# Software and Data Engineering

## Term Project: Database Tuning with NLP

Presented by:
Sasikanth Kotti

# Presentation Sub-sections

- Introduction
- Objective and Summary
- Results and Comparison
- Project Schematic
- Conclusion

# Introduction

- Databases play a significant role in almost all enterprise applications. These store critical information of applications
- Among different database types, relational databases are predominantly used across applications
- Performance of these databases need to monitored and different steps are to be taken for improvement
- Performance improvements are done both by tuning the queries which are specific to application as well as tuning the global database parameters
- There are general instructions available in database manuals to tune these global parameters, but in almost all deployments there are not sufficient to improve the performance

# Objective and Summary

Database administration consists of monitoring the database to maintain its health. In addition to this, performance tuning also need to be performed both at the query level and also at global level using database configuration parameters. The information that is required to perform tuning is obtained from the manuals and other documents by administrators. In this work, we show how NLP techniques can be used to automate this process and obtain much better database performance. We had also discussed the limitations of this technique and proposed a simple heuristic that showed marginal performance improvements from NLP only based methods. These improvements are shown in MySQL and Postgres databases by executing TPC-H benchmark

# Objective and Summary

**We had successfully performed the below tasks:**

1. Executed sqltuner for both mysql and postgresql database to obtain the baseline recommendations
2. Trained NLP transformer models and obtained recommendations of database settings for both postgresql and mysql
3. Executed TPC-H benchmark for postgresql and mysql databases to verify the performance of baseline and NLP recommended settings for global parameters
4. Argued the limitations of NLP only recommended parameters and proposed a simple heuristic called as Hybrid
5. Hybrid is defined as "Select the Highest Value between NLP Method Vs SQLTuner Method, for binary parameter consider True > False"
6. Demonstrated that parameter recommendations obtained by the Hybrid method showed marginal improvement in performance when compared to NLP only method.

# Results and Comparison

Below are the results for postgresql

| Parameter | Default (As per postgresql_tuner report) | Baseline (As per Paper) | NLP_based (As per paper) | NLP_based (Replicated) | Hybrid |
|---|---|---|---|---|---|
| shared_buffers | 128MB | 16GB | 16GB | 8GB | 16GB |
| maintenance_work_mem | 64MB | 1GB | (default) | (default) | 1GB |
| checkpoint_completion_target | 0.5 | 0.9 | (default) | (default) | 0.9 |
| effective_cache_size | 4GB | (default) | 4GB | 16GB | 4GB |

# Results and Comparison

Performance results for postgresql

| Throughput(Sec) | | | | |
|---|---|---|---|---|
| System | Default | Baseline | NLP_based | Hybrid |
| Postgres (Replicated) | 42 | 31 | 31 | 30 |
| Postgres (Paper) | 141 | 121 | 119 | n/a |

# Results and Comparison

Below are the results for mysql

| Parameter | Default | Baseline | NLP_based | Hybrid |
|---|---|---|---|---|
| innodb_buffer_pool_size | 128 MB (134217728 Bytes) | 2.9GB | 16GB | 16GB |
| innodb_log_file_size | 48 MB (50331648 Bytes) | 16MB | (default) | 48 MB (50331648 Bytes) |
| query_cache_size | 16 MB (16777216 Bytes) | 0 | (default) | 16 MB (16777216 Bytes) |
| query_cache_type | 0 (OFF) | 0 | (default) | 0 (OFF) |

# Results and Comparison

Below are the results for mysql (Contd…)

| Parameter | Default | Baseline | NLP_based | Hybrid |
|---|---|---|---|---|
| innodb_buffer_pool_instances | 1 | (default) | 8 | 8 |
| innodb_flush_log_at_trx_commit | 1 | (default) | 0 | 1 |
| join_buffer_size | 256 KB (262144 Bytes) | (default) | 4GB | 4GB |

# Results and Comparison

Performance results for MySQL

| Throughput(Sec) | | | | |
|---|---|---|---|---|
| System | Default | Baseline | NLP_based | Hybrid |
| MySQL (Replicated) | 113 | 67 | 63 | 62 |
| MySQL (Paper) | 307 | 97 | 82 | n/a |

# Results - NLP Recommended Values Postgresql

Postgresql Database

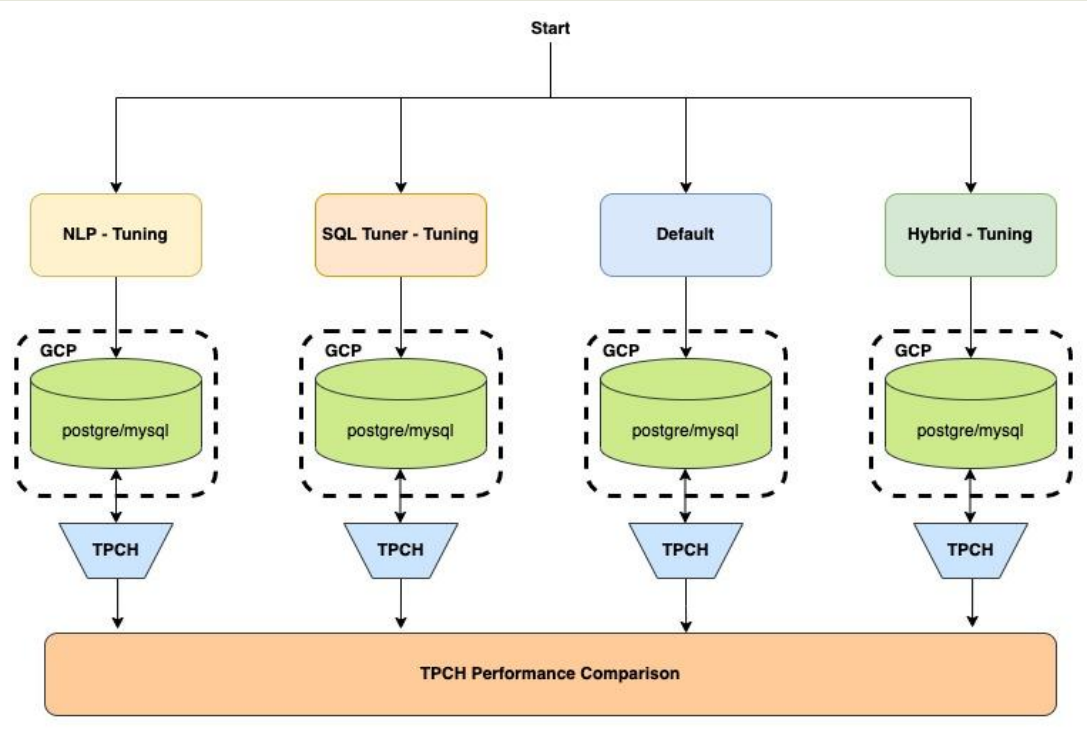
| sentence | context | pred_type | params | values |
|---|---|---|---|---|
| If you have a system with 1GB or more of RAM, a reasonable starting value for shared_buffers is 1/4 of the memory in your system | If you have a system with 1GB or more of RAM, a reasonable starting value for shared_buffers is 1/4 of the memory in your system | 4 | {'shared_buffers'} | ['1GB', '1/4'] |
| A reasonable value would be 50% of the RAM | effective_cache_size | 4 | {'effective_cache_size'} | ['50%'] |

# Results - NLP Recommended Values MySQL

MySQL Database

| sentence | context | pred_type | params | values |
|---|---|---|---|---|
| The following variables are largely dependent on your hardware:innodb_buffer_pool_size Generally, set to 50% – 70% of your total RAM as a starting point | The following variables are largely dependent on your hardware:innodb_buffer_pool_size Generally, set to 50% – 70% of your total RAM as a starting point | 4 | {'innodb_buffer_pool_size'} | ['on', '50%', '70%'] |
| innodb_buffer_pool_instances A best practice is to set this to “8” unless the buffer pool size is < 1G, in which case set to “1” | innodb_buffer_pool_instances A best practice is to set this to “8” unless the buffer pool size is < 1G, in which case set to “1” | 4 | {'innodb_buffer_pool_instances'} | ['8', '1G', '1'] |
| Setting to “0” or “2” will give more performance, but less durability | innodb_flush_log_at_trx_commit Setting to “1” (default in 5.7) gives the most durability | 4 | {'innodb_flush_log_at_trx_commit'} | ['0', '2'] |
| For example, join_buffer_size set to 4GB when the total DB size is less than 1GB | For example, join_buffer_size set to 4GB when the total DB size is less than 1GB | 4 | {'join_buffer_size'} | ['4GB', '1GB'] |

# Project Schematic




- Obtain the parameter values using NLP based method.
- Update postgresql and mysql with these parameters
- Execute TPCH benchmark and record results
- Perform the above 3 steps with parameter values recommended by SQL Tuner, default and Hybrid method
- Perform comparison of TPCH performance of all 4 methods

# Methodology

- As part of this project an instance from GCP of type e2-standard-8 is created. This has 8 CPU's and 32 GB of RAM.
- Postgresql database and MySQL database were installed on this machine
- TPC-H benchmark related to both these databases were setup as per the README files of the respective repositories
- Pipeline as per the authors description is executed to obtain recommendations for parameters of both the databases
- In the same way, recommendations were also obtained by executing SQL tuner tools for both the databases
- TPC-H benchmark is executed to record the performance for both of the above set of parameters. Additionally performance is also obtained with the default parameters
- The parameters obtained by Hybrid method are also setup and performance is obtained for TPC-H benchmark

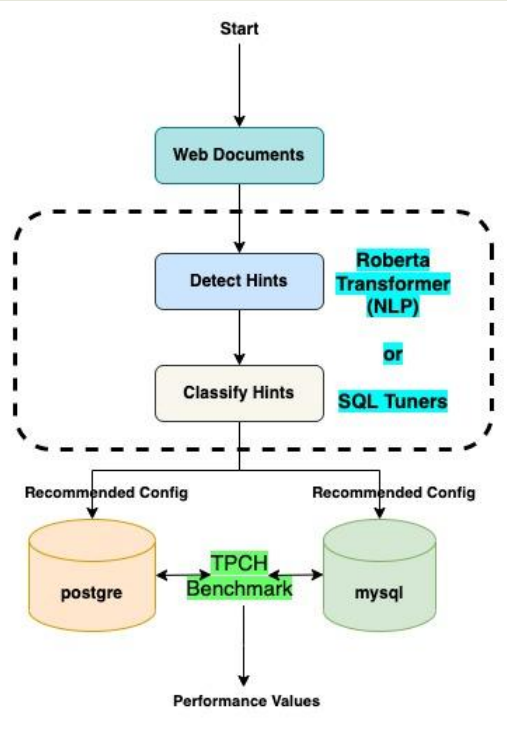
# Results Discussion

- It was observed that the absolute values of the throughput obtained is much better than reported in the paper by the authors
- Also we are able to verify that the performance of the parameters recommended by NLP method is much better than that of heuristic methods (SQL Tuner)
- Also, we found that for postgresql the values recommended by the NLP pipeline are different than that of the values mentioned in the paper by the authors.
- Even using these values showed that NLP method is better than that of heuristic and default settings
- The shortfall of NLP based method is it assumes the configuration of the machine to be high (32 GB in our case)
- However, in most of the cases there can be variety of hardware configurations on which the databases are deployed.
- To also take this factor into consideration we propose a Hybrid approach.

# Results Discussion

- The “Hybrid Method” is defined as follows :
  - Obtain the recommendations using heuristic method (SQL Tuner)
  - Obtain the recommendations using NLP based approach
  - Select the values of different parameters using the rule **“Select the Highest Value between NLP Method Vs SQLTuner Method, for binary parameter consider True > False”**
- It was observed that “Hybrid Method” showed improvement to performance albeit marginally
- However, this will now incorporate both the hardware conditions as well the knowledge from web mined using NLP algorithms
- Also as part of the TPC-H benchmark , contribution to existing github repo is made via a pull request (Ref : https://github.com/Data-Science-Platform/tpch-pgsql/pull/23)

# NLP Method Schematic




- GCP Instance of type e2-standard-8
- 8 vCPUs and 32 GB RAM
- PostgreSQL 10.19 (Ubuntu 10.19-0ubuntu0.18.04.1)
- MySQL 5.7.37-0ubuntu0.18.04.1

# Github Repo & Video Link

Github Repo : https://github.com/ksasi/sde

Video Link : https://youtu.be/2PHzNQ0zVPg

# Conclusion

- We are able to successfully verify that NLP methods can be effectively used to mine knowledge from web and obtain hints for parameters values needed for database configuration
- It was also shown that gives better performance than heuristic methods
- We have also argued that NLP method alone doesn't take into consideration the environments in which databases are deployed
- To also incorporate this , we proposed a Hybrid method and showed that this improved performance albeit marginally
- We hypothesize that advanced algorithms such as reinforcement learning can be used to incorporate environmental conditions as well as knowledge from the web dynamically

# References


- https://dl.acm.org/doi/abs/10.1145/3503780.3503788
- http://vldb.org/pvldb/vol14/p1159-trummer.pdf
- https://github.com/Data-Science-Platform/tpch-pgsql.git
- https://github.com/jfcoz/postgresqltuner
- https://github.com/major/MySQLTuner-perl
- https://github.com/catarinaribeir0/queries-tpch-dbgen-mysql.git
- https://github.com/itrummer/dbbert
- https://tinyurl.com/9crrjezv
- https://github.com/acshulyak/mysql_tpch.git


THANK YOU!