# Описание ПО системы хранения данных BAUM STORAGE AI

Программное обеспечение системы хранения данных BAUM STORAGE AI предназначено для надежного, централизованного хранения больших объёмов пользовательских данных и предоставления доступа к этим данным по блочным и файловым протоколам.

## 1. Описание работы системы

### 1.1 Программная архитектура.

В системе используется микросервисная архитектура. Программное обеспечение состоит из отдельных программных модулей - микросервисов, которые реализуют определенный набор функций управления аппаратным обеспечением через передачу команд управления ядру операционной системы, используя интерфейс командной строки, либо посредствам системных утилит. Программные модули представляют собой событийно-управляемые асинхронные программы, реализованные по единой схеме, в которых для обработки приходящих сообщений реализована «машина состояний». Модули выполняют обмен сообщениями с интерфейсом пользователя и друг с другом.

В общем случае, механизм управления работает следующим образом:

1. Пользователь, в графическом интерфейсе управления системой, изменяет некоторые параметры формируя командую последовательность, программный модуль управляющий пользовательским интерфейсом инициирует отправку управляющего сообщения одному из модулей, отвечающих за различные параметры аппаратного обеспечения. В сообщении содержится имя вызываемого метода, реализованного в этом программном модуле и необходимые параметры.

2. Программный модуль получивший сообщение, в свою очередь отправляет некоторую последовательность команд операционной или файловой системам.

3. Операционная /файловая система возвращает результат выполнения команды, который определенным образом интерпретируется программным модулем. Далее модуль формирует ответное сообщение и направляет его модулю – отправителю, в данном случае модулю пользовательского интерфейса. Изменение параметров системы также заносится в хранилище конфигурации и регистрируется в системных логах.

4. Получив результат управляющего воздействия, программный модуль пользовательского интерфейса интерпретирует его и выводит в том или ином виде в интерфейс пользователя.

Модель взаимодействия программных модулей программно-аппаратного комплекса представлена на следующей диаграмме.

При проектировании программной архитектуры BAUM Storage AI за основу была взята архитектура межсервисного взаимодействия, реализующая асинхронную модель обмена сообщениями. Каждый сервис (программный модуль) реализует свой программный интерфейс приложения (API), через который выполняется управление, а также интерфейс JRPC (JSON RPC) представляющий собой TCP сервер и клиента.

Все программные модули обмениваются между собой сообщениями - запросами, содержащими вызовы команд внутреннего API. Запросы имеют формат JSON пакетов, которые в общем случае содержат следующую информацию:

- Имя источника запроса;
- Имя получателя запроса;
- Имя API метода программного модуля и его параметры;
- Идентификатор запроса (action_ID).

## 1.2 Управление хранилищем.

Для управления хранилищем используется файловая система ZFS (Zettabyte File System, которая была доработана под задачи проекта. ZFS является проектом с открытым исходным кодом и лицензируется под CDDL (Common Development and Distribution License). Использование ZFS снимает необходимость применения специализированных RAID контроллеров для создания отказоустойчивых дисковых массивов. Все диски системы могут быть объединены в один или несколько дисковых пулов, резервирование и контроль целостности данных на которых выполняется самой файловой системой ZFS. В отличие от традиционных файловых систем, которые располагаются на одном устройстве, ZFS строится поверх виртуальных пулов хранения данных. Пул строится из виртуальных устройств, каждое из которых является либо физическим устройством, либо зеркалом (RAID 1) одного или нескольких устройств, либо группой устройств (RAID 5, 6). Это позволяет собирать пулы с разными уровнями отказоустойчивости (RAID 1, 5, 6, 10, 50, 60). Ёмкость всех виртуальных устройств затем доступна для создания ресурсов хранения -

томов и файловых систем. Схема объединения дисковых накопителей в пулы показана на рисунке ниже:

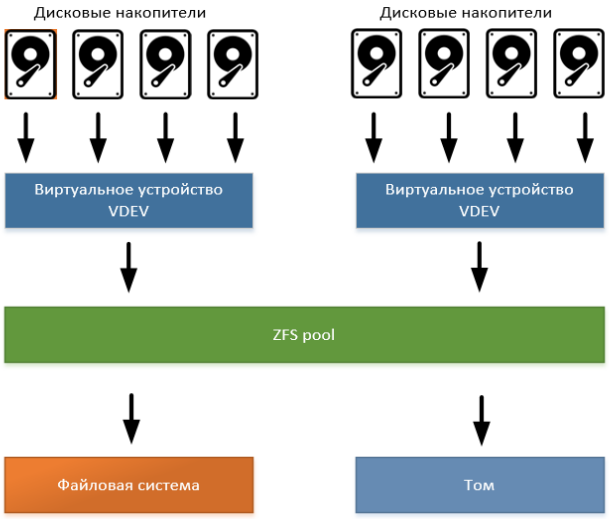


**Рисунок 1. Схема организации хранения данных**

## 1.3 Доступ к ресурсам хранилища.

Доступ к ресурсам хранилища предоставляется по файловым протоколам доступа NFS и SMB. Для автоматизации настройки выделения ресурсов по этим протоколам в системе используются программные модули, управляющие привязкой файловых систем к протоколам и установкой разрешений на доступ к файлам. Используется два режима контроля доступа: фильтрация клиентов по IP-адресу и импорт пользователей из служб Active Directory и LDAP. Также имеется возможность интеграции в СХД компонента поддержки шифрования файлов. Для работы модуля шифрования файлов, на компьютере пользователя должен быть установлен программный компонент, обеспечивающий автоматизацию работы с ключами шифрования и управление шифрованием файлов. После активации режима шифрования папки, дальнейшие действия по шифрованию и расшифровке файлов происходит в автоматическом режиме, прозрачно для пользователя. Поддержку шифрования на стороне СХД выполняет модуль шифрования файлов.

## 1.4 Хранение конфигурации СХД.

Задача хранения настроек системы и их защиту от несогласованных изменений на разных узлах кластера решается при помощи централизованного хранении всей конфигурационной информации в одном месте и получения доступа к ней при помощи специализированного модуля хранения конфигурации. Этот программный модуль решает задачу согласованности записываемых настроек конфигурации при помощи использования

транзакций. Для хранения конфигурационной информации используется СУБД SQLite, реплицируемая между узлами кластера. При записи или чтении конфигурационных настроек программными модулями комплекса, они обмениваются с модулем хранения конфигурации пакетами данных, в которых содержится набор конфигурационных параметров. При этом все действия по сохранению либо чтению необходимых внешним модулям параметров выполняет только модуль хранения конфигурации.

## 1.5 Описание принципа управления компонентами системы.

Задача управления сервисами операционной системы решена при помощи разработки специализированных программных модулей, принимающих команды управления и выполняющих определенную последовательность команд управления над подчиненными им системными сервисами, через интерфейс командной строки операционной системы (CLI). Каждый такой модуль реализует некоторый набор методов - команд внутреннего программного интерфейса приложения (API), которые могут быть вызваны удаленно. Команду на выполнение того или иного метода и её параметры, программный модуль получает в управляющем пакете. После выполнения команды, программный модуль читает стандартный поток вывода и формирует ответный пакет на основании полученных данных о результате работы команды.

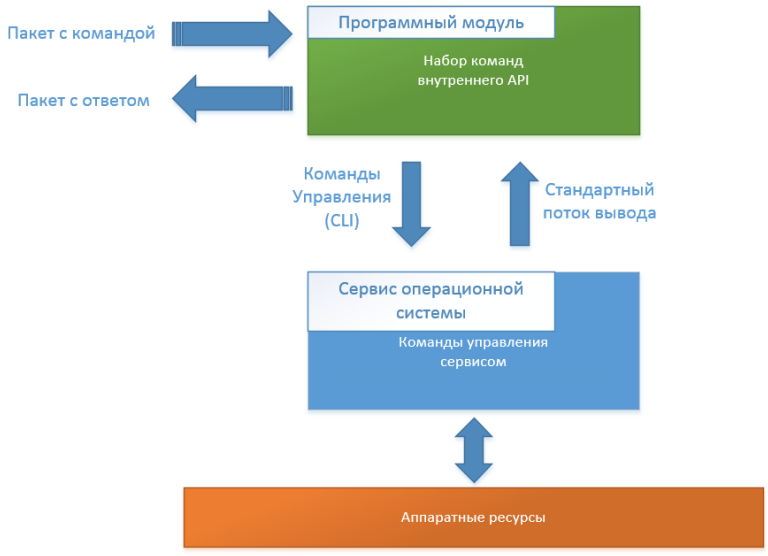


**Рисунок 2. Схема управления сервисами операционной системы**

## 2. Описание программных модулей

**Модуль RPC_router** отвечает за пересылку сообщений (пакетов) между модулями. Получив пакет от модуля – отправителя, RPC_router читает в пакете имя модуля получателя и пересылает ему пакет. В системе используются пакеты двух типов: пакеты уведомления, не предполагающие получения ответа и пакеты запросы на которые ответ обязателен.

Модуль, получивший запрос извлекает из него имя выполняемого метода и его параметры, и выполнив его отправляет обратно ответ. Для отслеживания пакетов RPC_router присваивает им уникальные четырёхзначные идентификаторы. После пересылки ответа на запрос идентификатор освобождается и может быть выдан другому запросу. При передаче пакетов между узлами (нодами) кластера, модули RPC_router обмениваются пакетами через сетевой интерконнект – прямое соединение сетевых интерфейсов двух контроллеров.

**Модуль Веб-сервер** – обеспечивает взаимодействие пользователя с СХД посредствам графического интерфейса управления (GUI). Веб-сервер реализован на основе ПО Node.js. Модуль Веб-сервер для взаимодействия с другими модулями отправляет им JSON пакеты, содержащие имя требуемого метода программного интерфейс API и дополнительные параметры. Обратно передается результат выполненных действий. Запуск Веб-сервера и контроль его работоспособности выполняет разработанный для этих целей демон baum_web_ui. При «зависании» модуля Веб-сервера, демон baum_web_ui выполняет его перезагрузку.

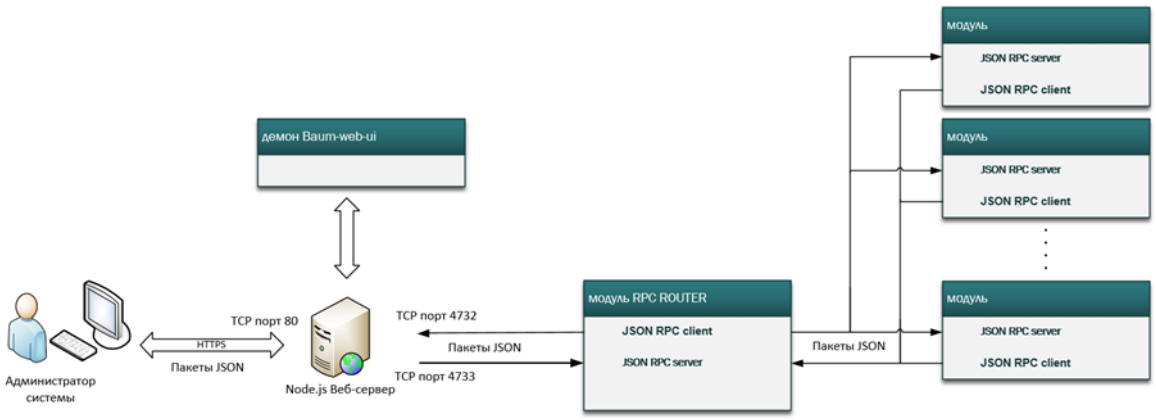


**Рисунок 3. Реализация интерфейса пользователя**

Обмен информацией между web-сервером Node.js и клиентом реализован на основе архитектуры взаимодействия компонентов распределенного приложения RESTful API (Representational State Transfer - «передача состояния представления»). Для соединения с клиентом используется порт TCP/80. После установки соединения происходит переключение на протокол HTTPS.

Получив команду от интерфейса управления, Веб-сервер формирует и отправляет управляющий пакет к указанному в команде модулю. После выполнения команды модуль-получатель отправляет JSON пакет с ответом обратно веб-серверу. Веб-сервер, в свою очередь, формирует JSON пакет и отправляет его клиенту.

Сценарий запуска веб-сервера следующий: после старта демона baum-web-ui он запускает процесс Node.js, который регистрируется в модуле RPC router, отправляет запрос на инициализацию модулю Log_service, получает список контроллеров (нод) кластера запрашивая модуль Heartbeet, после чего запускает веб-сервер на управляющем интерфейсе (данные об интерфейсе он получает от модуля BNCD). Далее модуль Веб-сервер посылает запрос списка ресурсов модулю Config Keeper и запускает цикл опроса статуса системы. При возникновении ошибки в процессе выполнения этого сценария, он выполняется повторно.

**Модуль Config Keeper** выполняет обработку запросов на сохранение и чтение конфигурационных параметров всех модулей в базе конфигурации. Модуль получает и отправляет запросы (JSON пакеты) через интерфейс JRPC. Конфигурационные параметры программных модулей системы (сервисов) хранятся централизованно в базе данных конфигурации, реализованной на базе ПО SQL lite. При получении запроса от других модулей системы, модуль Config Keeper отправляет запрос базе данных. Записав или получив данные из БД, модуль отправляет их в ответном пакете.

**Модуль Bpool** отвечает за управление хранилищем данных, выполняя конфигурирование файловой системы ZFS. Получение и отправка JSON пакетов выполняется через интерфейс JRPC.

Получив команду, поступившую от другого программного модуля, например, от веб-сервера, модуль Bpool отправляет серию команд управления файловой системе. Выполнив запрошенное действие, программный модуль Bpool отправляет ответный пакет с уведомлением содержащем результат выполненных действий модулю – источнику запроса.

При конфигурировании пулов Bpool запрашивает данные о дисковых накопителях у модуля Disk Manager, который хранит данные об установленных в системе устройствах хранения и отслеживает их статус, с целью выяснить, имеются ли в данный момент доступные для создания пула дисковые накопители.

**Модуль Disk Manager** выполняет функции управления дисковыми накопителями и контроля их параметров используя встроенную в накопители систему самодиагностики - S.M.A.R.T. Модуль выполняет получение и отправку запросов от других модулей через интерфейс JRPC.

При физическом отключении диска, модуль Disk Manager отправляет извещение модулю Веб-сервер, который выводит предупреждение об аппаратной ошибке в интерфейсе пользователя. Также отправляются извещения другим модулям:

**Модуль SRS_service** управление системным raid массивом.

При запуске модуль запрашивает информацию о второй ноде у модуля Heartbeet. Далее сервис инициализирует таймер, который срабатывает раз в минуту и получает от ядра системы информацию о системном рейде, которой обменивается с соседней нодой. Модуль отправляет данные модулю Веб-сервер и получает от него команды управления системным raid массивом. Получение и отправка JSON пакетов выполняется через интерфейс JRPC.

**Модуль BNCD_service** отвечает за настройку сетевых (Ethernet) интерфейсов и их закрепление за службами (привязку). Получение и отправка JSON пакетов выполняется через интерфейс JRPC. При получении запросов от модуля Веб-сервер (команды пользовательского интерфейса) на настройку параметров сетевых интерфейсов, модуль отправляет команды управления ядру операционной системы. Получив подтверждение выполнения команды модуль формирует ответный пакет и отправляет его модулю Веб-сервер, который обновляет интерфейс пользователя.

Фиксация системных событий и событий, возникающих при действиях пользователя выполняется при помощи утилиты rsyslog.

**Модуль Log_service** выполняет регистрацию сервисов в rsyslog и управляет выборкой записей журнала событий системы для предоставления их пользователю. Журналы событий пишутся отдельно для каждого модуля и находятся в каталоге /var/log/. При возникновении сбоя ядра системы, модуль записывает сопутствующую информацию (dump), которая сохраняется в каталоге /var/log/crash/.

За мониторинг параметров системы отвечают два модуля:

**Модуль HWMON** – мониторинг аппаратной части комплекса;

**Модуль BestMon** – мониторинг производительности.

Для работы систем внешнего мониторинга используется протокол SNMP (Simple Network Management Protocol), а также служба SNMP, считывающая показания счетчиков производительности модуля BestMon.

**Модуль SNMP_service** управляет службой SNMP. Модуль предоставляет данные о статусе службы модулю Веб-сервера. Модуль обеспечивает следующий функционал:

- включение/ выключение системной службы SNMP;
- предоставление статуса службы (включена либо выключена).

**Модуль HWMON** выполняет сбор показаний датчиков аппаратного обеспечения и загрузки центрального процессора, а также получает данные о подключенных дисковых полках и установленных в системе адаптерах Fibre Channel и SAS. Приём и отправка JSON пакетов выполняется через интерфейс JRPC.

Модуль предоставляет данные о загрузка центрального процессора, список дисковых полок с их содержимым, показание встроенных датчиков аппаратного обеспечения как самого контроллера, так и дисковых полок.Данные считываются в режиме реального времени.

**Модуль BestMon** выполняет сбор статистики скорости чтения / записи и задержек при выполнении операций с дисками, томами, работе кэша. Приём и отправка JSON пакетов выполняется через интерфейс JRPC.

Модуль выполняет предоставление статистики операций чтения/записи при работе по сетевым протоколам, статистики операций чтения записи пулов и томов, статистики операций чтения/записи и задержек дисковых накопителей.

Для получения статистики модуль отправляет команды управления ядру операционной системы. Полученные данные передаются модулю Веб-интерфейса и отдаются по протоколу SNMP.

**Модуль HB_service** выполняет мониторинг состояния кластера. Модуль выполняет наблюдение за работоспособностью соседней ноды кластера и управление миграцией ресурсов. Наблюдение за статусом соседней ноды реализовано за счет регулярного обмена

информационными пакетами через сетевой интерконнект, адрес которого запрашивается у модуля BNCD_service. Этом алгоритм получил название – сетевой heartbeat.

Во избежание запуска механизма переподключения (миграции) ресурсов при пропадании связи через интерконнект, проверка статуса соседней ноды также дублируется через общую дисковую подсистему кластера, путем записи и чтения нодами данных в определенных секторах дисковых накопителей. Этот алгоритм называется «дисковый heartbeat».

При обнаружении отсутствия активности одного из узлов (нод) кластера, выполняется процедура миграции ресурсов, зарегистрированных на отключившейся ноде. При этом на работающую ноду автоматически переподключаются сетевые соединения и выполняется передача дисковых пулов под её управление. Список ресурсов, которые необходимо подключить, модуль Heartbeat запрашивает у модуля управления конфигурацией.

При выполнении миграции ресурсов с рабочей ноды кластера, например, при проведении планового обслуживания, запускается другой сценарий миграции. Сначала нода, с которой забирают ресурсы, выполняет отключение ресурсов от своих служб в определенной последовательности. Эти ресурсы переподключаются к одноименным службам второй ноды. После передачи своих ресурсов нода устанавливает статус «отдал ресурсы», вторая нода устанавливает статус «принял ресурсы». Статусы записываются в базу конфигурации через модуль Config Keeper.

**Модуль Access Control service** выполняет управление пользователями интерфейса управления системы и их авторизацию. Также модуль управляет списками доступа к ресурсам (клиентами и группами), и предоставляет данные для подключения к домену Active Directory и серверу LDAP. Приём и отправка JSON пакетов выполняется через интерфейс JRPC.

При запросе авторизации пользователя в интерфейсе управления, модуль Веб-сервер отправляет запрос модулю Access Control service, который в свою очередь отправляет запрос модулю Config Keeper. Получив данные пользователя модуль Access Control отправляет их модулю Веб-сервер.

При создании либо модификации пользователя системы и присвоении ему роли, модуль выполняет запросы к модулю Config Keeper.

**Модуль SMB_service** выполняет управление службами Samba и Windbind. Модуль читает и записывает конфигурацию связанных с протоколом SMB файловых ресурсов,

отправляя пакеты модулю Config Keeper. От модуля Веб-сервер модуль получает команды на подключение к домену, привязку протокола SMB к файловым системам, управление правами доступа к данным. От модуля Heartbeet модуль получает команды на подъём или опускание SMB ресурсов. Приём и отправка JSON пакетов выполняется через интерфейс JRPC.

**Модуль NFS_service** выполняет управление службами NFS и LDAP. Модуль читает и записывает конфигурацию связанных с протоколом NFS файловых ресурсов, отправляя пакеты модулю Config Keeper. От модуля Веб-сервер модуль получает команды на привязку протокола NFS к файловым системам, управление правами доступа к данным. От модуля Heartbeet модуль получает команды на подъём или опускание NFS ресурсов. Приём и отправка JSON пакетов выполняется через интерфейс JRPC.

**Модуль SR_service** – модуль асинхронной репликации, выполняет перемещение или репликацию данных томов и файловых систем как в ручном режиме, так и по расписанию.

Для реализации асинхронной репликации нужны передающая и приёмная стороны. Передающая сторона создаёт снимок данных - снапшот, и передает его на приёмную сторону. После приёма на обеих сторонах будет одинаковый набор снапшотов. Модуль Веб-сервер отправляет модулю SR_service параметры для создания задачи приёма реплики и параметры для задачи передачи данных и, если это необходимо, время начала выполнения репликации. После начала выполнения задачи репликации, модуль SR_service регулярно отправляет модулю Веб-сервер уведомления, содержащие количество переданных данных. Настройки созданных задач репликации сохраняются модулем Config Keeper в базе конфигурации. Приём и отправка JSON пакетов выполняется через интерфейс JRPC.

**Модуль UPD_service** – модуль обновления программного обеспечения системы. Модуль выполняет развертывание заданного образа системы с помощью стандартных утилит Linux в автоматическом режиме. Путь к файлу, содержащему обновление ПО UPD_service получает от модуля Веб-сервер, а после выполнения процедуры обновления, модулю Веб-сервер отправляется уведомление о результате операции. Приём и отправка JSON пакетов выполняется через интерфейс JRPC.

**Модуль Time_service** – модуль выполняет синхронизацию времени на узлах кластера.

### 2.1 Список программных модулей ПО BAUM STORAGE AI:

1. /usr/bin/uds_ac_service
2. /usr/bin/uds_afp_service
3. /usr/bin/uds_asr_service
4. /usr/bin/uds_bestmon_service
5. /usr/bin/uds_bncd_service
6. /usr/bin/uds_hwmon_service
7. /usr/bin/uds_ck_service
8. /usr/bin/uds_hb_service
9. /usr/bin/uds_log_service
10. /usr/bin/uds_nfs_service
11. /usr/bin/uds_rpc_router
12. /usr/bin/uds_smb_service
13. /usr/bin/uds_snmp_ext
14. /usr/bin/uds_snmp_service
15. /usr/bin/uds_srs_service
16. /usr/bin/uds_time_service
17. /usr/bin/uds_upd_service
18. /usr/bin/uds_vm_service
19. /usr/bin/uds_health_service

### 2.2 Список поддерживаемого функционала

| № | Поддерживаемый функционал | |
|---|---|---|
| 1 | Поддерживаемый объем хранимых данных, Пбайт | не более 2 |
| 2 | Поддержка типов RAID | 1,10,5,50,6,60 |
| 3 | Кэш записи | есть |
| 4 | Кэш чтения | есть |
| 5 | Моментальные снимки (snapshot) и клоны | есть |
| 6 | Компрессия | есть |
| 7 | Дедупликация | есть |
| 8 | Протокол NFS | есть |
| 9 | Протокол SMB (CIFS) | есть |
| 10 | Квоты | есть |

| № | Поддерживаемый функционал | |
|---|---|---|
| 11 | Мониторинг производительности | есть |
| 12 | WORM | только на SMB |
| 13 | Шифрование | по требованию |
| 14 | Работа в кластере | да |
| 15 | Количество узлов в кластере | 2 |
| 16 | Агрегация LACP | есть |
| 17 | Vlan | есть |
| 18 | Виртуальные интерфейсы | есть |
| 19 | Интеграция с Active Directory | есть |
| 20 | Интеграция с LDAP | есть |
| 21 | Поддержка горячего резервирования дисков (Hot spare) | для каждого пула |