数据库之智能调参

汇报人：周帆

DSC
分布式存储与计算实验室

2022.08.26

目录

01 智能调参背景概述

01 智能调参背景概述

分布式存储与计算实验室

1.1 背景概述

现状：

1．数据库参数配置选项多

2．DBA根据经验调参

过程：

1．监控其工作负载是否存在任何性能故障或减速

2．将数据库复制到另一台机器

3．根据调优指南或经验手动调参

问题：

1．繁琐、昂贵且效果未必最佳

01 智能调参背景概述

影响：

1. 若缓存太小，导致页面频繁换入换出，性能降低；缓存太大，资源浪费
2. 若刷盘频率太高，导致大量磁盘I/O操作，影响性能；频率太低，占用大量内存，recovery时间长

方向：将数据库与AI结合，实现自动调参

01 智能调参背景概述

DSC
分布式存储与计算实验室

1.2 智能调参类型

- 基于规则的数据库调参

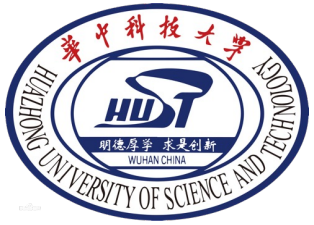
  BestConfig（启发式搜索）
- 基于学习的数据库调参

  OtterTune、CDBTune、Qtune、Hunter

（2017）基于大规模机器学习的数据库参数自动化管理系统 OtterTune

（2019）基于深度强化学习的端到端云数据库调参系统 CDBTune

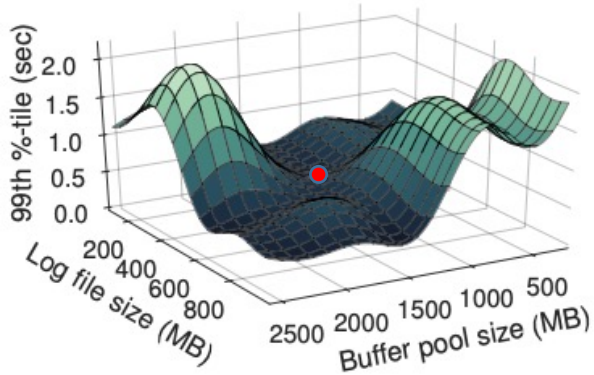
（2019）基于强化学习查询感知数据库调参系统 Qtune



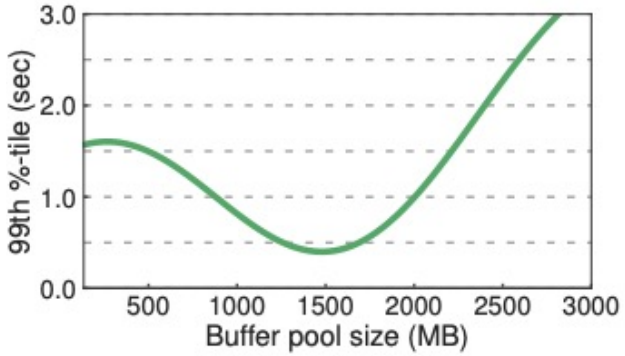
（2022）在线云数据库混合调优系统 Hunter

01 智能调参背景概述

1.3 调参挑战



(a) Dependencies



(b) Continuous Settings

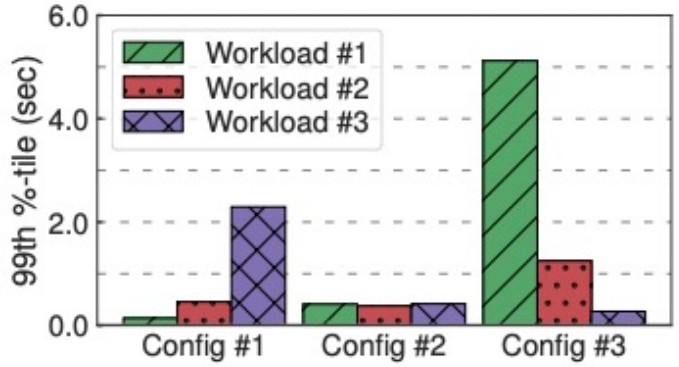
1．依赖性

每个knob不是独立存在，调整一个knob，可能会影响其他knob

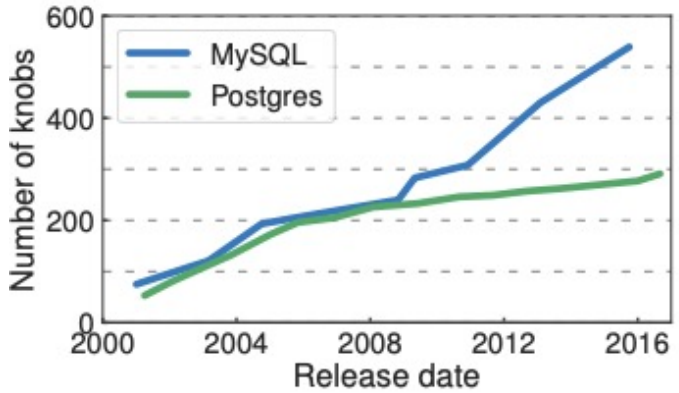
2．连续设置

参数调整范围是连续的，参数组合数量较大

智能调参背景概述



(c) Non-Reusable Configurations



(d) Tuning Complexity

3．不可重用配置

不同工作负载对应不同最优参数配置

4．调整复杂性

数据库参数成千上百，并逐年增加

02 / OtterTune

02

OtterTune

2.1 简要概述

核心思想

收集历史数据，并使用模型学习以往经验，实现自动化调优

主要步骤

I. 去掉冗余工作负载

II.选择最具影响力的旋钮

III.将目标工作负载映射到已有工作负载

IV.推荐旋钮设置

解决问题

在构建模型之前，尽可能减少构建参数的维度、个数等，以此降低模型的复杂性

OtterTune

2.2 系统架构

客户端（本地）

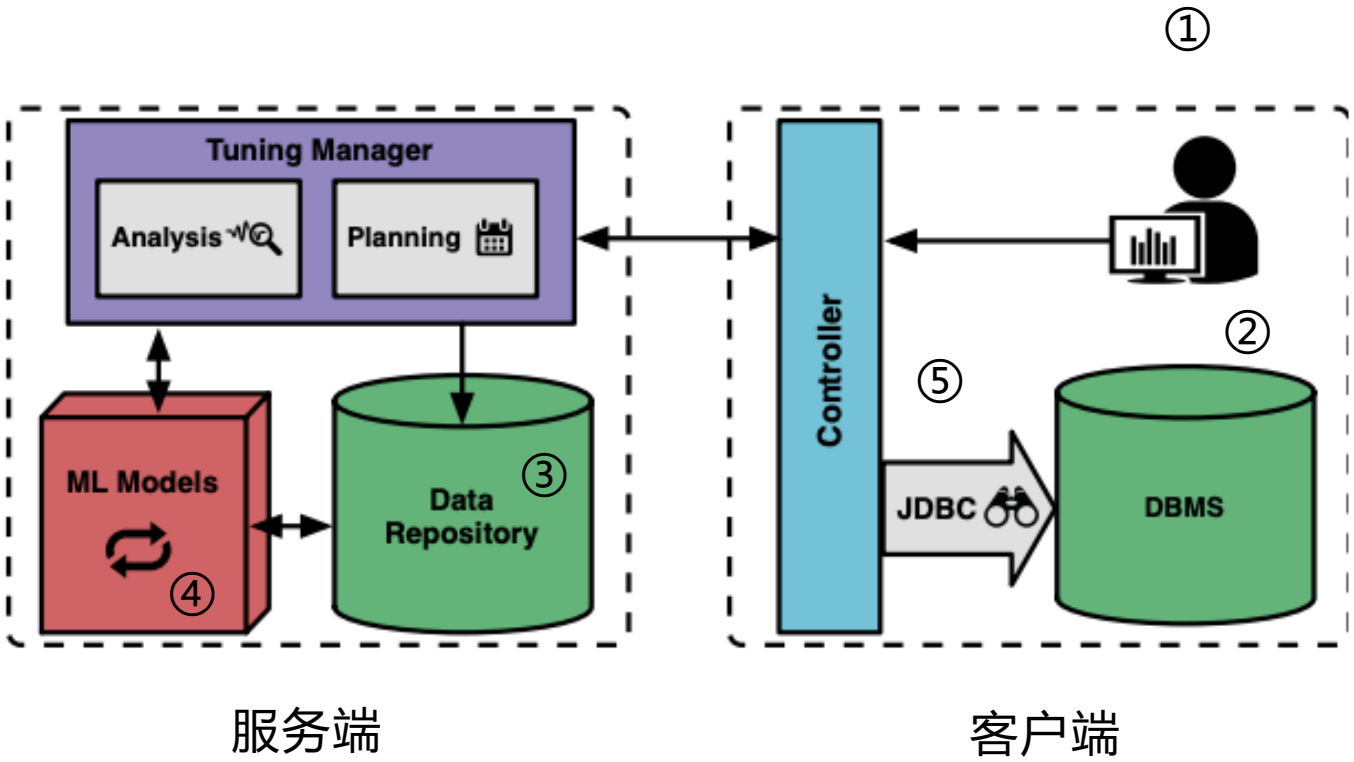
I. 收集DBMS统计信息，如工作负载，参数配置，性能

II. 给DSBMS安装新配置，并收集性能测试结果

服务端（云端）

Data Repository：存储客户端数据

ML Models：调优模型



OtterTune

2.3 调优过程

1. 确定需优化哪些外部指标，如延迟，吞吐量
2. 开始第一个观察期，对于OLTP工作负载，**以固定观察期为准**；对于OLAP工作负载，**可变长度观察期**
3. 观察期结束，收集内部指标（参数配置）、外部指标、工作负载
4. 将信息存储在存储库，并判断是否需要调参

   调参步骤主要分为两步：

   I. 将目标工作负载映射到已有工作负载

   II.按照工作负载分类，进行模型训练，最后推荐knob配置
5. 循环调参过程，直到达到调优效果，但为了降低调优时间，一般将调优次数设置为5

OtterTune

2.4 去掉冗余工作负载

工作负载收集方法：使用DBMS的内部运行时指标来描述工作负载

问题：

1. 相同指标对应不同单位的值
2. 指标之间完全是强相关的

方法：因子分析（FA）、k-means

表示方式：KV存储

M = （m1，m2...mk）

1.
2.

```
mysql> SHOW GLOBAL STATUS;
+---------------------------------------+-----------+
| METRIC_NAME                           | VALUE     |
+---------------------------------------+-----------+
| ABORTED_CLIENTS                       | 0         |
| ABORTED_CONNECTS                      | 0         |
...
| INNODB_BUFFER_POOL_BYTES_DATA         | 129499136 |
| INNODB_BUFFER_POOL_BYTES_DIRTY        | 76070912  |
| INNODB_BUFFER_POOL_PAGES_DATA         | 7904      |
| INNODB_BUFFER_POOL_PAGES_DIRTY        | 4643      |
| INNODB_BUFFER_POOL_PAGES_FLUSHED      | 25246     |
| INNODB_BUFFER_POOL_PAGES_FREE         | 0         |
| INNODB_BUFFER_POOL_PAGES_MISC         | 288       |
| INNODB_BUFFER_POOL_PAGES_TOTAL        | 8192      |
| INNODB_BUFFER_POOL_READS              | 15327     |
| INNODB_BUFFER_POOL_READ_AHEAD         | 0         |
| INNODB_BUFFER_POOL_READ_AHEAD_EVICT   | 0         |
| INNODB_BUFFER_POOL_READ_AHEAD_RND     | 0         |
| INNODB_BUFFER_POOL_READ_REQUESTS      | 2604302   |
| INNODB_BUFFER_POOL_WAIT_FREE          | 0         |
| INNODB_BUFFER_POOL_WRITE_REQUESTS     | 562763    |
| INNODB_DATA_FSYNCS                    | 2836      |
| INNODB_DATA_PENDING_FSYNCS            | 1         |
| INNODB_DATA_WRITES                    | 28026     |
...
| UPTIME                                | 5996      |
| UPTIME_SINCE_FLUSH_STATUS             | 5996      |
+---------------------------------------+-----------+
```

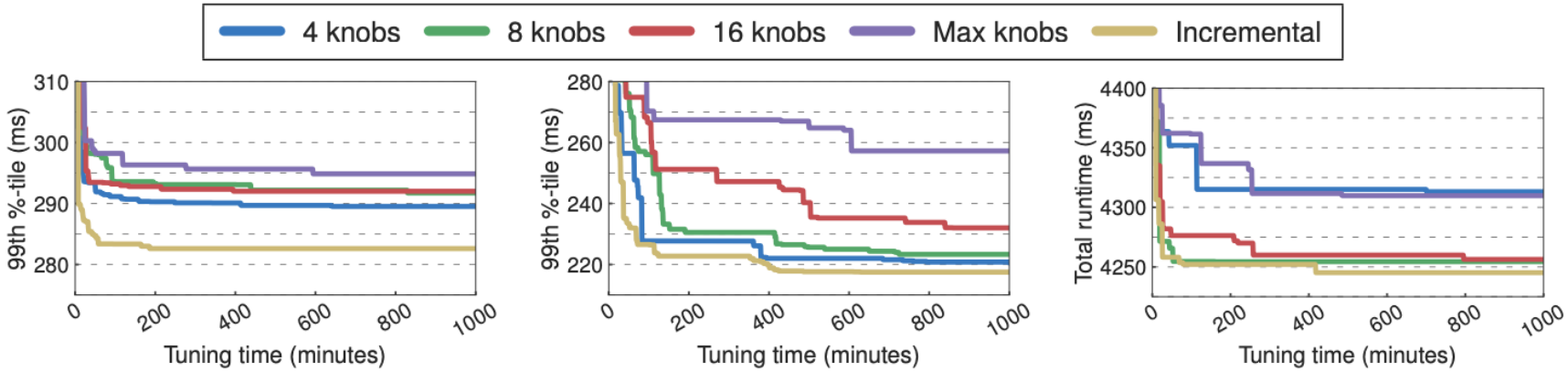
OtterTune

2.5 knobs重要性排序

问题：

1. knobs越多，调优过程越复杂

方法：Lasso回归模型对参数重要性进行排序，调优时，逐渐增加knob个数（每1小时，计数增加两个）



(a) MySQL (TPC-C) (b) Postgres (TPC-C) (c) Vector (TPC-H)

Figure 5: Number of Knobs – The performance of the DBMSs for TPC-C and TPC-H during the tuning session using different configurations generated by OtterTune that only configure a certain number of knobs.

OtterTune

2.6 调优模型

工作负载映射 ----- 寻找相似负载

方法一：两个工作负载的相似度可用其统计量M之间的距离来衡量

方法二：将统计量M也作为机器学习模型的向量一起训练。即模型的输入向量从参数配置演变为[参数配置、工作负载]

OtterTune

2.6 调优模型

构建模型

无数据时：拉丁超立方采样

横坐标：参数组合 X = [x1,x2 ... xk]

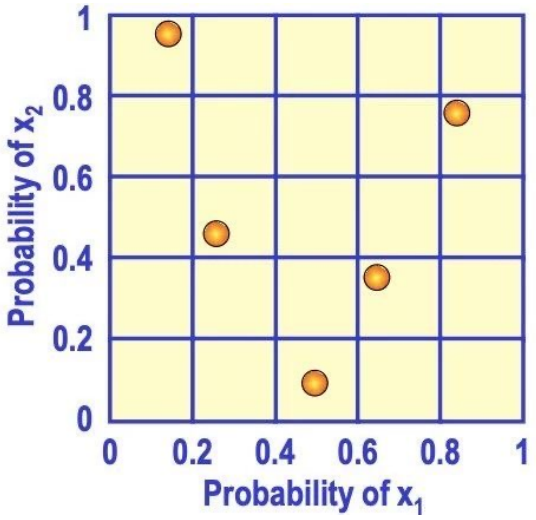
纵坐标：性能

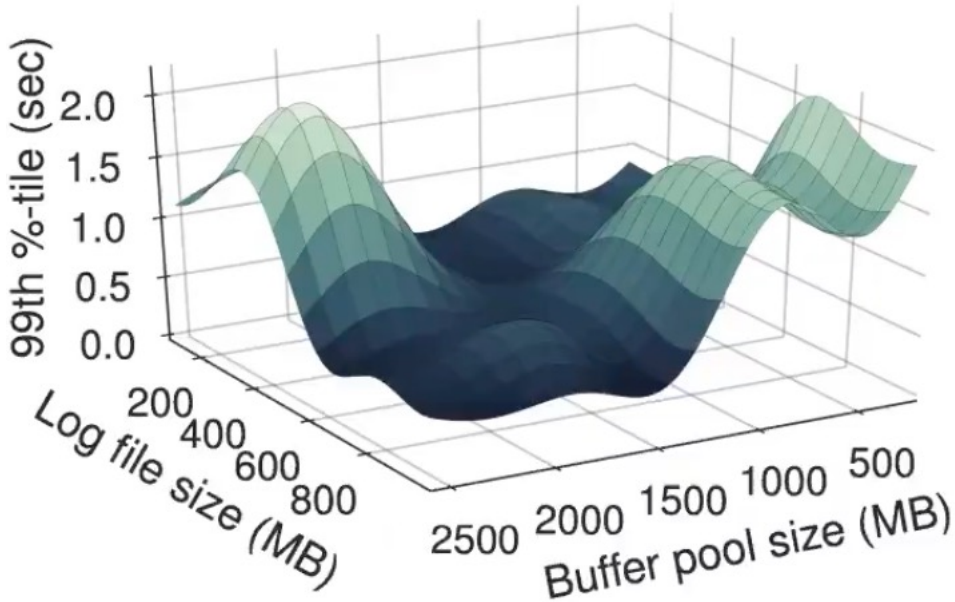
模型训练：得到近似函数f

输出：参数组合

训练近似函数f的模型可以是：

- 高斯过程回归





OtterTune

2.6 调优模型

如何找到一个优秀的参数配置X？

核心思想：数据多，则利用；数据少，则探索

- 高斯回归

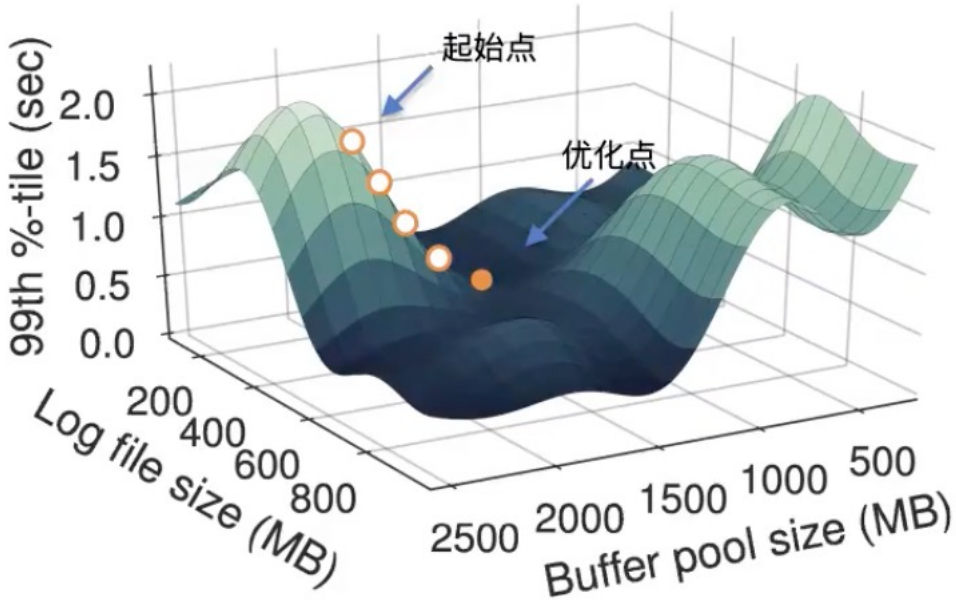
利用 ↑

$$U(x) = M(x) + k * S(x)$$

↓ 探索

- 梯度下降。

  从起始点开始，按函数梯度的方向不断尝试，直到找到优化点



2.7 总结

优点

1. 利用大量历史数据
2. 几乎不需要人工参与
3. 能够适应不同的数据库管理系统

缺点：

1. 所采用的训练模型都是流水线模型
2. 过度依赖大规模高质量训练样本
3. 无法在高维连续空间中推荐合理的配置，存在次优情况

03 / CDBTune

CDBTune

3.1 简要概述

1. 核心思想-放弃机器学习，引入深度强化学习，实现端到端自动调优系统
2. 优化点
   I. 采用试错法，以有限的样本数学习最佳旋钮
   II.利用深度确定性策略梯度法在高维连续空间中寻找最优配置
   III.设计了一个有效的奖励函数，代替传统回归

3.2 训练数据收集

1. 冷启动

初始时，利用标准的工作负载测试工具（如 Sysbench）来生成一组查询工作负载。并采用try-and-error策略来训练四元组，得到更多的训练数据

2. 增量培训

online训练，即在线强化模型

CDBTune

3.3 推荐模型

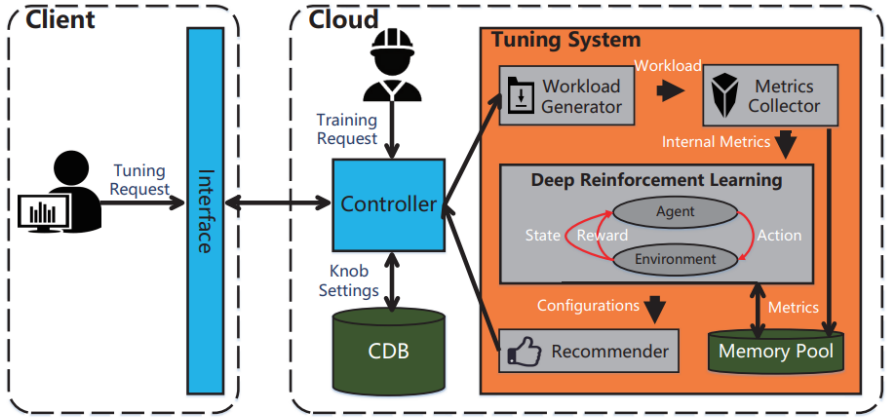


Figure 2: System Architecture.

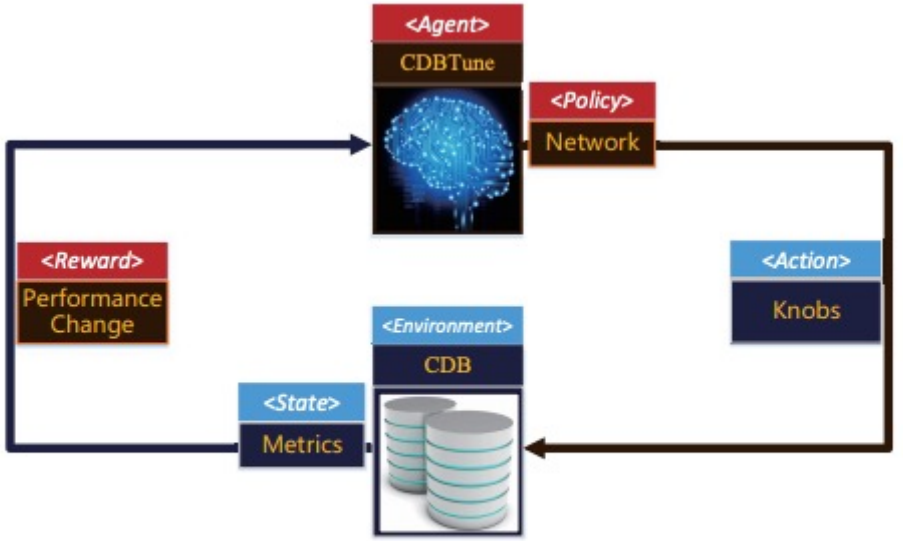


Figure 3: The correspondence between RL elements and CDB configuration tuning.

CDB实例：收集最近一段时间SQL语句集合，重新执行

CDBTune

3.4 Deep deterministic policy gradient (DDPG)

预测过程

仿真环境

a_t

$(s_t / r_t, s_{t+1})$

样本数据

Memory pool

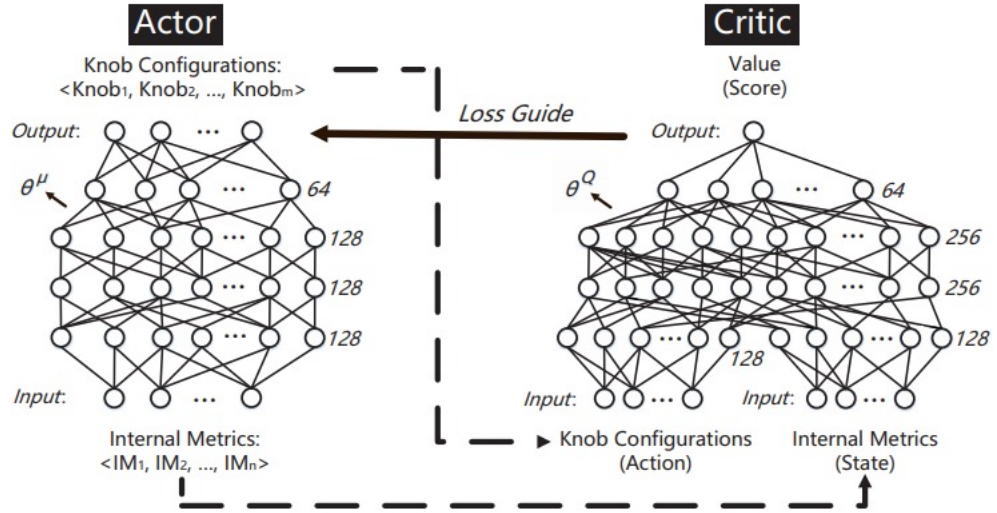


Figure 4: DDPG for CDBTune.

s_t, a_t, r_t, s_{t+1}

训练过程

CDBTune

3.5 Reward Function

T：吞吐量

L：延迟

Δ：判断正负相关

L_0:初始性能

r：奖励函数

C_T，C_L：系数，相加为1，默认都为0.5。

参数约大，权重约重

公式（7）：最终奖励

$$\Delta T = \begin{cases} \Delta T_{t\to 0} = \dfrac{T_t - T_0}{T_0} \\ \Delta T_{t\to t-1} = \dfrac{T_t - T_{t-1}}{T_{t-1}} \end{cases} \tag{4}$$

$$\Delta L = \begin{cases} \Delta L_{t\to 0} = \dfrac{-L_t + L_0}{L_0} \\ \Delta L_{t\to t-1} = \dfrac{-L_t + L_{t-1}}{L_{t-1}} \end{cases} \tag{5}$$

$$r = \begin{cases} ((1+\Delta_{t\to 0})^2 - 1)|1+\Delta_{t\to t-1}|, \Delta_{t\to 0} > 0 \\ -((1-\Delta_{t\to 0})^2 - 1)|1-\Delta_{t\to t-1}|, \Delta_{t\to 0} \leqslant 0 \end{cases} \tag{6}$$

$$r = C_T * r_T + C_L * r_L \tag{7}$$

CDBTune

3.6 性能对比

Table 2: Detailed online tuning steps and time of CDBTune and other tools.

Tuning Tools	Total Steps	Time of One Step (mins)	Total Time (mins)
CDBTune	5	5	25
OtterTune	5	11	55
BestConfig	50	5	250
DBA	1	516	516

相同负载下，CDBTune只需5步可以找到最优负载，并且总训练时长最短

3.7 总结

优点

1. 化繁为减，实现的是一个端到端的推荐
2. 自我学习，探索各种可能的调参动作
3. 减小陷入局部最优的可能性

缺点：

1. 冷启动，需要初始化生成负载进行训练，导致调优时间大大增加
2. 只支持粗粒度调优（只读，只写，读写），不支持细粒度的调优（select，update等）

04 /Hunter

04 Hunter

4.1 简要概述

问题

1. 训练初期，调优效果差，收敛速度慢（冷启动）
2. 搜索空间大，网络结构复杂（模型优化）

目标：在没有任何数据的情况下短时间内推荐最优配置（在线调优系统）

优化点

I. 克隆CDB实例，以此并行进行压力测试，同时，采用GA算法，以此生成高质量的样本

II.参照OtterTune，采用PCA降维，并利用随机森林给Knob排序

4.2 系统架构

GA算法推荐参数

降维与knob排序

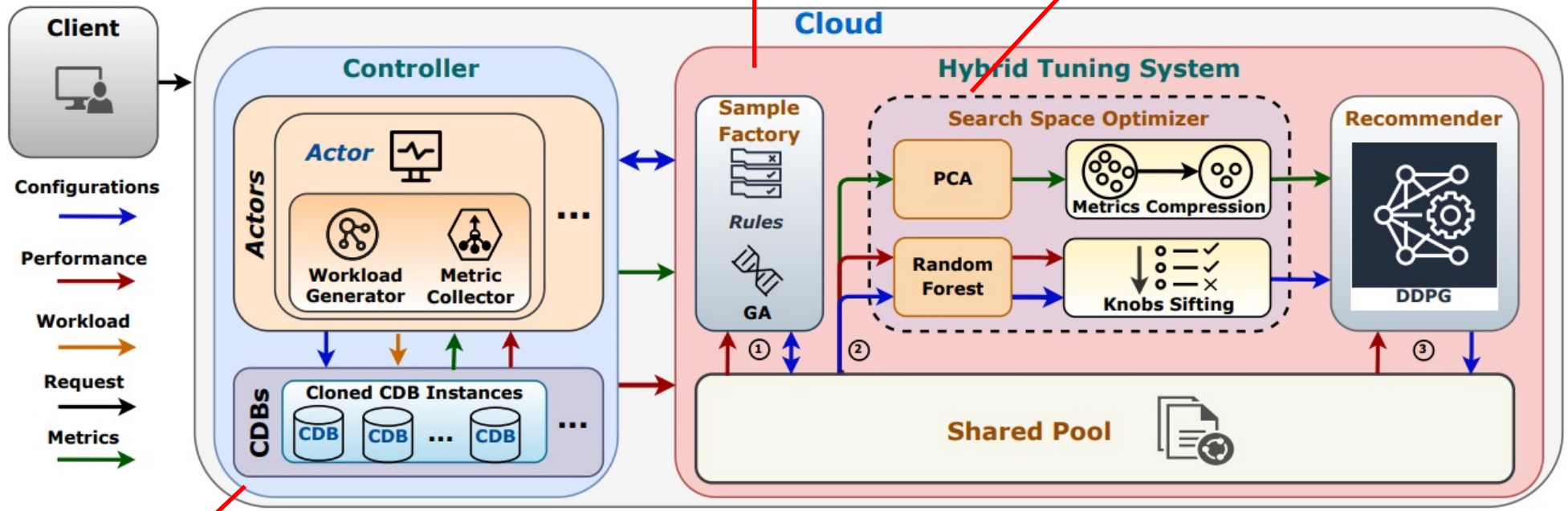


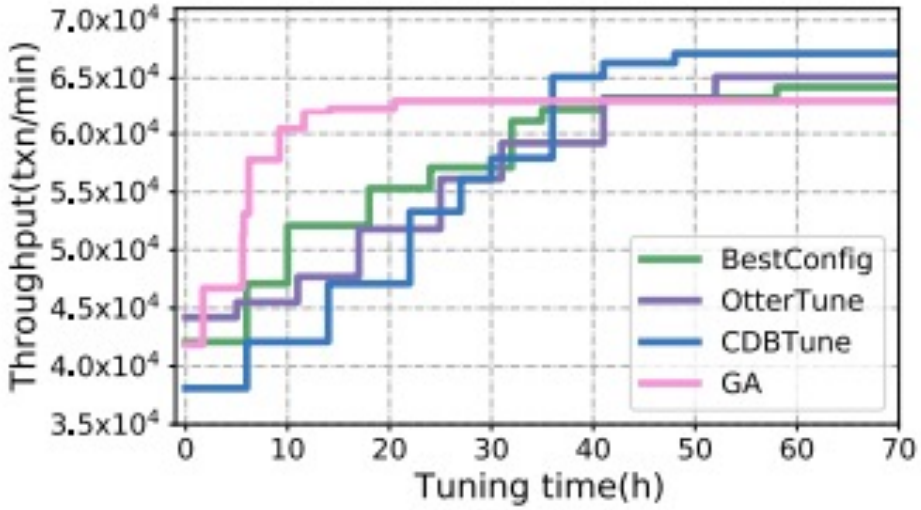
Figure 2: Overview Architecture and Workflow of HUNTER.

并行压力测试

4.3 GA算法

Algorithm 1: Genetic Algorithm

```
1  Execute Initialization to generate POP;
2  while Number of samples is not enough do
3      Execute Selection Strategy and save K_BEST into POP_i;
4      for k = 1 to n do
5          Execute Selection Strategy ;
6              Execute Fitness Function ;
7          Execute Crossover Strategy ;
8          Execute Mutation Strategy ;
9          Save new individuals into POP_j;
10     end
11     POP = POP_i + POP_j;
12 end
```



(a) MySQL(TPC-C)

- 利用高性能之间的相似性，通过交叉和变异快速学习
- 可以在初步阶段获得更优样本

4.4 降维与knob排序

Hunter与OtterTune使用方法对比

	PCA	FA
原理	基于方差最大化降维	依据变量间的依赖关系降维
优势	信息保留度>90%	降维效果更优
不足	降维效果不佳	时间过长

	随机森林	Lasso回归
原理	由多个决策树组成的投票算法	线性回归计算特征系数
优势	引入随机性，准确率高	可以去除无关变量，降低维度
不足	训练时间过长	不适用于共性变量

4.5 总结

优点

1. 利用并发压力测试，解决冷启动问题，显著缩短在线调优时间
2. GA算法短时间内生成高质量样本，提高DDPG调优效果

缺点：

1. 难以快速判断性能是否达到“最优”，目前需要额外的时间判断

下一步研究方向：研究估计最优性能的方法，从而减少额外的调优时间

05 / QTune

05

QTune

分布式存储与计算实验室

5.1 简要概述

问题

1. CDBTune只提供粗粒度的调优（读写级别调优），不能提供细粒度调优（OLTP、OLAP级别的调优）

优化点

I. QTune 提供三种数据库调优粒度：查询级、工作负载级和集群级调优

II.加入SQL查询信息，提出一种双状态深度确定性策略梯度（DS-DDPG）模型

QTune

5.2 系统架构

Query2Vector：为给定查询生成一个特征向量

Vector2Pattern：离散模型，学习配置的离散值（类似-1，0，1）

Pattern2Cluster：根据离散模式分类

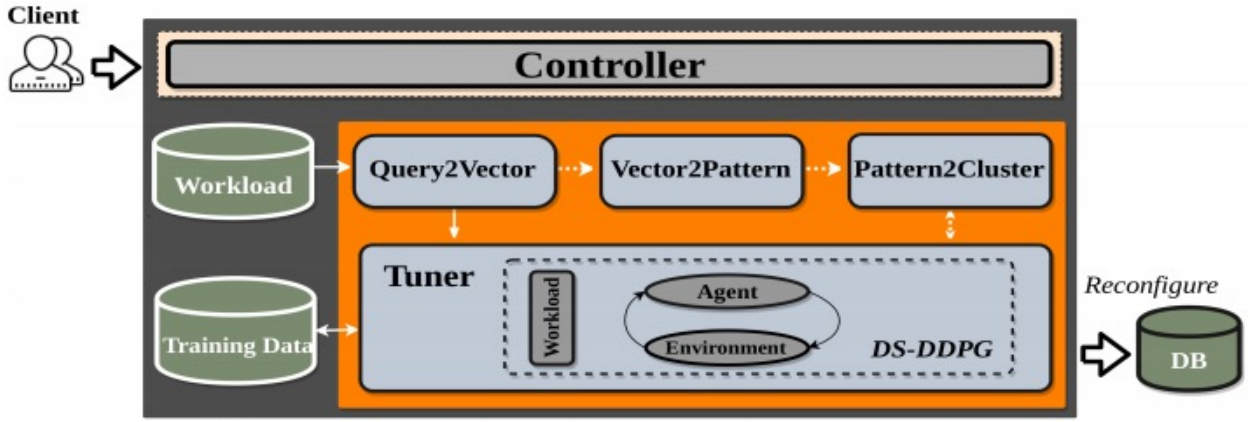
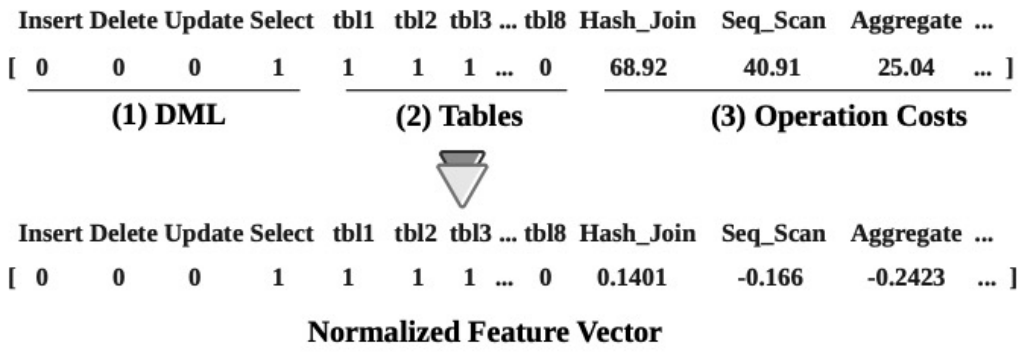


Figure 1: The QTune Architecture

5.4 查询向量化

1. 捕获查询信息，例如查询中涉及哪些表
2. 捕获查询成本，例如选择成本和连接成本

```
SELECT      MIN(tbl3.movie_id)
FROM        tbl1,  tbl2,  tbl3
WHERE       tbl1.info = '%act%'
  AND       tbl1.id = tbl3.type_id
  AND       tbl2.movie_id = tbl3.movie_id
```



5.5 总结

优点

1. 提供细粒度调优，提高调优性能

缺点：

1. 训练时间太长，模型复杂

06 总结

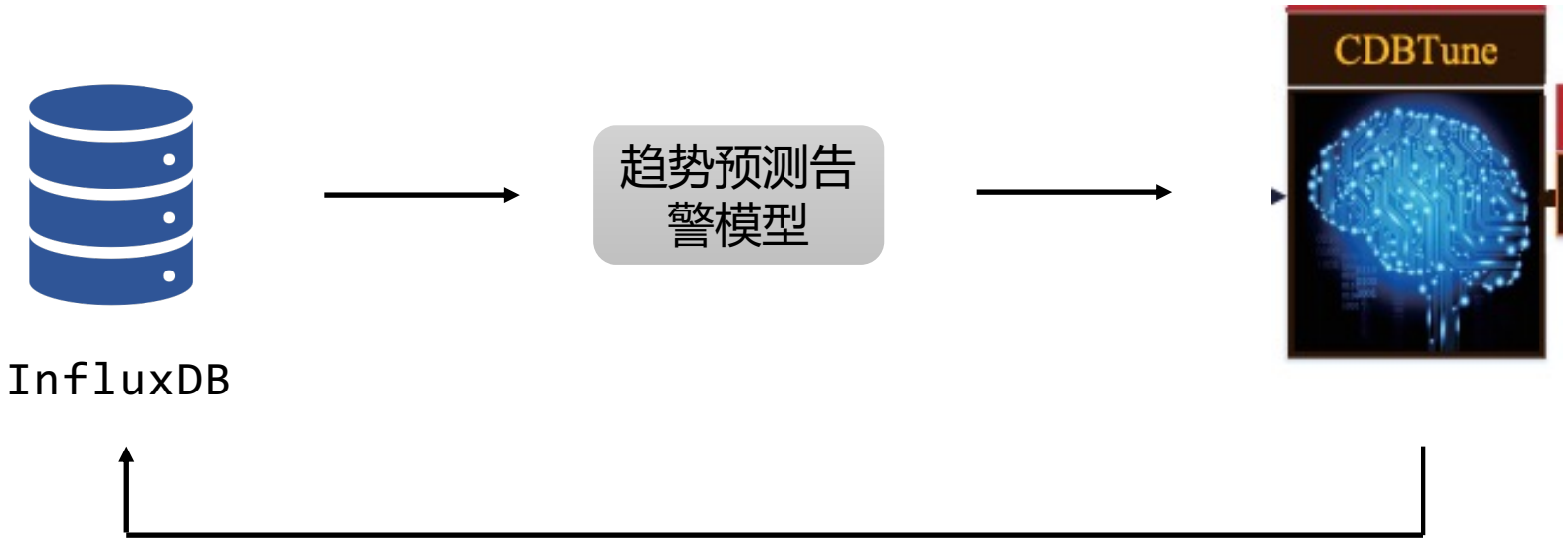
06 总结

DSC 分布式存储与计算实验室

	原理	优点	缺点
OtterTune	1. 分阶段式调优 2. 引入机器学习	1. 根据历史数据调优 2. 减少人工操作	1. 过度依赖高质量样本 2. 在高维空间调优效果不佳
CDBTune	1. 引入强化学习，实现端到端优化	1. 自我学习，探索各种可能的调参动作 2. 减小陷入局部最优的可能性	1. 冷启动 2. 缺少细粒度调优
Hunter	1. 利用并发压力测试 2. 使用GA算法预热DDPG模型	1. 优化冷启动	1. 缺少快速判断性能结果的方法
QTune	1. 提供三种调优粒度 2. 输入加入SQL语句特征	1. 解决粗粒度问题	1. 训练时间太长，模型复杂

自监督：监控多个指标，并预测每个指标未来变化趋势，提前告警

自配置：结合深度强化学习，实现自动化调参



06 文献

分布式存储与计算实验室

[01] - 李国良,周煊赫,孙佶,余翔,袁海涛,刘佳斌,韩越.基于机器学习的数据库技术综述[J].计算机学报,2020,43(11):2019-2049.

[02] - Van Aken D, Pavlo A, Gordon G J, et al. Automatic database management system tuning through large-scale machine learning[C]//Proceedings of the 2017 ACM international conference on management of data. 2017: 1009-1024.

[03] - Zhang J, Liu Y, Zhou K, et al. An end-to-end automatic cloud database tuning system using deep reinforcement learning[C]//Proceedings of the 2019 International Conference on Management of Data. 2019: 415-432.

[04] - Baoqing Cai, Yu Liu, Ce Zhang, Guangyu Zhang, Ke Zhou, Li Liu, Chunhua Li, Bin Cheng, Jie Yang, and Jiashu Xing. 2022. HUNTER: An Online Cloud Database Hybrid Tuning System for Personalized Requirements. In Proceedings of the 2022 International Conference on Management of Data (SIGMOD '22). Association for Computing Machinery, New York, NY, USA, 646–659.

[05] - Li G, Zhou X, Li S, et al. Qtune: A query-aware database tuning system with deep reinforcement learning[J]. Proceedings of the VLDB Endowment, 2019, 12(12): 2118-2130.