УТВЕРЖДАЮ

Генеральный директор ООО «ППГ Разработка»

\_\_\_\_\_\_\_\_\_\_\_ В.В. Попов

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2022 г.

Отчет

по результатам аудита состояния БД ib\_stmt кластера

СУБД PostgreSQL

Версия №1 от 11.10.2022г.

на 20 листах

## **Оглавление**

[**Оглавление** 2](#_Toc1)

[1. **Цели и задачи аудита** 3](#_Toc2)

[2. **Анализ используемых программных и аппаратных средств** 4](#_Toc3)

[2.1. **Анализ используемых аппаратных средств** 4](#_Toc4)

[2.2. **Анализ используемых программных средств** 5](#_Toc5)

[3. **Общие выводы** 6](#_Toc6)

[4. **Рекомендации по оптимизации запросов** 7](#_Toc7)

[5. **Дубли индексов** 16](#_Toc8)

[6.  **Рекомендации по конфигурации параметров операционной системы** 17](#_Toc9)

[6.1. **Настройка HugePages** 17](#_Toc10)

[6.2. **Настройка Transparent Hugepage** 17](#_Toc11)

[7.  **Поиск разросшихся таблиц и индексов** 19](#_Toc12)

[8.  **Поиск large object** 21](#_Toc13)

[9.  **Конфигурация СУБД PostgreSQL** 22](#_Toc14)

[9.1. **Настройка фонового процесса записи** 22](#_Toc15)

[9.2. **Настройка процесса автоматической очистки** 22](#_Toc16)

[9.3. **Дополнительные параметры по настройке СУБД** 22](#_Toc17)

## **Цели и задачи аудита**

Целью проводимых работ является выдача рекомендаций по конфигурации операционной системы, СУБД Postgresql и ответы на вопросы заказчика.

## **Анализ используемых программных и аппаратных средств**

### **Анализ используемых аппаратных средств**

Ниже приведены характеристики сервера **p0dbop-pg5003lp.region.vtb.ru** с СУБД PostgreSQL, являющегося мастером от 05.10.2022:

|  |  |
| --- | --- |
| **Сервер** | **p0dbop-pg5003lp.region.vtb.ru** |
| Роль сервера | Мастер |
| Тип сервера | Виртуальный сервер |
| Операционная система | Red Hat Enterprise Linux Server release 7.9 (Maipo) 3.10.0-1160.42.2.el7.x86\_64 |
| CPU (ядро) | Кол-во сокетов: 2  Кол-во ядер на сокет: 24  Кол-во трэдов на ядро: 2  Идентификатор вендора: GenuineIntel  Наименование модели: Intel(R) Xeon(R) Gold 6248R CPU @ 3.00GHz  NUMA Nodes: 2 |
| RAM | 503GB |
| Диск | Filesystem Type Size Used Avail Use% Mounted on  devtmpfs devtmpfs 252G 0 252G 0% **/**dev  tmpfs tmpfs 252G 2.7M 252G 1% **/**dev**/**shm  tmpfs tmpfs 252G 2.1G 250G 1% **/**run  tmpfs tmpfs 252G 0 252G 0% **/**sys**/**fs**/**cgroup  **/**dev**/**mapper**/**vg\_01**-**lv\_root xfs 10G 3.2G 6.8G 32% **/**  **/**dev**/**sda2 xfs 506M 183M 323M 37% **/**boot  **/**dev**/**mapper**/**vg\_01**-**lv\_tmp xfs 3.0G 34M 3.0G 2% **/**tmp  **/**dev**/**mapper**/**vg\_01**-**lv\_pgaudit xfs 200G 353M 200G 1% **/**pg\_audit  **/**dev**/**mapper**/**vg\_01**-**lv\_etcd xfs 15G 403M 15G 3% **/**app**/**etcd  **/**dev**/**mapper**/**vg\_01**-**lv\_home xfs 5.0G 38M 5.0G 1% **/**home  **/**dev**/**mapper**/**vg\_p0dbop\_pg5003lp\_pgdata**-**lv\_pgdata xfs 4.0T 2.7T 1.3T 68% **/**pg\_data  **/**dev**/**mapper**/**vg\_01**-**lv\_var xfs 8.0G 1.8G 6.2G 23% **/**var  **/**dev**/**mapper**/**vg\_p0dbop\_pg5003lp\_pgbackup**-**lv\_pgbackup xfs 4.0T 2.7T 1.3T 68% **/**pg\_backup  **/**dev**/**sda1 vfat 200M 9.9M 190M 5% **/**boot**/**efi  tmpfs tmpfs 51G 0 51G 0% **/**run**/**user**/**653  **/**dev**/**mapper**/**vg\_pgwal**-**lv\_pgwal xfs 2.0T 98G 2.0T 5% **/**pg\_walarchive  tmpfs tmpfs 51G 0 51G 0% **/**run**/**user**/**1376548563  /dev/mapper/vg\_pgwal-lv\_pgwal xfs 2.0T 210G 1.8T 11% /pg\_walarchive  tmpfs tmpfs 51G 0 51G 0% /run/user/1376548563 |
| Директория данных СУБД | /pg\_data |

**Таблица 2.1.1. Характеристики сервера** **p0dbop-pg5003lp.region.vtb.ru**

### **Анализ используемых программных средств**

СУБД работает под управлением ПО patroni 2.1.2, в таблице ниже представлены используемая на сервере версия от 01.09.2022:

|  |  |
| --- | --- |
| Сервер | Версия |
| Мастер  (p0dbop-pg5003lp.region.vtb.ru  ) | PostgreSQL 11.14 on x86\_64-pc-linux-gnu, compiled by gcc (GCC) 4.8.5 20150623 (Red Hat 4.8.5-44), 64-bit |

**Таблица 2.2.1. Используемая версия СУБД**

В настоящий момент СУБД обновлена до версии 13.

## **Общие выводы**

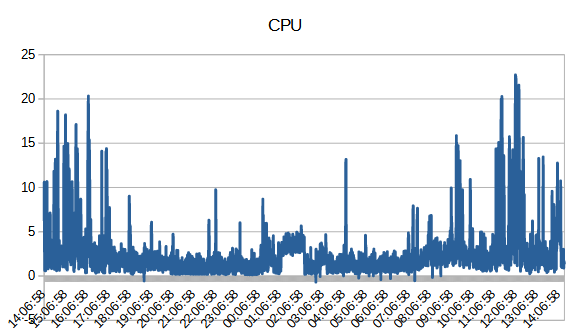
Выдан ряд рекомендаций по:

1. Оптимизации работы запросов
2. Настройке параметров операционной системы
3. Настройке параметров СУБД PostgreSQL

Ожидается, что рекомендации позволят снизить нагрузку на CPU и дисковую подсистему.

Рекомендуется провести повторный анализ после проведения работ согласно данным рекомендациям.

Также требуется сравнить графики нагрузки на CPU до и после оптимизации. Ниже приведён график потребления CPU за день до оптимизации:



**Рис 3.1. График потребления CPU до оптимизации**

## **Рекомендации по оптимизации запросов**

1. Обнаружен запрос к таблицам st\_contractor\_statement, st\_contractor\_account, st\_contract\_account\_relation, st\_account\_statement\_previous\_version\_chain, максимальное время работы которого 18503 мс, минимальное время выполнения 36.764 мс, среднее время выполнения 8651 мс, за час он был выполнен 33 раза

Текст запроса представлен в файле **contractor\_statement.sql**.

Поскольку в запросе выводится только поле client\_account\_id таблицы **st\_contractor\_statement**, то join других таблиц можно заменить на exists. Предлагаемый вариант запроса описан в файле **proposals/contractor\_statement\_prop.sql**

Также для ускорения доступа к данным таблиц **st\_contractor\_account\_relation** и **st\_contractor\_statement** предлагается создать следующие индексы:

**CREATE** **INDEX** st\_contractor\_statement\_client\_account\_date\_idx

**ON** stmt**.**st\_contractor\_statement**(**client\_account\_id**,** statement\_date**,** contractor\_account\_id**,** contractor\_bank\_id**,** credit**);**

**CREATE** **INDEX** st\_car\_contr\_acc\_contr\_id\_ix

**ON** stmt**.**st\_contractor\_account\_relation**(**contractor\_account\_id**,** contractor\_id**);**

**Листинг 4.1. Индексы для таблиц st\_contractor\_account\_relation и st\_contractor\_statement**

1. Обнаружен запрос к таблицам **st\_contractor\_bank и st\_contractor\_statement**, минимальное время выполнения 7869.453 мс, максимальное 22131.736 мс, среднее 15000 мс, 595 мс. Количество выполнений за час 10.

Текст запроса представлен в файле **st\_contractor\_bank.sql**

Поскольку в запросе требуются поля только таблицы **st\_contractor\_bank**, то join можно заменить на exists, distinct при этом будет не нужен. Предлагаемый вариант запроса описан в файле **proposals/st\_contractor\_bank.sql**

Также для ускорения доступа к данным st\_contractor\_statement предлагается создать следующий индекс:

**CREATE** **INDEX** stcs\_st\_dt\_cl\_acc\_id\_ca\_cb\_id\_ix

**ON** stmt**.**st\_contractor\_statement**(**statement\_date**,** client\_account\_id**,** contractor\_account\_id**,** contractor\_bank\_id**);**

**Листинг 4.2. Индекс для ускорения доступа к st\_contractor\_statement**

1. Обнаружен запрос добавления данных в **st\_contractor\_account\_relation и st\_contractor\_account,** являющийся лидером по количеству записи и чтения блоков из временных файлов, за час 9080217 блоков прочитано, 19063109 блоков записано.

Текст представлен в файле **stmt\_st\_contractor\_account\_rel\_ins.sql**

Минимальное время выполнения 20 мс, максимальное время выполнения 566.804 мс, среднее время выполнения 346.904 мс, количество выполнений за час 3220.

Поскольку в запросе содержится конструкция **on conflict do nothing**, то предлагается её заменить на NOT EXISTS, поскольку данные добавляются в таблицу только в случае их отсутствия. Текст запроса представлен в следующем файле:

**sql/proposals/stmt\_st\_contractor\_account\_rel\_ins.sql**

Необходимые индексы присутствуют в таблицах st\_aoperation, st\_contractor, st\_contractor\_account, однако, требуется получить план выполнения для поиска неоптимальных фрагментов. Его можно получить следующим образом:

* Взять значения statement\_id, полученные из запросов ниже

**SELECT** statement\_id AS stid\_credit

**,** **COUNT(\*)** **AS** cnt

**FROM** stmt**.**st\_aoperation a

**WHERE** a**.**credit

**GROUP** **BY** statement\_id

**ORDER** **BY** cnt **DESC**

**LIMIT** 20**;**

**SELECT** statement\_id AS stid\_not\_credit

**,** **COUNT(\*)** **AS** cnt

**FROM** stmt**.**st\_aoperation a

**WHERE** NOT a**.**credit

**GROUP** **BY** statement\_id

**ORDER** **BY** cnt **DESC**

**LIMIT** 20**;**

**Листинг 4.3. Определение значений statement\_id для использования в качестве параметров запроса**

* Собрать статистику по таблицам **st\_aoperation, st\_contractor\_account** и **st\_contractor** с помощью VACUUM(ANALYZE)
* Выполнить **explain** запроса из **stmt\_st\_contractor\_account\_rel.sql**

Также для ускорения доступа к данным таблиц st\_contractor и st\_contractor\_account предлагается создать следующие индексы:

**CREATE** **UNIQUE** **INDEX** st\_contractor\_inn\_unique\_idx **ON** stmt**.**st\_contractor**(**inn**,** id**);**

**CREATE** **UNIQUE** **INDEX** st\_contractor\_account\_number\_bic\_idx **ON** stmt**.**st\_contractor\_account**(**"number"**,** bic**,** id**);**

**Листинг 4.4. Индексы для ускорения доступа к данным таблиц st\_contractor и st\_contractor\_account**

1. Обнаружен запрос добавления данных в таблицу st\_contractor\_statement, который также является лидером по чтению и записи блоков временных файлов. За час было прочитано 9083661 блоков, записано 19064118 блоков.

Минимальное время выполнения 10 мс, максимальное время выполнения 582.383 мс,среднее время выполнения 286.908 мс, за час запрос выполнился 3071 раз.

Текст запроса представлен в файле **st\_contractor\_statement\_ins.sql**

В данном случае также нужно получить план выполнения для поиска неоптимальных фрагментов, алгоритм действий аналогичен алгоритму пункта 3.

Для ускорения доступа к таблице **st\_contractor\_bank** предлагается создать следующий индекс:

**CREATE** **UNIQUE** **INDEX** st\_contractor\_bank\_number\_bic\_idx **ON** stmt**.**st\_contractor\_bank**(**bic**,** id**);**

**Листинг 4.5. Индекс для ускорения доступа к таблице st\_contractor\_bank**

1. Обнаружен запрос добавления данных в таблицу st\_contractor\_account, он аналогичен запросам пунктов 3 и 4. Текст запроса представлен в файле **stmt\_st\_contractor\_account\_ins.sql**

Поскольку в запросах содержится **on conflict do nothing**, то предлагается её заменить на NOT EXISTS, поскольку данные добавляются в таблицу только в случае их отсутствия. Текст запроса представлен в файле:

**sql/proposals/stmt\_st\_contractor\_account\_ins.sql**

В данном случае также нужно получить план выполнения для поиска неоптимальных фрагментов, алгоритм действий аналогичен алгоритму пункта 3.

1. Обнаружены запросы к таблице st\_notification\_account\_change, минимальное время выполнения 6625.650 мс, максимальное время 17974.333 мс, среднее время 7824.886 мс. За час запросы выполнились 20 раз.

Тексты запросов представлены в файле **notif\_acc\_change.sql**

Для оптимизации поиска по полю client\_id предлагается использовать следующий индекс:

**CREATE** **INDEX** stnac\_cl\_id\_status\_ix **ON** stmt**.**st\_notification\_account\_change**(**client\_id**,** status**);**

**Листинг 4.6. Индекс для ускорения поиска по полю client\_id таблицы st\_notification\_account\_change**

1. Ниже приведён листинг представления st\_account\_client\_extension, в котором присутствует конструкция GROUP BY .

**SELECT** rel**.**account\_id

**,** cln**.**short\_name

**,** cln**.**inn

**FROM** stmt**.**st\_acc\_account\_client\_relation rel

**JOIN** stmt**.**st\_cln\_client\_common\_dto prof

**ON** prof**.**id **=** rel**.**client\_id

**JOIN** stmt**.**st\_cln\_client\_master cln

**ON** cln**.**id **=** prof**.**client\_master\_id

**GROUP** **BY** rel**.**account\_id**,** cln**.**id**;**

**Листинг 4.7. Запрос представления st\_account\_client\_extension**

account\_id, client\_id - первичный ключ таблицы **stmt.st\_acc\_account\_client\_relation**. id - первичный ключ таблицы **stmt.st\_cln\_client\_common\_dto**, поэтому комбинация **rel.account\_id, rel.client\_id и prof.id** также является уникальной. Значит, комбинация **rel.account\_id, cln.id** также является уникальной. Поэтому конструкция group by здесь не нужна, она не позволяет сразу отфильтровывать по полю account\_id.

1. Обнаружен запрос к данным представления **stmt.st\_statement\_version**, минимальное время выполнения 417.68 мс, максимальное время 947.40 мс, среднее время выполнения 762.63 мс, количество выполнений за час 10.

Текст запроса представлен в файле **stmt\_version.sql**.

Запрос представляет собой объединение данных таблиц **stmt.st\_asstatement** и **stmt.st\_cstatement** с соединением по таблице **stmt.st\_account\_statement.** Таким образом, фильтры применяются только после забора всех данных, что не является оптимальным решением.

Предлагаемый вариант запроса описан в файле **proposals/stmt\_version.sql**

1. Обнаружен запрос с последовательным сканированием к таблице **st\_acc\_account\_dbo**, текст представлен в файле **account\_dto\_simple.sql.** Данный запрос является лидером по количеству прочитанных блоков из общего буферного кэша.

Для ускорения доступа к данным таблицы **stmt.st\_acc\_account\_dto** предлагается создать следующий индекс:

**CREATE** **INDEX** st\_acc\_dtio\_bc\_dto\_id\_tpa\_vfdbo\_ix

**ON** stmt**.**st\_acc\_account\_dto**(**branch\_common\_dto\_id**,** third\_party\_account**,** visible\_for\_dbo**);**

**Листинг 4.8. Индекс для ускорения доступа к данным таблицы stmt.st\_acc\_account\_dto**

1. Обнаружены запросы с последовательным сканированием данных таблицы stmt.st\_request. Тексты запросов представлены в файле **st\_request.sql**. Ниже приведена таблица с временами выполнения запросов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № запроса | Минимальное время выполнения (мс) | Максимальное время выполнения (мс) | Среднее время выполнения (мс) | Количество выполнений в час |
| 1 | 3220.097 | 3697.921 | 3563.597 | 42 |
| 2 | 5719.339 | 6344.385 | 6071.959 | 107 |
| 3 | 3085.538 | 6162.525 | 3849.498 | 33 |
| 4 | 3333.633 | 6149.090 | 5620.879 | 7 |
| 5 | 3234.972 | 6201.866 | 5523.397 | 7 |
| 6 | 5286.933 | 5966.796 | 5502.457 | 7 |

**Таблица 4.1. Времена выполнения запросов к таблице st\_request**

Для ускорения доступа к данным таблицы st\_request предлагается создать индекс следующего вида:

**CREATE** **INDEX** st\_req\_rec\_dt\_ix

**ON** stmt**.**st\_request**(**received\_date**,** status**);**

**CREATE** **INDEX** st\_req\_rec\_dt\_bic\_ix

**ON** stmt**.**st\_request**(**received\_date**,** bic**);**

**CREATE** **INDEX** st\_req\_req\_dt\_status\_ix

**ON** stmt**.**st\_request**(**requested\_date**,** status**);**

**Листинг 4.9**. **Индекс для ускорения доступа к данным таблицы st\_request**

В условии ниже received\_date is not null является лишним:

**where** received\_date **>=** now**()** **-** $2 **::**interval

**and** received\_date **is** **not** **null**

**Листинг 4.10. Условие фильтрации данных в запросе к таблице st\_request**

В **sql/proposals/stmt\_version.sql** представлена версия запроса с применением фильтров по полям account\_id и normalized\_date таблиц **st\_astatement** и **st\_cstatement**.

1. Обнаружен запрос поиска по представлению **stmt.st\_statement\_version** с последующим соединением по таблице **st\_contractor\_statement**. Как и в пункте 11, фильтры по полям account\_id и normalized\_date не могут быть сразу применены.

Рекомендуется использовать версию запроса из файла **sql/proposals/stmt\_version.sql**

1. Обнаружены запросы к таблице st\_aoperation, работающие 2 минуты. Тексты представлены в файле **st\_aoperation.sql.** Причиной может быть использование offset, из-за чего запрос выбирает много данных и пропускает большую её часть.

Также предлагается вместо индекса st\_aoperation\_st\_id\_idx создать следующий объект:

**CREATE** **INDEX** st\_ao\_st\_id\_oper\_type\_del\_at\_ix

**ON** stmt**.**st\_aoperation**(**statement\_id**,** oper\_type**,** deleted\_at**);**

**Листинг 4.11. Вариант замены индекса st\_aoperation\_st\_id\_idx**

1. Обнаружен запрос с последовательным сканированием таблицы **stmt.st\_background\_operations** по полю sending\_status, текст запроса представлен в файле **st\_background\_operations.sql**.

Минимальное время выполнения 93.692 мс, максимальное время выполнения 336.949 мс, среднее время выполнения 118.234 мс, количество выполнений за час 203

Для ускорения доступа к данным предлагается создать следующий индекс:

**CREATE** **INDEX** st\_bo\_send\_st\_ix

**ON** stmt**.**st\_background\_operations**(**sending\_status**);**

**Листинг 4.12. Индекс для ускорения доступа к таблице st\_background\_operations по полю sending\_status**

1. Обнаружен запрос к таблице st\_contractor\_account с последующим соединением по таблице st\_contractor\_interaction.

Запрос представлен в файле **st\_contractor\_account\_interaction.sql**

Поскольку в запросе извлекаются только поля таблицы stmt.st\_contractor\_account то join с distinct можно заменить на exists.

Изменённый вариант запроса представлен в следующем файле:

**sql/proposals/st\_contractor\_account\_interraction.sql**

Возможно, что в некоторых случаях поле statement\_date позволит отфильтровать больше данных, чем поле client\_account\_id. Для ускорения доступа к данным предлагается создать следующий индекс:

**CREATE** **INDEX** stci\_stdt\_caid\_contr\_aid\_ix

**ON** stmt**.**st\_contractor\_interaction**(**statement\_date**,** client\_account\_id**,** contractor\_account\_id**);**

**Листинг 4.13. Индекс для ускорения доступа к таблице currc\_currency\_contract**

1. Обнаружен запрос с поиском по полю **contractor\_account\_id** таблицы stmt.st\_contractor\_account\_relation, текст запроса представлен в файле **st\_contractor\_list.sql**

Минимальное время выполнения 16460.753 мс, максимальное время выполнения 48678.345 мс, среднее время выполнения 29926.691 мс.

Поскольку в запросе извлекаются только поля таблицы **stmt.st\_contractor**, то join с distinct можно заменить на exists. Изменённый вариант запроса представлен в **sql/proposals/st\_contractor\_list.sql**

Поиск осуществляется по индексу **st\_contractor\_account\_relation\_idx**, в котором поле contractor\_account\_id не является лидирующем.

Предлагается создать следующий индекс:

**CREATE** UNIQUE **INDEX** st\_contr\_acc\_rel\_ux

**ON** stmt**.**st\_contractor\_account\_relation**(**contractor\_account\_id**,** contractor\_id**);**

**Листинг 4.14.** **Индекс для ускорения поиска по полю contractor\_account\_id**

Стоит отметить, что с 14-ой версии СУБД PostgreSQL улучшен механизм поиска по списку значений. До этого он был линейным, сейчас он осуществляется с помощью хэширования.

1. Обнаружен запрос поиска данных по таблице **st\_request** и **st\_account\_statement,** являющийся лидером по количеству ожиданий выполнения операций ввода-вывода. Текст запроса представлен в файле **st\_req\_acc\_stmt.sql**

У поля **statement\_id** таблицы **stmt.st\_account\_statement** есть ограничение is not null. Поэтому условие statement\_id is null может быть только тогда, когда для строки таблицы **stmt.st\_request** не нашлось соответствующей записи в таблице **stmt.st\_account\_statement**

Если normalized\_date и statement\_date позволяет отфильтровать больше строк, чем account\_id, то стоит создать следующие индексы

**CREATE** **INDEX** str\_nd\_acc\_id\_ix

**ON** stmt**.**st\_request**(**normalized\_date**,** account\_id**);**

**CREATE** **UNIQUE** **INDEX** sas\_st\_dt\_acc\_id\_ix

**ON** stmt**.**st\_account\_statement**(**statement\_date**,** account\_id**);**

**Листинг 4.15. Индексы для st\_request и st\_account\_statement**

1. Обнаружен запрос поиска по таблицам st\_acc\_account\_dto, st\_acc\_account\_client\_relation, st\_branch\_common, st\_account\_client\_extension, st\_acc\_account\_balance\_dto. Текст запроса представлен в файле **st\_account\_dto.sql**:

stmt.st\_account\_client\_extension – представление, в составе которой GROUP BY, поэтому сначала данные выбираются из таблиц stmt.st\_acc\_account\_client\_relation, stmt.st\_cln\_client\_common\_dto, stmt.st\_cln\_client\_master. Затем они группируются и только потом соединяются с основным множеством.

Если конструкция **group by** является избыточной, то тогда соединение с таблицей stmt.st\_acc\_account\_client\_relation по полю account\_id может быть осуществлено с помощью индекса. Причина избыточности group by описана в пункте 7.

1. Обнаружен запрос удаления данных из таблицы stmt.st\_contractor\_interaction с использованием данных таблицы stmt.st\_account\_statement. Является одним из лидеров по чтению блоков из буферного кэша.

Текст запроса представлен в файле **st\_account\_statement\_contract\_interraction\_del.sql**.

Для ускорения работы запроса предлагается использовать следующие индексы:

**CREATE** **INDEX** stci\_ix

**ON** stmt**.**st\_contractor\_interaction**(**client\_account\_id**,** statement\_date**);**

**CREATE** **UNIQUE** **INDEX** st\_acc\_st\_st\_id\_acc\_id\_st\_dt\_ux

**ON** stmt**.**st\_account\_statement**(**statement\_id**,** account\_id**,** statement\_date**);**

**Листинг 4.16. Индексы для ускорения работы запроса st\_account\_statement\_contract\_interraction\_del.sql**

1. Обнаружены запросы к таблице **st\_notification\_account\_change**, тексты представлены в файле **st\_notification\_account\_change.sql.** Ниже приведена таблица с временами выполнения:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № запроса | Минимальное время выполнения в мс | Максимальное время выполнения в мс | Среднее время выполнения в мс | Количество выполнений в час |
| 1 | 9280.167 | 9914.388 | 9547.078 | 18 |
| 2 | 6625.650 | 17974.333 | 7824.886 | 10 |

**Таблица 4.2. Времена выполнения запросов к таблице** **st\_notification\_account\_change**

В данной таблице отсутствует индекс, в котором поле client\_id является лидирующим, поэтому предлагается его создать:

**CREATE** **INDEX** stnac\_cl\_id\_status\_ix

**ON** stmt**.**st\_notification\_account\_change**(**client\_id**,** status**);**

**Листинг 4.17. Индекс с лидирующим полем client\_id**

## **Дубли индексов**

В ходе анализа схемы данных обнаружены многочисленные дубликаты индексов, что замедляет операции изменения, добавления и удаления данных. Список представлен в **ib\_stmt\_dupl\_idx.csv**. Их нужно будет удалить в технологическое окно

## **Рекомендации по конфигурации параметров операционной системы**

### **Настройка HugePages**

Анализ значений файла /proc/meminfo от 01.09.2022 показал, что 9.2GB системы тратится на PageTables, т.е, на таблицы соответствия страниц виртуальной и физической памяти:

PageTables: 2955148 kB

**Листинг 6.1.1. Значение поля PageTables**

Для уменьшения потребления памяти рекомендуется активировать HugePages согласно следующей формуле:

num\_pages = 1.1 **\*** shared\_buffers\_in\_kb **/** hugepage\_size

**Листинг 6.1.2. Формула расчёта количества HugePages**

**shared\_buffers\_in\_kb** – объём памяти для буферного кэша СУБД PostgreSQL в килобайтах

**hugepage\_size** = 2048kB или 1GB. В данном случае, 2048kB

Таким образом, значение для num\_pages при **shared\_buffers = 128GB** равно 72090

Для активации HugePages после перезапуска системы необходимо сохранить вычисленное значение в файле **/etc/systcl.conf** как **vm.nr\_hugepages = num\_pages,** затем применить sysctl -p /ect/sysctl.conf.

Стоит отметить, что HugePages не выгружаются в swap

### **Настройка Transparent Hugepage**

Анализ значений файла /proc/meminfo от 01.09.2022 показал использование Transparent Hugepage.

AnonHugePages**:** 36864 kB

**Листинг 6.2.1. Значение поля AnonHugePages**

Для отключения THP нужно выполнить следующие команды пользователем root:

echo never **>** **/**sys**/**kernel**/**mm**/**transparent\_hugepage**/**enabled

echo never **>** **/**sys**/**kernel**/**mm**/**transparent\_hugepage**/**defrag

**Листинг 6.2.2. Команды отключения THP**

Также для отключения THP необходимо добавить **transparent\_hugepage=never** в конец опции GRUB\_CMDLINE\_LINUX в файл /etc/default/grub.

После этого от пользователя root нужно выполнить:

grub2-mkconfig -o /boot/grub2/grub.cfg

Это позволит сохранить настройку даже после перезагрузки сервера.

Далее нужно перезапустить СУБД PostgreSQL для последующего использования выделенных HugePages.

## **Поиск разросшихся таблиц и индексов**

Разрастание таблиц и индексов приводит к замедлению скорости поиска данных в них и, как следствие, к замедлению скорости всей системы.

Кроме того, избыточное пространство в индексах замедляет работу процесса автоматической очистки, поскольку последнему нужно пройти каждый блок каждого индекса таблицы.

Для поиска таких объектов использовались запросы **bloated\_tables.sql** и **bloated\_indexes.sql из директории sql**.

Таблицы с процентом возрастания, большим 50%, не обнаружено. Так, в частности, процент разрастания таблицы **st\_aoperation** всего 5, при этом её размер 1633GB.

Однако необходимо проработать механизм её секционирования, поскольку количество строк в ней 1907115136, что, в свою очередь, может привести:

1. К замедлению создания и перестроения индексов.
2. К снижению точности статистики для планировщика.
3. К замедлению работы процесса автоматической очистки.
4. К замедлению скорости работы запросов.

Ниже приведена статистика по проценту разрастания индексов:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Схема | Таблица | Индекс | Размер | Избыточный размер | Процент избыточности |
| stmt | st\_aoperation | st\_aoperation\_init\_ref\_idx | 47 GB | 25 GB | 54.11 |
| stmt | st\_astatement | st\_astatement\_acc\_bic\_ndate\_dact\_idx | 2408 MB | 1266 MB | 52.57 |
| stmt | st\_astatement | st\_astatement\_account\_id\_normalized\_date\_idx | 1056 MB | 527 MB | 49.9 |
| stmt | st\_account\_statement | st\_account\_statement\_asys\_idx | 477 MB | 216 MB | 45.2 |
| stmt | st\_astatement | st\_astatement\_rid\_idx | 1950 MB | 655 MB | 33.5 |
| stmt | st\_astatement | st\_astatement\_req\_id\_un | 1950 MB | 654 MB | 33.5 |
| stmt | st\_aoperation | st\_aoperation\_st\_id\_idx | 54 GB | 18 GB | 33.48 |
| stmt | st\_aoperation | st\_aoperation\_saved\_doc\_ref\_idx | 74 GB | 24 GB | 32.80 |
| stmt | st\_aoperation | st\_aoperation\_ndd\_idx | 51 GB | 15 GB | 29.8 |

**Таблица 7.1. Список индексов с наибольшим процентом избыточного места**

Рекомендуется перестроить указанные выше индексы в период минимальной нагрузки на систему, например, с помощью pg\_repack. Также рекомендуется проработать механизм разделения таблицы **st\_aoperation** на секции

## **Поиск large object**

На пустой схеме определить некорректное использование large object невозможно, это нужно делать в период минимальной нагрузки следующим образом:

1. Найти поля типа text
2. Найти значения полей из шага 1, в которых хранятся только цифры.
3. Сверить значения из шага 2 с полем loid таблицы pg\_largeobject
4. Проверить по коду приложения использование LargeObjectManager для принятия решения о корректности использования типа large object.

Нужно помнить, что тип поля для large object должен быть oid или loid

## **Конфигурация СУБД PostgreSQL**

### **Настройка фонового процесса записи**

В таблице ниже приведены текущие и рекомендуемые настройки конфигурации фонового процесса записи от 01.09.2022 исходя из текущих серверных мощностей.

|  |  |  |
| --- | --- | --- |
| Параметр | Текущее значение | Рекомендуемое значение |
| bgwriter\_lru\_maxpages | 100 | 4000 |
| bgwriter\_lru\_multiplier | 2 | 4 |

**Таблица 9.1.1. Текущие и рекомендуемые настройки конфигурации фонового процесса записи**

### **Настройка процесса автоматической очистки**

В таблице ниже приведены текущие и рекомендуемые настройки конфигурации процесса автоматической очистки от 01.09.2022 исходя из текущих серверных мощностей.

|  |  |  |
| --- | --- | --- |
| Параметр | Текущее значение | Рекомендуемое значение |
| autovacuum\_naptime | 1min | 30s |
| autovacuum\_vacuum\_cost\_limit | -1 | 3200 |

**Таблица 9.2.1. Текущие и рекомендуемые настройки конфигурации процесса автоматической очистки**

### **Дополнительные параметры по настройке СУБД**

Ниже приведены значения параметров СУБД от 01.09.2022 и рекомендации по их изменению.

|  |  |  |
| --- | --- | --- |
| Параметр | Текущее значение | Рекомендуемое значение |
| default\_statistics\_target | 100 | 800 |
| join\_collapse\_limit | 8 | 20 |
| from\_collapse\_limit | 8 | 20 |

**Таблица 9.3.1.** **Текущие и рекомендуемые параметры работы планировщика**