비선형 기계학습 기반의 Redis 파라미터 튜닝 연구

서주연 ○ 이지은 김경훈 JIN HUIJUN 박상현†
연세대학교 컴퓨터과학과
{sjy9728s, jieun199624, kkh115505, jinhuijun, sanghyun}@yonsei.ac.kr

A Study on Redis Parameter Tuning Based on Non-linear Machine Learning

Juyeon Seo○ Jieun Lee Kyeonghun Kim JIN HUIJUN Sanghyun Park†
Dept. of Computer Science, Yonsei University

요 약

싱글 스레드 기반의 인메모리 데이터베이스인 Redis는 신속한 데이터 처리 성능 보존을 위해, 데이터 지속성, TTL(Time-To-Live) 처리와 같이 시스템 성능 저하를 유발하는 작업은 백그라운드로 수행한다. 그러나 지속성 방법과 백그라운드 작업은 데이터 처리 지연 및 메모리 사용량 증가와 같은 추가적인 부하를 초래한다. 이러한 Redis의 작업 부하는 파라미터 튜닝을 수행하여 완화할 수 있다. DBMS 파라미터 튜닝을 위한 선행 연구에서는 영향력이 높은 파라미터를 추출하기 위하여 선형적 관계만을 고려한다. 이를 개선하기 위해 본 논문에서는 비선형 기계학습 기법을 사용한 파라미터 튜닝을 수행하였다. 해당 모델을 적용한 Redis는 데이터 처리 성능을 향상시킬 수 있는 최적의 Redis 파라미터 조합을 도출할 수 있다. Redis의 지속성 방법에 따른 단위 시간당 처리량을 측정한 결과, 튜닝한 파라미터를 사용하였을 때 기존 파라미터 대비 최대 45.9%의 성능 향상을 확인하였다.

1. 서 론

스마트시티에서는 사람들의 일상 활동으로부터 방대한 양의 데이터가 양산되고 있다. 각종 센서, 스마트 디바이스, GPS로부터 수집되는 데이터를 통해, 도시 경쟁력 강화 및 국민 삶의 질을 향상시키기 위한 실시간 데이터 분석이 요구되고 있다[1]. 실시간 데이터 생성 속도 증가로 인해 우수한 데이터 처리 성능을 가진 서비스가 요구됨에 따라, 인메모리 데이터베이스(In-Memory Database)의 사용이 증가하고 있다. 인메모리 데이터베이스는 메인 메모리를 데이터 저장 공간으로 사용한다. 따라서 디스크 입출력이 빈번한 디스크 기반의 데이터베이스보다 응답 속도가 빠른 장점이 있다.

인메모리 데이터베이스인 Redis[2]는 모든 데이터를 메모리에 유지하기 때문에 데이터 접근 지연이 적다. 또한, 데이터를 키-값 쌍으로 저장하여 데이터 조회 속도가 빨라 실시간 데이터 처리가 요구되는 환경에서 사용되고 있다. Redis는 DRAM의 휘발성으로부터 데이터를 보존하기 위한 지속성 방법을 제공한다. 지속성 방법은 로그 레코드를 생성하여 데이터 유실을 방지하지만, 데이터 처리가 지연되고 추가적인 메모리 사용을 필요로 한다.

Redis는 지속성 방법으로 RDB(Redis Database)와 AOF(Append-Only File)라는 두 가지 기법을 제공한다. RDB는 데이터셋에 대해 일정 간격으로 스냅샷(Snapshot)을 생성하는 방법이고, AOF는 데이터셋을 변경하는 명령에 대한 로그 레코드를 생성하여 로그 파일에 덧붙인다. 두 방법을 통하여 데이터 지속성을 높일 수 있지만, 로깅 작업으로 인해 데이터 처리 성능 저하와 같은 추가적인 부하가 발생한다.

Redis는 지속성 방법 외에도 다양한 백그라운드 작업을 수행한다. 싱글 스레드 기반의 프로그램인 Redis는 한 번에 하나의 명령만 실행할 수 있다. 만약 지속성 작업이 메인 프로세스에서 수행될 경우, 요청된 명령어는 처리되지 않고 지연된다. 따라서 특정 시간이 초과된 클라이언트의 연결을 해제하거나, 만료된 키를 삭제하는 작업, 데이터 지속성 작업과 같이, 작업 수행이 완료되기까지 오랜 시간이 요구되는 작업들은 백그라운드로 처리한다. 그러나 이러한 노력에도 불구하고, 여전히 데이터 처리 성능이 저하하는 현상이 발생한다.

Redis에서 수행하는 지속성 방법과 부가적인 백그라운드 작업으로 인해 유발되는 부하는 Redis의 파라미터 값을 조절하여 완화시킬 수 있다. 다양한 워크로드에서 최적의 파라미터 값을 튜닝하는 작업은 데이터 처리 성능을 높이는 데에 중요한 단계다. 그러나 Redis의 파라미터 개수와 값이 광범위하기 때문에, 전문가가 최적의 파라미터 값을 찾는 데는 한계가 있다.

데이터베이스 성능에 영향력 높은 파라미터를 튜닝하기 위해 기계학습 기법을 사용한 연구가 진행되고 있다. 선행 연구인 OtterTune[3]은 지도, 비지도 기계학습 기법을 적용하여, 서로 다른 워크로드에서 얻은 결과를 통해 특정한 워크로드에서의 최적 파라미터 값을 찾는다. 그러나 해당 연구는 추출해낸 데이터의 선형적 관계만을 고려하여 최적 파라미터를 선정한다. 이를 개선하기 위해 본 연구에서는 OtterTune의 구조를 착안하여, 비선형적 기계학습 기법을 Redis에 적용한 RS-OtterTune(Redis Simplified OtterTune)을 제안한다. 본 모델은 비선형 회귀 기반의 기계학습 기법으로 RandomForest[4]와 XGBoost[5]를 사용한다. RS-OtterTune의 성능 향상을 검증하기 위해 Redis의 기본 성능과 선행 연구와의 비교실험을 진행하였고, 최대 45.9%의 성능 향상을 확인하였다.

2. 모 델

본 연구는 Redis 파라미터 튜닝을 위해 비선형 기계학습 기법을 적용한 RS-OtterTune을 제안한다. RS-OtterTune은 그림 1과 같이 샘플 생성(Sample Generation), Metrics 단순화(Metrics Simplification), Knobs 랭킹(Knobs Ranking) 그리고 추천(Recommendation) 단계로 구성되며, 결과적으로 최적의 configuration을 도출한다. knob은 하나의 파라미터를 뜻하고, configuration은 knob의 집합을 의미한다.

* 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

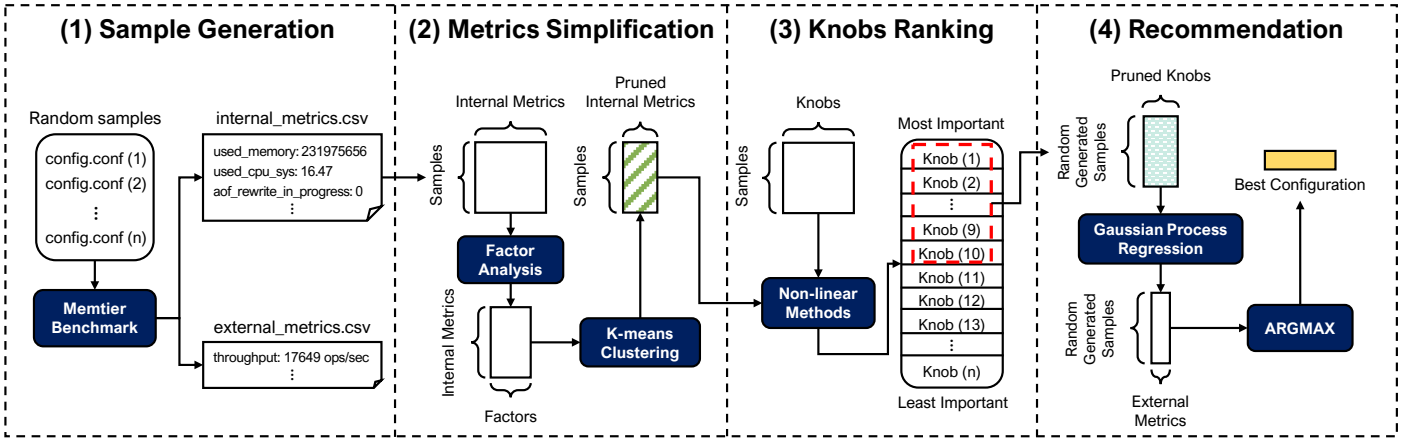

그림 1. RS-OtterTune 모델 구조

2.1 데이터 샘플 생성

공개되어 있는 Redis 워크로드 데이터셋이 없으므로, 본 연구는 훈련에 필요한 데이터 샘플을 생성하는 단계가 요구된다. 다양한 샘플을 얻기 위해 랜덤한 파라미터 값을 가지는 Redis configuration 파일들을 생성한다. 그다음, 각 파일들의 내용으로 설정된 Redis 서버에서 Memtier-Benchmark[6]를 통해 내부(Internal) metrics와 외부(External) metrics를 측정하여 별도의 파일로 저장한다. 내부 metrics는 Redis 서버 정보와 메모리 사용량과 같은 통계값을 저장하고, 외부 metrics는 데이터 처리 성능을 평가하기 위해 단위 시간당 처리량(Throughput) 등을 포함한다.

2.2 Metrics 단순화

내부 metrics는 다수의 실행 결과값을 포함하므로, Metrics 단순화 단계에서는 비슷한 특징을 가지는 metrics를 찾아 단순화시킨다. 인자 분석(Factor Analysis)을 통해 내부 metrics 사이의 상관 관계에 대한 분산을 찾는다. 추출한 인자들을 이용하여 K-평균 군집화(K-means Clustering)로 비슷한 성격을 갖는 metrics로 k개의 군집을 얻는다. 군집의 중심에 가장 가까운 metrics를 선택하여 k개의 pruned metrics로 단순화한다.

2.3 Knobs 랭킹

Knobs 랭킹 단계에서 RS-OtterTune은 Metric 단순화 단계에서 얻은 pruned metrics를 이용해 성능에 영향을 주는 knob들을 선정한다. 기계학습 기법을 통해 knob들의 영향 정도를 측정하여 정렬하고, 상위 10개의 knob들을 활용한다. 본 연구는 기계학습 기법으로 비선형 회귀 모델 RandomForest와 XGBoost를 사용한다.

2.3.1 RandomForest

RandomForest는 훈련을 통해 생성된 여러 의사결정트리(Decision Tree)들로부터 분류를 취합하여, 결과를 예측 또는 회귀 분석을 수행하는 배깅(Bagging) 방법을 사용하는 앙상블 모델이다. RandomForest 회귀 모델은 각 특징의 불순도(Impurity)를 계산하고, 불순도평균감소량을 통해 분류에 대한 특징 기여도를 산출한다. 회귀 모델에서 상대적으로 중요도가 높은 변수를 추출해낼 수 있다. 또한, 대용량 데이터 처리에 용이하고, 과적합(Overfitting)을 방지하는 특징이 있다.

2.3.2 XGBoost

배깅 방법을 이용하는 RandomForest와 달리 XGBoost는 부스팅(Boosting) 방법을 사용하는 기계학습 앙상블 모델이다. 부스팅 방법은 약한 예측 모형들의 학습 에러에 가중치를 두고, 순차적으로 다음 학습 모델에 반영하여 강한 예측 모형을 만든다. 기존 부스팅 방법을 사용하는 알고리즘들은 계산량이 많고 과적합이 발생한다는 문제점이 있다. XGBoost는 병렬처리를 이용하여 기존 모델들보다 빠르고, 과적합을 방지하기 위한 규제항이 포함되어 있다. 수식은 다음과 같다.

$$\hat{y} = \alpha \times tree_A + \beta \times tree_B + \gamma \times tree_C + \cdots \quad (1)$$

$$L = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k) \quad (2)$$

수식 (1)에서 $tree$는 트리 기반의 예측 모형이고, 각 예측모형에는 가중치(α, β, γ)를 곱한 뒤 더해 예측값 $\hat{y}$를 구한다. 목적 함수 L은 $\hat{y}$와 실제값 y 사이의 손실 함수와 정규화 항 Ω을 더함으로써 구한다. 이때, n은 데이터 샘플의 개수, K는 트리의 개수를 의미한다.

2.3.3 상위 Knobs 추출

RS-OtterTune은 RandomForest와 XGBoost를 통해 얻은 성능에 대한 중요도를 기준으로 knob들을 정렬한다. knob들은 Redis 성능과 관여가 깊을수록 상위에 위치하게 된다. 본 연구에서는 정렬되어 있는 knob들 중에서 상위 10개를 추출하여 추천 단계에서 사용한다.

2.4 추천

RS-OtterTune의 추천 단계에서는 특정한 워크로드에 대한 최적의 파라미터 조합을 추천하기 위해 GP(Gaussian Process)[7] 회귀 모형을 사용한다. GP 모형에서 x는 Knobs 랭킹에서 얻은 pruned knobs이며, y는 외부 metrics으로 학습한다. 그리고 pruned knobs을 기반으로 랜덤한 값을 가지는 다수의 Redis configuration 파일을 생성한 후, 학습된 GP 모델을 통해 데이터 처리 성능을 예측한다. 추출한 예측값들 중 최댓값을 가지는 Redis configuration 파일을 최종적으로 추천하게 된다.

3. 실험 및 결과 분석

3.1 실험 환경

표 1. 시스템 실험 환경

OS		Ubuntu 16.04.7 LTS
CPU		Intel® Xeon® CPU @ 2.00GHz
RAM	Data Generation	DIMM 4G
	Tuning	DIMM 16G
Redis Version		5.0.2

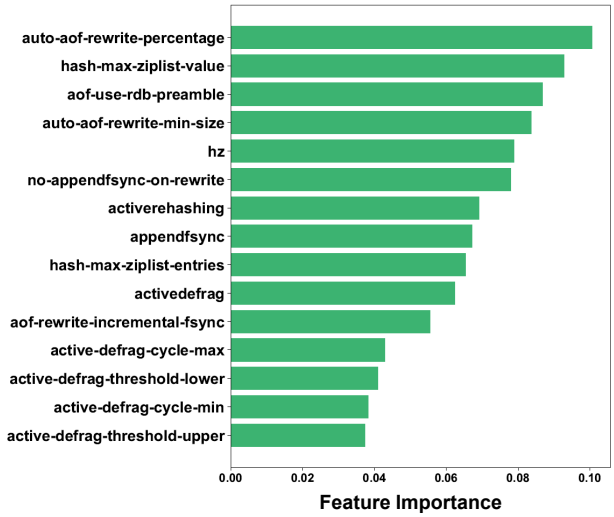

그림 2. Write-Only_AOF의 파라미터 기여도 분석

본 논문에서는 데이터 샘플 생성과 다양한 기계학습 기법의 성능 비교 실험을 Google Cloud Platform(GCP)에서 진행하였다. 데이터 생성을 위한 32개의 노드와 파라미터 튜닝을 위한 2개의 노드를 생성하였고, 시스템 실험 환경은 표 1과 같다. 모든 실험은 메모리 기반 데이터베이스 성능 측정을 위한 벤치마크인 Memtier-benchmark를 사용하였다.

본 실험은 RandomForest와 XGBoost를 모델에 적용하였을 때 성능을 비교하기 위해, 두 가지 워크로드에서 단위 시간당 처리량을 측정하였다. 워크로드는 데이터 적재만 수행하는 Write-Only, 데이터 조회와 적재 비율이 같은 Read-Write(1:1)에 대해, 지속성 방법으로 RDB와 AOF를 사용한 경우를 구분하여 진행하였다. 적재되는 1,000,000건의 키-값 데이터의 키의 크기는 16 B로 고정된 크기를 가지며, 값의 크기는 128 B로 설정하였다.

3.2 결과 분석

그림 2는 Redis가 지속성 방법으로 AOF를 사용하여 데이터를 적재할 때, XGBoost를 통해 파라미터 기여도를 평가한 결과이다. 각 파라미터의 중요도는 Gain 방법을 통해 계산된다[5]. 분석한 파라미터들을 중요도 순으로 정렬한 후, 상위 10개의 파라미터 값을 최적화하여 configuration 파일에 반영하였다.

Redis의 파라미터를 튜닝하지 않은 Default와 선형 기법인 Lasso를 사용한 OtterTune, 비선형 기법인 RandomForest와 XGBoost를 사용한 RS-OtterTune을 통해 튜닝한 Redis의 성능 비교 실험을 진행하였다. 그림 3은 Redis의 지속성 방법과 워크로드를 달리 하여 측정한 데이터 처리 성능을 나타낸다. 파라미터를 튜닝하지 않은 기존 Redis보다 기계학습 기법을 이용하여 파라미터 값을 최적화한 경우 더 높은 데이터 처리 성능을 보였다. 또한, 의미 있는 knob들을 추출할 때 선형 기법인 Lasso를 활용한 OtterTune보다 비선형 기법을 이용한 RS-OtterTune의 경우 더 높은 성능을 이끌어낼 수 있다. 본 실험을 통해 knob과 metrics의 선형적인 관계를 얻는 것보다 비선형성을 이용할 때 유용한 관계 정보를 추출함을 검증하였다.

Redis는 지속성 방법으로 AOF를 사용할 때 과도한 디스크 입출력과 메모리 사용량 급증이 발생한다. 특히, 데이터 적재가 빈번한 환경에서 다수의 로그 레코드를 생성하여 디스크의 파일에 작성하기 때문에, 그림 3(b)의 Default의 경우 가장 낮은 데이터 처리 성능을 보였다. 기계학습 기법을 이용하여 파라미터 튜닝 후 비교 실험을 수행한 결과, 해당 워크로드에서 XGBoost를 활용하여 튜닝하였을 때 Default보다 약 45.9% 향상되었다.

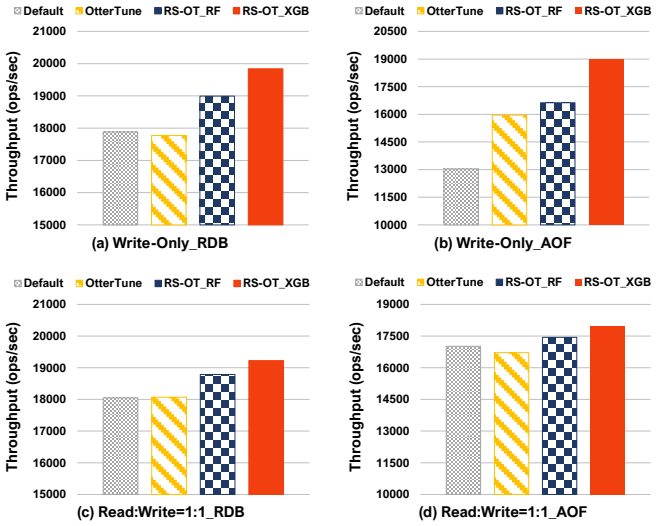

그림 3. 워크로드 별 데이터 처리 성능

4. 결 론

본 논문에서는 Redis의 작업 부하를 완화하기 위해 기계학습을 이용하여 Redis의 파라미터 값을 최적화하는 연구를 수행하였다. DBMS 파라미터 튜닝을 위해 선형 기계학습 기법을 사용한 선행 연구와 달리, 비선형 기법을 활용한 RS-OtterTune을 통하여 의미 있는 metric과 파라미터를 분석하고, 추출한 결과를 configuration 파일에 반영하였다. 데이터 처리 성능 비교 실험을 통해 Redis에서 발생하는 추가적인 부하를 파라미터 튜닝으로 개선할 수 있음을 확인하였다. 또한, 선형 기법보다 비선형 기법을 사용하였을 때 Redis의 데이터 처리 성능이 크게 향상되었다.

본 연구에서는 외부 metrics 중 단위 시간당 처리량만을 고려했지만, 지연 시간(Latency)도 중요한 지표로 여겨진다. 향후 연구에서는 데이터 처리량과 지연 시간을 모두 고려할 수 있도록 모델을 개선시키고자 한다. 또한, 데이터 적재와 조회 비율을 조절하여, Redis가 다양한 워크로드에서 최적의 성능을 이끌어낼 수 있는 파라미터 조합을 도출하는 모델을 개발할 계획이다.

참 고 문 헌

[1] Rathore, M. Mazhar, et al, "Real-time secure communication for Smart City in high-speed Big Data environment," *Future Generation Computer Systems*, Vol.83, pp.638-652, 2018.

[2] Redis. https://redis.io

[3] Memtier-Benchmark. https://github.com/RedisLabs/memtier_benchmark

[4] Breiman, L, "Random forests," *Machine learning*, Vol.45, No.1, pp. 5-32, 2001.

[5] 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.

[6] Van Aken, Dana, et al, "Automatic database management system tuning through large-scale machine learning," *Proceedings of the 2017 ACM International Conference on Management of Data*, 2017.

[7] Rasmussen, Carl Edward, "Gaussian processes in machine learning," *Summer school on machine learning*, Springer, Berlin, Heidelberg, 2003.