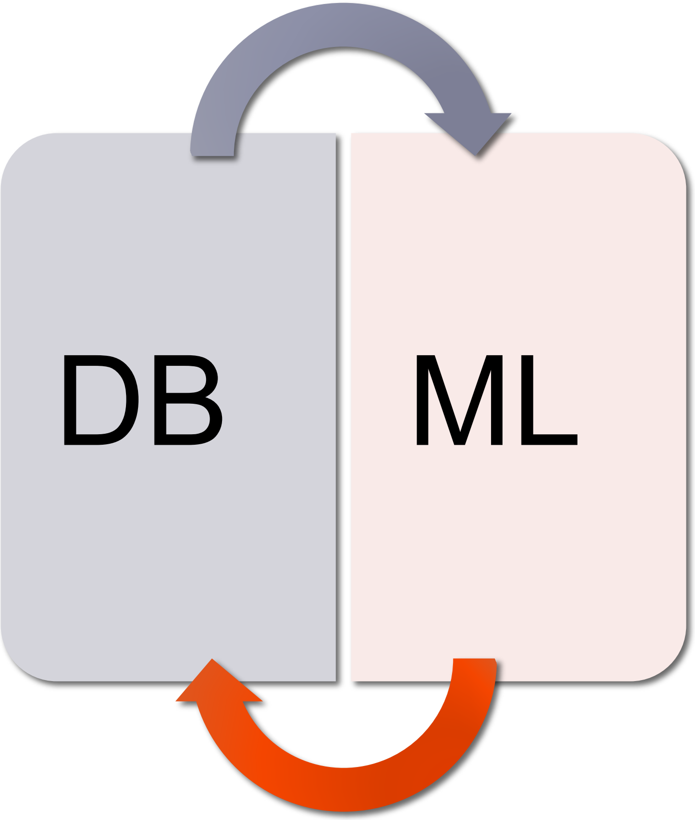
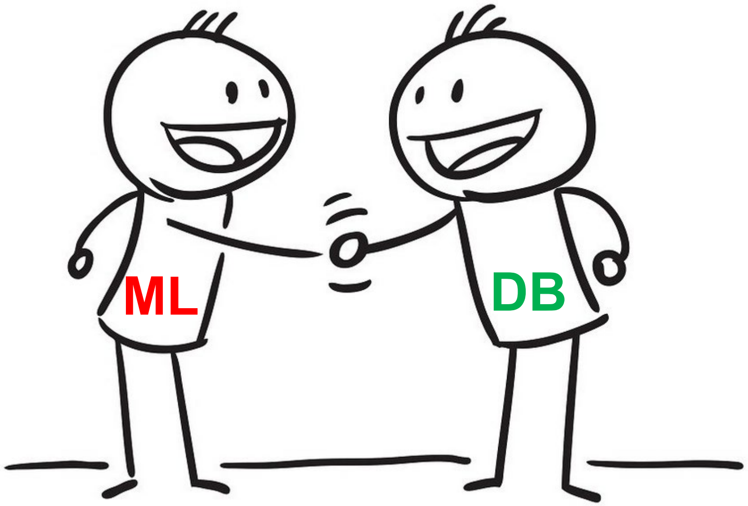

DB
ML

Interplay

AI/ML for DB - (our focus)

- Self Optimization
- Self Configuration
- Self Monitoring
- Self Healing
- Self Security
- Self Design



DB for AI/ML

- Declarative AI, AI Optimization, Data Governance, Model Mgmt. etc.

AI/ML for DB

- AI advised optimizations like **knob tuning**, index advisor, view advisor, buffer tuning, logical design tuning, metadata statistics, data partitioning, ...

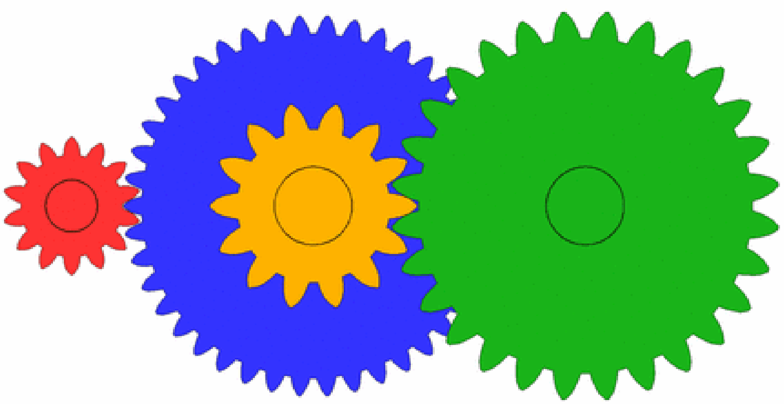
- AI assisted online processes like workload scheduling, fault diagnosis, self healing, query tuning, ...

- AI enhanced core components like building learned indexes, learned cost estimator, learning based join-order selection, query engine customization, …

AI/ML for DB (contd.)

- AI for assembling various alternatives for an operation
  - E.g.: creating learned ensemble of cost based, rule based and learning based optimizer.

- AI designed DB, i.e. self designing
  - Data structures design , transaction design, storage design, …



DBMS TUNING

DBMSs are complex systems with many tunable options (knobs) that control nearly all aspects of their runtime operation.

Optimal Knob-Configuration
= f(hardware, software-implementation, query-workload)

PostgreSQL Configuration “Knobs”

- File Locations (data dir, auth-file, …)
- Connections and Authentication
- Resource Usage
- Write Ahead Log
- Query Tuning
- Lock Management, Error Reporting, …

Some knobs are useless

PostgreSQL Configuration “Knobs”

- File Locations
- Connections and Authentication (max_connections, auth timeout, …)
- Resource Usage
- Write Ahead Log
- Query Tuning
- Lock Management, Error Reporting, …

Increasing max_connections costs ~400 bytes of shared memory per connection slot, plus lock space

PostgreSQL Configuration “Knobs”

- File Locations
- Connections and Authentication
- Resource Usage
  - Memory (shared buffers, temp buffers, work mem, …)
  - Disk (temp file limit)
  - Kernel Resource Usage (max files per process, …)
  - Background Writer (bgwriter delay, bgwriter lru maxpages, …)
  - Asynchronous Behavior (effective io concurrency, max worker processes)
- Write Ahead Log
- Query Tuning
- Lock Management, Error Reporting, …

PostgreSQL Configuration “Knobs”

- File Locations
- Connections and Authentication
- Resource Usage
- Write Ahead Log
  - Settings (buffers, level, commit delay, …)
  - Checkpoints (timeout, warning, …)
- Query Tuning
- Lock Management, Error Reporting, …

PostgreSQL Configuration “Knobs”

- File Locations
- Connections and Authentication
- Resource Usage
- Write Ahead Log
- Query Tuning
  - Planner Method Configuration
  - Planner cost constants
  - Genetic Query Optimizer, …
- Lock Management, Error Reporting, …

PostgreSQL Configuration “Knobs”

- File Locations
- Connections and Authentication
- Resource Usage
- Write Ahead Log
- Query Tuning
- Lock Management, Error Reporting …

What is currently done in practice?

- Hire expensive experts to configure the knobs for the expected workload manually.
  - Personnel is estimated to be ~50% of the total ownership cost of a large-scale DBMS!
  - Many DBAs spend nearly 25% of their time on tuning!

EDB POSTGRES

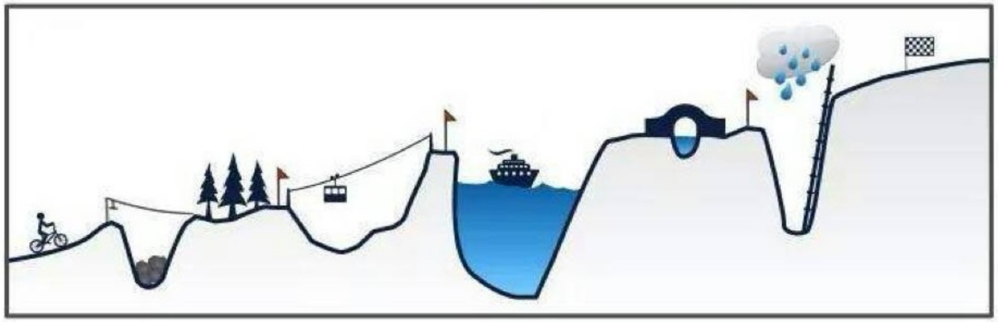
40% of engagement requests are for tuning and knob configurations issues.

- Many automated tools shortcomings:
  - Engine-specific
  - Too much human intervention
  - No knowledge transfer from one deployment to the other

Performance optimization

- *target objective: throughput, latency, etc.*

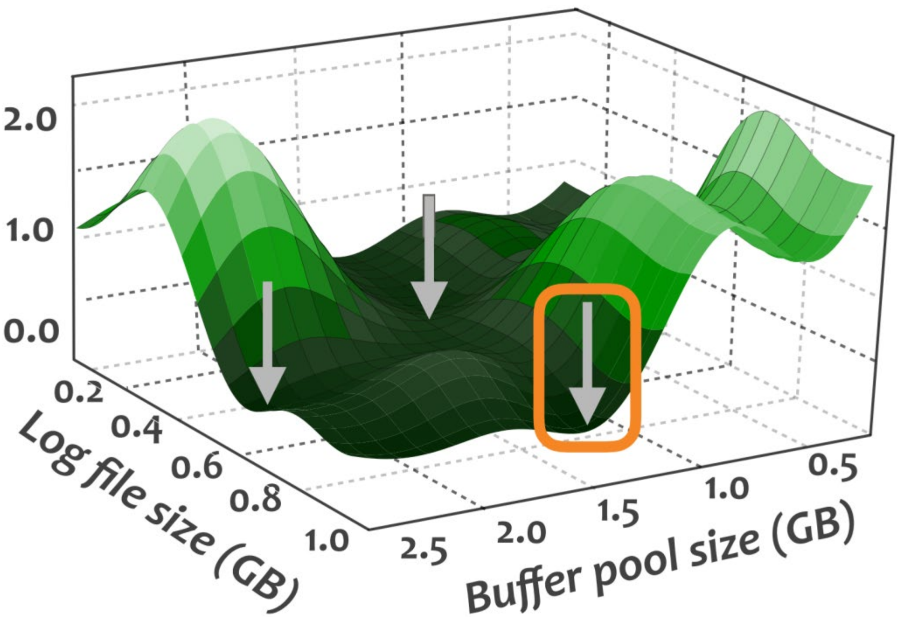
*Tuning even one DBMS deployment is **HARD**.*



Finding optimal configuration is NP-Hard!

#1: Dependencies

99^{th} %-tile latency (sec)
lower is better

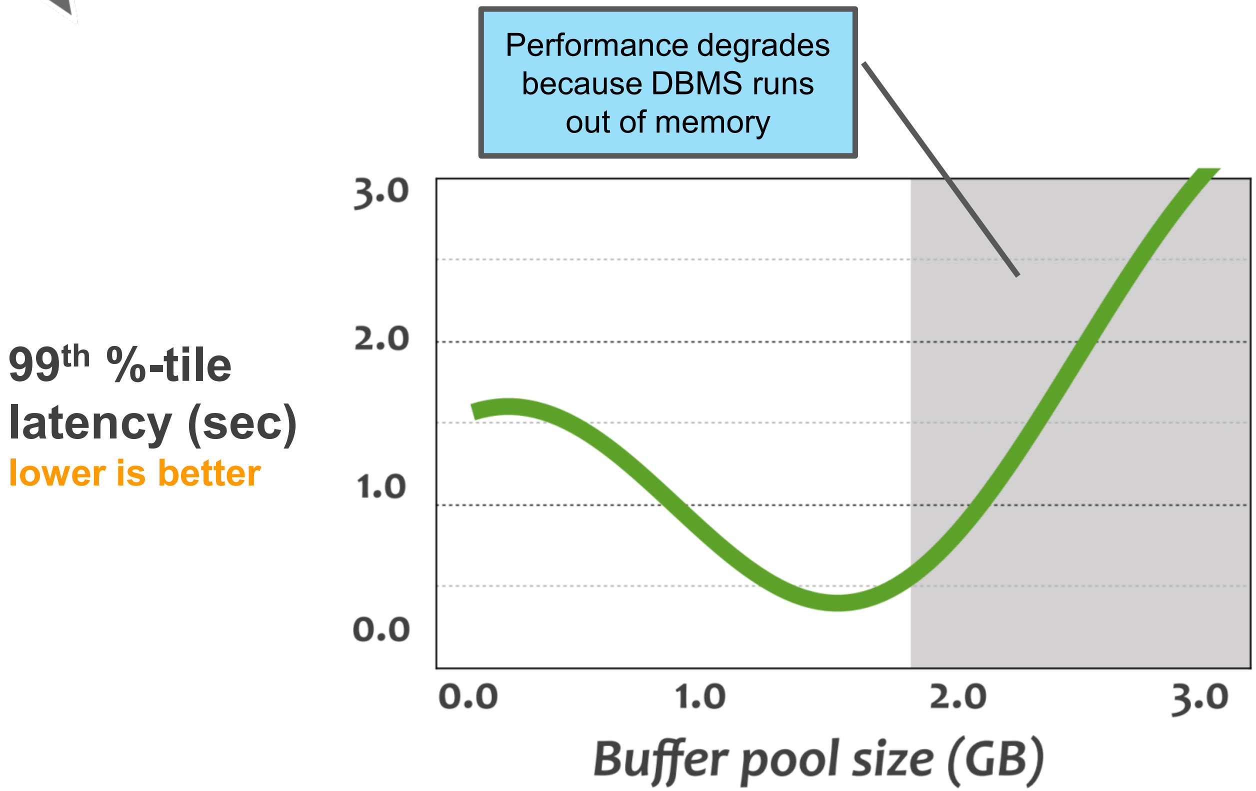


- MySQL (v5.6)   - YCSB* Workload A   - VM: 2 GB RAM, 2 vCPUs

* Yahoo! Cloud Service Benchmark: consists of 6 different workloads.
  - Workload A (Update Heavy) has a mix of 50/50 reads and writes

THE CHALLENGE

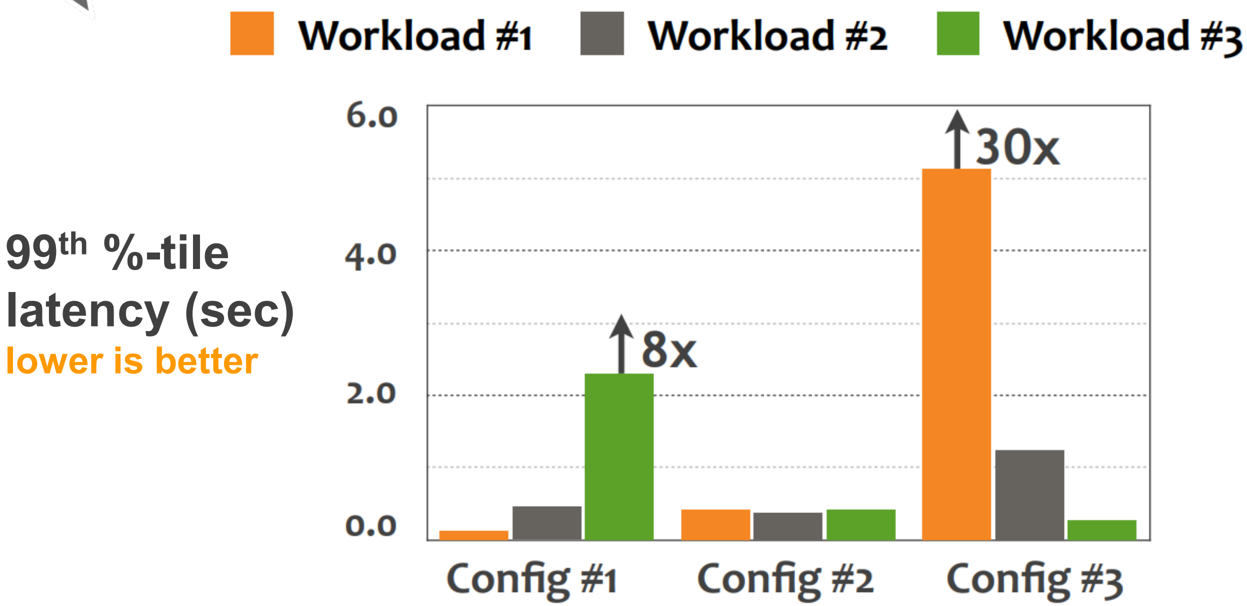
#2: Continuous Settings



- MySQL (v5.6) - YCSB Workload A - VM: 2 GB RAM, 2 vCPUs

THE CHALLENGE

#3: Non-Reusability

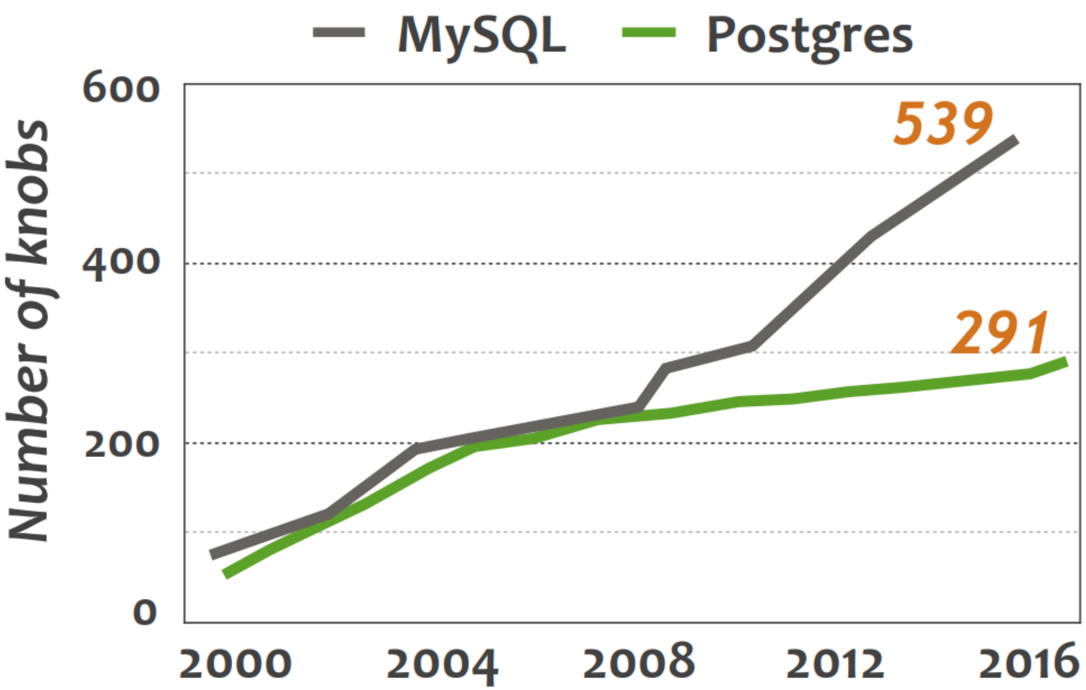


- MySQL (v5.6) - YCSB Workloads (3 different)

Optimal configuration is different for every workload.

THE CHALLENGE

#4: Tuning Complexity



MySQL ↑7x

PostgreSQL ↑5x

Number of configuration knobs in MySQL and Postgres releases
(16 years)

Summary so far…

- Database systems have *numerous configuration knobs*.
- *Tuning knobs is critical* for performance.
- Performance is measured in terms of a *target objective*.
  - Latency, throughput
- Choosing knob configuration depends on *hardware, software implementation* and *query workload*.
- The *complexity* of knobs and *interdependence* between them make the optimization problem challenging.

This paper…

AUTOMATIC TUNING THROUGH MACHINE LEARNING

SIGMOD 2017, VLDB 2018 (demo)

Goal:
Reuse historical performance data from tuning "past" DBMS deployments to tune "new" DBMS deployments.

OtterTune

Key Assumptions

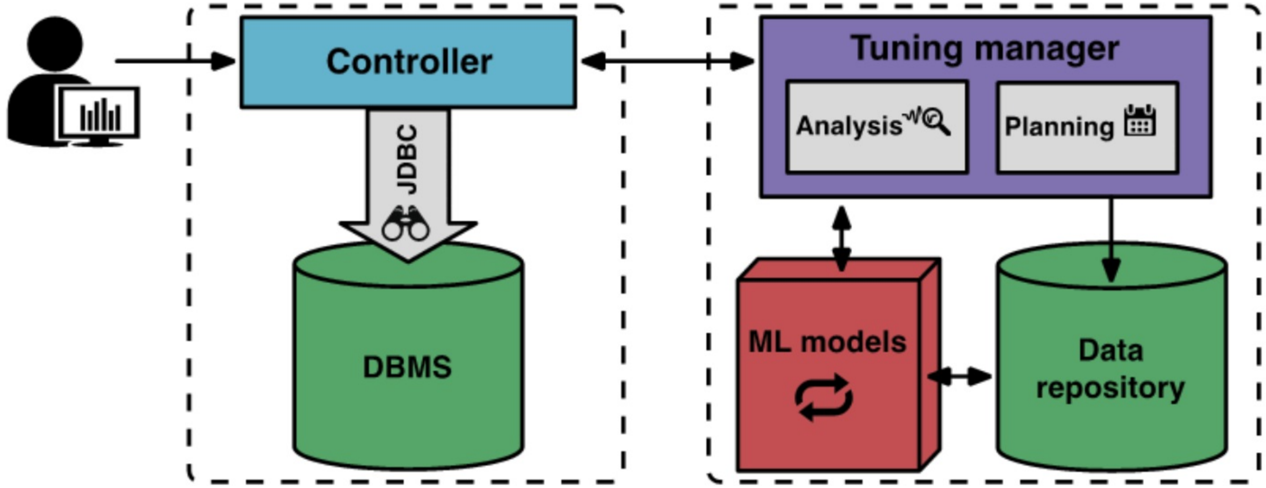
- The physical design (indexes, views) of the database is assumed to be reasonably good.
- Many knobs require DBMS restart after alternation. DBMS restart cost is neglected.

Key Contributions

Models designed for:

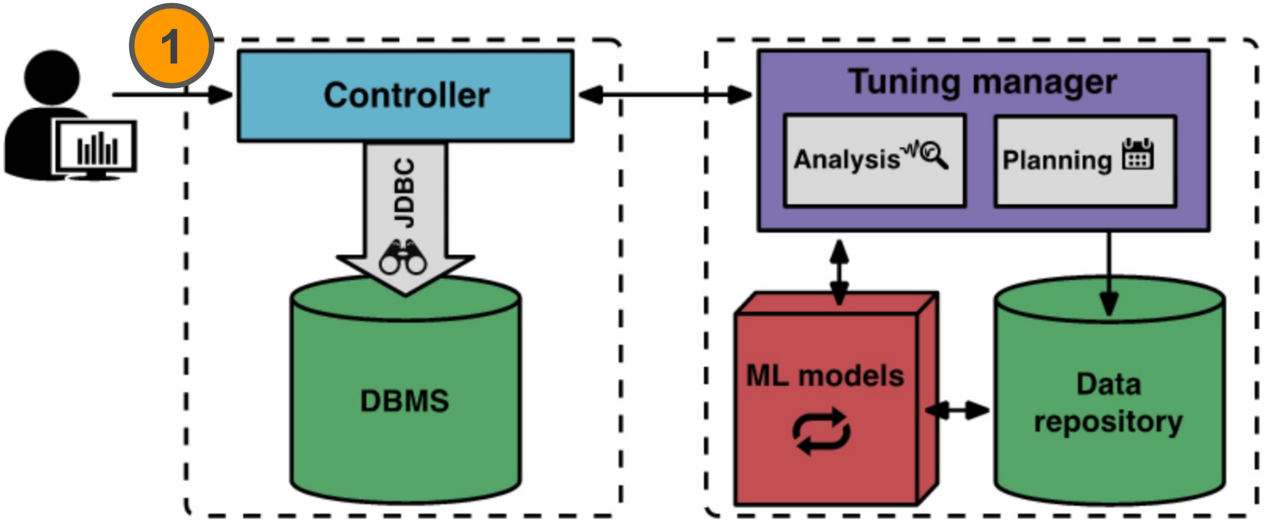
- Identifying most impactful knobs.
- Workload Mapping: Map unseen database workloads to previous workloads for helping knowledge transfer.
- Recommend knob configuration for target objective.

System Overview



- *Controller* interacts with the DBMS to collect runtime information, install new configuration and collect performance measurements.
- *Tuning Manager*
    - stores the above information in a repository. This is further used by background processes for constructing/refining the models.
    - Using the models, the next configuration is recommended. Each recommendation provides more information in a feedback loop.

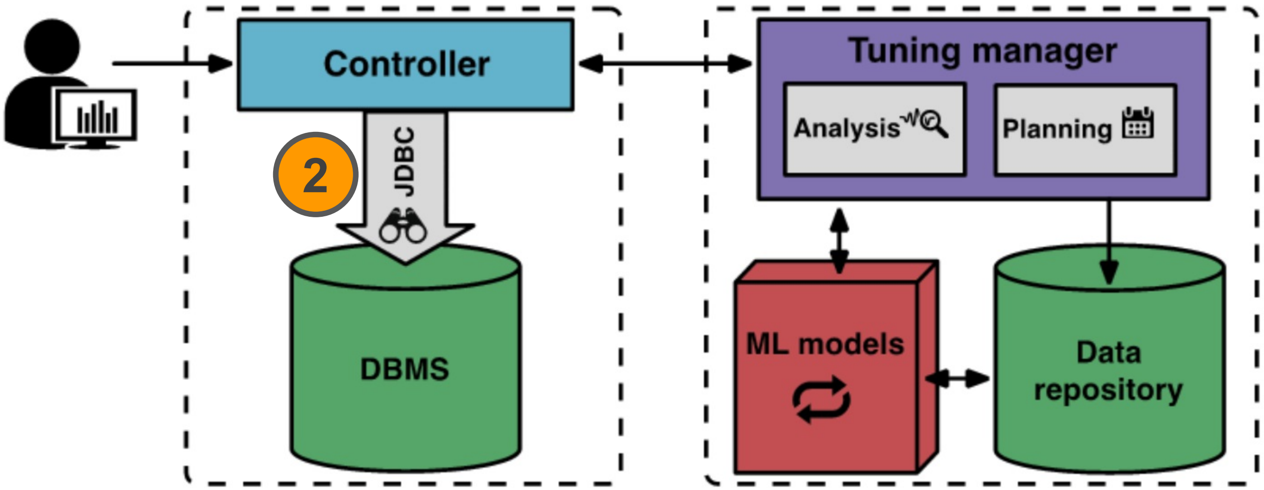
System Overview



1 At the start of a tuning session,

User specifies the *target objective*
– which "metric" to optimize?

System Overview

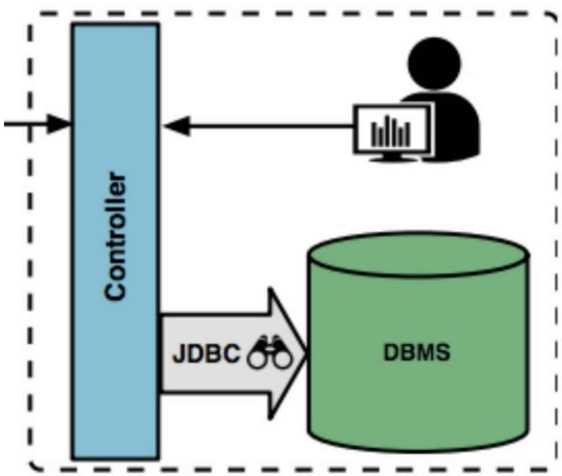


2 **Controller** connects to the target DBMS and collect hardware profile and current knob configuration.

It then starts the *observation period.*

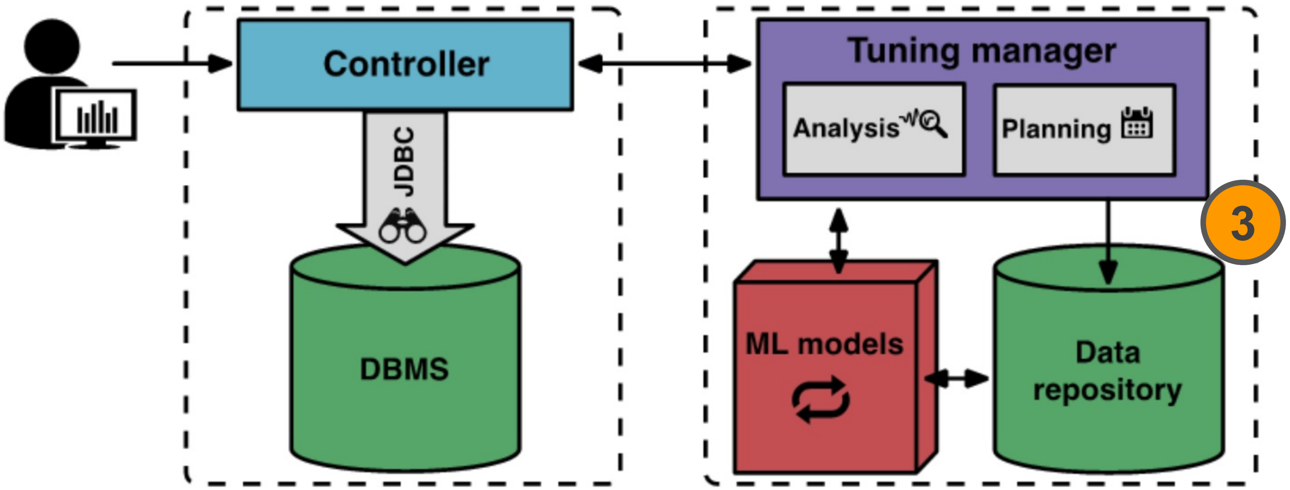
Observation Period

- Execute either (specified by the DBA)
  - a set of queries for a fixed time
    - fixed observation period, suitable for OLTP.
  - a specified workload trace
    - variable observation period, suitable for OLAP.
- Observe DBMS & measure target metric.



- At the end of the observation period collect the additional DBMS-specific internal metrics.
  - E.g.: counter of pages written to/read from the disk

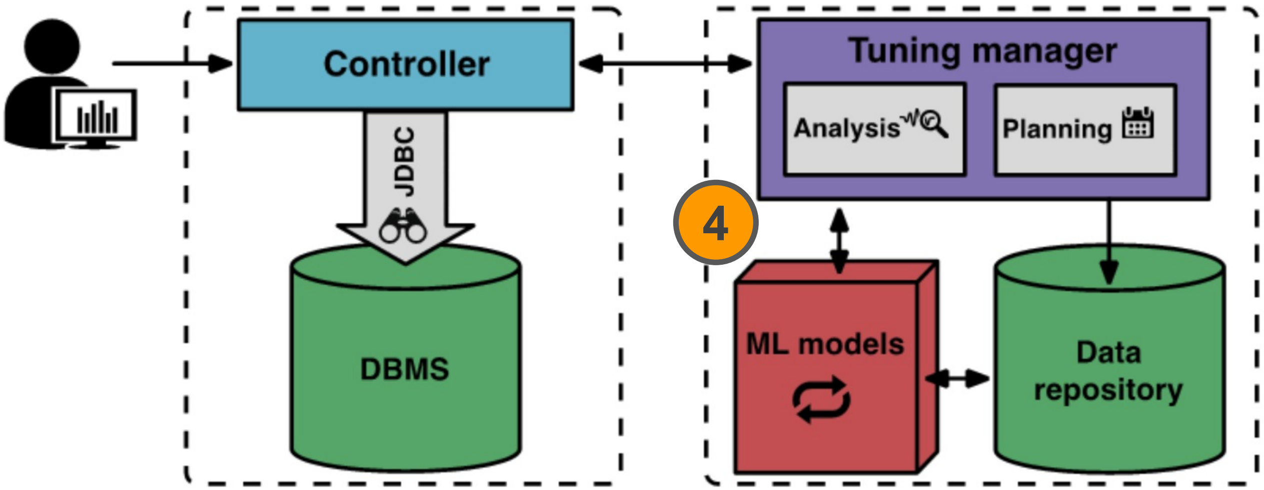
System Overview



3 **Tuning manager** receives information from controller and stores it in a *repository*.

Repository has data organized per hardware profile and major DBMS versions.

System Overview

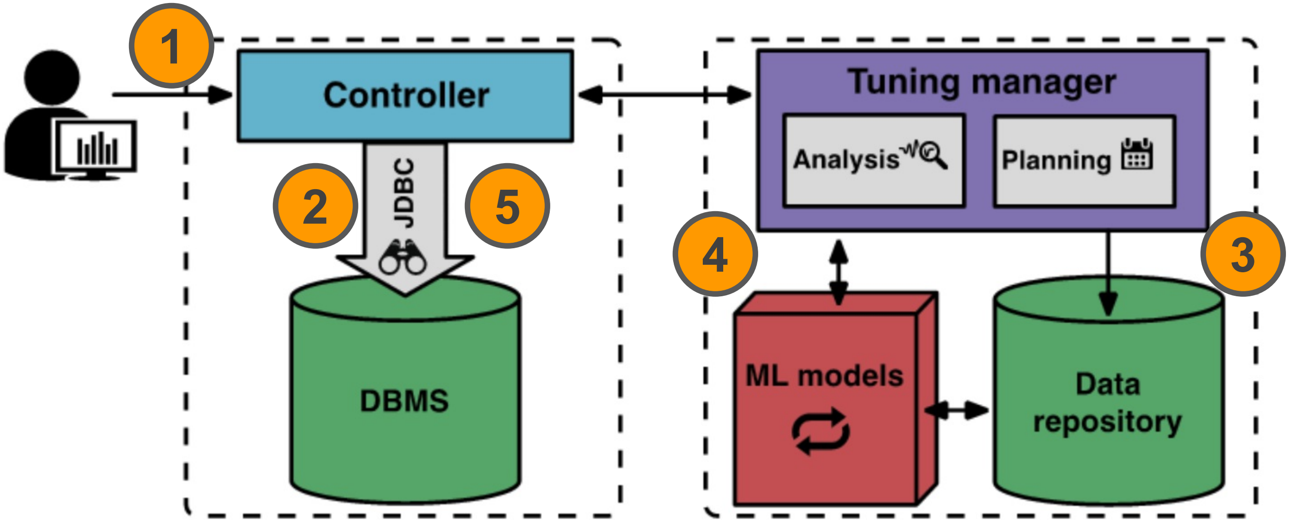


4 **Tuning Manager** recommends next configuration using background process that continuously analyze data and refine internal *ML models*.

ML models allow to

- understand target workload and map it to a workload for same DBMS and hardware profile that it has seen (and tuned).
- recommend knob configuration that is designed to improve objective for current workload, DBMS and hardware.

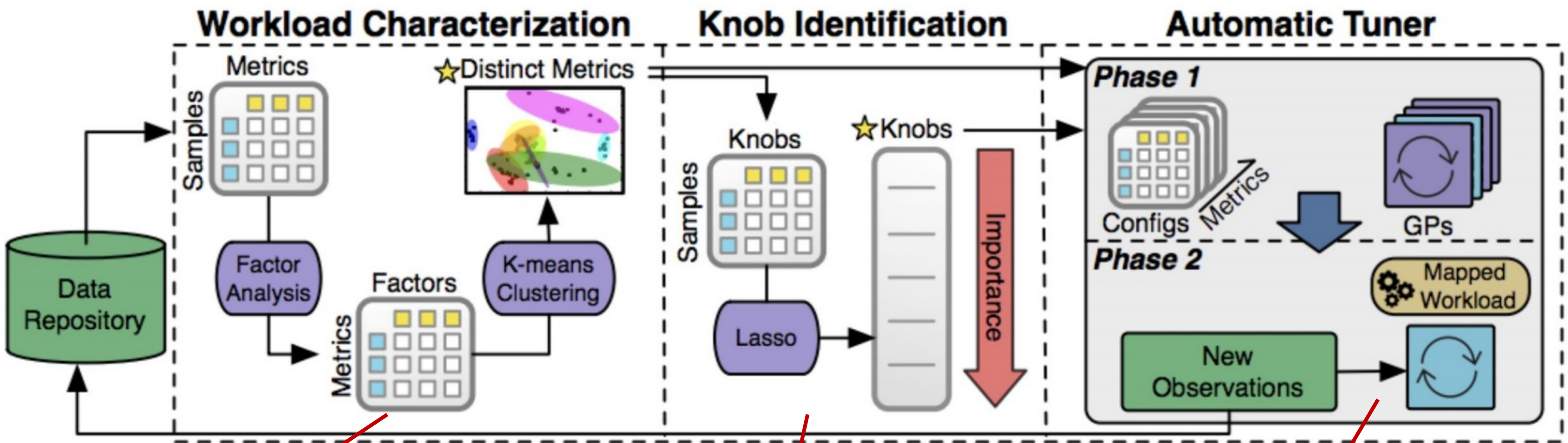
System Overview



Termination

- OtterTune provides an estimate of how much better recommended configuration is compared to the best configuration that it has seen so far.
- If controller decides to use the recommendation, then the suggested configuration is installed, and measurements are collected.
- Tuning continues until user is satisfied with improvement.

Tuning System



Compute non-redundant metrics that can help to characterize the workload

Identify the knobs having the strongest impact on the objective function

Based on the collected data, recommend the next configuration

WORKLOAD CHARACTERIZATION

Construct smallest set of metrics that capture the variability in performance and distinguishing characteristics for different workloads.

DBMS-Internal Metrics

```
mysql> SHOW GLOBAL STATUS;
+-------------------------------------+-----------+
| METRIC_NAME                         | VALUE     |
+-------------------------------------+-----------+
| ABORTED_CLIENTS                     | 0         |
| ABORTED_CONNECTS                    | 0         |
...
| INNODB_BUFFER_POOL_BYTES_DATA       | 129499136 |
| INNODB_BUFFER_POOL_BYTES_DIRTY      | 76070912  |
| INNODB_BUFFER_POOL_PAGES_DATA       | 7904      |
| INNODB_BUFFER_POOL_PAGES_DIRTY      | 4643      |
| INNODB_BUFFER_POOL_PAGES_FLUSHED    | 25246     |
| INNODB_BUFFER_POOL_PAGES_FREE       | 0         |
| INNODB_BUFFER_POOL_PAGES_MISC       | 288       |
| INNODB_BUFFER_POOL_PAGES_TOTAL      | 8192      |
| INNODB_BUFFER_POOL_READS            | 15327     |
| INNODB_BUFFER_POOL_READ_AHEAD       | 0         |
| INNODB_BUFFER_POOL_READ_AHEAD_EVICT | 0         |
| INNODB_BUFFER_POOL_READ_AHEAD_RND   | 0         |
| INNODB_BUFFER_POOL_READ_REQUESTS    | 2604302   |
| INNODB_BUFFER_POOL_WAIT_FREE        | 0         |
| INNODB_BUFFER_POOL_WRITE_REQUESTS   | 562763    |
| INNODB_DATA_FSYNCS                  | 2836      |
| INNODB_DATA_PENDING_FSYNCS          | 1         |
| INNODB_DATA_WRITES                  | 28026     |
...
| UPTIME                              | 5996      |
| UPTIME_SINCE_FLUSH_STATUS           | 5996      |
+-------------------------------------+-----------+
```

- Directly affected by the knobs' settings

Buffer pool size is too small:

$$\frac{\text{\#buffer pool misses}}{\text{total \#buffer pool requests}}$$

- Problem: *Redundancy*
  - Same but different units
  - Highly correlated
- Solution: *Prune* them!

Prune Redundant Metrics

For each hardware profile and DBMS version, a set of non-redundant metrics have to be identified.

Factor Analysis

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

K-means Clustering

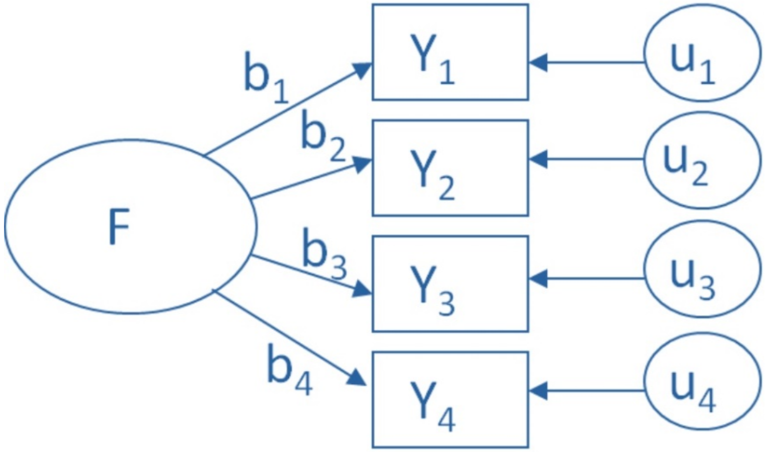
- Find groups of metrics similar to each other.
- Select one metric from each group.

Factor Analysis

Given: A set of real-valued variables that contain arbitrary correlations.

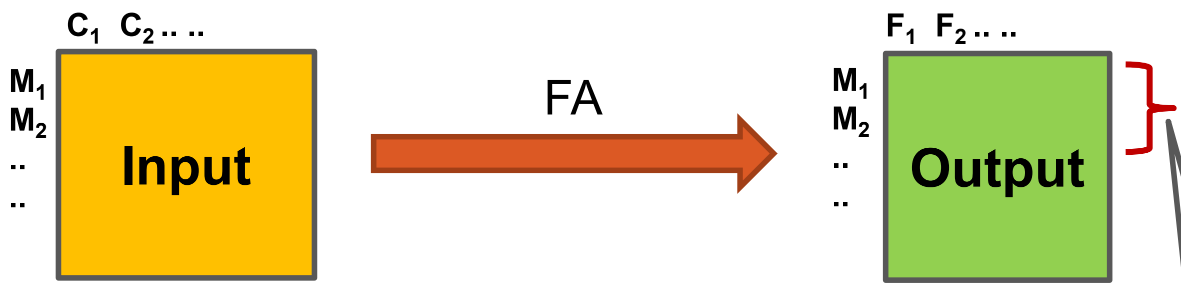
FA aims to find a smaller set of latent factors that explain (underlie) the observed variables.

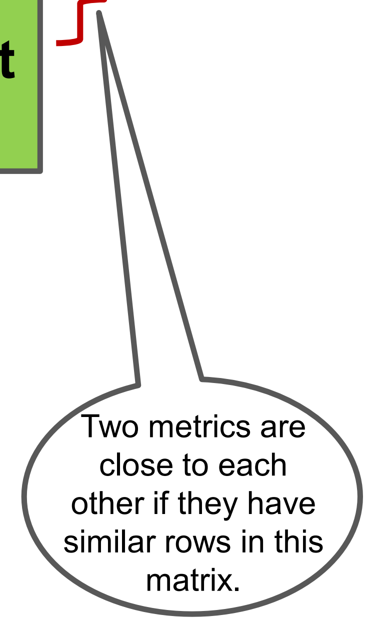
- These factors capture the correlation patterns of the original variables.



$$Y_1 = b_1{*}F + u_1$$
$$Y_2 = b_2{*}F + u_2$$
$$Y_3 = b_3{*}F + u_3$$
$$Y_4 = b_4{*}F + u_4$$

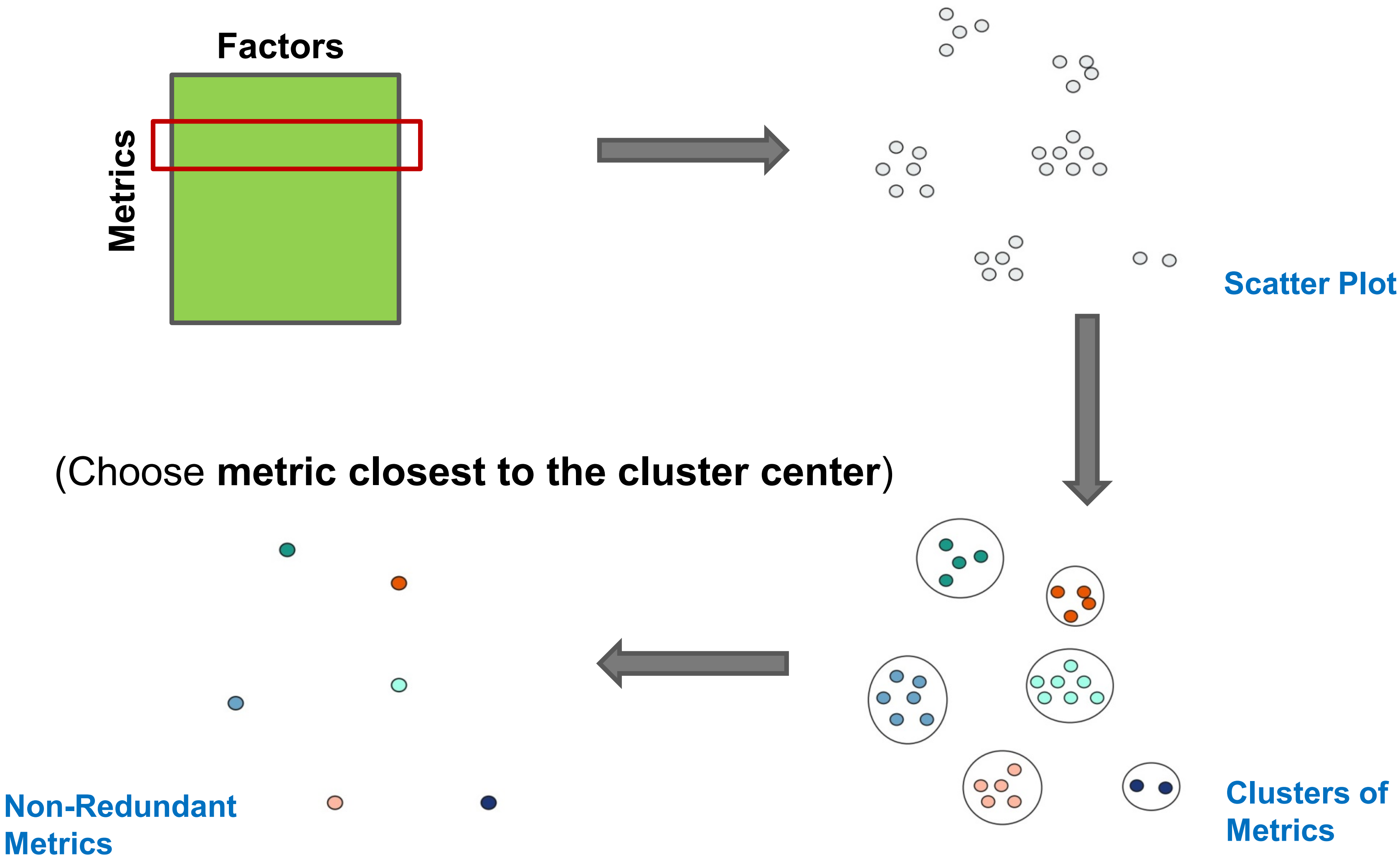
Factor Analysis (Contd.)





- Factors are ordered by the amount of variability in the original data.
- Most of the variability is captured by first few factors.
- From the output, closely correlated metrics can be identified and pruned.

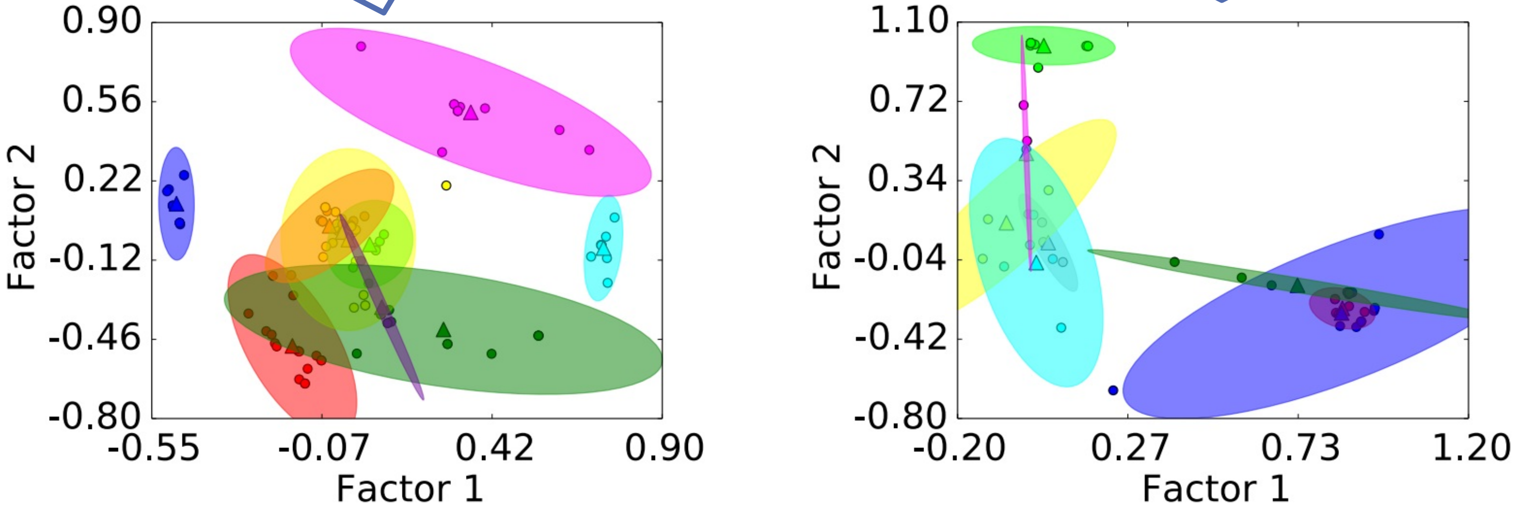
K-means Clustering



2-D Projection of the Scatter Plot

131 metrics
9 clusters!

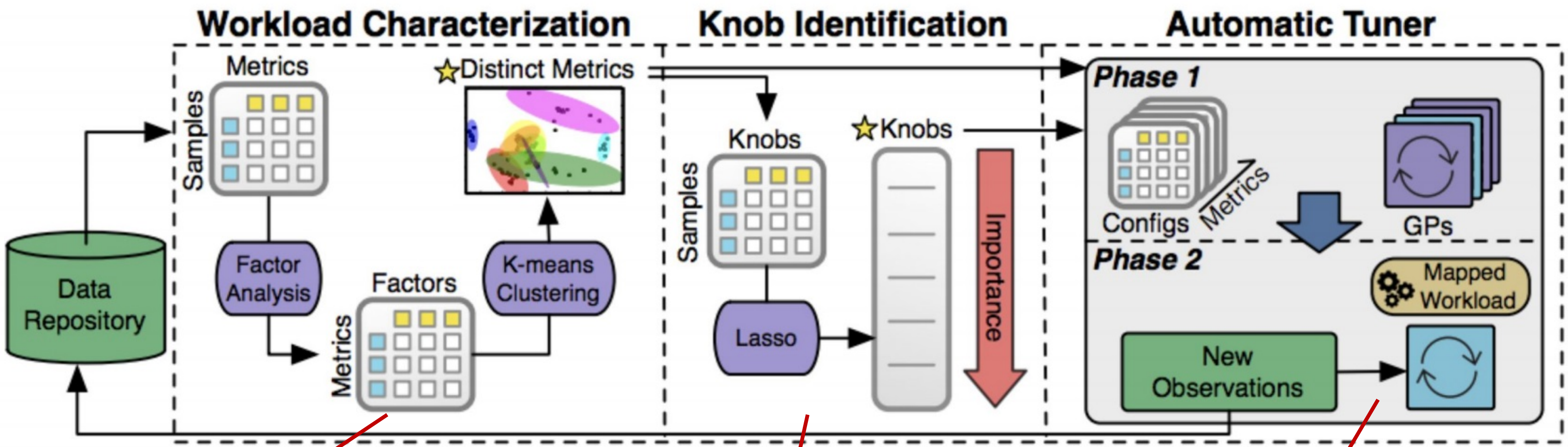
57 metrics
8 clusters!



(a) MySQL (v5.6) **(b)** Postgres (v9.3)

Each cluster corresponded to a distinct aspect of performance

Tuning System



Compute non-redundant metrics that can help to characterize the workload

Identify the knobs having the strongest impact on the objective function

Based on the collected data, recommend the next configuration

IMPORTANT KNOBS IDENTIFICATION

Given various configurations and the corresponding metric value, identify knobs that influence the metric the most

Configuration Knobs

```
mysql> SHOW GLOBAL VARIABLES;
+-----------------------------------+-----------------+
| KNOB_NAME                         | KNOB_VALUE      |
+-----------------------------------+-----------------+
| AUTOCOMMIT                        | ON              |
| AUTOMATIC_SP_PRIVILEGES           | ON              |
...
| INNODB_BUFFER_POOL_SIZE           | 134217728       |
| INNODB_CHANGE_BUFFERING           | all             |
| INNODB_FLUSH_LOG_AT_TRX_COMMIT    | 1               |
| INNODB_FLUSH_METHOD               |                 |
| INNODB_FORCE_LOAD_CORRUPTED       | OFF             |
| INNODB_FORCE_RECOVERY             | 0               |
| INNODB_IO_CAPACITY                | 200             |
| INNODB_LARGE_PREFIX               | OFF             |
| INNODB_LOCKS_UNSAFE_FOR_BINLOG    | OFF             |
| INNODB_LOCK_WAIT_TIMEOUT          | 50              |
| INNODB_LOG_BUFFER_SIZE            | 8388608         |
| INNODB_LOG_FILES_IN_GROUP         | 2               |
| INNODB_LOG_FILE_SIZE              | 5242880         |
...
| SORT_BUFFER_SIZE                  | 2097152         |
| SQL_AUTO_IS_NULL                  | OFF             |
...
```

- Knobs have varying degrees of impact on the performance
  - Some have high impact
  - Some have no impact
  - For many, it depends on the workload
- Problem: Which knob matters?
- Solution: *Feature Selection*

Least Absolute Shrinkage and Selection Operator (LASSO) Regression

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

$$\min\left(||Y - X\theta||_2^2 + \lambda||\theta||_1\right)$$

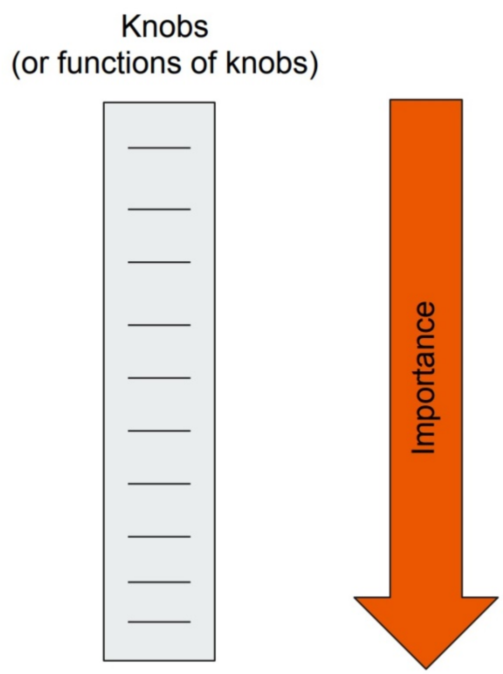
Y = vector of metrics
X = vector of knobs
θ = weights for different knobs
λ = regularization parameter (penalty)

Feature Selection with LASSO

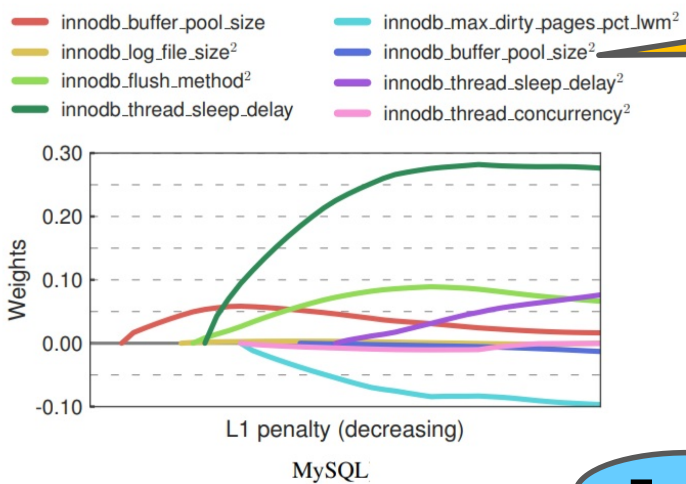
Aim: find relationship between knobs (or polynomial functions of knobs) and metrics.

Feature Selection:

- Start by adding high penalty thereby removing all knobs (weights shrink to zero).
- Decrease penalty in small increments, recompute regression and track what features are added.
- **Order knobs** by order of appearance.
- How many knobs to choose?
  - **Incremental approach**: Dynamically increase the number of knobs used in a tuning session over time.

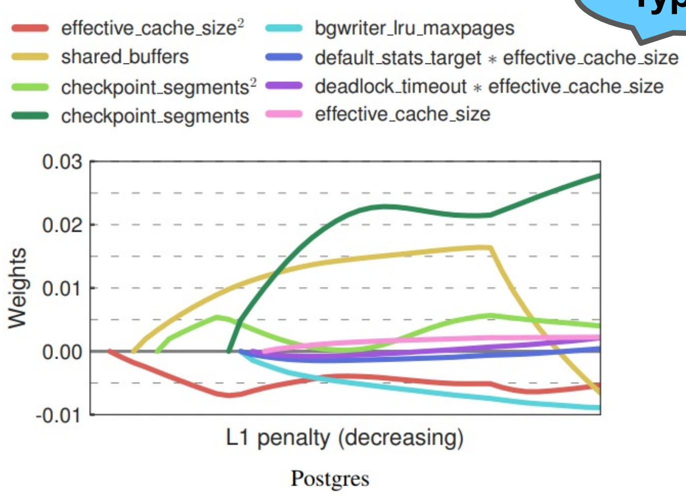


Identifying Important Knobs



MySQL

Type 1

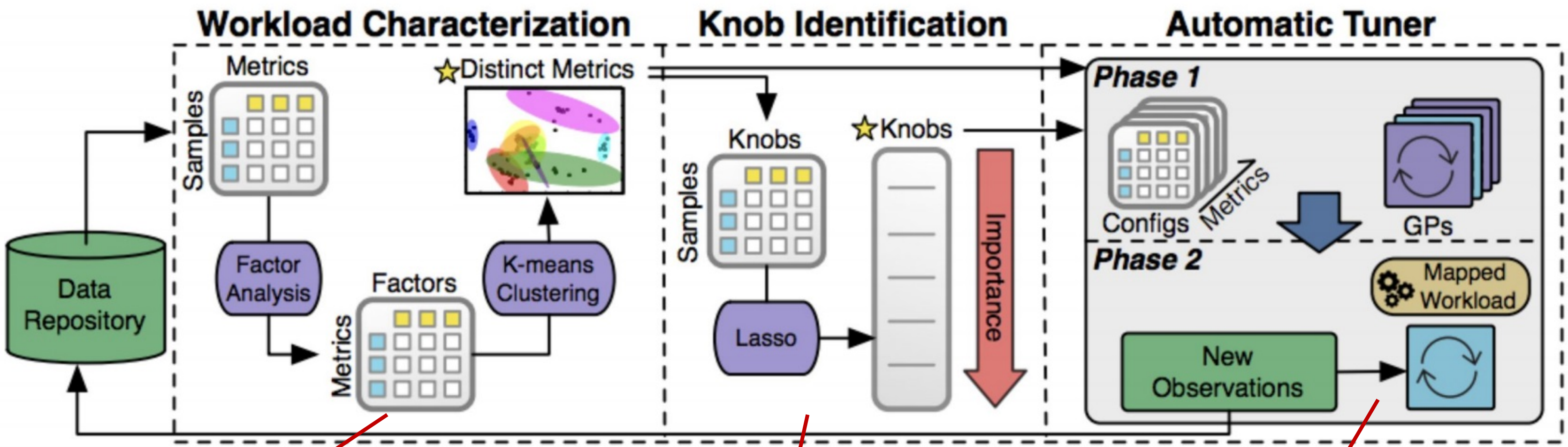


Postgres

- Lasso paths for 99th %-tile latency.
- Eight most impactful features.
- Second degree polynomial features – 2 types.

1. **Product of two knobs**
   - Useful for detecting pairs of knobs that are non-independent.
2. **Product of single knob**
   - Reveals quadratic relationship between a knob and a target

Tuning System



Compute non-redundant metrics that can help to characterize the workload

Identify the knobs having the strongest impact on the objective function

Based on the collected data, recommend the next configuration

AUTOMATIC TUNING

Input: Set of non-redundant metrics, impactful knobs and data from the previous tuning sessions

Output: Configuration recommendation having the best expected performance improvement.

Automatic Tuning

Workload Mapping

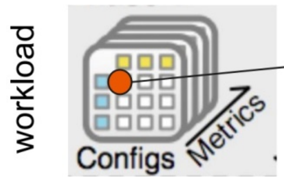
Find workload in the data repository similar to the target workload.

↓

Configuration Recommendation

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

Workload Mapping

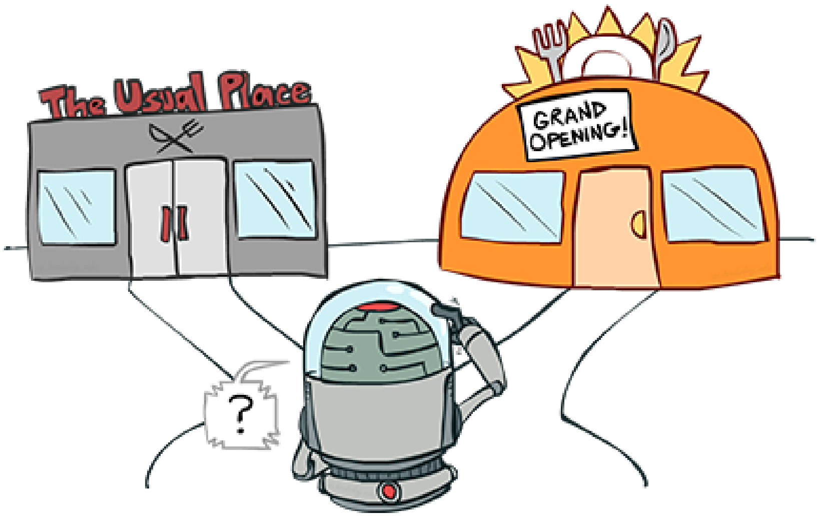


X_{mij} = value of metric m when executing workload i for configuration j

- For each “seen” workload w, compute a *score*
  - For each metric, compute **Euclidean distance** between target workload and w.
  - Compute score for w by averaging distance over all possible metrics.
- Select workload with lowest score.
- Dynamic mapping used
  - With each iteration quality of match increases with the amount of data gathered

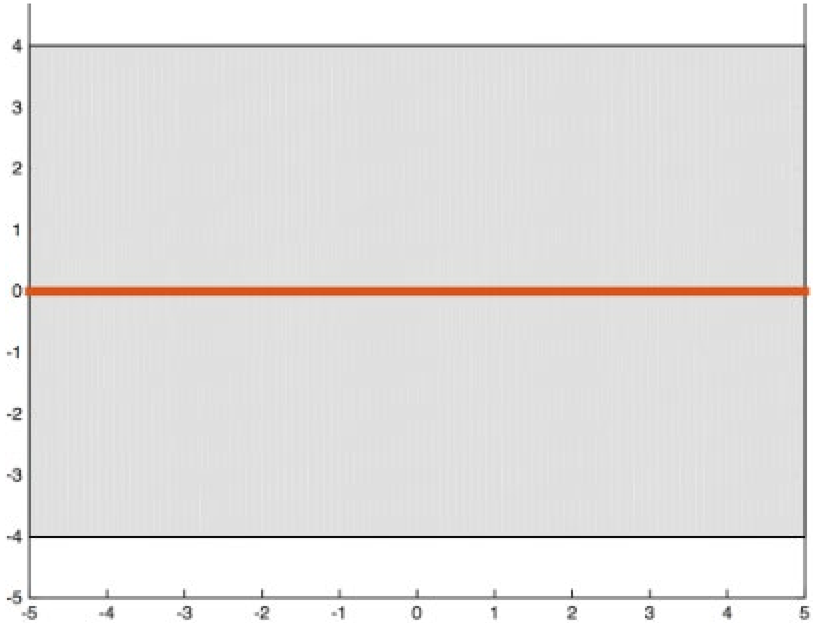
Configuration Recommendation

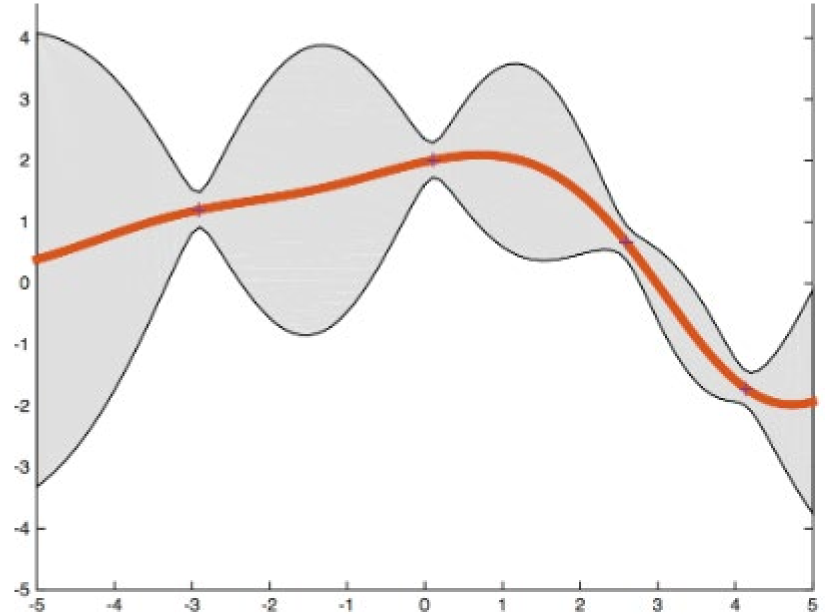
- OtterTune uses Gaussian Process (GP) Regression
- For each observation period, OtterTune tries to find a better configuration than the best seen configuration
- This is done by either
  - Exploration: searching unknown regions
  - Exploitation: searching near best configuration seen so far



GP Regression

- Assumption
  - P(y|D,x) ~ N(μ, ∑)
  - The target objective value for neighbouring points are similar.
- Red line: prediction of the target objective value





GP Regression Contd.

How to decide which config to recommend?

- Depends on variance of the data points in its GP model
- Configuration with the greatest expected improvement in the objective

Say function f captures the greatest expected improvement in terms of the mean and variance in GP model

- Expected improvement is near 0 at sampled points and higher between them

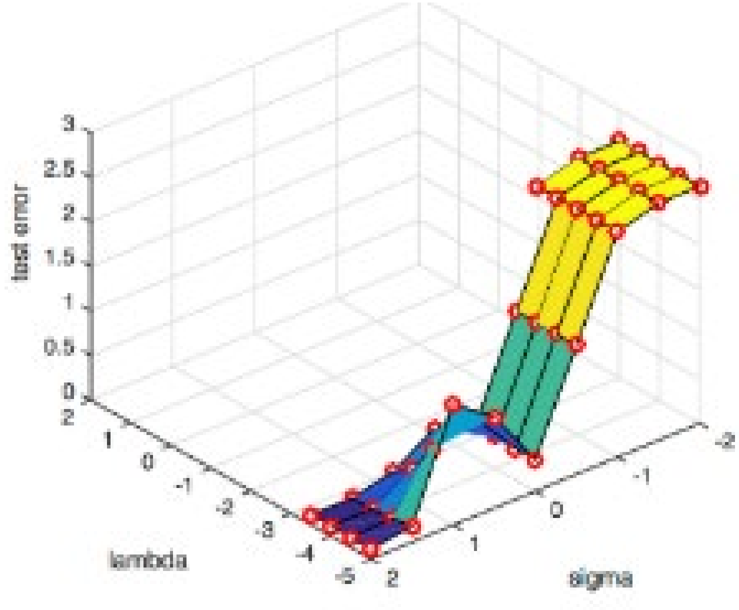
f is optimized using gradient descent

- initial set comprising of top performing configurations and random configurations in 1:10 ratio
- OtterTune selects from the optimized configurations the one that maximizes the potential improvement to run next

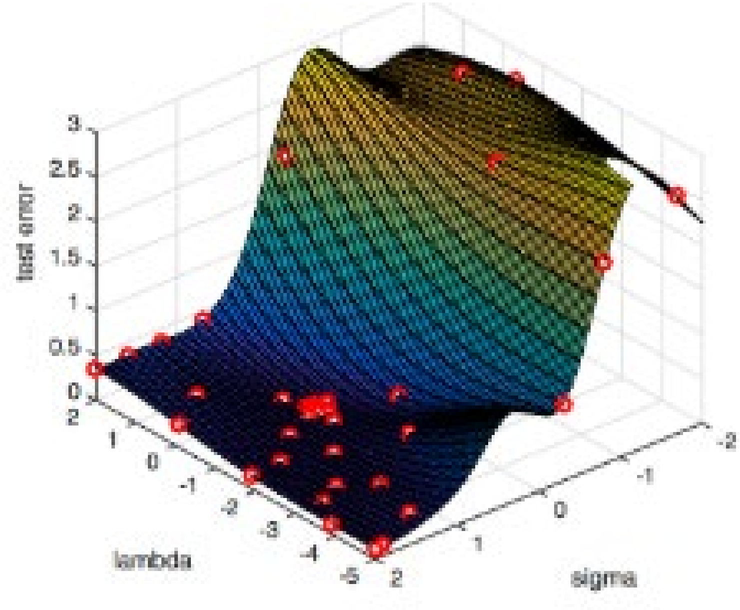
GP Regression Contd.

GP Regression is preferred because:

- Theoretically justifies way to tradeoff between exploration vs exploitation
- Provides confidence intervals
- Quick convergence



Grid Search

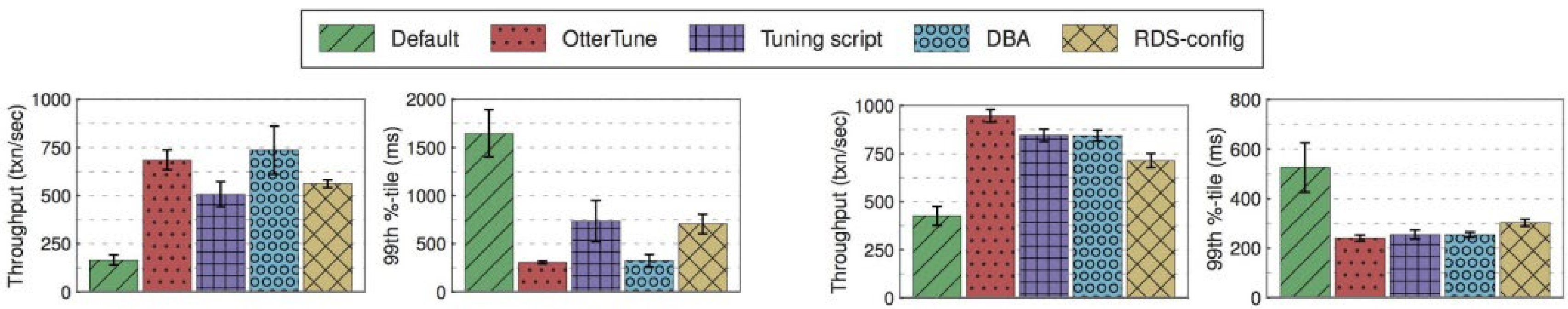


GP Regression

Experimental Evaluation

Efficacy Evaluation

- Default: The configuration provided by the DBMS
- Tuning script: The configuration generated by an open source tuning advisor tool
- DBA: The configuration chosen by a human DBA
- RDS: The configuration customized for the DBMS that is managed by Amazon RDS

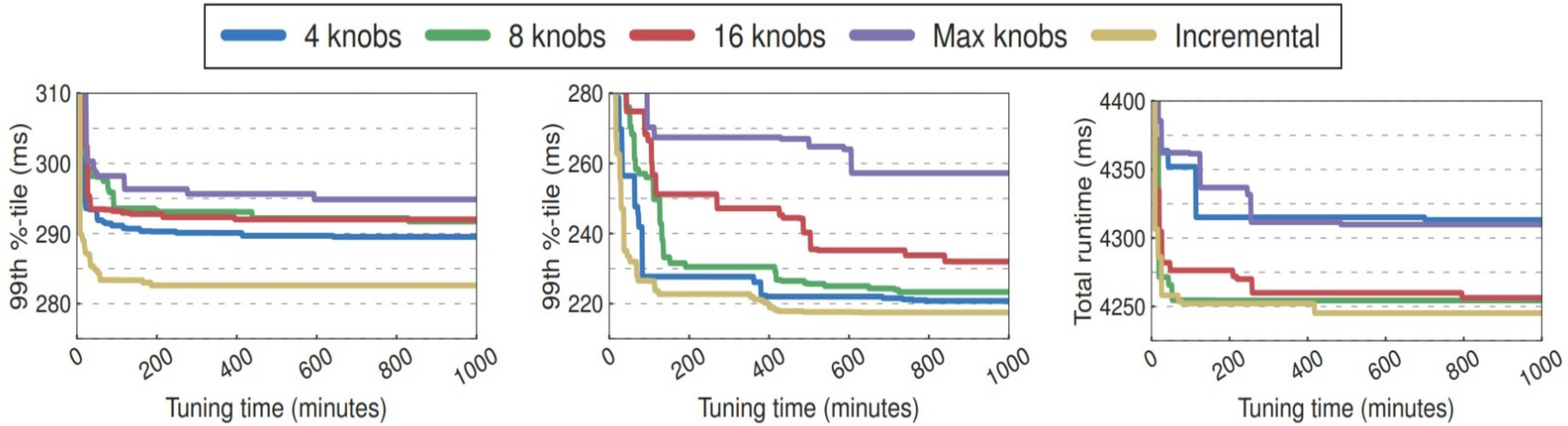


(a) TPC-C (Throughput) (b) TPC-C (99%-tile Latency) (a) TPC-C (Throughput) (b) TPC-C (99%-tile Latency)

MySQL **Postgres**

Experimental Evaluation

- Influence of the *number of knobs* used in the performance.
  - The *incremental* approach works best for all DBMSs.



(a) MySQL (TPC-C) **(b)** Postgres (TPC-C) **(c)** Vector (TPC-H)

The End

Some of the content has been sourced from the following:

(a) blogs:

i. https://blog.acolyer.org/2017/08/11/automatic-database-management-system-tuning-through-large-scale-machine-learning/

ii. https://aws.amazon.com/blogs/machine-learning/tuning-your-dbms-automatically-with-machine-learning/

(b) presentation:

i. https://www.percona.com/live/e17/sites/default/files/slides/Automatic%20Database%20Management%20System%20Tuning%20Through%20Large-Scale%20Machine%20Learning%20-%20FileId%20-%20118513.pdf

ii. https://pdfs.semanticscholar.org/1f1f/47da8fff8da53589d7eab36d6bae32b2c3d2.pdf

iii. Guoliang Li, AI-native Database, SMDB Workshop, ICDE 2020

(c) lecture: https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote15.html