DEVSISTERS

RDS online alter부터 CPU 100% 장애까지

데브시스터즈 쿠키런: 오븐브레이크 서버 프로그래머
황재영

RDS online alter

쿠키런 : 오븐브레이크

22년 8월 신규 기능. **패키지 매출 랭킹**

요구 사항:

현재 오픈되어 있는 패키지 중

최근 n시간동안 패키지의 판매 매출 (**판매 가격 * 판매 수량**) 순위를 구하는 것

RDS online alter

요구 사항:

최근 n시간동안 각 패키지의 판매 매출 **(판매 가격 * 판매 수량)** 순위를 구하는 것

쿠키런: 오븐브레이크의 인앱 구매 기록은 RDS Aurora로 관리

총 4개의 스토어별 별도의 테이블

- AppStore
- PlayStore
- OneStore
- GalaxyStore

RDS online alter

문제: 일부 스토어 테이블에 **구매 시점** 정보가 없다

- 삽입 시점(insert_dt)가 있는데, 정확히는 구매 시점(purchase_dt)와는 다름
- 쿠키런: 오븐브레이크는 구매 시점에 따라 다른 패키지로 고려될 수 있는 상품이 있음

따라서 위 테이블에 구매 시점(purchase_dt) column을 추가해야할 필요성 대두

```
ALTER TABLE $TABLE_NAME
ADD COLUMN purchase_dt datetime NOT NULL DEFAULT 0,
ADD KEY purchase_dt (purchase_dt);
```

RDS online alter

진짜 문제: alter cost가 너무 크다

- 너무 방대한 테이블. 지난 n년간의 구매 영수증이 모두 존재
- 단순한 alter table로는 table lock이 상당히 오래 잡힐 것으로 예상
- 쿠키런 오븐브레이크의 주기적인 점검(2시간) 내에 진행하기도 어려운 상황

이 기능을 위해서 별도의 점검 시간을 거는 것은 유저 경험적으로 별로일 수 있다고 판단.

Online alter table 플랜을 준비하기 시작함

RDS online alter

Plan 1. Inplace Algorithm

그래도 혹시 inplace 알고리즘으로 되지 않을까?
Staging 서버의 clone cluster를 만들어 테스트

```
ALTER TABLE $TABLE_NAME
ADD COLUMN purchase_dt datetime NOT NULL DEFAULT 0,
ADD KEY purchase_dt (purchase_dt),
ALGORITHM=inplace, LOCK=none;
```

RDS online alter

Plan 1. Inplace Algorithm

```
mysql> ALTER TABLE $TABLE_NAME
    -> ADD COLUMN purchase_dt datetime NOT NULL DEFAULT 0,
    -> ADD KEY purchase_dt (purchase_dt),
    -> ALGORITHM=inplace, LOCK=none;
ERROR 1062 (23000): Duplicate entry '73817' for key 'PRIMARY'
```

왜 안 되지? 일단 다시 해보자

```
mysql> ALTER TABLE $TABLE_NAME
    -> ADD COLUMN purchase_dt datetime NOT NULL DEFAULT 0,
    -> ADD KEY purchase_dt (purchase_dt),
    -> ALGORITHM=inplace, LOCK=none;
Query OK, 0 rows affected (2.72 sec) Records: 0 Duplicates: 0 Warnings: 0
```

왜 되지? 무슨 에러였던 걸까

RDS online alter

Plan 1. Inplace Algorithm

MySQL doc. 15.12.8 Online DDL Limitations

…

When the DML operations are applied, it is possible to encounter a duplicate key entry error (ERROR 1062 (23000): Duplicate entry), even if the duplicate entry is only temporary and would be reverted by a later entry in the online log.

…

online ddl의 한계였던 것

production 환경에서는 더 발생할 가능성이 높으므로 이 방안은 폐기

RDS online alter

Plan 2. Percona Toolkit

online alter 사례를 찾다보니 percona toolkit이라는 툴을 발견

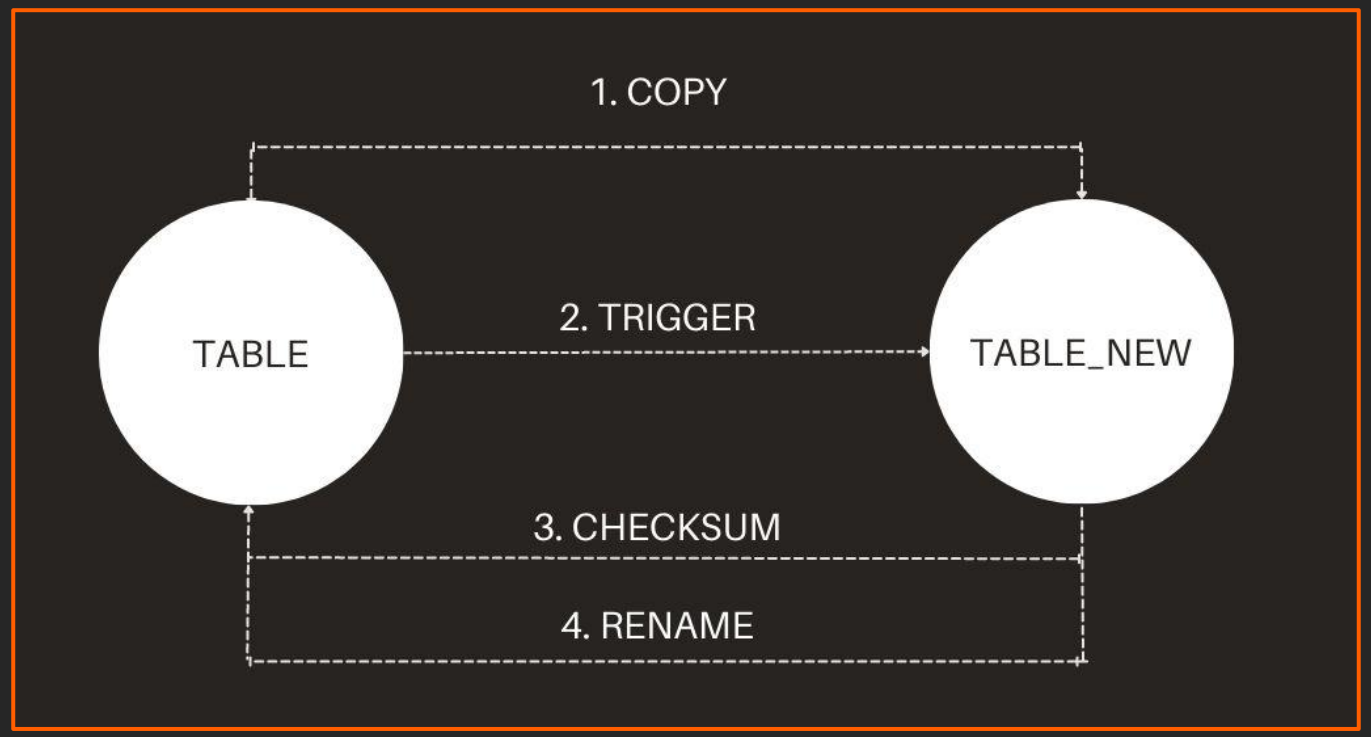
https://docs.percona.com/percona-toolkit/pt-online-schema-change.html

***pt-online-schema-change** - ALTER tables without locking them.*

어떤 원리로 작동하는 것일까?

RDS online alter

Plan 2. Percona Toolkit



몇 가지 제한점이 있지만, 현재 목적에서는 특별히 걸리는 부분이 없었다
시도해보는 것으로 결정하였다

RDS online alter

Plan 2. Percona Toolkit

실행 스텝

1. Staging RDS Clone (기본 검증)
2. Production RDS Clone (production RDS 적용 검증)
3. Staging RDS (적용과 service 병행 테스트)
4. Production RDS (실제 적용)

RDS online alter

Plan 2. Percona Toolkit

```
> pt-online-schema-change --host ${DBHOST} --user ${USERNAME} --ask-pass --dry-run
--alter 'ADD COLUMN purchase_dt datetime NOT NULL DEFAULT 0, ADD KEY purchase_dt (purchase_dt)'
D=$DB_NAME,t=$TABLE_NAME

Enter MySQL password:
Operation, tries, wait:
  analyze_table, 10, 1
  copy_rows, 10, 0.25
  create_triggers, 10, 1
  drop_triggers, 10, 1
  swap_tables, 10, 1
  update_foreign_keys, 10, 1
Starting a dry run.  `DB_NAME`.`TABLE_NAME` will not be altered.  Specify --execute instead of
--dry-run to alter the table.
```

RDS online alter

Plan 2. Percona Toolkit

```
Creating new table...
Created new table DB_NAME._TABLE_NAME_new OK.
Altering new table...
Altered `DB_NAME`.`_TABLE_NAME_new` OK.
Not creating triggers because this is a dry run.
Not copying rows because this is a dry run.
Not swapping tables because this is a dry run.
Not dropping old table because this is a dry run.
Not dropping triggers because this is a dry run.
2022-07-13T11:52:58 Dropping new table...
2022-07-13T11:52:58 Dropped new table OK.
Dry run complete.  `DB_NAME`.`TABLE_NAME` was not altered.
```

dry run에서는 생성도, 복사도, 스왑도 하지 않는다
real run으로 넘어가보자

RDS online alter

Plan 2. Percona Toolkit

```
> pt-online-schema-change --host ${DBHOST} --user ${USERNAME} --ask-pass --execute
--alter 'ADD COLUMN purchase_dt datetime NOT NULL DEFAULT 0, ADD KEY purchase_dt (purchase_dt)'
D=$DB_NAME,t=$TABLE_NAME

Enter MySQL password:
Operation, tries, wait:
  analyze_table, 10, 1
  copy_rows, 10, 0.25
  create_triggers, 10, 1
  drop_triggers, 10, 1
  swap_tables, 10, 1
  update_foreign_keys, 10, 1
```

RDS online alter

Plan 2. Percona Toolkit

```
Altering `DB_NAME`.`TABLE_NAME`...
Creating new table...
Created new table DB_NAME._TABLE_NAME_new OK.
Altering new table...
Altered `DB_NAME`.`_TABLE_NAME_new` OK.
2022-07-13T11:55:11 Creating triggers...
2022-07-13T11:55:11 Created triggers OK.
2022-07-13T11:55:11 Copying approximately 9980 rows...
...
2022-07-13T11:55:34 Copied rows OK.
2022-07-13T11:55:34 Analyzing new table...
2022-07-13T11:55:34 Swapping tables...
2022-07-13T11:55:34 Swapped original and new tables OK.
2022-07-13T11:55:34 Dropping old table...
2022-07-13T11:55:34 Dropped old table `DB_NAME`.`_TABLE_NAME_old` OK.
2022-07-13T11:55:34 Dropping triggers...
2022-07-13T11:55:34 Dropped triggers OK.
Successfully altered `DB_NAME`.`TABLE_NAME`.
```

row가 적은 stg라서 1분만에 완료

RDS online alter

Plan 2. Percona Toolkit

Staging RDS clone으로 기본적인 검증을 완료

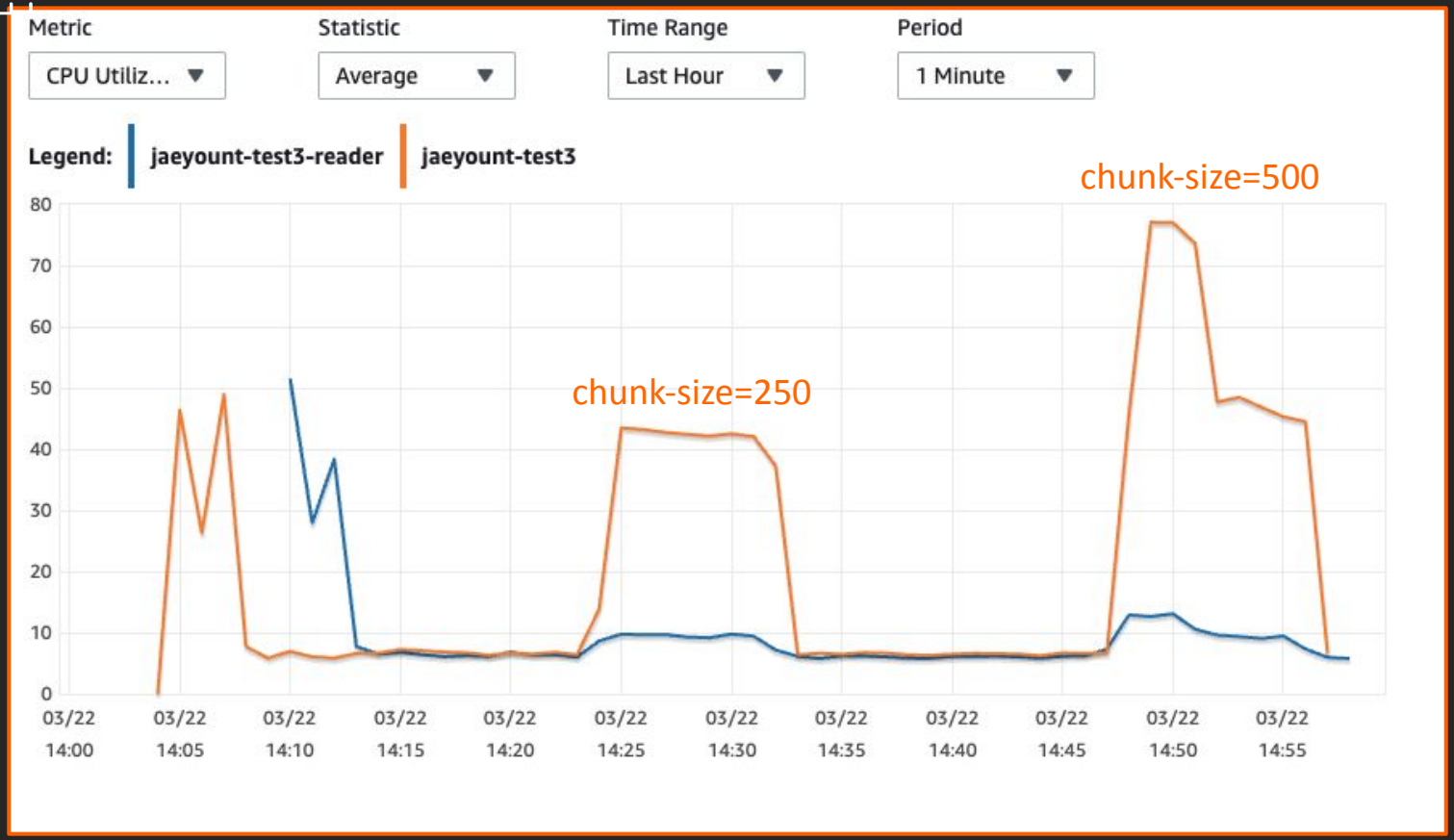
Production RDS clone에서도 테스트를 진행해보았다

1. Reader cluster의 유무 차이
2. Row 수 차이로 인한 실행 시간 차이
3. chunk-size 옵션 조절을 통한 CPU 조절

RDS online alter

Plan 2. Percona Toolkit

해당 RDS는 평소에 CPU를 많이 쓰지 않아서 50% 정도의 CPU를 유지하는 것이 목표



RDS online alter

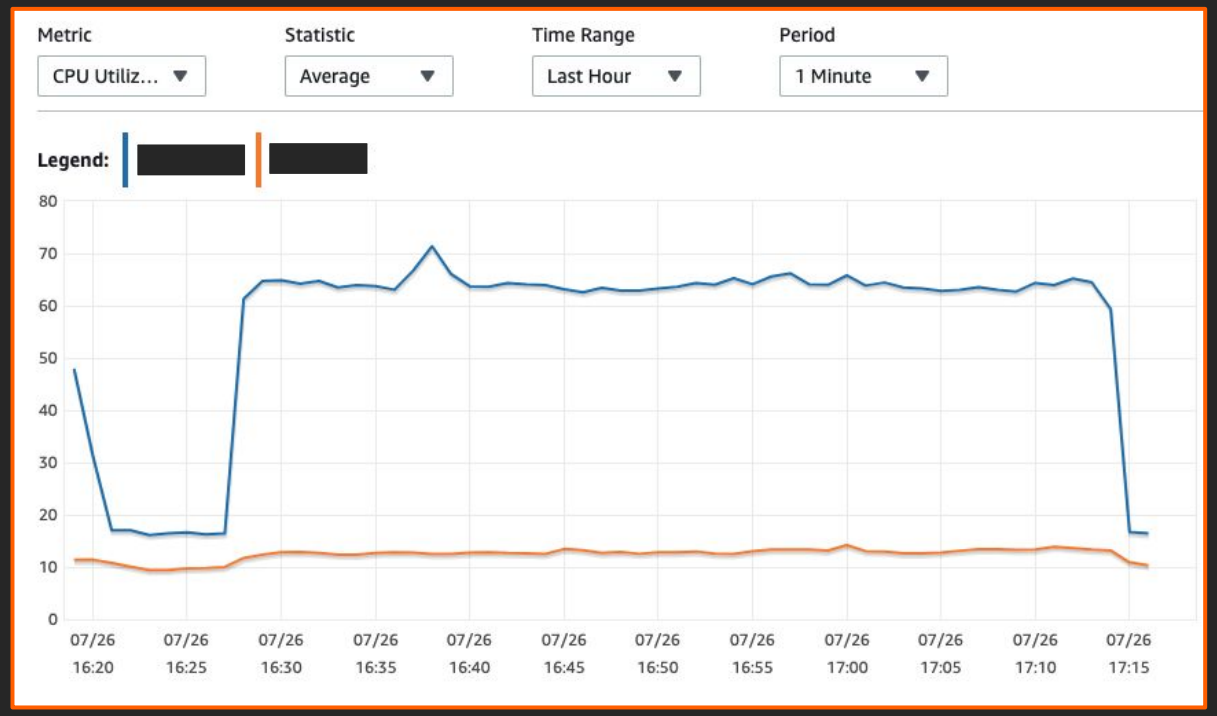
Plan 2. Percona Toolkit

Staging RDS에서 진행하면서, 실제 인게임 구매와 병행하여 문제가 없는지 체크

-> 설명할 게 없어서 머쓱할 정도로 문제 없이 진행

Production RDS에 적용

- > 혹시 모를 상황에 대비하여 snapshot을 준비하고 진행



RDS CPU

그리고 업데이트 당일

CPU가 좀 높아졌네?

쿼리가 좀 무겁기는 했지. 최적화를 좀 해봐야겠다

RDS CPU 100%

서버A: 다른 서버 스펙에 자그마한 버그 있어요. 새로운 서버로 롤링 업데이트하겠습니다

기획B: 리소스에서 수정할 수 있게 해서 새로운 리소스로 업로드하겠습니다

그래프야 어디가니?

RDS CPU 100%

RDS CPU 100%로 인한 정상적인 게임 접속 불가
원인 파악 이전에 일단 긴급하게 서버 점검을 공지하고 서버를 닫고 분석을 시작

업데이트 전후로 RDS 사용 패턴에서 유일하게 바뀐 패키지 매출 랭킹을 중점적으로 확인

그리고 확인된 문제

```
SELECT 상품정보, COUNT(*) As count
FROM $TABLE_NAME
WHERE
(상품정보 = 'aaa' and FROM_UNIXTIME(111) < purchase_dt and purchase_dt <= FROM_UNIXTIME(222)) and
(상품정보 = 'bbb' and FROM_UNIXTIME(111) < purchase_dt and purchase_dt <= FROM_UNIXTIME(222))
...
```

일단 해당 기능을 OFF할 수 있는 기능을 만들어서 서버를 배포. 1차적으로 문제 해결

RDS CPU 100%

```
SELECT 상품정보, COUNT(*) As count
FROM $TABLE_NAME
WHERE
(상품정보 = 'aaa' and FROM_UNIXTIME(111) < purchase_dt and purchase_dt <= FROM_UNIXTIME(222)) and
(상품정보 = 'bbb' and FROM_UNIXTIME(111) < purchase_dt and purchase_dt <= FROM_UNIXTIME(222))
...
```

1. 최적화 0%의 쿼리. 쿼리 자체가 매우 무겁다
2. 서버 단위의 캐시. 새로운 서버가 뜰 때마다 해당 쿼리를 RDS로 요청
3. 캐시 키가 리소스에 따라 변동. 새로운 리소스가 나오면 이전 캐시는 사실상 Invalidate

그리고 해당 기능을 낮은 부하로 사용할 방법을 고민

- 쿼리 최적화
- 캐시 최적화

쿼리 최적화

```
SELECT 상품정보, COUNT(*) As count
FROM $TABLE_NAME
WHERE
(상품정보 = 'aaa' and FROM_UNIXTIME(111) < purchase_dt and purchase_dt <= FROM_UNIXTIME(222)) and
(상품정보 = 'bbb' and FROM_UNIXTIME(111) < purchase_dt and purchase_dt <= FROM_UNIXTIME(222))
...
```

```
CREATE INDEX $INDEX_NAME
ON $TABLE_NAME(상품정보,purchase_dt);
```

1. (상품정보, purchase_dt) 인덱스 추가
   -> Clone cluster를 만들어 테스트. 효과가 10% 정도로 미미해서 폐기

쿼리 최적화

```
SELECT 상품정보, COUNT(*) As count
FROM $TABLE_NAME
WHERE
(상품정보 = 'aaa' and FROM_UNIXTIME(111) < purchase_dt and purchase_dt <= FROM_UNIXTIME(222)) and
(상품정보 = 'bbb' and FROM_UNIXTIME(111) < purchase_dt and purchase_dt <= FROM_UNIXTIME(222))
...
```

```
SELECT 상품정보, COUNT(*) As count
FROM DB_NAME
WHERE
((상품정보 = 'aaa' or 상품정보 = 'bbb') and FROM_UNIXTIME(111) < purchase_dt) and
...
```

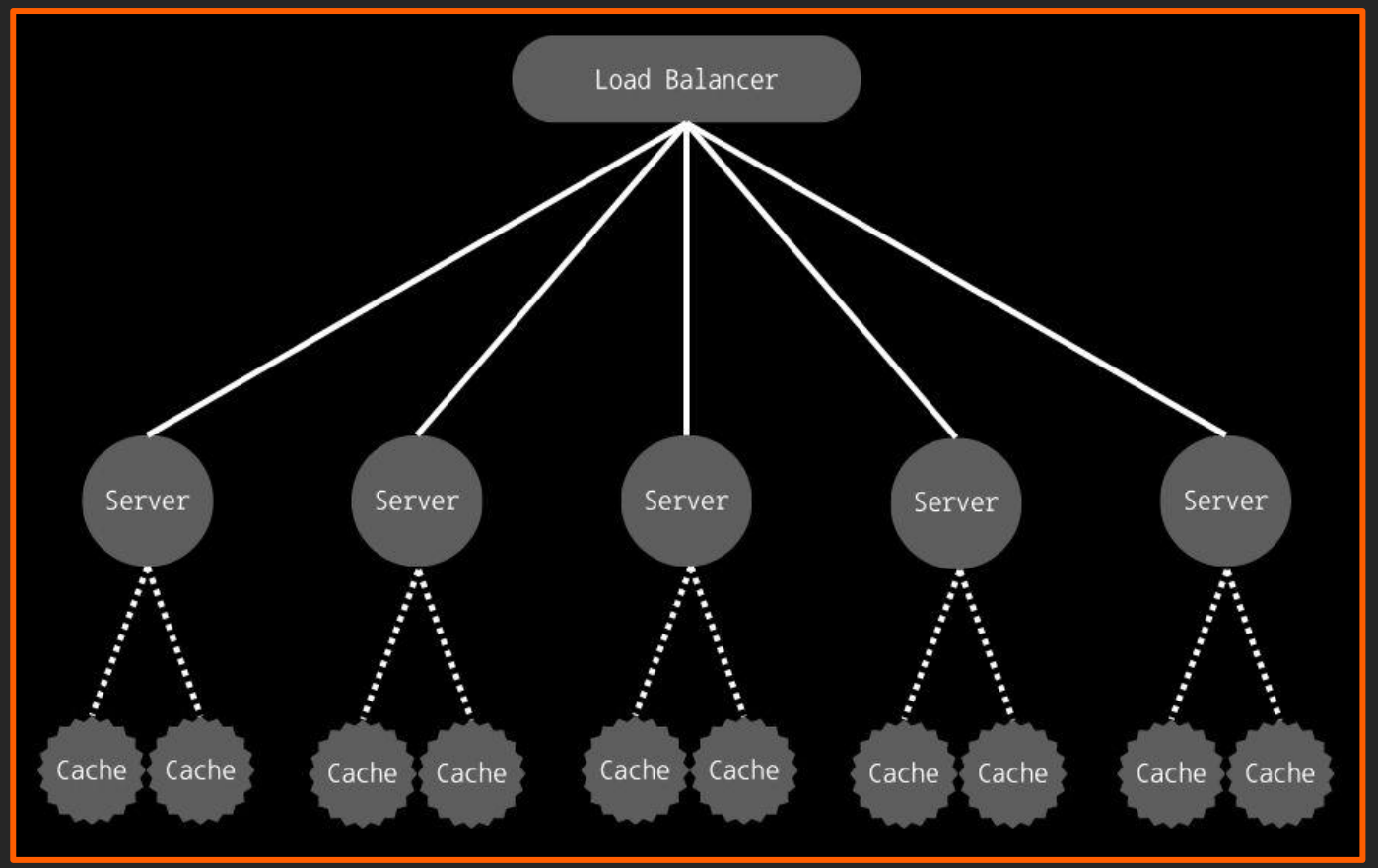
2. 쿼리를 상품 단위로 묶고, 종료 시간 condition 제거

   -> Clone cluster를 만들어 테스트. 효과가 크지는 않았지만, 디메리트가 없어서 적용

캐시 최적화

재활용되지 않는 캐싱 결과

- 수평확장된 서버마다의 인메모리 캐시
- 캐시 키가 리소스에 영향받는 캐시



캐시 최적화

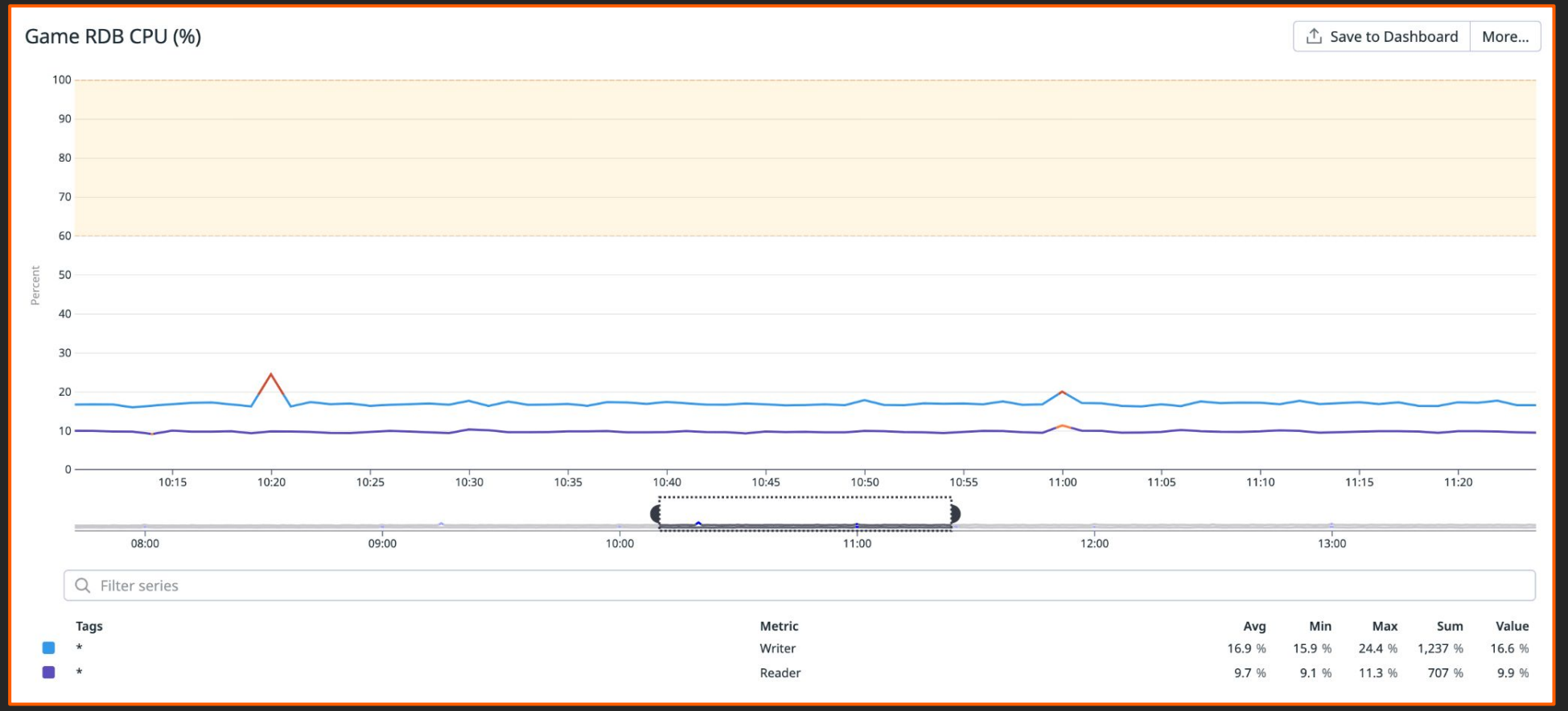
서버 간 캐시 공유, 리소스와 무관한 캐시 사용

주기적으로 요청을 처리하는 별도의 서버(스케쥴러 서버)를 사용해서 단 하나의 캐시 운용

1. 서버는 쿼리 대신 스케쥴러 서버로의 요청을 큐잉
2. 스케쥴러 서버는 주기적으로 큐에서 요청을 꺼내서 합친 후, 결과를 계산

   -> 부하가 있는 쿼리의 실행 자체를 일정 시간동안 1번으로 줄임
3. 스케쥴러 서버가 계산한 결과를 원격 저장소에 저장(캐싱)
4. 서버는 쿼리 대신 elasticache에서 저장된 결과를 조회

RDS CPU 100%



2가지 개선 점을 모두 적용한 이후
드디어 기능을 정상적으로 사용하는데 성공

RDS CPU 100%

배운점

1. percona toolkit은 production 환경에서도 적용할만 하다
2. ~~online alter로 시간 아낄 필요가 없었다~~
3. 새로운 쿼리를 도입할 때는 벤치마킹부터 진행해보고 상황에 따라 적절한 최적화 방법을 적용해야 한다

감사합니다