Automatic Database Management System Tuning Through Large-scale Machine Learning

Van Aken Dana *et al.* [1]

LSDPO (2017/2018)
Paper Presentation: Ioana Bica (ib354)

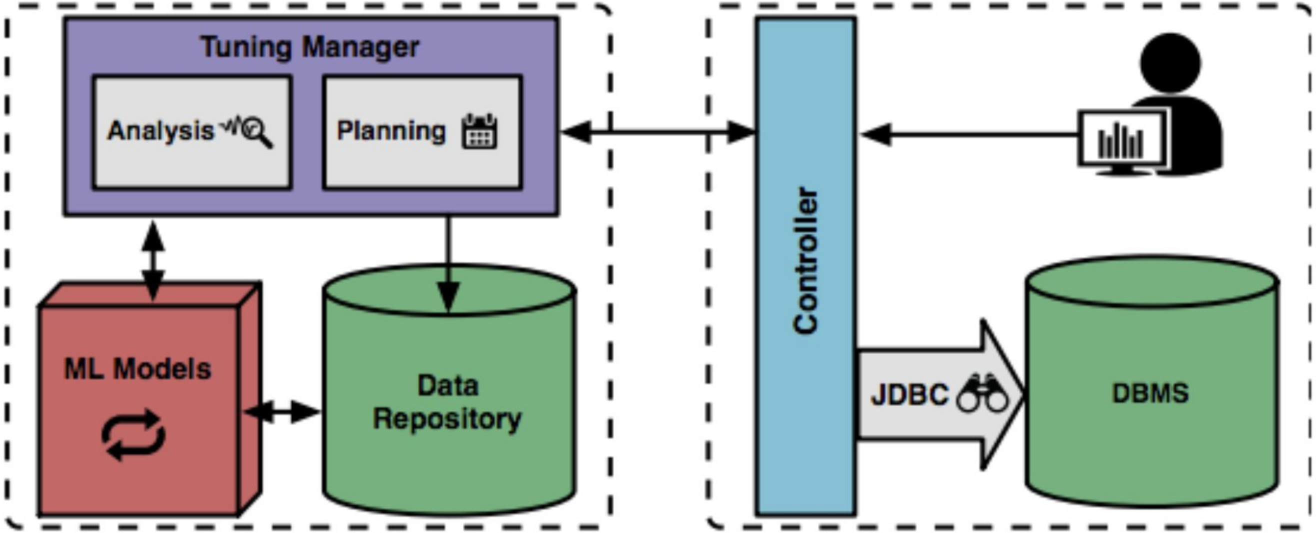
Problems with database management systems (DBMS) configuration tuning

- Standard approach: employ a database administrator (DBA) to tweak knobs through “trial-and-error”
- Main problems:
  - Dependencies
  - Continuous Setting
  - Non-reusable configurations
  - Tuning complexity

OtterTune

- Reduces the required input from the DBA.
- Works for any DBMS.
- Uses machine learning models through different stages of the system.
- Continuously uses new data and reuses previous training data to incrementally improve the models used for predicting good configurations.

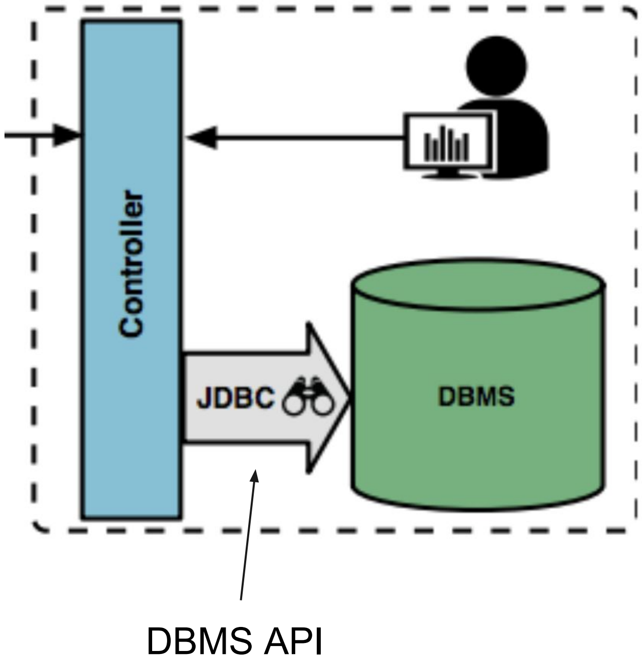
System architecture



System architecture

At the beginning of the tuning session:

- DBA specifies which metric OtterTune needs to improve.
- Controller connects to target DBMS and starts observation period.



Observation period

Aim: Collect current knob configuration and runtime statistics for both DBMS-independent external metric and DBMS-specific internal metric.

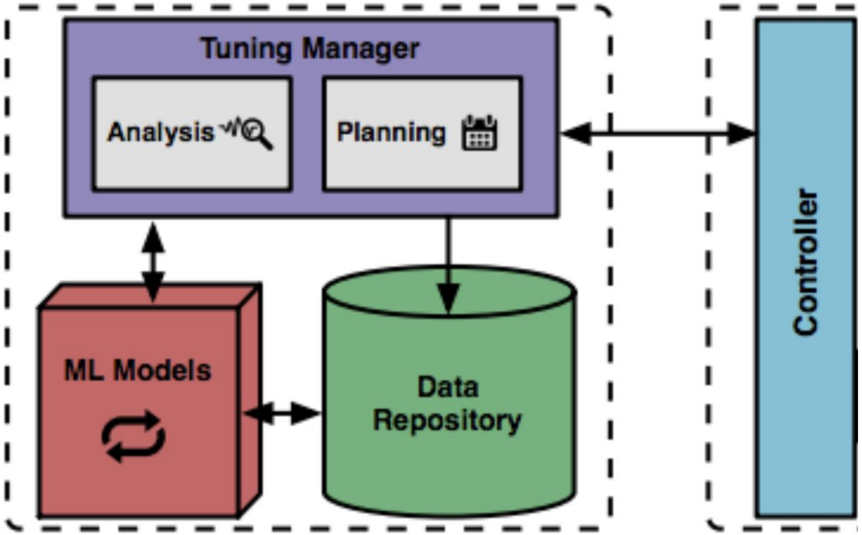
Main steps performed by the controller:

1. Reset statistics for target DBMS.
2. Execute some workload trace or a set of queries specified by the DBA.
3. Observe DBMS and measures metrics specified by DBA.
4. At the end, collect additional DBMS-specific internal metrics.
5. Store metrics with the same name as a single sum scalar value.

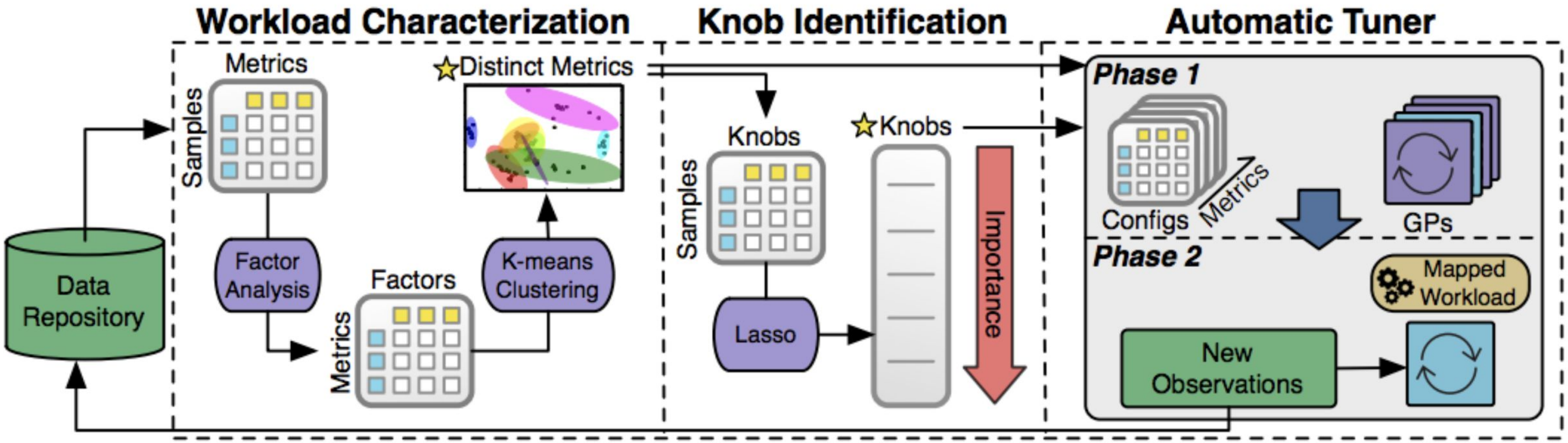
System architecture

After the observation period:

- Controller sends results to the tuning manager.
- Tuning manager stores all of the information in the data repository.
- OtterTune identifies the next configuration that should be installed on the DBMS.



Machine Learning Pipeline



Workload identification

Aim: identify characteristic aspects of target workload.

- Make use of the runtime statistics recorded while executing workload.
- OtterTune is DBMS independent since metrics collected do not need to be labelled.

Prune redundant metrics

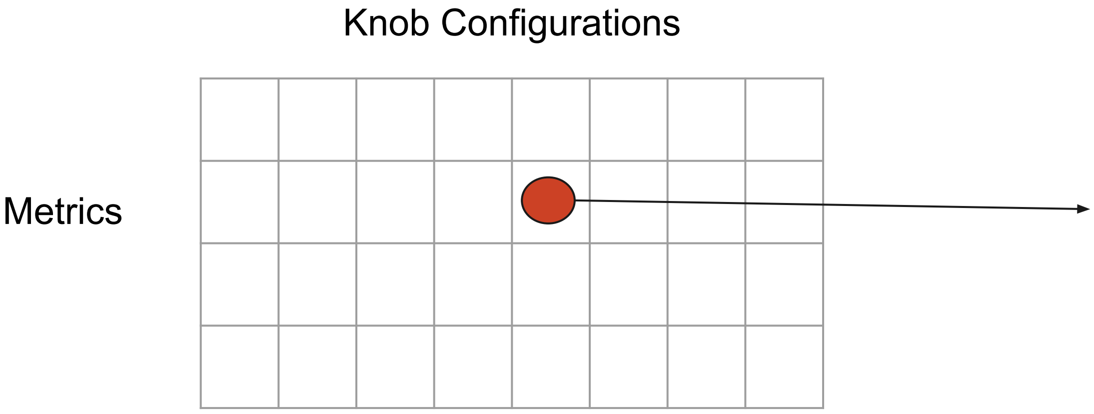
Factor Analysis

- Pre-processing step.
- Dimensionality reduction.
- Reduce the noise in the data.

k-means clustering

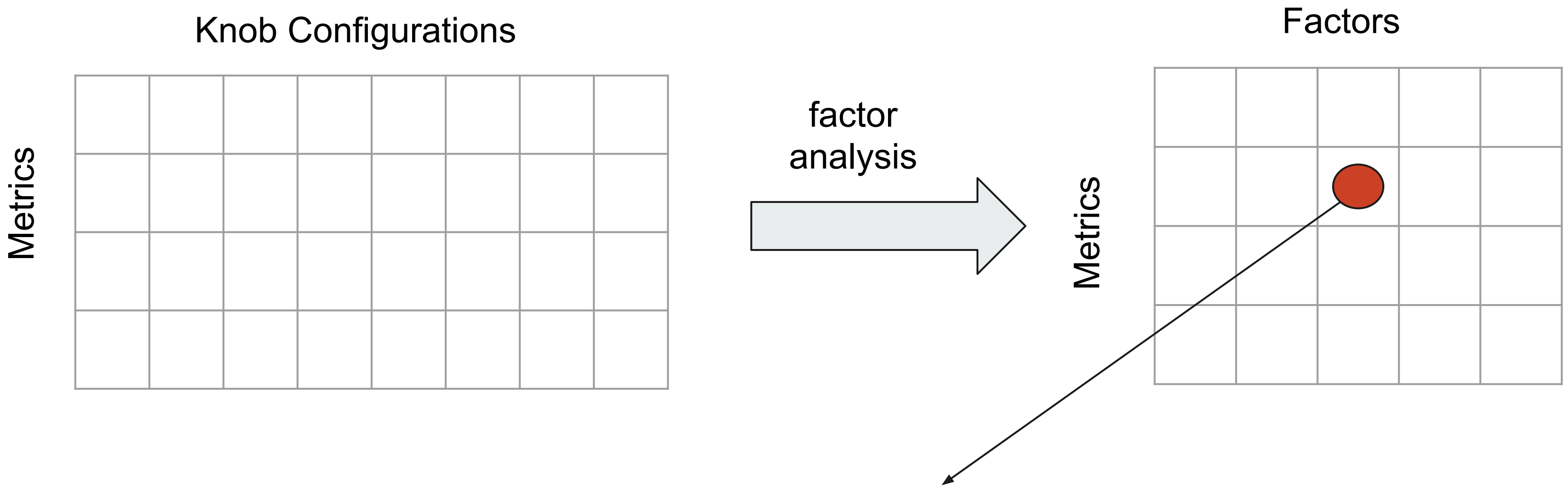
- Find groups of metrics similar to each other.
- Select one metric from each group.

Factor analysis



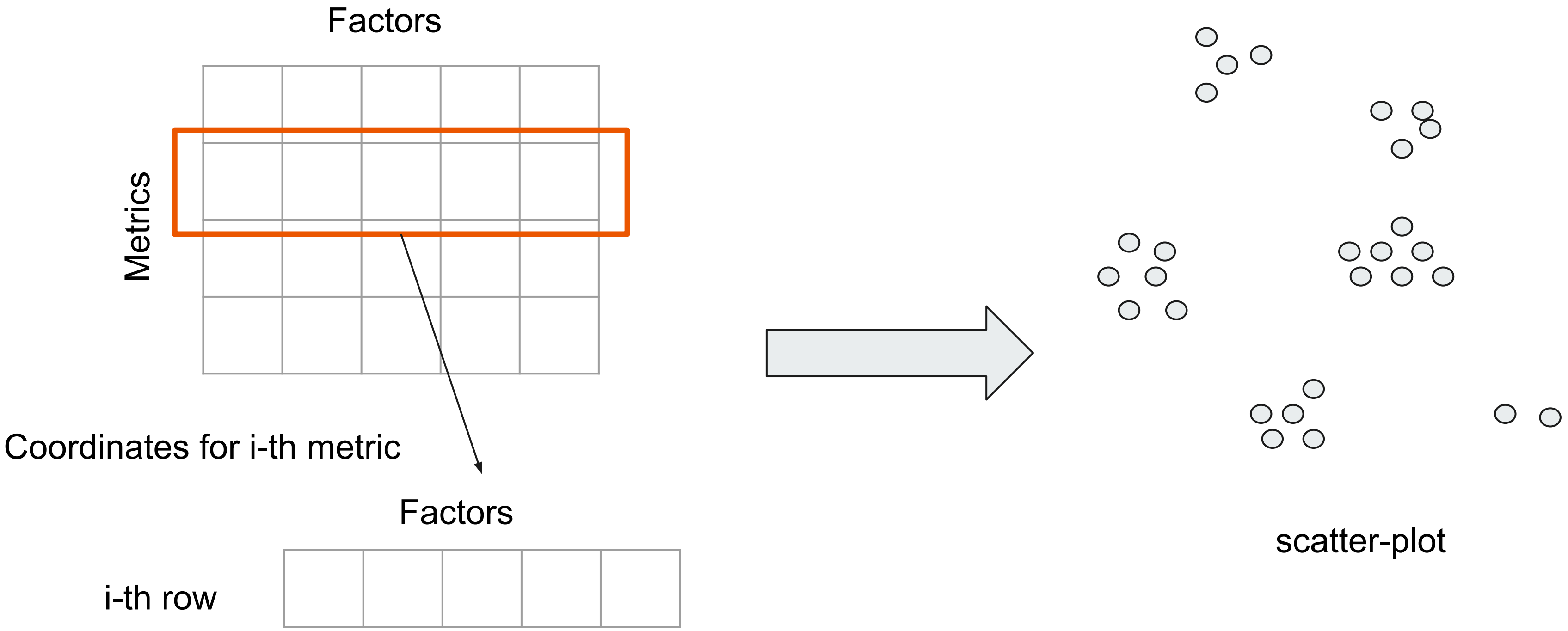
X_{ij} = value of metric i under configuration j

Factor analysis

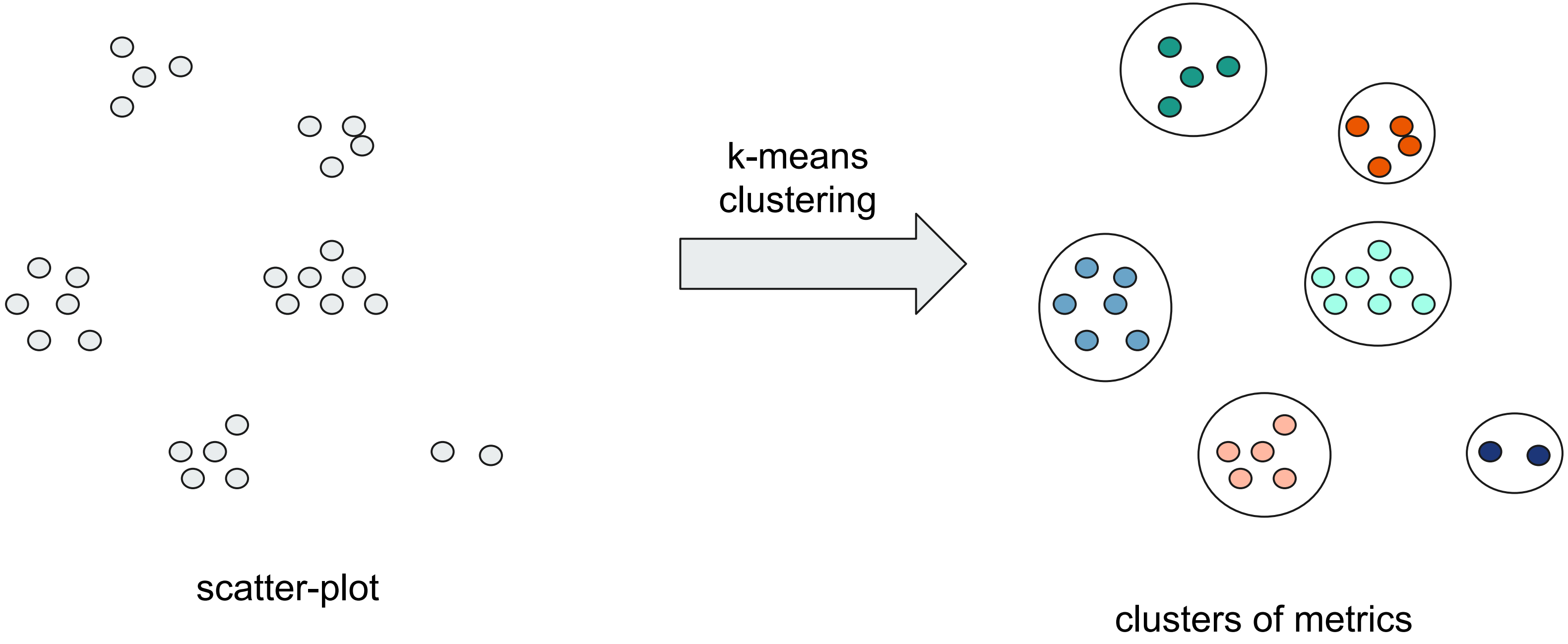


U_{ij} = coefficient of metric i in factor j

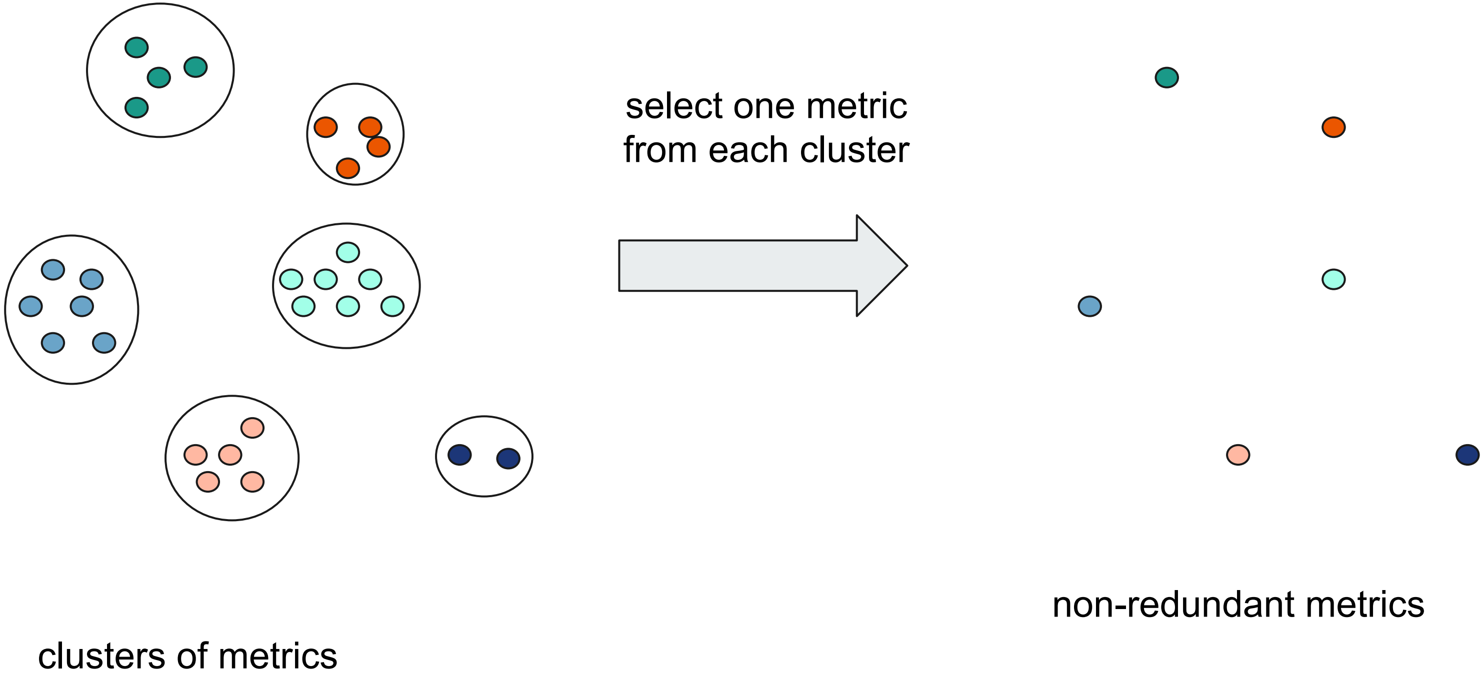
Scatter-plot



k-means clustering



Select one metric from each cluster



Identify important knobs

- Find knobs that affect system’s performance.

- Identify dependencies between knobs by adding polynomial features.

- Dynamically increase the number of knobs used in the tuning session.

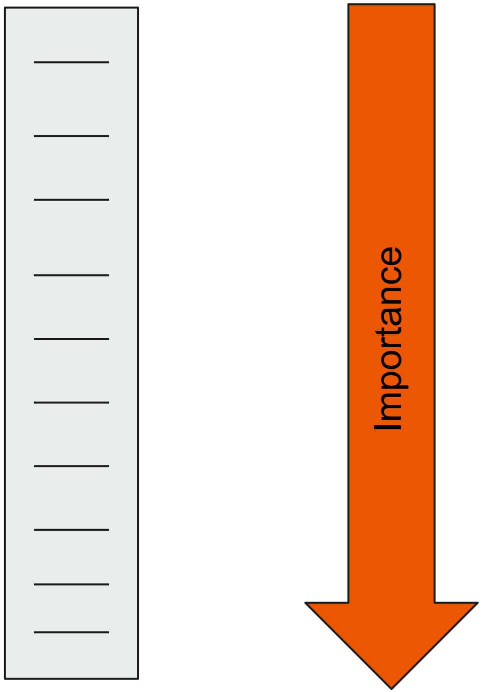
Lasso regression

Aim: find relationship between knob (or functions of knobs) and metrics.

- Variant of linear regression.
- Adds an L1 penalty to the loss function.

- Remove irrelevant knobs by shrinking their weights to zero.
- Orde knobs by order of appearance in regression.

Knobs
(or functions of knobs)



Automatic tuning

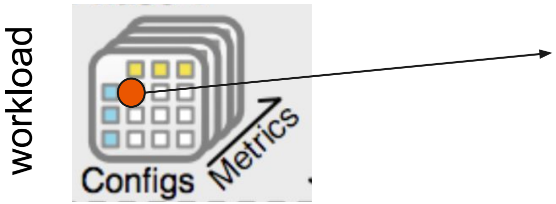
Workload mapping

- Find workload in the data repository similar to the target workload.

Configuration Recommendation

- Use Gaussian Process (GP) regression to find knob configuration that would target metric.

Workload mapping

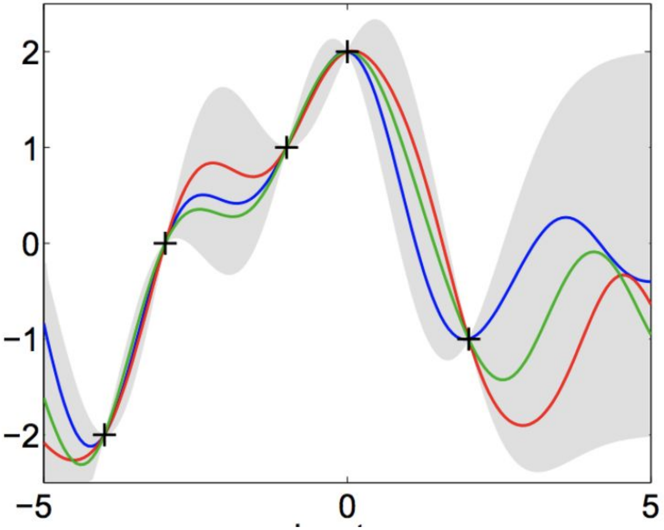


X_{mij} = value of metric m when executing workload i for configuration j

- For each metric m:
  - Compute Euclidean distance between target workload and each other workloads i.
- Compute score for workload i by averaging distance over all possible metrics.
- Select workload with lowest score.

Gaussian Process (GP) regression

- Use data from mapped workload to train a GP model.
- Update model by adding observed metrics from target workload.



http://mlg.eng.cam.ac.uk/teaching/4f13/1718/

Exploration

- Search unknown areas of the GP.
- Useful for getting more data.
- Helps identify configurations with knob values beyond limits tried in the past.

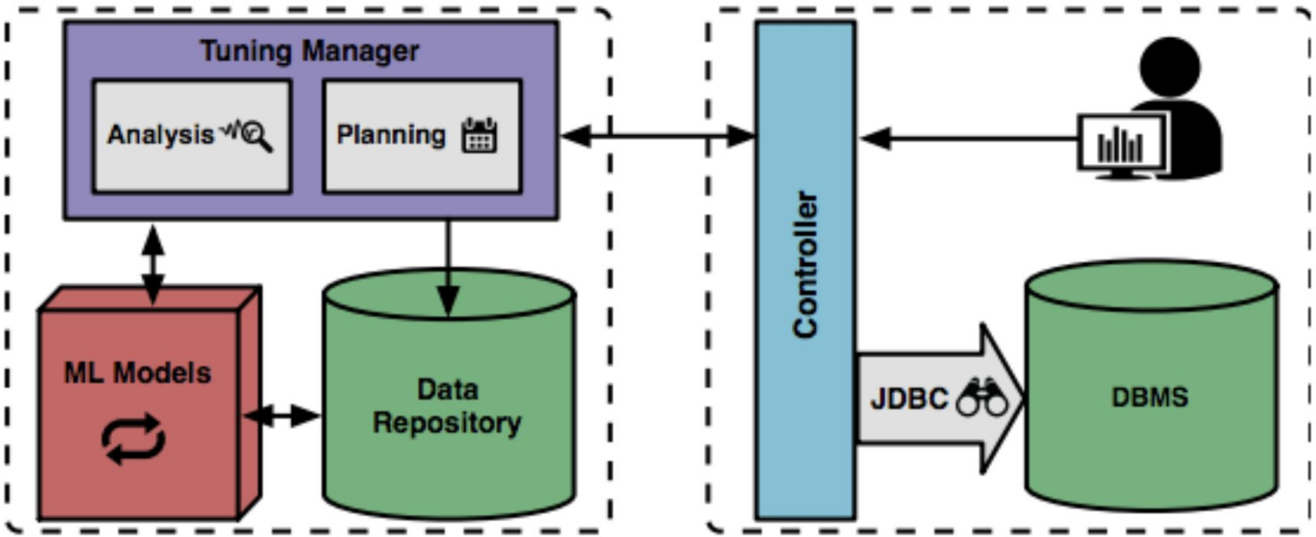
Exploitation

- Select configuration similar to best configuration found in the GP.
- Makes slight modifications to previously known good configurations.

Configuration recommendation

- Exploration/Exploitation strategy depends on variance of data points.
- Always select configuration with greatest expected improvement.
- Use gradient descent to find the configuration that maximizes potential improvement.
  - Initialization set: top-performing configurations + configurations for which knob values are selected randomly.
  - Finds local optimum on surface predicted by GP.

System architecture



Evaluation

DBMS evaluated

MySQL	PostgreSQL	Actian Vector
OLTP DBMS	OLTP DBMS	OLAP DBMS

Workloads

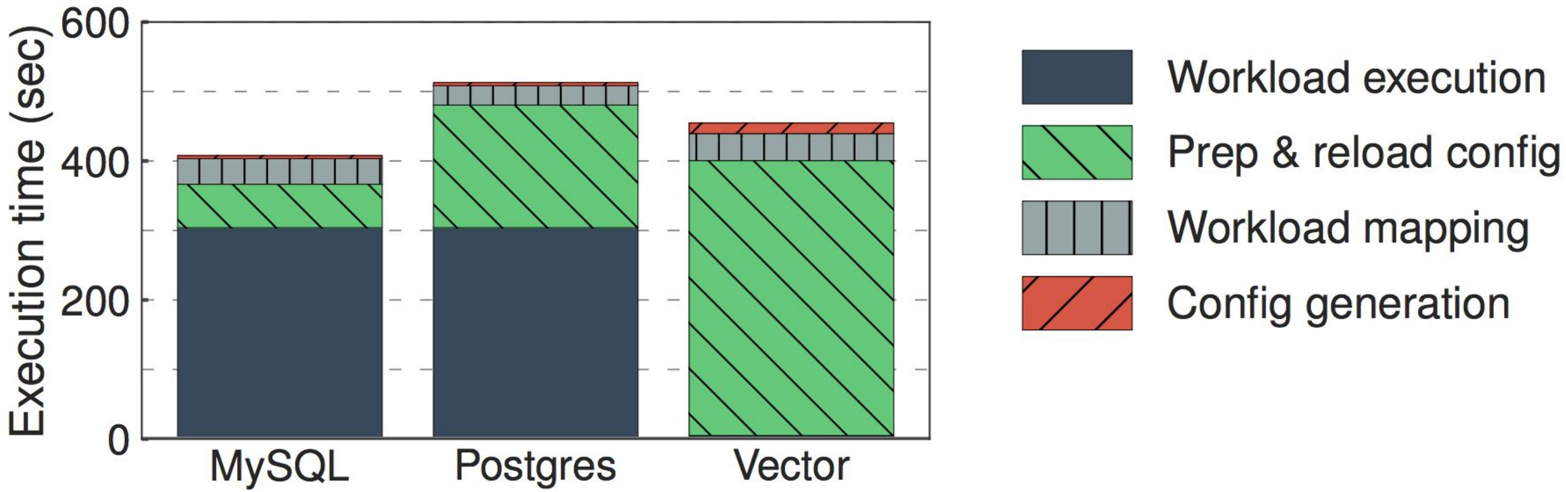
YCSB	• Yahoo! Cloud Serving Benchmark (OLTP) • Simple workload with high scalability requirement. (18m tuples)
TPC-C	• OLTP benchmark • Simulates an order processing application. (200 workhouses)
Wikipedia	• OLTP benchmark • Transactions -> most common operations in Wikipedia for article and “watchlist” management. (100k articles)
TPC-H	• Simulates OLAP environment • Little prior knowledge of queries.

Elements evaluated

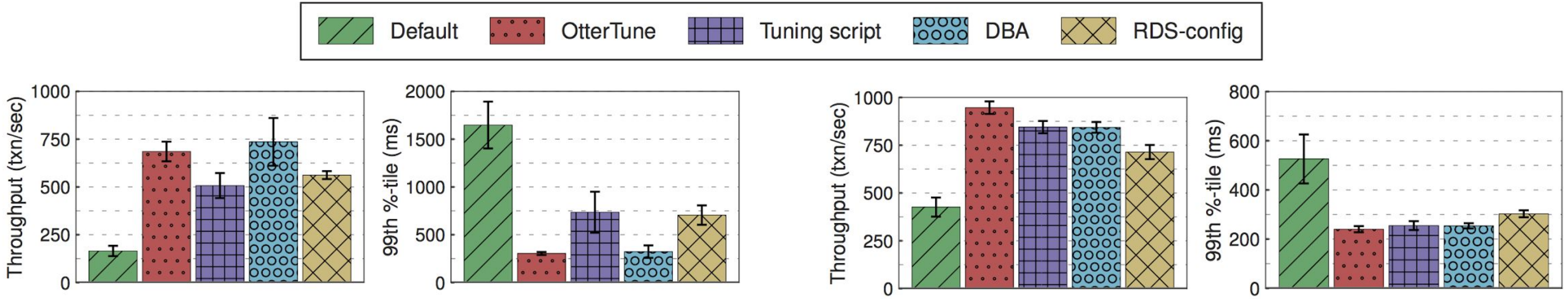
- Influence of the number of knobs used in the performance.
  - The incremental approach works best for all DBMSs.
  - OtterTune identifies the optimal number of knobs that should be tuned.
- Comparison with iTuned [2].
  - Demonstrates that continuously integrating new training data helps with performance.
  - OtterTune works much better on OLTP workloads, but it has similar performance with ITuned on OLAP workloads.

Note: Before starting the evaluation, training data was obtained to bootstrap OtterTune's repository.

Execution time breakdown



Efficacy Evaluation



(a) TPC-C (Throughput) **(b)** TPC-C (99%-tile Latency)

Figure 10: Efficacy Comparison (MySQL) – Throughput and latency measurements for the TPC-C benchmark using the (1) default configuration, (2) OtterTune configuration, (3) tuning script configuration, (4) Lithuanian DBA configuration, and (5) Amazon RDS configuration.

(a) TPC-C (Throughput) **(b)** TPC-C (99%-tile Latency)

Figure 11: Efficacy Comparison (Postgres) – Throughput and latency measurements for the TPC-C benchmark using the (1) default configuration, (2) OtterTune configuration, (3) tuning script configuration, (4) expert DBA configuration, and (5) Amazon RDS configuration.

Assumptions and limitations of OtterTune

Assumptions

- Assume that the OtterTune controller has administrative privileges on the DBMS.
  - If not, DBA needs to deploy a second copy for trials.

- Assume that the DBA is aware of dangerous knobs which they can add to a blacklist of knobs that OtterTune does not change.

- Assume that physical design of database is reasonable. (e.g. proper indices already installed)

Limitations

- OtterTune only considers global knobs.

- It also ignores the cost of restarting the DBMS when suggesting configurations.

Problems deferred as future work....

- Automatically identify knobs that require DBMS restarting.
- Taking into consideration the cost of restarting when recommending configurations.
- Automatically determining if certain knobs can cause application to lose data.
- Consider tuning table or component-specific knobs.

Summary

Contributions of the paper

OtterTune:

- Can find good configurations for a much larger number of knobs than previous automatic database tuning system.
- Can also identify dependencies between knobs.
- Generates configurations much faster than previous systems.
- Leverages machine learning techniques and data from past configurations.

Criticism (my opinion)

- Details are not very well explained.
- OtterTune still needs significant input from the DBA.
- Approach is overly complicated and has a lot of limitations.
- Not being able to determine which knobs can cause data loss is dangerous.

References

[1] Van Aken, Dana, et al. "Automatic Database Management System Tuning Through Large-scale Machine Learning." *Proceedings of the 2017 ACM International Conference on Management of Data*. ACM, 2017.

[2] Duan, Songyun, Vamsidhar Thummala, and Shivnath Babu. "Tuning database configuration parameters with iTuned." *Proceedings of the VLDB Endowment* 2.1 (2009): 1246-1257.

Thank you!
Questions?