Утвержден

РДЦП.10001-02-УД

Подп. и дата	
Инв. № дубл	
Взам. инв. №	
Подп. и дата	
Инв. № подл	

ПРОГРАММНЫЙ КОМПЛЕКС «СРЕДСТВА ВИРТУАЛИЗАЦИИ «БРЕСТ»

Руководство администратора

РДЦП.10001-02 95 01

Листов 140

2021

Литера $О_1$

АННОТАЦИЯ

Настоящий документ является руководством администратора программного комплекса «Средства виртуализации «Брест» (ПК СВ «Брест») РДЦП.10001-02 (далее — ПК СВ).

В документе приведено описание установки ПК СВ, настройки и порядка применения на персональном компьютере с учетом особенностей операционной системы специального назначения «Astra LInux Special Edition» РУСБ.10015-01 (далее по тексту — ОС СН), под управлением которой функционирует ПК СВ. В документе описан порядок использования среды виртуализации, обеспечения отказоустойчивости, масштабирования и виртуализации дисковых пространств, способы развертывания и организации облачных решений с использованием платформы OpenNebula.

Дополнительная информация о порядке эксплуатации, а также варианты реализации отдельных решений приведены на официальном сайте `wiki.astralinux.ru/docs`.

СОДЕРЖАНИЕ

1. Базовые средства организации защищенной среды виртуализации 10
1.1. Установка основных компонентов 10
1.2. Графическая утилита virt-manager 10
2. Управление виртуальными машинами 12
2.1. Создание виртуальной машины 12
2.2. Доступ к рабочему столу виртуальной машины по протоколам VNC и SPICE . . . 18
2.2.1. Особенности использования VNC 19
2.2.2. Особенности использования протокола SPICE 20
2.2.2.1. Изменение разрешения экрана при подключении по протоколу SPICE 21
2.2.2.2. Подключение более одного монитора по протоколу SPICE 21
2.3. Управление конфигурацией виртуальных машин с помощью графической утилиты и инструмента командной строки 21
2.4. Инструменты командной строки 23
2.5. Управление файлами образов дисков виртуальных машин 24
2.6. Управление виртуальными сетевыми интерфейсами и сетями 25
3. Подключение физического устройства к ВМ 29
3.1. Подключение USB-устройства к ВМ 29
3.2. Подключение жесткого диска к ВМ 31
3.3. Подключение PCI-устройства к ВМ 32
3.4. Подключение последовательного порта к ВМ 36
4. Взаимодействие администратора с СЗИ 38
4.1. Настройка дискреционного и мандатного управление доступом к ВМ 38
4.2. Ограничения по конфигурированию и запуску виртуальных машин в условиях мандатного управления доступом 39
5. Средства обеспечения отказоустойчивости 41
5.1. Программное обеспечение кластера высокой доступности. Pacemaker и Corosync 41
5.2. Отказоустойчивые сервисы Samba и NFS 41
6. Средства виртуализации дисковых пространств 48
6.1. Организация сетевого RAID 48
6.1.1. Установка и настройка DRBD 48
6.2. Распределенные параллельные файловые системы с защитой от сбоев 50

6.3. Средства интеграции и дифференциации блочных устройств 50
6.3.1. DRBD 50
6.3.2. ISCSI 50
6.3.2.1. Установка цели 51
6.3.2.2. Пример реализации цели 52
6.3.2.3. Установка инициатора 54
6.3.2.4. Пример реализации инициатора 54
7. Программный коммутатор Open vSwitch 55
7.1. Установка 55
7.2. Особенности конфигурирования 55
8. Облачная системная архитектура OpenNebula 57
8.1. Обзор архитектуры и определение основных характеристик облака 57
8.2. Фронтальная машина 58
8.2.1. Характеристики фронтальной машины 58
8.2.2. Предварительная настройка ОС СН 59
8.2.3. Конфигурирование фронтальной машины 59
8.2.4. Алгоритм Raft 60
8.3. Конфигурация сервиса oned 60
8.3.1. Параметры настройки сервиса 60
8.3.2. Виртуальные сети 63
8.3.3. Хранилища 63
8.3.4. Сборщик информации 65
8.3.5. Информационный драйвер 65
8.3.6. Система хуков 66
8.3.6.1. Хуки виртуальной машины VM_HOOK 67
8.3.6.2. Хуки узла HOST_HOOK 67
8.4. Мониторинг 68
8.5. Узлы виртуализации 68
8.5.1. Конфигурирование узла KVM 69
8.5.1.1. Предварительная настройка ОС СН 69
8.5.1.2. Установка программного обеспечения 70
8.5.1.3. Конфигурация ssh без пароля 70
8.5.1.4. Конфигурация сети 70

8.5.1.5. Конфигурация хранилища 71
8.5.1.6. Добавление узла 71
8.5.1.7. Импорт виртуальных машин 74
8.6. Хранилище данных 74
8.7. Организация сети 75
8.8. Аутентификация 75
8.9. Отказоустойчивость виртуальной машины 76
8.9.1. Автостарт виртуальных машин 77
8.10. Дополнительные компоненты 78
8.11. Управление пользователями 78
9. Особенности настройки узла 80
9.1. Требования 80
9.2. Настройка 80
9.2.1. Настройка KVM 80
9.2.2. Драйверы 80
9.2.3. Динамическое перемещение для других параметров кэширования 80
9.3. Использование 80
9.3.1. Специальные опции KVM 80
9.3.2. Графическая подсистема 81
9.3.3. Virtio 81
9.3.4. Горячее подключение/отключение Disk/NIC 81
9.3.5. Подключение QEMU Guest Agent 82
9.3.6. Импорт виртуальной машины 82
9.3.7. Сервисный режим узла 82
9.4. Управление хостами по IPMI 82
9.5. Перенаправление PCI-устройств 83
9.5.1. Требования 83
9.5.2. Настройка гипервизора 83
9.5.2.1. Конфигурация ядра 83
9.5.2.2. Загрузка драйвера vfio в initrd 83
9.5.2.3. Блокировка драйверов 83
9.5.2.4. Привязка устройств к vfio 84
9.5.2.5. Конфигурация QEMU 84

9.5.3. Настройка драйвера 85
9.5.4. Настройка использования устройств PCI 85
9.5.4.1. Настройка в CLI 86
9.5.4.2. Настройка в Sunstone 87
9.5.5. Использование устройств PCI в качестве сетевых интерфейсов 87
9.5.5.1. Настройка через CLI 88
9.5.5.2. Настройка через Sunstone 88
10. Установка открытого облачного хранилища 90
10.1. Хранилище образов 90
10.1.1. Хранилище данных файловой системы 90
10.1.1.1. Схема хранилища данных 91
10.1.1.2. Установка фронтальной машины 92
10.1.1.3. Установка узла 93
10.1.1.4. Настройка OpenNebula 93
10.1.1.5. Дополнительные параметры 94
10.1.2. Хранилище данных LVM 94
10.1.2.1. Схема хранилища данных 95
10.1.2.2. Установка фронтальной машины 96
10.1.2.3. Установка узла 96
10.1.2.4. Настройка OpenNebula 96
10.1.3. Хранилище данных Ceph 97
10.1.3.1. Схема хранилища данных 98
10.1.3.2. Установка Ceph-кластера 98
10.1.3.3. Установка фронтальной машины 98
10.1.3.4. Установка узла 98
10.1.3.5. Настройка OpenNebula 100
10.1.3.6. Дополнительные параметры 101
10.1.4. Хранилище Raw Device Mapping 101
10.1.4.1. Схема хранилища 101
10.1.4.2. Установка фронтальной машины 102
10.1.4.3. Установка узла 102
10.1.4.4. Настройка OpenNebula 102
10.1.4.5. Использование хранилища 103

10.1.5. Хранилище iSCSI-Libvirt . . . 103
10.1.5.1. Установка фронтальной машины . . . 103
10.1.5.2. Установка узла . . . 103
10.1.5.3. Аутентификация iSCSI CHAP . . . 103
10.1.5.4. Настройка OpenNebula . . . 104
10.1.5.5. Использование хранилища . . . 105
10.2. Хранилище файлов . . . 106
10.2.1. Требования . . . 106
10.2.2. Настройка . . . 106
10.2.3. Настройка узла . . . 107
11. Установка открытой облачной сети . . . 108
11.1. Установка узла . . . 108
11.1.1. Сетевой режим Bridged . . . 108
11.1.1.1. Требования . . . 108
11.1.1.2. Настройка . . . 108
11.1.2. Сетевой режим VLAN . . . 109
11.1.2.1. Требования . . . 109
11.1.2.2. Настройка . . . 109
11.1.3. Сетевой режим VXLAN . . . 109
11.1.3.1. Требования . . . 109
11.1.3.2. Настройка . . . 109
11.1.4. Сетевой режим Open vSwitch . . . 109
11.1.4.1. Требования . . . 109
11.1.4.2. Настройка . . . 110
11.2. Сетевой мост . . . 110
11.2.1. Особенности и ограничения . . . 110
11.2.2. Настройка OpenNebula . . . 110
11.2.3. Определение мостовой сети . . . 110
11.2.4. Режим ebtables VLAN . . . 111
11.3. Сети 802.1Q VLAN . . . 111
11.3.1. Настройка OpenNebula . . . 111
11.3.2. Определение сети 802.1Q . . . 112
11.4. Сети VXLAN . . . 112

11.4.1. Особенности и ограничения . . . 113
11.4.2. Настройка OpenNebula . . . 113
11.4.3. Определение сети VXLAN . . . 113
11.5. Сети Open vSwitch . . . 114
11.5.1. Особенности конфигурирования . . . 115
11.5.2. Агрегирование физических интерфейсов . . . 116
11.5.3. Зеркалирование портов . . . 116
11.5.4. Настройка OpenNebula . . . 117
11.5.5. Определение сети Open vSwitch . . . 118
11.5.6. Многоканальные сети VLAN (VLAN транкинг) . . . 118
11.5.7. Правила OpenFlow . . . 118
11.5.7.1. MAC-спуфинг . . . 118
11.5.7.2. IP-захват . . . 119
11.5.7.3. Черные порты . . . 119
11.5.7.4. ICMP-игнорирование . . . 119
12. Мониторинг и учет . . . 120
12.1. Мониторинг . . . 120
12.1.1. Конфигурация OpenNebula . . . 120
12.1.1.1. Подключение драйвера мониторинга . . . 120
12.1.1.2. Параметры настройки мониторинга . . . 121
12.1.2. Отчет системы мониторинга . . . 121
12.1.3. Настройка и расширение . . . 122
12.1.3.1. Корректировка интервалов мониторинга . . . 122
12.1.3.2. Тесты . . . 122
12.2. Учет использования ВМ . . . 122
12.2.1. Получение информации о потреблении ресурсов с помощью CLI . . . 122
12.2.2. Получение информации о потреблении ресурсов с помощью Sunstone . . . 131
12.2.3. Тонкая настройка и расширение . . . 132
12.3. Логирование . . . 133
12.3.1. Настройка системы регистрации . . . 133
12.3.2. Регистрационные ресурсы . . . 133
12.3.3. Регистрационный формат . . . 133
12.3.4. Ошибки виртуальной машины . . . 134

12.3.5. Ошибки узла 135

13. Ограничения при использовании 136

Перечень сокращений 137

1. БАЗОВЫЕ СРЕДСТВА ОРГАНИЗАЦИИ ЗАЩИЩЕННОЙ СРЕДЫ ВИРТУАЛИЗАЦИИ

Основными пакетами, необходимыми для виртуализации персонального компьютера, являются пакеты средства эмуляции аппаратного обеспечения на основе QEMU и сервера виртуализации libvirt из состава ОС СН, а также пакет графического интерфейса управления виртуализацией virt-manager.

Описание средств эмуляции аппаратного обеспечения на основе QEMU, сервера виртуализации libvirt, а также порядок настройки доступа к серверу виртуализации libvirt приведены в документе РУСБ.10015-01 95 01-1 «Операционная система специального назначения «Astra Linux Special Edition». Руководство администратора. Часть 1» из состава ОС СН.

1.1. Установка основных компонентов

Установка и настройка компонентов выполняется от имени администратора используя механизм `sudo`.

Для установки основных компонентов необходимо выполнить следующие действия, при этом репозиторий ОС СН должен быть уже подключен:

1) подключить репозиторий ПК СВ:
2) обновить список доступных пакетов, выполнив команду:

```
sudo apt update
```

3) установить клиентскую часть ПК СВ, выполнив команду:

```
sudo apt install virt-manager
```

Примечания:

1. Для функционирования системы в режиме замкнутой программной среды (ЗПС) необходимо дополнительная установка пакета `brest-digsig-key`.
2. Для возможности полного управления ВМ администратор должен входить в группы `disk`, `kvm`, `libvirt`, `libvirt-admin`, `libvirt-qemu`, `astra-console` и `astra-admin`.
3. Для работы с ПК СВ обычному пользователю необходимо вхождение пользователя в локальные группы `kvm` и `libvirt-qemu`.

ВНИМАНИЕ! Операции по изменению состава или конфигурации виртуальных машин требуют вхождения пользователя в специальную локальную административную группу `libvirt-admin`.

1.2. Графическая утилита virt-manager

В состав ПК СВ входит графическая утилита управления виртуальными машинами virt-manager, предоставляющая доступ к возможностям сервера виртуализации libvirt из

графического интерфейса пользователя. Внешний вид окна утилиты приведен на рис. 1.

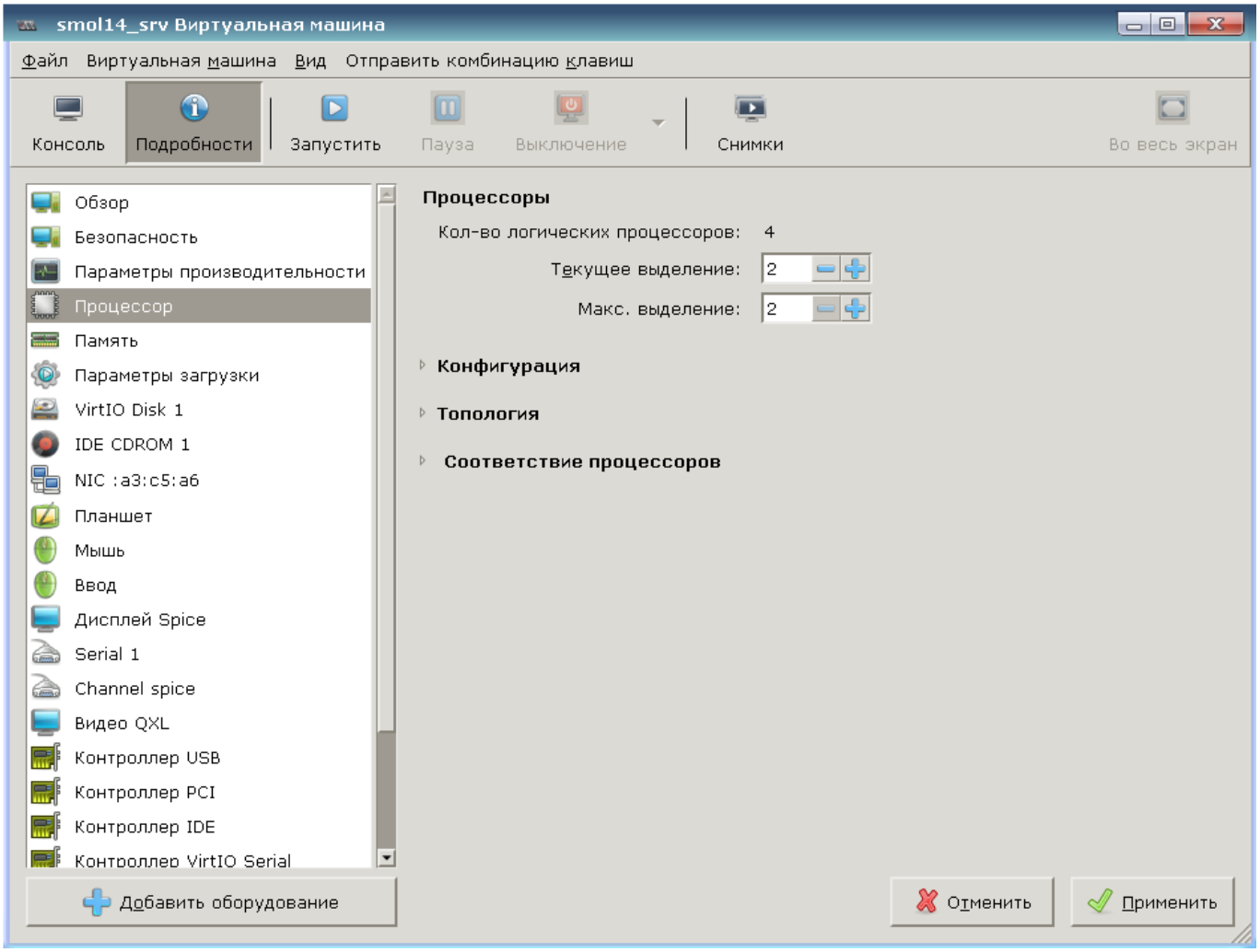

Рис. 1

Утилита позволяет выполнять действия по созданию виртуальных машин, управлению их конфигурацией и файлами-образов дисковых носителей. Также обеспечивает удаленный доступ к рабочему столу выбранной виртуальной машины по протоколам VNC и SPICE. Более подробная информация по работе с программой приведена в разделе 2.

2. УПРАВЛЕНИЕ ВИРТУАЛЬНЫМИ МАШИНАМИ

Управление виртуальными машинами в ПК СВ осуществляется с помощью инструмента командной строки `virsh` или графической утилиты virt-manager путем обращения к соответствующему серверу виртуализации libvirt.

ВНИМАНИЕ! Операции по изменению состава или конфигурации виртуальных машин требуют вхождения пользователя в специальную локальную административную группу `libvirt-admin`.

ВНИМАНИЕ! Существуют ограничения по конфигурированию и запуску виртуальных машин в условиях мандатного управления доступом (см. 4.2).

2.1. Создание виртуальной машины

Для создания ВМ с помощью графической утилиты необходимо войти в ОС СН под учетной записью пользователя, включенного в соответствующие группы согласно примечаний на стр. 10, и запустить утилиту virt-manager одним из приведённых способов:

1) открыть меню «Пуск», перейти во вкладку «Системные» и выбрать «Менеджер виртуальных машин»;
2) нажать комбинацию клавиш **<Alt+F2>** для запуска утилиты «Выполнить команду», в строке ввести `virt-manager`;
3) нажать комбинацию клавиш **<Alt+T>** для запуска утилиты «Терминал Fly», в строке ввести `virt-manager`.

Примечание. Если пользователь не включен в соответствующие группы согласно примечаний на стр. 10, при запуске утилиты потребуется пароль администратора системы.

По умолчанию устанавливается подключение к локальному серверу виртуализации. Для доступа к удаленному серверу виртуализации требуется задать параметры подключения в соответствии с рис. 2.

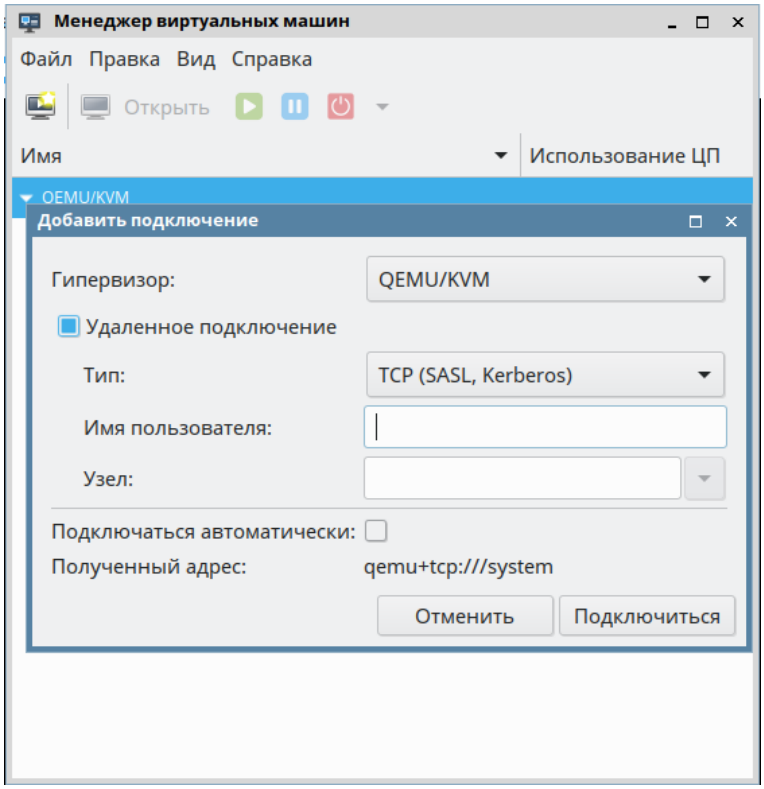

Рис. 2

После установки соединения с сервером виртуализации можно приступать к созданию виртуальных машин. Для этого необходимо:

1) нажать на кнопку **[Создать виртуальную машину]** или в панели меню выбрать «Файл — Создать виртуальную машину»;

2) в открывшемся окне, в соответствии с рис. 3, выбрать метод установки операционной системы из предлагаемого списка, например: «Локальный ISO или CD-ROM»;

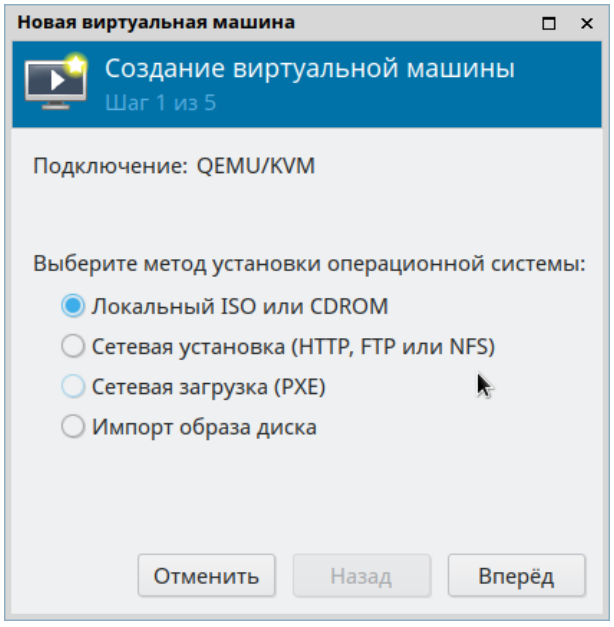

Рис. 3

3) для установки гостевой ОС указать расположение установочного носителя, ко-

торым может являться ISO-образ установочного диска ОС СН или DVD-диск с дистрибутивом ОС СН в соответствии с рис. 4. ISO-образ предварительно должен быть добавлен в соответствующий пул хранения согласно 2.5, DVD-диск должен быть установлен в устройство чтения CD/DVD-дисков;

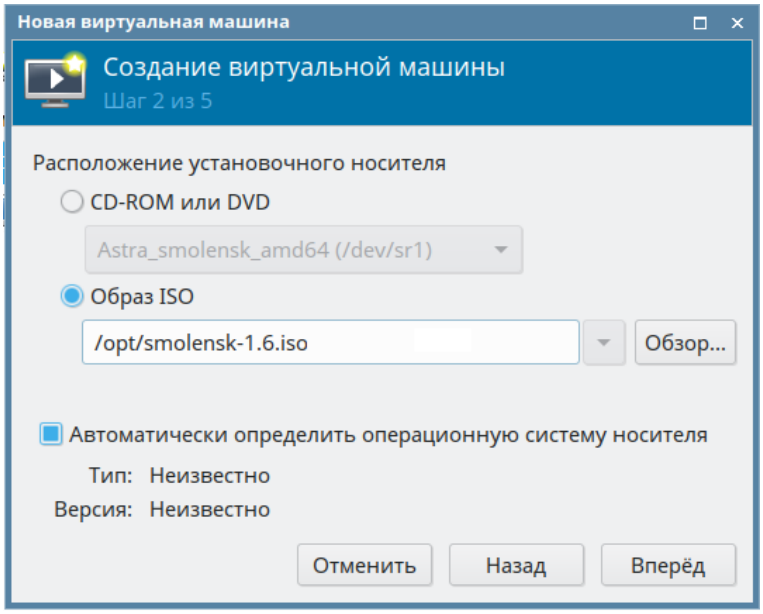

Рис. 4

4) в соответствии с рис. 5 задать размер памяти и количество процессоров для ВМ, но не более указанных доступных значений;

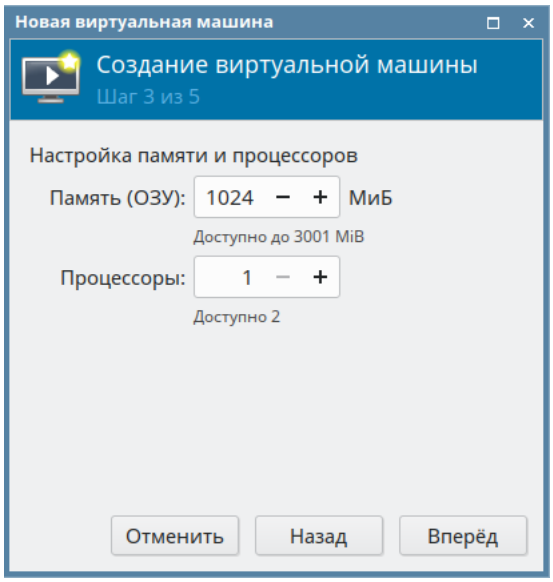

Рис. 5

5) в соответствии с рис. 6 указать параметры для создания файла образа основного дискового носителя, задав его размер не менее 8 ГБ;

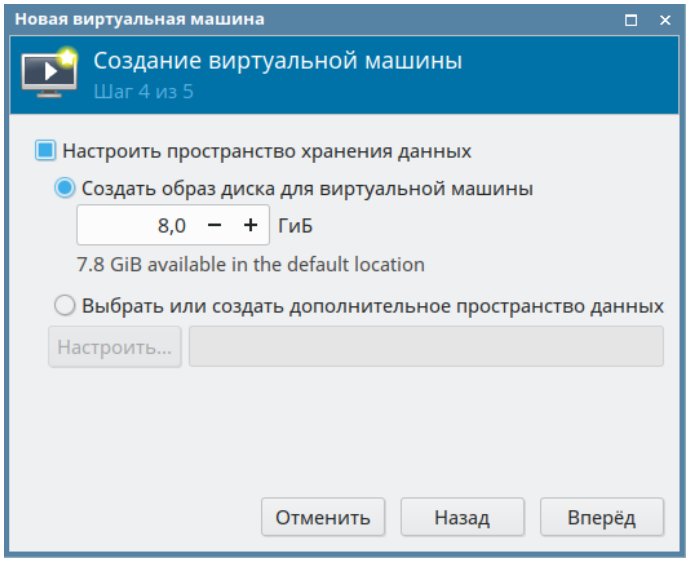

Рис. 6

6) для завершения конфигурирования ВМ в окне, приведенном на рис. 7, необходимо установить флаг в поле «Проверить конфигурацию перед установкой» и нажать кнопку **[Готово]**.

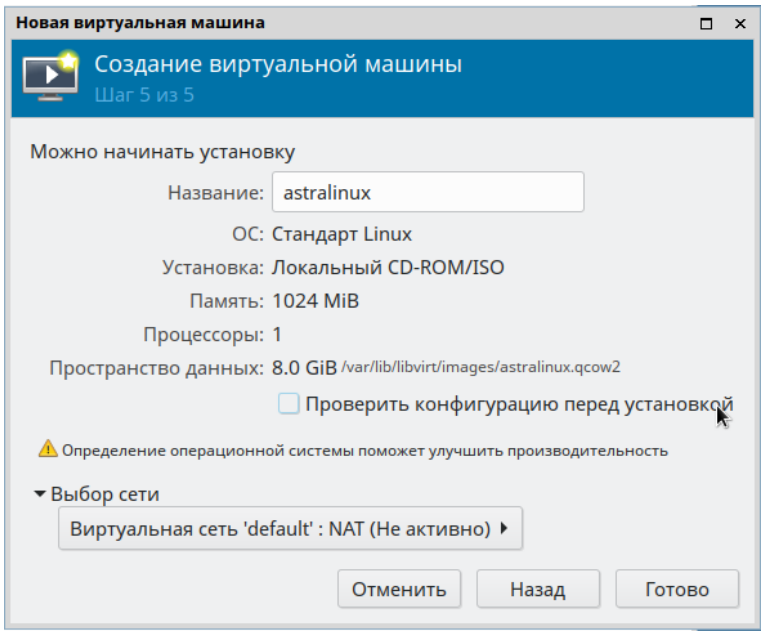

Рис. 7

В появившемся окне проверить параметры создаваемой ВМ.

Примечание. Для улучшения работы ВМ рекомендуется дополнительно выполнить следующие настройки:

а) в пункте меню «Процессоры» установить флаг «Копировать конфигурацию ЦП хоста» согласно рис. 8;

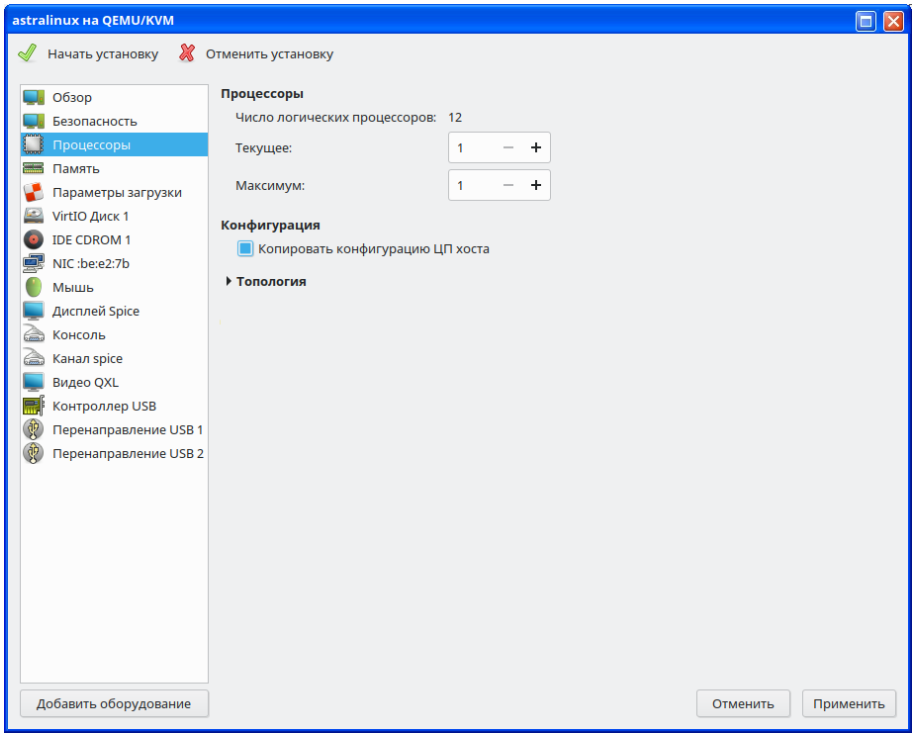

Рис. 8

б) в пункте меню «VirtIO Диск 1» раскрыть секцию «Дополнительные параметры» и в поле «Шина диска» выбрать из раскрывающегося списка значение «VirtIO» согласно рис. 9;

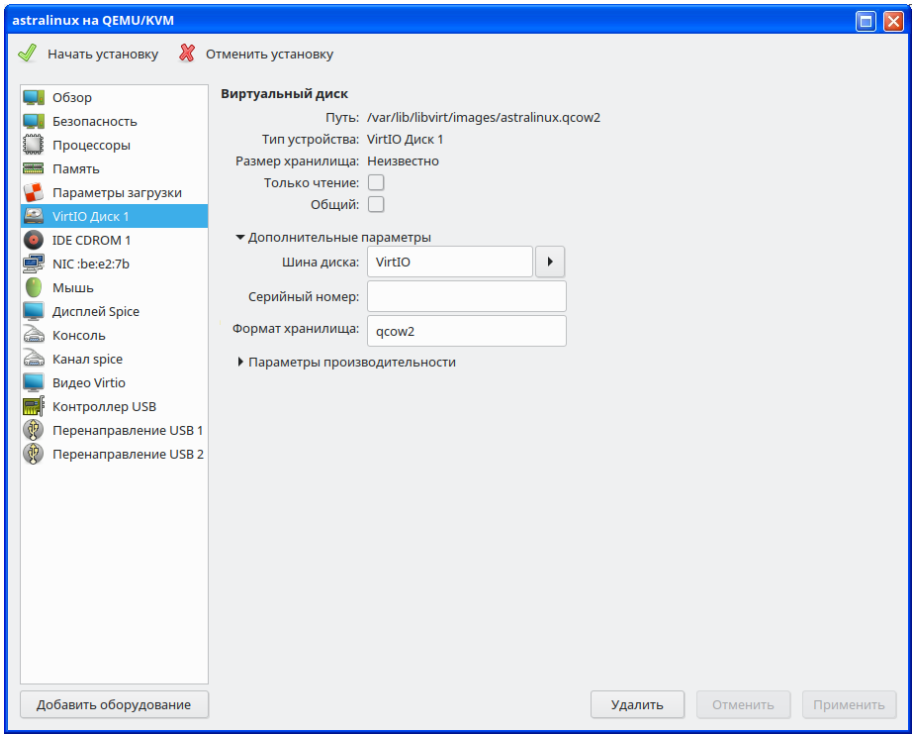

Рис. 9

в) в пункте меню «Видео Virtio» в поле «Модель» выбрать из выпадающего списка значение «Virtio»;

7) для начала установки ОС виртуальной машины нажать в левом верхнем углу кнопку **[Начать установку]**.

ВНИМАНИЕ! В случае отмены установки, настройки ВМ не сохранятся.

После завершения настройки запустится созданная ВМ с окном установки ОС СН как показано на рис. 10.

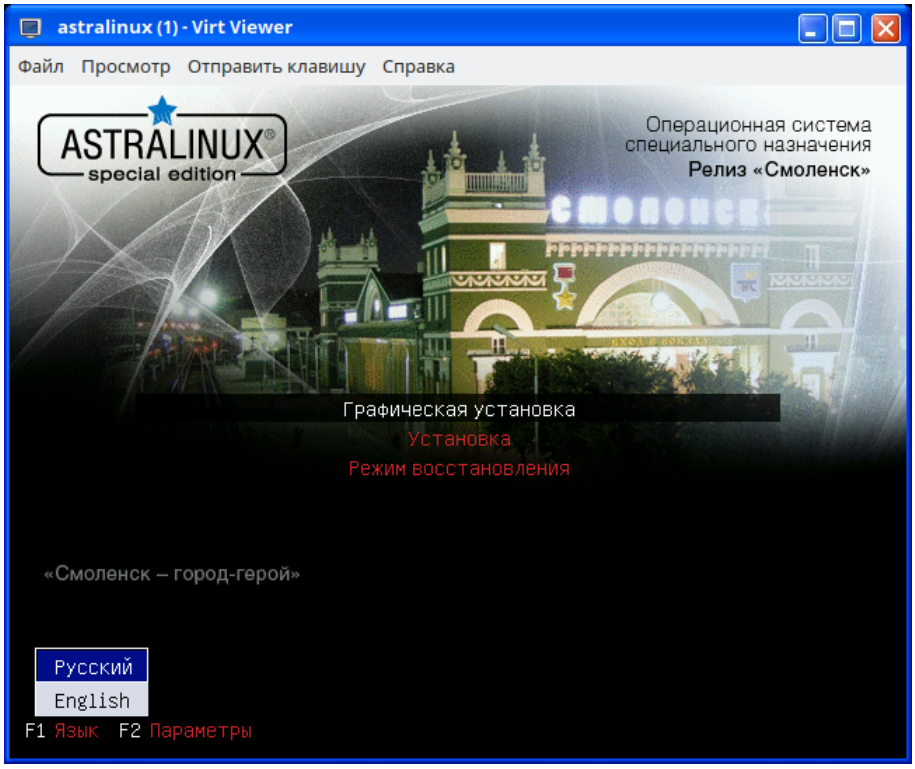

Рис. 10

Для создания виртуальных машин с помощью консольной утилиты необходимо войти в ОС СН под учетной записью пользователя, включенного в группы `astra-admin` и `astra-console`, запустить Терминал Fly (нажав комбинацию клавиш **<Alt+T>** либо нажав комбинацию клавиш **<Alt+F2>** для запуска утилиты «Выполнить команду» и в строке ввести `fly-term`) и выполнить команду:

```
virt-install --connect qemu:///system -n smolensk16 -r 2048 --vcpus=2
-c /path/to/iso/smolensk.iso --disk
pool=default,size=8,bus=virtio,format=qcow2 --os-type=linux
```

где `-n smolensk16` — название ВМ;

`-r 2048` — объем оперативной памяти;

`--vcpus=2` — количество процессоров;

`-c /path/to/iso/smolensk.iso` — путь к ISO-образу установочного диска ОС СН;

`pool=default` — хранилище по умолчанию;

`size=8` — размер хранилища в ГБ;

`bus=virtio` — шина создаваемого хранилища (диска ВМ);

`format=qcow2` — размер хранилища в ГБ;

`--os-type=linux` — тип операционной системы.

В результате запустится созданная виртуальная машина с окном установки ОС СН (рис. 10).

2.2. Доступ к рабочему столу виртуальной машины по протоколам VNC и SPICE

Рабочим столом ВМ можно управлять из встроенного в `virt-manager` «просмотрщика» или с помощью входящих в состав ПК СВ отдельных приложений `remote-viewer` и `virt-viewer`, позволяющих получить доступ (VDI) по протоколам VNC и SPICE. Приложения имеют схожий функционал, но различаются синтаксисом вызова. На рис. 11 представлены строка вызова и сам интерфейс доступа к ВМ с помощью `virt-viewer`. На момент подключения к рабочему столу виртуальная машина должна быть уже запущена.

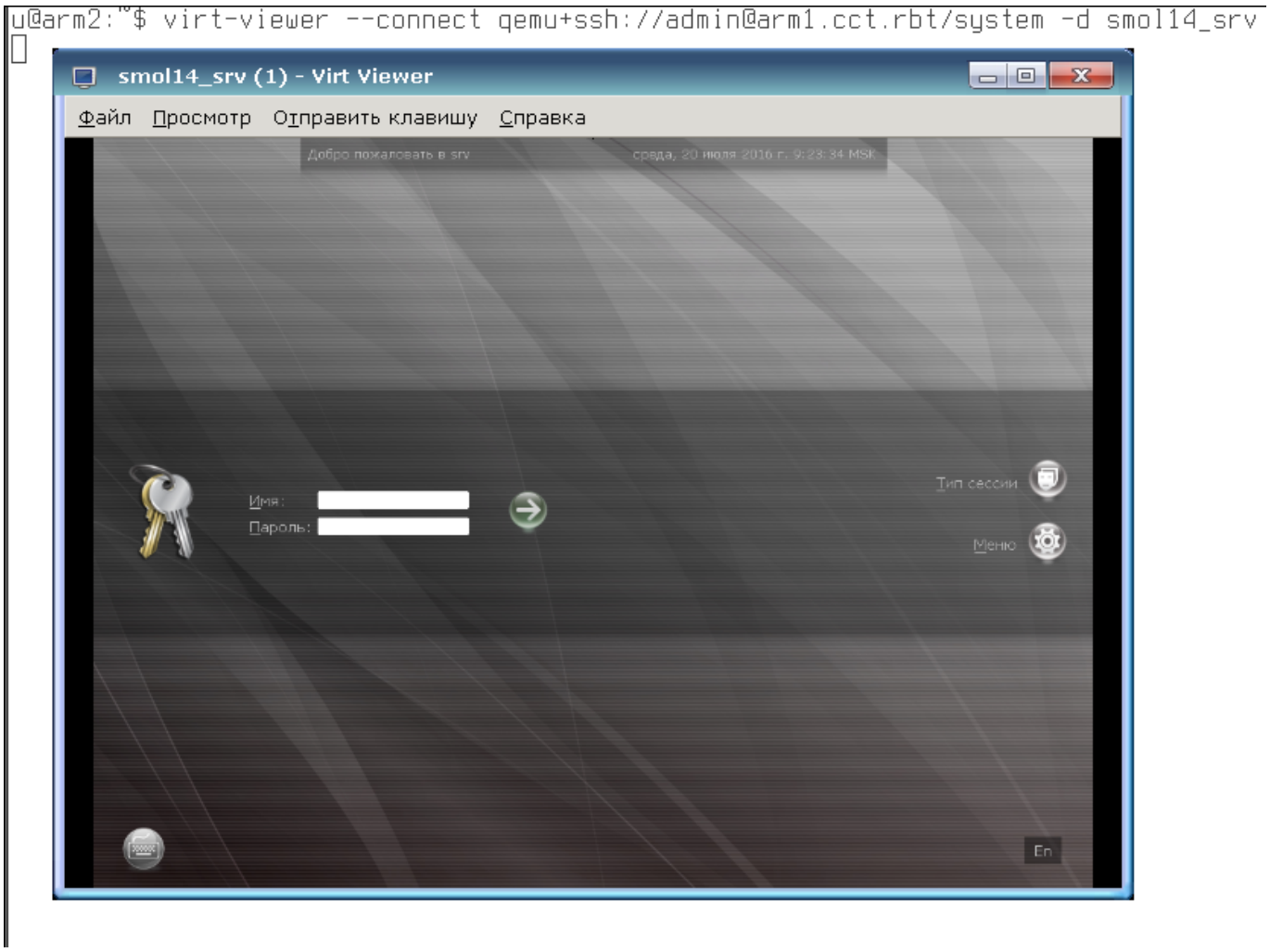

Рис. 11

2.2.1. Особенности использования VNC

При использовании VNC-сервера в качестве графической подсистемы ВМ у администратора появляется возможность указать режим доступа к дисплею.

Доступны следующие режимы доступа к дисплею:

- «Общий» — доступ к VNC-дисплею предоставляется без ограничений количества подключений;
- «Монопольный» — VNC-подключение доступно только первому подключившемуся;
- «Эксклюзивный» — первый клиент получает доступ к VNC-дисплею, все остальные подключения отбрасываются. Если первый клиент укажет опцию «shared», другие клиенты тоже смогут подключится, так же указав эту опцию;
- «Запретить эксклюзивный» — доступ к VNC-дисплею разрешается только с указанием опции «shared».

Режим доступа к дисплею устанавливается путем выбора из выпадающего списка соответствующего значения как показано на рис. 12. Порядок перехода к параметрам ВМ и управление ими описаны в 2.3.

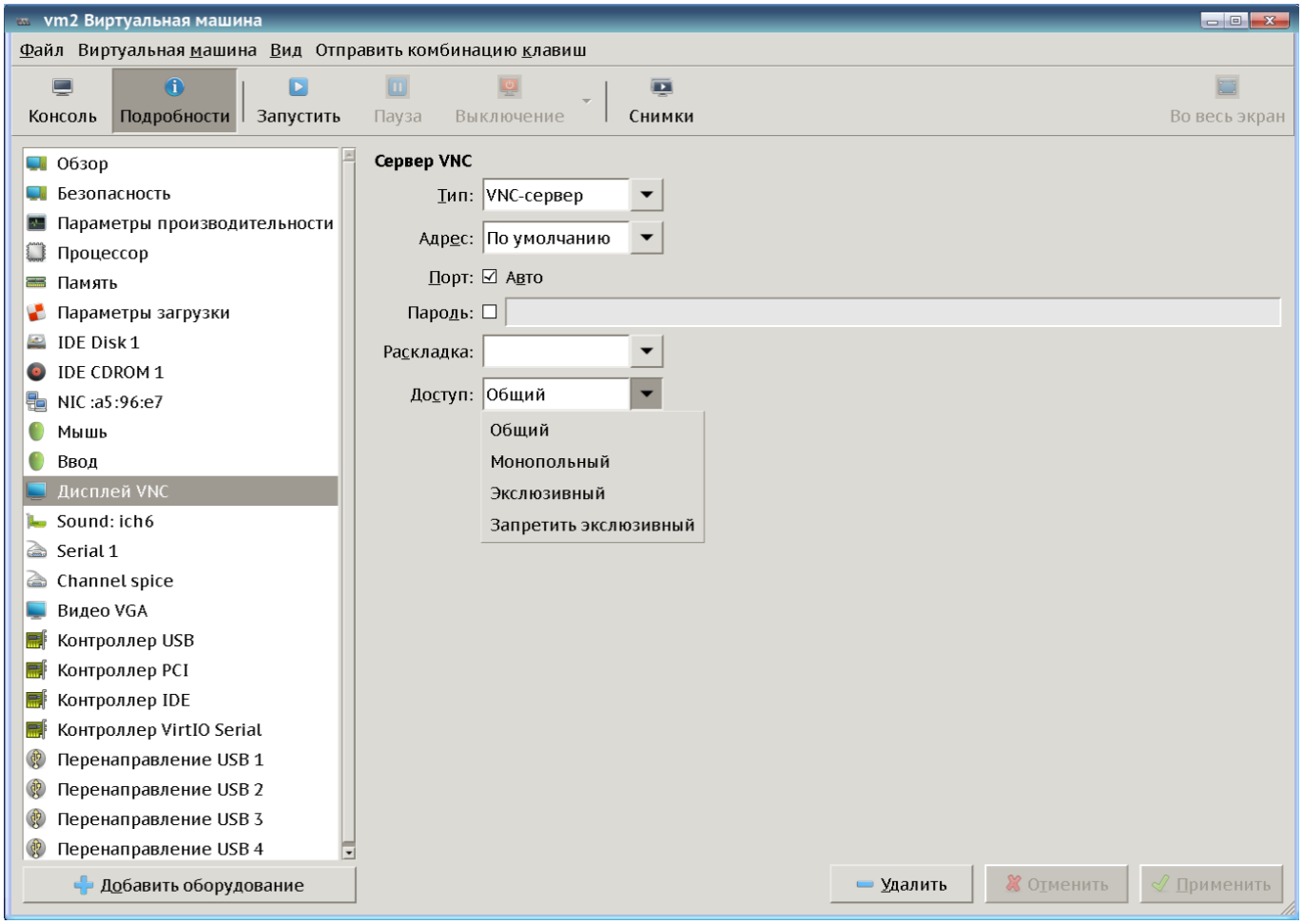

Рис. 12

Пример

Подключение к VNC-серверу (с режимом «Запретить эксклюзивный») с помощью клиента vncviewer

```
vncviewer -shared astra-server:5900
```

2.2.2. Особенности использования протокола SPICE

Для полноценного использования протокола SPICE в гостевой машине с ОС СН необходимо наличие установленных пакетов QXL-драйвера `xserver-xorg-video-qxl` и службы-клиента `spice-vdagent`. Установка осуществляется путем выполнения команды:

```
apt install xserver-xorg-video-qxl spice-vdagent
```

После установки указанных пакетов и перезагрузки ВМ у пользователя появится возможность изменять разрешение экрана, автоматически подключать звуковые устройства узла в ОС виртуальной машины, а также использовать более одного монитора.

В качестве службы-клиента SPICE может использоваться `remote-viewer`.

Пример

Вызов из командной строки подключения к SPICE-серверу `astra-server`

```
remote-viewer spice://astra-server:5900
```

2.2.2.1. Изменение разрешения экрана при подключении по протоколу SPICE

Для изменения разрешения экрана при подключении по протоколу SPICE необходимо на гостевой ОС в домашней директории пользователя в файл `.profile` добавить следующую строку вызова:

```
xrandr --output Virtual-0 --mode 1920x1080
```

где `1920x1080` — требуемое разрешение экрана.

Далее при входе указанного пользователя в сеанс разрешение экрана будет меняться на заданное значение.

2.2.2.2. Подключение более одного монитора по протоколу SPICE

При использовании протокола SPICE пользователь может подключить до четырех мониторов. Для этого необходимо:

1) в соответствующем клиенте (`remote-viewer` или `virt-viewer`) отметить требуемые мониторы в соответствии с рис. 13;

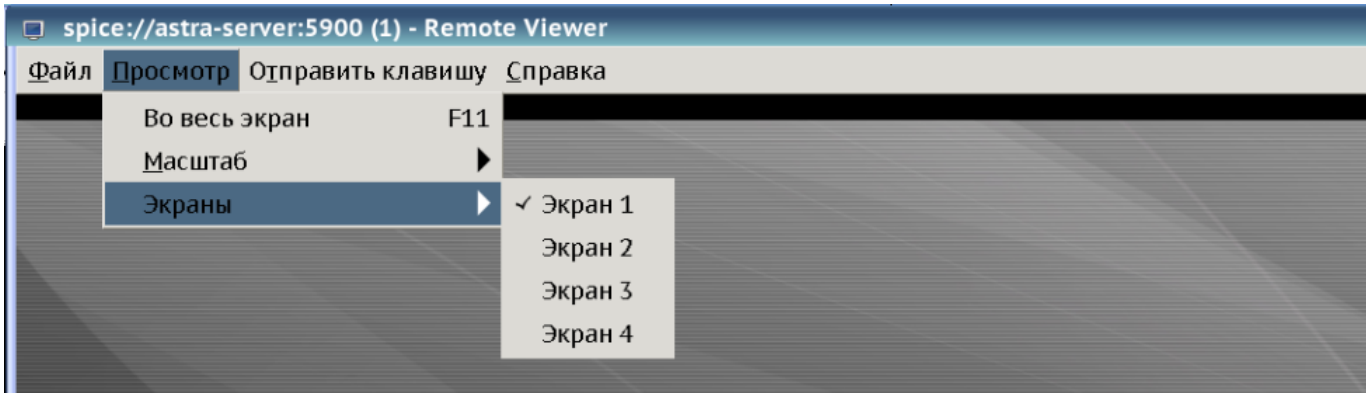

Рис. 13

2) в гостевой ОС с помощью стандартной утилиты `xrandr` указать использование дополнительных мониторов. Например, для подключения двух мониторов необходимо выполнить следующую команду:

```
xrandr --output Virtual-0 --mode 1920x1080 --pos 0x0 --output Virtual-1
    --mode 1280x1024 --pos 1920x0
```

где `1920x1080` — требуемое разрешение первого монитора;

`1280x1024` — требуемое разрешение второго монитора;

`1920x0` — смещение второго монитора относительно первого.

Если описанная конфигурация будет использоваться постоянно, необходимо указанную команду добавить в файл `.profile` в домашней директории пользователя.

2.3. Управление конфигурацией виртуальных машин с помощью графической утилиты и инструмента командной строки

Для управления конфигурацией виртуальных машин с помощью менеджера виртуальных машин `virt-manager` необходимо в основном окне `virt-manager` выбрать ВМ из

списка и нажать на кнопку **[Открыть]** в соответствии с рис. 14 для открытия окна управления данной ВМ. Затем нажать **[Подробности]** в соответствии с рис. 15 для перехода в раздел конфигурирования ВМ.

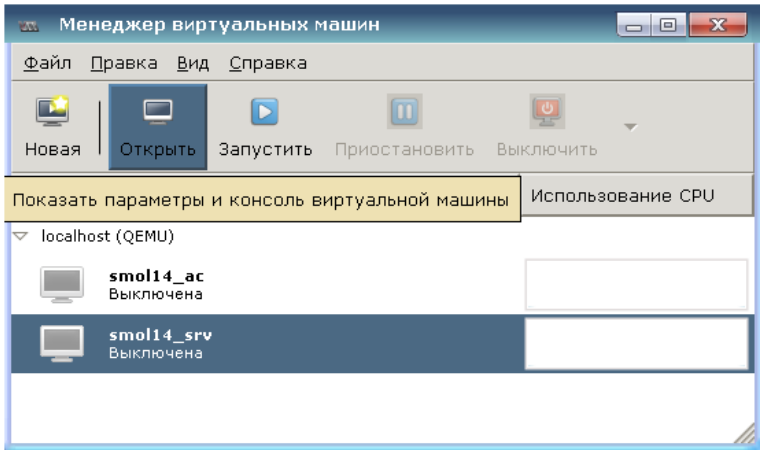

Рис. 14

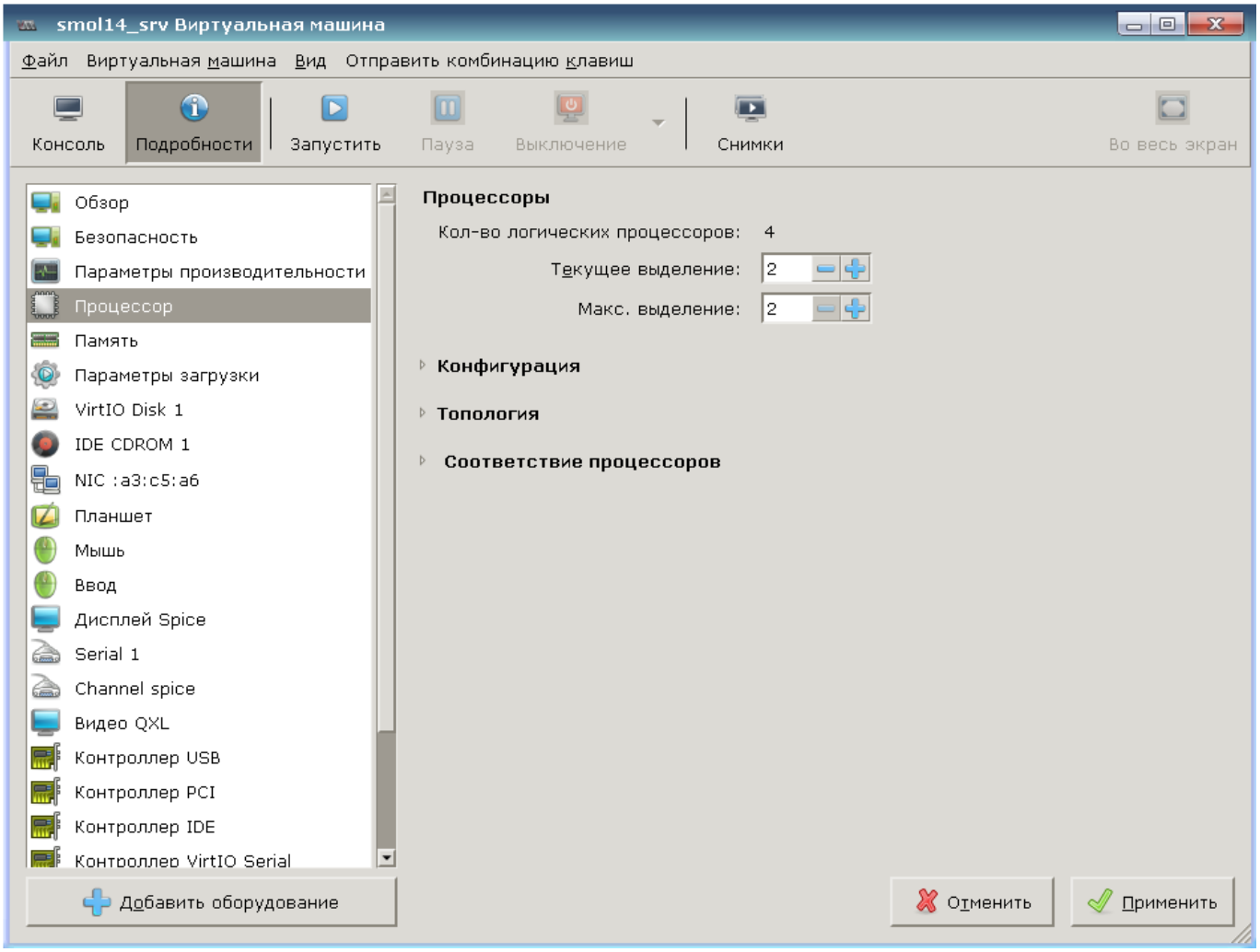

Рис. 15

Для изменения параметра необходимо перейти на соответствующую вкладку, внести изменения и подтвердить путем нажатия кнопки **[Применить]**.

Управление конфигурацией виртуальных машин из командной строки производится с помощью инструмента virsh. Например, для изменения размера оперативной памяти и количества виртуальных процессоров необходимо выполнить следующие команды соответственно:

```
virsh --connect qemu:///system setmaxmem --size 4194304 smolensk15_srv
virsh --connect qemu:///system setvcpus --config smolensk15_srv 3 -maximum
```

Более подробное описание команд инструмента virsh приведено в руководстве man.

2.4. Инструменты командной строки

В состав ПК СВ входят следующие инструменты командной строки:

- qemu-img — инструмент работы с образами дисков виртуальной машины QEMU. Позволяет выполнять операции по созданию образов различных форматов, конвертировать файл образа из одного формата в другой, получать информацию об образах и объединять снимки виртуальных машин для тех форматов, которые это поддерживают;
- qemu-io — инструмент диагностирования образов виртуальных носителей QEMU;
- qemu-nbd — инструмент экспортирования файлов образов виртуальных носителей QEMU с использованием протокола NBD;
- qemu-system-i386 — средство эмуляции 32-разрядной платформы x86;
- qemu-system-x86_64 — средство эмуляции 64-разрядной платформы x86;
- virsh — инструмент управления виртуальными машинами;
- virt-clone — инструмент клонирования виртуальных машин;
- virt-convert — инструмент конвертирования виртуальной машины из одного формата в другой и с одной программно-аппаратной платформы на другую;
- virt-host-validate — инструмент проверки конфигурации узла сервера виртуализации;
- virt-image — инструмент создания виртуальной машины по их XML-описанию;
- virt-install — инструмент создания виртуальной машины с помощью параметров командной строки;
- virt-pki-validate — инструмент проверки корректности настройки системы PKI (сертификатов) сервера виртуализации libvirt;
- virt-sanlock-cleanup — инструмент удаления устаревших файлов блокировки виртуальной машины;
- virt-xml — инструмент редактирования XML-описаний виртуальной машины с помощью параметров командной строки;
- virt-xml-validate — инструмент проверки корректности XML-описаний вирту-

альной машины.

Возможности и способы использования данных инструментов описаны в соответствующих разделах руководства `man`.

2.5. Управление файлами образов дисков виртуальных машин

В ПК СВ работа с файлами образов носителей производится с помощью пулов носителей. Пул носителей определяет физическое расположение файлов образов и может быть различных типов.

Для настройки пулов носителей с помощью графической утилиты необходимо войти в ОС СН под учетной записью пользователя, включенного в группы `astra-admin` и `astra-console`, и запустить менеджер виртуальных машин virt-manager.

Далее в окне менеджера виртуальных машин выбрать подключение к нужному серверу виртуализации и перейти во вкладку «Хранилище». Для создания нового пула носителей нажать **[+]**, а для создания образа нажать **[Новый том]** в соответствии с рис. 16.

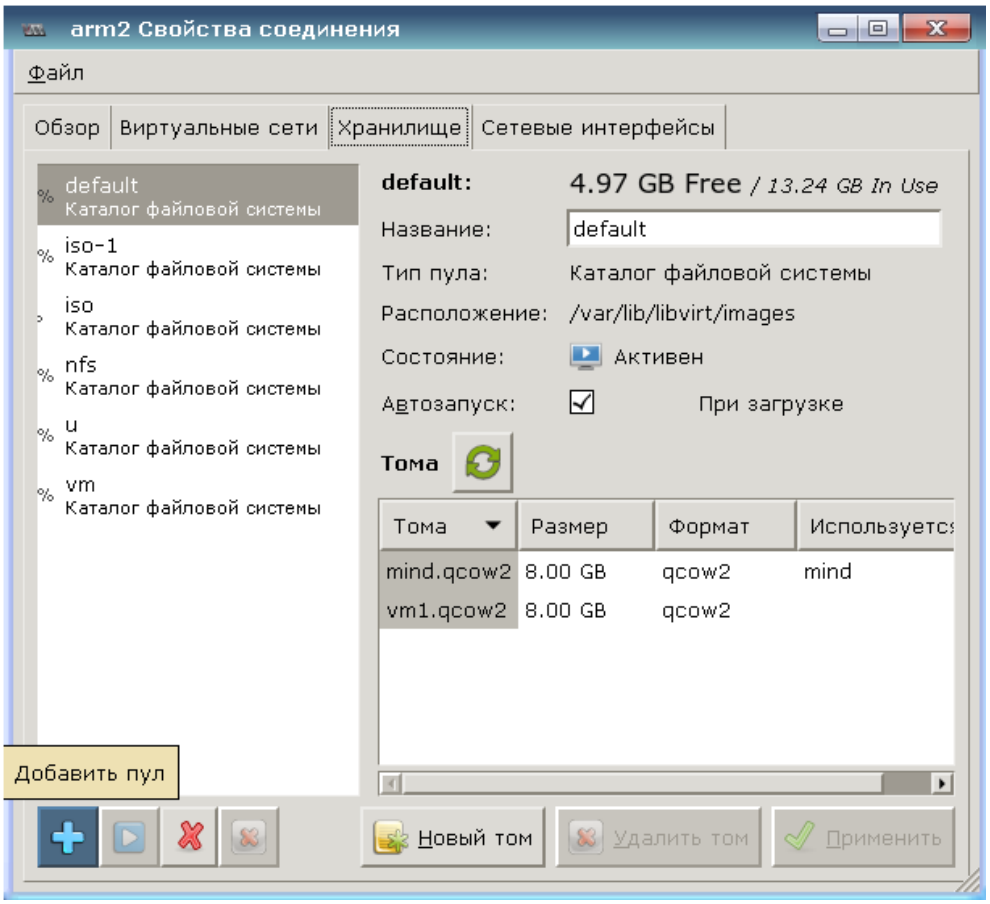

Рис. 16

На рис. 17 приведен внешний вид диалоговых окон при создании пула носителей и образа нового диска. В выпадающих списках, соответственно, необходимо выбрать тип пула или формат образа носителя.

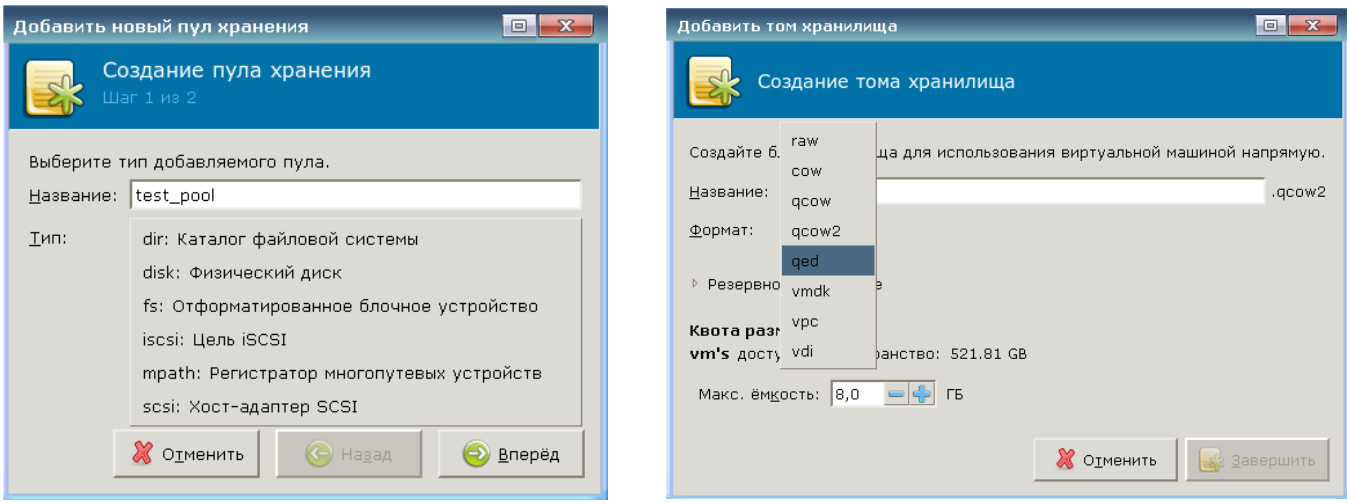

Рис. 17

ВНИМАНИЕ! В ПК СВ используются средства эмуляции аппаратного обеспечения на основе QEMU, которые поддерживают следующие форматы образов: `raw` (raw-формат, является фактически представлением физического диска) и формат `qcow2` (собственный формат QEMU, поддерживающий возможности сжатия, использования снимков и другие дополнительные возможности).

Примечание. Существует возможность использования форматов образов других средств эмуляции аппаратного обеспечения, например, VirtualBox. Но для запуска виртуальной машины требуется конвертация в собственный формат QEMU.

После создания образ может быть добавлен в конфигурацию виртуальной машины, например, с помощью окна свойств виртуальной машины в менеджере виртуальных машин virt-manager.

2.6. Управление виртуальными сетевыми интерфейсами и сетями

В ПК СВ существует возможность настройки виртуальных сетевых интерфейсов и сетей для обеспечения сетевого взаимодействия виртуальных машин как между собой, так и с операционной системой узла.

Для настройки сетевых интерфейсов с помощью графической утилиты необходимо войти в ОС СН под учетной записью пользователя, включенного в группы `astra-admin` и `astra-console`, и запустить менеджер виртуальных машин virt-manager.

Далее в окне менеджера виртуальных машин выбрать подключение к нужному серверу виртуализации, перейти во вкладку «Сетевые интерфейсы» и для добавления нового сетевого интерфейса нажать **[+]** в соответствии с рис. 18.

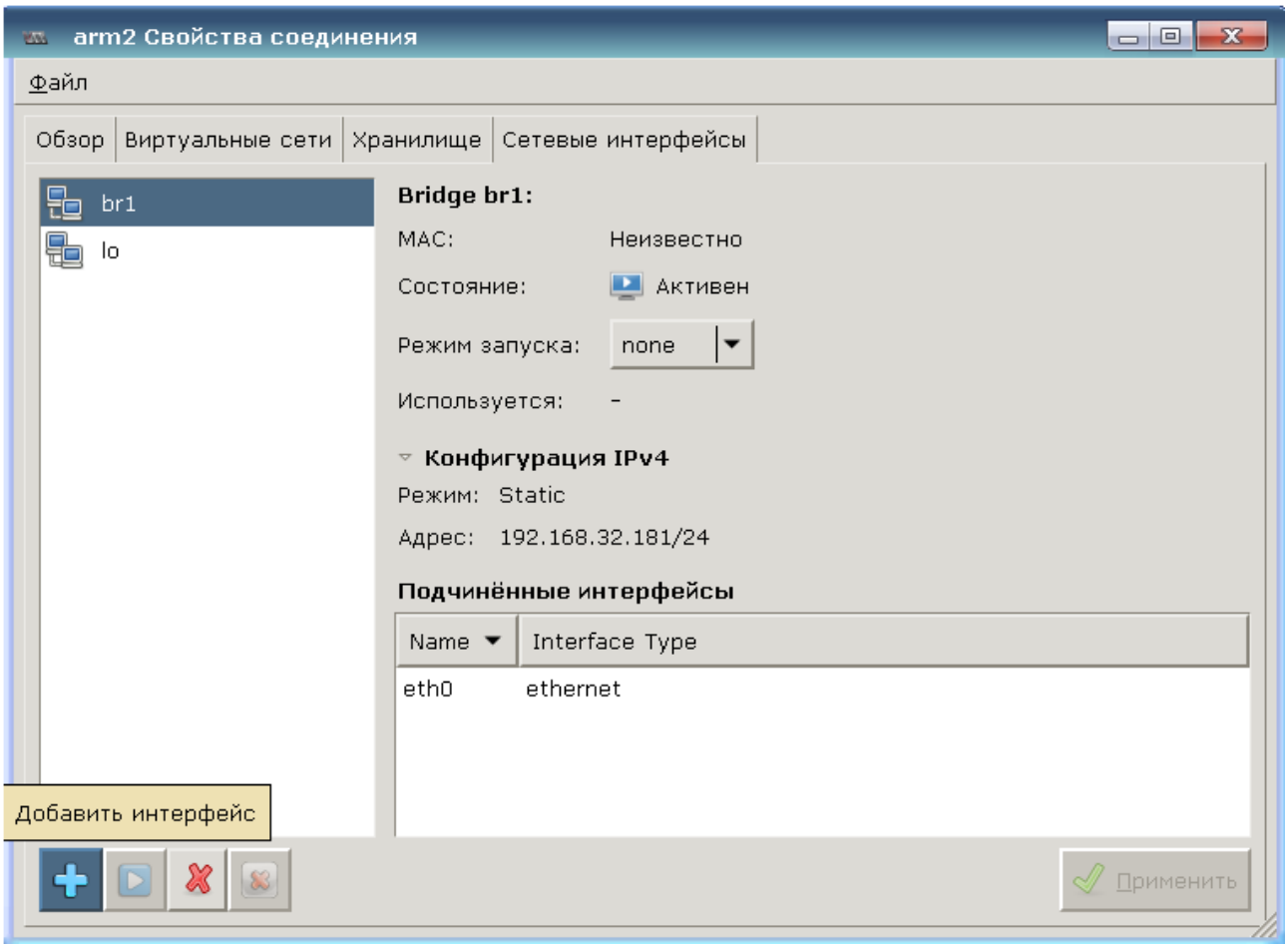

Рис. 18

Затем выбрать тип сетевого интерфейса, задать название и выбрать из выпадающего списка режим запуска в соответствии с рис. 19.

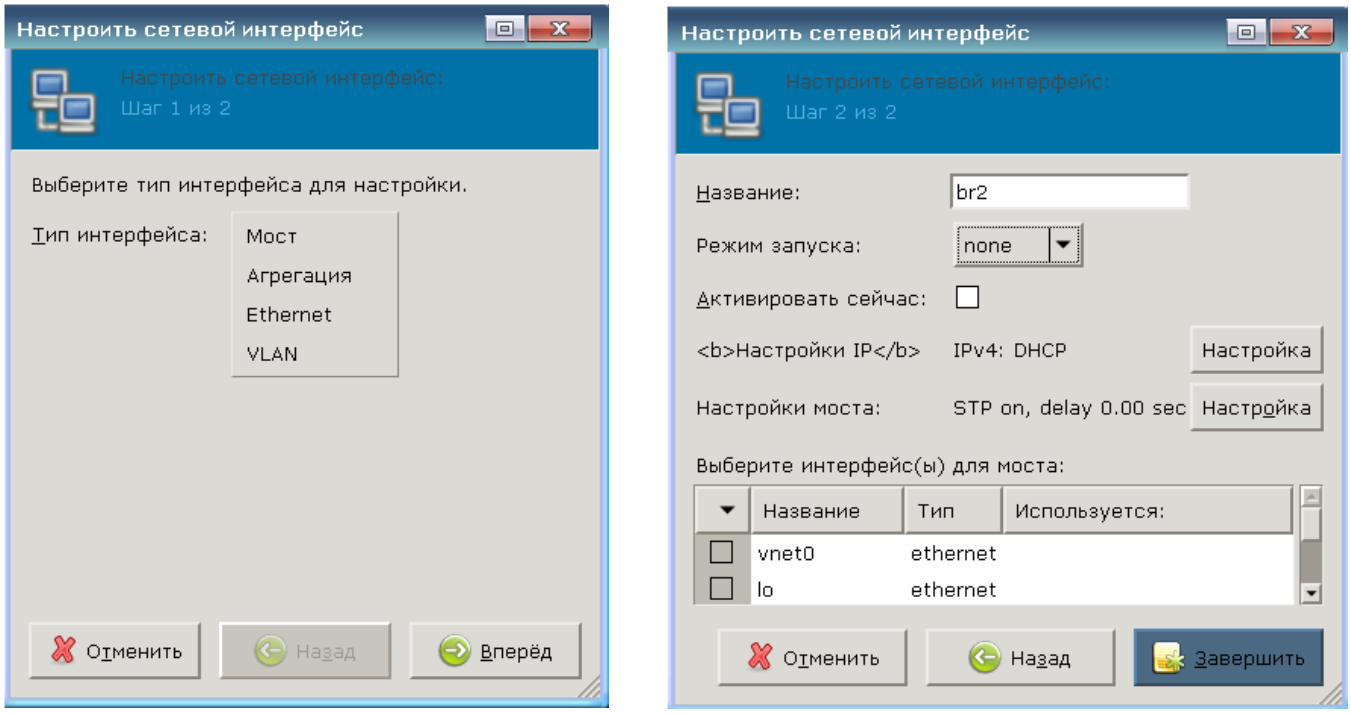

Рис. 19

Выбор виртуального сетевого интерфейса для ВМ выполняется в разделе конфигурирования данной ВМ (см. 2.3) как показано на рис. 20.

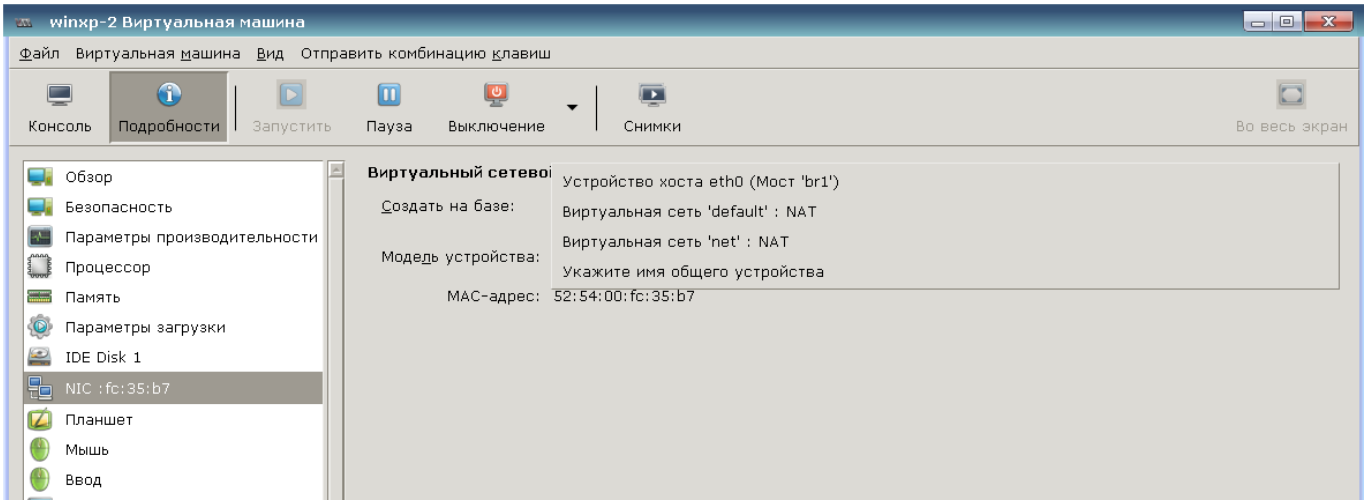

Рис. 20

Для настройки виртуальных сетей с помощью графической утилиты необходимо войти в ОС СН под учетной записью пользователя, включенного в группы `astra-admin` и `astra-console`, и запустить менеджер виртуальных машин virt-manager.

Далее выбрать подключение к нужному серверу виртуализации, перейти во вкладку «Виртуальные сети» и для добавления виртуальной сети нажать **[+]** в соответствии с рис. 21.

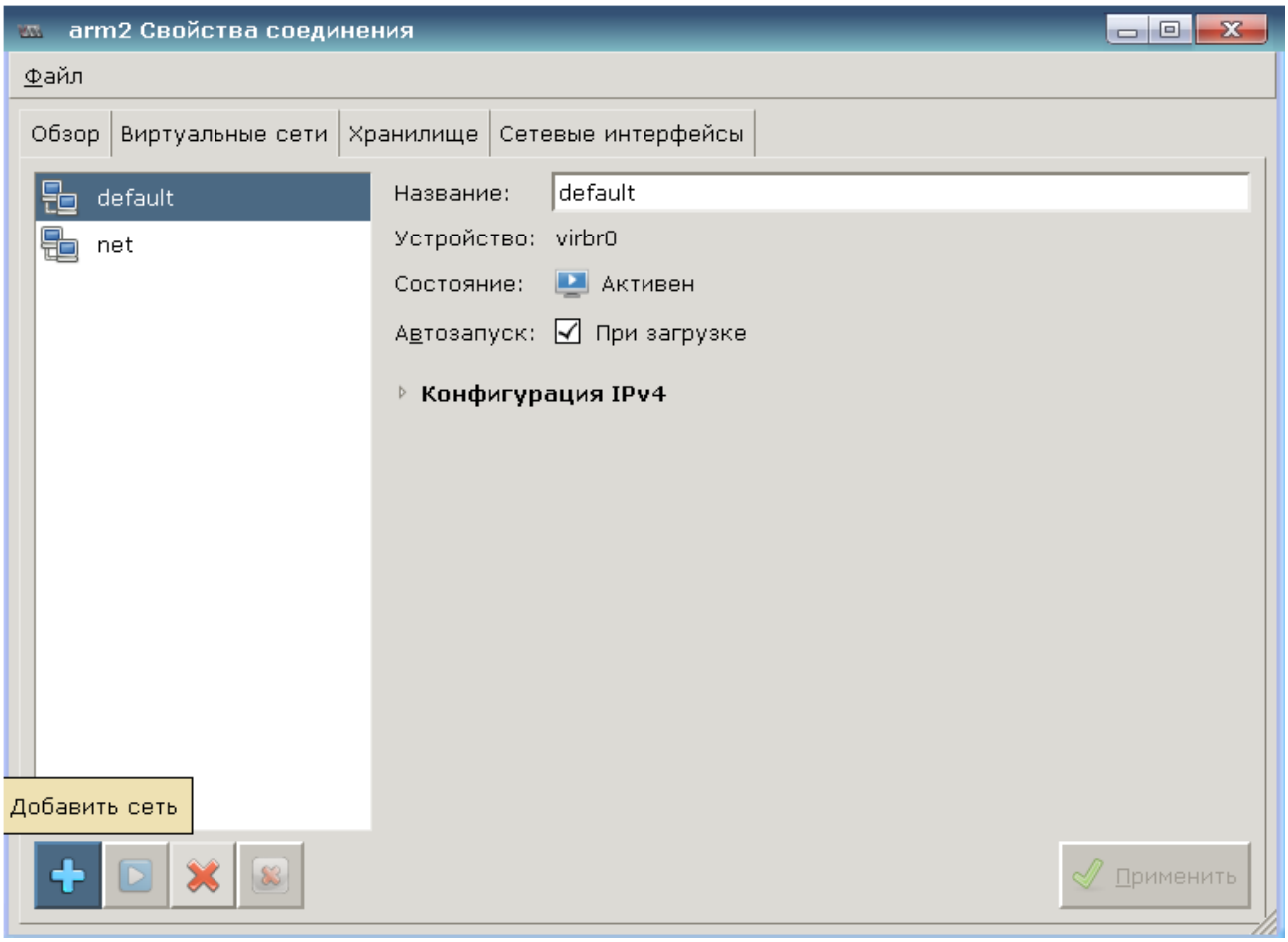

Рис. 21

Затем задать название и необходимые настройки (подсеть и подключение к физической сети) в соответствии с рис. 22.

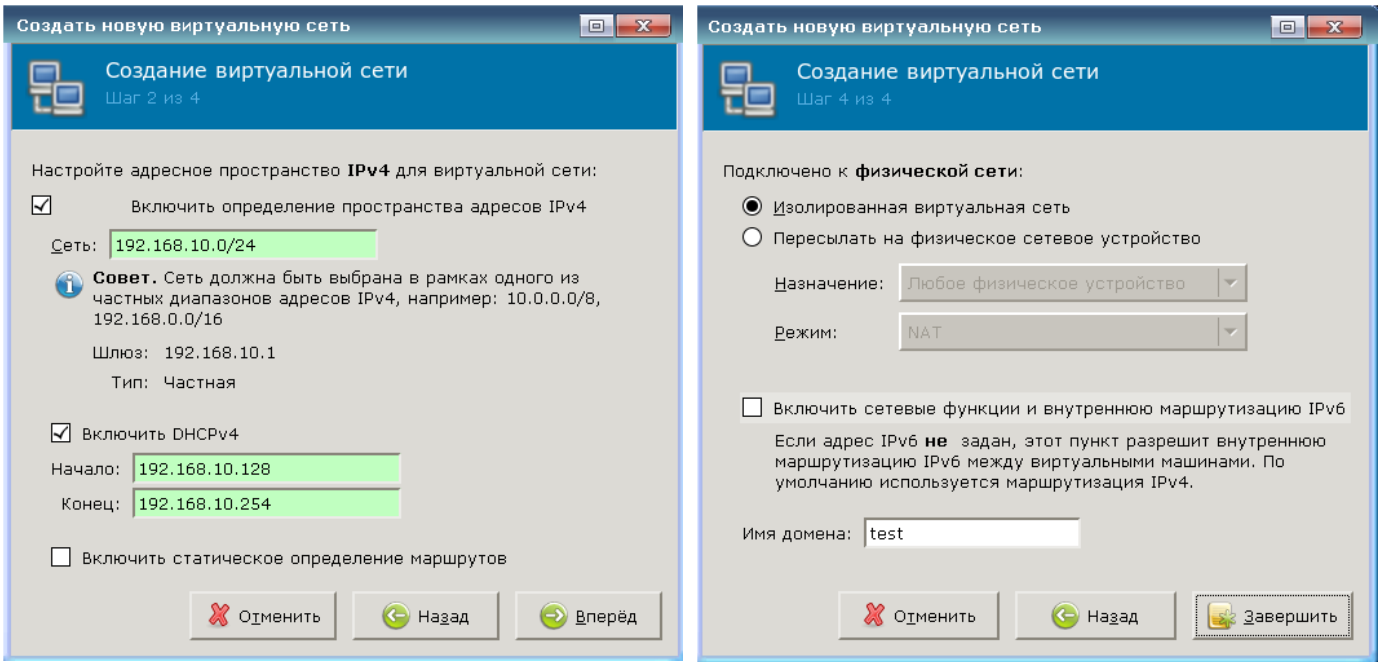

Рис. 22

3. ПОДКЛЮЧЕНИЕ ФИЗИЧЕСКОГО УСТРОЙСТВА К ВМ

3.1. Подключение USB-устройства к ВМ

Функционал ПК СВ обеспечивает подключение USB-устройства к ВМ и позволяет использовать его из сессии ВМ, одновременно ограничивая доступ к нему с узла.

Для подключения USB-устройства к ВМ требуется выполнить следующие действия:

1) настроить права доступа к устройству:

   а) запустить на узле утилиту `fly-admin-smc`;

   б) выбрать пункт меню «Устройства и правила — Устройства» и нажать **[+]** («Создать новый элемент»);

   в) подключить USB-устройство к узлу в соответствии с рис. 23;

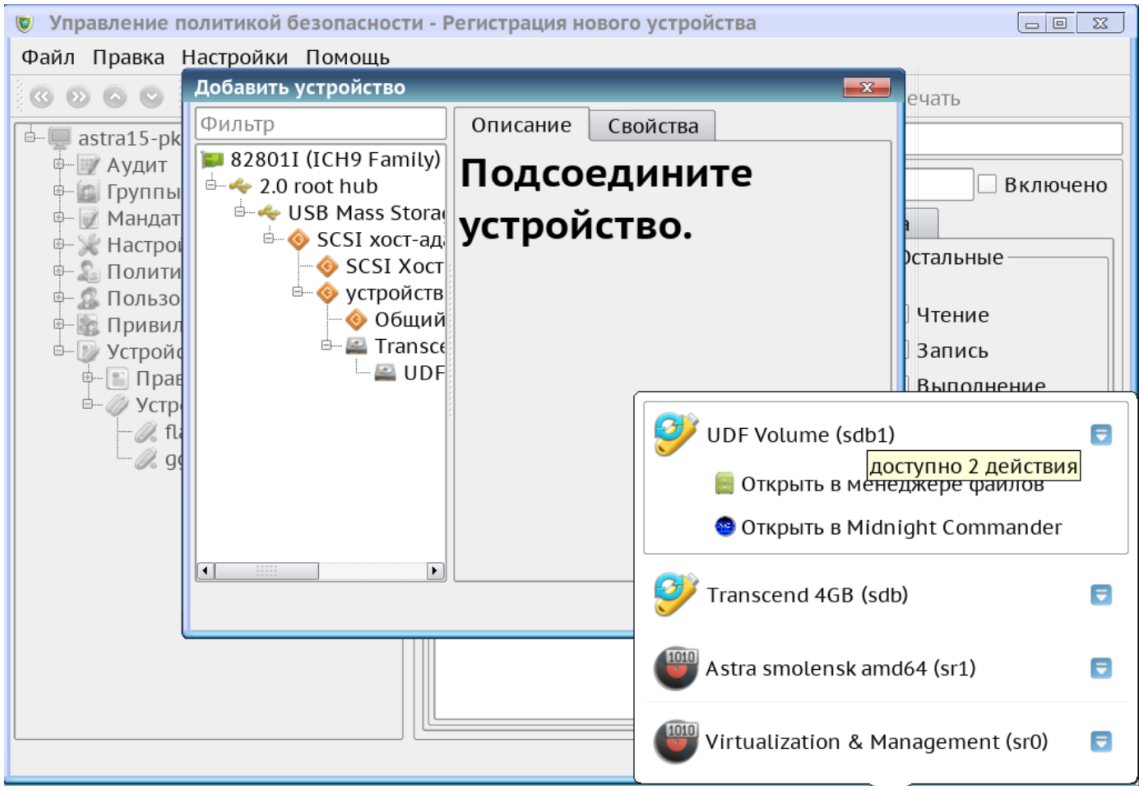

Рис. 23

   г) выбрать «устройство SCSI» и нажать кнопку **[Да]** в соответствии с рис. 24;

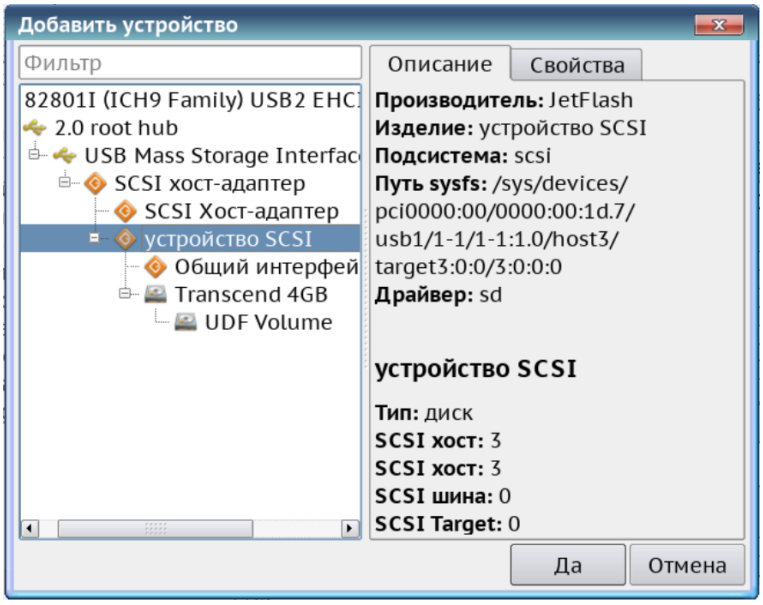

Рис. 24

д) задать название правила, указать пользователя, группу и атрибуты (права на устройство должны быть таковы, чтобы пользователь, от имени которого запускается ВМ, имел к нему доступ), установить флаг «Включено» и сохранить;

2) указать USB-устройство в настройках ВМ:

а) запустить virt-manager;

б) открыть требуемую ВМ и перейти в раздел конфигурирования;

в) нажать на кнопку **[Добавить оборудование]**;

г) выбрать тип «USB Host Device»;

д) выбрать необходимое USB-устройство в соответствии с рис. 25;

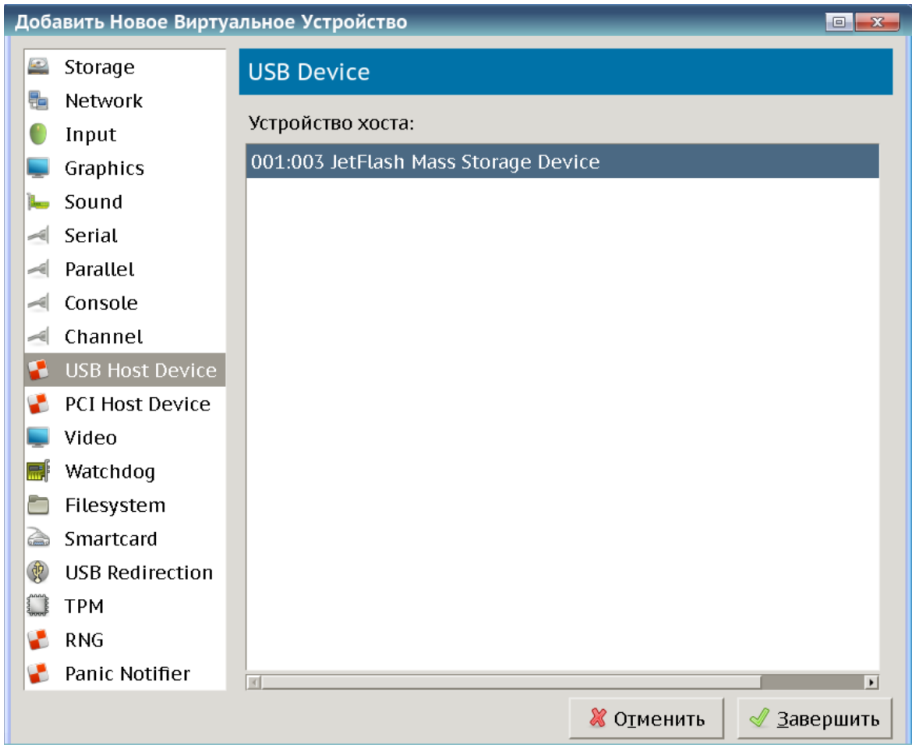

Рис. 25

3) перезагрузить узел.

3.2. Подключение жесткого диска к ВМ

Для подключения жесткого диска целиком к ВМ (на примере устройства `/dev/sdb`), необходимо выполнить следующие действия:

1) определить на узле требуемый диск:

```
ls -l /dev/disk/by-id/ | grep sdb
```

Результат выполнения команды будет содержать следующие строки:

```
... ata-WDC_WD1003FZEX-00K3CA0_WD-WCC6Y7YXUX4S -> ../../sdb
... scsi-SATA_WDC_WD1003FZEX-_WD-WCC6Y7YXUX4S -> ../../sdb
```

2) создать правило доступа к устройству в политиках безопасности:

а) запустить на узле утилиту `fly-admin-smc` («Пуск — Настройки — Политика безопасности»);

б) выбрать пункт меню «Устройства и правила — Устройства», нажать **[+]** («Создать новый элемент»), затем **[Да]**;

в) в окне настройки в секции «Свойства» нажать кнопку **[+]**. В появившейся таблице задать параметры устройства:

- «Тип»: `ENV`;
- «Ключ»: `ID_SERIAL`;
- «Операция»: `==`;
- «Значение»: `WDC_WD1003FZEX-00K3CA0_WD-WCC6Y7YXUX4S`;

г) задать наименование правила, указать пользователя, группу и атрибуты (права на устройство должны быть таковы, чтобы пользователь, от имени которого запускается ВМ, имел доступ к устройству), установить флаг для поля «Включено» и сохранить настройки в соответствии с рис. 26.

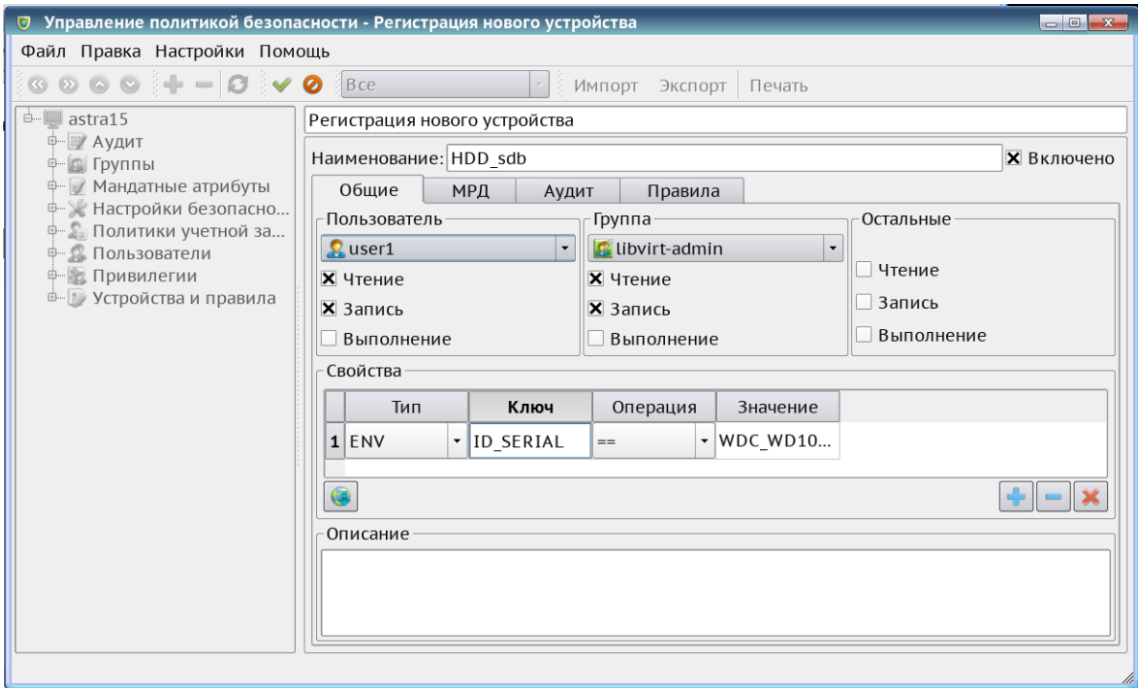

Рис. 26

3) указать HDD-устройство в настройках ВМ, отредактировав их с помощью инструмента командной строки `virsh`. Для этого выполнить команду:

```
virsh --connect qemu:///system edit vmname
```

и отредактировать файл, добавив в секцию `devices` описание блочного устройства:

```
<disk type='block' device='disk'>
<driver name='qemu' type='raw'/>
<source dev='/dev/sdb'/>
<target dev='sdb' bus='sata'/>
</disk>
```

затем сохранить конфигурацию;

4) перезагрузить узел.

3.3. Подключение PCI-устройства к ВМ

Подключение PCI-устройства к ВМ позволит использовать все возможности узла в ОС виртуальной машины. Для возможности подключения PCI-устройства к ВМ узел должен иметь материнскую плату с поддержкой блока управления памятью ввода/вывода (I/O MMU) — технология AMD-Vi(AMD) или VT-d(Intel). Для соответствующей технологии в BIOS материнской платы узла должно быть установлено значение «Enable».

В примере приведен алгоритм подключения к ВМ PCI-видеокарты. Аналогичным образом выполняется подключение других PCI-устройств, например, сетевой карты.

П р и м е р

Подключение PCI-видеокарты (Nvidia) к ВМ на узле на базе Intel

1) включить I/O MMU в ОС узла:

а) узнать «VendorID» и «SerialID» видеокарты, для этого в консоли выполнить:

```
lspci -nn | grep NVIDIA
```

Результат выполнения команды:

```
01:00.0 VGA compatible controller [0300]: NVIDIA Corporation Device
[10de:0fc6] (rev a1)

01:00.1 Audio device [0403]: NVIDIA Corporation Device
[10de:0e1b] (rev a1)
```

Первое устройство — видеокарта (`[10de:0fc6]`), второе — встроенная в нее звуковая карта (`[10de:0e1b]`);

б) включить в ядре соответствующие модули. Для этого в файл `/etc/modules` необходимо добавить строки:

```
pci_stub
vfio
vfio_iommu_type1
vfio_pci
vfio_virqfd
```

и пересобрать `initrd`, выполнив команду:

```
update-initramfs -u -k all
```

в) добавить в `grub` необходимые инструкции. Для этого в файле `/etc/default/grub` для параметра `GRUB_CMDLINE_LINUX_DEFAULT` добавить следующие данные:

```
intel_iommu=on pci-stub.ids=10de:0fc6,10de:0e1b
```

и обновить `grub`:

```
update-grub
```

г) добавить в `blacklist` драйвера Nvidia. Для этого в файл `/etc/modprobe.d/blacklist.conf` внести следующие строки:

```
blacklist nvidia
blacklist nouveau
blacklist lbm-nouveau
options nouveau modeset=0
alias nouveau off
alias lbm-nouveau off
```

д) создать исполняемый скрипт `/usr/sbin/vfio-bind` для подключения устройства к VFIO-PCI:

```
#!/bin/bash

modprobe vfio-pci
```

```
for dev in "$@"; do
vendor=$(cat /sys/bus/pci/devices/$dev/vendor)
device=$(cat /sys/bus/pci/devices/$dev/device)
if [ -e /sys/bus/pci/devices/$dev/driver ]; then
echo $dev > /sys/bus/pci/devices/$dev/driver/unbind
fi
echo $vendor $device > /sys/bus/pci/drivers/vfio-pci/new_id
done
```

выставить права, выполнив команду:

```
chmod 755 /usr/sbin/vfio-bind
```

и добавить скрипт в автозапуск с указанием ID устройств, полученных в результате выполнения действия согласно пункту а) настоящего перечисления. Для этого в файл `/etc/rc.local` добавить следующую строку:

```
/usr/bin/vfio-bind 0000:01:00.0 0000:01:00.1
```

е) выполнить перезагрузку узла. После перезагрузки следует проверить статус `IOMMU`:

- для процессора Intel выполнив команду:

  ```
  dmesg | grep -iE "(IOMMU|VT-d)"
  ```

- для процессора AMD выполнив команду:

  ```
  dmesg | grep -iE "(IOMMU|AMD-Vi)"
  ```

Результат выполнения команды должен содержать следующие строки:

```
...
[    0.000000] DMAR: IOMMU enabled
[    0.046886] DMAR-IR: IOAPIC id 2 under DRHD base
0xfed91000 IOMMU 1
...
[    0.591348] iommu: Adding device 0000:01:00.0 to group 1
[    0.591352] iommu: Adding device 0000:01:00.1 to group 1
[    0.892308] [drm] VT-d active for gfx access
```

ж) проверить pci-stub командой:

```
dmesg | grep pci-stub
```

Результат выполнения команды должен содержать следующие строки:

```
...
[    8.678781] pci-stub: add 10DE:0FC6 sub=FFFFFFFF:FFFFFFFF
cls=00000000/00000000
[    8.678786] pci-stub: add 10DE:0E1B sub=FFFFFFFF:FFFFFFFF
cls=00000000/00000000
```

2) создать и сконфигурировать ВМ:

а) создать ВМ с виртуальным набором микросхем (chipset) `pc-q35-2.1` с помощью инструмента `virt-install`:

```
virt-install --connect qemu:///system -n vmname -r 4096 --vcpus=2
     -c /path/to/iso/smolensk.iso --disk
     pool=default,size=20,bus=virtio,format=qcow2 --os-type=linux
     --graphics vnc --video cirrus --machine pc-q35-2.1
```

где `vmname` — название виртуальной машины;

`--vcpus=2` — количество ядер процессора;

`4096` — объем оперативной памяти;

`/path/to/iso/smolensk.iso` — путь к ISO-образу установочного диска ОС СН;

`--disk pool=default` — дисковое хранилище;

`size=20` — размер хранилища в ГБ;

`--graphics vnc` — тип дисплея гостевой системы;

`--video cirrus` — тип видеоустройства гостевой системы.

После выполнения команды необходимо запустить virt-manager и установить гостевую ОС. После завершения установки ОС выключить ВМ;

б) отредактировать конфигурационный файл ВМ с помощью инструмента `virsh` выполнив команду:

```
virsh --connect qemu:///system edit vmname
```

и внеся в файл следующие изменения:

1) заменить блок:

```
<domain type='kvm'>
```

на блок:

```
<domain type='kvm'
xmlns:qemu='http://libvirt.org/schemas/domain/qemu/1.0'>
```

2) заменить блок:

```
<features>
<acpi/>
<apic/>
<pae/>
</features>
```

на блок:

```
<features>
<acpi/>
<apic/>
```

```
<pae/>
<kvm>
<hidden state='on'/>
</kvm>
</features>
```

3) для подключения PCI-устройства добавить в конце конфигурационного файла блок:

```
<qemu:commandline>
<qemu:arg value='-device'/>
<qemu:arg value='ioh3420,bus=pcie.0,addr=1c.0,
multifunction=on,port=1,chassis=1,id=root.1'/>
<qemu:arg value='-device'/>
<qemu:arg value='vfio-pci,host=01:00.0,bus=root.1,addr=00.0,
multifunction=on,x-vga=on'/>
<qemu:arg value='-device'/>
<qemu:arg value='vfio-pci,host=01:00.1,bus=root.1,addr=00.1'/>
</qemu:commandline>
```

где `host=01:00.0` и `host=01:00.1` — адреса шин видео и звука соответственно.

Сохранить изменения конфигурационного файла;

в) запустить ВМ. В списке устройств должны появится новые устройства — видеокарта и встроенная в нее звуковая карта. Необходимо установить драйвера и перезагрузить гостевую систему.

3.4. Подключение последовательного порта к ВМ

Подключение устройства, подключенного к последовательному порту (например, `/dev/ttyS0`), можно выполнить двумя способами:

1) в конфигурационном файле ВМ добавить следующий блок:

```
<devices>
  <serial type="dev">
    <source path="/dev/ttyS0"/>
    <target port="1"/>
  </serial>
</devices>
```

2) в менеджере виртуальных машин virt-manager в разделе конфигурирования ВМ необходимо нажать на кнопку **[Добавить оборудование]** и в появившемся окне указать необходимое устройство в соответствии с рис. 27.

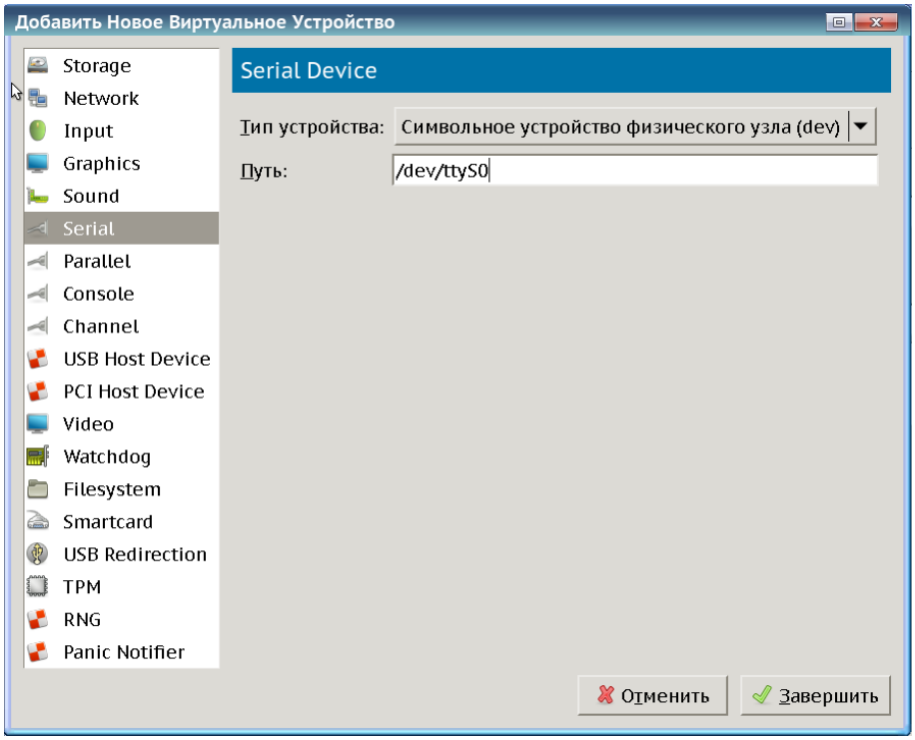
Добавить Новое Виртуальное Устройство
Storage
Network
Input
Graphics
Sound
Serial
Parallel
Console
Channel
USB Host Device
PCI Host Device
Video
Watchdog
Filesystem
Smartcard
USB Redirection
TPM
RNG
Panic Notifier
Serial Device
Тип устройства:
Символьное устройство физического узла (dev)
Путь:
/dev/ttyS0
Отменить
Завершить

Рис. 27

4. ВЗАИМОДЕЙСТВИЕ АДМИНИСТРАТОРА С СЗИ

Взаимодействие администратора с СЗИ ПК СВ состоит из обязательного прохождения процедуры аутентификации на сервере виртуализации libvirt и работы в условиях применения дискреционного и мандатного управления доступом.

Основными задачами администрирования являются:

- настройка удаленного доступа к серверу виртуализации libvirt;
- настройка удаленного доступа к рабочим столам виртуальных машин по протоколам VNC и SPICE;
- дискреционное управление доступом к виртуальным машинам;
- мандатное управление доступом к виртуальным машинам;
- управление режимом запрета модификации файлов-образов виртуальных машин.

Примечание. Настройка сервера виртуализации для работы в ЕПП ОС СН, а также идентификация и аутентификация при доступе к серверу виртуализации libvirt и к рабочему столу виртуальных машин выполняется в соответствии с документом РУСБ.10015-01 95 01-1.

ВНИМАНИЕ! Дискреционное и мандатное управление доступом, а также управление режимом запрета модификации файлов-образов виртуальных машин осуществляются СЗИ из состава ОС СН.

ВНИМАНИЕ! Операции по изменению состава или конфигурации ВМ требуют вхождения пользователя в специальную локальную административную группу `libvirt-admin`.

4.1. Настройка дискреционного и мандатного управление доступом к ВМ

Настройка дискреционного и мандатного управления доступом осуществляется посредством web-интерфейса OpenNebula в самом экземпляре ВМ во вкладке «Security» («Безопасность») в соответствии с рис. 28.

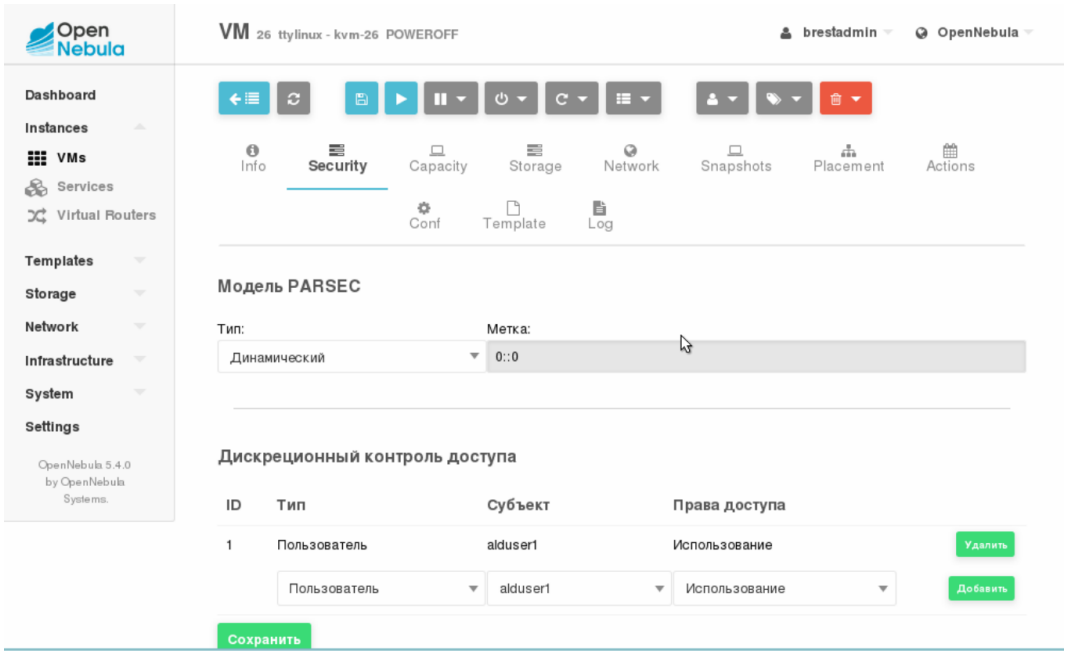

Рис. 28

В OpenNebula различаются два типа доступа к виртуальной машине:

- «Использование» — просмотр свойств, запуск и работа с виртуальной машиной;
- «Администрирование» — полный доступ к ВМ, включая запуск, правку ее свойств и управление правами доступа к ней.

Для настройки дискреционного управления доступом необходимо в соответствующих полях формы, приведенной на рис. 28, выбрать «Тип» (пользователь или группа), «Субъект» и уровень доступа и нажать на кнопку **[Добавить]**.

Для настройки мандатного управления доступа требуется указать модель `parsec` (динамическая или статическая). При статической модели в соответствующем поле необходимо также указать метку.

После внесения изменений необходимо нажать на кнопку **[Сохранить]**.

ВНИМАНИЕ! Настройка доступа возможно только при выключенной ВМ.

4.2. Ограничения по конфигурированию и запуску виртуальных машин в условиях мандатного управления доступом

Для запуска виртуальной машины от имени пользователя с его мандатными атрибутами безопасности необходимо войти в ОС СН под учетной записью данного пользователя с заданным мандатным уровнем. После чего запустить менеджер виртуальных машин virt-manager, выбрать виртуальную машину из списка и нажать на кнопку **[Запустить]** в соответствии с рис. 29.

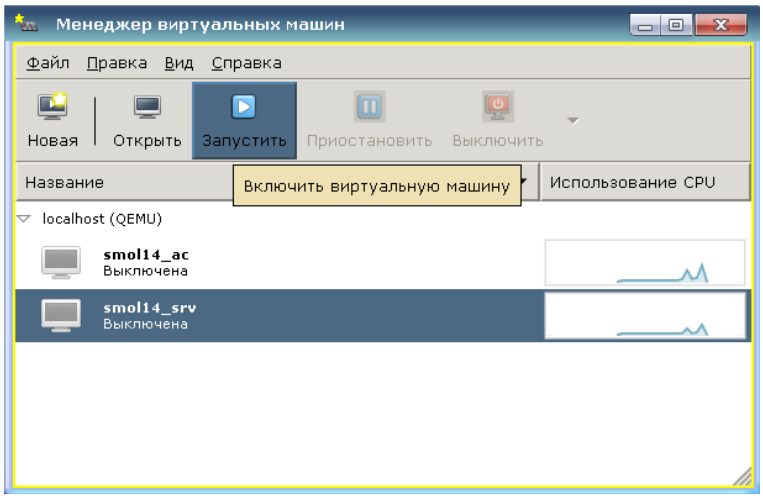

Рис. 29

Примечания:

1. При запуске ВМ приобретает динамические метки безопасности, включающие идентификатор запустившего ее пользователя и его мандатные атрибуты. После этого при доступе к данной ВМ применяются дискреционное и мандатное управление доступом.

2. В случае использования в качестве гостевой системы ОС СН виртуальная машина не должна запускаться в мандатном контексте. Вместо этого необходимо выполнять удаленный вход с требуемым мандатным уровнем доступа средствами ОС СН.

3. При запуске ВМ в ненулевом мандатном контексте рекомендуется использовать режим только на чтение.

ВНИМАНИЕ! Только запустивший ВМ пользователь или пользователь, входящий в группу администраторов `libvirt-admin`, могут управлять функционированием ВМ (приостанавливать, останавливать и т.п.).

ВНИМАНИЕ! При использовании ВМ с ненулевым мандатным контекстом в качестве сетевого адаптера не может быть выбрано устройство `virtio`.

5. СРЕДСТВА ОБЕСПЕЧЕНИЯ ОТКАЗОУСТОЙЧИВОСТИ

5.1. Программное обеспечение кластера высокой доступности. Pacemaker и Corosync

Pacemaker и Corosync являются программным обеспечением для построения кластерных систем высокой доступности. Описание и пример настройки Pacemaker и Corosync для создания кластера высокой доступности приведены в РУСБ.10015-01 95 01-1.

5.2. Отказоустойчивые сервисы Samba и NFS

Для настройки отказоустойчивых сервисов Samba и NFS из состава ОС СН необходимо последовательно выполнить действия согласно 5.2.1–5.2.4 для каждого из сервисов. Действия согласно 5.2.1–5.2.3 выполняются идентично для обоих сервисов.

5.2.1. Настроить сеть:

1) объединить в один логический интерфейс на серверах кластера путем установки пакета `ifenslave-2.6`:

```
apt install ifenslave-2.6
```

2) в файле `/etc/network/interfaces` указать настройки для сетевых интерфейсов:

- первого сервера:

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
      address 10.10.10.99
netmask 255.255.255.0
gateway 10.10.10.2

auto eth1
iface eth1 inet static
     address 192.168.25.8
     netmask 255.255.255.0

auto bond0
iface bond0 inet static
     address 192.168.122.10
         netmask 255.255.255.0
        gateway 192.168.122.1
```

```
        bond_mode balance-rr
        bond_miimon 100
        bond_downdelay 200
        bond_updelay 200
        slaves eth2 eth3
```

- второго сервера:

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
      address 10.10.10.100
netmask 255.255.255.0
gateway 10.10.10.2

auto eth1
iface eth1 inet static
     address 192.168.25.9
     netmask 255.255.255.0

auto bond0
iface bond0 inet static
     address 192.168.122.11
         netmask 255.255.255.0
        gateway 192.168.122.1
     bond_mode balance-rr
     bond_miimon 100
     bond_downdelay 200
     bond_updelay 200
     slaves eth2 eth3
```

где `eth0` — интерфейс для доступа к отдельным узлам кластера;

`eth1` — heartbeat-интерфейс, по которому осуществляется доступ к сервису;

`bond0` — логический интерфейс (bonding) на базе физических `eth2` и `eth3` и используется для синхронизации DRBD-ресурсов;

3) на обоих серверах выполнить следующие команды:

```
modprobe bonding
echo 'bonding' >> /etc/modules
```

```
init 6
```

5.2.2. Установить и настроить блочное устройство Distributed Replicated Block Device (DRBD) в соответствии с 6.1.1.

5.2.3. Установить и настроить Pacemaker и Corosync в соответствии с РУСБ.10015-01 95 01-1.

5.2.4. Настроить отказоустойчивость сервиса:

1) Samba-сервер:

а) перед настройкой ресурсов кластера на обоих серверах необходимо убрать из автозагрузки скрипты, которые в дальнейшем будут контролироваться `corosync`:

```
update-rc.d -f drbd remove
update-rc.d -f samba remove
```

б) создать ресурсы, объединить их в группы и определить порядок их запуска. Все действия выполнять на первом сервере:

- создать ресурс DRBD-раздела:

```
crm configure primitive drbd_main ocf:linbit:drbd params
     drbd_resource="main" op monitor interval="60s" op start
     interval="0" timeout="240s" op stop interval="0"
     timeout="240s" op monitor role=Master interval="10s" op
     monitor role=Slave interval="30s"
```

- указать определение ведущего (master)/ведомого (slave) для созданного DRBD-ресурса :

```
crm configure ms ms_drbd_main drbd_main meta master-max="1"
     master-node-max="1" clone-max="2" clone-node-max="1"
     notify="true"
```

- создать ресурс точки монтирования DRBD-раздела, например, каталог `/opt/main`:

```
crm configure primitive fs_main ocf:heartbeat:Filesystem params
     device="/dev/drbd0"\ directory="/opt/main" fstype="ext4"
     options="noatime"
```

- создать ресурс отказоустойчивого IP-адреса (по данному адресу будет выполняться обращение к Samba-серверу):

```
crm configure primitive failover_ip ocf:heartbeat:IPaddr2 params
     ip=192.168.25.10 nic=eth1
```

- создать ресурс `samba`:

```
crm configure primitive samba lsb:samba
```

- организовать созданные ресурсы служб/сервисов в группы:

```
crm configure group group_main fs_main failover_ip samba
```

- указать правило монтирования — монтировать ресурсы на узле, где

DRBD-ресурсы находятся в состоянии ведущих. Задать имя, например, cluster_resources:

```
crm configure colocation cluster_resources
     inf: fs_main ms_drbd_main:Master
```

- указать порядок запуска сервисов — после запуска соответствующих DRBD-разделов. Задать имя, например, all_after_drbd:

```
crm configure order all_after_drbd
     inf: ms_drbd_main:promote group_main:start
```

- установить предпочтение запуска ресурсов ведущего/ведомого:

```
crm configure location prefer_main_server-1
     ms_drbd_main 100: server-1
crm configure location prefer_main_server-2
     ms_drbd_main 50: server-2
crm configure location prefer_group_main_server-1
     group_main 100: server-1
crm configure location prefer_group_main_server-2
     group_main 50: server-2
```

Из вышеуказанных настроек следует, что основным сервером для запуска службы Samba является server-1.

Настройка завершена, проверка состояния выполняется командой crm status.

Пример

Результат проверки

```
============
Last updated: Thu Jun 16 10:58:33 2016
Last change: Thu Jun 16 10:58:10 2016 via cibadmin on server-2
Stack: openais
Current DC: server-1 - partition with quorum
Version: 1.1.7-ee0730e13d124c3d58f00016c3376a1de5323cff
2 Nodes configured, 2 expected votes
5 Resources configured.
============

Online: [ server-1 server-2 ]

Master/Slave Set: ms_drbd_main [drbd_main]
Masters: [ server-1 ]
Slaves: [ server-2 ]
```

```
Resource Group: group_main
fs_main    (ocf::heartbeat:Filesystem):  Started server-1
failover_ip    (ocf::heartbeat:IPaddr2): Started server-1
samba    (lsb:samba):    Started server-1
```

в) настроить Samba-сервер. Для этого на каждом сервере указать настройки для директории `/opt/main` в файле `/etc/samba/smb.conf`:

```
[share-drbd]
comment = Astra Linux Samba Share
path = /opt/main
browsable = yes
guest ok = yes
read only = no
create mask = 0755
```

2) NFS-сервер:

а) перед настройкой ресурсов кластера на обоих серверах необходимо убрать из автозагрузки скрипты, которые в дальнейшем будут контролироваться `corosync`:

```
update-rc.d -f drbd remove
update-rc.d -f nfs-common remove
update-rc.d -f nfs-kernel-server remove
```

б) создать ресурсы, объединить их в группы и определить порядок их запуска. Все действия выполнять на первом сервере:

- создать ресурс DRBD-раздела:

```
crm configure primitive drbd_main ocf:linbit:drbd params
     drbd_resource="main" op monitor interval="60s" op start
     interval="0" timeout="240s" op stop interval="0"
     timeout="240s" op monitor role=Master interval="10s"
     op monitor role=Slave interval="30s"
```

- указать определение ведущего/ведомого для созданного DRBD-ресурса:

```
crm configure ms ms_drbd_main drbd_main meta master-max="1"
     master-node-max="1" clone-max="2" clone-node-max="1"
     notify="true"
```

- создать ресурс точки монтирования DRBD-раздела, например, каталог `/opt/main`:

```
crm configure primitive fs_main ocf:heartbeat:Filesystem
     params device="/dev/drbd0"\ directory="/opt/main"
     fstype="ext4" options="noatime"
```

- создать ресурс отказоустойчивого IP-адреса (по данному адресу будет выполняться обращение к NFS-серверу):

```
crm configure primitive failover_ip ocf:heartbeat:IPaddr2
     params ip=192.168.25.10 nic=eth1
```

- создать ресурсы nfs:

```
crm configure primitive nfs-kernel-server lsb:nfs-kernel-server
crm configure primitive nfs-common lsb:nfs-common
```

- организовать созданные ресурсы служб/сервисов в группы:

```
crm configure group group_main fs_main failover_ip
     nfs-kernel-server nfs-common
```

- указать правило монтирования — монтировать ресурсы на узле, где DRBD-ресурсы находятся в состоянии ведущих. Задать имя, например, cluster_resources:

```
crm configure colocation cluster_resources inf: fs_main
     ms_drbd_main:Master
```

- указать порядок запуска сервисов — после запуска соответствующих DRBD-разделов. Задать имя, например, all_after_drbd:

```
crm configure order all_after_drbd inf: ms_drbd_main:promote
     group_main:start
```

- установить предпочтение запуска ресурсов ведущего/ведомого:

```
crm configure location prefer_main_server-1
     ms_drbd_main 100: server-1
crm configure location prefer_main_server-2
     ms_drbd_main 50: server-2
crm configure location prefer_group_main_server-1
     group_main 100: server-1
crm configure location prefer_group_main_server-2
     group_main 50: server-2
```

Из вышеуказанных настроек следует, что основным сервером для запуска службы NFS является server-1.

Настройка завершена, проверка состояния выполняется командой crm status.

Пример

Результат проверки

```
============
Last updated: Sun Jun 19 21:20:06 2016
Last change: Sun Jun 19 21:18:23 2016 via cibadmin on
server-2
Stack: openais
Current DC: server-1 - partition with quorum
```

```
Version: 1.1.7-ee0730e13d124c3d58f00016c3376a1de5323cff
2 Nodes configured, 2 expected votes
6 Resources configured.
============
Online: [ server-1 server-2 ]

Master/Slave Set: ms_drbd_main [drbd_main]
Masters: [ server-1 ]
Slaves: [ server-2 ]
Resource Group: group_main
fs_main (ocf::heartbeat:Filesystem): Started server-1
failover_ip (ocf::heartbeat:IPaddr2): Started server-1
nfs-kernel-server (lsb:nfs-kernel-server): Started server-1
nfs-common (lsb:nfs-common): Started server-1
```

в) настроить NFS-сервер. Для этого необходимо на каждом сервере указать настройки для директории `/opt/main` в файле `/etc/exports`:

```
/opt/main *(ro,sync)  # Только чтение для всех
```

6. СРЕДСТВА ВИРТУАЛИЗАЦИИ ДИСКОВЫХ ПРОСТРАНСТВ

6.1. Организация сетевого RAID

ПК СВ поддерживает работу с набором инструментов DRBD из состава ОС СН для организации сетевого RAID.

Distributed Replicated Block Device (DRBD) — распределенное реплицируемое блочное устройство, предназначенное для построения отказоустойчивых кластерных систем. DRBD полностью отражает по сети все операции с блочным устройством. DRBD является сетевым RAID-1.

Технология DRBD поддерживает только два узла, более сложные конструкции могут строиться путем использования drbd-устройства в качестве локального для еще одного drbd-устройства. Узлы могут работать в режиме первичного (primary) или вторичного (secondary) узла. Вторичный узел хранит данные, но не позволяет осуществить к ним локальный доступ, первичный позволяет осуществить доступ. DRBD поддерживает режим «первичный — первичный», при котором возможен доступ к обоим узлам.

DRBD работает локально на узле и обеспечивает репликацию на удаленный узел содержимого локального блочного устройства. Для использования создается новое устройство, обычно `/dev/drbdX`, где `X` — число. Для нормальной работы DRBD должен быть запущен на обоих узлах. Если узел работает в режиме вторичного, то на нем установлено соответствующее drbd-устройство, доступ к которому запрещен. В случае повышения режима работы узла до первичного, доступ к drbd-устройству открывается. Большинство операций осуществляется с помощью утилиты `drbdadm`, при этом фактическая работа выполняется на уровне ядра. Для повышения скорости и надежности работы необходимо наличие отдельного сетевого интерфейса или нескольких интерфейсов, с помощью которых будет происходить репликация между узлами.

6.1.1. Установка и настройка DRBD

Для установки поддержки технологии DRBD необходимо выполнить следующее:

1) из репозитория ПК СВ установить следующие пакеты:

```
apt install lvm2 drbd8-utils
```

2) запустить модуль DRBD:

```
modprobe drbd
```

Пример

Настройка DRBD на двух серверах с ОС СН. На каждом сервере имеется неиспользуемый раздел `sdb1`

Для настройки необходимо выполнить следующий порядок действий:

1) подготовить разделы для синхронизации с использованием технологии LVM. На

каждом сервере создать логический том `main` размером 100 ГБ:

```
pvcreate /dev/sdb1
vgcreate vg0 /dev/sdb1
lvcreate -L100G -n main vg0
```

2) на обоих серверах создать одинаковый конфигурационный файл `/etc/drbd.d/main.res`:

```
resource main {
protocol C;

disk {
fencing resource-only;
}

handlers {
fence-peer "/usr/lib/drbd/crm-fence-peer.sh";
after-resync-target "/usr/lib/drbd/crm-unfence-peer.sh";
}

net {
after-sb-0pri discard-least-changes;
after-sb-1pri call-pri-lost-after-sb;
after-sb-2pri call-pri-lost-after-sb;
}
syncer {
rate 1000M;
}
on server-1
{
device /dev/drbd0;
disk /dev/vg0/main;
address 192.168.122.10:7794;
meta-disk internal;
}
on server-2
{
device /dev/drbd0;
disk /dev/vg0/main;
address 192.168.122.11:7794;
```

```
meta-disk internal;
}
}
```

В секции `syncer` указывается максимальная скорость синхронизации данных между узлами. В секциях `on server-1` и `on server-2` располагаются параметры каждого из серверов. Для параметра `disk` указываются синхронизируемые LVM-тома, а для параметра `address` — IP-адрес и порт сервера, по которому будет проходить синхронизация;

3) создать описанный ресурс DRBD, выполнив на обоих серверах следующие команды:

```
drbdadm create-md main
drbdadm up all
```

4) на первом сервере сделать созданный DRBD-ресурс первичным (primary):

```
drbdadm -- --overwrite-data-of-peer primary all
```

Проверка статуса выполняется с помощью команды:

```
service drbd status
```

5) на первом сервере создать файловую систему, выполнив команды:

```
mkfs.ext4 /dev/drbd0
tune2fs -m 0 /dev/drbd0
```

Далее необходимо дождаться синхронизации, после которой блочное устройство `/dev/drbd0` готово к использованию.

6.2. Распределенные параллельные файловые системы с защитой от сбоев

Распределенные файловые системы используются в высокоскоростных вычислениях и фокусируются на высокой доступности, производительности и масштабируемости. ПК СВ поддерживает использование распределенной файловой системы Ceph из состава ОС СН. Описание и настройка Ceph приведены в РУСБ.10015-01 95 01-1.

6.3. Средства интеграции и дифференциации блочных устройств

6.3.1. DRBD

Подробная информация по работе с DRBD представлена в 6.1.

6.3.2. ISCSI

iSCSI — протокол, базирующийся на TCP/IP и разработанный для установления взаимодействия и управления системами хранения данных, серверами и клиентами. iSCSI описывает:

- транспортный протокол для SCSI, который работает поверх TCP;
- механизм инкапсуляции SCSI команд в IP-сети;

- протокол для нового поколения систем хранения данных, которые будут использовать «родной» TCP/IP.

Системы на основе iSCSI могут быть построены с использованием любой достаточно быстрой технической конфигурации, поддерживающей протокол IP, например, Gigabit Ethernet или 10G Ethernet. Использование стандартного протокола позволяет применять стандартные средства контроля и управления потоком.

При описании протокола iSCSI используется следующая терминология:

- initiator — объект-инициатор, устанавливающий соединение с целью (target), в общем случае — узел. Осуществляет ввод/вывод информации на блочные устройства;
- target — экспортируемый объект. В зависимости от контекста целью является целиком экспортирующий узел или только экспортируемый объект. Сам объект может делиться на несколько LUN;
- portal — группа целей (targets), которые анонсируются вместе. В общем случае один узел хранения соответствует одной группе целей;
- iqn — полное имя участника взаимодействия. Существует у инициатора и у цели;
- endpoint — уточненное имя ресурса, в общем случае включает в себя iqn, номер LUN и указание на конкретный метод доступа к нему (например, номер соединения, LUN и IP-адрес, с которого следует получать доступ к устройству);
- LUN (Logical Unit Number) — номер объекта внутри цели (target). Ближайшим аналогом является раздел диска или отдельный том.

В ПК СВ в качестве цели используется программный комплекс Linux-IO Target (LIO), в качестве инициатора — open-iscsi.

6.3.2.1. Установка цели

Для поддержки iSCSI-цели LIO необходимо установить следующий пакет:

```
apt install targetcli-fb
```

Для управления целью и ее настройки используется командная оболочка, доступ к которой можно получить, выполнив в терминале команду `targetcli`. Появится специальная командная строка с ограниченным набором инструкций для настройки ISCSI-целей.

Команда `targetcli` представляет цель в виде иерархически построенной структуры и делит цель на back-end и front-end части. На рис. 30 представлена структура цели.

```
/> ls
o- / ..... [...]
  o- backstores ..... [...]
  | o- block ..... [Storage Objects: 0]
  | o- fileio ..... [Storage Objects: 0]
  | o- pscsi ..... [Storage Objects: 0]
  | o- ramdisk ..... [Storage Objects: 0]
  o- iscsi ..... [Targets: 0]
  o- loopback ..... [Targets: 0]
  o- vhost ..... [Targets: 0]
/>
```

Рис. 30

Структура цели включает:

- `Backstores` — раздел, в котором отображаются объекты-хранилища. Backstores является скрытой частью iSCSI-цели;
- `fileio` — файл;
- `block` —жесткий диск, символические ссылки на диск или LVM;
- `pscsi` (SCSI pass-through) — любое устройство для хранения информации, поддерживающее SCSI команды напрямую, то есть без эмуляции SCSI. Не рекомендуется к использованию, т.к. может привести к повреждению оборудования;
- `iSCSI` — видимая часть iSCSI-цели. Содержит список целей, LUN, права доступа и т.п.;
- `loopback` — модуль, предоставляющий доступ к цели локально.

В таблице 1 представлено описание основных команд, используемых в командной оболочке `targetcli`.

Таблица 1 – Список команд `targetcli`

Команда	Описание
`help [topic]`	Вывести на экран справку
`ls [path] [depth]`	Вывести на экран структуры цели `path` глубиной `depth`
`pwd`	Вывести на экран текущий путь дерева
`get [group] [parameter...]`	Получить значение параметра группы
`set [group] [parameter=value...]`	Установить значение параметра в группе
`saveconfig`	Сохранить конфигурацию
`exit`	Выйти из командной оболочки `targetcli`

6.3.2.2. Пример реализации цели

Настройка цели сервера с ОС СН (IP-адрес: `192.168.25.10`). В качестве блочного устройства с предоставленным общим доступом выступает диск `/dev/sdb`:

1) создать блочное устройство с именем `disk1`:

```
cd /backstores/block
create name=disk1 dev=/dev/sdb
```

2) создать цель (экспортируемый объект):

```
cd /iscsi
create
```

В результате внутри iSCSI будет создана структура, приведенная на рис. 31. Для просмотра структуры выполнить команду `ls /`.

Рис. 31

В структуре, приведенной на рис. 31:

- `iqn.2003-01.org.linux-iscsi.node1.x8664:sn.af8d10a0e170` — имя цели;
- `tpg1` (Target Portal Group) — список IP-адресов и TCP портов, которые будет слушать данная цель;
- `acl` — список адресов, с которых можно будет подключиться к цели;
- `luns` — список LUN;
- `portals` — список IP-адресов и портов, которые будет слушать цель.

3) создать LUN на основе созданного хранилища `disk1`:

```
cd /iscsi/iqn.2003-01.org.linux-iscsi.node1.x8664:sn.af8d10a0e170/tpgt1/
          luns
create /backstores/block/disk1
```

4) добавить в список доступа (ACL) имя инициатора, которое хранится в файле `/etc/iscsi/initiator.name` на машине-инициаторе (см. 6.3.2.4):

```
cd /iscsi/iqn.2003-01.org.linux-iscsi.node1.x8664:sn.af8d10a0e170/tpgt1/
          acls
create iqn.1993-08.org.debian:01:b43c731f4920
```

Результат настроек представлен на рис. 32.

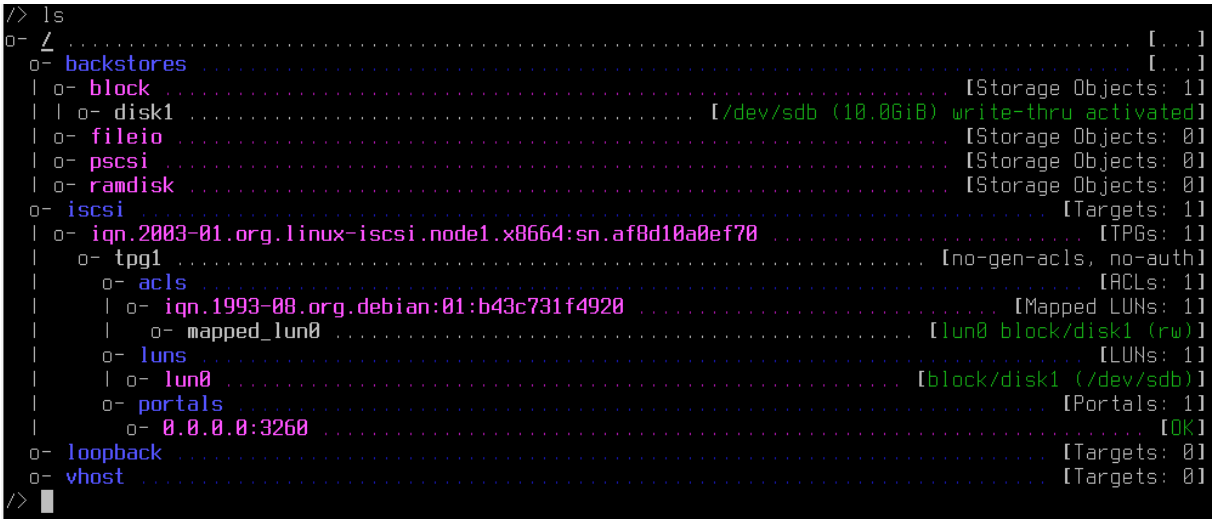

Рис. 32

5) выполнить сохранение настроек и выйти из утилиты:

```
/ saveconfig
exit
```

6.3.2.3. Установка инициатора

Для подключения iSCSI-устройства необходимо установить пакет `open-iscsi`:

```
apt install open-iscsi
```

6.3.2.4. Пример реализации инициатора

Порядок действий по установке и настройке инициатора на примере клиента с IP-адресом `192.168.25.11`:

1) после установки пакета `open-iscsi` запустить одноименный сервис:

```
/etc/init.d/open-iscsi start
```

После первого запуска сгенерируется уникальный код инициатора, который можно просмотреть в файле `/etc/iscsi/initiatorname.iscsi`:

```
InitiatorName=iqn.1993-08.org.debian:01:c1a92326f6b8
```

Данное имя используется в acl при настройке цели (см. 6.3.2.1).

2) переключить параметр `node.startup` в файле `/etc/iscsi/iscsid.conf` в значение `automatic`:

```
node.startup = automatic
```

3) перезапустить сервис:

```
/etc/init.d/open-iscsi start
```

4) выполнить сканирование для поиска iSCSI-целей и подключение таргета:

```
sudo iscsiadm -m discovery -t st -p 192.168.25.10
sudo iscsiadm -m node -l
```

где `192.168.25.10` — IP-адрес цели.

7. ПРОГРАММНЫЙ КОММУТАТОР OPEN VSWITCH

Open vSwitch (OVS) — программный многоуровневый коммутатор, обеспечивающий агрегацию портов, обнаружение петель, зеркалирование портов, сбор статистики о трафике на NetFlow-коллектор, изоляцию сети с помощью сетей VLAN путем тегирования портов, а также фильтрацию базовой сети и централизованное управление программными коммутаторами с помощью протокола OpenFlow.

Архитектура OVS состоит из трех основных компонентов: базы данных, программного коммутатора и управляющего контроллера. На каждом из физических узлов вместе с гипервизором располагаются собственные БД и коммутатор. БД обеспечивает хранение всей конфигурации своего узла: настройки интерфейсов, портов, различные правила и прочее. Коммутатор передает пакеты. Распределенность OVS достигается с помощью контроллера.

Коммутация пакетов происходит на уровне ядра, также поддерживается коммутация в пользовательском пространстве. При коммутации в пользовательском пространстве снижается производительность по причине частых переключений между режимами ядра и пользователя.

ВНИМАНИЕ! Данное программное средство в текущей версии не поддерживает классификационные метки и может использоваться только на минимальном уровне конфиденциальности.

7.1. Установка

На каждом узле, предназначенном для включения в сеть, должен быть установлен пакет `openvswitch-switch` программного коммутатора OVS . Для установки пакета используется команда:

```
apt openvswitch-switch
```

7.2. Особенности конфигурирования

Конфигурация всех Open vSwitch коммутаторов, портов, настройки поддерживаемых протоколов хранятся в собственной базе данных OVS (OVSDB). В стандартной конфигурации в OVSDB существуют следующие таблицы:

- Open_vSwitch — Схема
- Bridge
- Port
- Interface
- Flow_Table — конфигурация OpenFlow
- QoS

- Mirror
- Controller — параметры подключения к контроллеру OpenFlow
- Manager — конфигурация OVSDB
- NetFlow
- SSL
- sFlow
- IPFIX
- Flow_Sample_Collector_Set

Изначально почти все таблицы пусты, так как конфигурация отсутствует. Утилита `ovs-vsctl` используется для внесения изменений в OVSDB. Команда для внесения изменений в БД имеет следующий синтаксис:

```
ovs-vsctl <команда> <таблица> <запись> <ключ=значение>
```

Для создания программного коммутатора используется команда вида:

```
ovs-vsctl add-br <имя_коммутатора>
```

Для добавления порта коммутатора используется команда:

```
ovs-vsctl add-port <имя_коммутатора> <имя_порта>
```

Просмотреть записи, присутствующие в таблице, описывающей порты, можно с помощью команды:

```
ovs-vsctl list port
```

Для вывода списка портов, включенных в конкретный VLAN, необходимо выполнить команду:

```
ovs-vsctl find port tag=10
```

Конфигурация OVS заданная командами `ovs-vsctl` автоматически применяется и сохраняется в файле `/etc/network/interfaces`.

8. ОБЛАЧНАЯ СИСТЕМНАЯ АРХИТЕКТУРА OPENNEBULA

При создании надежного и эффективного облака предварительно необходимо выбрать его архитектуру в соответствии с предполагаемым использованием. Для этого требуется определить, какие компоненты центра обработки и хранения данных (ЦОХД) должны входить в облако. В общем случае в состав облака входят все компоненты инфраструктуры, такие как инструментальные средства организации сети, хранения, авторизации и виртуализации.

Также при создании облака рассчитывается его возможный размер, характеристики рабочей нагрузки, количество пользователей и другие параметры, обеспечивающие высокую степень доступности системы, определяется обеспечение рабочего процесса, т. е. как конечные пользователи будут изолироваться и использовать облако.

8.1. Обзор архитектуры и определение основных характеристик облака

При использовании платформы OpenNebula применяемая физическая инфраструктура должна иметь кластероподобную архитектуру с машиной предварительной обработки данных (фронтальная машина) и несколькими узлами, на которых располагаются виртуальные машины. При этом физическая сеть должна соединять все узлы с машиной предварительной обработки данных (рис. 33).

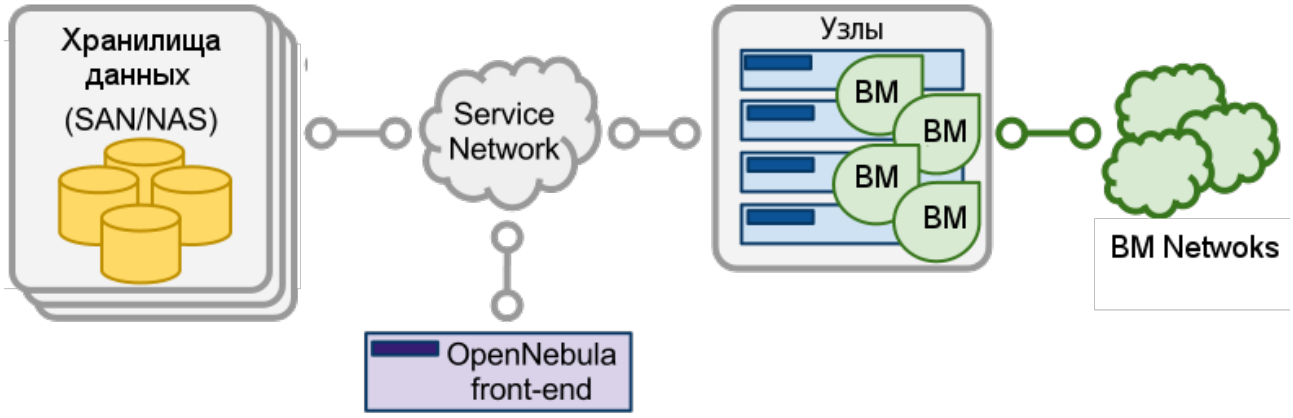

Рис. 33

Основными компонентами облачной архитектуры являются: хранилище данных, архитектура сети и виртуализация. Базовыми компонентами системы OpenNebula являются:

- машина предварительной обработки данных (фронтальная машина) — служит для установки OpenNebula и выполнения ее сервисов;
- узлы с поддержкой гипервизоров Kernel-based Virtual Machine (KVM) — предоставляют необходимые для виртуальных машин ресурсы;
- хранилища данных (Datastores) — служат для хранения базовых образов виртуальных машин;

- физическая сеть — используются для поддержки базовых сервисов, таких как объединение серверов хранения данных, операций управления OpenNebula и виртуальных сетей для виртуальных машин. Может быть использовано несколько физических сетей.

Размер инфраструктуры облака возможно определить непосредственно на основании предполагаемой рабочей нагрузки (т.е. виртуальных машин), которую должна выдерживать облачная инфраструктура. В 8.2–8.10 приведены основные показатели, которые необходимо учитывать при определении параметров облака OpenNebula.

8.2. Фронтальная машина

8.2.1. Характеристики фронтальной машины

Минимальные рекомендуемые характеристики фронтальной машины для OpenNebula приведены в таблице 2.

Таблица 2

Ресурсы	Минимальная рекомендуемая конфигурация
Память	2 ГБ
ЦП	1 ЦП (2-ядерный)
Размер диска	100 ГБ
Сеть	2 NICS

Максимальное количество серверов (узлов виртуализации), которым можно управлять с помощью одного экземпляра OpenNebula, зависит от производительности и масштабируемости инфраструктуры базовой платформы и главным образом от подсистемы хранения данных. Не рекомендуется использовать один экземпляр платформы для управления более чем 500 серверами.

Фронтальная машина должна иметь сетевое соединение со всеми узлами и, по возможности, доступ к хранилищам данных (как локальным, так и сетевым). Для базовой установки OpenNebula требуется не более 150 МБ.

В состав сервисов OpenNebula входят:

- сервис `oned` (присоединенная программа управления);
- планировщик `mm_sched`;
- web-интерфейс Sunstone для работы с сервером (`sunstone-server`);

Примечание. Данные компоненты связываются через XML-RPC и могут устанавливаться на различных машинах.

По умолчанию после установки OpenNebula настроена на использование с БД PostgreSQL.

8.2.2. Предварительная настройка ОС СН

Для использования возможностей ПК СВ необходимо выполнить следующие предварительные настройки ОС СН:

- повысить максимальный уровень целостности системы с 63 до 127 и установить флаг загрузки параметра ядра `parsec.ccnr_relax`. Для этого необходимо в файле `/etc/default/grub` в строке `GRUB_CMDLINE_LINUX_DEFAULT` заменить:

```
parsec.max_ilev=63
```

на:

```
parsec.max_ilev=127
parsec.ccnr_relax=1
```

- для сохранения изменений от имени администратора выполнить команды:

```
sudo pdpl-user -i 127 <user>
sudo update-grub
```

где `<user>` — имя пользователя, который должен иметь высокий уровень целостности;

- выполнить перезагрузку системы.

8.2.3. Конфигурирование фронтальной машины

Для развертывания облачной платформы OpenNebula на фронтальной машине под управлением ОС СН необходимо, чтобы она входила в домен ALD или FreeIPA. Далее приводится вариант настройки для домена ALD.

ВНИМАНИЕ! В настройках домена должны быть разрешены локальные группы `kvm`, `libvirt-admin`, `libvirt`, `libvirt-qemu`, `disk`, `astra-admin` и `astra-console`. Для этого на контроллере домена в файле `/etc/ald/ald.conf` в секции `ALLOWED_LOCAL_GROUPS` указать данные группы.

После добавления ПК СВ в список источников (файл `sources.list`) следует выполнить на сервере команды:

```
$ sudo aptitude install brestcloud
$ sudo ald-libvirt-qemu-configure
```

Затем запустить мастер конфигурирования ПК СВ:

```
$ sudo brestcloud-configure
```

Для возможности выполнения функций администратора добавить доменного пользователя `brestadmin` в группы `astra-admin` и `astra-console`. Затем перегрузить сервер.

Управление облаком осуществляется с помощью web-интерфейса по адресу `http://main-server-ip/`.

Для доступа к web-интерфейсу с помощью браузера Mozilla Firefox необходимо в настройках браузера указать серверы, для которых доступна аутентификация `negotiate`:

1) в браузере открыть страницу настроек, введя в адресной строке `about:config`,

согласиться с предупреждением;

2) установить значение «http://,https://» для параметров:

   а) network.negotiate-auth.delegation-uris;

   б) network.negotiate-auth.trusted-uris.

8.2.4. Алгоритм Raft

Алгоритм Raft позволяет объединять несколько экземпляров фронтальной машины в зону, конфигурацию которой можно менять (добавлять и удалять узлы), не прерывая работу облака. Для этой зоны выделяется плавающий (способный при необходимости переходить от одного узла к другому) IP-адрес. Из доступных узлов выбирается лидер, которому присваивается ранее выделенный IP-адрес. Лидер обслуживает все входящие запросы. Все изменения на лидере синхронизируются с остальными узлами зоны. Если работа лидера прерывается на 100 миллисекунд, то выбирается новый лидер из числа исправных узлов. Выделенный для зоны IP-адрес присваивается новому лидеру. Таким образом обеспечивается высокая доступность фронтальной машины.

Для работы Raft должны быть соблюдены следующие требования:

1) настроен по крайней мере один контроллер домена;
2) настроена фронтальная машина облака;
3) имеется четное число (не менее двух) дополнительных узлов;
4) ни на одном из узлов не развернут сервис apache2 в режиме «AstraMode off»;
5) выделен IP для настройки плавающего IP-адреса кластера.

8.3. Конфигурация сервиса oned

Сервис oned управляет узлами кластера, виртуальными сетями, виртуальными машинами, пользователями, группами и базами данных хранилищ. Конфигурационный файл для сервиса oned.conf находится в каталоге /etc/one.

8.3.1. Параметры настройки сервиса

Файл конфигурации сервиса поддерживает настройку параметров, приведенных в таблице 3.

Таблица 3

Параметр	Описание
MANAGER_TIMER	Время в секундах, необходимое ядру для оценки периодических функций
MONITORING_INTERVAL	Время в секундах между циклами мониторинга. Параметр не может иметь значение меньше, чем параметр MANAGER_TIMER

Окончание таблицы 3

Параметр	Описание
MONITORING_THREADS	Максимальное количество потоков, используемых для обработки сообщений контролирующей программы
HOST_PER_INTERVAL	Количество узлов, контролируемых в каждый интервал времени
HOST_MONITORING_EXPIRATION_TIME	Время в секундах, через которое истекает срок контрольной информации. Использовать 0 для отключения записи мониторинга узла
VM_INDIVIDUAL_MONITORING	Контрольная информация ВМ получается вместе с информацией об узле. Для некоторых специальных драйверов может потребоваться активация индивидуального процесса мониторинга ВМ
VM_PER_INTERVAL	Количество ВМ, контролируемых в каждый интервал времени
VM_MONITORING_EXPIRATION_TIME	Время в секундах, через которое истекает срок контрольной информации. Использовать 0 для отключения записи мониторинга ВМ
SCRIPTS_REMOTE_DIR	Удаленный путь для хранения скрипта мониторинга и управления ВМ
PORT	Порт, на котором oned будет принимать запросы xml-rpc
LISTEN_ADDRESS	IP-адрес узла для приема запросов xml-rpc (по умолчанию все IP-адреса)
DB	Параметры настройки БД, используется для БД PostgreSQL
VNC_PORTS	Пул портов VNC для автоматического назначения портов VNC, по возможности, устанавливать порт на START+VMID: - start — первый назначаемый порт; - reserved — список зарезервированных портов, разделенный запятыми. Два номера, разделенные двоеточием, указывают диапазон
VM_SUBMIT_ON_HOLD	Принудительное создание ВМ в состоянии удержания вместо состояния ожидания. Возможные значения ДА или НЕТ
LOG	Настройка системы регистрации: - SYSTEM — возможные значения: file (по умолчанию), syslog или std; - DEBUG_LEVEL — устанавливает уровень отладки зарегистрированных сообщений. Возможные значения: - 0 — ошибка; - 1 — предупреждение; - 2 — информация; - 3 — отладка

Пример

```
#*******************************************************************************
# Атрибуты настройки сервиса
#*******************************************************************************

LOG = [
SYSTEM = "file", DEBUG_LEVEL = 3
]

#MANAGER_TIMER = 15

MONITORING_INTERVAL = 60
MONITORING_THREADS= 50

 #HOST_PER_INTERVAL = 15
 #HOST_MONITORING_EXPIRATION_TIME = 43200
 #VM_INDIVIDUAL_MONITORING =  "no"
 #VM_PER_INTERVAL  = 5
 #VM_MONITORING_EXPIRATION_TIME = 14400

 SCRIPTS_REMOTE_DIR=/var/tmp/one
 PORT = 2633
 LISTEN_ADDRESS = "0.0.0.0"

 #DB = [ BACKEND = "sqlite" ]
 # Пример конфигурации для PostgreSQL
 DB = [ BACKEND = "pgsql",
 SERVER = "localhost",
 PORT = 5432,
 USER = "postgres",
 PASSWD = "postgres",
 DB_NAME = "opennebula" ]

 VNC_PORTS = [
    START= 5900
    #RESERVED = "6800, 6801, 9869"
 ]
```

```
#VM_SUBMIT_ON_HOLD = "NO"
```

8.3.2. Виртуальные сети

Виртуальные сети поддерживают настройку параметров, приведенных в таблице 4.

Таблица 4

Параметр	Описание
NETWORK_SIZE	Определяет размер по умолчанию для виртуальных сетей
MAC_PREFIX	MAC-префикс по умолчанию, предназначенный для создания автоматически генерируемых MAC-адресов (может переписываться шаблоном виртуальной сети)
VLAN_IDS	Пул VLAN ID для автоматического назначения VLAN_ID. Данный пул предназначен для сетей 802.1Q (Open vSwitch и драйверы 802.1Q). Драйвер будет сначала пытаться назначить VLAN_IDS[ЗАПУСК]+VNET_ID: - start — первый VLAN_ID, который может использоваться; - reserved — список номеров VLAN_ID, разделенных запятыми. Два номера, разделенные двоеточием, указывают диапазон
VXLAN_IDS	Автоматическое назначение идентификатора сети VXLAN (VNI). Используется для сетей VXLAN. start — первый VNI, который может использоваться

Пример

```
#*******************************************************************************
# Конфигурация физических сетей
#*******************************************************************************

NETWORK_SIZE = 254
MAC_PREFIX = "02:00" VLAN_IDS = [
START = "2",
RESERVED = "0, 1, 4095"
]

VXLAN_IDS = [
START = "2"
]
```

8.3.3. Хранилища

Система хранилищ позволяет пользователям устанавливать образы, которые могут быть образами ОС или данных, для использования в виртуальных машинах. Данные образы могут использоваться несколькими виртуальными машинами одновременно, а также совместно с другими пользователями. Подробное описание хранилищ приведено в разделе 10.

В хранилищах и шаблонах образов можно настроить значения по умолчанию, приведенные в таблице 5.

Таблица 5

Параметр	Описание
DATASTORE_LOCATION	Путь к хранилищам. Одинаков для всех узлов и фронтальной машины. По умолчанию /var/lib/one/datastores (в независимом режиме устанавливается по умолчанию $ONE_LOCATION/var/datastores). Каждое хранилище имеет собственный каталог, называемый BASE_PATH, в виде: $DATASTORE_LOCATION/<datastore_id> При необходимости можно присвоить символьную ссылку на этот каталог любому пути. Значение BASE_PATH генерируется из данного параметра каждый раз при запуске oned
DATASTORE_CAPACITY_CHECK	Проверяет наличие достаточного пространства до создания нового образа. Значение по умолчанию Yes
DEFAULT_IMAGE_TYPE	Значение по умолчанию для поля TYPE, если оно отсутствует в шаблоне. Возможные значения: - OS — файл образа, содержащий операционную систему; - CDROM — файл образа, содержащий CDROM; - DATABLOCK — файл образа, содержащий блок данных, создаваемый как пустой блок
DEFAULT_DEVICE_PREFIX	Значение по умолчанию для поля DEV_PREFIX, если оно отсутствует в шаблоне. Отсутствующее поле DEV_PREFIX заполняется, когда создаются образы, поэтому изменение префикса не повлияет на существующие образы. Возможные значения: - префикс hd — для устройства IDE; - префикс sd — для устройства SCSI; - префикс vd — для виртуального диска KVM
DEFAULT_CDROM_DEVICE_PREFIX	Аналогично DEFAULT_DEVICE_PREFIX, но для устройств CDROM

Пример

```
#*******************************************************************************
# Конфигурация репозитория образов
#*******************************************************************************

DATASTORE_CAPACITY_CHECK="yes"

DEFAULT_IMAGE_TYPE="OS"
DEFAUL_ DEVICE_PREFIX ="hd"
```

```
DEFAUL_CDROM_DEVICE_PREFIX="hd"
```

8.3.4. Сборщик информации

За сборку информации отвечает драйвер `collectd`. Драйвер поддерживает установку параметров, приведенных в таблице 6.

Таблица 6

`-a:`	Адрес для привязки `collectd` сокета, по умолчанию 0.0.0.0
`-p:`	UDP-порт для перехвата контрольной информации, по умолчанию 4124
`-f:`	Интервал в секундах для записи на диск собранной информации, по умолчанию 5
`-t:`	Количество потоков для сервера, по умолчанию 50
`-i:`	Время в секундах цикла push мониторинга. Данный параметр должен быть меньше, чем `MONITORING_INTERVAL` (см. таблицу 26), в противном случае push мониторинг не будет эффективным

ВНИМАНИЕ! Данный драйвер не может назначаться для узла и должен использоваться с драйверами KVM (см. 8.3.5).

Пример

```
IM_MAD=[
name           ="collectd"
executable     ="collectd",
arguments      ="-p 4124-f S-t 50 -i20"
]
```

8.3.5. Информационный драйвер

Информационные драйверы используются для сбора информации с узлов кластера. Поддерживают установку параметров, приведенных в таблице 7.

Таблица 7

`name`	Имя информационного драйвера
`executable`	Путь исполняемого модуля информационного драйвера, может быть абсолютным или относительным `/usr/lib/one/mads/`
`arguments`	Конфигурационный файл для исполняемого драйвера, может быть абсолютным или относительным `/etc/one/`

8.3.6. Система хуков

Хуки в OpenNebula являются программами, выполняемыми при изменении состояния ВМ или узлов. Хуки могут выполняться как локально, так и удаленно в узле, где работает ВМ. Для настройки системы хуков необходимо установить следующие значения в конфигурационном файле OpenNebula:

- `executable` — путь исполняемого модуля драйвера хука, может быть абсолютным или относительным `/usr/lib/one/mads/`;
- `arguments` — конфигурационный файл для исполняемого драйвера, может быть абсолютным или относительным `/etc/one/`.

Пример

```
HM_MAD = [
executable = "one_hm"
]
```

8.3.6.1. Хуки виртуальной машины VM_HOOK

Хуки ВМ определяются по следующим параметрами:

- name — имя хука;
- on — условия выполнения хука:
  - CREATE — при создании ВМ;
  - PROLOG — при нахождении ВМ в состоянии PROLOG;
  - RUNNING — после успешной загрузки ВМ;
  - UNKNOWN — при нахождении ВМ в неизвестном состоянии;
  - SHUTDOWN — после отключения ВМ;
  - STOP — после остановки ВМ (включая передачу образов ВМ);
  - DONE — после удаления или отключения ВМ;
  - CUSTOM — определяемое пользователем конкретное состояние STATE и комбинация состояний LCM_STATE для запуска хука;
- command — путь может быть абсолютным или относительным /usr/share/one/hooks;
- arguments — аргументы для хука. Можно просмотреть информацию по ВМ с помощью команды $:
  - $ID — идентификатор ВМ;
  - $TEMPLATE — шаблон ВМ в формате xml с кодированием base64;
  - PREV_STATE — предыдущее состояние ВМ;
  - PREV_LCM_STATE предыдущее LCM-состояние ВМ;
- remote — удаленное выполнение. Возможные значения:
  - YES — захватчик выполняется на узле, где установлена ВМ;
  - NO — захватчик выполняется на сервере OpenNebula. Является значением по умолчанию.

8.3.6.2. Хуки узла HOST_HOOK

Хуки узла определяются по следующим параметрами:

- name — имя хука;
- on — условия выполнения хука:
  - CREATE — при создании узла (использование команды onehost create);
  - ERROR — при нахождении узла в состоянии сбоя;
  - DISABLE — после отключения узла;

- command — путь может быть абсолютным или относительным /usr/share/one/hooks;
- arguments — аргументы для хука. Можно использовать следующую информацию об узле:
  - $ID — идентификатор узла;
  - $TEMPLATE — шаблон узла в формате xml с кодированием base64;
- remote — удаленное выполнение. Возможные значения:
  - YES — захватчик выполняется на узле;
  - NO — захватчик выполняется на сервере OpenNebula. Является значением по умолчанию.

Пример

```
VM_HOOK = [
      name = "advanced_hook",
      on = "CUSTOM",
      state = "ACTIVE", lcm_state = "BOOT_UNKNOWN", command = "log.rb",
      arguments = "$ID $PREV_STATE $PREV_LCM_STATE"
]
```

8.4. Мониторинг

Подсистема мониторинга собирает следующую информацию об узлах и виртуальных машинах:

- статус узла;
- основные показатели производительности;
- статус виртуальной машины;
- потребление вычислительных ресурсов.

Сбор информации производится путем выполнения статических тестов OpenNebula. Фронтальная машина отвечает за сбор информации и ее обработку с помощью соответствующего модуля. Каждый узел периодически передает контрольные данные на фронтальную машину. Данная схема является масштабируемой и ее ограничения (количество контролируемых виртуальных машин в секунду) зависят от производительности сервера, на котором выполняется oned, и сервера БД. Подробная информация о подсистеме мониторинга приведена в разделе 12.

8.5. Узлы виртуализации

Узлы — это физические машины, на которых выполняются виртуальные машины. Подсистема виртуализации — это компонент, который отвечает за связь с гипервизором, установленным на узлах, и выполнение действий, необходимых для каждого этапа жизненного

цикла ВМ.

Узлы виртуализации имеют следующие характеристики и их рекомендованные значения:

1) центральный процессор (ЦП) — без последующих дополнительных нагрузок каждый модуль ЦП, закрепленный за одной ВМ, должен соответствовать физическому ядру ЦП в случае, если необходимо минимизировать конкуренцию ВМ за процессорные ядра. Например, при нагрузке в 40 виртуальных машин с двумя ЦП каждая для облака потребуются 80 физических ЦП. При этом 80 физических ЦП могут распределяться по различным узлам: 10 серверов с восемью ядрами каждый или пять серверов с 16 ядрами каждый. При необходимости последующих дополнительных нагрузок архитектуру ЦП можно планировать заранее с помощью элементов CPU и VCPU: CPU определяет физические ЦП, закрепленные за виртуальными машинами, а VCPU — виртуальные ЦП, передаваемые гостевой операционной системой;

2) память — по умолчанию в OpenNebula отсутствует избыточно выделяемая память. Как правило, рекомендуется всегда предусматривать резерв 10 % по ресурсам, потребляемым гипервизором. Например, для нагрузки в 45 виртуальных машин с 2 ГБ оперативной памяти каждая необходимо 90 ГБ физической памяти. Важным параметром является количество узлов, поскольку в связи с применением гипервизоров на каждый из них приходится 10 % затрат ресурсов. Например, 10 гипервизоров с 10 ГБ оперативной памяти каждый предоставят по 9 ГБ памяти, поэтому они смогут выдержать планируемую нагрузку.

По умолчанию OpenNebula поддерживает один гипервизор с открытым исходным кодом — гипервизор KVM — и настраивается на взаимодействие с узлами, на которых выполняется KVM.

По возможности конфигурация узлов должна быть однородной в части установленных программных компонентов, пользовательского управления `root`, доступности хранилища и структуры коммутации сети. Однородные узлы могут группироваться в кластеры OpenNebula.

8.5.1. Конфигурирование узла KVM

В 8.5.1.2-8.5.1.7 описаны основные действия, достаточные для конфигурирования узла и завершения развертывания OpenNebula. Дополнительная информация о настройке узла приведена в разделе 9. Конфигурирование таких подсистем, как хранилище и сеть, описано в разделах 10 и 11 соответственно.

8.5.1.1. Предварительная настройка ОС СН

Предварительная настройка ОС СН выполняется в соответствии с 8.2.2.

8.5.1.2. Установка программного обеспечения

Узел должен находиться в одном домене с фронтальной машиной. В случае, если настройка осуществляется в домене ALD, в файле `/etc/ald/ald.conf` для пользователей должны быть разрешены локальные группы `astra-admin`, `astra-console`, `kvm`, `libvirt-admin`, `libvirt-qemu`, `libvirt`.

Для настройки узла необходимо установить следующие пакеты:

```
$ sudo aptitude install ald-libvirt-qemu opennebula-node qemu-block-extra
```

Далее запустить мастер конфигурирования:

```
$ sudo ald-libvirt-qemu-configure
```

8.5.1.3. Конфигурация ssh без пароля

Фронтальная машина OpenNebula связывается с узлами гипервизора через ssh. Соответственно, необходимо настроить беспарольную авторизацию между фронтальной машиной и узлами. Для этого на фронтальной машине от имени администратора `brestadmin` выполнить команды:

```
$ KEY=$(sudo cat /root/.ssh/id_rsa.pub)
$ ssh brestadmin@<node1> "sudo bash -c \"echo $KEY >>
     /root/.ssh/authorized_keys\""
$ ssh brestadmin@<node2> "sudo bash -c \"echo $KEY >>
     /root/.ssh/authorized_keys\""
$ ssh brestadmin@<node3> "sudo bash -c \"echo $KEY >>
     /root/.ssh/authorized_keys\""
$ ...
```

8.5.1.4. Конфигурация сети

Сервисам, работающим на фронтальной машине, необходим доступ к узлам для управления гипервизорами, мониторинга, а также для перемещения файлов образов. Для этой цели рекомендуется использовать выделенную сеть. Конфигурация сети зависит от выбранной модели в соответствии с разделом 11.

Пример

При выборе сети типа «Мост» требуется установить linux-мост и подключить физическое устройство к мосту. Далее при определении сети в OpenNebula указать созданное соединение типа «Мост». Таким образом виртуальная машина будет подключена к данному мосту для обеспечения возможности связи с физическим сетевым устройством . Типовой узел с двумя сетями должен иметь два моста в соответствии с рис. 34.

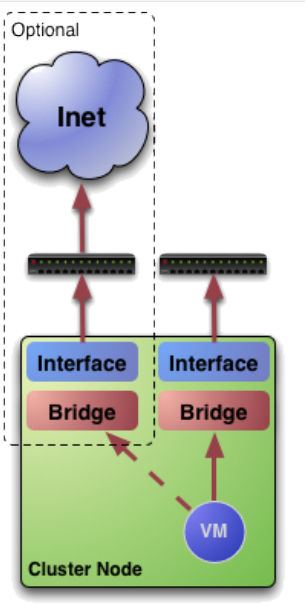

Рис. 34

Одна сеть для публичных IP-адресов, например, подключенных к eth0 NIC, другая — для частных виртуальных сетей, например, NIC eth1:

```
$ brctl show
bridge name bridge id               STP enabled interfaces
               br08000.001e682f02ac noeth0
               br18000.001e682f02ad noeth1
```

Примечание. Настройка сети необходима только на узлах. Точное имя ресурсов (`br0`, `br1` и т. д.) значения не имеет, но важно, чтобы мосты и сетевые карты NIC имели одно и то же имя на всех узлах.

8.5.1.5. Конфигурация хранилища

В конфигурации OpenNebula по умолчанию для хранения образов используется локальное хранилище фронтальной машины, а для работы виртуальных машин — локальное хранилище гипервизоров. Возможно использовать другие хранилища, например, Ceph, NFS, LVM и т. д. Подробное описание использования хранилищ приведено в разделе 10.

8.5.1.6. Добавление узла

Регистрация узла, установленного на фронтальной машине, необходима, чтобы OpenNebula могла запускать соответствующие виртуальные машины. Регистрацию можно выполнить в графическом интерфейсе пользователя Sunstone или через консольный интерфейс Command Line Interface (CLI).

Добавление узла через Sunstone

Запустить Sunstone. В меню слева перейти к списку узлов «Infrastructure — Hosts» и нажать кнопку **[+]** в соответствии с рис. 35.

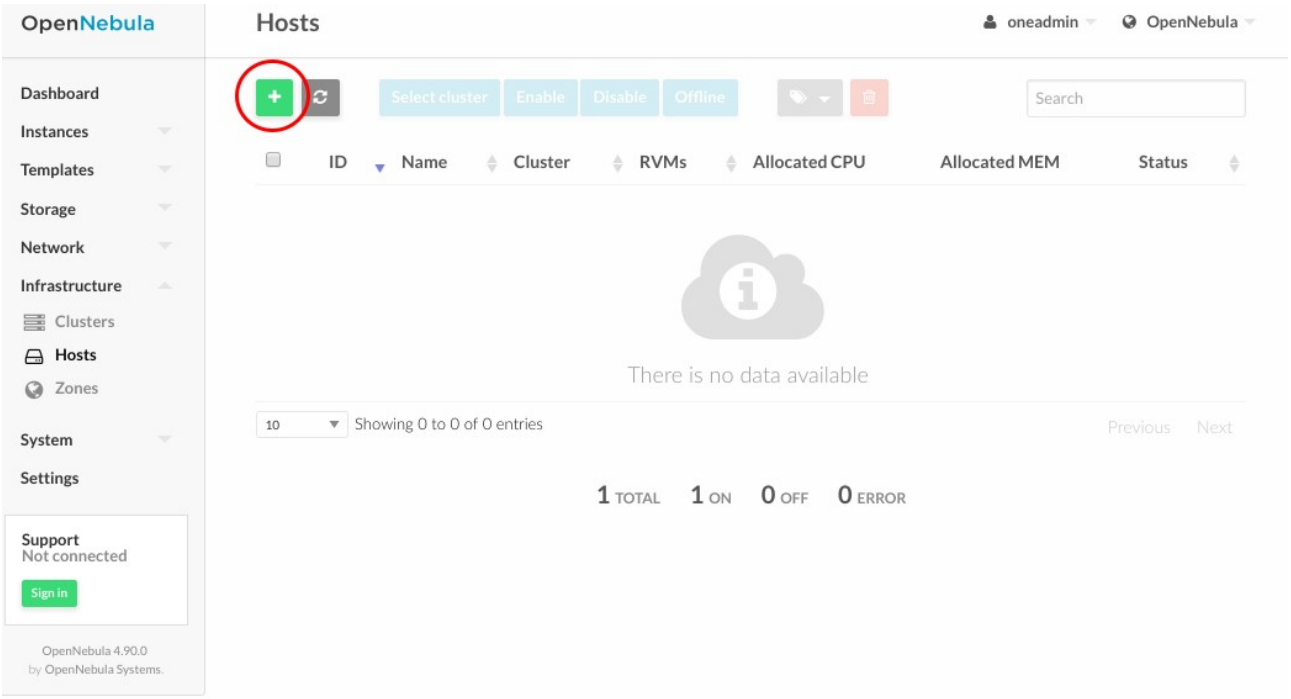

Рис. 35

Ввести имя узла в поле «Hostname», например, `node01` и нажать кнопку **[Create]** в соответствии с рис. 36.

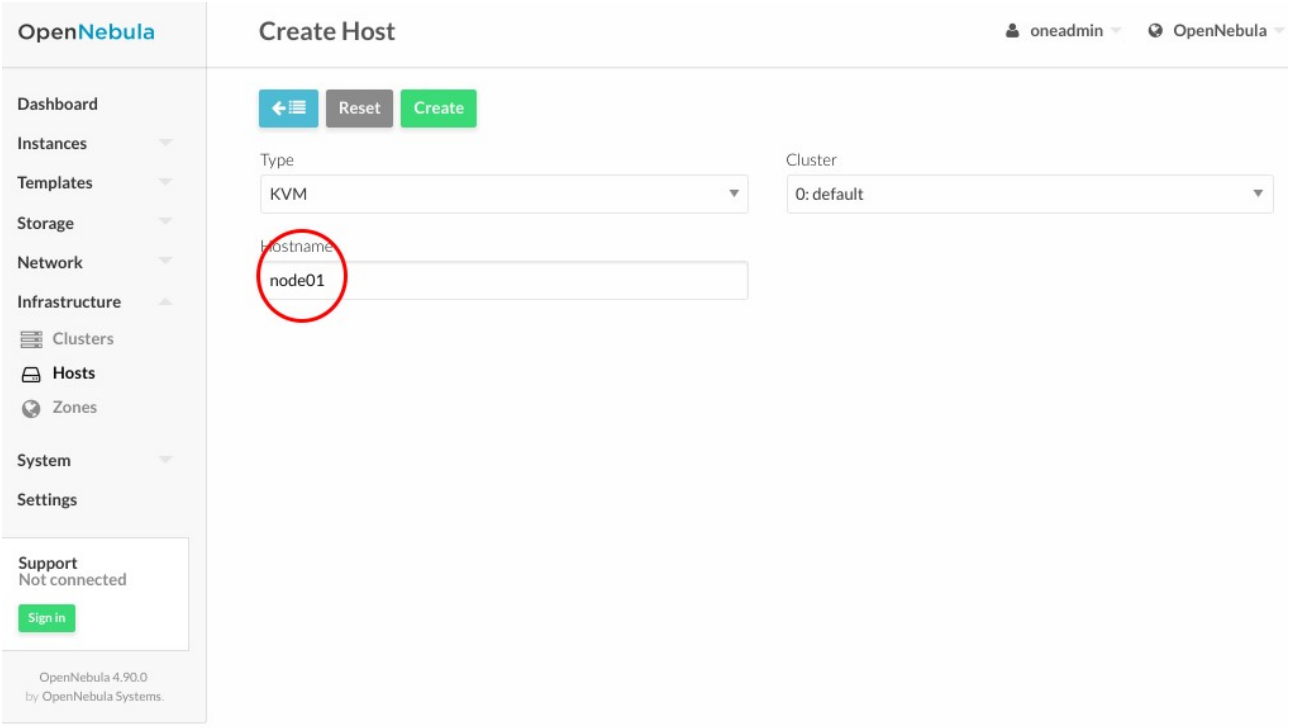

Рис. 36

Вернуться к списку узлов «Infrastructure — Hosts» и убедиться, что переключатель «Status» для добавленного узла находится в состоянии «ON». Изменение статуса узла может занять от 20 до 60 с. Для отображения актуальной информации на текущий момент требуется выполнить обновление страницы щелчком левой кнопки мыши по соответствующей кнопке

как показано на рис. 37.

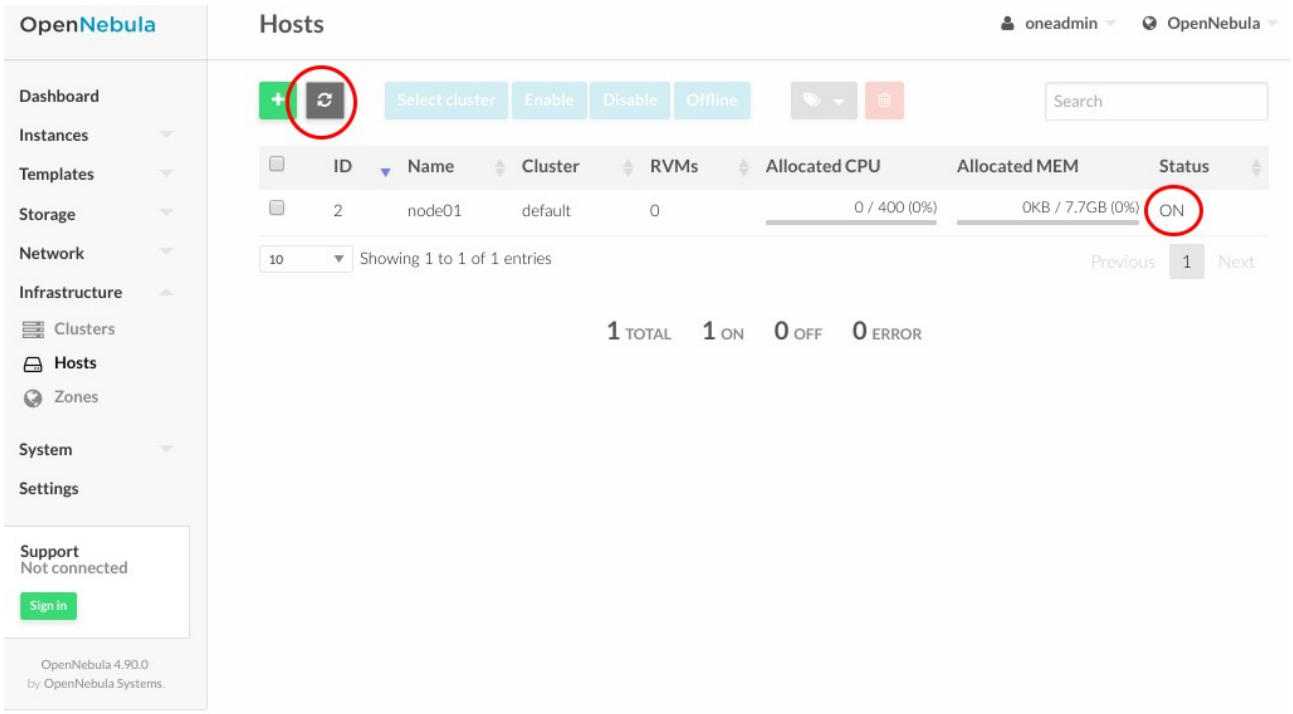

Рис. 37

Если состояние узла «ERR» вместо «ON», то необходимо проверить содержание файла `/var/log/one/oned.log`. Наиболее вероятной причиной ошибки является проблема с ssh.

Добавление узла через CLI

На фронтальной машине от имени пользователя, входящего в группы `astra-admin` и `astra-console`, выполнить следующую команду:

```
$sudo onehost create <node01> -i kvm -v kvm
```

Проверить добавленный узел с помощью команды:

```
$sudo onehost list
ID  NAME          CLUSTER    RYM    ALLOCATED_CPU    ALLOCATED_MEM    STAT
1   localhost     default0                  -    -int
```

В выводе результата выполнения команды список установленных узлов. Добавление узла займет от 20 до 60 с:

```
$sudo onehost list
ID  NAME          CLUSTER    RYM    ALLOCATED_CPU    ALLOCATED_MEM      STAT
0   node01        default0     0    0 / 400 (0 %)    OK / 7.7 G (0 %)   on
```

Убедиться, что параметр узла `STAT` находится в состоянии `on`. Если указано состояние узла `err`, то необходимо проверить содержание файла `/var/log/one/oned.log`. Наиболее вероятной причиной ошибки является проблема с ssh.

8.5.1.7. Импорт виртуальных машин

Импорт работающих в текущий момент виртуальных машин можно выполнять в любое время. Функцию также возможно использовать для просмотра ранее развернутых в OpenNebula виртуальных машин.

8.6. Хранилище данных

Для хранения образов дисков виртуальных машин OpenNebula использует хранилища данных. Хранилище данных — это любой носитель данных с поддержкой серверов SAN/NAS. К каждому хранилищу данных должен обеспечиваться доступ через фронтальную машину с помощью технологий SAN/NAS либо через систему хранения данных с прямым подключением.

При развертывании виртуальной машины ее образы передаются из хранилища данных на узлы. В зависимости от используемой технологии хранения данных образы передаются путем копирования, по символьной ссылке или путем установки тома LVM (управление логическими томами).

Хранилище данных OpenNebula подразделяется на три типа хранилищ:

- хранилище образов (Images Datastore) — используется для хранения всех зарегистрированных образов, которые могут использоваться для создания ВМ, описание данного типа хранилища приведено в 10.1;
- системное хранилище (System Datastore) — используется для хранения дисков виртуальных машин, работающих в текущий момент. Образы дисков перемещаются, или клонируются, в хранилище образов или из него при развертывании и отключении виртуальных машин, при подсоединении или фиксировании мгновенного состояния дисков;
- хранилище файлов и ядер (Files & Kernels Datastore) — используется для хранения обычных файлов. Файлы могут использоваться как ядра kernels, псевдодиски ramdisks или файлы контекста. Описание данного типа хранилища приведено в 10.2.

Одним из основных способом управления хранилищем данных является ограничение хранилища, доступного для пользователей, путем определения квот по максимальному количеству виртуальных машин, а также максимального объема энергозависимой памяти, который может запросить пользователь, и обеспечения достаточного пространства хранения системных данных и образов, отвечающего предельным установленным квотам. При необходимости OpenNebula позволяет администратору добавлять хранилища системных данных и образов.

Примечание. По умолчанию хранилища данных и система настроены на использование файловой системы с драйверами передачи ssh.

8.7. Организация сети

Для обеспечения надежности облачной инфраструктуры во фронтальной машине (публичной и сервисной) должны быть установлены две или три сетевые платы (в зависимости от серверной части может потребоваться доступ к сети хранения данных).

В каждом узле виртуализации в зависимости от конфигурации хранилища и сети должно быть установлено до четырех сетевых плат: для приватной, публичной, сервисной сетей и сети хранения данных.

OpenNebula обеспечивает адаптируемую и настраиваемую подсистему сети для интеграции конкретных сетевых требований существующих ЦОХД. Так же необходимо наличие минимум двух различных физических сетей:

- сервисная сеть (Service Network) — используется сервисами фронтальной машины OpenNebula для обеспечения доступа к узлам с целью управления и мониторинга гипервизоров и перемещения файлов образов;
- сеть экземпляров (Instance Network) — обеспечивает возможность сетевого подключения к виртуальным машинам через различные узлы. Для эффективного развертывания виртуальных машин, следует создать одну или несколько физических сетей.

Администратор OpenNebula может соотносить один из следующих драйверов с каждым узлом:

- dummy — используется по умолчанию, не выполняет сетевых операций, правила межсетевого экранирования игнорируются;
- fw — правила межсетевого экранирования применяются, но изоляция сетевого соединения игнорируется;
- 802.1Q — ограничивает сетевой доступ через VLAN-тегирование, для чего требуется поддержка аппаратных переключателей;
- ebtables — ограничивает сетевой доступ по правилам Ebtables. Особая конфигурация аппаратных средств не требуется;
- ovswitch — ограничивает сетевой доступ с помощью программного коммутатора Open vSwitch (OVS);
- vxlan — сегментирует VLAN в изолированные сети с помощью протокола инкапсуляции VXLAN.

Порядок настройки сети описан в разделе 11.

8.8. Аутентификация

Для обеспечения доступа к OpenNebula поддерживаются доменная аутентификация.

8.9. Отказоустойчивость виртуальной машины

В OpenNebula интегрирован механизм (модуль) обеспечения отказоустойчивости ВМ с сохранением мандатных и дискретных атрибутов безопасности.

Для работы модуля необходимо соблюдение следующих условий:

- наличие в кластере минимум двух рабочих узлов виртуализации;
- общее хранилище дисков ВМ, доступное на каждом из узлов;
- поддержка со стороны узлов технологии IPMI (Intelligent Platform Management Interface). Данные для авторизации по IPMI должны быть указаны в настройках каждого узла виртуализации в соответствии с рис. 38.

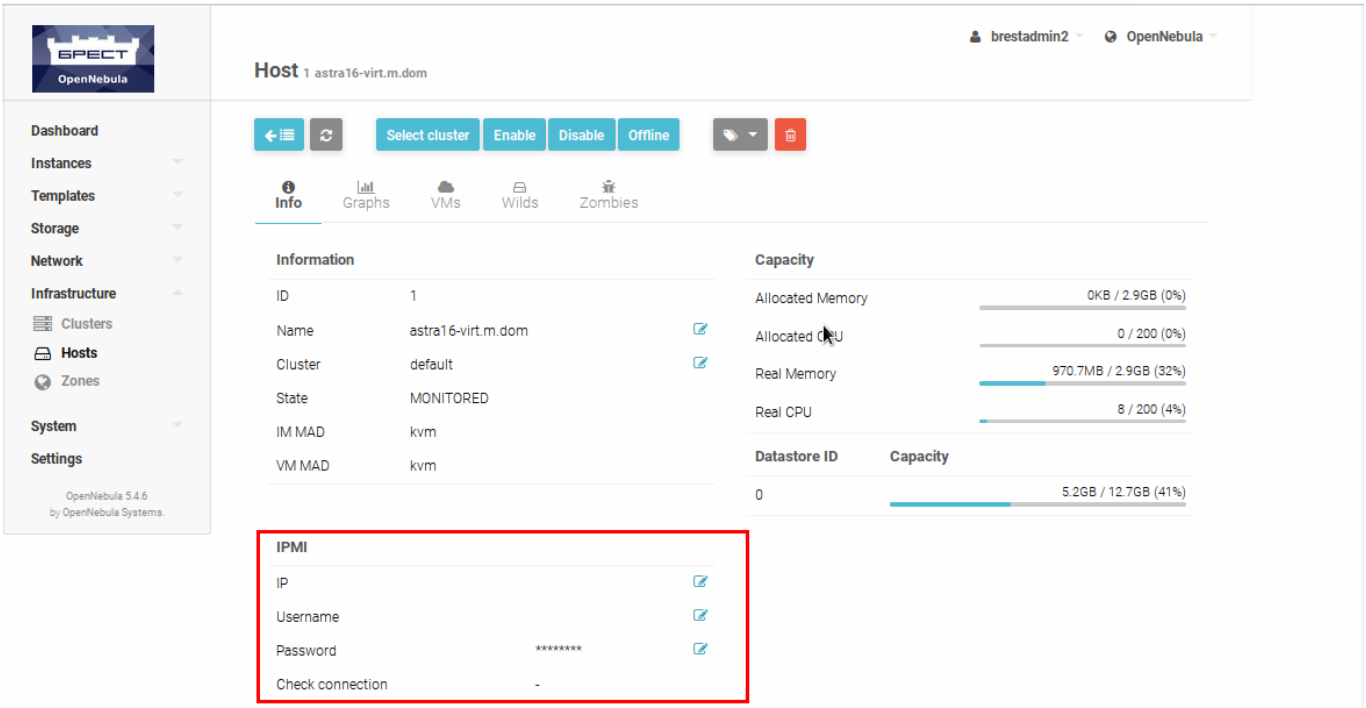

Рис. 38

- для ВМ должен быть установлен признак отказоустойчивости (High Availability) в соответствии с рис. 39.

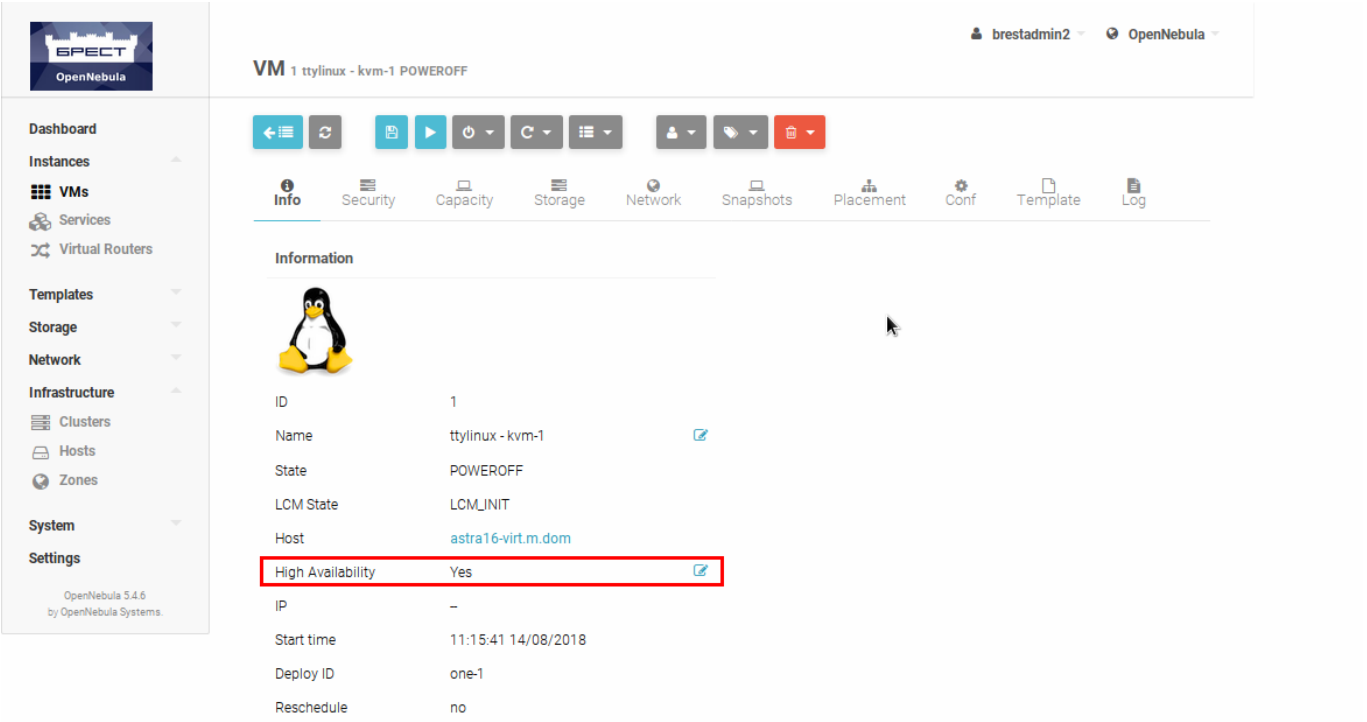

Рис. 39

В процессе работы модуля:

- OpenNebula получает сигнал о неисправности узла виртуализации;
- считывается список запущенных на неисправном узле ВМ с установленным признаком отказоустойчивости (High Availability);
- считываются сохранившиеся атрибуты безопасности ВМ (мандатная метка и имя пользователя, запустившего виртуальную машину) и производится ее перезапуск на рабочем узле.

8.9.1. Автостарт виртуальных машин

Механизм автостарта ВМ позволяет автоматически запустить ВМ, настроенные на автостарт, на узлах при восстановлении облака после аварийного отключения.

Для включения автостарта на ВМ следует в свойствах данной ВМ во вкладке ?Общее? перевести настройку «Autostart» в состояние «On».

Автостарт происходит следующим образом:

1) при восстановлении работы облака фронтальная машина запускается в первую очередь;

2) фронтальная машина проверяет состояние узлов, ожидая их запуска;

3) фронтальная машина проверяет наличие на запущенных узлах ВМ, для которых настроен автостарт;

4) если такие ВМ присутствуют, фронтальная машина автоматически запускает их.

Если используется алгоритм Raft (описанный в 8.2.4), то для корректной работы автостарта необходимо в файл `/lib/systemd/system/libvirtd.service`, находящийся

на фронтальной машине, добавить строку:

```
After=opennebula.service
```

8.10. Дополнительные компоненты

После установки и запуска облака OpenNebula можно установить следующие дополнительные компоненты:

- приложения Multi-VM и Auto-scaling (автомасштабирование): OneFlow позволяет пользователям и администраторам определять, выполнять и осуществлять управление многоуровневыми приложениями или сервисами, состоящими из взаимосвязанных в части развертывания виртуальных машин. В части развертывания и управления каждая группа виртуальных машин является единым объектом, полностью интегрированным в дополнительное управление пользователями и группами OpenNebula;
- Application Insight (Анализатор приложений): OneGate позволяет гостевым пользователям виртуальных машин отправлять контрольную информацию в OpenNebula. Пользователи и администраторы могут использовать его для сбора показателей, определения проблем в своих приложениях и активации правил автомасштабирования OneFlow.

8.11. Управление пользователями

Для управления пользователями на основном узле OpenNebula используется инструмент командной строки `brestuser`.

ВНИМАНИЕ! При работе с данным инструментом командной строки необходимо, чтобы все узлы кластера OpenNebula были в режиме online. Все вызовы должны осуществляться пользователем, обладающим правами администратора.

Для просмотра списка доступных команд инструмента выполнить:

```
$ sudo brestuser help
```

Перечень основных команд инструмента `brestuser` приведен в таблице 8.

Таблица 8

Команда	Описание
`create <имя_пользователя>`	Создание пользователя. Команда должна выполняться от имени администратора
`passwd <имя_пользователя>`	Изменение пароля существующего пользователя
`del <имя_пользователя>`	Удаление существующего пользователя из системы (в том числе из контроллера домена)
`list`	Просмотр списка зарегистрированных пользователей OpenNebula

Окончание таблицы 8

Команда	Описание
`sync`	Синхронизация пользователей с узлами кластера. **ВНИМАНИЕ!** Команду следует выполнять каждый раз после добавления нового узла в кластер для предоставления существующим пользователям доступа к новому узлу. По умолчанию существующие пользователи не имеют доступа к добавляемым узлам

Если домен работает на основе FreeIPA, управление доменными пользователями и группами можно осуществлять через пункт меню «System» графического интерфейса OpenNebula.

Автостарт ВМ возможно производить от имени служебного пользователя. Для этого следует в свойствах ВМ во вкладке «Общее» включить настройку «Service VM».

9. ОСОБЕННОСТИ НАСТРОЙКИ УЗЛА

9.1. Требования

По возможности необходимо использовать virtio для сети (в случае, если не используется мандатное управление доступом) и дисков. Информация о virtio приведена в 9.3.3. Применение эмулированных аппаратных средств для сети и дисков оказывает влияние на производительность и не раскрывает все имеющиеся функциональные возможности. Например, если не используется virtio для драйверов диска, невозможно превысить ограничения по небольшому количеству дисков, подсоединенных к контроллеру, и оно не действует, пока работает ВМ (подключение диска без выключения).

9.2. Настройка

9.2.1. Настройка KVM

Пакеты OpenNebula выполняют настройку KVM автоматически.

9.2.2. Драйверы

Драйвер KVM в OpenNebula подключен по умолчанию.

9.2.3. Динамическое перемещение для других параметров кэширования

В случае использования дисков с установкой кэша, отличной от «none», могут возникнуть трудности с динамическим перемещением в зависимости от версии libvirt. Перемещение можно активировать, добавив параметр `--unsafe` к команде `virsh`. Необходимо в файле `/var/lib/one/remotes/vmm/kvm/kvmrc` раскомментировать строку `MIGRATE_OPTIONS=--unsafe` и затем от имени администратора выполнить команду:

```
sudo onehost sync --force
```

9.3. Использование

9.3.1. Специальные опции KVM

В таблице 9 приведено описание основных параметров шаблонов, характерных для KVM.

Таблица 9

Параметр	Описание
Устройство `DISK`	
`TYPE`	Определяет тип носителя, выделяемого для ВМ. Возможные значения: `disk` (по умолчанию), `cdrom` или `floppy`. Соответствует значению `media` параметра `-driver` команды `kvm`
`DRIVER`	Определяет формат образа диска. Возможные значения: `raw`, `qcow2`. Соответствует значению `format` параметра `-driver` команды `kvm`

Окончание таблицы 9

Параметр	Описание
CACHE	Указывает опциональный механизм кэширования. Возможные значения: default, none, writethrough и writeback
IO	Устанавливает политику ввода/вывода. Возможные значения: threads и native
Устройство NIC	
TARGET	Имя tun устройства, созданного для ВМ. Соответствует значению ifname параметра –net команды kvm
SCRIPT	Имя скрипта оболочки, который должен выполняться после создания tun устройства для ВМ. Соответствует значению script параметра –net команды kvm
MODEL	Аппаратные средства Ethernet, которые необходимо эмулировать. Список доступных моделей можно получить командой $ kvm –net nic,model=? –nographic /dev/null
FILTER	Определяет правило фильтрации сети для интерфейса. Libvirt включает в себя некоторые предварительно заданные правила, например, очистки трафика, которые могут использоваться. Список правил в системе можно просмотреть командой $ virsh –c qemu:///system nwfilter-list

9.3.2. Графическая подсистема

Libvirt и KVM при корректной настройке могут работать с протоколом SPICE. Для его выбора необходимо добавить в раздел GRAPHICS параметр:

```
TYPE = SPICE
```

Использование SPICE обеспечивает также подачу драйвером конкретной конфигурации для данных машин. Настройки можно изменить в конфигурационном файле драйвера (переменная SPICE_OPTIONS).

9.3.3. Virtio

Virtio является фреймворком виртуализации ввода-вывода в KVM. Для «гостя» потребуется ядро linux с драйверами virtio. Для использования драйвера virtio необходимо добавить к следующим устройствам соответствующие параметры и их значения:

- DISK — добавить параметр DEV_PREFIX="vd";
- NIC — добавить параметр MODE="virtio".

9.3.4. Горячее подключение/отключение Disk/NIC

KVM поддерживает горячее подключение к virtio и шинам SCSI. Способ подключения диска определяется значением параметра DEV_PREFIX шаблона диска, например:

- DEV_PREFIX=vd — горячее подключение к virtio (рекомендуется);
- DEV_PREFIX=sd — горячее подключение к SCSI (по умолчанию).

При передаче параметра TARGET вместо DEV_PREFIX применяются те же правила (т.е. OpenNebula создает TARGET на основании DEV_PREFIX, если TARGET не предоставляется).

Настройки для типа кэша по умолчанию для вновь подключенных дисков выполняются в файле `/var/lib/one/remotes/vmm/kvm/kvmrc`:

```
DEFAULT_ATTACH_CACHE=none
```

В отношении дисков и NIC, если гостевой ОС виртуальной машины является ОС семейства Linux, гостю необходимо однозначно указать на необходимость повторного просмотра шины PCI. Это можно сделать путем выполнения следующей команды с правами администратора:

```
# echo 1 > /sys/bus/pci/rescan
```

9.3.5. Подключение QEMU Guest Agent

QEMU Guest Agent обеспечивает возможность взаимодействия с гостевой ОС. Для отправки и получения команд данный агент использует последовательное соединение virtio. Он позволяет зафиксировать файловую систему до выполнения снимка, при этом в снимке не будет большой части записанных данных. Фиксация файловой системы возможна только с драйверами хранилищ Ceph и qcow2. Необходимым условием использования агента является установка пакета `qemu-guest-agent` внутри гостевой ОС.

9.3.6. Импорт виртуальной машины

Виртуальная машина, работающая на гипервизоре KVM, которая не была запущена через OpenNebula, может быть импортирована в OpenNebula. При этом в ходе импорта будет создан клон ВМ, а оригинал останется неизменным.

9.3.7. Сервисный режим узла

В случае необходимости проведения технического обслуживания или других работ на узле рекомендуется перевести данный узел в сервисный режим.

Для перевода узла в сервисный режим следует в свойствах узла выставить для настройки «Resched VMs when host disable» значение «Yes», затем отключить узел нажатием на кнопку **[DISABLE]**. Перед отключением узла размещенные на нем ВМ будут мигрированы на другие доступные узлы, что позволит им продолжить работу.

9.4. Управление хостами по IPMI

Утилита `ipmitool` служит для управления хостами по протоколу IPMI. В большинстве случаев вызов `ipmitool` имеет следующий формат:

```
ipmitool [<опции>] <команда>
```

Пример

```
ipmitool -I lanplus -H 192.168.17.12 -U USER -P PASSWORD  chassis power off
ipmitool -I lanplus -H 192.168.17.12 -U USER -P PASSWORD  chassis power on
ipmitool -I lanplus -H 192.168.17.12 -U USER -P PASSWORD  chassis power reset
```

где `192.168.17.12` — IP-адрес IPMI-интерфейса;

`USER` — пользователь;

`PASSWORD` — пароль.

Указанные команды позволяют выключить, включить, и перезагрузить сервер соответственно.

Более подробное описание утилиты приведено в man ipmitool.

9.5. Перенаправление PCI-устройств

9.5.1. Требования

Узел, который предполагается использовать для виртуализации, должен поддерживать I/O MMU (Intel VT-d, AMD-Vi).

9.5.2. Настройка гипервизора

9.5.2.1. Конфигурация ядра

Конфигурация ядра должна выполняться с учетом необходимости поддержки I/O MMU и блокировки любых драйверов, которые могут осуществлять доступ к устройствам PCI, предполагаемым для использования в виртуальных машинах. Параметр для подключения I/O MMU:

```
intel_iommu=on
```

Необходимо также разрешить ядру загружать драйвер vfio-pci и блокировать драйверы для выбранных карт. Например, для графической карты Nvidia можно применять следующие параметры:

```
rd.driver.pre=vfio-pci rd.driver.blacklist=nouveau
```

9.5.2.2. Загрузка драйвера vfio в initrd

Следующие модули для vfio должны быть добавлены в initrd:

- `vfio`;
- `vfio_iommu_type1`;
- `vfio_pci`;
- `vfio_virqfd`.

Например, если система использует `dracut`, добавить в файл `/etc/dracut.conf.d/local.conf` строку:

```
add_drivers+="vfio vfio_iommu_type1 vfio_pci vfio_virqfd"
```

и сгенерировать initrd:

```
# dracut --force
```

9.5.2.3. Блокировка драйверов

Блокировка, которая определяется в параметрах ядра, должна быть определена и в настройках системы.

Пример

Файл /etc/modprobe.d/blacklist.conf для графической карты Nvidia

```
blacklist nouveau blacklist lbm-nouveau options nouveau modeset=0 alias nouveau
off
alias lbm-nouveau off
```

Данный конфигурационный драйвер vfio должен загружаться передачей id карт PCI, которые предполагается подключить к ВМ. Например, для графической карты Nvidia Grid K2 передается id 10de:11bf в файле /etc/modprobe.d/local.conf:

```
options vfio-pci ids=10de:11bf
```

9.5.2.4. Привязка устройств к vfio

I/O MMU разделяет карты PCI на группы для изолирования работы памяти между устройствами и ВМ. Для добавления карт в vfio и назначения им группы можно использовать совместно выполняемые скрипты.

Пример

Скрипт привязывает карту к vfio, прописывается в файле /usr/local/bin/vfio-bind

```
#!/bin/sh
modprobe vfio-pci
for dev in "$@"; do
vendor=$(cat /sys/bus/pci/devices/$dev/vendor) device=$(cat
/sys/bus/pci/devices/$dev/device)
if [ -e /sys/bus/pci/devices/\$dev/driver ]; then
echo $dev > /sys/bus/pci/devices/$dev/driver/unbind
fi
echo $vendor $device > /sys/bus/pci/drivers/vfio-pci/new_id
done
```

Этот скрипт необходимо добавить в автозапуск при загрузке системы.

Конфигурация прописывается в файле /etc/sysconfig/vfio-bind. Карты указываются с PCI-адресами. Адреса можно получить командой lspci, добавив в начало домен, как правило, 0000.

Пример

```
DEVICES="0000:04:00.0 0000:05:00.0 0000:84:00.0 0000:85:00.0"
```

9.5.2.5. Конфигурация QEMU

После привязки устройств к vfio необходимо предоставить QEMU-доступ к данным устройствам для групп, назначенных картам PCI. Список карт PCI и их I/O MMU группу можно получить с помощью команды:

```
# find /sys/kernel/iommu_groups/ -type l
```

Пример

Для карт с группами 45, 46, 58 и 59 в файл `/etc/libvirt/qemu.conf` добавить конфигурацию:

```
cgroup_device_acl = [
"/dev/null", "/dev/full", "/dev/zero", "/dev/random", "/dev/urandom",
"/dev/ptmx", "/dev/kvm", "/dev/kqemu", "/dev/rtc","/dev/hpet",
"/dev/vfio/vfio", "/dev/vfio/45", "/dev/vfio/46", "/dev/vfio/58",
"/dev/vfio/59"
]
```

9.5.3. Настройка драйвера

Единственной необходимой настройкой является настройка фильтра для теста мониторинга, который получает список карт PCI. По умолчанию тест перечисляет все карты, имеющиеся в узле. Для изменения данного списка необходимо изменить настройки фильтра в файле `/var/lib/one/remotes/im/kvm-probes.d/pci.rb` и установить список с таким же форматом lspci:

```
# Данная функция содержит фильтры для мониторинга карт PCI. Формат
# такой же, как lspci, и можно добавить несколько фильтров через запятые.
# Нулевой фильтр обеспечит извлечение всех карт PCI.
#
# Из раздела помощи lspci:
# -d [<vendor>]:[<device>][:<class>]
#
# Например
#
# FILTER = '::0300' # все карты VGA
# FILTER = '10de::0300' # все карты NVIDIA VGA
# FILTER = '10de:11bf:0300' # только GK104GL [GRID K2]
# FILTER = '8086::0300,::0106' # все карты Intel VGA и любые контроллеры SATA
```

9.5.4. Настройка использования устройств PCI

Основным назначением является просмотр информации узла в CLI и в Sunstone, обнаружение доступных устройств PCI и добавление желаемого устройства в шаблон ВМ. Устройства PCI можно добавлять указанием значений `vendor` (продавец), `device` (устройство) и `class` (класс). OpenNebula будет развертывать ВМ только на узле с имеющимся устройством PCI. Если таких узлов нет, в журнале планировщика появится сообщение об ошибке.

9.5.4.1. Настройка в CLI

Для просмотра списка устройств PCI узла необходимо выполнить команду:

```
onehost show <номер_узла>
```

Результат выполнения команды для узла с номером `0` приведен на рис. 40.

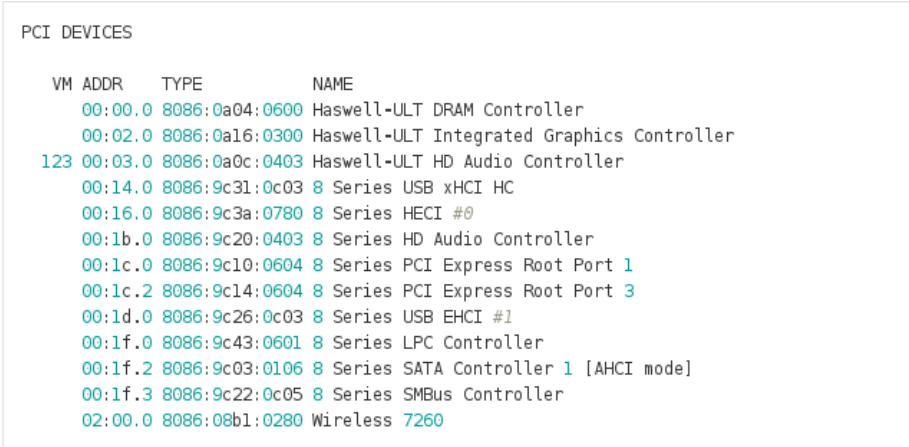

Рис. 40

В соответствии с рис. 40:

- `VM` — идентификационный номер ВМ, использующей данное устройство. Не указывается, если это устройство не используется ни одной ВМ;
- `ADDR` — адрес на шине PCI;
- `TYPE` — значения описания устройства. Это `vendor:device:class`. Данные значения используются при выборе устройства PCI для подключения;
- `NAME` — имя устройства PCI.

Для обеспечения использования одного из устройств PCI в ВМ должна быть добавлена новая функция, с помощью которой производится выбор устройства для использования.

Пример

Запрос `Haswell-ULT HD Audio Controller`

```
PCI = [
VENDOR = "8086", DEVICE = "0a0c", CLASS = "0403"
]
```

Устройство может быть также указано без типовых значений.

Пример

Получение любых портов PCI Express Root Ports

```
PCI = [
CLASS = "0604"
]
```

Для подключения более одного устройства PCI к ВМ необходимо добавить дополнительные параметры `PCI`.

9.5.4.2. Настройка в Sunstone

В Sunstone информация о доступных устройствах PCI узла отображается во вкладке «PCI» в соответствии с рис. 41.

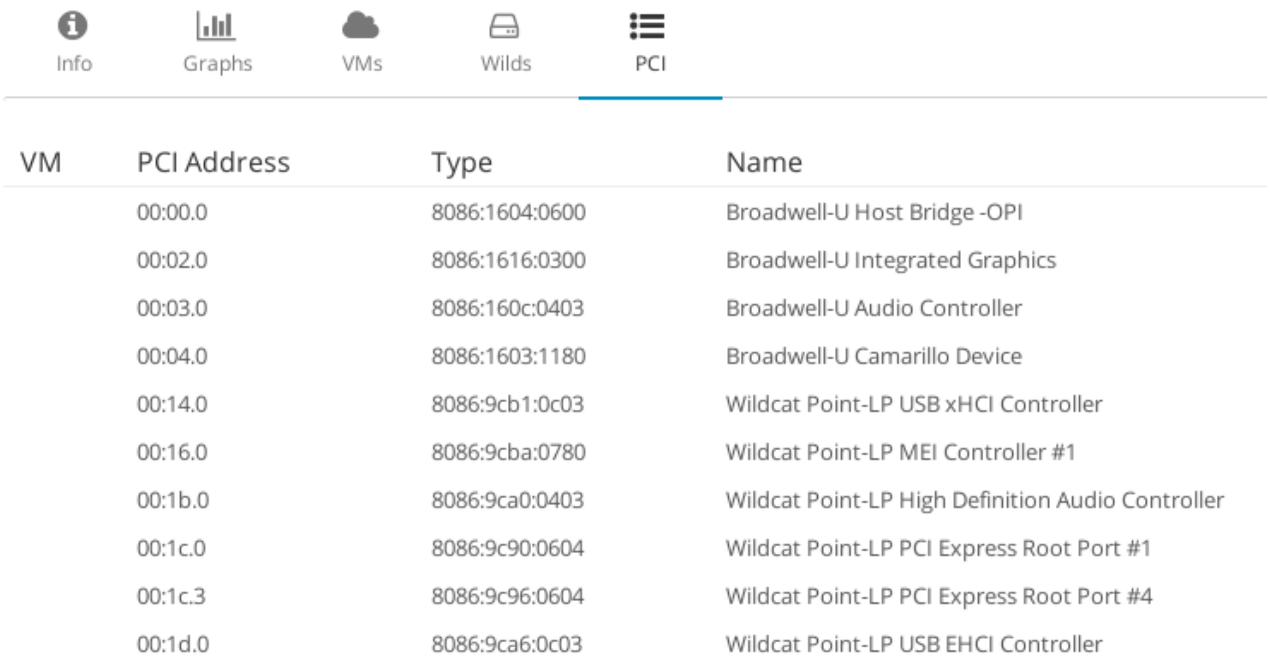

Рис. 41

Для добавления устройства PCI в шаблон ВМ перейти во вкладку «Other» и заполнить приведенную на рис. 42 форму для соответствующего устройства PCI.

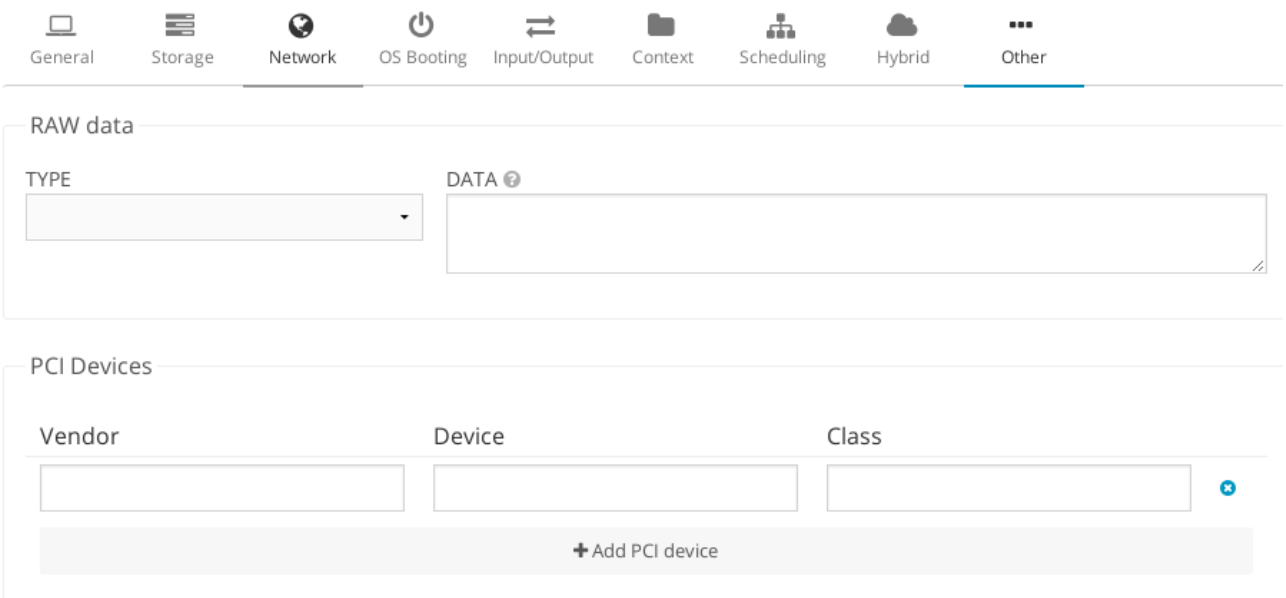

Рис. 42

9.5.5. Использование устройств PCI в качестве сетевых интерфейсов

Устройство PCI можно использовать как сетевой интерфейс непосредственно в OpenNebula. Для этого необходимо в настройках ВМ через интерфейс Sunstone изменить

драйвер устройства. После изменения драйвера устройства при определении сети данный драйвер будет использоваться для плат сетевого интерфейса PCI passthrough. При этом, если используется VLAN, рекомендуется указывать фиктивный (dummy) сетевой драйвер или 802.1Q. Необходимо указать любое произвольное значение в поле BRIDGE, и оно будет игнорироваться. Для 802.1Q поле PHYDEV можно оставить пустым.

Пакеты context поддерживают настройку следующих опций:

- MAC — изменяет mac-адрес соответствующего сетевого интерфейса на MAC, назначенный OpenNebula;
- IP — назначает IPv4-адрес интерфейсу при условии использования маски подсети /24;
- IPV6 — назначает IPv6-адрес интерфейсу при условии использования маски подсети /128;
- VLAN_ID — при наличии будет создавать тегированный интерфейс и присваивать IP тегированному интерфейсу.

9.5.5.1. Настройка через CLI

Если раздел PCI в шаблоне ВМ содержит параметр TYPE="NIC", данное устройство будет рассматривается в качестве сетевого, и OpenNebula присвоит ему MAC-адрес, VLAN_ID, IP и т.д.

Пример

Раздел PCI интерфейса, принимаемого за NIC

```
PCI=[
NETWORK="passthrough", NETWORK_UNAME="brestadmins", TYPE="NIC",
CLASS="0200", DEVICE="10d3", VENDOR="8086" ]
```

9.5.5.2. Настройка через Sunstone

Во вкладке «Network» в пункте дополнительных параметров «Advanced options» найти раздел «Hardware», установить флаг «PCI Passthrough» и заполнить поле PCI-адреса. Выбрать сетевое устройство из выпадающего списка. Значения «Vendor», «Device», «Class» для выбранного устройства приведены в свойствах узла — вкладка «PCI», столбец «Type» (см. рис. 41). Эти значения следует вручную ввести в соответствующие поля, как показано на рис. 43.

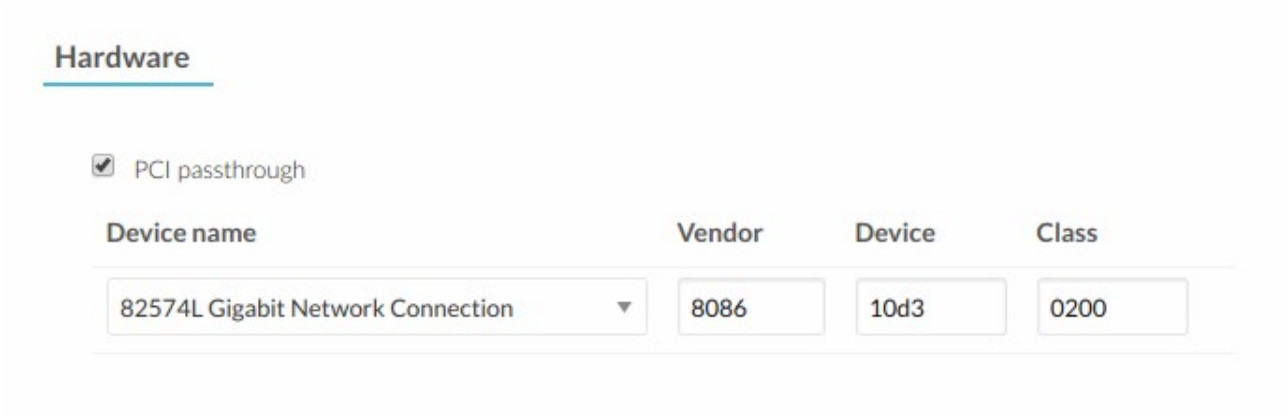
Hardware
PCI passthrough
Device name
Vendor
Device
Class
82574L Gigabit Network Connection
8086
10d3
0200

Рис. 43

10. УСТАНОВКА ОТКРЫТОГО ОБЛАЧНОГО ХРАНИЛИЩА

10.1. Хранилище образов

В зависимости от базовой технологии хранения различают следующие типы хранилищ образов:

- Filesystem — хранилище данных файловой системы для хранения образов в форме файла;
- LVM — хранение образов в логических томах LVM (менеджер логических томов);
- Ceph — хранение образов с помощью блочных устройств Ceph;
- Raw Device Mapping — прямое подключение к ВМ существующих блочных устройств в узлах;
- iSCSI-Libvirt Datastore — доступ к устройствам iSCSI через встроенную поддержку QEMU.

Образы дисков передаются между хранилищем образов и системным хранилищем с помощью драйверов программного обеспечения (ПО) Transfer Manager (TM). Эти драйверы представляют собой специальные элементы ПО, которые выполняют низкоуровневые операции хранения. В таблице 10 приведено описание доступных методов передачи данных для хранилища каждого типа.

Таблица 10

Хранилище	Методы передачи данных между хранилищем образов и системным хранилищем
Filesystem	shared — образы экспортируются в совместно используемую файловую систему; sh — образы копируются с помощью ssh-протокола; qcow2 — аналогично shared, но для файлов формата `qcow2`
Ceph	ceph — все образы экспортируются в Ceph-пулы; shared — энергозависимые (volatile) и контекстные (context) диски, экспортируемые в совместно используемую файловую систему (shared FS)
LVM	fs_lvm — образы, экспортируемые в совместно используемую файловую систему, но разгружаемые в LV
Raw Devices	dev — образы представляют собой существующие блочные устройства в узлах
iSCSI libvirt	iscsi — образы представляют собой компоненты iSCSI target

10.1.1. Хранилище данных файловой системы

Хранилище данных файловой системы позволяет хранить образы ВМ в виде файла. Рекомендуется иметь несколько хранилищ данных файловой системы для:

- распределения операций ввода-вывода между серверами хранения данных;
- использования различных хранилищ для различных узлов кластера;

- применения различных режимов передачи данных в различные образы;
- применения различных SLA-политик, например, резервирования, к различным типам ВМ или пользователям;
- добавления нового хранилища к облаку.

10.1.1.1. Схема хранилища данных

Образы сохраняются в соответствующий каталог хранилища (`/var/lib/one/datastores/<datastore_id>`). Для каждой рабочей ВМ существует каталог с названием по идентификационному номеру ВМ в соответствующем системном хранилище. В данных каталогах содержатся диски ВМ и дополнительные файлы, например, файлы контрольных точек или файлы снимков.

Примечание. Стандартный путь для `/var/lib/one/datastores` можно изменить в файле `oned.conf` через опцию настройки `DATASTORE_LOCATION`.

Режимы передачи Shared и Qcow2

Драйвер совместной передачи (shared transfer driver) предполагает, что хранилище установлено на всех узлах кластера. Как правило, это достигается с помощью распределенной файловой системы (distributed FS), такой как NFS.

При создании ВМ ее диски (файлы `disk.i`) копируются в соответствующий каталог системного хранилища. Данные файловые операции всегда выполняются удаленно на узле назначения (см. рис. 44).

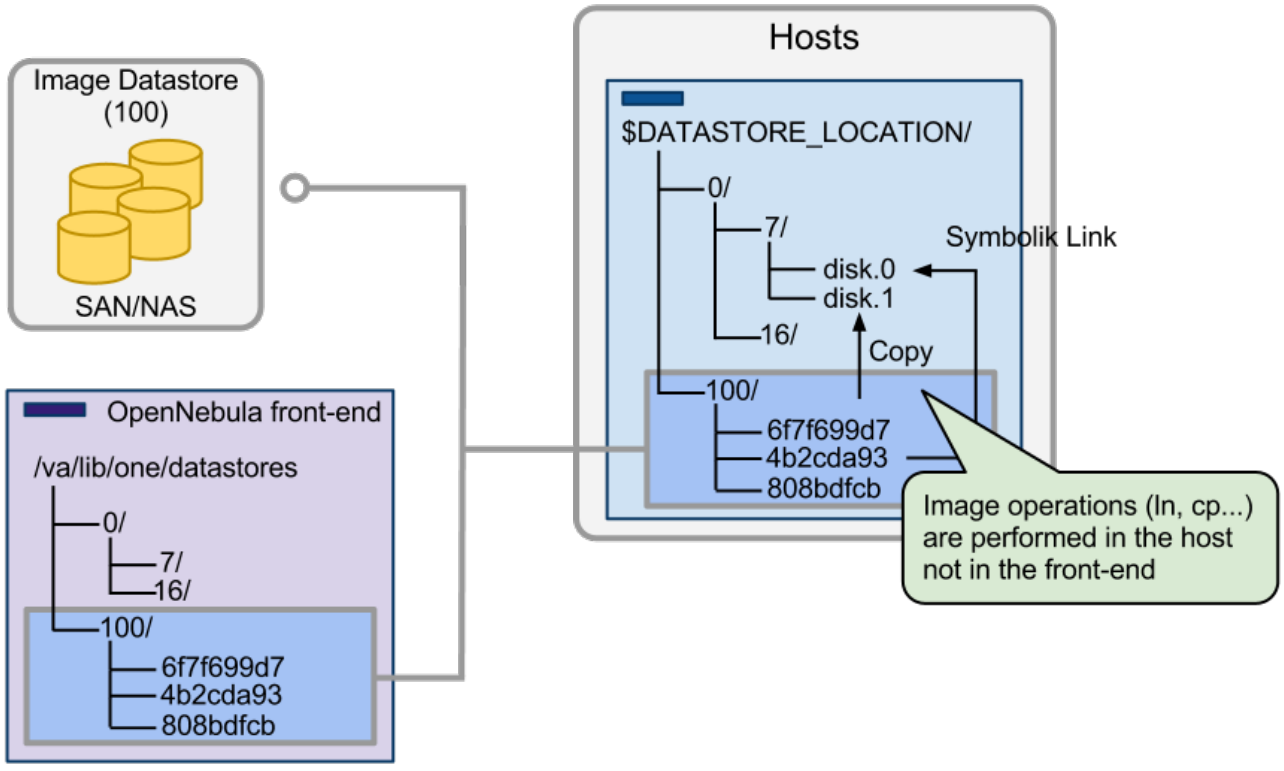

Рис. 44

Данный метод передачи сокращает время развертывания ВМ и обеспечивает воз-

можность динамического перемещения. Однако возможно снижение производительности виртуальных машин, если виртуализированные службы оказывают интенсивные нагрузки на диск. Это ограничение можно преодолеть путем:

- использования серверов с различными файловыми системами для хранилищ образов с распределением фактической пропускной способности подсистемы ввода-вывода;
- альтернативного использования ssh-системного хранилища, при котором образы копируются локально на каждый узел;
- настройки или улучшения серверов файловых систем.

Режим передачи SSH

При данном режиме передачи системное хранилище распределяется между узлами. Драйвер передачи ssh использует локальное хранилище узлов для размещения образов работающих виртуальных машин. Впоследствии все операции выполняются локально, но образы всегда необходимо копировать на узлы. Данный драйвер также предотвращает использование динамических перемещений между узлами (см. рис. 45).

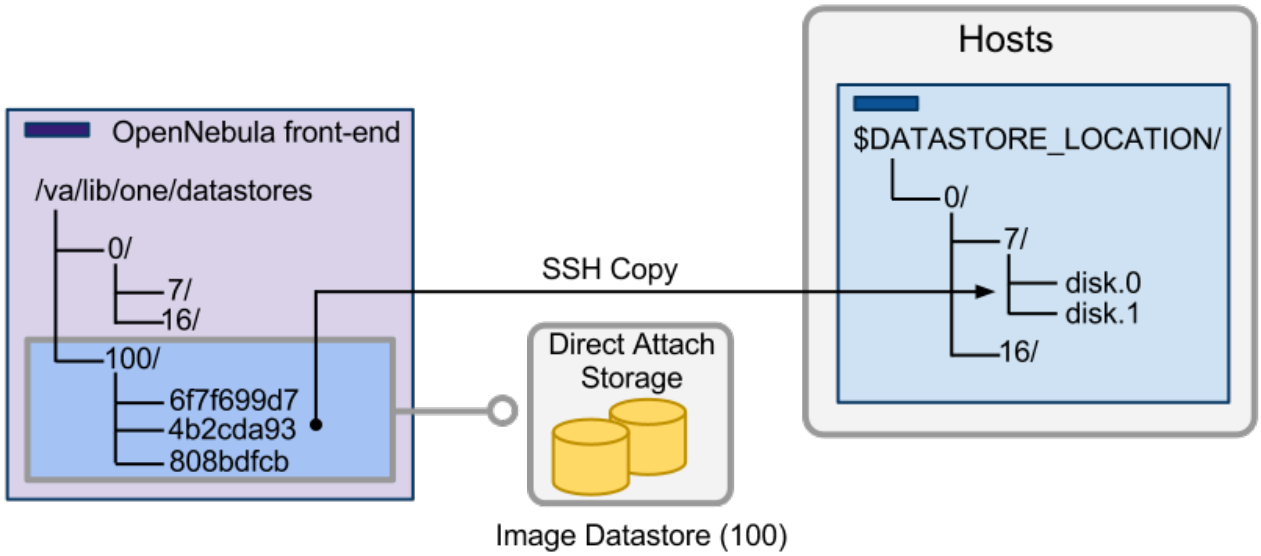

Рис. 45

10.1.1.2. Установка фронтальной машины

Перед установкой фронтальной машины требуется подготовить память для:

- хранилищ образов;
- системных хранилищ.

Режимы передачи Shared и Qcow2

Смонтировать каталог хранилища образов на фронтальной машине по адресу `/var/lib/one/datastores/<datastore_id>`. Если все хранилища одного типа, можно смонтировать весь каталог `/var/lib/one/datastores`.

ВНИМАНИЕ! Для фронтальной машины необходимо смонтировать только хранилища образов.

Для монтирования NFS-томов рекомендуются опции `soft`, `intr`, `rsize=32768`, `wsize=32768`, `no_root_squash`.

Режим передачи SSH

Убедиться, что в разделе `/var/lib/one/datastores` достаточно места для хранения образов и дисков остановленных и не развёртываемых виртуальных машин. При этом `/var/lib/one/datastores` может монтироваться из любого SAN/NAS сервера в сети.

10.1.1.3. Установка узла

Режимы передачи Shared и Qcow2

Смонтировать на каждом узле каталоги хранилищ образов и системных хранилищ по адресу
`/var/lib/one/datastores/<datastore_id>`. Если все хранилища одного типа, можно смонтировать весь каталог `/var/lib/one/datastores`.

Режим передачи SSH

Убедиться, что в разделе `/var/lib/one/datastores` на данном узле достаточно места для хранения дисков рабочих виртуальных машин.

ВНИМАНИЕ! Убедиться в том, что все узлы, включая фронтальную машину, могут осуществлять ssh-передачу на любой другой узел, включая самих себя. В противном случае перемещения не будут выполняться.

10.1.1.4. Настройка OpenNebula

После установки хранилища файловой системы настройка OpenNebula выполняется в два этапа:

- создание системного хранилища;
- создание хранилище образов.

Создание системного хранилища

При создании нового системного хранилища необходимо указать его имя, тип и способ передачи данных в соответствии с таблицей 11.

Таблица 11

Параметр	Значение
`NAME`	Имя хранилища
`TYPE`	`SYSTEM_DS`
`TM_MAD`	`shared` — для режима совместной передачи; `qcow2` — для режима передачи qcow2; `ssh` — для режима передачи ssh

Пример

Создание системного хранилища с режимом совместной передачи как в Sunstone, так и в CLI

```
$ cat systemds.txt
NAME = nfs_system TM_MAD = shared TYPE = SYSTEM_DS
$ onedatastore create systemds.txt ID: 101
```

Создание хранилища образов

При создании нового хранилища образов необходимо указать его имя, тип и способ передачи данных в соответствии с таблицей 12.

Таблица 12

Параметр	Значение
NAME	Имя хранилища
TYPE	fs
TM_MAD	shared — для режима совместной передачи; qcow2 — для режима передачи qcow2; ssh — для режима передачи ssh

Пример

Создание хранилища образов с использованием драйверов совместной передачи данных

```
$ cat ds.conf
NAME = nfs_images DS_MAD = fs TM_MAD = shared
$ onedatastore create ds.conf ID: 100
```

ВНИМАНИЕ! Обязательно использовать одинаковое значение параметра TM_MAD для системного хранилища и для хранилища образов.

Примечание. Дополнительно можно установить атрибуты шаблона хранилища.

10.1.1.5. Дополнительные параметры

Драйверы qcow2 являются разновидностью совместно используемых драйверов для работы с форматом qcow2 для образов дисков. Образы создаются и передаются с помощью команды qemu-img с использованием оригинального образа в качестве опорного файла. Стандартные параметры можно отправить на клонирование qemu-img с помощью функции QCOW2_OPTIONS в /var/lib/one/remotes/tm/tmrc.

10.1.2. Хранилище данных LVM

Драйвер хранилища данных LVM обеспечивает OpenNebula возможность использования LVM-томов вместо обычных файлов для хранения образов. При использовании данного типа хранилища отсутствует необходимость в организации файловой системы.

10.1.2.1. Схема хранилища данных

Образы хранятся как обычные файлы, стандартный путь размещения в хранилище образов `/var/lib/one/datastores/<id>`, но при создании ВМ они выгружаются в логические тома (LV). Виртуальные машины запускаются из LV на узле.

На рис. 46 приведена рекомендуемая схема для расширения функциональных возможностей, которую следует применять при наличии сети хранения данных high-end SAN. LUN можно экспортировать на все узлы, при этом виртуальные машины смогут работать непосредственно из SAN.

Примечание. Для хранилища LVM не требуется настройка кластерного управления логическими томами (CLVM) в кластере. Драйверы обновляют метаданные LVM каждый раз, когда образ требуется в другом узле.

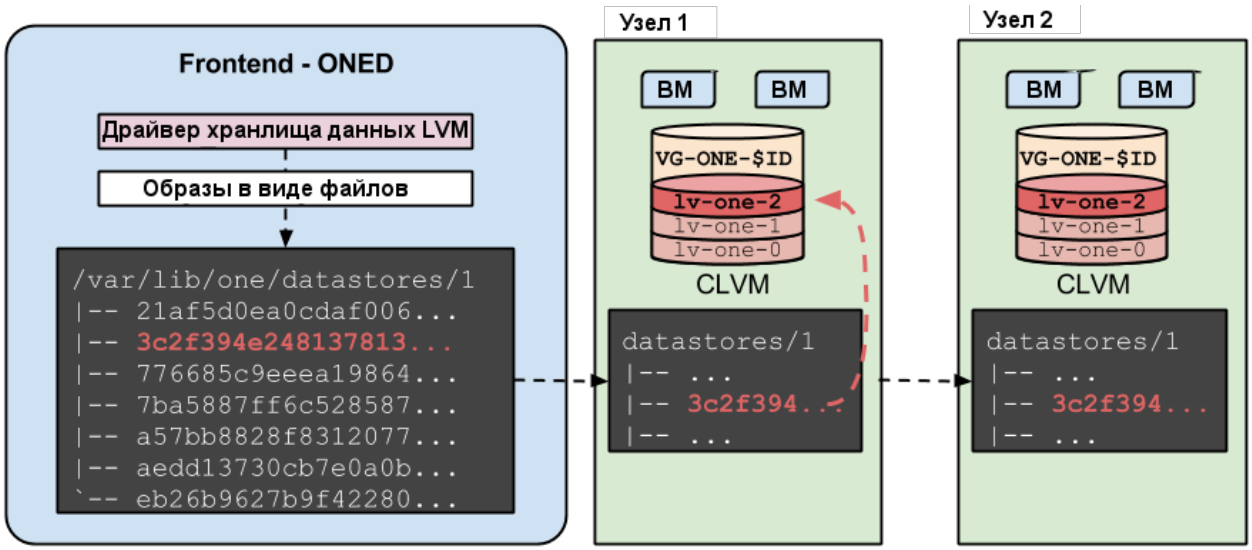

Рис. 46

Пример

В системе две виртуальные машины (9 и 10) используют диск с ID 0, работающий в хранилище LVM. На узлах выполнена настройка совместно используемого LUN и создана группа тома с именем `vg-one-0`. Хранилище будет иметь следующую схему:

```
# lvs
  LV           VG        Attr        LSize  Pool  Origin  Data%  Meta%  Move
  lv-one-10-0  vg-one-0  -wi-------  2.20g
  lv-one-9-0   vg-one-0  -wi-------  2.20g
```

ВНИМАНИЕ! Образы хранятся в совместно используемом хранилище в виде файла, например, NFS. При этом необходимо выполнить настройку каталогов хранилища и точек монтирования как стандартного совместно используемого хранилища образов (см. 10.1.1). Рекомендуется сначала выполнить развертывание совместно используемого хранилища

файловой системы, затем, убедившись в его корректной работе, заменить соответствующее системное хранилище на хранилище LVM.

10.1.2.2. Установка фронтальной машины

Дополнительная настройка не требуется.

10.1.2.3. Установка узла

Узлы должны отвечать следующим требованиям:

- на узлах должен быть LVM2;
- `lvmetad` должен быть отключен. Для этого в файле `/etc/lvm/lvm.conf` установить следующие значение для параметра:

```
use_lvmetad = 0
```

- все узлы должны иметь доступ к одним и тем же LUN;
- на одном узле должен быть создан LVM VG в совместно используемых LUN для каждого хранилища с именем `vg-one-<system_ds_id>`;
- диски виртуальных машин являются символьными ссылками на устройства блока. Дополнительные файлы ВМ, такие как файлы контрольной точки или файлы развертывания, хранятся в каталоге `/var/lib/one/datastores/<id>`;
- все узлы должны иметь доступ к хранилищу образов, для чего создается соответствующий каталог.

10.1.2.4. Настройка OpenNebula

После установки хранилища настройка OpenNebula выполняется в два этапа:

- создание системного хранилища;
- создание хранилища образов.

Создание системного хранилища

Системные хранилища LVM должны создаваться со значениями, приведенными в таблице 13

Таблица 13

Параметр	Значение
NAME	Имя хранилища
TM_MAD	fs_lvm
TYPE	SYSTEM_DS

Пример

```
> cat ds.conf
NAME = lvm_system
TM_MAD = fs_lvm
TYPE = SYSTEM_DS
```

```
BRIDGE_LIST = "NODE1 NODE2"

> sudo onedatastore create ds.conf
ID: 100
```

Создание хранилища образов

Хранилища образов должны создаваться со значениями, приведенными в таблице 14

Таблица 14

Параметр	Значение
NAME	Имя хранилища
TYPE	IMAGE_DS
DS_MAD	fs
TM_MAD	fs_lvm
DISK_TYPE	BLOCK
BRIDGE_LIST	Список узлов, разделенных пробелами, с доступом к совместно используемому хранилищу, предназначенному для копирования новых образов

Пример

Создание хранилища LVM с использованием конфигурационного файла. Узел NODE1 используется как один из узлов, подключаемых OpenNebula LVM

```
> cat ds.conf
NAME = production
DS_MAD = fs
TM_MAD = fs_lvm
DISK_TYPE = "BLOCK"
TYPE = IMAGE_DS
SAFE_DIRS="/var/tmp /tmp"
BRIDGE_LIST = "NODE1 NODE2"

> sudo onedatastore create ds.conf
ID: 101
```

10.1.3. Хранилище данных Ceph

Драйвер хранилища Ceph обеспечивает пользователям OpenNebula возможность использования блочных устройств Ceph в качестве виртуальных образов.

ВНИМАНИЕ! Для работы данного драйвера необходимо, чтобы узлы OpenNebula, использующие драйвер Ceph, являлись Ceph-клиентами работающего Ceph-кластера.

10.1.3.1. Схема хранилища данных

Образы и диски виртуальных машин хранятся в одном Ceph-пуле. Каждый образ имеет в пуле имя one-<IMAGE_ID>. Виртуальные машины будут использовать эти RBD-тома для своих дисков если образы являются устойчивыми, в противном случае создаются новые моментальные снимки состояния в виде one-<IMAGE_ID>-<VM_ID>-<DISK_ID>.

10.1.3.2. Установка Ceph-кластера

Предварительно Ceph-кластер должен быть развернут. Дополнительно необходимо выполнить следующие настройки:

- создать пул для хранилищ OpenNebula. Записать имя пула для его включения в описание хранилищ:

```
$ ceph osd pool create one 128
$ ceph osd lspools
0 data,1 metadata,2 rbd,6 one,
```

- определить Ceph-пользователя, который будет иметь доступ к пулу хранилищ. Данный пользователь будет также использоваться libvirt для доступа к образам дисков, например:

```
$ ceph auth get-or-create client.libvirt mon 'allow rwx' osd \
'allow rwx pool=one'
```

Кроме того, необходимо получить копию ключа данного пользователя для ее дальнейшей передачи на узлы OpenNebula, например:

```
$ ceph auth get-key client.libvirt | tee client.libvirt.key
$ ceph auth get client.libvirt -o ceph.client.libvirt.keyring
```

- несмотря на то, что RBD-формат «1» поддерживается, настоятельно рекомендуется использовать RBD-формат «2», для этого в файле ceph.conf указать:

```
[global]
rbd_default_format = 2
```

- выбрать группу клиентских узлов кластера для использования в качестве мостов хранилищ storage bridges. Эти узлы будут использоваться для импорта образов в Ceph-кластер из OpenNebula. В данных узлах должна быть установлена команда qemu-img.

10.1.3.3. Установка фронтальной машины

Для фронтальной машины не требуется специальная установка Ceph. Фронтальная машина будет выполнять доступ к Ceph-кластеру через мосты хранилищ.

10.1.3.4. Установка узла

Для использования Ceph-кластера необходимо выполнить следующую настройку узлов:

- должны быть установлены клиентские инструментальные средства Ceph;

- демон MON должен быть определен в ceph.conf для всех узлов, поэтому значения hostname и port не требуется указывать в Ceph-командах;
- от имени пользователя, входящего в группы astra-admin и astra-console, скопировать набор ключей Ceph-пользователя ceph.client.libvirt.keyring на узлы в каталог /etc/ceph и ключ пользователя client.libvirt.key в каталог /var/lib/one:

```
$ scp ceph.client.libvirt.keyring admin@node:/tmp
$ scp client.libvirt.key admin@node:/tmp
$ ssh admin@node "sudo mv /tmp/ceph.client.libvirt.keyring /etc/ceph"
$ ssh admin@node "sudo mv /tmp/client.libvirt.key /var/lib/one"
$ ssh admin@node "sudo ln -s /var/lib/one/client.libvirt.key
   /root/client.libvirt.key"
```

- сгенерировать секретный ключ для Ceph-пользователя и скопировать его на узлы в каталог /var/lib/one. Запомнить универсальный уникальный идентификатор (UUID) для дальнейшего использования:

```
$ UUID='uuidgen'
$ cat > secret.xml <<EOF
<secret ephemeral='no' private='no'>
   <uuid>$UUID</uuid>
   <usage type='ceph'>
            <name>client.libvirt secret</name>
   </usage>
</secret> EOF

$ scp secret.xml admin@node:/tmp
$ ssh admin@node "sudo mv /tmp/secret.xml /var/lib/one"
$ ssh admin@node "sudo ln -s /var/lib/one/secret.xml /root/secret.xml"
```

- установить секретный ключ libvirt и удалить файлы ключа на узлах:

```
$ ssh admin@node "sudo virsh -c qemu:///system secret-define secret.xml"
$ ssh admin@node "sudo virsh -c qemu:///system secret-set-value --secret
     $UUID --base64 $(cat /root/client.libvirt.key)"
$ ssh admin@node "sudo rm /root/client.libvirt.key"
$ ssh admin@node "sudo rm /var/lib/one/client.libvirt.key"
```

- убедиться, что Ceph-клиент имеет корректные настройки на узле:

```
$ ssh admin@node "sudo rbd ls -p one --id libvirt"
```

- убедиться, что на узлах выделено достаточно места для хранения вспомогательных файлов виртуальных машин, таких как context-диски, файлы развертывания и файлы контрольной точки.

10.1.3.5. Настройка OpenNebula

Для использования Ceph-кластера с OpenNebula необходимо определить системное хранилище и хранилище образов. Оба хранилища совместно используют одни и те же опции конфигурации, приведенные в таблице 15, и Ceph-пул.

Примечание. Можно добавить дополнительные хранилища образов и системные хранилища, указав другие пулы с отличными от Ceph политиками распределения ресурсов/репликации.

Таблица 15

Параметр	Описание	Обязательный
POOL_NAME	Имя Ceph-пула	ДА
CEPH_USER	Имя Ceph-пользователя, используемое командами libvirt и rbd	ДА
TM_MAD	ceph	ДА
CEPH_CONF	Нестандартный конфигурационный файл Ceph, если необходим	НЕТ
RBD_FORMAT	По умолчанию будет использоваться RBD-формат «2»	НЕТ

Создание системного хранилища

Создать системное хранилище через Sunstone или CLI.

Пример

Создание хранилища используя CLI

```
$ cat systemds.txt
NAME = ceph_system
TM_MAD = ceph
TYPE = SYSTEM_DS
POOL_NAME = one
CEPH_USER = libvirt

$ sudo onedatastore create systemds.txt ID: 101
```

Примечание. Также Ceph может работать с системным хранилищем типа Filesystem в режиме совместной передачи согласно 10.1.1. В этом случае энергозависимые диски и диски подкачки создаются в виде обычных файлов в системном хранилище. Кроме Ceph-кластера необходимо выполнить установку совместно используемой файловой системы.

Создание хранилища образов

При создании хранилища образов дополнительно к параметрам, приведенным в таблице 15, устанавливаются параметры, указанные в таблице 16.

Таблица 16

Параметр	Описание	Обязательный
DS_MAD	ceph	ДА
DISK_TYPE	RBD	ДА
BRIDGE_LIST	Список мостов хранилища для доступа к Ceph-кластеру	ДА
CEPH_HOST	Разделенный пробелами список Ceph-мониторов, например, NODE1:port1 NODE2:port2 host4:port4	ДА
CEPH_SECRET	UUID секретного ключа libvirt	ДА
STAGING_DIR	Путь по умолчанию для операций с образом в мостах	НЕТ

Пример

Хранилище образов

```
> cat ds.conf

NAME = "cephds"
DS_MAD = ceph
TM_MAD = ceph
DISK_TYPE = RBD
POOL_NAME = one
CEPH_HOST = NODE1:port1 NODE2:port2
CEPH_USER = libvirt
CEPH_SECRET = "6f88b54b-5dae-41fe-a43e-b2763f601cfc"
BRIDGE_LIST = NODE1 NODE1

> sudo onedatastore create ds.conf
ID: 101
```

10.1.3.6. Дополнительные параметры

Следующие значения по умолчанию для Ceph-драйверов находятся в файле /var/lib/one/remotes/datastore/ceph/ceph.conf:

- POOL_NAME — группа томов;
- STAGING_DIR — путь для операций с образом;
- RBD_FORMAT — формат для RBD томов.

10.1.4. Хранилище Raw Device Mapping

Хранилище Raw Device Mapping (RDM) является хранилищем образов, обеспечивающим динамический доступ к блочным устройствам узла.

10.1.4.1. Схема хранилища

Хранилище RDM предназначено для регистрации уже существующих блочных устройств узла. Устройства должны быть установлены и доступны, а виртуальные машины,

использующие эти устройства, должны быть настроены для работы в подготовленных для них узлах. Дополнительные файлы виртуальных машин, такие как файлы развертывания или энергозависимые диски, создаются как обычные файлы.

10.1.4.2. Установка фронтальной машины

Дополнительная установка не требуется.

10.1.4.3. Установка узла

Дополнительная установка не требуется.

10.1.4.4. Настройка OpenNebula

После установки хранилища настройка OpenNebula выполняется в два этапа:

- создание системного хранилища;
- создание хранилища образов.

Создание системного хранилища

Хранилище RDM может работать с хранилищами файловой системы в следующих режимах:

- в режиме совместно используемой передачи;
- в режиме передачи ssh.

Хранилище файловой системы используется только для энергозависимых дисков и context-устройств.

Создание хранилища образов

Хранилища образов должны создаваться со значениями, приведенными в таблице 17.

Таблица 17

Параметр	Значение
NAME	Имя хранилища
TYPE	IMAGE_DS
DS_MAD	dev
TM_MAD	dev
DISK_TYPE	BLOCK

Пример

```
> cat rdm.conf

NAME = rdm_datastore
TYPE = "IMAGE_DS"
DS_MAD = "dev"
TM_MAD = "dev"
```

```
DISK_TYPE = "BLOCK"


> sudo onedatastore create rdm.conf
ID: 101
```

10.1.4.5. Использование хранилища

В хранилище можно добавлять новые образы с указанием пути. При использовании CLI нельзя применять сокращенные параметры, т.к. CLI проверяет, существует ли файл и устройство на фронтальной машине.

Пример

Шаблон образа, в который добавляется диск узла `/dev/sdb`

```
NAME=scsi_device
PATH=/dev/sdb
PERSISTENT=YES
```

Примечание. Данное хранилище является контейнером для существующих устройств, и образы используют его память. Все зарегистрированные устройства имеют размер 0, а хранилище устройств в целом занимает не более 1 МБ доступного пространства.

10.1.5. Хранилище iSCSI-Libvirt

Данное хранилище предназначено для регистрации уже существующих томов iSCSI, доступных узлам гипервизора.

10.1.5.1. Установка фронтальной машины

Дополнительная настройка не требуется.

10.1.5.2. Установка узла

Дополнительная настройка не требуется.

10.1.5.3. Аутентификация iSCSI CHAP

Для использования аутентификации по протоколу CHAP необходимо создать секретный ключ `libvirt` на всех узлах.

При использовании аутентификации CHAP необходимо учесть следующее:

- поле `incominguser` в файле аутентификации iSCSI должно соответствовать параметру хранилища `ISCSI_USER`;
- поле `<target>` в `XML`-файле секретного ключа должно содержать параметр `ISCSI_USAGE`;
- действия должны быть выполнены на всех узлах.

Для настройки аутентификации CHAP необходимо:

1) создать файл аутентификации iSCSI-источника:

```
<target iqn.2013-07.com.example:iscsi-pool>
```

```
backing-store /home/tgtd/iscsi-pool/disk1
backing-store /home/tgtd/iscsi-pool/disk2
incominguser myname mysecret
</target>
```

2) создать файл iscsi-secret.xml:

```
$ cat > iscsi-secret.xml <<EOF
<secret ephemeral='no' private='yes'>
<description>Passphrase for the iSCSI example.com server</description>
<usage type='iscsi'>
<target>libvirtiscsi</target>
</usage>
</secret> EOF
```

3) зарегистрировать созданный XML-файл в libvirt:

```
$ sudo virsh secret-define iscsi-secret.xml
Secret c4dbe20b-b1a3-4ac1-b6e6-2ac97852ebb6 created

$ sudo virsh secret-list
 UUID                                  Usage
------------------------------------------------------------
 c4dbe20b-b1a3-4ac1-b6e6-2ac97852ebb6 iscsi libvirtiscsi

$ MYSECRET='printf %s "mysecret" | base64'
$ sudo virsh secret-set-value c4dbe20b-b1a3-4ac1-b6e6-2ac97852ebb6 $MYSECRET
Secret value set
```

10.1.5.4. Настройка OpenNebula

После установки хранилища настройка OpenNebula выполняется в два этапа:

- создание системного хранилища;
- создание хранилища образов.

Создание системного хранилища

Хранилище RDM может работать с хранилищами файловой системы в следующих режимах:

- в совместном режиме;
- в режиме ssh-транспорта.

Хранилище файловой системы используется только для энергозависимых дисков и context-устройств.

Создание хранилища образов

Хранилища образов должны создаваться со значениями, приведенными в таблице 18

Таблица 18

Параметр	Значение
NAME	Имя хранилища
TYPE	IMAGE_DS
DS_MAD	iscsi
TM_MAD	iscsi
DISK_TYPE	iscsi
ISCSI_HOST	Узел iSCSI. Например, host или host:port

При необходимости использовать аутентификацию CHAP добавить к хранилищу параметры, приведенные в таблице 19.

Таблица 19

Параметр	Значение
ISCSI_USAGE	Использование секретного ключа со строкой аутентификации CHAP
ISCSI_USER	Аутентификация iSCSI CHAP пользователя

Пример

```
> cat iscsi.ds

NAME = iscsi
DISK_TYPE = "ISCSI"
DS_MAD = "iscsi"
TM_MAD = "iscsi"
ISCSI_HOST = "the_iscsi_host"
ISCSI_USER = "the_iscsi_user"
ISCSI_USAGE = "the_iscsi_usage"

> sudo onedatastore create iscsi.ds
ID: 101
```

ВНИМАНИЕ! Образы, создаваемые в данном хранилище, должны быть устойчивыми. Если образы не являются устойчивыми, появляется возможность использования данного устройства более чем одной ВМ, что может привести к возникновению проблем и повреждению данных.

10.1.5.5. Использование хранилища

Можно добавлять новые образы с указанием полного пути. При использовании CLI не следует применять сокращенные параметры, т.к. CLI проверят, существует ли файл и устройство на фронтальной машине.

Пример

Шаблон образа, в который добавляется диск узла iqn.1992-01.com.example:storage:diskarrays-sn-a8675309

```
NAME = iscsi_device
PATH = iqn.1992-01.com.example:storage:diskarrays-sn-a8675309
PERSISTENT = YES
```

ВНИМАНИЕ! Данное хранилище является контейнером для существующих устройств, и образы используют его память. Все зарегистрированные устройства имеют размер 0, а хранилище устройств в целом занимает не более 1 МБ доступного пространства.

Примечание. Можно отменить опции ISCSI_HOST, ISCSI_USER, ISCSI_USAGE и ISCSI_IQN в шаблоне образа. Отмененные опции вступят в действие для новых виртуальных машин.

Пример

Шаблон iSCSI LUN, использующий диспетчер передачи iSCSI

```
NAME=iscsi_device_with_lun
PATH=iqn.2014.01.192.168.50.61:test:7cd2cc1e/0
ISCSI_HOST=192.168.50.61
PERSISTENT=YES
```

10.2. Хранилище файлов

Хранилище файлов позволяет хранить обычные файлы, которые могут использоваться как ядра виртуальных машин (kernels), временные диски (ramdisks) или контекстные файлы (context-файлы). Хранилище файлов не является особым механизмом хранения. Файловое хранилище создается по умолчанию после установки OpenNebula (хранилище с ID: 2).

10.2.1. Требования

Специальные требования отсутствуют. В ходе работы используются стандартные утилиты, например, cp, ln, mv, tar, mkfs, которые установлены в системе по умолчанию.

10.2.2. Настройка

Большинство критериев настройки, используемых для хранилищ образов дисков, применяются к файловому хранилищу.

Особые атрибуты для драйвера данного хранилища перечислены в таблице 20.

Таблица 20

Параметр	Описание
TYPE	Использовать FILE_DS для установки хранилища файлов
DS_MAD	Тип хранилища. Применить fs для использования файловых драйверов
TM_MAD	Драйверы передачи для хранилища. Использовать режим ssh для передачи файлов

Пример

Создание хранилища файлов

```
> cat kernels_ds.conf

NAME = kernels
DS_MAD = fs TM_MAD = ssh TYPE = FILE_DS
SAFE_DIRS = /var/tmp/files

> sudo onedatastore create kernels_ds.conf
ID: 100

> sudo onedatastore list
 ID   NAME          CLASTER    IMAGES    TYPE     DS      TM MAD
  0   system           -           0     sys      -       dummy
  1   default          -           0     img      dumm    dummy
  2   files            -           0     fil      fs      ssh
100   kernels          -           0     fil      fs      ssh
```

Значения параметров DS и TM MAD можно впоследствии изменить командой onedatastore update. Подробные значения параметров хранилища можно просмотреть с помощью команды onedatastore show.

10.2.3. Настройка узла

Рекомендуемый драйвер ssh для хранилища файлов не требует особой настройки. Достаточно убедиться, что в разделе $DATASTORE_LOCATION достаточно места для хранения файлов ВМ на фронтальной машине и на узлах.

11. УСТАНОВКА ОТКРЫТОЙ ОБЛАЧНОЙ СЕТИ

При запуске новой ВМ OpenNebula подсоединяет свои сетевые интерфейсы, определяемые опцией NIC, к физическим устройствам гипервизора в соответствии с настройками виртуальной сети. Это позволяет ВМ иметь доступ к публичным и частным сетям.

OpenNebula поддерживает четыре сетевых режима:

- режим Bridged (сетевой мост) — ВМ напрямую соединяется с существующим мостом в гипервизоре. Данный режим может быть настроен на использование групп безопасности и изоляции сети;
- режим VLAN — виртуальные сети внедряются с применением технологии назначения портов на виртуальные локальные сети VLAN стандарта IEEE802.1Q (VLAN-тегирование стандарта IEEE802.1Q);
- режим VXLAN — виртуальные сети задействуют сети VLAN, используя протокол VXLAN, основанный на UDP-инкапсуляции и групповой адресации IP;
- режим Open vSwitch — аналогичен режиму VLAN, но использует программный коммутатор Open vSwitch (OVS) вместо Linux bridge. Группы безопасности данным режимом не поддерживаются.

При создании новой сети необходимо добавить параметр `VN_MAD` в шаблон, в качестве значения указав сетевой режим для использования.

Сетевой стек OpenNebula может объединяться с внешним диспетчером IP-адресов (IPAM). Для этого необходимо добавить связующий элемент.

11.1. Установка узла

11.1.1. Сетевой режим Bridged

11.1.1.1. Требования

Для настройки сетевого режима необходимо выполнение следующих требований:

- должен быть установлен пакет узла OpenNebula в соответствии с 8.5.1;
- необходимо установить на узлы пакет `ebtables`, который по умолчанию обеспечивает изоляцию сети.

11.1.1.2. Настройка

Создать Linux bridge для каждой сети, в которой будут работать виртуальные машины. Использовать одно имя сети на всех узлах. Добавить физический сетевой интерфейс к мосту.

Пример

Узел с двумя сетями: публичной, подключенной к eth0, и частной,подключенной к eth1, должен иметь два моста

```
brctl show
```

```
bridge name bridge id
STP enabled interfaces
br08000.001e682f02ac noeth0
br18000.001e682f02ad noeth1
```

11.1.2. Сетевой режим VLAN

11.1.2.1. Требования

Для настройки сетевого режима необходимо выполнение следующих требований:

- должен быть установлен пакет узла OpenNebula (см. 8.5.1);
- модуль 802.1Q должен быть загружен в ядро;
- наличие переключателя сети, способного направлять VLAN-тегированный трафик. Физические порты переключателя должны быть каналами связи VLAN.

11.1.2.2. Настройка

Дополнительная настройка не требуется.

11.1.3. Сетевой режим VXLAN

11.1.3.1. Требования

Для настройки сетевого режима необходимо выполнение следующих требований:

- должен быть установлен пакет узла OpenNebula (см. 8.5.1);
- при подключении всех узлов к одному домену передачи данных групповой трафик не должен фильтроваться правилами iptables в узлах. Если групповой трафик должен проходить через маршрутизаторы, необходимо настроить в сети многоадресный протокол, например, IGMP.

11.1.3.2. Настройка

Дополнительная настройка не требуется.

11.1.4. Сетевой режим Open vSwitch

11.1.4.1. Требования

Для настройки сетевого режима необходимо выполнение следующих требований:

- должен быть установлен пакет узла OpenNebula (см. 8.5.1);
- на каждом узле должен быть установлен Open vSwitch (пакет `openvswitch-switch`).

Пример

Узел, который направляет трафик виртуальных сетей через сетевой интерфейс `enp0s8`, должен создавать следующий переключатель Open vSwitch:

```
$ sudo ovs-vsctl show

c61ba96f-fc11-4db9-9636-408e763f529e Bridge "ovsbr0"
```

```
Port "ovsbr0"
Interface "ovsbr0" type: internal
Port "enp0s8"
Interface "enp0s8"
```

11.1.4.2. Настройка

Для настройки необходимо создать Open vSwitch для каждой сети, в которой будут работать виртуальные машины. На всех узлах необходимо использовать одно имя для Open vSwitch. Затем добавить физический сетевой интерфейс к Open vSwitch.

11.2. Сетевой мост

В данном режиме трафик ВМ напрямую передается через существующий Linux bridge в узлах. Мосты могут работать в трех различных режимах в зависимости от дополнительной фильтрации трафика, выполняемой OpenNebula:

- Dummy — фильтрация не выполняется;
- Security Group — устанавливаются правила iptables для внедрения правил групп безопасности;
- ebtables VLAN — аналогично Security Group с дополнительными правилами ebtables для изоляции (L2) всех виртуальных сетей.

11.2.1. Особенности и ограничения

При изоляции трафика необходимо учитывать следующее:

- в режимах Dummy и Security Group можно добавлять тегированные сетевые интерфейсы для обеспечения сетевой изоляции. Данный режим является рекомендуемой стратегией развертывания в работающих системах (не тестовых);
- режим ebtables VLAN предназначен для небольших сред без соответствующей аппаратной поддержки для внедрения сетей VLANS. Данный режим ограничен сетями /24 и IP-адреса не могут перекрываться в виртуальных сетях. Рекомендуется только для целей тестирования.

11.2.2. Настройка OpenNebula

Мостовая сеть не требует специальных настроек.

11.2.3. Определение мостовой сети

Для создания сети необходимо добавить параметры, приведенные в таблице 21.

Таблица 21

Параметр	Значение	Обязательный
VN_MAD	Dummy — для режима Dummy Bridged mode; fw — для режима Bridged с группами безопасности; ebtables — для режима Bridged с изоляцией ebtables	ДА
BRIDGE	Имя linux bridge в узлах	ДА

Пример

Сеть типа «мост» с использованием режима групп безопасности (Security Groups mode)

```
NAME = "bridged_net"
VN_MAD = "fw"
BRIDGE = vbr1
...
```

11.2.4. Режим ebtables VLAN

Правила ebtables, которые создаются при необходимости отладки настройки:

```
# Игнорировать пакеты, которые не соответствуют MAC-адресу сети
-s ! <mac_address>/ff:ff:ff:ff:ff:0 -o <tap_device> -j DROP
# Предотвратить MAC-спуфинг
-s ! <mac_address> -i <tap_device> -j DROP
```

11.3. Сети 802.1Q VLAN

Данный драйвер создает мост для каждой виртуальной сети OpenNebula и подключает VLAN-тегированный сетевой интерфейс к мосту. Механизм совместим со стандартом IEEE 802.1Q.

Идентификационный номер VLAN будет одинаковым для всех интерфейсов в конкретной сети и автоматически рассчитываться OpenNebula. Возможно также принудительно указать значение параметра VLAN_ID в шаблоне виртуальной сети.

11.3.1. Настройка OpenNebula

Значение параметра VLAN_ID рассчитывается в соответствии с параметром настройки oned.conf:

```
# VLAN_IDS: Пул VLAN ID для автоматического назначения VLAN_ID. Данный пул
# предназначен для сетей 802.1Q (Open vSwitch и драйверы 802.1Q).

VLAN_IDS = [
      START = "2",
      RESERVED = "0, 1, 4095"
]
```

Изменением данной опции можно зарезервировать некоторые сети VLAN, и они не будут назначаться виртуальной сети. Можно также указать первый номер VLAN_ID. При создании новой изолированной сети OpenNebula находит свободный номер VLAN_ID из пула VLAN. Этот пул является глобальным и совместно используется с сетевым режимом Open vSwitch.

В файле /var/lib/one/remotes/vnm/OpenNebulaNetwork.conf можно откорректировать параметр настройки validate_vlan_id. Установив значение true можно проверить, что другие сети VLAN не подсоединены к мосту.

11.3.2. Определение сети 802.1Q

Для создания сети по стандарту 802.1Q необходимо задать значения, приведенные в таблице 22.

Таблица 22

Параметр	Значение	Обязательный
VN_MAD	802.1Q	ДА
PHYDEV	Имя физического сетевого устройства, которое будет подключено к мосту	ДА
BRIDGE	Имя linux bridge, назначается по умолчанию onebr.<net_id> или onebr.<vlan_id>	НЕТ
VLAN_ID	Идентификационный номер сети VLAN. Будет сгенерирован, если не указан, а AUTOMATIC_VLAN_ID устанавливается на YES	ДА (если не AUTOMATIC_VLAN_ID)
AUTOMATIC_VLAN_ID	Обязательный и должен быть установлен на YES, если VLAN_ID определен	ДА (если не VLAN_ID)
MTU	Максимальный передаваемый модуль данных (MTU) для тегированного интерфейса и моста	НЕТ

Пример

Определение сети 802.1Q

```
NAME = "hmnet" VN_MAD = "802.1Q"
PHYDEV= "eth0"
VLAN_ID = 50
BRIDGE= "brhm"
```

В данном примере драйвер проверяет наличие моста brhm. Если он не существует, то будет создан. Eth0 будет тегирован (eth0.50) и подсоединен к brhm.

11.4. Сети VXLAN

Данный драйвер создает мост для каждой виртуальной сети OpenNebula и подключает VXLAN-тегированный сетевой интерфейс к мосту.

Идентификационный номер VLAN будет одинаковым для всех интерфейсов в конкретной сети и автоматически рассчитываться OpenNebula. Возможно также принудительно указать значение параметра VLAN_ID в шаблоне виртуальной сети.

Кроме того, каждая сеть VLAN назначает групповой адрес для инкапсуляции транслирования L2 и группового трафика. Данный адрес назначается по умолчанию диапазону 239.0.0.0/8 в соответствии с RFC 2365 (административно назначаемая групповая адресация IP). В частности, групповой адрес получается добавлением VLAN_ID к основному адресу 239.0.0.0/8.

11.4.1. Особенности и ограничения

Драйвер работает со стандартным UDP-портом сервера 8472.

Трафик VXLAN направляется на физическое устройство, которое может быть установлено как VLAN-тегированный интерфейс, но в этом случае необходимо убедиться в том, что тегированный интерфейс будет создан вручную изначально на всех узлах.

11.4.2. Настройка OpenNebula

Можно указать начальный VLAN_ID путем корректировки файла /etc/one/oned.conf:

```
# VXLAN_IDS: Автоматическое назначение идентификационного номера сети VXLAN (/
   VNI). Используется
# для сетей vxlan.
# запуск: Первый VNI, который может использоваться

VXLAN_IDS = [
      START = "2"
]
```

Параметры настройки, приведенные в таблице 23, можно откорректировать в файле /var/lib/one/remotes/vnm/OpenNebulaNetwork.conf.

Таблица 23

Параметр	Описание
vxlan_mc	Основной групповой адрес для каждой сети VLAN. Групповой адрес: vxlan_mc + vlan_id
vxlan_ttl	Время жизни (TTL) должно быть меньше 1 в маршрутизируемых многоадресных сетях (IGMP)
validate_vlan_id	Установить на true для проверки, что другие сети VLAN не подсоединены к мосту

11.4.3. Определение сети VXLAN

Для создания сети VXLAN необходимо задать значения, приведенные в таблице 24.

Таблица 24

Параметр	Значение	Обязательный
VN_MAD	vxlan	ДА
PHYDEV	Имя физического сетевого устройства, которое будет подключено к мосту	ДА
BRIDGE	Имя linux bridge, назначается по умолчанию onebr.<net_id> или onebr.<vlan_id>	НЕТ
VLAN_ID	Идентификационный номер сети VLAN. Будет сгенерирован, если не указан	НЕТ
MTU	Максимальный передаваемый модуль данных (MTU) для тегированного интерфейса и моста	НЕТ

Пример

Определение сети VXLAN

```
NAME = "vxlan_net"
VN_MAD = "vxlan"
PHYDEV = "eth0"
VLAN_ID = 50 # optional
BRIDGE = "vxlan50" # optional
```

В данном примере драйвер проверяет наличие моста vxlan50. Если он не существует, то будет создан. eth0 будет тегирован (eth0.50) и подсоединен к vxlan50. eth0 может иметь 802.1Q тегированный интерфейс, если предполагается изолировать трафик сети VXLAN OpenNebula.

11.5. Сети Open vSwitch

Сети Open vSwitch создаются на базе программного коммутатора Open vSwitch.

Open vSwitch — программный многоуровневый коммутатор, обеспечивающий изоляцию сети с помощью сетей VLAN путем тегирования портов и фильтрацию базовой сети с помощью OpenFlow.

Архитектура OVS состоит из трех основных компонентов: базы данных, непосредственно программного коммутатора и управляющего контроллера. На каждом из физических узлов вместе с гипервизором располагаются собственные БД и коммутатор. Эти два компонента образуют отдельно стоящий коммутатор, не знающий о других программных коммутаторах на соседних узлах.

ВНИМАНИЕ! Данный драйвер несовместим с группами безопасности.

11.5.1. Особенности конфигурирования

Конфигурация всех Open vSwitch коммутаторов, портов, настройки поддерживаемых протоколов хранятся в собственной базе данных OVS (OVSDB). В стандартной конфигурации в OVSDB существуют следующие таблицы:

- Open_vSwitch — Схема
- Bridge
- Port
- Interface
- Flow_Table — конфигурация OpenFlow
- QoS
- Mirror
- Controller — параметры подключения к контроллеру OpenFlow
- Manager — конфигурация OVSDB
- NetFlow
- SSL
- sFlow
- IPFIX
- Flow_Sample_Collector_Set

Изначально почти все таблицы пусты, так как конфигурация отсутствует. Утилита `ovs-vsctl` предоставляет интерфейс для внесения изменений в БД. Для внесения изменений используется команда вида:

```
$ sudo ovs-vsctl <команда> <таблица> <запись> <ключ=значение>
```

Для создания программного коммутатора с именем `ovs-sw0` необходимо выполнить следующую команду:

```
$ sudo ovs-vsctl add-br ovs-sw0
```

После появляется возможность подключения ВМ. При этом ВМ, подключенные к `ovs-sw0`, будут работать в изолированной сети. Для того, чтобы предоставить им доступ к внешней сети, необходимо подключить к `ovs-sw0` в качестве порта физический интерфейс `eth0`, выполнив команду:

```
$ sudo ovs-vsctl add-port ovs-sw0 eth0
```

Для того чтобы разрешить порту `eth0` пропускать во внешнюю сеть трафик из определенных VLAN, необходимо выполнить команду:

```
$  ovs-vsctl set port eth0 trunks=10,20,30,40,50
```

Ниже представлен список вариантов команд с параметрами вызова:

- `list <таблица> <запись>`
- `find <таблица> <условие>`
- `get <таблица> <запись> <ключ=значение>`

- add <таблица> <запись> <ключ=значение>
- remove <таблица> <запись> <ключ=значение>
- clear <таблица> <запись> <ключ>
- create <таблица> <запись> <ключ=значение>
- destroy <таблица> <запись>
- wait-until <таблица> <запись> <ключ=значение>

Для просмотра записей, присутствующих в таблице, описывающей порты, выполнить команду:

```
$ sudo ovs-vsctl list port
```

Для вывода списка портов, включенных в VLAN, необходимо выполнить команду:

```
$ sudo ovs-vsctl find port tag=10
```

11.5.2. Агрегирование физических интерфейсов

Для повышения пропускной способности и уровня отказоустойчивости в Open vSwitch коммутатор могут быть включены несколько физических интерфейсов с задействованной на них агрегацией по протоколу LACP (Link Aggregation Control Protocol). Выполняется на канальном уровне путем создания объединенного интерфейса (Bonding).

Для создания объединенного интерфейса на базе физических интерфейсов `eth` и `eth1` необходимо выполнить следующую команду:

```
$ sudo ovs-vsctl add-bond ovs-sw0 bond0 eth0 eth1
```

После следует включить `lacp` на созданном объединенном интерфейсе:

```
$ sudo ovs-vsctl set port bond0 lacp=active
```

На этом настройка отказоустойчивости сетевых интерфейсов завершена.

11.5.3. Зеркалирование портов

Open vSwitch позволяет направлять копию потока трафика из одного или нескольких интерфейсов в другой. Так же он может организовать перенаправление трафика из всей VLAN в конкретный порт или наоборот. Зеркалироваться может только входящий, только исходящий или оба типа трафика. Использование такой возможности позволит вести контроль сетевого трафика, передаваемого между ВМ с целью обнаружения (предупреждения) компьютерных атак.

Пример

Зеркалирование трафика из интерфейса `vnet2`, принадлежащего одной ВМ, в специально созданный для прослушивания порт `mirror0` с типом `internal`.

```
$ sudo ovs-vsctl -- set Bridge ovs-sw0 mirrors=@m -- --id=@mirror0 get Port
     mirror0 -- --id=@vnet2 get Port vnet2 -- --id=@m create Mirror
     name=mymirror select-dst-port=@vnet2 select-src-port=@vnet2
     output-port=@mirror0
```

Краткое пояснение:

- --id=@<имя_переменной> — использование переменной;

- set Bridge ovs-sw0 mirrors=@m — создание зеркала, имя и параметры которого получаются из переменной @m;

- --id=@mirror0 get Port mirror0 -- --id=@vnet2 get Port vnet2 — в переменные @mirror0, @vnet2 записываются идентификаторы соответствующих портов.

- --id=@m create Mirror name=mymirror select-dst-port=@vnet2 select-src-port=@vnet2 output-port=@mirror0 — в переменную @m записывается идентификатор моста mymirror, которая подставляется в первой команде в переменную @m.

С помощью tcpdump, запущенного на узле, можно просматривать весь трафик ВМ:

```
$ tcpdump -i mirror0
```

Также можно организовать ретрансляцию всех пакетов, пришедших на порт eth0 или eth1 на порт eth2:

```
$ sudo ovs-vsctl -- set Bridge ovs-sw0 mirrors=@m -- --id=@eth0 get Port eth0
     -- --id=@eth1 get Port eth1 -- --id=@eth2 get Port eth2 -- --id=@m create
     Mirror name=mymirror -- select-dst-port=@eth0,@eth1
     select-src-port=@eth0,@eth1 output-port=@eth2
```

где select-dst-port — зеркалирование входящего трафика на порты @eth0 и @eth1;

select-src-port — зеркалирование исходящего трафика;

output-port — место перенаправления трафика.

Для отмены зеркалирования выполнить команду:

```
$ sudo ovs-vsctl remove Bridge ovs-sw0 mirrors mymirror
```

11.5.4. Настройка OpenNebula

Значение VLAN_ID рассчитывается в соответствии с параметром настройки oned.conf:

```
# VLAN_IDS: Пул VLAN ID для автоматического назначения VLAN_ID. Данный пул
# предназначен для сетей 802.1Q (Open vSwitch и драйверы 802.1Q).

VLAN_IDS = [
START = "2",
RESERVED = "0, 1, 4095"
]
```

Изменением данного параметра можно зарезервировать некоторые сети VLAN, и они не будут назначаться виртуальной сети. Можно также указать первый номер VLAN_ID.

При создании новой изолированной сети OpenNebula находит свободный номер VLAN_ID из пула VLAN. Этот пул является глобальным, а также совместно используется с сетевым режимом 802.1Q VLAN.

В файле /var/lib/one/remotes/vnm/OpenNebulaNetwork.conf можно откорректировать параметр настройки arp_cache_poisoning, отвечающий за подключение правила предотвращения изменения кэша ARP (ARP Cache Poisoning).

ВНИМАНИЕ! Необходимо выполнить команду onehost sync для применения файла ко всем узлам.

11.5.5. Определение сети Open vSwitch

Для создания сети Open vSwitch необходимо задать значения, приведенные в таблице 25.

Таблица 25

Параметр	Значение	Обязательный
VN_MAD	ovswitch	ДА
BRIDGE	Имя переключателя Open vSwitch, который должен использоваться	ДА
VLAN_ID	Идентификационный номер сети VLAN. Будет сгенерирован, если не указан	НЕТ

Пример

Определения сети Open vSwitch

```
NAME = "ovswitch_net" VN_MAD = "ovswitch"
BRIDGE = vbr1
VLAN_ID = 50 # optional
...
```

11.5.6. Многоканальные сети VLAN (VLAN транкинг)

VLAN транкинг поддерживается путем добавления тега VLAN_TAGGED_ID: к элементу NIC в шаблоне ВМ или шаблоне виртуальной сети. Тег позволяет указать диапазон сетей VLAN, подлежащий тегированию, например, 1,10,30,32.

11.5.7. Правила OpenFlow

11.5.7.1. MAC-спуфинг

Данные правила предотвращают выход любого трафика с порта, если был изменен MAC-адрес.

Пример

```
in_port=<PORT>,dl_src=<MAC>,priority=40000,actions=normal
```

```
in_port=<PORT>,priority=39000,actions=normal
```

11.5.7.2. IP-захват

Данные правила предотвращают выход любого трафика с порта для IPv4, если не настроен IP-адрес для ВМ.

Пример

```
in_port=<PORT>,arp,dl_src=<MAC>priority=45000,actions=drop
in_port=<PORT>,arp,dl_src=<MAC>,nw_src=<IP>,priority=46000,actions=normal
```

11.5.7.3. Черные порты

Применяется одно правило на порт.

Пример

```
tcp,dl_dst=<MAC>,tp_dst=<PORT>,actions=drop
```

11.5.7.4. ICMP-игнорирование

С помощью данной настройки можно, например, заблокировать ping-запросы к ВМ.

Пример

```
icmp,dl_dst=<MAC>,actions=drop
```

12. МОНИТОРИНГ И УЧЕТ

12.1. Мониторинг

В OpenNebula используется распределенная подсистема мониторинга. Сервис подсистемы мониторинга `collectd` выполняется на фронтальной машине и перехватывает UDP-соединения от узлов на порте 4124.

ВНИМАНИЕ! Межсетевой экран фронтальной машины должен разрешать получение UDP-пакетов с узлов по порту 4124.

При первичном запуске системы мониторинга OpenNebula подключается к узлу с помощью ssh и запускает на нем сервис, который выполняет тесты и затем отсылает собранные данные сервису `collectd` на фронтальной машине. Интервал передачи собранных данных в секундах определяется опцией `-i` в `collectd IM_MAD`. При этом системе мониторинга не требуется выполнять новые ssh-соединения для получения данных. Схема работы системы мониторинга приведена на рис. 47.

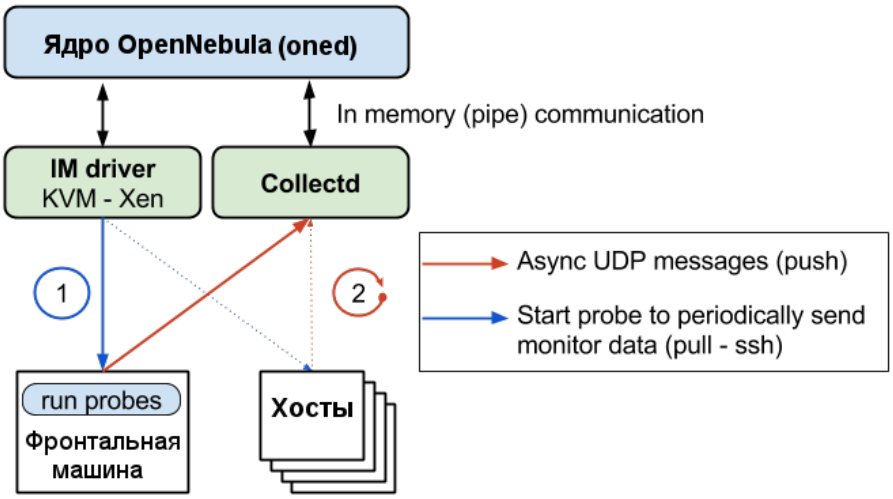

Рис. 47

В случае остановки сервиса на каком-либо из узлов, OpenNebula перестанет получать данные мониторинга и с помощью ssh перезапустит сервис на данном узле.

12.1.1. Конфигурация OpenNebula

12.1.1.1. Подключение драйвера мониторинга

Для подключения подсистемы мониторинга в файле `/etc/one/oned.conf` для KVM должен быть включен драйвер `collectd`. Описание параметров драйвера приведено в 8.3.4.

Пример

```
#-------------------------------------------------------------------------------
# Настройка диспетчера драйверов информации UDP-проталкивания KVM
```

```
# -r количество попыток при мониторинге узла
# -t количество потоков, т.е., узлов, контролируемых одновременно
#-------------------------------------------------------------------------------
IM_MAD = [
NAME = "kvm",
SUNSTONE_NAME = "KVM",
EXECUTABLE = "one_im_ssh",
ARGUMENTS = "-r 3 -t 15 kvm" ]
#-------------------------------------------------------------------------------
```

12.1.1.2. Параметры настройки мониторинга

OpenNebula позволяет настроить общее поведение всей системы мониторинга используя параметры, приведенные в таблице 26.

Таблица 26

Параметр	Описание
MONITORING_INTERVAL	Время в секундах между циклами мониторинга узла и ВМ. Должно быть больше, чем время цикла push мониторинга (см. 12.1.1.1)
HOST_PER_INTERVAL	Количество узлов, контролируемых в каждый интервал времени

12.1.2. Отчет системы мониторинга

Каждый интервал времени, заданный опцией monitoring_push_cycle, OpenNebula получает данные мониторинга по каждой ВМ и узлу.

Пример

```
Tue May 24 16:21:47 2016 [Z0][InM][D]: Host thost087 (0) successfully monitored.
Tue May 24 16:21:47 2016 [Z0][VMM][D]: VM 0 successfully monitored: STATE=a
     CPU=0.0 MEMORY=113404 NETRX=648 NETTX=398
Tue May 24 16:22:07 2016 [Z0][InM][D]: Host thost087 (0) successfully monitored.
Tue May 24 16:22:07 2016 [Z0][VMM][D]: VM 0 successfully monitored: STATE=a
     CPU=0.0 MEMORY=113516 NETRX=648 NETTX=468
Tue May 24 16:22:11 2016 [Z0][VMM][D]: VM 0 successfully monitored:
     DISK_SIZE=[ID=0, SIZE=27] DISK_SIZE=[ID=1,SIZE=1]
Tue May 24 16:22:27 2016 [Z0][InM][D]: Host thost087 (0) successfully monitored.
Tue May 24 16:22:27 2016 [Z0][VMM][D]: VM 0 successfully monitored: STATE=a
     CPU=0.0 MEMORY=113544 NETRX=648 NETTX=468
```

При этом если в oned.log выполняется периодический активный мониторинг узла каждый интервал времени, заданный параметром MONITORING_INTERVAL, то мониторинг работает некорректно.

Пример

```
Tue May 24 16:24:23 2016 [Z0][InM][D]: Monitoring host thost087 (0)
Tue May 24 16:25:23 2016 [Z0][InM][D]: Monitoring host thost087 (0)
Tue May 24 16:26:23 2016 [Z0][InM][D]: Monitoring host thost087 (0)
```

12.1.3. Настройка и расширение

12.1.3.1. Корректировка интервалов мониторинга

Для точной настройки системы мониторинга OpenNebula необходимо откорректировать опцию `-i` функции `collectd IM_MAD` (цикл push мониторинга). В работе системы мониторинга могут происходить сбои из-за недостаточной пропускной способности БД. OpenNebula записывает в БД данные мониторинга объемом приблизительно 4 КБ для каждой ВМ. При большом количестве ВМ и коротком цикле push мониторинга OpenNebula не сможет записать такой объем данных в БД.

12.1.3.2. Тесты

Тесты (probes) представляют собой специальные программы, которые обеспечивают получение контрольных показателей мониторинга. Тесты определяются для каждого гипервизора и находятся по адресу `/var/lib/one/remotes/im/kvm-probes.d`.

Необходимо выполнять синхронизацию тестов мониторинга на узлах с помощью команды `onehost sync`.

12.2. Учет использования ВМ

Инструмент для ведения учета позволяет получать данные от гипервизора о потреблении ресурсов виртуальных машин.

12.2.1. Получение информации о потреблении ресурсов с помощью CLI

Для вывода информации о потреблении ресурсов виртуальных машин используется команда `oneacct`. Описание параметров команды приведено в таблице 27.

Таблица 27

Параметр	Описание
`-s, --start TIME`	Предоставить информацию за период после указанной даты. Может использоваться совместно с командой `-e, --end TIME`
`-e, --end TIME`	Предоставить информацию за период до указанной даты. Может использоваться совместно с командой `-s, --start TIME`
`-u, --userfilter user`	Имя или идентификатор пользователя для фильтрации предоставляемой информации
`-g, --group group`	Имя или идентификатор группы для фильтрации предоставляемой информации
`-H, --host HOST`	Имя или идентификатор узла для фильтрации предоставляемой информации

Окончание таблицы 27

Параметр	Описание
`--xpath XPATH_EXPRESSION`	Выражение `Xpath` для фильтрации предоставляемой информации. Например, `oneacct --xpath 'HISTORY[ETIME>0]'`
`-x, --xml`	Предоставить информацию в формате `xml`
`-j, --json`	Предоставить информацию в формате `json`
`--split`	Вывести информацию в таблице отдельно для каждой ВМ
`-v, --verbose`	Запустить в режиме Verbose
`-h, --help`	Вывести описание параметров команды
`-V, --version`	Отобразить версию и информацию об авторских правах
`--describe`	Вывести описание столбцов
`-l, --list x,y,z`	Выбрать столбцы для отображения при выполнении перечня команд
`--csv`	Вывести таблицы в формате `csv`
`--user name`	Имя пользователя для подключения к OpenNebula
`--password password`	Пароль пользователя для аутентификации в OpenNebula
`--endpoint endpoint`	URL-адрес OpenNebula XML-RPC front-end

Дата записывается в формате `month/day/year hour:minute:second` (месяц/день/год час:минута:секунда), либо в любом другом аналогичном формате, например, `month/day hour:minute` (месяц/день час:минута).

Для внедрения данного инструмента в другие системы могут использоваться метки `-j`, `-x` или `--csv`, обеспечивающие вывод информации в простом машиночитаемом формате.

С помощью команды `oneacct` можно отобразить сохраненные записи для отдельной ВМ. Для одной ВМ можно получить записи о потреблении ресурсов для каждого действия перемещения/остановки. Новая запись также будет создана в случае изменения размера или подключения диска/сетевой карты.

Каждая запись содержит полную информацию о ВМ, включая ее контрольную информацию. По умолчанию отчеты создаются только о потреблении сетевых ресурсов. Более подробная информация приведена в 12.2.3.

При фильтрации результатов с помощью параметров `-s` (начало периода отчета) и/или `-e` (окончание периода отчета) отображаются все записи в хронологическом порядке, которые были активны в заданный период времени.

Пример

Запрос информации о ВМ, работавшей с 1 мая по 1 июня

```
$ sudo oneacct -s 05/01 -e 06/01
Showing active history records from 2016-05-01 00:00:00 +0200 to 2016-06-02
```

```
      00:00:00
+0200

# User 0
```

В приведенном выше примере в записи будет отображена полная история, а также общее потребление сетевых ресурсов. Будут отображены данные о потреблении только за май.

Также необходимо иметь в виду, что активные хронологически записи с END_TIME обновляют свою контрольную информацию при каждом мониторинге ВМ. Если ВМ была отключена, перемещена или остановлена, END_TIME устанавливается, и собираемая контрольная информация замораживается. Окончательные значения отражают общие результаты для суммарных атрибутов, например, NETRX/NETTX.

Пример

Получение всей имеющей учетной информации

```
VID  HOSTNAME  ACTION         REAS  START_TIME     END_TIME       MEMORY  CPU
NETRX  NETTX   DISK

13   host01    nic-attach     user  05/17 17:10:57 05/17 17:12:48  256M    0.1
19.2K  15.4K   8G

13   host01    nic-detach     user  05/17 17:12:48 05/17 17:13:48  256M    0.1
36.9K  25K     8G

13   host01    nic-attach     user  05/17 17:13:48 05/17 17:14:54  256M    0.1
51.2K  36.4K   8G

13   host01    nic-detach     user  05/17 17:14:54 05/17 17:17:19  256M    0.1
79.8K  61.7K   8G

13   host01    nic-attach     user  05/17 17:17:19 05/17 17:17:27  256M    0.1
79.8K  61.7K   8G

13   host01    terminate-hard user  05/17 17:17:27 05/17 17:37:52  256M    0.1
124.6K 85.9K   8G

14   host02    nic-attach     user  05/17 17:38:16 05/17 17:40:00  256M    0.1
16.5K  13.2K   8G
```

```
14   host02    poweroff       user  05/17 17:40:00 05/17 17:53:40  256M    0.1
38.3K  18.8K   8G

14   host02    terminate-hard user  05/17 17:55:55 05/18 14:54:19  256M    0.1
1M 27.3K      8G
```

Описание значений столбцов из примера приведено в таблице 28.

Таблица 28

Столбец	Описание
VID	Идентификатор виртуальной машины
HOSTNAME	Имя узла
ACTION	Действие виртуальной машины, на основании которого создана новая запись
REASON	Причина изменения состояния ВМ: - none: ВМ до сих пор работает; - error: возникновение ошибки ВМ; - user: действие ВМ, инициированное пользователем
START_TIME	Время начала
END_TIME	Время окончания
MEMORY	Выделенная память. Это запрошенная память, а не потребляемая
CPU	Число ЦП. Это запрошенное количество разделяемых ЦП узла, а не потребление ЦП
NETRX	Данные, полученные по сети
NETTX	Данные, отправленные по сети

Пример

Получение учетной информации для определенного пользователя

```
$ oneacct -u 0 --split
# User 0

VID  HOSTNAME    ACTION    REAS   START_TIME      END_TIME     MEMORY   CPU
NETRX  NETTX     DISK
12   host01                     05/09 19:20:42  19:35:23             1
29.8M  638.8K    0K

VID  HOSTNAME    ACTION    REAS   START_TIME      END_TIME     MEMORY   CPU
NETRX  NETTX     DISK
14   host02                     17:38:16
```

```
16.5K            8G
14   host02
38.3K            8G
14   host02
               8G

VID  HOSTNAME    ACTION    REAS   START_TIME      END_TIME        MEMORY   CPU
NETRX  NETTX     DISK
29   host02      none      none   05/27 17:09:28  -               256M     1
2.4M  1.3K       10G
```

Если используется параметр `--csv` для вывода в формате CSV, то будет отображен заголовок с названием каждого столбца, а затем данные.

Пример

```
$ oneacct --csv
UID,VID,HOSTNAME,ACTION,REASON,START_TIME,END_TIME,MEMORY,CPU,NETRX,NETTX,DISK
0,12,host01,none,user,05/09 19:20:42,05/09 19:35:23,1024M,1,29.8M,638.8K,0K
0,13,host01,nic-attach,user,05/17 17:10:57,05/17 17:12:48,256M,0.1,19.2K,15.4K,
     8G
0,13,host01,nic-detach,user,05/17 17:12:48,05/17 17:13:48,256M,0.1,36.9K,25K,8G
0,13,host01,nic-attach,user,05/17 17:13:48,05/17 17:14:54,256M,0.1,51.2K,36.4K,
     8G
0,13,host01,nic-detach,user,05/17 17:14:54,05/17 17:17:19,256M,0.1,79.8K,61.7K,
     8G
0,13,host01,nic-attach,user,05/17 17:17:19,05/17 17:17:27,256M,0.1,79.8K,61.7K,
     8G
0,13,host01,terminate-hard,user,05/17 17:17:27,05/17 17:37:52,256M,0.1,124.6K,
     85.9K,8G
0,14,host02,nic-attach,user,05/17 17:38:16,05/17 17:40:00,256M,0.1,16.5K,13.2K,
     8G
0,14,host01,poweroff,user,05/17 17:40:00,05/17 17:53:40,256M,0.1,38.3K,18.8K,8G
0,14,host02,terminate-hard,user,05/17 17:55:55,05/18 14:54:19,256M,0.1,1M,27.3K,
     8G
0,29,host02,none,none,05/2717:09:28,-,256M,1,2.4M,1.3K,10G
```

Если используется параметр `-x`, то будет выведен XML-файл, который определен xsd.

Пример

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<xs:schema  xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified"
targetNamespace="http://opennebula.org/XMLSchema"
xmlns="http://opennebula.org/
XMLSchema">

<xs:element name="HISTORY_RECORDS">
   <xs:complexType>
    <xs:sequence maxOccurs="1" minOccurs="1">
          <xs:element ref="HISTORY" maxOccurs="unbounded" minOccurs="0"/>
    </xs:sequence>
   </xs:complexType>
</xs:element>

<xs:element name="HISTORY">
  <xs:complexType>
   <xs:sequence>
        <xs:element name="OID" type="xs:integer"/>
        <xs:element name="SEQ" type="xs:integer"/>
        <xs:element name="HOSTNAME" type="xs:string"/>
        <xs:element name="HID" type="xs:integer"/>
        <xs:element name="CID" type="xs:integer"/>
        <xs:element name="STIME" type="xs:integer"/>
        <xs:element name="ETIME" type="xs:integer"/>
        <xs:element name="VM_MAD" type="xs:string"/>
        <xs:element name="TM_MAD" type="xs:string"/>
        <xs:element name="DS_ID" type="xs:integer"/>
        <xs:element name="PSTIME" type="xs:integer"/>
        <xs:element name="PETIME" type="xs:integer"/>
        <xs:element name="RSTIME" type="xs:integer"/>
        <xs:element name="RETIME" type="xs:integer"/>
        <xs:element name="ESTIME" type="xs:integer"/>
        <xs:element name="EETIME" type="xs:integer"/>

        <!-- REASON values:
          NONE = 0 History record is not closed yet
          ERROR = 1 History record was closed because of an error
```

```
USER = 2 History record was closed because of a user action
-->
<xs:element name="REASON" type="xs:integer"/>

<!-- ACTION values:

  NONE_ACTION                =0
  MIGRATE_ACTION             =1
  LIVE_MIGRATE_ACTION        =2
  SHUTDOWN_ACTION            =3
  SHUTDOWN_HARD_ACTION       =4
  UNDEPLOY_ACTION            =5
  UNDEPLOY_HARD_ACTION       =6
  HOLD_ACTION                =7
  RELEASE_ACTION             =8
  STOP_ACTION                =9
  SUSPEND_ACTION             =10
  RESUME_ACTION              =11
  BOOT_ACTION                =12
  DELETE_ACTION              =13
  DELETE_RECREATE_ACTION     =14
  REBOOT_ACTION              =15
  REBOOT_HARD_ACTION         =16
  RESCHED_ACTION             =17
  UNRESCHED_ACTION           =18
  POWEROFF_ACTION            =19
  POWEROFF_HARD_ACTION       =20
  DISK_ATTACH_ACTION         =21
  DISK_DETACH_ACTION         =22
  NIC_ATTACH_ACTION          =23
                             =24
  DISK_SNAPSHOT_CREATE_ACTION =25
  DISK_SNAPSHOT_DELETE_ACTION =26
  TERMINATE_ACTION           =27
  TERMINATE_HARD_ACTION      =28
-->
<xs:element name="ACTION" type="xs:integer"/>
```

```
        <xs:element name="VM">
          <xs:complexType>
             <xs:sequence>
               <xs:element name="ID" type="xs:integer"/>
               <xs:element name="UID" type="xs:integer"/>
               <xs:element name="GID" type="xs:integer"/>
               <xs:element name="UNAME" type="xs:string"/>
               <xs:element name="GNAME" type="xs:string"/>
               <xs:element name="NAME" type="xs:string"/>
               <xs:element name="PERMISSIONS" minOccurs="0" maxOccurs="1">
                    <xs:complexType>
                      <xs:sequence>
                           <xs:element name="OWNER_U" type="xs:integer"/>
                           <xs:element name="OWNER_M" type="xs:integer"/>
                           <xs:element name="OWNER_A" type="xs:integer"/>
                           <xs:element name="GROUP_U" type="xs:integer"/>
                           <xs:element name="GROUP_M" type="xs:integer"/>
                           <xs:element name="GROUP_A" type="xs:integer"/>
                           <xs:element name="OTHER_U" type="xs:integer"/>
                           <xs:element name="OTHER_M" type="xs:integer"/>
                           <xs:element name="OTHER_A" type="xs:integer"/>
                       </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="LAST_POLL" type="xs:integer"/>
        <!-- STATE values,
        seehttp://docs.opennebula.org/stable/user/references/vm_states.html
        -->
        <xs:element name="STATE" type="xs:integer"/>
        <!-- LCM_STATE values, this sub-state is relevant only when
STATE is ACTIVE (4)
        seehttp://docs.opennebula.org/stable/user/references/vm_states.html
        -->
        <xs:element name="LCM_STATE" type="xs:integer"/>
        <xs:element name="PREV_STATE" type="xs:integer"/>
        <xs:element name="PREV_LCM_STATE" type="xs:integer"/>
        <xs:element name="RESCHED" type="xs:integer"/>
        <xs:element name="STIME" type="xs:integer"/>
```

```
          <xs:element name="ETIME" type="xs:integer"/>
          <xs:element name="DEPLOY_ID" type="xs:string"/>
          <xs:element name="MONITORING">
          <!--
                  minOccurs="0" maxOccurs="1"/><xs:complexType>
                  <xs:all>
                        <- Percentage of 1 CPU consumed (two fully
consumed cpu is 200) ->
                        <xs:element name="CPU" type="xs:decimal"
                        minOccurs="0" maxOccurs="1"/><- MEMORY
                      consumption in kilobytes ->
                        <xs:element name="MEMORY" type="xs:integer"
                        minOccurs="0" maxOccurs="1"/><- NETTX: Sent
                      bytes to the network ->
                        <xs:element name="NETTX" type="xs:integer"
                        minOccurs="0" maxOccurs="1"/><- NETRX: Received
                      bytes from the network ->
                        <xs:element name="NETRX" type="xs:integer"
                  </xs:all>
                  </xs:complexType>
           -->
           </xs:element>
           <xs:element name="TEMPLATE" type="xs:anyType"/>
           <xs:element name="USER_TEMPLATE" type="xs:anyType"/>
           <xs:element name="HISTORY_RECORDS">
           </xs:element>
           <xs:element name="SNAPSHOTS" minOccurs="0" maxOccurs="unbounded">
              <xs:complexType>
                <xs:sequence>
<xs:element name="DISK_ID" type="xs:integer"/>
<xs:element name="SNAPSHOT" minOccurs="0"
          maxOccurs="unbounded">
                           <xs:complexType>
                                 <xs:sequence>
                                   <xs:element name="ACTIVE" type=
"xs:string" minOccurs="0" maxOccurs="1"/>
                                   <xs:element name="CHILDREN" type=
"xs:string" minOccurs="0" maxOccurs="1"/>
```

```
                                        <xs:element name="DATE" type=
"xs:integer"/>
                                        <xs:element name="ID" type=
"xs:integer"/>
                                        <xs:element name="NAME" type=
"xs:string" minOccurs="0" maxOccurs="1"/>
                                        <xs:element name="PARENT" type=
"xs:integer"/>
                                        <xs:element name="SIZE" type=
"xs:integer"/>
                                    </xs:sequence>
                                </xs:complexType>
                            </xs:element>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
  </xs:sequence>
 </xs:complexType>
</xs:element>
</xs:schema>
```

12.2.2. Получение информации о потреблении ресурсов с помощью Sunstone

Информация о потреблении ресурсов доступна для режимов просмотра User и Admin. В режиме просмотра Cloud можно получать доступ к учетной информации в панели мониторинга, при этом администратору группы она доступна в разделе пользователей. Окно с информацией о потреблении ресурсов приведено на рис. 48.

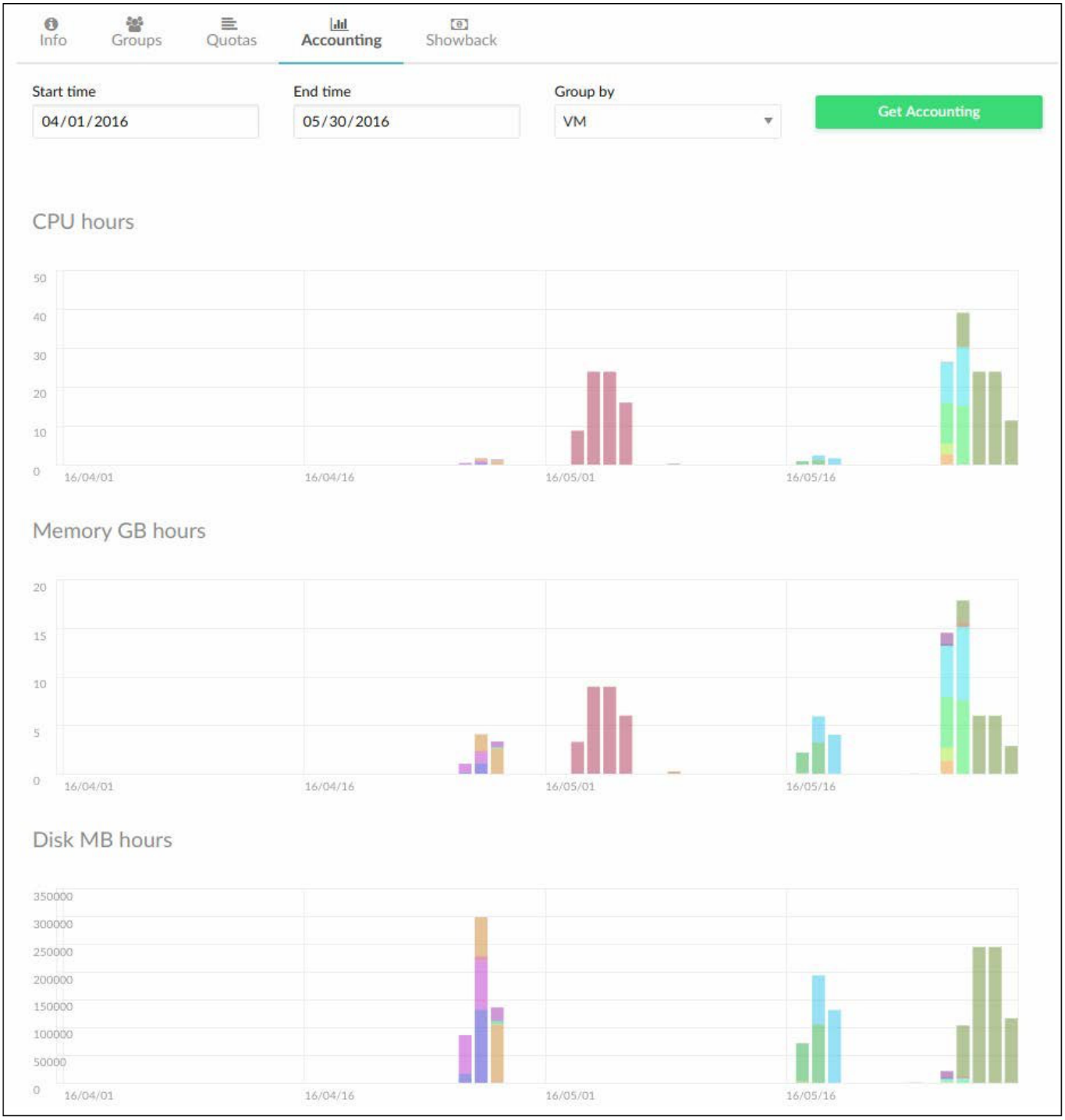

Рис. 48

12.2.3. Тонкая настройка и расширение

Существует два типа получаемой информации о потреблении :

1) текущие значения — например, VM/CPU или VM/MEMORY показывают данные о потреблении памяти, полученные в последний раз;

2) суммарные значения — например, VM/NETRX и VM/NETTX отображают общее потребление сетевых ресурсов с момента начала ведения учета.

12.3. Логирование

OpenNebula обеспечивает ведение журналов для большинства ресурсов. Поддерживается три системы регистрации: файловая система регистрации, регистрация системных журналов и регистрация в стандартный поток ошибок.

При использовании файловой регистрации OpenNebula ведет отдельные файлы журналов для каждого активного компонента, при этом все они хранятся в каталоге `/var/log/one`. Пользователи и администраторы могут иметь доступ к некоторым сообщениям об ошибке из CLI или Sunstone.

12.3.1. Настройка системы регистрации

Система регистрации может быть изменена в файле `/etc/one/oned.conf` путем корректировки раздела `LOG`. Возможны изменения следующих параметров:

- `SYSTEM` — тип системы регистрации. Принимаемые значения:
  - `syslog` — регистрация системных журналов;
  - `file` — файловая система регистрации, используется по умолчанию;
  - `std`— регистрация в стандартный поток ошибок;
- `DEBUG_LEVEL` — определяющий уровень детализации регистрации.

Для планировщика система регистрации может быть изменена таким же способом в файле `/etc/one/sched.conf`.

12.3.2. Регистрационные ресурсы

Существуют различные регистрационные ресурсы, соответствующие различным компонентам OpenNebula:

- сервис ONE — core-компонент OpenNebula, выгружает всю регистрационную информацию в файл `/var/log/ one/oned.log`. Детализация регулируется параметром `DEBUG_LEVEL` в файле `/etc/one/oned.conf`. По умолчанию команда `one start up` будет резервировать последний файл `oned.log` используя текущее время, например, `oned.log.20121011151807`. Альтернативный вариант — данный ресурс может загружаться в системный журнал;
- планировщик — вся информация планировщика выгружается в файл `/var/log/one/sched.log`. Данный ресурс может загружаться в системный журнал;
- виртуальные машины — информация, относящаяся к ВМ, будет выгружаться в файл журнала `/var/log/one/<vmid>.log`.

12.3.3. Регистрационный формат

Сообщения OpenNebula для файловой системы регистрации имеют следующую структуру:

```
date [Z<zone_id>][module][log_level]: message body
```

Сообщения OpenNebula для регистрации системных журналов имеют следующую структуру:

```
date hostname process[pid]: [Z<zone_id>][module][log_level]: message
```

Модулем является любой из внутренних компонентов OpenNebula: `VMM`, `ReM`, `TM` и т.д. Параметр `log_level` представляет собой отдельный символ, указывающий уровень регистрации: `I` для информации, `D` для отладки и т.д.

При регистрации системных журналов OpenNebula также будет регистрировать события ВМ.

Пример

```
date hostname process[pid]: [VM id][Z<zone_id>][module][log_level]: message
```

Для регистрации типа `stderr`, для `oned` и событий ВМ формат будет следующим:

```
date [Z<zone_id>][module][log_level]: message
date [VM id][Z<zone_id>][module][log_level]: message
```

12.3.4. Ошибки виртуальной машины

Ошибки ВМ могут проверяться с помощью команды `onevm show`.

Пример

```
$ sudo onevm show 0
VIRTUAL MACHINE 0 INFORMATION ID : 0
NAME : one-0
USER : brestadmin
GROUP : brestadmins
STATE : ACTIVE
LCM_STATE : PROLOG_FAILED
START TIME : 07/19 17:44:20
END TIME : 07/19 17:44:31
DEPLOY ID : -

VIRTUAL MACHINE MONITORING NET_TX : 0
NET_RX : 0
USED MEMORY : 0
USED CPU : 0

VIRTUAL MACHINE TEMPLATE CONTEXT=[
FILES=/tmp/some_file,
 TARGET=hdb ]
CPU=0.1
ERROR=[
 MESSAGE="Error excuting image transfer script: Error copying /tmp/some_file /
```

```
    to /var/lib/one/0/images/isofiles",
 TIMESTAMP="Tue Jul 19 17:44:31 2011" ]
MEMORY=64
 NAME=one-0 VMID=0
```

Ошибка, приведенная в примере, указывает на то, что было невозможно скопировать файл; возможно, что он не существует. Также можно проверить файлы журналов для ВМ в `/var/log/one/<vmid>.log`.

12.3.5. Ошибки узла

Ошибки узла можно проверить командой `onehost show`.

Сообщение об ошибке появляется в поле `ERROR` мониторинга. Дополнительную информацию можно просмотреть в файле `/var/log/one/oned.log`.

13. ОГРАНИЧЕНИЯ ПРИ ИСПОЛЬЗОВАНИИ

Существует ряд ограничений при использовании ПК СВ в условиях мандатного управления доступом:

1) подключение USB-устройства с компьютера пользователя по протоколу TCP (SASL+Kerberos) с использованием программ virt-viewer и remote-viewer возможна только в рамках одного домена, т.е. компьютер пользователя, узел виртуализации и ВМ должны быть в одном домене;

2) в качестве сетевого адаптера ВМ не может быть использовано устройство virtio;

3) при разворачивании ВМ с ОС семейства Windows не рекомендуется использовать виртуальные диски на шине virtio;

4) не рекомендуется подключать к ВМ различные физические устройства (видеокарта, USB-токен, источник бесперебойного питания по USB-шине и т.д.), т. к. имеется возможность несанкционированной записи на данные устройства;

5) не рекомендуется устанавливать пакет `spice-vdagent` в ВМ с ненулевой мандатной меткой.

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

БД — база данных
ВМ — виртуальная машина
ГИП — графический интерфейс пользователя (GUI)
ЕПП — единое пространство пользователей
ОС — операционная система
ОС СН — операционная система специального назначения «Astra Linux Special Edition»
ПК СВ — программный комплекс «Средства виртуализации «Брест»
ПО — программное обеспечение
СЗИ — средства защиты информации
ФС — файловая система
ЦОХД — центр обработки и хранения данных
ЦП — центральный процессор

ALD — Astra Linux Directory (единое пространство пользователей)
CHAP — Challenge Handshake Authentication Protocol (протокол аутентификации с косвенным согласованием, предусматривающим передачу не самого пароля пользователя, а косвенных сведений о нём)
CephFS — Ceph File System (POSIX-совместивамя файловая система на базе кластера Ceph)
CLI — Command Line Interface (интерфейс командной строки)
CLVM — (кластерное управление логическими томами)
CPU — Central Processing Unit (центральный процессор)
DRBD — Distributed Replicated Block Device (распределённое реплицируемое блочное устройство)
I/O MMU — Input/Output Memory Management Unit (блок управления памятью для операций ввода-вывода)
IPMI — Intelligent Platform Management Interface (интерфейс, обеспечивающий автономный мониторинг, восстановление и журналирование работы функций, встроенных непосредственно в аппаратное и микропрограммное обеспечения серверных платформ)
iSCSI — Internet Small Computer System Interface (протокол на базе TCP/IP для взаимодействия и управления системам хранения данных, серверов и клиентов)
KVM — Kernel-based Virtual Machine (программное решение, обеспечивающее виртуализацию в среде Linux на платформе, которая поддерживает аппаратную виртуализацию на базе Intel VT (Virtualization Technology) либо AMD SVM (Secure

Virtual Machine))

LIO — Linux-IO Target (программный комплекс)

LUN — Logical Unit Number (номер объекта внутри цели target)

LV — Logical Volume (логический том)

LVM — Logical Volume Manager (менеджер логических томов)

MDS — Metadata Server (сервер кластера Ceph, отслеживающий метаданные файловой иерархии для CephFS)

MON — Monitor (демон, отслеживающий состояние кластера Ceph)

NBD — Network Block Device (сетевое блочное устройство)

NFS — Network File System (сетевая файловая система)

OSD — Object Storage Device (основное устройстве хранения объектов Ceph, обычно связанное с одним физическим диском, в котором хранятся фактические данные пользователя)

OVS — Open vSwitch (программный многоуровневый коммутатор для работы в гипервизорах и на компьютерах с виртуальными машинами)

OVSDB — база данных OVS

PCI — Peripheral component interconnect (шина ввода-вывода для подключения периферийных устройств к материнской плате компьютера)

QEMU — Quick Emulator (средства эмуляции аппаратного обеспечения)

RADOS — Reliable Autonomic Distributed Object Store (хранилище, отвечающее за хранение объектов кластера Ceph независимо от их типа данных)

RBD — Rados block device (блочное хранилище кластера Ceph, которое может отображаться, форматироваться и монтироваться в точности как любой другой диск в сервере)

RDM — Raw Device Mapping (используется для прямого подключения к виртуальной машине существующих блочных устройств в узлах)

SCSI — Small Computer System Interface (системный интерфейс малых компьютеров)

SPICE — Simple Protocol for Independent Computing Environments (простой протокол для независимой вычислительной среды)

SSH — Secure Shell Protocol (протокол защищенной передачи информации)

TLS — Transport Layer Security (безопасность транспортного уровня)

TM — Transfer Manager (драйвер передачи)

UDP — User Datagram Protocol (протокол пользовательских дейтаграмм)

UUID — Universally Unique Identifier (универсальный уникальный идентификатор)

VCPU — Virtual Central Processing Unit (виртуальный центральный процессор)

VDI — Virtual Desktop Infrastructure (инфраструктура виртуальных рабочих столов)

VLAN — Virtual Local Area Network (виртуальная локальная вычислительная сеть)

VNC — Virtual Network Computing (система удалённого доступа к рабочему столу компьютера)

VXLAN — Virtual Extensible Local Area Network (виртуальная масштабируемая локальная вычислительная сеть)

Лист регистрации изменений									
Изм.	Номера листов (страниц)				Всего листов (страниц) в документе	Номер документа	Входящий номер сопро-водительного документа и дата	Подпись	Дата
	изменен-ных	заменен-ных	новых	аннулиро-ванных					
1		Все				РДЦП.02-18			21.08.18
2		Все				РДЦП.03-19			11.11.19
3		Все				РДЦП.02-20			06.07.20
4		Все				РДЦП.01-21			05.02.21