인메모리 키-값 데이터베이스의 데이터 보존성을 위한 부하에 따른 디바이스 활용*

최원기[0] 박상현[1†]
연세대학교 컴퓨터과학과
cwk1412@yonsei.ac.kr[0], sanghyun@yonsei.ac.kr[1]

Evaluating Overhead for Data persistency in In-memory Key-Value store

WonGi Choi Sanghyun Park
Department of Computer Science, Yonsei University

요 약

레디스는 데이터를 저장하는 주매체가 메인 메모리인 인메모리 기반의 키-값 데이터베이스로써 높은 처리율로 실시간으로 데이터 저장 및 처리를 요구하는 환경에서 각광받고 있다. 인메모리 기반의 키-값 데이터베이스는 시스템의 종료 시 데이터가 손실되는 메인 메모리의 휘발성 특성으로 인하여, 데이터 영속성을 위한 복구 메커니즘을 제공해야 한다. 레디스[1]에서는 RDB파일과 AOF파일을 디스크에 기록하여 프로그램 재시작에도 종료 직전의 데이터를 복구할 수 있도록 한다. 그러나 RDB/AOF파일은 디스크 입출력을 야기해 레디스의 전체적 성능에 영향을 미친다. 본 논문에서 다양한 저장 매체에 RDB/AOF파일을 기록하는 실험을 진행하면서 데이터 영속성을 위한 레디스의 부하가 상당하며 데이터 크기가 늘어남에 따라 저장 디바이스 매체에 대해 나타나는 성능 저하 정도가 상이함을 확인하고 적절한 디바이스의 활용이 필요함을 제시한다.

1. 서 론

최근 대용량의 데이터의 활용도에 대해 주목받기 시작하면서, 다양한 분야에서 실제로 여러 가지 종류의 데이터 수집이 이루어지고 있으며, 데이터를 효율적으로 저장하는 플랫폼에 대한 연구도 활발히 진행되고 있다. 초창기 대용량 데이터를 수용할 수 있도록 다수의 클러스터를 이용하여 분산 파일 시스템 환경을 구축한 하둡(Hadoop)[2]이 주목을 받았다. 하지만 데이터를 일괄적으로 처리하는 작업에 최적화되어 있는 하둡 구조의 특성상 크기가 작은 데이터를 실시간으로 처리하는 작업에 대해서는 취약한 단점을 가지고 있다. 따라서 IoT의 센서 데이터나 모바일 어플리케이션의 메시지와 같이 크기가 작고 형태가 다양한 데이터를 저장하는 환경의 경우, 키와 데이터를 간단한 형태로 효율적으로 저장하는 키-값 데이터베이스가 각광받고 있다.

키-값 데이터베이스의 경우, 지원하는 데이터의 종류나 데이터를 저장하는 매체의 종류에 따라 다양하게 분류된다. 그 중에서도 메인 메모리를 매체로 사용하는 인메모리 기반 키-값 데이터베이스의 경우, 높은 처리율과 사용자의 요청에 대한 빠른 반응성으로 기업 및 연구 단체에서 주목받고 있다.[3][4][5] 그 중 레디스(Redis)의 경우, 오픈 소스 프로젝트로 공개되어 데이터의 실시간 처리가 요구되는 환경에서 많이 사용되고 있다.

인메모리 기반의 키-값 데이터베이스의 경우, 데이터를 저장하고 읽는 성능에서 뛰어난 효과를 보인다. 하지만 사용자의 프로그램의 종료 요청이나 서버의 비정상적인 오류로 인하여 강제로 종료될 때 휘발성인 메인 메모리의 특성으로 인하여 데이터가 손실될 우려가 있다.

레디스에서는 데이터의 보존성을 보장하고 데이터 손실 시 복구를 위하여 고유의 로깅 동작을 지원한다. 레디스는 메모리 구조체에 저장된 데이터를 스냅샷(Snapshot) 형태로 파일에 기록하는 RDB 파일과 레디스 데이터베이스에 요청된 명령어를 기록하는 AOF파일(Append Only File)을 제공한다. 레디스가 종료로 인하여 메모리에 기록한 데이터를 손실했을 때, 스냅샷 형태의 RDB 파일을 읽어 데이터를 복구하거나, AOF파일에 기록된 연산을 재실행하여 레디스에 종료 이전의 값들을 복구할 수 있도록 한다. 그러나 RDB파일과 AOF파일의 경우, 디스크에 기록하는 연산을 필요로 하므로 메인 메모리를 주 매체를 사용하는 레디스의 입장에서 성능을 저하시키는 요인이 될 수 있다.

본 논문에서는 RDB파일과 AOF파일을 다양한 저장 매체에 기록하여 성능을 평가하고자 한다. 레디스의 데이터 보존성을 위해 필요한 부하의 정도에 대해 측정하고 다양한 데이터 크기와 저장 매체에 대하여 실험을 진행하고 결과에 대해 논의함으로써 레디스 사용자에게 최적화된 성능을 제공하는 환경을 제시하고자 한다.

2. 배 경

2.1 RDB 파일

RDB파일은 사용자가 설정한 특정 시간 조건을 만족할 때 메모리상에 저장되어 있는 모든 데이터에 대해서 스

* "본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 SW컴퓨팅산업원천기술개발사업(SW스타랩)의 연구결과로 수행되었음" (IITP-2017-0-00477)

† 교신저자

냅샷(Snapshot)을 만들어 저장되는 파일이다. 레디스는 임시 RDB파일을 생성한 후, 메모리에 있는 데이터를 바이너리 포맷의 형태로 기록한다. 그림 1과 같이 RDB파일은 RDB파일의 고유의 번호(Magic number)를 기록하고 데이터베이스의 번호를 기록한 후 번호에 해당하는 데이터베이스에 저장된 키-값을 순차적으로 기록한다. RDB파일에 데이터의 입력이 모두 완료되면 레디스는 이전 RDB파일을 삭제하고 새롭게 생성된 임시 RDB파일로 대체한다.

레디스의 복구 작업에 RDB파일을 사용하는 경우, RDB파일에 저장된 키, 값들을 읽어 메모리에 저장하는 방식으로 진행된다. 따라서 RDB파일을 사용하는 경우 복구 시간이 빠르지만, RDB파일의 작성은 특정 시간 간격마다 이루어지기 때문에 RDB파일 생성 후 다음 RDB파일 생성 전에 시스템 오류로 인하여 데이터가 손실된다면 그 사이에 삽입된 데이터에 대하여 복구가 불가능하다. 따라서 일반적인 환경에서는 AOF파일과 같이 작성하여 데이터의 손실율을 줄이고, 복구 시에 활용된다.

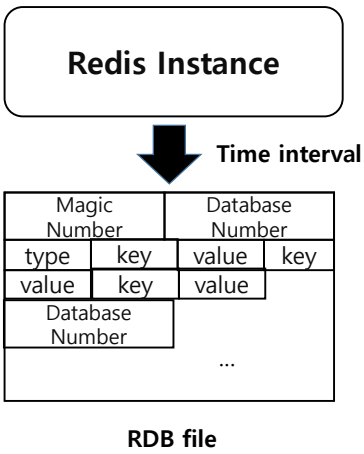

그림 1 RDB 파일의 구조

2.2 AOF 파일 (Append Only File)

레디스가 수신한 명령 중 메모리 내부 데이터에 수정이 이루어진 명령에 대해 AOF파일에 기록한다. 기존 관계형 데이터베이스에서도 사용되는 로깅 작동 방식과 유사하게 그림 2와 같이 명령어, 키, 값의 정보가 포함된 로그 레코드가 순차적으로 AOF파일에 기록되는 방식이다.

AOF파일의 경우 복구 시에 기록된 로그를 순차적으로 읽으면서 초기화된 레디스 데이터베이스에 명령어를 재실행하는 방법으로 복구 작업을 수행하게 된다. 따라서 AOF파일의 크기가 증가함에 따라 복구 시에 수행해야 할 명령어가 많고 복구시간 또한 증가함을 알 수 있다. AOF의 크기가 무한정으로 증가하는 것을 방지하기 위해, 수정한 연산을 반영하여 로그 레코드의 수를 줄이기 위한 AOF 다시쓰기(AOF rewrite) 작업이 수행된다. 하지만 AOF 다시쓰기 작업은 데이터의 일관성을 위해 처리를 지연하기 때문에 레디스의 명령어 처리율에도 영향을 미친다. 또한 다시쓰기를 위한 버퍼 할당으로 인하여 메모리 사용량이 급격하게 증가하게 하여 메모리의 크기가 충분하지 않은 인메모리 환경에서 치명적인 오류를 발생하게 할 수 있다.[6]

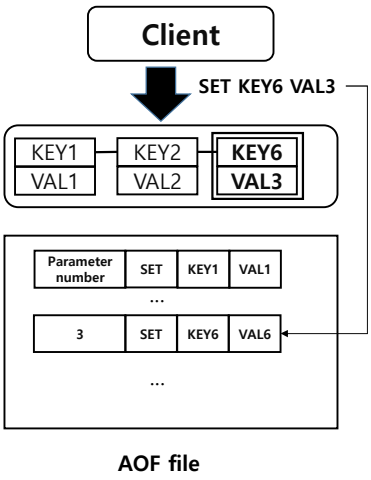

그림 2 AOF 파일의 구조

3. 성능 실험

레디스의 데이터 영속성을 위해 RDB파일 및 AOF파일을 저장하는데 필요한 디스크 부하를 알아보기 위한 실험을 진행하였다. 레디스에서 제공하는 벤치마크(redis-benchmark)를 사용하여 데이터 크기를 8 byte에서 4096 byte까지 변경해가며, 다양한 저장 매체에 RDB/AOF파일을 저장하는 실험을 수행했다.

실험 환경은 아래 표와 같다. RDB/AOF파일을 저장 매체는 하드디스크, SATA SSD, NVMe, 배터리가 부착된 DRAM의 형태인 NVRAM을 사용하여 측정하였으며, 모두 ext4 파일 시스템으로 포맷한 블락 기반의 디바이스로 사용하였다. 추가적으로 RDB/AOF파일의 부하를 확인하기 위하여 RDB/AOF파일 쓰기에 대한 옵션을 비활성화한 실험도 추가하였다. 레디스의 경우 3.2 버전 기준으로 실험하였으며 클러스터 환경이 아닌 Standalone 모드로 실행하였다. 레디스는 기본적으로 샤딩(Sharding)을 이용하여 데이터의 독립적인 분배가 이루어지기 때문에 데이터 영속성에 관련한 동작들은 네트워크 부하와 관련이 적으므로 Standalone 모드로 실험을 진행하였다.

실험은 RDB/AOF 파일을 저장하는 매체를 지정한 상태에서 디스크 부하를 생성하는 SET 명령어를 데이터 크기를 변경해가며 100만 번 실행하였다.

표 1 실험 환경

CPU	Intel i7 6700k
RAM	DDR4 64GB
HDD	Seagate 3TB Barracuda ST3000DM001
SSD	SAMSUNG 850 PRO 256GB
NVMe	NVMe SSD SAMSUNG 850 PRO 256GB
NVRAM	Flashtec NV1616
OS	CentOS 7
Redis	Redis 3.2 (Standalone)

레디스 벤치마크는 삽입되는 키의 범위를 지정할 수 있는데, 본 실험에서는 일반적인 환경에서의 RDB/AOF파일 쓰기의 부하를 확인하기 위하여, 키의 범위를 넓게 하여 수정 연산이 빈번히 일어나는 경우는 배제하였다.

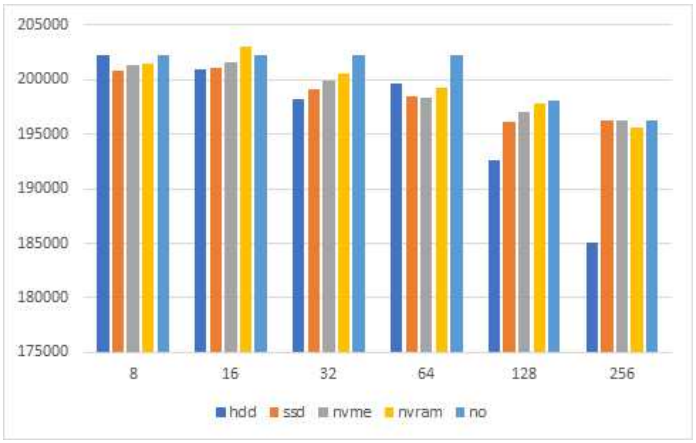

그림 3 디바이스에 따른 데이터 크기 (8-256byte) 별 성능 비교

그림 3와 그림 4는 해당 실험에 대한 결과를 그래프로 나타낸 그림이다. 세로축은 레디스에서 1초당 몇 건의 SET 연산을 수행했는지 표시 하였고 가로축은 RDB/AOF파일 저장 매체로 사용된 매체를 표시하였다.

그림 3을 보면 사용된 데이터의 크기를 8 byte에서 256 byte의 크기로 변경하며 실험을 진행 했을 때의 그래프이다. 데이터의 크기가 작은 경우, 성능에 있어서 크게 차이를 보이지 않으나 일반적으로 디바이스의 처리량에 비례하여 성능이 하드디스크, SSD, NVMe, NVRAM 순으로 미세하게 향상됨을 알 수 있었다.

성능이 좋은 NVMe, NVRAM은 RDB/AOF 파일 옵션을 비활성화하여 메모리 연산 작업만 진행하는 경우와 비교하였을 때 큰 차이가 없었다. 하지만 데이터의 크기가 128 byte, 256 byte로 증가하였을 때, 하드디스크의 경우 RDB/AOF 파일 옵션을 비활성화 하였을 때와 비교했을 때 처리율이 최대 1.1x까지 차이가 나는 것을 확인할 수 있었고, 데이터 영속성을 위한 레디스의 부하가 존재함을 확인할 수 있었다.

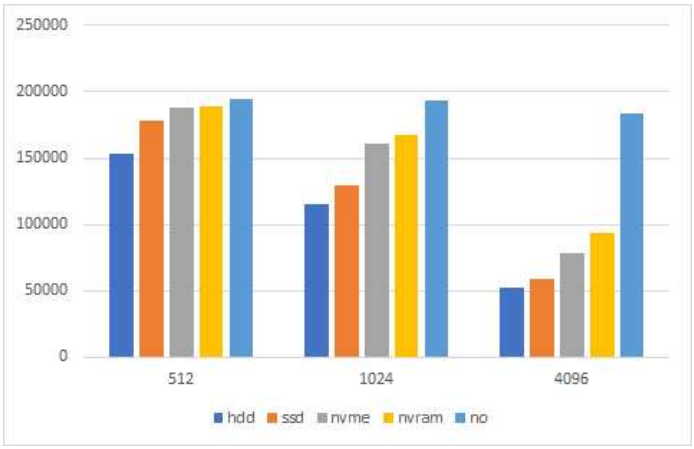

그림 4 디바이스에 따른 데이터 크기 (512-4096byte) 별 성능 비교

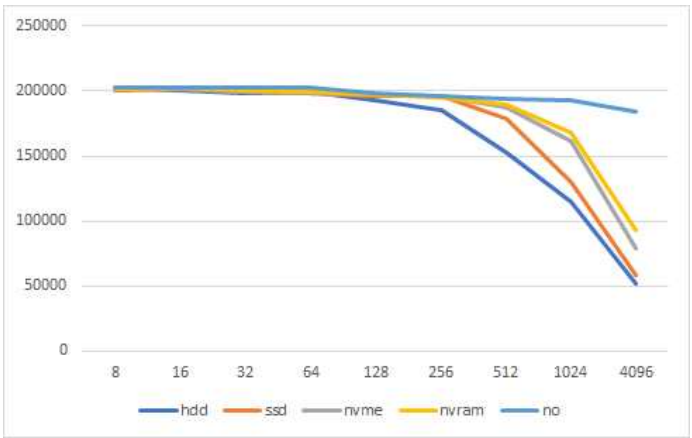

그림 5 디바이스에 따른 성능 감소 추이

그림 4의 경우, 데이터 크기를 512 byte, 1024 byte, 4096 byte로 설정했을 때 측정한 실험에 대해 표시한 그래프이다. 데이터 크기가 증가함에 따라 영속성을 위한 RDB/AOF파일의 크기도 함께 증가하게 된다. 그림 4의 추이를 보면 확인할 수 있듯이, 디스크에 기록되는 RDB/AOF파일의 크기가 증가하면서 디바이스의 성능에 따른 전체 레디스 성능의 차이가 심화되었고 처리율이 좋은 디바이스를 활용하는 것이 레디스의 전체적인 성능이 향상시키는 것을 확인하였다. 하드디스크의 경우 RDB/AOF파일 쓰기 옵션을 비활성화 했을 때에 비해

3.5x나 성능이 감소함을 확인하였다. 또한 앞선 실험과 다르게 비활성화 경우와 유사할 정도로 성능이 좋았던 NVMe, NVRAM의 경우에도 비활성화의 경우에 비해 성능이 감소하는 것으로 보아 데이터 크기가 증가할수록 RDB/AOF파일을 기록하는 데 필요한 부하가 상당함을 확인할 수 있었다. 그림 5는 그림 3와 그림 4에 대한 실험 결과를 종합하여 나타낸 그래프이며 데이터 크기가 증가함에 따라 처리율이 감소함을 알 수 있다. 로그 파일의 특성상 RDB/AOF파일의 경우에 임의 쓰기 패턴이 상대적으로 적을 것으로 보이며, 이에 따라 128 byte 미만의 데이터 크기가 작은 경우에는 성능 차이가 많이 나지 않음을 확인하였다. 그러나 데이터 크기가 증가함에 따라 기록해야할 RDB/AOF파일의 크기 또한 서서히 증가해감에 따라 NVMe 및 NVRAM등의 성능이 좋은 디바이스 매체를 활용하는 것이 레디스의 전체 성능 감소폭이 완화함을 확인하였다. 레디스 사용자의 사용 환경과 입력되는 데이터의 크기에 대한 분석을 통해 RDB/AOF 파일을 어떠한 디바이스 매체에 저장해야 가격 대비 최고의 성능을 도출할 수 있는 지를 예측할 수 있다.

4. 결 론 및 향후 계획

레디스 내부의 데이터의 영속성을 유지하기 위해 기록하는 RDB/AOF 파일의 전체적인 부하를 측정하였고, 파일을 기록하는 디바이스 성능에 따라 레디스의 전체적인 성능에도 영향이 있음을 확인하였다. 데이터베이스로서 데이터 손실에 대한 복구는 필수적인 요소이므로, 레디스에서 RDB/AOF파일 쓰기 옵션 활성화는 보장되어야 한다. 따라서 사용자는 디바이스의 가격 혹은 용량 대비 성능을 고려하여 RDB/AOF파일 저장하는 적합한 매체를 선택해야 한다. 그리고 데이터의 크기에 따라서도 디바이스 별 성능 양상이 상이하므로, 레디스에 주로 저장되는 데이터의 크기 및 형태에 대한 분석도 필요하다.

향후 연구로는 SET 명령어 이외의 연산들에 대한 성능 평가를 진행할 예정이다. 또한 로깅 동작에서 성능에 영향을 미칠 수 있는 AOF 다시 쓰기 연산에 대해 세분화된 분석과 성능 평가를 진행할 예정이다. 또한 데이터 크기에 따라 RDB/AOF파일 쓰기 작업을 다른 디바이스에 기록할 수 있도록 디바이스 계층 구조를 설계할 예정이다. 추가적으로, 자체적으로 데이터의 영속성을 보장하는 NVDIMM 디바이스를 사용할 수 있는 라이브러리를 지원하는 레디스 환경을 구축하여 성능을 평가하고 RDB/AOF파일을 기록하지 않았을 때, 얼마나 향상되었는지 확인할 예정이다. 또한 NVDIMM 디바이스를 사용함에 따라 발생할 수 있는 새로운 부하에 대해 조사할 예정이다.[7][8]

5. 참고 문헌

[1] https://redis.io

[2] http://hadoop.apache.org

[3] H. Lim, B Fan, D. G Andersen, and M. Kaminsky, “SILT: A memory-efficient, high-performance key-value store”, In Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles, pp. 1-13, Oct. 2011.

[4] W. Cao, S. Sahin, L. Liu, and X. Bao, “Evaluation and Analysis of In-Memory Key-Value Systems”, IEEE International Congress on Big Data(BigData Congress), pp. 26-33, Jul, 2016.

[5] Garcia-Molina, Hector and Kenneth Salem, “Main memory database systems: An overview”. IEEE Transactions on knowledge and data engineering, Vol. 4, No. 6, pp. 509-516, Dec. 1992.

[6] 진민화, 박상현, “Redis에서 AOF Rewrite 기법의 오버헤드 분석”, 한국정보기술학회논문지, 15권, 3호, 1~10쪽, 2017년, 3월.

[7] J. Arulraj, M. Perron, A. Pavlo, “Write-behind logging”, Proceedings of the VLDB Endowment, 10.4: 337-348, 2016.

[8] S. Gao, J. Xu, T. Härder, B. He, B. Choi and H. Hu, “PCMLogging: optimizing transaction logging and recovery performance with PCM”, IEEE Transactions on Knowledge and Data Engineering, 27(12), 3332-3346, 2015.