

МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
“ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”

Факультет *компьютерных наук*

Кафедра информационных систем

Разработка web-интерфейса управления RAID-массивом

ВКР *Бакалаврская работа*

09.03.02 *Информационные системы и технологии*

Информационные системы и сетевые технологии

Допущено к защите в ГЭК

Зав. кафедрой _____ *Д.Н. Борисов, к. т. н., доцент* __.__.2022

Обучающийся _____ *В.Р. Григоренко, 4 курс, д/о*

Руководитель _____ *А.А. Головкин, ассистент*

Воронеж 2022

МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет компьютерных наук

Кафедра информационных систем

УТВЕРЖДАЮ

заведующий кафедрой

подпись, расшифровка подписи

__._.2022

ЗАДАНИЕ

НА ВЫПОЛНЕНИЕ ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

ОБУЧАЮЩЕГОСЯ Григоренко Виктора Руслановича

фамилия, имя, отчество

1. Тема работы «Разработка web-интерфейса управления RAID-массивом», утверждена решением ученого совета _____ факультета от __.__.2022

2. Направление подготовки 09.03.02 Информационные системы и технологии

3. Срок сдачи законченной работы __.__.2022

4. Календарный план: (строится в соответствии со структурой ВКР)

№	Структура ВКР	Сроки выполнения	Примечание
1	Введение	20.03.2022	
2	Постановка задачи	27.03.2022	
3	Средства реализации	01.04.2022	
4	Цели создания web-приложения	05.04.2022	
5	Анализ предметной области	07.04.2022	

6	Терминология	13.04.2022	
7	Уровни RAID-массивов	15.04.2022	
8	Разработка	16.04.2022	
9	Интерфейс управления RAID-массивами	20.04.2022	
10	Web-интерфейс взаимодействия с разработанным RAID-менеджером	22.04.2022	
11	Сравнение создания RAID-массива	23.04.2022	
12	Заключение	26.04.2022	
13	Список использованных источников	30.04.2022	

Обучающийся

Подпись

расшифровка подписи

Руководитель

Подпись

расшифровка подписи

Реферат

Бакалаврская работа 44 с., 32 рис.

RAID-МАССИВ, ЗЕРКАЛИРОВАНИЕ, ЧЕРЕДОВАНИЕ, ЧЁТНОСТЬ,
УРОВНИ RAID-МАССИВОВ, MDADM, WEB-ПРИЛОЖЕНИЕ.

Объектом исследования являются механизмы взаимодействия интерфейсов и средств ОС Linux, создание и управление RAID-массивами.

Цель работы – программная реализация web-интерфейса для управления RAID-массивами.

В процессе выполнения работы проводились:

- изучение архитектуры RAID-массивов;
- изучение утилит и команд для управления RAID-массивами;
- проектировка понятного и лаконичного внешнего вида для клиентской части web-приложения.

В результате было реализовано web-приложение для взаимодействия с RAID-массивами с возможностями:

- упрощённого процесса создания RAID-массивов;
- управление RAID-массивами;
- мониторинга состояния RAID-массивов.

Содержание

Введение.....	6
1 Постановка задачи.....	7
1.1 Средства реализации.....	8
1.2 Цели создания web-приложения.....	8
2 Анализ предметной области.....	9
2.1 Терминология	9
2.2 Уровни RAID-массивов.....	15
2.3 Язык программирования	22
2.4 Фреймворк	23
2.5 Операционная система	23
3 Разработка	24
3.1 Интерфейс управления RAID-массивами.....	24
3.2 Web-интерфейс взаимодействия с разработанным RAID-менеджером.....	28
3.3 Сравнение создания RAID-массива	32
Заключение	44
Список использованных источников	45

Введение

В настоящее время компьютер тесно связан с нашей жизнью, так как компьютер применяется практически во всех сферах жизни, появляется проблема хранения больших объёмов данных, эту проблему помогают решить RAID-массивы. Они помогают сохранять данные и повысить производительность систем.

Большое количество RAID-массивов используются на unix подобных систем, так как особенность этой ОС в её безопасности и конфиденциальности. Большинство unix – подобных операционных систем бесплатны, так же стоит отметить, что в данных ОС потребляется гораздо меньше аппаратных ресурсов. Данные операционные системы более гибкие в настройке. Мой выбор был сделан на дистрибутив Ubuntu, так как он наиболее популярен среди всех unix-подобных систем, так же разработчики часто выпускают обновления для улучшения производительности и устранения возникших неполадок.

RAID–массивы применяются в крупных компаниях, в серверных, а так же в домашнем использовании. Технология RAID (Redundant Array of Independent Disks) – избыточный массив независимых дисков. Принцип работы данной технологии состоит в том, чтобы из набора накопителей(дисков) создаётся массив, который определяется в системе, как большой логический диск. Надёжность хранения информации выполняется дублированием данных, а высокое быстродействие системы выполняется за счёт параллельного выполнения операций ввода-вывода данных.

Выбор RAID-массива определяется в зависимости от требуемых задач, компонентов системы и цены. Существует 2 подхода в реализации RAID-массивов: программный и аппаратный. В данной работе выбор был сделан в программной реализации, так как не требует наличие RAID-контроллера.

Для создания RAID-массива будут использоваться утилиты из Linux. Для простоты использования реализация будет в web-интерфейсе.

1 Постановка задачи

Главной целью данной работы является разработка интерфейса для управления RAID-массивами, на основе которого будет проектироваться и реализовываться web-приложение для управления RAID-массивами.

Разрабатываемый интерфейс для управления RAID-массивами должен уметь выполнять все необходимые действия по предварительной настройке для создания самого RAID-массива, а также его создание, мониторинг состояния созданных RAID-массивов и остановка работающих RAID-массивов.

Разрабатываемое web-приложение должно быть интуитивно понятно для пользователя, иметь современный вид и выполнять весь функционал, который будет реализован в интерфейсе для управления RAID-массивами.

На основе поставленных выше целей были сформулированы следующие задачи:

- исследовать и изучить утилиты и команды для создания, управления, мониторинга состояния и удаления RAID-массивов;
- спроектировать и реализовать интерфейс для работы с RAID-массивами, исходя из результатов выполнения первой задачи;
- спроектировать и реализовать интуитивно понятный и лаконичный внешний вид клиентской части будущего web-приложения;
- спроектировать и реализовать сервер приложений будущего web-приложения;
- реализовать взаимодействие между разработанным интерфейсом для работы с RAID-массивами, сервером приложений и клиентской частью web-приложения;
- провести тестирование на работоспособность разработанного web-приложения для работы с RAID-массивами и сравнить затраченное время и трудоёмкость на аналогичные действия в терминале.

1.1 Средства реализации

Реализация web-приложения будет осуществляться на базе языка программирования Python, а именно на определённых его модулях:

- os и subprocess с помощью которых реализовывается интерфейс для работы с RAID-массивами;
- Flask и его компоненты для реализации web-интерфейса и связывания его с разработанным интерфейсом для работы с RAID-массивами.

Также понадобятся утилиты mdadm, mkfs, fdisk и некоторые другие для реализации интерфейса работы с RAID-массивами.

Разработанное web-приложение будет работать на базе операционной системы Ubuntu.

1.2 Цели создания web-приложения

Основными целями разработки и реализации web-приложения являются:

- упрощение процесса создания, управления и мониторинга состояния RAID-массива для обычных пользователей;
- экономия времени пользователей на изучение способов создания, управления и мониторинга состояния RAID-массива.

2 Анализ предметной области

В данном разделе выделим и рассмотрим сущности, которые необходимо будет реализовать в будущем web-приложении. Также проведём анализ инструментов для реализации данных сущностей.

2.1 Терминология

Разберём некоторые термины, ознакомление с которыми позволит свободнее понимать тему данной работы.

2.1.1 RAID

В переводе с английского «RAID» (Redundant Arrays of Inexpensive Disks) означает «избыточный массив независимых дисков». RAID – это дисковой массив. Данный массив создан для повышения безопасности хранения данных и/или для повышения скорости записи и/или чтения информации[1].

Отказоустойчивость добивается за счёт избыточности. Доля дискового места становится недоступной для пользователя. Увеличение скорости системы достигается за счёт параллельной/независимой работы нескольких дисков. Данные разбиваются на определенные блоки и синхронно записываются на диски. Пример: блок имеет размер 10 Кбайт, в массиве имеется 5 дисков, а размер файла 50 Кбайт, при выполнении записи файлов на диск, файл делится на 5 блоков и запись происходит синхронно. Так же бывают ситуации, когда размер записываемой информации, имеет меньший размер, чем блок, тогда запись происходит на один диск[1].

При модели независимого доступа вся информация, которая выполняется одной командой запроса, то запись или чтение происходит на каждое отдельное устройство.

Технология RAID имеет ряд преимуществ [2]:

- использование нескольких устройств увеличивает производительность системы, показатели скорости чтения и записи информации, значительно повышаются;
- гарантированная отказоустойчивость, то есть повышение надёжности сохранения данных, при отказе одного из дисков, функционирование системы и доступ к данным продолжается.

Данная технология имеет также и недостатки [3]:

- комбинация уровней рейд, повышает стоимость реализации технологии, так как требует большее количество дисков;
- не все уровни рейд, могут сохранять скорость чтения и записи, при выходе одного из устройств;
- при выходе из строя диска в некоторых уровнях, сохранение информации не гарантировано или может быть восстановлена не в полном объёме;
- работа по восстановлению данных или починки диска, может занимать трудоёмкий процесс, требующий большого количества времени и знаний.

Различные уровни RAID используют различные методы и принципы. Уровень RAID – характеризует отношения между устройствами хранения данных.

2.1.2 ЕСС-память

ЕСС-память – это тип модуль корреляции ошибок, который позволяет исправить или опознать ошибку.

2.1.3 Массив

Массив – это несколько связанных устройств хранения данных, которые управляются, настраиваются и форматируются одним единым центром [5]. Логический массив – это более сложное представление накопителей без учитывания физических характеристик системы. Следовательно, логические диски по объёму

и количеству могут не совпадать с физическими параметрами устройств. Операционная система определяет массив как один большой диск.

2.1.4 Метод зеркалирования

Метод зеркалирования – это метод копирования одинаковых данных на один и более диск, входящих в массив для надёжности хранения информации (рисунок 1.1) [5]. Этот метод позволяет увеличить надёжность системы. При выходе из строя одного из устройств хранения данных, данные остаются на втором устройстве. Данный метод имеет 100% избыточность.

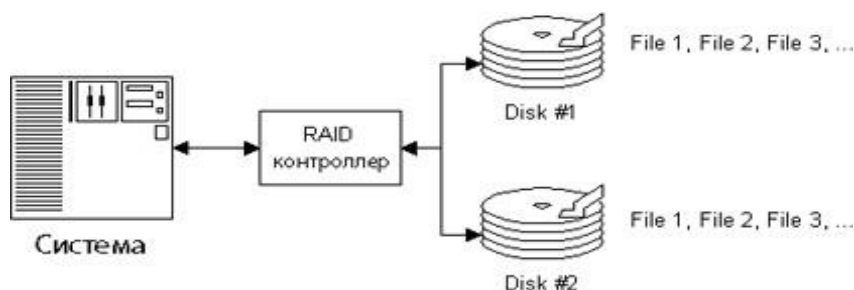


Рисунок 1.1 – Зеркалирование

2.1.5 Чередование

Чередование – метод позволяющий увеличить быстродействие системы. (рисунок 1.2) [5]. В данном методе, запись и чтение происходит следующим образом, файл делится на определённый размер (от 1 байта и больше) и направляется или считывается параллельно на все устройства хранения данных. Метод не имеет избыточности, но при выходе из строя одного из дисков, информация будет потеряна.

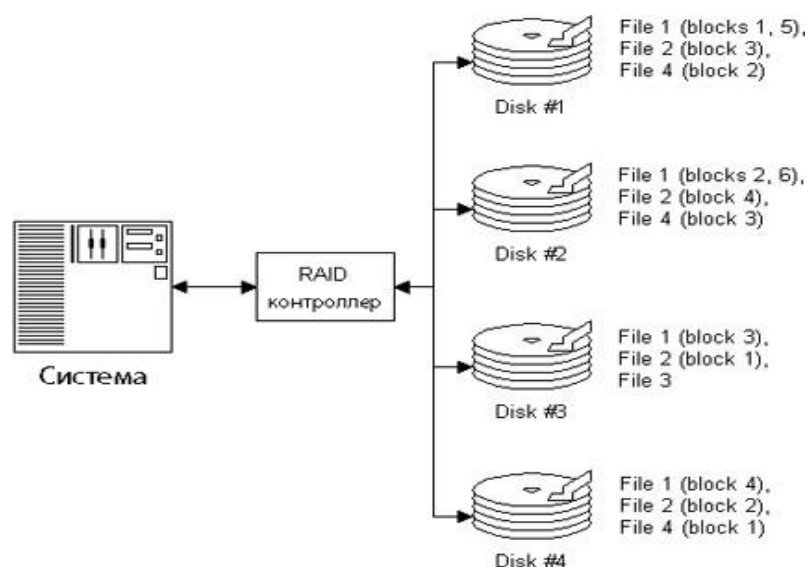


Рисунок 1.2 – Чередование

2.1.6 Дуплекс

Дуплекс – это метод зеркалирования, но с использованием вдвое большего количества накопителей для создания копий (рисунок 1.3) [5]. Данный метод имеет высокий уровень надёжности, но также имеет дополнительные затраты. В случае выхода из строя устройства хранения данных и/или RAID-контроллера система продолжает работу.

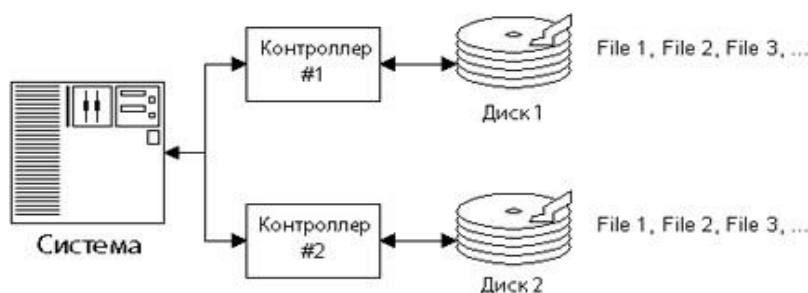


Рисунок 1.3 – Дуплекс

2.1.7 Чётность

Чётность – это технология, в которой объединяются зеркалирование и чередование [5]. Данный метод имеет высокую скорость работы и надёжность. Данный метод работает по принципу чётности оперативной памяти.

2.1.8 Программный и аппаратный RAID-массив

RAID-массивы разделяются по способу реализации на аппаратный и программный. С помощью RAID-контроллера или RAID -карты в аппаратном RAID можно создавать и управлять массивами RAID, независимо от операционной системы [6]. В этих контроллерах имеется специальный процессор для управления устройствами RAID. Преимущество аппаратного RAID: на управление устройствами хранения не тратятся дополнительные ресурсы, так как это выполняет RAID контроллер; так как массив управляется без программного обеспечения, то на RAID -массив устанавливается главная файловая система. Недостатки: качественные контроллеры, стоят дорого; при поломке RAID, необходим аналогичный контроллер.

Программный RAID настраивается с помощью операционной системы. Преимущества: нет дополнительных затрат, так как не нужно аппаратное обеспечение; гибкость в управлении RAID-массивом, можно перенести RAID массив с одной операционной системы на другую, без потери доступа к данным [6].

2.1.9 Web-приложение

Web-приложения – это приложение в котором, клиент взаимодействует с веб-сервисом с помощью браузера [7]. Данный вид приложений логически делится серверную и клиентскую часть. В большинстве случаев данные хранятся на сервере, а обмен информацией происходит по сети. Web-приложение пишется с помощью нескольких языков программирования, так как оно делится на backend и frontend.

В данном случае будет использоваться подход к разработке МРА (Multi Page Application). Данный подход характеризуется приложением с множеством страниц, в которых загрузка новой информации, также изменение данных перезагружают страницу для их отображения [8].

На рисунке 1.4 представлена теоритическая схема будущего web-приложения.

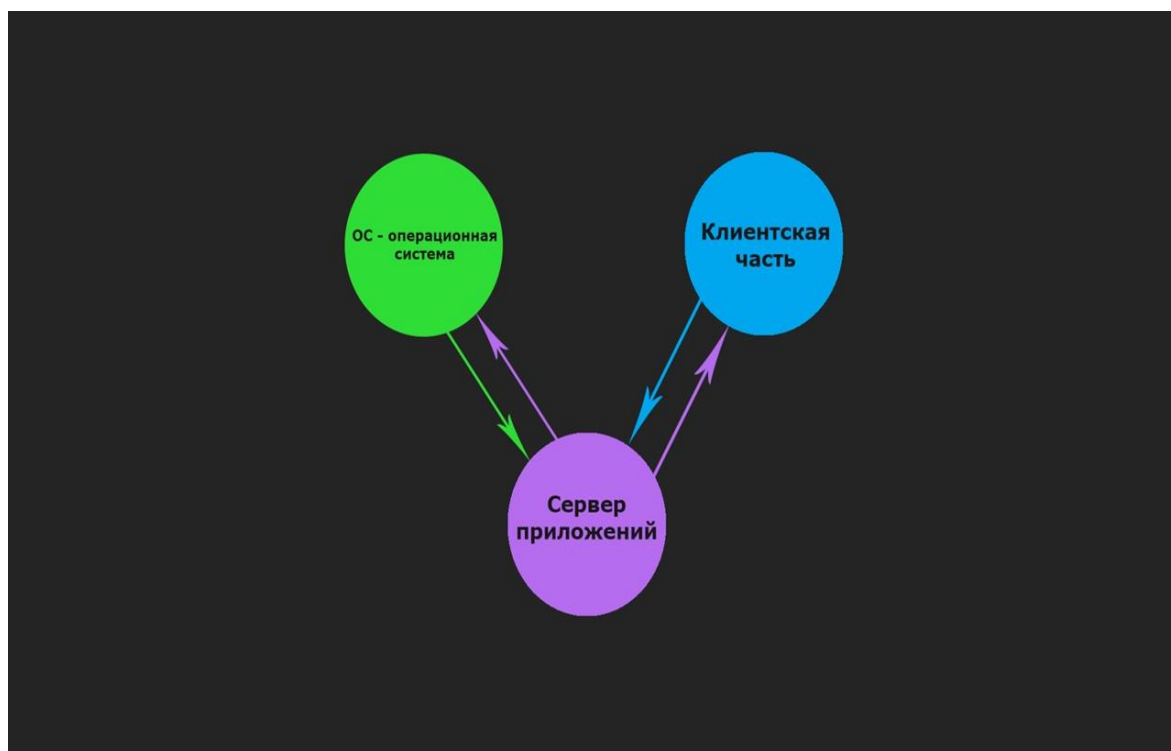


Рисунок 1.4 – Web-приложение

Клиентская часть приложения, так же называемая frontend, это пользовательский интерфейс, который виден пользователю [8]. В данной части приложения клиент обращается с запросами к серверу. Значимость клиентской части состоит в том, чтобы информировать сервер, о том, что совершить с данными, которые он передаёт, либо с данными, которые, находятся в базе данных. Главной задачей frontend части для пользователя состоит в том, чтобы предоставлять данные в удобном виде и использовать механизмы для обновления данных. Для реализации frontend был использован стандартный язык разметки html и фреймворк bootstrap. В данной работе клиентом является браузер.

Серверная часть приложения, она же backend, принимает запросы от клиентской части [8]. Роль серверной части в том, чтобы обрабатывать и индексировать информацию, предоставлять доступ к данным и выполнять запросы клиента.

В данной работе для backend части был выбран язык программирования Python и микрофреймворк Flask.

2.2 Уровни RAID-массивов

Рассмотрим уровни raid-массивов. Уровни от RAID 0 до RAID 5 считаются стандартизированными. Существуют фирменные уровни (RAID 6,7) и комбинированные. Уровни 0, 1, 3, 5 считаются наиболее распространёнными.

2.2.1 RAID 0

RAID уровня 0 (рисунок 1.5), объединяет два и более устройства хранения данных, затем информация делится на блоки и записывается на отдельные устройства, путём чередования информации [9]. В результате синхронного ввода-вывода информации с разных дисков, предоставляется максимальная скорость передачи данных. В основном этот уровень используется в тех случаях, когда необходима быстрая передача больших объёмов информации.

Из недостатков уровня можно отметить следующее:

- при выходе из строя одного из дисков, весь массив перестает работать и все данные будут потеряны;
- данные из этого уровня невозможно восстановить.

Также есть и ряд преимуществ данного вида рейда:

- увеличивает производительность системы;
- нет дополнительных затрат.

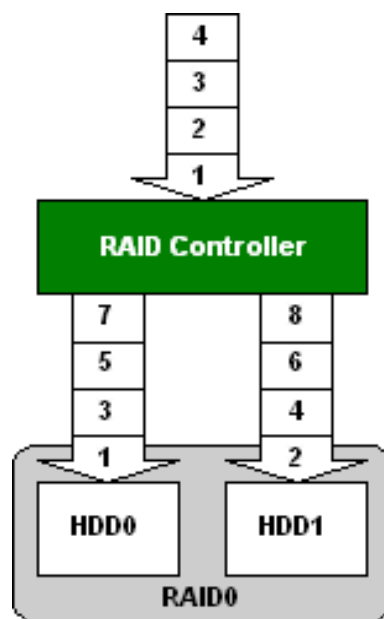


Рисунок 1.5 – RAID 0

2.2.2 RAID 1

RAID уровня 1 (рисунок 1.6) – это массив, при котором данные зеркалируются, то есть данные из 1 устройства полностью копируются на другое устройство хранения данных [9]. Данный уровень имеет большой уровень надёжности. Такой уровень используется при наивысшем приоритете надёжности сохранения данных. Преимущество: при выходе из строя одного устройства все данные сохраняются.

В данном уровне присутствуют недостатки:

- высокая стоимость реализации данного уровня;
- высокая избыточность, так как один диск просто используется для хранения одинаковых данных, в итоге пользователь получит из 2 дисков объём одного диска.

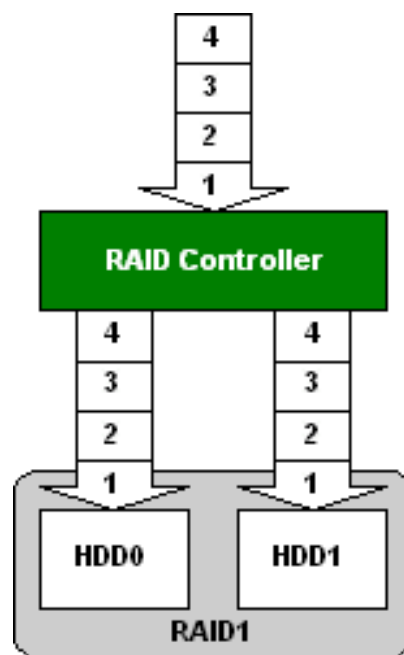


Рисунок 1.6 – RAID 1

2.2.3 RAID 2

RAID уровня 2 (рисунок 1.7) — это массив данных, использующий чередование дисков, в котором некоторые диски выделяются, и в них записывается информация о исправлении ошибок и проверке, так же используется чётность кода Хемминга для обнаружения ошибок и их корреляции, но так как современные устройства имеют само контролирующийся код Хемминга, то такой уровень считается устаревшим [9].

К преимуществам можно отнести: операции с информацией происходят быстрее в сравнении со скоростью одного диска.

Можно выделить главные недостатки уровня:

- реализация уровня, требует специальные дорогие контроллеры, что влечёт за собой увеличения стоимости данного уровня;
- только при 7 дисках, его использование рационально, так как он требует меньшее количество дисков, чем RAID 1.

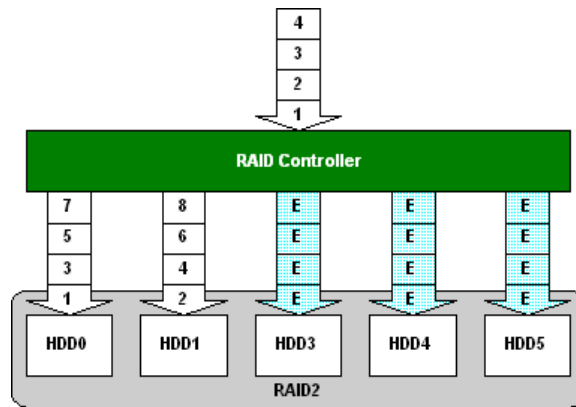


Рисунок 1.7 – RAID 2

2.2.4 RAID 3

RAID уровня 3 (рисунок 1.8) – это массив в котором данные, разделяются на подблоки на уровне байтов, после этого синхронно записываются на все устройства памяти кроме одного, так как он используется для хранения контрольной информации [9].

К достоинствам уровня можно отнести:

- при выходе из строя любого устройства хранения данных, данные можно восстановить по контрольным данным и данным, оставшимся на исправных дисках;

- повышение производительности системы;
- при работе с большими файлами показывает хорошие результаты.

К недостаткам уровня относятся:

- увеличенная нагрузка на контрольный диск;
- при малом размере блока, требуется больше времени для чтения.

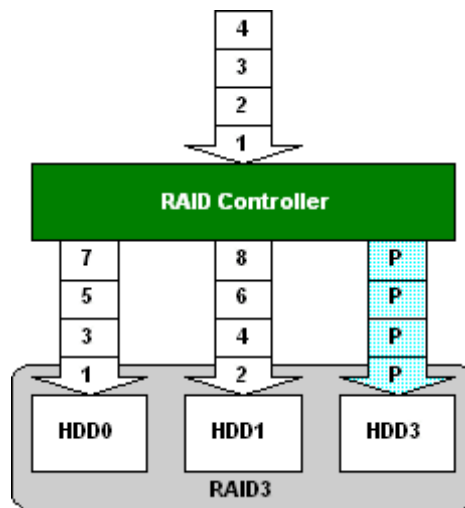


Рисунок 1.8 – RAID 3

2.2.5 RAID 4

RAID уровня 4 – это массив с одним устройством, в котором хранятся контрольные суммы (рисунок 1.9) [9]. Данный уровень похож на 3, но с увеличенным размером блока записываемой информации. Контрольная сумма записывается на выделенный диск и благодаря этому возможно синхронное выполнение нескольких операций чтения. К недостатку уровня можно отнести следующие: производительность снижается, из-за того, что все записи идут на блок чётности.

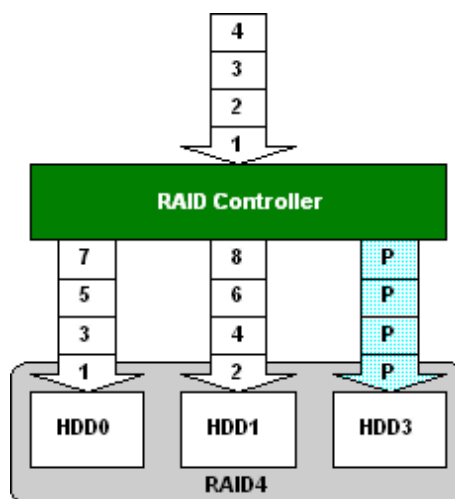


Рисунок 1.9 – RAID 4

2.2.6 RAID 5

RAID уровня 5 – это отказоустойчивый массив с распределением хранения контрольных сумм и чередованием (рисунок 1.10). В данном уровне все устройства хранения данных имеют одинаковый размер, но на один диск меньше, так как 1 устройство хранения данных отводится на контрольную информацию. Данный уровень очень распространён в различных системах [9].

К недостаткам можно отнести следующие пункты:

- потеря производительности при записи в произвольном порядке;
- при выходе из строя одного из дисков, отказоустойчивость приравнивается к RAID уровня 0.

Можно выделить следующие достоинства уровня:

- высокая скорость записи и чтения информации;
- относительная экономия стоимости в сравнении с RAID уровня 10; присутствует отказоустойчивость.

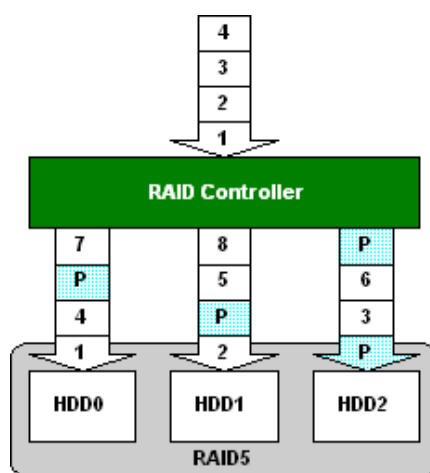


Рисунок 1.10 – RAID 5

2.2.7 RAID 6

RAID уровня 6 (рисунок 1.11) – это массив с двумя устройствами хранения данных, которые определены для хранения контрольных сумм с методом чередования. Данный уровень может сохранить данные при выходе из строя двух дис-

ков, скорость ввода-вывода информации достаточно высока [10]. Для данного уровня типичные преимущества: максимальная надёжность из вышеперечисленных RAID -массивов. Также можно выделить недостатки:

- большая стоимость;
- уменьшение производительности, если сравнивать с RAID уровня 5.

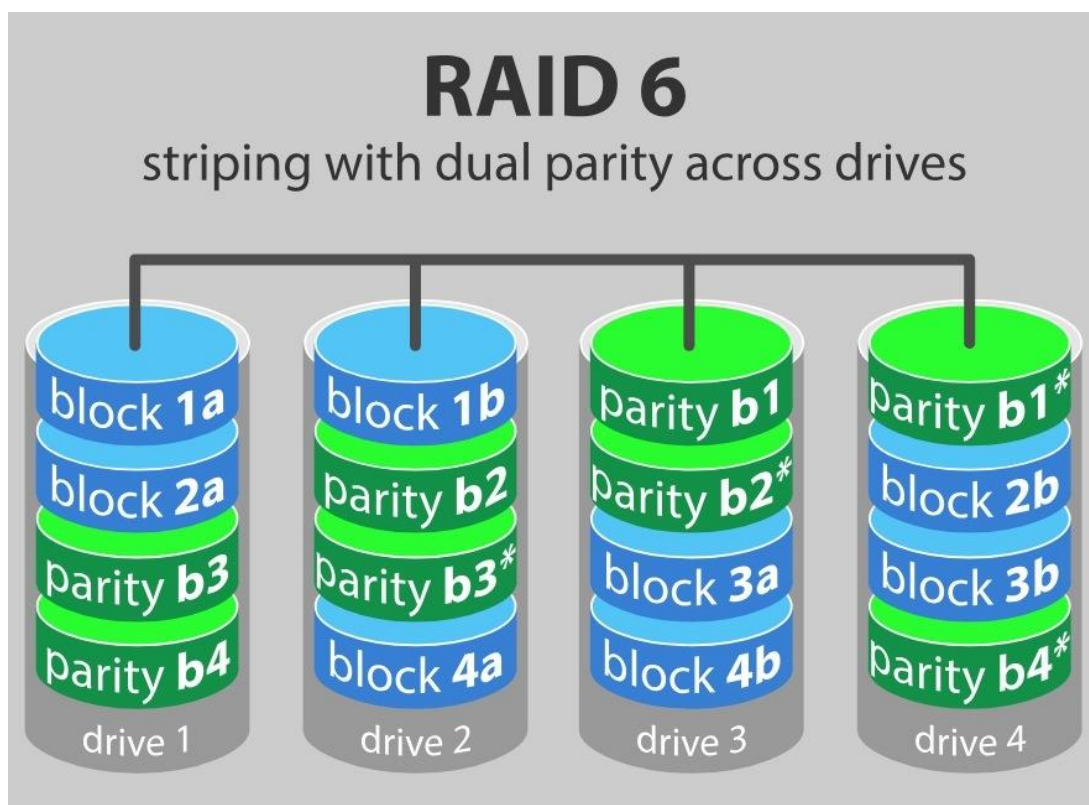


Рисунок 1.11 – RAID 6

2.2.8 RAID 7

RAID уровня 7 (рисунок 1.12) – это фирменный массив, который принадлежит компании «Storage Computer Corporation», в основе данного уровня лежит 3 и 4 уровень RAID-массива [11].

Преимущества:

- высокая отказоустойчивость;
- скорость обработки данных выше, чем у RAID уровня 3.

Недостатки:

- весьма трудоёмкая реализация;
- трудоёмкое восстановление данных;
- невысокая скорость записи данных;
- большая стоимость, которая вытекает из монополии создания контроллеров для данного массива.

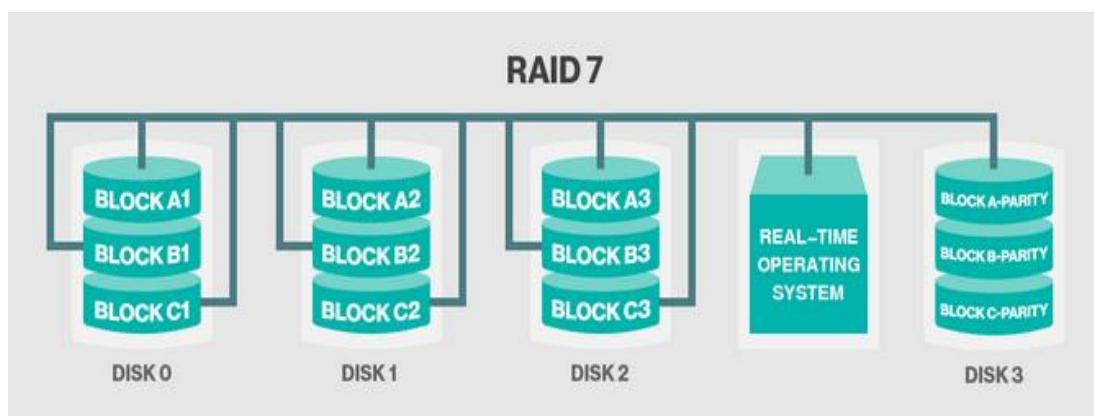


Рисунок 1.12 – RAID 7

2.3 Язык программирования

Для разработки будущего web-приложения был выбран высокоуровневый язык программирования Python. Он в сравнении с другими языками является достаточно простым в освоении, применим в широком спектре задач. Одним из плюсов является кроссплатформенность и использование в web-проектах, что и необходимо. Одной из его особенностей, является читабельностью и скоростью разработки. Так же данный язык, динамично и постоянно развивается, количество библиотек постоянно увеличивается, и также у него большое количество фреймворков, которые упрощают и ускоряют процесс разработки.

2.4 Фреймворк

Среди различных фреймворков для создания web-приложения, был выбран Flask. Данный фреймворк не перегружен библиотеками, которые не будут использованы, для небольшого проекта он более удобен. Главные черты Flask это минимализм и простота, также он позволяет выбирать только те компоненты, которые будут необходимы для создания конкретного проекта [12].

2.5 Операционная система

Операционная система была выбрана Ubuntu. Данный дистрибутив один из самых популярных дистрибутивов Linux. Из-за популярности Ubuntu имеет большое количество документации, так же стоит отметить стабильность, безопасность, постоянную поддержку различного оборудования, доступность различных программ и библиотек. Данный дистрибутив достаточно прост и гибок в настройке, по умолчанию не требует высоких системных требований.

3 Разработка

В данном разделе будут рассмотрены этапы при работе над практической частью дипломной работы, а именно разработка интерфейса для работы с RAID-массивами и разработка web-интерфейса для работы с ранее разработанным программным обеспечением для работы с RAID-массивами.

3.1 Интерфейс управления RAID-массивами

Для разработки программного обеспечения, с помощью которого можно управлять программными RAID-массивами будет использована утилита mdadm. Следует ознакомиться с её встроенными возможностями чтобы понимать на что будет способен будущий интерфейс.

Также нам необходимо будет работать с файловыми системами дисковых устройств для RAID-массивов и изменение типов разделов данных дисков, что означает и использование специальных утилит для этого. Для работы с файловыми системами достаточно будет встроенной утилиты mkfs, а для работы с типами разделов – fdisk.

При разработке интерфейса взаимодействия с данными утилитами на языке Python, в основном, будут использованы библиотеки os и subprocess. Они позволяют вызывать нужные нам утилиты от лица пользователя в автоматическом режиме.

3.1.1 Утилита mdadm

Утилита mdadm является утилитой для Linux-систем, которая позволяет работать с программными RAID-массивами, а именно: создавать, удалять и изменять по необходимости.

У утилиты mdadm есть несколько режимов работы:

– Assemble (сборка) – данный режим позволяет из ранее построенного массива собрать компоненты в массив. Составляющие построенного массива можно указывать или не указывать, если не указывать, то дальше идёт их поиск по суперблокам

– Build (построение) – данный режим позволяет построить массив из составляющих, у которых отсутствуют суперблоки. Создание и сборка не имеют различий, так как проверки не выполняются.

– Create (создание) – режим позволяет на основе устройств, которые указал пользователь создать новый массив, в котором суперблоки будут размещены на каждое устройство.

– Monitor /Follow (наблюдение) – режим позволяет просматривать изменения и откликаться на их состояние. Данный режим не целесообразен для линейных массивов и RAID 0, но для многоканальных типов RAID-массивов он применим и необходим.

– Grow (расширение или уменьшение) – режим, который позволяет сжать (уменьшить количество дисков), увеличить или переконфигурировать массив. С помощью данного режима возможно модифицировать размер, составляющих в RAID 1,2,3,5,6, и изменить численность действующих устройств в RAID 1.

– Incremental Assembly (инкрементальная сборка) – режим позволяет добавить устройство в массив.

– разнообразные операции по управлению массивом, такие как замена диска и пометка как сбойного;

– Manage (управление) – позволяет совершать разные операции по управлению массивом, подобные как добавление новых свободных дисков и удаление дисков, которые помечены, как сбойные.

– Misc (разное) – данный режим позволяет совершать операции с массивом, которые не могут совершать режимы, которые указаны выше. К примеру, остановка массивов, которые активны, изменять и просматривать суперблоки массива.

– Auto-detect (автообнаружение) – режим возможностью, которого является автоматическое обнаружение массивов в ядре Linux.

В нашем случае мы будем использовать, в большей части, режим Create, который будет создавать RAID-массив с выбранными пользователем параметрами: тип RAID, количество дисков и сами диски.

3.1.2 Утилиты fdisk и mkfs

Утилита fdisk является общим названием некой семьи утилит, позволяющих работать с разделами жёсткого диска. В нашем случае мы будем использовать её в специальном скрипте, который будет эмулировать работу пользователя и в автоматическом режиме выполнять необходимые действия с выбранными дисками.

Текст данного скрипта представлен в листинге 1.

Листинг 1 – Код скрипта для работы с утилитой fdisk

```
#!/bin/bash

fdisk $1 <<EOF
t
fd
w
EOF
exit 0
```

В листинге 1 каждая буква выполняет определённые действия:

- t – сменить тип раздела;
- fd – сменить тип раздела на Linux RAID autodetect;
- w – сохранить изменения и выйти из fdisk.

Утилита mkfs будет использована для создания файловых систем, которые будут выбраны пользователем, на выбранных пользователем дисках для создания

RAID-массива. В нашем случае синтаксис вызова данной команды будет выглядеть примерно следующим образом – `mkfs.file_system raid_name`. Названия, использованные в примере, имеют следующие обозначения:

- `file_system` – выбранный тип файловой системы;
- `raid_name` – название RAID-массива, на котором будет устанавливаться файловая система.

Используется утилита `mkfs`, как и `fdisk`, с помощью `bash`-скрипта, код которого представлен в листинге 2.

Листинг 2 – Код `bash`-скрипта для запуска утилиты `mkfs`

```
#!/bin/bash
```

```
$1 $2
```

Скрипт из листинга 2 запускается с помощью метода `subprocess.Popen` и имеет следующий вид: `subprocess.Popen([f'{path}/mkfs_helper.sh', f'mkfs.{filesystem}', f'{disk}'], stdout=subprocess.PIPE, stderr=subprocess.PIPE, text=True)`. Параметры для запуска скрипта описаны выше.

3.1.3 Библиотеки `os` и `subprocess`

Обе эти библиотеки позволяют работать с терминальными командами Linux-систем с помощью специальных методов.

Например, некоторые команды необходимо выполнять в режиме супер-пользователя, а для того, чтоб зайти в этот режим, необходимо выполнить следующий код, который представлен в листинге 2.

Листинг 3 – Код для входа в режим супер-пользователя

```
os.popen('sudo su', 'w').write(self.sudo_pass)
```

```
res = os.popen(f'команда_для_выполнения').read()
os.popen('exit')
```

В листинге 3 с помощью конструкции `os.popen('sudo su', 'w').write(self.sudo_pass)` выполняется вход в режим супер-пользователя, с помощью заранее введённого пароля пользователя, который сохраняется в поле `sudo_pass`.

Далее выполняется нужная нам команда, заносся результат в переменную `res`, чтоб можно было вернуть результат выполнения, а после выполнения операции выполняется выход из режима супер-пользователя с помощью конструкции `os.popen('exit')`, после чего можно возвращать результат работы.

Библиотеку `subprocess`, по некоторым техническим причинам, мы будем использовать только для запуска скрипта, представленного в листинге 1, с помощью следующей конструкции:

```
proc = subprocess.Popen([f'./utils/fdisk_helper.sh',
f'{disk}'], stdout=subprocess.PIPE), где:
```

- `proc` – переменная для сохранения результата работы скрипта;
- `disk` – название диска для смены его разделов на Linux `raid autodetect`.

3.2 Web-интерфейс взаимодействия с разработанным RAID-менеджером

При разработке web-интерфейса использовался микрофреймворк `Flask` и отдельные его компоненты: `request` и `render_template`.

Для более привлекательного вида страниц web-интерфейса была использована библиотека с заготовленными стилями `Bootstrap`.

В разработанном web-интерфейсе есть несколько основных страниц и путей, по которым они будут отображаться:

- «/» – отображение главной страницы по данному пути;
- «/create» – отображение страницы с формой для создания RAID-массива и вывода результата создания;
- «/control» – отображение страницы с формой для управления созданными RAID-массивом;
- «/status» – отображение страницы с состоянием работы RAID-массивов.

3.2.1 Компоненты микрофреймворка Flask – request и render_template

Компонент request нам необходим для обработки данных, которые отсылаются на сервер с клиента при заполнении форм для создания и управления RAID-массивами на определённых страницах web-приложения.

Метод render_template нам необходим для написания контроллера с выбранными путями для верного отображения страниц web-интерфейса. Также с помощью шаблонизатора, который входит в данный компонент, мы сможем генерировать страницы HTML с динамическими данными, передавая их при отображении того или иного шаблона.

3.2.2 Использование библиотеки Bootstrap

Библиотека Bootstrap является специальным инструментом для облегчения создания web-страниц, имея большой список заготовленных css-стилей для различных компонентов, таких как кнопки, формы, контейнеры, меню сайта и прочие.

В нашем случае мы использовали Bootstrap для облегчения и ускорения разработки дизайна web-интерфейса, а также для уменьшения вероятности неправильного отображения на различных мониторах, так как их заготовленные стили сразу поддерживают адаптивность.

3.3.3 Внешний вид страниц web-интерфейса

В разработанном web-интерфейсе существует 4 основные страницы - главная страница, страница для создания RAID, страница с информацией о RAID и страница с управлением созданных RAID.

На рисунках 3.1 – 3.4 представлены внешние виды вышеперечисленных страниц.

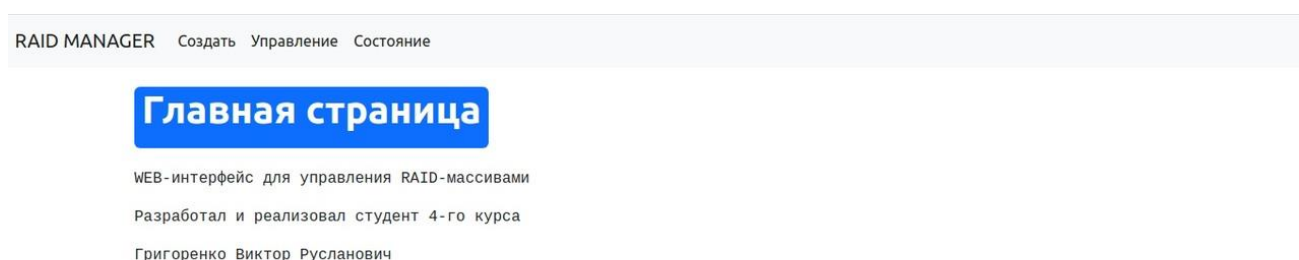


Рисунок 3.1 – Главная страница web-интерфейса

RAID MANAGER Создать Управление Состояние

Настройка

Вид рейда ▾ Устройства

☐ /dev/sdb 28,9G
☐ /dev/sdc 14,8G

Файловая система ▾

/dev/md Номер для RAID, который не

Создать

Результат работы

Создание RAID-массива может занять некоторое время.
Пожалуйста, не перезагружайте страницу для предотвращения критической ошибки во время создания RAID-массива

Рисунок 3.2 – Страница с созданием RAID

RAID MANAGER Создать Управление Состояние

Управление RAID-массивами

/dev/md Введите номер RAID для остановки

Остановить

Рисунок 3.3 – Страница для управления RAID-массивами

Статус RAID-массива

```
Personalities : [linear] [multipath] [raid0] [raid1] [raid6] [raid5] [raid4] [raid10]
md1 : active raid1 sdc[1] sdb[0]
15437824 blocks super 1.2 [2/2] [UU]

[>.....] resync = 2.0% (317952/15437824) finish=34.9min speed=7210K/sec
unused devices: <none>
```

Рисунок 3.4 – Страница с отображением статуса RAID-массивов

3.3 Сравнение создания RAID-массива

Сравним процесс создания RAID-массива с помощью выполнения необходимого списка команд в терминале и с помощью разработанного web-приложения.

3.3.1 Создание RAID-массива с помощью терминала

Разберём по шагам создание RAID-массива через терминал с использованием необходимых для этого утилит.

3.3.1.1 Выбор дисков

Перед созданием RAID-массива, необходимо выбрать жёсткие диски, которые мы хотим использовать в будущем RAID-массиве. Для этого можно использовать команду «lsblk».

Команда «lsblk» выведет нам информацию об имеющихся дисках, в том числе о созданных на них разделах, их размеры и точки монтирования.

Результат выполнения команды «lsblk» представлен на рисунке 3.5.

```
victor@GVR-Laptop:~$ lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
loop0        7:0      0   27,1M  1 loop /snap/chromium-ffmpeg/28
loop1        7:1      0    4K    1 loop /snap/bare/5
loop2        7:2      0   22,5M  1 loop /snap/chromium-ffmpeg/26
loop3        7:3      0  110,6M  1 loop /snap/core/12834
loop4        7:4      0    9M    1 loop /snap/canonical-livepatch/132
loop5        7:5      0    9M    1 loop /snap/canonical-livepatch/138
loop6        7:6      0   55,5M  1 loop /snap/core18/2344
loop7        7:7      0  111,7M  1 loop /snap/core/13250
loop8        7:8      0   55,5M  1 loop /snap/core18/2409
loop9        7:9      0   61,9M  1 loop /snap/core20/1434
loop10       7:10     0   65,2M  1 loop /snap/gtk-common-themes/1519
loop11       7:11     0  164,8M  1 loop /snap/gnome-3-28-1804/161
loop12       7:12     0  724,8M  1 loop /snap/pycharm-professional/282
loop13       7:13     0   54,2M  1 loop /snap/snap-store/558
loop14       7:14     0  254,1M  1 loop /snap/gnome-3-38-2004/106
loop15       7:15     0  724,8M  1 loop /snap/pycharm-professional/285
loop16       7:16     0   44,7M  1 loop /snap/snapd/15904
loop17       7:17     0  248,8M  1 loop /snap/gnome-3-38-2004/99
loop18       7:18     0   82,9M  1 loop /snap/discord/132
loop19       7:19     0   81,3M  1 loop /snap/gtk-common-themes/1534
loop20       7:20     0    71M   1 loop /snap/flameshot/180
loop21       7:21     0   44,7M  1 loop /snap/snapd/15534
loop22       7:22     0  424,2M  1 loop /snap/kde-frameworks-5-qt-5-15-3-core20/8
loop23       7:23     0   61,9M  1 loop /snap/core20/1494
loop24       7:24     0   82,9M  1 loop /snap/discord/135
sda          8:0      0  931,5G  0 disk
├─sda1        8:1      0    16M  0 part
├─sda2        8:2      0 683,6G  0 part /windowsd
├─sda3        8:3      0   93,1G  0 part /
└─sda4        8:4      0   61,5G  0 part /home
sdc          8:32     1   14,8G  0 disk
sde          8:64     1   28,9G  0 disk
nvme0n1     259:0    0  238,5G  0 disk
├─nvme0n1p1  259:1    0   100M  0 part /boot/efi
├─nvme0n1p2  259:2    0    16M  0 part
├─nvme0n1p3  259:3    0  237,9G  0 part /windowssc
└─nvme0n1p4  259:4    0   521M  0 part
```

Рисунок 3.5 – Результат выполнения команды «lsblk»

3.3.1.2 Изменение типа разделов

Для того чтобы создать RAID-массив на ранее выбранных нами дисках, - в нашем случае это «/dev/sdc» и «/dev/sde» -, необходимо установить на них специальный тип раздела – «Linux raid autodetect».

Установить необходимый нам тип раздела можно с помощью утилиты «fdisk». Этот процесс занимает довольно длительный промежуток времени, зависящий от характеристик компьютера, в том числе и выбранных дисков.

Процесс применения и результат работы утилиты «fdisk» представлен на рисунках 3.6 и 3.7.

```

victor@GVR-Laptop:~$ sudo fdisk /dev/sde
[sudo] пароль для victor:

Добро пожаловать в fdisk (util-linux 2.34).
Изменения останутся только в памяти до тех пор, пока вы не решите записать их.
Будьте внимательны, используя команду write.

Команда (м для справки): o
Создана новая метка DOS с идентификатором 0x15749fd2.

Команда (м для справки): n
Тип раздела
  р    основной (0 первичный, 0 расширенный, 4 свободно)
  е    расширенный (контейнер для логических разделов)
Выберите (по умолчанию - р): р
Номер раздела (1-4, по умолчанию 1): 1
Первый сектор (2048-60628991, по умолчанию 2048): 2048
Last sector, +/-sectors or +/-size[K,M,G,T,P] (2048-60628991, по умолчанию 60628991): 60628991

Создан новый раздел 1 с типом 'Linux' и размером 28,9 GiB.
Раздел #1 содержит сигнатуру ext4.

Удалить сигнатуру? [Y] Да/[N] Нет: y

Сигнатура будет удалена командой записи.

Команда (м для справки): t
Выбранный раздел 1
Шестнадцатеричный код (введите L для получения списка кодов): 1
Тип раздела 'Linux' изменен на 'FAT12'.

Команда (м для справки): p
Диск /dev/sde: 28,93 GiB, 31042043904 байт, 60628992 секторов
Disk model: USB DISK 2.0
Единицы: секторов по 1 * 512 = 512 байт
Размер сектора (логический/физический): 512 байт / 512 байт
Размер I/O (минимальный/оптимальный): 512 байт / 512 байт
Тип метки диска: dos
Идентификатор диска: 0x15749fd2

```

Рисунок 3.6 – Процесс работы с утилитой «fdisk»

```

Команда (м для справки): w

Таблица разделов была изменена.
Вызывается ioctl() для перечитывания таблицы разделов.
Синхронизируются диски.

victor@GVR-Laptop:~$

```

Рисунок 3.7 – Результат работы с утилитой «fdisk»

Команды, которые были использованы при работе с утилитой «fdisk», уже были описаны выше в пункте 3.1.2.

3.3.1.3 Создание RAID-массива

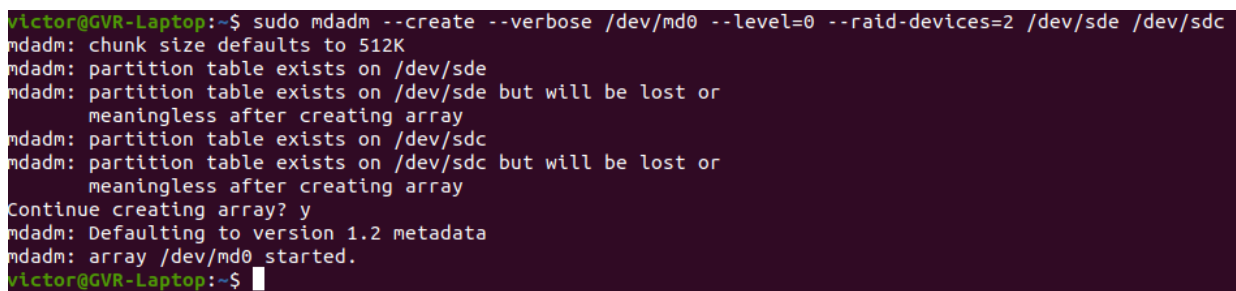
После того, как необходимые нам диски выбраны и их тип разделов изменён на «Linux raid autodetect», можно приступать к непосредственному созданию RAID-массива на этих дисках.

RAID-массив будем создавать с помощью утилиты «mdadm». Команда для создания RAID-массива на выбранных нами дисках выглядит следующим обра-

зом: «mdadm --create --verbose /dev/md0 --level=0 --raid-devices=2 /dev/sde /dev/sdc», где:

- «--create» – ключ создания RAID-массива;
- «/dev/md0» – название будущего RAID-массива;
- «--level=0» – уровень будущего RAID-массива (в нашем случае RAID 0);
- «--raid-devices=2» – количество использованных устройств для будущего RAID-массива;
- «/dev/sde» и «/dev/sdc» – перечисление устройств для будущего RAID-массива.

Результат выполнения данной команды можно увидеть на рисунке 3.8.



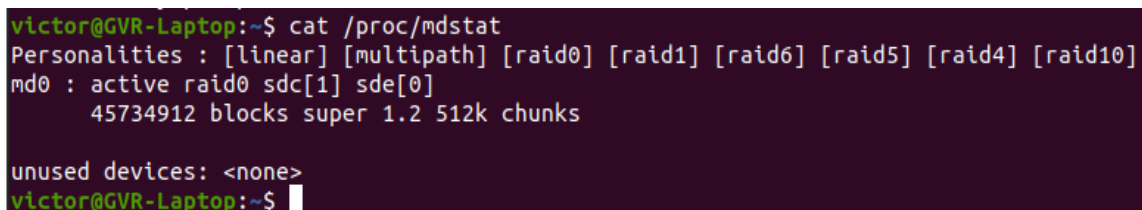
```
victor@GVR-Laptop:~$ sudo mdadm --create --verbose /dev/md0 --level=0 --raid-devices=2 /dev/sde /dev/sdc
mdadm: chunk size defaults to 512K
mdadm: partition table exists on /dev/sde
mdadm: partition table exists on /dev/sde but will be lost or
      meaningless after creating array
mdadm: partition table exists on /dev/sdc
mdadm: partition table exists on /dev/sdc but will be lost or
      meaningless after creating array
Continue creating array? y
mdadm: Defaulting to version 1.2 metadata
mdadm: array /dev/md0 started.
victor@GVR-Laptop:~$
```

Рисунок 3.8 – Результат создания RAID-массива

3.3.1.4 Проверка состояния созданного RAID-массива

После создания RAID-массива необходимо проверить правильно ли он функционирует и функционирует ли вовсе. Сделать это можно с помощью команды «cat /proc/mdstat».

Результат выполнения данной команды представлен на рисунке 3.9.



```
victor@GVR-Laptop:~$ cat /proc/mdstat
Personalities : [linear] [multipath] [raid0] [raid1] [raid6] [raid5] [raid4] [raid10]
md0 : active raid0 sdc[1] sde[0]
      45734912 blocks super 1.2 512k chunks

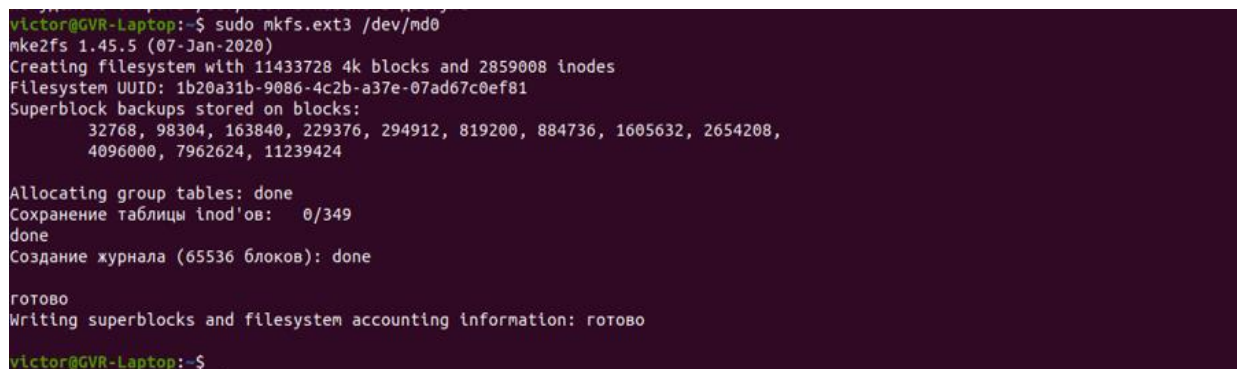
unused devices: <none>
victor@GVR-Laptop:~$
```

Рисунок 3.9 – Результат выполнения команды

3.3.1.5 Создание файловой системы на RAID-массиве

После создания и проверки работоспособности RAID-массива необходимо установить на него файловую систему. Сделать это можно с использованием утилиты «mkfs». Данный процесс, как и работа с утилитой «fdisk» в пункте 3.3.1.2, занимает немалый промежуток времени и может исчисляться несколькими минутами.

Процесс работы с утилитой «mkfs» и результат её работы представлены на рисунке 3.10.



```
victor@GVR-Laptop:~$ sudo mkfs.ext3 /dev/md0
mke2fs 1.45.5 (07-Jan-2020)
Creating filesystem with 11433728 4k blocks and 2859008 inodes
Filesystem UUID: 1b20a31b-9086-4c2b-a37e-07ad67c0ef81
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
    4096000, 7962624, 11239424

Allocating group tables: done
Сохранение таблицы inode'ов: 0/349
done
Создание журнала (65536 блоков): done

готово
Writing superblocks and filesystem accounting information: готово
victor@GVR-Laptop:~$
```

Рисунок 3.10 – Процесс и результат работы утилиты «mkfs»

3.3.1.6 Создание конфигурационного файла

После создания RAID-массива и установки на него файловой системы, необходимо создать конфигурационный файл «mdadm.conf», в котором хранится информация для системы о том какие RAID-массивы нужно создавать и какие компоненты входят в данные RAID-массивы.

С помощью команды «echo "DEVICE partitions" > /etc/mdadm/mdadm.conf» мы можем создать конфигурационный файл. С помощью команды «mdadm --detail --scan --verbose | awk '/ARRAY {print}' >> /etc/mdadm/mdadm.conf» мы можем записать необходимую информацию, полученную из утилиты «mdadm», сразу в конфигурационный файл.

Результат создания конфигурационного файла и запись в него необходимых строк представлен на рисунке 3.11.

```
root@GVR-Laptop:~# echo "DEVICE partitions" > /etc/mdadm/mdadm.conf
root@GVR-Laptop:~# mdadm --detail --scan --verbose | awk '/ARRAY/ {print}' >> /etc/mdadm/mdadm.conf
```

Рисунок 3.11 – Результат создания конфигурационного файла и запись в него необходимых строк

3.3.1.7 Монтирование RAID-массива

После пройденных этапов создания установки необходимых типов разделов на выбранные диски, создания RAID-массива и установки на него файловой системы, необходимо смонтировать его для дальнейшей работы с RAID-массивом.

Для этого необходимо сначала создать папку для монтирования RAID-массива. Сделать это можно с помощью команды «mkdir /raid», которая создаст папку «/raid», в которую мы будем монтировать наш RAID-массив.

Смонтировать RAID-массив в папку можно с помощью команды «mount /dev/md0 /raid». Создание папки для монтирования представлено на рисунке 3.12, а монтирование RAID-массива представлено на рисунке 3.13.

```
victor@GVR-Laptop:~$ sudo mkdir /raid
victor@GVR-Laptop:~$
```

Рисунок 3.12 – Создание папки /raid для монтирования RAID-массивов

```
root@GVR-Laptop:/home/victor# mount /dev/md0 /raid
root@GVR-Laptop:/home/victor# cat /proc/mdstat
Personalities : [linear] [multipath] [raid0] [raid1] [raid6] [raid5] [raid4] [raid10]
md0 : active raid0 sdc[1] sde[0]
      45734912 blocks super 1.2 512k chunks

unused devices: <none>
root@GVR-Laptop:/home/victor#
```

Рисунок 3.13 – Результат монтирования RAID-массива и проверка его состояния

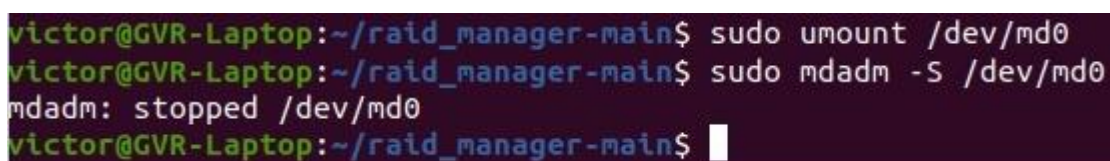
3.3.1.8 Размонтирование и остановка RAID-массива

Когда созданный RAID-массив будет уже не нужен нам, необходимо его размонтировать и остановить. Это необходимо для предотвращения ненужной нагрузки на выбранные диски, которая может им навредить со временем.

Размонтирование созданного RAID-массива производится с помощью команды «`sudo umount /dev/md0`», где «`/dev/md0`» – имя RAID-массива. Остановка работающего RAID-массива производится с помощью команды «`sudo mdadm -S /dev/md0`», где:

- «`-S`» – ключ остановки работающего RAID-массива;
- «`/dev/md0`» – имя работающего RAID-массива.

Результат размонтирование и остановки работы RAID-массива представлен на рисунке 3.14.



```
victor@GVR-Laptop:~/raid_manager-main$ sudo umount /dev/md0
victor@GVR-Laptop:~/raid_manager-main$ sudo mdadm -S /dev/md0
mdadm: stopped /dev/md0
victor@GVR-Laptop:~/raid_manager-main$
```

Рисунок 3.14 – Результат размонтирования и остановки RAID-массива

3.3.2 Создание RAID-массива с помощью web-приложения

Разберём по шагам создание RAID-массива с помощью разработанного web-приложения и сделаем выводы насколько это быстрее и удобнее.

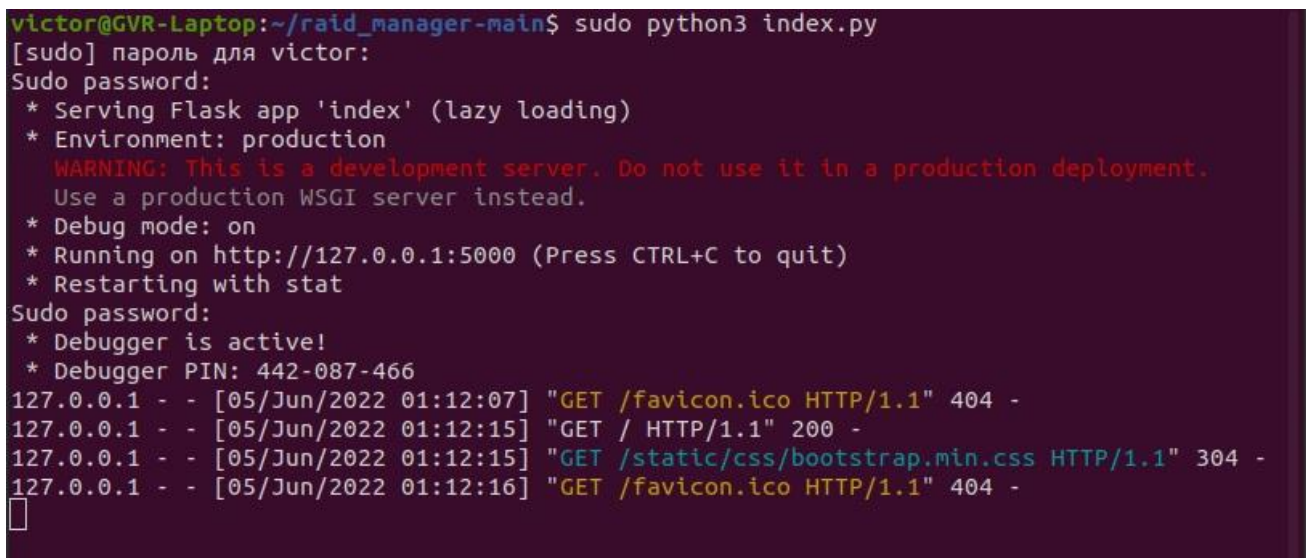
3.3.2.1 Запуск web-приложения

Для того чтоб начать работать с разработанным web-приложением управления RAID-массивами, его необходимо сначала запустить в терминале.

Запуск приложения происходит с помощью команды «`sudo python3 /путь/index.py`». Эта команда запустит сервер приложения, который будет обрабатывать входящие запросы с клиента.

После запуска сервера приложений, web-приложение попросит ввести пароль пользователя для выполнения необходимых действий от лица супер-пользователя.

Результат запуска web-приложения представлен на рисунке 3.15.



```
victor@GVR-Laptop:~/raid_manager-main$ sudo python3 index.py
[sudo] пароль для victor:
Sudo password:
* Serving Flask app 'index' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://127.0.0.1:5000 (Press CTRL+C to quit)
* Restarting with stat
Sudo password:
* Debugger is active!
* Debugger PIN: 442-087-466
127.0.0.1 - - [05/Jun/2022 01:12:07] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [05/Jun/2022 01:12:15] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [05/Jun/2022 01:12:15] "GET /static/css/bootstrap.min.css HTTP/1.1" 304 -
127.0.0.1 - - [05/Jun/2022 01:12:16] "GET /favicon.ico HTTP/1.1" 404 -
```

Рисунок 3.15 – Результат запуска web-приложения

3.3.2.2 Заполнение формы и создание RAID-массива

Далее необходимо запустить браузер и перейти по адресу «127.0.0.1:5000» для загрузки web-приложения. После перейти во вкладку «Создание» в меню сверху.

На вкладке «Создание» необходимо заполнить форму для создания будущего RAID-массива с нужными характеристиками. После заполнения формы остаётся нажать на кнопку «Создать» и подождать пока страница перезагрузится, отдав ответ о результате создания RAID-массива.

Заполненная форма для создания RAID-массива и результат его создания представлены на рисунках 3.16 и 3.17 соответственно.

RAID MANAGER Создать Управление Состояние

Настройка

RAID 1

Устройства

☒ /dev/sdb 28,9G
☒ /dev/sdc 14,8G

ext4

/dev/md 1

Создать

Результат работы

Создание RAID-массива может занять некоторое время.

Пожалуйста, не перезагружайте страницу для предотвращения критической ошибки во время создания RAID-массива

Рисунок 3.16 – Заполненная форма для создания RAID-массива

RAID MANAGER Создать Управление Состояние

Настройка

Вид рейда

Устройства

☐ /dev/sdb 28,9G
☐ /dev/sdc 14,8G

Файловая система

/dev/md Номер для RAID, который не

Создать

Результат работы

```
class Exception >
Создание точки монтирования для RAID-массива и монтирование его:
RAID-массив /dev/md1 успешно смонтирован в /raid
Создание конфигурационного файла и запись в него необходимых данных:
Файл успешно обновлён
```

Рисунок 3.17 – Результат создания RAID-массива

На рисунке 3.18 можно увидеть результат создания RAID-массива при помощи встроенной в Ubuntu программы для работы с дисковыми устройствами.

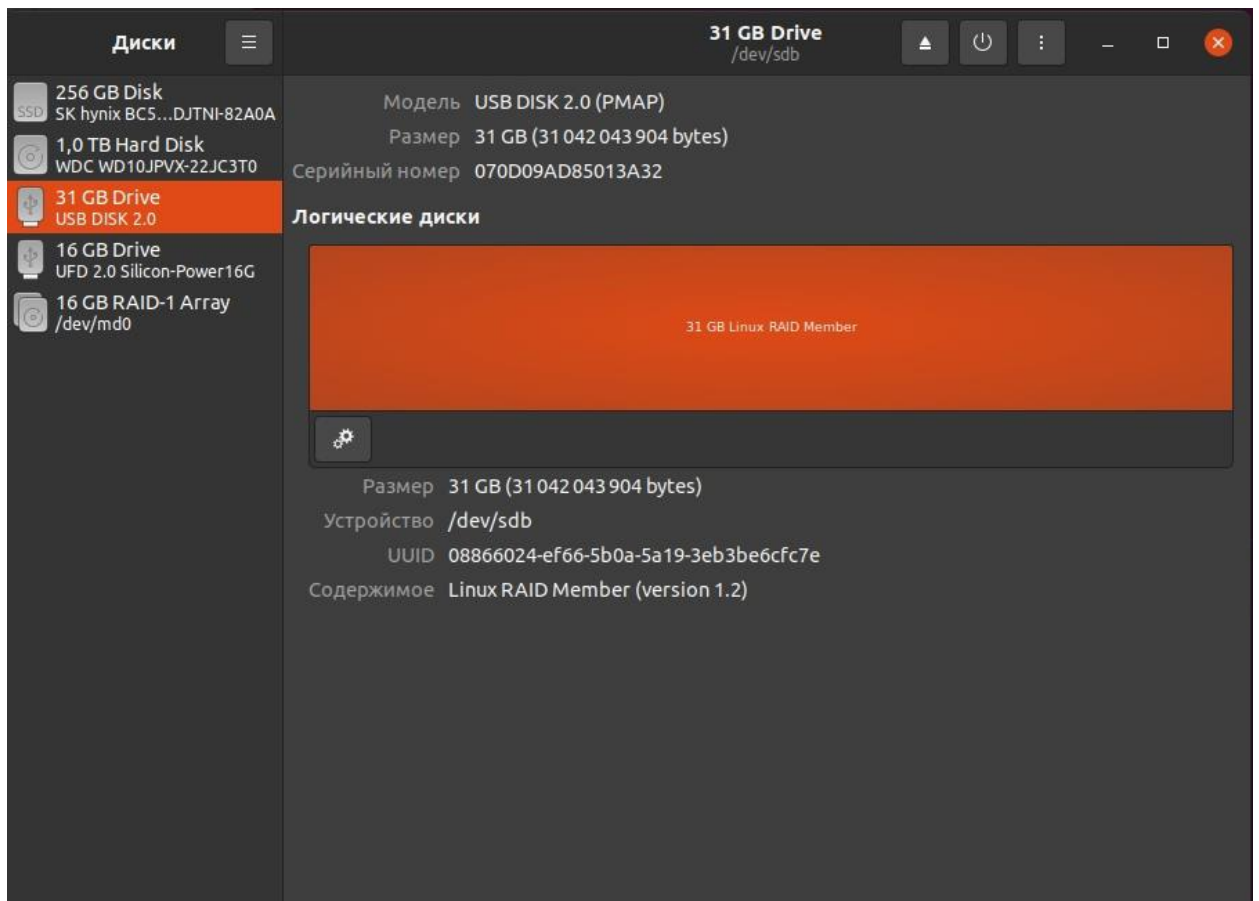


Рисунок 3.18 – Результат создания RAID-массива

В конечном итоге, многие действия, занимающие много времени и концентрации внимания, выполняются за несколько секунд и требуется лишь подождать пока создастся RAID-массив. В автоматическом режиме выполняются:

- размонтирование и изменение типа разделов выбранных дисков на «Linux raid autotect»;
- создание выбранного типа RAID-массива на выбранных устройствах;
- установка выбранной файловой системы на созданный RAID-массив;
- монтирование созданного RAID-массива;
- создание и обновление конфигурационного файла.

Исходя из вышесказанного, можно сделать вывод, что разработанное web-приложение значительно сокращает время для создания RAID-массива, а также упрощает сам процесс создания.

3.3.2.3 Проверка состояния RAID-массива

Чтобы проверить состояние созданного RAID-массива, необходимо перейти во вкладку «Состояние» в меню сверху.

Результат вывода состояния созданного RAID-массива представлен на рисунке 3.19.



Рисунок 3.19 – Результат вывода состояния созданного RAID-массива

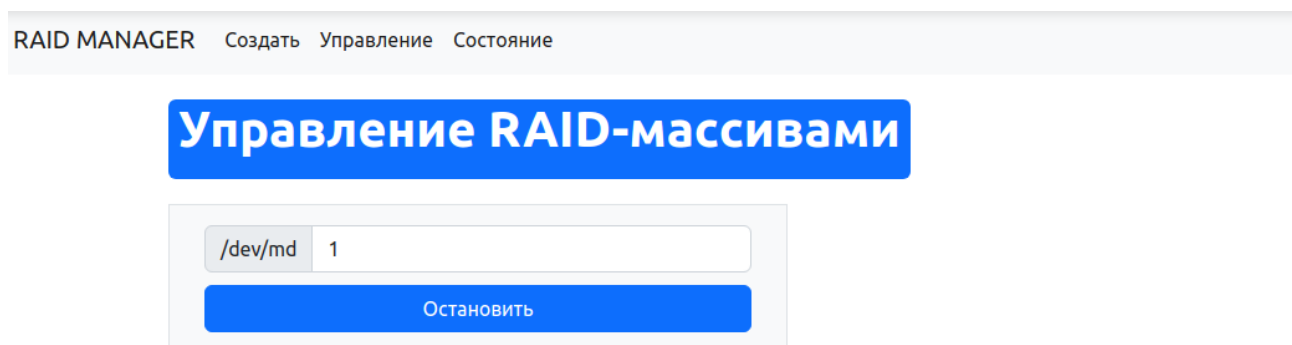
В данном случае затраченное время на получение статуса созданного RAID-массива не особо отличается от затраченного времени на получение статуса созданного RAID-массива в терминале, но информация не засоряет экран и лаконично представлена пользователю.

3.3.2.4 Размонтирование и остановка RAID-массива

Чтобы размонтировать и остановить работающий RAID-массив, необходимо перейти во вкладку «Управление» в меню сверху. На данной вкладке необхо-

димо написать номер работающего RAID-массива, который вы хотите размонтировать и остановить, после чего нажать кнопку «Остановить».

Заполненная форма для остановки работающего RAID-массива представлена на рисунке 3.20.



RAID MANAGER Создать Управление Состояние

Управление RAID-массивами

/dev/md 1

Остановить

Рисунок 3.20 – Заполненная форма для остановки работающего RAID-массива

В данном случае время, затраченное на размонтирование и остановку работающего RAID-массива через терминал и через разработанное web-приложение не особо различается. Однако у web-приложения существует явное преимущество, так как достаточно просто ввести номер нужного работающего RAID-массива и нажать на кнопку «Остановить».

Заключение

В ходе выполнения ВКР были изучены способы создания, управления и мониторинга состояния RAID-массива с помощью терминала Ubuntu, результаты которого легли основой для создания интерфейса для создания интерфейса для работы с RAID-массивом на языке программирования Python.

Исходя из поставленных задач в разделе «Постановка задачи», в котором были также описаны средства реализации web-приложения и цели его создания, разработанное web-приложение и его функционал выполняют их.

Анализ предметной области позволил выделить главные сущности разрабатываемого web-приложения, также определить терминологию, используемую в ВКР, описать выбор языка программирования, фреймворка и операционной системы.

Языком программирования был выбран Python, ведь он удовлетворяет всем нашим требованиям, а также имеет необходимые нам модули для работы с операционной системой – `os` и `subprocess`, позволившие реализовать интерфейс для работы с RAID-массивами, и фреймворком Flask, который позволил нам реализовать web-приложение и связать его с интерфейсом для работы с RAID-массивами.

Раздел с реализацией web-приложения отчётливо показывает, что все поставленные цели к разрабатываемому web-приложению достигнуты в полном объёме. А сравнение со способом создания, управления и мониторинга состояния RAID-массива через терминал показывает, что разработанное web-приложение имеет явные преимущества по времени и простоте использования.

Список используемых источников

- 1 – Практические советы по созданию RAID-массивов на домашних ПК [Электронный источник] – URL: <https://compress.ru/article.aspx?id=21065#01> (Дата обращения: 25.03.2022);
- 2 – Что такое RAID-массивы [Электронный ресурс] – URL: <https://eternalhost.net/blog/sistemnoe-administrirovanie/raid-massiv#p4> (Дата обращения: 25.03.2022);
- 3 – Введение в raid: основные термины и подходы [Электронный ресурс] – URL: <https://www.8host.com/blog/vvedenie-v-raid-osnovnye-terminy-i-podxody/> (Дата обращения: 27.03.2022);
- 4 – RAID Levels [Электронный ресурс] – URL: <https://www.ixbt.com/storage/raids.html> (Дата обращения: 27.03.2022);
- 5 – Подробное знакомство с RAID-массивами [Электронный ресурс] – URL: <https://www.ferra.ru/review/computers/s26107.htm> (Дата обращения: 27.03.2022);
- 6 – Урок 1. Web-приложение: понятие, компоненты и принципы работы [Электронный ресурс] – URL: <https://smartiqa.ru/courses/web/lesson-1> (Дата обращения: 27.03.2022);
- 7 – Стек технологий для разработки веб-приложений: что важно знать бизнесу [Электронный ресурс] – URL: <https://www.azoft.ru/blog/web-development-stack/> (Дата обращения: 01.04.2022);
- 8 – Описание RAID массивов (RAID 4, RAID 5, RAID 6, RAID 7) [Электронный ресурс] – URL: <https://www.nstor.ru/ru/catalog/StorageSystems/info/93.html> (Дата обращения: 05.04.2022);
- 9 – Flask против Django: почему Flask может быть лучше [Электронный ресурс] – URL: <https://python-scripts.com/flask-vs-django> (Дата обращения: 07.04.2022).