데이터베이스 성능 향상을 위한 기계학습 기반의 RocksDB 파라미터 분석 연구

김휘군, 최원기, 최종환, 성한승, 박상현[1]
연세대학교 컴퓨터과학과
e-mail : {jinhuijun, cwk1412, mathcombio, hssung, sanghyun}@yonsei.ac.kr

A Study on the Analysis of RocksDB Parameters Based on Machine Learning to Improve Database Performance

Huijun Jin, Won Gi Choi, Jonghwan Choi, Hanseung Sung, Sanghyun Park[1]
Dept. of Computer Science, Yonsei University

요 약

Log Structured Merged Tree(LSM-Tree)구조를 사용하여 빠른 데이터 쓰기 성능을 보유한 RocksDB 에는 쓰기 증폭과 공간 증폭 현상이 발생한다. 쓰기 증폭은 과도한 쓰기 연산을 유발하여 데이터 처리 성능 저하와 플래시 메모리 기반 장치의 수명 저하를 초래하며, 공간 증폭은 데이터 저장 공간 점유로 인한 저장 공간 부족 문제를 야기한다. 본 논문에서는 쓰기 증폭과 공간 증폭 완화를 위해 RocksDB 의 성능에 영향 주는 주요 파라미터를 추출하고, 기계학습 기법인 랜덤 포레스트를 사용하여 추출한 파라미터가 쓰기 증폭과 공간 증폭에 미치는 영향을 분석하였다. 실험결과 쓰기 증폭과 공간 증폭에 영향을 많이 주는 주요 요소를 선별하였고 다른 파라미터에 대비해서 성능 격차가 61.7% 더 나타낸 것을 발견하였다.

1. 서론

정형화된 데이터가 양산되던 이전과 달리, 빅데이터 환경에서는 비정형 데이터가 대량으로 생산 및 소비된다. 정형 데이터를 다루는 관계형 데이터베이스 시스템으로는 비정형 데이터를 처리하기에 어려움이 있다. 전통적인 시스템의 한계를 극복하기 위해, 비정형 데이터를 키-값 쌍의 형태로 저장 및 관리하는 키-값 데이터베이스가 등장했다.

RocksDB 는 높은 데이터 처리 성능을 위해 Log-Structured Merge-Tree (LSM-Tree) 구조를 사용하는 디스크 기반의 키-값 데이터베이스이다 [1][2]. LSM-Tree 구조는 순차쓰기를 유도하여 우수한 쓰기 성능을 보이지만, 데이터 관리를 위해 수행되는 컴팩션(Compaction) 중에 부가적인 쓰기와 공간 사용을 필요로 한다 [3]. 부가적인 쓰기는 데이터베이스 처리 성능 저하를 초래함과 동시에, 과도한 쓰기 연산은 데이터베이스 파일을 Solid-State Drive (SSD)에 저장하는 경우 SSD 수명 단축의 원인이 된다 [4]. 또한, 실제로 저장한 데이터보다 많은 양의 데이터가 저장되어 데이터베이스 운영 측면에서 부담을 가중시킨다.

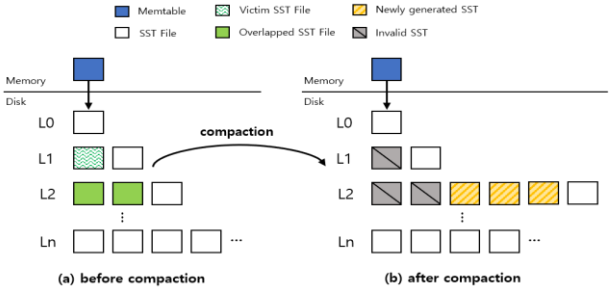
RocksDB 에서 발생하는 쓰기 증폭과 공간 증폭 문제를 개선하기 위해, RocksDB 의 파라미터 최적화가 필요하다. 그러나 많은 파라미터를 가지고 있는 RocksDB 의 성능을 튜닝하는 것은 데이터베이스 전문가들도 많은 시간이 소요되는 어려운 과제이다. 따라서, 본 논문에서는 기계학습 알고리즘 중 하나인 랜덤 포레스트 알고리즘을 이용하여 데이터베이스 성능에 관여하는 중요 RocksDB 파라미터들을 식별하고, 이로써 데이터베이스 성능 튜닝의 부하를 줄이고자 한다.

2. 배경

2.1. RocksDB 의 데이터 저장방식

RocksDB 의 데이터 관리를 위해 수행되는 컴팩션(Compaction)은 부가적인 쓰기 연산을 유발한다. 컴팩션 과정은 그림 1 과 같다. 컴팩션은 서로 다른 크기를 가진 각 계층의 임계점을 초과하면 동작된다. 먼저, 컴팩션 작업이 호출되면 임계점을 초과한 계층에 존재하는 파일(Victim SST)을 선택한다. 그리고, 임계점을 초과한 계층의 하위 계층에서, 선택된 파일의 키 범위와 중복된 키 범위를 가진 파일들(Overlapped SST)을 추가적으로 선택한다. 선택된 파일들은 병합 정렬을 통해 새로운 파일(Newly generated SST)로 생성되어 하위 계층에 저장된다. 컴팩션에서 선택된 파일은 유효

[1] 교신저자(corresponding author)



(그림 1) LSM-tree 의 컴팩션 도식화

하지 않은 파일(Invalid SST)로 관리되며, 추후 시스템의 가비지 컬렉션(Garbage Collection)을 통하여 삭제된다.

2.2. Write Amplification(쓰기 증폭) & Space Amplification(공간 증폭)

컴팩션은 임계점을 초과한 계층에 존재하는 파일을 하위 계층으로 내려 보내 공간을 확보하는 작업이다. 그림 1 과 같이 컴팩션 대상 파일(Victim SST) 외에도 추가적인 파일(Overlapped SST)을 조회하고 기록한다. 하나의 파일을 내려 보내기 위해서는 추가적인 파일 쓰기 연산이 발생하며, 이와 같은 추가적인 쓰기가 발생하는 현상을 Write Amplification (WA)이라고 한다. 멀티 스레드 기반의 프로그램인 RocksDB 는 컴팩션 작업을 동시 다발적으로 수행하여 부가적인 쓰기 연산이 기하급수적으로 증가한다. 이때, 과도한 디스크 입출력 연산이 수행됨에 따라 데이터 처리 성능이 저하된다. 또한, WA 는 데이터 처리 성능 저하와 더불어, 데이터베이스에서 파일을 SSD 에 쓸 경우, SSD 와 같이 쓰기 횟수에 제한이 있는 플래시 메모리 기반 장치들의 수명을 단축시킨다.

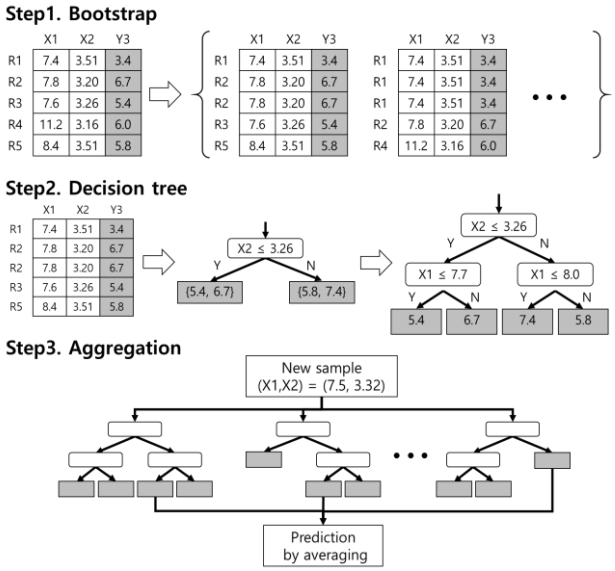
RocksDB 에는 WA 현상 외에도 데이터 저장 공간 활용 측면에서도 문제가 있다. 그림 1 에 나타나 있듯이, 컴팩션으로 인하여 더 이상 필요가 없어진 파일(Invalid SST)은 가비지 컬렉션에 의해 삭제될 때까지, 디스크 공간을 차지하고 있으며, 이러한 공간 차지 현상을 Space Amplification (SA)이라고 한다. 동시 다발적으로 수행되는 컴팩션 작업으로 인하여 Invalid SST 파일이 지나치게 축적되며, 상당한 공간을 차지하는 문제가 있다.

3. 분석 방법

3.1. 랜덤 포레스트 및 기여도 분석

랜덤 포레스트(random forest)는 부트스트랩(bootstrap)을 통해 서로 다른 훈련데이터를 여러 개 생성하고, 각 훈련데이터로 훈련된 의사결정트리(decision tree)들을 결합(aggregation)하여 예측 또는 회귀분석을 수행하는 기계학습 모델이다 [5]. 랜덤 포레스트 분류모델(classifier)은 엔트로피(entropy) 또는 지니계수(Gini index)를 이용하여 각 특징의 불순도(impurity)를 계산하고, 불순도를 가장 크게 감소시키는 특징의 분기(node)를 구축한다. 유사한 방법으로, 랜덤 포레스트 회귀모델(regressor)은 평균제곱오차(mean squared error)를 이용하여 불순도를 정의하여 최적의 분기를 구축해간다.

랜덤 포레스트는 여러 의사결정트리에서 산출된 각 특징의 불순도를 이용하여 불순도평균감소량(mean decrease impurity; MDI)을 구하고, 이를 기반으로 분류 및 회귀에 대한 특징 기여도를 산출할 수 있다 [5]. 불순도평균감소량이 클수록 중요한 특징으로 해석되며, 해당 특징이 어떻게 기여하는지는 추가적인 분석을 통해 살펴봐야 한다.



(그림 2) 랜덤 포레스트 도식화

3.2. 회귀모델 결정계수

랜덤 포레스트 회귀모델이 훈련데이터를 잘 학습하였는지를 평가하기 위해 결정계수(coefficient of determination;R^2)을 계산하였다. 결정계수는 식(1)과 같이 종속변수 실제값 y_i 들의 총제곱합(total sum of squares)과 예측값 f_i에 대한 오차제곱합(sum of squares of residuals)의 비로 계산된다.

$$R^2 = 1 - \frac{\sum_i (y_i - f_i)^2}{\sum_i (y_i - \bar{y})^2} \quad (1)$$

식(1)에서 $\bar{y}$은 실제값들의 평균값을 나타낸다. 결정계수의 값은 0 에서 1 사이에 있으며, 1 에 가까울수록 모델이 훈련데이터를 잘 설명한다고 해석한다.

3.3. 분산분석

범주형 변수(categorical variable)에 따른 성능지표 차이를 분석하기 위해 분산분석(analysis of variance; ANOVA)을 수행하였다. ANOVA 분석은 주어진 범주형 변수에 대해서 범주 항목에 따른 종속변수 분산들의 합과 전체 종속변수 값의 분산의 비를 계산하여 항목 간의 종속변수 분포 차이를 통계적으로 검정한다 [6]. 검정통계치로부터 유의 확률 p-value 를 계산하고, 일반적으로 p-value 가 0.05 보다 작으면 변수값에 따른 성능 차이가 있다고 해석한다.

4. 실험환경과 실험결과 및 분석

본 실험은 표 1 과 같은 실험환경에서 수행되었으며, RocksDB 성능측정을 위해 db_bench 를 사용하였다. db_bench

<표 1> 실험환경

OS	CentOS 7.3.1611 (x86_64)
CPU	Intel® Xeon® CPU E5-2660 v2 @ 2.20GHz
RAM	Samsung M393B2G70QH0-YK 16GB*4
NVMe	Intel SSDPEDMD800G4 DC P3700 800GB
RocksDB	Version 6.13

는 RocksDB 에서 제공하는 성능측정 도구로써, RocksDB 계층 크기, 압축 기법, 데이터 크기 및 개수 외에도 다양한 파라미터를 제공하는 벤치마크이다 [7].

본 실험에서는 1,000,000 건의 키-값 쌍을 적재하여 RocksDB 성능을 측정하였다. 적재되는 개별 키와 값의 크기는 각각 16 B 와 1 KB 로 고정된 값을 가지며, 이외에 RocksDB 의 약 200 여개의 파라미터 중에서 WA 와 SA 에 관련된 30 개의 파라미터를 선별하였고, 이러한 파라미터들은 무작위 값을 가진다. 총 1,000 개의 무작위 조합에 대한 데이터를 생성하였으며, 데이터 생성시간을 효율적으로 단축시키기 위해 LSM-tree 의 크기를 기본값의 1/64 배로 축소하였다.

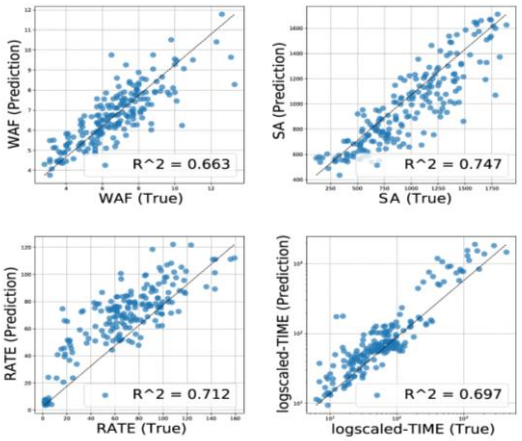
WA, SA 에 영향주는 파라미터들이 RocksDB 의 기본 성능지표인 데이터 처리 속도와 처리 시간에 주는 영향도 동시에 보기 위해, 본 연구에서는 총 4 가지 성능지표 Write Amplification Factor (WAF), 적재된 데이터 크기(SA), 처리 속도(RATE), 처리 시간(TIME)을 이용하여 파라미터에 따른 RocksDB 성능 변화를 분석하였다. WAF 은 WA 의 비율을 나타내는 성능지표로서 스토리지에 기록하는 데이터의 양과 데이터 베이스에 쓰여진 데이터 양의 비율을 나타낸다. WAF 계산식은 식(2)와 같다.

$$WAF = \frac{Bytes\ written\ to\ Storage}{Bytes\ written\ to\ Database} \qquad (2)$$

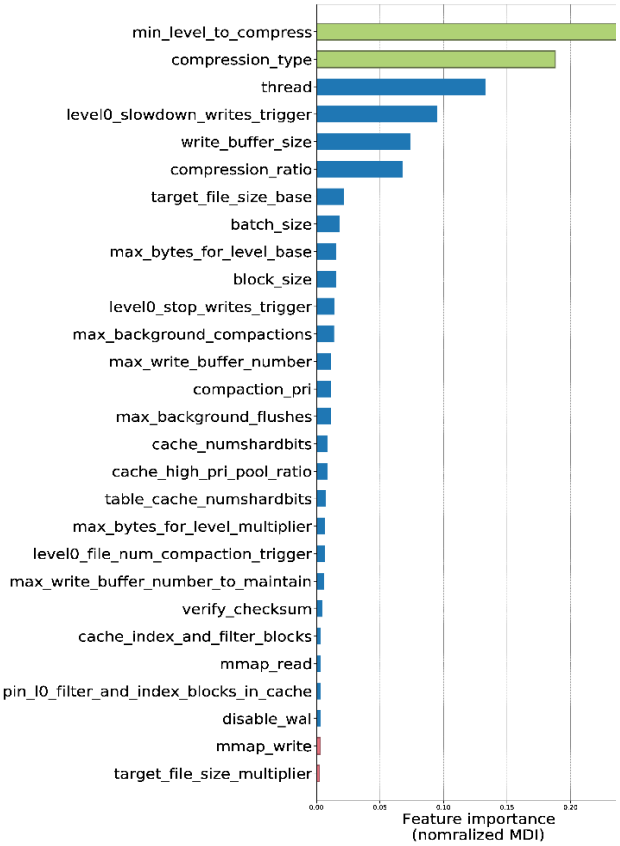
적재된 데이터 크기는 실제 LSM-Tree 에 최종적으로 기록된 데이터의 크기를 의미한다. WAF 와는 다르게, 압축알고리즘의 영향으로 인하여 최종적으로 데이터베이스에 적재된 데이터 양은 사용자가 기록한 논리적인 양보다 적다. 따라서, SA 를 측정하기 위해서는 논리적 데이터 양과 실제 데이터 양의 비율보다, LSM-Tree 에 최종적으로 기록된 데이터 크기로 표시하는 것이 바람직하다. 데이터 처리 속도는 RocksDB 가 모든 데이터를 데이터베이스에 적재하는 평균속도를 의미하며, 데이터 기록을 시작하는 시점부터 종료되기까지 소요되는 시간을 처리 시간으로 표시하였다.

4.1. 랜덤 포레스트 학습 평가

그림 3 는 WAF, SA, RATE, TIME 등 4 개의 성능지표를 랜덤 포레스트 학습을 통해 분석한 결과이다. 랜덤 포레스트의 성능을 평가하기 위해 교차검증을 수행하였으며, 데이터 1,000 건을 8:2 로 훈련데이터와 시험데이터로 분할하여 사용하였다. 시험데이터에 대하여 WAF 예측에 대한 결정계수는 0.663, SA 결정계수는 0.747, RATE 는 0.712, 그리고 TIME 은 0.697 로 확인되어 학습이 잘 되었다고 판단할 수 있었다. 학습된 랜덤 포레스트는 계속해서 파라미터 기여도 분석에 활용된다.



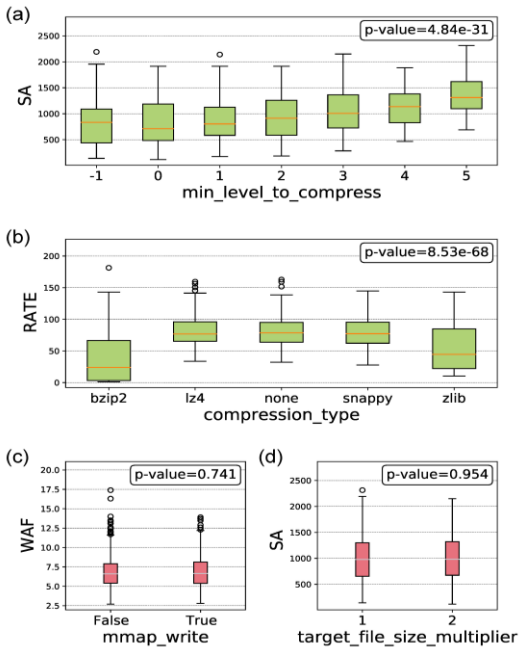
(그림 3) RocksDB 성능 지표별 예측 정확도



(그림 4) 지니평균감소량 기반의 파라미터 기여도

4.2. 파라미터 기여도 분석

그림 4 는 지니평균감소량 기반으로 파라미터 기여도를 분석한 결과이다. 결과에 의하면 min_level_to_compress 의 기여도가 가장 큰 것으로 확인되었으며, 그 다음은 차례대로 compression_type, thread, level0_slowdown_writes_trigger, write_buffer_size, compression_ratio 등이 중요한 것으로 나타났다. Min_level_to_compress 는 RocksDB 가 해당 파라미터 값과 동일한 계층부터 압축을 수행하도록 설정한다. 해당 파라미터보다 낮은 계층들은 압축이 적용되지 않는다. 기본값은 -1



(그림 5) 최상위 최하위 파라미터 비교분석

로 모든 계층에서 압축을 수행한다. Compression_type 은 적용할 압축 알고리즘을 가리키며, 기본 알고리즘은 snappy 이고 이외에 none, zlib, bzip2, lz4 등을 사용할 수 있다. Thread 는 요청된 명령을 수행할 스레드의 개수를 나타내고 기본값은 1 이다. Level0_slowdown_writes_trigger 은 RocksDB 의 쓰기 속도를 저하시키는 L0 의 파일 개수이며 기본값은 20 이다. Write_buffer_size 는 압축 전 RocksDB 의 memtable 에서 버퍼링 할 바이트 수이고 기본값은 64MB 이다. Compression_ratio 는 데이터 압축률이며 기본값은 0.5 이다. 실제로, compression_type 은 여러 연구팀에서 RocksDB 성능 향상을 위해 지속적으로 연구되고 있는 중요 파라미터이다 [8].

그림 5 은 그림 4 의 파라미터 기여도를 기반으로 최상위, 최하위에서 각각 2 개씩 파라미터를 선택하여 분산분석을 수행한 결과이다. 그림 (a)와 (b)는 최상위 파라미터에 해당하는 min_level_to_compress 와 compress_type 에 따른 SA, RATE 의 성능차이를 보여준다. 각각의 성능차이를 분산분석을 통해 통계적으로 검정하면 p-value 가 각각 4.84e-31, 8.53e-68 로 유의수준 0.05 보다 훨씬 작아 min_level_to_compress 및 compress_type 에 따라 성능차이가 크게 나타남을 확인하였다. 그림 (c)와 그림 (d)는 최하위 파라미터에 해당하는 mmap_write, target_file_size_multiplier 에 따른 WAF 와 SA 의 성능차이를 분석한 결과이다. 최상위 파라미터 결과와 달리, p-value 가 각각 0.741, 0.954 로 유의수준보다 높은 값을 보여주고 있어 두 파라미터는 RocksDB 성능에 크게 관여하지 않는 것으로 확인되었다. 실제로, min_level_to_compress 값에 따른 최소평균 SA 와 최대평균 SA 를 비교하였을 때, 각각 821.5, 1328 로 61.7%의 성능차이가 나타났으며, compression_type 에 따라서는 평균 RATE 가 최소 36.57 에서 최대 80.16 까지의 격차를 보여 RocksDB 최적화에 몹시 중요한 것으로 드러났다. 이와는 대조적으로 최하위 파라미터에 해당하는 mmap_write 와 target_file_size_multiplier 에 대한 WAF, SA 의 최대 최소 격차는 최소평균 대비 각각 3.77%, 0.51%으로 확인되어 RocksDB 성능 향상에 거의 관련성이 없는 것으로 나타났다.

5. 결론

본 논문에서는 RocksDB 의 200 여개의 파라미터 중에서 RocksDB 의 주요성능지표인 WA, SA 에 관련된 파라미터 30 개를 선별하였고, 각각의 파라미터들이 WA, SA 에 주는 영향을 랜덤 포레스트를 통해 분석하였고, 중요도가 가장 높은 파라미터들 min_level_to_compress, compression_type, thread, level0_slowdown_writes_trigger, write_buffer_size, compression_ratio 를 식별하였다. 분산분석을 통해 상기 파라미터들이 WA 및 SA 성능에 크게 관여하는 것으로 밝혀졌고, 하위 파라미터들의 관련성은 낮은 것으로 확인되었다. 따라서, 식별된 중요 파라미터들의 값을 적절히 선택하면 RocksDB 의 성능을 크게 향상시킬 수 있을 것으로 기대되며, 나아가 식별된 파라미터들은 LSM-Tree 를 사용하는 모든 데이터베이스에서 사용되는 것들이기에 다른 데이터베이스 시스템에서도 중요하게 작용할 것으로 보인다. 또한, 식별된 파라미터를 최적화하여 WA 및 SA 를 낮춤으로써 값비싼 저장 디바이스의 불필요한 부하 및 수명단축 문제를 완화하여 경제적인 비용 절약이 가능할 것으로 기대된다.

향후 연구에서는 식별된 파라미터들이 자동적으로 최적화가 가능하도록, 최적화 알고리즘 기반의 RocksDB 자동 최적화시스템 연구개발을 하고자 한다.

Acknowledgments

이 논문은 2020 년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원(IITP-2017-0-00477, (SW 스타랩) IoT 환경을 위한 고성능 플래시 메모리 스토리지 기반 인메모리 분산 DBMS 연구개발)과 국토교통부의 스마트시티 혁신인재육성사업의 지원을 받아 수행된 연구임

참고문헌

[1] Mei, Fei, et al. "LSM-tree managed storage for large-scale key-value store." IEEE Transactions on Parallel and Distributed Systems 30.2 (2018): 400-414.
[2] https://rocksdb.org/
[3] O'Neil, Patrick, et al. "The log-structured merge-tree (LSM-tree)." Acta Informatica 33.4 (1996): 351-385.
[4] Hu, Xiao-Yu, et al. "Write amplification analysis in flash-based solid state drives." Proceedings of SYSTOR 2009: The Israeli Experimental Systems Conference. 2009.
[5] Breiman, Leo. "Random forests." Machine learning 45.1, pp. 5-32, 2001
[6] Howell, David, Statistical Methods for Psychology, Duxbury, Pacific Grove, CA, 2002
[7] https://github.com/facebook/rocksdb/wiki/Benchmarking-tools
[8] Dong, Siying, et al. "Optimizing Space Amplification in RocksDB." CIDR. Vol. 3. 2017.