

ПРОЕКТ на тему:

"Учёт по нагрузке ПК с внесением данных в БД."

ОГЛАВЛЕНИЕ:

Задача проекта.....	2
Что делает проект.....	3
Некоторые нюансы.....	9
Настройки, для развертывания на ПК.....	11
Возможные проблемы.	12
Указатели и сноски.....	13



Сартаков Алексей

OTUS-2023-C01-SARTAKOV-AP ·
he/him

Проект выполнил:

Сартаков Алексей Петрович

Тел: *+7-910-797-8034*

Город: *(Нижний Новгород)*

Телеграмм:
https://t.me/Sartakov_Aleksey

Дата выполнения: *декабре 2023г.*

Ученик курса: *«Программист Си»*

Платформа: *ОТУС (Otus.ru)*

Проверяющий проекта (код ревьюер):

Андрей Кравчук

<https://t.me/awkravchuk>

Задача проекта.

Первоначально планировалось, что это будет вспомогательная задача для работы с БД.

Предполагались, что основная программа, ради которой и писалась подпрограмма по нагрузке сервера, будет потреблять много ресурсов по оперативной памяти и по нагрузке на центральный процессор. И все это на продолжительном периоде.

Поэтому, хотелось понять, сколько от зарезервированных ресурсов для БД PostgreSQL 15 можно будет отнять ресурсов. Ориентиров было 3:

- Нагрузка на процессор
- Температура процессора
- Наличие чтения и записи на SWAP диске (СВОП память).

Кроме того, все расчёты делаются на ноутбуке, где охлаждение не такое сильное и требуется принудительная установка промежуточных пауз в глобальных циклах для его охлаждения. Этот подпункт удачно реализован в основном/втором проекте.

Позже, в ходе работы, при чтении инструкций по ядру Linux, были добавлены еще некоторые интересные индикаторы, количество чтения записи с диска – видна излишняя нагрузка на БД.

Сами записи сохраняются как исторические срезы нагрузки на сервер за последний год (можно и дольше), при этом размер данных будет занимать около $12 \cdot 24 \cdot 365 \cdot (29 \text{ колонок по } 8 \text{ байт}) \sim \mathbf{24 \text{ Mb/год}}$.

В ходе написания этой программы, она выросла в полноценное решение с гибкими настройками и выводом графиков через Apache2 сервер, через сайт на чистом Питоне (сайт был реализован уже гораздо раньше).

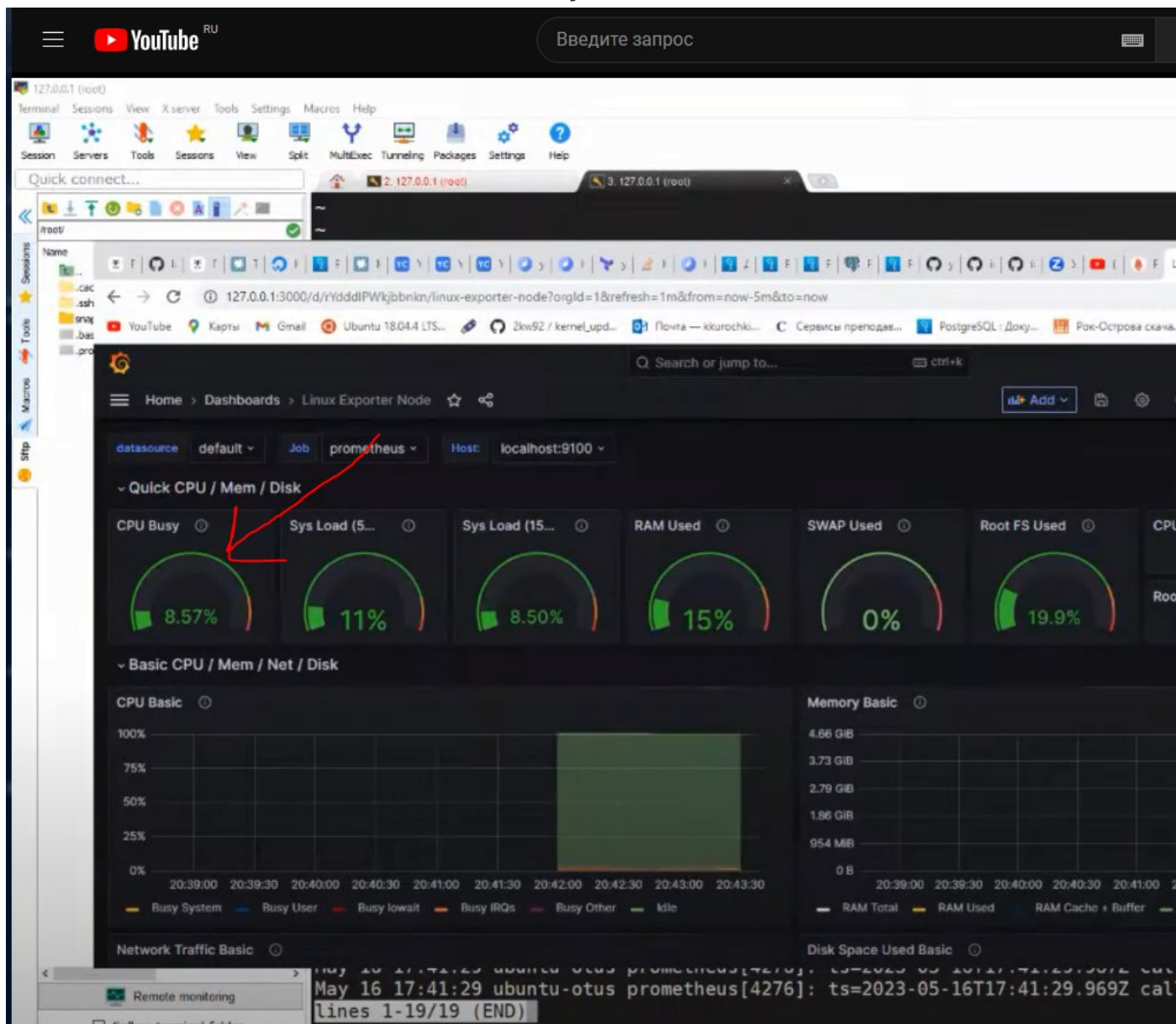
Сама База Данных на PostgreSQL, настройки программы, код на Python для вывода графиков на сайте лежат [в хранилище на github](#). Там же лежат и примеры отчетов.

Что делает проект.

Честно говоря, я пытался использовать существующие бесплатные решения (Monitorix). Но нагрузки в них в холостом режиме была 8-12% что явно было за гранью разумного (возможно были не оптимальные настройки).

Были решения и для самого PostgreSQL [Prometheus и Grafana] (в OTUS.ru даже был открытый урок по их настройке **примеч_1*), но там с настройками нужно очень сильно было повозиться. Да и нагрузка там так же была около 8-9% (правда на виртуальной машине).

Рисунок №1.



После создания моего приложения, оказалось, что программа практически ничего не потребляет в плане ресурсов. По факту около 0,3-0,4% в рабочем режиме и это с учетом работы самой операционной системы.

С интервалом от 1 до 15 секунд (настраиваются по желанию) делаются замеры нагрузки сервера. Один раз в начале минуты делаются записи в БД (обновляются). Каждые 5 минут (кратно 5 минутам) создаётся новая запись в БД PostgreSQL. (рисунок №2).

Рисунок №2

нагрузка_системы											
Свойства Данные Диаграмма											
Введите SQL выражение чтобы отфильтровать результаты											
Таблица	k	123 cpu t min	123 cpu t max	123 cpu t midle	123 ssd read block	123 ssd write block	время	123 ram full free	123 ram доступно max	123 ram до	
1	0	38	42	39,8	1	385 312	2023-12-07 16:35:00.000	47,7	53,1		
2	0	38	61	42,6	257 944	634 848	2023-12-07 16:30:00.000	47,7	55		
3	0	38	50	40,5	1	656 336	2023-12-07 16:25:00.000	49,8	54,3		
4	0	39	43	40,4	448	742 568	2023-12-07 16:20:00.000	49,8	54,3		
5	0	38	61	41,8	72	770 760	2023-12-07 16:15:00.000	49,9	54,4		
6	0	38	63	48,1	416 152	685 496	2023-12-07 16:10:00.000	50	57		
7	0	38	43	40,2	1 048	571 688	2023-12-07 16:05:00.000	53,9	57,8		
8	0	38	43	40,2	520	613 080	2023-12-07 16:00:00.000	54,8	57,8		
9	0	38	45	40,8	100 048	658 264	2023-12-07 15:55:00.000	54,8	57,9		
10	0	39	54	41,6	217 464	500 184	2023-12-07 15:50:00.000	55,3	58,7		
11	0	39	45	41,4	8	505 688	2023-12-07 15:45:00.000	56,6	58,7		
12	0	39	43	40,9	1	500 760	2023-12-07 15:40:00.000	56,6	58,7		
13	0	39	43	40,9	1	493 928	2023-12-07 15:35:00.000	56,6	58,7		
14	0	39	43	40,9	8	484 992	2023-12-07 15:30:00.000	56,7	58,8		
15	0	39	44	40,8	1	492 536	2023-12-07 15:25:00.000	56,6	58,8		
16	0	39	45	41,3	1	504 768	2023-12-07 15:20:00.000	56,6	58,7		
17	0	39	43	40,8	1	498 536	2023-12-07 15:15:00.000	56,6	58,7		
18	0	39	44	41,2	1	526 040	2023-12-07 15:10:00.000	56,8	58,7		
19	0	39	43	40,8	1	545 480	2023-12-07 15:05:00.000	56,6	58,7		
20	0	39	44	41,2	1	598 296	2023-12-07 15:00:00.000	56,7	58,7		
21	0	39	45	40,8	1	601 800	2023-12-07 14:55:00.000	56,7	58,7		
22	0	39	44	41	1	577 344	2023-12-07 14:50:00.000	56,9	58,8		
23	0	39	43	41	1	591 088	2023-12-07 14:45:00.000	56,6	58,7		
24	0	39	43	40,9	1	583 080	2023-12-07 14:40:00.000	56,6	58,8		
25	0	39	45	41,2	1	570 136	2023-12-07 14:35:00.000	56,7	58,7		
26	0	39	45	41,2	1	576 032	2023-12-07 14:30:00.000	56,7	58,7		
27	0	39	44	40,9	1	530 512	2023-12-07 14:25:00.000	56,7	58,8		
28	0	39	45	40,8	1	543 296	2023-12-07 14:20:00.000	56,7	58,7		
29	0	39	44	41,6	1	611 760	2023-12-07 14:15:00.000	56,8	58,7		
30	0	39	43	40,8	1	542 824	2023-12-07 14:10:00.000	56,8	58,7		

Внутри этих 5 минут, делаются замеры каждые N секунд и счётчик количества этих замеров. По истечении 5 минут сумму делим на кол-во замеров и получаем среднее значение за период. Это относится к нагрузке процессора и температуры процессора. Остальные параметры вычисляются как разница в начале и в конце периода в 5 минут или как минимум и максимум опять же внутри этого 5 минутного периода. В общем, каждый параметр считается по своему.

Через сайт (если у вас есть сайт, связанный с вашей БД), можно посмотреть статистику нагрузки по дням (с 00-00 до 23-59). Примеры срезов нагрузок прилагаются в файлах pdf [в проекте на github.com](#)



Какие данные считывает из системы программа и записывает в БД:

1. Оперативная память, которая полностью свободна (никем не зарезервирована). То есть если её вынуть из сервера, то ничего страшного (в текущий момент времени) не произойдет, компьютер просто не заметит её отсутствия.
2. Оперативная память, занятая пользовательскими приложениями (программами, запущенные пользователями), но в основном этот параметр делался, чтобы определить – как много съедают проекты, написанные на Си. Предполагалось, что проекты, написанные на Си будут съедать процентов 20-30. По факту, эти цифры оказались существенно меньше.
3. Оперативная память, которая никак не может быть очищена. Это ядро ОС (оперативная система) плюс запущенные приложения и память которая занята этими приложениями (именно занята, а не зарезервирована). Так, приложение может зарезервировать 50% от всей оперативной памяти, занимать всего 5%, то фактически будет отображаться эти 5% + память занятая ОС.
4. Суммарная нагрузка сервера (всех его ядер). Минимум, максимум и средняя за период в 5 минут. Каждые 5 минут данные обнуляются и считаются заново.
5. Температура процессора (первого ядра, самого горячего). Можно заметить, что примерно в 4 утра идут большие нагрузки – это сервер обрабатывает данные (делает расчеты) в течении 15 минут каждый день, кроме выходных.



6. SSD диск. Количество считанных и записанных блоков за 5 минут. Иногда доходит до 10^7 степени (это 10 млн блоков по 512 байт) за 5 минут (**более 5 Гб**). Это очень большая нагрузка на SSD диск (пример на рисунке №3).
7. Работа с SWAP диском (когда из оперативки скидываются данные на жёсткий диск из-за нехватки оперативной памяти). Делалось с упором, насколько можно отобрать памяти у БД, прежде чем она начнет использовать файл подкачки. Отобрал с 10Gb до 7Gb и пока никаких эксцессов не было.

Рисунок №3



8. Блоков ждет очереди записи на диск (отложенная запись). Берётся максимальное значение внутри 5 минут. В среднем около 10 тыс блоков (около 5 мб), но иногда скачет до 4Гб (работа с копированием БД).
9. Блоков сейчас (в текущий момент) пишется на диск. Пока всегда был равен 0.
10. Оперативная память, доступная для использования. Сюда входит память, которая зарезервирована программами, и может быть освобождена.
11. Оперативная память, отведённая под КЭШ и буферы обмена (зарезервирована программами, к примеру той же БД PostgreSQL).
12. В 00-15 и в 7-45 каждый день, программа запускает очистку оперативной памяти. Но она это может сделать, если программа запускается от прав root пользователя. Если вам очистка оперативной памяти не нужна, то программу можно запускать от

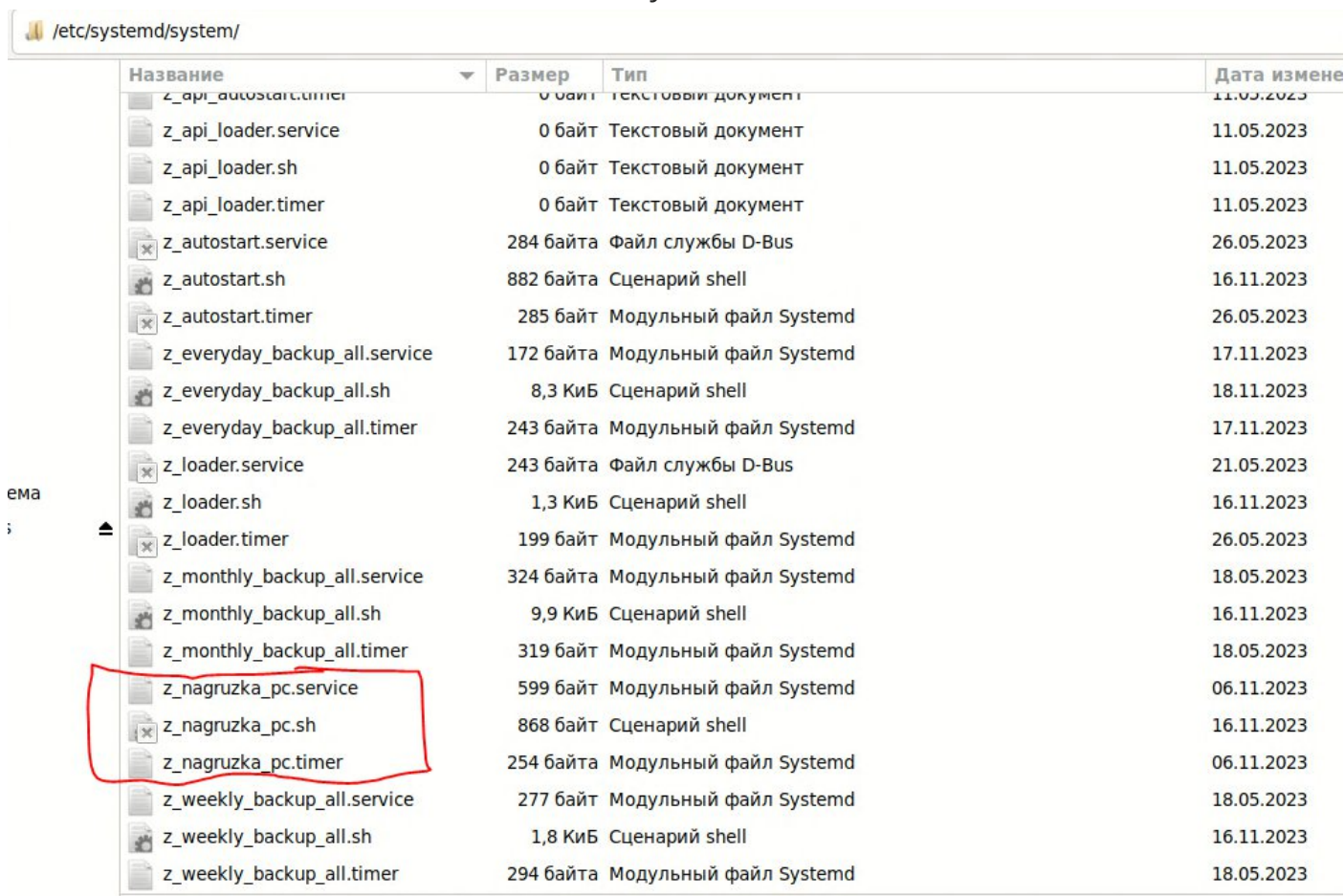
имени любого пользователя, главное, чтобы были верные настройки для соединения с БД PostgreSQL.

13. Программа ведёт автономный лог файл, который можно настроить, указав нужную папку. Каждую полночь – в журнал делается запись о работе программы, чтобы видеть, что программа работает. Проверка возможности работы с лог файлом при первом запуске программы. Очистка лог-файла, если его размер стал более 3 Мб.
14. Учет ошибок в журнале программы сразу от трех источников: как ошибок самой программы по нагрузке ПК, так и языка Си и самой операционной системы. То есть три разных уровня и описания ошибок.
15. Настраиваемое время очисток памяти с учетом локального времени.
16. Запросы к БД komponуются в один сложный, чтобы снизить нагрузку на систему и на саму БД.
17. В лог файле время наступления событий пишется на русском языке в доступном для понимания форме.

При запуске программы, через командную строку вводится от 9 до 12 параметров для её нормального функционирования. Справка-помощь к программе прилагается, просто запустив программу без ключей.

Так же, программа очень хорошо запускается через демоны загрузок на Linux (systemd), см рисунок №4.

Рисунок №4.



Имя	Размер	Тип	Дата измене
z_api_autostart.timer	0 байт	Текстовый документ	11.05.2023
z_api_loader.service	0 байт	Текстовый документ	11.05.2023
z_api_loader.sh	0 байт	Текстовый документ	11.05.2023
z_api_loader.timer	0 байт	Текстовый документ	11.05.2023
z_autostart.service	284 байта	Файл службы D-Bus	26.05.2023
z_autostart.sh	882 байта	Сценарий shell	16.11.2023
z_autostart.timer	285 байт	Модульный файл Systemd	26.05.2023
z_everyday_backup_all.service	172 байта	Модульный файл Systemd	17.11.2023
z_everyday_backup_all.sh	8,3 КиБ	Сценарий shell	18.11.2023
z_everyday_backup_all.timer	243 байта	Модульный файл Systemd	17.11.2023
z_loader.service	243 байта	Файл службы D-Bus	21.05.2023
z_loader.sh	1,3 КиБ	Сценарий shell	16.11.2023
z_loader.timer	199 байт	Модульный файл Systemd	26.05.2023
z_monthly_backup_all.service	324 байта	Модульный файл Systemd	18.05.2023
z_monthly_backup_all.sh	9,9 КиБ	Сценарий shell	16.11.2023
z_monthly_backup_all.timer	319 байт	Модульный файл Systemd	18.05.2023
z_nagruzka_pc.service	599 байт	Модульный файл Systemd	06.11.2023
z_nagruzka_pc.sh	868 байт	Сценарий shell	16.11.2023
z_nagruzka_pc.timer	254 байта	Модульный файл Systemd	06.11.2023
z_weekly_backup_all.service	277 байт	Модульный файл Systemd	18.05.2023
z_weekly_backup_all.sh	1,8 КиБ	Сценарий shell	16.11.2023
z_weekly_backup_all.timer	294 байта	Модульный файл Systemd	18.05.2023

Где можно указать, чтобы она запускалась после перезагрузки сервера (очень удобно, по факту работа программы никогда не прерывается) и передать программе список всех необходимых ключей для нормального запуска. Настройки для автозапуска программы в системе Debian приведены ниже в текущем описании.

Некоторые нюансы.

Запустить саму программу можно двумя разными способами:

- Первый – настройка демона загрузки (не путать с cron-задачами)
- **Второй** – запуск вручную с указанием всех ключей.

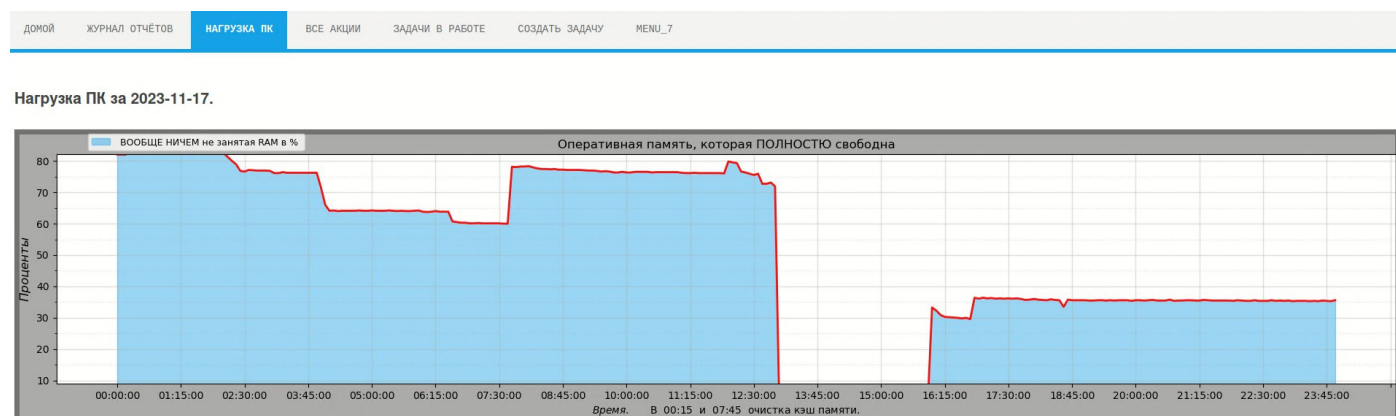
Лично мне больше нравится первый вариант. Настроил и забыл. Было уже не одна переагрузка сервера, а дырок в данных и сбоев в работе программ не было ни одного.

Деление по программам, по потокам, по дискам – считаю не критично, т.к. нужно видеть картину в целом, так сказать фоновая работа. А если есть сильные всплески, то обычно запущена какая то конкретная программа и уже по этой программе делать доскональный анализ более детально. Температура первого процессора всегда самая высокая, да и температур других ядер процессора в своей системе я не нашел.

В моём случае, были сделан ряд оптимизаций на стороне БД и с оперативной памятью (**«echo 3 > /proc/sys/vm/drop_caches»**). Результаты в целом оказались интересные.

Первый раз программу лучше запустить в ручном режиме, с указанием, что это тестовая работа. Тогда все возможные ошибки сразу вылезут при работе программы.

Нижe пример VACUUM FULL для БД PostgreSQL. Оперативна в 12-40 была вся исчерпана.

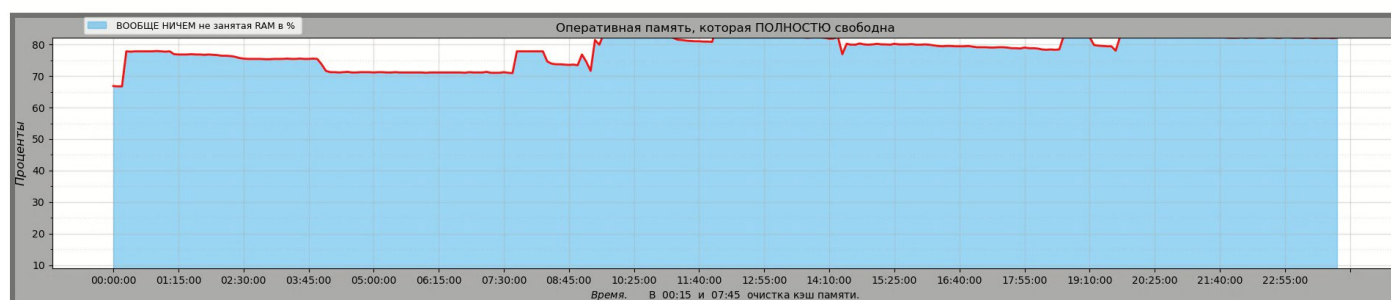


Продолжение графика выше (следующие календарные сутки).



Полная перезагрузка сервера (на уровне железа). Доступная память достигла более 80%.

Нагрузка ПК за 2023-11-16.



Так же очень сильные проблемы с ежемесячным резервированием БД (около 70Гб), но она запускается ночью, когда сервер вообще ничем не занят.

Каких то проблем, косяков в работе программы за время её работы с 08 ноября 2023 не выявлено.

Некоторые поля показывают нулевые значения, но их отлов так же показывал и в первоисточниках те же нулевые значения.

Настройки, для развертывания на ПК.

Для запуска и настройки самой программы достаточно запустить её без ключей. Появится справка по работе с программой и настройкам ключей.

Рисунок №5.

```
postgres@honor-srv: ~/Документы/_отдельные_программы_си/001_demon_нагрузка_cpu/demon_нагрузка_cpu/build-Debug/bin
Файл Правка Вкладки Справка
bash: ./demon_нагрузка_cpu: нет такого файла или каталога
postgres@honor-srv: ~/Документы/_отдельные_программы_си/001_demon_нагрузка_cpu/demon_нагрузка_cpu/build-Debug/bin$ ./demon_нагрузка_cpu

Программа НЕ имеет нужного количества параметров на входе. Сейчас указан(о) 1 параметр(ов).

*** Инструкция по использованию ключей для корректной работы программы ***
=====
Данная программа каждые 60 секунд обновляет данные о нагрузке операционной системы в БД PostgreSQL.
Программа сохраняет информацию по следующим параметрам:
НАГРУЗКА CPU. Записывается максимальная, минимальная и средне арифметическая нагрузка системы.
ОПЕРАТИВНАЯ ПАМЯТЬ. Записывается максимальная, минимальная и средне арифметическое использование памяти.
ИСПОЛЬЗОВАНИЕ SWOP диска. Указывает количество записанных и считанных на/с SSD диска блоков памяти.
ТЕМПЕРАТУРА CPU. Записывается максимальная, минимальная и средне арифметическая температура процессора.
Среднее арифметическое считается как сумма всех замеров КАЖДЫЕ 2 или N секунды и делённое на количество этих замеров.

Каждые 5 (пять) минут в БД создаётся новая запись и последующие 5 минут обновляется именно эта запись.

Программу можно использовать для авто запуска после перезагрузки операционной системы (через DEMON операц.сист.).
Для разделения передаваемых в программу ключей нужно использовать пробел или табуляцию.
Если в каком то имени есть пробел, то такое имя нужно взять в двойные кавычки с каждой стороны, например:
-schema "схема тест"

Список ключей и их значение ЖИРНЫЙ - обязательный параметр:
-db          Название БД в PostgreSQL.
-schema      Имя схемы в БД в PostgreSQL, в которой находится таблица для внесения записей.
-table       Имя таблицы в БД в PostgreSQL, в которую вносятся данные о нагрузке ПК/сервера. По умолчанию = 'нагрузка_системы'
-host        Имя хоста. Можно указывать как localhost так и его IP адрес в сети (IP4). ПО УМОЛЧАНИЮ = localhost
-port        Имя порта (socket). По умолчанию PostgreSQL использует порт 5432, но это может быть и 5433 и 5434, ..
.. ПО УМОЛЧАНИЮ = 5432
-user        Имя пользователя (название роли) в БД PostgreSQL от имени которого будут вноситься записи в таблицу
базы данных.
-psw         Пароль пользователя для доступа к Базе Данных (таблице) в PostgreSQL.
-delay       Задержка, интервал в целых секундах между считыванием нагрузки операционной системы.
Варианты: 1,2,3,4,5,6,10,12,15 секунд. По умолчанию, замеры каждые 2 секунды.
-log_path    Полный и Абсолютный путь к лог файлу, куда будут писаться ошибки и результат работы программы, т.к.
она подсоединяется к БД, то важно контролировать ошибки.
Путь по умолчанию = '/var/log/z_nagruzka_pc.log' но с правами root.
Для хранения в другой папке, указанный путь (все папки и подпапки) уже должны существовать. При первом запуске, прог
рамма проверит возможность создать/дополнить лог файл о чём будет сделана запись внутри лог файла или выведено сообщ
ение об ошибке.
Если размер ЛОГ файла больше 10 млн байт (~9.53 Mb), то этот файл затирается.

-help 2      Вывод справочного файла-помощника. Просто выводится текущая справка.
-test 2      Тестовый режим - выводит сообщения в терминал. Использовать этот ключ самый последний!
Обязательно использовать этот ключ в связке с числом 2. Само число роли не играет.
Число используется как защита от сбоя при разборе ключей на входе в программу (один ключ = 2 поля).
```

Ниже приводится код для настройки запуска программы через systemd и должны быть настроены все три файла.

1. Эти файлы настроек должны быть в папке /etc/systemd/system/
2. Все настройки необходимо делать от прав суперпользователя (root).
3. Файл, в котором указаны ключи для запуска программы лучше закрыть для любого чтения всем пользователям и всем группам, оставив только для самого root пользователя, т.к там указан пароль для доступа к БД.
4. В настройке БД должен иметься/существовать пользователь от имени которого мы соединяемся из программы с БД PostgreSQL.
5. Для этого пользователя в БД лучше создать отдельные права на доступ именно к указанной таблиц(там потребуется подцепить так же несколько еще дополнительных таблиц). Настройки пользователя в БД PostgreSQL выходят за рамки проекта (там очень все непросто). Можете создать

в Посгрес пользователя с правами суперпользователя, хотя это и не рекомендуется.

6. Для работы с БД должен быть открыт доступ из вне (если соединяетесь по сети а не на локальном ПК).

Настройки PostgreSQL файла /etc/postgresql/15/имя_*/postgresql.conf

```
#-----  
# CONNECTIONS AND AUTHENTICATION  
#-----  
# - Connection Settings -  
listen_addresses = '*'          # what IP address(es) to listen on;
```

Настройки PostgreSQL файла /etc/postgresql/15/имя_*/pg_hba.conf

```
# TYPE      DATABASE      USER      ADDRESS      METHOD  
# "local" is for Unix domain socket connections only  
local      ваша_бд      пользователь_в_бд      peer  
local      all          all        scram-sha-256      # peer  
# IPv4 local connections:  
host      all          all        127.0.0.1/32      scram-sha-256  
host      all          all        192.168.2.0/24      scram-sha-256
```

Желательно, то же имя, что указано в БД имелось в вашей системе Debian/Ubuntu. Тогда вы сможете запускать скрипты для работы с БД без ввода пароля (эта тема выходит за рамки текущего проекта).

МИНИМАЛЬНЫЙ НАБОР. Если вам нужно создать только саму таблицу в БД для записи исторических данных по нагрузке ПК и функцию по формированию ответа на запрос с сайта о нагрузке ПК, то эти модули вынесены в отдельные файлы и находятся в папке ["код для сайта" на github.com](#)

Для разворачивания ПОЛНОЦЕННОЙ копии БД **test_si_2023** текущего проекта на вашем сервере. (сама БД PostgreSQL уже должна быть установлена на вашем ПК) нужно скачать архив с БД, разархивировать его и запустить следующий код по переносу содержимого БД.

1. Сперва создаём саму БД на вашем сервере: подключаемся к серверу на котором будем создавать тестовую БД, к примеру **psql -p 5432**
2. **create database test_si_2023;**
3. **alter role postgres password '123';** если у вас еще не активирована учетная запись в БД (только создали БД).

4. Выходим из psql через \q
5. Распаковываем БД из архива
6. В окне терминала запускаем код по переносу БД:
`psql -h localhost -p 5432 -d test_si_2023 -U postgres` <
`/home/postgres/адрес_к_бд/имя_бд.sql`
7. Подключаемся к БД PostgreSQL через DBeaver 23.*
8. Начинаем работу.

Настройки systemd = **z_nagruzka_pc.service**

[Unit]

Description=Запуск программы, которая отслеживает нагрузку ПК

[Service]

Type=exec #notyfi

ExecStart=/etc/systemd/system/z_nagruzka_pc.sh

запускается ТОЛЬКО БЕЗ КЛЮЧЕЙ !!!!!

Restart=restart-always

StandardError=journal

Type=oneshot: Этот тип указывает, что процесс будет недолговечным и система должна ждать

завершения процесса, прежде чем продолжить работу с другими устройствами.

Настройки systemd = **z_nagruzka_pc.timer**

[Unit]

Description=Запуск программы, которая отслеживает нагрузку ПК в режиме реального времени.

[Timer]

OnBootSec=0h 1m

Unit=z_nagruzka_pc.service

[Install]

WantedBy=multi-user.target

```

Настройки systemd = z_nagruzka_pc.sh
#!/bin/sh
# Запуск программы, которая отслеживает нагрузку ПК в режиме реального
времени
# и скидывает все данные (каждую 1 минуту) в ЮД PostgreSQL. Нагрузка
процессора, его температура,
# жесткий диск Память, СВОП, Кэш и прочие вещи. Автоматический запуск
и после перезагрузки ПК.

/usr/local/bin/z_nagruzka_pc -db test_si_2023 -port 5432 -host localhost -
user имя_пользователя -psw ваш_пароль_для_пользователя_бд

# -test 2 = тестовый запуск - сработает ли корректно и только потом
боевой, убрав ключи « -test 2 »

```

ВНИМАНИЕ! демоны можно создавать только на латинице, иначе может быть беда с выводом названия в списке задач или при запасах по демонам. А вот описание демона запросто можно делать на русском - с этим полный порядок.

Все демоны размещены в папке: /etc/systemd/system/ Их название начинается с «z_»

Сами программы в каталоге /usr/local/bin/ доступ от имени root.

Затем нужно активировать все эти настройки. Если активация не происходит, то где то ошиблись. Не пытайтесь добавить ключи в сам файл *.service, они там не сработают!!

Для мониторинга нагрузки ПК = z_nagruzka_pc.timer + z_nagruzka_pc.service + z_nagruzka_pc.sh Демоны являются автозагрузками, то есть после перезагрузки сервера он будет продолжать выполнять свою работу.

Они работают в паре (timer + Service). Для их активации нужно запустить команду (один раз) и долька для блока, в котором есть раздел = [Install].

1. # sudo systemctl enable z_nagruzka_pc.timer так мы добавляем в список запуска после перезагрузок сервера

- # sudo systemctl daemon-reload
- sudo systemctl list-units --type=timer
- sudo systemctl list-timers

2. Затем нужно активировать сам юнит: # sudo systemctl start z_nagruzka_pc.timer

3. Проверить статус (работает ли наш юнит) нужно через программу # sudo systemctl status z_nagruzka_pc.timer

4. Если вдруг потребовалось удалить наши юнты (ну стали не нужны, написали к примеру другие), то тогда нужно:

- `# sudo systemctl stop z_nagruzka_pc.timer + sudo systemctl stop z_nagruzka_pc.service`

5. Затем удалить данные юниты из авто старта после перезагрузки системы:

- `# sudo systemctl disable z_nagruzka_pc.timer`

6. Перезагрузка каталога systemd

- `sudo systemctl daemon-reload`

7. Затем удалить оба файла из папки `/etc/systemd/system/`

Сами скрипты расположены в папке `/etc/systemd/system/z_....sh` . При перемещении файлов в другую папку — работчпособность НЕ ГАРАНТИРУЕТСЯ!

Возможные проблемы.

Имеется ряд подводных камней.

Папка измерения температуры отличалась от примеров в интернет глубиной вложенности. Поэтому, возможно, что на вашем ПК она может быть немного в другом месте. Но в целом примерно там же. Требуется просто ручная проверка - открыть пару раз файл по указанному адресу папки и посмотреть, меняется ли цифра и совпадают ли она с температурой, что выводит система (примерно).

Жёсткий диск. Он взят как диск в целом. У меня в системе стоит только один диск, поэтому был просто взят как `nvme0n1`. На сервере, у которых два и более диска, будет считываться данные только по **одному, самому первому, верхнему в перечне дисков**. Но в принципе, можно изменить код, добавив несколько дисков, но это потребует изменений и при запросах к БД, на стороне самой БД и при запросах данных на сайте для вывода статистики. Если диск ключевой только один, а остальные вспомогательные, то достаточно просто изменить указатель на диск с которого нужно считывать данные.

Выглядят данные по диску вот так: **259** **0 nvme0n1** 687966 231117
56075223 105895 99899297 4752443 1023626818 108883062 0 171576384 167982189
63056 0 913469712 23096 44959795 58970136

Ниже перечень путей, которые могут измениться на вашей системе.

- Процессор/CPU `/proc/cpuinfo`
- Память RAM `/proc/meminfo`
- Температура CPU `/sys/class/hwmon/hwmon0/temp1_input`
- SSD диск `/proc/diskstats` (взять в целом как диск `nvme0n1`)
- Нагрузка CPU `/proc/stat`

Проверялось все на Debian 10/11 (Linux). Точно должно работать на Ubuntu старше 2016 года.

Данные о нагрузке сетевой карты не брались, т .к. в моём случае они были не важны.

По остальным значениям/блокам - читал в первоисточнике по ядру Linux и больше там ничего интересного не оказалось.

Указатели и сноски.

*Примечание_1. [Мониторинг PostgreSQL grafana+prometheus // Курс «PostgreSQL для администраторов баз данных».](#)

*Примечание_2. [Оптимизация производительности кластера PostgreSQL // Демо-занятие курса «PostgreSQL Advanced»/ Dhtvz = 29м 30сек](#)

*Примечание_3. Описание настроек железа при работе в ядре Linux. То есть какие данные и из какой папки можно получить при работе в Linux.
<https://docs.kernel.org/admin-guide/index.html>
<https://docs.kernel.org/admin-guide/sysctl/index.html>
<https://docs.kernel.org/admin-guide/iostats.html?highlight=diskstats>

На github.com выкладываются (>>>перейти):

- копия БД PostgreSQL (её урезанная, но полностью рабочая в данном проекте). **Суперпользователь postgres, пароль 123**
- Код для запросов к БД PostgreSQL и вывода графиков на сайте (в Python).
- Кусок кода index.html для перехода к нужной странице по выводу графиков нагрузки ПК.
- Функция как отдельный код на стороне БД **для выборки данных из исторической таблицы** по нагрузке сервера, генерации/компоновки данных и отправки их Питону на сайте (для тех кто уже имеет установленную БД PostgreSQL).
- Сама функция **по созданию таблицы** на стороне Постгрес со всеми настройками столбцов. Для тех кто уже имеет установленную БД PostgreSQL.
- Примеры отчетов.
- **КОД ПРОГРАММЫ НА СИ**
- Описание проекта (текущая инструкция).

Система протестирована на тестовой копии БД. Проблем с работой программы нет.