Литература

1. Nurmi, D. QBETS: Queue Bounds Estimation from Time Series [Текст] / D. Nurmi, J. Brevik, R. Wolski // JSSPP 2007. LNCS, vol. 4942 Springer, Heidelberg. – 2008. – С. 76 - 101.
2. Brevik, J. Predicting Bounds on Queuing Delay for Batch-Scheduled Parallel Machines [Текст]/ J. Brevik, D. Nurmi, R. Wolski // PPoPP 2006: Proceedings of the Eleventh ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. – 2006. – С. 110 - 118.
3. Fernandez-Rodriguez, F. Nearest-Neighbour Predictions in Foreign Exchange Markets / F. Fernandez-Rodriguez, S. Sosvilla-Rivero, J. Andrada-Felix // Fundacion de Estudios de Economia Aplicada [Электронный ресурс]. – 2002, No.5. 36 p. URL: http://documentos.fedea.net/pubs/dt/2002/dt-2002-05.pdf (дата обращения 21.02.2016)
4. Чучуева, И.А. Модель прогнозирования временных рядов по выборке максимального подобия [Текст] / И.А. Чучуева // Дисс. на соиск. уч. ст. к. т. н., Москва. – 2012. – 155 с.
5. Артамонов, Ю.С. Применение облачного сервиса Templet Web при проведении лабораторных практикумов на суперкомпьютере «Сергей Королев» [Текст] / Ю.С. Артамонов, С.В. Востокин // X Международная научно-практическая конференция «Современные информационные технологии и ИТ-образование», МГУ, Москва, 2015. Том 2. – С. 409 - 414.
6. Лукашин, Ю.П. Адаптивные методы краткосрочного прогнозирования временных рядов [Текст] / Ю.П. Лукашин – Москва: Финансы и статистика, 2003 – 415 с.

И.П. Болодурина, Д.И. Парфёнов

СТРУКТУРНАЯ МОДЕЛЬ ПРОГРАММНО-УПРАВЛЯЕМОЙ ИНФРАСТРУКТУРЫ ВИРТУАЛЬНОГО ЦОД

(Оренбургский государственный университет)

В настоящее время, проблема эффективного использования вычислительных ресурсов в центрах обработки данных (ЦОД) является актуальной задачей. Современные коммуникационные технологии создали среду для многих критически важных бизнес-приложений и сервисов на основе облачных вычислений. В этой статье мы остановимся на вопросе моделирования инфраструктуры виртуальных центров обработки данных. В основе современных виртуальных ЦОД используется программно-конфигурируемая инфраструктура (SDI), на базе которой разворачиваются облачные приложения и сервисы по схемам IaaS и PaaS с использованием программно-конфигурируемой сети (SDN) [1-3]. Модель будет использована в дальнейшем для создания эффективных алгоритмов управления виртуальными ЦОД, включая алгоритмы планирования работы виртуальных машин, облачных приложений и сервисов, а также реконфигурации маршрутов сетевого трафика.

На рисунке 1 представлены уровневая схема программно-управляемой инфраструктуры виртуального ЦОД.

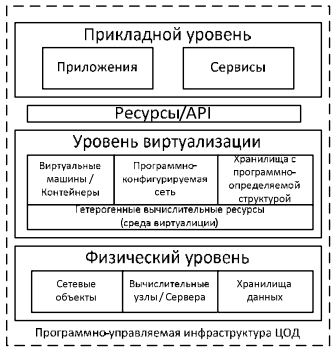

Рис. 1. Уровневая схема программно-управляемой инфраструктуры виртуального ЦОД

Основой структурной модели является программно-конфигурируемые сеть, которая может быть представлена в виде взвешенного ориентированного мультиграфа:

$$SDN = (Nodes, Links), \quad (1)$$

где $Nodes = \{Node_i\}_{i=\overline{1,N}}$ – множество сетевых устройств (узлов / серверов / и т.д.).; $Links = \{Link_j\}_{j=\overline{1,M}}$ – множество дуг, представляющих собой сетевое соединение. Каждое сетевое устройство характеризуется следующим кортежем:

$$Node_i = (L_i, P, C, M, T) \quad (2)$$

где L_i – множество дуг, исходящих из вершины; $P : L_i \to Z^+ \cup \{0\}$ – функция, которая характеризует текущую задержку для каждой дуги; $C : L_i \to Z^+ \cup \{0\}$ – пропускная способность; $M : L_i \to Z^+ \cup \{0\}$ – максимальная пропускная способность; $T \in \{host, switch\}$ – тип устройства.

Структурная модель программно-конфигурируемые инфраструктуры ЦОД может быть определена как направленный мультиграф:

$$SDI = (Seg, Connect(t)), \quad (3)$$

где вершины графа $Seg = \{Seg_1, Seg_2, ..., Seg_s\}$ – множество сетевых сегментов ЦОД; дуги графа $Connect(t) = \{(Seg_i, Seg_j)\}$ – направленная связь между сегментами.

Сегмент $Seg_k \in Seg$ программно-конфигурируемые инфраструктуры ЦОД представим в виде взвешенного неориентированного мультиграфа:

$$Seg_k = (Devices_k, Links_k, Flows_k(t)). \quad (4)$$

Вершины графа являются разбиением множества:

$$Devices_k = Nodes_k \cup Switches_k \quad (5)$$

где $Nodes_k = \{Node_{k1}, Node_{k2}, ..., Node_{kn_k}\}$ – набор вычислительных узлов;

$Switches_k = \{Switch_{k1}, Switch_{k2}, ..., Switch_{km_k}\}$ – сетевые элементы;

Вычислительные узлы $Node_{ki} \in Nodes_k$ включают в себя разбиение на множества:

$$Node_{ki} = VM_k \cup C_k \cup GW_k \cup B_k \cup Stg_k \quad (6)$$

где $VM_k = \{VM_{k1}, VM_{k2}, ..., VM_{kz_k}\}$ – виртуальные машины;

$C_k = \{C_{k1}, C_{k2}, ..., C_{kz_k}\}$ – контролеры OpenFlow;

$GW_k = \{GW_{k1}, GW_{k2}, ..., GW_{kw_k}\}$ – шлюзы;

$B_k = \{B_{k1}, B_{k2}, ..., B_{kb_k}\}$ – балансиры OpenFlow;

$Stg_k = \{Stg_{k1}, Stg_{k2}, ..., Stg_{kh_k}\}$ – хранилища данных (NAS/SAN).

Ребра мультиграфа $Links_k = \{(p_{ki}, p_{kj})\}$ это двусторонний сетевые соединения между портами сетевого устройства.

Каждый вычислительный узел $Node_{ki} \in Nodes_k$ имеет следующие параметры и динамические характеристики:

$$Node_{ki} = (M_{ki}, D_{ki}, Core_{ki}, S_{ki}, m_{ki}(t), d_{ki}(t), s_{ki}^{node}(t)) \quad (7)$$

$M_{ki} \in N$ и $D_{ki} \in N$ – соответственно размер оперативной памяти (Мб) и емкость диска (Мб); $Core_{ki} \in N$ – число вычислительных ядер; $S_{ki} \in R^+$ – производительность ядер; $m_{ki}(t) \in [0,1]$ и $d_{ki}(t) \in [0,1]$ – время отклика; $s_{ki}^{node}(t) \in \{"online","offline"\}$ – состояние узла.

На каждом вычислительном узле есть очередь задач $Q_{kij}^{node}(t) = \{Q_{kije}^{node}(t)\}$. Она используются для обеспечения QoS в соответствии с минимальной гарантированной пропускной способностью и максимальной задержкой для данной сети.

Для размещения данных в программно-конфигурируемой инфраструктуре используется особый тип хранилища с программно-определяемой структурой (SDS) [4]. Новая парадигма хранения позволяет задавать тип и структуру размещаемых данных. Размещаемые в хранилище данные можно представить в виде структуры

$$Data = (TypeS, TypeD, RDisk), \quad (8)$$

где $TypeS \in N$ - способ размещения; $TypeD \in N$ - вид физического устройства; $RDisk \in N$ – физическое устройство хранения.

Каждый сегмент виртуального ЦОД Seg_k имеет хранилище $Stg_{ki} \in Stg_k$ со следующими параметрами и динамическими характеристиками:

$$Stg_{ki} = \left(MaxV_{ki}, P_{ki}^{stg}, Vol_{ki}(t), \overline{R}_{ki}(t), \overline{W}_{ki}(t), s_{ki}^{stg}(t)\right), \quad (9)$$

где $MaxV_{ki} \in N$ – максимальная емкость для хранения (Mb); $P_{ki}^{stg} = \{p_{kij}^{stg}\}_j$ – множество сетевых портов; $Vol_{ki}(t) \in N \cup \{0\}$ – доступный объем для хранения

в Мб; $\overline{R}_{ki}(t)$ и $\overline{W}_{ki}(t)$ – скорость чтения и записи; $s_{ki}^{stg}(t) \in \{"online","offline"\}$ – состояние.

Каждый сегмент виртуального ЦОД Seg_k имеет хранилище $Stg_{ki} \in Stg_k$ со следующими параметрами и динамическими характеристиками:

$$Stg_{ki} = \left(MaxV_{ki}, P_{ki}^{stg}, Vol_{ki}(t), \overline{R}_{ki}(t), \overline{W}_{ki}(t), s_{ki}^{stg}(t)\right), \quad (10)$$

где $MaxV_{ki} \in N$ – максимальная емкость для хранения (Mb); $P_{ki}^{stg} = \{p_{kij}^{stg}\}_j$ – множество сетевых портов; $Vol_{ki}(t) \in N \cup \{0\}$ – доступный объем для хранения в Мб; $\overline{R}_{ki}(t)$ и $\overline{W}_{ki}(t)$ – скорость чтения и записи; $s_{ki}^{stg}(t) \in \{"online","offline"\}$ – состояние.

Хранилища с программно-определяемой структурой размещаются поверх физической инфраструктуры ЦОД в среде виртуализации. При этом его можно представить в виде структуры:

$$SoftStg = (VM, Lan, Stype, Dtype, RDisk(t), Vdisk), \quad (11)$$

где VM – виртуальная машина или сетевой контейнер, Lan – скорость доступа сетевого интерфейса, $Stype$ – поддерживаемый способ размещения данных, $Dtype$ вид физического устройства, на которых будет размещаться виртуальная машина; $RDisk(t)$ конкретное физическое устройство, содержащее виртуальную машину в данный момент времени t; ,$Vdisk$ – общий объем хранилища данных.

Кроме перечисленных элементов в состав программно-управляемой инфраструктуры виртуального ЦОД входят облачные приложения и сервисы, размещаемые в виртуальных машинах и контейнерах.

Облачное приложение представляет собой взвешенный ориентированный ацикличный граф зависимостей данных, в котором вершинами являются пользователи, виртуальные машины, объекты программно-конфигурируемого хранилища (БД, СХД) и прочие ресурсы, дуги – зависимости по данным между соответствующими вершинами. Каждая вершина – характеризуется предъявляемыми ресурсными требованиями (к количеству ядер, размерам оперативной и дисковой памяти, наличию специальных библиотек или оборудования на физических или виртуальных узлах, используемых для запуска процессов), количеством запускаемых процессов, оценкой времени выполнения. Оригинальность заключается в том, что дуга характеризуется типом доступа (доступ к файлу в СХД, к локальному файлу, к распределенной БД, к сервису данных), оценкой объемов передаваемых данных, требованиями к QoS.

Облачный сервис, как и облачное приложение, описывается в виде ориентированного графа зависимостей по данным, отличие заключается в том, что с точки зрения пользователя облачный сервис представляет собой закрытую систему. Также все его приложения распределены между виртуальными машинами или физическими серверами предустановленного набора, новые их экземпляры масштабируются динамически в зависимости от количества поступающих запросов на выполнение функции конкретного сервиса со стороны облачного

приложения, конечного пользователя или других облачных сервисов. Новизна последних двух моделей заключается в одновременном описании подзадач и источников данных разного типа с указанием направления и способа передачи данных.

Разработанная структураная модель позволяет описать основные элементы программно-управляемой инфраструктуры виртуального ЦОД.

Исследования выполнены при поддержке РФФИ (проекты № 16-37-60086 и № 16-07-01004).

Литература

1. Bolodurina I., Parfenov D., Shukhman A. Approach to the effective controlling cloud computing resources in data centers for providing multimedia services // Proceedings of International Siberian Conference on Control and Communications (SIBCON), Omsk, Russia, 2015. pp.:1-6,DOI: 10.1109/SIBCON.2015.7147170

2. Полежаев П.Н., Ушаков Ю.А., Шухман А.Е. Система управления ресурсами для высокопроизводительных вычислений, основанная на использовании программно-конфигурируемой сети // "Системы управления и информационные технологии: научно-технический журнал", 2013. - №4(54). - С. 65-69.

3. Парфёнов, Д.И. Управление потоками данных в высоконагруженных информационных системах, построенных на базе облачных вычислений / Болодурина И.П., Парфёнов Д.И. // Системы управления и информационные технологии: научно-технический журнал. – 2015. - № 1.1. – С. 111-118.

4. Mishra M., Sahoo A. “On theory of VM placement: Anomalies in existing methodologies and their mitigation using a novel vector based approach” Cloud Computing (CLOUD), IEEE International Conference, 2011, Washington: IEEE Press, p.275-282.

Э.И. Ватутин, С.Ю. Валяев, В.С. Титов

АНАЛИЗ РЕЗУЛЬТАТОВ ПРИМЕНЕНИЯ МЕТОДА СЛУЧАЙНОГО ПЕРЕБОРА ПРИ ПОСТРОЕНИИ РАЗБИЕНИЙ ГРАФ-СХЕМ ПАРАЛЛЕЛЬНЫХ АЛГОРИТМОВ В ЗАВИСИМОСТИ ОТ РАЗМЕРНОСТИ ЗАДАЧИ И СИЛЫ ОГРАНИЧЕНИЙ

(Юго-Западный государственный университет)

При проектировании цифровых устройств их структура традиционно делится на две части: операционную и управляющую. Операционная часть, как правило, включает в своем составе схемы, обеспечивающие обработку информации в соответствии с требованиями решаемой задачи, а управляющая обеспечивает координацию ее работы путем определения ее состояния в ходе получении вектора сигналов логических условий $X = \{x_1, x_2, ..., x_P\}$ и выдачи сигналов микроопераций $Y = \{y_1, y_2, ..., y_Q\}$ в соответствии с заданным алгоритмом

управления. Так, например, в качестве сигнала логического условия может служить единичное значение бинарного признака завершения работы одной из схем в составе операционной части [1], достижение счетчиком заданного значения и т.п.; управляющие сигналы обычно используются для фиксации значений в запоминающих элементах схемы (ОЗУ, триггерах и пр.). При проектировании относительно простых систем логического управления (СЛУ), ориентированных на реализацию небольших алгоритмов управления, производится разметка соответствующей граф-схемы, составление графа переходов и проектирование соответствующего управляющего автомата Мили или Мура [2]. Если же алгоритм логического управления $G = \langle A, V \rangle$, где A – множество его вершин, V – множество дуг, является комплексным и содержит большое количество вершин (как правило сотни), параллельные фрагменты, циклы различного вида (в том числе параллельные), то традиционный подход сталкивается с существенными трудностями. С целью их преодоления в [3–5] было предложено использование однородных многомодульных мультисистем, именуемых логическими мультиконтроллерами (ЛМК), включающих в своем составе определенное количество относительно простых логических контроллеров, каждый из которых может выполнять последовательный подалгоритм $A_i \subseteq A$, $\bigcup_{i=1}^{H} A_i = A$, $\forall i, j = \overline{1, H}, i \neq j: A_i \cap A_j = \varnothing$ ограниченной сложности, являющейся частью исходного комплексного алгоритма управления G. Каждый контроллер в составе ЛМК имеет память микропрограмм ограниченной емкости $W_{\max}$, в которой хранится соответствующий подалгоитм A_i, и заданное число ножек (или портов ввода/вывода) для приема сигналов логических условий $X_{\max}$ и выдачи сигналов микроопераций $Y_{\max}$. При обеспечении координации работы отдельных подалгоритмов, назначаемых на различные контроллеры в составе ЛМК, можно добиться выдачи логических условий, соответствующий заданному алгоритму управления G, т.е., другими словами, реализации СЛУ в базисе ЛМК. При этом возникает ряд задач дискретной комбинаторной оптимизации [6], связанных с разбиением заданного алгоритма управления на подалгоритмы (блоки разбиения) в соответствии с ограничениями базиса ЛМК [3–5], размещением полученных блоков в контроллерах, имеющих заданную топологию связей (например, матричную) [3], координацией параллельной работы модулей [3], оптимизацией структуры ЛМК [7] и пр.

В указанной работе рассматривается задача поиска субоптимального разбиения заданного алгоритма логического управления (ее подробное описание можно найти в монографиях [4–5]). При ее решении учитываются структурные и функциональные ограничения базиса ЛМК, к которым относятся соответственно емкость памяти контроллера $W_{\max}$, число ножек $X_{\max}$ и $Y_{\max}$ и требование отсутствия параллельных вершин в составе блоков разбиения (что вынуждает решать частную задачу классификации бинарных отношений вершин граф-схемы G [8]), и минимизируются значения множества показателей качества, к