# OtterTune: Automatic Database Management System Tuning Through Large-Scale Machine Learning

15-300 Fall 2017
Shuli Jiang
shulij@andrew.cmu.edu
Nov.30.2017

## Project Description

I'll be working with Professor Andy Pavlo and his PhD student Data Van Aken at CMU Database Group (CMUDB) on the OtterTune project. OtterTune aims to apply large scale Machine Learning on tuning database configuration to improve the performance of database systems.

Database management system (DBMS) configuration tuning is essential to the performance of database. This is historically a difficult task because DBMSs have hundreds of configuration "knobs" that control everything in the system. However, these knobs are not standardized (i.e., two DBMSs use a different name for the same knob), not independent (i.e., changing one knob can impact others), and not universal (i.e., what works for one application may be sub-optimal for another). Information about the effects of the knobs typically comes only from expensive experiments by DBMS experts.

OtterTune is the new tool to overcome these challenges. It would use past experience of tuning knobs and new information to recommend new DBMS configurations. The basic pipeline of OtterTune consists of the following parts: (1) select the most impactful knobs, (2) map unseen database workloads to previous workloads from which we can transfer experience, and (3) recommend knob settings.

The OtterTune pipeline at current stage is only a prototype. It is still unclear how to recommend knob settings across different versions of database. So my research task is to figure out how to recommend knob settings of a new version database based on past knowledge of tuning an old version. It would essentially be a large scale machine learning based recommendation system specifically for database.

## Project Website

http://www.andrew.cmu.edu/user/shulij/15400

## Project Goals

75%: Complete the second version of the OtterTune pipeline. OtterTune could effectively recommend new knobs settings of a new version from past knowledge for at least one type of database.

100%: Complete and refine the second version of the OtterTune pipeline. OtterTune could effectively recommend new knobs settings of a new version from past knowledge for at least four database — which is currently using for testing and development— including PostgreSQL, MySQL, HANA and RocksDB. These four types of database include popular database of different types.

125%: Test and evaluate OtterTune on a real-world database from the industry.

**Milestone**

**1st Technical Milestone for 15-300**

We need to refine the whole OtterTune system first. On the highest level, the OtterTune system would consist of a controller and a server. The controller would be responsible for collecting data, namely knobs and metrics (database performance measurements) of the database, from querying a database under experiment in a certain period of time. The server is the pipeline which analyze data collected form the controller and generate knob settings recommendations. My task now is to refine the controller part. In the OtterTune prototype, the controller for collecting data relies on part of another project called `oltpbenchmark`. We want to extract out this part from `oltpbenchmark` as an independent component for OtterTune and adapt it so that it collects the data we need for doing knob settings from the database under experiment recommendation and works smoothly with the server.

By the end of this semester, I will finish building the controller (which has done 90% now). The controller is responsible for the following jobs:
1. Collect metrics and knobs information of the experimenting database before the experiment starts.
2. Record start time of the experiment.
3. Sleep for a while. This is when queries into the experimenting database would happen.
4. Record end time of the experiment.
5. Collect metrics information about the experimenting database again. Since knobs would not be affected by the experiment, it will not collect information on knobs again.
6. Output four files: knobs.json, metrics_before.json (metrics information collected before the experiment), metrics_after.json (metrics information collected after the experiment), summary.json.
The summary.json would contain information about the start and end time, database type (i.e. MySQL, PostgreSQL, etc.) and database version.
7. Send the output files to the remote server.

**Bi-weekly Milestone for 15-400 (subject to change)**

At this stage, the focus would be mostly on the server's side. The output files about metrics and knobs information from an experiment from the controller's side into the pipeline would often contain redundant metrics and all knobs are treated equally. In order to efficiently recommend knobs configuration, it is essential to prune redundant metrics and identify the most important knobs first. These two tasks would be the first two milestones.

**January 31st:** Finish building a module in the server that would effectively prune redundant metrics.

**February 14th:** Finish building a module in the server that could identify the most important knobs.

After this step, the focus would be more on workload mapping and the actual machine learning recommendation part, which has been proposed in Prof.Pavlo's first paper on OtterTune as a Gaussian Process(GP) regression. The recommendation part is still a prototype. So we need to further implement and refine it.

First, we want the recommendation system to generate and send tokens to the controller. Tokens would tell the controller the progress in the server because it might take the several more than 10 minutes to finish its job. So if the controller asks the server its current progress, it is necessary for the server to generate some feedback to the controller.

**February 28th:** Finish building the interaction token feedback system between the controller and the server.

At this stage, all the necessary parts before and after the GP recommendation system has been set up. The next steps would primarily focus on the GP recommendation system itself.

**March 21st, April 4th:** Finish first trial of implementing an actual GP recommendation system and test it against `oltpbenchmark`, a benchmark for testing database performances on four types of database, PostgreSQL, MySQL, HANA and RocksDB.

**April 18th:** Improve and refine the GP recommendation system. See what could be improved with GP regression. Try other machine learning methods for improving the recommendation system.

**May 2nd:** Integrate each part of the whole OtterTune system and make sure it works as expected. Start testing OtterTune on a real-world setting if everything is going on well.

**Literature Search**

[1] Dana Van Aken, Andrew Pavlo, Geoffrey J. Gordon, Bohan Zhang. "Automatic Database Management System Tuning Through Large-scale Machine Learning".

Proceedings of the 2017 ACM International Conference on Management of Data Pages 1009-1024.
http://db.cs.cmu.edu/papers/2017/p1009-van-aken.pdf

[2] Songyun Duan, Vamsidhar Thummala, Shivnath Babu. “Tuning Database Configuration Parameters with iTuned”.
https://users.cs.duke.edu/~shivnath/papers/ituned.pdf

**Resources Needed**

1) Install database for testing. Currently I only need to install PostgreSQL and MySQL on my laptop.
2) Clone the repository, oltpbenchmark, on GitHub. ([link] https://github.com/oltpbenchmark/oltpbench) This is the starting point of building the controller as mentioned before.
3) Access of CMUDB organization on GitHub. All codes, both the legacies from the past and future contributions, would go into this repository. ([link] https://github.com/cmu-db/ottertune)
4) Access of CMUDB dev machines. These machines have various of database installed. Using this machines for testing OtterTune is necessary to prevent my laptop form running out of disk space.