Automatic Database Management System Tuning Through Large Scale Machine Learning

Authors: Dana Van Aken, Andrew Pavlo, Geoffrey J. Gordon, Bohan Zhang

Presented by: Wesley Wotring

Main Idea and Context

The development of an automated Database management systems tuning tool called Ottertune.

Ottertune was developed in order to automatically tune Database management system knob configurations using past knob configuration data along with machine learning methods in order to optimize the knob configurations to best fit a target workload.

The developers from Carnegie Mellon University developed Ottertune using supervised learning, however reinforcement learning could be an even more optimal and efficient method to perform the task of Database management system knob configuration tuning. [2]
How can we extend OtterTune for a future upgrade?(Shreya, Pranitha)

Why was there a need to automate this?

As of when this study and development was conducted, the main method of optimizing the tuning of knob configurations was to hire a professional to come out and do the job MANUALLY while also taking a duplicate copy of the database in order to try trial and error tuning on individual knobs.

What are the other automatic tuning system apart from OtterTune? (Soumen)

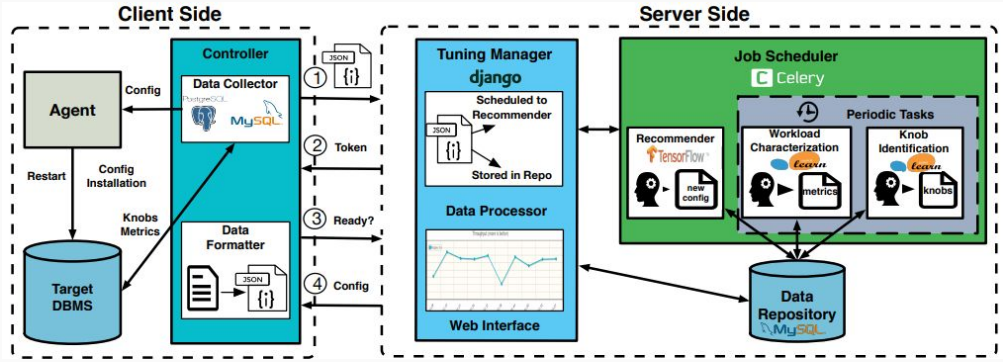
Other tuning systems include MySQLTuner, DBMS Auto-tuning by IBM, Query Load Balancing by Microsoft, iTuned, and Apache Ignite.

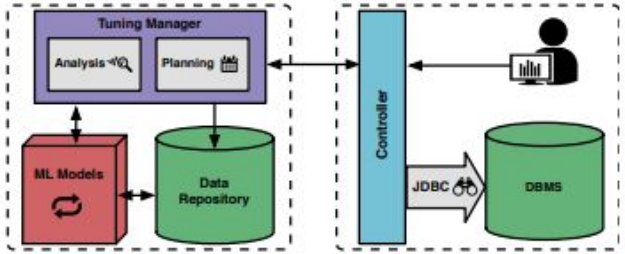
Why Ottertune if other tools are available?

Ottertune was built under the idea of versatility to be available for a multitude of database management systems where the ones listed previously only address their respective database management system. [2]

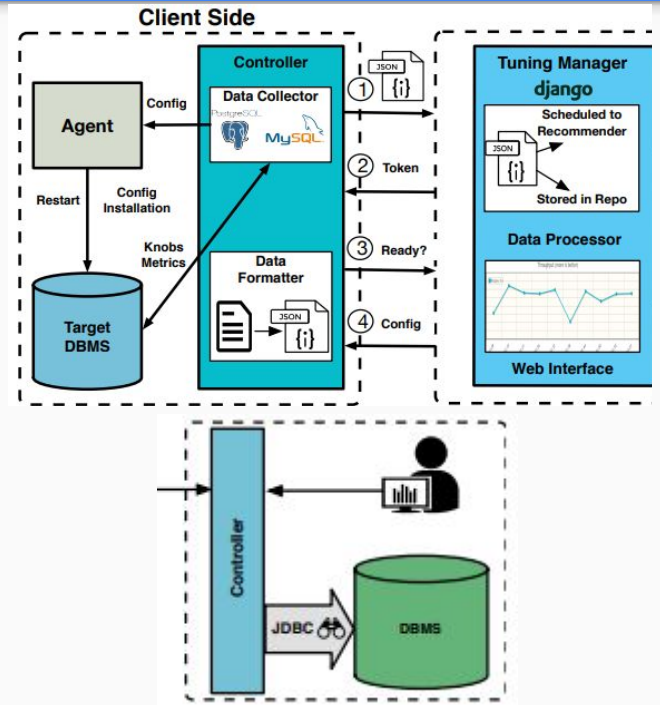
How Ottertune Works

- Ottertune has 2 components being a client side controller and a server side tuning manager. [1]
- The two components work in conjunction with one another in order to tune the knob configurations. [1]





Client Controller

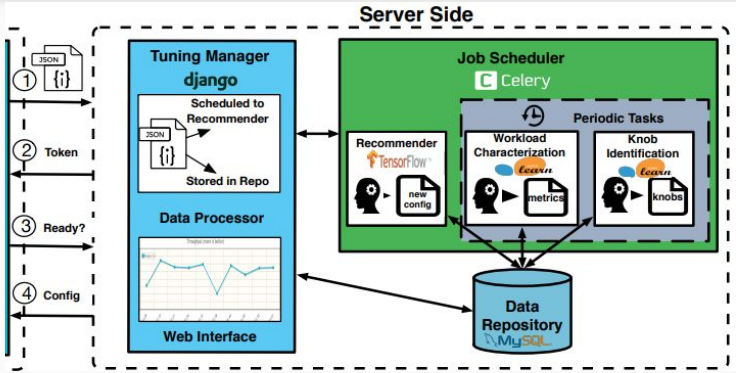


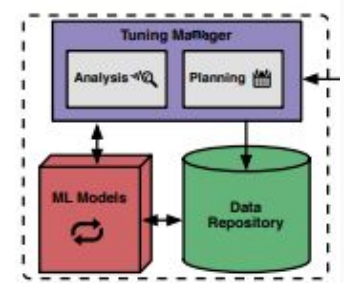
The client controller first connects to the database management system to collect the current knob configuration and takes a note of the metrics of the system. [1]

The controller then converts the information taken into a schema and sends it to the manager. [1]

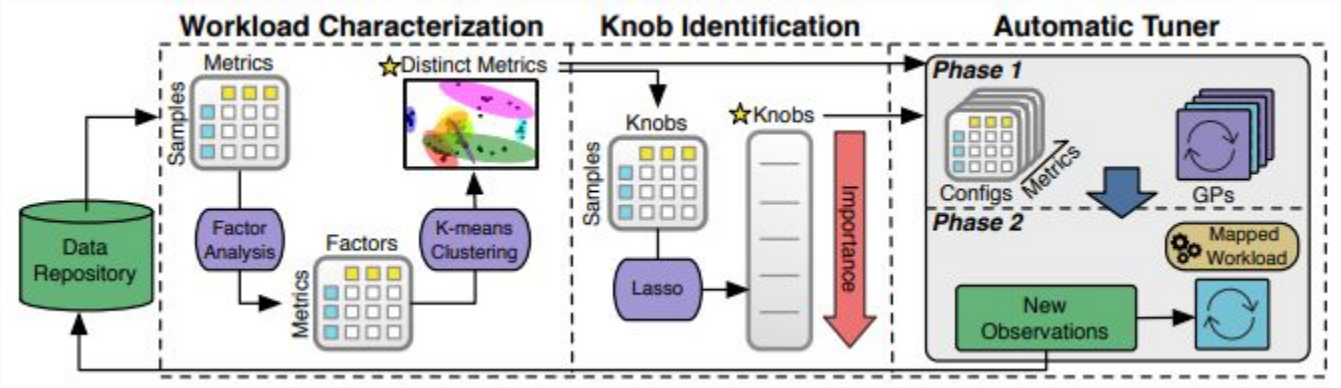
Tuning Manager

Once the Client Controller sends the data to the tuning manager, the tuning manager then uses a Celery asynchronous task in order to test the new incoming data to machine learning models. [1]





Ottertune Machine Learning Pipeline



Machine Learning Introduction

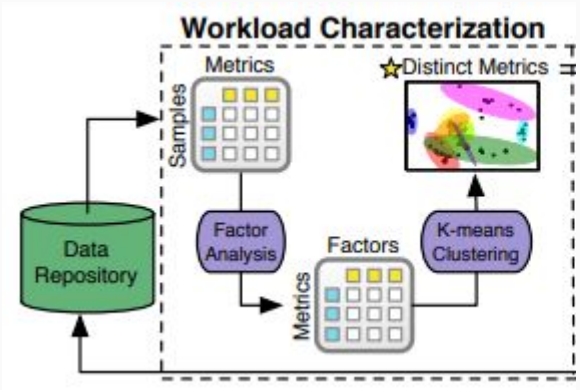
- Why does the use of ML make sense for this problem?
Machine learning makes sense for the problem at hand with the idea that this machine learning pipeline allows for the automation of tuning knob configurations, but also optimizes them to become more efficient over time. [2]

- How does it build upon, utilize, or extend other work?
Other work it only really builds upon database management system knobs and how the current method of tuning these knobs by now automating the process instead of having a paid professional come out and do it manually with a similar process. [2]

Workload Characterization

The approach taken to achieve this was to use DBMS runtime metrics to characterize how a workload behaves.

Because all modern DBMS expose large amounts of information about the system, this information helps provide accurate depiction of the workload runtime behavior. [2]



Workload Characterization Challenges

Collection of Statistics:

This was solved by prefixing the sub elements into single sum scalar values due to Ottertune only considering global knobs.

In 4.2 paper mentions about superfluous metrics, what exactly is that and what was the need to remove it? (Atharva) It is important to remove the irrelevant metrics that do not aid in the tuning in order to obtain the smallest set of metrics to accurately capture the variability of workload performance. This improves speed, accuracy, and generalizability.

Redundant Metrics:

The researchers used Factor Analysis that transforms high dimensional data into lower dimensional data along with *k-means* to cluster this lower dimensional data into groups. This is to reduce noise. [2]

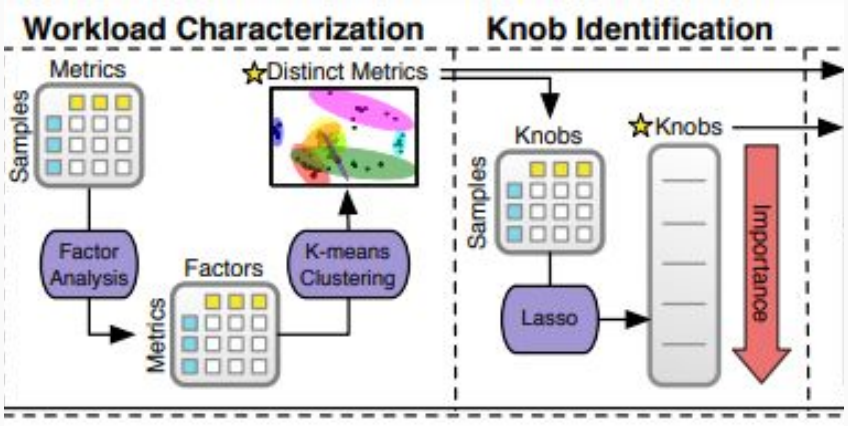
Pruning process

In the Pruning Redundant Metrics part, as for the FA algorithm, is there any measuring standard for removing metrics? I mean, how close are those redundant metrics if we will remove them? (Wangjiaxuan)

Factor Analysis algorithm takes in a matrix (X) where the rows correspond to metrics and columns correspond to knob configurations that are tried.

The Factor Analysis then reduces the dimensionality making a smaller matrix now with rows of metrics and factors. Which can be used to make a scatter-plot where now the metrics removed are the metrics that are too close to one another on the plot.

Knob Identification



In the second phase of the machine learning pipeline the developers used the Lasso regression algorithm on the scatter plotted data in order to identify the most important features or metrics while also diminishing the coefficients of the lesser important metrics to zero. Creating a ranked list of the important metrics. [2]

Lasso and Linear Regression

In the Feature Selection with Lasso part, the regularized version is adding an L1 penalty, which is that a constant λ times the sum of absolute weights to the loss function, so why not L2 penalty, i.e., replace the sum of absolute weights with the square root for the sum of weights' squares? (Wangjiaxuan)

The difference between L1 and L2 penalties in this case is that L1 penalty shrinks coefficients to zero in order to make the more important features weigh more than the redundant ones. L2 penalties spread out the coefficients values which would hurt the knob selection process.

Peer Questions Regarding Lasso

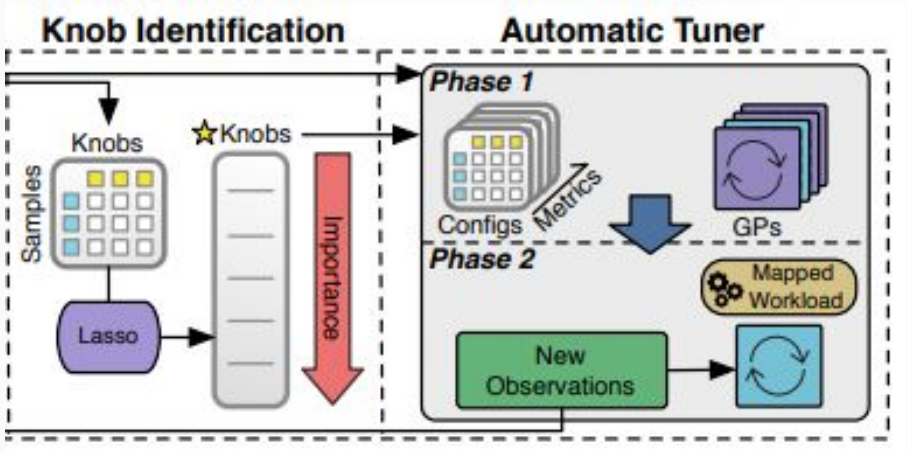
What can be used other than lasso regression? (Atharva)

Why do they use the Lasso algorithm? Why not use a more complex algorithm? (Yihe)

Ridge regression, Elastic Net regression, Random Forest regression, Support Vector regression, and Neural Networks could all replace Lasso regression. However the developers chose Lasso regression because it provides a sparse solution which is good for identifying the knobs that have the largest impact on efficiency.

Automatic Tuning

The goal of the Workload mapping step/phase is to now match the target Workload to a similar workload from the data repository based on performance measurements. [2]



Workload Mapping

The goal of workload mapping is to now identify similarities and differences between workloads and to use this information to predict the performance of a configuration on a new workload.

In order to achieve this Ottertune takes the target workload vector from the *k-means* and calculates the Euclidean distance between the target's vector and each workload in the matrix and computes a score for each workload.

The algorithm then selects the workload with the lowest score as the closest related workload. [2]

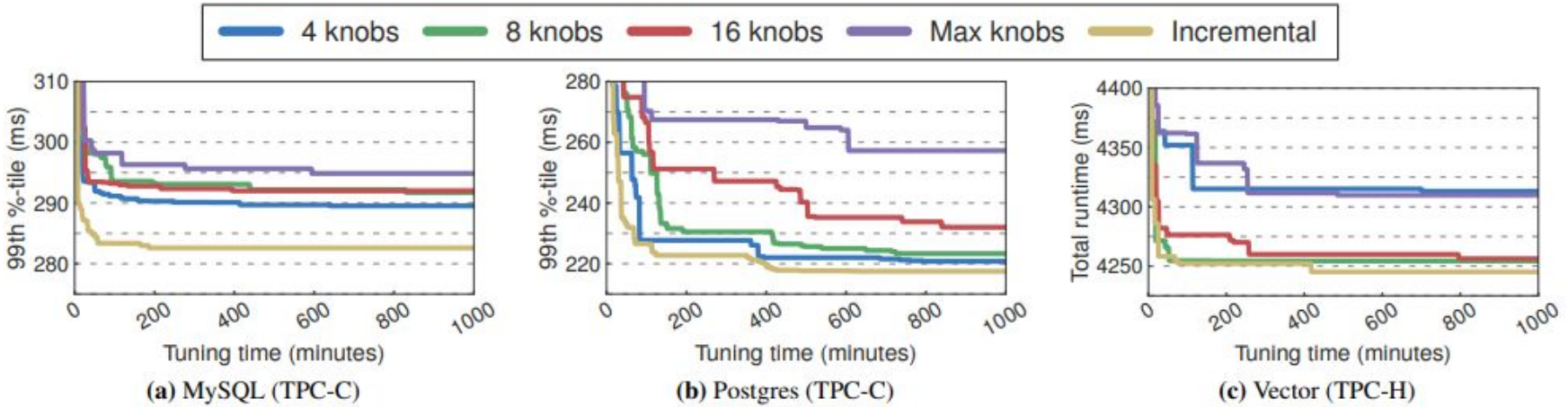
Configuration Recommendation

Ottertune starts the recommendation step by reusing the data from the workload that was selected in the mapping phase to train a Gaussian Process model (GP).

Then updates the model with adding metrics from the target workload that has been observed. Although since the mapped workload is not identical to the targeted workload the system does not fully trust the prediction and is handled with increasing the variance of noise for all points in the GP model.

Then there are 2 strategies that the tuner can take in exploitation and exploration where the model chooses to either search an unknown region in the GP which helps Ottertune look for unknown configurations or exploitation where a good configuration was found and Ottertune slightly modifies the knobs to try and improve performance. [2]

Evaluation Pt. 1 Knob Number



(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.

Evaluation Question

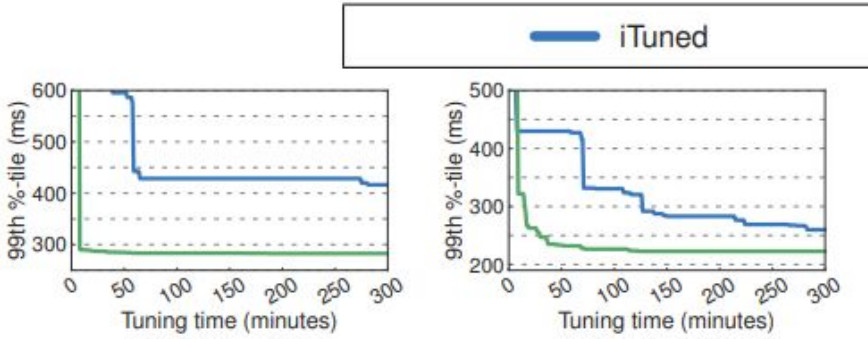
What is 99%-tile? (Xiangcheng)

The 99th percentile is the author's way of saying that the 99% of the queries were completed within that amount of time with the fixed number of knobs to configure.

What are the exact numbers of knobs in each incremental experiment? Are they all over 16? (Xiangcheng)

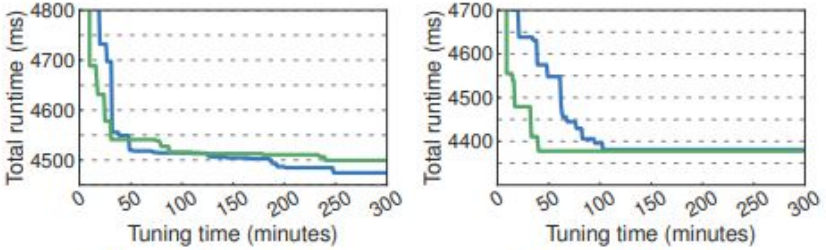
The number of knobs in the incremental experiment starts out with 4 but increases every 60 minutes by two in order to investigate how performance is impacted by increasing the number of knobs and help identify the knobs most crucial on improving performance.

Evaluation Pt. 2 iTuned Comparison



(a) MySQL **(b)** Postgres

Figure 6: Tuning Evaluation (TPC-C) – A comparison of the OLTP DBMSs for the TPC-C workload when using configurations generated by OtterTune and iTuned.



(a) Vector (TPC-H #1) **(b)** Vector (TPC-H #2)

Figure 8: Tuning Evaluation (TPC-H) – Performance measurements for Vector running two sub-sets of the TPC-H workload using configurations generated by OtterTune and iTuned.

Concluding Results

OtterTune produces configurations that achieve up to 94% lower latency compared to their default settings or configurations generated by other tuning advisors and generates configurations in under 60 min that are comparable to ones created by human experts. [2]

Additional Questions

Give some real world examples of OtterTune being used and how did they benefit from it? (Shreya, Tansuri, Pranitha)

Outside of Carnegie Mellon University (The authors who conducted the study), Companies such as Amazon, Google, and even Uber have used this for their databases and Uber even found that Ottertune was able to optimize their Database performance with throughput up to 30% and reduce latency by 40%.

Given the active environment of this problem and the goal of continuous improvement, is it possible to instead apply a reinforcement learning based approach to this problem? Is there some reason such an approach would not be effective? (Chris, Uzochi)

Yes reinforcement learning could be applied to Ottertune and could be used to find optimal configuration parameters, however the size or possible configurations might be to vast for the agent to find optimal solutions, but can be addressed with deep neural networks.

Resources

[1] B. Zhang, D. Van Aken, J. Wang, T. Dai, S. Jiang, J. Lao, S. Sheng, A. Pavlo, and G. J. Gordon, “A demonstration of the OTTERTUNE Automatic Database Management System Tuning Service,” Proceedings of the VLDB Endowment, vol. 11, no. 12, pp. 1910–1913, 2018.

[2] D. Van Aken, A. Pavlo, G. J. Gordon, and B. Zhang, “Automatic Database Management System tuning through large-scale machine learning,” Proceedings of the 2017 ACM International Conference on Management of Data, 2017.

Thank you for your time, any additional questions?