УТВЕРЖДАЮ

Генеральный директор ООО «ППГ Разработка»

\_\_\_\_\_\_\_\_\_\_\_ В.В. Попов

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

Отчет

по результатам аудита состояния БД ib\_2 кластера

СУБД PostgreSQL

Версия №1 от 17.10.2022г.

на 21 листе

## **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** 12](#_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** 20](#_Toc13)

[9.  **Конфигурация СУБД PostgreSQL** 21](#_Toc14)

[9.1. **Настройка фонового процесса записи** 21](#_Toc15)

[9.2. **Настройка процесса автоматической очистки** 21](#_Toc16)

[9.3. **Дополнительные параметры по настройке СУБД** 21](#_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-pg5002lp.region.vtb.ru** с СУБД PostgreSQL, являющегося мастером от 01.09.2022:

|  |  |
| --- | --- |
| **Сервер** | **p0dbop-pg5002lp.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 23M 252G 1% **/**dev**/**shm  tmpfs tmpfs 252G 2.2G 250G 1% **/**run  tmpfs tmpfs 252G 0 252G 0% **/**sys**/**fs**/**cgroup  **/**dev**/**mapper**/**vg\_01**-**lv\_root xfs 10G 3.2G 6.9G 32% **/**  **/**dev**/**sda2 xfs 506M 183M 323M 37% **/**boot  **/**dev**/**sda1 vfat 200M 9.9M 190M 5% **/**boot**/**efi  **/**dev**/**mapper**/**vg\_p0dbop\_pg5002lp\_pgdata**-**lv\_pgdata xfs 4.0T 359G 3.6T 9% **/**pg\_data  **/**dev**/**mapper**/**vg\_p0dbop\_pg5002lp\_pgbackup**-**lv\_pgbackup xfs 5.9T 345G 5.6T 6% **/**pg\_backup  **/**dev**/**mapper**/**vg\_01**-**lv\_pgaudit xfs 200G 546M 200G 1% **/**pg\_audit  **/**dev**/**mapper**/**vg\_01**-**lv\_etcd xfs 15G 467M 15G 4% **/**app**/**etcd  **/**dev**/**mapper**/**vg\_01**-**lv\_home xfs 5.0G 34M 5.0G 1% **/**home  **/**dev**/**mapper**/**vg\_01**-**lv\_tmp xfs 3.0G 34M 3.0G 2% **/**tmp  **/**dev**/**mapper**/**vg\_01**-**lv\_var xfs 8.0G 1.9G 6.2G 23% **/**var  tmpfs tmpfs 51G 0 51G 0% **/**run**/**user**/**653  **/**dev**/**mapper**/**vg\_pgwal**-**lv\_pgwal xfs 2.0T 27G 2.0T 2% **/**pg\_walarchive  tmpfs tmpfs 51G 0 51G 0% **/**run**/**user**/**1376548563 |
| Директория данных СУБД | /pg\_data |

**Таблица 2.1.1. Характеристики сервера** **p0dbop-pg5002lp.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-pg5002lp.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**

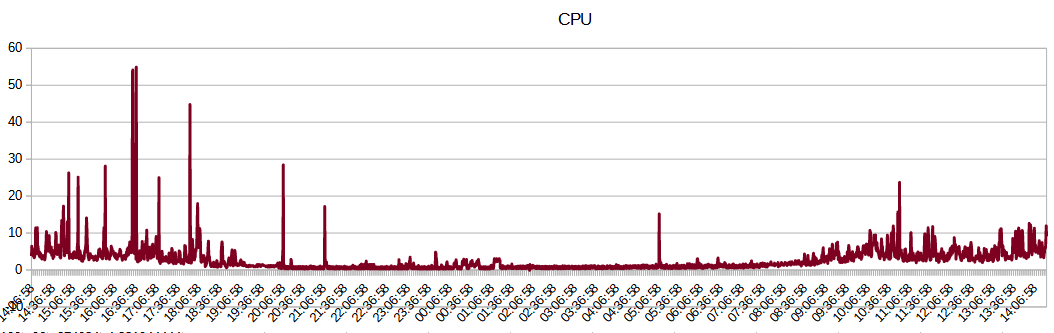
Выдан ряд рекомендаций по:

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 запросы к таблице **fxrate.fxrt\_bank\_rate**, максимальное время работы которого 163.033 мс, минимальное время выполнения 61.662 мс, среднее время выполнения 78.379 мс, за час он был выполнен 3053 раза.

Тексты запросов представлен в файле **sql**/**bankrate.sql**.

Для ускорения доступа к данным таблицы **fxrate.fxrt\_bank\_rate** предлагается создать следующие индексы:

**CREATE** **INDEX** fbnkr\_c1num\_code\_c2ac\_bedt\_rt\_t\_ix

**ON** fxrate**.**fxrt\_bank\_rate**(**cur1num\_code**,** cur2alpha\_code**,** begin\_dt**,** end\_dt**,** rate\_type**,** term**);**

**CREATE** **INDEX** fbnkr\_c2num\_code\_c1ac\_bedt\_rt\_t\_ix

**ON** fxrate**.**fxrt\_bank\_rate**(**cur2num\_code**,** cur1alpha\_code**,** begin\_dt**,** end\_dt**,** rate\_type**,** term**);**

**Листинг 4.1. Индексы для таблицы fxrate.fxrt\_bank\_rate**

1. Обнаружен запрос соединения таблиц **cln\_client** и **cln\_client\_master** по cross join, минимальное время выполнения 183.100 мс, максимальное 263.224 мс, среднее 204.136 мс, количество выполнений за час 401.

Текст запроса представлен в файле **sql/cl\_client\_master.sql**

cross join нужно заменить на join, вариант запроса представлен в файле **sql/proposals/cl\_client\_master.sql**

С 14-ой версии СУБД PostgreSQL улучшен механизм работы с большими списками значений.

1. Обнаружен запрос к таблицам **cln\_employee** и **cln\_client\_branch\_relation**, минимальное время выполнения 55.891 мс, максимальное время выполнения 115.138 мс, среднее время выполнения 68.349 мс, количество выполнений за час 8201

Текст представлен в файле **sql/cln\_employee.sql**

Предлагается заменить **in** на **exists**, а **like** на **ilike**, что позволит убрать вызов функции lower() для поля **full\_name** таблицы **cln.cln\_employee**. Данный вариант запроса представлен в файле:

**sql/proposals/cln\_employee.sql**

Предположительно, что поиск по полю **full\_name** обладает высокой селективностью, однако в индексе **cln\_employee\_fts\_trigram\_index** поле оно последнее. В gist-индексе порядок столбцов имеет значение. Поэтому предлагается создать gist-индекс по полю full\_name.

**CREATE** **INDEX** cln\_empl\_full\_name\_trg\_ix

**ON** cln**.**cln\_employee **USING** GIST**(**full\_name extensions**.**gist\_trgm\_ops**);**

**Листинг 4.2. Индекс по полю full\_name таблицы cln\_employee**

Если поиск по full\_name обладает высокой селективностью, то для таблицы cln\_client\_branch\_relation может понадобиться создать следующий индекс:

**CREATE** **UNIQUE** **INDEX** cbr\_client\_id\_branch\_id\_ux

**ON** cln**.**cln\_client\_branch\_relation**(**client\_id**,** branch\_id**);**

**Листинг 4.3. Индекс для таблицы cln\_client\_branch\_relation**

1. Обнаружены запросы поиска данных по таблицам **cln\_client**, **cln\_client\_master** и **cln\_client\_branch\_relation**. Ниже приведена таблица с временами выполнения:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № запроса | Минимальное время выполнения в мс | Максимальное время выполнения в мс | Среднее время выполнения в мс | Кол-во выполнений в час |
| 1 | 271.989 | 31753.902 | 3363.963 | 10 |
| 2 | 293.045 | 34057.228 | 15798.967 | 10 |
| 3 | 294.097 | 34969.397 | 14575.738 | 17 |

**Таблица 4.1. Времена выполнения запросов по триграммам.**

Тексты запросов представлены в файле **sql/cln\_ms.sql**

Для второго запроса cross join не нужен, его можно заменить на join.

Основной вопрос заключается в следующем фрагменте, поскольку сначала соединяются строки трёх таблиц, а затем фильтруются по ILIKE в узле Join Filter.

**select** client3\_**.**id

**from** cln\_client client3\_

**left** **join** cln\_client\_master clientmast4\_

**on** clientmast4\_**.**id **=** client3\_**.**client\_master\_id

--left outer join cln\_client\_master\_kpp fiscalreas5\_

--on fiscalreas5\_.client\_master\_id = clientmast4\_.id

**left** **join** cln\_client\_master\_kpp fiscalreas6\_

**on** fiscalreas6\_**.**client\_master\_id **=** clientmast4\_**.**id

**left** **join** cln\_client\_master clientmast7\_

**on** clientmast7\_**.**id **=** client3\_**.**client\_master\_id

**where** **(CAST(**client3\_**.**id **AS** TEXT**)** ILIKE $101 **OR**

**CAST(**client3\_**.**user\_short\_name **AS** TEXT**)** ILIKE $102 **OR**

**CAST(**clientmast7\_**.**id **AS** TEXT**)** ILIKE $103 **OR CAST(**clientmast7\_**.**full\_name **AS** TEXT**)** ILIKE $104 **OR**

**CAST(**clientmast7\_**.**international\_full\_name **AS** TEXT**)** ILIKE $105 **OR**

**CAST(**clientmast7\_**.**international\_name **AS** TEXT**)** ILIKE $106 **OR**

**CAST(**clientmast7\_**.**inn **AS** TEXT**)** ILIKE $107 **OR CAST(**clientmast7\_**.**ed\_author **AS** TEXT**)** ILIKE $108 **OR**

**CAST(**clientmast7\_**.**ogrn **AS** TEXT**)** ILIKE $109 **OR** **CAST(**clientmast7\_**.**short\_name **AS** TEXT**)** ILIKE $110 **OR**

**CAST(**fiscalreas6\_**.**kpp **AS** TEXT**)** ILIKE $111**)** **=** $112**;**

**Листинг 4.4. Фрагмент запроса поиска по ILIKE**

В этом запросе нужно понять, какое из условий фильтрации обладает большей селективностью. Если это поиск по триграммам, то предлагается рассмотреть возможность использования варианта запроса в **sql/proposals/cln\_ms.sql**

Кроме того, поиск по триграммам работает приемлемо до 4 млн записей, после этого время работы может увеличиться до 10 минут.

Таким образом, нужно повторно рассмотреть архитектуру модуля, в ходе работы которого вызывается данный запрос.

1. Обнаружены запросы к таблице **entry\_journal.ejrnl\_entry\_journal**, минимальное время выполнения 473.052 мс, максимальное время выполнения 511.072 мс, среднее время выполнения 485.667 мс, кол-во выполнений в час 120.

Текст запросов представлен в файле **sql/ejrnl\_entry\_journal.sql**

В таблице entry\_journal.ejrnl\_entry\_journal отсутствует индекс, в котором поле created\_at было первым. Поэтому предлагается создать следующие индексы:

**CREATE** **INDEX** ejrnl\_ej\_created\_at\_ix

**ON** entry\_journal**.**ejrnl\_entry\_journal**(**created\_at**,** certificate\_cn**,** success**);**

**CREATE** **INDEX** ejrnl\_ej\_created\_at\_ip\_addr\_success\_ix

**ON** entry\_journal**.**ejrnl\_entry\_journal**(**created\_at**,** ip\_address**,** success**);**

**Листинг 4.5. Индексы для оптимизации запросов к ejrnl\_entry\_journal по полю created\_at.**

1. Обнаружен запрос с последовательным сканированием таблицы fxrate.fxrt\_bank\_rate, минимальное время выполнения 72.882 мс, максимальное время 82.909 мс, среднее время 79.559 мс. За час запрос выполнился 86 раз.

Текст запроса представлен в файле **sql**/**fxrt\_br\_cdt\_ix.sql**

Для оптимизации поиска по полю change\_dt предлагается использовать следующий индекс:

**CREATE** **INDEX** fxrt\_br\_cdt\_ix

**ON** fxrate**.**fxrt\_bank\_rate**(**change\_dt**);**

**Листинг 4.6. Индекс для ускорения поиска по полю** change\_dt **таблицы fxrt\_bank\_rate**

1. Обнаружен запрос с последовательным сканированием таблицы ntn.ntn\_action\_event по полю **event\_id**. Минимальное время выполнения 10.885 мс, максимальное время выполнения 33.831 мс, среднее время выполнения 14.683 мс, кол-во выполнений за час 3665.

Текст запроса представлен в файле **sql/ntn\_action\_event.sql**

Предлагается создать индекс по полю event\_id.

**CREATE** **INDEX** nae\_event\_id\_ix

**ON** ntn**.**ntn\_action\_event**(**event\_id**);**

**Листинг 4.7. Индекс по полю event\_id таблицы ntn.ntn\_action\_event.sql**

1. Обнаружен запрос к данным таблицы **ntn.ntn\_event** по полям **expiration\_time** иprocessed.

Текст запроса представлен в файле **ntn\_event.sql**.

Для поля expiration\_time нет индекса, поэтому предлагается его создать:

**CREATE** **INDEX** ntn\_event\_exp\_time\_ix

**ON** ntn**.**ntn\_event**(**expiration\_time**,** processed**);**

**Листинг 4.8. Индекс по полям expiration\_time, processed таблицы ntn\_event**

1. Обнаружены запросы к таблице **ntn\_message**, тексты представлены в файле **sql/ntn\_message.sql**. Для части запросов среднее время выполнения запроса достигает 6846.419 мс и 3292.629 мс.

Предлагается:

1. Вместо индексов

ntn\_index\_message\_user\_id\_exp\_noti\_external\_idx, ntn\_index\_message\_user\_id\_expiration\_time\_news\_idx, ntn\_index\_message\_user\_id\_expiration\_time\_notification\_idx

создать следующий объект:

**CREATE** **INDEX** ntn\_uid\_et\_cid\_st\_cat\_ut\_ix

**ON** ntn**.**ntn\_message**(user\_id,** **COALESCE(**expiration\_time**,** 'infinity'**::**timestamp **with** time **zone),**

channel\_id**,** status**,** category**,** user\_type**);**

**Листинг 4.9. Замена трёх индексов**

1. Создать индексы, в которых поля created\_at и user\_id будут лидирующими

**CREATE** **INDEX** nm\_cat\_cid\_status\_ix

**ON** ntn**.**ntn\_message**(**created\_at**,** channel\_id**,** status**);**

**CREATE** **INDEX** ntn\_mes\_user\_id\_type\_ix

**ON** ntn**.**ntn\_message**(user\_id,** user\_type**);**

**Листинг 4.10. Индексы для ускорения поиска по полям created\_at и user\_id**

1. Обнаружены запрос с последовательным сканированием данных таблицы rndr.rndr\_background\_operations. Текст запроса представлен в файле **rndr\_background\_operations.sql**. Минимальное время выполнения запроса 42.354 мс, максимальное время выполнения 54.557 мс, среднее время выполнения 43.588 мс, кол-во выполнений в час 120.

Для ускорения доступа к данным предлагается создать следующий индекс:

**CREATE** **INDEX** rbo\_send\_status\_ix

**ON** rndr**.**rndr\_background\_operations**(**sending\_status**);**

**Листинг 4.11. Индекс для ускорения доступа к данным таблицы rndr\_background\_operations**

## **Дубли индексов**

В ходе анализа схемы данных обнаружены многочисленные дубликаты индексов, что замедляет операции изменения, добавления и удаления данных. Список представлен в **ib\_2\_dupl\_idx.csv**. Их нужно будет удалить в технологическое окно.

Также выявлены следующие особенности:

1. В таблице **rrko.rrko\_account\_balance\_snapshot** обнаружены два частичных индекса. Поскольку поле id является первичным ключом, то есть сомнения в том, что условие по is\_blocked позволит существенно ускорить выполнение запросов. Поэтому указанные ниже индексы предлагается удалить:

"rrko\_account\_balance\_snapshot\_id\_blocked\_idx" **UNIQUE,** btree **(**id**)** **WHERE** is\_blocked **=** **true**

"rrko\_account\_balance\_snapshot\_id\_not\_blocked\_idx" **UNIQUE,** btree **(**id**)** **WHERE** **NOT** is\_blocked **=** **true**

**Листинг 5.1. Частичные индексы таблицы rrko.rrko\_account\_balance\_snapshot**

1. В таблице **rrko.rrko\_activated\_model** предлагается убрать частичный индекс **rrko\_activated\_model\_uuid\_idx**.

Предлагается использовать индекс **activated\_model\_entity\_id\_idx,** поскольку его можно использовать при фильтрации данных по полям uuid, type\_id. А у частичного индекса как раз есть условие type\_id = 11. Ниже приведён листинг с описанием упомянутых индексов.

"activated\_model\_entity\_id\_idx" btree **(**uuid**,** **type\_id,** client\_ext\_id**)**

"rrko\_activated\_model\_uuid\_idx" btree **(**uuid**)** **WHERE** **type\_id** **=** 11

**Листинг 5.2. Полный и частичный индексы таблицы rrko.rrko\_activated\_model**

1. В таблице rrko.rrko\_cancel\_request обнаружены следующие индексы:

"rrko\_cancel\_request\_uuid\_key" **UNIQUE** **CONSTRAINT,** btree **(**uuid**)**

"cancel\_req\_entity\_id\_idx" btree **(**uuid**,** **type\_id,** client\_ext\_id**)**

"rrko\_cancel\_request\_uuid\_idx" btree **(**uuid**)**

**Листинг 5.3. Дубликаты индексов таблицы rrko.rrko\_cancel\_request**

Предлагается удалить индексы rrko\_cancel\_request\_uid\_key и rrko\_cancel\_request\_uuid\_idx, а индекс cancel\_req\_entity\_id\_idx сделать уникальным.

1. В таблице **rrko.rrko\_complementary\_signature** обнаружены следующие индексы:

"rrko\_complementary\_signature\_edoc\_ref\_id\_sign\_type" btree **(**edoc\_ref\_id**,** sign\_type**)**

"rrko\_complementary\_signature\_edoc\_ref\_id\_technical" btree **(**edoc\_ref\_id**)** **WHERE** sign\_type**::**text **=** 'TECHNICAL'

**Листинг 5.4. Полный и частичный индекс таблицы rrko.rrko\_complementary\_signature**

Предлагается убрать частичный индекс.

1. В таблице **rrko.rrko\_esignature** обнаружены следующие индексы:

"esignature\_entity\_id\_idx" btree **(**uuid**,** **type\_id,** client\_ext\_id**)**

"rrko\_esignature\_uuid\_idx" btree **(**uuid**)** **WHERE** **type\_id** **=** 11

**Листинг 5.5. Полный и частичный индексы таблицы** **rrko.rrko\_esignature**

Предлагается удалить частичный индекс.

1. В таблице **rrko.rrko\_history\_entry** обнаружены следующие индексы:

"rrko\_entity\_id\_type\_id\_history\_entry\_idx" btree **(**entity\_id**,** **type\_id)**

"rrko\_history\_entry\_entity\_id\_type\_id\_11\_idx" btree **(**entity\_id**)** **WHERE** **type\_id** **=** 11

**Листинг 5.6. Полный и частичный индексы таблицы** **rrko.rrko\_history\_entry**

Предлагается удалить частичный индекс.

1. Обнаружены частичные индексы в таблице **rrko\_ru\_payment,** листинг представлен в файле **sql/rrko\_ru\_payment\_idx.sql**

Нужно выяснить, позволяют ли условия в индексах отфильтровать большое количество строк. Для этого предлагается выполнить запрос из **sql/rrko\_ru\_payment\_idx\_filter\_stats.sql**

Если количества строк, удовлетворяющих условиям частичных индексов будут близки к количеству строк в таблице, то тогда их использование не позволит ускорить выполнение запросов.

В этом случае нужно определить корректный индексный состав, как с точки зрения улучшения скорости работы DML-операций, так и скорости работы запросов.

1. В таблице cln.cln\_employee обнаружены следующие индексы:

"cln\_employee\_pkey" **PRIMARY** **KEY,** btree **(**id**)**

"cln\_employee\_id" btree **(**id**)** **WHERE** is\_deleted **=** **false** **AND** user\_not\_found **IS** **NULL**

**Листинг 5.7. Полный и частичный индексы таблицы cln.cln\_employee**

Предлагается удалить частичный индекс.

Также в таблицы cln.cln\_employee обнаружены следующие индексы по полю external\_user

"cln\_employee\_external\_user\_doc\_id" btree **(**external\_user**)**

"cln\_employee\_external\_user\_idx" btree **(**external\_user**)**

"cln\_employee\_external\_user\_partial\_idx" btree **(**external\_user**)** **WHERE** is\_deleted **=** **false**

**Листинг 5.8. Дубликаты индексов по полю external\_user**

Предлагается их заменить следующим индексом:

**CREATE** **INDEX** cln\_employee\_external\_user\_is\_deleted\_ix

**ON** cln**.**cln\_employee**(**external\_user**,** is\_deleted**);**

**Листинг 5.9. Вариант замены индексов листинга 5.8**

1. В таблице **impr.impr\_import\_record** обнаружены частичные индексы, однако для оценки их эффективности предлагается выполнить следующий запрос:

**SELECT** **COUNT(\*)** **AS** cnt

**,** **COUNT(\*)** FILTER**(WHERE** i**.**record\_status **=** 'IMPORTED'**)** **AS** cnt1

**,** **COUNT(\*)** FILTER**(WHERE** i**.**record\_status **IN** **(**'IMPORTED'**,** 'IMPORT\_ERROR'**,** 'CANCELED'**,** 'CANCELED\_ERROR'**))** **AS** cnt2

**,** **COUNT(\*)** FILTER**(WHERE** i**.**record\_status **=** 'CHECKED'**)** **AS** cnt3

**FROM** impr**.**impr\_import\_record i**;**

**Листинг 5.10. Запрос оценки эффективности частичных индексов**

Если значения **cnt1**, **cnt2** и **cnt3** будут близки к значению cnt, значит, частичные индексы не являются эффективными с точки зрения скорости выполнения запросов.

В этом случае нужно будет определить корректный индексный состав.

1. В таблице nsi.nsi\_swift\_gpi\_status обнаружены следующие частичные индексы:

"nsi\_swift\_gpi\_code\_reason\_code\_uk1\_idx" **UNIQUE,** btree **(**code**,** reason\_code**)** **WHERE** reason\_code **IS** **NOT** **NULL**

"nsi\_swift\_gpi\_code\_reason\_code\_uk2\_idx" **UNIQUE,** btree **(**code**)** **WHERE** reason\_code **IS** **NULL**

**Листинг 5.11. Частичные индексы таблицы nsi.nsi\_swift\_gpi\_status**

Предлагается их заменить на следующий индекс:

**CREATE** **UNIQUE** **INDEX** nsi\_swift\_gpi\_code\_reason\_code\_uk\_idx

**ON** cln**.**cln\_employee**(**code**,** reason\_code**);**

**Листинг 5.12. Вариант замены индексов листинга 5.11**

1. В таблице nsi.nsi\_wbank2\_world\_bank2 обнаружены следующие индексы:

"nsi\_wbank2\_world\_bank2\_swiftcode\_status\_idx" **UNIQUE,** btree **(**swift\_code**,** status**)** **WHERE** status **=** 0

"nsi\_wbank2\_world\_bank2\_char\_code\_2\_idx" btree **(**swift\_code**)**

**Листинг 5.13. Полный и частичный индексы таблицы nsi.nsi\_wbank2\_world\_bank2**

Предлагается их заменить на следующее:

**CREATE** **INDEX** nsi\_wbank2\_world\_bank2\_swiftcode\_status\_idx

**ON** nsi**.**nsi\_wbank2\_world\_bank2**(**swift\_code**,** status**);**

**Листинг 5.14. Вариант замены индексов листинга 5.13**

1. В таблице **ntn.ntn\_message** обнаружены следующие индексы:

"ntn\_message\_pk" **PRIMARY** **KEY,** btree **(**id**)**

"ntn\_message\_id\_channel\_status\_partial\_idx" btree **(**id**)**

**WHERE** **(**channel\_id **=** **ANY** **(**'{0,1}'**::**integer**[]))** **AND** **(**status **=** **ANY** **(**'{0,5,6}'**::**integer**[]))**

**Листинг 5.15. Полный и частичный индекс по полю id**

Поскольку поле id является первичным ключом, то индексируемые значения являются уникальными. Скорее всего, условия частичного индекса не позволят ускорить выполнение запросов.

Предлагается удалить индекс ntn\_message\_id\_channel\_status\_partial\_idx

1. В таблице **ntn.ntn\_message** обнаружены частичные индексы, предлагается оценить их эффективность с помощью запроса ниже:

**SELECT** **COUNT(\*)** **AS** cnt

**,** **COUNT(\*)** FILTER**(WHERE** n**.**channel\_id **IN** **(**2**,** 3**,** 4**)** **AND** n**.**status **=** 4 **AND**

n**.**category **=** 'NEWS' **AND** n**.**user\_type **=** 'EXTERNAL'**)** **AS** cnt1

**,** **COUNT(\*)** FILTER**(WHERE** n**.**channel\_id **IN** **(**2**,** 3**,** 4**)** **AND** n**.**status **IN** **(**0**,** 3**)** **AND** n**.**category **=** 'NOTIFICATION' **AND** n**.**user\_type **=** 'EXTERNAL'**)** **AS** cnt2

**FROM** ntn**.**ntn\_message n**;**

**Листинг 5.16. Запрос оценки эффективности условий частичных индексов таблицы ntn.ntn\_message**

Если cnt1 и cnt2 близки к cnt, то следующие частичные индексы не позволяют улучшить скорость выполнения запросов:

1. ntn\_index\_message\_user\_id\_expiration\_time\_news\_idx
2. ntn\_index\_message\_user\_id\_expiration\_time\_notification\_idx

В этом случае предлагается их заменить на следующий объект:

**CREATE** **INDEX** ntn\_index\_message\_user\_id\_expiration\_time\_cr\_at\_ix

**ON** ntn**.**ntn\_message**(user\_id,** **COALESCE(**expiration\_time**,** 'infinity'**::**timestamp **with** time **zone),** created\_at **DESC);**

**Листинг 5.17. Вариант замены частичных индексов таблицы ntn.ntn\_message**

1. В таблице settings.st\_value обнаружены следующие индексы:

"st\_1\_st\_value\_user\_id\_is\_deleted\_idx" btree **(user\_id)** **WHERE** **NOT** is\_deleted

"st\_1\_st\_value\_user\_id\_spec\_id\_not\_deleted\_idx" btree **(user\_id,** spec\_id**,** is\_deleted**)**

"st\_2\_st\_value\_user\_id\_spec\_id\_not\_deleted\_idx" btree **(user\_id,** spec\_id**)** **WHERE** **NOT** is\_deleted

"st\_spec\_id\_is\_deleted" btree **(**spec\_id**)** **WHERE** **NOT** is\_deleted

**Листинг 5.18. Полные и частичные индексы таблицы settings.st\_value**

Предлагается из заменить на следующее:

**CREATE** **INDEX** st\_1\_st\_value\_user\_id\_spec\_id\_not\_deleted\_idx

**ON** settings**.**st\_value**(user\_id,** spec\_id**,** is\_deleted**);**

**CREATE** **INDEX** st\_spec\_id\_is\_deleted

**ON** settings**.**st\_value**(**spec\_id**,** is\_deleted**);**

**Листинг 5.19. Вариант замены индексов листинга 5.18**

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

### **Настройка HugePages**

Анализ значений файла /proc/meminfo от 01.09.2022 показал, что 6.7GB системы тратится на PageTables, т.е, на таблицы соответствия страниц виртуальной и физической памяти:

PageTables: 6995064 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 70400

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**:** 69632 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.

Ниже приведена статистика по проценту разрастания индексов:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Схема | Таблица | Индекс | Размер | Избыточный размер | Процент избыточности |
| attch | att\_attachment\_v2 | att\_attachment\_v2\_file\_name\_idx | 1226 MB | 764 MB | 62.34 |
| ntn | ntn\_event | ntn\_event\_announcement\_id\_idx | 2812 MB | 1700 MB | 60.47 |
| ntn | ntn\_event | ntn\_event\_event\_type\_fk\_idx | 2742 MB | 1603 MB | 58.46 |
| ntn | ntn\_message | ntn\_message\_jms\_message\_id\_idx | 5228 MB | 2784 MB | 53.26 |
| ntn | ntn\_message | ntn\_index\_message\_user\_id\_expiration\_time\_notification\_idx | 14 GB | 7720 MB | 52.91 |
| ntn | ntn\_message | ntn\_user\_id\_created\_at\_exter\_idx | 15 GB | 7636 MB | 51.14 |
| ntn | ntn\_message | ntn\_message\_channel\_idx | 4987 MB | 2543 MB | 51 |
| ntn | ntn\_message | ntn\_msg\_evgr\_fk\_idx | 6846 MB | 3484 MB | 50.88 |
| ntn | ntn\_event | ntn\_event\_pk | 2294 MB | 1156 MB | 50.37 |
| ntn | ntn\_addressee | ntn\_addressee\_pk | 4617 MB | 2314 MB | 50.12 |
| impr | impr\_import\_record | impr\_import\_record\_pkey | 831 MB | 678 MB | 81.51 |
| impr | impr\_import\_record | impr\_import\_record\_ext\_obj\_id\_idx | 765 MB | 552 MB | 72.13 |
| impr | impr\_import\_record | impr\_import\_record\_session\_id\_idx | 518 MB | 364 MB | 70.33 |
| attch | att\_attachment\_v2 | att\_attachment\_v2\_last\_usage\_idx | 826 MB | 567 MB | 68.67 |

**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.

## **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**