Project: Automatic Database Management System Tuning Through Large-scale Machine Learning

1. Project Description
Modern Database Management Systems (DBMSs) have tens or hundreds of knobs to tune. These knobs control different buffer sizes, log file sizes, flush methods, lock wait timeout, etc., which affect the running time (latency) of queries significantly. Tuning of DBMS knobs to minimize latency is a hard problem due to a large number of choices for these knobs. To address this challenge, this paper presents an automated approach that uses several machine learning methods to automatically recommend configurations of DBMS knobs based on the historical data and the data newly collected for the current query workload.

Specifically, this project will mainly focus on the latency prediction for running a query workload under a certain configuration. The goal of the project is to predict latencies for 100 <workload, configuration> pairs. You will need to implement a set of techniques described in the paper and finish the project with the following guidelines.

2. Reading Material
Please read the paper from the beginning up to the first two paragraphs in Section 6.2.

3. Algorithms to Implement
In this project, you are asked to implement the following techniques:

1. Pruning Redundant Metrics [Section 4.2]

In our dataset, we have hundreds of metrics being observed for running a query workload, including latency as an external metrics and hundreds of DBMS-specific internal metrics. Please implement “Factor Analyze” + “K-means” (Section 4.2) to prune the redundant metrics in order to speed up the ML algorithms and increase the likelihood that the models will fit in memory.

Note: Please keep at most 20 metrics after pruning.

2. Latency Prediction through Workload Mapping

The project will ask you to predict the latency for target workloads, each of which has only been observed under 5 different configurations. You are supposed to (1) apply the workload mapping (Section 6.1) to map the target workload to a well observed workload in the past, and (2) using the Gaussian Process Regression (GPR) model to predict the latency for a target workload under a specific configuration (see the first 2 paragraphs in Section 6.2)
**Note1:** By default, you can use the Squared Exponential kernel (aka the Gaussian Kernel) for GPR and set its hyperparameter length scale and the output variance to 1.

**Note2:** To compute the Euclidean distance for workload mapping, you will need to use the pruned metrics of the target workload under the 5 observed configurations and the same metrics from well observed past workloads under the same 5 configurations. It is possible that even the well observed workloads have not tried the 5 configurations used in the target workload. Thus, you need to build GPR models for each pruned metric to enable such workload mapping.

**Hint1:** To have a better latency prediction, you can tune the choice of the kernel function and its corresponding hyperparameters.

**Hint2:** Normalize the metrics into the same order of magnitude before compute the Euclidean distance.

### 4. Results to Produce

To produce the output of this project, you need to fill in the latency column for the test.CSV file we will provide, which contains the 100 <workload, configuration> pairs.

**Dataset**

Our dataset is the parsed traces when running 258 data analysis workloads in a Spark cluster, including:

1. 58 “offline” workloads as “well-observed” workloads, each with ~300 configurations (data points);  
2. 100 “online” workloads B, each with 6 data points;  
3. 100 “online” workloads C, each with 5 data points and 1 configuration for testing, where each “data point” consists of  
   (1) A workload id,  
   (2) A configuration, including 4 integer knobs, 1 Boolean knob, and 7 continuous knobs,  
   (3) An observed latency,  
   (4) 572 runtime metrics, which were produced from a feature engineering method.

**test.CSV** has 14 columns, including “workload id”, 12 knobs, and “latency prediction”, we will provide all the information for the first 13 columns and need you to fill the predicted latency in the “latency prediction” column.

In your final report, you should also describe the following items:

- How you handle the raw dataset including  
  o How to split data set into train/validation/test sets;  
  o How to normalize your data (metrics/latency/knobs), if so, how; if not, why.  
  o How to deal with the integer and Boolean knobs  
  o How to guarantee that your prediction performance is stable  
- What are the metrics retained after pruning (show in the attachment)  
- What is the “nearest neighbor” for each “online” workload (show in the attachment)
• What is the kernel function you used and what are the parameter values you used in your kernel function?

5. Ideas for Extension (20%)
This project also requires an extension. We offer a few ideas below. Please feel free to choose one idea to explore in your project. Students are also welcome to propose their own idea.

Extension 1: new approach to metrics pruning
• Your proposed pruning approach
• Compare the one in the paper and your own, explain the pros and cons, and the reason for favoring one to the other.

Extension 2: new approach to workload mapping
• Your proposed workload mapping approach
• Compare the one in the paper and your own, explain the pros and cons, and the reason for favoring one to the other.

Extension 3: new approach to latency prediction
• Your proposed latency prediction approach
• Are there any hyperparameters in your new approach? If so, how do you choose them?
• Compare the one in the paper and your own, explain the pros and cons, and the reason for favoring one approach to the other.