Министерство образования и науки Российской Федерации
ФГАОУ ВПО «УрФУ имени первого Президента России Б. Н. Ельцина»
Институт радиоэлектроники и информационных технологий - РтФ
Департамент информационных технологий и автоматики

Автоматизированное развертывание кластера с Intel MPI.

ОТЧЕТ

по проекту по модулю

Преподаватель: Лукач Юрий Саулович
Студент: Сухоплюев Илья Владимирович
Группа: РИ-440001

Екатеринбург
2018

Содержание

Введение

Задача данного проекта по модулю возникла в рамках соревнования по высокопроизводительному программированию Asia Supercomputer Community (ASC) [1]. В рамках этого соревнования требуется собрать и настроить кластерную систему, после чего решить с ее использованием ряда задач: тестирование полученной системы на производительность; оптимизации решений, предоставленных организаторами и решение прикладных задач с нуля.

Чтобы уделить решению задач наибольшее время нужно минимизировать, на сколько это возможно, время настройки кластерной системы. В момент написания данного проекта пока не известна вся конфигурация кластерной системы, поэтому основной задачей является автоматическое разворачивание n-узловой кластерной системы на виртуальных машинах с операционной системой CentOS7 и установленной MPI-системой. Коректность работы MPI будем проверять запуском тестовых програм.

Число узлов n в работе подразумеваем не большим (не более 10–20). Хотя вполне возможно, что полученное решение будет позволять развернуть большее количество узлов, либо требовать небольшую доработку полученного решения. Например, расширение адресного пространства. Тем не менее данные вопросы будут опущены по предположению, что целевая кластерная система будет содержать порядка 8-10 узлов.

Подзадачи:

— Автоматическая установка CentOS7;
— Настройка сетевой конфигурации;
— Установка и настройка Network File System (NFS)[2];
— Установка и настройка Intel MPI.

1 Исспользуемые технологии

В данной разделе рассматриваются используемые программы и технологии, используемые в данной работе и цель их использования.

1.1 Kernel-based Virtual Machine (KVM) и libvirt

Кластерная система – это группа компьютеров, объединенных высокоскоростными каналами связи, которая будет представлять для пользователя единый аппаратный ресурс.

Для того, чтобы полноценно протестировать установку и настройку такой системы, нам потребовалось несколько компьютеров и коммутатор(-ов) соединяющих их между собой, что не очень целесообразно для тестовой задачи. Для этого в настоящий момент времени удобно создавать такие системы с использованием виртуальных машин и виртуальных сетей, которые будут эмулировать поведение реального оборудования.

В Ubuntu рекомендуется использовать гипервизор (менеджер виртуальных машин) KVM[3] и библиотеку libvirt[4] в качестве инструментария управления им.

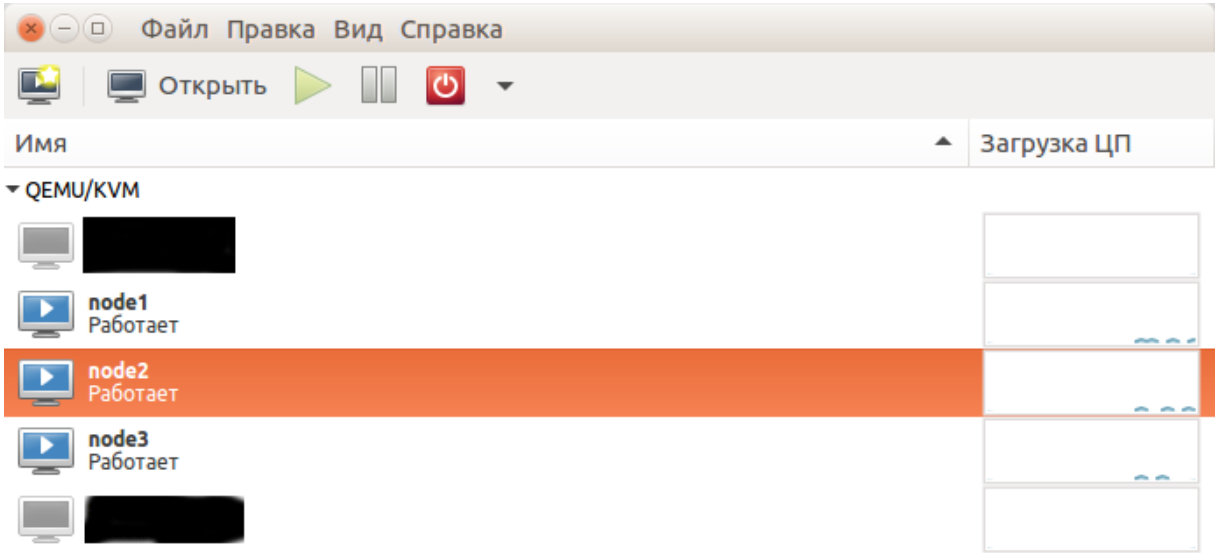

Рисунок 1.1 — Virt-manager – GUI-интерфейс для управления виртуальными машинами

Также для простоты использования к данным инструментам можно поставить virt-manager(Рис. 1.1) – GUI-интерфейс, позволяющий легко создавать виртуальные машины и управлять ими.

1.2 Операционная система – CentOS7

Каждый узел (или нода) в нашей кластерной системе будет представлять отдельный компьютер с установленной системой CentOS7[5], основанной на пакетном менеджере yum и впервые выпущенной 14 мая 2004 года. Эта операционная система основана на коммерческом проекте Red Hat Enterprise Linux[6] и совместима с ним.

Данная операционная система позволяет автоматизировать установку с помощью kickstart-файла[7], описывающего выбор вариантов, которые предлагаются перед установкой системы.

1.3 Network File System (NFS)

Network File System (NFS) — протокол сетевого доступа к файловым системам, первоначально разработан Sun Microsystems в 1984 году. За основу был взят протокол вызова удалённых процедур (RPC). Позволяет подключать (монтировать) удалённые файловые системы через сеть.

В нашей задаче данная технология потребуется для запуска MPI-программ, которые подразумевают что запускаемая программа доступна на всех узлах системы. Конечно, для этих целей можно просто провести копирование исполняемого файла на все узлы, но постоянная работа таким образом является утомительной, что в конечном итоге повлечет за собой создание скрипта или другого механизма по автоматизации этого процесса. Чтобы избежать этого мы создадим каталог, доступный через с помощью NFS всем узлам.

1.4 MPI (Intel MPI)

Message Passing Interface (MPI, интерфейс передачи сообщений) – стандарт программного интерфейса (API) для передачи информации, ко-

торый позволяет обмениваться сообщениями между процессами, выполняющими одну задачу. Разработан Уильямом Гроуппом, Эвином Ласком (англ.) и другими[8].

Существует множество реализаций MPI. В данной работе мы будем разворачивать Intel MPI[9], в предположении что эта реализация наибыстрейшая.

1.5 Архитектура кластера

Обсудим общую схему создаваемой системы. Краткое описание можно увидеть на рисунке 1.2. В текущем примере мы будем создавать кластер из 3-х узлов, представляющих виртуальные машины с CentOS7, соединенных двумя сетями.

Первая сеть будет необходима для доступа в сеть Интернет и скачивания требуемых rpm-пакетов. Вторая сеть будет использоваться для взаимодействия между узлами и будет изолированной, поскольку протокол NFS подразумевает подключение по доверенной сети.

Конфигурация первой сети:

```
IP-пространство: 192.168.127.0/24
Шлюз по-умолчанию: 192.168.127.1 (Рабочая станция, NAT до Интернета)
DHCP-pool: 192.168.127.128 - 192.168.127.254
```

Конфигурация второй сети:

```
статическое IP-пространство: 10.20.30.0/24
```

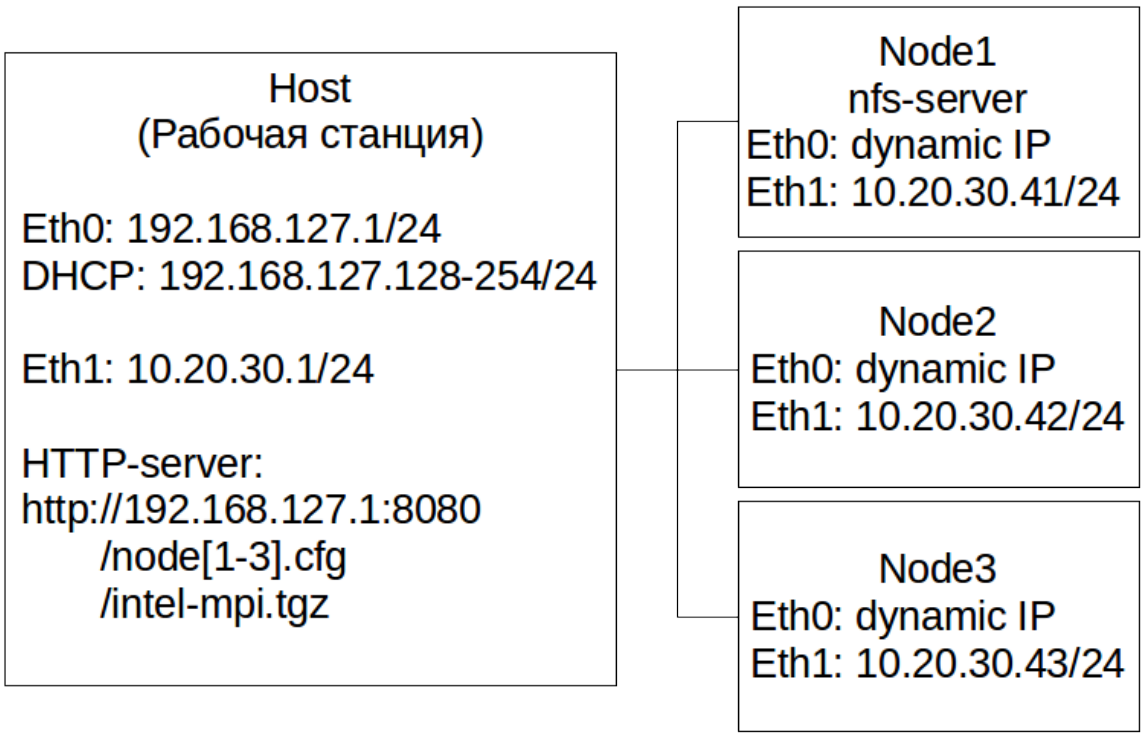

Рисунок 1.2 — Общая схема разворачиваемой системы

Все узлы будут иметь названия хоста (node1/node2/node3). node1-узел будет главным, а также будет размещать на себе NFS-сервер и предоставлять папку /nfs двум другим узлам, которые будут NFS-клиентами.

Для того, чтобы обеспечить автоматическую установку системы CentOS, потребуется обеспечить доступ к kickstart-файлам, которые будут сгенерированны для каждого компьютера и располагаться на локальном HTTP-сервере. Там же будет располагаться архив mpi.tgz с Intel MPI-установочником, который будет установлен после основной установки системы.

2 Создание кластерной системы

В этой главе рассматриваются пошаговые действия при подготовке автоматизированной установки кластерной системы из 3-ех нод, описанной в предыдущей главе.

2.1 Создание виртуальных сетей

Для создания новой виртуальной сети в virt-manager перейдем в "Правка"→"Свойства подключения"→"Виртуальные сети"и в открывшемся диалоговом окне найдем кнопку по созданию сети. Конфигурации сетей изображены на Рис. 2.1 и 2.2.

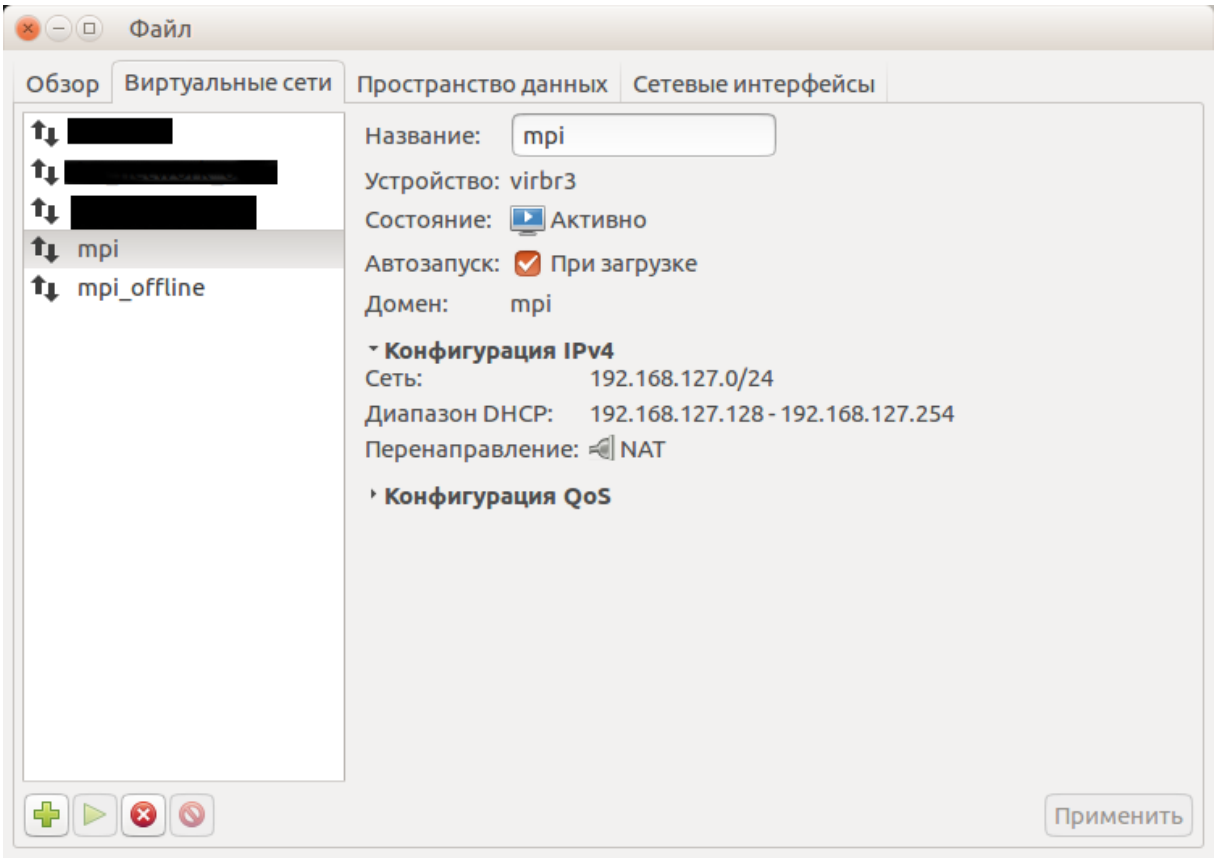

Рисунок 2.1 — Конфигурация первой сети для доступа в Интернет

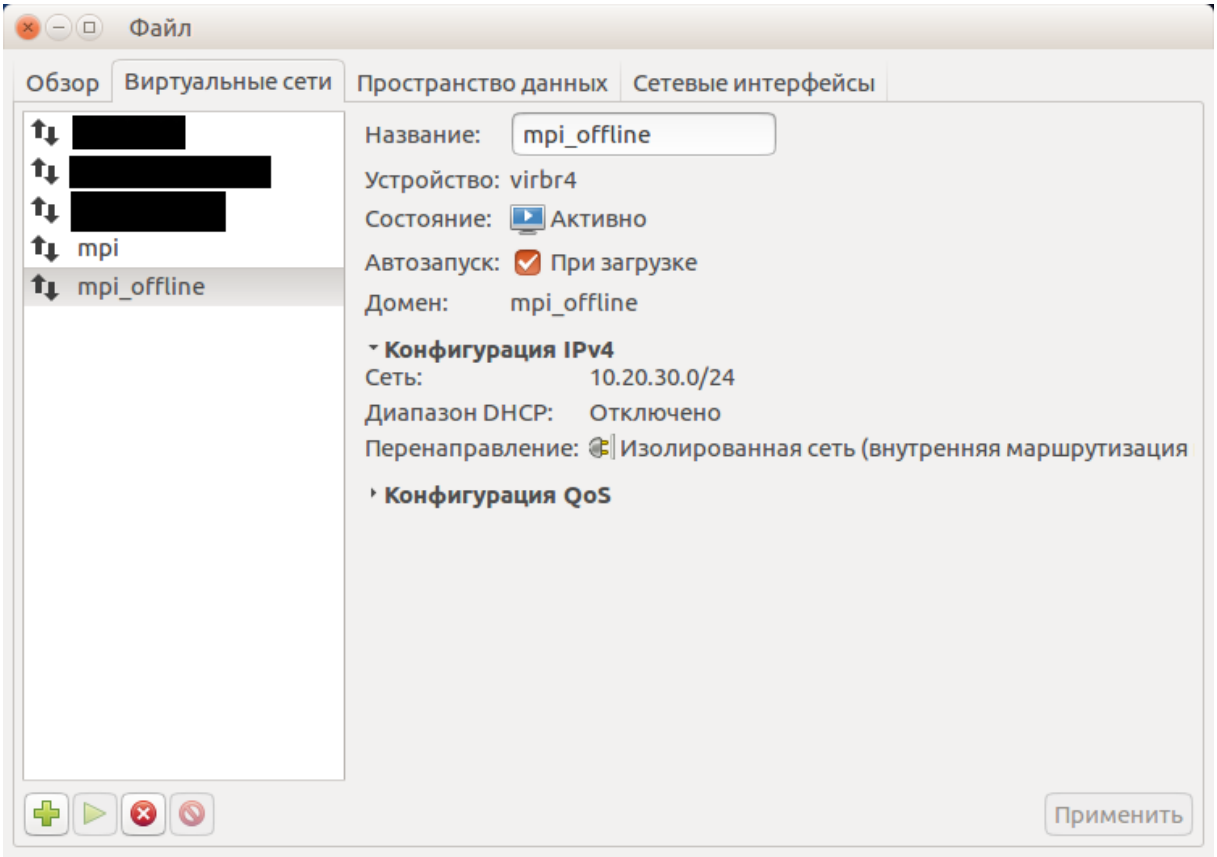

Рисунок 2.2 — Конфигурация второй, внутренней сети

2.2 Получение основного kickstart-файла

Следующим шагом, мы должны создать kickstart-файл, где будут описаны все основные этапы установки системы. Kickstart-файл представляет из себя простой текстовый файл (plain text) который разбит на обязательные и опциональные секции. Создание и отладка такого файла – довольно трудоемкая задача. Для упрощения его получения, можно провести тестовую установку системы, после чего в домашней папке пользователя root будет лежать файл anaconda-ks.cfg, сохонивший в себе все этапы выбора при установке.

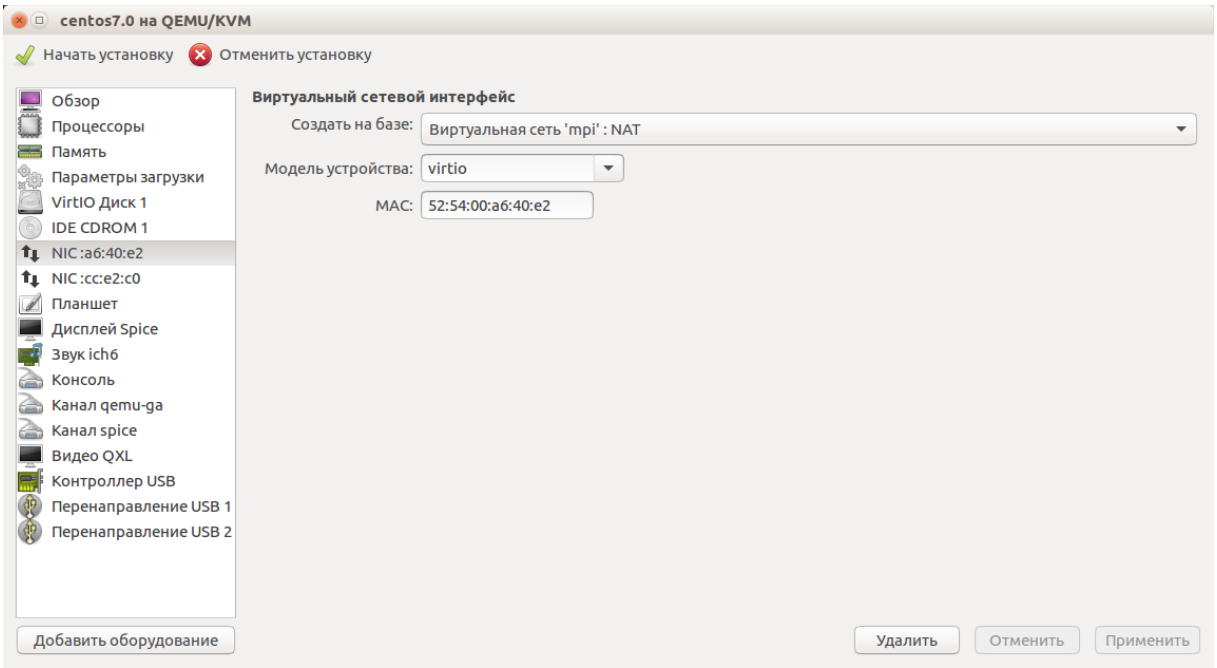

Рисунок 2.3 — Добавление второго сетевого интерфейса

Для этого выберем пункт "Создать виртуальную машину"и проследуем мастеру настройки, указав скаченный с официального сайта образ CentOS и созданную сеть с доступом в Интернет. Также выберем пункт "Дополнить конфигурацию перед установкой чтобы добавить второй интерфейс к доступу в изолированную сетью (Рис2.3), модель интерфейсов virtio, а для ускорения установки в параметрах жесткого диска можно указать "unsafe кэширования.

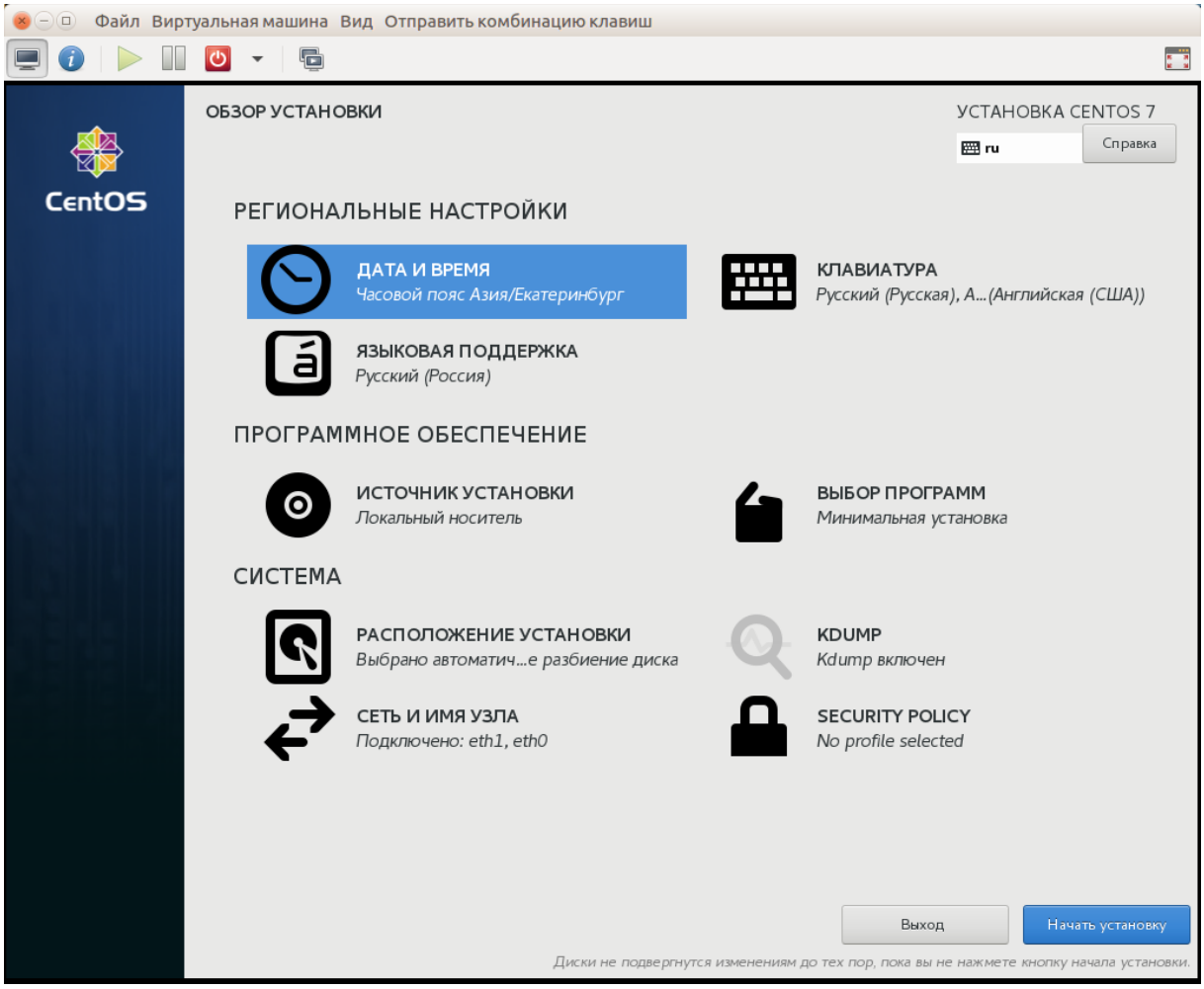

Рисунок 2.4 — Процесс установки CentOS7

После настройки физической конфигурации машины, мы можем нажать "Начать установку"и перед нами раскроется окно с эмуляцией монитора созданной машины. В ней мы увидим приглашение к установке CentOS7, в которой можем произвести всю необходимую настройку (Рис. 2.4). После того, как мы нажмем "Начать установку мы сможем настроить пользователей системы.

Наконец, перезагрузив систему и зайдя от root-пользователя мы найдем файл anaconda-ks.cfg. При желании полученный файл можно найти в приложении А.1.

2.3 Генерация kickstart-файла с помощью python

Для установки нескольких узлов, нам потребуется создать копии полученного kickstart-файла. При этом данные файлы будут отличаться только именем узла и статическим IP-адрессом в изолированной сети. Мы можем создать из полученного файла Jinja-шаблон[10], и сгенерировать полученные конфиги автоматически, в зависимости от номера узла:

```
#!/usr/bin/env python3

from jinja2 import Environment, FileSystemLoader

env = Environment(loader=FileSystemLoader('.'))
template = env.get_template('template.cfg')

for i in range(1, 4):
    with open('node'+str(i)+'.cfg', 'w') as f:
        f.write(template.render({
            'ip': '10.20.30.4' + str(i),
            'host': 'node' + str(i)
        }))
```

А в шаблоне заменить изменяющиеся параметры:

```
# ...

network  --bootproto=static --device=eth1 --ip={{ip}} \
          --netmask=255.255.255.0 --ipv6=auto --activate
network  --hostname={{host}}

# ...
```

После чего мы можем просто раздать http-сервером эти файлы. И при установке дописав параметр, который запустит автоматическую установку:

```
  inst.ks=http://192.168.127.1:8080/node1.cfg
```

2.4 Настройка NFS

Рассмотрим настройку NFS-подсистемы, которая сделает общий подкаталог /nfs для всех узлов. Данный критерий не обязателен при работе с MPI, но очень упрощает разработку.

2.4.1 Настройка NFS-сервера

Для настройки nfs в CentOS есть пакеты nfs-utils и nfs-utils-lib. В процессе установки использовался DVD-образ CentOS7, и эти пакеты были выбраны при установке. Поэтому остается только провести настройку узлов:

```
mkdir -p /nfs # создадим общую
vim /etc/exports # правим конфигурацию nfs-сервера
```

В /etc/exports вписываем следующую строку:

```
/nfs 10.20.30.0/24(rw,sync,no_root_squash,no_all_squash)
```

При этом:

— /home/nfs – расшариваемая директория;

— 10.20.30.40/24 – IP адрес клиента (или, как в моем случае, возможность подключения для всей подсети);

— rw – разрешение на запись;

— sync – синхронизация указанной директории;

— no_root_squash – включение root привилегий;

— no_all_squash — включение пользовательской авторизации;

После чего включаем все необходимые сервисы:

```
systemctl enable rpcbind
systemctl enable nfs-server
systemctl enable nfs-lock
systemctl enable nfs-idmap
systemctl start rpcbind
systemctl start nfs-server
```

```
systemctl start nfs-lock
systemctl start nfs-idmap
```

2.4.2 Настройка NFS-клиента

Со стороны клиента требуется гораздо меньше забот. Достаточно включить необобходимые сервисы:

```
systemctl enable rpcbind
systemctl enable nfs-server
systemctl enable nfs-lock
systemctl enable nfs-idmap
systemctl start rpcbind
systemctl start nfs-server
systemctl start nfs-lock
systemctl start nfs-idmap
```

И примонтировать папку с сервера:

```
mkdir -p /nfs
mount -t nfs 10.20.30.41:/nfs /nfs
```

После чего можно проверить работоспособность:

```
echo "Hello, NFS!" > /nfs/hello
# на node1 должен появится файл /nfs/hello
```

И наконец, чтобы не монтировать NFS при каждом запуске, подправим файл /etc/nfs:

```
# /etc/fstab
# ...
/dev/mapper/centos\_node2-root / xfs defaults 0 0
UUID=827da158-cf9b-44b9-9f1f-be86ccd21d49 /boot xfs defaults 0 0
/dev/mapper/centos\_node2-swap swap swap defaults   0 0
10.20.30.41:/nfs /nfs nfs rw,sync,hard,intr 0 0
```

После чего проверяем коррекность внесенных правок:

```
mount -fav
```

2.5 Установка Intel MPI

Для установки Intel реализации MPI требуется пройти регистрацию на их официальном сайте[9], после чего можно загрузить как коммерческую, так и бесплатную версию для ознакомления (что мы и сделали).

Для полноценной работы с MPI нам потребуется поставить набор необходимых компиляторов: gfortran/gcc/gcc-c++, а также сделать доступным все хосты по имени:

```
# /etc/hosts
# ...
10.20.30.41 node1
10.20.30.42 node2
10.20.30.43 node3
```

И так как пользоваться MPI должен обычный пользователь, мы должны сделать доступным SSH-доступ к каждому узлу скопировав публичный ключ:

```
ssh-copy-id user@{node2,node3}
```

Когда это сделано и архив загружен на рабочую станцию, можно раздать его на виртуальные узлы:

```
cd /nfs # установка потребуется на всех узлах
wget http://192.168.127.1:8080/mpi.tgz
tar -xf mpi.tgz
```

Далее достаточно запустить процесс установки, отвечая на инструкции установочника (запустить на всех узлах):

```
cd ./mpi
./install.sh
```

Теперь, для проверки, можно попробовать запустить на всех узлах какую-нибудь UNIX-программу. hostname для нас отлично подойдет, так как мы сможем легко убедимся в правильности запущенной программы:

```
# Найдем скрипт, настраивающий переменные среды Intel MPI
find / -type f -name 'mpivars.sh'

# Запустим это user-пользователем
# В последствии, можно добавить этот скрипт в /etc/.bashrc,
# чтобы Intel MPI окружение настраивалось при входе
. /opt/intel/compilers_and_libraries_2018.1.163/linux/mpi\\
  /intel64/bin/mpivars.sh

# тестовый запуск
# -ppn [число процессов на узел]
# -n [число процессов в общем]
# --hosts [список узлов через запятую]
mpirun -ppn 1 -n 3 \
     -hosts 10.20.30.41,10.20.30.42,10.20.30.43 hostname
```

Вывод программы (что и ожидалось):

```
node1
node2
node3
```

Для удобства использования, все хосты кластера задать в файле /nfs/hosts и создать алиас в .bashrc-файле:

```
alias mpirun='mpirun -f /nfs/hosts'
```

После чего, параметр –hosts можно опустить:

```
mpirun -ppn 2 -n 6 hostname
```

2.6 Запуск тестовой программы

Для полноценной проверки работоспособности MPI запустим тестовую программу, распространяемую вместе с Intel MPI реализацией:

```
cd /nfs/user
# скопируем программу в рабочий котолог
cp -r /opt/intel/impi/2018.1.163/test/ ./
cd test

# скомпилируем программу, написанную на C
mpicc -o test test.c

# запустим скомпилированную программу
mpirun -ppn 2 -n 6 ./test
```

Каждый узел этой программы отправляет свой номер, кол-во процессов и имя узла на 0-вой узел, который в свою очередь производит печать поулченных данных. Таким образом вывод нашей программы:

```
Hello world: rank 0 of 6 running on node1
Hello world: rank 1 of 6 running on node1
Hello world: rank 2 of 6 running on node2
Hello world: rank 3 of 6 running on node2
Hello world: rank 4 of 6 running on node3
Hello world: rank 5 of 6 running on node3
```

Заключение

Список использованных источников

1. Asia Supercomputer Community (ASC) 2018. — http://www.asc-events.org/ASC18/.

2. Linux Network File System Overiew, FAQ and HOWTO Documents. — http://nfs.sourceforge.net/.

3. Kernel-based Virtual Machine (KVM) – Документация Ubuntu. — http://help.ubuntu.ru/wiki/kvm.

4. libvirt – Документация Ubuntu. — https://goo.gl/vTvxi8.

5. CentOS – официальный сайт. — https://www.centos.org/.

6. Red Hat Enterprise Linux (RHEL) – официальный сайт. — https://www.redhat.com/en/technologies/linux-platforms/enterprise-linux.

7. Kickstart Installations – Документация Red Hat Enterprise Linux 7. — https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/installation_guide/chap-kickstart-installations.

8. MPI Forum. — http://mpi-forum.org/.

9. Intel MPI. — https://software.intel.com/en-us/intel-mpi-library/.

10. Jinja - официальный сайт. — http://jinja.pocoo.org/.

Приложение А Конфигурационные файлы и программы

А.1 kickstart-файл, полученный после первой установки

В файле опущены хеши паролей и длинные строки разбиты для лучшей наглядности после печати.

```
#version=DEVEL
# System authorization information
auth --enableshadow --passalgo=sha512
# Use CDROM installation media
cdrom
# Use graphical install
graphical
# Run the Setup Agent on first boot
firstboot --enable
# Keyboard layouts
keyboard --vckeymap=us --xlayouts='ru','us' \
         --switch='grp:alt_shift_toggle'
# System language
lang ru_RU.UTF-8

# Network information
network  --bootproto=dhcp --device=eth0 --ipv6=auto --activate
network  --bootproto=static --device=eth1 --ip=10.20.30.41 \
         --netmask=255.255.255.0 --ipv6=auto --activate
network  --hostname=localhost.localdomain

# Root password
rootpw --iscrypted ...
# System services
services --enabled="chronyd"
# System timezone
timezone Asia/Yekaterinburg --isUtc
```

```
user --groups=wheel --homedir=/nfs/user --name=user \
     --password=... --iscrypted --gecos="user"
# System bootloader configuration
bootloader --append=" crashkernel=auto" --location=mbr \
           --boot-drive=vda
autopart --type=lvm
# Partition clearing information
clearpart --none --initlabel

%packages
@^compute-node-environment
@base
@core
@debugging
@development
@directory-client
@guest-agents
@hardware-monitoring
@infiniband
@network-file-system-client
@performance
@remote-system-management
@scientific
@security-tools
chrony
kexec-tools

%end

%addon com_redhat_kdump --enable --reserve-mb='auto'

%end
```

```
%anaconda
pwpolicy root --minlen=6 --minquality=1 --notstrict --nochanges \
              --notempty
pwpolicy user --minlen=6 --minquality=1 --notstrict --nochanges \
              --emptyok
pwpolicy luks --minlen=6 --minquality=1 --notstrict --nochanges \
              --notempty
%end
```