14. Huo X., Huo M., Karimi H.R. Attitude stabilization control of a quadrotor UAV by using backstepping approach. *Mathematical Problems in Engineering*, 2014, vol. 2014, pp. 749803. DOI: https://doi.org/10.1155/2014/749803.
15. Liu Y., Ma J., Tu H. Robust command filtered adaptive backstepping control for a quadrotor aircraft. *Journal of Control Science and Engineering*, 2018, vol. 2018, pp. 1854648. DOI: https://doi.org/10.1155/2018/1854648.
16. Gen K., Chulin N.A. Quadcopter guidance algorithm with obstacle avoidance and planned route tracking based on normal acceleration control. *Problemy sovremennoj nauki i obrazovanija*, 2016, no. 31 (73), pp. 6–28. (In Russian.)
17. Honglei A. et al. Backstepping-based inverse optimal attitude control of quadrotor. *International Journal of Advanced Robotic Systems*, 2013, vol. 10, pp. 223.
18. Bouabdallah S. *Design and Control of Quadrotors with Application to Autonomous Flying*. PhD Thesis. Lausanne, Switzerland: École Polytechnique Federale de Lausanne, 2007. 155 p.
19. Farrell J.A. et al. Command filtered backstepping. *IEEE Transactions on Automatic Control*, 2009, vol. 54, no. 6, pp. 1391–1395. DOI: https://doi.org/10.1109/TAC.2009.2015562.
20. Besekerskij V.A., Popov E.P. *Theory of Automatic Control Systems*. Moscow: Nauka, 1975, 768 p. (In Russian.)
21. Isaev A.M. et al. Hardware and software complex for simulating the flight of a multi-rotor UAV. *Infokommunikacionnye tehnologii*, 2020, vol. 19, no. 2, pp. 177–187. DOI: https://doi.org/10.18469/ikt.2020.18.2.08. (In Russian.)

Received 10.07.2020

УДК 004.75

ВИРТУАЛИЗАЦИЯ СЕРВЕРНОЙ ИНФРАСТРУКТУРЫ КОРПОРАТИВНЫХ ЦЕНТРОВ ОБРАБОТКИ ДАННЫХ

Мочалов В.П., Линец Г.И., Палканов И.С.
Северо-Кавказский федеральный университет, Ставрополь, РФ
E-mail: kbytw@mail.ru

Объектом исследования являются кластерные системы центров обработки данных, содержащие определенное множество серверов приложений, файл серверов, систем хранения данных, систему ввода-вывода связанных между собой системой коммутации и каналами связи. Целью работы является повышение эффективности использования виртуализированных кластерных систем путем разработки и применения рационального метода распределения виртуальных машин по физическим элементам центров обработки данных. В основу предложенного метода положен итерационный жадный алгоритм и процедура ограниченного перебора, позволяющие увеличить производительность виртуализированной системы за счет рационального распределения данных. Разработана имитационная модель функционирования виртуализированной системы, проведено ее экспериментальное исследование. Представленные подходы обеспечивают определение требуемого числа узлов кластера виртуализированной системы при непредсказуемой интенсивности нагрузки и непредсказуемого уровня запрашиваемых ресурсов виртуальными машинами.

Ключевые слова: *центры обработки данных, виртуальные машины, физические серверы, алгоритмы распределения ресурсов, жадный алгоритм, ограниченный перебор*

Введение

Современные центры обработки данных (ЦОД), используя технологии программно-конфигурируемых сетей, средства виртуализации серверной инфраструктуры FirstFit, RandomFit, Min-Min, Max-Min, CPULoad, Windows Azure и поддерживающие разнообразные ИТ-приложения, предоставляют пользователям разнообразные услуги и сетевые сервисы с учетом соглашений об уровнях обслуживания (SLA). Виртуализация серверной инфраструктуры, направленная на повышение эффективности управления ресурсами ЦОД и основанная на технологиях поддержки гипервизоров, обеспечивает распределение виртуальных машин (ВМ) между физическими серверами ЦОД, используя при этом алгоритмы оптимизации их загрузки.

Применяемые гипервизоры (например, VMware Infrastructure, VMware ESXi Server, Microsoft Hyper-V) используют как методы прямого распределения приложений, при которых каждому приложению предоставляется отдельная ВМ, а перераспределение возможно только в процессе их реализации, как и настройки, подобранные опытным путем и основанные, как правило, на статистике прогнозирования использования ресурсов [1–3; 7].

При превышении значения интенсивности нагрузки и ее объемов заранее заданного порогового значения сервер считается перегруженным и осуществляет миграцию нагрузки на менее нагруженные хосты. Использование планировщика ресурсов ЦОД на базе алгоритмов платформы OpenStack, обеспечивающих случайное распределение поступающих приложений на множество ресурсов, не учитывает в полной мере динамически изменяющуюся, непредсказуемую интенсивность нагрузки, динамику происходящих процессов.

Управляющие воздействия формируются здесь одновременно с появлением перегрузки и не обеспечивают решения задачи оптимального разделения ресурсов физических серверов между виртуальными машинами и сетевыми приложениями в реальном времени, а значит, и гарантированного обеспечения SLA [4–6; 9]. Повышение эффективности функционирования аппаратно-программной платформы ЦОД определяет необходимость реализации оптимизационных алгоритмов размещения ВМ на физических серверах в реальном времени на основе балансировки нагрузки.

Алгоритм виртуализации серверной инфраструктуры

Состав современного ЦОД включает в себя серверный комплекс, устройства хранения данных (сетевая система хранения данных); телекоммуникационное оборудование, обеспечивающее интеграцию всех устройств в единую систему и связь с внешними системами; систему управления ЦОД.

При решении задачи оптимального распределения ВМ по физическим серверам аппаратно-программной платформы ЦОД необходимо учитывать не только объемы ресурсов данного оборудования и его загрузку, но и особенности построения систем хранения данных (с прямым подключением DAS, сетевая NAS, сети хранения данных SAN, контентно-адресуемое хранилище CAS, иерархическая система HSM и др.), оказывающих существенное влияние на параметры, объем и скорость доступа к информации [10–12; 8]. При этом возможно применение как статического, так и динамического подходов. Статический подход применим в случае достаточно стабильной, предсказуемой интенсивности нагрузки и ее объемов и реализуется путем настройки гипервизора с использованием статистических методов. Он не учитывает в реальном времени динамику происходящих процессов и определяет момент перегрузки аппаратной платформы ЦОД после того, как она произойдет.

Некоторые подходы к решению подобных задач представлены в моделях Э.Х. Гимади и др. [13–15]. Реализованные здесь модели основаны на точных и приближенных эвристических решениях задачи многомерной упаковки в контейнерах и методах целочисленной линейной оптимизации.

Динамический подход используется при непредсказуемой интенсивности и объемах нагрузки, при существенном увеличении объемов решаемых задач, росте числа пользователей, воздействии других трудно формализуемых факторов и основан на оптимизационных алгоритмах долгосрочного прогноза использования ресурсов. Он должен обеспечивать в реальном времени выделение ресурсов ЦОД для каждой ВМ без предварительного планирования и распределения, определять соответствие ВМ определенному физическому серверу при использовании минимального их количества и поддерживать уровни производительности ВМ в соответствии с SLA (например, платформа виртуализации VMware vSphere, реализующая технологию динамического распределения нагрузки Assignable Hardware, или технология Live Migration) [16–18]. Структура системы управления аппаратно-программной платформы ЦОД приведена на рисунке.

По требованию пользователей ЦОД она производит формирование ВМ с определенными ресурсными параметрами. Локальный менеджер осуществляет контроль загрузки физических серверов и размещение запросов на ресурсы ВМ. Система мониторинга ЦОД осуществляет передачу полученной информации глобальному менеджеру, который осуществляет миграцию ВМ и управляет перераспределением физических серверов. Основной проблемой реализации динамического подхода является определение моментов перегрузки серверов ЦОД в будущем для проведения миграции приложений в реальном времени, т. е. адаптация распределения ресурсов физических серверов к нагрузкам ВМ. Долгосрочный про-

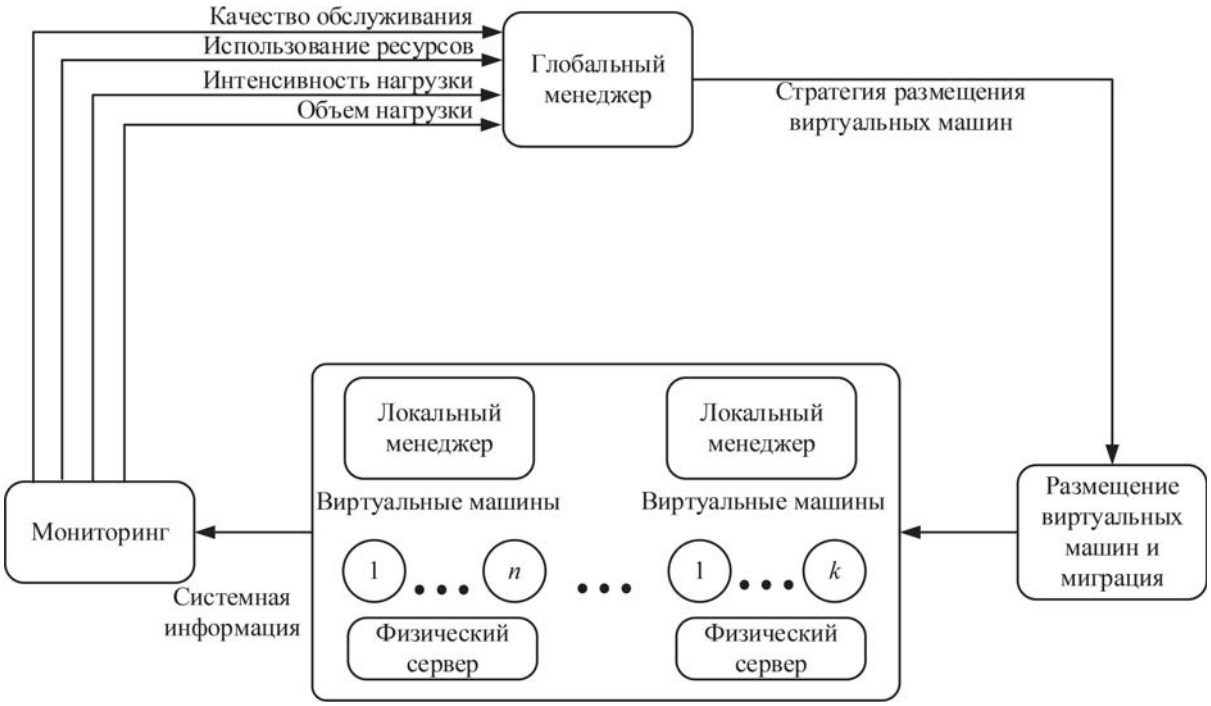

Рисунок. Структура системы управления аппаратно-программной платформой ЦОД

гноз использования ресурсов ЦОД может быть основан на регрессионных моделях (модель Бокса – Дисенникса), вариационной нейронной сети, машинном обучении, теории принятия решений. Субоптимальные решения краткосрочных и долгосрочных прогнозов осуществляют, используя различные оптимизационные алгоритмы, алгоритмы эвристики, генетические алгоритмы, нейронные сети [19].

Реализация статического подхода распределения ресурсов ЦОД, удовлетворяющего требованиям приложений, может быть проведена поэтапно. На первом этапе для реализации запросов приложений определяются требуемое количество и тип ВМ. В качестве технических характеристик ВМ при этом можно использовать, например, типовой ряд ВМ компании Amazon EC2 как одного из лидеров средств виртуализации. На втором этапе для предоставления ресурсов заданному числу ВМ, обеспечивающих выполнение соглашения об уровнях обслуживания (SLA), определяется необходимое число физических серверов. Реализация приложений ЦОД предполагает формирование совокупности ВМ, характеризующихся определенными ресурсными показателями. Локальный планировщик ресурсов определяет текущую загрузку оборудования ЦОД, распределяет ВМ на вычислительные устройства при условии обеспечения ими необходимой производительности, объемов памяти, пропускной способности системы коммутации, каналов связи и сетевых интерфейсов. Объединяя характеристики ВМ для каждого типа ресурсов, можно получить необходимые параметры оборудования ЦОД.

Распределение программных приложений $F = \{f_1, f_2, ..., f_n\}$ по $k \in K$ ВМ осуществляется с учетом ограниченных объемов ресурсов ЦОД. При этом необходимо определить рациональное число задействованных ВМ и состав распределенных на них программных приложений с учетом следующих ограничений:

– приложение может быть реализовано только на одной из ВМ:

$$\sum_n a_{jn} \le 1, \qquad (1)$$

где

$$a_{jn} = \begin{cases} 1, \text{ если приложение} \\ \text{выполняется на } n \text{ ВМ;} \\ 0, \text{ в противном случае,} \end{cases}$$

$$j = \overline{1,n}, \; n = \overline{1,k};$$

– назначенные на ВМ приложения запрашивают суммарные объемы ресурсов не больше имеющихся в наличии:

$$\sum_n a_{jn} C_j \le C_n, \quad \sum_n a_{jn} m_j \le M_n, \qquad (2)$$

где C_n, M_n – производительность и память j ВМ;

– реализуется условие неделимости приложений для всех временных интервалов t_n их выполнения:

$$\sum_{t=0}^{t_n} \sum_n a_{jn} = t_j \sum_n a_{jn}. \qquad (3)$$

Данная задача относится к классу NP-полных. Трудоемкость ее решения при полном переборе составляет k^n операций. Для приближенного ее решения с целью сокращения времени расчета можно использовать приведенный ниже жадный эвристический алгоритм [12].

1. Упорядочиваем программные приложения $j \in r$ в порядке убывания их запросов на объемы ресурсов ВМ.

2. Ранжируем ВМ $k \in K$ в порядке возрастания их производительности. При этом производительность измеряется в ECU (EC2 Compute Unit). Один ECU соответствует 1–1,2 ГГц процессора AMD Opteron или Intel Xeon.

3. Для каждого очередного приложения $j(k)$ на каждом временном интервале t_n подбираем ВМ, способную реализовать соответствующее приложение.

4. Если ВМ не удается подобрать, то приложение не будет выполнено. В противном случае на ВМ фиксируются требуемые объемы ресурсов. Данный алгоритм имеет полиномиальную сложность и обеспечивает близкое к оптимальным распределение [15].

ЦОД содержит определенное множество серверов приложений S_i, $i = \overline{(1, L)}$ и файл-серверов H_j, $j = \overline{(1, M)}$, необходимо распределить по ним множество BM_k, $k = \overline{(1, N)}$, при соблюдении следующих ограничений:

– каждая ВМ может быть размещена только на одном сервере

$$\sum_{i=1}^{L}\sum_{j=1}^{M}\sum_{k=1}^{N} x_{ijk} = 1, \qquad (4)$$

где

$$x_{ijk} = \begin{cases} 1, \text{ если } BM_k \text{ располагается} \\ \text{на } S_i \text{ или } H_j; \\ 0, \text{ в противном случае;} \end{cases}$$

– требования ВМ, предъявляемые к ресурсу оперативной памяти RAM физических серверов, должны удовлетворять ограничению

$$\sum_{i=1}^{L}\sum_{j=1}^{M}\sum_{k=1}^{N} RAM_k x_{ijk} < RAM_{ij}. \qquad (5)$$

Используемые ресурсы всех ВМ не должны быть больше ресурсов серверов;

– ограничения на производительность серверов

$$\sum_{i=1}^{L}\sum_{j=1}^{M}\sum_{k=1}^{N} CPU_k x_{ijk} < CPU_j; \qquad (6)$$

– ограничение на объем памяти дисков

$$\sum_{i=1}^{L}\sum_{j=1}^{M}\sum_{k=1}^{N} S_k x_{ijk} < S_{ij}; \qquad (7)$$

– ограничение на производительность системы ввода-вывода данных

$$\sum_{i=1}^{L}\sum_{j=1}^{M}\sum_{k=1}^{N} IOPS_k x_{ijk} < IOPS_{ij}. \qquad (8)$$

Инструментальная платформа Windows Performance Monitor позволяет осуществлять оценку вычислительной нагрузки оборудования ЦОД. Оптимальным соотношением эффективности использования ресурсов и запасом ресурсов на изменение нагрузки считают уровень 70–80 % [14].

Физические серверы и системы хранения данных соединены между собой системой коммутации. Поэтому при распределении ресурсов физических серверов необходимо учитывать основные параметры данной системы и виртуальных каналов связи между ВМ и системой хранения данных.

Суммарная пропускная способность всех каналов связи и системы коммутации должны удовлетворять следующим условиям:

$$\sum_{i=1}^{n}\sum_{l=1}^{L} r_i l < \Pi, \quad \sum_{i=1}^{n}\sum_{l=1}^{L} r_i l < K, \qquad (9)$$

где $r_i l$ – требуемая пропускная способность i-й ВМ при использовании l виртуальных каналов; П и K – суммарная пропускная способность физических каналов и системы коммутации.

В качестве критериев оптимальности будем использовать показатели:

– наиболее полной загрузки оборудования ЦОД

$$K_1 = \sum_{i=1}^{L}\sum_{j=1}^{M}\sum_{k=1}^{N} CPU_k x_{ijk} \cdot RAM_k x_{ijk}; \qquad (10)$$

– максимальной производительности системы ввода-вывода

$$K_2 = \sum_{i=1}^{L}\sum_{j=1}^{M}\sum_{k=1}^{N} IOPS_k x_{ijk}; \qquad (11)$$

– максимальной производительности системы коммутации и передачи данных

$$K_3 = \sum_{i=1}^{n}\sum_{l=1}^{L} r_i l. \qquad (12)$$

Задача оптимального распределения ресурсов ВМ является NP-трудной, поэтому для ее решения необходимо воспользоваться не полиномиальными алгоритмами офлайн-, или онлайн-размещения [17]. Подобные подходы обеспечивают приемлемую точность распределения относительно обладающих большой сложностью полиномиальных алгоритмов. Наиболее часто применяемые алгоритмы решения подобных задач приведены в таблице 1.

Таблица 1. Неполиномиальные алгоритмы распределения

Алгоритмы	NF	FF	BF	NED	FED	BED
Точность алгоритма	2	1,7	1,7	1,691	1,222	1,222

Например, особенностью приближенного алгоритма «в первый подходящий в порядке убывания» FFD [4] является применение упорядочения ВМ в порядке уменьшения требуемых ими ресурсов и их поочередный перенос на наименее занятые серверы. Данный алгоритм решает задачу размещения с точностью, не более чем на 22 % отличающуюся от оптимального. Ни один из известных приближенных алгоритмов не обеспечивает существенно лучшего результата [4].

Применение подобных алгоритмов целесообразно при загрузке ресурсов ЦОД до 50 %. В случае более высокой загрузки данные алгоритмы неприемлемы. Это вызывает необходимость разработки эвристических алгоритмов. Предлагаемый в данной работе алгоритм производит последовательную обработку запросов на размещение ВМ. При этом производится их ранжирование по убыванию запрашиваемых ресурсов. Внешний планировщик ЦОД выделяет каждой ВМ максимальное значение ресурсов CPU, RAM, disk, IOPS и т. д. Определяется физический ресурс с минимальной остаточной суммой его параметров, и осуществляется размещение на нем ВМ в соответствии со следующей схемой.

1. Производится ранжирование ВМ по показателю [4] $V_i = \sum_{j=1}^{K} C_j V_j^i$, где V_j^i – требование BM_i к ресурсу j, $j = \overline{1,K}$, C_j – коэффициент значимости ресурса.

2. В соответствии с исходной схемой выбирается ВМ с наибольшими требованиями к ресурсам и размещается на физическом элементе с минимальным ресурсом. Наиболее целесообразно использование при этом итерационного алгоритма, основанного на подходах динамического программирования.

Например, итерационный жадный алгоритм работает следующим образом [14]. В качестве входных процессов a_i множества $S = \{a_1, a_2, \ldots, a_n\}$ выступают конечные моменты f_i их реализации. Последовательно выбранные процессы объединяются процедурой Greedy Activity Selector (s, f) в множество A, в котором f_i является максимальным временем окончания всех процессов $f_i = \max\{f_k : a_k \in A\}$. Greedy Activity Selector (s, f):

– $n \leftarrow length[s]$;
– $A \leftarrow \{a_1\}$;
– $i \leftarrow 1$;
– *for* $m \leftarrow 2$ *to* n;
– *do if* $S_m \geq f_i$;
– *then* $A \leftarrow A \cup \{a_m\}$;
– $i \leftarrow m$;
– *return A.*

3. Внешний планировщик ЦОД осуществляет оценку производительности размещенной на сервере ВМ.

4. Если производительность ВМ ниже установленного уровня, внешний планировщик перемещает ее на физический элемент с более высоким уровнем ресурсов и переходит на шаг 3.

5. Если производительность ВМ выше установленного уровня, внешний планировщик перемещает ее на физический элемент с более низким уровнем ресурсов и переходит на шаг 3.

6. В случае невозможности распределения ВМ осуществляется переход к процедуре ограниченного перебора. Глубина перебора, определяющая максимальное число серверов, для которых назначается распределение, должна обеспечивать требуемый баланс между качеством сервиса и временем распределения ВМ, то есть находить приемлемое решение за допустимое время. Качество сервиса можно оценить вероятностью блокировки P_δ запросов на обслуживание, иначе вероятностью отсутствия допустимых серверов в единицу времени. Для этого можно использовать формулу Эрланга

$$P_\delta = \frac{a^n / n!}{\sum_{k=0}^{n} a^k / k!}, \qquad (13)$$

где P_δ – вероятность блокировки; n – число серверов; a – интенсивность запросов.

Значение данной вероятности определяется рекуррентным способом [8]

$$P_\delta = B(n, a) = \frac{B(n-1, a)}{B(n-a) + n/a}, \qquad (14)$$

где $n = 1, 2, \ldots$, $B(0, a) = 1$. В случае достаточно больших значений n и a можно уменьшить время вычисления P_δ путем задания разницы $P_\delta(i) - P_\delta(i-1) < \delta$ и остановки выполнения процедуры перестановки при ее достижении [10].

Таблица 2. Процент использования хостов

Число хостов	1	2	3	4	5	6	7	8	9	10
% использования хостов до размещения	85	63	100	81	42	37	34	45	30	40
% использования хостов после размещения	73	75	70	75	–	–	–	–	–	–

Таблица 3. Процент использования оперативной памяти

Число хостов	1	2	3	4	5	6	7	8	9	10
% использования памяти до размещения	60	65	70	45	27	25	20	32	40	43
% использования памяти после размещения	80	65	40	70	–	–	–	–	–	–

Таблица 4. Время работы алгоритма

Число ВМ	5	10	15	20	25	30	35	40
Время размещения, с	12	19	25	31	50	75	85	90

Значение $P_{\delta}(i) - P_{\delta}(i-1) < \delta$ определяет здесь точность вычислений.

7. Если ресурс не найден, то происходит размещение следующей ВМ, требование к уровню ресурсов предыдущей ВМ понижается, и она становится в общую упорядоченную очередь.

8. Если ресурс найден, то ВМ удаляется из очереди.

9. Выбор наилучшего по времени размещения при соблюдении всех ограничений.

Модельный эксперимент

Имитационное моделирование предложенного метода виртуализации проводилось для кластера, реализованного на базе платформы Huawei 2488, процессор Intel(R) Xeon(R) Gold 6154, 512 Gb оперативной памяти, содержащего 10 хост и 40 ВМ. Уровень запрашиваемых ресурсов ВМ равномерно распределен в интервале 20–80 % уровня ресурсов сервера. Соотношение процессорного времени и оперативной памяти распределено равномерно. На пропускную способность системы коммутации наложены ограничения до 1 Гб/с. Длительность перемещения ВМ задавалась нормальным законом распределения.

В таблицах 2–4 приведены результаты имитационного моделирования процесса размещения ВМ на вычислительном кластере ЦОД.

Из результатов моделирования следует, что предлагаемый алгоритм обеспечивает допустимые показатели распределения при количестве ВМ, не превышающем 20. С увеличением количества ВМ время распределения растет по экспоненциальному закону. Реализация представленного алгоритма снижает количество используемых хост ЦОД на 60 % при числе размещаемых ВМ, не превышающем 20.

Заключение

В работе рассматривается актуальная задача построения механизма распределения виртуальных машин по физическим элементам центра обработки данных. Предложена методика решения поставленной задачи, сочетающая эвристический жадный алгоритм с процедурой ограниченного перебора. Показано, что решение данной задачи можно провести в два этапа: применить алгоритм оптимального распределение программных приложений по виртуальным машинам и реализовать распределение виртуальных машин по физическим узлам кластера ЦОД с учетом статистики заявок.

Разработана имитационная модель алгоритма функционирования системы управления виртуализацией программно-конфигурируемой сети (свидетельство о государственной регистрации программы для ЭВМ № 2019664647 от 11.11.2019), проведено ее экспериментальное исследование с целью получения и последующего анализа таких характеристик как процент используемых ресурсов хост до реализации предложенной процедуры распределения и после ее реализации, а также прогнозируемое время работы алгоритма распределения. Использование предлагаемой методики позволит обоснованно выполнять размещение

программных приложений по ВМ корпоративного ЦОД, выбирать состав виртуальных машин и решать задачи рационального их размещения по физическим серверам ЦОД.

Финансирование

Данные исследования выполнены при финансовой поддержке Российского фонда фундаментальных исследований (РФФИ), грант № 19-07-00856\20.

Литература

1. Hewitt C. ORGs for scalable, robust, privacy-friendly client cloud computing // C. 46. The Private Cloud Report. 2010. № 3. 42 p.
2. Cloud computing and grid computing 360-degree compared / I. Foster [et al.] // Grid Computing Environments Workshop. Chicago, 2008. P. 1–10.
3. Cloud services for the fermilab scientific stakeholders / S. Timm [et al.] // J. Phys. Conf. Ser. 2015. Vol. 664. № 2. P. 022039.
4. JINR cloud infrastructure evolution / A.V. Baranov [et al.] // Physics of Particles and Nuclei Letters. 2016. Vol. 13. № 5. P. 672–675.
5. Feller E., Rilling L., Morin C. Energy-aware ant colony based workload placement in clouds // Proceedings of the 12th IEEE/ACM International Conference on Grid Computing. Lyon, France. 2011. P. 00626042.
6. Farahnakian F., Liljeberg P., Plosila J. LiRCUP: Linear regression based CPU usage prediction algorithm for live migration of virtual machines in data centers // 39th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA). 2013. P. 357–364.
7. Beloglazov A., Buyya R. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers // Concurrency and Computation: Practice and Experience (CCPE). 2012. Vol. 24 (13). P. 1397–1420.
8. Mastroianni C., Meo M., Papuzzo G. Probabilistic consolidation of virtual machines in self-organizing cloud data centers // IEEE Transaction of Cloud Computing. 2013. Vol. 1. P. 215–228.
9. Mosa A., Paton N.W. Optimizing virtual machine placement for energy and SLA in clouds using utility functions // Journal of Cloud Computing: Advances, Systems and Applications. 2016. Vol. 5. №. 1. P. 17.
10. Monil M.A.H., Rahman R.M. VM consolidation approach based on heuristics, fuzzy logic, and migration control // Journal of Cloud Computing: Advances, Systems and Applications. 2016. Vol. 5. № 1. P. 8.
11. Guenter B., Jain N., Williams C. Managing cost, performance, and reliability tradeoffs for energy-aware server provisioning // Proc. of the 30th Annual IEEE Intl. Conf. on Computer Communications (INFOCOM). 2011. P. 1332–1340.
12. Balashov N., Baranov A., Korenkov V. Optimization of over-provisioned clouds // Physics of Particles and Nuclei Letters. 2016. Vol. 13. № 5. P. 609–612.
13. McNab A., Stagni F., Luzzi C. LHCb experience with running jobs in virtual machines // J. Phys. Conf. Ser. 2015. Vol. 664. № 2. P. 022030.
14. Computing Center of the Institute of High Energy Physics (IHEP-CC) «VCondor – virtual computing resource pool manager based on HTCondor». URL: https://github.com/hep-gnu/VCondor (дата обращения: 17.06.2020).
15. McNab A., Love P., MacMahon E. Managing virtual machines with Vac and Vcycle // J. Phys. Conf. Ser. 2015. Vol. 664. № 2. P. 022031.
16. Feller E., Rilling L., Morin C. Snooze: A scalable and autonomic virtual machine management framework for private clouds // Proceedings of the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid). 2012. P. 482–489.
17. Beloglazov R.B. OpenStack neat: A framework for dynamic and energy-efficient consolidation of virtual machines in OpenStack clouds // Concurrency and Computation: Practice and Experience (CCPE). 2015. Vol. 27. № 5. P. 1310–1333.
18. Orgerie A.-C., Lef`ever L. When clouds become green: The green open cloud architecture // International Conference on Parallel Computing (ParCo). 2009. P. 228–237.
19. Ward J.S., Barker A. Observing the clouds: A survey and taxonomy of cloud monitoring // Journal of Cloud Computing: Advances, Systems and Applications. 2014. Vol. 3. № 1. P. 24.

Поступило 02.09.2020

Мочалов Валерий Петрович, д.т.н., профессор кафедры инфокоммуникаций (ИК) Северо-Кавказского федерального университета (СКФУ). 355028, Российская Федерация, Ставропольский край, г. Ставрополь, пр. Кулакова, 2 (корп. 9). Тел. +7 865 295-69-97. E-mail: mochalov.valery2015@yandex.ru

Линец Геннадий Иванович, д.т.н., заведующий кафедрой ИК СКФУ. 355028, Российская Федерация, Ставропольский край, г. Ставрополь, пр. Кулакова, 2 (корп. 9). Тел. +7 865 295-69-97. E-mail: kbytw@mail.ru

Палканов Илья Сергеевич, аспирант кафедры ИК СКФУ. 355028, Российская Федерация, Ставропольский край, г. Ставрополь, пр. Кулакова, 2 (корп. 9). Тел. +7 865 295-69-97. E-mail: ilya0693@yandex.ru

VIRTUALIZATION OF SERVER INFRASTRUCTURE AT THE CORPORATE DATA CENTERS

Mochalov V.P., Linets G.I., Palkanov I.S.
North-Caucasus Federal University, Stavropol, Russian Federation
E-mail: kbytw@mail.ru

The object of the research is the cluster systems of data centers containing a certain set of application servers, a servers file, data storage systems, an input-output system connected by a switching system and communication channels. The goal of the work is to increase the efficiency of using virtualized cluster systems by developing and applying a rational method for distributing virtual machines among the physical elements of the data centers. The proposed method is based on iterative greedy algorithm and a limited search procedure allowing to increase the performance of virtualized system due to rational data distribution. A simulation functioning model of the virtualized system was developed, its experimental study was carried out. The presented approaches provide determination of the required number of cluster nodes of the virtualized system with unpredictable load intensity and unpredictable level of resources requested by virtual machines.

Keywords: *data centers, virtual machines, physical servers, resource allocation algorithms, greedy algorithm, limited search*

DOI: 10.18469/ikt.2020.18.3.07

Mochalov Valeriy Petrovich, North-Caucasus Federal University, 2 (building 9), Kulakov Avenu, Stavropol, Stavropol Territory, 355028, Russian Federation; Professor of Infocommunication Department, Doctor of Technical Science. Tel. +7 865 295-69-97. E-mail: mochalov.valery2015@yandex.ru

Linets Gennadiy Ivanovich, North-Caucasus Federal University, 2 (building 9), Kulakov Avenu, Stavropol, Stavropol Territory, 355028, Russian Federation; Head of Infocommunication Department, Doctor of Technical Science. Tel. +7 865 295-69-97. E-mail: kbytw@mail.ru

Palkanov Ilya Sergeevich, North-Caucasus Federal University, 2 (building 9), Kulakov Avenu, Stavropol, Stavropol Territory, 355028, Russian Federation; PhD Student of Infocommunication Department. Tel. +7 865 295-69-97. E-mail: ilya0693@yandex.ru

References

1. Hewitt C. ORGs for scalable, robust, privacy-friendly client cloud computing. *C. 46. The Private Cloud Report*, 2010, no. 3, 42 p.
2. Foster I. et al. Cloud computing and grid computing 360-degree compared. *Grid Computing Environments Workshop*, Chicago, 2008, pp. 1–10.
3. Timm S. et al. Cloud services for the fermilab scientific stakeholders. *J. Phys. Conf. Ser*., 2015, vol. 664, no. 2, p. 022039.
4. Baranov A.V. et al. JINR cloud infrastructure evolution. *Physics of Particles and Nuclei Letters*, 2016, vol. 13, no. 5, pp. 672–675.

5. Feller E., Rilling L., Morin C. Energy-aware ant colony based workload placement in clouds. *Proceedings of the 12th IEEE/ACM International Conference on Grid Computing*, Lyon, France, 2011, p. 00626042.
6. Farahnakian F., Liljeberg P., Plosila J. LiRCUP: Linear regression based CPU usage prediction algorithm for live migration of virtual machines in data centers. *39th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA)*, 2013, pp. 357–364.
7. Beloglazov A., Buyya R. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. *Concurrency and Computation: Practice and Experience (CCPE)*, 2012, vol. 24 (13), pp. 1397–1420.
8. Mastroianni C., Meo M., Papuzzo G. Probabilistic consolidation of virtual machines in self-organizing cloud data centers. *IEEE Transaction of Cloud Computing*, 2013, vol. 1, pp. 215–228.
9. Mosa A., Paton N.W. Optimizing virtual machine placement for energy and SLA in clouds using utility functions. *Journal of Cloud Computing: Advances, Systems and Applications*, 2016, vol. 5, no. 1, p. 17.
10. Monil M.A.H., Rahman R.M. VM consolidation approach based on heuristics, fuzzy logic, and migration control. *Journal of Cloud Computing: Advances, Systems and Applications*, 2016, vol. 5, no. 1, p. 8.
11. Guenter B., Jain N., Williams C. Managing cost, performance, and reliability tradeoffs for energy-aware server provisioning. *Proc. of the 30th Annual IEEE Intl. Conf. on Computer Communications (INFOCOM)*, 2011, pp. 1332–1340.
12. Balashov N., Baranov A., Korenkov V. Optimization of over-provisioned clouds. *Physics of Particles and Nuclei Letters*, 2016, vol. 13, no. 5, pp. 609–612.
13. McNab A., Stagni F., Luzzi C. LHCb experience with running jobs in virtual machines. *J. Phys. Conf. Ser*., 2015, vol. 664, no. 2, p. 022030.
14. *Computing Center of the Institute of High Energy Physics (IHEP-CC) «VCondor – virtual computing resource pool manager based on HTCondor»*. URL: https://github.com/hep-gnu/VCondor (accessed 17.06.2020).
15. McNab A., Love P., MacMahon E. Managing virtual machines with Vac and Vcycle. *J. Phys. Conf. Ser*., 2015, vol. 664, no. 2, p. 022031.
16. Feller E., Rilling L., Morin C. Snooze: A scalable and autonomic virtual machine management framework for private clouds. *Proceedings of the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, 2012, pp. 482–489.
17. Beloglazov R.B. OpenStack neat: A framework for dynamic and energy-efficient consolidation of virtual machines in OpenStack clouds. *Concurrency and Computation: Practice and Experience (CCPE)*, 2015, vol. 27, no. 5, pp. 1310–1333.
18. Orgerie A.-C., Lef`ever L. When clouds become green: The green open cloud architecture. *International Conference on Parallel Computing (ParCo)*, 2009, pp. 228–237.
19. Ward J.S., Barker A. Observing the clouds: A survey and taxonomy of cloud monitoring. *Journal of Cloud Computing: Advances, Systems and Applications*, 2014, vol. 3, no. 1, p. 24.

Received 02.09.2020