PDTune: 파라미터 간 의존성을 고려한 데이터베이스 파라미터 최적화

백슬기[01], 권세인[2], 김휘군[2], 이지은[2], 박상현[2†]
동국대학교 컴퓨터공학과[1],
연세대학교 컴퓨터과학과[2]

seulgi.baek@dgu.ac.kr, {seinkwon97, jinhuijun, jieun199624, sanghyun}@yonsei.ac.kr

PDTune: Database Parameter Dependency-Aware Optimization

Seulgi Baek[01], Sein Kwon[2], Huijun Jin[2], Jieun Lee[2], Sanghyun Park[2†]
Dept. of Computer Science, Dongguk University[1]
Dept. of Computer Science, Yonsei University[2]

요 약

데이터베이스의 성능을 높이기 위해서는 데이터베이스 내 다양한 파라미터 값을 적절히 조정하는 데이터베이스 파라미터 튜닝 작업이 필요하다. 데이터베이스 튜닝을 효과적으로 수행하기 위해서는 최적의 파라미터 조합에 근사한 탐색공간으로 빠르고 효율적으로 수렴할 수 있는 모델이 요구된다. 데이터베이스 파라미터는 독립적으로 작용하지 않고 상호 간 의존성을 가지며 성능 변화에 영향을 미치지만 기존 기계학습 기반 튜닝 모델은 파라미터 간 복잡한 의존성 관계를 충분히 반영하지 못한다는 한계점이 있다. 본 논문에서는 풍부한 도메인 지식을 가진 LLM을 활용해 데이터베이스 파라미터 최적화 과정에서 파라미터 간 의존성을 파악하고, 이를 탐색 방향에 반영하여 탐색 효율성을 높이는 PDTune 모델을 제안한다.

1. 서 론

스마트 시티에서는 생산 및 활용되는 데이터 규모가 폭발적으로 증가하였고 최근 빅데이터 활용이 급증하면서 대규모 데이터를 효율적으로 관리하고 처리하는 데이터베이스 관리 시스템의 중요성이 더욱 커지고 있다[1]. 이에 따라 고성능 데이터베이스에 대한 요구가 증가하고 있으며, 최적의 파라미터 조합을 탐색하는 데이터베이스 파라미터 튜닝은 성능 향상의 핵심 방법론으로 주목받고 있다. 기존에는 데이터베이스 관리자(Database Administrator, DBA)가 수동으로 파라미터를 조정했지만, 파라미터 수가 많아지고 시스템 환경이 자주 바뀌면서 튜닝을 새롭게 진행해야 하는 비효율성이 존재한다.

이러한 한계를 해결하기 위해, 최근에는 기계학습 기반 자동 파라미터 최적화 연구가 진행되고 있다[2,3,4]. 대표적인 모델로 OtterTune은 고차원 검색 공간을 축소하기 위해 Lasso 알고리즘을 이용해 주요 파라미터를 선별하고[2], CDBTune은 DQN(Deep-Q-Network)과 액터-크리틱이 혼합된 DDPG(Deep Deterministic Policy Gradient) 알고리즘을 사용하여 파라미터 최적화를 진행한다[3]. RGPE 방법론은 베이지안 최적화(Bayesian Optimization, BO) 기반 앙상블 모델로 워크로드 정보와 과거 파라미터 최적화 수행 정보를 혼합하는 방법을 제안한다[4]. 그러나 기존 방법론은 데이터베이스 파라미터 간의 복잡한 의존성 관계를 충분히 반영하지 못하며[2,5], DBA나 커뮤니티로부터 축적된 도메인 지식들을 활용하지 못한다는 한계가 있다[6]. 이로 인해 튜닝 과정에서 중요한 파라미터 또는 조합을 초기에 식별하지 못해 넓은 탐색 공간을 비효율적으로 탐색하는 문제가 있다.

*이 논문은 2025년도 정부(과기정통부)의 재원으로 한국연구재단의 지원(No. RS-2023-00229822)과 국토교통부의 스마트시티 혁신인재육성사업으로 지원을 받아 수행된 연구임.

†교신 저자: sanghyun@yonsei.ac.kr

본 연구는 기존 방법론의 한계를 극복하기 위해 BO 과정 중 성능이 일정 수준 이상 향상된 시점을 탐지하고, 해당 시점의 전후 파라미터 구성 및 성능 정보를 대규모 언어 모델(Large Language Model, LLM)에 입력하여 파라미터 간의 동적 의존성을 추론한다. LLM은 학습된 도메인 지식을 기반으로 파라미터 간 의존 관계를 도출하고, 이를 BO 탐색 방향에 반영하여 보다 효과적인 최적화 탐색 방향을 유도해 기존 방법론의 한계를 보완한다.

2. 모델 구조

본 연구는 LLM을 활용해 BO 탐색 과정 중 성능이 향상되는 구간의 파라미터 의존성을 분석하고, 이를 탐색 방향에 반영하는 PDTune을 제안한다. PDTune은 BO 탐색 중 특정 임계치 이상의 성능 향상이 발생하는 구간에서 성능 향상 전후의 파라미터 조합과 성능 데이터를 LLM 입력으로 제공하여 파라미터 간 의존성을 추론한다. 추론 결과를 기반으로 파라미터 간 의존성 점수를 계산하고 이를 BO의 획득 함수에 가중치로 적용해 파라미터 간 의존성을 반영한 방향으로 탐색을 유도한다.

2.1 워크로드 인코딩(Workload Encoding)

튜닝 프로세스 시작 단계에서는 특정 워크로드에 대한 튜닝 요청을 수신하고, 해당 워크로드를 수치형 특성 벡터로 인코딩한다. 본 연구에서는 단일 SQL 문장에서 쿼리 명령어의 구조적 특징을 추출한 후, 이를 기반으로 여러 SQL 문장의 특성을 하나의 워크로드 벡터로 통합한다. 사용된 SQL 특성은 **표 1**에 제시되며, 쿼리 타입은 'SELECT', 'UPDATE', 'INSERT', 'DELETE', 'CREATE', 'ALTER', 'DROP' 7가지로 구성된다. 단일 워크로드는 각 SQL 문장에서 추출된 특성 벡터의 평균값을 쿼리 타입 비율과 결합하여 수식 (1)과 같이 워크로드를 대표하는 단일 벡터로 생성된다. 이 벡터는 특성 평균(12차원)과 쿼리 타입 비율(7차원)을 접합해 총 19차원으로 표현된다.

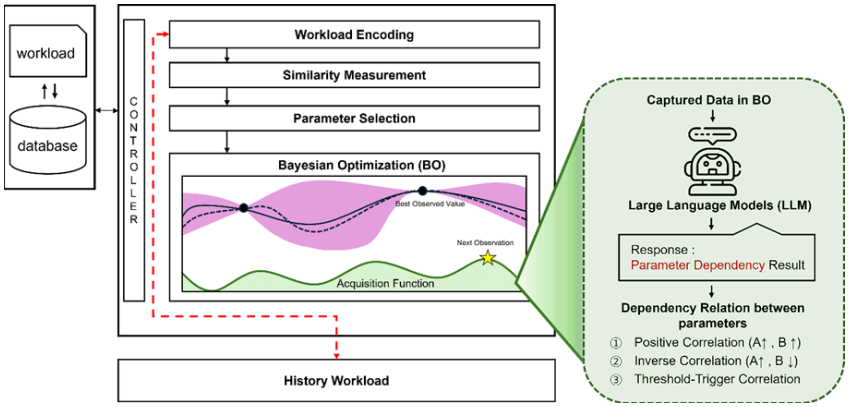


그림 1 PDTune 흐름도

2.2 워크로드간 유사도 계산(Similarity Measurement)

현재 워크로드 트성 벡터와 과거 워크로드들의 특성 벡터 간 코사인 유사도를 계산한다. 계산된 코사인 유사도 점수를 기반으로, 현재 워크로드와 가장 유사한 워크로드를 가져온다.

표 1 SQL 인코딩 시 사용된 주요 특성

특성	설명	인덱스
쿼리타입	One-hot Encoding 방식으로 7가지 쿼리 타입 유형 구분	0-6
테이블 수	테이블 개수	7
JOIN 수	JOIN 사용 횟수	8
WHERE 절	WHERE 조건절에 사용된 조건문의 개수	9
GROUP BY	GROUP BY 사용 여부 (0: 미사용, 1: 사용)	10
ORDER BY	ORDER BY 사용 여부 (0: 미사용, 1: 사용)	11

$$Workload = Mean(SQLVector) \bullet QueryTypeRatio \quad (1)$$

2.3 파라미터 선정(Parameter Selection)

BO는 고차원 공간에서의 탐색 시 차원의 저주 문제가 발생할 수 있으므로[1], 데이터베이스 성능에 큰 영향을 미치는 상위 10개의 주요 파라미터에 대해서만 최적화를 수행한다. 이를 위해 XGBoost[7] 기반 성능 예측 모델을 학습시키고, SHapley Additive exPlanations(SHAP) 알고리즘[8]을 사용해 성능에 중요한 영향을 미치는 상위 10개 파라미터를 추출한다.

2.4 파라미터 최적화

2.4.1 BO를 이용한 데이터베이스 파라미터 최적화

본 연구에서는 파라미터 간 의존성을 반영한 가중치를 BO 과정 내 획득 함수에 적용한다. 의존성 점수로 획득함수를 보정해, 의존성이 높은 파라미터 조합 우선 탐색을 유도한다.

$$biasAcq(x) = acq(x) \times (1 + \alpha \times dependencyScore) \quad (2)$$

수식 (2)에서 α는 의존성 함수의 가중치 영향력을 조정하는 하이퍼파라미터이고, $acq(x)$은 Expected Improvement(EI) 함수[9]를 기반으로 한 획득 함수를 의미한다.

2.4.2 BO 탐색 중 LLM을 활용한 파라미터 간 의존성 추론

BO 과정 중 i와 $i+1$번째 반복 사이에 성능이 1.2배 이상 향상되는 경우, 해당 시점의 파라미터 구성과 성능 데이터를 LLM의 입력으로 제공한다. 이때 LLM에게 전달되는 프롬프트는 **그림 2**의 구조를 갖는다. LLM은 입력 데이터를 기반으로 파라미터 간 의존성을 추론해 상호 의존관계를 갖는 파라미터 A, B와 의존성 유형 정보를 반환한다. 데이터베이스 파라미터간 의존성 유형은 두 파라미터가 함께 증가하거나 감소하는 Positive Correlation, 한 파라미터의 증가가 다른 파라미터의 감소로 이어지는 Inverse Correlation, 한 파라미터가 특정 임계치(T)를 초과할 때 다른 파라미터의 변화가 촉발되는 Threshold-Triggered Correlation으로 구분하여 정의한다.

```
Task:
Infer that parameter dependency pattern that explains the performance
improvement, based on two snapshots of {configuration, performance} data.
Configuration:
Use scaled parameter values and performance metrics. Choose one pattern
type from the defined options (e.g., POSITIVE, INVERSE, THRESHOLD,
NOTHING).
Output:
Return the pattern type, involved parameters and reasoning in a
structured format.
```

그림 2 LLM 프롬프트 구성

본 연구에서는 파라미터 간 의존성 관계를 반영하기 위해 새로운 $dependencyScore$ 함수를 설계해 제안한다. 수식 (3)의 A_i, B_i는 i번째 BO 반복지점에서 파라미터 A, B 값을 나타낸다.

$$\Delta A = (A_{i+1} - A_i),\ \Delta B = (B_{i+1} - B_i) \quad (3)$$

의존성 유형에 따른 가중치 계산 시 Positive Correlation은 수식 (4), Inverse Correlation은 수식 (5), Threshold-Triggered Correlation은 수식 (6)에 해당하는 $dependencyScore$ 함수를 채택한다.

$$dependencyScore = \begin{cases} \exp(-\alpha \times (\Delta A - \Delta B)^2) & (4) \\ \exp(-\alpha \times (\Delta A + \Delta B)^2) & (5) \\ \begin{cases} \text{if } (A_{curr} > T) & \exp(-\gamma \times (\Delta B - \theta)^2) \\ else & 0 \end{cases} & (6) \end{cases}$$

여기서 α, β, γ는 3가지 유형의 의존성 함수에 곱해지는 민감도 조절 계수로, 값이 클수록 입력값 간의 미세한 차이에도 점수변화가 크게 발생한다. θ는 B의 변화량(ΔB)이 일정 수준 이상일 때만 의존성이 있다고 판단하기 위한 기준값이다. T와 θ값은 LLM 추론 결과의 출력으로 제공된다. 계산된 스코어는 각 파라미터 구성 당 하나의 스칼라 값으로 산출되며, 획득 함수에 가중치로 곱해져 해당 의존성을 만족하는 파라미터 구성이 다음 탐

색 후보로 선택될 확률을 높이는 방식으로 적용된다. 이를 통해 파라미터 간 의존성을 반영한 파라미터 조합을 우선 탐색하도록 유도하는 역할을 한다.

3. 실험 및 결과

3.1 실험 환경

본 연구에서는 MySQL 5.7 버전을 사용하여 데이터베이스 성능 최적화 실험을 수행하였고 벤치마크 도구로 Benchbase를 사용하였다. 실험에 사용된 워크로드는 YCSB 기반 A, E, F 시나리오이며, 구체적인 구성은 **표 2**에 제시하였다. 성능 지표는 처리량(Throughput)과 지연시간(Latency)을 사용하였다. BO 탐색 과정에서 탐색된 파라미터에 대한 평가 함수로는 $(2*Throughput)/Latency$을 사용하였다. 의존성 함수의 민감도 계수 α,β,γ는 각 30으로 설정했다. 획득 함수는 EI를 사용하였고, BO 반복 횟수는 500회로 고정하여 실험을 수행하였다.

표 2 워크로드 환경에 따른 데이터셋 구성

WORKLOAD_INDEX	A	E	F
Scale Factor	12000	12000	12000
ReadRecord	50	0	50
InsertRecord	0	5	0
ScanRecord	0	95	0
UpdateRecord	50	0	0
DeleteRecord	0	0	0
ReadModify WriteRecord	0	0	50

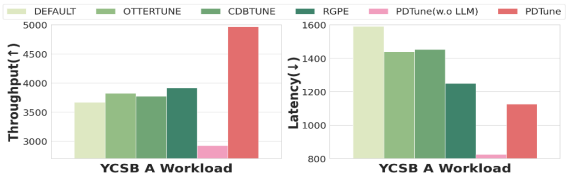
3.2 실험 결과

3가지 워크로드에 대해 YCSB 기반 A, E, F 워크로드를 대상으로 MySQL 기본 설정(DEFAULT), OtterTune, CDBTune, RGPE 베이스라인 모델, PDTune(w.o LLM), 그리고 제안 기법인 PDTune 모델의 성능을 비교 평가했다. PDTune(w.o LLM)은 제안 모델에서 LLM을 활용한 파라미터 의존성 추론이 제외된 모델[9]이다.

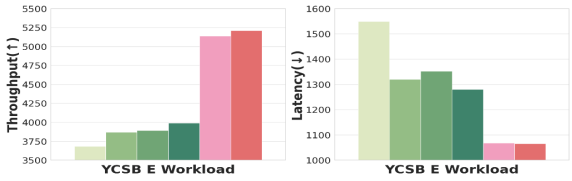
그림 3의 성능 비교 결과에 따르면 PDTune은 모든 워크로드에서 기존 베이스라인 모델 및 PDTune(w.o LLM) 대비 가장 높은 처리량과 가장 낮은 지연 시간을 기록했다. YCSB A 워크로드에서는 기존 베이스라인 모델 중 가장 우수한 성능을 보인 RGPE와 비교해도 처리량은 26.8% 향상되었고, 지연시간은 34% 감소했다. YCSB E 워크로드에서도 PDTune은 RGPE 대비 30.6% 높은 처리량과 16.8% 낮은 지연시간을 달성해 베이스라인 모델과 비교했을 때 월등히 높은 성능을 보였다. 또한, PDTune은 PDTune(w.o LLM)과 비교하여 워크로드 A에서 약 77.7%의 처리량 향상, 27%의 지연시간 감소를 보였고, 다른 워크로드에서도 PDTune(w.o LLM)과 비교했을 때 PDTune이 모든 워크로드에 대해 더 좋은 성능 기록했다. 이는 파라미터 간 의존성을 반영하여 데이터베이스 파라미터 최적화를 진행하는 방식이 다양한 워크로드 환경에서 성능 개선에 기여함을 검증한다.

4. 결 론

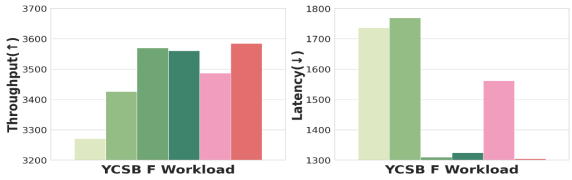
본 연구는 기존 모델들이 파라미터 튜닝 수행시 파라미터 간 복잡하고 동적인 의존성 관계를 충분히 반영하지 못하는 문제를 해결하고자 하였다. 이를 위해 BO 탐색 과정 중 성능이 일정 수준 이상 향상된 시점 전후의 파라미터 구성과 성능 데이터를 LLM에 입력하여 파라미터 간 의존성을 추론하고, 이를 BO 탐색 방향에 반영하는 모델 PDTune을 제안한다. 기존 베이스라인 모델과 PDTune(w.o LLM)과의 성능 비교 실험을 진행함으로써



(a) YCSB A Workload 성능 비교



(b) YCSB E Workload 성능 비교



(c) YCSB F Workload 성능 비교

그림 3 YCSB 워크로드 A, E, F에 대한 PDTune과 베이스라인 모델 성능 비교

LLM의 성능 영향도를 검증했다. 본 연구는 LLM의 광범위한 도메인 지식과 추론 능력을 활용하여 파라미터 간 의존성을 포착하고, 성능에 실질적 영향을 미치는 요인을 보다 정밀하게 탐색 과정에 반영한다는 점에서 의의를 갖는다. 향후 다양한 DBMS 환경에서도 적용할 수 있도록 범용성을 강화하고, 기존의 LLM 기반 튜닝 기법들과의 비교 실험을 통해 제안한 모델의 성능을 체계적으로 검증할 계획이다.

참고문헌

[1] Lu.Jiaheng, et al. "Automatic database management system tuning through large-scale machine learning."Lu.Jiaheng, et al. "Speedup your analytics: Automatic parameter tuning for databases and big data systems." *Proceedings of the VLDB Endowment* 12.12(2019): 1970-1973.

[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.

[3] Zhang, Ji, et al. "An end-to-end automatic cloud database tuning system using deep reinforcement learning." *Proceedings of the 2019 international conference on management of data*. 2019.

[4] Feurer, Matthias, Benjamin Letham, and Eytan Bakshy. "Scalable meta-learning for bayesian optimization using ranking-weighted gaussian process ensembles." *AutoML Workshop at ICML*. Vol. 7. 2018.

[5] Duan, Songyun, Vamsidhar Thummala, and Shivnath Babu. "Tuning database configuration parameters with ituned." *Proceedings of the VLDB Endowment* 2.1 (2019): 1246-1257.

[6] Lao Jiale, et al. "GpTuner: A manual-reading database tuning system via gtp-guided bayesian optimization." *arXiv preprint arXiv:2311.03157* (2023).

[7] 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.

[8] Lundberg, Scott M., and Su-In Lee. "A unified approach to interpreting model predictions." *Advances in neural information processing systems* 30 (2017).

[9] Frazier, Peter I. "A tutorial on Bayesian optimization." *arXiv preprint arXiv:1807.02811* (2018).