计算机科学与工程学院
School of computer science and engineering

E2ETune: End-to-End Knob Tuning via Fine-tuned Generative Language Model

VLDB'25

Xinmei Huang[1], Haoyang Li[1], Jing Zhang[1], Xinxin Zhao[1], Zhiming Yao[1], Yiyan Li[1], Tieying Zhang[2], Jianjun Chen[2], Hong Chen[1], Cuiping Li[1]

[1] Renmin University of China

[2] ByteDance Inc

Reporter: Yuxiang Lu

2025.9.18

作者团队

研究方向

- Big data analysis and mining
- Machine learning-based database systems
- Deep learning
- Graph representation learning

科研成果

- [ICLR’25] Streamlining Redundant Layers to Compress Large Language Models
- [TKDE’25]LIOF: Make the Learned Index Learn Faster with Higher Accuracy
- [ACL’24] SP3: Enhancing Structured Pruning Via PCA Projection

陈红
中国人民大学
信息学院教授

李翠平
中国人民大学
信息学院教授

目录

目录

背景介绍

数据库(DataBase, DB)

- 数据是计算机进行计算、执行任务的基础，而数据库就是存放数据的仓库，其本质上是一个以某种方式组织、存储和管理数据的集合。可以将其想象成一个高度结构化的电子化文件柜。

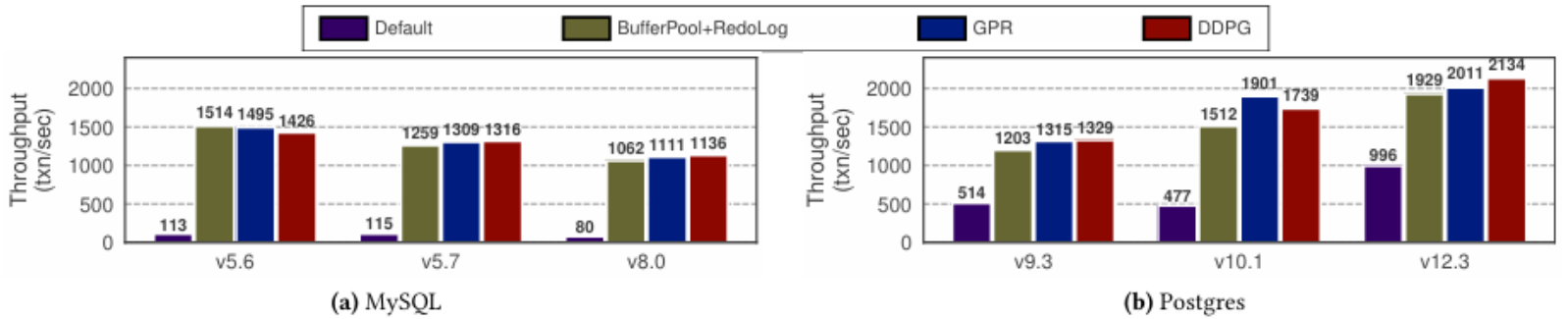
数据库中的knobs

- 在现代数据库的管理系统中有成百上千个可调节的Knobs，代表着对数据库不同方面状态的控制。
- Knobs取值的组合情况决定数据库系统在处理到来的工作负载时的性能表现。

Knobs名称	Knobs功能
innodb_buffer_pool_size、shared_buffers	内存管理
optimizer_switch、work_mem	查询优化策略
sync_binlog、wal_level	同步策略控制
innodb_thread_concurrency、max_worker_processes	线程数控制

常见的Knobs及功能

背景介绍



Knobs调优前后的性能对比[1]

图中统计的数据是在两个不同的数据库实例上处理同一种工作负载TPC-C得到的，实验采用了默认配置和三种经过算法调优后的配置

通过Knobs调优选择更优的配置对提升数据库的性能是效果显著的！

1 数据统计来自于[VLDB'21] An Inquiry into Machine Learning Based Automatic Configuration Tuning Services on Real-World Database Management Systems

目录

相关工作与动机

现有的Knob调优方法可以分为四类：

- Heuristic-based methods

  基于[illegible]工制作的规则或一组预定义的启发式规则来迭代探索搜索空间。

- Ba[illegible]-based methods

  [illegible]模型从给定的特征评估数据库的性能指标，通过不断迭代获[illegible]的合适的配置。

- Reinf[illegible]ning (RL)-based methods

  基于强化学习的Knobs调优方法使用强化学习算法来调整Knobs。具有actor-critic框架的深度确定性策略梯度（Deep Deterministic Policy Gradient, DDPG）方法是常用的方法，通过actor选择配置，critic评估有效性的迭代过程生成合适的配置。

- Deep Learning (DL)-based methods

  基于DL的Knobs调优方法将神经网络训练为成本模型，根据给定特征预测数据库性能，以此促进快速的配置迭代探索。

现有的调优方法都采用多次迭代的方法寻找最优配置

推荐配置 -> 执行工作负载 -> 获取性能反馈 -> 更新模型 -> 推荐新配置...

相关工作与动机

为解决多次迭代搜索带来的效率和成本问题，研究人员开发出了多种方法来缩短迭代的过程，包括：

1. 工作负载映射：使用过去的调优数据来改善基于BO的调优器的初始状态。
2. 模型预训练：使用历史数据预训练RL模型，然后在新的工作负载上对其进行微调，以更快地收敛。
3. 模型集成：将过去任务中的模型组合起来以适应新的工作负载。
4. Knob剪枝：通过根据以往经验选择关键Knob及其最优范围来细化配置空间，减少搜索空间。

局限：如何解决遇到新负载就需要在线迭代的问题？进一步提升调优效率？

相关工作与动机

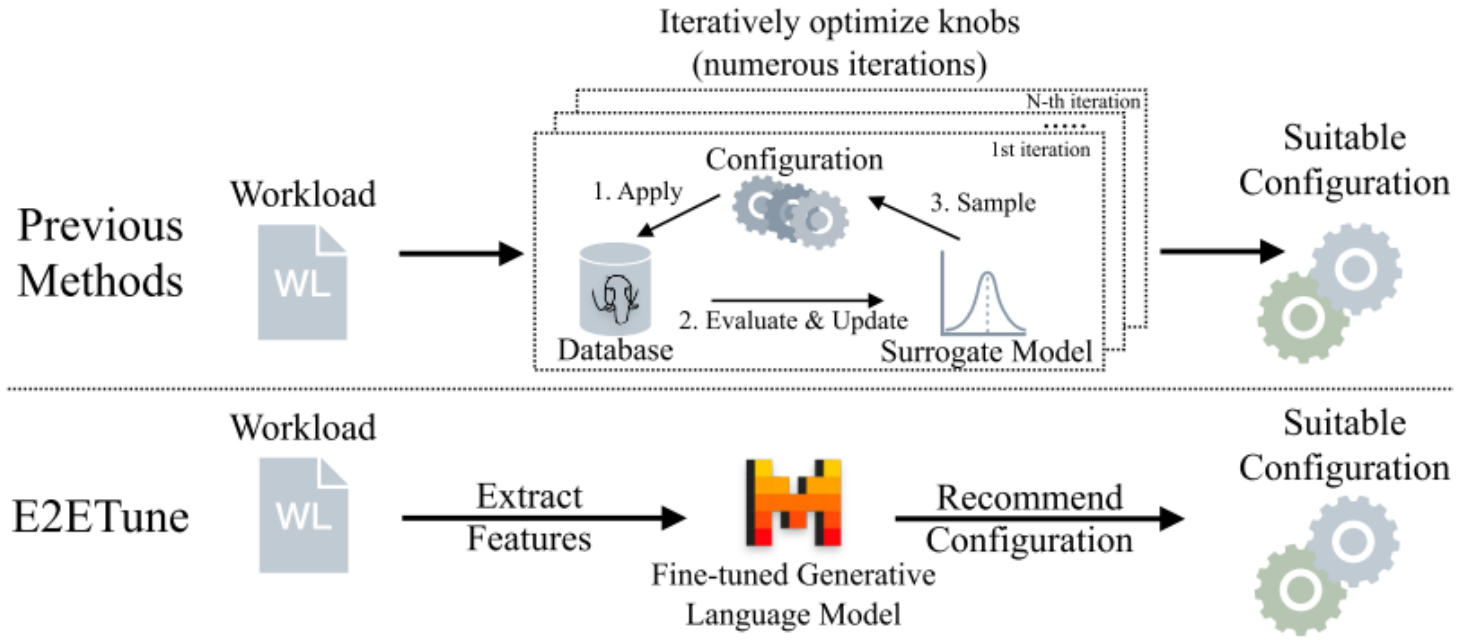
- 为了彻底解决多次迭代带来的效率低下问题，作者希望开发一种端到端的调优方法，改变在线的“搜索问题”，避免配置Knobs过程中的大量迭代。

- 通过观察，作者发现熟练的数据库管理员经常能够利用他们丰富的经验，手动为新的工作负载确定合适的配置。

机遇：通过训练一个能够理解分布映射关系的模型，建立一个端到端的Knob调优解决方案

相关工作与动机

- 在开发能够理解分布映射关系的模型时，作者首先尝试了传统的机器学习算法，如多层感知机（MLP）和随机森林，发现由于其有限的学习能力，不能有效地从训练数据中学习分布映射。
- 因此作者设计了一个能力更强大的解决方案：以端到端生成语言模型（如GPT-4和LLaMA-3）强大的建模能力为基础，对语言模型进行微调以学习和捕获复杂的分布映射关系。



多次迭代

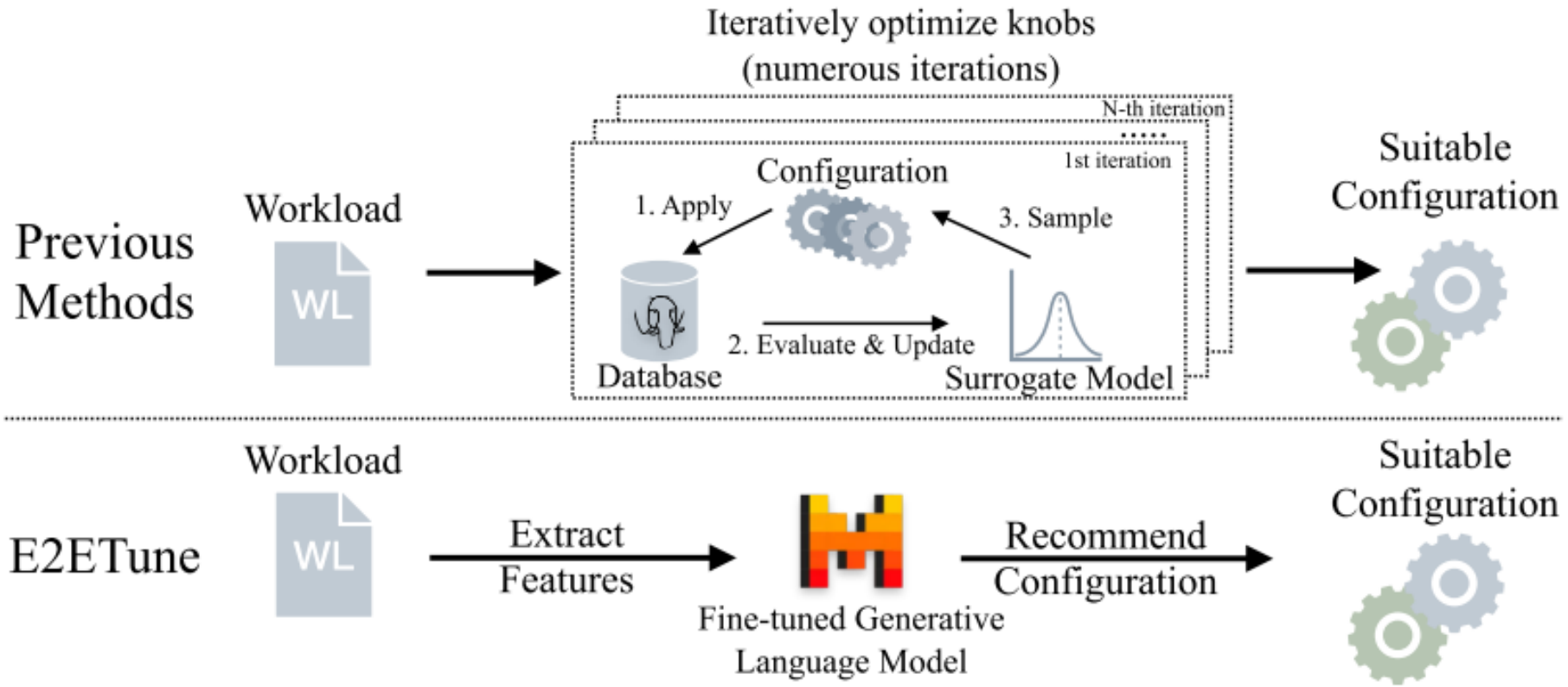
直接预测

Previous knob tuning methods *vs.* E2ETune

目录

系统设计

本文提出了一种新颖的数据库Knob调优方法E2ETune，通过使用历史数据将Knob调优过程建模为端到端的序列生成任务，避免配置过程中的迭代



Previous knob tuning methods *vs.* E2ETune

系统设计

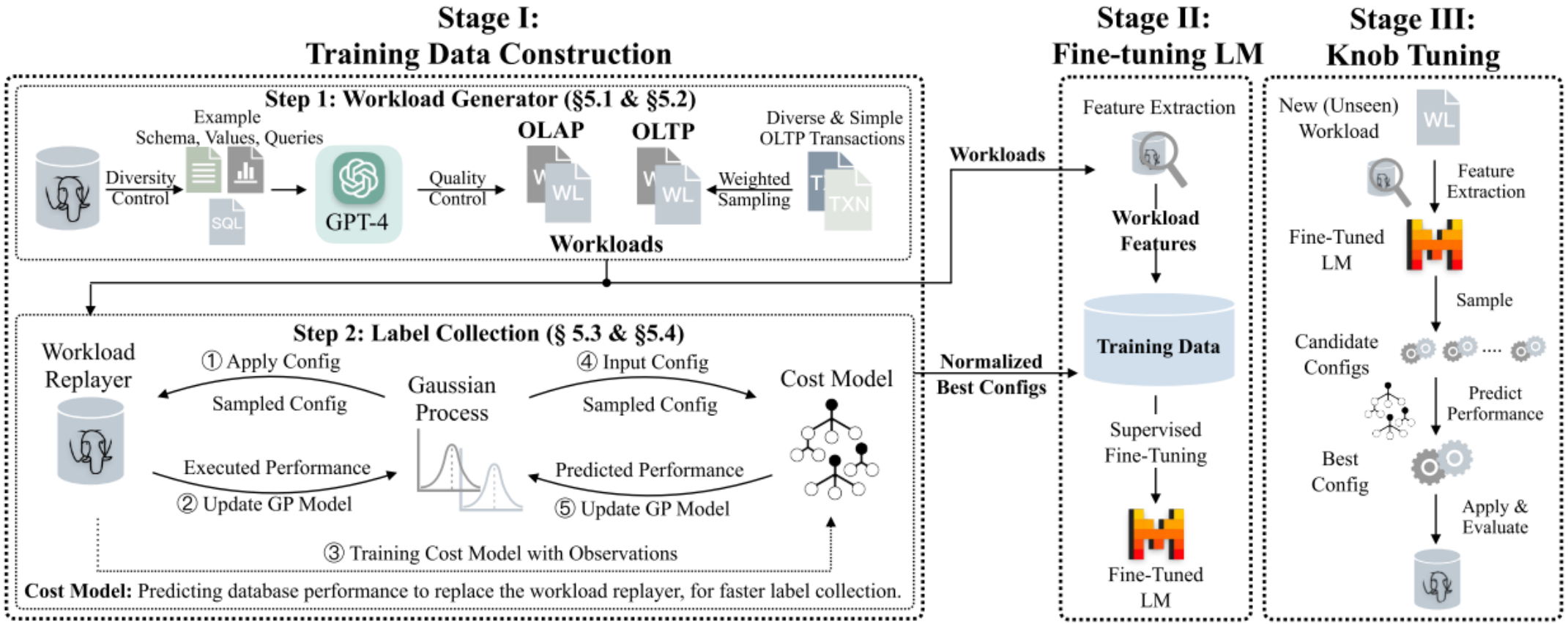
本文提出了一种新颖的数据库Knob调优方法E2ETune，通过使用历史数据将Knob调优过程建模为端到端的序列生成任务，避免配置过程中的迭代

挑战：缺乏高质量的包含<工作负载，有前途的配置>对的数据集，如何训练模型

需要设计一个新的训练数据生成流程

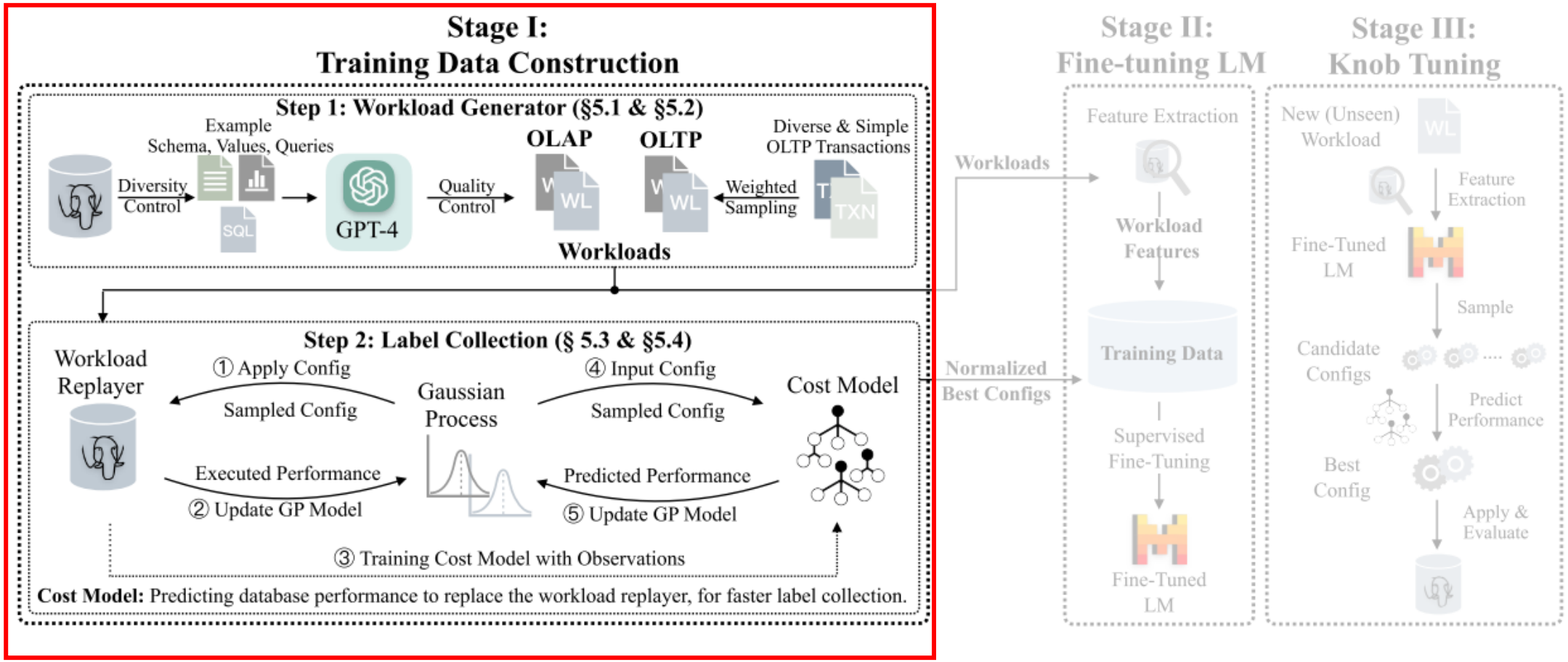
Previous knob tuning methods *vs.* E2ETune

系统设计



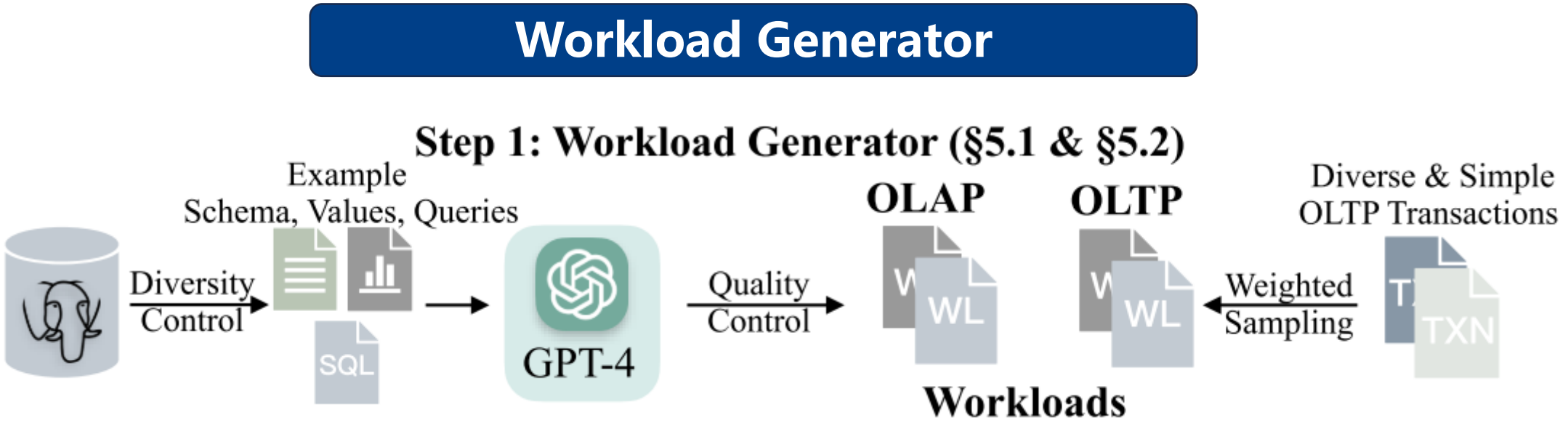
Overview of E2Etune

系统设计



Overview of E2Etune

Data Construction——Workload Generator



- **数据库系统中的工作负载被分类为两类：OLAP和OLTP**
  - OLAP工作负载侧重于复杂的分析查询，其性能主要是通过延迟（执行查询所花费的时间）来衡量的。
  - OLTP工作负载负责实时事务处理，其性能通过吞吐量来评估。

Data Construction——Workload Generator

OLAP负载生成

- **OLAP工作负载通常是固定的查询模板，其中包含有限数量的可变值参数**
- **直接根据现有模板改变值来生成OLAP工作负载的方法会导致多样性不足**
- **作者利用GPT-4-Turbo生成真实且复杂的SQL查询，并在拥有足够数量的查询后，选择这些查询的子集来获得新的OLAP负载**

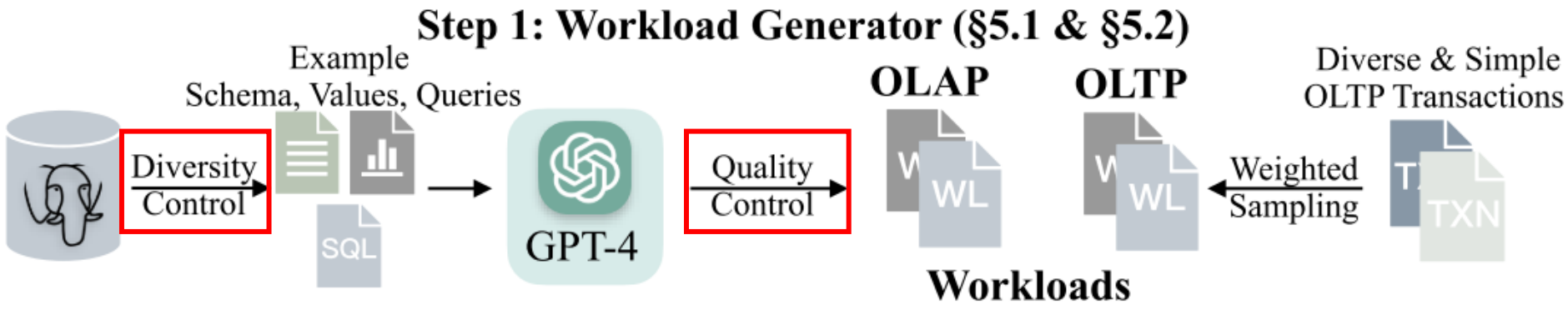
为了确保生成工作负载的质量，采用了两种控制机制：

- 多样性控制

  输入给大模型的prompt中会随机组合数据库的参考信息，以保证组合的多样性

- 质量控制

  使用数据库的EXPLAIN指令来评估生成的查询的语法准确性，对于出错的进行修复。执行时间过长的查询会被直接清除



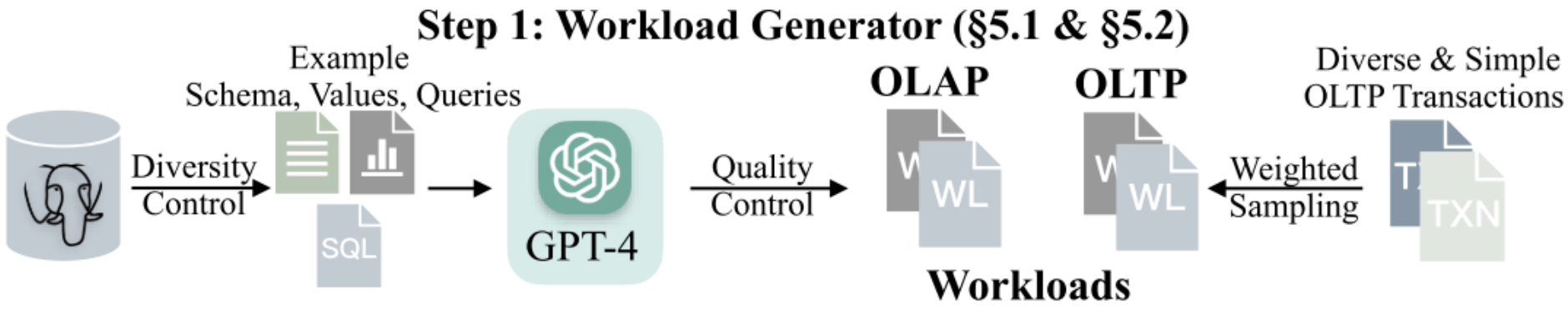
Data Construction——Workload Generator

OLAP负载生成

- OLAP工作负载通常是包含有限数量的具有可变值的查询模板
- 直接根据现有模板改变值来生成OLAP工作负载的方法可能会导致多样性不足
- 作者利用GPT-4-Turbo生成真实且复杂的OLAP负载
- 通过采用多样性控制和质量控制两个机制保证生成的多样性与质量

OLTP负载生成

- OLTP工作负载通常由一组预定义的事务组成
- 通过观察得知，调整单个事务的权重能够获得不同的工作负载
- 作者通过改变事务的权重来生成大量新的OLTP工作负载



Data Construction——Label Collection

Label Collection

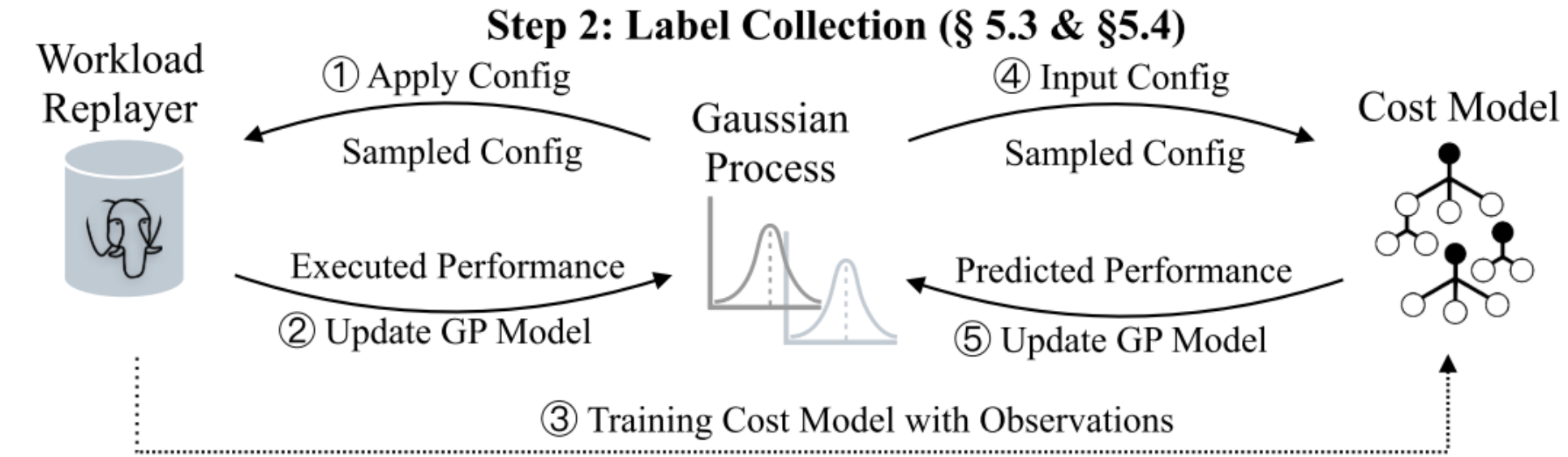
使用一种贝叶斯优化方法的变体HEBO作为数据的标签来源，也就是在数据构建阶段为不同的工作负载执行HEBO来探索所需的配置。

HEBO在寻找最佳配置的方面具有良好的性能，在绝大多数基准测试上都实现了比其他基线方法更好的性能。

HEBO需要进行多次迭代来实现，为一个工作负载生成一个配置常常需要几个小时，并且一个设备在同一时间只能为一个工作负载进行调优。

Data Construction——Label Collection

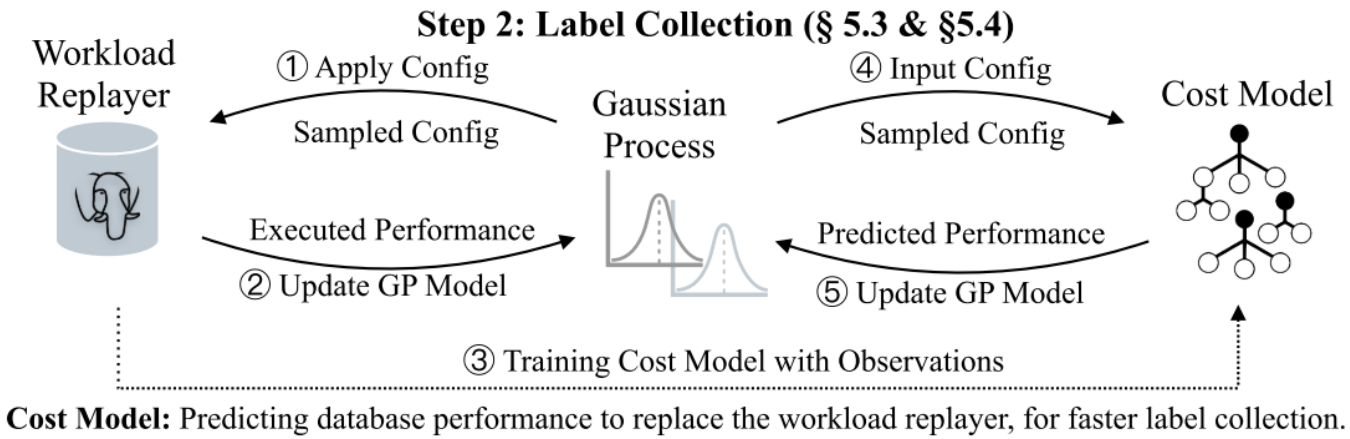
为解决HEBO方法的效率问题，E2Etune引入了Cost Model来加速HEBO的搜索过程，并且能够实现对多个工作负载并发调优。



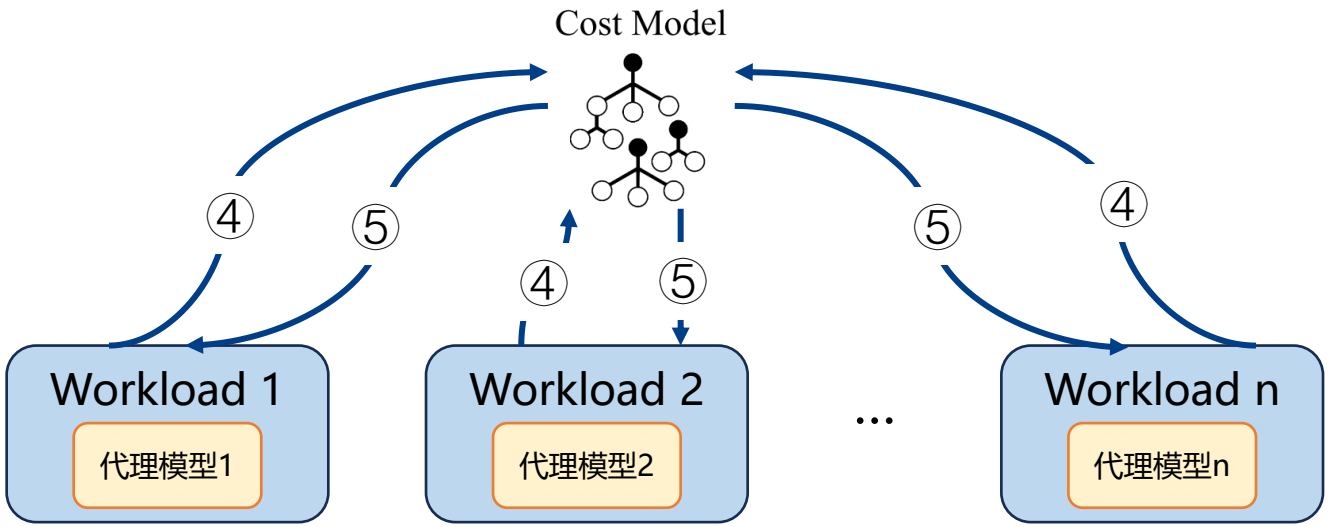
Cost Model: Predicting database performance to replace the workload replayer, for faster label collection.

采用适合成本估计任务的回归模型梯度增强回归（GBR）和随机森林回归（RFR）作为Cost Model，最终的评估结果是聚合两种模型的生成得到的。

Data Construction——Label Collection

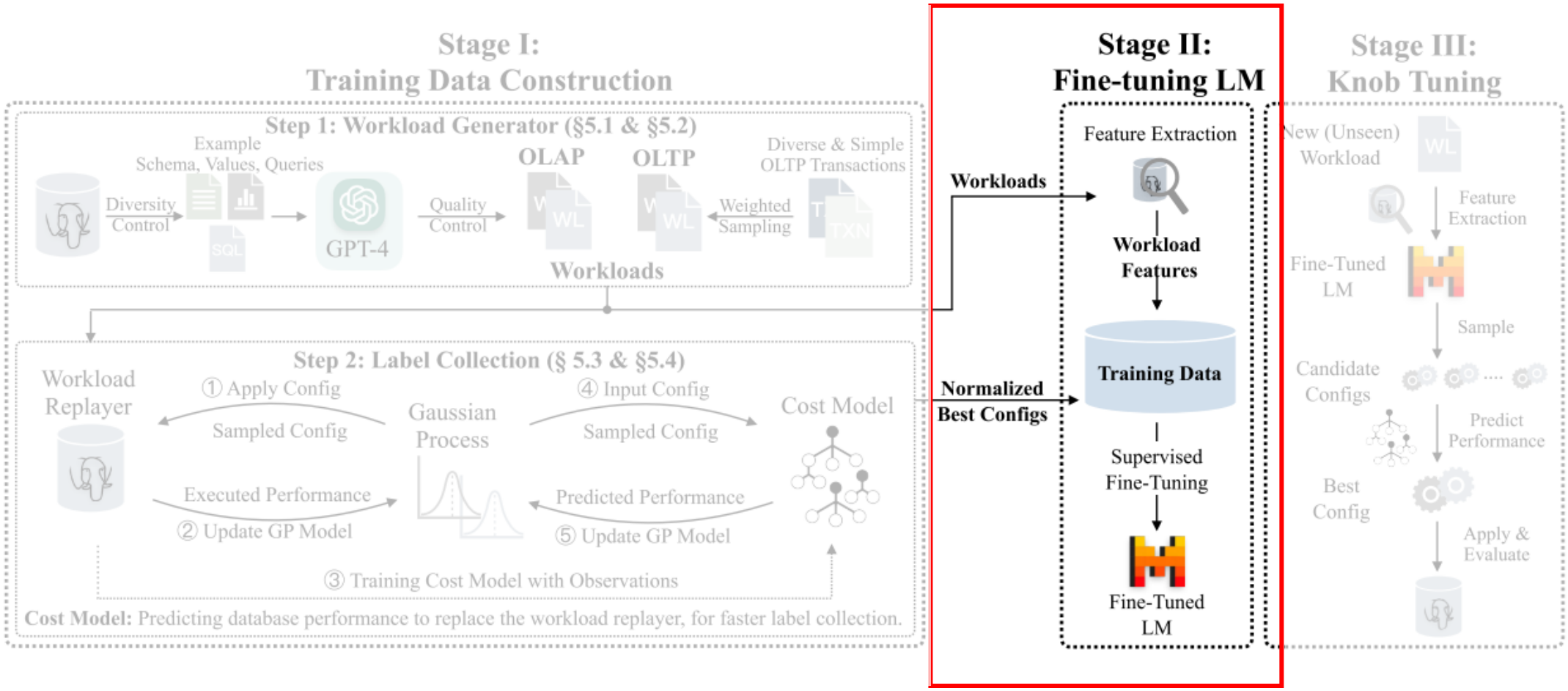


Cost Model: Predicting database performance to replace the workload replayer, for faster label collection.



- **执行过程**
  - **整个标签收集过程可以分为两个循环：一个循环是Replayer和代理模型的循环另一个是Cost Model和代理模型的循环**
  - **Replayer和代理模型的循环用于获取真实的性能指标以指导Cost Model的训练**
  - **Cost Model和代理模型的循环则是使用Cost Model替代了配置重放过程，加速评估配置并更新模型并且可以并行执行**
- **加速的实现**
  - **工作负载的特征中Knobs的值会按取值范围进行最大-最小值归一化处理，再输入给Cost Model训练。**
  - **因此训练出的一个共享的Cost Model可以对所有的工作负载实现性能评估，从而实现并行执行，加速数据的生成过程。**

Fine-Tuning LM



Fine-Tuning LM

- **输入序列**
  - 输入序列主要包含工作负载的特征，需要使模型能够理解和解释工作负载
  - 作者研究了工作负载三个维度的特征：
    1. 工作负载统计（工作负载级别的信息）
    2. 查询计划（查询级别的信息）
    3. 内部度量（系统级别的信息）

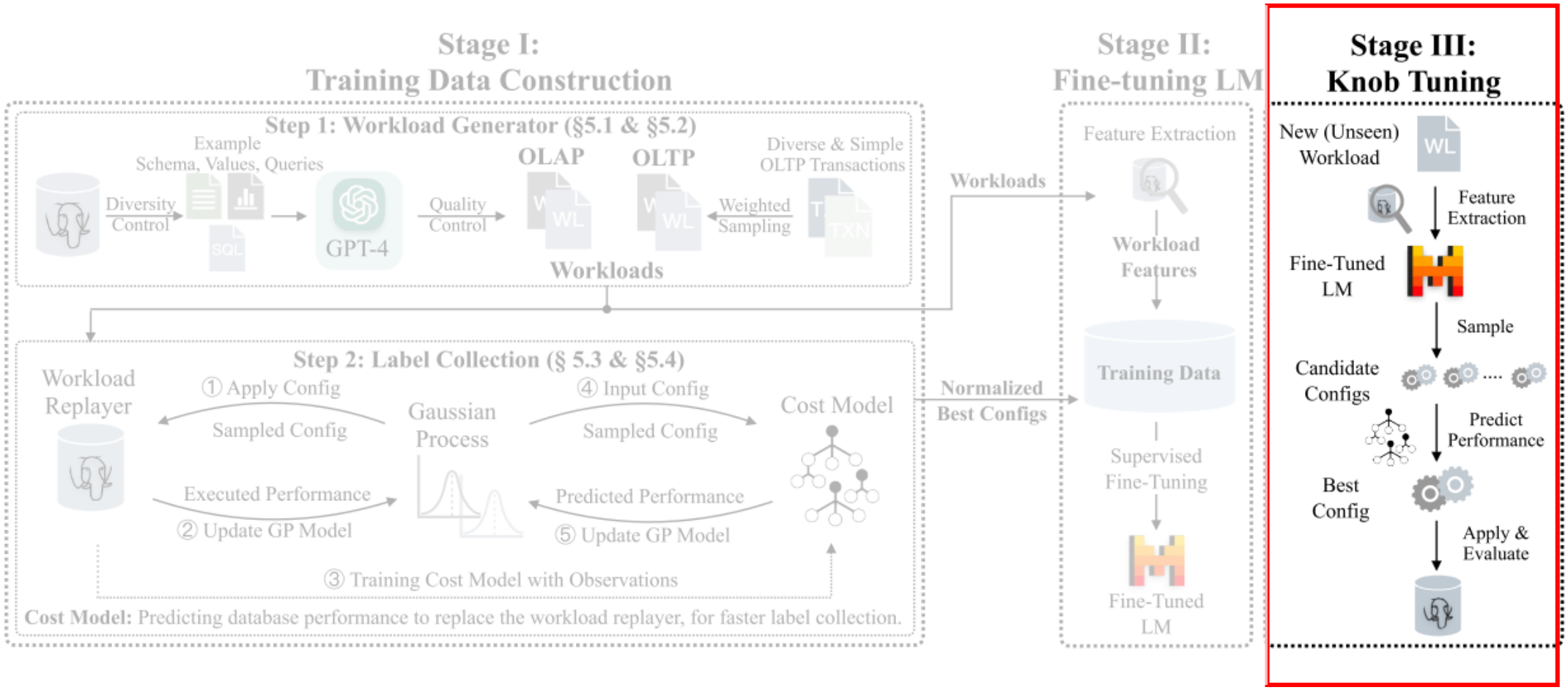
工作负载统计	查询计划	内部度量
• 每个表的访问频率 • SQL语句的总数 • 读写比率 • 每个SQL查询的平均谓词数 • ORDER BY、GROUP BY和聚合函数等关键操作符的比例	• 所有SQL语句相关的查询计划	14个经过仔细筛选能够指示配置运行状态并指导Knob调整的指标

Fine-Tuning LM

- **输入序列**
  - 输入序列主要包含工作负载的特征，需要使模型能够理解和解释工作负载
  - 作者研究了工作负载三个维度的特征：
    1. 工作负载统计（工作负载级别的信息）
    2. 查询计划（查询级别的信息）
    3. 内部度量（系统级别的信息）

- **输出序列**
  - 由于不同knob之间的数值尺度不同，直接生成knob的数值是有困难的。
  - 因此，E2Etune首先对值进行了规范化，然后将标准化后的值离散成buckets，模型的生成变为“knob的值应该在哪个buckets中”这样一个分类问题，这使得语言模型能够更容易进行学习。

Knob Tuning

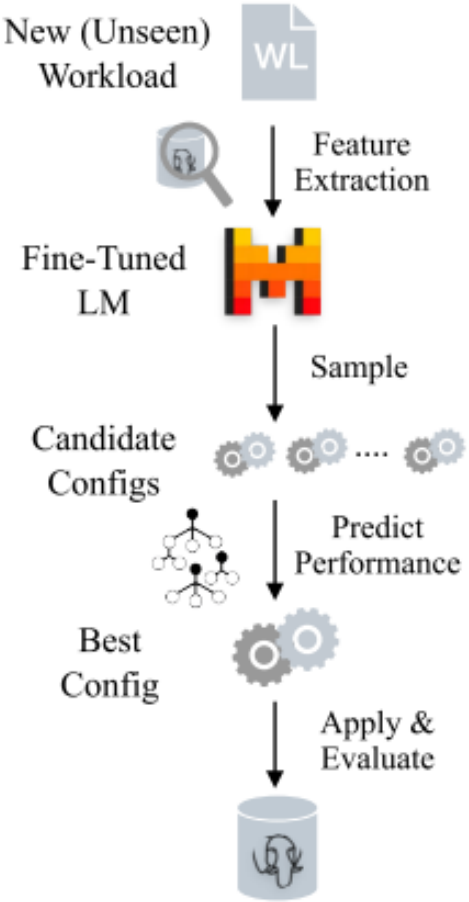

Stage I:
Training Data Construction
Step 1: Workload Generator (§5.1 & §5.2)
Example
Schema, Values, Queries
Diversity
Control
GPT-4
Quality
Control
OLAP
OLTP
Weighted
Sampling
Diverse & Simple
OLTP Transactions
Workloads
Step 2: Label Collection (§ 5.3 & §5.4)
Workload
Replayer
① Apply Config
Sampled Config
Gaussian
Process
④ Input Config
Sampled Config
Cost Model
Executed Performance
② Update GP Model
Predicted Performance
⑤ Update GP Model
③ Training Cost Model with Observations
Cost Model: Predicting database performance to replace the workload replayer, for faster label collection.
Workloads
Normalized
Best Configs
Stage II:
Fine-tuning LM
Feature Extraction
Workload
Features
Training Data
Supervised
Fine-Tuning
Fine-Tuned
LM
Stage III:
Knob Tuning
New (Unseen)
Workload
Feature
Extraction
Fine-Tuned
LM
Sample
Candidate
Configs
Predict
Performance
Best
Config
Apply &
Evaluate

Knob Tuning

- **在得到经过微调的LM后，E2Etune遵循“sampling-then-ranking”的推理策略进行Knob调优。**

  - “sampling”是指在推理阶段对LM的输出分布进行采样，以获得多个候选配置。以此大大提高生成配置的多样性和创新性，使LM能够探索更广泛的潜在解决方案并确定最佳配置。

  - “ranking”是在生成多个候选配置之后，对它们进行排序以确定最佳选择。 这个步骤是利用前面介绍过的Cost Model来评估和排序采样配置，最终选择最佳配置作为最终推荐配置。



目录

实验设置

- **数据集**
  - 使用了10广泛采用的基准：
    - 5个OLAP基准（TPC-H、JOB、SSB、SSB-flat、TPC-DS）
    - 5个OLTP基准（TPC-C、Smallbank、YCSB、Twitter、Wikipedia）

  每个基准测试包括一个数据库实例和用于SQL查询（OLAP）或事务（OLTP）的预定义模板。

  基于数据生成框架，总共有2953个LM的训练样本。实际执行调优以训练Cost Model的工作负载有130个，剩余2,823个样本的标签是通过Cost Model生成的。

  - 同时还考虑了来自现实场景的三个基准：StackOverflow 、SSAG 和AMPS

- **指标**
  - OLAP负载比较时延
  - OLTP负载比较吞吐量

$$\Delta = \frac{\text{default latency} - \text{optimized latency}}{\text{default latency}}$$

$$\Delta = \frac{\text{optimized tps} - \text{default tps}}{\text{default tps}}$$

- **基线**
  - 传统方法：SMAC和HEBO
  - 基于RL的方法：CDBTune
  - Cost Model代替负载replay：SMAC + Cost Model，HEBO + Cost Model和CDBTune + Cost Model
  - Workload Mapping：SMAC和HEBO
  - Model Ensemble： SMAC和HEBO
  - Model Pre-training：CDBTune（有或没有在线调优）
  - 随机抽样+Cost Model
  - Knob Pruning：OpAdviser、DB-BERT、GPTuner

- **实现细节**
  - PostgreSQL 12.2
  - Base Model：Mistral7B-Instruct-v0.2
  - PyTorch 2.1和Hugging face Transformer进行微调

- **环境**
  - Intel Xeon E5 2650 v4、1个 Intel Xeon Gold 5218 CPU, 256GB RAM和4个 NVIDIA GeForce RTX 3090 GPUs

主要结果

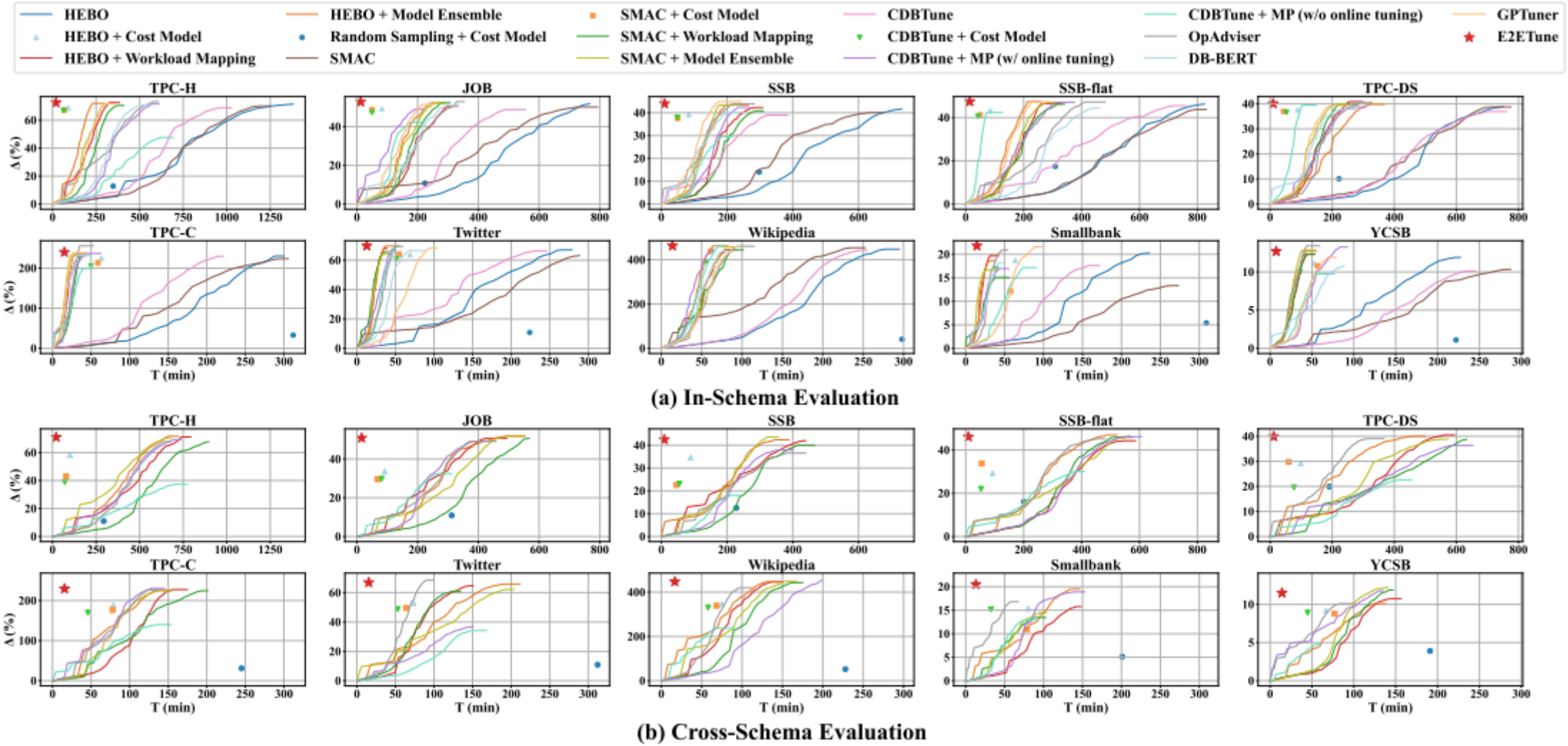


Figure 3: Best performance improvement over tuning time across 10 representative benchmarks. "MP" is the abbreviation of Model Pre-training. (top-left is better)

主要结果

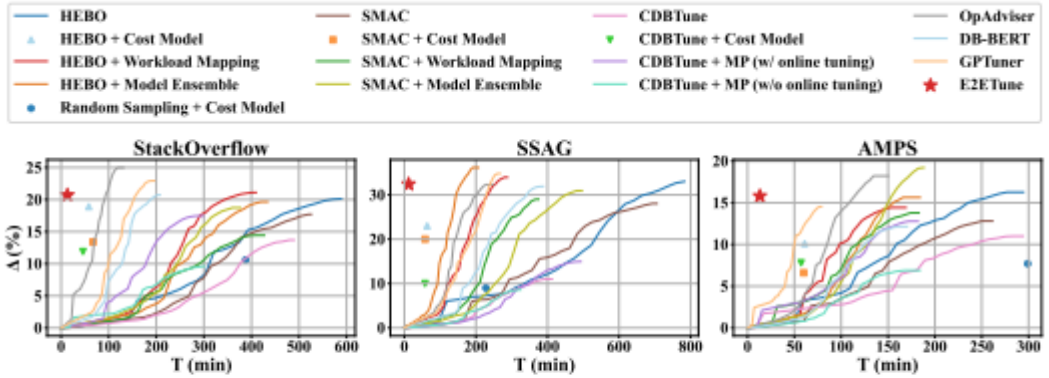


Figure 4: Maximum performance improvement over tuning time on three real-world benchmarks. (top-left is better)

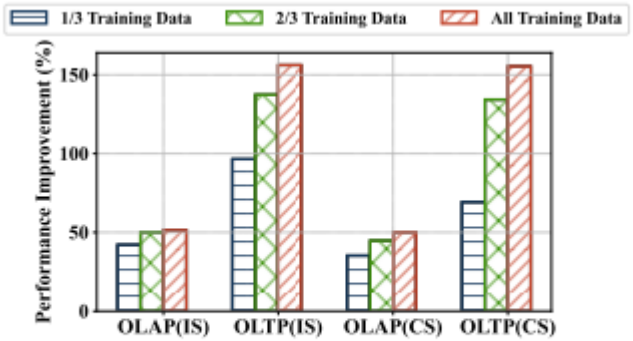


Figure 5: Ablation study of the scale of training data.

Table 2: Ablation studies of E2ETune. We present the average performance improvements Δ (%) ↑ on OLAP and OLTP benchmarks. "IS" and "CS" represent "in-schema" and "cross-schema" settings, respectively.

	OLAP(IS)	OLTP(IS)	OLAP(CS)	OLTP(CS)
E2ETune	51.4	156.2	50.1	150.6
Input Features				
- w/o internal metrics	49.8	152.5	48.6	142.8
- w/o workload features	47.0	148.9	48.5	148.7
- w/o query plans	44.5	145.6	34.9	53.2
LM Inference Strategy				
- w/o sampling-then-ranking	50.1	152.8	47.0	144.3
Knob Output Format				
- specific values	15.1	45.3	9.4	36.8
LM Backbone				
- CodeLLaMA-7B [59]	51.3	152.7	50.5	152.8
- LLaMA2-7B [68]	50.4	152.4	50.1	153.7
- DeepSeekCoder-7B [23]	48.8	159.6	48.8	135.3
- DeepSeekCoder-1B [23]	42.0	142.4	41.0	116.1
- Mistral-7B w/o Pre-training	NA	NA	NA	NA
LM Learning Strategy				
- Few-shot Mistral-7B	33.8	114.2	28.6	109.5
- Few-shot GPT-4	35.3	115.4	30.9	116.4

主要结果

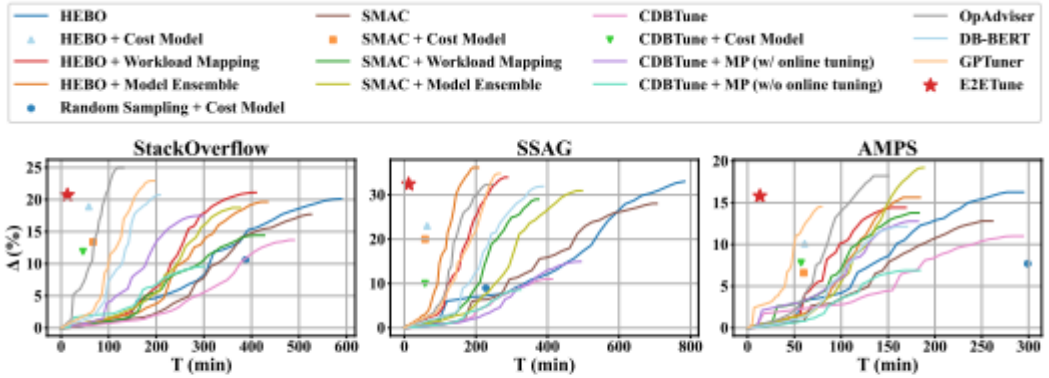


Figure 4: Maximum performance improvement over tuning time on three real-world benchmarks. (top-left is better)

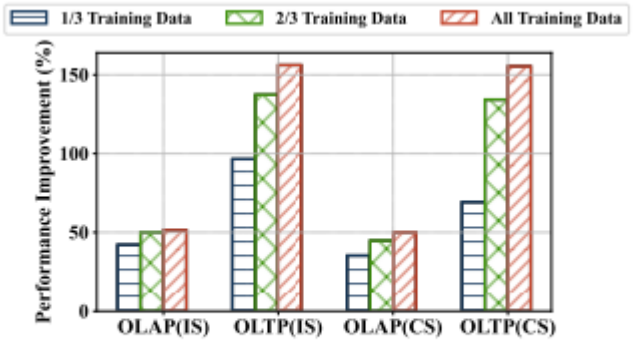


Figure 5: Ablation study of the scale of training data.

Table 2: Ablation studies of E2ETune. We present the average performance improvements Δ (%) ↑ on OLAP and OLTP benchmarks. "IS" and "CS" represent "in-schema" and "cross-schema" settings, respectively.

	OLAP(IS)	OLTP(IS)	OLAP(CS)	OLTP(CS)
E2ETune	51.4	156.2	50.1	150.6
Input Features				
- w/o internal metrics	49.8	152.5	48.6	142.8
- w/o workload features	47.0	148.9	48.5	148.7
- w/o query plans	44.5	145.6	34.9	53.2
LM Inference Strategy				
- w/o sampling-then-ranking	50.1	152.8	47.0	144.3
Knob Output Format				
- specific values	15.1	45.3	9.4	36.8
LM Backbone				
- CodeLLaMA-7B [59]	51.3	152.7	50.5	152.8
- LLaMA2-7B [68]	50.4	152.4	50.1	153.7
- DeepSeekCoder-7B [23]	48.8	159.6	48.8	135.3
- DeepSeekCoder-1B [23]	42.0	142.4	41.0	116.1
- Mistral-7B w/o Pre-training	NA	NA	NA	NA
LM Learning Strategy				
- Few-shot Mistral-7B	33.8	114.2	28.6	109.5
- Few-shot GPT-4	35.3	115.4	30.9	116.4

主要结果

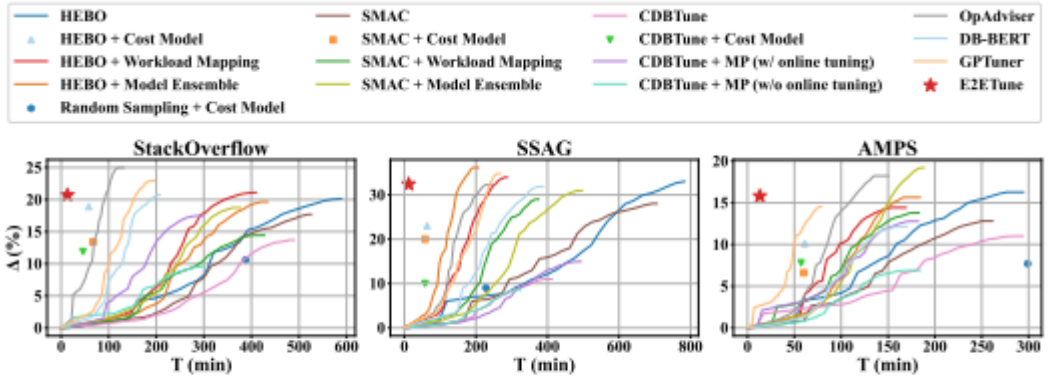


Figure 4: Maximum performance improvement over tuning time on three real-world benchmarks. (top-left is better)

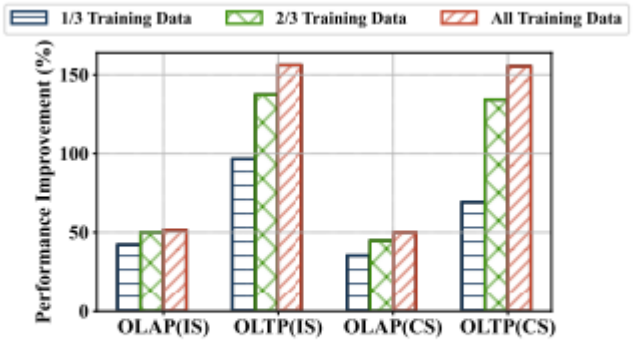


Figure 5: Ablation study of the scale of training data.

Table 2: Ablation studies of E2ETune. We present the average performance improvements Δ (%) ↑ on OLAP and OLTP benchmarks. "IS" and "CS" represent "in-schema" and "cross-schema" settings, respectively.

	OLAP(IS)	OLTP(IS)	OLAP(CS)	OLTP(CS)
E2ETune	51.4	156.2	50.1	150.6
Input Features				
- w/o internal metrics	49.8	152.5	48.6	142.8
- w/o workload features	47.0	148.9	48.5	148.7
- w/o query plans	44.5	145.6	34.9	53.2
LM Inference Strategy				
- w/o sampling-then-ranking	50.1	152.8	47.0	144.3
Knob Output Format				
- specific values	15.1	45.3	9.4	36.8
LM Backbone				
- CodeLLaMA-7B [59]	51.3	152.7	50.5	152.8
- LLaMA2-7B [68]	50.4	152.4	50.1	153.7
- DeepSeekCoder-7B [23]	48.8	159.6	48.8	135.3
- DeepSeekCoder-1B [23]	42.0	142.4	41.0	116.1
- Mistral-7B w/o Pre-training	NA	NA	NA	NA
LM Learning Strategy				
- Few-shot Mistral-7B	33.8	114.2	28.6	109.5
- Few-shot GPT-4	35.3	115.4	30.9	116.4

目录

总结与讨论

- **总结**

  E2ETune 提出了一种新颖的数据库参数调优方法，它通过微调生成式语言模型准确捕捉工作负载与其有希望的配置之间的复杂映射关系，实现了一种端到端的数据库调优框架。为了训练这个模型，作者设计了一个自动化数据生成框架，实现快速高效生成工作负载及其有希望的配置。实验表明，E2ETune显著提高了相对于现有方法的调优效率，而且在in-schema和cross-schema设置下均展示了显著的数据库性能提升。

- **基于本文**
  - 当前的数据生成与大模型微调过程都是完全离线的，未来可以设计一种信息交互机制，收集在线执行过程中的数据，指导离线进一步训练，以不断提升性能，增强泛化性。
  - 文章将工作负载简单分成了OLTP和OLAP两种类型，并且分别对应延迟需求和吞吐量需求两种单一优化目标。在后续的工作中是否可以更实现进一步的细致分类、判断不同工作负载对应的性能要求，以及一些针对一些可能存在的多目标优化需求？

总结与讨论

- **泛化性**
  - E2ETune的一个核心贡献就在于学习了工作负载与可用配置之间的深层映射关系，因此在面对新的数据库模式和工作负载时，也能做到类似于知识迁移的效果，给出合适的配置。这种希望学习深度映射关系以适应新的情况的想法是值得借鉴的。

- **用到我们自己的工作上?**
  - 这篇文章训练数据生成的框架的设计思路是可以参考的，因为我目前的工作也是缺少一个高质量的有<任务，有希望的配置>这样成对数据的数据集。借鉴这文章的方法，可以尝试把他生成配置的迭代过程换成资源调度方案生成的过程，以此获得微调大模型的数据集，感觉是可以实现的。
  - 另一个让我觉得可以借鉴的点在于E2Etune对于将微调后的语言模型要完成的任务从数值生成任务转换成分类任务的设计，因为经过前段时间的尝试有发现，完整的资源调度问题对大模型来说还是太复杂了，可能将任务进行一些拆解和转化，比如从具体的调度策略生成转化为一些问题建模上的指导或是分类判断可能容易让大模型学习。

请老师同学们批评指正!