Redis Persistence Introduction

Persistence Introduction

두 가지 방식

- 디스크 쓰기는 AOF와 RDB 두 가지 방식을 제공합니다.
  AOF는 Append Only File로 명령이 실행될때 마다 기록되는 파일입니다. 그러므로 데이터의 손실이 거의 없습니다.
  RDB 파일은 특정한 간격마다 메모리에 있는 레디스 데이터 전체를 디스크에 씁니다. 특정 시점 마다 백업을 받을 때 사용합니다.
  AOF 방식부터 알아보도록 하겠습니다.
  RDB 방식을 먼저 보려면 여기를 클릭하세요.

AOF(Append Only File)

AOF 방식 설명

- AOF 파일은 default로 appendonly.aof 파일에 기록됩니다. 입력/수정/삭제 명령이 실행될 때 마다 기록됩니다. 조회 명령은 기록되지 않습니다. 기록 시점에 대한 parameter는 여기를 보세요.
- AOF는 계속 추가하면서 기록됩니다. 하지만 특정 시점에 데이터 전체를 다시 쓰는 기능이 있습니다.
  왜 이런 기능이 필요할까요? AOF는 계속 추가하면서 기록되기 때문에 파일 사이즈가 계속 커집니다.
  너무 커지면 OS 파일 사이즈 제한에 걸려서 기록이 중단될 수도 있고, 레디스 서버 시작 시 로드 시간이 많이 걸릴 수 있습니다.
  그럼 rewrite 하면 파일 사이즈가 작아 질까요? 작아집니다.
  예를 들어 SET 명령이 key는 같고 값을 다른 조건에서 5번 수행되었다고 하면, 메모리에는 마지막 수행된 값만 남아있습니다. 하지만 AOF에는 5번 모두 남아 있습니다. 다른 대표적인 명령은 INCR 입니다.
  INCR key가 1000번 수행되었다고 하면 메모리에는 key 1000 이 하나만 남아 있습니다. AOF에는 INCR key 명령이 1000번 기록되어 있습니다. Rewrite를 수행하면 이전 기록은 모두 사라지고 최종 데이터가 기록됩니다. 즉, INCR key 1000번은 SET key 1000 하나로 기록됩니다.
  Rewrite에 대한 parameter는 여기를 보세요.
- AOF 파일은 text 파일이므로 edit 가능합니다. 실수로 FLUSHALL 명령으로 메모리에 있는 데이터 전체를 날렸을 경우, 즉시 레디스 서버를 shutdown하고, appendonly.aof 파일에서 FLUSHALL 명령을 제거 한 후 레디스를 다시 시작하면 데이터 손실없이 DB를 살릴 수 있습니다.
  AOF를 열어보면 마지막에 아래와 같이 3줄이 있을 것이다. 이것을 삭제하고 레디스 서버를 start하면 됩니다.

```
*1
$8
flushall
```

AOF 관련 redis.conf 파라미터

- appendonly yes 또는 no : yes는 AOF기능을 사용합니다. no는 사용하지 않습니다.
  레디스 서버 시작 시 이 값이 yes 일때만 AOF 파일을 읽어들입니다.
  no 이면 AOF 파일이 있어도 읽어들이지 않습니다.
- appendfilename "appendonly.aof" : AOF 파일명을 지정합니다.
  Path는 지정할 수 없습니다. Path는 working directory에 따릅니다.
- appendfsync : AOF에 기록하는 시점을 정합니다.
  - always : 명령 실행 시 마다 AOF에 기록합니다. 데이터 유실의 염려는 없으나, 성능이 매우 떨어집니다.
  - everysec : 1초마다 AOF에 기록합니다. 1초 사이 데이터 유실될 수 있으나, 성능에 거의 영향을 미치지 않으면서 데이터를 보존할 수 있어, 일반적으로 권장합니다. default 값입니다.
  - no : AOF에 기록하는 시점을 OS가 정합니다. 일반적으로 리눅스의 디스크 기록 간격은 30초입니다. 데이터가 유실될 수 있습니다.
- AOF rewrite 설정
  - auto-aof-rewrite-percentage 100 : AOF 파일 사이즈가 100% 이상 커지면 rewrite합니다.
    그러면 100%의 기준은 무었일까요? 처음에는 레디스 서버가 시작할 시점의 AOF 파일 사이즈를 기준으로 합니다.
    Rewrite를 하면 rewrite 후 파일 사이즈를 기준으로 계산합니다.
    그럼 레디스 서버 시작 시 AOF 파일 사이즈가 0이었다면 어떻게 계산할까요?
    그것은 아래 min-size를 기준으로 합니다.
    0으로 설정하면 rewrite를 하지 않습니다.
  - auto-aof-rewrite-min-size 64mb : AOF 파일 사이즈가 64mb 이하면 rewrite를 하지 않습니다.
    파일이 작을때 rewrite가 자주 발생하는 것을 막아줍니다.

AOF Rewrite 동작 순서

- Step 1: Child process를 fork() 한다.
- Step 2: Child process는 데이터를 새 AOF temp 파일에 쓴다.
- Step 3: 동시에 Parent process는 새로운 명령을 메모리 버퍼에 기록하면서 현재 AOF 파일에도 쓴다.
  이렇게 하는 이유는 rewrite 작업이 실패해도 데이터를 안전하게 보존하기 위해서 이다.
- Step 4: Child process가 fork()된 시점의 데이터 쓰기 완료(1차 쓰기)되면 Child process는 Parent Process에게 stop 시그널을 보낸다.
- Step 5: Parent Process는 stop 시그널을 받으면, Child process가 쓰는 동안 새로 발생한 데이터을 Child process에게 보내고, Child process는 이 데이터를 받아 2차로 임시 AOF 파일에 쓴다. 쓰기가 완료되면 파일을 닫고 Parent process에게 완료 시그널을 보낸다.
- Step 6: Parent process는 임시 AOF 파일을 열고, 2차 쓰기 동안 추가로 발생한 데이터를 AOF 파일에 쓴다. 이것을 3차 쓰기라고 하자.
- Step 7: 3차 쓰기가 완료되면 현 AOF 파일을 삭제하고 새 파일로 교체한 다음, 새 파일에 쓰기를 시작한다.

BGREWRITEAOF 명령 수행 정보 확인

- AOF Rewrite는 설정에 따라 자동으로 수행되기도 하지만 BGREWRITEAOF 명령으로 명시적으로 수행시킬 수 있다.
- Redis-cli 에서 수행시 나타나는 메지지:

  ```
  127.0.0.1:6379> BGREWRITEAOF
  Background append only file rewriting started
  ```

- info persistence 로 볼 수 있는 정보 : 필요한 데이터만 간추렸다.

  ```
  127.0.0.1:6379> info persistence
  aof_enabled:1   AOF 기능이 활성화 되어 있음: redis.conf 에 appendonly yes 일때
  aof_rewrite_in_progress:1   현재 rewrite가 진행중임을 나타낸다. 아닐때는 0
  aof_last_rewrite_time_sec:4   지난 번 rewrite 하는데 걸린 시간
  aof_current_rewrite_time_sec:6   새 파일에 rewrite 를 시작하고 현재까지 경과 시간
  ```

aof_last_bgrewrite_status:ok 지난 번 rewrite 상태
aof_current_size:125799616 현재 AOF 파일 사이즈
aof_base_size:119990550 베이스 AOF 파일 사이즈, base size와 current size를 비교해서 rewrite-percentage 값 이상이되면 자동으로 rewrite 한다.

- 레디스 서버 로그

25947:M 12:48:09.402 * Background append only file rewriting started by pid 26370
Step 1: parent process가 child process를 fork() 했다.
25947:M 12:48:15.525 * AOF rewrite child asks to stop sending diffs.
Step 4 (1차 쓰기 완료)
26370:C 12:48:15.525 * Parent agreed to stop sending diffs. Finalizing AOF...
26370:C 12:48:15.525 * Concatenating 0.41 MB of AOF diff received from parent.
Step 5: 1차 쓰기 중에 발생한 데이터를 다시 AOF에 추가한다. (2차 쓰기 완료)
26370:C 12:48:15.529 * SYNC append only file rewrite performed
26370:C 12:48:15.529 * AOF rewrite: 26 MB of memory used by copy-on-write
25947:M 12:48:15.577 * Background AOF rewrite terminated with success
25947:M 12:48:15.577 * Residual parent diff successfully flushed to the rewritten AOF (0.02 MB) Step 6: 3차 쓰기 완료
25947:M 12:48:15.577 * Background AOF rewrite finished successfully

RDB(snapshot)

RDB(snapshot) 방식 설명

- RDB는 특정 시점의 메모리에 있는 데이터 전체를 바이너리 파일로 저장하는 것이다.
  AOF 파일보다 사이즈가 작다. 따라서 로딩 속도가 AOF 보다 빠르다.
  일반적으로 AOF 가 10초 걸린다면, RDB는 7초 정도로 생각하면 될 것이다.
- 저장 시점을 정하는 redis.conf의 파라미터는 save 이다.
  SAVE 조건은 여러 개를 지정할 수 있고, 모두 or 조건이다. 즉 어느 것 하나라도 만족하면 저장한다.

save 900 1 900초(15분) 동안 1번 이상 key 변경이 발생하면 저장
save 300 10 300초(5분) 동안 10번 이상 key 변경이 발생하면 저장
save 60 10000 60초(1분) 동안 10000번 이상 key 변경이 발생하면 저장

- RDB를 저장하지 않으려면 redis.conf 에서 SAVE를 모두 주석 처리하면 된다.
- RDB 파일 이름 지정은 redis.conf에서 dbfilename dump.rdb 이다.
- 명령으로도 RDB 파일을 생성할 수 있다. BGSAVE 또는 SAVE 이다.

BGSAVE로 RDB 파일 생성

- 파일 쓰기 작업은 Child process가 생성되어 background로 실행되므로 쓰기 작업 중에도 레디스 서버는 클라이언트의 명령을 정상적으로 처리할 수 있다.
- 동작 순서
  - Step 1: Child process를 fork() 한다.
  - Step 2: Child process는 데이터를 새 RDB temp 파일에 쓴다.
  - Step 3: 쓰기가 끝나면 기존 파일을 지우고, 이름을 변경하다.
- Redis-cli 에서 수행시 나오는 메시지

127.0.0.1:6379> BGSAVE
Background saving started

- info persistence 로 볼 수 있는 정보 : 필요한 데이터만 간추렸다.

127.0.0.1:6379> info persistence
rdb_bgsave_in_progress:1 현재 Rewrite가 진행중임을 나타냄
rdb_last_save_time:1434352287 지난 번 RDB 파일 저장 시간
rdb_last_bgsave_status:ok 지난 번 RDB 파일 저장 성공 여부

rdb_last_bgsave_time_sec:7 지난 번 RDB 파일 저장에 걸린 시간
rdb_current_bgsave_time_sec:4 현재 RDB 파일 저장 시작하고 현재까지 경과 시간

- 레디스 서버 로그

  30854:M 16:10:16.006 * Background saving started by pid 1578
  Step 1: parent process가 child process를 fork() 했다.
  1578:C 16:10:23.782 * DB saved on disk Step 2 완료
  1578:C 16:10:23.782 * RDB: 278 MB of memory used by copy-on-write
  30854:M 16:10:23.830 * Background saving terminated with success

- Step 2에서 자식 프로세스가 임시 파일을 쓰는 중에 파일명은 temp-pid.rdb 이다. 이 경우는 temp-1578.rdb 이다.

SAVE로 RDB 파일 생성

- 파일 쓰기 작업을 레디스 Main process가 직접하므로 끝날때까지 클라이언트의 명령을 처리할 수 없다.
- 동작 순서
  - Step 1: Main process가 데이터를 새 RDB temp 파일에 쓴다.
  - Step 2: 쓰기가 끝나면 기존 파일을 지우고, 새 파일로 교체한다.
- Redis-cli 에서 수행시 나오는 메시지: 완료되면 수행 시간이 표시된다.

  127.0.0.1:6379> SAVE
  OK
  (4.62s)

- 레디스 서버 로그: 딱 한 줄 나온다.

  30854:M 15 Jun 16:11:27.468 * DB saved on disk

RDB 관련 redis.conf 파라미터

- **stop-writes-on-bgsave-error : yes or no**, default yes
  이 값이 yes 일때, 레디스는 RDB 파일을 디스크에 저장하다 실패하면, 모든 쓰기 요청을 거부한다.
  쓰기에 문제가 발생했으니, 빨리 조치를 취하라는 의미다. default는 yes이다.
  이 값을 no 로 설정하면, 디스크 저장에 실패하더라도, 레디스는 쓰기 요청을 포함한 모든 동작을 정상적으로 처리한다.
  서비스를 계속하는 것이 더 중요하고 모니터링이 잘되어 있다면 no로 설정하는 것이 좋다.
  info persistence 명령에서 rdb_last_bgsave_status 값이 err 면 실패한 것이다.
  디스크 쓰기에 실패하는 경우는 여유 공간이 부족하거나, 권한(permission) 부족, 디스크 물리적 오류 등이 있을 수 있다.
  이 파라미터는 save 이벤트에만 해당한다. BGSAVE 명령을 직접 입력했을 경우에는 해당하지 않는다.
  이 경우에 쓰기는 정상적으로 처리된다.

  이때 Redis-cli에서 set 명령을 실행하면 아래와 같은 메시지가 나타난다.

  (error) MISCONF Redis is configured to save RDB snapshots, but is currently not able to persist on disk.
  Commands that may modify the data set are disabled. Please check Redis logs for details about the error.

  쓰기가 실패한 경우 레디스 서버 로그 : 아래와 같은 메시지가 반복해서 나타난다.

  6930:M 19:16:07.036 * 1 changes in 60 seconds. Saving... redis.conf 에 save 60 1 로 설정함
  6930:M 19:16:07.038 * Background saving started by pid 6957 Child process를 fork() 함
  6957:C 19:16:07.039 # Failed opening .rdb for saving: Permission denied 권한 부족으로 쓰기 실패
  6930:M 19:16:07.140 # Background saving error

  info persistence로 알 수 있는 에러 정보 : 필요한 것만 간추렸다.

이 **err**가 save 이벤트에 의한 실패인지, BGSAVE 명령에 의한 실패인지는 구분할 수 없다.

```
127.0.0.1:6379> info persistence
rdb_last_bgsave_status:err
```

이때 주의할 점은 서버 터미널(daemonize no)에서 ctrl+C를 입력해도 서버가 down되지 않고, 아래와 같은 메시지를 내보낸다.

```
^C6930:signal-handler (1434363420) Received SIGINT scheduling shutdown...
6930:M 19:17:00.701 # User requested shutdown...
6930:M 19:17:00.701 * Saving the final RDB snapshot before exiting.
6930:M 19:17:00.701 # Failed opening .rdb for saving: Permission denied
6930:M 19:17:00.701 # Error trying to save the DB, can't exit.
6930:M 19:17:00.701 # SIGTERM received but errors trying to shut down the server, check the logs for more information
```

Redis-cli 에서도 shutdown 명령만 입력하면 아래와 같은 메시지를 내보내고 서버가 죽지 않는다.
shutdown nosave 를 입력해야 한다.

```
(error) ERR Errors trying to SHUTDOWN. Check logs.
```

- **rdbcompression : yes or no**, default yes
  RDB 파일을 쓸때 압축 여부를 정한다. 압축 알고리즘은 LZF이다.
  압축률이 그다지 높지 않다. default인 yes로 놓고 사용한다.
- **rdbchecksum : yes or no**, default yes
  RDB 파일 끝에 CRC64 checksum 값을 기록한다. default인 yes로 놓고 사용한다.
- **dbfilename dump.rdb**
  RDB 파일명을 지정한다. Path는 지정할 수 없습니다. Path는 working directory에 따릅니다.

기타

- RDB File Format 정보는 여기를 보세요.

어느 것을 선택할까

AOF(Append Only File)

- AOF를 기본으로 하고, RDB를 Option으로 한다. AOF 시간 설정은 everysec로 하고, AOF Rewrite를 사용한다. AOF를 사용해도 성능에 거의 영향을 미치지 않습니다. 성능 자료는 여기를 보세요.
- Master 노드이고 Auto-restart 기능을 사용할때는 AOF를 사용하는 것이 안전하다. Master 노드가 자동 재시작했을때 AOF 파일은 없고, 몇 분 전 RDB 파일만 있다면 slave 노드들의 데이터도 마스터와 같이 몇 분 전 RDB 파일의 데이터를 받게 된다. 마스터 자동 시작과 Persistence 기능 사용은 여기를 보세요.

DB 복구

- AOF와 RDB 파일 양쪽이 모두 존재할 경우 어느 것을 먼저 읽어들일 까요?
  그것은 redis.conf 설정에 달려 있습니다.
  appendonly yes 인 경우에는 AOF 파일을 읽어 들입니다.
  appendonly no 인 경우 RDB 파일을 읽어 들입니다.
- 그럼 appendonly yes 인데 AOF 파일은 없고 RDB 파일만 있을 경우 RDB 파일을 읽어 들일까요?
  읽어 들이지 않습니다.
  반대로 appendonly no 인데 RDB 파일은 없고 AOF 파일만 있을 경우 AOF 파일을 읽어 들일까요?
  읽어 들이지 않습니다.
- AOF 파일을 읽어 들일때 메시지: DB loaded from append only file: 6.406 seconds
  RDB 파일을 읽어 들일때 메시지: DB loaded from disk: 4.510 seconds

- 데이터 파일을 읽어 들이는 명령이 있을까요? 없습니다.
  레디스 서버 시작 시 읽어 들입니다.

<< TIME | Persistence Intro | AOF Internal >>

redisgate@gmail.com 02.503.2235
서울시 강남구 강남대로 342 역삼빌딩 5층 (역삼동) 우 06242