|  |
| --- |
| Redis 파라미터 분류 및 단계적 Bayesian Optimization을 통한 파라미터 튜닝 연구 |
|  |
|  |
|  |
| A Study on Parameter Tuning for Redis via Parameter Classification and Phased Bayesian Optimization |
|  |
|  |
|  |
| **요 약**  DBMS 파라미터 튜닝이란 데이터베이스에서 제공하는 다양한 파라미터의 값을 조율하여, 최적의 성능을 도출하는 과정이다. 파라미터 튜닝은 성능 향상을 위해 사용하는 응용 소프트웨어를 변경하는 작업보다 요구되는 자원과 비용이 적다는 장점이 있다. 하지만 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 BO를 수행하는 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 8가지 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)[2]을 이용하여 최적의 파라미터 값을 추출한다. BO란 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.

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[3]. 이러한 문제점을 개선하기 위해서 본 연구에서는 통계적 기법과 기계학습을 통해 파라미터를 분류한 후 단계적으로 BO를 진행하는 PBO(Phased Bayesian Optimization)를 제안한다. 요인 분석(Factor Analysis)과 K-평균 군집화 기법(K-means Clustering)을 사용해 Redis의 내부 정보를 대상으로 군집화를 진행하고 상관 계수를 이용해 파라미터를 매칭한다. 그리고 탐색 공간을 줄이기 위해 각 파라미터 분류별로 BO를 진행하여 파라미터 값을 추출한다. PBO를 통해 실제 성능이 향상되었는지 확인하기 위해 파라미터를 분류하지 않고 BO를 진행한 결과와 default 값의 결과를 총 8가지 회귀 모델을 이용하여 비교 실험을 진행하였다. 실험 결과 PBO 방법이 BO를 진행한 경우보다 단위 시간당 처리량이 모두 높았고 대부분의 모델에서 default 값의 결과보다 높게 나타나는 것을 확인하였다.

본 연구에서 제안하는 PBO는 파라미터 분류별로 단계적으로 BO를 진행하여 탐색 공간을 줄이게 되며 모델 훈련 시 필요한 데이터 셋 양이 감소한다. 또한 종속성이 강한 파라미터들끼리 분류한 후 단계적으로 각 분류 별로 BO를 진행하기 때문에 surrogate 모델의 예측 정확도가 높아진다.

Diagram

Description automatically generated

(그림2) PBO 진행 과정

1. 관련 연구

**2.1 파라미터 튜닝**

튜닝이란 응용 프로그램, 데이터베이스 시스템, 그리고 운영체제의 조율을 통하여 시스템의 성능을 향상하기 위한 조율 과정이다[4]. 데이터베이스 시스템 튜닝은 정보 시스템의 성능을 향상시키기 위하여 수행하는 튜닝의 한 종류이며 파라미터 튜닝이 이에 해당된다.

파라미터 튜닝이란 데이터베이스가 제공하는 파라미터 값을 변경하여 성능을 향상시키는 방법이다. 데이터베이스의 성능은 설정된 파라미터 조합에 따라 쉽게 변화할 수 있다. 그러나 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[5]. 본 논문에서는 선행 연구와 같이 파라미터 개수가 수십 개인 Redis를 대상으로 기계학습을 통해 값을 추천하는 방식을 제안한다.

**2.2 파라미터 선별을 통한 차원 축소**

기계학습 모델을 사용하여 파라미터 값을 추출할 때 파라미터 종류가 많아지면 입력값에 대한 차원이 커지는 문제점이 있다. 그 결과 과적합이 발생할 수 있으며 이에 따른 필요한 학습 데이터 양이 늘어난다. 그리고 컴퓨터의 연산량이 많아지고 시간이 오래 걸리는 문제가 발생한다.

[6]에서는 RocksDB를 대상으로 10개의 파라미터를 선별하여 차원을 축소하는 방법을 제안한다. 내부(Internal) metrics와 연구자의 경험을 토대로 RocksDB의 구조를 3부분으로 나눈 후 가장 영향력 높은 10개의 파라미터를 각 부분에 매칭시켜 BO를 통해 값을 추출한다. 또한, RocksDB의 내부 구조를 나누는 과정과 파라미터를 선별하고 각 구조에 매칭시키는 과정에서 연구자의 주관적인 판단이 포함되기 때문에 객관성이 떨어진다. 이와 달리, 본 논문에서는 정해진 알고리즘대로 통계적인 방법을 사용하여 파라미터를 분류하는 방법을 제시한다.

1. 모델

본 논문에서는 학습 데이터를 구축한 후 내부 metrics를 대상으로 요인 분석과 K-평균 군집화를 진행한다. 그리고 각 군집과 모든 파라미터별로 상관분석을 통해 파라미터와 내부 metrics간의 연관성을 조사하여 상관계수가 가장 높은 군집으로 파라미터를 매칭한다. 그리고 분류된 파라미터들로 BO를 단계적으로 수행하여 데이터베이스의 성능을 최대로 할 수 있는 파라미터를 추출한다.

**3.1 학습 데이터 구축**

파라미터들에 랜덤한 값을 할당하여 Redis configuration 파일을 생성한다. 그리고 각 파일을 통해 Memtier-Benchmark[7]를 실행해 내부 metrics와 외부(External) metrics 값을 추출한다. 내부 metrics는 Redis 서버 정보, 메모리 사용량과 같이 info 명령어로 조회되는 값이며, 외부 metrics는 단위 시간당 처리량, 지연 시간 등 데이터베이스의 성능을 나타내는 값이다. 내부 metrics는 군집화된 후 파라미터를 매칭시키는 데에 사용되고 외부 metrics는 BO의 최적화 대상으로 목적함수 내부 모델을 훈련시킬 때 사용된다.

**3.2 내부 metrics 군집화**

요인 분석을 실시하여 내부 metrics에서 요인(Factor)들을 추출한다. 내부 metrics 간의 상관성을 기반으로 잠재된 공통 요인을 파악한다. 공통 요인 수는 1이상의 고유값(Eigenvalue)을 갖는 요인으로 결정한다[8]. 그리고 요인 분석 결과를 바탕으로 K-평균 군집화를 진행해 내부 metrics를 분류한다. K 값은 엘보우 기법(Elbow Method)을 통해 SSE(Sum of Squares for error) 값 차이가 가장 큰 지점을 선택한다[9]. SSE는 군집 간의 거리 합을 나타내며 수식 (1)과 같이 계산된다.

|  |  |
| --- | --- |
|  | (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.

|  |  |
| --- | --- |
|  | (2) |
|  | (3) |

수식 (2)는 파라미터와 내부 metrics 간의 상관계수를 구하는 수식이다. *IMK*란 군집 내부의 *k* 번째 내부 metrics를 의미한다. 파라미터와 *IMK*의 공분산을 구하고, 각각의 표준편차 의 곱을 나눠서 상관계수 를 구한다. 특정 군집의 모든 내부 metrics와 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 Phased Bayesian Optimization**

파라미터 분류별로BO를 단계적으로 진행한다. 그림**2**와 같이 default 파라미터 값이 할당된 리스트에서 첫 번째 군집(Cluster1)에 대해 첫 번째BO를 진행하여 값을 추출한다. 이때, 첫 번째 군집에 속하지 않은 나머지 파라미터들은 default 값을 유지한 채 예측 모델에 입력된다. 그리고 두 번째 BO는 앞서 추출된 값을 고정하고, 두 번째 군집(Cluster2)에 해당하는 파라미터 값으로 진행한다. 결과적으로, BO는 단계적으로 군집 개수만큼 진행하여 최적의 파라미터 조합을 얻는다.

1. 실험 및 결과 분석

**4.1 실험 환경**

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

<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 Redis의 default 성능과 분류되지 않은 파라미터로 BO를 진행한 결과를 통해 비교 실험을 진행하였다. 워크로드는 Read-Write(1:1), Write-Only에 대해서 Redis의 지속성 기법 RDB와 AOF 방식으로 구분하였으며, BO의 예측 모델로 DT(Decision Tree), GBM(Gradient Boosting Machine), Lasso, LGBM(Light-GBM), LR(Linear Regression), RF(Random Forest), Ridge, XGB(XGBoost) 8가지 회귀 모델을 통해 성능을 평가하였다.

<표 2> RDB에서 워크로드 별 성능 수치

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Read-Write(1:1) | | Write-Only | |
| BO | PBO | BO | PBO |
| DT | 59377 | 60401 | 58754 | 60648 |
| GBM | 56246 | 60629 | 55798 | 59807 |
| Lasso | 54363 | 60538 | 56056 | 56719 |
| LGBM | 57734 | 60218 | 59342 | 61105 |
| LR | 59429 | 60766 | 55539 | 60036 |
| RF | 57265 | 60467 | 56872 | 58705 |
| Ridge | 57964 | 58515 | 56212 | 58831 |
| XGB | 56484 | 57254 | 57047 | 59432 |

<표 3> AOF에서 워크로드 별 성능 수치

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Read-Write(1:1) | | Write-Only | |
| BO | PBO | BO | PBO |
| DT | 46237 | 48181 | 44777 | 47974 |
| GBM | 46215 | 47223 | 38096 | 45391 |
| Lasso | 42204 | 45938 | 42195 | 45073 |
| LGBM | 46798 | 47599 | 36879 | 44520 |
| LR | 47610 | 48321 | 44658 | 46739 |
| RF | 43553 | 44560 | 43649 | 46795 |
| Ridge | 45654 | 48138 | 43128 | 46167 |
| XGB | 45805 | 46738 | 44005 | 45988 |

**표 2, 3은 Redis의 지속성 방법과 워크로드를 구분하여 단위 시간당 처리량을 표기하였다. 비교 결과 RDB와 AOF 작동 방식에서 두 워크로드 별로 모두 PBO가 BO 보다 높은 성능을 보였다. ~~~~표 2, 3은 Redis의 지속성 방법과 워크로드를 구분하여 단위 시간당 처리량을 표기하였다. 비교 결과 RDB와 AOF 작동 방식에서 두 워크로드 별로 모두 PBO가 BO 보다 높은 성능을 보였다. ~~~~**

**그림 3은 RDB, AOF 작동 방식 별로 두 워크로드를 구분하여 PBO와 BO 그리고 default 성능 값과 비교한 결과이다. 그림 3의 점선이 default 수치 값이다. 그림 3 (a), (b)는 RDB 방식에서의 비교 결과이다. PBO가 BO보다 모든 모델에서 높은 성능을 보였다. 그리고 BO의 경우 default 성능 값과 비교했을 때 (a)에서 3개, (b)에서 4개의 모델이 낮은 성능을 가졌으며, PBO는 모델 전체에서 더 높은 성능을 보였다. 그림 3(c), (d)는 AOF 방식에서의 비교 결과이다. PBO가 BO보다 전체 모델에서 높은 성능을 보였다. default 성능 값과 비교했을 때 PBO는 RF를 제외한 모든 모델에서 처리 성능이 높았으며, BO는 (c)는 2개, (d)는 6개의 모델에서 낮은 성능을 가진다. 본 실험을 통해, K-평균 군집화를 통해 BO를 단계적으로 진행하는 PBO방법이 최적의 파라미터 값을 도출하는 것을 확인하였다.**

Chart, bar chart

Description automatically generated

(그림3) 워크로드별 데이터 처리 성능

1. 결론

본 논문에서는 BO를 통해 파라미터 튜닝 작업 시, 파라미터를 분류하여 탐색 공간을 줄이는 연구를 수행하였다. 파라미터 전체를 탐색 공간으로 BO를 진행하는 선행연구와는 달리 통계적 기법을 사용하여 파라미터를 분류한 후 BO를 단계적으로 진행하는 PBO방식을 제안하였다. 연구 결과 PBO 방식이 분류하지 않고 BO를 진행한 경우보다 성능이 모두 높았으며, default 설정값과 비교했을 때도 단위 시간당 처리량이 대부분 높은 것을 확인하였다.

**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

[2] Snoek, Jasper, Hugo Larochelle, and Ryan P. Adams. "Practical bayesian optimization of machine learning algorithms.", Advances in neural information processing systems, 2012, 25

[3] Reunanen, Juha. "Overfitting in making comparisons between variable selection methods.",  Journal of Machine Learning Research 3, 1371-1382, 2003

[4] 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

[5] 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

[6] Alabed, Sami, and Eiko Yoneki. "High-Dimensional Bayesian Optimization with Multi-Task Learning for RocksDB.", Proceedings of the 1st Workshop on Machine Learning andn Systems, 2021, pp. 111-119

[7] Memtier-Benchmark. [https://github.com/RedisLabs/mem](https://github.com/RedisLabs/memt)

  tier\_benchmark

[8] 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

[9] Likas, Aristidis, Nikos Vlassis, and Jakob J. Verbeek. "The global k-means clustering algorithm.", Pattern recognition 36.2, 2003, 451-461