OUTLINE

Discussion

- Workload Compression & Data Management

Tutorial

- Advanced File System (FAST'21)
- Routine for Machine Learning

OUTLINE

Discussion

- **Workload Compression & Data Management**

Tutorial

- Advanced File System (FAST'21)
- Routine for Machine Learning

Discussion →Workload Compression & Data Management

Workload Compression & Data Management

- Critical features for AI (*domain knowledge like access pattern*)
- Less Overhead will be involved if metadata is available.
  - Aggregated info before real IO; E.g.

    (1) metadata from NAS controller,

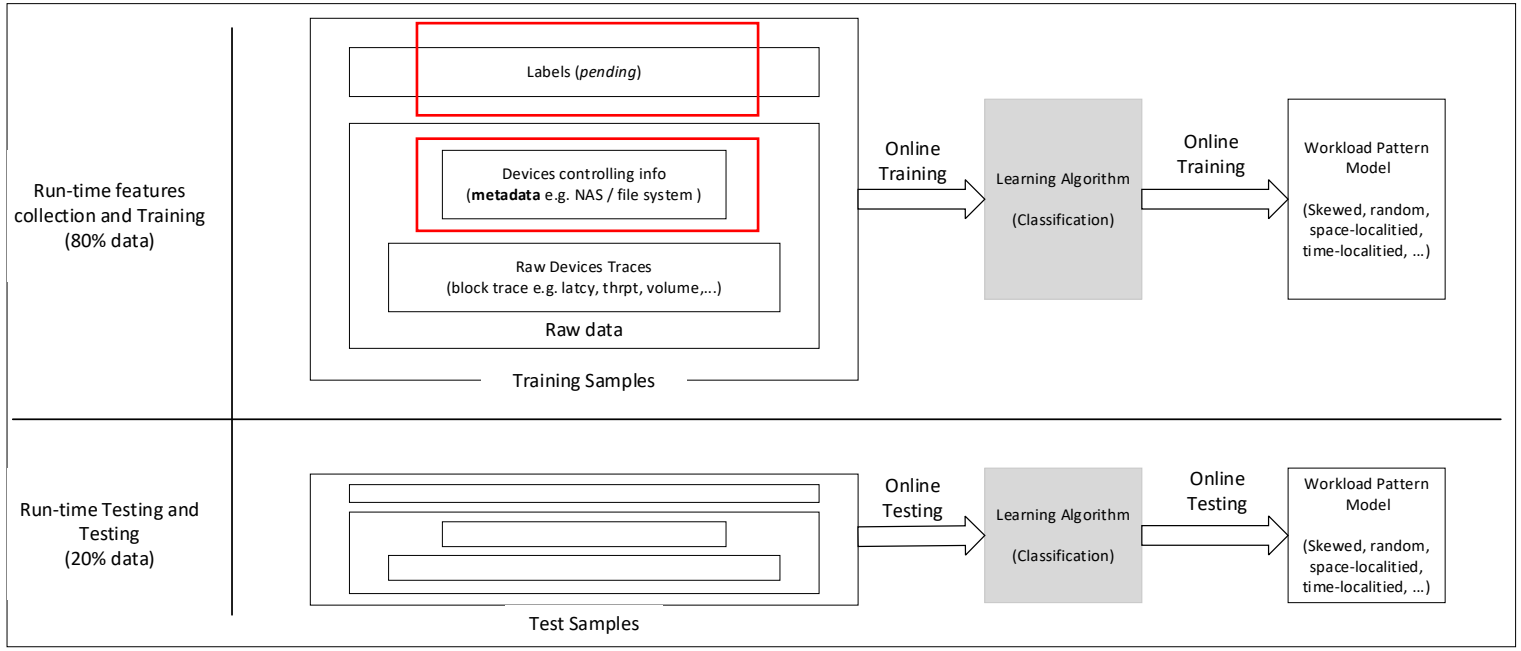
    (2) critical **command** of log-structured system can be retrieved with minimum overhead.
- Talk about the architecture (data management) before doing ML.

Discussion →Workload Compression & Data Management

Workload Compression

- Which workload features should be considered. (labels)
- Combining with device status (NAS) / application processing (database, file system)
- Lightweight online algorithm (scheme)

- Using *classification* to collect workload pattern. *(compression)* → *e.g., CNN*
- Using *Prediction* for workload replaying. (redrawing) → e.g., RL



OUTLINE

Discussion

- Workload Compression & Data Management

Tutorial

- **Advanced File System (FAST'21)**
- Routine for Machine Learning

OUTLINE

Discussion

- Workload Compression & Data Management

Tutorial

- Advanced File System (FAST'21)
- **Routine for Machine Learning**

Tutorial → Machine Learning

Introduction for ML

- 李宏毅2020机器学习深度学习
- RL Course by David Silver

Paper Reading

- An End-to-End Automatic Cloud Database Tuning System Using Deep Reinforcement Learning.
- Query-based Workload Forecasting for Self-Driving Database Management Systems

An End-to-End Automatic Cloud Database Tuning System Using Deep Reinforcement Learning.

Ji Zhang§, Yu Liu§, Ke Zhou§, Guoliang Li‡, Zhili Xiao†, Bin Cheng†, Jiashu Xing†, Yangtao Wang§, Tianheng Cheng§, Li Liu§, Minwei Ran§, and Zekang Li§ §Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology, China ‡Tsinghua University, China, †Tencent Inc., China

- Proposes end-to-end automatic database tuning system by adopting deep RL learning, so that performance can be enhanced even training data are not sufficient.
- **Background & Motivation**
  - Cloud database are widely adopted while it is difficult for users to figure out and handle the performance degradation which are caused by system configuration.
  - Cloud service provider are required to adjust the database system parameters for the user in a timely manner to ensure that the performance of the database is maintained in a better state. It is not realistic to entirely rely on database experts for tuning.
- **Basic Principles:**
  - CDBTune: (1) Offline training with database **pressure test**, collecting training data and training a preliminary configuration recommendation model. (2) When users or experts have optimization requirements, they can submit an **online parameter tuning request** through interface. The controller sends online tuning request to the model and adjusts the previous parameters according to the current load. (3) Repeat.
  - Model: **RL** is adopted because it can **train the model while generating data**, learning from success or failure. Hence, it does **not require** high **quality** of pre-training **samples**. RL optimizes the configuration of the network through signals (**changes in database performance**).

An End-to-End Automatic Cloud Database Tuning System Using Deep

Ji Zhang§, Yu Liu§, Ke Zhou§, Guoliang Li‡, Zhili Xiao†, Bin Cheng†, Jiashu Xing†, Yangtao Wang§, Tianheng

Laboratory for Optoelectronics, Huazhong University of Science and Technology, China ‡Tsinghua University,

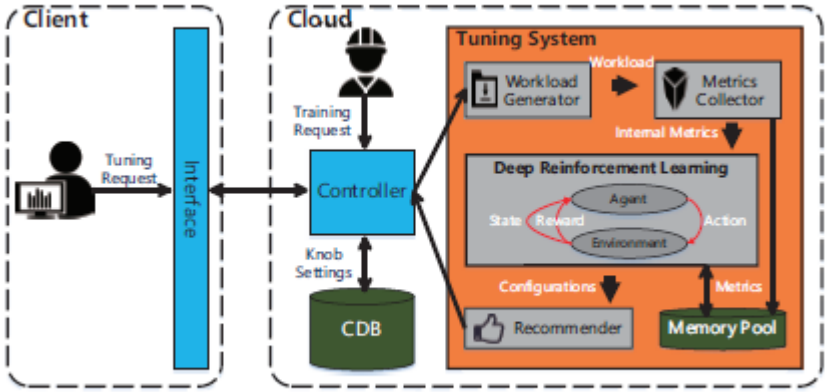


Figure 2: System Architecture.

- Proposes end-to-end automatic database tuning system by adopting deep RL lea training data are not sufficient.
- **Background & Motivation**
  - Cloud database are widely adopted while it is difficult for users to figure out and handle the performance degradation which are caused by system configuration.
  - Cloud service provider are required to adjust the database system parameters for the user in a timely manner to ensure that the performance of the database is maintained in a better state. It is not realistic to entirely rely on database experts for tuning.
- **Basic Principles:**
  - CDBTune: (1) Offline training with database **pressure test**, collecting training data and training a preliminary configuration recommendation model. (2) When users or experts have optimization requirements, they can submit an **online parameter tuning request** through interface. The controller sends online tuning request to the model and adjusts the previous parameters according to the current load. (3) Repeat.
  - Model: **RL** is adopted because it can **train the model while generating data**, learning from success or failure. Hence, it does **not require** high **quality** of pre-training **samples**. RL optimizes the configuration of the network through signals (**changes in database performance**).

An End-to-End Automatic Cloud Database Tuning System Using Deep Rei

Ji Zhang§, Yu Liu§, Ke Zhou§, Guoliang Li‡, Zhili Xiao†, Bin Cheng†, Jiashu Xing†, Yangtao Wang§, Tianheng Cheng

Laboratory for Optoelectronics, Huazhong University of Science and Technology, China ‡Tsinghua University, China,

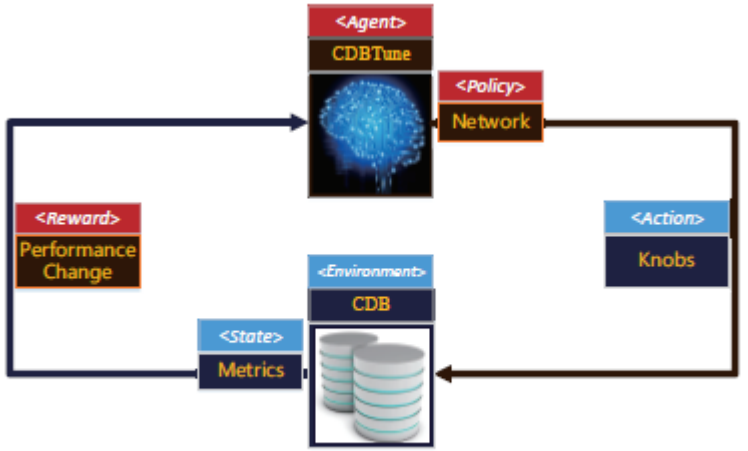


Figure 3: The correspondence between RL elements and CDB configuration tuning.

- Proposes end-to-end automatic database tuning system by adopting deep RL learning training data are not sufficient.
- **Background & Motivation**
  - Cloud database are widely adopted while it is difficult for users to figure out and handle the performance degradation which are caused by system configuration.
  - Cloud service provider are required to adjust the database system parameters for the user in a timely manner to ensure that the performance of the database is maintained in a better state. It is not realistic to entirely rely on database experts for tuning.
- **Basic Principles:**
  - CDBTune: (1) Offline training with database **pressure test**, collecting training data and training a preliminary configuration recommendation model. (2) When users or experts have optimization requirements, they can submit an **online parameter tuning request** through interface. The controller sends online tuning request to the model and adjusts the previous parameters according to the current load. (3) Repeat.
  - Model: **RL** is adopted because it can **train the model while generating data**, learning from success or failure. Hence, it does **not require** high **quality** of pre-training **samples**. RL optimizes the configuration of the network through signals (**changes in database performance**).

Query-based Workload Forecasting for Self-Driving Database Management Systems

Lin Ma, Dana Van Aken, Ahmed Hefny, Gustavo Mezerhane, Andrew Pavlo, Geoffrey J. Gordon; CMU

- **Background & Motivation**
  - For the Self-Driving DB, it is important to predict the workload in the future. If only referring to the past behavior when choosing an optimization method, it may be "short-sighted". In severe cases, DBMS will continuously initiate optimization operations.
  - The access speed of different queries of apps is different, resulting in the model needs to predict the access characteristics of each type of query. The composition and number of app queries will change over time, causing the model to need to be continuously updated.
- **QueryBot5000:**
  - The SQL queries will first enter the Pre_Processor, remove the parameter part and convert it into a SQL template, and record the number of accesses to each template to achieve the purpose of reducing computational complexity and storage overhead.
  - Enter the Clusterer component, and cluster the SQL templates according to the access frequency to further reduce the **computational complexity**.
  - The clustered data is used for **prediction**, and the final result is obtained.
  - Using ***access frequency*** as a feature for clustering, because such features *will not change with changes in the physical structure of the database* and can *accurately describe the workload.*
  - The **clustering center in this paper represents the entire class for predicting workload**, and this can *reduce computational overhead*, and is more suitable for ***online clustering scenarios***.

Query-based Workload Forecasti

Lin Ma, Dana Van Aken, Ahmed Hefny, Gustavo Mez

- **Background & Motivation**
  - For the Self-Driving DB, it is impo
    method, it may be "short-sighted".
  - The access speed of different queries of apps is different, resulting in the model needs to predict the access characteristics of each type of query. The composition and number of app queries will change over time, causing the model to need to be continuously updated.

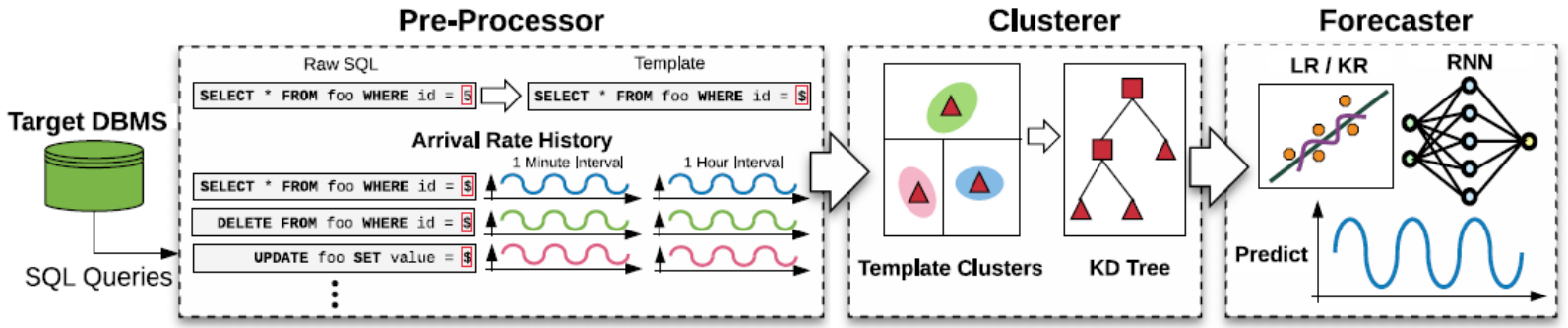


Figure 2: QB5000 Workflow – The framework receives SQL queries from the DBMS. This data is first passed into the Pre-Processor that identifies distinct templates in the workload and records their arrival rate history. Next, the Clusterer combines the templates with similar arrival rate patterns together. This information is then fed into the Forecaster where it builds models that predict the arrival rate of templates in each cluster.

- **QueryBot5000:**
  - The SQL queries will first enter the Pre_Processor, remove the parameter part and convert it into a SQL template, and record the number of accesses to each template to achieve the purpose of reducing computational complexity and storage overhead.
  - Enter the Clusterer component, and cluster the SQL templates according to the access frequency to further reduce the **computational complexity**.
  - The clustered data is used for **prediction**, and the final result is obtained.
  - Using ***access frequency*** as a feature for clustering, because such features *will not change with changes in the physical structure of the database* and can *accurately describe the workload.*
  - The **clustering center in this paper represents the entire class for predicting workload**, and this can *reduce computational overhead*, and is more suitable for ***online clustering scenarios***.