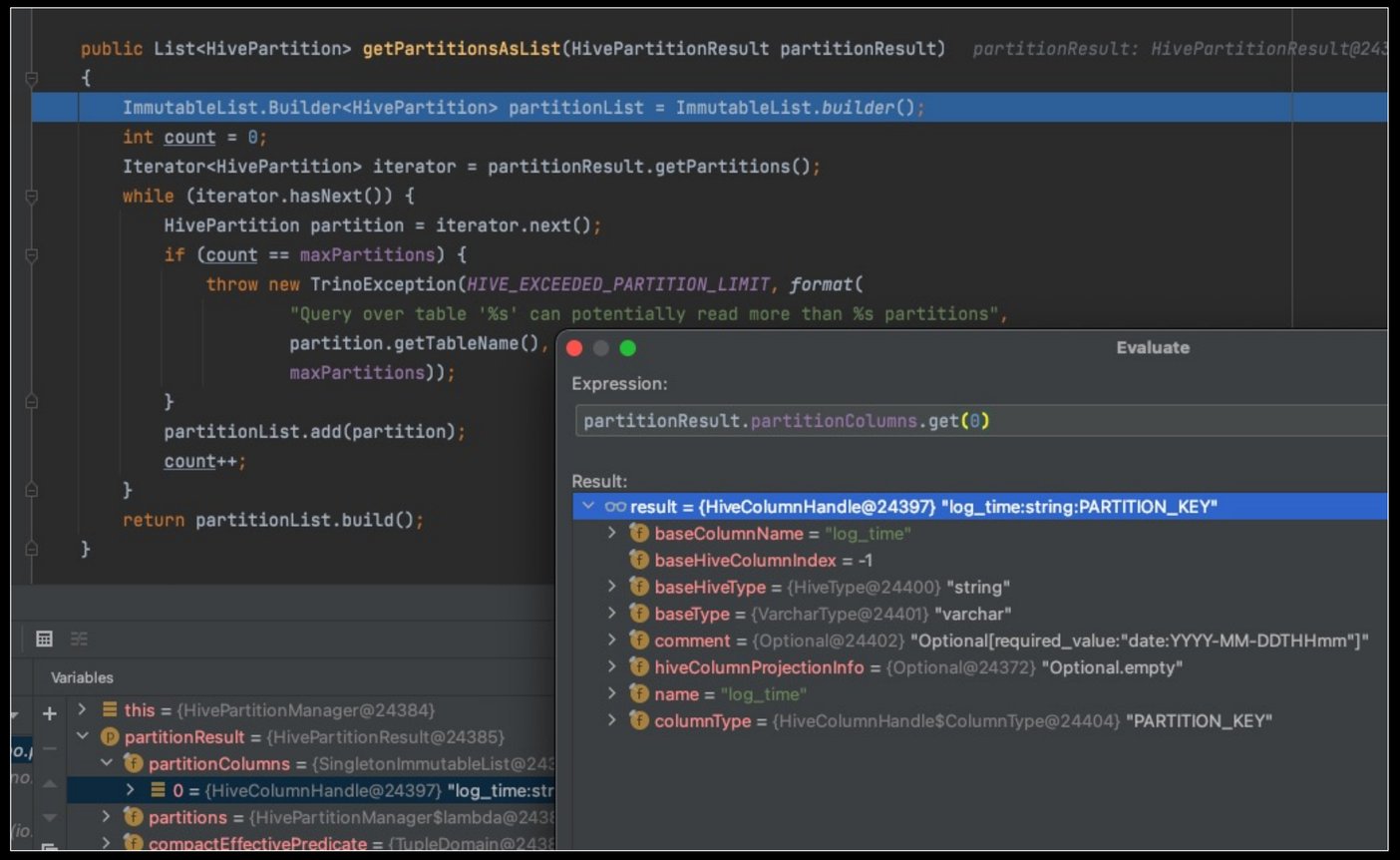
ByteBuddy: ASM library를 감싸서 Java Class를 쉽게 고치게 해주는 툴

- Abusing 쿼리 제한을 위해 오픈소스를 AIDA CQuery Trino에 알맞게 수정하는 방식이 아니라 Bytecode를 runtime에 수정하는 Java Agent 라는 instrumentation API를 이용하는 방식 선택
- 위 작업을 편리하게 해주는 ByteBuddy Library 에서 premain 방법 이용
  - Premain 단계에서 class transformer 등록하고 main이 실행하기 때문에 클래스 파일을 읽을 때 변경된 코드 실행

4.3 Guard – Java Agent Hooking

NAVER
DEVIEW
2023

HivePartitionManager 클래스 getPartitionsAsList 함수를 Hooking



- getPartitionsAsList 함수의 partitionResult 객체가 쿼리에서 사용되는 모든 파티션 정보 보유
- 파티션 개수 계산해서 어뷰징으로 판단될 경우 Exception을 던져서 쿼리 종료
- 예) log_time 파티션 필드 제한

4.3 Guard – Java Agent Hooking

NAVER
DEVIEW
2023

ByteBuddy에서 사용할 agent builder 생성

```
/**
 * Java agent for Trino Hooking
 */
public class HookAgent {
    public static void premain(String arguments, Instrumentation instrumentation) {
        // register a hook, advice in terms of byte-buddy, for io.trino.plugin.hive.HivePartitionManager#getPartitionsAsList() method
        // HivePartitionManager#getPartitionsAsList() has one parameter; result of fetching Partitions in the HivePartitionManager interface.
        [illegible]
    }
}
```

- Hooking할 HivePartitionManager 클래스의 getPartitionAsList 함수와 직접 작성한 어뷰징 제한 코드를 이용해서 Agent Builder 생성

4.3 Guard – Java Agent Hooking

NAVER
DEVIEW
2023

일자별 파티션 (log date) 기준 최대 1년 제한

```
trino> SELECT count(*) FROM                    ."ctr_query" WHERE log_date = '2022-02-15';
  _col0
------------
            
(1 row)

Query 20220222_074842_00003_u5rup, FINISHED, 3 nodes
Splits: 123 total, 123 done (100.00%)
0.91 [                                    ]

trino> SELECT count(*) FROM                    ."ctr_query" WHERE log_date between '2020-01-01' and '2022-02-20';
Query 20220222_074852_00004_u5rup failed: The field( log_date ) range (partition size) exceeds maximum size "365 + 5 days", current size [ 782 days ], location [                    .ctr_query ].
```

- 최대 1년 기준은 운영하면서 최대 분석 적정 기간으로 판단
- **조회 기간이 제한 범위 이내면 쿼리가 정상적으로 실행**

4.4 Stability

NAVER
DEVIEW
2023

최대 1년 제한 뒀음에도 쿼리 복잡도에 따라 메모리 한도 초과 문제 발생

Error Information

Error Type	INSUFFICIENT_RESOURCES
Error Code	EXCEEDED_LOCAL_MEMORY_LIMIT (131079)

Stack Trace

```
io.trino.ExceededMemoryLimitException: Query exceeded per-node total memory limit of N GB [Allocated: [redacted] GB, Delta: 1
        at io.trino.ExceededMemoryLimitException.exceededLocalTotalMemoryLimit(ExceededMemoryLimitException.java:46)
        at io.trino.memory.QueryContext.enforceTotalMemoryLimit(QueryContext.java:378)
        at io.trino.memory.QueryContext.updateSystemMemory(QueryContext.java:211)
        at io.trino.memory.QueryContext$QueryMemoryReservationHandler.reserveMemory(QueryContext.java:351)
        at io.trino.memory.context.RootAggregatedMemoryContext.updateBytes(RootAggregatedMemoryContext.java:37)
        at io.trino.memory.context.ChildAggregatedMemoryContext.updateBytes(ChildAggregatedMemoryContext.java:38)
        at io.trino.memory.context.ChildAggregatedMemoryContext.updateBytes(ChildAggregatedMemoryContext.java:38)
        at io.trino.memory.context.ChildAggregatedMemoryContext.updateBytes(ChildAggregatedMemoryContext.java:38)
```

- query_max_memory_per_node 설정: worker 노드당 쿼리가 사용할 수 있는 메모리 제한 (N GB)
- 장비 추가 투입하여 어느정도 해결이 되었으나 UNION, JOIN 등 메모리 과다 사용시 실패

↓

- Spill to Disk **기능 사용**
- **일시적으로 한도 이상 메모리 사용** (OS의 Page Swapping과 흡사)

4.4 Stability – Spill to Disk

NAVER DEVIEW 2023

Revocable Memory

- 필요한 만큼 추가로 메모리 사용을 허용해주되, 상황에 따라 메모리 관리자가 허용한 메모리를 다시 회수
- **한도 이상 메모리를 사용하되 과도한 리소스 사용으로 인해 발생할 수 있는** deadlock **방지**

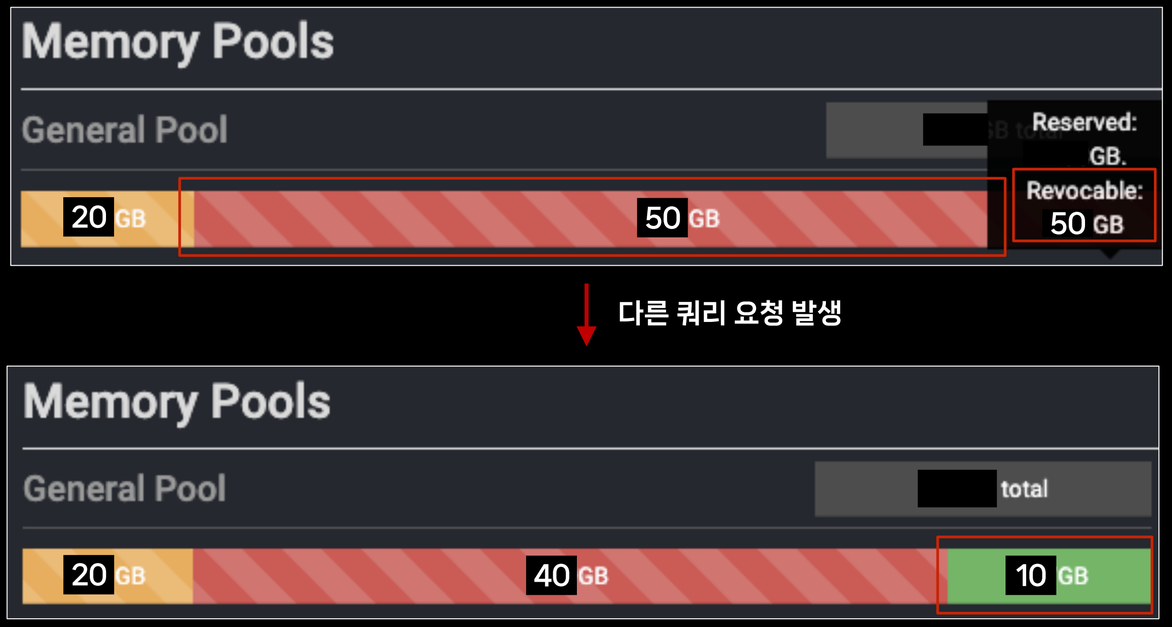
- **메모리 사용 제한**(20GB) **넘어선** 50GB **추가 사용** → 50 GB 중에 **일부 회수되어** 40GB **만 사용, 총** 10GB **여유 메모리 확보**

4.4 Stability – Spill to Disk

NAVER
DEVIEW
2023

Spill-to-disk를 이용하여 Revocable 메모리 데이터를 디스크에 임시 보관 후 사용

- 임시 보관하고 메모리 확보가 다시 가능한 시점에 쿼리를 재게 (메모리 확보될 때까지 쿼리 수행 중지)




```
[                    spill-path]$ ll -lh
total 10 GB
1.1G Aug 25 18:15 spill1162861998266928566.bin
1.1G Aug 25 18:16 spill12293847191661479208.bin
7.5M Aug 25 18:16 spill12344887048236082927.bin
1.1G Aug 25 18:14 spill14801098577679614139.bin
1.1G Aug 25 18:15 spill162474956787051774.bin
1.1G Aug 25 18:15 spill18274658173329399338.bin
1.1G Aug 25 18:16 spill2630188357860452690.bin
1.1G Aug 25 18:16 spill536866287967207889.bin
213M Aug 25 18:16 spill5638484848812972089.bin
1.1G Aug 25 18:16 spill6169298546350652686.bin
163M Aug 25 18:16 spill735734140055730647.bin
```

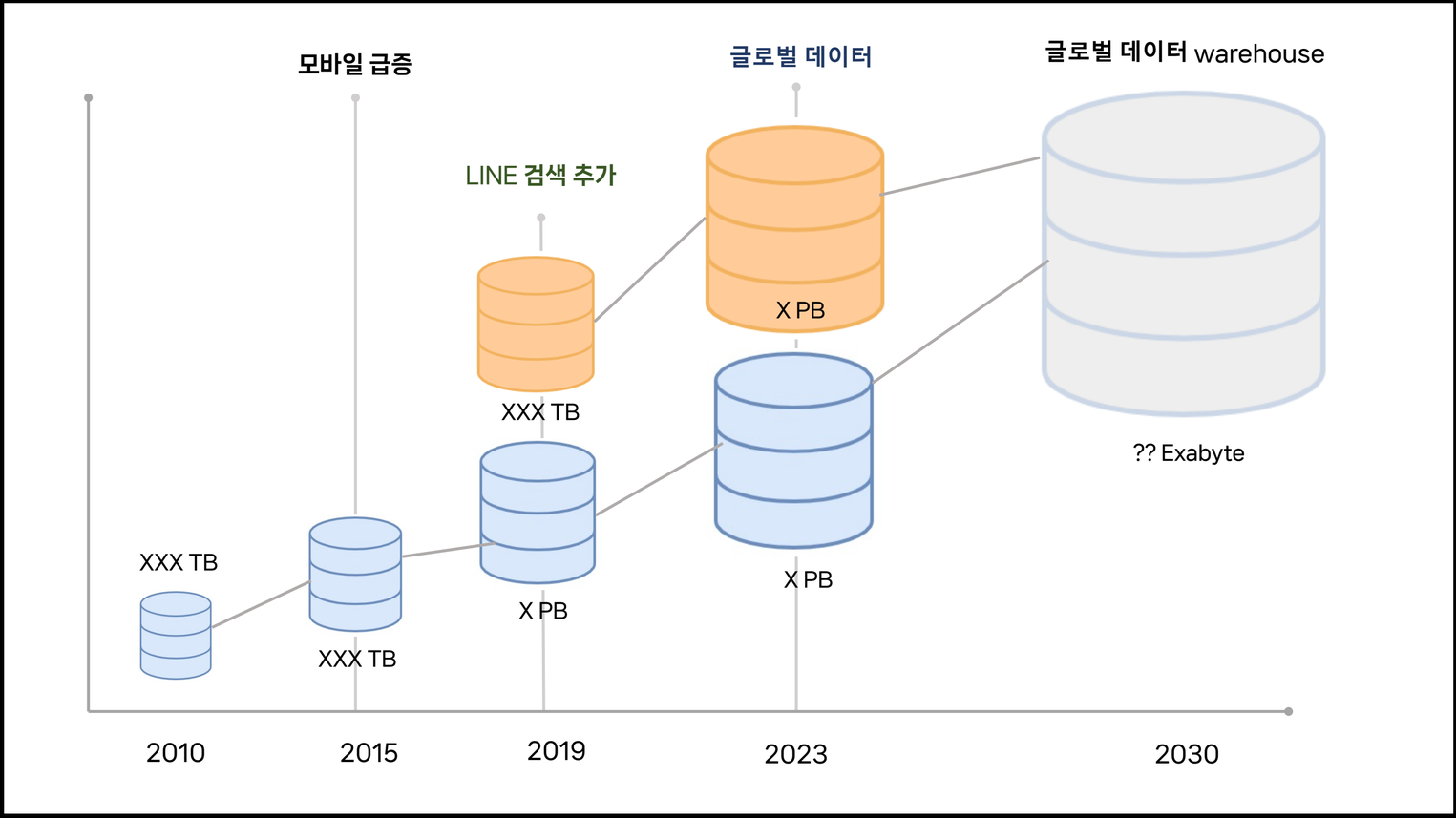
회수된 메모리 만큼 disk에 임시 저장

5. 미래를 설계해요. (To be)

- CXL 기반 메모리 PoC
- Workload Specific Cluster
- Iceberg

5.1 CXL 기반 메모리 PoC

NAVER
DEVIEW
2023



- 어뷰징 막고, 리소스를 효율적으로 사용하는 등의 SW적인 부분으로 해결하는데 한계
- SW 튜닝 대비 데이터 증가 속도가 빨라 성능 개선에 한계
- 저장하는 데이터 용량은 매해 증가하고 대용량 쿼리 분석을 위한 근본적인 문제 고민

→ Scale Up

5.1 CXL 기반 메모리 PoC

NAVER
DEVIEW
2023

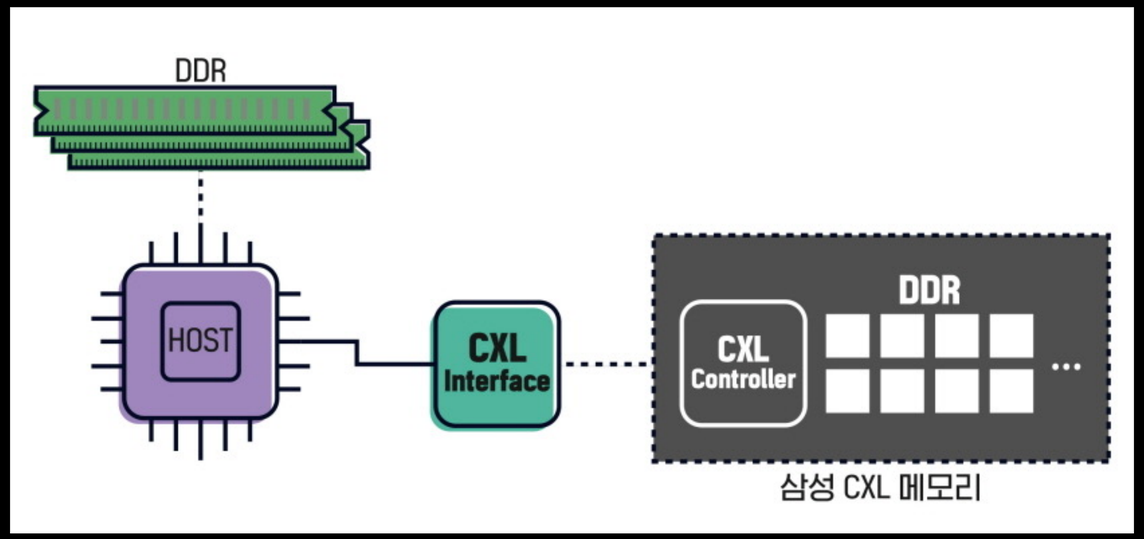


그림7 - https://www.aitimes.kr/news/articleView.html?idxno=20994

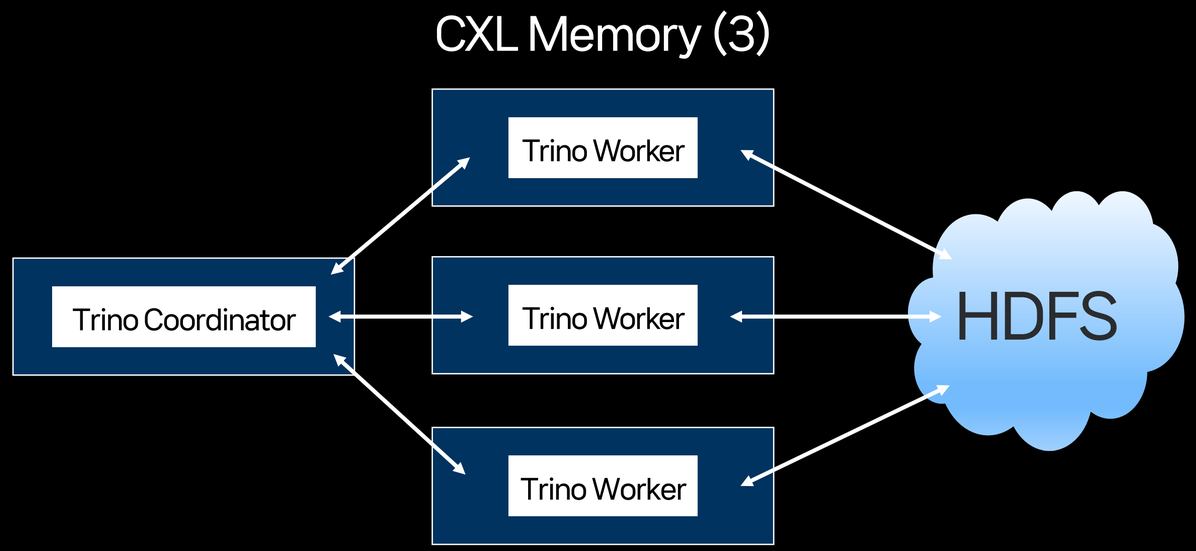
DRAM은 비싸고 현 아키텍처로는 늘리기 힘듬

DRAM 보다는 느리지만 저가의 메모리 사용 가능

- 메인 메모리 DRAM (256GB) + CXL 기반 저가 메모리 (2TB) 장착
- 시스템의 메모리 용량과 대역폭을 획기적으로 확장 가능
  - 2TB → JVM 1.6TB → 1.09TB 가용 메모리

5.1.1 CXL 기반 메모리 PoC – 테스트 환경

NAVER
DEVIEW
2023

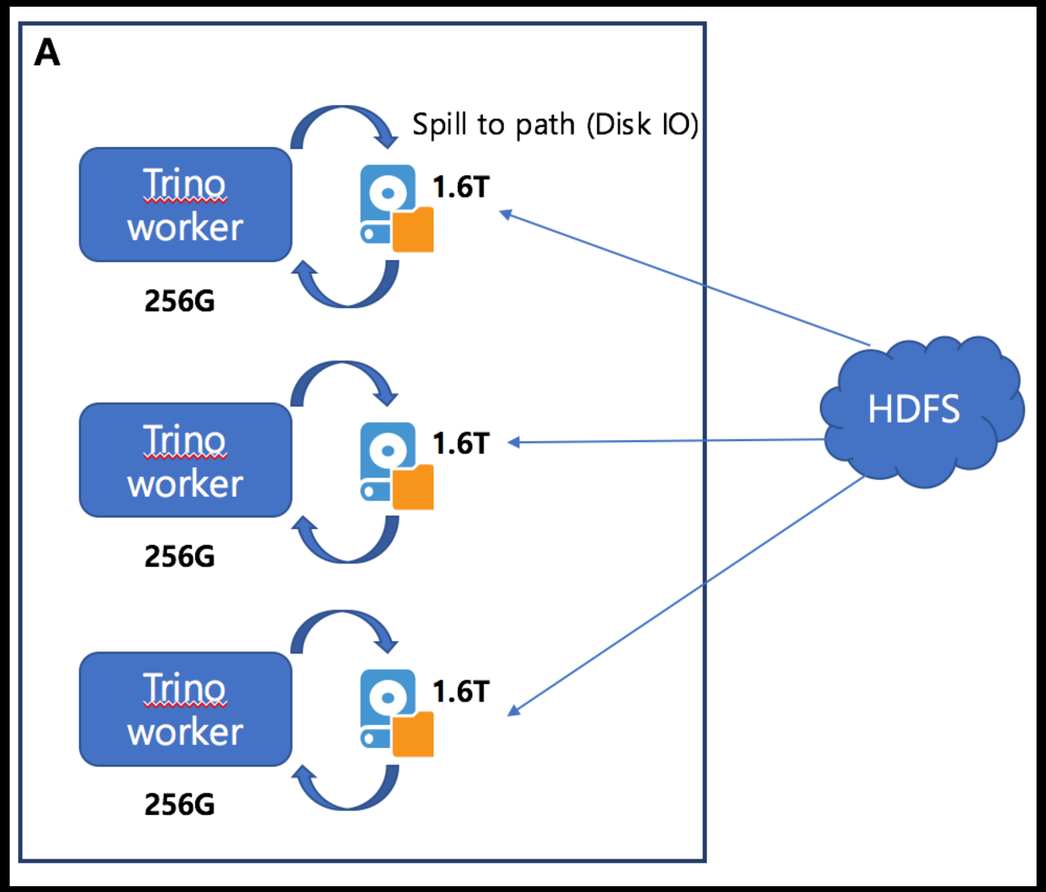


	기존 A 군	메모리 확장 B군
Specification	DRAM + Spill to Disk	CXL 저가 메모리
CPU	Icelake * 2ea	Icelake * 2ea
Memory	256GB	2TB

5.1.1 CXL 기반 메모리 PoC – 테스트 환경

NAVER
DEVIEW
2023

A군 : DRAM + Spill to Disk

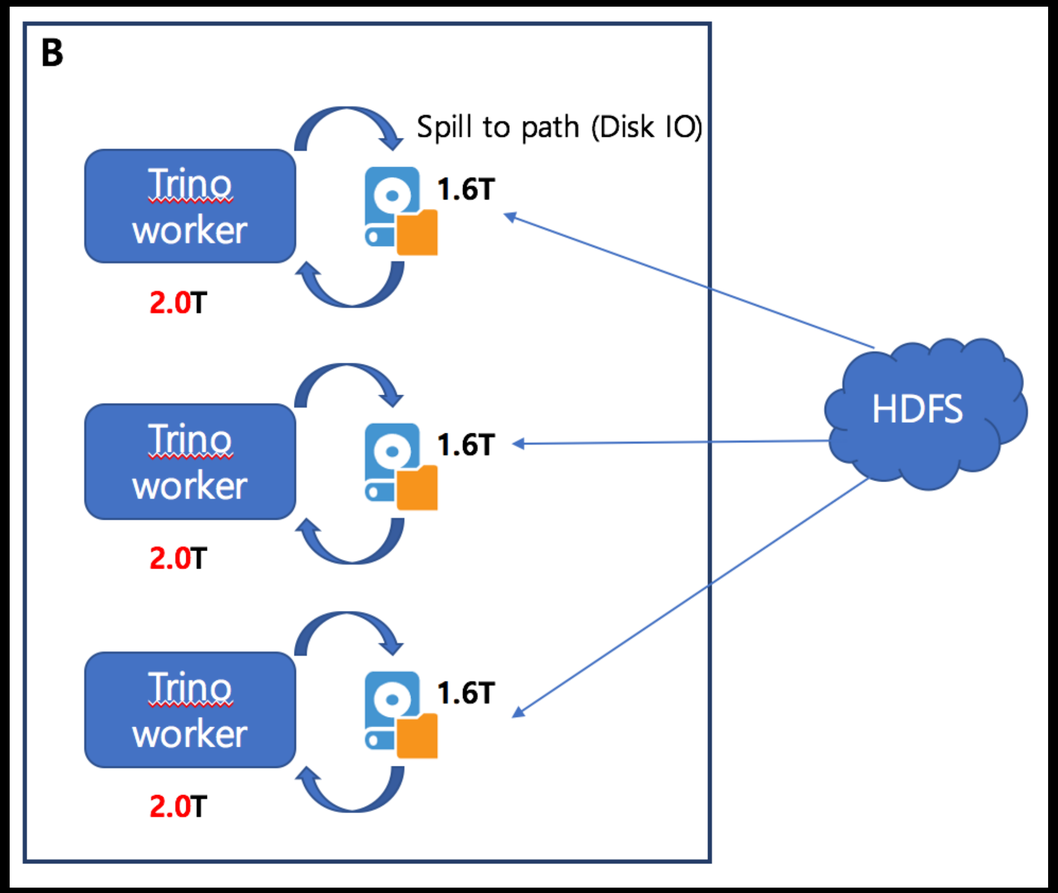


- 가용 메모리를 다 사용한 경우 Spill to Disk 옵션을 통해 disk에 정보 저장
- 메모리를 disk로 대체하며 쿼리 실패는 안 되지만 disk IO로 성능 크게 저하
- Total Memory : 256GB * 4 = 768 GB
- Total SSD : 1.6TB * 3 = 4.8TB

5.1.1 CXL 기반 메모리 PoC – 테스트 환경

NAVER DEVIEW 2023

B군 : CXL 저가 메모리



- CXL 저가 메모리를 이용하여 2TB 사용 가능
- 메모리 기반 컴퓨팅 자원 확보로 Trino 성능 향상 기대
- A군에 비해 메모리가 장비당 2TB 즉 2048 GB로 8배 확장
- Total Memory : 2T * 3 = 6 T
- Total SSD : 1.6 T * 3 = 4.8 T (비교군인 A와 동일)

5.1.2 CXL 기반 메모리 PoC – 결과

NAVER
DEVIEW
2023

저용량 데이터 처리율에서 8% 성능 하락

	A군	B군
Peak Total Memory	357G	351G
Spill to Disk Data	0	0
Elapsed Time	3.52m	3.82m

- Spill to Disk, Disk IO가 발생하지 않는 범위의 저용량 쿼리 처리 요청
- DRAM만 사용할 때 대비 PMEM으로 메모리를 확장해서 사용할 경우 쿼리가 3.52m → 3.82m 로 대략 8% 정도 성능 저하
- CXL 저가 메모리는 DRAM에 비해 성능 저하

5.1.2 CXL 기반 메모리 PoC – 결과

NAVER
DEVIEW
2023

대용량 데이터 처리율에서 28.3%, 58.1% 성능 향상

	A군	B군
Peak Total Memory	377G	709G
Spill to Disk Data	349G	0
Elapsed Time	13.11m	9.40m

2배 데이터 크기

- A군의 경우 Memory가 부족하여 Spill to Disk 으로 중간 데이터를 생성했기에 쿼리 연산 및 Disk IO 발생
- 메모리만 사용한 B군에 비해 13.10m → 9.40m 으로 28.3% 감소

	A군	B군
Peak Total Memory	378G	1.55T
Spill to Disk Data	1.17T	0
Elapsed Time	44.17m	18.50m

4배 데이터 크기

- 데이터가 2배 이상 증가해도 Spill이 없어 성능 차이가 크게 발생
- 44.18m → 18.50m 으로 쿼리 실행 시간 절반 이상 58.1% 감소

5.1.2 CXL 기반 메모리 PoC – 결과

NAVER
DEVIEW
2023

고가인 DRAM 보다 저가인 CXL 메모리 구성의 경쟁력 확인

- 확장된 CXL 저가 메모리는 spilled data가 전혀 발생하지 않음. 따라서 온전히 메모리 기반 쿼리 분석 가능
- 메모리 기반 분석에 최적화된 CQuery Trino 환경에서 대용량 데이터 처리 시 가용 메모리를 확장하면 Disk IO까지 사용하는 빈도가 확연히 줄어 빠른 쿼리 성능을 기대해 볼 수 있음
- 메모리를 최대로 끌어 올릴 수 있는 하드웨어 스펙이 CQuery Trino 클러스터에서는 이점이 있을 수 있음
- CQuery 사용패턴은 메모리에서 읽는 것이 대부분이니 DRAM과 CXL 메모리 성능이 크게 차이가 나지 않음. 따라서 고가인 DRAM보다는 저가인 CXL 메모리로 구성하는 것이 가격 경쟁력이 있음
- 향후 삼성 CXL, AMD Genoa 등 도입 고려

5.2 Workload Specific Cluster – Gateway

NAVER DEVIEW 2023

Gateway를 이용해 Query 특성으로 Cluster 자동 전환

- Scheduled Cluster, Ad-hoc Cluster, Short-term Cluster
- Testbed Cluster 임시 클러스터 생성 반납 프로세스 구현
  TTL 후 자동 소멸 기능 고려
- 반드시 성공을 보장 받아야하는 쿼리는 클러스터 독립
- 네이버 서비스에 바로 사용되는 경우 지표 추출 시간 보장
  - 예) 검색 추천, 모델링 등에 사용되는 데이터

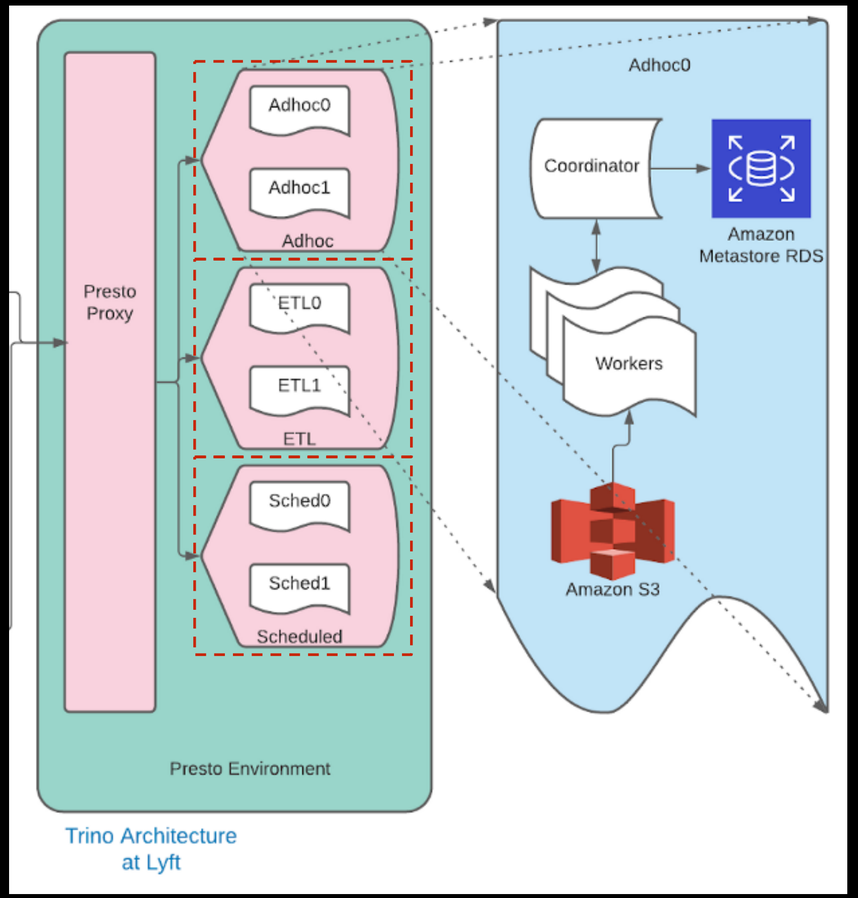


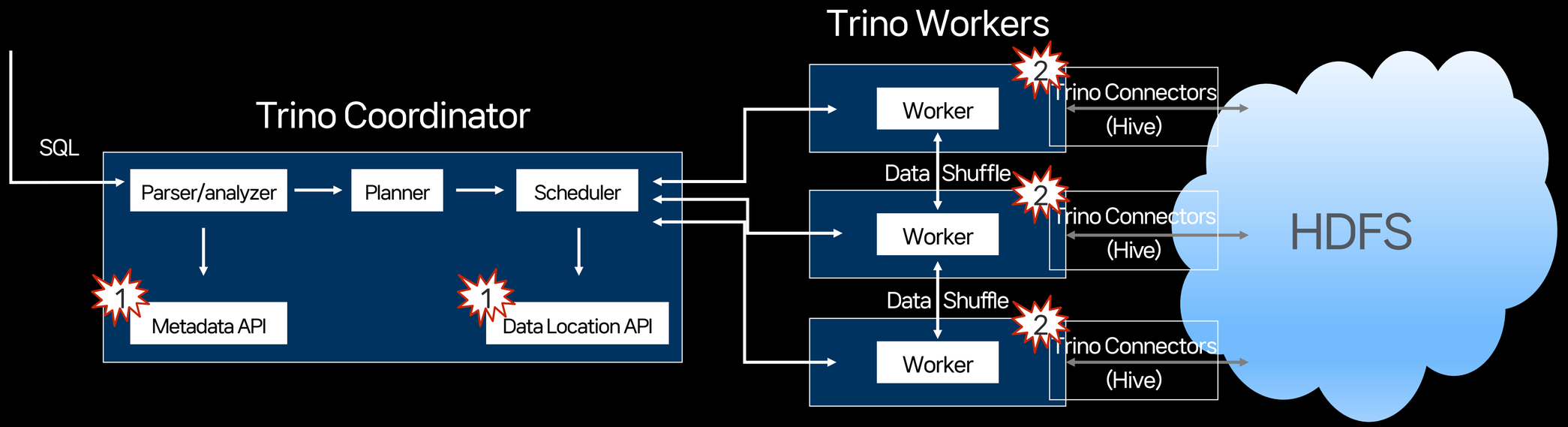
그림8 - https://eng.lyft.com/trino-open-source-infrastructure-upgrading-at-lyft-83f26b099fa

5.3 Iceberg

NAVER
DEVIEW
2023

Hive Table Format → Iceberg Format

- **PB 단위를 이미 넘어섰고 갈수록 증가하는 데이터를 다루는 AIDA CQuery 입장에서는 Hive MetaStore의 RDBMS, Hadoop NameNode 병목이 가장 큰 문제**



1. Parser, Query planning, CBO, Scheduler → 매번 HMS 서버에서 table schema/statistics 정보 요청
   - RDBMS(mysql)로 구성되어 있어서 HDFS 처럼 scale out이 자유롭지 못한 병목지점
2. Directory file listing HDFS NameNode / Access File Data Node
   - Metastore에서는 파티션 디렉토리 정보 까지만 전달, 실제 파일은 Name Node에서 listing
   - Name Node도 한 개만 존재, 파일이 정말 많으면 데이터 조회보다 directory안에 파일 리스팅 하는데 시간이 걸림

5.3 Iceberg

NAVER
DEVIEW
2023

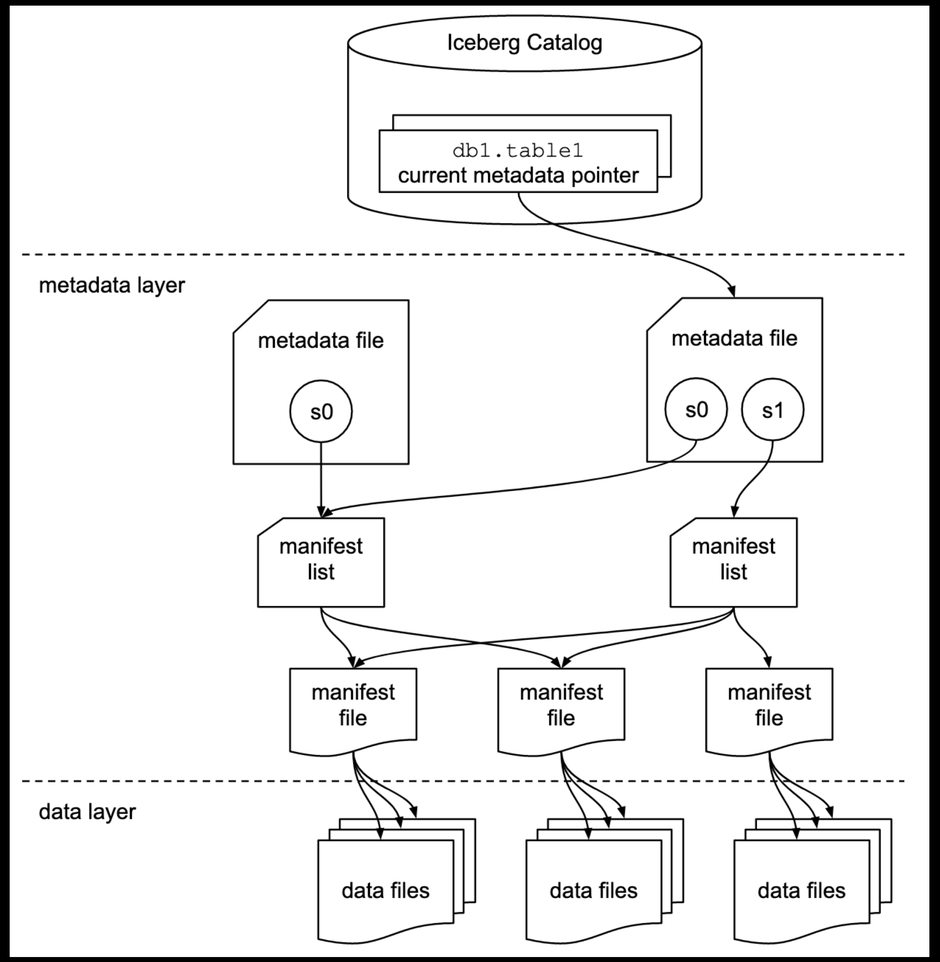


그림9 - https://iceberg.apache.org/spec/

- **대규모 데이터 분석을 위해** Netflix **에서 개발한 테이블 형식**
  - Schema evolution, ACID, scan planning and file filtering 등 제공
- Hive ORC**는 스키마 확장성을 미지원 문제** → Schema Evolution
- HDFS**를 사용하면서** Update**를 지원하지 않아서 오는** Transaction **처리 문제** → ACID **지원**
- https://trino.io/episodes/40.html

6. 우리 함께 해요. (Contribute)

- Open Source Contributions
- 채용

6.1 기능 추가/성능 개선

NAVER
DEVIEW
2023

Trino

- https://github.com/trinodb/trino/pull/10561 delegation 토큰이 자동 재갱신 기능 추가
- https://github.com/trinodb/trino/pull/15389 Resource Group DB 읽는 주기 변경 기능 추가

Trino Go Client

- https://github.com/trinodb/trino-go-client/pull/14 실행 중 현재 상태 정보 획득 방법 추가

6.2 버그 수정

NAVER
DEVIEW
2023

Trino

- https://github.com/trinodb/trino/pull/13304 ENUM 타입 conversion 버그
- https://github.com/trinodb/trino/pull/15380 OOM 발생하는 preventive GC 사용 안하도록 설정

Trino Go Client

- https://github.com/trinodb/trino-go-client/pull/5 hidden arg 버그 1
- https://github.com/trinodb/trino-go-client/pull/7 hidden arg 버그 2
- https://github.com/trinodb/trino-go-client/pull/38 커스텀 Scanner 객체가 초기화 버그
- https://github.com/trinodb/trino-go-client/pull/61 json 형태를 잘 못 읽는 버그

6.3 채용 관련

NAVER
DEVIEW
2023

AI & Data Platform 소개

https://naver-career.gitbook.io/kr/service/search/ai-and-data-platform

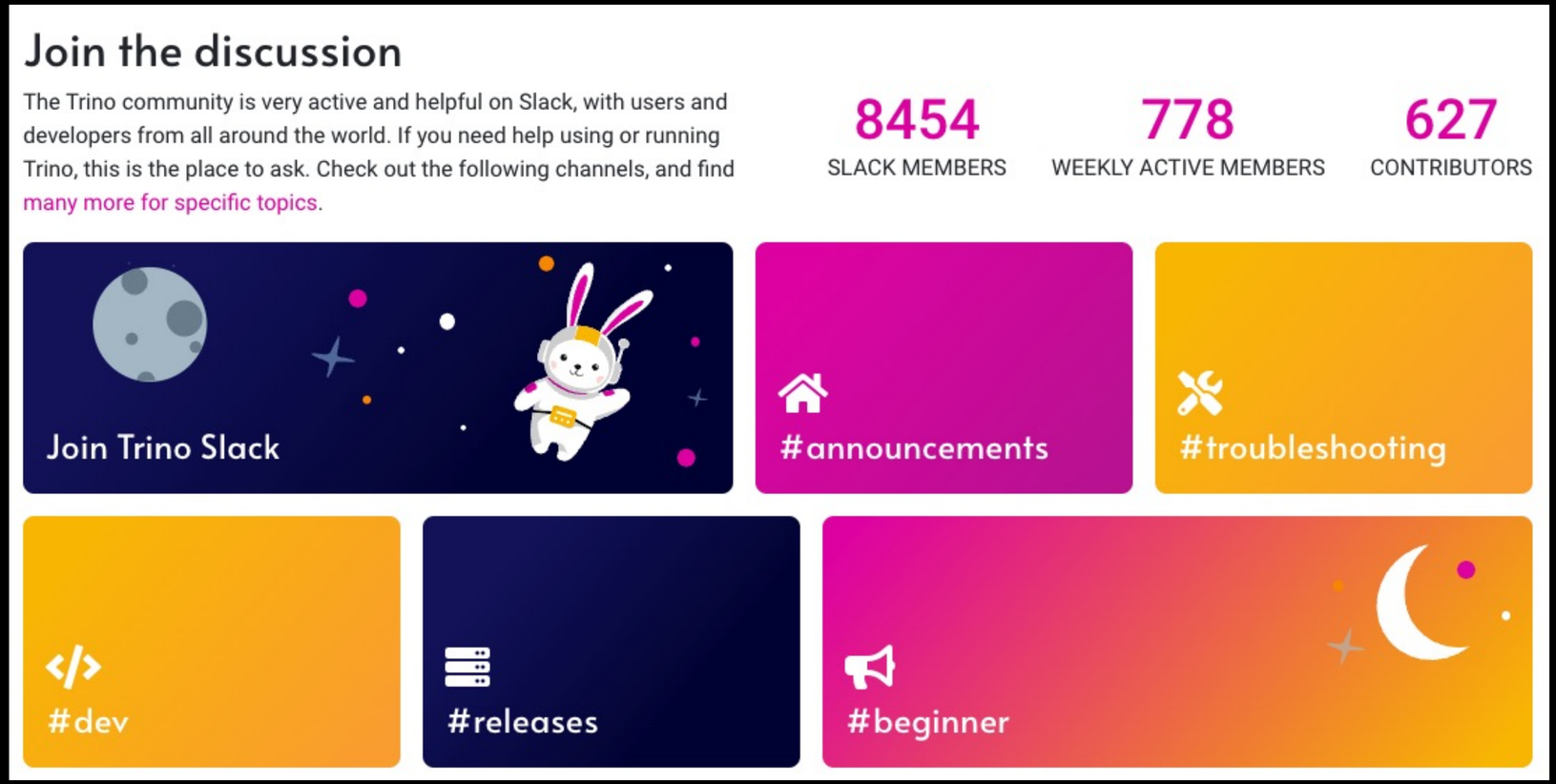
경력 등록

https://d2.naver.com/news/7591059

Trino 관련 문의

Slack을 통해서 커뮤니티 제공

- 기능 개선 / 버그 / 향후 개발 방향성 등 모든 Trino 관련 문의
- 한인 채널 별도 지원 (general-ko)
- https://trino.io/slack.html



참고자료

- 그림1 - https://naver-career.gitbook.io/kr/service/search/ai-and-data-platform
- 그림2 - https://deview.kr/2021/sessions/515
- 그림3 - https://blog.treasuredata.com/blog/2015/03/20/presto-versus-hive/
- 그림4 - https://www.youtube.com/watch?v=Pu80FkBRP-k&t=2086s
- 그림5 - https://www.oreilly.com/library/view/trino-the-definitive/9781098107703/ch04.html
- 그림6 – Presto(Trino) SQL On Everything IEEE Paper
- 그림7 - https://www.aitimes.kr/news/articleView.html?idxno=20994
- 그림8 - https://eng.lyft.com/trino-open-source-infrastructure-upgrading-at-lyft-83f26b099fa
- 그림9 - https://iceberg.apache.org/spec/

Thank You