Redis 파라미터 분류 및 단계적 베이지안 최적화를 통한 파라미터 튜닝 연구

조성운*, 박상현**
*동국대학교 통계학과
**연세대학교 컴퓨터공학과
jsw1660@dgu.ac.kr, sanghyun@yonsei.ac.kr

A Study on Parameter Tuning for Redis via Parameter Classification and Phased Bayesian Optimization

Seong-Woon Jo*, Sang-Hyun Park**
*Dept. of Statistics, Dong-Guk University
**Dept. of Computer Science, Yon-Sei University

요 약

DBMS 파라미터 튜닝이란 데이터베이스에서 제공하는 다양한 파라미터의 값을 조율하여, 최적의 성능을 도출하는 과정이다. 데이터베이스 종류에 따라 파라미터 개수가 수십 개에서 수백 개로 다양하며, 각 기능이 모두 다르기 때문에 최적의 조합을 찾는 것은 쉽지 않다. 선행 연구에서는 BO 기법을 사용하여 적절한 파라미터 값을 추출했지만, 파라미터 개수에 비례하여 차원이 커지는 문제가 발생한다. 본 논문에서는 통계적으로 파라미터를 분류하여 탐색 공간을 줄인 다음 단계적으로 BO 를 수행하는 PBO 방식을 제안한다. 파라미터 값을 랜덤하게 할당하여 벤치마킹한 결과값을 군집화한 후, 각 군집별로 파라미터와의 연관성을 분석해 높은 상관관계를 가진 파라미터를 매칭시켜 분류한다. 제안하는 방법론을 검증하기 위하여 8 가지 회귀 모델과의 비교 실험을 통해 제안한 방법론의 우수성을 검증하였다.

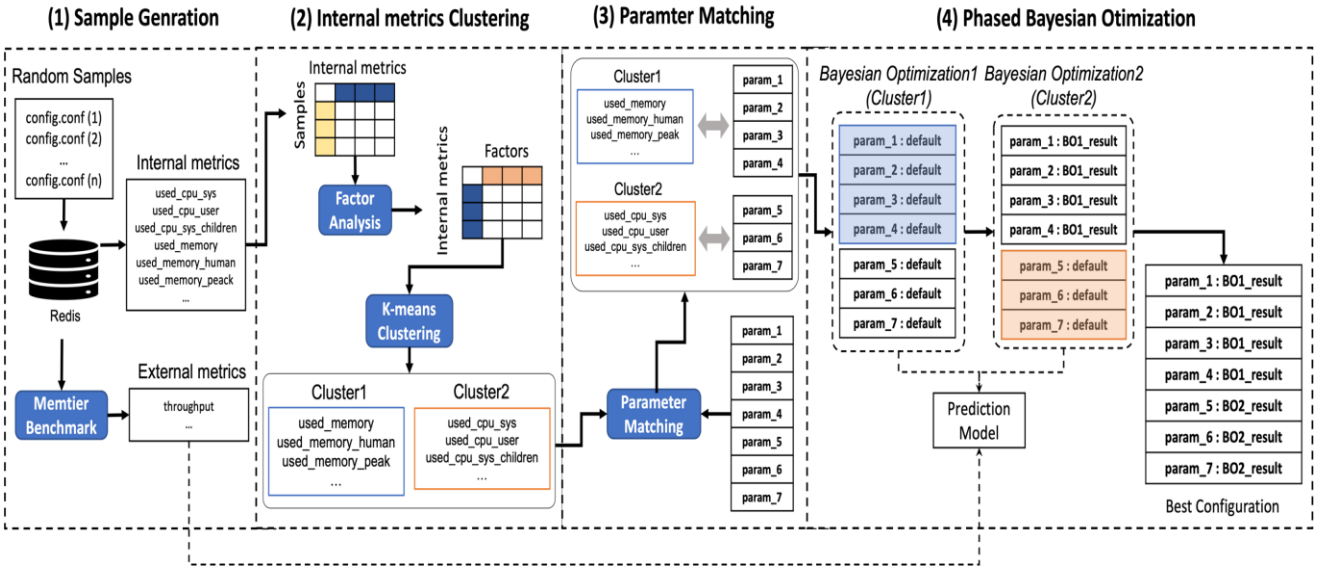
1. 서론

Redis[1]를 포함한 대부분의 데이터베이스에는 다양한 파라미터가 존재하고 조합에 따라 성능이 달라질 수 있다. 사용자는 목적에 맞는 파라미터 조합을 찾음으로써 데이터베이스의 성능 저하를 완화할 수 있다. 그러나 전문가가 직접 최고의 성능을 이끌어낼 수 있는 파라미터 조합을 찾는 데에는 한계가 있다. 수많은 파라미터들은 시스템의 모든 부분을 제어하기 때문에 각 파라미터의 영향을 고려하여 찾는 것은 현실적으로 불가능하다[2]. 또한, 특정 파라미터끼리 종속적이기 때문에 연관성도 고려해주어야 한다.

선행 연구에서는 BO(Bayesian Optimization)[3]를 이용하여 최적의 파라미터 값을 추출한다. BO 란 베이지안 이론(Bayesian Theory) 기반으로 사전(Prior) 데이터를 반영하여 목적함수(Objective-Function)를 최적화하는 기법이다. 입력값을 받는 미지의 목적함수를 상정하여, 함수값을 최대 또는 최소로 만드는 최적의 입력값 집합을 찾게 된다. 현재까지 탐색된 데이터를 토대로 목적함수를 추정하는 Surrogate 모델과 추정된 모델을 바탕으로 그 다음에 입력할 최적의 데이터를 추천하는 함수인 Acquisition 함수로 구성된다. 데이터베이스의 파라미터들을 목적함수의 입력값으로 설정하고 단위 시간당 처리량(Throughput) 또는 지연 시간(Latency) 등 최적화하려는 값을 결과값으로 설정하여 최적의 파라미터 조합을 추출해낼 수 있다.[1]

하지만 데이터베이스에 따라 제공하는 파라미터 개수가 많게는 수백 개에 이르기 때문에 이를 모두 BO 목적함수의 입력값으로 지정하게 되면 차원이 커지는 문제가 발생한다. 탐색 공간이 커지므로 공간 복잡도와 시간 복잡도가 올라가며 과적합이 발생할 수 있어, 이에 따라 필요한 학습 데이터가 많아지는 결과를 초래한다[4]. 이러한 문제점을 개선하기 위해서 본 연구에서는 통계적 기법과 기계학습을 통해 파라미터를 분류한 후 단계적으로 BO 를 진행하는 PBO(Phased Bayesian Optimization)를 제안한다. 요인 분석(Factor Analysis)과 K-평균 군집화 기법(K-means Clustering)을 사용해 Redis 의 내부 정보를 대상으로 군집화를 진행하고, 상관 계수를 이용해 파라미터를 매칭한다. 그리고 파라미터 분류별로 BO 를 진행하여

**교신저자(corresponding author)



(그림 1) PBO 진행 과정

탐색 공간을 줄인 후 값을 추출한다. PBO 를 통해 실제 성능이 향상되었는지 확인하기 위해 파라미터를 분류하지 않고 BO 를 진행한 결과와 default 값의 결과를 총 8 가지 회귀 모델을 이용하여 비교 실험하였다. 또한 회귀 모델별로 PBO 의 결과를 비교하였다. 실험 결과 PBO 방법이 BO 를 진행한 경우보다 단위 시간당 처리량이 모두 높았고 대부분의 모델에서 default 값의 결과보다 높게 나타나는 것을 확인하였다. 그리고 회귀 모델 중 LGBM 과 DT 모델에서 가장 높은 성능 결과가 나타나는 것을 확인하였다.

본 연구의 기여는 다음과 같다. 1) 통계적 기법을 이용한 파라미터 분류를 통해 BO 를 진행하여 탐색 공간을 줄여, 시간, 공간 복잡도를 낮춘다. 2) 차원이 줄어들기 때문에 학습 시 필요한 데이터 셋 양이 줄어든다. 3) 서로 종속성이 강한 파라미터들을 대상으로 단계적으로 BO 를 진행하여 Surrogate 모델의 예측 정확도가 높아진다.

2. 관련 연구

2.1 파라미터 튜닝

파라미터 튜닝이란 데이터베이스가 제공하는 파라미터 값을 변경하여 성능을 향상시키는 방법이다[5]. 데이터베이스의 성능은 설정된 파라미터 조합에 따라 쉽게 변화할 수 있다. 하지만 파라미터 종류가 다양하고 기능이 모두 다르기 때문에 사용자가 직접 값을 설정하는 것은 현실적으로 쉽지 않다. 사용하는 데이터베이스에 대한 전문적인 지식이 없다면 적절한 파라미터 값을 선택하는 것은 더욱 어려울 것이다. 이와 같은 문제를 해결하기 위해 기계학습을 통해 선형, 비선형적으로 값을 추천해주는 선행 연구가 존재한다[6]. 본 논문에서는 선행 연구와 같이 파라미터 개수가 수십 개인 Redis 를 대상으로 기계학습을 통해 값을 추천하는 방식을 제안한다.

2.2 파라미터 선별을 통한 차원 축소

[7]에서는 RocksDB 를 대상으로 10 개의 파라미터를 선별하여 차원을 축소하는 방법을 제안한다. 연구자의 경험을 토대로 내부(Internal) metrics 와 RocksDB 의 구조를 3 부분으로 나눈 후 가장 영향력 높은 10 개의 파라미터를 각 부분에 매칭시켜 BO 를 통해 값을 추출한다. 실험 결과 고차원 문제를 해결해 RocksDB 의 성능이 향상되었다. 하지만 내부 구조를 나누고, 각 구조에 파라미터를 매칭할 때 연구자의 주관이 들어가 객관성이 떨어진다. 이와 달리, 본 논문에서는 정해진 알고리즘대로 통계적인 방법을 사용하여 파라미터를 분류하는 방법을 제시한다.

3. 제안하는 모델

본 연구에서 제안하는 PBO 모델은 그림 1 과 같이 샘플 생성(Sample Generation), 내부 metrics 군집화(Internal metrics Clustering), 파라미터 매칭(Parameter Matching) 그리고 단계적 BO(Phased Bayesian Optimization)로 구성되며 결과적으로 데이터베이스의 성능을 최대로 끌어올릴 수 있는 최적의 파라미터 값을 추출한다.

3.1 학습 데이터 생성

파라미터들에 랜덤한 값을 할당하여 Redis configuration 파일을 생성한다. 그리고 각 파일을 통해 Memtier-Benchmark[8]를 실행해 내부 metrics 와 외부(External) metrics 값을 추출한다. 내부 metrics 는 Redis 서버 정보, 메모리 사용량과 같이 info 명령어로 조회되는 값이며, 외부 metrics 는 단위 시간당 처리량, 지연 시간 등 데이터베이스의 성능을 나타내는 값이다.

내부 metrics 는 군집화된 후 파라미터를 매칭시키는 데에 사용되고 외부 metrics 는 BO 의 최적화 대상으로 목적함수 내부 모델을 훈련시킬 때 사용된다.

3.2 내부 metrics 군집화

요인 분석을 실시하여 내부 metrics 에서 요인(Factor)들을 추출한다. 내부 metrics 간의 상관성을 기반으로 잠재된 공통 요인을 파악한다. 공통 요인 수는 1 이상의 고유값(Eigenvalue)을 갖는 요인으로 결정한다[9]. 그리고 요인 분석 결과를 바탕으로 K-평균 군집화를 진행해 내부 metrics 를 분류한다. K 값은 엘보우 기법(Elbow Method)을 통해 SSE(Sum of Squares for error) 값 차이가 가장 큰 지점을 선택한다[10]. SSE 는 군집 간의 거리 합을 나타내며 수식 (1)과 같이 계산된다.

$$SSE = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \qquad (1)$$

3.3 파라미터 매칭

파라미터마다 군집과의 상관관계를 바탕으로 연관성이 가장 높은 군집에 매칭시킨다. 군집과의 연관성은 군집에 속한 내부 metrics 전체와 파라미터 간의 상관계수를 구한 후 절댓값들의 평균값을 계산하여 판단한다.

$$r_{IMK} = \frac{cov(Param,\ IMK)}{\sigma_{Param}\sigma_{IMK}} \qquad (2)$$

$$\frac{|r_{IM1}|+|r_{IM2}|+\cdots+|r_{IMN}|}{Total\ number\ of\ Internal\ Metrics} \qquad (3)$$

수식 (2)는 파라미터와 내부 metrics 간의 상관계수를 구하는 수식이다. IMK 란 군집 내부의 k 번째 내부 metrics 를 의미한다. 파라미터와 IMK 의 공분산을 구하고, 각각의 표준편차 σ 의 곱을 나눠서 상관계수 γ_{IMK} 를 구한다. 특정 군집의 모든 내부 metrics 와 상관계수를 구한 다음 수식 (3)을 통해 절댓값의 평균을 계산하여 해당 군집 간의 연관성을 계산한다. 모든 군집마다 연관성을 파악해 가장 높은 값을 가지는 군집에 파라미터를 매칭한다. 모든 파라미터가 각 군집에 매칭되면 결과적으로 군집별로 파라미터가 분류된다.

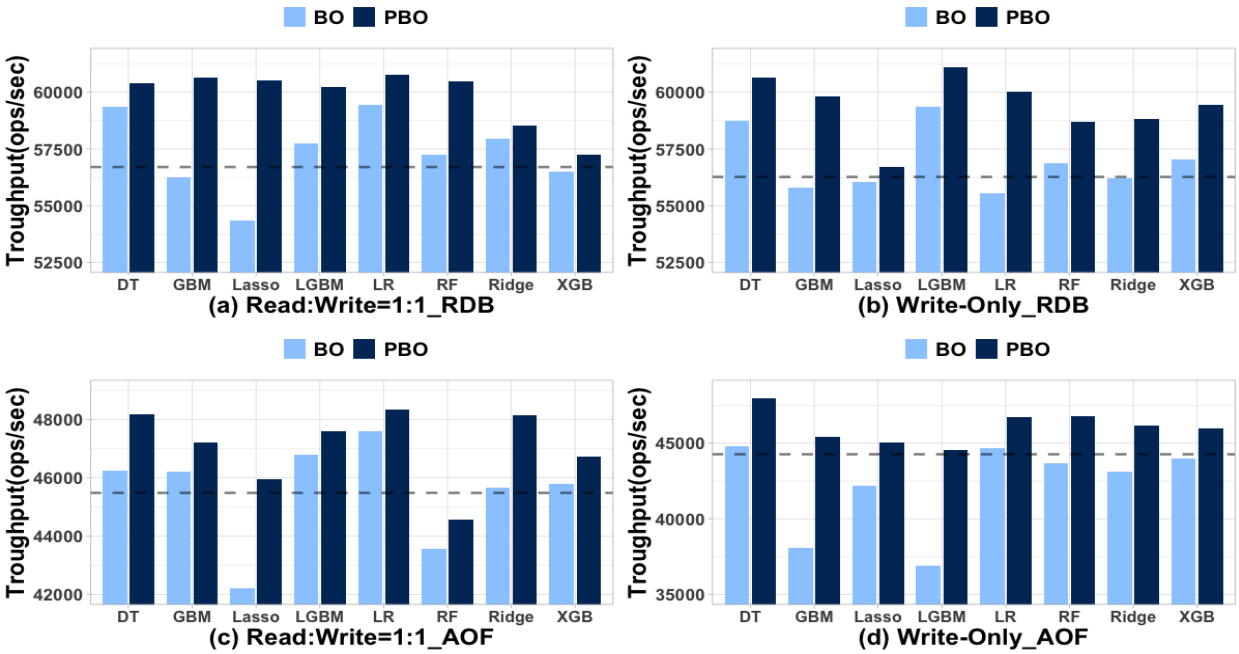
3.4 단계적 베이지안 최적화

파라미터 분류별로 BO 를 단계적으로 진행한다. 그림 1(4)와 같이 default 파라미터 값이 할당된 리스트에서 첫 번째 군집($Cluster1$)에 대해 첫 번째 BO 를 진행하여 값을 추출한다. 이때, 첫 번째 군집에 속하지 않은 나머지 파라미터들은 default 값을 유지한 채 예측 모델에 입력된다. 그리고 두 번째 BO 는 앞서 추출된 값을 고정하고, 두 번째 군집($Cluster2$)에 해당하는 파라미터 값으로 진행한다. 결과적으로, BO 는 단계적으로 군집 개수만큼 진행하여 최적의 파라미터 조합을 얻는다.

4. 실험 및 결과 분석

4.1 실험 환경

본 논문에서 제안하는 기법의 성능 향상을 측정하기 위해 표 1 과 같은 시스템 환경에서 실험을 진행하였다.



(그림 2) 워크로드별 데이터 처리 성능

<표 1> 시스템 실험 환경

OS	CentOS Linux release 7.6.1810 (Core)
CPU	Intel® Core™ i7-6700K CPU @ 4.00GHz
RAM	16384 MB
Redis Version	6.2.1

4.2 결과 분석

제안하는 방법론의 성능 향상을 확인하기 위해 Redis 의 default 성능과 분류되지 않은 파라미터로 BO 를 진행한 결과를 통해 비교 실험을 진행하였다. 워크로드는 Read-Write(1:1), Write-Only 에 대해서 Redis 의 지속성 기법 RDB 와 AOF 방식으로 구분하였으며, BO 의 예측 모델로 DT(Decision Tree), GBM- (Gradient Boosting Machine), Lasso, LGBM(LightGBM), LR(Linear Regression), RF(Random Forest), Ridge, XGB(XGBoost) 8 가지 회귀 모델을 통해 성능을 평가하였다. 또한 지속성 기법별로 워크로드의 평균 값을 계산해 회귀 모델을 비교하는 실험을 진행하였다.

그림 2 은 PBO 와 BO 그리고 default 성능 값을 두 워크로드와 지속성 기법을 구분하여 나타내었다. 그림 2(a), (b)는 RDB 방식에서의 비교 결과이다. PBO 가 BO 보다 모든 모델에서 높은 성능을 보였다. 그리고 BO 의 경우 default 성능 값과 비교했을 때 (a)에서 3 개, (b)에서 4 개의 모델이 낮은 성능을 가졌으며, PBO 는 모델 전체에서 더 높은 성능을 보였다. 그림 2(c), (d)는 AOF 방식에서의 비교 결과이다. PBO 가 BO 보다 전체 모델에서 높은 성능을 보였다. default 성능 값과 비교했을 때 PBO 는 RF 를 제외한 모든 모델에서 처리 성능이 높았으며, BO 는 (c)는 2 개, (d)는 6 개의 모델에서 낮은 성능을 가진다.

표 2,3 은 RDB, AOF 에서 PBO 결과를 8 가지 회귀 모델별로 두 워크로드의 평균값을 계산해서 나타내었다. 표 2 의 RDB 에는 GBM 계열의 비선형 모델인 LGBM 성능이 가장 높으며, 표 3 의 AOF 에서는 트리 계열의 비선형 모델인 DT 에서 가장 높은 성능을 보였다.

본 실험을 통해, 통계 기법으로 분류한 파라미터들에 BO 를 단계적으로 진행하는 PBO 방법이 최적의 파라미터 값을 도출하는 것을 확인하였다. 그리고 RDB 에서는 LGBM 과 AOF 에서는 DT 모델이 가장 높은 성능을 보이는 것을 확인하였다. 결과적으로 서로 연관성 있는 파라미터별로 분류하여 탐색 공간을 줄인 경우, 더 높은 성능 값을 도출하며 이때 LGBM 과 DT 예측 모델을 통해 가장 높은 성능을 이끌어 낼 수 있다.

<표 2> RDB 에서 PBO 모델별 결과 비교

	Read-Write(1:1)	Write-Only	Average
DT	60401	60648	60524.5
GBM	60629	59807	60218
Lasso	60538	56719	58628.5
LGBM	60218	61105	**60661.5**
LR	60766	60036	60401
RF	60467	58705	59586
Ridge	58515	58831	58673
XGB	57254	59432	58343

<표 3> AOF 에서 PBO 모델별 결과 비교

	Read-Write(1:1)	Write-Only	Average
DT	48181	47974	**48077.5**
GBM	47223	45391	46307
Lasso	45938	45073	45505.5
LGBM	47599	44520	46059.5
LR	48321	46739	47530
RF	44560	46795	45677.5
Ridge	48138	46167	47152.5
XGB	46738	45988	46363

5. 결론

본 논문에서는 BO 를 통해 파라미터 튜닝 작업 시, 파라미터를 분류하여 탐색 공간을 줄이는 연구를 수행하였다. 파라미터 전체를 탐색 공간으로 BO 를 진행하는 선행연구와는 달리 통계적 기법을 사용하여 파라미터를 분류한 후 BO 를 단계적으로 진행하는 PBO 방식을 제안하였다. 연구 결과 PBO 방식이 분류하지 않고 BO 를 진행한 경우보다 성능이 모두 높았으며, default 설정값과 비교했을 때도 단위 시간당 처리량이 대부분 높은 것을 확인하였다. 그리고 회귀 모델 중에서는 LGBM 과 DT 에서 가장 높은 성능이 나타나는 것을 확인하였다.

Acknowledgement

이 논문은 2021 년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (IITP-2017-0-00477, (SW 스타랩) IoT 환경을 위한 고성능 플래시 메모리 스토리지 기반 인메모리 분산 DBMS 연구개발)

참고문헌

[1] https://redis.io/
[2] 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, pp.1009-1024
[3] Snoek, Jasper, Hugo Larochelle, et al. "Practical bayesian optimization of machine learning algorithms.", Advances in neural information processing systems, 2012, 25
[4] Reunanen, Juha. "Overfitting in making comparisons between variable selection methods.", Journal of Machine Learning Research 3, 1371-1382, 2003
[5] Yong-Lak Choi, Byungkwon Yoon, and Kiwon Chong. "Database Management System Parameter Tuning Processes for Improving Database System Performance.", The Journal of Korean Institute of CALS/EC, vol. 7, no. 1, pp. 107-127, 2002
[6] Juyeon Seo, Jieun Lee, et al. "A Study on Redis Parameter Tuning Based on Non-linear Machine Learning.", The Korean Institute of Information Scientists and Engineers, 2021, 69-71
[7] Alabed, Sami, et al. "High-Dimensional Bayesian Optimization with Multi-Task Learning for RocksDB.",Proceedings of the 1st Workshop on Machine Learning andn Systems, 2021,pp. 111-119
[8] Memtier-Benchmark. https://github.com/RedisLabs/memtier_benchmark
[9] Yong, An Gie, and Sean Pearce. "A beginner's guide to factor analysis: Focusing on exploratory factor analysis.", Tutorials in quantitative methods for psychology 9.2, 2013, 79-94
[10] Likas, Aristidis, Nikos Vlassis, et al. Verbeek. "The global k-means clustering algorithm.", Pattern recognition 36.2, 2003, 451-461