УТВЕРЖДАЮ

Генеральный директор ООО «ППГ Разработка»

\_\_\_\_\_\_\_\_\_\_\_ В.В. Попов

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

Отчет

по результатам аудита состояния БД ib кластера

СУБД PostgreSQL

Версия №1 от 05.10.2022г.

на 22 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-pg5005lp.region.vtb.ru** с СУБД PostgreSQL, являющегося мастером от 05.10.2022:

|  |  |
| --- | --- |
| **Сервер** | **p0dbop-pg5005lp.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 16M 252G 1% /dev/shm  tmpfs tmpfs 252G 2.5G 250G 1% /run  tmpfs tmpfs 252G 0 252G 0% /sys/fs/cgroup  /dev/mapper/vg\_01-lv\_root xfs 10G 3.3G 6.8G 33% /  /dev/sda2 xfs 506M 183M 323M 37% /boot  /dev/sda1 vfat 200M 9.9M 190M 5% /boot/efi  /dev/mapper/vg\_p0dbop\_pg5005lp\_pgdata-lv\_pgdata xfs 4.0T 2.0T 2.0T 51% /pg\_data  /dev/mapper/vg\_p0dbop\_pg5005lp\_pgbackup-lv\_pgbackup xfs 4.0T 2.1T 1.9T 54% /pg\_backup  /dev/mapper/vg\_01-lv\_home xfs 5.0G 38M 5.0G 1% /home  /dev/mapper/vg\_01-lv\_var xfs 8.0G 1.9G 6.2G 23% /var  /dev/mapper/vg\_01-lv\_pgaudit xfs 200G 3.1G 197G 2% /pg\_audit  /dev/mapper/vg\_01-lv\_etcd xfs 15G 403M 15G 3% /app/etcd  /dev/mapper/vg\_01-lv\_tmp xfs 3.0G 33M 3.0G 2% /tmp  tmpfs tmpfs 51G 0 51G 0% /run/user/653  /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-pg5005lp.region.vtb.ru**

### **Анализ используемых программных средств**

СУБД работает под управлением ПО patroni 2.2.2, в таблице ниже представлены используемая на сервере версия от 01.09.2022:

|  |  |
| --- | --- |
| Сервер | Версия |
| Мастер  (p0dbop-pg5005lp.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**

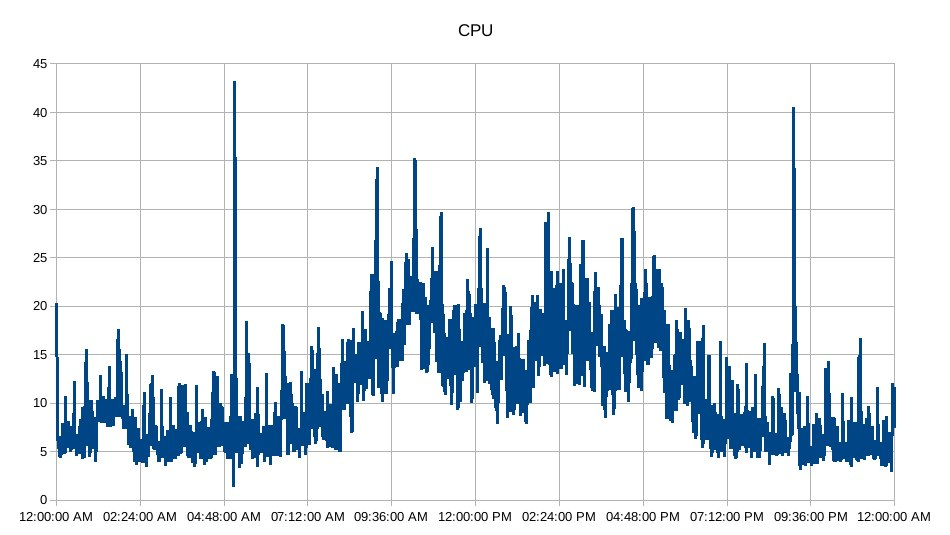
Выдан ряд рекомендаций по:

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 таблице **migrc\_process\_continue\_signal**, максимальное время работы которого 260 мс, за час он был выполнен 80677 раз, среднее время выполнения 28.837 мс. Размер таблицы 23MB, текст запроса представлен в файле **migr\_process\_continue\_signal.sql**.

В предложении ORDER BY используется функция random(), из-за чего запрос может вернуть разный набор данных.

Есть предположение, что в некоторых случаях желательно выполнять поиск по process\_id. В настоящий момент нет индекса, в котором это поле было лидирующим. Поэтому для оптимизации запроса предлагается создать следующий индекс:

**CREATE** **INDEX** continue\_signal\_process\_instance\_idx

**ON** migrc**.**migrc\_process\_continue\_signal**(**process\_id**,** instance**);**

**Листинг 4.1. Индекс для таблицы migrc\_process\_continue\_signal**

1. Обнаружены запросы к таблице **salary\_empl.empl\_salary\_employee**, Тексты представлены в файле **empl\_salary\_employee.sql**

Максимальное время работы 1001.292 мс, среднее время работы 308.189 мс, за час количество выполнений 7865 раз

В них периодически встречаются следующие условия фильтрации.

employee0\_**.**client\_id **in** **(**$1**,** $2**,** $3**,** $4**,** $5**,** $6**)** **and** employee0\_**.**client\_id**=**$7

**Листинг 4.2. Условия фильтрации по полю client\_id**

Скорее всего, первое условие фильтрации является избыточным при наличии второго.

Отсутствуют индексы по полям client\_id и employee\_id, что приводит к последовательному сканированию таблицы **salary\_empl.empl\_salary\_employee** размером 733MB.

Для ускорения работы запросов предлагается создать следующие индексы, они позволят снизить нагрузку на CPU и дисковую подсистему:

**CREATE** **INDEX** empl\_salary\_employee\_client\_id\_surname\_first\_name\_patronymic\_ix

**ON** salary\_empl**.**empl\_salary\_employee**(**client\_id**,** surname**,** first\_name**,** patronymic**);**

**CREATE** **UNIQUE** **INDEX** empl\_identity\_card\_emp\_id\_main\_doc\_ux

**ON** salary\_empl**.**empl\_identity\_card**(**employee\_id**,** main\_document**,** id**);**

**Листинг 4.3. Индексы для таблицы empl\_salary\_employee**

1. Обнаружены запросы к таблице **account.acc\_account** по полям **account\_state\_id** и **branch\_id**. По ним отсутствуют индексы, что приводит к последовательному сканированию таблицы размером 58MB. Тексты запросов представлены в файле acc\_account.sql

Минимальное время выполнения 694.760 мс, максимальное время выполнения 878.954 мс, среднее время выполнения 772.493 мс, за час запрос выполнился 589 раз.

Для обеспечения индексного доступа к данным предлагается создать следующие объекты:

**CREATE** **INDEX** account\_account\_state\_id\_idx

**ON** account**.**acc\_account**(**account\_state\_id**,** is\_deleted**,** visible\_for\_dbo**,** third\_party\_account**);**

**CREATE** **INDEX** account\_branch\_id\_account\_state\_id\_idx

**ON** account**.**acc\_account**(**branch\_id**,** account\_state\_id**,** is\_deleted**,** visible\_for\_dbo**,** third\_party\_account**);**

**Листинг 4.4. Индексы для таблицы acc\_account**

1. В описании таблицы acc\_account обнаружен индекс, у которого все поля преобразованы в тип text и используются для поиска по триграммам.

"acc\_account\_extension\_fts\_trigram\_index" gist **((**id**::**text**)** extensions**.**gist\_trgm\_ops**,** short\_name extensions**.**gist\_trgm\_ops**,** inn extensions**.**gist\_trgm\_ops**)**

**Листинг 4.5. Индекс для поиска по триграммам**

Для создания индексов, в состав которых входят текстовые и не текстовые поля, можно использовать расширение **btree\_gin**.

**CREATE** **INDEX** acc\_account\_fts\_trigram\_index

**ON** account**.**acc\_account

**USING** GIST **(**id**,** account\_number extensions**.**gist\_trgm\_ops**,** cur\_numeric\_code extensions**.**gist\_trgm\_ops**,**

cur\_alphabetic\_code extensions**.**gist\_trgm\_ops**,** okv\_name extensions**.**gist\_trgm\_ops**,**

account\_active\_passive extensions**.**gist\_trgm\_ops**,**

account\_name extensions**.**gist\_trgm\_ops**,** bic extensions**.**gist\_trgm\_ops**);**

**Листинг 4.6. Пример создания btree\_gin индекса**

1. Обнаружены запросы к таблице **ec\_cti\_currency\_transactions\_info**, в состав которых входят подзапросы к таблицам **ec\_cti\_acc\_account\_user\_name\_snapshot** и **ec\_cti\_edoc\_ref.** Тексты запросов можно найти в файле **ec\_cti\_currency\_transactions\_info.sql.**

Максимальное время выполнения 105.770 мс, минимальное 17.148 мс, среднее время выполнения 44.796 мс, за час запросы выполнились 18728 раз.

Выяснено, что эти подзапросы выполняются последовательным сканированием, что значительно увеличивает продолжительность работы основного запроса, поскольку подзапросы выполняются для каждой строки. Размеры таблиц **ec\_cti\_acc\_account\_user\_name\_snapshot** и **ec\_cti\_edoc\_ref** 4MB и 33MB соответственно.

Для ускорения работы предлагается создать следующие индексы:

**CREATE** **INDEX** ec\_cti\_auns\_account\_user\_id\_is\_deleted\_ix

**ON** ec\_cti**.**ec\_cti\_acc\_account\_user\_name\_snapshot**(**account\_id**,** **user\_id,** is\_deleted**)**

**INCLUDE(**account\_user\_name**);**

**CREATE** **INDEX** ec\_cti\_edoc\_ref\_edoc\_branch\_snapshot\_id\_ix

**ON** ec\_cti**.**ec\_cti\_edoc\_ref**(**edoc\_id**,** branch\_snapshot\_id**);**

**CREATE** **INDEX** ec\_cti\_edoc\_ref\_edoc\_client\_snapshot\_id\_ix

**ON** ec\_cti**.**ec\_cti\_edoc\_ref**(**edoc\_id**,** client\_snapshot\_id**);**

**Листинг 4.7. Индексы для ускорения работы подзапросов**

1. Обнаружен запрос к таблице **salary\_regs.regs\_salary\_register\_for\_salary**, минимальное время выполнения которого 271.623 мс, а максимальное 1152.256 мс. Текст запроса можно найти в файле **regs\_salary\_register\_for\_salary.sql**. Размер таблицы **salary\_regs.regs.salary\_register\_for\_salary** 1074MB.

Причина заключается в отсутствии индекса по полям **abs\_client\_id, file\_name, document\_date, delete\_date,** что приводит к последовательному сканированию. Создание индекса позволит ускорить время выполнения запроса, ниже приведён листинг.

**CREATE** **INDEX** salary\_regs**.**rsrfs\_abs\_cl\_id\_file\_name\_doc\_del\_date\_ix

**ON** salary\_regs**.**regs\_salary\_register\_for\_salary**(**abs\_client\_id**,** **file\_name,** document\_date**,** delete\_date**);**

**Листинг 4.8. Индекс для ускорения запроса к таблице** **regs\_salary\_register\_for\_salary**

1. Обнаружен запрос к таблице ac\_actor, в ходе работы которого читается большое количество блоков, размер таблицы 281MB. Текст можно найти в **ac\_actor.sql**

Минимальное время время выполнения запроса 20 мс, максимальное время выполнения 200 мс, за час количество выполнений 17806.

Найдены дубликаты условий соединения:

actorroles1\_**.**role\_instance\_id**=**roleinstan2\_**.**id **and** **(**actorroles1\_**.**role\_instance\_id**=**roleinstan2\_**.**id**)**

**Листинг 4.9. Дубликаты условий соединения**

Также обнаружены следующие частичные индексы, но поскольку поле id является первичным ключом, то смысла в таких индексах нет.

"ac\_actor\_id\_need\_repl\_idx" btree **(**id**)** **WHERE** need\_replication **=** **true**

"ac\_actor\_id\_not\_need\_repl\_idx" btree **(**id**)** **WHERE** need\_replication **=** **false**

**Листинг 4.10. Частичные индексы по полю id таблицы ac\_ac\_actor**

Для ускорения доступа к таблице ac\_actor предлагается вместо индекса ac\_actor\_user\_id\_idx создать следующий индекс:

**CREATE** **INDEX** ac\_actor\_user\_id\_block\_type\_ix **ON** ac**.**ac\_actor**(user\_id,** block\_type**);**

**Листинг 4.11. Индекс для ускорения доступа к данным таблицы ac.ac\_actor**

1. Обнаружен запрос с последовательным сканированием к таблице **erm\_cti\_attachment,** текст которого приведён в файле **erm\_cti\_attachment**.**sql**. Размер таблицы 27MB

Минимальное время выполнения запроса 14.269 мс, максимальное время выполнения 688.072 мс, среднее время выполнения 18.9 мс, за час количество выполнений 31127.

Для ускорения доступа к данным таблицы **erm\_cti\_attachment** предлагается создать следующий индекс:

**CREATE** **INDEX** erm\_cti\_attachment\_currency\_operation\_id\_ix **ON** erm\_cti**.**erm\_cti\_attachment**(**currency\_operation\_id**);**

**Листинг 4.12. Индекс для ускорения доступа к данным таблицы erm\_cti.erm\_cti\_attachment**

1. Обнаружен запрос с последовательным сканированием к таблице pc.pc\_collection\_outgoing\_payment\_claim, текст которого представлен в pc\_collection\_outgoing\_payment\_claim.sql. Размер таблицы 574 MB

Минимальное время выполнения 172.221 мс, максимальное время 454.092 мс, среднее время выполнения 215.930 мс, количество выполнений за час 6213.

Для ускорения доступа к данным предлагается создать следующий индекс:

**CREATE** **INDEX** pcopc\_cdre\_dd\_ix **ON** pc**.**pc\_collection\_outgoing\_payment\_claim**(**cabs\_doc\_ref\_ext**,** delete\_date**);**

**Листинг 4.13. Индекс для ускорения доступа к данным таблицы pc\_collection\_outgoing\_payment\_claim**

1. Обнаружен запрос с последовательным сканированием таблицы erm\_cti.erm\_cti\_edvk\_export, текст представлен в файле erm\_cti\_edvk\_export.sql. Размер таблицы 104 MB

Минимальное время выполнения запроса 31.076 мс, максимальное время выполнения 621.377 мс, среднее время выполнения 57 мс, количество выполнений за час 7418.

Для ускорения доступа к данным предлагается создать следующий индекс:

**CREATE** **INDEX** erm\_cti\_edvk\_export\_req\_id\_ix

**ON** erm\_cti**.**erm\_cti\_edvk\_export**(**request\_id**);**

**Листинг 4.14. Индекс для ускорения доступа к данным таблицы**

1. Обнаружен запрос с последовательным сканированием таблицы ermcdr\_conf\_doc\_ref\_asvcb\_event, текст представлен в файле ermcdr\_conf\_doc\_ref\_asvcb\_event.sql.

Минимальное время выполнения запроса 8.368 мс, максимальное 73.814 мс, среднее время 17.7 мс, количество выполнений за час 58941, размер таблицы 69MB

Для ускорения доступа к данным предлагается создать следующий индекс:

**CREATE** **INDEX** ecdrae\_conf\_doc\_ref\_id\_ix

**ON** erm\_docs\_ref**.**ermcdr\_conf\_doc\_ref\_asvcb\_event**(**conf\_doc\_ref\_id**);**

**Листинг 4.15. Индекс для ускорения доступа к таблице erm\_cti\_edvk\_export**

1. Обнаружен запрос с последовательным сканированием таблицы pc.pc\_batch\_doc, текст представлен в файле pc\_batch\_doc.sql, Размер таблицы 1MB.

Минимальное время выполнения 4 мс, максимальное время выполнения 33 мс, количество выполнений за час 16901

Для ускорения доступа к данным предлагается создать следующий индекс:

**CREATE** **UNIQUE** **INDEX** pc\_batch\_doc\_batch\_id\_doc\_pushed\_status\_id\_ix

**ON** pc**.**pc\_batch\_doc**(**batch\_id**,** doc\_pushed\_status**,** id**);**

**Листинг 4.16. Индекс для ускорения доступа к таблице pc\_batch\_doc**

1. Обнаружен запрос с последовательным сканированием таблицы currc\_currency\_contract, текст представлен в файле currc\_currency\_contract.sql. Размер таблицы 62 MB.

Минимальное время выполнения запроса 20.274 мс, максимальное время выполнения 80.603 мс, среднее время 20.274 мс.

Для ускорения доступа к данным предлагается создать следующий индекс:

**CREATE** **INDEX** currc\_cur\_contract\_fucn\_ix

**ON** ec\_currctr**.**currc\_currency\_contract**(**full\_unique\_contract\_number**,** master\_id**,** status**);**

**Листинг 4.17. Индекс для ускорения доступа к таблице currc\_currency\_contract**

1. Обнаружен запрос к таблицам cc\_subscription, cc\_account и cc\_cln\_client\_informational\_id\_snapshot, текст представлен в файле corp\_card.sql

Минимальное время выполнения 90782.383 мс, максимальное время выполнения 107139.493 мс, среднее время выполнения 95893.365 мс

Предположительно, данные таблиц извлекаются с помощью последовательного сканирования. Размеры таблиц приведены ниже:

1. corp\_card.cc\_cln\_client\_informational\_id\_snapshot - 24 MB
2. corp\_card.cc\_subscription - 56 kB
3. corp\_card.cc\_account - 21MB
4. corp\_card.cc\_cln\_client\_informational\_id\_snapshot - 24 MB

Для ускорения доступа к данным предлагается создать следующие индексы:

**CREATE** **INDEX** cc\_client\_inf\_id\_snapshot\_ix

**ON** corp\_card**.**cc\_cln\_client\_informational\_id\_snapshot**(**system**,** is\_deleted**);**

**CREATE** **INDEX** cc\_subscr\_client\_id\_ix

**ON** corp\_card**.**cc\_subscription**(**client\_id**);**

**CREATE** **INDEX** cc\_account\_client\_type\_stat\_id\_ix

**ON** corp\_card**.**cc\_account**(**client\_id**,** **type\_id,** state\_id**);**

**CREATE** **INDEX** cccc\_iis\_client\_id\_system\_is\_del\_ix

**ON** corp\_card**.**cc\_cln\_client\_informational\_id\_snapshot**(**client\_id**,** system**,** is\_deleted**);**

**Листинг 4.18. Индексы для ускорения запроса corp\_card.sql**

1. Обнаружен запрос с последовательным сканированием таблицы acc\_dclib\_jms\_requests\_history, текст представлен в файле acc\_dclib\_jms\_requests\_history.sql. Размер таблицы 8309 MB

Минимальное время выполнения запроса 2549.915 мс, максимальное время выполнения 5968.177 мс, среднее время выполнения 4642.146 мс, количество выполнений за час 105.

Для ускорения доступа к данным предлагается создать следующий индекс:

**CREATE** **UNIQUE** **INDEX** adjrh\_creation\_date\_correl\_id\_ix

**ON** account**.**acc\_dclib\_jms\_requests\_history**(**creation\_date**,** correl\_id**);**

**Листинг 4.19. Индекс для ускорения доступа к таблице acc\_dclib\_jms\_requests\_history**

Также стоит отметить, что запрос может возвращать разные строки из-за отсутствия ORDER BY

1. Обнаружен запрос следующего вида:

**select** confdoc\_at0\_**.**rev **as** col\_0\_0\_

**,** confdoc\_at0\_**.**conf\_doc\_id **as** col\_0\_1\_

**,** confdoc\_at0\_**.**id **as** col\_0\_2\_

**,** attachment1\_**.**id **as** col\_1\_0\_

**,** attachment1\_**.**rev **as** col\_1\_1\_

**from** ermcdr\_conf\_doc\_attachment\_aud confdoc\_at0\_

**cross** **join** ermcdr\_attachment\_aud attachment1\_

**where** confdoc\_at0\_**.**conf\_doc\_id**=**$1

**and** attachment1\_**.**rev**=(select** **max(**attachment2\_**.**rev**)**

**from** ermcdr\_attachment\_aud attachment2\_

**where** attachment2\_**.**rev**<=**$2

**and** attachment1\_**.**id**=**attachment2\_**.**id

**)**

**and** confdoc\_at0\_**.**rev**=(select** **max(**confdoc\_at3\_**.**rev**)**

**from** ermcdr\_conf\_doc\_attachment\_aud confdoc\_at3\_

**where** confdoc\_at3\_**.**rev**<=**$3

**and** confdoc\_at0\_**.**conf\_doc\_id**=**confdoc\_at3\_**.**conf\_doc\_id

**and** confdoc\_at0\_**.**id**=**confdoc\_at3\_**.**id

**)**

**and** confdoc\_at0\_**.**revtype**<>**$4

**and** attachment1\_**.**revtype**<>**$5

**and** **(**confdoc\_at0\_**.**id**=**attachment1\_**.**id **or**

confdoc\_at0\_**.**id **is** **null** **and** attachment1\_**.**id **is** **null)**

**Листинг 4.20. Запрос с cross join**

При cross join условие confdoc\_at0\_.id is NULL and attachment1\_.id is null не может быть истинным. Следовательно, cross join заменяется на inner join

Также подзапросы выполняются с использованием последовательного сканирования, что значительно увеличивает время выполнения основного запроса. Ниже приведены размеры таблиц:

1. ermcdr\_conf\_doc\_attachment\_aud - 5MB
2. ermcdr\_attachment\_aud - 18MB

Минимальное время выполнения запроса 7.186 мс, максимальное время выполнения 1886.646, среднее время выполнения 860.174 мс, количество выполнений за час 1823.

Для ускорения работы подзапросов предлагается создать следующие индексы:

**CREATE** **UNIQUE** **INDEX** ecda\_conf\_doc\_id\_ux

**ON** erm\_docs\_ref**.**ermcdr\_conf\_doc\_attachment\_aud**(**conf\_doc\_id**,** revtype**,** rev**,** id**);**

**CREATE** **UNIQUE** **INDEX** ecda\_id\_conf\_doc\_id\_rev\_ux

**ON** erm\_docs\_ref**.**ermcdr\_conf\_doc\_attachment\_aud**(**id**,** conf\_doc\_id**,** rev**);**

**Листинг 4.21. Индексы для ускорения работы подзапросов**

1. Обнаружен запрос с последовательным сканированием таблицы salary\_empl.empl\_identity\_card, текст представлен в empl\_identity\_card.sql. Размер таблицы 278 MB

Минимальное время выполнения запроса 26.7 мс, максимальное время выполнения 93.9 мс, среднее время выполнения 36.7 мс. За час запрос выполнился 10362 раза.

Для ускорения работы запроса предлагается использовать следующий индекс:

**CREATE** **INDEX** eic\_employee\_id\_ix **ON** salary\_empl**.**empl\_identity\_card**(**employee\_id**);**

**Листинг 4.22. Индекс для ускорения доступа к данным таблицы empl\_identity\_card**

1. Обнаружен запрос, при выполнении которого выполняется последовательное сканирование таблицы **rgacc.rgacc\_account\_link\_edoc\_param**. Текст представлен в файле **rgacc\_account\_link\_edoc\_param.sql**

Минимальное время выполнения запроса 9.31 мс, максимальное время выполнения 45.5 мс, количество выполнений за час 1165.

Для ускорения работы запроса предлагается создать следующий индекс:

**CREATE** **INDEX** rgaccalep\_account\_link\_id\_ix

**ON** rgacc**.**rgacc\_account\_link\_edoc\_param**(**account\_link\_id**);**

**Листинг 4.23. Индекс для ускорения доступа к таблице rgacc\_account\_link\_edoc\_param**

1. Обнаружен запрос с последовательным сканированием таблицы **mbo\_request\_stop**, текст представлен в **mbo\_request\_stop.sql**, размер таблицы 20 MB.

Минимальное время выполнения запроса 13.88 мс, максимальное время выполнения 51.18 мс, среднее время выполнения 20 мс.

Для ускорения работы запроса предлагается создать следующий индекс:

**CREATE** **INDEX** mrs\_request\_id\_ix **ON** migr\_bo**.**mbo\_request\_stop**(**request\_id**)**

**Листинг 4.24**. **Индекс для ускорения доступа к таблице** **mbo\_request\_stop**

1. Обнаружен запрос с последовательным сканированием таблицы migrc.migrc\_request. Текст представлен в файле migrc\_request.sql, размер таблицы 37 GB

Минимальное время выполнения запроса 65.29 мс, максимальное время выполнения 204.20 мс, среднее время выполнения 102 мс, количество выполнений за час 3000.

Для ускорения работы запроса предлагается создать следующий индекс:

**CREATE** **INDEX** mr\_cr\_dt\_status\_ix

**ON** migrc**.**migrc\_request**(**create\_date**,** status**);**

**Листинг 4.25**. **Индекс для ускорения доступа к таблице** **migrc\_request**

1. Обнаружен неоптимальный запрос поиска полей первичного ключа, текст запроса представлен в файле find\_pk\_cols.sql.

Минимальное время выполнения 70.766 мс, максимальное время выполнения 426.277, среднее время выполнения 76.353 мс, за час запрос выполнился 12325 раз

Предлагается заменить его на запрос find\_pk\_cols\_new.sql, его среднее время выполнения 0.083 мс.

1. Обнаружен неоптимальный запрос поиска данных последовательным сканированием по полю **ext\_user\_info\_master\_record** таблицы **eus\_ext\_user\_info\_mapping**. Текст запроса представлен в **eus\_ext\_user\_info\_mapping.sql**

Минимальное время выполнения запроса 2 мс, максимальное время выполнения 663 мс, среднее время выполнения 20 мс.

Для ускорения доступа к данным предлагается создать следующий индекс

**CREATE** **INDEX** eeuimrid\_ix

**ON** eus**.**eus\_ext\_user\_info\_mapping**(**ext\_user\_info\_master\_record\_id**);**

**Листинг 4.26. Индекс для ускорения доступа к таблице eus\_ext\_user\_info\_mapping**

1. Обнаружен неоптимальный запрос с последовательным сканированием данных таблицы **ec\_cti\_attachment** по полю **currency\_operation\_id**. Текст запроса представлен в файле **ec\_cti\_attachment.sql**

Минимальное время выполнения запроса 6.2 мс, максимальное время выполнения 23.5 мс, среднее время выполнения 10 мс, за час запрос выполнился 42296 раз.

Для ускорения доступа к данным предлагается создать следующий индекс:

**CREATE** **INDEX** eca\_curr\_op\_id\_ix

**ON** ec\_cti**.**ec\_cti\_attachment**(**currency\_operation\_id**);**

**Листинг 4.27. Индекс для ускорения доступа к таблице ec\_cti\_attachment**

## **Дубли индексов**

В ходе анализа схемы данных обнаружены многочисленные дубликаты индексов, что замедляет операции изменения, добавления и удаления данных. Список представлен в **ib\_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: 9264616 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 = 125GB** равно 64000

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.

## **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**.

Ниже приведена статистика по проценту избыточного места таблиц и индексов:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Схема | Таблица | Количество строк | Размер | Избыточный размер | Процент избыточности |
| kpi | kpi\_dm\_metrics\_20211201 | 12743 | 121 MB | 119 MB | 98.82 |
| vktr | vktr\_contract\_registration\_request\_aud | 27546 | 290 MB | 241 MB | 83.123 |
| rup\_oc | rup\_oc\_ru\_payment\_snapshot | 2644098 | 14 GB | 9813 MB | 70.87 |
| rup\_oc | rup\_oc\_united\_scroller\_outbox\_entity | 2549671 | 13 GB | 8701 MB | 67.77 |
| rup\_oc | rup\_oc\_batch\_notification\_sms\_outbox\_entity | 132571 | 543 MB | 327 MB | 60.29 |
| fraud | frd\_process | 116771 | 1345 MB | 912 MB | 67.77 |
| ec\_cti | ec\_cti\_currency\_operation | 349698 | 172 MB | 103 MB | 59.63 |
| salary\_regc | regc\_history\_entry | 57604 | 298 MB | 176 MB | 58.96 |
| salary\_empl | empl\_salary\_employee | 2303418 | 733 MB | 345 MB | 47.09 |
| migrc | migrc\_sync\_process | 2746008 | 48 GB | 22 GB | 45.58 |

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Схема | Таблица | Индекс | Размер | Избыточный размер | Процент избыточности |
| rup\_oc | rup\_oc\_united\_scroller\_outbox\_entity | rup\_oc\_united\_scroller\_outbox\_entity\_pk | 1420 MB | 1293 MB | 91 |
| migrc | migrc\_meta\_data | bg\_meta\_index | 5820 MB | 4428 MB | 76.08 |
| migrc | migrc\_meta\_data | meta\_data\_log\_id\_idx | 1421 MB | 1080 MB | 76.01 |
| migrc | migrc\_meta\_data | task\_unit\_root\_entity\_idx | 9978 MB | 6969 MB | 69.8 |
| migrc | migrc\_request | request\_create\_date\_idx | 2433 MB | 1636 MB | 67.22 |
| migrc | migrc\_meta\_data | source\_entity\_idx | 2821 MB | 1822 MB | 64.6 |
| ntn\_migrated | ntn\_event | ntn\_event\_announcement\_id\_idx | 2056 MB | 1308 MB | 63.6 |
| salary\_regs | regs\_employee\_payroll | regs\_employee\_payroll\_pk | 1247 MB | 784 MB | 62.84 |
| ac | ac\_permission | ac\_permission\_expression\_id\_idx | 1992 MB | 1251 MB | 62.84 |
| ntn\_migrated | ntn\_message | ntn\_message\_jms\_message\_id\_idx | 3362 MB | 2005 MB | 59.64 |
| ntn\_migrated | ntn\_message | ntn\_index\_message\_user\_id\_expiration\_time\_notification\_idx | 9301 MB | 5527 MB | 59.4 |
| migrc | migrc\_meta\_data | migrc\_meta\_data\_pk | 3561 MB | 2114 MB | 59.37 |
| ltrs | ltrs\_access | ltrs\_access\_stage\_role\_edocref\_uk | 50 GB | 29 GB | 57.89 |
| ntn\_migrated | ntn\_addressee | ntn\_addressee\_pk | 2893 MB | 1588 MB | 54.89 |
| ntn\_migrated | ntn\_message | ntn\_message\_pk | 3045 MB | 1664 MB | 54.66 |
| ntn\_migrated | ntn\_event | ntn\_event\_pk | 1674 MB | 913 MB | 54.52 |
| ntn\_migrated | ntn\_message | ntn\_msg\_evgr\_fk\_idx | 4234 MB | 2303 MB | 54.40 |
| ltrs | ltrs\_access | ltrs\_access\_id\_pk | 24 GB | 12 GB | 51.03 |

**Таблица 7.2. Список индексов с наибольшим процентом избыточного места**

Рекомендуется перестроить указанные выше таблицы и индексы в период минимальной нагрузки на систему, например, с помощью pg\_repack.

## **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**