Оценка производительности программного обеспечения в виртуализованном окружении на основе атомарных тестов

П.А. Клеменков
parser@cs.msu.su
МГУ имени М.В. Ломоносова

Аннотация. В настоящее время все больше растет интерес к использованию платформ виртуализации (VMWare, XEN и др.) в различных сферах, включая консолидацию серверов, организацию хостинга и облачные вычисления. Производительность приложения в виртуальной машине может очень сильно отличаться от производительности вне виртуализованного окружения из-за взаимодействий с гипервизором и другими виртуальными машинами.

В этой статье описывается обобщенный подход к оценке требуемых программному обеспечению ресурсов при переносе его в виртуализованное окружение. Основной принцип предложенного подхода заключается в представлении сложной нагрузки в виде комбинации простых задач и замены этих простых задач на синтетические атомарные тесты. Оценка производительности атомарных тестов в среде виртуализации и вне нее позволяет определить накладные расходы на виртуализацию.

Ключевые слова: виртуализация; оценка производительности по; атомарные тесты; xen

1. *Введение*

Виртуализация и автоматизация – это основные составляющие современных вычислительных центров, позволяющие создавать более динамичную IT-инфраструктуру. Виртуализация помогает изолировать владельца вычислительных ресурсов от владельца программного обеспечения, позволяя стандартизировать и автоматизировать процессы настройки, управления и мониторинга. Скрывая детали распределения ресурсов от пользователей, виртуализация способствует оптимальному разделению вычислительных мощностей между приложениями, выполняющимися в разных виртуальных

машинах, благодаря возможности быстрого перепрофилирования ресурсов по запросу. Такие функции позволяют лучше удовлетворять потребности приложений и более эффективно отвечать на изменяющиеся требования бизнеса.

В таких условиях необходимыми становятся удобные и точные модели оценки производительности, которые позволяют консолидировать приложения и предоставлять необходимые для их выполнения ресурсы. Для того чтобы понять, какие нагрузки и на каких серверах можно объединять, требуется провести планирование мощностей и анализ существующей нагрузки. В простейшем случае можно собрать информацию о пиковых нагрузках, создаваемых множеством приложений, и оценить совместные требования к ресурсам как сумму пиковых значений. Однако такой подход может привести к выделению излишних мощностей, т.к. он не учитывает эффекты, возникающие при разделении ресурсов между взаимодополняющими программами.

Более перспективный и точный подход основывается на трассировке поведения приложения [1], [2]. Метод заключается в создании профиля использования ресурсов за некоторый период времени (обычно 3-6 месяцев). Затем этот профиль используется для планирования мощностей и распределения нагрузки при консолидации [3], [4]. Суть подхода заключается в том, что поведение приложения в будущем можно попытаться определить на основе наблюдений за создаваемой им нагрузкой в прошлом.

Однако планирование ресурсов при переходе в виртуализованное окружение усложняется возникновением дополнительных накладных расходов, вызванных взаимодействием с гипервизором. Эти накладные расходы зависят от типа виртуализации и конкретной реализации решения [5], [6], [7]. Зачастую дополнительные накладные расходы при использовании ЦПУ прямо пропорциональны объему выполняемого ввода-вывода [8], [9]. Существующие решения на основе трассировки имеют возможности масштабировать оценку на основе заданного множителя, чтобы учитывать различия в аппаратном обеспечении, однако такой подход не всегда применим при переходе в среду виртуализации, где объем накладных расходов оценить не так просто.

2. Оценка производительности на основе «микрозамеров»

Существует и другой подход к оценке производительности приложений при переносе их в виртуализованную среду [10]. Основу подхода составляют два процесса:

1. Проведение заданного набора микрозамеров для профилировки накладных расходов на виртуализацию данной платформы.
2. Создание модели, связывающей профиль потребления ресурсов вне среды виртуализации с профилем внутри среды, на основе

регрессионного алгоритма. Полученная модель позволяет предсказывать потребление ресурсов любым приложением на данной платформе.

Во время проведения каждого микрозамера собирается информация о потреблении ресурсов для построения профиля, используемого как обучающий набор для модели. Вне среды виртуализации собирается информация о метриках, связанных с потреблением ресурсов ЦПУ, сетевой активности и дисковом вводе-выводе. Далее, для нахождения закономерностей между потреблением ресурсов в среде виртуализации и вне нее, используются профили, собранные при проведении микрозамеров на «голом железе» и в виртуализованном окружении. Используя данные профили, формулируется система уравнений, выражающая потребление ресурсов ЦПУ как линейную комбинацию различных метрик:

$$\begin{aligned} U^1_{virt} &= c_0 + c_1 * M^1_1 + c_2 M^1_2 + \mathrm{K} + c_k * M^1_k \\ U^2_{virt} &= c_0 + c_1 * M^2_1 + c_2 M^2_2 + \mathrm{K} + c_k * M^2_k \\ &\mathrm{K} \end{aligned} \quad (1)$$

где

- M^j_i - это значение метрики M_i, измеренное во временной интервал j вне среды виртуализации
- U^j_{virt} - это мера потребления ресурсов ЦПУ в среде виртуализации, измеренная во временной интервал j

Пусть c^{virt}_0, c^{virt}_1, K, c^{virt}_k являются приближенным решением системы уравнений (1), тогда объем потребляемых ресурсов ЦПУ для произвольного приложения в виртуализованном окружении можно оценить как:

$$\hat{U}^j_{virt} = c^{virt}_0 + \sum_{i=1}^{k} M^j_i \cdot c^{virt}_i \quad (2)$$

Для получения приближенного решения c^{virt}_0, c^{virt}_1, K, c^{virt}_k можно применить один из известных регрессионных методов. Одним из самых известных является метод наименьших квадратов, который минимизирует ошибку:

$$e = \sqrt{\sum_j \left(\hat{U}^j_{virt} - U^j_{virt}\right)^2}$$

Набор коэффициентов $c^{virt}_0, c^{virt}_1, \mathrm{K}, c^{virt}_k$ и определяют модель, описывающую связь между потреблением ресурсов ЦПУ в среде виртуализации и потреблением ресурсов сервера вне нее.

Авторы метода утверждают, что данный подход достаточно полно характеризует различные накладные расходы на виртуализацию, а медиана ошибки предсказания модели на тестах RUBiS и TPC-W составляет менее 5%.

3. Ограничения метода «микрозамеров»

Описанный выше подход показывает достаточно хорошие результаты на избранных нагрузках. «Микрозамеры» предлагают решение задачи оценки производительности ПО в виртуализованном окружении в общем, но зачастую важнее делать предположения, основываясь на преобладающих характеристиках нагрузки. Очевидно, что некоторые приложения производят интенсивный ввод-вывод, некоторые большую часть времени исполняются в пространстве ядра, производя системные вызовы и т.д. Однако каждая программа представляет собой совокупность различных активностей. Эти активности воздействуют друг на друга и на производительность всего приложения. С другой стороны, одна и та же активность может вызывать различные события в разных гипервизорах. Именно поэтому так важно учитывать эти особенности при оценке производительности.

Другой проблемой метода «микрозамеров» является природа тестов. Каждый микрозамер по своей сути является синтетической нагрузкой. Поэтому выбор подходящего микрозамера является сложной задачей, которую трудно решить с удовлетворительной точностью.

4. Метод разделения сложной нагрузки на атомарные тесты

Описываемый мной подход основан на предположении о том, что некоторые типы сложных нагрузок могут быть представлены как совокупность более простых активностей. Пусть T - время выполнения некоторой нагрузки. Это может быть создание процесса, обработка запроса на ввод-вывод и т.д. T может быть выражена как линейная комбинация нескольких атомарных активностей операционной системы (обработка исключения, переключение контекста, обработка ошибки страницы и т.п.):

$$T = \sum_{i=1}^{N} \alpha_i r_i + U + F(\bar{r}) \quad (3)$$

где

- r_i - время выполнения атомарной активности i;

- $\bar{r}$ - вектор времен выполнения всех атомарных активностей;
- α_i - коэффициенты;
- U - константа, выражающая неучтенные значения;
- $F(\bar{r})$ - отклонение модели. Функция также выражает взаимовлияние атомарных активностей r_i.

Для того чтобы воспользоваться уравнением (3) необходимо измерить все r_i. Это довольно сложная задача, несмотря на то, что существует целый ряд подходящих инструментов [11], [12]. Еще сложнее произвести измерения с удовлетворительной точностью. Для практического применения необходимо упростить модель.

Для упрощения предлагается заменить атомарные активности r_i так называемыми атомарными тестами. Атомарный тест – это простое приложение, которое эмулирует поведение атомарной активности операционной системы. Для примера, мы можем измерить время переключения контекста с помощью приложения, создающего два процесса, взаимодействующих через неименованный канал. Обозначим время выполнения атомарного теста через t_i. Далее выразим t_i:

$$t_i = \beta_i r_i + u_i \qquad (4)$$

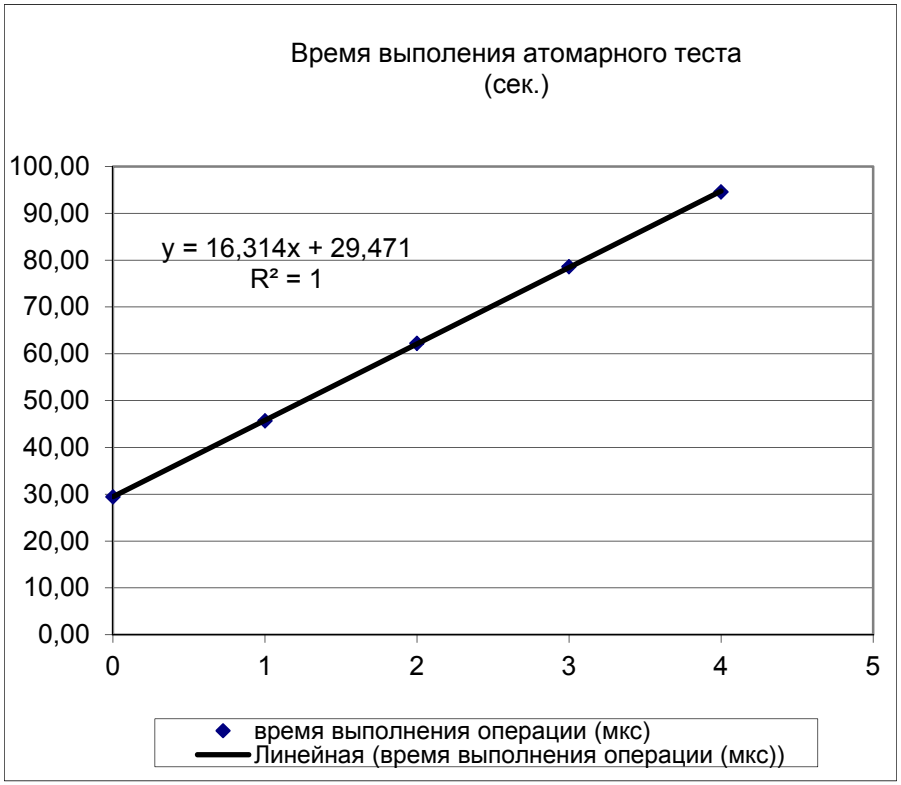
где

- β_i - коэффициенты;
- u_i - константы, выражающие неучтенные эффекты.

Теперь докажем линейность выражения (4). Для этого внесем некоторые изменения в гипервизор и выполним тест в виртуализованном окружении. Для того чтобы повлиять на t_i, можно реализовать в мониторе виртуальных машин искусственную задержку. Время выполнения атомарного теста t_i должно расти пропорционально:

$$\begin{aligned} t_i^1 &= \beta_i(r_i + 1\cdot\Delta) + u_i \\ t_i^2 &= \beta_i(r_i + 2\cdot\Delta) + u_i \\ t_i^3 &= \beta_i(r_i + 3\cdot\Delta) + u_i \\ &\Lambda \\ t_i^k &= \beta_i(r_i + k\cdot\Delta) + u_i \end{aligned} \qquad (5)$$

В качестве доказательства линейности выражения (4) была разработана упомянутая выше программа взаимодействия двух процессов через неименованный канал. Выполнение программы осуществлялось в виртуальной машине под управлением гипервизора XEN. Задержка была реализована в виде простого цикла в функции, обновляющей контрольный регистр CR3 при переключении контекста. Линейная зависимость времени выполнения атомарного теста от числа итераций цикла показана на графике:



5. Оценка производительности по методу атомарных тестов

Теперь можно выразить r_i из формулы (4) и сделать соответствующие замены в (3):

$$T = \sum_{i=1}^{N}\frac{\alpha_i}{\beta_i}t_i - \sum_{i=1}^{N}\frac{\alpha_i}{\beta_i}u_i + U + F(\bar{r}) = \sum_{i=1}^{N}\gamma_i t_i + U^* + F(\bar{r}) \qquad (6)$$

Далее уравнение (6) можно преобразовать к более удобному, для применения на практике, виду:

$$T = T_0 + \sum_{i=1}^{N} \gamma_i \Delta t_i + F^*(\bar{r}) \quad (7)$$

Практическая ценность формулы (7) заключается в том, что время выполнения сложной нагрузки T_0 можно измерить только один раз, а затем использовать его для оценки времени работы этой нагрузки в виртуализованном окружении T, подставляя характеристические коэффициенты γ_i используемого гипервизора. Норма функции $\left\|F^*(\bar{r})\right\|$ выражает отклонение измеренного времени T_0 от времени, предсказанного моделью.

Также необходимо отметить некоторые особенности приведенной модели:

- Условие (4) должно выполняться всегда, т.к. атомарные тесты обязаны линейно зависеть от атомарных активностей.
- Система должна быть полной относительно выражения (3). Это значит, что набор атомарных тестов должен полностью покрывать рассматриваемую нагрузку. Отследить наличие неучтенных атомарных активностей можно по изменениям значения константы U.

Следует отметить, что значение константы U необходимо подбирать для каждого гипервизора в отдельности, ограничивая норму функции $\left\|F^*(\bar{r})\right\|$.

6. *Заключение*

Перенос программного обеспечения в виртуализованное окружение всегда приводит к снижению общей производительности. Поэтому важной задачей является оценка требуемых приложениями ресурсов перед их консолидацией. В этой статье был представлен подход к оценке производительности ПО, основанный на атомарном тестировании. Суть подхода заключается в том, что любая сложная нагрузка может быть представлена, как совокупность простых активностей. А предсказать время работы приложения в виртуальной машине, можно собрав данные о производительности атомарных тестов в используемом виртуализованном окружении и вне него.

На данном этапе исследования сформулированной задачи, была представлена теоретическая модель, подтвержденная некоторыми практическими данными. Дальнейшая работа будет направлена на доказательство справедливости

модели и ее уточнение, путем расширения набора атомарных тестов и сбора данных в реальной продуктивной среде.

Список литературы

[1] Gmach D., Rolia J., Cherkasova L., Kemper A.: Capacity Management and Demand Prediction for Next Generation Data Centers. Proceedings of the International IEEE Conference on Web Services, 2007, pp. 43-50.

[2] Rolia J., Cherkasova L., Arlitt M., Andrzejak A.: A Capacity Management Service for Resource Pools. Proceedings of the 5th international workshop on software and performance, 2005, pp. 229-237.

[3] HP Integrity Essentials Capacity Advisor. http://h71036.www7.hp.com/enterprise/cache/262379-0-0-0-121.html. Дата обращения 10.09.2011.

[4] VMware Capacity Planner, http://www.vmware.com/products/capacity-planner/index.html. Дата обращения 10.09.2011.

[5] Barham P., Dragovic B., Fraser K., Hand S., Harris T., Ho A., Neugebauer R., Pratt I., Warfield A.: Xen and the art of virtualization. Proceedings of the nineteenth ACM symposium on Operating systems principles, 2003, pp. 164-177.

[6] King S., Dunlap G., Chen P.: Operating system support for virtual machines. Proceedings of the USENIX Annual Technical Conference, 2003, p. 6.

[7] Sugerman J., Venkitachalam G., Lim B.-H.: Virtualizing I/O Devices on VMware Workstation's Hosted Virtual Machine Monitor. Proceedings of the General Track: 2002 USENIX Annual Technical Conference, 2001, pp. 1-14.

[8] Cherkasova L., Gardner R.: Measuring CPU overhead for I/O processing in the Xen virtual machine monitor. Proceedings of the USENIX Annual Technical Conference, 2005, p. 24.

[9] Gupta D., Cherkasova L., Gardner R., Vahdat A.: Enforcing Performance Isolation Across Virtual Machines in Xen. Proceedings of the ACM/IFIP/USENIX 2006 International Conference on Middleware, 2006, pp. 342-362.

[10] Wood T., Cherkasova L., Ozonat K., Shenoy P.: Profiling and Modeling Resource Usage of Virtualized Applications. Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware, 2008, pp. 366-287.

[11] McVoy L., Staelin C.: Portable Tools for Performance Analysis. Proceedings of the USENIX Annual Technical Conference, 1996, p. 23.

[12] Staelin C., McVoy L.: Anatomy of a micro-benchmark. Proceedings of the USENIX Annual Technical Conference, 1998, p.13.