УДК 004.051
DOI: 10.25559/SITITO.15.201901.72-80

Повышение эффективности виртуального рабочего окружения распределенной разработки программ

П. В. Колясников[1], И. Н. Силаков[2], Д. Ю. Ильин[2], А. А. Гусев[3], Е. В. Никульчев[2*]
[1] Российская академия образования, г. Москва, Россия
[2] МИРЭА – Российский технологический университет, г. Москва, Россия
[*] nikulchev@mail.ru
[3] Кубанский государственный университет, г. Краснодар, Россия

Аннотация

Для распределенной разработки программных систем широко применяются виртуальные рабочие окружения, которые позволяют использовать технологии виртуализации для создания модели создаваемого проекта и автоматизации процессов синхронизации и интеграции компонентов. Виртуальное рабочее окружение позволяет избежать различий между локальными рабочими окружениями разработчиков, а также конечной платформой, и значительно упростить установку и настройку окружений при внедрении систем в эксплуатацию. В статье рассмотрена система организации виртуальных рабочих окружений Vagrant. В качестве альтернативы была рассмотрена технология Docker. На примере проекта по разработке открытой цифровой платформы массовых психологических исследований проведены оценки параметров и предложены механизмы и технологии повышения эффективности функционирования облачного виртуального окружения. Определен набор основных критериев оценки эффективности конфигурации рабочего окружения: быстрое развёртывание; увеличение быстродействия и уменьшение объема используемых ресурсов; повышение скорости обмена данными между хост-машиной и виртуальной машиной. Приведены результаты экспериментальных оценок параметров, определяющих сформулированные критерии эффективности: задействованные ресурсы процессора (процент); объём задействованной оперативной памяти (ГБ); время запуска виртуальных машин (секунды); время выполнения полной сборки компонента Build и повторной сборки Watch (секунды). Для повышения эффективности рассмотрен драйвер доступа к файловой системе на основе протокола NFS. По результатам экспериментальных оценок выявлен значительный рост производительности и скорости работы при использовании единой виртуальной машины и драйвера на основе NFS. Таким образом, определен набор критериев, методов оценивания и предложены механизмы повышения эффективности виртуального рабочего окружения при распределенной разработке больших программных систем.

Ключевые слова: распределенная разработка программ, виртуальное рабочее окружение, повышение эффективности разработки, виртуальные машины, Vagrant, Docker, NFS, Webpack.

Благодарности: работа выполнена при финансировании Российского фонда фундаментальных исследований, проект № 17-29-02198 (офи_м) «Разработка открытой экспериментально-аналитической веб-платформы для сбора и интеллектуального анализа данных междисциплинарных исследований в области психического здоровья».

Для цитирования: *Колясников П. В., Силаков И. Н., Ильин Д. Ю., Гусев А. А., Никульчев Е. В.* Повышение эффективности виртуального рабочего окружения распределенной разработки программ // Современные информационные технологии и ИТ-образование. 2019. Т. 15, № 1. С. 72-80. DOI: 10.25559/SITITO.15.201901.72-80

© Колясников П.В., Силаков И.Н., Ильин Д.Ю., Гусев А.А., Никульчев Е.В., 2019

Контент доступен под лицензией Creative Commons Attribution 4.0 License.
The content is available under Creative Commons Attribution 4.0 License.

Increasing the Efficiency of the Virtual Development Environment for Distributed Software Development

P. V. Kolyasnikov[1], I. N. Silakov[2], D. Yu. Ilin[2], A. A. Gusev[3], E. V. Nikulchev[2*]
[1] Russian Academy of Education, Moscow, Russia
[2] MIREA – Russian Technological University, Moscow, Russia
[*] nikulchev@mail.ru
[3] Kuban State University, Krasnodar, Russia

Abstract

Virtual distributed environments are widely used for distributed development of software systems, which enable the use of virtualization technologies to create a model of project and to automate processes of synchronization and integration of its components. The virtual development environment allows us to avoid the differences between the local environments of developers and between the local environments and the final platform, and greatly simplifies the installation and configuration of environments during the implementation of systems in operation. The article studies the system of organization of virtual development environments in Vagrant. Alternatively, Docker technology was considered. Using the example of a project aimed at developing an open digital platform for mass psychological research, parameter estimates were made and mechanisms and technologies to improve the efficiency of the cloud virtual environment were proposed. A set of basic criteria for evaluating the effectiveness of the configuration of the development environment has been defined to be: rapid deployment; increase in speed and decrease in the volume of resources used; increase in the speed of data exchange between the host machine and the virtual machine. The results of experimental estimates of the parameters that define the formulated efficiency criteria are given as: processor utilization involved (percentage); the amount of RAM involved (GB); virtual machine startup time (seconds); the time to complete the building of the Build component and to reassemble the Watch component (seconds). To improve the efficiency, a file system access driver based on the NFS protocol is considered. According to the results of experimental assessments, a significant increase in productivity and speed of operation was revealed when using a single virtual machine and the driver based on the NFS. Thus, the set of criteria and estimation methods has been defined and the mechanisms have been proposed for increasing the efficiency of the virtual development environment in the distributed development of large software systems.

Keywords: Distributed Software Development, Virtual Development Environment, Increase Development Efficiency, Virtual Machines, Vagrant, Docker, NFS, Webpack.

Acknowledgements: The work was done with the financing of the Russian Foundation for Basic Research, project No. 17-29-02198 (ofi_m) “Developing an open experimental-analytical web platform for collecting and mining data from interdisciplinary research in the field of mental health”.

For citation: Kolyasnikov P.V., Silakov I.N., Ilin D.Yu., Gusev A.A., Nikulchev E.V. Increasing the Efficiency of the Virtual Development Environment for Distributed Software Development. *Sovremennye informacionnye tehnologii i IT-obrazovanie* = Modern Information Technologies and IT-Education. 2019; 15(1):72-80. DOI: 10.25559/SITITO.15.201901.72-80

1. Введение

При командной распределенной разработке больших проектов по созданию программного обеспечения в настоящее время широко применяются виртуальные облачные рабочие окружения [1]. Данная технология использует виртуализацию и средства управления конфигурациями виртуальных машин [2] для применения необходимых параметров и установки требуемых компонентов, автоматизируя процесс синхронизации, настройки и запуска рабочего окружения.

Использование виртуальных рабочих окружений позволяет избежать различий между локальными рабочими окружениями разработчиков, а также конечной платформой, и значительно упростить установку и настройку окружений на новых машинах. В статье рассматривается система для подготовки виртуальных рабочих окружений Vagrant[1], позволяющая создавать воспроизводимые виртуальные рабочие окружения [3], снижая ряд сложностей, возникающих по причине несовместимости программно-аппаратных средств, используемых разработчиками [4].

Использование системы управления разработкой облегчает одновременную распределенную работу над несколькими компонентами разрабатываемого программного обеспечения [5, 6], а также автоматизируют процесс установки и настройки всех необходимых компонентов среды разработки [7-9]. Процессы обновления и модификации используемых компонентов также упрощаются [10], так как достаточно только внести изменения в конфигурационные файлы.

Статья содержит результаты исследования по повышению эффективности виртуальной разработки программного обеспечения цифровой платформы психологических исследований [11]. При использовании для разработки структуры виртуальных машин, максимально приближенных к структуре реальных серверов, были обнаружены характерные проблемы производительности [12]: низкая скорость обмена данными, а также нестабильность работы.

Целью работы являются разработка методики исследований и разработка механизмов повышения эффективности виртуального рабочего окружения при распределенной разработке больших программных систем.

2. Исходные данные

В качестве исходного рабочего окружения использована структура виртуальных машин, максимально приближенная к структуре серверов, задействованных в проекте (рисунок 1). При данном подходе каждому компоненту платформы соответствует отдельная конфигурация виртуальной машины.

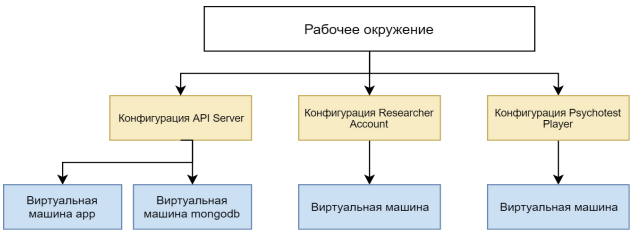

Рис. 1. Схема исходного рабочего окружения
Fig. 1. Diagram of the original working environment

При разработке исходного рабочего окружения были выбраны следующие параметры виртуальных машин для компонентов:

- **API Server**: ОС ubuntu/xenial64, версия 20180424.0.0, 1 CPU, 1024 MB RAM;
- **Researcher Account**: ОС ubuntu/xenial64, версия 20180424.0.0, 2 CPU, 1024 MB RAM;
- **Psychotest Player**: ОС ubuntu/xenial64, версия 20180424.0.0, 2 CPU, 512 MB RAM.

При разработке с использованием подобной структуры рабочего окружения возникает необходимость в одновременном запуске нескольких виртуальных машин для одновременной работы нескольких взаимосвязанных компонентов. Во время практического использования разработанной структуры виртуального рабочего окружения на компьютерах разработчиков было замечено значительное падение производительности. В ходе дальнейших наблюдений были выявлены следующие проблемы использованного рабочего окружения:

- высокая нагрузка на рабочих машинах разработчиков;
- низкая скорость обмена данными виртуальных машин с основной системой;
- высокое время сборки компонентов;
- нестабильность работы из-за возрастающих нагрузок;
- необходимость ручного выполнения большого количества операций при запуске окружения.

В ходе разработки крупных проектов количество разрабатываемых компонентов возрастает, что при использовании подобной структуры может привести к ещё большему понижению производительности компьютеров и понижению эффективности разработчиков [13]. Помимо этого, при разработке также необходимо использовать дополнительное программное обеспечение (среда разработки, веб-браузер, сетевые драйверы ввода данных [14]), что также приводит к значительному росту нагрузки на рабочие машины разработчиков, понижению стабильности работы системы и повышению времени выполнения программного обеспечения, задействованного в разработке.

Низкая скорость обмена данными основной системы и виртуальной машины, наблюдаемая при использовании разработанной структуры виртуального окружения, может быть следствием двух факторов: значительного роста нагрузки на машину разработчика, но также используемого виртуальной машиной драйвера для доступа к родительской файловой системе.

Можно выделить задачу необходимости выполнения большого количества повторяющихся действий вручную при запуске рабочего окружения, что также требует затрат времени на ожидание окончания запуска каждой из виртуальных машин для перехода к запуску остальных компонентов.

Описанные выше проблемы значительно влияют на скорость разработки программ в связи с высокими потерями производительности и временными затратами. При расширении общей структуры проекта параллельная разработка нескольких компонентов становится практически невозможной из-за ещё большего роста требований к ресурсам.

Таким образом, можно сделать вывод о необходимости исследования подходов к созданию рабочих окружений разработчиков и разработке усовершенствованной структуры. Исходя из выявленных в ходе наблюдений проблем выделены основные

[1] Vagrant. [Электронный ресурс]. URL: https://www.vagrantup.com (дата обращения: 3.02.2019).

требования к искомому решению для рабочих окружений разработчиков:

- быстрое развёртывание;
- повышение быстродействия и понижение используемых ресурсов;
- повышение скорости обмена данными между хост-машиной и виртуальной машиной.

Необходимо провести оценку альтернативных технологий и разработать решение, соответствующее описанным требованиям. Также необходимо произвести экспериментальную оценку используемому и альтернативному решениям, и оценить целесообразность перехода на альтернативную структуру рабочего окружения.

3. Методика исследования

Для проведения экспериментов использовались замеры следующих показателей:

- задействованные ресурсы процессора (процент);
- объём задействованной оперативной памяти (ГБ);
- время запуска виртуальных машин (секунды);
- время выполнения полной сборки компонента Build (секунды);
- время выполнения повторной сборки компонента Watch (секунды).

Все эксперименты проводились на одной рабочей машине со следующей конфигурацией:

- материнская плата: Dell 00TMJ3;
- процессор: Intel Core i5-5250U;
- оперативная память: DDR3 12 ГБ, частота 800 МГц;
- внутренний диск: Samsung SSD 860 EVO 500 ГБ;
- операционная система: Windows 10.

Для каждого из компонентов виртуального рабочего окружения проводилось 10 экспериментов для замера каждого из показателей. Для оценки отклонения полученных значений использован коэффициент Стьюдента [15] при доверительной вероятности P=0.95.

Для измерения задействованных ресурсов процессора и оперативной памяти был разработан программный сценарий на языке bash, который сопоставлял показатели до запуска виртуального рабочего окружения с показателями после полного завершения запуска виртуальной машины. Для получения данных о необходимых показателях в коде сценария была задействована системная команда «WMIC», предоставляющая возможность получения необходимых данных посредством командного интерфейса.

Все замеры показателей проводились в состоянии ожидания виртуальной машины – после завершения её запуска и при работе ключевых (веб-сервер, сервер баз данных и т.д.) компонентов.

Для оценки времени запуска виртуальных машин использовалась системная утилита «time», запускаемая с Vagrant в качестве префикса командой «time vagrant up» и предоставляющая после завершения информацию о времени, затраченном на выполнение. При необходимости запуска нескольких компонентов рабочего окружения запуск необходимых контейнеров Vagrant осуществлялся параллельно.

Для оценки времени выполнения задач сборки (полной сборки «build» и повторной сборки в режиме отслеживания «watch») использовались данные, указываемые после выполнения задачи инструментом для сборки веб-приложений Webpack.

4. Оценка исходного рабочего окружения

С целью оценки нагрузки и скорости работы используемого рабочего окружения был проведён эксперимент, включающий замеры времени, затрачиваемого на запуск, а также оценку задействованных ресурсов центрального процессора и оперативной памяти. Результаты эксперимента представлены в таблице 1.

Таблица 1. Результаты оценки потребления ресурсов рабочими окружениями

Table 1. Results of assessing resource consumption by work environments

Конфигурация	Кол-во ВМ	Время запуска, сек	Нагрузка на ЦП, %	Занято ОЗУ, ГБ
Исходная конфигурация (API Server)	2	289,4 ± 2,8	24,3% ± 6,7%	1,5 ± 0,1
Исходная конфигурация (Researcher Account)	1	118,5 ± 13	21,8% ± 6%	0,5 ± 0,08
Исходная конфигурация (Psychotest Player)	1	155,3 ± 19,1	16,1% ± 3%	0,7 ± 0,04
Исходная конфигурация (API Server, Researcher Account)	3	347,4 ± 4,4	35,4% ± 9,1%	2 ± 0,2
Исходная конфигурация (API Server, Psychotest Player)	3	290,1 ± 4,5	29% ± 5,3%	2 ± 0,06
Исходная конфигурация (3 компонента)	4	404,8 ± 4,1	49% ± 5%	2,2 ± 0,18

На основе данных, полученных в ходе эксперимента, можно сделать вывод об объективности проблем, описанных в ходе наблюдений, что подкрепляет необходимость рассмотрения альтернативных решений.

5. Разработка механизмов повышения эффективности окружения

В связи с высокой нагрузкой, возникающей из-за использования виртуальных машин, используемых в основе Vagrant, целесообразно рассмотреть альтернативные технологии для организации рабочих окружений. В качестве альтернативы была рассмотрена технология Docker[2], которая также может применяться в целях организации синхронизируемых рабочих окружений [16].

Система Docker основана на использовании абстрагирования при помощи встроенных в ядро Linux возможностей виртуализации для изоляции различных используемых компонентов в рамках автономных контейнеров с обособленным окружением [17].

Изначально подобный подход был ориентирован на доставку разработанных программных решений до сервера, однако

[2] Docker [Электронный ресурс]. URL: https://www.docker.com (дата обращения: 3.02.2019).

позднее платформа также стала применяться для замены виртуальных рабочих окружений. При этом Docker может выступать как самостоятельным решением [18], так и работать в качестве основы решения на основе Vagrant, тем самым заменяя собой использование виртуальных машин.

Для оценки Docker в качестве альтернативы Vagrant был выделен ряд критериев, по которым было проведено сравнение технологий, результаты которого приведены в таблице 2.

Таблица 2. Сравнение Vagrant и Docker
Table 2. Comparison of Vagrant and Docker

Критерий	**Vagrant**	**Docker**
Лицензия	MIT	Apache 2.0
Разработчик	Hashi Corp.	Docker, Inc.
Поддерживаемые платформы	Linux, Windows, macOS	Linux, Windows, macOS
Назначение	Виртуальные рабочие окружения	Контейнеризация приложений и автоматизация задач, доставка компонентов до сервера
Простота эксплуатации	Запуск окружения одной командой	Необходимо понимание системы контейнеров и зависимостей; запуск окружения одной командой в комбинации с Vagrant
Простота конфигурирования	Конфигурационный файл на языке Ruby	Собственный формат конфигурационных файлов
Структура контейнеров	Контейнер включает в себя все зависимости, указанные в конфигурации, является лишь средой выполнения	Каждый компонент и его зависимости являются отдельными контейнерами

Можно сделать вывод, что обе технологии предоставляют сравнимые преимущества при разных подходах к выполнению схожих задач. Использование Docker для синхронизации и быстрой настройки рабочих окружений разработчиков потенциально может являться допустимым решением благодаря преимуществам контейнеризации и решениям, используемым в рамках реализации Docker. Тем не менее, подобный подход также обязывает разработать новую структуру компонентов с использованием Docker контейнеров.

В рамках разрабатываемой платформы важной задачей выделялось сохранение максимальной приближенности к окружению рабочих систем, выполняемых на удалённых серверах. В подобном случае изменение структуры компонентов для использования в качестве контейнеров Docker неизбежно становится причиной расхождения окружения разработчика и системного окружения сервера, а также значительно осложняет поддержку существующего решения в рамках разрабатываемой платформы. В связи с этим использование технологии контейнеризации становится нецелесообразным.

Другим возможным решением в данном случае может выступить сохранение Vagrant в качестве основы выбранного решения, но с использованием единой виртуальной машины для выполнения всех компонентов.

В качестве альтернативного решения стоит рассмотреть конфигурацию на основе единого виртуального окружения Vagrant. В этом случае каждый компонент выполняется в рамках одной виртуальной машины, используя также единую модульную конфигурацию Vagrant (рисунок 2).

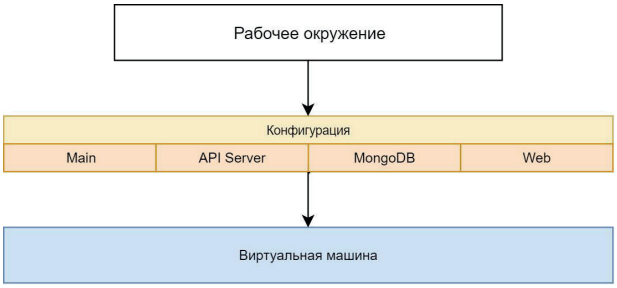

Рис. 2. Альтернативная структура рабочего окружения
Fig. 2. Alternative structure of the working environment

Использование подобной структуры должно понизить нагрузку на компьютер за счёт использования лишь одной виртуальной машины, при этом использование модульной конфигурации, основанной исходном решении, должно упростить поддержку и минимизировать расхождения с окружением сервера.

Для сравнения исходной и альтернативной структур рабочего окружения была проведена экспериментальная оценка потребляемых ресурсов (таблица 3). В качестве конфигурации виртуальной машины в основе альтернативного решения использовалась ОС ubuntu/xenial64, версия 20180424.0.0 с 2 CPU, 2048 MB RAM.

Таблица 3. Результаты замеров выполнения рабочих окружений
Table 3. The results of measurements of the performance of working environments

Конфигурация	**Кол-во ВМ**	**Время запуска, сек**	**Нагрузка на ЦП, %**	**Занято ОЗУ, ГБ**
Исходная конфигурация (3 компонента)	4	404,8 ± 4,1	49% ± 5%	2,2 ± 0,18
Альтернативная конфигурация (3 компонента)	1	210,3 ± 4,6	23,8% ± 7,7%	1,3 ± 0,13

В целях дополнительного повышения эффективности также стоит рассмотреть альтернативный драйвер обмена данными с родительской системой, основанный на протоколе NFS [16], так как подобное решение может значительно повысить скорость работы с файловой системой и выполнение задач [19].

Для использования NFS на машинах под управлением ОС Windows требуется использование дополнительного драйвера[3]. Кроме того, для корректной работы протокола NFS для

[3] Vagrant WinNFSd – GitHub [Электронный ресурс]. URL: https://github.com/winnfsd/vagrant-winnfsd (дата обращения: 3.02.2019).

обмена данными между родительской системой и виртуальной машиной необходимо подключение расширения bindfs[4], позволяющего перенести права доступа родительской системы.

В связи с архитектурной особенностью протокола NFS, не предоставляющего реализацию системных сигналов при отслеживании изменения файлов [20], также возникла необходимость модифицировать конфигурацию ПО Webpack[5], используемого для сборки веб компонентов. Были внесены следующие изменения:

- для обеспечения совместимости с NFS была установлена опция «watchOptions.poll», реализующая обход отслеживаемых файлов по заданному временному интервалу;
- для исключения неизменяемых файлов библиотек из опроса была использована опция «watchOptions.ignore».

Были проведены дополнительные замеры времени, затрачиваемого на сборку компонента при внесённых изменениях конфигурации (таблица 4).

Таблица 4. Результаты замеров времени сборки компонентов (в секундах)

Table 4. Results of component assembly time measurements (in seconds)

Конфигурация	Researcher Account		Psychotest Player	
	Build	Watch	Build	Watch
Исходная конфигурация (3 компонента)	120,4 ± 2,7	5 ± 0,4	131 ± 3,1	4,4 ± 0,1
Альтернативная конфигурация (3 компонента)	100,7 ± 2,5	3,6 ± 0,3	82,4 ± 2,4	3,3 ± 0,1
Улучшенная альтернативная конфигурация (3 компонента)	90,1 ± 2,4	1,2 ± 0,3	79 ± 1,3	1,1 ± 0,2

Подобные изменения конфигурации также повлекли за собой рост времени запуска (с 210,3 до 227,7 секунд в среднем), связанный с добавлением времени ожидания запуска драйвера NFS и монтирования директорий в виртуальной машине, а также рост используемых ресурсов процессора (с 24% до 29% в среднем. Тем не менее, показатели потребляемых ресурсов всё ещё остаются значительно более низкими в сравнении с полным запуском всех компонентов исходного рабочего окружения. При этом было замечено резкое понижение временных затрат на операции сборки компонентов при использовании предложенных улучшений.

6. Результаты экспериментальных оценок

Итоговые результаты проведенных экспериментов по оценке времени запуска и нагрузки на компьютер различных конфигураций виртуального рабочего окружения разработчиков приведены в таблице 5.

Результаты экспериментов по оценке времени выполнения полной сборки компонента Build и повторной сборки компонента Watch для компонентов платформы Researcher Account и Psychotest Player – представлены в таблице 6.

Таблица 5. Результаты оценки нагрузки и времени запуска рабочих окружений

Table 5. The results of the evaluation of the load and start-up time

Конфигурация	Кол-во ВМ	Время запуска, сек	Нагрузка на ЦП, %	Занято ОЗУ, ГБ
Исходная конфигурация (API Server)	2	289,4 ± 2,8	24,3% ± 6,7%	1,5 ± 0,1
Исходная конфигурация (Researcher Account)	1	118,5 ± 13	21,8% ± 6%	0,5 ± 0,08
Исходная конфигурация (Psychotest Player)	1	155,3 ± 19,1	16,1% ± 3%	0,7 ± 0,04
Исходная конфигурация (API Server, Researcher Account)	3	347,4 ± 4,4	35,4% ± 9,1%	2 ± 0,2
Исходная конфигурация (API Server, Psychotest Player)	3	290,1 ± 4,5	29% ± 5,3%	2 ± 0,06
Исходная конфигурация (3 компонента)	4	404,8 ± 4,1	49% ± 5%	2,2 ± 0,18
Альтернативная конфигурация (3 компонента)	1	210,3 ± 4,6	23,8% ± 7,7%	1,3 ± 0,13
Улучшенная альтернативная конфигурация (3 компонента)	1	227,7 ± 3,4	28,7% ± 5,4%	1,2 ± 0,09

Таблица 6. Результаты оценки времени выполнения сборки компонентов

Table 6. Time assessment for component assembly

Конфигурация	Researcher Account		Psychotest Player	
	Build	Watch	Build	Watch
Исходная конфигурация (API Server)	101,9 ± 2,9	3 ± 0,4	86,1 ± 2,1	3,1 ± 0,1
Исходная конфигурация (Researcher Account)	100,6 ± 2,5	2,7 ± 0,4	91,8 ± 2,4	3,7 ± 0,1
Исходная конфигурация (Psychotest Player)	99 ± 2,4	3,6 ± 0,4	89,3 ± 3	3,7 ± 0,1
Исходная конфигурация (API Server, Researcher Account)	98,7 ± 2,2	3,7 ± 0,5	119,2 ± 2,4	4 ± 0,2
Исходная конфигурация (API Server, Psychotest Player)	94,6 ± 2,7	4,1 ± 0,4	137,7 ± 3,6	4 ± 0,2
Исходная конфигурация (3 компонента)	120,4 ± 2,7	5 ± 0,4	131 ± 3,1	4,4 ± 0,1
Альтернативная конфигурация (3 компонента)	100,7 ± 2,5	3,6 ± 0,3	82,4 ± 2,4	3,3 ± 0,1
Улучшенная альтернативная конфигурация (3 компонента)	90,1 ± 2,4	1,2 ± 0,3	79 ± 1,3	1,1 ± 0,2

Из результатов проведённых экспериментов видно, что предложенная альтернативная конфигурация оказывает значительно меньшую нагрузку на центральный процессор (рисунок 3). Улучшенная альтернативная конфигурация использует несколько больше ресурсов процессора, что связано с использованием дополнительного драйвера, однако даже в этом случае нагрузка оказывается значительно ниже, чем при полном запуске исходного рабочего окружения.

Подобные изменения заметны также при сравнении используемой оперативной памяти (рисунок 4).

Понижение нагрузки на процессор, а также понижение количества используемых виртуальных машин, повлекло за собой зна-

[4] Vagrant bindfs – GitHub. [Электронный ресурс]. URL: https://github.com/gael-ian/vagrant-bindfs (дата обращения: 3.02.2019).

[5] Webpack. [Электронный ресурс]. URL: https://webpack.js.org/ (дата обращения: 3.02.2019).

чительное понижение среднего времени, необходимого для запуска виртуального рабочего окружения (рисунок 5). Улучшенная альтернативная конфигурация, как и в случае с нагрузкой на процессор, уступает по этому показателю альтернативной конфигурации без дополнительных модификаций. Это также может быть следствием использования дополнительного драйвера и необходимости ожидания его инициализации.

Тем не менее, улучшенная альтернативная конфигурация лидирует по показателям при выполнении полной сборки компонентов Researcher Account и Psychotest Player (рисунок 6), а также демонстрирует значительно лучшие показатели при выполнении их повторной сборки (рисунок 7). Подобные улучшения показателей являются прямым следствием использования драйвера NFS, повышающего скорость обмена данными с хост-машиной, что позволяет значительно сократить временные затраты при сборке компонентов.

Таким образом, полученная улучшенная альтернативная конфигурация рабочего окружения для Vagrant демонстрирует значительное понижение нагрузки на процессор и загрузки оперативной памяти в сравнении с полным запуском исходного рабочего окружения. При этом также демонстрируется значительное ускорение процесса сборки компонентов, что понижает время ожидания со стороны разработчиков, тем самым повышая их эффективность.

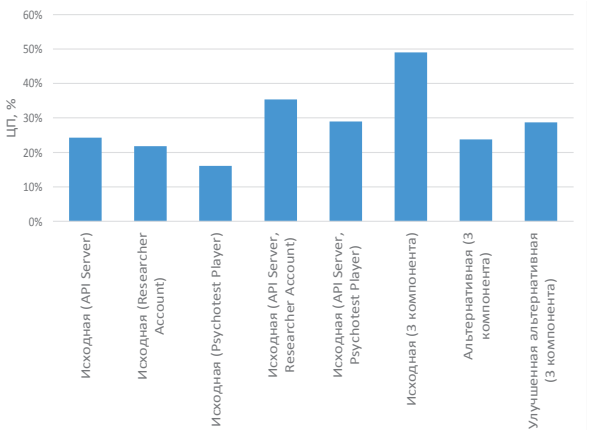

Рис. 3. Средняя нагрузка на центральный процессор, в процентах

Fig. 3. Average load on the central processor, in percentage

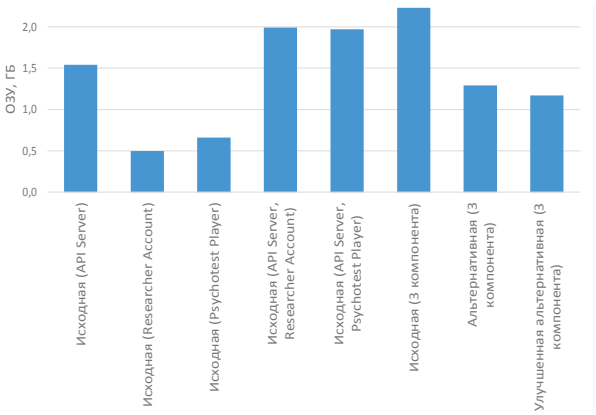

Рис. 4. Занято ОЗУ в среднем, ГБ

Fig. 4. Average RAM occupied, GB

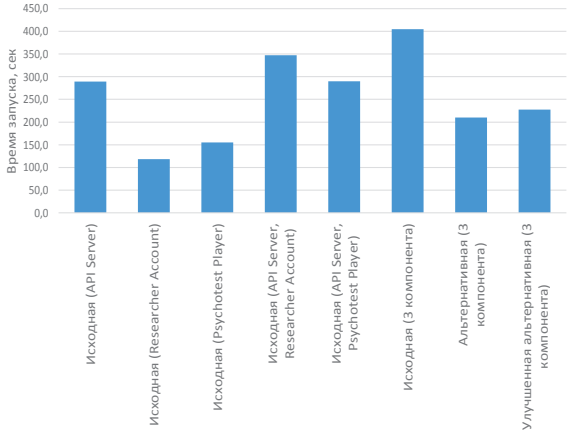

Рис. 5. Среднее время запуска рабочего окружения, сек.

Fig. 5. Average start time of the working environment, sec.

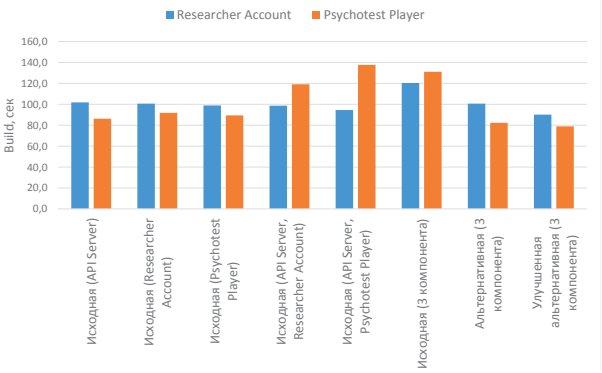

Рис. 6. Среднее время выполнения полной сборки компонентов Researcher Account и Psychotest Player (Build), сек.

Fig. 6. The average execution time of the full assembly of the components Researcher Account and Psychotest Player (Build), sec.

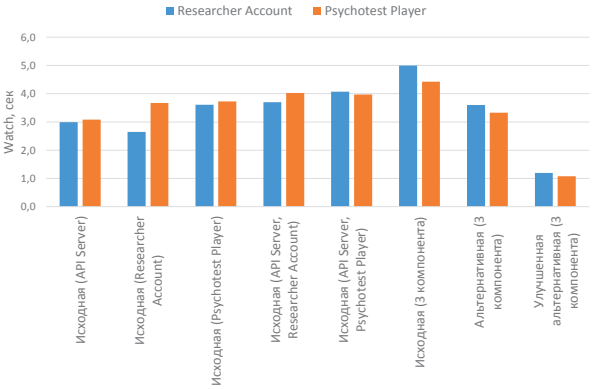

Рис. 7. Среднее время выполнения повторной сборки компонентов Researcher Account и Psychotest Player (Watch), сек.

Fig. 7. The average execution time of complete reassembling of the Researcher Account and Psychotest Player (Watch) components, sec.

7. Заключение

В связи с ростом сложности разрабатываемых веб-проектов и настройки рабочих окружений, возникает потребность в использовании виртуальных рабочих окружений.

При использовании виртуального рабочего окружения, в структуре которого каждый компонент использовал автономную виртуальную машину, были замечены проблемы с высоким потреблением ресурсов и падением производительности рабочих машин разработчиков, в связи с чем возникла необходимость в рассмотрении альтернативных решений.

В работе был проведен анализ причин снижения производительности при использовании рабочих окружений на основе виртуализации. Рассмотрены основные технологии, применяемые для разработки виртуальных рабочих окружений, а также предложена улучшенная структура. Кроме того, была произведена экспериментальная оценка исходного и альтернативного решений.

Реализованное альтернативное решение на основе единого рабочего окружения показало значительно меньшее потребление ресурсов, а также понижение времени выполнения задач сборки, благодаря драйвера для доступа к файловой системе на основе NFS.

Таким образом, использование виртуального рабочего окружения на основе единой виртуальной машины с использованием драйвера NFS позволяет значительно снизить нагрузку на компьютеры, повышает быстродействие и снижает временные затраты, тем самым повышая эффективность разработчиков.

References

[1] Caballer M., Blanquer I., Moltó G., de Alfonso C. Dynamic management of virtual infrastructures. *Journal of Grid Computing*. 2015; 13(1):53-70. (In Eng.) DOI: 10.1007/s10723-014-9296-5

[2] Giannakopoulos I., Konstantinou I., Tsoumakos D., Koziris N. Cloud application deployment with transient failure recovery. *Journal of Cloud Computing*. 2018; 7(1):11. (In Eng.) DOI: 10.1186/s13677-018-0112-9

[3] Spanaki P., Sklavos N. Cloud Computing: Security Issues and Establishing Virtual Cloud Environment via Vagrant to Secure Cloud Hosts. In: Daimi K. (eds) Computer and Network Security Essentials. Springer, Cham, 2018, pp. 539-553. (In Eng.) DOI: 10.1007/978-3-319-58424-9_31

[4] Hashimoto M. Vagrant: Up and Running: Create and Manage Virtualized Development Environments. O'Reilly Media Inc., 2013. 156 pp. (In Eng.)

[5] Nam Pham Nguyen Xuan, Sungmin Lim, and Souhwan Jung. 2017. Centralized management solution for vagrant in development environment. Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication (IMCOM '17). ACM, New York, NY, USA, 2017. Article 37, 6 pp. (In Eng.) DOI: https://doi.org/10.1145/3022227.3022263

[6] Thompson C. Vagrant Virtual Development Environment Cookbook. Packt Publishing Ltd., 2015. 250 pp. (In Eng.)

[7] Mouat A. Using Docker: Developing and Deploying Software with Containers. O'Reilly Media Inc., 2016. 358 pp. (In Eng.)

[8] Sammons G. Learning Vagrant: Fast Programming Guide. CreateSpace Independent Publishing Platform, 2016. 68 pp. (In Eng.)

[9] Peacock M. Creating Development Environments with Vagrant. Packt Publishing Ltd., 2015. 153 pp. (In Eng.)

[10] Iuhasz G., Pop D., Dragan I. Architecture of a Scalable Platform for Monitoring Multiple Big Data Frameworks. *Scalable Computing: Practice and Experience*. 2016; 17(4):313-321. (In Eng.) DOI: 10.12694/scpe.v17i4.1203

[11] Nikulchev E., Ilin D., Kolyasnikov P., Belov V., Zakharov I., Malykh S. Programming Technologies for the Development of Web-Based Platform for Digital Psychological Tools. *International Journal of Advanced Computer Science and Applications*. 2018; 9(8):34-45. (In Eng.) DOI: 10.14569/IJACSA.2018.090806

[12] Kashyap S., Min C., Kim T. Opportunistic spinlocks: Achieving virtual machine scalability in the clouds. *ACM SIGOPS Operating Systems Review*. 2016; 50(1):9-16. (In Eng.) DOI: 10.1145/2903267.2903271

[13] Saikrishna P.S., Pasumarthy R., Bhatt N.P. Identification and Multivariable Gain-Scheduling Control for Cloud Computing Systems. *IEEE Transactions on Control Systems Technology*. 2017; 25(3):792-807. (In Eng.) DOI: 10.1109/TCST.2016.2580659

[14] Li J., Xue S., Zhang W., Ma R., Qi Z., Guan H. When I/O Interrupt Becomes System Bottleneck: Efficiency and Scalability Enhancement for SR-IOV Network Virtualization. *IEEE Transactions on Cloud Computing*. 2017. pp. 1-1. (In Eng.) DOI: 10.1109/TCC.2017.2712686

[15] Zabell S.L. On Student's 1908 Article "The Probable Error of a Mean". *Journal of the American Statistical Association*. 2008; 103(481):1-7. (In Eng.) DOI: 10.1198/016214508000000030

[16] Chen M., Bangera G.B., Hildebrand D., Jalia F., Kuenning G., Nelson H., Zadok E. vNFS: Maximizing NFS Performance with Compounds and Vectorized I/O. *ACM Transactions on Storage*. 2017; 13(3):21. (In Eng.) DOI: 10.1145/3116213

[17] Kane S.P., Matthias K. Docker: Up & Running: Shipping Reliable Containers in Production. O'Reilly Media Inc, 2018. 352 pp. (In Eng.)

[18] Peinl R., Holzschuher F., Pfitzer F. Docker Cluster Management for the Cloud - Survey Results and Own Solution. *Journal of Grid Computing*. 2016; 14(2):265-282. (In Eng.) DOI: 10.1007/s10723-016-9366-y

[19] Krieger M.T., Torreno O., Trelles O., Kranzlmüller D. Building an open source cloud environment with auto-scaling resources for executing bioinformatics and biomedical workflows. *Future Generation Computer Systems*. 2017; 67:329-340. (In Eng.) DOI: 10.1016/j.future.2016.02.008

[20] Dani A.S. JavaScript by Example. Packt Publishing, 2017. 334 pp. (In Eng.)

Поступила 3.02.2019; принята к публикации 10.03.2019;
опубликована онлайн 19.04.2019.
Submitted 3.02.2019; revised 10.03.2019;
published online 19.04.2019.

Об авторах:

Колясников Павел Владимирович, аналитик, лаборатория мониторинга эффективности и качества научных исследований информационно-аналитического центра, Российская академия образования (119121, Россия, г. Москва, ул. Погодинская, д. 8), ORCID: http://orcid.org/0000-0003-3633-5913, pavelkolyasnikov@gmail.com

Силаков Илья Николаевич, магистрант, кафедра автоматизированных систем управления, МИРЭА — Российский техноло-

гический университет (119454, Россия, г. Москва, пр. Вернадского, д. 78), ORCID: http://orcid.org/0000-0003-3985-0312, me@ileamare.ru
Ильин Дмитрий Юрьевич, аспирант, кафедра управления и моделирования систем, МИРЭА — Российский технологический университет (119454, Россия, г. Москва, пр. Вернадского, д. 78), ORCID: http://orcid.org/0000-0002-0241-2733, i@dmitryilin.com
Гусев Александр Алексеевич, аспирант, кафедра прикладной математики, Кубанский государственный университет (350040, Россия, г. Краснодар, ул. Ставропольская, д. 149), ORCID: http://orcid.org/0000-0003-2437-8537, alexandrgsv@gmail.com
Никульчев Евгений Витальевич, профессор, кафедра управления и моделирования систем, МИРЭА — Российский технологический университет (119454, Россия, г. Москва, пр. Вернадского, д. 78), доктор технический наук, ORCID: http://orcid.org/0000-0003-1254-9132, nikulchev@mail.ru

Все авторы прочитали и одобрили окончательный вариант рукописи.

About the authors:

Pavel V. Kolyasnikov, Analyst of the Laboratory for Monitoring the Efficiency and Quality of Scientific Research of the Information and Analytical Center, Russian Academy of Education (8 Pogodinskaya St., Moscow 119121, Russia), ORCID: http://orcid.org/0000-0003-3633-5913, pavelkolyasnikov@gmail.com
Ilya N. Silakov, Master's student, Department of Automated Control Systems, MIREA — Russian Technological University (78 Vernadsky Av., Moscow 119454, Russia), ORCID: http://orcid.org/0000-0003-3985-0312, me@ileamare.ru
Dmitry Yu. Ilin, Postgraduate Student, Department Control and System Modeling, MIREA — Russian Technological University (78 Vernadsky Av., Moscow 119454, Russia), ORCID: http://orcid.org/0000-0002-0241-2733, i@dmitryilin.com
Alexander A. Gusev, Postgraduate Student, Department of Applied Mathematics, Kuban State University (149 Stavropol St., Krasnodar 350040, Russia), ORCID: http://orcid.org/0000-0003-2437-8537, alexandrgsv@gmail.com
Evgeny V. Nikulchev, Professor, Department Control and System Modeling, MIREA — Russian Technological University (78 Vernadsky Av., Moscow 119454, Russia), Dr.Sci. (Engineering), ORCID: http://orcid.org/0000-0003-1254-9132, nikulchev@mail.ru

All authors have read and approved the final manuscript.