REINFORCE 알고리즘을 통한 RocksDB 파라미터 튜닝

조준흠, 김경훈, PIAO SHENGMIN, 박상현†[1]
연세대학교 컴퓨터과학과
{chojh11c, kkh115505, sungmin630, sanghyun†}@yonsei.ac.kr

RocksDB Parameter Tuning through REINFORCE Algorithm

Junheum Cho, Kyeonghun Kim, PIAO SHENGMIN, Sanghyun Park†
Dept. of Computer Science, Yonsei University

요 약

RocksDB는 Facebook에서 개발한 LevelDB 기반의 오픈소스 데이터베이스로, 테라바이트 급의 빅데이터 처리가 요구되는 서버 어플리케이션 개발에 주로 활용된다. RocksDB는 파라미터에 따라 데이터 처리 성능이 크게 상이하여, 최적의 성능을 확보하기 위해서는 파라미터 튜닝 작업이 필수적이다. 본 연구에서는 강화학습 알고리즘을 활용한 RocksDB 파라미터 자동화 튜닝 모델을 제안한다. 제안하는 모델은 RocksDB 파라미터 구성을 상태로, 하나의 파라미터 수정을 액션으로, 수정된 파라미터 적용에 따른 RocksDB 성능을 보상으로 정의하여 튜닝 작업을 수행한다. 벤치마킹 데이터를 통해 제안 모델이 RocksDB의 데이터처리성능을 대폭 향상시키는 것을 확인하였다.

1. 서 론

Facebook에서 개발한 RocksDB는 LevelDB 기반의 오픈소스 데이터베이스로, 최대 테라바이트급의 빅데이터를 다루는 워크로드 처리에 적합한 고성능 저장소 역할을 맡을 수 있다[1]. 이러한 RocksDB는 스마트시티와 같이 작은 규모의 데이터가 대량 생산 및 처리가 필요한 어플리케이션에서 전통적인 관계형 데이터베이스 관리시스템보다 뛰어난 적합성을 지니고 있어서[2], RocksDB 성능 개선을 위한 연구가 활발히 이루어지고 있다[3,4]

RocksDB는 Log Structured Merged Tree (LSM Tree) 구조를 갖는다. RocksDB는 메모리 영역과 디스크 영역으로 구분해 데이터를 저장한다. 메모리 영역의 memtable의 용량이 넘치면 디스크에 데이터를 저장한다. 이때 디스크는 내부에 레벨을 나눠 데이터를 저장하는데, memtable에서 넘어온 데이터를 상위 레벨과 하위 레벨 간의 병합 정렬을 통해 순차적으로 저장한다. 이 과정에서 쓰기 증폭이 발생한다. 쓰기 증폭은 과도한 쓰기 연산을 발생시켜 성능 저하를 야기하며 장치의 수명을 줄인다. 동시에 LSM Tree는 데이터를 저장할 때 덮어쓰기가 아닌 새로운 공간에 저장한다. 이러한 이유로 데이터 저장공간이 부족하게 되는 공간 증폭이 발생한다.

본 논문에서는 RocksDB의 파라미터 튜닝을 통해 이러한 문제를 해결하려 한다. 파라미터 튜닝을 위해 XGBoost 기법과 강화학습 (reinforcement learning) 알고리즘을 사용했다. XGBoost는 파라미터와 성능 사이의 모델을 훈련하는 과정에서 사용되었다. 강화학습은 주어진 state에서 action을 통해 reward를 얻어 최종적으로 얻을 수 있는 reward의 합을 최대로 만드는 것을 목표로 하는 알고리즘으로 본 논문에서는 튜닝할 파라미터를 선택하여 이에 따른 성능 수치 최대화하는 방향으로 학습을 진행한다.

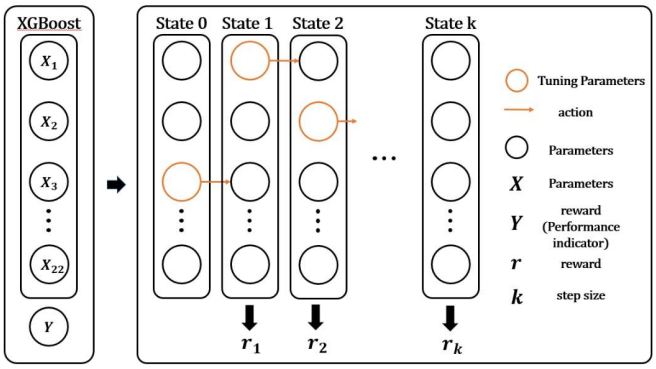
2. 제안 모델

본 논문에서는 XGBoost를 통한 모델 훈련과 강화학습을 통해 파라미터 튜닝을 학습시켜 RocksDB의 파라미터 튜닝을 시도했다. 그림 (1)을 통해 모델의 전반적인 구조를 확인할 수 있다.

3.1 강화학습 reward 계산을 위한 XGBoost

XGBoost는 Gradient Boost[5] 기반의 앙상블 모델로서, 효과적인 예측/분류 성능과 더불어 효율적인 연산량으로 각광받고 있는 모델이다[6]. 특히 명목형 데이터와 연속형 데이터 모두 유연하게 다룰 수 있고, 추가적인 정규화항(regularization)을 사용하여 과적합을

* 이 논문은 2021년도 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 혁신인재육성사업으로 지원을 받아 수행된 연구임.
† 교신 저자: sanghyun@yonsei.ac.kr



(그림 1) 전체 모델 도식화

방지하여, 뛰어난 성능을 가진다. 따라서 강화학습 시 필요한 reward 예측에 XGBoost를 채택하여 사용하고자 한다.

본 논문에서는 XGBoost를 활용하여, 선별한 22개의 파라미터와 튜닝의 목표로 삼은 성능 지표를 모델에 훈련시켰다. 이 모델을 통해 튜닝한 파라미터의 값이 어떠한 성능을 보여줄지 예측하였으며 최종 모델의 성능을 테스트할 때에도 사용하였다.

3.2 RocksDB 파라미터 튜닝을 위한 REINFORCE

강화학습은 주어진 상태(state)에서 행동(action)을 통해 얻은 보상(reward)을 최대화하는 것을 목표로 한다. 즉 모든 보상의 합을 최대로 만드는 정책(policy)을 찾는 것이 강화학습의 목표이다.

본 논문에서는 REINFORCE 알고리즘을 선택하여 정책을 찾는다[7]. REINFORCE 알고리즘은 N 번의 에피소드에서 행동을 결정하는 k 번의 step을 거쳐 학습을 한다. 하나의 에피소드에서 각 단계에서 얻는 reward r 에 $discount\ factor$ γ 를 적용하여 선행된 action a 으로 얻게 되는 reward의 가치를 높여주고 이를 모두 합한다. 이렇게 더해진 최종 값을 $Q-value$라 부르며 각 에피소드에서의 $Q-value$는 식(1)과 같이 정의될 수 있다.

$$Q_N = \sum_{i=1} \gamma^{i-1} r_i \qquad (1)$$

하나의 에피소드가 끝나면 식(2)와 같이 손실 함수를 계산할 수 있다.

$$Loss = -Q(s,a) log\pi(a|s) \quad (2)$$

본 논문에서는 RocksDB 파라미터 구성을 하나의 state로, 파라미터 중 하나의 파라미터를 선택하여 이를 튜닝하는 과정을 action a로 정의한다. 튜닝은 파라미터 값p 에 $-0.5 \sim 0.5$ 사이 임의의 값n을 곱한 다음 나온 값을 기존 값에 더하는 방식으로 식 (3)과 같이 구할 수 있다.

$$p_i' = p_i + p_i * n \quad (-0.5 < n < 0.5) \qquad (3)$$

이 과정에서 얻은 값을 파라미터 값 p' 으로 치환하여 성능을 측정하는데, 이 과정을 반복하여 얻은 값들 중 성능을 가장 많이 향상시킨 값을 선택하여 최종적으로 파라미터를 튜닝한다. Reward의 경우 튜닝을 통해 얻은 성능 지표의 값으로 정의한다.

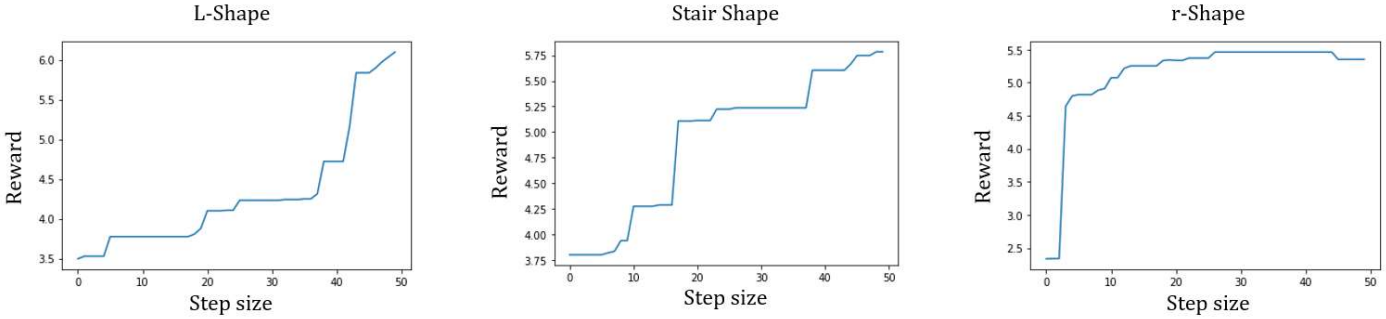
4. 실험 및 결과분석

4.1 실험환경

본 논문에서는 RocksDB 파라미터 튜닝을 위한 강화학습 모델 훈련 및 평가를 <표1>과 같은 환경에서 수행하였다.

4.2 벤치마킹 데이터

본 연구에서는 Jin et. al.[8] 에서 생성된 벤치마킹 데이터를 이용하여 제안모델을 훈련 및 성능평가를 수행하였다. 벤치마킹 데이터는 RocksDB 벤치마킹 소프트웨어인 db_bench[9]를 통해 생성되었으며, 2만개의 튜플을 가지고 있고, 각 튜플은 22개의 RocksDB 파라미터 구성에 따른 RocksDB 데이터



(그림 2) $Step$에 따른 $Reward$ 양상

<표 1> 실험환경

OS	Ubuntu 18.04.5 LTS
CPU	Intel(R) Xeon(R) Silver 4110 CPU @ 2.10GHz
GPU	Tesla V100-PCIE-32GB
CUDA	11.2

<표 2> step에 따른 reward 상승률

step	상승률(%)
10	93.5
30	101.7
50	110.1

처리속도(RATE) 데이터를 제공한다.

4.3 제안모델 성능평가

XGBoost를 사용하여 파라미터와 성능 지표 사이의 관계 모델을 훈련시켰다. 이때 성능 지표는 처리 속도(RATE)로 선정하여 학습을 진행하였다.

강화학습 단계에서 다양한 초기 state를 제공하기 위하여 데이터셋 20000개의 파라미터 구성 중에서 5000개의 샘플 데이터를 초기 state로 입력하였다.

훈련된 모델의 평가를 위하여, 100개의 임의의 테스트셋에 대하여 튜닝 이후의 reward를 계산하였다. <표2>는 각 step에 따라 계산된 초기 reward 대비 튜닝 이후 reward의 상승률의 평균을 나타낸다. <표2>의 결과와 같이, 100개의 테스트셋에서 10, 30, 50 step 마다 평균 reward가 93.5%, 101.7%, 110% 향상된 것을 확인하였다.

또한 훈련된 모델이 파라미터 튜닝을 진행하는 경과를 확인하기 위하여, 각 step에 따라 계산되는 reward를 시각화 하였다. (그림2)는 100개의 테스트셋 중 임의 샘플링을 통해 정성적인 분석 결과로 도출된, 대표적인 3가지의 reward 상승 곡선을 나타낸다. step의 후반부에 성능이 높게 향상되는 L형 그래프 (L-Shape), 점진적인 성능 향상을 보여주는 계단형 그래프 (Stair Shape), step 초반에 높은 성능 향상을 보여주는 r 모양 그래프 (r-Shape)와 같이 3가지 양상을 보였다. 이는 초기 입력된 state에 모델의 성능이 의존적임을 나타낸다. 그럼에도 불구하고, 최종 reward의 값은 일관된 성능 향상을 나타내는 것을 보여주고 있다.

5. 결론

본 논문에서는 XGBoost를 통해 파라미터와 성능 지표 사이의 관계를 훈련시킨 다음, 강화학습을 통해 RocksDB의 파라미터를 튜닝하였다. 강화학습 알고리즘은 주어진 state에서 가장 영향력 있는 파라미터를 선택하는 방향으로 학습되었다. 하지만 최선의 파라미터를 선택하기 위해서는 각 파라미터들 사이의 관계도 함께 고려해야 한다. 따라서 향후 연구에서 각 파라미터들 사이의 관계를 고려하여 튜닝을 진행하는 연구를 진행하려 한다.

참고문헌

[1] https://github.com/facebook/rocksdb/wiki

[2] Alsaig, Alaa, et al. "Characterization and efficient management of big data in iot-driven smart city development." *Sensors* 19.11 (2019): 2430.

[3] 김도영, et al. "GPU 가속화 필터를 적용한 디스크 기반 키-값 데이터베이스." *정보과학회 컴퓨팅의 실제 논문지* 26.3 (2020): 161-166.

[4] Li, Xiang, et al. "Improving Read Performance of LSM-Tree Based KV Stores via Dual Grained Caches." *2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*. IEEE, 2019.

[5] J. Friedman. Greedy function approximation: a gradient boosting machine. Annals of Statistics, 29(5):1189–1232,2001.

[6] Chen, Tianqi, and Carlos Guestrin. "Xgboost: A scalable tree boosting system." *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 2016.

[7] Williams, Ronald J. "Simple statistical gradient-following algorithms for connectionist reinforcement learning." *Machine learning* 8.3 (1992): 229-256.

[8] Jin, Huijun, et al. "Improvement of RocksDB Performance via Large-scale Parameter Analysis and Optimization." *Journal of Information Processing Systems* (2021) (accepted)

[9] https://github.com/facebook/rocksdb/wiki/Benchmarking-tools