УДК 004.457+004.031.2+004.382.2
DOI: 10.15827/2311-6749.17.4.4

ДИНАМИЧЕСКИЙ КОНФИГУРАТОР ВИРТУАЛЬНОЙ РАСПРЕДЕЛЕННОЙ ВЫЧИСЛИТЕЛЬНОЙ СРЕДЫ

Б.М. Шабанов, к.т.н., доцент, директор, shabanov@jscc.ru; П.Н. Телегин, к.т.н., ведущий научный сотрудник, ptelegin@jscc.ru; А.В. Баранов, к.т.н., доцент, ведущий научный сотрудник, antbar@mail.ru
(Межведомственный суперкомпьютерный центр Российской академии наук – филиал ФГУ «Федеральный научный центр Научно-исследовательский институт системных исследований Российской академии наук», Ленинский просп., 32а, г. Москва, 119334, Россия)
Д.В. Семёнов, научный сотрудник, sdvbox@gmail.com;
А.В. Чуваев, научный сотрудник, achuv@yandex.ru
(ФГУП «Научно-исследовательский институт «Квант», 4-й Лихачевский пер., 15, г. Москва, 125438, Россия)

В настоящее время при объединении высокопроизводительных вычислительных установок в единую распределенную вычислительную среду широко применяются технологии виртуализации, позволяющие решить проблему бинарной переносимости пользовательских заданий между разными вычислительными установками. Для эффективного управления ресурсами распределенной вычислительной среды авторы статьи предлагают подход, основанный на разбиении пользовательских заданий на классы в зависимости от ресурсных требований. Каждому классу заданий ставится в соответствие один или несколько классов виртуальных вычислителей, экземпляры которых динамически разворачиваются на ресурсах распределенной вычислительной среды. За разворачивание и сворачивание экземпляров виртуальных вычислителей отвечает конфигуратор виртуальной вычислительной среды, который, взаимодействуя с метапланировщиком этой среды, обеспечивает равномерную загрузку ее вычислительных ресурсов и позволяет свести к минимуму число реконфигураций.

Ключевые слова: *распределенная вычислительная среда, виртуальный вычислитель, метапланировщик грид, виртуальная вычислительная среда, планирование параллельных заданий.*

С целью повышения производительности и надежности расчетов отдельные высокопроизводительные *вычислительные установки* (ВУ) нередко объединяются в *распределенные вычислительные среды* (РВС), или грид-системы (рис. 1). Наибольшее распространение получила многоуровневая форма организации РВС, когда в качестве ВУ используются высокопроизводительные вычислительные кластеры [1].

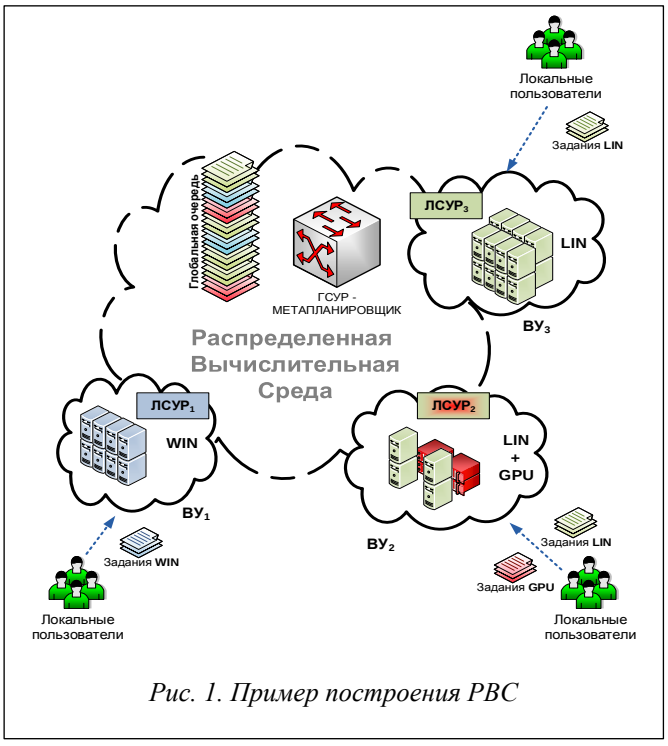

Рис. 1. Пример построения РВС

Для управления отдельной ВУ применяется *локальная система управления ресурсами* (ЛСУР), действия всех ВУ координирует *глобальная система управления ресурсами* (ГСУР). Несмотря на это, в РВС, как правило, отсутствует единый центр управления, а ГСУР строится по децентрализованной схеме. После включения ВУ в состав РВС вычислительные ресурсы этой ВУ не отчуждаются от их владельца и продолжают использоваться для выполнения локальных заданий, которые образуют локальный поток заданий [2]. Совместно с заданиями локального потока на вычислительные ресурсы ВУ поступают задания с других ВУ – из глобального потока. В алгоритмах планирования (распределения) пользовательских заданий глобального потока на вычислительные ресурсы РВС, используемых в известных на сегодняшний день метапланировщиках грид

[1, 3], как правило, предполагается, что пользовательское задание может быть выполнено на любой ВУ, входящей в состав РВС.

При подобной организации РВС возникает проблема бинарной несовместимости исполняемых модулей пользовательских прикладных программ. Поскольку включаемые в РВС вычислительные ресурсы не отчуждаются от своих владельцев, в отношении своих ресурсов владельцы продолжают применять собственные политики администрирования, безопасности и планирования заданий. Это приводит к невозможности единообразия стека системного и инструментального ПО и, как следствие, к невозможности перенесения пользовательских заданий с одной ВУ на другую на уровне исполняемых программных модулей. Попытки решить проблему путем сборки исполняемых модулей для каждой ВУ повлекли существенное усложнение организации вычислений, а использование кроссплатформенных и/или интерпретируемых языков программирования типа Java привели к неприемлемым накладным расходам.

Решение проблемы бинарной несовместимости пришло с развитием технологий виртуализации, позволившим создавать пользовательские программные среды (платформы) и получать виртуальный вычислитель с любыми *операционной системой* (ОС) и необходимым набором библиотек. В работах [4, 5] проведена оценка накладных расходов на гипервизорную и контейнерную виртуализацию, которая позволяет сделать вывод о применимости этих технологий в высокопроизводительных вычислениях, в том числе и при построении распределенных систем. В этом случае пользовательское задание оформляется в виде образа виртуальной машины или контейнера, содержащего необходимую для выполнения задания программную среду. При распределении этого задания на одну из ВУ РВС происходят развертывание программной среды из образа (шаблона) и выполнение пользовательского задания в развернутой среде.

При такой организации вычислений возникают две трудности: во-первых, развертывание программной среды из образа может занять достаточно длительное время, во-вторых, неизбежна неравномерная загрузка разнородных ВУ, образующих РВС, что снижает эффективность использования вычислительных ресурсов. Для решения необходимо перераспределить задания между очередями ЛСУР [6]. В настоящей работе рассматриваются формирование распределенной виртуальной вычислительной среды на базе ресурсов РВС и планирование пользовательских заданий на *виртуальные вычислители* (ВВ) с целью повышения эффективности использования вычислительных ресурсов и уменьшения среднего времени нахождения пользовательского задания в очереди за счет балансировки нагрузки.

Общая схема управления виртуальными ресурсами РВС

В общем случае РВС состоит из гетерогенных ВУ, к каждой из которых существует очередь пользовательских заданий с определенными ресурсными запросами (требованиями), позволяющими выполнить задание на одном из гетерогенных вычислительных ресурсов (рис. 1). Можно все множество заданий РВС в зависимости от ресурсных требований разбить на определенное количество классов и каждому классу поставить в соответствие определенный ВВ. Для описания ресурсных требований заданий и классов можно ввести соответствующий алфавит: $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ где α_i – ресурсное требование i-го типа.

Данный вычислитель представляет собой одну или несколько объединенных виртуальной коммуникационной средой виртуальных машин (контейнеров), используемых для выполнения пользовательских заданий с определенными ресурсными требованиями. Конфигурация ВВ может характеризоваться количеством ядер, объемом оперативной памяти, размером файлового хранилища, типом ОС, набором библиотек и иными параметрами, необходимыми для выполнения пользовательских заданий. Чтобы ВВ был способен выполнять задания своего класса, он должен быть развернут на физических ресурсах ВУ. Развертывание ВВ осуществляется с помощью платформы виртуализации из специально подготовленного шаблона. Из одного шаблона может быть развернуто несколько экземпляров ВВ с заданными характеристиками (определенной конфигурации). При этом на одном физическом ресурсе ВУ может быть размещено несколько ВВ. Таким образом, РВС может быть представлена в виде множества ВВ определенных конфигураций.

Так, если некоторый ВВ характеризуется вектором $\beta(\alpha) = \{\beta_1(\alpha_1), \beta_2(\alpha_2), \ldots, \beta_n(\alpha_n)\}$, где β_i – имеющийся объем ресурсов i-го типа, то задание с вектором ресурсных требований $a(\alpha) = \{ a_1(\alpha_1), a_2(\alpha_2), \ldots, a_n(\alpha_n)\}$ может быть выполнено на данном ВВ, если $a_i \le \beta_i$, $\forall i \in \overline{1,n}$.

Развернутый на физических ресурсах ВУ ВВ представляется в ЛСУР в виде *логического вычислительного устройства* (ЛВУ), на которое возможно планирование заданий (рис. 2). Фактически осуществляется переход на новый абстрактный уровень доступных для ЛСУР вычислительных ресурсов, при котором мультипрограммная вычислительная среда суперкомпьютера подстраивается под ресурсные требования пользовательских заданий. Однако для реализации такой возможности ЛСУР ВУ должна иметь возможность задавать произвольные типы ресурсов и распределять пользовательские задания по очередям к ресурсам соответствующего типа. Поддержка таких механизмов осуществляется во многих современных системах планирования заданий: PBS, SLURM, Moab, СУППЗ и других.

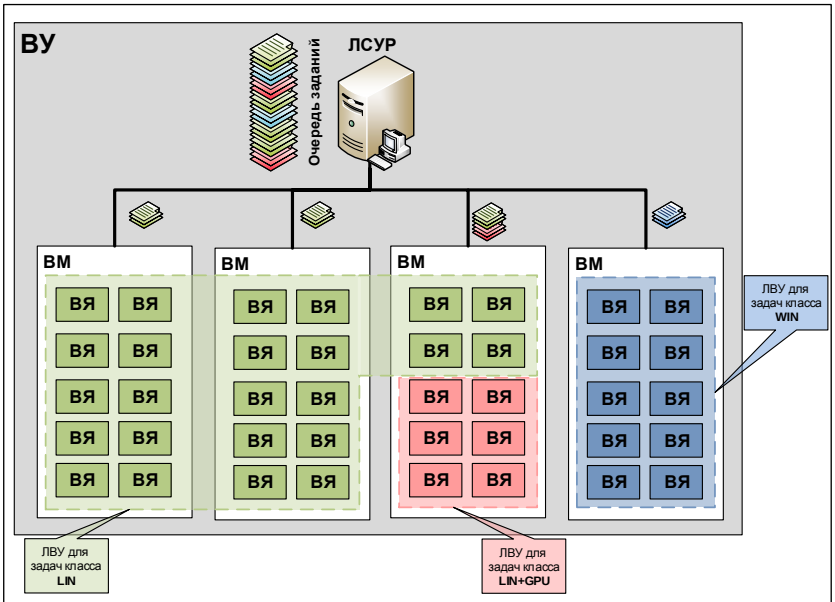

Рис. 2. Размещение ВВ по вычислительным ядрам (ВЯ) вычислительных модулей (ВМ) ВУ

Для обеспечения возможности настройки вычислительной среды ВУ под текущие ресурсные требования пользовательских заданий, находящихся в очереди ЛСУР, авторами разработан функционирующий на ВУ специальный программный компонент – конфигуратор виртуальной вычислительной среды. Конфигуратор принимает решение о динамическом изменении состава развернутых ВВ в соответствии с загруженностью вычислительных ресурсов, оценивает состояние очередей ЛСУР, состояние заданий, выполняющихся на ВВ, список доступных ресурсов и осуществляет автоматическое перераспределение доступных ВВ по физическим ресурсам ВУ. Для выполнения задания конфигуратор может развернуть экземпляр ВВ с характеристиками, соответствующими классу задания, при этом один или несколько неиспользуемых ВВ могут быть свернуты. Действия конфигуратора по разворачиванию и сворачиванию ВВ назовем реконфигурацией ВУ. Очевидно, что каждая реконфигурация занимает некоторое время и привносит в процесс вычислений накладные расходы. С целью уменьшения числа реконфигураций конфигуратор взаимодействует с метапланировщиком РВС для получения в очередь ЛСУР пользовательских заданий определенного класса (наиболее подходящих заданий).

Подход с использованием конфигуратора виртуальной вычислительной среды позволяет интегрировать ВУ различной архитектуры в единую РВС, в которой задача бинарной переносимости заданий решается за счет формирования на физических ресурсах ВВ с необходимыми характеристиками. При этом эффективное использование вычислительных ресурсов обеспечивается как на уровне метапланировщика РВС за счет балансировки нагрузки на уже развернутые ВВ, так и на уровне ВУ за счет динамического реконфигурирования виртуальной среды в зависимости от классов пользовательских заданий.

Далее детально рассмотрим механизмы и алгоритмы, используемые для управления виртуальной РВС.

Формирование классов пользовательских заданий и набора ВВ

При формировании набора ВВ необходимо наиболее полно охватить все возможные варианты конфигураций пользовательских заданий [7]. С этой целью проводится статистический анализ потока заданий, поступивших в РВС за определенный период.

На возможность задания быть выполненным на том или ином вычислительном ресурсе $a(\alpha) = \{a_1(\alpha_1), a_2(\alpha_2), \ldots, a_n(\alpha_n)\}$ влияет множество характеристик, которые разделяются на фиксированные и расходуемые. К фиксированным характеристикам можно отнести тип ОС, архитектуру центрального процессора, наличие и тип ускорителя и др. Классификация заданий в ходе статистического анализа проводится в основном по этим характеристикам. Расходуемые характеристики, такие как количество вычислительных ядер, объем оперативной памяти и т.п., используются для разбиения классов на подклассы с целью повышения эффективности заполнения физических ресурсов РВС ВВ за счет многообразия вариантов.

Формирование набора ВВ осуществляется по следующим правилам:

– для любого задания из глобальной и локальных очередей РВС должна существовать как минимум одна конфигурация ВВ, позволяющая выполнить это задание;

– для любой ВУ из состава РВС должна существовать как минимум одна конфигурация ВВ, которую можно было бы развернуть на ресурсах этой ВУ.

Формирование набора ВВ на основе статистических данных о поступивших за определенный период пользовательских заданиях можно осуществлять как вручную, так и с применением методов интеллектуального анализа данных (например кластерного). Проиллюстрируем процесс формирования набора ВВ следующим примером.

Предположим, что РВС состоит из трех ВУ, каждая из которых обладает набором ресурсов с определенными характеристиками. В таблице 1 приведены выделенные значимые характеристики ВУ, а также информация о потоке пользовательских заданий, поступивших на каждую ВУ за определенный период.

Таблица 1

Характеристики ВУ РВС из примера формирования набора ВВ

№	Тип ОС	Наличие ускорителя (GPU)	Число ядер	Количество пользовательских заданий, поступивших на ВУ (число заданий с требованием ускорителя на GPU)
1	Windows	–	200	4 000
2	Linux	+	400	7 000 (3 000)
3	Linux	–	800	9 000

Для разбиения потока пользовательских заданий на классы в качестве фиксированных были выбраны такие характеристики, как «Тип операционной системы» и «Наличие ускорителя». Число требуемых для задания процессорных ядер учитывалось как расходуемая характеристика.

К тестовому потоку пользовательских заданий был применен кластерный анализ [8] с использованием алгоритма «к-средних» (k-means). После 30 итераций алгоритма сформировалось 11 устойчивых центров, каждый из которых соответствует своему классу пользовательских заданий. Результаты кластерного анализа потока заданий приведены в таблице 2.

Таблица 2

Результат деления пользовательских заданий на классы с помощью кластерного анализа

№ класса	Тип ОС	Наличие ускорителя	Число требуемых ядер	Заданий этого класса (%)
1	Linux	+	160	7
2	Linux	+	320	6
3	Linux	+	400	5
4	Linux	–	160	15
5	Linux	–	320	14
6	Linux	–	400	13
7	Linux	–	560	7
8	Linux	–	720	7
9	Linux	–	800	6
10	Windows	–	80	9
11	Windows	–	200	11

Приведенные в таблице значения расходуемых характеристик являются ограничением сверху, то есть пользовательское задание определенного класса должно выполниться на ВВ, ресурсные характеристики которого соответствуют верхней границе класса. Как следует из таблицы 2, получившееся разбиение соответствует правилам формирования ВВ, ведь для любого пользовательского задания из анализируемого потока существует хотя бы один соответствующий ему класс, и на каждой ВУ (табл. 1) можно разместить один или более ВВ, соответствующих классам из таблицы 2.

С целью более эффективного размещения ВВ на физических ресурсах РВС предлагается число ядер ВВ сделать кратным количеству процессорных ядер на физическом *вычислительном модуле* (ВМ). Если в разных ВУ количество процессорных ядер на ВМ различается, необходимо выбрать общий знаменатель.

Далее на каждой ВУ из состава РВС создается набор шаблонов ВВ, ресурсные характеристики которых позволяют им быть размещенными на физических ресурсах той или иной ВУ. На ВУ № 1 из таблицы 1 можно разместить ВВ, соответствующие классам №№ 4, 10, 11, на ВУ № 2 – ВВ для всех классов из таблицы 2, кроме №№ 7–9, для ВУ № 3 подойдут все классы, кроме первых трех.

Первоначальное заполнение физических ресурсов РВС множеством ВВ

После определения классов пользовательских заданий и формирования соответствующих шаблонов ВВ определяется первоначальная конфигурация виртуальной вычислительной среды.

Для создания первоначального набора ВВ, размещаемых на физических ресурсах РВС, необходимо определить соотношение мощностей классов заданий $|C_1|:|C_2|:\ldots:|C_n|$. В соответствии с характеристиками ВУ из состава РВС и полученного соотношения мощностей классов заданий $|C_1|:|C_2|:\ldots:|C_n|$ формируется необходимое количество ВВ, способное обеспечить максимальную утилизацию вычислительных ресурсов.

Задаче закрепления набора ВВ за ВС РВС посвящено множество работ [9–11]. В общем случае данную задачу принято сводить к решению задачи динамического программирования об оптимальном заполнении ранца (задаче о рюкзаке). При ее решении находится оптимальное распределение определенного количества ВВ по физическим ВУ. Под целевой функцией при решении оптимизационной задачи могут пониматься минимизация стоимости обслуживания в ходе эксплуатации, минимизация энергопотребления, минимизация простоя вычислительной техники либо другие критерии оценки эффективности использования вычислительных ресурсов.

В общем виде задачу размещения ВВ на ВУ из состава РВС можно сформулировать следующим образом. Пусть в РВС имеется множество ВУ $N = \{N_1, N_2, \ldots, N_n\}$, где n – число ВУ. Вычислительные ресурсы i-й ВУ, характеризующейся числом ядер и объемом оперативной памяти $\{N_i^{CPU}, N_i^{RAM}\}$, подлежат распределению между множеством ВВ: $V = \{V_1, V_2, \ldots, V_m\}$, где m – число ВВ. Каждый j-й ВВ запрашивает V_j^{CPU} ядер и V_j^{RAM} оперативной памяти.

Необходимо найти

$$\min_{x_{ij}} \sum_{j}^{m} Cost\left(x_{ij}\right),$$

где $Cost(x)$ – некая целевая функция, например, стоимость эксплуатации ВУ или ее энергопотребление [12, 13]. При этом действуют ограничения:

$$\begin{cases} \sum_{i=1}^{n} V_i^{CPU}\left(x_{ij}\right) < N_j^{CPU}, j = [1..m], \\ \sum_{i=1}^{n} V_i^{RAM}\left(x_{ij}\right) < N_j^{RAM}, j = [1..m]. \end{cases}$$

Результатом решения оптимизационной задачи будет матрица $X = \{x_{ij}\}$, $i = [1..n]$, $j = [1..m]$, где x_{ij} равно 1, если i-й ВВ закреплен за j-й ВУ, и 0, если нет.

Вопросы управления виртуальными ресурсами достаточно хорошо исследованы [14, 15]: существуют алгоритмы размещения виртуальных машин на физических ресурсах ВУ, сформулированы оптимизационные задачи, возникающие при миграции виртуальных машин в соответствии с требованиями энергоэффективности ВУ и минимизации стоимости их обслуживания. Кроме того, исследованы методы обеспечения ресурсами виртуальных машин в зависимости от их фактической загруженности (фактического использования), рассмотрены различные схемы оптимального плана предоставления ресурсов в виде виртуальных машин с учетом минимизации стоимости владения ими. Отдельно рассматриваются перспективы предварительной классификации заданий в РВС с целью оптимизации их дальнейшего размещения по вычислительным ресурсам РВС. Однако в литературе почти не рассматриваются вопросы (методы и алгоритмы) динамического изменения состава ВВ на физических ресурсах ВУ (РВС) в соответствии с изменениями интенсивности потока пользовательских заданий. Стоит отметить, что в используемых на практике метапланировщиках РВС, как правило, используются алгоритмы прямого распределения заданий, что приводит к конкуренции за ресурсы между локальными заданиями ВУ и заданиями из глобальной очереди РВС.

Реконфигурация ВВ в процессе функционирования РВС

С целью обеспечения равномерной загрузки вычислительных ресурсов РВС в процессе ее функционирования необходимо осуществлять перераспределение ресурсов между различными классами ВВ, то есть производить динамическую реконфигурацию ВВ. Эту задачу решает упоминавшийся ранее конфигуратор виртуальной вычислительной среды, который оценивает загруженность развернутых экземпляров ВВ каждого класса и перераспределяет физические вычислительные ресурсы, разворачивая дополнительные экземпляры вычислителей наиболее востребованных классов.

Для принятия управленческого решения по реконфигурации ВВ конфигуратор использует следующие характеристики заданий и развернутых ВВ:

– площадь $S_{\text{задания}}$ пользовательского задания, под которой понимается произведение запрашиваемого заданием числа ядер $N_{\text{задания}}$ на запрашиваемое время $T_{\text{задания}}$ выполнения задания: $S_{\text{задания}} = N_{\text{задания}} \times T_{\text{задания}}$;

– суммарная площадь заданий $S_{\text{ВВ}}$, находящихся в очереди ЛСУР для ВВ определенного класса и выполняющихся на развернутых экземплярах ВВ этого класса;

– загруженность $C_{ВВ}$ ВВ некоторого класса, под которой понимается отношение суммарной площади заданий для этого класса ВВ к общему числу N_{CPU} процессорных ядер, выделяемых для развертывания экземпляров ВВ; характеристика показывает, сколько времени в среднем будут заняты вычислениями развернутые экземпляры ВВ данного класса;

– граница $LIM_{C_{ВВ}}$ загруженности ВВ ограничивает сверху загруженность развернутого экземпляра ВВ.

Разница между границей и значением текущей загруженности ВВ характеризует суммарную площадь заданий, которые можно поставить в очередь конкретной ЛСУР для данного класса ВВ.

Для минимизации потерь при реконфигурации виртуальной вычислительной среды предлагается ввести коэффициент соответствия K_{map} ресурсных характеристик различных ВВ, определяемый разницей в ядрах, памяти и других расходуемых характеристиках между двумя различными ВВ. Например, коэффициент соответствия K_{map} может быть определен на основе трех основных характеристик ВУ: числа процессорных ядер a_{CPU}, объеме оперативной памяти a_{RAM} и размере дискового пространства a_{HDD}, задействованных для развертывания экземпляра ВВ. Для каждой характеристики вводятся весовые коэффициенты: K_{CPU}, K_{RAM}, K_{HDD}. Тогда для ВВ с характеристиками, заданными вектором $a = \{a_{CPU}, a_{RAM}, a_{HDD}\}$, коэффициент соответствия ВВ с ресурсными характеристиками, заданными вектором $b = \{b_{CPU}, b_{RAM}, b_{HDD}\}$, будет определяться как

$K_{map} = K_{CPU}(a_{CPU} - b_{CPU}) + K_{RAM}(a_{RAM} - b_{RAM}) + K_{HDD}(a_{HDD} - b_{HDD})$.

Минимизация потерь при динамической реконфигурации виртуальной вычислительной среды достигается за счет выбора на замену сворачиваемому ВВ наиболее «соответствующего» ВВ из числа доступных. Рассмотрим предлагаемый авторами алгоритм динамического изменения состава ВВ в соответствии с состоянием очереди заданий и загруженности ВВ.

1. Для каждого класса ВВ конфигуратор определяет показатель загруженности $C_{ВВ}$.

2. Из числа ВВ, для которых текущая загруженность больше установленной границы, то есть $C_{ВВ} > LIM_{C_{ВВ}}$, выбирается ВВ с максимальным значением показателя загруженности $C_{ВВ}$, и для него определяется возможность запуска дополнительного экземпляра. Запуск дополнительного экземпляра возможен при наличии достаточного количества свободных физических ресурсов ВУ. Если свободных ресурсов нет, конфигуратор переходит к высвобождению ресурсов, то есть к сворачиванию экземпляров ВВ некоторого класса. Сворачивание возможно, если для ВВ выполняется одно из условий:

– на ВУ присутствуют развернутые экземпляры ВВ, для которых нет заданий в очереди и на выполнении;

– количество развернутых экземпляров ВВ определенного класса равно n, и выполняется условие, что при $n - 1$ развернутых экземплярах ВВ уровень их загруженности не превысит установленную границу $LIM_{C_{ВВ}}$.

3. Конфигуратор при принятии решения о замене ВВ использует коэффициент соответствия K_{map}, выбирая для сворачивания наиболее подходящие ВВ. Из всех подходящих ВВ для сворачивания выбирается наименее загруженный в соответствии с показателем загруженности $C_{ВВ}$. Для развернутых ВВ рассчитываются обобщающие показатели:

$K_{all}(K_{map}, C_{ВВ}) = \mu_{map} K_{map} + \mu_{ВВ} C_{ВВ}$,

где μ_{map}, $\mu_{ВВ}$ – весовые коэффициенты. В соответствии со значениями обобщающих показателей определяется наименее загруженный ВВ, максимально подходящий по ресурсным характеристикам, один развернутый экземпляр которого сворачивается.

4. На месте высвобожденных ресурсов конфигуратор разворачивает экземпляр ВВ, определенного на шаге 2.

Если в системе нет ВВ, для которых превышена граница загруженности, конфигуратор пытается перераспределить неиспользуемые вычислительные ресурсы между задействованными классами ВВ. Для этого осуществляется поиск ВВ, для которых выполняется условие $S_{ВВ} = 0$ (простой ВВ), и высвобождаются выделенные под него ресурсы. На месте высвобожденных ресурсов разворачивается экземпляр для ВВ с максимальной загруженностью.

Выделение ресурсов для нового класса ВВ

Если в одну из очередей РВС поступает задание, для которого нет развернутых экземпляров требуемого класса ВВ, реконфигурация виртуальной вычислительной среды производится следующим образом. Загруженность нового класса ВВ считается максимальной, и требуется обязательное высвобождение под него вычислительных ресурсов, $C_{ВВ} = \infty$. Для этого повторяются шаги 1–5 алгоритма динамической реконфигурации. Если алгоритм не приводит к успеху, для запуска первого экземпляра нового класса ВВ

выбирается наименее загруженный класс ВВ с количеством развернутых экземпляров больше 1 и $K_{map} > 0$ и принудительно высвобождаются занятые им ресурсы.

Если в РВС развернуто по одному экземпляру ВВ каждого класса, среди них выбирается наименее загруженный класс ВВ с $K_{map} > 0$ и для него выставляется запрет на получение заданий из глобальной очереди. После выполнения всех заданий для данного класса ВВ его ресурсы высвобождаются в пользу нового класса ВВ.

Планирование и распределение заданий глобальной очереди РВС

Для обеспечения эффективного использования вычислительных ресурсов в РВС необходимо обеспечить перераспределение разнородных заданий по ЛСУР. В современных метапланировщиках РВС для этого используются различные эвристические алгоритмы, базирующиеся на принципах минимальной загруженности и минимальной достаточности [6, 16]. Проводятся исследования эффективности использования алгоритмов, основанных на пакетировании [17], учета характеристик коммуникационной среды и метаданных заданий.

Как уже упоминалось, на практике ресурсы локальных ВУ являются неотчуждаемыми, и их состав динамически изменяется в зависимости от потребностей их владельцев. Поэтому при планировании заданий в РВС необходимо учитывать интересы владельцев вычислительных ресурсов. Для этого применяются подходы, основанные на экономических принципах распределения ресурсов [6, 18]. Примеры распределения ресурсов на основе организации аукционов представлены в [19]. С помощью аукционов при планировании заданий в РВС достигается эффективное использование доступных ресурсов с учетом интересов основных участников вычислений. Аукционы между различными ВУ обеспечивают обратное распределение заданий, когда каждая локальная ВУ сама запрашивает для себя необходимое количество заданий. Кроме того, такой подход к распределению заданий уменьшает конкуренцию за вычислительные ресурсы между локальными заданиями и заданиями, поступившими из глобальной очереди РВС.

В качестве альтернативы проведению аукционов в настоящей статье предлагается алгоритм распределения заданий на основе выставленных каждой ВУ приоритетов w_i для задания конкретного класса.

Конфигуратор виртуальной вычислительной среды взаимодействует с метапланировщиком РВС в режиме «запрос–ответ», каждый раз уточняя наличие заданий требуемого класса. Конфигуратор оценивает состояние очередей ЛСУР, загруженность ВВ и определяет приоритетные вычислители, для которых необходимо запросить дополнительные задания.

Сформированная виртуальная среда ВС характеризуется вектором $X = (X_1, X_2, \ldots, X_n)$, где X_i – ВВ с заданными вектором $\beta = (\beta_1, \beta_2, \ldots, \beta_n)$ ресурсными характеристиками. Конфигуратор определяет разницу между границей и значением текущей загруженности для всех типов ВВ на ВУ. Полученные значения используются для формирования запроса к метапланировщику РВС на получение заданий из глобальной очереди. Запрос представляет собой вектор $Y = Y_1^{w_1}, Y_2^{w_2} .. Y_n^{w_n}$ где $Y_i^{w_i}$ – идентификатор класса задания. Для каждого класса конфигуратор рассчитывает вес w_i, $w_i = LIM_{C_{\text{ВВ}}} - C_{\text{ВВ}}$. Сравнение весов показывает, для какого класса ВВ в системе не хватает заданий. Соответственно, наиболее приоритетными считаются задания для менее загруженных ВВ. На время реконфигурации виртуальной вычислительной среды конфигуратор может выставить запрет на задания для некоторого класса ВВ, в этом случае вес $w_i = -1$.

Расходуемые характеристики позволяют разбивать классы пользовательских заданий на подклассы с целью повышения эффективности заполнения физических ресурсов РВС ВВ за счет многообразия вариантов. Поэтому метапланировщик при планировании пользовательского задания на вычислительные ресурсы может выбирать из множества ВВ, развернутых на физических ресурсах РВС и являющихся подклассами одного класса. Для сопоставления множества классов задач $C = (C_1, C_2, \ldots, C_m)$ классам ВВ $X = (X_1, X_2, \ldots, X_n)$ используется матрица соответствия A размерности $m \times n$, элемент которой $a_{ij} = 1$ означает, что для исполнения задания класса C_i может быть использован вычислитель класса X_j. Матрица соответствия может быть сформирована вручную на этапе формирования классов задач. Для распределения задания метапланировщику РВС остается с помощью матрицы соответствия определить набор виртуальных вычислителей, способных выполнить задание, и выбрать ВУ, которая наиболее заинтересована в задании данного класса (для которой значение веса-приоритета w_i максимально).

При планировании пользовательского задания, для которого не существует развернутого экземпляра ВВ ($w_i = 0$ для $\forall Y \in \left\{ Y_1^{w_1}, Y_2^{w_2} .. Y_n^{w_n} \right\}$), метапланировщик РВС действует следующим образом. Задание назначается на ВУ, в которой развернут наименее загруженный ВВ с максимально подходящими значениями расходуемых характеристик, выбор осуществляется на основе максимального значения свертки показателей $\{K_{map}, w_i\}$.

Заключение

При объединении высокопроизводительных ВУ в единую РВС проблема бинарной переносимости пользовательских заданий может быть решена за счет применения технологий гипервизорной или контейнерной виртуализации. В этом случае на базе ресурсов ВУ организуется динамически реконфигурируемая виртуальная вычислительная среда, для которой актуально решение двух задач: минимизации накладных расходов за счет снижения числа реконфигураций и обеспечения балансировки вычислительной нагрузки.

Для решения этих задач авторами предложен комплексный подход, заключающийся в разбиении пользовательских заданий на классы в соответствии с их ресурсными запросами. Каждому классу пользовательских заданий ставится в соответствие один или несколько ВВ, экземпляры которых могут быть развернуты на физических ресурсах ВУ РВС. В разворачивании и/или сворачивании заключается реконфигурация виртуальной вычислительной среды, за которую отвечает разработанный авторами специальный программный компонент – конфигуратор виртуальной вычислительной среды.

Для сокращения числа реконфигураций и балансировки загрузки ВУ авторами предложен метод динамической реконфигурации виртуальной вычислительной среды, основанный на алгоритмах взаимодействия конфигуратора и метапланировщика РВС и перераспределения заданий в РВС. Предложенный метод, по мнению авторов, способен обеспечить равномерную загрузку вычислительных ресурсов, при этом сводит к минимуму число реконфигураций за счет извлечения из глобальной очереди «наиболее подходящих» пользовательских заданий.

Ближайшими перспективами работы являются построение имитационной модели конфигуратора и проведение вычислительных экспериментов на макете РВС с целью определения эффективности рассмотренных в статье подходов.

Работа выполнена при поддержке программы президиума РАН «1.5П Развитие гетерогенной распределенной вычислительной инфраструктуры для суперкомпьютерных приложений с использованием технологий грид и облачных вычислений».

Литература

1. Савин Г.И., Шабанов Б.М., Корнеев В.В., Телегин П.Н., Семенов Д.В., Киселев А.В., Кузнецов А.В., Вдовикин О.И., Аладышев О.С., Овсянников А.П. Создание распределенной инфраструктуры для суперкомпьютерных приложений // Программные продукты и системы. 2008. № 2. С. 2–7.
2. Корнеев В.В., Семенов Д.В., Телегин П.Н., Шабанов Б.М. Отказоустойчивое децентрализованное управление ресурсами грид // Изв. вузов: Электроника. 2015. Т. 1. № 111. С. 83–90.
3. GridWay Metascheduler: Metascheduling Technologies for the Grid. URL: http://gridway.org (дата обращения: 31.08.2017).
4. Аладышев О.С., Баранов А.В., Ионин Р.П., Киселёв Е.А., Орлов В.А. Сравнительный анализ вариантов развертывания программных платформ для высокопроизводительных вычислений // Вестн. УГАТУ. 2014. № 3. С. 295–300.
5. Баранов А.В., Николаев Д.С. Использование контейнерной виртуализации в организации высокопроизводительных вычислений // Программные системы: теория и приложения. 2016. Т. 7. № 1. С. 117–134.
6. Корнеев В.В., Киселев А.В., Голосов П.Е. Планирование заданий в грид на основе экономической модели // Матер. 2-й Междунар. конф. 2006. URL: http://grid2006.jinr.ru/docs/129_Golosov_Dubna.pdf (дата обращения: 31.08.2017).
7. Феоктистов А.Г. Методология концептуализации и классификации потоков заданий масштабируемых приложений в разнородной распределенной вычислительной среде // Системы управления, связи и безопасности. 2015. № 4. С. 1–25. URL: http://sccs.intelgr.com/archive/2015-04/01-Feoktistov.pdf (дата обращения: 31.08.2017).
8. Барсегян А.А., Куприянов М.С., Степаненко В.В., Холод И.И. Технологии анализа данных: Data Mining, Visual Mining, Text Mining, OLAP. СПб: БХВ-Петербург, 2007. 384 с.
9. Самоваров О.И., Гайсарян С.С. Архитектура и особенности реализации платформы UniHUB в модели облачных вычислений на базе открытого пакета OpenStack // Тр. ИСП РАН. 2014. Т. 26. Вып. 1. С. 403–420. DOI: 10.15514/ISPRAS-2014-26(1)-17. URL: http://www.ispras.ru/proceedings/docs/2014/26/1/isp_26_2014_1_403.pdf (дата обращения: 31.08.2017).
10. Ролик А.И., Боданюк М.Е. Алгоритмы последовательного анализа вариантов в задаче распределения виртуальных машин по физическим серверам в центрах обработки данных // Адаптивні системи автоматичного управління. 2012. № 21 (41). С. 61–69. URL: http://asac.kpi.ua/article/download/30682/27343 (дата обращения: 31.08.2017).
11. Теленик С.Ф., Ролик А.И., Савченко П.С., Боданюк М.Е. Управляемый генетический алгоритм в задачах распределения виртуальных машин в ЦОД / // Вісн. ЧДТУ. 2011. № 2. С. 104–113.

12. Ворожцов А.С., Тутова Н.В., Тутов А.В. Оптимизация размещения облачных серверов в центрах обработки данных // T-Comm: Телекоммуникации и транспорт. 2015. Т. 9. № 6. С. 4–8. URL: http://media-publisher.ru/old/pdf/Nom-6-2015-sait.pdf (дата обращения: 31.08.2017).

13. Verma A., Ahuja P., and Neogi A. pMapper: Power and Migration Cost Aware Application Placement in Virtualized Systems. Proc. 9th Intern. Middleware Conf. Belgium, 2008, December 1–5, pp. 243–264. DOI: 10.1007/978-3-540-89856-6_13.

14. Xu J., Fortes J. Multi-objective Virtual Machine Placement in Virtualized Data Center Environments. In Green Computing and Communications (GreenCom). Proc., IEEE/ACM Int. Conf. & Int. Conf. 2010, pp. 179–188. DOI: 10.1109/GreenCom-CPSCom.2010.137.

15. Козловский А.Л. Модели, методы и алгоритмы распределения ресурсов виртуализованных вычислительных кластеров: дис. … канд. технич. наук. М., 2012, 163 с. URL: http://www.dissercat.com/content/modeli-metody-i-algoritmy-raspredeleniya-resursov-virtualizovannykh-vychislitelnykh-klastero#ixzz4rLi0zP7N (дата обращения: 31.08.2017).

16. Коновалов М.Г., Малашенко Ю.Е., Назарова И.А. Управление заданиями в гетерогенных вычислительных системах // Изв. РАН: Теория и системы управления. 2011. № 2. С. 43–61.

17. Баранов А.В., Ляховец Д.С. Влияние пакетирования на эффективность планирования параллельных заданий // Программные системы: теория и приложения. 2017. Т. 8. № 1. С. 193–208.

18. Топорков В.В., Емельянов Д.М. Экономическая модель планирования и справедливого разделения ресурсов в распределенных вычисления // Программирование. 2014. Т. 40. № 1. С. 54–65. URL: http://naukarus.com/ekonomicheskaya-model-planirovaniya-i-spravedlivogo-razdeleniya-resursov-v-raspredelennyh-vychisleniyah (дата обращения: 31.08.2017).

19. Baranov A., Telegin P., Tikhomirov A. (2017) Comparison of Auction Methods for Job Scheduling with Absolute Priorities. Malyshkin V. (Eds). PaCT, 2017. Lecture Notes in Computer Science, vol. 10421. Springer, Cham. DOI: 10.1007/978-3-319-62932-2_37.