ПОДХОД К РЕАЛИЗАЦИИ МОДЕЛИРОВАНИЯ ПРОИЗВОДИТЕЛЬНОСТИ ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ

APPROACH TO IMPLEMENTATION OF MODELING THE PERFORMANCE OF COMPUTER SYSTEM

A. Kalistratov
G. Afanasyev

Summary. The article describes an approach to a computer system performance modeling. Two methods of performance modeling for a virtualized environment are described. Questions of improvement of accuracy of such model by its consecutive adaptation during work are specified. The process of building a performance model is depicted. The authors describe the process of collecting data on the operation of processes and services and their representation as components of the model. The conclusion about possible development of this approach and ways of development is made.

Keywords: performance, performance measurement, performance modeling, virtual machines.

Калистратов Алексей Павлович
Аспирант, МГТУ им. Н. Э. Баумана
akalistratov@gmail.com
Афанасьев Геннадий Иванович
К.т.н., доцент, МГТУ им. Н. Э. Баумана
gaipcs@bmstu.ru

Аннотация. В статье описывается подход к моделированию производительности вычислительной системы. Приводится описание двух способов моделирования производительности для виртуализированной среды. Уточняются вопросы улучшения точности такой модели путем ее последовательной адаптации в процессе работы. Описывается процесс построения модели производительности. Авторы описывают процесс сбора данных по работе процессов и служб и их представление в виде компонентов модели. Делается вывод о возможном развитии данного подхода и путях развития.

Ключевые слова: производительность, измерение производительности, моделирование производительности, виртуальные машины.

Введение

Существует несколько подходов к решению задачи предсказания и моделирования производительности виртуальных машин. Однако, она зависит не только от количества выделенных под виртуальную машину системных ресурсов, но и от характера использования этих ресурсов системой-хостом. В случае параллельного использования этих ресурсов другими виртуальными машинами производительность будет отличаться. Таким образом, необходимо учитывать изменения в доступе к выделенным виртуальной машине ресурсам.

В данной работе авторы делают допущение, что параллельно могут использоваться только ресурсы процессора. Несмотря на то, что фактически это не так (параллельно могут использоваться ресурсы процессора, оперативной памяти, жесткого диска), это допущение делается для упрощения измерения влияния одного из компонентов на общую производительность системы.

Влияние параллельного использования процессорных ядер и оперативной памяти может быть отслежено напрямую с помощью программного обеспечения, загрузка этих ресурсов видна в гостевой операционной системе. Также на результат может влиять параллельное использование таких ресурсов как процессорный кэш и шина памяти, при этом мы не можем никак отследить загрузку этих узлов, т.к., они управляются без участия пользователя [1].

Стоит отметить, что необходимо учитывать также и особенности поддержки и реализации технологий виртуализации в различных гипервизорах или аппаратном обеспечении [2]. Это тоже может внести свой вклад в изменение значений производительности и/или работу с предоставленным объемом системных ресурсов. Однако, в данной статье влияние этого фактора не учитывается. По мнению авторов, на данном этапе изучения вопроса влияния возможных сценариев использования заданного количества системных ресурсов и их влияния на производительность разумно будет сконцентрироваться на изучении одного определенного гипервизора.

В данной работе рассматриваются некоторые вопросы, возникающие при решении задачи моделирования производительности вычислительной системы и некоторых проблем, возникающих в ходе решения данной задачи. Особенность моделирования в данном случае заключается в том, что стоит задача смоделировать производительность виртуализированной вычислительной системы, что заставляет учитывать не только «традиционные» факторы, влияющие на производительность, но также и некоторые особенности, связанные с применением виртуализации.

Подходы к моделированию производительности

Для того, чтобы перейти к вопросу моделирования производительности виртуальных машин, необходимо определить доступные способы моделирования производительности. Это может быть сделано следующим образом:

1) Оффлайн-моделирование. При таком подходе мы предполагаем, что производительность может быть заранее измерена на нескольких платформах хостовых систем, в различных конфигурациях оборудования и установок виртуальных машин. Также мы предполагаем, что вся нагрузка (как минимум на уровне виртуальной машины) может быть разделена на отдельные части (например, на процессы и службы) с целью измерения их влияния на поведение виртуальной машины. Таким образом, при измерениях и наблюдениях основной целью является настройка модели для предсказания производительности различных конфигураций системы в будущем.

2) Онлайн-моделирование. При этом подходе мы предполагаем, что нам по тем или иным причинам недоступно оффлайн-моделирование и факторы, влияющие на производительность виртуальных машин, должны быть оценены в режиме реального времени. В этом случае необходимо создать такую модель, которая бы отражала производительность виртуальных машин определенного типа для определенной платформы.

В дополнение к этому, необходимо учитывать и разделить следующие виды системных ресурсов, так как они могут быть использованы различным образом (как параллельно, так и отдельно друг от друга), что, как было определено в предыдущем разделе, значительно влияет на производительность:

1) Видимые ресурсы: системные ресурсы, показатели которых доступны для измерения с помощью средств операционной системы (ОС) или гипервизора. Например, загрузка ядра, объем занятой оперативной памяти, операции ввода-вывода жесткого диска.

2) Невидимые ресурсы: системные ресурсы, которые на данный момент недоступны для получения параметров использования с помощью операционной системы или гипервизора, например, занятое пространство в кэш-памяти или параметры конвейера ядра. Соответственно, кэш-память, занятая некой виртуальной машиной, не может быть отслежена и измерена гипервизором. Аналогичная ситуация с ресурсами конвейера при их использовании потоком в процессорном ядре. Однако, есть показатели производительности, напрямую зависящие от того, используется ли кэш или ядро параллельно, или нет, например, количество тактов за инструкцию или количество промахов в кэше на инструкцию. Их можно считать с помощью некоторых инструментов мониторинга системы [3].

Требования к модели

Предлагаемый подход к моделированию производительности виртуальных машин являет собой получение примерного расчета объема системных ресурсов платформы, которые будут в распоряжении каждой виртуальной машины на момент ее запуска в гипервизоре, а затем их распределение по компонентам-потребителям с получением результата их выполнения, включающего в себя запуск и работу системных служб.

Для онлайн-моделирования производительности важно поддерживать синхронизацию модели производительности с моделируемой системой. В противном случае, как только модель производительности системы построена, она может быстро устареть и, таким образом, больше не будет представлять реальную систему. Изменения конфигурации и развертывания распространены в современных корпоративных системных средах [4]. Например, прямо во время работы запускаются или изменяются службы, приложения, иногда даже конфигурации виртуальных машин. При изменении системы представления модели производительности также должны быть обновлены [5]. Следует реализовать привязку модели производительности к среде системы, т.е. в постоянной адаптации модели во время работы системы. Модель должна предоставлять актуальную и точную информацию о системе для обеспечения точных прогнозов производительности. В терминах моделей среды выполнения модель должна быть причинно-связным представлением системы. В этом случае она будет постоянно отражать структуру и поведение системы, относящееся к производительности. Для достижения такого зеркального отображения нужно следовать предложению о конвергенции прогнозирования производительности, то есть моделирования и прогнозирования как взаимосвязанных видов деятельности [6]. Во время выполнения компоненты системы реализуются и развертываются в целевой рабочей среде. Это позволяет получать репрезентативные оценки различных параметров модели с учетом реальной среды исполнения. Кроме того, для итеративного уточнения точности параметры модели можно непрерывно корректировать [7]. Также информация, относящаяся к производительности, может отслеживаться и описываться на уровне экземпляра компонента, а не только на уровне типа, что характерно для существующих образцов онлайн-моделирования производительности. Однако в процессе эксплуатации мы не имеем возможности проводить произвольные

эксперименты, так как система находится в производстве и используется реальными заказчиками, выдающими запросы.

Таким образом, наша задача сводится к интеграции моделей производительности на уровне архитектуры и системных сред. Интеграция осуществляется двумя способами:

- методом полуавтоматического извлечения информации об изменениях модели на основе данных мониторинга;
- методом автоматического сбора данных о потреблении ресурсов приложениями во время их запуска и завершения.

В каждом случае мы различаем статическую структурную информацию системной среды (например, типы используемых приложений) и динамические параметры (например, потребности приложений в ресурсах), которые отражаются в моделях. Как полуавтоматический, так и автоматический способы используют данные мониторинга в качестве входных данных для изменения исходных данных для модели производительности.

Помимо общего мониторинга функций, таких как контроль использования ресурсов, контрольная инфраструктура должна позволять отслеживать системные запросы и предоставлять средства для контроля затрат и контроля операций [8]. Также следует принимать во внимание то, что данные измерений используются для характеристики модели и мы должны хранить такие параметры, как, например, время отклика на системные запросы, количество занимаемой приложением оперативной памяти или же процессорное время. Необходимо не только запоминать собранные данные, но и аккумулировать временные ряды в виде функций вероятности, что позволит значительно сократить объем хранимых и обрабатываемых данных.

Системные запросы можно отслеживать с помощью метода, который мы обозначаем как трассировка пути вызова. Выполняемый системный запрос преобразуется в путь через граф потока управления, ребра которого являются базовыми блоками, т.е. ребро представляет часть кода в приложении только с одной точкой входа и только с одной точкой выхода [9]. Путь через график потока управления может быть представлен последовательностью ссылок на базовые блоки. Для простоты запросы с поведением разветвления игнорируются, такие запросы будут преобразовываться в дерево в графе потока управления. Мы предполагаем, что мы можем отслеживать и сохранять запросы в так называемые записи событий, определяемые как кортеж e = (l, t, s), где l ссылается на начало или конец базового блока, t является меткой времени и s идентифицирует системный запрос. Запись события указывает, что l был достигнут s в момент времени t. Для отслеживания отдельных системных запросов необходимо получить набор записей событий во время их выполнения. Затем набор собранных записей событий должен быть упорядочен, для чего он разбивается на классы эквивалентности [a]R в соответствии со следующим соотношением эквивалентности: пусть a = (l_1; t_1; s_1); b = (l_2; t_2; s_2) — записи событий, полученные с помощью мониторинга или иных способов. Тогда a относится к b, т.е. E., a ~ b, тогда и только тогда, когда $\Sigma_1 = \Sigma_2$.

Описания моделируются как случайные переменные, каждое из которых представлено вероятностным распределением по темпу выборки. Учитывая, что выборочные пространства времени отклика, количества итераций цикла и потребности в ресурсах бесконечны, а в случае времени отклика и потребностей в ресурсах также непрерывны, индивидуальная вероятность невозможна в случае, если имеется большое количество выборок. Вместо того, чтобы использовать функцию вероятности для характеристики случайной величины, необходимо аппроксимировать функцию плотности вероятности. Другими словами, учитывая набор измерительных выборок, необходимо построить гистограмму, представляющую эмпирическую функцию плотности. Количество ячеек гистограммы и их размеры должны быть выбраны для того, чтобы упростить представление распределения, обеспечивая при этом репрезентативную форму функции плотности. Существует множество подходов к построению гистограмм, мы различаем статические гистограммы, когда базовый набор выборок должен быть доступен в начале построения гистограммы, и динамические гистограммы, когда гистограмма может быть последовательно уточнена по мере появления новых образцов измерений [10].

Построение модели

Процесс построения модели представляется состоящим из четырех шагов:

1) Составляется аппаратная архитектура измеряемой системы (виртуальная машина, физическая машина).
2) Составляется программная архитектура для измерения производительности.
3) Из заранее накопленных данных составляются параметры модели производительности.
4) Полученная из компонентов модель итеративно корректируется до тех пор, пока не будет обеспечена приемлемая точность.

Методы извлечения параметров модели эквивалентны методам, которые используются для сбора параметров. Учитывая, что мы используем данные мониторинга, мы рассматриваем только те компоненты, которые

используются измеримо во время выполнения. Таким образом, игнорируются те компоненты, для которых отсутствуют данные мониторинга. Для выделения компонентов мы используем подход, называемый компонентизацией. Компонентизация — это процесс разбиения рассматриваемой архитектуры приложения на компоненты. Границы компонентов могут быть получены различными способами, например, заданы вручную архитектором системы или извлечены автоматически с помощью статического анализа кода. Степень детализации идентифицированных компонентов определяет степень детализации единиц, производительность которых должна быть охарактеризована, и, следовательно, степень детализации результирующей модели производительности. В контексте автоматического извлечения модели граница компонента задается как набор программных строительных блоков, рассматриваемых как единое целое. Например, это может быть набор классов или набор сервлетов Java. После определения границ компонентов связи между компонентами можно автоматически определять на основе данных мониторинга. Мы определяем контрольный поток между идентифицированными компонентами, используя трассировку пути вызова. Учитывая список путей вызовов и знания о границах компонентов, можно определить список эффективно используемых компонентов, а также их фактические записи (предоставляемые услуги) и выходы (необходимые услуги). После извлечения компонентов и соединений между ними необходимо извлечь абстракции поведения службы. Различают три уровня абстракции: черного ящика, грубая и точная. Каждый из уровней требует различной точности входных данных и своего подхода к их сбору.

Абстракция черного ящика захватывает представление поведения службы с точки зрения потребителя службы без каких-либо дополнительных сведений о поведении службы. Модель черного ящика может быть извлечена путем параметризации ее с измеренным временем отклика

Грубая абстракция фиксирует поведение компонента при наблюдении снаружи на границах компонента. Модель грубой абстракции может быть получена путем параметризации частоты внешних вызовов компонента и общих требований ресурса компонента. Таким образом, нам потребуется информация об общем потреблении ресурсов компонента, однако информация о внутреннем поведении компонента не рассматривается.

Для точной абстракции потребуется информация о потоке управления компонентами и ее процессах. Поток управления, относящийся к производительности, состоит из внутреннего ресурсоемкого поведения компонента и информации о том, как она использует внешние вызовы компонента. Разницу можно заметить в случае, если компонента вызывается один или десять раз в процессе выполнения приложения. Модель точной абстракции, которую мы стремимся извлечь, является абстракцией фактического контрольного потока. Действия, связанные с производительностью,— это внутренние вычислительные задачи и вызовы внешних компонент, следовательно, также циклы и ветви, в которых вызываются внешние компоненты. Набор путей вызова, полученных на предыдущем шаге, предоставляет информацию о том, как предоставляемая компонента связана с вызовами внешнего компонента.

Пусть множество XC-набор служб компоненты C, а Y — искомый набор служб, где $Y^C = y_1^C, \dots, y_n^C$.Затем обозначим системные вызовы компонент как функцию $G: x \rightarrow S_x$, где $x \in X, S_x \in \{(l_1, \dots, l_n) | n \in N, l_n \in Y\}$. G является отображением x на S_x и является собой набор последовательностей наблюдаемых внешних вызовов компоненты C.

Существует несколько подходов для определения вызовов служб для набора компонент, позволяющих определить влияние этого ПО на производительность. Путем отслеживания падения производительности одновременно запускаемых компонент можно определить однотипные службы и заложить основу для их оптимизации путем одновременного выполнения одной службы для нескольких компонент.

Заключение

Накапливая зависимости и функции вероятности для поведения компонент и служб, можно выделить области действия определенных переменных для модели производительности, основанной на влиянии запуска различных служб в пределах данных компонент. В случае, если переменная модели не имеет указанной области, мы полагаем, что данная переменная уникальна и начинаем накапливать для нее зависимости без учета предыдущего опыта. При этом, накопленные для в процессе всех наблюдений за одинаковыми службами данные могут использоваться для всех компонент, содержащих в себе эти службы, равно как и их вызовы. Этот процесс можно автоматизировать, что потребует кластерного анализа «сверху вниз» для накапливаемых значений переменных. Цель состоит в том, чтобы подобрать кластеризацию элементов так, чтобы значения, наблюдаемые в одном экземпляре компонента, могли быть назначены точно одному кластеру и один кластер для всех наблюдаемых значений будет соответствовать пустому набору областей для переменной модели.

Кроме того, в модель можно добавить перекрестные зависимости служб. В зависимости от того, может ли зави-

симость службы быть явно охарактеризована или должна быть просто помечена для эмпирической характеристики на основе наблюдений, должен быть настроен тип отношений, ее характеризующих. Автоматическое определение зависимостей соответствующих служб потребует анализа чувствительности всех задействованных параметров. Учитывая, что набор всех задействованных параметров будет содержать все входные параметры службы или компонента и все переменные для модели производительности, на данном этапе авторы полагают автоматизированное обнаружение нецелесообразным, т.к., накопление данных приведет к преждевременному их устареванию, а перед запуском системы накопить данные в процессе экспериментов будет невозможно.

Итоговой целью работы является автоматизация процесса построения моделей производительности путем наблюдения за поведением системы во время ее выполнения. При этом факторы, значительно влияющие на производительность, должны автоматически извлекаться с использованием данных мониторинга с минимальным вмешательством человека. Это позволит сделать процесс оценки производительности очень быстрым и точным.

ЛИТЕРАТУРА

1. Калистратов А.П., Афанасьев Г. И., Ревунков Г. И., Семкин П. С., Влияние распределения системных ресурсов на производительность виртуальных машин, Динамика сложных систем — XXI век. 2017. Т. 11. № 4. С. 46–50.
2. Горбачевская Елена Николаевна, Марфин Сергей Григорьевич Мониторинг виртуальной вычислительной системы // Вестник ВУиТ. 2010. № 16. URL: https://cyberleninka.ru/article/n/monitoring-virtualnoy-vychislitelnoy-sistemy (дата обращения: 10.04.2018).
3. Калистратов А.П., Федосеев Д. А., Измерение производительности виртуальных машин, 19-я Молодежная международная научно-техническая конференция «Наукоемкие технологии и интеллектуальные системы 2017».— г. Москва, 19 апреля 2017 г., НУК ИУ МГТУ им. Н. Э. Баумана. 387 с.
4. Saavedra-Barrera, Rafael Hector. CPU performance evaluation and execution time prediction using narrow spectrum benchmarking. Diss. University of California, Berkeley, 1992.
5. Venkataraman, Shivaram, et al. «Ernest: Efficient Performance Prediction for Large-Scale Advanced Analytics.» NSDI. 2016.
6. Wang, Kewen, and Mohammad Maifi Hasan Khan. «Performance prediction for apache spark platform.» High Performance Computing and Communications (HPCC), 2015 IEEE7th International Symposium on Cyberspace Safety and Security (CSS), 2015 IEEE12th International Conferen on Embedded Software and Systems (ICESS), 2015 IEEE17th International Conference on. IEEE, 2015.
7. Романчук, В. А., В. В. Лукашенко. «Моделирование структуры нейрокомпьютерного вычислительного кластера.» Вестник Пермского национального исследовательского политехнического университета. Электротехника, информационные технологии, системы управления 21 (2017).
8. Сорокин С.А., Чудинов С. М., Сорокин А. П., Болгова Е. В. Методы оценки производительности вычислительных комплексов // Научные ведомости Белгородского государственного университета. Серия: Экономика. Информатика. 2017. № 9 (258). URL: https://cyberleninka.ru/article/n/metody-otsenki-proizvoditelnosti-vychislitelnyh-kompleksov (дата обращения: 10.04.2018).
9. Felter, Wes, et al. «An updated performance comparison of virtual machines and linux containers.» Performance Analysis of Systems and Software (ISPASS), 2015 IEEE International Symposium On. IEEE, 2015.
10. Van Werkhoven, Ben, et al. «Performance models for CPU-GPU data transfers.» Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on. IEEE, 2014.

Журнал «Современная наука: актуальные проблемы теории и практики»