Санкт-Петербургский государственный университет

Программная инженерия

Волков Григорий Валерьевич

Автоматическая подстройка параметров конфигурации СУБД

Выпускная квалификационная работа

Научный руководитель:
к. ф.-м. н., доцент
Луцив Д. В.

Рецензент:
старший менеджер группы автоматизации
ARC департамента
Synopsys Inc.
Никодимов Г. С.

Санкт-Петербург
2022

SAINT-PETERSBURG STATE UNIVERSITY

Software engineering

Grigory Volkov

Automatic tuning of the DBMS configuration parameters

Master's Thesis

Scientific supervisor:
C.Sc., docent D.V. Luciv

Reviewer:
Senior Manager of the automation group
at Synopsys Inc. G.S. Nikodimov

Saint-Petersburg
2022

Оглавление

Введение

Система управления базами данных (СУБД) – неотъемлемая часть многих клиент-серверных приложений. Базы данных могут использоваться для хранения состояния приложения, результатов бенчмарков, протоколирования событий системы и т.д. Конфигурация СУБД напрямую влияет на её производительность и, как следствие, на производительность приложения и других зависящих от СУБД процессов. Конфигурация СУБД включает в себя большое количество переменных, что усложняет подбор оптимальных значений параметров и поиск узкого места в конфигурации. Помимо большого количества возможных конфигураций СУБД, задача нахождения оптимального набора параметров усложняется в виду следующих причин.

1. Большое количество факторов, которые необходимо учитывать при конфигурации СУБД (типы запросов, характеристики сервера и т.д.).

2. Настройка параметров СУБД требует определённой квалификации, поэтому возникает потребность в найме дорогостоящих специалистов [6].

3. Не существует единственно оптимального решения проблемы оптимальной конфигурации СУБД. Поэтому переиспользование конфигурации, оптимизированной под определённые условия, не может гарантировать наилучшую производительность.

Исходя из перечисленных выше особенностей, появляется необходимость автоматизации процесса подстройки параметров СУБД.

1 Постановка задачи

Целью данной работы является создание инструмента автоматической подстройки параметров СУБД.
Для достижения цели были поставлены следующие задачи.

1. Собрать и сформулировать требования к инструменту.
2. Разработать архитектуру инструмента.
3. Реализовать инструмент.
4. Апробировать инструмент на реальных данных и открытых бенчмарках.

2 Предметная область

2.1 Параметры конфигурации

Каждая СУБД имеет своё множество переменных конфигурации. Это множество часто меняется со временем. Например, количество параметров InnoDB[1] с 2013 по 2020 год увеличилось с 138 до 170, т.е. выросло на 20%. [11, 12].

Параметры, используемые в различных СУБД можно разделить на следующие типы:

- числовые (размер различных буферов, время ожидания и т.д.);
- перечисление[2] (настройка различных опций);
- путь на файловой системе (расположение файлов журнала, файлов-идентификаторов процесса, и т.д.).

Также, параметры делятся на следующие типы:

- редактируемые во время исполнения;
- нередактируемые во время исполнения.

2.2 Производительность

Производительность работы СУБД оценивается по следующим аспектам:

- время выполнения запросов;
- использование ресурсов (CPU, RAM, HDD и т.д.).

Второй тип метрик производительности, зависящий от утилизации аппаратных ресурсов, имеет особое значение при ипользовании облачных решений хостинга СУБД. В рамках работы в основном внимание

[1] InnoDB – универсальная система хранения MySQL, используемая по умолчанию

[2] Данный вид также включает в себя логический тип параметра (как частный случай перечисления)

будет уделено времени выполнения запросов как наиболее важной метрике.

3 Требования

На основе консультации с двумя инженерами Synopsys, занимающимися поддержкой внутренних корпоративных сервисов, мною были сформулированы следующие функциональные и нефункциональные требования к инструменту автоматической оптимизации конфигурации СУБД.

3.1 Функциональные требования

Ниже перечислены функциональные требования.

1. Возможность запуска процесса оптимизации на любом сервере внутри корпоративной сети (не только на сервере, на котором работает СУБД).

2. Инструмент должен принимать как параметры:

   - набор запросов, используемых для измерения производительности;
   - набор метрик оценки производительности;
   - параметры конфигурации СУБД, которые можно изменять в процессе оптимизации, и область определения параметров (возможные значения);
   - алгоритм оптимизации.

3. Потенциальная возможность применения инструмента к различным СУБД, независимо от типа (реляционные, документоориентированные, поисковые движки и т.д.).

4. Инструмент должна работать в следующих режимах:

   - тестовый (разрешается при необходимости перезапускать СУБД);
   - производственный (запуск СУБД исключён, и, как следствие, есть возможность изменения лишь параметров, не требующих перезапуска процесса СУБД).

5. В качестве результата работы ожидается 2 артефакта:

    - набор оптимальных параметров СУБД;
    - ресурс, хранящий значения метрик производительности в зависимости от набора параметров конфигурации.

3.2 Нефункциональные требования

Ниже перечислены нефункциональные требования.

1. Инструмент должен работать на операционной системе CentOS 6.6 и выше.

2. Инструмент должен работать без прав суперпользователя.

4 Обзор существующих решений

В этой главе описаны существующие на данный момент пути решения задачи подстройки параметров конфигурации СУБД.

4.1 Литература общего плана

Существует множество книг, описывающих различные аспекты работы с СУБД (администрирование, оптимизации запросов, конфигурация и т.д.) и являющихся выражением практического опыта авторов[17, 16]. В такой литературе, как правило, даются общие рекомендации по решению проблем конфигурации СУБД.

4.2 Общие практики

Документация многих СУБД содержит общие рекомендации по улучшению производительности [8, 13]. При этом в документации описываются общие советы, которые можно использовать для улучшения производетльности, например, на какие параметры стоит обратить особое внимание.
Инструменты автоматического поиска узких мест конфигурации СУБД часто ориентированы на физические и логические аспекты дизайна баз данных. Например, Database Engine Tuning Advisor (DTA) [9] делает рекомендации по некоторым аспектам базы данных (создание или модификация индексов; секционирование[1] таблиц, советы по изменению дизайна базы данных и т.д.).

4.3 Научные работы

Многие статьи на тему автоматического поиска оптимальной конфигурации СУБД имеют ограничения в функциональности, например, ориентированы под оптимизацию размеров буферов или под конкретную СУБД [1, 4].

[1]Секционирование – разделение базы данных или ее составных элементов на отдельные независимые части

Ниже представлены обзоры на научные работы, результатом которых является иснтрумент с функциональностью максимально соответствующей сформулированным ранее требованиям.

4.3.1 iTuned

iTuned [15] – инструмент автоматического подбора оптимальной конфигурации СУБД. К сожалению, исходный код iTuned недоступен, поэтому единственным источником информации является статья, посвященная этому инструменту.
В виду описанных ниже причин, iTuned не может быть применён для решения поставленной задачи.

1. В основе работы инструмента лежит модель гауссовского процесса без возможности измнения метода оптимизации параметров.

2. Представлена лишь высокоуровневая архитектура iTuned. К сожалению, не представлена техническая сторона исследования (какие технологии использовались, какие именно параметры подвергались оптимизации и каким образом и т.д.).

4.3.2 OtterTune

OtterTune [5] – инструмент автоматической конфигурации СУБД, переросший в стартап. OtterTune имеет открытый исходный код[1].

1. По аналогии с iTuned, OtterTune использует задачу регрессии для гауссовского процесса. К сожалению, нет возможности модульно изменить оптимизационный алгоритм.

2. OtterTune распространяется в виде Docker образа. Из этого следует, что для запуска инструмент необходимы права суперпользователя, что противоречит сформулированным требованиям.

[1]По состоянию на 26.03.2021 автор по неизвестным причинам заархивировал репозиторий, и сейчас он доступен только на чтение (https://github.com/cmu-db/ottertune)

5 Архитектура инструмента

Одним из требований является потенциальная поддержка различных СУБД. Это требование необходимо учесть в процессе создания архитектуры инструмента. Также, инструмент должен быть способен работать с различными:

- метриками производительности СУБД;
- форматами входных данных (описаний бенчмарков);
- параметрами СУБД;
- методами изменения параметров СУБД[1].

Задача будущего расширения возможностей инструмента решается с использованием модульного подхода в создании архитектуры инструмента.

На Рис. 1 представлена диаграмма компонент с описанием потоков данных.

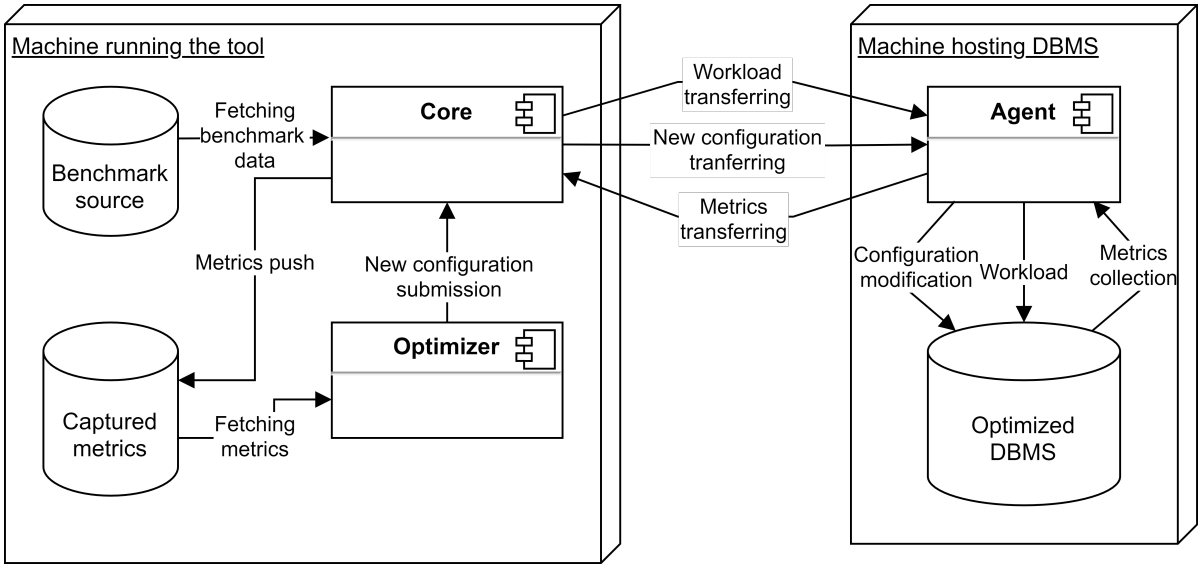

Рис. 1: Высокоуровневая архитектура

[1]Изменения параметров посредством редактирования конфигурационного файла или выполнения запроса на изменение параметра.

Ниже представлено краткое описание компонент.

1. **Core** – ядро инструмента. Ядро выступает в ролях:

   - менеджера репозитория с метриками, полученными в ходе работы инструмента;
   - прослойки между оптимизирующим алгоритмом и агентом, запущенном на сервере СУБД;
   - передатчика бенчмарков агенту для последующей отправки СУБД.

2. **Optimizer** – оптимизирующий алгоритм, получающий на вход данные, хранящиеся в репозитории с метриками. На выход алгоритм выдаёт новую конфигурацию СУБД.

3. **Agent** – агент, который работает на сервере, на котором запущена СУБД.
   Задачи агента заключаются в:

   - изменении конфигурации СУБД, переданной ядром;
   - сборе метрик до и после выполнения единицы тестирования.

На Рис. 1 Агент вынесен на сервер, на котором расположена СУБД по следующим причинам:

- Некоторые метрики оптимальнее собирать с самого сервера, а не удалённо, например, время выполнения запроса, поскольку не происходит искажение метрик за счёт задержки сети.
- Некоторые операции требуют локального (для сервера СУБД) исполнения. К таким операциям можно отнести перезапуск сервера.

6 Реализация

В качестве инструмента реализации был выбран язык Python по следующим причинам.

1. Python используется для написание похожих инструментов (напр., Ansible[1]).

2. Большое количество библиотек научных вычисления для Python (напр., scikit-learn, tensorflow).

3. Один из основных минусов Python как инструмента реализации – производительность. Основной аспект инструмента, потенциально требующий производительности – работа оптимизрующего алгоритма. Поскольку библиотеки научных вычислений зачастую используют функции, работающие в кодах целевой машины[2], при выборе Python в качестве инструмента реализации производительность не будет являться критичной проблемой.

6.1 Протокол коммуникации ядра и агента

Коммуникация между ядром и агентом реализована при помощи протокола WebSocket. Некоторые бенчмарки могут работать продолжительное время. Как следствие, метрики могут быть получены после того, как истечёт время задержки. Протокол WebSocket позволяет поддерживать двустороннюю передачу сообщений без ограничений по времени, что идеально подходит в случае продолжительного запуска бенчмарка.

[1]Ansible – проект с открытым исходным кодом (https://github.com/ansible/ansible), используемый для автоматизации и менеджмента конфигураций, имеющих модульную структуру

[2]Например, NumPy, SciPy

6.2 Поддерживаемые СУБД и параметры конфигурации

Архитектура инструмента позволяет добавлять поддержку различных СУБД. На данный момент добавлена поддержка СУБД MySQL, поскольку она будет использоваться в апробации. Инструмент также абстрагирован от параметров конфигурации, однако добавление поддержки новых параметров конфигурации требует добавление соответствующего плагина.

6.3 Детали реализации

На Рис. 2 представлена диаграмма пакетов реализованного инструмента.

Ниже представлены комментарии к диаграмме пакетов и другие детали реализации.

Перед реализацией инструмента встала задача выбора вида интерфейса взаимодействия с инструментом. Было принято решение использовать интерфейс командной строки для реализации инструмента в виду его универсальности и простоты. Пакет `cli`, являющийся частью пакетов `Core` и `Agent`, занимается обработкой коммандно-строковых параметров.

Ниже представлено описание пакетов ядра и агента с деталями реализации.

Пакеты `agent_commutator` и `core_commutator` реализуют функциональность коммуникации между ядром и агентом инструмента. На момент написания работы, коммуникация реализована с помощью протокола WebSocket[1].

6.3.1 Пакеты ядра

Реализация обработки различных форматов бенчмарков относится к пакету ядра `workload`. В качестве входных бенчмарков поддерживает-

[1]Причина выбора данного протокола описана в главе 6.16.1

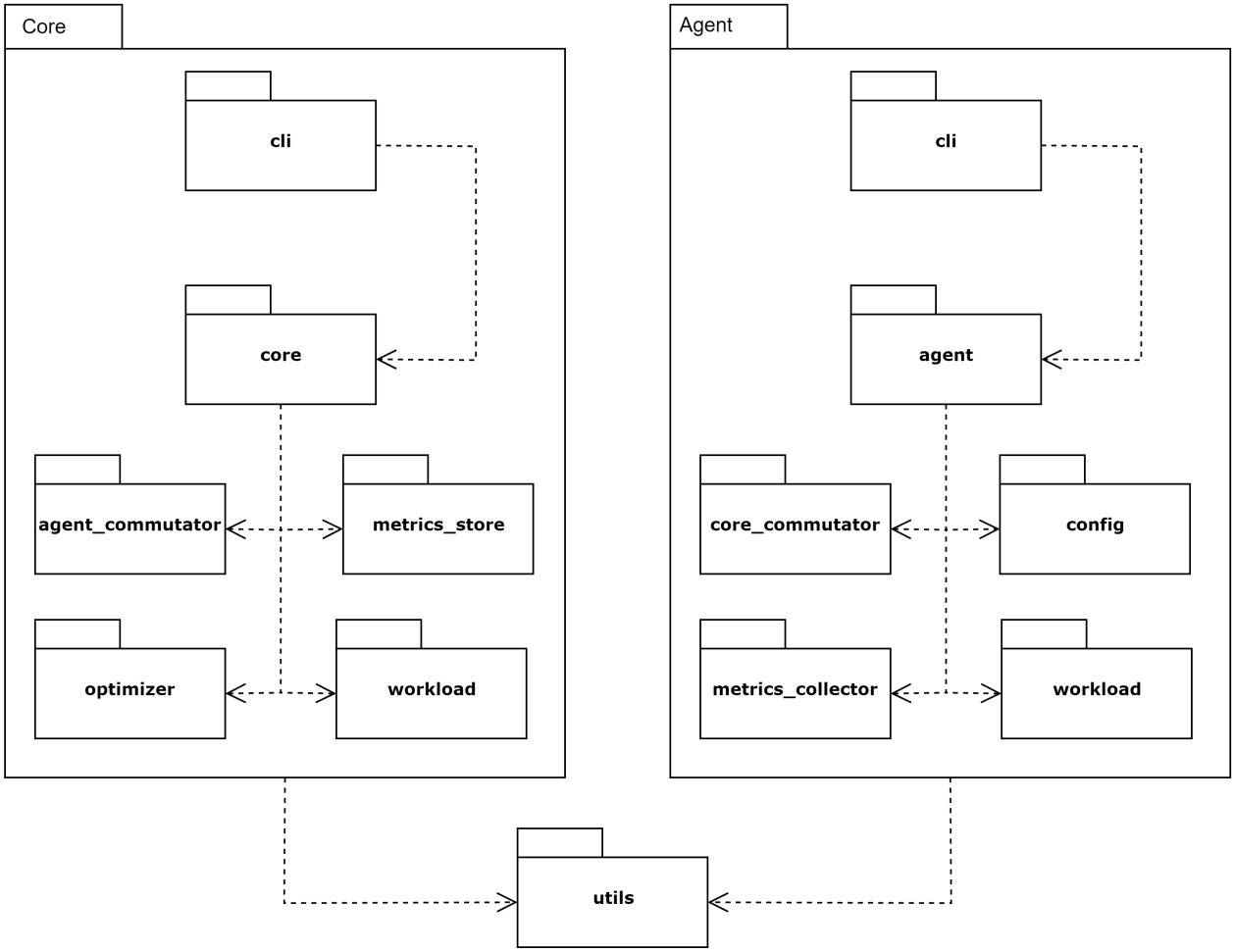

Рис. 2: Диаграмма пакетов

ся файл журналирования SQL запросов MySQL[1] и SQL скрипт. Также была реализована поддержка инструмента BenchBase[2][10]. BenchBase – использующий JDBC[3] фреймворк запуска бенчмарков для СУБД. BenchBase является образовательным проектом группы баз данных Университета Карнеги-Меллона. Фреймворк был апробирован в нескольких успешных исследованиях[2, 3]. BenchBase поддерживает многие виды бенчмарков, среди которых:

- TPC-C;
- TPC-H;

[1]В документации встречается название ”General log”

[2]https://github.com/cmu-db/benchbase (08.03.2022)

[3]JDBC – API взимодействия Java-приложений с различными базами данных

- Wikipedia;
- AuctionMark.

При проведении экспериментов, инструмент показал нестабильную работу, поэтому он не участвовал в апробации.

Метрики сохраняются в файлы формата JSON или CSV, по выбору пользователя инструмента. Поддержка данных способов хранения метрик реализована в пакете `metrics_store`. В полученных артефактах содержится информация о конфигурациях СУБД, которые были испытаны, и полученных метриках.

В пакете `optimizer` содержатся реализации оптимизирующих алгоритов. В качестве алгоритмов оптимизации используется 2 метода: Grid search и Tree of Parzen Estimators (TPE). Выбор Grid search обусловлен простотой реализации и интегрирования в инструмент. Алгоритм TPE был выбран после изучения применения его применения в аналогичных задачах[14][7].

6.3.2 Пакеты агента

В пакете `workload` со стороны агента реализована нагрузка базы данных полученными от ядра бенчмарками. На момент написания работы реализована поддержка нагрузки СУБД MySQL[1].

Время выполнения бенчмарков является главной метрикой, которую инструмент стремится минимизировать. Сбор метрик реализован в пакете `metrics_collector`.

Работа с конфигурационными файлами реализована в пакете `config` с помощью Python библиотеки Configparser[2]. На момент написания работы это единственный способ модификации конфигурационных параметров. Однако, архитектура инструмента позволяет гибко настроить способ изменения конфигурации индивидуально для каждого параметра. Например, одновременно изменять одни параметры через конфигурационный файл, а другие – с помощью SQL запросов.

[1]https://github.com/PyMySQL/PyMySQL (06.05.2022)

[2]https://docs.python.org/3/library/configparser.html (06.05.2022)

7 Апробация

7.1 Подготовка продуктовых данных для апробации

Под подготовкой продуктовых данных понимается подготовка бенчмарка, на основе которого будет производится оптимизация конфигурации СУБД. Бенчмарк был сгенерирован на основе запросов к MySQL базе данных, хранящей результаты тестов. Для генерации бенчмарка было произведено журналирование запросов к базе данных. Было собрано порядка 1000 SQL запросов. На основе собранной информации было создано 3 бенчмарка по типу нагрузки, моделирующие:

- добавление данных в базу;
- анализ результатов тестов (для построения отчётов);
- смешанный (включающий в себя 2 предыдущих бенчмарка).

Анализ работы инструмента с использованием вышеописанных бенчмарков позволит оценить работу инструмента с продуктовыми данными.

7.2 Результаты

Описание окружения и конфигурационные параметры, использующиеся при апробации, представлены в приложении.

Следующие алгоритмы использовались для апробации:

- Tree of Parzen Estimators (TPE);
- Grid search.

Ниже представлена таблица[1] с временем работы бенчмарков в производственной среде без оптимизации конфигурации. Изначальная конфигурация была подобрана инженером компании.

Вид нагрузки	Время работы, сек
1	14
2	165
3	179

Таблица 1: Результаты работы бенчмарков без оптимизации.

Ниже приведена таблица с результатами апробации инструмента в промежуточной среде.

Алгоритм	Вид нагрузки бенчмарка	Количество опробованных конфигураций	Лучшее время работы нагрузки, сек	Количество запусков инструмента
Grid search	1	972	12	3
	2		153	
	3		169	
TPE	1	50	12	10
	2		154	
	3		173	

Таблица 2: Результаты апробации инструмента в промежуточной среде.

Результаты апробации в промежуточной среде показали, что расхождения в результатах работы двух алгоритмов незначительны. По этой причине при апробации в производственной среде использовался только алгоритм TPE.

Ниже представлена таблица с результатами апробации инструмента в производственной среде (в таблице представлено среднее трёх запусков).

[1]Нумерация видов нагрузки приведена в главе, описывающей подготовку продуктовых данных для апробации

Количество опробованных конфигураций	Вид нагрузки	Лучшее время работы нагрузки, сек
10	1	13
	2	148
	3	173
30	1	11
	2	153
	3	169
50	1	11
	2	148
	3	162

Таблица 3: Результаты апробации инструмента в промежуточной среде.

8 Заключение

В рамках выполнения работы были выполнены следующие задачи.

1. Были собраны и сформулированы функциональные и нефункциональные требования к инструменту.
2. Разработана модульная архитектура инструмента, абстрагирующаяся от:
   - вида бенчмарка;
   - СУБД и конфигурационных параметров;
   - алгоритма оптимизации.
3. Реализован инструмент с интерфейсом командной строки, удовлетворяющий требованиям.
4. Был сгенерирован бенчмарк на основе нагрузки на производственную базу данных.
5. Инструмент был апробирован на сгенерированном бенчмарке и продуктовом сервере. В результате апробации, производительность базы данных на бенчмарке улучшилась на 9%.

Список литературы

[1] A. J. Storm C. Garcia-Arellano S. Lightstone Y. Diao and Surendra M. Adaptive Self-tuning Memory in DB2. — 2006.

[2] Difallah Djellel Eddine, Pavlo Andrew, Curino Carlo, and Cudré-Mauroux Philippe. BenchPress: Dynamic Workload Control in the OLTP-Bench Testbed // Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. — 2015. — Vol. 7, no. 4. — P. 1069–1073. — Access mode: `https://db.cs.cmu.edu/papers/2015/p1069-vanaken.pdf` (online; accessed: 08.03.2022).

[3] Difallah Djellel Eddine, Pavlo Andrew, Curino Carlo, and Cudré-Mauroux Philippe. Benchmarking OLTP/Web Databases in the Cloud: The OLTP-Bench Framework // Proceedings of the Fourth International Workshop on Cloud Data Management. — 2012. — Vol. 7, no. 4. — P. 17–20. — Access mode: `http://www.cs.cmu.edu/~pavlo/static/papers/oltpbench.pdf` (online; accessed: 08.03.2022).

[4] D. N. Tran P. C. Huynh Y. C. Tay and Tung A. K. H. A new approach to dynamic self-tuning of database buffers. — 2008.

[5] D.V. Aken A. Pavlo G. J. Gordon B. Zhang. Automatic Database Management System Tuning Through Large-scale Machine Learning. — 2017.

[6] Doyly A. IT Jobs: Career Options, Job Titles, and Descriptions // The Balance Careers. — 2021. — Access mode: `https://www.thebalancecareers.com/list-of-information-technology-it-job-titles-2061498` (online; accessed: 23.03.2021).

[7] Jorge Francisco Madrigal Diaz Camille Maurice Frédéric Lerasle. Hyper-parameter optimization tools comparison for Multiple Object Tracking applications. — 2018. — Vol. 17, no. 15. — Access mode:

https://hal.archives-ouvertes.fr/hal-01897032/document (online; accessed: 05.07.2022).

[8] Microsoft Corporation. Performance Monitoring and Tuning Tools // SQL Server technical documentation. — 2017. — Access mode: https://docs.microsoft.com/en-us/sql/relational-databases/performance/performance-monitoring-and-tuning-tools?view=sql-server-ver15 (online; accessed: 23.03.2021).

[9] Microsoft Corporation. Database Engine Tuning Advisor // SQL Server technical documentation. — 2021. — Access mode: https://docs.microsoft.com/en-us/sql/relational-databases/performance/database-engine-tuning-advisor?view=sql-server-ver15 (online; accessed: 23.03.2021).

[10] Difallah Djellel Eddine, Pavlo Andrew, Curino Carlo, and Cudré-Mauroux Philippe. OLTP-Bench: An Extensible Testbed for Benchmarking Relational Databases // PVLDB. — 2013. — Vol. 7, no. 4. — P. 277–288. — Access mode: http://www.vldb.org/pvldb/vol7/p277-difallah.pdf.

[11] Oracle Corporation. 14.14 InnoDB Startup Options and System Variables // MySQL 5.6 Reference Manual. — 2021. — Access mode: https://dev.mysql.com/doc/refman/5.6/en/innodb-parameters.html (online; accessed: 23.03.2021).

[12] Oracle Corporation. 14.14 InnoDB Startup Options and System Variables // MySQL 8.0 Reference Manual. — 2021. — Access mode: https://dev.mysql.com/doc/refman/8.0/en/innodb-parameters.html (online; accessed: 23.03.2021).

[13] Oracle Corporation. Chapter 8 Optimization // MySQL 5.6 Reference Manual. — 2021. — Access mode: https://downloads.mysql.com/docs/refman-5.6-en.a4.pdf (online; accessed: 23.03.2021).

[14] Roman Vasiliev Dmitrij Koznov George Chernishev Aleksandr Khvorov Dmitry Luciv Nikita Povarov. TraceSim: a method for calculating

stack trace similarity // Proceedings of the 4th ACM SIGSOFT International Workshop on Machine-Learning Techniques for Software-Quality Evaluation. — 2020. — Vol. 7, no. 4. — P. 25–30. — Access mode: https://arxiv.org/pdf/2009.12590.pdf (online; accessed: 08.03.2022).

[15] S. Duan V. Thummala and Babu S. Tuning database configuration parameters with iTuned. — 2009.

[16] Б.А. Новиков Г.Р. Домбровская. Книга Настройка приложений баз данных. — БХВ-Петербург, 2006.

[17] Б.А. Новиков Е.А. Горшкова Н.Г. Графеева. Основы технологий баз данных. — ДМК Пресс, 2020.

Приложение А Апробационное окружение

Окружение, в котором была выполнена апробация, обладает следующими параметрами:

- CPU: 4 Intel(R) Xeon(R);
- RAM: 32Gb;
- Версия MySQL: 5.6.12;
- Размер базы данных: 24Gb.

На основе консультации с двумя инженерами Synopsys, занимающимися поддержкой внутренних корпоративных сервисов, были выбраны 7 параметров MySQL [11] для апробации:

- `innodb_buffer_pool_size`;
- `innodb_buffer_pool_instances`;
- `innodb_checksum_algorithm`;
- `innodb_doublewrite`;
- `innodb_io_capacity`;
- `innodb_compression_level`;
- `innodb_random_read_ahead`.