На правах рукописи

Чубахиро Амисси

СИСТЕМА УПРАВЛЕНИЯ РАСПРЕДЕЛЕННЫМИ ВИРТУАЛЬНЫМИ КЛАСТЕРАМИ

Специальность: 05.13.15 - Вычислительные машины, комплексы и компьютерные сети

Автореферат

диссертации на соискание ученой степени

кандидата технических наук

Санкт-Петербург – 2019

Работа выполнена в ФГАОУ ВО «Санкт-Петербургский государственный электротехнических университет «ЛЭТИ» им. В.И. Ульянова (Ленина)», кафедра вычислительной техники

Научный руководитель: доктор технических наук, доцент **Дегтярев Александр Борисович**, профессор кафедры компьютерного моделирования и многопроцессорных систем ФГАОУ ВО «Санкт-Петербургский государственный университет», г. Санкт-Петербург.

Официальные оппоненты:

Гордеев Александр Владимирович, доктор технических наук, профессор ФГАОУ ВО «Санкт-Петербургский государственный университет аэрокосмического приборостроения», профессор кафедры вычислительных систем и сетей, г. Санкт-Петербург;

Петров Олег Николаевич, к.т.н., доцент, ФГАОУ ВО «Санкт-Петербургский государственный морской технический университет», доцент кафедры вычислительной техники и информационных технологий, г. Санкт-Петербург.

Ведущая организация:

Объединенный институт ядерных исследований (ОИЯИ), г. Дубна.

Защита состоится “24” декабря 2019 года в 15-30 часов на заседании диссертационного совета Д 212.238.01 Санкт-Петербургского государственного электротехнического университета «ЛЭТИ» им. В.И. Ульянова (Ленина) по адресу: 197376, Санкт-Петербург, ул. Проф. Попова, 5.

С диссертацией можно ознакомиться в библиотеке ФГАОУ ВО «Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина)» и на сайте университета www.etu.ru в разделе «Подготовки кадров высшей квалификации» - «Объявление о защитах».

Автореферат разослан “23” октября 2019 года.

Ученый секретарь
Диссертационного совета Д 212.238.01
канд. тех. наук, доцент А. А. Пазников

Актуальность темы и степень разработанности проблемы

Актуальность темы обусловлена тем фактом, что эффективная система управления кластером — ключ к решению многих проблем, возникающих в сфере распределенных вычислений. Широкое распространение вычислительных кластеров привело к активной разработке систем управления задачами, однако, многие вопросы, в частности вопросы миграции задач, до сих пор не решены. В данной работе рассматривается вариант организации виртуального кластера, созданного с помощью виртуализации на уровне операционной системы (ОС). Виртуализация на уровне ОС — перспективная технология, предлагающая выгодное соотношение «производительность/изолированность». В рамках решений данного типа обеспечивается высокая производительность, сравнимая с производительностью работы обыкновенной ОС, за счет низких накладных расходов. Однако за счет меньшей изолированности отдельных сущностей (обычно называемых контейнерами - аналог гостевой ОС в случае аппаратной виртуализации), как следствие, потенциально более низкой безопасности. К примеру, в случае Linux работа контейнеров обеспечивается изменениями в ядре ОС, позволяющими создавать отдельные пространства имен — «namespaces». Все контейнеры разделяют общее ядро ОС. И в общем случае возможность использования других «гостевых ОС», как правило, ограничена. Но довольно часто требуется лишь запуск различных сервисов в отдельных контейнерах. Указанные изменения в ядре позволяют гибко конфигурировать настройки контейнеров (к примеру, они все могут разделять единое сетевое пространство имен, а могут работать в отдельных сетевых пространствах имен с разными виртуальными сетевыми адаптерами). Высокая производительность при необходимом уровне безопасности и гибкость конфигурирования в пределах данной ОС — основные причины использования данного типа виртуализации.

В то же время многие вопросы в данной области остаются открытыми. Данный тип виртуализации используется все шире в рамках различных ОС. Развиваются комплексные решения, использующие данный тип виртуализации

(например, Docker, Singularity). В данный момент можно наблюдать активное развитие этого направление, формирование новых ветвей.

Динамическая миграция отдельных процессов и контейнеров — сложная задача, озвученная в прошлом, однако, так и не нашедшая полноценной реализации в промышленных системах. Сложность этой задачи в рамках кластера обусловлена жесткими требованиями, большим набором параметров состояния процессов и контейнеров, возможностью использования специализированного оборудования. Возможность восстановления состояния контейнера на другом узле — непростая задача, требующая разработки и реализации множества подсистем. Миграция контейнеров и процессов намного сложнее миграции привычных виртуальных машин из-за тесной интеграции в ОС и возможности работы с отдельными компонентами оборудования напрямую: потребуется восстановление состояния отдельных подсистем, в то время как в случае традиционной виртуальной машины (ВМ); ВМ «работает» с виртуальным оборудованием, предоставленным гипервизором, а состояние гостевой ОС находится «внутри» самой ВМ. Миграция процессов и контейнеров — активно развивающееся направление в настоящее время.

Виртуальные вычислительные сети — широко распространенный тип сетей. Отдельные компоненты могут использоваться для решения самых разных задач. В качестве примера можно привести виртуальный сетевой мост Linux, который способен объединить как физические узлы в сети (при их подключении через физические сетевые интерфейсы узла), так и виртуальные машины. Поиск эффективной, производительной архитектуры виртуальной сети — важная задача, ведь от ее решения зависит производительность виртуальных кластеров.

Исследованию виртуализации на уровне ОС и отдельным ее аспектам посвящен ряд научных статей российских и зарубежных исследований. Многие проприетарные решения используют возможности данного направления. Пристальное внимание как со стороны научной среды, так и со стороны бизнеса говорит об актуальности данной темы в настоящий момент, ведь решение

вопросов в этой сфере позволит получить эффективные решения при низких накладных расходах.

Объект исследования

Вычислительные кластеры.

Предмет исследования

Система управления виртуальными вычислительными кластерами, использующими виртуализацию на уровне ОС.

Цель работы

Создание системы управления виртуальными кластерами, построенными на основе разнородных контейнеров.

Задачи исследования

1. Разработка методики управления распределенными разнородными виртуальными кластерами, использующими виртуализацию на уровне ОС;
2. Разработка методики запуска задач с возможностью динамической миграции;
3. Разработка архитектуры виртуальной вычислительной сети;
4. Реализация программных компонентов для развертывания архитектуры;
5. Тестирование архитектуры на различных классах задач.

Методы исследования

Методы математической статистики, теория графов, теория вероятности, технологии проектирования программного обеспечения.

Научная новизна

1. Предложена методика управления распределенными разнородными виртуальными кластерами, использующими виртуализацию на уровне ОС;
2. Предложена методика обеспечения надежной и отказоустойчивой работы и балансировки нагрузки виртуальных вычислительных кластеров за счет динамической миграции задач в рамках виртуального кластера;
3. Предложена архитектура виртуальной вычислительной сети для вычислительного кластера, минимизирующая накладные расходы, связанные с обменом данными, под конкретную задачу.

Практическая значимость

Практическая значимость состоит в том, что результаты исследования могут быть применены для широко распространенного в настоящее время типа вычислительных систем: виртуальных кластеров, использующих виртуализацию на уровне ОС. Реализация разработанных методик для случая Linux оправдана распространенностью и надежностью данной системы. Виртуальные вычислительные сети нашли применение в современных вычислительных комплексах для широкого класса задач.

Таким образом, перечисленные ниже пункты являются практически значимыми.

1. Методика управления распределенными разнородными виртуальными кластерами, использующими виртуализацию на уровне ОС.
2. Методика обеспечения надежной и отказоустойчивой работы и балансировки нагрузки вычислительных кластеров за счет динамической миграции задач в рамках виртуального кластера.
3. Архитектура виртуальной вычислительной сети для вычислительного кластера, минимизирующая накладные расходы, связанные с обменом данными, под конкретную задачу.

Положения, выносимые на защиту

1. Методика управления распределенными разнородными виртуальными кластерами, использующими виртуализацию на уровне ОС.
2. Методика обеспечения надежной и отказоустойчивой работы и балансировки нагрузки вычислительных кластеров за счет динамической миграции задач в рамках виртуального кластера.
3. Архитектура виртуальной вычислительной сети для вычислительного кластера, минимизирующая накладные расходы, связанные с обменом данными, под конкретную задачу.

Соответствие паспорту специальности

Работа соответствует пунктам 2, 4, 6 паспорта специальности 05.13.15.

Внедрение результатов

Работа выполнялась в рамках учебного вычислительного кластера кафедры компьютерного моделирования и многопроцессорных систем факультета ПМ-ПУ СПбГУ.

Апробация работы

На данный момент текущие результаты работы были представлены на 5 научных конференциях:

1. The 15th International Conference on Computational Science and Its Applications (ICCSA 2015). Канада, 2015 г.
2. 10th International Conference on Computer Science and Information Technologies (CSIT 2015). Армения, 2015 г.
3. Научно-техническая конференция профессорско-преподавательского состава СПбГЭТУ “ЛЭТИ”, 2016 г.
4. The 7th International Conference «Distributed Computing and Grid-technologies in Science and Education» (GRID 2016). Россия, 2016 г.
5. The 8th International Conference «Distributed Computing and Grid-technologies in Science and Education» (GRID 2018). Россия, 2018 г.

Публикации

Основные результаты работы были опубликованы в 7 статьях, из них 6 по теме диссертации, среди которых 3 публикации в ведущих рецензируемых изданиях, рекомендованных в действующем перечне ВАК. Также 3 публикации индексированы в Scopus. Доклады были озвучены на 5 международных, всероссийских и межвузовских научно-практических конференциях.

Структура и объем диссертации

Диссертация состоит из введения, 4 глав, заключения, списка используемой литературы. Работа содержит 114 страниц основного текста, 3 таблицы, 14 рисунков. Список использованной литературы включает в себя 100 наименований.

Основное содержание работы

Во **введении** приводится обоснование актуальности выбранной темы, показана научная новизна и практическая значимость выполненного исследования, представлены положения, выносимые на защиту.

В **первой главе** рассмотрены существующие решения в области виртуализации в целом и виртуализация на уровне ОС, в частности, решения для организации запуска задач на вычислительных кластерах. Проанализированы плюсы и минусы тех или иных решений. Приводится обоснование выбора виртуализации на уровне ОС как основы для создания виртуальных вычислительных кластеров, приводится обоснование выбора контейнеров в ОС Linux.

Во **второй главе** рассмотрены подходы к организации вычислений на кластере, приводится обоснование необходимости тщательного анализа аппаратных и программных средств для развертывания виртуальных кластеров контейнеров для обеспечения надежной и отказоустойчивой работы.

Создание виртуальных кластеров — комплексная задача, требующая решения многих вопросов. В данной работе рассматривается случай виртуализации на уровне ОС. Данный тип виртуализации предоставляет высокую производительность, однако, этот случай — один самых сложных с точки зрения выбора аппаратных и программных средств. В случае виртуализации на уровне ОС приходится выбирать пару «архитектура — ОС», которая бы оптимально подошла для решаемых задач. Нужно учитывать намного больше параметров, чем в случае других видов виртуализации, например, аппаратной. Для балансировки нагрузки необходимо предусмотреть возможность миграции контейнеров. Возможность выполнения миграции для контейнера зависит от еще большего числа параметров.

Как результат, для создания эффективной инфраструктуры требуется разбиение исходного множества узлов на кластеры, схожие как по аппаратным, так и по программным характеристикам. В рамках предлагаемой методики предусмотрен выбор подмножества физических узлов для развертывания

виртуальных кластеров. Полученное разбиение на отдельные кластеры предлагается подавать как входные данные для систем управления ресурсами.

Алгоритм «BIRCH» (Balanced Iterative Reducing and Clustering using Hierarchies) позволяет обрабатывать большие наборы данных. Он разбивает объекты на кластеры, вычисляет характеристики этих кластеров, а затем позволяет выполнить кластеризацию полученных кластеров с использованием различных алгоритмов, как если бы они были входными данными (объектами). Такой подход позволяет обрабатывать большие объемы данных в условиях ограниченных ресурсов. Для преставления кластера вводится понятие «Clustering Feature (CF)», которое позволяет представлять кластеры, используя построение деревьев. Узлы хранят указатели на другие поддеревья и CF, характеризующие их. Количество таких указателей ограничено параметрами, отражающими доступный объем памяти. В данном случае термин «кластер» из кластерного анализа будет соответствовать вычислительному кластеру, так как задачей является получение групп схожих узлов, то есть получение отдельных вычислительных кластеров.

Алгоритм «BIRCH» был выбран в качестве основы, так как реализации данного алгоритма дают хорошие результаты кластеризации при высокой производительности. Данный алгоритм учитывает ограниченность доступной оперативной памяти и обращение к диску.

Оптимизация алгоритма «BIRCH» для данного случая возможна благодаря следующим особенностям: среди данных нет шума (все узлы необходимо использовать); одинаковые узлы могут быть указаны один за другим во входных данных (это важно, так как результаты работы алгоритма могут меняться в зависимости от порядка поступления данных). Оптимизация в данном случае заключается в том, что фазы 2 и 3 алгоритма могут не выполняться, а значение «T» — Threshold value и полученное дерево сохраняются для будущих запусков при необходимости (при добавлении нового узла вычислительного кластера).

Таким образом, методику можно сформулировать следующим образом:

1. Провести разбиение узлов по признаку принадлежности к подсети (так как сеть имеет высокое влияние на производительность виртуальных кластеров).
2. Провести разбиение узлов в рамках каждой подсети по признакам «ОС» и «архитектура процессора».
3. В рамках каждой полученной группы выполнить кластеризацию узлов при помощи оптимизированной версии алгоритма BIRCH.
4. Полученные в результате работы алгоритма деревья сохранить для случая добавления новых физических узлов (для быстрого определения нужного кластера).
5. Конфигурацию итоговых кластеров подать как входные данные для системы управления ресурсами.
6. Для запуска задачи развернуть контейнеры на выбранных для задачи узлах с необходимым уровнем изоляции ресурсов.

   Особенности запуска контейнеров для задач описаны в главе 4.

В **третьей главе** рассматриваются варианты организации виртуальной вычислительной сети для кластера контейнеров. Сеть является одним из главных факторов, сдерживающих производительность вычислительного кластера, в том числе и виртуального, поэтому вопросам организации сети уделено особое внимание в данной работе. Кроме того, виртуальная сеть необходима для методики, рассматриваемой в четвертой главе.

Для организации виртуальной вычислительной сети было предложено четыре варианта архитектуры сети. Для работы контейнеров с сетью использовались интерфейсы veth и macvlan. При необходимости создания единой локальной сети «поверх» глобальной сети использовался интерфейс vxlan.

В первом варианте архитектуры виртуальной сети предлагается использование интерфейса veth. Это парный интерфейс, он предполагает наличие двух виртуальных устройств одинакового типа. Кадры, поступившие на одно устройство «появляются» на другом — с точки зрения ОС, она получает новый кадр, поступивший со второго устройства (как в случае получения кадра на

физическом сетевом интерфейсе). При этом передача кадров работает в обоих направлениях: с первого устройства на второе и со второго на первое. Одно из устройств «перемещается» в сетевое пространство имен, связанное с контейнером, а второе «остается» в сетевом пространстве имен по умолчанию. Таким образом можно наладить связь между хостовой ОС (сетевое пространство имен по умолчанию) и контейнером или двумя контейнерами.

Однако, как правило, требуется обмен данными между большим числом контейнеров или между контейнером и узлами во внешней сети. В этом случае, как правило, используют виртуальный интерфейс bridge. Это по сути виртуальный мост, работающий, как и обычный мост, на втором уровне модели OSI. Он передает поступившие кадры на нужный порт. Если порт назначения неизвестен, он передает кадры на все порты, кроме того, с которого поступил пакет. В мост также добавляют физический сетевой адаптер, подключенный ко внешней сети, благодаря чему обеспечивается связь со внешней сетью. Кроме этого к мосту возможно подключение tap-устройства, обеспечивающего сетью виртуальную машину.

В другом варианте предлагается использование интерфейса macvlan. Для данного случая использование виртуального моста не требуется, так как macvlan-устройство само может работать в режиме моста. Для такого устройства необходим выбор «нижележащего» устройства. Таким устройством, как правило, является физический сетевой адаптер. С одним «нижележащим» сетевым устройством может быть связно несколько устройств macvlan. Если все эти устройства работают в режим «bridge», то данные между ними будут передаваться, не покидая узла. Когда же требуется отправить пакет во внешнюю сеть, используется «нижележащее» устройство. При этом в случае macvlan проверяется MAC-адрес назначения кадра. При отправке кадра с macvlan-устройства в режим «bridge» если MAC-адрес назначения равен MAC-адресу одного из других macvlan-устройств, и это устройство также работает в режиме «bridge», кадр передается этому устройству — с точки зрения ОС она получит кадр на втором интерфейсе macvlan. Устройства macvlan могут быть

ассоциированы с разными сетевыми пространствами имен. Благодаря этому можно наладить связь между контейнерами: в рамках контейнеров будут использоваться macvlan-устройства в режиме «bridge», связанные с одним и тем же «нижележащим» устройством. Также есть возможность соединения и с виртуальными машинами: в случае, если ВМ использует виртуальное macvtap-устройство, и MAC-адрес виртуального сетевого адаптера ВМ равен MAC-адресу macvtap-устройства, возможно использование macvtap-устройства для обычного обмена данными. При этом можно обеспечить связь между ВМ и контейнерами (для этого macvtap-устройство также должно работать в режиме «bridge» и быть ассоциировано с тем же «нижележащим» устройством). Для связи со внешней сетью используется «нижележащий» физический сетевой интерфейс. На рисунке 1 приведен второй вариант архитектуры сети с использованием macvlan. С помощью macvtap-устройств возможно совместная работа контейнеров и ВМ.

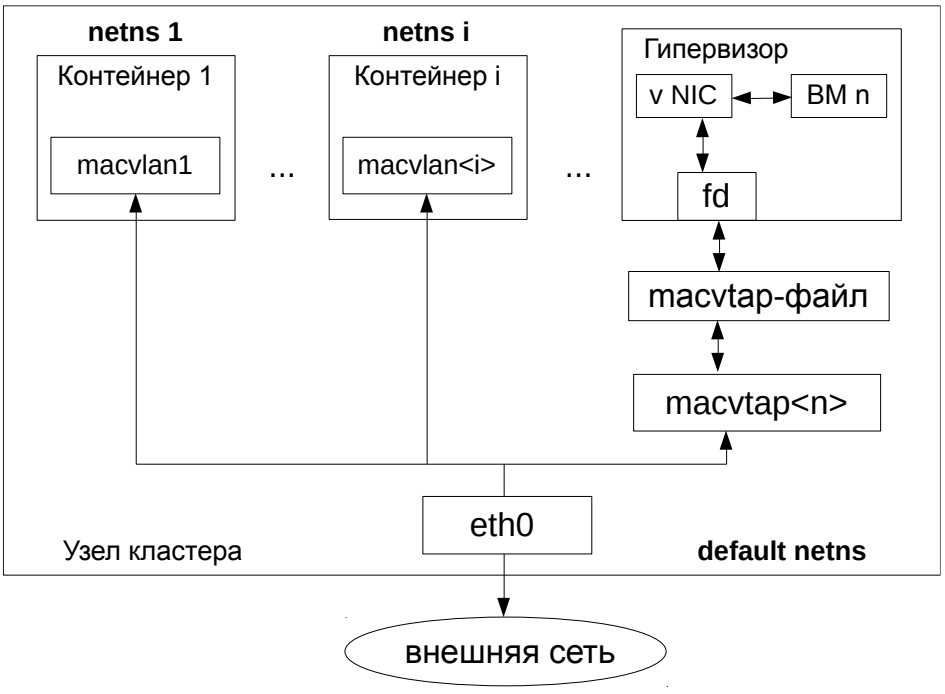

Рисунок 1. Архитектура сети с использованием macvlan

Для выбора архитектуры виртуальной вычислительной сети для кластера было выполнено тестирование различных вариантов организации виртуальной сети на двух различных типах вычислительных узлов. Рассматривались варианты использования физического сетевого адаптера без виртуализации, использование

двух виртуальных интерфейсов — veth и macvlan, а также использование этих интерфейсов при объединении узлов с помощью туннеля - при помощи виртуального интерфейса vxlan (интерфейс vxlan может быть использован и для случая одного физического интерфейса). В таблице 1 приведены результаты тестирования интерфейсов veth и macvlan на узлах второго типа и результаты для физического сетевого адаптера — «eth» (без использования виртуальных сетевых устройств).

Таблица 1. Результаты тестирования сетевых интерфейсов

	iperf, Мбит/с	LINPACK, % от eth	interFoam, с	rhoPimple Foam, с	GROMACS, с
eth	94.1	100	62	91	242.3
veth	94	99.1	70	91	242.4
macvlan	94	99.7	62	91	242.9

Для случая тестов LINPACK, Iperf и GROMACS разницы в производительности для указанных сетевых интерфейсов по сути нет. Разница между случаем veth и macvlan практически незаметна на большинстве тестов, за исключением теста для солвера «interFoam». Однако причина этого различия может быть объяснена особенностями работы сетевого стека: в случае veth стеку TCP/IP реже удается объединять данных в рамках больших пакетов, получается больше пакетов небольшой длины, как следствие — связанные с их обработкой накладные расходы. Причина того, что данные объединяются реже — проверка TCP «Small Queue Check».

Для случая использования vxlan присутствуют некоторые накладные расходы, однако они не являются существенными (к примеру, на тесте Iperf для macvlan и vxlan было получено 90.9 Мбит/с вместо 94 Мбит/с для случая macvlan адаптера без vxlan). При этом использование vxlan дает возможность создания общей сети «поверх» глобальной сети.

По результатам тестов предлагается следующая архитектура сети для виртуального кластера.

1. Использование устройств macvlan для контейнеров (позволяет изолировать контейнеры при низких накладных расходах).
2. Устройства macvlan работают в режиме «bridge» (пакеты, адресованные контейнерам на одном узле, передаются в рамках одного узла).
3. Каждый контейнер работает в отдельном сетевом пространстве имен и использует отдельное устройство macvlan.
4. Устройства macvlan создаются на базе физического сетевого адаптера сервера.
5. В случае необходимости объединения удаленных узлов можно использовать vxlan (создавать macvlan устройства на базе устройства vxlan). Для больших возможностей по объединению сетей лучше использовать вариант «vxlan+bridge+veth».

В **четвертой главе** рассматривается методика обеспечения отказоустойчивой и надежной работы вычислительного кластера, использующего контейнеры, за счет миграции вычислительных задач.

Миграцию виртуальных машин, использующих аппаратную виртуализацию, на сегодняшний день довольно часто можно назвать стандартной процедурой. Различные решения для виртуализации предлагают такой сервис своим клиентам. Миграция таких ВМ, как правило, не вызывает проблем, поскольку гипервизор предоставляет ВМ абстракцию оборудования, которая должна быть обеспечена гипервизорами на всех узлах. Самое главное — внутреннее состояние ОС «мигрирует» вместе с ВМ, остается неизменным. Миграция контейнеров и задач — совершенно другое дело из-за тесной интеграции с ОС: на удаленном узле потребуется изменить внутреннее состояние ОС для возобновления контейнера или задачи. Миграция вычислительных задач ставит множество вопросов, до конца так и нерешенных. Поэтому системы управления задачами, как правило, или не поддерживают миграцию задач, или задают значительные ограничения. Даже checkpoint/restart задачи на одном и том же узле требует решения множества вопросов.

Некоторые приложения сами по себе предоставляют возможности checkpoint/restart благодаря сохранению промежуточных данных и перезапуску расчетов с момента checkpoint. Однако такую возможность предоставляют далеко не все приложения, поэтому задачу checkpoint/restart нужно рассматривать в общем случае. А в общем случае для checkpoint/restart даже на одном узле требуется использование специальных средств.

Выполнение C/R (Checkpoint/Restart) в рамках широко распространенных систем управления задачами накладывает множество ограничений. В общем случае может понадобиться перекомпиляция приложения. Иногда поддерживается C/R лишь последовательных задач на одном узле. Довольно часто требуются отдельные плагины для выполнения C/R. А иногда пользователю предлагается самому реализовать необходимую функциональность, возможно, используя уже упомянутые средства.

В работе рассмотрено несколько вариантов выполнения C/R. В итоге был выбран вариант C/R с помощью CRIU (Checkpoint/Restore In Userspace). CRIU дает беспрецедентные возможности, это успешный, активно развивающийся проект. К числу несомненных достоинств CRIU можно отнести тот факт, что он обеспечивает сохранение и восстановление сетевых соединений, сохранение и восстановление состояния дерева процессов с учетом большого числа параметров (при этом перекомпиляция для приложения не требуется). Кроме того, CRIU предлагает даже восстановление контейнеров.

Миграция же задач и вовсе, как правило, не поддерживается или поддерживается с ограничениями. Трудности, связанные с миграцией, могут быть объяснены, например, привязкой к конкретному IP-адресу узла, на котором задача была запущена изначально. Однако миграция задач могла бы дать множество преимуществ: задачи можно было бы мигрировать для обеспечения балансировки нагрузки, для приоритетного запуска задач, для вывода проблемных узлов из кластера. Без миграции все эти задачи, скорее всего, будут решены обычным завершением задачи с потерей результата. А ведь задача может считаться несколько месяцев.

Для решения проблемы миграции задач в рамках данной работы предлагается использование контейнеров. В общем случае задача запускается на нескольких узлах. При этом на каждом из узлов может быть запущено множество процессов задачи, считающихся на нескольких процессорных ядрах.

При запуске без контейнера процессы используют сетевой интерфейс (или интерфейсы), доступный на узле, сетевое взаимодействие осуществляется с использованием этого интерфейса, используется один из заранее сконфигурированных IP-адресов. В таком случае миграция задач, использующих сеть, затруднена, так как процесс задачи использует IP-адрес узла. При миграции этот IP-адрес потребуется задать на другом узле, однако это невозможно, так как это приведет к конфликту IP-адресов в сети (узел, на котором считался процесс задачи и тот узел, на который он процесс был мигрирован должны получить одинаковые IP-адреса). Использование нескольких IP-адресов для одного и того же сетевого интерфейса на узле для разных задач тоже проблематично: в этом случае потребуется ограничивать использование того или иного IP-адреса той или иной задачей, ведь при миграции IP-адрес должен быть удален с одного узла и добавлен на другом, однако, на первом узле этот IP-адрес может использоваться другой задачей.

Решением указанной проблемы может быть использование контейнеров. В этом случае для каждой группы процессов каждой задачи на узле создается отдельный контейнер, в котором эти процессы и работают. При этом контейнер работает в отдельном сетевом пространстве имен с отдельным виртуальным сетевым интерфейсом. При этом для каждого такого интерфейса выделяется уникальный IP-адрес, а также MAC-адрес.

При этом при перемещении задачи между узлами MAC-адрес интерфейса контейнера может оказаться «за другим портом» физического коммутатора. Для скорейшего обновления записи в таблице коммутатора можно отправить тестовый кадр. На рисунке 2 представлена архитектура системы для работы методики.

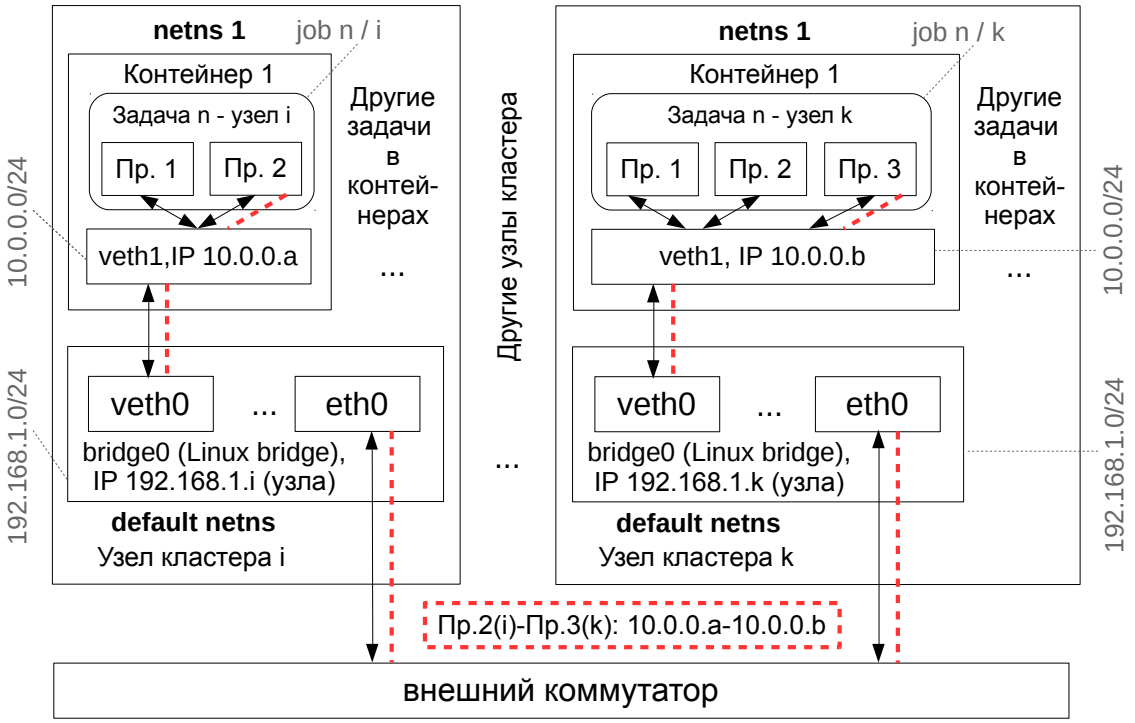

Рисунок 2. Архитектура системы для миграции задач

Таким образом, методику можно сформулировать следующим образом:

1. При запуске задачи выделить набор свободных IP-адресов (по количеству узлов для задачи), получить уникальные MAC-адреса.
2. Развернуть виртуальный кластер: на каждом узле для задачи создать контейнеры, необходимые виртуальные сетевые интерфейсы, настроить их, запустить необходимые сервисы (например, sshd).
3. Создать файл с узлами для задачи: вместо указания имен узлов, задать IP-адреса виртуального кластера.
4. Запустить скрипт задачи в рамках контейнера, запуск процессов на узлах должен быть выполнен с учетов файла со списком узлов. Таким образом, процессы будут запущены в рамках контейнеров виртуального кластера.
5. При необходимости приостановить задачи (checkpoint).
6. В нужный момент — возобновить задачу (restart), предварительно восстановив настройки контейнера
7. Checkpoint/Restart выполняются с помощью CRIU (успешный C/R зависит от задачи, настроек ОС и параметров C/R).

Описанная методика приведена для случая использования реализации PBS для управления запуском задач. При этом в саму архитектуру системы PBS

необходимо добавить сервис выдачи адресов (как IP, так и MAC), а также измененную PBS MOM. Такая PBS MOM выполняла бы необходимую работу по созданию и настройке контейнеров.

Выполнение C/R предполагается производить вручную. Однако в дальнейшем возможна интеграция в существующую систему PBS — такие системы могут предлагать пользователю самому указать средство (программу, скрипт) как для checkpoint, так и для restart. При этом, к примеру, в случае выполнения checkpoint и остановки задачи через систему PBS будет вызван указанный скрипт: скрипт выполнит chekpoint и остановку процессов задачи, а система PBS изменит статус задачи.

Для успешной миграции задачи накладывается множество ограничений. На узлах должно быть обеспечено одинаковое окружение для успешного выполнения миграции. Для успешной миграции может быть предъявлено большое количество требований, однако, конкретный набор определяется системой, обеспечивающей C/R и миграцию, и приложением, которое будет мигрировано. Однако рассмотрение конкретных требований выходит за рамки данной работы. В рамках данной работы предлагается методика, обеспечивающая миграции задач при возможности успешного C/R на разных узлах. Для успешного C/R требуется выбрать группы схожих узлов, что можно сделать при помощи методики, описанной в главе 1. Таким образом, вопрос о миграции в случае конкретного приложения на конкретных узлах сводится к характеристикам узлов и возможности выполнения C/R для данного приложения.

Использование рассматриваемой методики при миграции задачи обеспечивает следующее: надежность, так как при использовании методики отсутствует привязка задач к определенным узлам кластера, может быть выполнен checkpoint/restart для задач на схожих узлах; отказоустойчивость, так как создание checkpoints позволяет восстановить задачу после сбоя; балансировку нагрузки, так как теперь задачи могут быть мигрированы между узлами кластера в любой момент.

В **заключении** сформулированы основные результаты работы:

1. Методика управления распределенными разнородными виртуальными кластерами, использующими виртуализацию на уровне ОС;
2. Методика обеспечения надежной и отказоустойчивой работы и балансировки нагрузки вычислительных кластеров за счет динамической миграции задач в рамках виртуального кластера;
3. Архитектура виртуальной вычислительной сети для вычислительного кластера, минимизирующая накладные расходы, связанные с обменом данными, под конкретную задачу.

Публикации в изданиях, рекомендованных ВАК России

1. Чубахиро, А. Методика управления распределенными разнородными виртуальными кластерами, использующими виртуализацию на уровне ОС [Текст] / А. Чубахиро // «Наука и Бизнес: пути развития». - № 1, 2019(январь). – С. 65-69. – ISSN 2221-5182.
2. Чубахиро, А. Виртуальные вычислительные сети для случая аппаратной виртуализации [Текст] / А. Чубахиро, М. Каманде // «Современная наука: актуальные проблемы теории и практики. Серия «Естественные и технические науки»». — № 6, 2018(Июнь). – С. 158-163. – ISSN:2223-2966.
3. Каманде, М. Виртуальные вычислительные сети для контейнеров [Текст] / М. Каманде, А. Чубахиро, // «Современная наука: актуальные проблемы теории и практики. Серия «Естественные и технические науки»». — № 6, 2018(Июнь). – С. 63-68. ISSN: 2223-2966.

Публикации в изданиях, индексируемых в Scopus

4. Bogdanov A. Profiling scheduler for efficient resource utilisation [Текст] / A. Bogdanov, V. Gaiduchok, N. Ahmed, A. Cubahiro, I. Gankevich // Proceedings of the 15th International Conference on Computational Science and Its Applications(ICCSA2015), Banff, Canada, 22-25 June 2015. — Springer Verlag. — 2015. — p.299-310, volume 9158. — ISBN: 978-331921409-2.

5. Bogdanov A. User interface for a computational cluster: resource description approach [Текст] / A. Bogdanov, V. Gaiduchok, N. Ahmed, P. Ivanov, M. Kamande, A. Cubahiro // Selected Papers of the 7th International Conference «Distributed Computing and Grid-technologies in Science and Education», Dubna, Russia, July 4-9, 2016. — CEUR Workshop Proceedings. — 2017 — P. 145-149, Volume 1787.
6. Korkhov, V. Distributed virtual cluster management system [Текст] / V. Korkhov, S. Kobyshev, A. Degtyarev, A. Cubahiro, L. Gaspary, X. Wang, Z. Wu // Selected Papers of the 8th International Conference «Distributed Computing and Grid-technologies in Science and Education», Dubna, Russia, September 10-14, 2018. — CEUR Workshop Proceedings. — 2018 — P. 383-387, Volume 2267. – ISSN: 1613-0073.

Другие статьи и материалы конференций

7. Korkhov V. Experience in Building Virtual Private Supercomputer [Текст] / V. Korkhov, I. Gankevich, A. Degtyarev, A. Bogdanov, V. Gaiduchok, N. Ahmed, A. Cubahiro // Proceedings of International Conference on Computer Science and Information Technologies (CSIT), 2015. — p. 220-223. — ISBN 978-5-8080-0797-0.