УДК 007.52:621.39
doi:10.31799/1684-8853-2019-5-30-37

Научные статьи
Articles

Подход к реализации распределенной системы виртуальных машин для самоорганизующихся сетей

С. В. Кулешов[a], *доктор техн. наук, главный научный сотрудник, orcid.org/0000-0002-8454-5598*
А. А. Зайцева[a], *канд. техн. наук, старший научный сотрудник, orcid.org/0000-0002-1345-8550, vaz1976@mail.ru*
И. О. Шальнев[a], *аспирант, orcid.org/0000-0001-9383-1089*
[a]*Санкт-Петербургский институт информатики и автоматизации РАН, 14-я линия В. О., 39, Санкт-Петербург, 199178, РФ*

Введение: *активные данные, представляя собой фрагменты исполнимого кода, передаваемого между узлами активной сети, являются эффективным механизмом функционирования программно-реконфигурируемых распределенных систем. Ранее в работах, посвященных активным данным, не уделялось достаточное внимание реализации среды выполнения (процессора) для исполнимого кода активных данных, а также вопросам построения гипервизоров и балансировки нагрузки в распределенных системах.* ***Цель:*** *разработка принципов построения виртуальных машин активных данных, обеспечивающих реконфигурируемость конечных устройств и гибкость сети в целом, определение возможности использования существующих подходов к балансировке нагрузки для сетей с активными данными.* ***Методы:*** *использованы принципы построения программно-определяемых систем, концепция активных данных, теоретические основы и технологии виртуализации.* ***Результаты:*** *рекомендовано в качестве среды выполнения активных данных использовать распределенную систему виртуальных машин, для построения которой выбран объектно-ориентированный подход к созданию распределенных приложений. При этом каждый узел такой распределенной системы виртуальных машин может выступать в качестве ведущего и ведомого узла при объектном взаимодействии. На основе сформулированного подхода предложено решение задачи построения сети ретрансляторов с применением активных данных, при этом беспилотный летательный аппарат рассматривается как элемент активной инфокоммуникационной сети, поддерживающий технологию активных данных. Так как распределенная система виртуальных машин допускает асимметричное распределение узлов децентрализованной сети, для распределенной системы, узлами которой являются беспилотные летательные аппараты и узел управления, разработан метод управления величиной асимметричности путем создания объектов различного уровня декомпозиции.* ***Практическая значимость:*** *представленные методы обеспечивают возможность управления ресурсопотреблением узлов распределенной программно-реконфигурируемой сети и количества передаваемых сетевых данных. Для динамического управления степенью загруженности узлов сети разработан вариант архитектуры менеджера ресурсов.*

Ключевые слова *— активные данные, виртуальная машина, распределенная сеть, беспилотный летательный аппарат, передача данных, программно-определяемые системы, самоорганизующиеся сети.*

Для цитирования: Кулешов С. В., Зайцева А. А., Шальнев И. О. Подход к реализации распределенной системы виртуальных машин для самоорганизующихся сетей. *Информационно-управляющие системы*, 2019, № 5, с. 30–37. doi:10.31799/1684-8853-2019-5-30-37

For citation: Kuleshov S. V., Zaytseva A. A., Shalnev I. O. Distributed system of virtual machines for self-organized networks. *Informatsionno-upravliaiushchie sistemy* [Information and Control Systems], 2019, no. 5, pp. 30–37 (In Russian). doi:10.31799/1684-8853-2019-5-30-37

Введение

Рост качества и скорости каналов связи делает популярными технические решения, основанные на распределенных приложениях с неравномерным распределением алгоритмической составляющей между узлами вычислительной сети.

Одновременно с этим новые объективные потребности создания программно-определяемых систем взамен морально устаревших программно-управляемых [1] диктуют необходимость исследования подходов, расширяющих функциональные возможности программного кода по изменению параметров аппаратных систем. Одним из таких подходов является подход активных данных [2–3].

Активные данные (АД), представляя собой фрагменты исполнимого кода, передаваемого между узлами активной сети, являются эффективным механизмом функционирования программно-реконфигурируемых распределенных систем. Ранее в работах, посвященных АД, не уделялось достаточное внимание среде выполнения исполнимого кода, а также вопросам построения гипервизоров и балансировки нагрузки в распределенных системах, когда узлы сети являются унифицированными. В данной работе рассматриваются вариант построения распределенной системы виртуальных машин для выполнения АД и существующие алгоритмы балансировки нагрузки в распределенных вычислительных сетях, определяются возможности исполь-

зования имеющихся подходов к балансировке нагрузки для сетей с АД.

Примечателен тот факт, что в вычислительных системах общего назначения, и в первую очередь в персональных компьютерах, существует обратная тенденция — разработка методов защиты от случайного или злонамеренного выполнения программного кода, расположенного в области данных.

В традиционном подходе (программно-управляемые, а не программно-определяемые вычисления [1]) память, используемую программой, делят на динамически распределяемую область памяти, сегменты данных и сегменты кода [4, 5]. Указатель команд при этом должен работать в пределах конкретного сегмента кода. Выход указателя за пределы этого сегмента и передача управления в другие области памяти может привести к выполнению процессором случайной последовательности команд, которое прервется только если встретится недопустимая битовая последовательность. В этом случае будет произведено аварийное завершение программы. Возможность выполнения команд из области памяти, в которой размещаются только данные, часто используется для несанкционированного доступа или внедрения вирусов.

Для предотвращения таких атак в операционных системах Linux, Mac OS X, Android и Windows была встроена функция безопасности DEP (Data Execution Prevention, технология предотвращения выполнения данных) [6], запрещающая приложениям доступ к области памяти, в которой размещаются только данные.

Однако в некоторых подходах используется возможность выполнения исполнимого кода в динамически распределяемой области памяти, а точнее, в «стеке» или «куче», например, в процессе оптимизации, динамической компиляции или отладки. Предлагаемая технология АД также основана на исполнении программного кода в области данных.

Одним из вариантов решения задачи обеспечения безопасности может быть применение виртуальных машин и гипервизоров для выполнения исполнимого кода АД.

Целесообразно в качестве среды выполнения АД использовать распределенную систему виртуальных машин, в основе которой выбран объектно-ориентированный подход для создания распределенных приложений.

Каждый узел распределенной системы может выступать в качестве как ведущего, так и ведомого узла при объектном взаимодействии. В этом случае ведомый узел содержит исполнимый код в библиотеках виртуальных машин с предопределенными в нем имплементациями объектов, к которым может обращаться исполнимый код АД. Таким образом, для прикладного программиста появляется возможность использовать объекты ведомых узлов, осуществляя управление ими из АД [7].

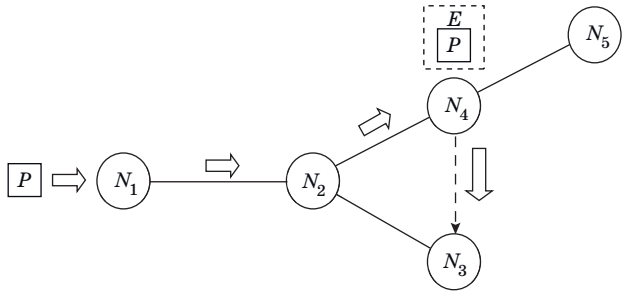
Компонентным объектом назовем совокупность объектов ведущей и ведомой частей, работающих совместно и представляющих собой единую программную сущность. Часть объекта, расположенная на управляющем узле, ответственна за программный интерфейс (API) описываемой сущности. Вторая часть объекта, расположенная на управляемой стороне, ответственна за функциональное представление сущности. Управление производится путем генерации управляющего кода, который должен быть передан на управляемый узел в пакете АД.

Решение задачи построения сети ретрансляторов с применением АД

Для формулирования требований к распределенной системе виртуальных машин рассмотрим задачу построения сети передачи данных на базе взаимодействия сети беспилотных летательных аппаратов (БЛА), каждый из которых может управляться и реконфигурироваться АД. Исполнимый код, передаваемый в виде данных, используется для сервисных (управление распространением, маршрутизация и навигация), а также конфигурационных (реконфигурация узлов) операций.

Таким образом, применение концепции АД дает возможность управлять поведением узлов сети (БЛА-ретрансляторов) при помощи самого потока данных без использования выделенного канала [8].

Одним из сценариев использования АД для задач инфокоммуникации является динамическая реконфигурация узлов. С узла-источника N_1 АД передаются на узел-получатель N_3. Программа P, выполняясь на узле N_4 (рис. 1), определяет возможность создания нового физического канала связи с узлом N_3 путем реконфигурирования аппаратной или протокольной части аппаратуры этого узла. В случае если такая реконфигурация будет оптимальной (в смысле минимизации вре-



■ ***Рис. 1.*** Использование динамической реконфигурации [2]

■ ***Fig. 1.*** The dynamic reconfiguration using [2]

мени доставки или минимизации ресурсов на осуществление передачи), то производится реконфигурирование и передача АД по вновь созданному физическому каналу связи.

Рассмотрим БЛА как элемент активной инфокоммуникационной сети, поддерживающий технологию АД. В этом случае согласно [2] каждый пакет данных может содержать фрагмент исполнимого кода, который запускается в виртуальной машине для АД — ADVM (Active Data Virtual Machine). Исполнимый код при этом может иметь доступ как к функциям коммуникационной аппаратуры, так и к транспортным функциям БЛА и компонентам стека программного обеспечения полетного контроллера (flight stack).

Асинхронный характер получения пакетов каждым узлом сети вызывает конкуренцию пакетов за право быть выполненным процессором ADVM. Разрешением подобных конфликтов должен заниматься арбитр АД в соответствии с политиками многопоточной обработки. Подобная проблема очень близка к задаче изоляции контекстов в мультизадачных операционных системах и может стать темой дальнейших исследований. В данной работе рассматривается вырожденный случай, и источником управляющих воздействий, передаваемых в АД, упрощенно считается единственный доверенный узел. В этом случае арбитраж активных пакетов заметно упрощается, а необходимостью переключения контекстов и сохранения их состояния можно пренебречь.

Управление степенью асимметричности децентрализованной сети

Рассмотрим возможность управления степенью асимметричности децентрализованной сети на примере построения распределенной системы, состоящей из управляемых (БЛА) и управляющих узлов. Разберем два варианта реализации задачи пролета БЛА по маршруту. В первом случае управляемый узел (БЛА) обладает большей алгоритмической составляющей, и программисту при реализации управляющего алгоритма достаточно вызвать один метод — полет по точкам маршрута. Во втором случае программа БЛА оперирует более низкоуровневыми примитивами: переместиться вперед, повернуться на заданный курсовой угол и т. п. Здесь алгоритмическая составляющая такого узла может быть меньше.

Опишем величину асимметричности децентрализованной сети. Величина алгоритмической составляющей узла определяется объемом кода вызываемых функций объектов (методов), доступных для вызова кодом из пакетов АД:

$$l(x_c) = \sum_{i=1}^{n} l(m_i), \qquad (1)$$

где $l(x_c)$ — объем программных компонентов узла; m_i — описание i-го метода; $l(m_i)$ — объем программных компонентов метода m_i; n — количество методов, необходимое для реализации узла.

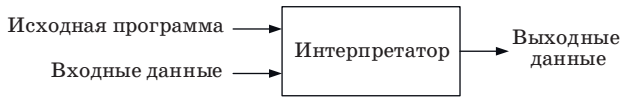
Алгоритмическую составляющую АД оценим через длину исполнимого байт-кода (bytecode) в пакетах АД $l(x_{\text{АД}})$. Тогда величину асимметричности децентрализованной сети T определим через отношение

$$T = \frac{l(x_{\text{АД}})}{\sum_{i=1}^{n} l(m_i)}. \qquad (2)$$

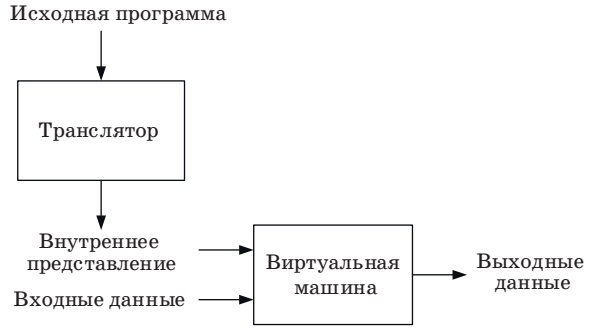
Для исполнения управляющего кода, принимаемого в пакетах АД, можно использовать интерпретатор, выполняющий операции, указанные в исходной программе, над входными данными (рис. 2).

Существует отдельный вид языкового процессора, называемый гибридным компилятором, который объединяет в себе компиляцию и интерпретацию (рис. 3). Исходная программа должна сначала компилироваться во внутреннее представление, называемое байт-кодом. Сформированный байт-код (рис. 4) интерпретируется виртуальной машиной [9].

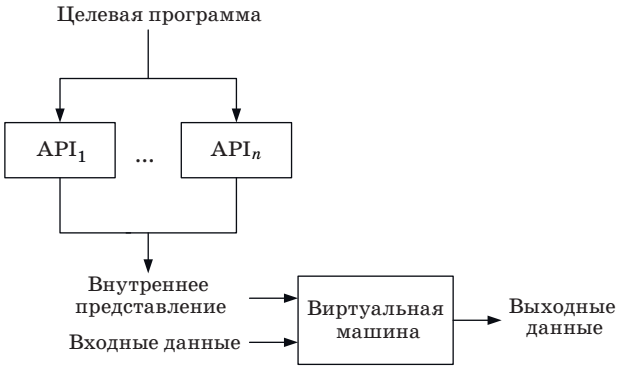
Внутреннее представление обычно является набором базовых команд, напоминающих машинно-ориентированные ассемблерные. Виртуальная машина представляется как целевая архитектура для таких команд, реализованная программно на архитектуре процессора узла. Целый класс языков программирования, таких как Java, C#, LISP



■ ***Рис. 2.*** Принцип работы интерпретатора
■ ***Fig. 2.*** The principle of the interpreter



■ ***Рис. 3.*** Структура гибридного компилятора
■ ***Fig. 3.*** The hybrid compiler structure



■ ***Рис. 4.*** Формирование байт-кода на управляющей стороне для интерпретации на узле

■ ***Fig. 4.*** The bytecode generation on the control side for interpretation on the node

и многие другие, используют данный подход. Виртуальные машины используются в основном для платформо-независимого исполнения кода и обеспечивают большую производительность по сравнению с интерпретацией и выполнением непосредственно исходного кода программы [10, 11]. Такая реализация позволяет строить гетерогенные распределенные сети, включающие узлы на процессорах с различными архитектурами.

Приведенная схема справедлива как для случая, когда инициирующей стороной выступает управляющий узел, так и для случая, когда управляемый узел формирует байт-код для исполнения на виртуальной машине другого узла. Таким образом, можно утверждать, что данный подход организует распределенное объектное взаимодействие посредством распределенной системы виртуальных машин.

Динамическое управление степенью загруженности узлов сети

В рассматриваемом примере АД, будучи запущенными на одном узле, распространяются по всем доступным узлам для выполнения полезной работы.

В этом случае возможны следующие стратегии распределения исполнимого кода по узлам сети.

1. Исполнимый код в АД сопровождает полезные данные и осуществляет задачу навигации этих полезных данных. Тогда программа АД, запускаясь на узле, производит оценку окружения (доступности последующих узлов, тестирования возможностей аппаратуры) и выбирает направления для следующего шага распространения полезных данных. При этом исполнимый код повторяет путь следования полезных данных, проходя через те же самые узлы.

2. Пересылка исполнимого кода, управляющего аппаратным обеспечением, в том числе пересылка кодеков для подготовки узлов к приему полезных данных. В этом случае исполнимый код должен быть доставлен на тот узел, на который адресован.

3. Выполнение дополнительной работы для задач кодирования/декодирования потока полезных данных, задач принятия решений об управлении физической конфигурацией группировки БЛА и т. п. В таком случае исполнимый код АД должен распространяться по узлам сети по правилам, известным из распределенных вычислительных и, в том числе, многопроцессорных систем. В связи с этим требуется решать задачу динамической балансировки нагрузки на узлы сети путем перераспределения ее алгоритмической составляющей.

Рассмотрим распределенную программно-реконфигурируемую систему, в узлах которой находятся виртуальные машины, как частный случай распределенной многопроцессорной системы.

В данном случае под многопроцессорной системой понимается система, содержащая несколько процессорных элементов, на которых размещаются виртуальные машины, без общей памяти, которые могут обмениваться сообщениями, содержащими АД.

В связи с тем, что для выполнения исполнимого кода АД используются виртуальные машины, можно считать рассматриваемую многопроцессорную систему гомогенной даже при гетерогенной структуре узлов (используются разнородные процессоры и различная физическая природа узлов: стационарные, мобильные, летающие).

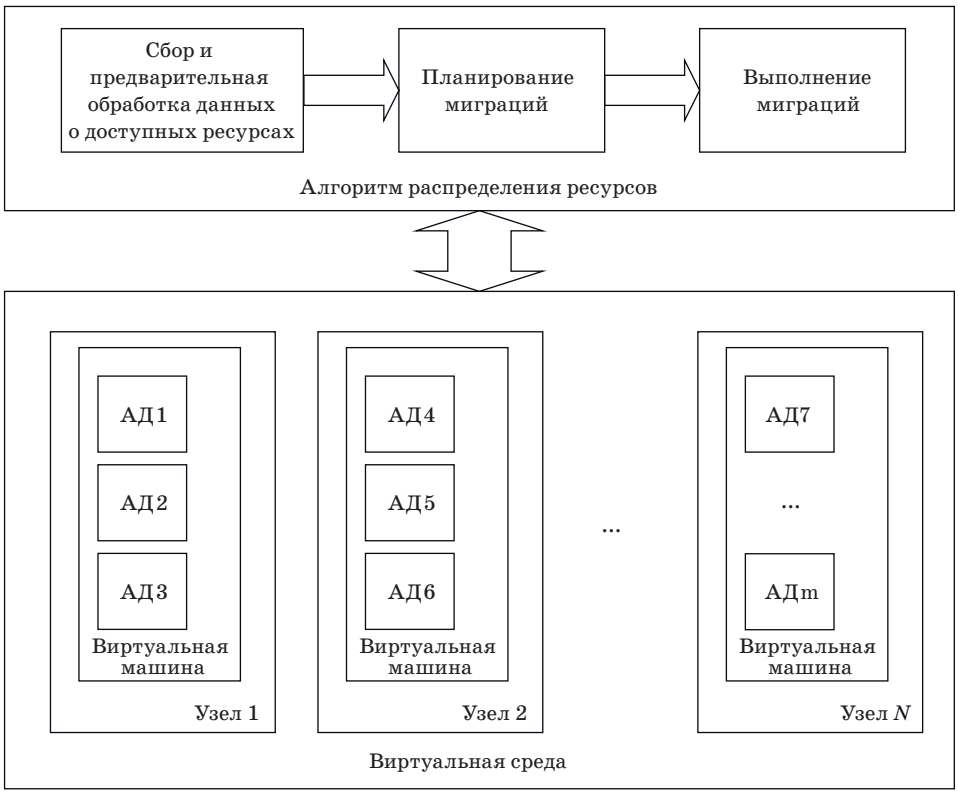
Традиционно в системах виртуализации за распределение ресурсов между физическими хостами отвечает гипервизор (hypervisor), используя алгоритмы балансировки нагрузки [12–15].

Существуют два основных подхода в балансировке нагрузки, которые можно обозначить как выталкивающие миграции и притягивающие миграции. Выталкивающие миграции — перенос полезной нагрузки на наименее нагруженный узел с более нагруженного. При использовании подхода с притягивающими миграциями нагрузка переносится на узел, когда этот узел начинает простаивать [16].

Управление ресурсами в рамках распределенной системы виртуальных машин представляет собой итерационный процесс, состоящий из следующих этапов (рис. 5).

1. Гипервизор (менеджер ресурсов) собирает статистические данные об используемых ресурсах за определенный временной интервал. На основе предварительной обработки принимается решение о запуске перераспределения ресурсов в случае нарушения предварительно заданных условий.

2. На этапе планирования миграций менеджер ресурсов на основании конкретного алго-



■ ***Рис. 5***. Архитектура менеджера ресурсов
■ ***Fig. 5***. The resource manager architecture

ритма балансировки формирует план миграции, который должен устранить или минимизировать возникшие нарушения условий.

3. Выполняется перераспределение алгоритмической составляющей между узлами путем передачи АД между узлами согласно плану миграции.

Основным отличием балансировки нагрузки в распределенной системе виртуальных машин от подходов к балансировке в облачных сервисах или механизмов RPC (Remote Procedure Call) можно считать, в частности, асинхронность процессов, а также отсутствие необходимости сериализации объектов.

Рассмотрим существующие алгоритмы балансировки нагрузки, используемые в виртуализации.

Размещение виртуальных машин с учетом трафика. В работе [17] предлагается рассматривать задачу балансировки как частный случай задачи о многомерном рюкзаке. В качестве измерений рюкзака выбраны оперативная память, процессорное время и ресурс межпроцессорной шины, используемые виртуальной машиной. Предложенный подход использует алгоритм «первый подходящий» для первичного размещения виртуальных машин на физической машине, а дальше на каждом шаге проверяется сбалансированность потребления процессорного времени.

Взвешенная рандомизированная миграция. Подход, описанный в статье [18], предлагает для каждой виртуальной машины находить наиболее подходящий узел, основываясь на штрафе за использование межпроцессорной шины. Исследования проводились на базе платформы визуализации Xen. Критерии штрафа опираются на значения, вычисляемые по счетчикам производительности процессора.

Циклический алгоритм Round Robin, или алгоритм кругового обслуживания, работает по принципу кругового цикла, или «карусели», запросы передаются по очереди каждому следующему серверу соответственно, а при достижении последнего сервера при следующем запросе производится обращение снова к первому. Более подробно данный алгоритм описан в работе [19].

Взвешенный циклический алгоритм [20] — усовершенствованная версия алгоритма Round Robin, отличается тем, что серверам ставятся в соответствие весовые коэффициенты, их значение определяется производительностью сервера.

Основная цель использования перечисленных алгоритмов состоит в перемещении объектов между виртуальными машинами с перераспределением нагрузки с перегруженных узлов, а также в обеспечении возможности динамически консолидировать объекты на наименьшем числе

узлов, чтобы обеспечивать энергоэффективность узлов при низкой загрузке [21].

Тем не менее непрерывная миграция имеет достаточно высокую цену, обоснованную потерей производительности виртуальных машин в процессе миграции, и соответствующее увеличение энергозатрат.

Для каждого узла в составе гипервизора добавлен менеджер ресурсов, который получает решения о распределении ресурсов виртуальных машин других узлов. Каждый менеджер ресурсов участвует в динамическом определении общих ресурсов, предназначенных для распределения на его виртуальной машине, и соответственно принимает решение о распределении. Отличительной особенностью менеджера ресурсов является дискретный принцип его работы: решения о распределении ресурсов принимаются в дискретные промежутки времени на следующий дискретный временной промежуток.

Заключение

В статье рассматривается развитие подхода АД для реализации динамической программно-определяемой системы на примере группировки БЛА-ретрансляторов.

Сложившееся на текущий момент времени противоречие в тенденциях создания автономных инфокоммуникационных систем требует разработки новых теоретических подходов и архитектурных решений. С одной стороны, в соответствии с современными тенденциями развития технологий программирования требуется четкое разделение на исполнимый код и код, защищенный от исполнения. С другой стороны, сама концепция построения программно-определяемых систем требует максимальной однородности исполняемого кода и данных. В качестве пути устранения такого противоречия предлагается использовать виртуальные машины, обладающие тьюринг-полнотой, но с ограниченными функциональными возможностями (режим «песочницы»), на которых выполняется исполнимый код АД.

Объединение отдельных виртуальных машин на узлах позволяет рассматривать их совокупность как распределенную систему виртуальных машин, с асимметричным распределением алгоритмической составляющей по узлам, построенную на принципах объектно-ориентированной парадигмы и ориентированную на исполнение байт-кода.

Особенностями такой архитектуры являются:

— динамическое управление объектами на управляемых узлах, дающее возможность удаленного реконфигурирования узла с реализацией концепции программно-управляемого подхода;

— использование объектно-ориентированной парадигмы для взаимодействия объектов в распределенной системе, что позволяет строить объектное взаимодействие в привычной для программистов форме;

— возможность управления величиной асимметричности путем создания объектов различного уровня декомпозиции и как следствие возможность регулирования энергопотребления узла и количества передаваемых сетевых данных.

Для случаев решения задач (кейсов), допускающих распараллеливание на сеть узлов, рассмотрены варианты применения балансировки нагрузки между узлами для получения наибольшей эффективности работы всей системы.

Предлагаемый подход является альтернативным решением построения самоорганизующихся сетей, основанным на агрегации принципов работы облачных сервисов и программно-определяемых систем, реализованных через выполнение программного кода АД на узлах сети.

Дальнейшие исследования предполагают экспериментальную отработку взаимодействия виртуальных машин для наиболее востребованных вариантов применения АД.

Финансовая поддержка

Работа выполнена в рамках реализации Государственного задания на 2019 г. № 0073-2019-0005.

Литература

1. **Кулешов С. В., Зайцева А. А., Аксенов А. Ю.** Развитие информационных технологий: программируемое и непрограммируемое. *Информатизация и связь*, 2017, № 3, с. 34–39.
2. **Кулешов С. В., Цветков О. В.** Активные данные в цифровых программно-определяемых системах. *Информационно-измерительные и управляющие системы*, 2014, № 6, с. 12–19.
3. **Alexandrov V. V., Kuleshov S. V., Zaytseva A. A.** *Active Data in Digital Software Defined Systems Based on SEMS Structures.* In: Smart Electromechanical Systems. Ed. A. Gorodetskiy. Studies in Systems, Decision and Control, Springer, Cham, 2016, vol. 49, pp. 61–69.
4. **Sharp B. L., Peterson G. D., Lok K. Y.** Extending hardware based mandatory access controls for memory to multicore architectures. *Proc. of 4th Annual Cyber Security and Information Intelligence Research*

Workshop: Developing Strategies to Meet the Cyber Security and Information Intelligence Challenges Ahead, 2008. doi:10.1145/1413140.1413167
5. **Stakem P. H.** *Computer Architecture & Programming of the Intel x86 Family*. 2013. 173 p.
6. *Data Execution Prevention — Technical Manual by Hewlett-Packard Development Company*. http://www1.hp.com/ctg/Manual/c00387685 (дата обращения: 12.03.2019).
7. **Шальнев И. О.** Подход к построению распределенных систем на основе балансировки объема исполняемого кода между узлами. *Технологическая перспектива в рамках Евразийского пространства: новые рынки и точки экономического роста: материалы 4-й Международной научной конференции*, Санкт-Петербург, 13–15 декабря 2018 г., СПб., Астерион, 2018, с. 165–172.
8. **Kuleshov S. V., Zaytseva A., Aksenov A. Y.** The conceptual view of unmanned aerial vehicle implementation as a mobile communication node of active data transmission network. *International Journal of Intelligent Unmanned Systems*, 2018, vol. 6, iss. 4, pp. 174–183. doi:10.1108/IJIUS-04-2018-0010
9. **Aho A. V., Lam M. S., Sethi R., Ulman J. D.** *Compilers. Principles, Techniques, & Tools*. Second Ed. Addison Wesley, 2007. 1038 p.
10. **Craig Iain D.** *Virtual Machines*. Springer-Verlag London, 2006. 269 p. doi:10.1007/978-1-84628-246-1
11. **Shi Y., Gregg D., Beatty A., Ertl M. A.** *Virtual Machine Showdown: Stack Versus Registers*. https://www.usenix.org/legacy/events/vee05/full_papers/p153-yunhe.pdf (дата обращения: 11.03.2019).
12. **Abeni L., Biondi A., Bini E.** Hierarchical scheduling of real-time tasks over Linux-based virtual machines. *Journal of Systems and Software*, 2019, vol. 149, pp. 234–249.
13. **Diener M., Cruz E. H. M., Navaux P. O. A.** Locality vs Balance: Exploring data mapping policies on NUMA systems. *23rd Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, Turku, Finland, 2015, vol. 1, pp. 9–16. doi:10.1109/PDP.2015.11
14. **Хританков А. С.** *Об одном алгоритме балансировки вычислительной нагрузки в распределенных системах*. http://www.ict.edu.ru/vconf/files/11951.pdf (дата обращения: 11.03.2019).
15. *Балансировка нагрузки в распределенных системах*. http://masters.donntu.org/2012/fknt/volokhova/library/article3.htm (дата обращения: 12.03.2019).
16. **Polenov M., Guzik V., Lukyanov V.** *Hypervisors comparison and their performance testing*. In: Advances in Intelligent Systems and Computing, Springer, Cham, 2019, vol. 763, pp. 148–157. doi:https://doi.org/10.1007/978-3-319-91186-1_16
17. **Cheng Y., Chen W., Wang Z., Yu X.** Performance-monitoring-based traffic-aware virtual machine deployment on NUMA systems. *IEEE Systems Journal*, 2017, vol. 11, no. 2, pp. 973–982. doi:10.1109/JSYST.2015.2469652
18. **Rao J., Wang K., Zhou X., Xu C.** Optimizing virtual machine scheduling in NUMA multicore systems. *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, Shenzhen, 2013, pp. 306–317. doi:10.1109/HPCA.2013.6522328
19. **Aymaz Ş., Çavdar T., Aymaz S., Öztürk E.** An analysis of load balancing strategies with wireshark in software defined networks. *2018 International Conference on Artificial Intelligence and Data Processing (IDAP)*, Malatya, Turkey, 2018, pp. 1–5. doi:10.1109/IDAP.2018.8620766 19
20. **Liu G., Wang X.** A modified Round-Robin load balancing algorithm based on content of request. *2018 5th International Conference on Information Science and Control Engineering (ICISCE)*, Zhengzhou, 2018, pp. 66–72. doi:10.1109/ICISCE.2018.00023
21. **Minarolli D., Mazrekaj A., Freisleben B.** Tackling uncertainty in long-term predictions for host overload and underload detection in cloud computing. *Journal of Cloud Computing*, 2017, vol. 6:4, no. 1. https://link.springer.com/content/pdf/10.1186%2Fs13677-017-0074-3.pdf (дата обращения: 13.03.2019).

UDC 007.52:621.39
doi:10.31799/1684-8853-2019-5-30-37

Distributed system of virtual machines for self-organized networks

S. V. Kuleshov[a], Dr. Sc., Tech., Principal Researcher, orcid.org/0000-0002-8454-5598
A. A. Zaytseva[a], PhD, Tech., Senior Researcher, orcid.org/0000-0002-1345-8550, vaz1976@mail.ru
I. O. Shalnev[a], Post-Graduate Student, orcid.org/0000-0001-9383-1089
[a]Saint-Petersburg Institute for Informatics and Automation of the RAS, 39, 14 Line, V. O., 199178, Saint-Petersburg, Russian Federation

Introduction: Active data, being fragments of executable code transmitted between the nodes of an active network, are an effective mechanism for the operation of software-reconfigurable distributed systems. Previously, in the works devoted to active data, not enough attention was paid to the implementation of the runtime environment (the processor) for the executable code of active data, as well as to the issues of building hypervisors and load balancing in distributed systems. **Purpose:** Developing principles for the construction of

virtual machines with active data, providing the reconfigurability of the target devices and network flexibility in general. Evaluating the possibility of using the existing approaches to load balancing for networks with active data. **Methods:** Our study uses the principles of software-defined system development, the conception of active data, theoretical foundations and technology of virtualization. **Results:** Is has been proposed to use a distributed system of virtual machines as an active data execution environment, based on the object-oriented approach to creating distributed applications. Each node of such a distributed system of virtual machines can act as either a control or slave node during the object interaction. Based on the developed approach, we proposed to solve the problem of building a network of repeaters using active data, considering an unmanned aerial vehicle as an element of an active info-communication network which supports the active data technology. Since a distributed system of virtual machines enables asymmetric distribution of decentralized network nodes, a method has been developed for a distributed system whose nodes are unmanned aerial vehicles and a control node, to control the asymmetry value by creating objects of various decomposition levels. **Practical relevance:** The proposed methods provide a way to control the resource consumption of the nodes of a distributed software-reconfigurable network and the amount of network data transmitted. For dynamic management of the load on the network nodes, a resource manager architecture and a resource allocation algorithm are developed.

Keywords — active data, virtual machine, distributed network, unmanned aerial vehicle, data transmission, software-defined systems, self-organized networks.

For citation: Kuleshov S. V., Zaytseva A. A., Shalnev I. O. Distributed system of virtual machines for self-organized networks. *Informatsionno-upravliaiushchie sistemy* [Information and Control Systems], 2019, no. 5, pp. 30–37 (In Russian). doi:10.31799/1684-8853-2019-5-30-37

References

1. Kuleshov S. V., Zaytseva A. A., Aksenov A. Y. Data processing technologies: programming and non-programming. *Informatization and Communication*, 2017, no. 3, pp. 34–39 (In Russian).
2. Kuleshov S. V., Tsvetkov O. V. Active data in digital software-defined systems. *Informatsionno-izmeritelnye i upravlyayushchie sistemy*, 2014, no. 6, pp. 12–19 (In Russian).
3. Alexandrov V. V., Kuleshov S. V., Zaytseva A. A. *Active Data in Digital Software Defined Systems Based on SEMS Structures*. In: *Smart Electromechanical Systems*. Ed. A. Gorodetskiy. Studies in Systems, Decision and Control, Springer, Cham, 2016, vol. 49, pp. 61–69.
4. Sharp B. L., Peterson G. D., Lok K. Y. Extending hardware based mandatory access controls for memory to multicore architectures. *Proc. of 4th Annual Cyber Security and Information Intelligence Research Workshop: Developing Strategies to Meet the Cyber Security and Information Intelligence Challenges Ahead*, 2008. doi:10.1145/1413140.1413167
5. Stakem P. H. *Computer Architecture & Programming of the Intel x86 Family*. 2013. 173 p.
6. *Data Execution Prevention — Technical Manual by Hewlett-Packard Development Company*. Available at: http://www1.hp.com/ctg/Manual/c00387685 (accessed 12 March 2019).
7. Shalnev I. O. Building distributed systems approach based on balancing of a source code capacity between network nodes. *Materialy 4-y Mezhdunarodnoy nauchnoy konferentsii "Tekhnologicheskaya perspektiva v ramkakh Yevraziyskogo prostranstva: novyye rynki i tochki ekonomicheskogo rosta"* [Materials of the 4th International Scientific Conference "Technological perspective in the framework of the Eurasian space: new markets and points of economic growth"], Sint-Petersburg, 2018, pp. 165–172 (In Russian).
8. Kuleshov S. V., Zaytseva A., Aksenov A. Y. The conceptual view of unmanned aerial vehicle implementation as a mobile communication node of active data transmission network. *International Journal of Intelligent Unmanned Systems*, 2018, vol. 6, iss. 4, pp. 174–183. doi:10.1108/IJIUS-04-2018-0010
9. Aho A. V., Lam M. S., Sethi R., Ulman J. D. *Compilers. Principles, Techniques, & Tools*. Second Ed. Addison Wesley, 2007. 1038 p.
10. Craig Iain D. *Virtual Machines*. Springer-Verlag London, 2006. 269 p. doi:10.1007/978-1-84628-246-1
11. Shi Y., Gregg D., Beatty A., Ertl M. A. *Virtual Machine Showdown: Stack Versus Registers*. Available at: https://www.usenix.org/legacy/events/vee05/full_papers/p153-yunhe.pdf (accessed 11 March 2019).
12. Abeni L., Biondi A., Bini E. Hierarchical scheduling of real-time tasks over Linux-based virtual machines. *Journal of Systems and Software*, 2019, vol. 149, pp. 234–249.
13. Diener M., Cruz E. H. M., Navaux P. O. A. Locality vs balance: Exploring data mapping policies on NUMA systems. *23rd Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, Turku, Finland, 2015, vol. 1, pp. 9–16. doi:10.1109/PDP.2015.11
14. Khritankov A. S. *Ob odnom algoritme balansirovki vychislitel'noy nagruzki v raspredelennykh sistemakh* [On one algorithm for balancing the computational load in distributed systems]. Available at: http://www.ict.edu.ru/vconf/files/11951.pdf (accessed 11 March 2019).
15. *Balansirovka nagruzki v raspredelennykh sistemakh* [Load balancing in distributed systems]. Available at: http://masters.donntu.org/2012/fknt/volokhova/library/article3.htm (accessed 12 March 2019).
16. Polenov M., Guzik V., Lukyanov V. *Hypervisors comparison and their performance testing*. In: *Advances in Intelligent Systems and Computing*. Springer, Cham, 2019, vol. 763, pp. 148–157. doi:https://doi.org/10.1007/978-3-319-91186-1_16
17. Cheng Y., Chen W., Wang Z., Yu X. Performance-monitoring-based traffic-aware virtual machine deployment on NUMA systems. *IEEE Systems Journal*, 2017, vol. 11, no. 2, pp. 973–982. doi:10.1109/JSYST.2015.2469652
18. Rao J., Wang K., Zhou X., Xu C. Optimizing virtual machine scheduling in NUMA multicore systems. *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, Shenzhen, 2013, pp. 306–317. doi:10.1109/HPCA.2013.6522328
19. Aymaz Ş., Çavdar T., Aymaz S., Öztürk E. An analysis of load balancing strategies with wireshark in software defined networks. *2018 International Conference on Artificial Intelligence and Data Processing (IDAP)*, Malatya, Turkey, 2018, pp. 1–5. doi:10.1109/IDAP.2018.8620766 19
20. Liu G., Wang X. A Modified Round-Robin load balancing algorithm based on content of request. *2018 5th International Conference on Information Science and Control Engineering (ICISCE)*, Zhengzhou, 2018, pp. 66–72. doi:10.1109/ICISCE.2018.00023
21. Minarolli D., Mazrekaj A., Freisleben B. Tackling uncertainty in long-term predictions for host overload and underload detection in cloud computing. *Journal of Cloud Computing*, 2017, vol. 6:4, no. 1. Available at: https://link.springer.com/content/pdf/10.1186%2Fs13677-017-0074-3.pdf (accessed 13 March 2019). doi:10.1186/s13677-017-0074-3