электронный научно-технический журнал
ИНЖЕНЕРНЫЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51036. ISSN 2307-0595

Профилирование распределенных систем

 # 02, февраль 2016

Лукьянчиков О. И.[1,*], Филатов В. В.[1]

УДК: 681.3.01

[1]Россия, МИРЭА «Московский государственный университет информационных технологий, радиотехники и электроники»
*luk-it6@yandex.ru

Введение

Одним из важных качеств информационных систем и отдельных приложений является временная эффективность реализации. Особо остро эффективность влияет на показатели качества обслуживания QoS в облачных системах.

Разработчик ещё в процессе разработки программных средств имеет представление о возможных «узких» местах, так как при ограничении времени принимается более удобная реализация, а не более эффективная. Для оценки эффективности в жизненном цикле разработки существует только один этап, в ходе которого возможно обнаружить и, при необходимости, исправить наиболее ресурсозатратные места программного обеспечения. Используя жизненный цикл согласно [1], этим этапом является испытание программы, включающего в себя проведение предварительных государственных, межведомственных, приёмо-сдаточных и других видов испытаний и корректировку программы и программной документации по результатам испытаний. Именно на этом этапе производится функциональное тестирование, тестирование безопасности проводится тестирование производительности, в которое включается нагрузочное тестирование, стресс-тестирование, тестирование стабильности или надежности и объемное тестирование. Профилирование относиться к нагрузочному тестированию.

Профилирование представляет собой процесс наблюдения и записи показателей о поведении участков кода или приложения в целом. Профилирование программных средств может проводится программистом в процессе разработки программных средств, реализуя в программном коде фиксацию времени выполнения основных операций, но удобней после разработки, с помощью CASE-средств, называемыми «профилировщики». Сегодня существует достаточно большое количество средств профилирования, которые как встроены в состав средств разработки, так и распространяемые отдельно (TAU Performance System, MPI Parallel Environment, GProf, Intel VTune Performance Analyzer), которые позволяют построить дерево выполнения по исходным кодам, контролировать время выполнения операций и даже контроль сразу нескольких потоков. Выполнив анализ «узких»

мест с помощью профилировщиков, разработчик предпринимает меры по повышении эффективности реализации, а именно: распараллеливает операции между потоками, уменьшает лишние проходы циклов, выбирает специализированные структуры хранения данных (списки, очереди, хеш таблицы и прочие).

В распределенных системах большой объем занимают сетевые операции и в этом случае профилировщики и могут простроить корректно дерево выполнения программного кода и верно определить время выполнения, так как оно влияет от множества факторов, таких как:

– выполнение операции производится на удаленных вычислительных узлах, характеристики которых не всегда известны и которые на момент выполнения операции случайно загружены, поэтому невозможно точно определить время выполнения операций;

– не учитывается нагрузка на сеть;

– постоянно меняется количество пользователей.

Поэтому методы профилирования не применимы для распределенных систем и необходимы другие методы, которые позволят найти в системе «узкие» места.

1. Методика профилирования распределенных систем

Выделяют три подхода к оценке времени исполнения задачи на вычислительном ресурсе [2]:

– Анализ исходного кода, на основе которого строится аналитическая модель внутренних алгоритмов системы и ключевых характеристик входных данных, пример, которого рассмотрен в [3].

– Профилирование кода, в ходе которого запускаются расчеты на тесовых наборах данных, и определяется производительность системы на исследуемом типе кода, например, профилировщиками.

– Статистические методы прогнозирования, в которых сохраняется результаты о прошедших запусках решения задач, и затем используются для прогнозирования времени работы новых, примеры которого описаны в работах [4,5], и целая система с БД для этого описана в работе [6].

Каждый подход имеет свои недостатки и большие погрешности при оценке, поэтому они должны дополнять друг друга, и использоваться совместно, особенно для оценки времени выполнения сетевых операций.

Для анализа исходного кода и профилирования стоится ориентированный граф, описывающий маршруты программного кода, в котором блоки программы являются узлами, соединенными ребрами, при условии, если управление может переходить с одного блока на другой. Данный граф строиться с помощью анализа программного кода разработчиком и с помощью профилировщиков, примеры этого описаны в работах [7,8]. При рассмотрении распределенных систем приоритетными для анализа являются сетевые операции, поэтому для упрощения данного графа незначительными локальными операциями можно

пренебречь. Также в условиях популярности языка UML удобней и корректней вместо графа описывать маршрут программного кода диаграммой активности в нотациях языка UML, в основном благодаря возможности обозначения параллельных процессов.

Разработчик распределенных систем знает об основных процессах, происходящих в системе, на основе чего может описать последовательность действий при сетевом обмене и построить граф или диаграмму активности. Однако разработчик пользуется сторонними технологиями и библиотеками для него могут представлять «черный ящик». Также при анализе программного кода разработчик может не учесть многие операции системы, или посчитать их не существенными, пренебрегая ими, поэтому провести анализ кода не всегда возможно сразу верно. Отсюда требуются эксперименты и анализ статистически собранных данных, подтверждающих корректность сделанной ранее оценки операций.

На основе вышесказанного предлагается следующий метод для профилирования распределенных систем:

1) Построение ориентированного графа или диаграмму активности в нотации языка UML, позволяющие описать маршруты программного кода.
2) Построение математической формулы оценки среднего времени выполнение операций.
3) Проведение испытаний по оценки времени выполнения операций, при изменяющемся количестве вычислительных узлов.
4) Корректировка математической формулы по результатам испытаний.
5) Если по результатам испытаний не удалось однозначно подтвердить корректность формулы, то добавление искусственных задержек и шаг 2, иначе 6.
6) Определение узких мест в системе, кардинально влияющих на производительность системы.

2. Профилирование распределенных систем, построенных на основе технологии агентно-реляционного отображения.

Для примера предложенного метода, рассмотрим профилирование распределенной системы, построенной на основе технологии агентно-реляционного отображения (АРО), принципы которой описаны в работах [9,10]. Данная технология представляет собой гибрид технологий объектно-реляционного отображения (ORM) и технологий удаленного взаимодействия (технологии удаленных объектов, вызова удаленных процедур и ориентированные на обработку сообщений). В результате технология АРО имеет интерфейс ORM и позволяет выполнять реляционные команды (select, update, insert, delete) как к СУБД, так и на других удаленных объектах. Данная технология, за счет своей универсальности, позволяет строить системы практически любой архитектуры.

Для исследований простроим распределенную систему, состоящую из одной БД и различного количества приложений, распределенных между несколькими вычислительными узлами, как показано на рис. 1. Технические характеристики системы:

– Вычислительный узел 1 CPU Pentium Dual-Core E5200 2.5 ГГц, ОЗУ 4Гб, ОС Windows 7, на котором располагается приложение инициатор операций, и засекающий время выполнения операций и СУБД PostgreSQL с БД на 5 тыс. записей.

– Множество вычислительных узлов в виде виртуальных машин, поднятые на отдельном сервере под управлением VMwareEXSi, под каждую выделен 1 CPU, ОЗУ 1Гб, ОС Windows XP.

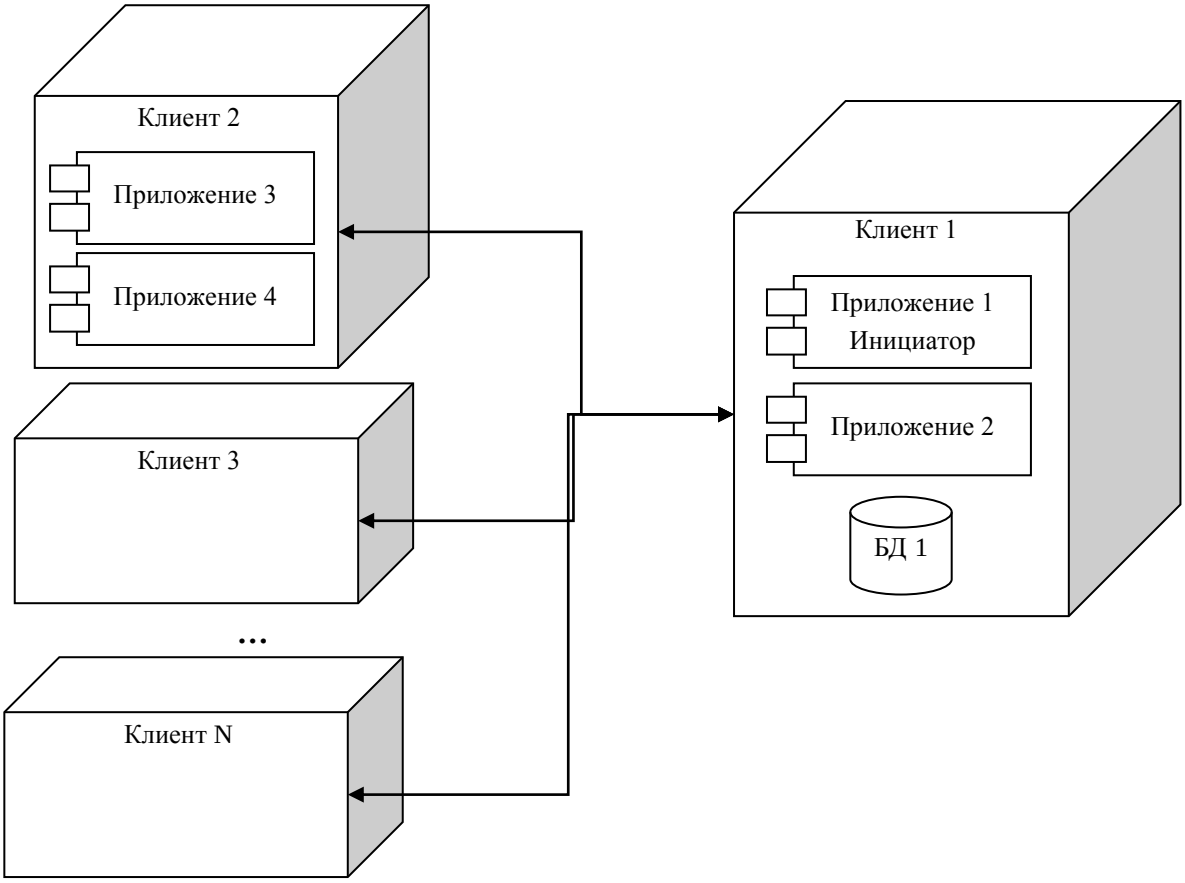

Рис. 1. Экспериментальный стенд.

Исследования эффективности данной архитектуры направлены на оценку задержек при синхронизации данных во время выполнения операций обновления. При данной операции происходит следующая последовательность действий:

– выполняется генерация команды;

– всем клиентам посылается сообщение с командой, также команда посылается всем БД;

– все удаленные объекты на различных вычислительных узлах приложения выполняют операцию асинхронно и при успешном выполнении отсылают обратно инициатору сообщение об успешном выполнении, параллельно операции выполняются и на всех распределенных БД;

– инициатор при окончательном сборе ото всех клиентов успешных сообщений фиксирует у себя изменение и отсылает всем остальным клиентам и БД сообщение с командой фиксации изменений.

На основе вышеописанного процесса построим диаграмму активности в нотации языка UML, показанную на рис. 2.

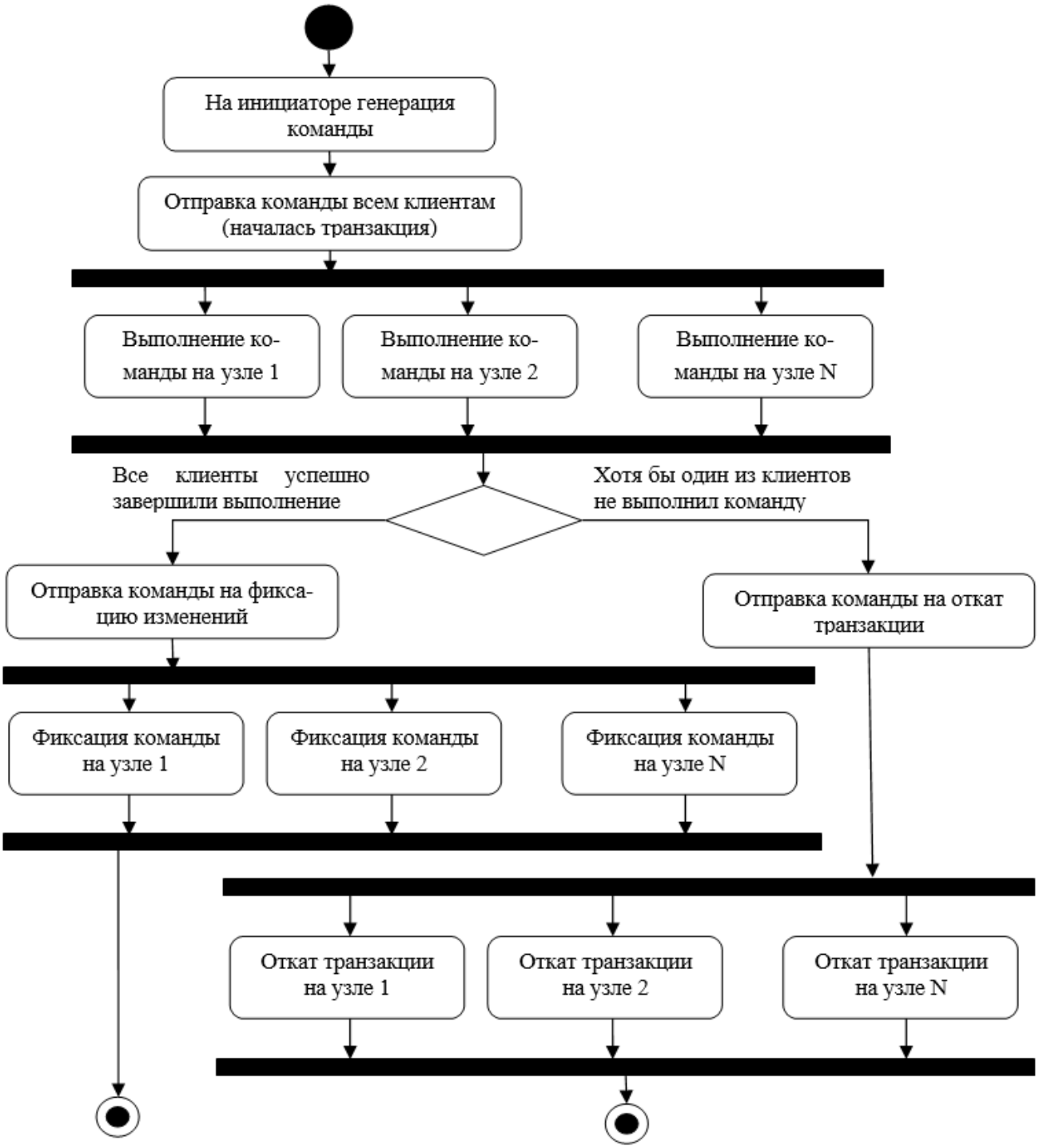

Рис. 2. Диаграмма активности выполнения операции обновления технологией агентно-реляционного отображения в нотации языка UML

Средняя математическая оценка производительности распределенной системы с распределенными БД при использовании технологии АРО будет иметь следующий вид:

$$\bar{T} = \bar{t}_g + 2\sum_{i=1}^{N+M}(\bar{t}_{z_i}) + \max\left(\max_{i=1..N}(k\bar{T}_{o_i} + \bar{t}_{a_i}),\ \max_{i=1..M}(\bar{T}_{db_i} + \bar{t}_{a_i})\right) + \bar{t}_c, \quad (1)$$

где

$\bar{T}$ – *средняя оценка времени выполнения одной операции;*

$\bar{t}_z$ – среднее время отправки заявки на выполнение операции;

$\bar{t}_g$ – среднее время генирации SQL запроса;

$\bar{T}_o$ – среднее время выполнения операции удаленным объектом;

$\bar{t}_a$ – среднее время ответа с результатом выполнения операции;

$\bar{T}_{db}$ – среднее время выполнения операции на СУБД;

$\bar{t}_c$ – среднее время фиксации операции;

N – количество удаленных узлов;

k – количество удаленных объектов на узле;

M – количество БД.

Данная оценка не включает в себя загрузку узлов, место хранение записи в структуре данных, вероятности неудачных выполнения операций и прочие факторы, поэтому все времена имеют среднее значение. Для подтверждения математической оценки времени выполнения операции обновления (1), проводится ряд опытов, среднестатистические результаты которых приведены на рис. 3. Как видно по данным графикам математическая оценка (1) имеет верный вид. Отклонение значений графика без задержек от значений графика с задержкой выполнения операции примерно различается на время принудительной задержки, тем самым подтверждая, что данная операция выполнятся параллельно на всех вычислительных узлах. Совершенно другой вид имеет график с задержкой на передачу данных, он имеет линейное увеличение как при арифметической прогрессии, тем самым показывая, что операция передачи данных является «узким» местом в системе.

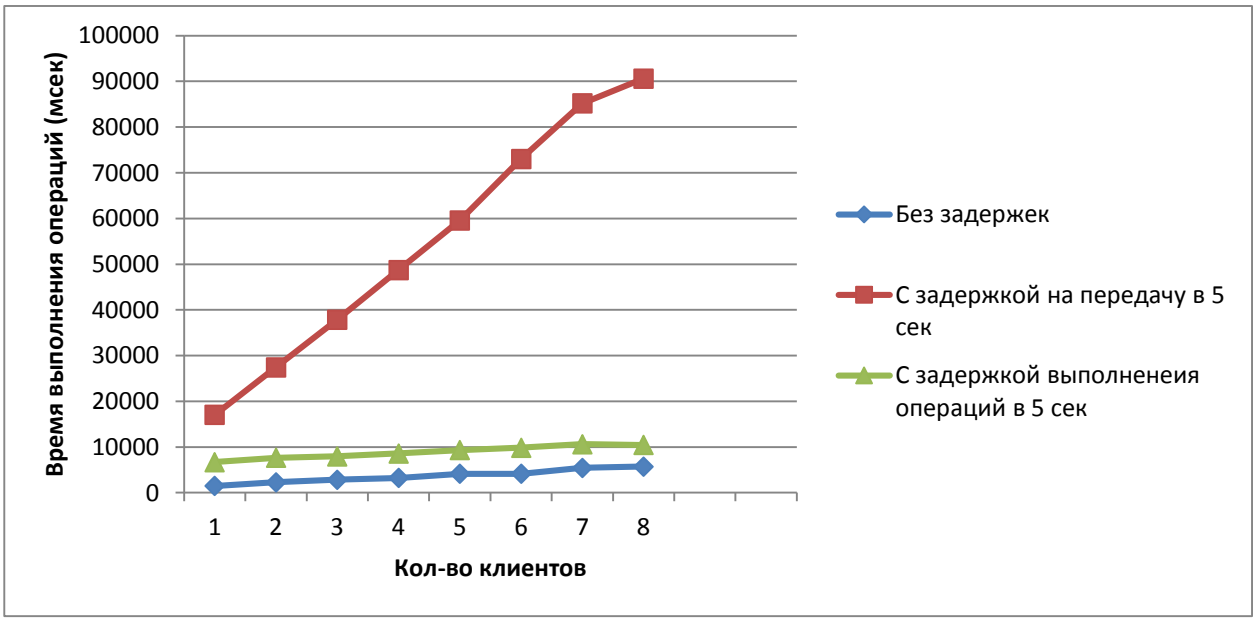

Рис. 3. Результаты экспериментов по оценке изменений времени выполнения операции изменения от меняющегося количества клиентов

Заключение

В статье рассмотрены различные подходы по временной оценке эффективности программных средств, и на основе этого предложен новый метод для профилирования распределенных систем. Пример применения данного метода приведен на распределенной системе, содержащую одну БД и множество клиентов, построенную на основе технологии APO. Предложенный метод позволит более точно оценить существующие архитектуры распределенных систем, выявить среди них наиболее эффективные и сформировать рекомендации по проектированию новых систем.

Список литературы.

[1]. ГОСТ 19.102-77 ЕСПД. Стадии разработки.

[2]. Iverson M.A., Ozguner F., Potter L.C. Statistical prediction of task execution times through analytic benchmarking for scheduling in a heterogeneous environment // Proc. Eighth Heterogeneous Computing Workshop (HCW'99). NewYork: IEEE Press. 1999. P.99–111.

[3]. Kerbyson D.J., Wasserman H.J., Hoisie A. Exploring advanced architectures using performance prediction //International Workshop on Innovative Architecture for Future Generation High Performance Processors and Systems.New York: IEEE Press. 2002. P.27–37

[4]. Devarakonda M.V., Iyer R.K. Predictability of process resource usage: a measurement-based study of UNIX // IEEE Trans. on Software Engineering. 1989. № 15. P. 1579–1586.

[5]. lghami O., Nau D.S. A general approach to synthesize problem-specific planners. Technical Report CS-TR-4597. College Park: Univ. of Maryland. 2004.

[6]. Булочникова Н.М., Горицкая В.Ю., Сальников А.Н. Система поддержки сбора и анализа статистики о работе вычислительной системы // Методы и средства обработки информации. Труды Второй всероссийской научной конференции. М.: Издательский отдел факультета ВМиК МГУ. 2005. С. 136-141.

[7]. Радченко Г.И., Соколинский Л.Б., Шамакина А.В. Модели и методы профилирования и оценки времени выполнения потоков работ в суперкомпьютерных системах // Вычислительные методы и программирование. 2013. Т. 14. № 3. С. 96-103.

[8]. Киселёв Е.А. Построение информационного графа параллельной программы на основе данных профилирования и трассировки // М.: Изд-во МГУ. 2011. С. 541-546.

[9]. Лукьянчиков О.И. Технология агентно-реляционного отображения // Современные проблемы науки и образования. 2015. № 1. Режим доступа: URL: http://www.science-education.ru/121-18623.

[10]. Лукьянчиков О.И. Технология объектно-реляционного отображения реляционных данных при агентно-ориентированном программировании // Задачи системного анализа, управления и обработки информации. Межвузовский сборник научных трудов. Выпуск 5. М.: Изд. МТИ. 2015. С. 85–96.