МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
федеральное государственное бюджетное образовательное учреждение высшего образования
КРАСНОЯРСКИЙ ГОСУДАРСТВЕННЫЙ ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ им.В.П.АСТАФЬЕВА
(КГПУ им.В.П.Астафьева)

Институт/факультет Институт математики, физики и информатики
(полное наименование института/факультета/филиала)

Выпускающая кафедра Базовая кафедра информатики и информационных технологий в образовании
(полное наименование кафедры)

Шпилевой Александр Александрович

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Тема **Построение вычислительного кластера из одноплатных компьютеров для образовательного учереждения**

Направление подготовки 44.03.01 Педагогическое образование
(код и наименование направления)

Профиль Информатика
(наименование профиля для бакалавриата)

ДОПУСКАЮ К ЗАЩИТЕ

Заведующий кафедрой
д.п.н., профессор Пак Н.И.
(ученая степень, ученое звание, фамилия, инициалы)

(дата, подпись)

Руководитель к.ф.-м.н., доцент
Шишаков С.А.
(ученая степень, ученое звание, фамилия, инициалы)

Дата защиты

Обучающийся Шпилевой А.А.
(фамилия, инициалы)

26.06.2018
(дата, подпись)

Оценка отлично
(прописью)

Красноярск 2018

Содержание

Введение

Суперкомпьютеры являются классическим примером гонки вооружений. Растущая производительность этих огромных систем позволяет решать все более сложные проблемы. Суперкомпьютеры – это предмет национальной и корпоративной гордости: страны и корпорации стремятся получить наивысшие результаты и наибольшее количество FLOPS. Первые суперкомпьютеры были разработаны для моделирования ударов ядерного оружие и вскрытия кодов шифровальных машин. Сегодня суперкомпьютеры применяются в самых различных областях и служат инструментами для решения проблем, связанных с огромными объемами вычислений – например, исследование климата, физическое, молекулярное моделирование, генетические алгоритмы, и, конечно же, Big Data.

К сожалению, настоящие, высокопроизводительные суперкомпьютеры, весьма недешевы, как по стоимости, так и в дальнейшей эксплуатации. Так, например, стоимость суперкомпьютера Sunway TaihuLight, занимающего первое место по вычислительной мощности на Ноябрь 2016 год, составляет около 273 млн. долларов. А первоначальная стоимость суперкомпьютера «Ломоносов», стоящего в МГУ им. М.В. Ломоносова, и сейчас занимающего 132-ю позицию в мировом рейтинге TOP500, составила 1,9 млрд рублей. Естественно, такие суммы являются неподъемными для средней школы.

Интересной особенностью суперкомпьютеров является то, что все они работают под управлением той или иной версии Linux, что позволяет собрать вычислительный кластер на основе практически любых компонентов, установить на них Linux-систему, развернуть Open Source менеджер задач и получить суперкомпьютер с таким же интерфейсом и программным обеспечением, как на «взрослых» суперкомпьютерах, пусть и с небольшой вычислительной мощностью.

Актуальность:

Обучение суперкомпьютерным технологиям требует использование суперкомпьютера, причём, в зависимости от целей обучения, возможно не только для обучению параллельному программированию и администрированию, но и для настройки его аппаратной составляющей. Одним из вариантов является использование кластера из учебных компьютеров. Более дешёвым и современным, а так же не требовательным к размеру помещения вариантом, является вычислительный кластер из одноплатных компьютеров.

Проблема исследования:

Невозможность, в первую очередь из-за финансовых причин, приобретения учебным учреждением полноценного суперкомпьютера. Отсутствие доступной информации и методических разработок по созданию вычислительного кластера из одноплатных компьютеров.

Цель работы:

разработать методические рекомендации по созданию вычислительного кластера из одноплатных компьютеров для образовательного учреждения

Объект исследования:

техническое обеспечение учебного процесса в области информатики

Предмет исследования:

разработка вычислительного кластера

Задачи:

1. Проанализировать возможность и пути создания кластера из одноплатных компьютеров

2. Сделать выбор одноплатного компьютера, подходящего для построения вычислительного кластера в образовательном учреждении

3. Разработать варианты конструкций кластеров из одноплатных компьютеров для образовательного учреждения

4. Собрать один из возможных кластеров из одноплатных компьютеров

5. Проанализировать выбор программного обеспечения для реализации вычислительного кластера

6. Апробировать установку одного из вариантов программного обеспечения на собранный кластер

7. Разработать рекомендации по реализации вычислительного кластера из одноплатных компьютеров в образовательном учреждении

Глава 1. Вычислительный кластер и его создание

1.1 Аппаратные средства высокопроизводительных вычислений

Проблема высокопроизводительных вычислительных систем относится к числу наиболее сложных научно-технических задач. Решение этой проблемы развивалось в нескольких направлениях – повышении мощности и быстродействия обычных компьютерных систем, создании специализированных суперкомпьютеров, формировании кластерных систем, применении GRID-технологий, разработке методов параллельного программирования.

Когда-то было дано полушуточное определение суперкомпьютера: "суперкомпьютер – это устройство, сводящее проблему вычислений к проблеме ввода/вывода". Смысл этого определения заключается в том, что суперкомпьютер должен выполнять вычисления настолько быстро, что время расчетов будет мало по сравнению со временем, необходимым для того, чтобы ввести в программу исходные данные и вывести результаты.

Более формальное определение звучит так: суперкомпьютер – это специализированная вычислительная машина, значительно превосходящая по своим техническим параметрам и скорости вычислений большинство существующих в мире компьютеров. Определение границ производительности, отделяющих суперкомпьютеры от просто мощных компьютеров очень быстро устаревает. Соответственно, и данное выше определение не оперирует абсолютными значениями производительности.

Вычислительные кластеры используются во всех сферах, где для решения задачи применяется численное моделирование, обработка большого количества данных в реальном времени, или решение задачи может быть найдено простым перебором множества значений множества исходных параметров.

Совершенствование методов численного моделирования происходило одновременно с совершенствованием вычислительных машин: чем сложнее были задачи, тем выше были требования к создаваемым машинам, и, соответственно, чем быстрее были машины, тем сложнее были задачи, которые на них можно было решать. Поначалу суперкомпьютеры применялись почти исключительно для оборонных задач: расчеты по ядерному и термоядерному оружию, ядерным реакторам, шифрование. Потом, по мере совершенствования математического аппарата численного моделирования, развития знаний в других сферах науки – суперкомпьютеры стали применяться и в «мирных» расчетах, создавая новые научные дисциплины, как то: численный прогноз погоды, биология и медицина, химия, гидродинамика, лингвистика и проч., – все области, где достижения информатики сливались с достижениями прикладной науки.

Эти уникальные решения с рекордными характеристиками обычно недешевы, поэтому, к сожалению, их нельзя пустить в массовое производство и широко использовать в бизнесе. Прогресс в области сетевых технологий сделал свое дело – появились недорогие, но эффективные решения, основанные на коммуникационных технологиях. Это и предопределило появление кластерных вычислительных систем, фактически являющихся одним из направлений развития компьютеров с массовым параллелизмом вычислительного процесса (Massively Parallel Processing – MPP).

Кластер — группа компьютеров, объединённых высокоскоростными каналами связи, представляющая с точки зрения пользователя единый аппаратный ресурсОдин из первых архитекторов кластерной технологии Грегори Пфистер дал кластеру следующее определение: «Кластер — это разновидность параллельной или распределённой системы, которая:

- состоит из нескольких связанных между собой компьютеров;

- используется как единый, унифицированный компьютерный ресурс».

Вычислительный кластер – это совокупность компьютеров, объединенных в рамках некоторой сети для решения крупной вычислительной задачи. В качестве узлов обычно используются доступные однопроцессорные компьютеры, двух- или четырехпроцессорные серверы. Каждый узел работает под управлением своей копии операционной системы, в качестве которой чаще всего используются *nux-семейства, Solaris и, иногда, Windows. В принципе, кластером можно считать как пару персональных компьютеров, связанных локальной сетью, так и обширную вычислительную систему, создаваемую в рамках крупного проекта, например SETI@HOME.

Состав и вычислительная мощность узлов вычислительного кластера может меняться даже в рамках самого кластера, давая возможность создавать обширные неоднородные системы с задаваемой вычислительной мощностью. Выбор конкретной коммуникационной среды определяется многими факторами: особенностями класса решаемых задач, финансированием, необходимостью последующего расширения кластера и т.п. Возможно включение в конфигурацию специализированных компьютеров, например файл-сервера, и, как правило, предоставлена возможность удаленного доступа к кластеру через Интернет.

В большинстве случаев, кластеры серверов функционируют на раздельных компьютерах. Это позволяет повышать производительность за счёт распределения нагрузки на аппаратные ресурсы, обеспечивает отказоустойчивость на аппаратном уровне и предоставляет возможность расширения.

В то время как обычный суперкомпьютер содержит множество процессоров, подключенных к локальной высокоскоростной шине,

распределенные, или GRID-вычисления, в целом являются разновидностью параллельных вычислений, которое основывается на обычных компьютерах (со стандартными процессорами, устройствами хранения данных, блоками питания и т.д.), подключенных к сети (локальной или глобальной) при помощи обычных протоколов, например TCP/IP.

Основным преимуществом распределенных вычислений является то, что отдельная ячейка вычислительной системы может быть приобретена как обычный неспециализированный компьютер. Таким образом можно получить практически те же вычислительные мощности, что и на обычных суперкомпьютерах, но с гораздо меньшей стоимостью.

1.2 Варианты реализации вычислительных кластеров

Вычислительные кластеры классифицируются, прежде всего, по характеру узловых процессоров (см. рис. 1).

В качестве узлов вычислительного кластера обычно используют персональные компьютеры, рабочие станции и SMP-сервера. Если в качестве узла кластера используются SMP-система, то такой вычислительный кластер называется SMP-кластером.

Если в качестве узлов вычислительного кластера используются персональные ЭВМ или рабочие станции, то обычной является ситуация, когда во время решения задачи на кластере на узлах этого кластера продолжают выполняться последовательные задания пользователей. В результате относительная производительность узлов кластера меняется случайным образом и в широких пределах. Решением проблемы было бы написание самоадаптирующейся пользовательской программы. Однако эффективное решение этой задачи представляется весьма проблематичным. Ситуация усугубляется, если среди узловых компьютеров вычислительного кластера имеются файловые серверы. При этом во время решения задачи на кластере в широких пределах может меняться загрузка коммуникационной среды, что делает непредсказуемыми коммуникационные расходы задачи.

Рис.1. Классификация узлов вычислительных кластеров.

Как и всякие MIMD-системы, вычислительные кластеры разделяются на однородные кластерные системы (однородные вычислительные кластеры) и гетерогенные кластерные системы (гетерогенные вычислительные кластеры).

Обычно, когда говорят о вычислительных кластерах, подразумевают однородные вычислительные кластеры. Однако часто при наращивании кластера приходится использовать процессоры, отличающиеся не только по производительности, но и по архитектуре, от узловых процессоров кластера. Поэтому постепенно однородный вычислительный кластер может стать неоднородным. Эта неоднородность создает следующие проблемы. Различие в производительности процессоров усложняет задачу распределения работ между процессорами. Различие в архитектуре процессоров требует подготовки разных выполняемых файлов для разных узлов, а в случае различий в представлении данных, может потребовать и преобразования их форматов при передаче сообщений между узлами.

Узлы вычислительного кластера могут представлять собой полнофункциональные компьютеры, которые могут работать и как самостоятельные единицы. Производительность такого кластера обычно невысока.

Для создания высокопроизводительных вычислительных кластеров системные блоки узловых компьютеров делают значительно более простыми, чем в первом случае (не полнофункциональными). Здесь нет необходимости снабжать компьютеры узлов графическими картами, мониторами, дисковыми накопителями и другим периферийным оборудованием. Периферийное оборудование устанавливается только на одном или немногих управляющих

компьютерах (HOST-компьютерах). Такой поход позволяет значительно уменьшить стоимость системы.

При классификации кластеров используется и ряд других классификационных признаков. Рассмотрим два из них (см. рис. 2.):

- классификация по стандартности комплектующих;
- классификация по функциональной направленности.

Рис.2. Классификация вычислительных кластеров.

С точки зрения стандартности комплектующих можно выделить два класса кластерных систем:

- вычислительный кластер строится целиком из стандартных комплектующих;
- при построении кластера используются эксклюзивные или нешироко распространенные комплектующие.

Вычислительные кластеры первого класса имеют низкие цены и простое обслуживание. Широкое распространение кластерные технологии получили как средство создания именно относительно дешевых систем суперкомпьютерного класса из составных частей массового производства.

Кластеры второго класса позволяют получить очень высокую производительность, но являются, естественно, более дорогими.

С функциональной точки зрения кластерные системы можно разделить на высокоскоростные кластерные системы (High Performance) – HP-кластеры и кластерные системы высокой готовности (High Availability) – HA-кластеры. Высокоскоростные кластеры используются в областях, которые требуют значительной вычислительной мощности. Кластеры высокой готовности используются везде, где стоимость возможного простоя превышает стоимость затрат, необходимых для построения отказоустойчивой системы.

Высокоскоростные кластеры. Производительность вычислительного кластера, очевидно, зависти от производительности его узлов. С другой стороны, производительность кластера, как и сякой системы с распределенной памятью, сильно зависит от производительности коммуникационной среды. Обычно при построении вычислительных кластеров используют достаточно дешевые коммуникационные среды. Такие среды обеспечивают производительность на один – два порядка более низкую, чем производительность коммуникационных сред суперкомпьютеров. Поэтому находится не так много задач, которые могут достаточно эффективно решаться на больших кластерных системах.

Влияние производительности коммуникационной среды на общую производительность кластерной системы зависит от характера выполняемой задачи. Если задача требует частого обмена данными между подзадачами, которые решаются на разных узлах вычислительного кластера, то быстродействию коммуникационной среды следует уделить максимум внимания. Соответственно, чем меньше взаимодействуют части задачи между собою, тем меньше внимания можно уделить быстродействию коммуникационной среды.

Место HP-кластеров среди современных высокопроизводительных систем иллюстрирует следующий пример: в списке 500 самых высокопроизводительных компьютеров мира «Top500» вычислительные кластеры из недорогих узлов занимают примерно половину списка.

В России крупнейший заказчик HP-кластеров — нефтегазовая отрасль, в которой кластеры все более широко используются для обработки трехмерных сейсмических данных, получаемых в процессе геологоразведки.

Кластеры высокой готовности. Среди многообразия типов современных вычислительных систем высокой готовности HA-кластеры обеспечивают высокий уровень отказоустойчивости при самой низкой стоимости.

Вообще говоря, для того, чтобы вычислительная система обладала высокими показателями готовности, необходимо, чтобы ее компоненты были максимально надежными, чтобы система была отказоустойчивой, а так же чтобы была возможной «горячая» замена компонентов (без останова системы). Благодаря кластеризации при отказе одного из компьютеров системы, задачи могут быть автоматически (операционной системой) перераспределены между другими (исправными) узлами вычислительного кластера. Таким образом, отказоустойчивость кластера обеспечивается дублированием всех жизненно важных компонент вычислительной системы. Самыми популярными коммерческими отказоустойчивыми системами в настоящее время являются двухузловые кластеры.

Выделяется еще один класс вычислительных кластеров – вычислительные сети (GRID), объединяющие ресурсы множества кластеров, многопроцессорных и однопроцессорных ЭВМ, которые могут принадлежать разным организациям и быть расположенными в разных странах.

Разработка параллельных программ для вычислительных сетей усложняется из-за следующих проблем. Ресурсы (количество узлов, их архитектура, производительность), которые выделяются задаче, определяется только в момент обработки сетью заказа на выполнение это задачи. Поэтому программист не имеет возможности разработать программу для конкретной конфигурации вычислительной сети. Программу приходится разрабатывать так, чтобы она могла динамически (без перекомпиляции) самонастраиваться на выделенную конфигурацию сети. Кроме того, к неоднородности коммуникационной среды добавляется изменчивость ее характеристик, вызываемая изменениями загрузки сети. В лучшем случае программа должна разрабатываться с учетом этой неоднородности коммуникационной среды, что представляет собой весьма непростую задачу. Как мы отмечали выше, подобная проблема имеет место и для вычислительных кластеров, построенных на основе персональных компьютеров или рабочих станций.

Эффективная производительность кластерных вычислительных систем (real applications performance – RAP) оценивается как 5–15% от их пиковой производительности (Peak Advertised Performance, PAP). Для сравнения: у лучших малопроцессорных систем из векторных процессоров это соотношение оценивается как 30–50%.

1.3 Программное обеспечение вычислительного кластера

Программное обеспечение кластеров состоит из двух компонент:

- средств разработки/программирования и
- средств управления ресурсами.

К средствам разработки относятся компиляторы для языков, библиотеки различного назначения, средства измерения производительности, а также отладчики, что, всё вместе, позволяет строить параллельные приложения.

Хотя для параллельной обработки существует очень много моделей программирования, но, на настоящий момент, доминирующим подходом является модель на основе "передачи сообщений" (message passing), реализованная в виде стандарта MPI (Message Passing Interface). MPI - это библиотека функций, с помощью которых в программах на языках C или Фортран можно передавать сообщения между параллельными процессами, а также управлять этими процессами.

Альтернативами такому подходу являются языки на основе так называемого "глобального распределенного адресного пространства" (GPAS - global partitioned address space), типичными представителями которых являются языки HPF (High Performance Fortran) и UPC (Unified Parallel C).

1.4 Создание вычислительного кластера на основе локальной сети

Одну из машин следует выделить в качестве центральной (головной) куда следует установить достаточно большой жесткий диск, возможно более мощный процессор и больше памяти, чем на рабочие узлы. Имеет смысл обеспечить (защищенную) связь этой машины с внешним миром.

Программное обеспечени. В качестве программной платформы системы выбран кластерный дистрибутив Rocks Clusters, основанный на открытой свободно распространяемой операционной системе CentOS. Unix-подобные операционные системы чаще всего становятся платформой, на которой разворачиваются кластеры (как вычислительные, так и кластеры высокой доступности), что связано прежде всего с зарекомендовавшим себя уровнем поддержки сетевых сервисов. Кроме того, Linux-кластеры являются гораздо более гибкими, масштабируемыми и доступными по цене, чем аналогичные решения, например, от Microsoft. Указанный кластерный дистрибутив Rocks является открытым свободно распространяемым продуктом. Он включает в себя основные автоматически настраиваемые сервисы для увязки узлов системы в единый вычислительный кластер. В числе прочего он обеспечивает:

- возможность использования автоматизированных средств расширения и контроля состояния кластера (в том числе система Ganglia);
- возможность адаптации логической структуры кластера к конкретному вычислительному эксперименту;
- использование единого набора компиляторов и библиотек как на управляющем, так и на вычислительных узлах, а также возможность переключения между различными реализациями библиотек (система Module);

Существуют и другие кластерные платформы, например системы MOSIX, OSCAR, PelicanHPC и другие, обладающие схожими возможностями. Кластерный дистрибутив Rocks Clusters был выбран исходя из нескольких соображений. Во-первых, в отличие от системы OSCAR он не требует наличия на узлах кластера предустановленной операционной системы, что упрощает процесс добавления в кластер вычислительных узлов. Во-вторых, эта система ориентирована на относительно небольшие вычислительные установки и не требует столь существенных затрат на поддержку и системное администрирование, как, например, система MOSIX, предназначенная для суперкомпьютеров промышленного масштаба. В то же время дистрибутив Rocks Clusters позволяет провести настройку основных параметров кластера и использовать имеющееся оборудование намного эффективнее по сравнению с системой PelicanHPC, которая не требует предварительной установки и загружается с внешнего (DVD или USB) носителя в заранее выбранной производителем стандартной конфигурации. Дополнительно к указанным системам на кластере установлен пакет компиляторов Intel для языков C/C++ и Fortran. Этот пакет поддерживает технологию распараллеливания для систем с общей памятью OpenMP и включает собственную реализацию библиотеки линейной алгебры BLAS (Basic Linear Algebra Subroutines). В качестве библиотеки для вычислений с распределенной памятью выбрана библиотека OpenMPI, скомпилированная с поддержкой компиляторов Intel. Также пользователям доступны открытые компиляторы GCC и GFortran с соответствующим вариантом библиотеки OpenMPI. Все перечисленное программное обеспечение является свободно распространяемым.

Кроме того, на управляющем узле установлено стандартное математическое программное обеспечение, включающее системы разработки ПО, системы визуализации полученных результатов и т.д.

Глава 2. Вычислительный кластер из одноплатных компьютеров

2.1 Требования к вычислительному кластеру в образовательном учреждении

Кроме повышенной надежности и быстродействия, есть еще несколько дополнительных требований, предъявляемых к кластерам в современных вычислительных системах. Они, в частности, должны обеспечивать единое внешнее представление системы, высокую скорость резервного копирования и восстановления данных и параллельный доступ к БД, обладать возможностями переноса нагрузки с аварийных узлов на исправные, иметь средства настройки высокого уровня готовности, гарантировать восстановление после аварии. По понятным причинам использование нескольких узлов кластера, которые одновременно обращаются к одним и тем же данным, увеличивает сложность процедуры резервного копирования и последующего восстановления информации. Перенос нагрузки с аварийного узла на исправный - это основной механизм обеспечения непрерывной работы приложений при условии оптимального использования ресурсов кластера. Для эффективной совместной работы кластерных систем и СУБД система должна иметь распределенный менеджер блокировок, обеспечивающий непротиворечивое изменение базы данных при поступлении последовательности запросов с разных узлов кластера. Настроить конфигурацию кластера и обеспечить высокую доступность приложений также непросто. Это связано в первую очередь со сложностью определения правил, по которым те или иные приложения переносятся с аварийных узлов кластера на исправные. Кластерная система должна позволять легко переносить приложения с одного узла кластера на другой, а также восстанавливать аварийное приложение на другом узле.

Следует отметить, что пользователь системы не обязан знать о том, что он работает с кластерной системой, поэтому для максимально комфортных условий работы пользователей кластер должен выглядеть извне как единый

компьютер. Он должен иметь единую файловую систему для всех узлов, единый IP-адрес и единое ядро системы. Даже самые надежные системы могут выйти из строя, если произойдет, например, стихийное бедствие (пожар, землетрясение, наводнение) или атака террористов. При глобальном масштабе современного бизнеса такие события не должны ему вредить, поэтому кластер может (или должен) быть распределенным.

2.2 Выбор одноплатного компьютера, подходящего для построения вычислительного кластера в образовательном учреждении

Raspberry Pi 2 Model B

Процессор Broadcom BCM2837 4x900MHz

Оперативная память 1Gb

Сетевой интерфейс 100Mb

Wi-Fi отсутствует

Стоимость ~2300руб.

Banana Pi BPI-M2

Процессор Allwinner A31 4x1GHz

Оперативная память 1Gb

Сетевой интерфейс 1 GbE

Wi-Fi присутствует

Стоимость ~2700руб.

NanoPC-T3

Процессор Samsung S5P6818 8x1.4GHz

Оперативная память 2Gb

Сетевой интерфейс 1 GbE

Wi-Fi присутствует

Стоимость ~3900руб.

Orange Pi PC

Процессор Allwinner H3 4x1.6GHz

Оперативная память 1Gb

Сетевой интерфейс 100MbE

Wi-Fi отсутствует

Стоимость ~800руб.

Исходя из проведенного анализа, можно сделать вывод, что плата Orange Pi PC является наиболее оптимальным решением для построения вычислительного кластера ввиду наилучшего соотношения цена-качество. Так, за ~15$ мы получаем процессор с 4 ядрами и тактовой частотой 1.6ГГц.

2.3 Сборка вычислительного кластера

Для демонстрационной модели были приобретены:

- 5 плат Orange Pi PC
- 1 коммутатор
- 1 блок питания
- 1 5v-вентилятор
- 5 microSD-карт памяти

В программе Google ScetchUp был спроектирован корпус устройства из акрила, чтобы было видно "внутренности" кластера, и в программе Incscape была создана развертка на него, чтобы можно было его вырезать лазером (рис. 3, 4, 5).

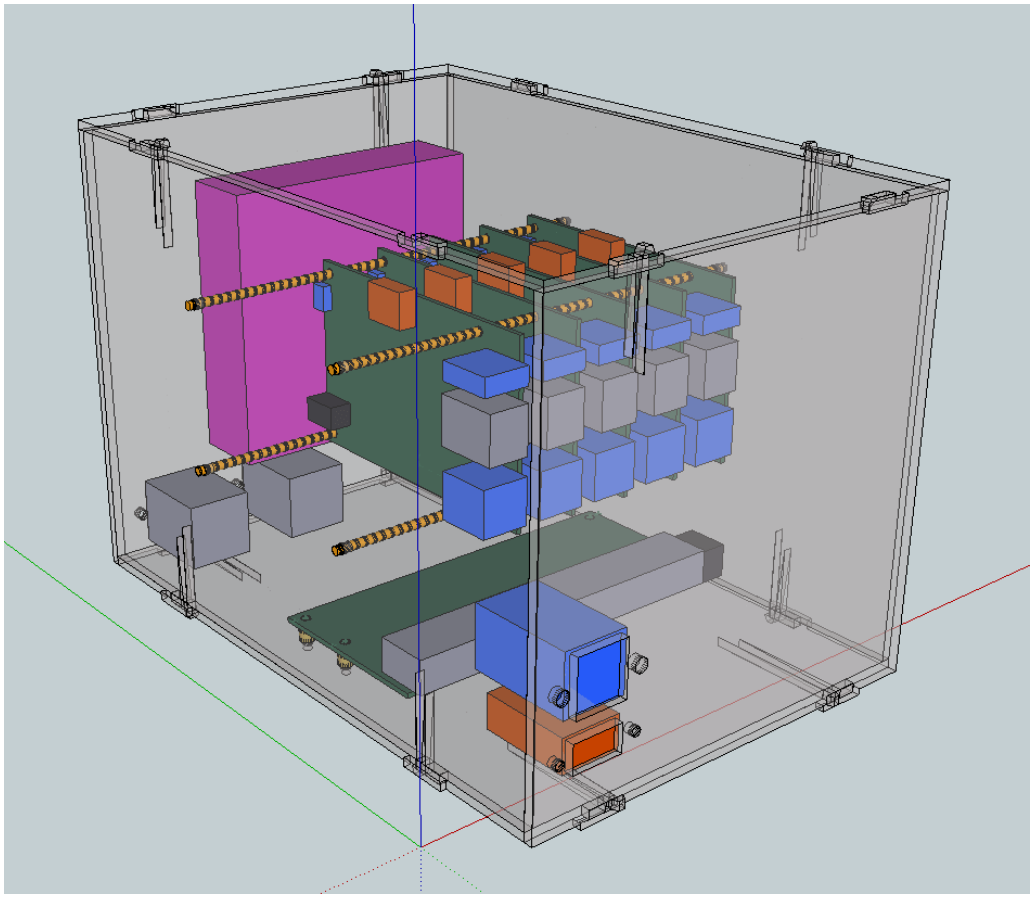

Рис. 3 Корпус устройства, вид спереди

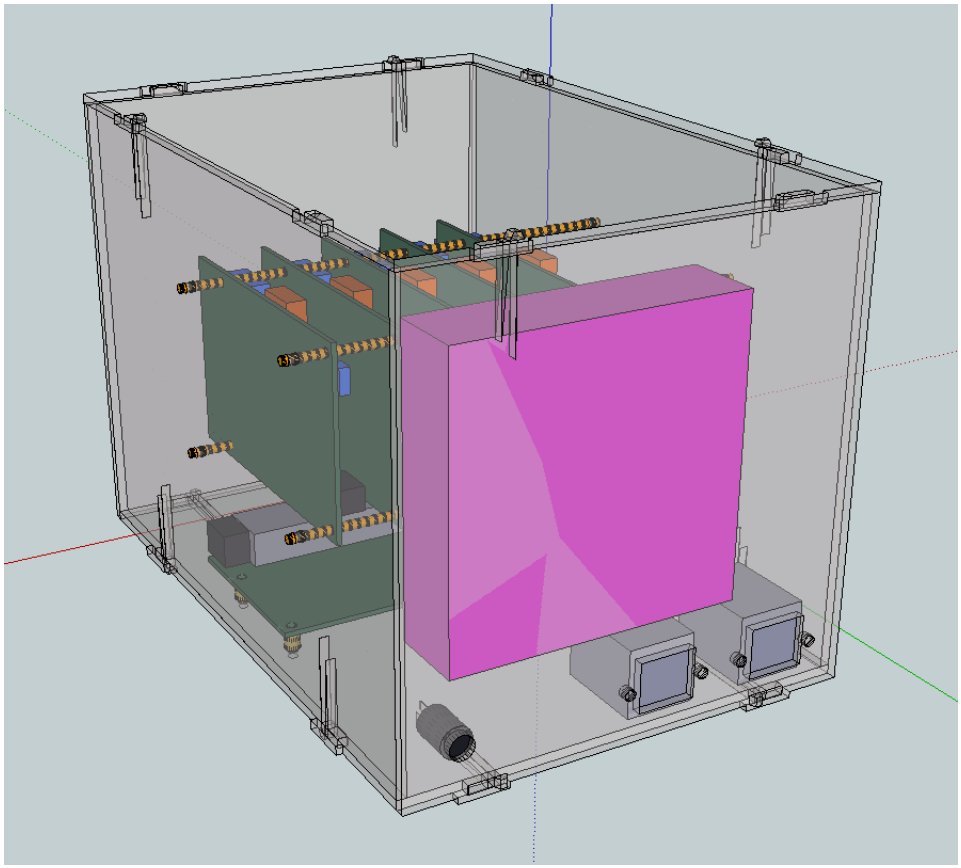

рис. 4 Корпус устройства, вид сзади

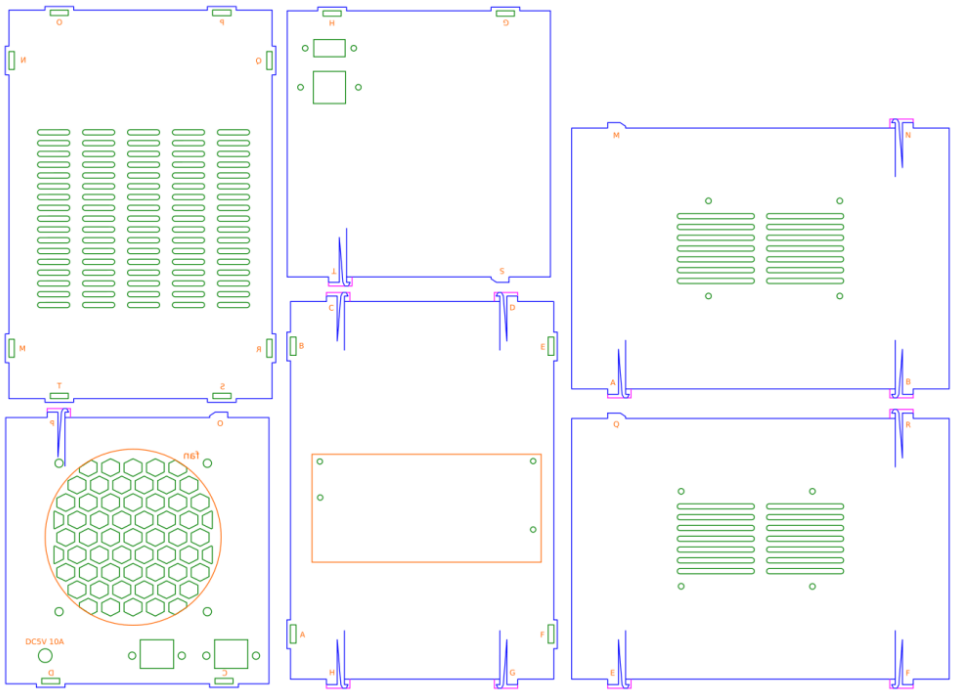

Рис. 5 Развертка корпуса для лазерной резки

2.4 Настройка вычислительного кластера

В первую очередь, необходимо установить операционную систему, на платы. Для этого необходимо скачать программу "SD Formatter", запустить её и подключить microSD-карту к компьютеру. Запустим SD Formatter, в пункте "Options" и в открывшемся окне "FORMAT TYPE" выбрать "FULL (OverWrite)", а в графе "FORMAT SIZE ADJUSTMENT" выбрать "ON". Далее нажимаем "OK" (рис. 6, 7).

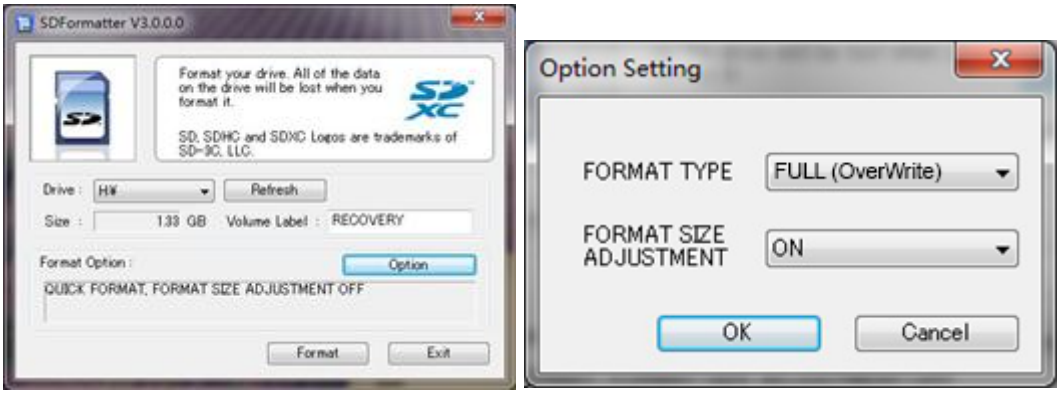

Рис. 6, 7

Нажимаем "Format" и дожидаемся полного форматирования карты.

Далее, скачиваем программу для записи образов ОС "Win32 Diskimage". Запускаем её, в открывшемся окне указываем путь к скаченной операционной системе, нажимаем "Write" и ожидаем, когда завершится процесс записи (рис. 8).

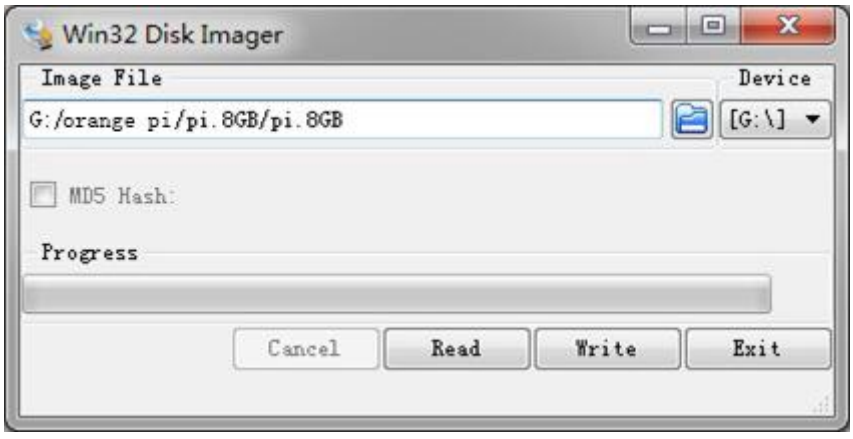

Рис. 8 Запись образа ОС

После проделанных манипуляций вытаскиваем карту из компьютера и вставляем её в Orange Pi.

Логин и пароль по умолчанию: orangepi/orangepi.

После того как на все узлы установлена операционная система, необходимо произвести конфигурацию вычислительного кластера. В первую очередь необходимо настроить локальную подсеть. Также процесс конфигурации вычислительного кластера подразумевает под собой настройку основных служб, таких как: iptables, DNS, NFS, NTP, NIS. Рассмотрим детальную установку каждой из них.

Структура IPTABLES Структуру можно представить следующим образом. *iptables → Tables → Chains → Rules* Пакет приходит на IPTABLES → далее передается на *таблицы* (*Tables*) → проходит по *цепочкам* (*Chains*) → проверяются правила (*Rules*). IPTABLES можно разделить на три части. • *таблицы*: набор цепочек; • *цепочки:* набор правил; • *правила*; 2.3.2 Таблицы IPTABLES IPTABLES включает в себя четыре встроенных типа таблиц: 1. Таблица Filter. Таблица по умолчанию. Если при создании/изменении правила не указана таблица – используется именно filter. Используется в основном для фильтрации пакетов. Например, можно выполнить DROP, LOG, ACCEPT или REJECT без каких либо сложностей, как в других таблицах. Использует 3 встроенных цепочки: o INPUT chain – входящие пакеты, используется только для пакетов, цель которых – сам сервер, не используется для транзитного (роутинга) трафика; o OUTPUT chain – исходящие пакеты, созданные локально и отправленные “за пределы” сервера; o FORWARD chain – пакеты, предназначенные другому сетевому интерфейсу (например, маршрут на другие машины сети). 2. Таблица NAT. Таблица nat используется главным образом для преобразования сетевых адресов (*Network Address Translation*). Эту таблицу проходит только первый пакет из потока.

Преобразования адресов автоматически применяется ко всем последующим пакетам. o PREROUTING chain – преобразование адресов DNAT (*Destination Network Address Translation*); o POSTROUTING chain – выполняется преобразование адресов SNAT (*Source Network Address Translation)*; o OUTPUT chain – NAT для локально сгенерированных пакетов; 3. Таблица Mangle. Таблица Mangle предназначена только для внесения изменения в некоторые заголовки пакетов – TOS (*Type of Service*), TTL (*Time to Live*), MARK (особая метка для IPTABLES или других служб). Включает в себя следующие цепочки: o PREROUTING chain o OUTPUT chain o FORWARD chain o INPUT chain o POSTROUTING chain 19 4. Таблица Raw. Применяется до передачи пакета механизму определения состояний (*state machine, connection tracking* – система трассировки соединений, при помощи которой реализуется межсетевой экран на сеансовом уровне (*stateful firewall*), позволяет определить, к какому соединению или сеансу принадлежит пакет, анализирует все пакеты кроме тех, которые были помечены NOTRACK в таблице raw). o PREROUTING chain o OUTPUT chain 2.3.3 Цепочки IPTABLES Выделяют 5 типов цепочек, встроенных в iptables:

• ПРЕРОУТИНГ □ ™└□ ⋃ ⌇ ◊ |◊└ ⌋ ⌇ ⌊Γ ⌊⟨□◊⟨ ⌊⋒ ⌊ ®J ⌊™□ ⅂⋃ |◊| ∑∫⌊®; • ІПΠУТ □ ™└□ ®J ⌊™□ ⅂⋃ |◊| ∑∫⌊®, ◊™□∑⟩⌊®◊⌇⌇ ⌋J ⌇∑| ⌊⟩□∑™⟩∫®∑⌇⌇ ⌊ локальному процессу (клиенту или серверу);

• ФОРΩАРΔ □ ™└□ ®J ⌊™□ ⅂⋃ |◊| ∑∫⌊® |∑□∑⌇◊|□◊®└∑⌇⌇ ⌋J ⌇◊ ®⌋J ⌊™; • О УТПУТ □ ™└□ |◊| ∑∫⌊® ©∑⌇ ∑□⌊□ ⌠∑⌠ ⌋J ⌊⌊ ◊└ ⌋⌇ ⌋⌠⌊ |□⌊⟩∑⟩◊⌠⌊; • ПОΣТРОУ ТІNГ □ ™└□ ⌊⌊ ⌊⌇ |◊∫∑└ ⌋⌇ ⌊Γ ⌊⟨□◊⟨ ⌊⋒ ⌊ ⌊⟩J ⌊™□ ⅂⋃ |◊| ∑∫⌊®. 2.3.4 ∉□◊®⌊◊ ІПТ АВΛЕΣ Правила имеют следующую структуру: *правило → цель → счётчик* Если пакет соответствует *правилу*, к нему применяется *цель*, и он учитывается *счетчиком*. Если *правило* не задано – то *цель* применяется ко всем проходящим через цепочку пакетам. Если не указаны ни *цель*, ни

правило – для правила будет срабатывать только *счётчик* пакетов. Если пакет не попадает под *правило* и *цель* – он передаётся следующему правилу в списке. 2.3.5 Цели (targets) IPTABLES Правило может содержать одно из следующих целей (или действий):

• АХХЕПТ □ |□\{□∫⎤ |◊| Σ∫, \ |Σ□Σ™◊∫⎤ ⟩⎣Σ™ ⌠⌋⎡Σ⎡ ⎤Σ|⎝⎢ Σ (⎣⎣\ приложению, или передать для дальнейшего роутинга);

• ΔNAT □ (*Δεστινατιον Νετωορκ Αδδρεσσ Τρανσλατιον*) используется для преобразования адреса места назначения в IP заголовке пакета; если пакет подпадает под критерий правила, выполняющего DNAT, то этот пакет, и все последующие пакеты из этого же потока, будут подвергнуты преобразованию адреса назначения и переданы на требуемое устройство, хост или сеть; действие DNAT может выполняться только в цепочках PREROUTING и OUTPUT таблицы nat, и во вложенных под-цепочках; важно отметить, что вложенные подцепочки, реализующие DNAT не должны вызываться из других цепочек, кроме PREROUTING и OUTPUT;

• ΔРОП □ |□⎝⌡⌠ □⟩⟨□◊⟩⌋®◊Σ∫□ |◊| Σ∫; □⟩⟨□⋃Σ⎨⎬⌋Σ□ |◊| Σ∫⌋ | ⎣⎬ ⎝⌡⎤⌋ прекращают свое движение, т.е. они не поступают на другие таблицы, в отличие от ACCEPT; данное действие может вызвать негативные последствия, поскольку могут остаться незакрытые сокеты как на стороне сервера, так и на стороне 20 клиента, приоритетнее использовать действия REJECT, особенно при защите от сканирования портов;

• ΛΟΓ □ ™Σ⌈⟩∫®\Σ,| ⌡∫⎝□⎝Σ ⟩⎣ ⌠⌠⌡ ™⎣□ ⌠ ⌠□⎨◊⎣\□⎝®◊⎨\□ ⎣™Σ⎣⎤⎬⌋⌋ пакетов и событий; в журнал могут заноситься заголовки IP пакетов и другая полезная информация; информация из журнала может быть затем прочитана с помощью dmesg или syslogd, либо с помощью других программ;

• МАРК □ ⎝⟩| ⎣⎦⎤| ⌠Σ∫⟩□ ™⎣□ ⌠⟩∫◊⎨ ⎝®| \ ⌠Σ∫⎣⎣ ™⎣□ ⎝| □Σ™Σ⎣Σ⎨⎬⌋⌋ пакетов; это действие может выполняться только в пределах таблицы mangle; установка меток обычно используется для нужд маршрутизации пакетов по различным маршрутам, для ограничения трафика и т.п.; “метка” пакета

существует только в период времени пока пакет не покинул брандмауэр, т.е. метка не передается по сети;

• MAΣΘYEPAΔE □ ® ⌊⟩⟨ ⌊®Σ ⟩®⌊Σ⌈ ⌈□Σ™⟩∫◊®⌊□Σ∫ ∫⌊ ⌈Σ ⟩◊⌈⌊Σ, ⌊∫⌊ и SNAT только не имеет ключа --to-source; причиной тому то, что маскарадинг может работать, например, с dialup подключением или DHCP, т.е. в тех случаях, когда IP адрес присваивается устройству динамически; если у вас имеется динамическое подключение, то нужно использовать маскарадинг, если же у вас статическое IP подключение, то лучше будет использование SNAT;

• MIPPOP □ ⌈⌊⌈Σ∫ ⌊⟩⌈ ⌊⌊⌉⌈ ⌊®◊∫⌉⟩□ ∫⌊⌊⌉⌈ ⌊ ™⌊□ ⟩⟨ ⟩⌈Σ□⌊⌈Σ⟨∫⌊®⌊ ® демонстрационных целях, так как это действие может привести к зацикливанию пакета и в результате к “Отказу от обслуживания”; в результате действия MIRROR в пакете, поля *source* и *destination* меняются местами и пакет отправляется в сеть; допускается использовать только в цепочках INPUT, FORWARD и PREROUTING;

• ΘYEYE □ ⟩∫◊®⌊∫ ⌈◊⌈ Σ∫ ® ⌊ ⌈Σ□Σ™⌉ ⟨◊ ⌊⟨□◊⟨⌊∫⌈ ⌈ ⌈⌊⌉⌈ ⌊®◊∫Σ⌊⌉⟩⌈ ⌊⌈ ⌈ процесс; может использоваться для нужд учета, дополнительной фильтрации пакетов или проксирования;

• PEΔIPEXT □ ®⌋⌈ ⌊⌊⟨ □Σ∫ ⌈Σ□Σ⟨◊⌈□◊®⌊Σ⟨⌊Σ ⌈◊⌈ Σ∫⌊®⌊⌈⌊∫⌊⌈ ⌊®⟨◊ другой порт текущей машины; можно пакеты, поступающие на HTTP порт перенаправить на порт HTTP-proxy; удобен для выполнения “прозрачного” проксирования (*transparent proxy*), когда машины в локальной сети даже не подозревают о существовании прокси; может использоваться только в цепочках PREROUTING и OUTPUT таблицы nat;

• PEϑEXT □ ⌊⟩⌈ ⌊⌊⌉⌈ ⌈Σ∫⟩□,⌈◊⌈ ⌈□◊®⌊⌊⌊, ® ∫Σ⌋ ⌈Σ ⟩◊⌈⌋⌋ ⟩⌊⌋ ⌈◊⟩⌊□⌋, ⌊∫⌊ и DROP, однако в отличие от DROP, команда REJECT выдает сообщение об ошибке на хост, отправивший пакет;

• PETYPN □ ⌈□Σ⌋ □◊⌉◊Σ∫ ™®⌊⌈Σ⟨⌊Σ ⌈◊⌈ Σ∫◊ ⌈⌊ ∫Σ⌋ ⌈⌉Σ⌈ ⌉Σ⌈⌊ ⌈⌊ Σ ⌈□◊®⌊⌊ ⌊ про изводит возврат следующему правилу в вызывающей (предыдущей) цепочке, если текущая цепочка была вложенной, или, если текущая цепочка лежит на

самом верхнем уровне (например INPUT), то к пакету будет применена политика по-умолчанию; обычно, в качестве политики по-умолчанию назначают действия ACCEPT или DROP;

• ΣNAT □ ⎩⎭| ⎣⎦|| ⎡∑∫⎭□ ™⎣□ |□∑⎩⟨□◊| ⎩®◊⎨⎩□ ⎭∑∫∑®⎦⎦ ◊™□∑⎭⎩® (*Σουρχε Νετωορκ Αδδρεσσ Τρανσλατιον*), т.е. изменение исходящего IP адреса в IP в заголовке пакета; SNAT допускается выполнять только в таблице nat, в 21 цепочке POSTROUTING; если исходящий адрес первого пакета в соединении преобразовался, то все последующие пакеты, из этого же соединения, будут преобразованы автоматически и не пойдут через эту цепочку правил;

• ΤΟΣ □ ⎩⎭| ⎣⎦|| ⎡∑∫⎭□ ™⎣□ ⎡⎭∫◊⎨ ⎩®| ⎩⟨⎧⎩® ® | ⎣⎦∑ *Τψπε οφ Σερϖιχε* ΙΠ | ◊©⎩⎣⎩®| ◊; | ⎣⎦∑ ΤΟΣ ⎭⎩™∑□⎡⎧ 8 ⟨⎧,| ⎧⎩□⎦∑ ⎩⎭| ⎣⎦|| ⎡⎦∫⎭□ ™⎣□ маршрутизации пакетов; важно помнить, что данное поле может обрабатываться различными маршрутизаторами с целью выбора маршрута движения пакета; в отличие от MARK, сохраняет свое значение при движении по сети, а поэтому может использоваться для маршрутизации пакета; лучше всего использовать это поле для своих нужд только в пределах WAN или LAN**;**

• ΤΤΛ □ ⎩⎭| ⎣⎦|| ⎡∑∫⎭□ ™⎣□ ⎧ ⎡∑⎨∑⎨⎩□ ⎭⎩™∑□⎡⎧⎡⎩©⎩ | ⎣⎦□ *Τιμε Το Λιϖε* в IP заголовоке;

• ΥΛΟΓ □ ⎭⎩⎧∫∑⎡◊ ⎛ ⎡□⎨ ◊⎣⎩□⎩®◊⎨⎩□ |◊| ∑⎧⎩®,| ⎧⎩□◊□ | ◊⎡∑⎨ □∑⎧ традиционно е действие LOG, базирующееся на системном журнале; при использовании этого действия, пакет, через сокеты netlink, передается специальному демону который может выполнять очень детальное журналирование в различных форматах (обычный текстовый файл, база данных MySQL и пр.), может формировать отчёты в CSV, XML**,** Netfilter’s LOG, Netfilter’s conntrack [4]**;** В листинге 2.1 приводится пример конфигурационного файла iptables ВС Unicluster. Листинг 2.1 – Конфигурационный файл iptables на ВС Unicluster.

```
*filter :OUTPUT ACCEPT [0:0] :FORWARD ACCEPT [0:0]
:INPUT ACCEPT [0:0] :Firewall - [0:0] -A INPUT -j
Firewall # localhost traffic -A Firewall -i lo -j
```

```
ACCEPT # Traffic from nodes (NFS, NTP, Web, ...) -A
Firewall -i eth0 -j ACCEPT -A Firewall -i eth1 -j
ACCEPT # ICMP -A Firewall -p icmp -m icmp --icmp-type
any -j ACCEPT # DNS -A Firewall -p tcp -m tcp --dport
53 -j ACCEPT --syn -A Firewall -p udp -m udp --dport 53
-j ACCEPT -A Firewall -p udp -m udp -s 0/0 -d 0/0 --
sport 53 -j ACCEPT # SSH -A Firewall -p tcp -m tcp -s
0/0 --dport 22 -j ACCEPT # FTP #-A Firewall -p tcp -m
tcp -s 0/0 --dport 21 -j ACCEPT #-A Firewall -m state -
-state NEW -m tcp -p tcp --dport 60000:64000 -j ACCEPT
# Web -A Firewall -p tcp -m tcp -s 0/0 --dport 80 -j
ACCEPT # REJECT 22 -A Firewall -j REJECT COMMIT # #
SNAT for nodes # *nat -A POSTROUTING -o eth1 -s
192.168.10.0/24 -j SNAT --to 91.196.245.221 COMMIT
```

Запуск службы iptables: `$ service iptables start` 2.4 DNS DNS – это служба, которая позволяет преобразовать буквенный адрес (доменное имя) в IP-адрес и обратно. Процесс преобразования доменного имени в IP-адресс происходит следующим образом: 1. Клиент (Служба DNS, установленная, на компьютере) отправляет запрос к DNS-серверу, который прописан на компьютере "Какой ip у адреса yandex.ru?" 2. DNS-Сервер, к которому дошел запрос клиента не знает IP-адреса сервера yandex.ru. В месте с этим он не знает какой DNS ответственен за данный домен. DNS-сервер отправляет запрос к корневому DNS-серверу "Какой ip у адреса yandex.ru?" 3. Корневой DNS-сервер отвечает: ?Нет, я не знаю IP нужного сервера, но я знаю DNS сервер, который отвечает за зону RU-можете спросить у него. 4. DNS сервер A продолжает обработку запроса и идет к серверу, который отвечает за доменную зону RU. Этот сервер тоже не знает какой адрес у yandex.ru. 5. DNS зоны RU отвечает-"Нет, я не знаю нужного адреса, но я знаю сервер, который отвечает за домен Russian.yandex - можешь спросить у него" 6. Только после этого DNS клиента обращается к

DNS-серверу, отвечающему за зону russian.yandex. Этот DNS russian.yandex отвечает не только за домен russian.yandex, но и за wiki.russian.yandex.ru, forum.russian.yandex.ru и т.д. Поэтому он отвечает серверу A сообщением "Доменному имени wiki.russian.yandex.ru соответствует ip адрес 89.108.109.5" 7. И DNS -сервер A пересылает этот ответ клиенту. На этом процесс резолвинга заканчивается. 2.4.1 Типы записей DNS Записи DNS, или Ресурсные записи (англ. Resource Records, RR) — единицы хранения и передачи информации в DNS. Каждая ресурсная запись состоит из следующих полей: • имя (NAME) — доменное имя, к которому привязана или которому «принадлежит» данная ресурсная запись, 23
• TTΛ (Τιμε Το Λιϖε) □ ™⌊⎪ ∫⟩∫⌊⌈⌊∑ ®□∑⌈□ ⌋ □◊⦚ ∑⦚⌊□ ™◊⦚⦚ ⌊⌈ □∑⟩ ⌈□⟩⦚ ⌊⌈ за писи в кэше неответственного DNS-сервера, • тип (TYPE) ресурсной записи — определяет формат и назначение данной ресурсной записи, • класс (CLASS) ресурсной записи; теоретически считается, что DNS может использоваться не только с TCP/IP, но и с другими типами сетей, код в поле класс определяет тип сети, • длина поля данных (RDLEN), • поле данных (RDATA), формат и содержание которого зависит от типа записи. Наиболее важные типы DNS-записей: • Запись A (*address record*) или запись адреса связывает имя хоста с адресом IP. Например, запрос A-записи на имя referrals.icann.org вернет его IP адрес — 192.0.34.164 • Запись AAAA (*IPv6 address record*) связывает имя хоста с адресом протокола IPv6. Например, запрос AAAA-записи на имя K.ROOTSERVERS.NET вернет его IPv6 адрес — 2001:7fd::1 • Запись CNAME (*canonical name record*) или каноническая запись имени (псевдоним) используется для перенаправления на другое имя • Запись MX (*mail exchange*) или почтовый обменник указывает сервер(ы) обмена почтой для данного домена. • Запись NS (*name server*) указывает на DNS-сервер для данного домена. • Запись PTR (*pointer*) или запись указателя связывает IP хоста с его каноническим именем. Запрос в домене in-addr.arpa на IP хоста в reverse форме вернёт имя (FQDN) данного хоста (см. Обратный

DNS-запрос). Например, (на момент написания), для IP адреса 192.0.34.164: запрос записи PTR 164.34.0.192.inaddr.arpa вернет его каноническое имя referrals.icann.org. В целях уменьшения объёма нежелательной корреспонденции (спама) многие серверы-получатели электронной почты могут проверять наличие PTR записи для хоста, с которого происходит отправка. В этом случае PTR запись для IP адреса должна соответствовать имени отправляющего почтового сервера, которым он представляется в процессе SMTP сессии. • Запись SOA (*Start of Authority*) или начальная запись зоны указывает, на каком сервере хранится эталонная информация о данном домене, содержит контактную информацию лица, ответственного за данную зону, *тайминги* (параметры времени) кеширования зонной информации и взаимодействия DNS-серверов. • Запись SRV (*server selection*) указывает на серверы для сервисов, используется, в частности, для Jabber и Active Directory [7]. 2.4.2 Настройка сервера DNS на Fedora 23 Установка пакетов `$ yum install bind 24` Также необходимо открыть 53 порт в iptables. Конфигурационные файлы располагаются в /var/named/. named.ca содержит адреса всех корневых DNS-серверов. named.empty - шаблон конфигурационного файла. named.localhost - зона localhost. named.loopback - файл обратной зоны 127.0.0.1. В листингах 2.2, 2.3, 2,4, 2,5 приводятся содержимое конфигурационных файлов DNS на ВК Unicluster. Листинг 2.2 – Конфигурация named.conf для кластера Unicluster.

```
options { listen-on port 53 { 127.0.0.1; 192.168.10.254; }; directory "/var/named"; dump-file "/var/named/data/cache_dump.db"; statistics-file "/var/named/data/named_stats.txt"; memstatistics-file "/var/named/data/named_mem_stats.txt"; allow-query { localhost; 192.168.10.0/24; }; forwarders { 91.196.245.221; 195.149.200.230; }; recursion yes; dnssec-enable yes; dnssec-validation yes; dnssec-
```

```
lookaside auto; /* Path to ISC DLV key */ bindkeys-file
"/etc/named.iscdlv.key"; }; logging { channel
default_debug { file "data/named.run"; severity
dynamic; }; }; view "internal" { match-clients {
localhost; 192.168.10.0/24; }; recursion yes; zone "."
IN { type hint; file "named.ca"; }; include
"/etc/named.rfc1912.zones"; include
"/etc/named.cluster.zones"; };
```

Листинг 2.3 – Конфигурационный файл named.cluster.zones для кластера Unicluster

```
zone "cluster.local"
{ type master; file "named-cluster.local.zone"; }; 25
zone "10.168.192.in-addr.arpa" { type master; file
"named-10.168.192.zone"; allow-update { none; }; };
```

Файл зоны будет прописан в файле /var/named/named-cluster.local.zone.

Листинг 2.4 – Конфигурационный файл cluster.local для кластера Unicluster

```
$TTL 1W @ IN SOA ns root ( 2016040202 ; serial 3H ;
refresh 15M ; retry 1W ; expiry 1D ) ; minimum @ IN NS
unicluster.cpct.sibsutis.ru. frontend IN A
192.168.10.254 cn1 IN A 192.168.10.1 cn2 IN A
192.168.10.2 cn3 IN A 192.168.10.3 cn4 IN A
192.168.10.4
```

Файл зоны 10.168.192.in-addr.arpa будет прописан в файле /var/named/named-10.168.192.zone соответственно. Листинг 2.5 – Конфигурационный файл 10.168.192.in-addr.arpa для кластера Unicluster

```
$TTL 3H @ IN SOA ns root ( 2016040202; serial (d.
adams) 3H ; refresh 1H ; retry 1W ; expiry 1H ) ;
minimum @ IN NS unicluster.cpct.sibsutis.ru. 254 IN PTR
frontend.cluster.local. 1 IN PTR cn1.cluster.local. 2
IN PTR cn2.cluster.local. 3 IN PTR cn3.cluster.local. 4
IN PTR cn4.cluster.local. 2.5 NFS NFS (Network File
System) – сетевая файловая система, решение в локальной
```

сети, где необходим быстрый обмен данными. Протокол NFS позволяет

монтировать удалённые файловые системы через сеть в локальное дерево каталогов, как если бы это была примонтирована дисковая файловая система. Тем самым приложения могут работать с удаленной файловой системой, как с локальной. 26 2.5.1 Настройка NFS на BC Unicluster Посредством NFS необходимо примонтировать каталоги frontend-a /home и /opt ко всем вычислительным узлам cn[1-4]. На frontend устанавливаем NFSсервер. `$ dnf -y install nfs-utils` Далее прописываем в файл /etc/exports какие клиенты могут обращаться к файлам на сервере, а также к каким каталогам и с какими правами будет предоставляться доступ клиентам: `/home 192.168.1.0/255.255.255.0(rw) /opt 192.168.1.0/255.255.255.0(ro)` Запуск NFS-сервера. `$ systemctl start nfs-server $ systemctl enable nfs-server` Далее необходимо настроить NFS-клиенты на всех вычислительных узлах. Инсталлируем NFS-клиенты. `$ dnf -y install nfs-utils` Для того, чтобы каталоги монтировались при запуске системы необходимо в конец файла /etc/fstab прописать строки, приведенные ниже. `frontend:/home /home nfs rw 0 0 frontend:/opt /opt nfs ro 0 0` Запуск NFS-клиента. `$ systemctl start nfs-mountd $ systemctl enable nfs-mountd 2.6 NTP Network Time Protocol — сетевой протокол, который позволяет` синхронизировать внутренние часы компьютера с применением сетей с переменной латентностью, основанных на коммутации пакетов. Традиционно для работы NTP используется протокол UDP, он также может работать и по средствам TCP. Время в системе NTP представляется 64-битным числом, где 32-битный счётчик секунд и 32-битный счётчик долей секунды, что позволяет передавать время в диапазоне 232 сек., с точностью 2- 32 сек.. В связи с тем, что временная шкала в NTP повторяется каждые 232 секунды (136 лет), пользователю необходимо хотя бы примерно знать текущее время (с точностью 50 лет). 2.6.1 Как работает NTP Серверы NTP

работают в иерархической сети, каждый уровень иерархии называется ярусом (stratum). Нулевой ярус – эталонные часы. Эталоном является сигнал GPS (Global Positioning System) или службы ACTS (Automated Computer Time Service). Серверы NTP не работают на нулевом ярусе. Серверы NTP первого 27 яруса синхронизируются с эталонными часами, получая данные о времени. Серверы NTP второго яруса получают данные от серверов первого яруса. Количество ярусов может достигать 15. За счет иерархической структуры протокол NTP является отказоустойчивым и избыточным. Два сервера второго яруса синхронизируются с шестью различными серверами первого яруса, каждый из которых работает по независимому каналу. Узлы внутри системы синхронизируются с внутренними серверами соответственно. Два сервера второго яруса координируют время друг с другом. В том случае, когда линия связи с сервером первого яруса откажет избыточный сервер второго яруса берет на себя процесс синхронизации. Аналогично устройства и узлы третьего яруса могут использовать сервера второго яруса. За счет наличия избыточной сети NTP-серверов гарантируется постоянная доступность серверов времени. Для расчета наиболее точного времени, NTP-сервер использует данные от всех источников, с которыми он синхронизируется. Протокол NTP не устанавливает время в чистом виде. NTP лишь корректирует локальные часы, используя временное смещение, разницы между временем на локальных часах и на сервере NTP. Если временное смещение слишком велико (более 1000 с.), то NTP сервер или клиент прекращают настройку времени и ожидают вмешательства администратора. 2.6.2 Настройка NTP на BC Unicluster Необходимо настроить иерархию таким образом, чтобы сервер, установленный на frontend, синхронизировался с серверами fedora.pool.ntp.org , а узлы вычислительной системы cn[1-4] синхронизировались с сервером, установленным на frontend-е. Инсталлируем ntpd на все узлы вычислительной системы: `$ dnf -y install ntp` Конфигурация NTP сервера настраивается в файле /etc/ntp.conf, для сервера на fronted ntp.conf

описан в листинге 2.6 (приведены лишь значащие строки конфигурационного файла). Листинг 2.6 – Конфигурационный файл ntp.conf для кластера Unicluster `restrict 192.168.10.0 mask 255.255.255.0 nomodify notrap # Permit all access over the loopback interface. This could # be tightened as well, but to do so would effect some of # the administrative functions. restrict 127.0.0.1 restrict ::1 # Hosts on local network are less restricted. #restrict 192.168.10.0 mask 255.255.255.0 nomodify notrap # Use public servers from the pool.ntp.org 28 project. # Please consider joining the pool (http://www.pool.ntp.org/join.html). server 0.fedora.pool.ntp.org iburst server 1.fedora.pool.ntp.org iburst server 2.fedora.pool.ntp.org iburst server 3.fedora.pool.ntp.org iburst` Конфигурационный файл для вычислительных узлов отличается тем, что в качестве сервера для них выступает сервер, установленный на frontend: `# Use public servers from the pool.ntp.org project. # Please consider joining the pool (http://www.pool.ntp.org/join.html). server 192.168.10.254` Запуск NTP сервера. `$ systemctl start ntpd $ systemctl enable ntpd 2.7 NIS` Служба NIS (Network Information Service – сетевая информационная служба) упрощает администрирование распространенных административных файлов, храня их в центральной базе данных и позволяя клиентам обращаться к серверу базы данных для выборки информации. Так же, как DNS предназначена для решения проблемы актуальности множества хранимых копий файлов /etc/hosts, NIS предназначена для поддержания актуальности не зависящих от системы конфигурационных файлов (таких как /etc/passwd). Основной целью администратора LAN является обеспечение прозрачности сети для пользователей. Один из аспектов этой прозрачности – предоставление

пользователям похожих сред, в том числе пользовательских имен и паролей при входе в различные машины. С точки зрения администратора, информация, поддерживающая среду пользователя, не должна реплицироваться, но должна храниться в центральном месте и распространяться по запросам, NIS упрощает эту задачу. 2.7.1 Настройка клиента NIS В первую очередь необходимо инсталлировать следующие пакеты. yp-tools ypbind Далее указываем имя домена NIS, для этого выполняем команду. `$ ypdomainname unicluster` В файле /etc/sysconfig/network добавляем "NISDOMAIN=unicluster". Запускаем ypbind. `$ systemctl start ypbind $ systemctl enable ypbind 29 2.7.2 Настройка сервера NIS` По аналогии с клиентом инсталлируем пакеты: yp-tools ypserv Далее указываем имя домена NIS, для этого выполняем команду. `$ ypdomainname unicluster` В файле /etc/sysconfig/network добавляем "NISDOMAIN=unicluster". Запускаем ypbind: `$ systemctl start ypbind $ systemctl enable ypbind` Необходимо обновить базу данных NIS. Листинг 2.7 – Процесс обновления базы данных NIS `$ /usr/lib64/yp/ypinit -m At this point, we have to construct a list of the hosts which will run NIS servers. dlp is in the list of NIS server hosts. Please continue to add the names for the other hosts, one per line. When you are done with the list, type a <control D>. next host to add: dlp.srv.world next host to add: # Ctrl + D key The current list of NIS servers looks like this: unicluster Is this correct? [y/n: y] y # yes We need a few minutes to build the databases... Building /var/yp/srv.world/ypservers... Running /var/yp/Makefile... gmake[1]: Entering directory `/var/yp/srv.world' Updating passwd.byname... Updating passwd.byuid... Updating shadow.byname... Updating group.byname... Updating group.bygid... Updating`

```
hosts.byname... Updating hosts.byaddr... Updating
rpc.byname... Updating rpc.bynumber... Updating
services.byname... Updating services.byservicename...
Updating netid.byname... Updating protocols.bynumber...
Updating protocols.byname... Updating mail.aliases...
gmake[1]: Leaving directory `/var/yp/srv.world'
dlp.srv.world has been set up as a NIS master server.
Now you can run ypinit -s dlp on all slave server. 30
```

Далее переходим в директорию /var/yp и делаем make. Процесс установки NIS завершен.

Конфигурирование системы управления ресурсами

SLURM – это высокомасштабируемый отказоустойчивый менеджер кластеров и планировщик заданий для больших кластеров вычислительных узлов. SLURM поддерживает очередь ожидающих заданий и управляет общей загрузкой ресурсов в процессе выполнения работы. Также SLURM управляет доступными вычислительными узлами в эксклюзивной или неэксклюзивной форме (как функция потребности в ресурсах). Наконец, в дополнение к мониторингу параллельных заданий вплоть до их завершения SLURM распределяет нагрузку по выделенным узлам. Не вдаваясь в подробности, можно сказать, что SLURM - это надежный, переносимый, масштабируемый на большое количество узлов, отказоустойчивый и, что самое важное, открытый менеджер кластеров (ориентированный больше на необходимые функции, чем на обеспечение дополнительных возможностей). SLURM начинал совместно разрабатываться несколькими компаниями (в их число входила Ливерморская национальная лаборатория имени Э. Лоуренса) как Open Source-менеджер ресурсов. Сегодня SLURM является лидером среди менеджеров ресурсов и используется на многих самых мощных суперкомпьютерах.

Архитектура SLURM реализована типичная архитектура управления кластером (см. рисунок 10). Верхний уровень управления – это резервированная пара контроллеров кластера (хотя резервирование не является обязательным). Эти контроллеры управляют вычислительным кластером и содержат демон управления под названием slurmctld. Демон slurmctld следит за вычислительными ресурсами, но, что более важно, он занимается распределением этих ресурсов между разными заданиями. Каждый вычислительный узел содержит демон под названием slurmd. Демон slurmd управляет узлом, в котором он запущен, в том числе занимается мониторингом выполняющихся на узле заданий, получением заданий от контроллера и их распределением по ядрам внутри узла. Кроме того, slurmd останавливает выполнение заданий по запросу контроллера. Внутри архитектуры существуют и другие демоны – например, демоны для безопасной аутентификации. Тем не менее, кластер – это не просто случайный набор узлов, поскольку в каждый момент времени часть из них логически связана с определенной задачей параллельных вычислений.

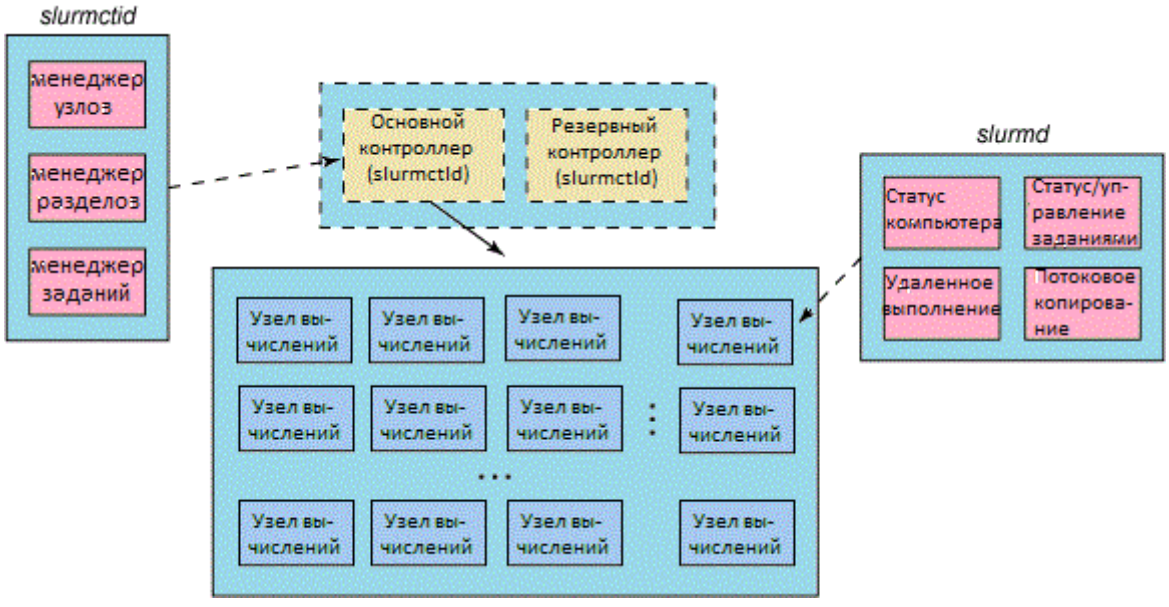

Рис. 10 – Высокоуровневое представление архитектуры SLURM

Набор узлов можно объединить в логическую группу, называющуюся разделом и обычно включающую в себя очередь входящих заданий. Для разделов можно задавать ограничения, например, указывать пользователей, которые могут х использовать, размер задания или предельный срок обработки. Другая особенность раздела заключается в том, что пользователю на определенный период времени выделяется часть узлов для выполнения работы, которая называется заданием. Каждое задание содержит один или несколько шагов,представляющих собой наборы задач, выполняющихся на выделенных узлах. Эта иерархия, более подробно показывающая разделение ресурсов на разделы в SLURM, изображена на рисунке 3. Обратите внимание на то, что при таком разделении учитывается близость ресурсов, что позволяет обеспечить низкие задержки при взаимодействии совместно работающих узлов.

Установка SLURM на BC Unicluster С официального сайта проекта SLURM (http://slurm.schedmd.com/) скачивается последняя версия реализации менеджера управления ресурсами SLURM 16.05. После того, как архив скачен, его необходимо распаковать и сконфигурировать SLURM в директорию /opt, в которой создается папка slurm-16.05.0, куда и будет инсталлироваться менеджер. В распакованном архиве необходимо выполнить следующие команды.

```
$ ./configure --prefix=/opt/slurm-16.05.0

$ make $ make install
```

После этого менеджер управления ресурсами SLURM установиться в указанную директорию. SLURM состоит из трех демонов:

slurmctld.service – управляющий демон SLURM;

slurmd.service – демон, работающий на узлах вычислительной системы;

Далее приведено содержание файлов slurmctld.service и slurmd.service соответственно.

Листинг 2.8 – Файл slurmctd.service для кластера Unicluster

```
[Unit]
Description=Slurm controller daemon
After=network.target
ConditionPathExists=/opt/slurm-16.05.0/etc/slurm.conf
[Service]
Type=forking
EnvironmentFile=-/etc/sysconfig/slurmctld
ExecStart=@sbindir@/slurmctld $SLURMCTLD_OPTIONS
ExecReload=/bin/kill -HUP $MAINPID
PIDFile=/var/run/slurmctld.pid
[Install]
WantedBy=multi-user.target
```

Листинг 2.9 – Файл slurmd.service для кластера Unicluster

```
[Unit]
Description=Slurm node daemon
After=network.target
ConditionPathExists=/opt/slurm-16.05.0/etc/slurm.conf
[Service]
Type=forking
EnvironmentFile=-/etc/sysconfig/slurmd
ExecStart=@sbindir@/slurmd $SLURMD_OPTIONS
ExecReload=/bin/kill -HUP $MAINPID
PIDFile=/var/run/slurmd.pid
KillMode=process
LimitNOFILE=51200
LimitMEMLOCK=infinity
```

```
LimitSTACK=infinity
[Install]
33
WantedBy=multi-user.target
```

Далее необходимо добавить демонов slurmctld.service и slurmd.service в systemd. Для для этого необходимо вышеперечисленные файлы поместить в каталог /usr/lib/systemd/system и выполнить следующие команды.

```
$ systemctl enable
/usr/lib/systemd/system/slurmd.service
$ systemctl start slurmd.service
$ systemctl enable
/usr/lib/systemd/system/slurmctld.service
$ systemctl start slurmctld.service
```

Демон slurmd запускается на узлах вычислительной системы, а slurmctld на управляющем узле frontend.

2.9 Установка средств разработки

В связи с тем, что студенты будут использовать вычислительный кластер Unicluster для разработки и исследования многопоточным программ, необходимо
установить средство, с помощью которого будут создаваться параллельные программы. Одним из таких решений – Mpich. Mpich – реализация стандарта MPI.

Message Passing Interface (MPI) – стандарт программного интерфейса коммуникационных функций для создания переносимых параллельных программ
в модели передачи сообщений (message passing) [2].

2.9.1 Как работают параллельные программы на кластере

Параллельные программы на вычислительном кластере работают в модели передачи сообщений (message passing). Это значит, что программа состоит из множества процессов, каждый из которых работает на своем процессоре и

имеет
свое адресное пространство. Причем непосредственный доступ к памяти другого
процесса невозможен, а обмен данными между процессами происходит с помощью операций приема и посылки сообщений. То есть процесс, который должен получить данные, вызывает операцию Receive (принять сообщение), и
указывает, от какого именно процесса он должен получить данные, а процесс,
который должен передать данные другому, вызывает операцию Send (послать сообщение) и указывает, какому именно процессу нужно передать эти данные. Эта модель реализована с помощью стандартного интерфейса MPI. Существует несколько реализаций MPI, в том числе бесплатные и коммерческие, переносимые и ориентированные на конкретную коммуникационную сеть. Как правило, MPI-программы построены по модели SPMD (одна программа - много данных), то есть для всех процессов имеется только один код программы, а различные процессы хранят различные данные и выполняют свои действия в зависимости от порядкового номера процесса.

Установка MPICH 3.1.4

Для установки MPICH необходимо скачать дистрибутив с официального сайта mpich.org. В разделе Downloads выбираем последнюю версию 3.1.4 и скачиваем архив mpich-3.1.4.tar.gz. Разархивируем его в текущей директории.

```
$ tar xvf mpich-3.1.4.tar.gz
```

Устанавливаем mpich в каталог /opt, для этого мы должны собрать пакет с следующими опциями.

```
    $ ./configure –prefix=/opt/mpich-3.1.4
```

Далее компилируем сборку.

```
$ make
```

И наконец, запускаем инсталляцию инсталлируем.

```
$ make install
```

Процесс инсталляции пакета mpich, являющийся реализацией стандарта MPI, завершен.

Заключение

В ходе проделанной работы был сконфигурирован вычислительный кластер на базе нескольких одноплатных компьютеров и выполнена его настройка. Были изучены детали использования одноплатных компьютеров, современные подходы к администрированию вычислительных кластеров, детали установки и настройки служб, необходимых для создания вычислительного кластера.