DOI 10.37882/2223–2966.2023.02–2.18

МОДЕЛЬ ГИБКОЙ ПРОГРАММНОЙ АРХИТЕКТУРЫ ДЛЯ УПРАВЛЕНИЯ РЕГИОНАЛЬНОЙ СИСТЕМОЙ ЗДРАВООХРАНЕНИЯ

A FLEXIBLE SOFTWARE ARCHITECTURE MODEL FOR MANAGING A REGIONAL HEALTHCARE SYSTEM

A. Konovalov
O. Romashkova

Summary. The article is devoted to the development of a client-server architecture, which is a platform for managing and servicing container workloads and services that form the basis for automating management processes in healthcare systems. The Kubernetes cluster is created using the calico pod network together with the main Kubelet, Kubeadm and Kubectl drivers. Secure Shell (SSH) protocol is used for secure shell, data management and authentication between client and server.

Keywords: data processing models, service orchestration software, healthcare system, automation of management processes, flexible client-server architecture.

Коновалов Артем Алексеевич
Аспирант, ГАОУ ВО «Московский городской педагогический университет (МГПУ)», г. Москва
bernadott94@yandex.ru
Ромашкова Оксана Николаевна
Доктор технических наук, профессор, ФГБОУ ВО «Российская академия народного хозяйства и государственной службы при Президенте РФ (РАНХиГС)», г. Москва
ox-rom@yandex.ru

Аннотация. Статья посвящена разработке клиент-серверной архитектуры, которая представляет собой платформу для управления и обслуживания контейнерных рабочих нагрузок и сервисов, образующих основу для автоматизации управленческих процессов в системах сферы здравоохранения. Кластер Kubernetes создается с помощью сети calico pod вместе с основными драйверами Kubelet, Kubeadm и Kubectl. Протокол Secure Shell (SSH) используется для защищенной оболочки, управления данными и аутентификации между клиентом и сервером.

Ключевые слова: модели обработки данных, программное обеспечение оркестровки сервисов, система здравоохранения, автоматизация процессов управления, гибкая клиент-серверная архитектура.

Распределенные вычисления — это область информационных технологий, которая занимается изучением распределенных информационных систем и баз данных. Такие системы имеют связь и координацию с каждым из своих узлов, которые взаимодействуют друг с другом для достижения общей цели: наиболее эффективной обработки данных для решения поставленной задачи. Возможности указанных систем способствуют внедрению гибких и эффективных приложений для управления деловыми процессами, в том числе в сфере здравоохранения [1, 2] .

Целью настоящего исследования является разработка клиент-серверной архитектуры, которая представляет собой платформу для управления и обслуживания контейнерных рабочих нагрузок и сервисов, образующих основу для автоматизации процессов управления [3]. Кластер Kubernetes создается с помощью сети calico pod вместе с основными драйверами Kubelet, Kubeadm и Kubectl. Протокол Secure Shell (SSH) используется для защищенной оболочки, управления данными и аутентификации между клиентом и сервером.

Особенности гибких программных архитектур

Kubernetes (иначе называемый k8s или "Kube") — это программное обеспечение оркестровки сервисов с открытым исходным кодом, которое автоматизирует значительную часть ручных процедур, связанных с транспортировкой, контролем и масштабированием контейнерных приложений. Таким образом появляется возможность объединять группы хостов, работающих под управлением операционной системы Linux, а также — продуктивно управлять ими. Kubernetes одинаково легко разворачивается как на локальных мощностях, так и в облаке. Также Kubernetes является идеальным средством для облегчения работы облачных приложений, требующих быстрого масштабирования, аналогичным постоянному потоку информации через Apache Kafka (брокер очередей). Контейнеры и этапы конфигурирования контейнеров, несомненно, представляют больше интереса по сравнению с обычной виртуализацией. Разделение завершается на уровне ядра без необходимости использования какого-либо стороннего программ-

ного обеспечения, поэтому контейнеры, несомненно, становятся все более эффективными, быстрыми и легкими. Docker в настоящее время является самой известной контейнерной средой. Docker предоставляет платформу для распараллеливаемых и распределенных вычислений контейнерных приложений. Но, тем не менее, возникла проблема координации и расписания запуска этих контейнеров. Kubernetes — это система для управления различными контейнерными приложениями (оркестрации) на нескольких хостах, предоставляющая базовые механизмы и функциональные возможности для развертывания, обслуживания и масштабирования различных и разных типов приложений [4] .

Виртуализация позволяет лучше использовать ресурсы как на физическом сервере, так и на облаке, обеспечивает лучшую адаптивность, поскольку приложение может быть эффективно развернуто или обновлено, что снижает затраты на аренду/закупку оборудования. Каждая виртуальная машина представляет собой пространство со своей операционной системой на которой можно запускать и выполнять программные приложения, аналогично обычным компьютерам. Механизм контейнерной среды действует частично как виртуальная машина (VM), но при этом исключаются все ненужные компоненты, которые не требуется для реализации сервиса, развернутого внутри контейнера. Соответственно, контейнеры намного легче стандартных приложений в рамках VM. Подобно виртуальной машине контейнер обладает своей файловой системой, процессором, памятью и процессорным пространством. Поскольку код сервисов, используемых контейнерами, используется только внутри контейнера, то это порождает феноменальную гибкость при разработке систем. Различные сервисы могут быть написаны на различных языках программирования, подходящих именно для конкретных поставленных задач.

Контейнеры стали необходимостью в современном мире из-за:

1. Корректировки трафика

Kubernetes может обнаружить контейнер, используя имя системы доменных имен (DNS) или используя их адрес. В случае, если объем трафика в контейнере слишком велик, Kubernetes может корректировать и адаптировать системный трафик до тех пор, пока он не станет устойчивым.

2. Распределения мощности

Kubernetes позволяет автоматически настроить распределение мощностей в зависимости от нагрузки, развернуть копии сервисов в случае критической нагрузки/падения контейнера и др.

3. Управления контейнеризацией

Полный контроль выполнения сервисов: возможность изменить состояние выполнения на идеальное состояние или общее состояние сервиса в зависимости от необходимости и требований.

В последние несколько лет наблюдается рост популярности облачных сервисов из-за их масштабируемости, гибкости. В то же время современные устройства теперь могут запускать контейнерные микросервисы, не требуя высокой мощности. Было проведено множество исследований по размещению сервисов в контейнерах и их развертыванию в кластерах. Кластерный контейнер, совместимый с Kubernetes, может улучшить распределение рабочей нагрузки по облаку и по обычным физическим кластерам [5, 6] .

Моделирование гибкой программной архитектуры для управления организациями в сфере здравоохранения

Целью разработки архитектуры является проектирование микросервисов, которые разворачиваются и масштабируются независимо друг от друга для создания единой системы управления региональным здравоохранением. Kubernetes предоставляет набор инструментов, на основе возможно получить доступ к методам развертывания, обслуживания, масштабирования и перезагрузки контейнеров, в которых размещается микросервис. Kubernetes выполняет абстракцию системы микросервисов и их обслуживание.

Для разработки данной архитектуры был выделен кластер, размещенный на локальной машине. Целью создания этого кластера было обеспечить простое развертывание и масштабируемость рабочих процессов управления организацией сферы здравоохранения. Пользователь, обладающий правами администратора, может беспрепятственно распределять ресурсы и предоставлять определенные интерфейсные функции (например, ведение журнала). Кластер нацелен на горизонтальное расширение, и в случае перегрузки какого-либо сервиса, Kuberntes создаст его копию. Также был использован Kubernetes networking — для придачи контейнерам свойства “виртуальных хостов”, которые могут взаимодействовать друг с другом через узлы, сочетая преимущества виртуальных машин с архитектурой микросервисов и контейнеризацией.

Для обоснования использования данного подхода в таблице 1 представлены результаты сравнения традиционных распределенных систем и Kubernetes с точки зрения ключевых факторов, таких как распределение

Таблица 1. Различия между традиционными распределенными системами и Kubernetes по ключевым факторам

Параметр	Традиционный подход	Подход Kubernetes
Распределение рабочей нагрузки	Ручное планирование, распределение и управление ресурсами	Основные, и рабочие узлы автоматически отслеживаются
Управление хранилищем	Для хранения данных используются центры обработки данных, что ставит под сомнение безопасность данных, а также ведет к росту финансовых затрат	Kubernetes позволяет монтировать хранилище в соответствии с выбором и потребностями пользователей. Пользователь может хранить данные в общедоступном, частном или гибридном режиме в зависимости от ценности данных
Масштабируемость	Проблема с масштабированием ресурсов в соответствии с потребностями и требованиями пользователя. Масштабирование ресурсов требует больших затрат труда и денежных вложений. Если один узел выйдет из строя, все развертывание кластера не сможет работать.	Kubernetes предлагает контейнеры для распределения работы, возможно масштабировать ресурсыв зависимости от задач. Масштабирование ресурсов позволит правильно распределить и распределить ресурсы работы. Если один контейнер выходит из строя, Kubernetes автоматически перезапускает контейнер, чтобы кластер не был поврежден
Системные требования	Повышенные требования к ОЗУ и частоте процессора в зависимости от ОС VM	Только ресурсы необходимые для выполнения сервиса

рабочей нагрузки, управление хранилищем, масштабируемость, системные требования.

В данной архитектуре используется несколько основных компонентов Kubernetes: Kube adm используется для обслуживания и питания всего кластера, Kubectl для управления кластерами. Kubelet — механизмы, которые выполняются на каждом узле и позволяет передавать информацию между каждым из узлов и в зависимости от запросов.Secure Socket Shell — протокол безопасности на базе UNIX, используемый для доступа к любой из ближайших или удаленных машин. Чтобы сделать возможным обмен данными между различными модулями через защищенную оболочку, используется протокол SSH. Таким образом, чтобы получить доступ к модулям, это можно сделать через сервис, который является доверенной точкой входа. Было проведено исследование возможностей Pod (базовая единица для запуска и управления приложениями: один или несколько контейнеров), было подтверждено, что pod можно масштабировать вверх и вниз в зависимости от использования и требований пользователя.

Если пользователь хочет большей масштабируемости, Kubernetes предоставляет возможность масштабирования до необходимого количества ресурсов узла.

Как правило, для облачных сред используется модель виртуализации с поддержкой контейнеров. Эта модель использует Docker вместе с Kubernetes для контейнерной системы Docker с несколькими хостами. Важным аспектом такой среды является то, что Kubernetes должен отслеживать требования к ресурсам и/или уровень потребления запущенных приложений. Он должен динамически корректировать ресурсы, выделяемые контейнерам, для обеспечения оптимальной производительности. На данный момент Kubernetes предоставляет доступный процесс динамического распределения ресурсов, который ограничен только потреблением процессора.

Планирование выполнения задач — важный аспект, который необходимо решить, чтобы сбалансировать рабочую нагрузку в кластере.

Был выбран круглый алгоритм планирования Робина для решения данной проблемы. Этот метод использует пакетный детектор для оценки рабочей нагрузки и выполнения любых необходимых мер. В данном алгоритме каждая готовая задача выполняется по очереди только в циклической очереди в течение ограниченного промежутка времени. Этот алгоритм также предлагает выполнение процессов без простоя. Предлагаемая система будет содержать главный узел, который будет подключен к нескольким клиентским узлам, которые обеспечивают уникальные функциональные возможности. Пользователи смогут воспользоваться этими функциями через главный узел. Система будет поддерживать четыре приложения (Node JS, Angular JS, Mongo DB), которые будут распределены по двум клиентским системам, развернутым на разных портах клиентов

На рисунке 1 показана схема архитектуры Kubernetes, использующая архитектуру «клиент-сервер».

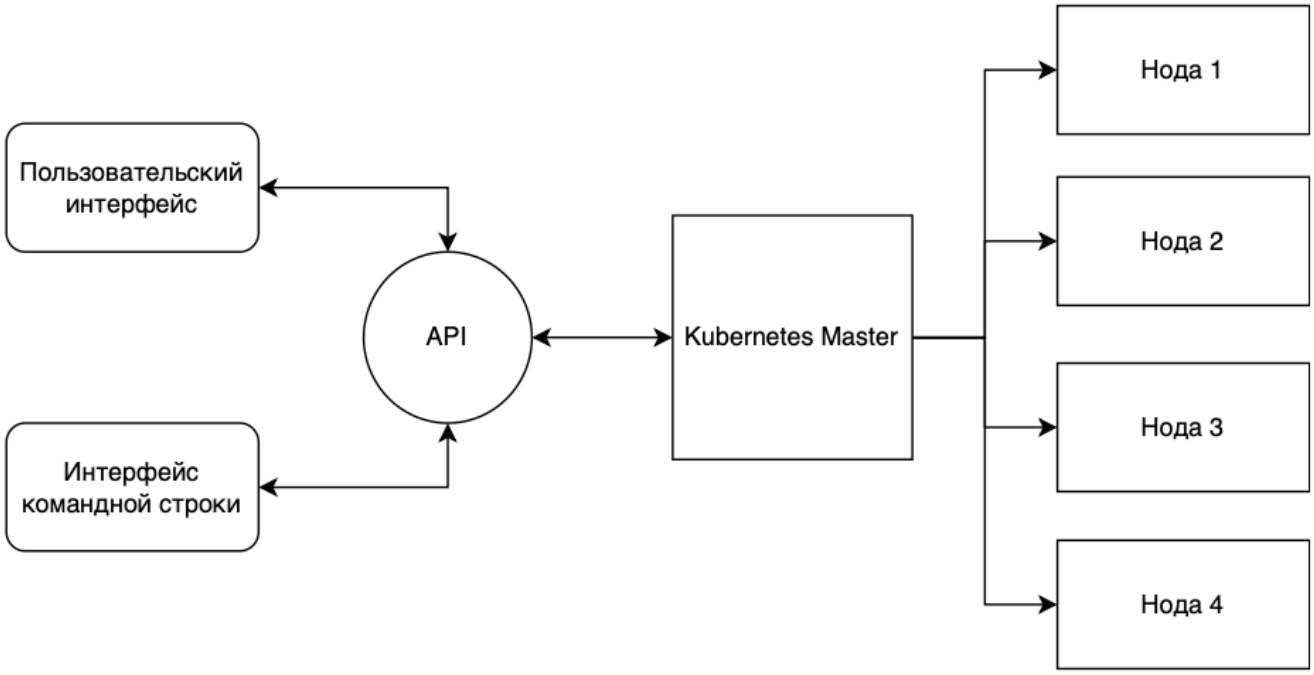

Рис. 1. Схема архитектуры Kubernetes

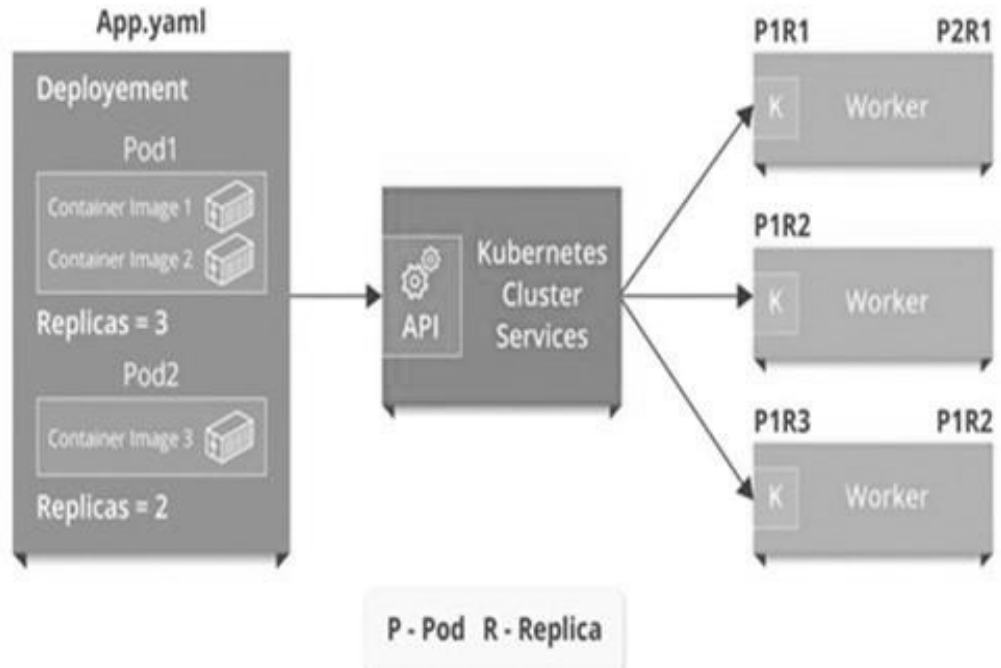

Рис. 2. Диаграмма модуля Kubernetes

Существует мастер, установленный на одной машине, и узел — на отдельных виртуальных машинах.

На рисунке 2 представлена диаграмма модуля, показывающая, что рабочая нагрузка главного модуля была разделена на рабочие узлы, и в зависимости от требований модуля для каждого из рабочих узлов разделена работа между контейнерами.

На рисунке 3 представлен пример web-приложения управления региональной системой здравоохранения, развернутого в рамках микросервисной архитектуры.

Надежность медицинского поставщика

Год 2022 Месяц Январь Рассчитать

КРП	Мин	Эталон	Макс	Вывод
Финансирование	0.4895	1.1830	129.1936	Норма
Прогноз банкротства	0,0949	0,5458	114,265	Норма
Материальный капитал	0,0356	0,3254	0,9658	Норма
Геополитический риск	-0,1252	0,0654	0,7523	Норма
Логистические затраты	0,0147	0,5476	1,1254	Норма
Рейтинг медицинских марок	0,1963	2,3265	4,2354	Норма
Прибыль и доходы	0,8029	0,4583	1,2684	Норма

Рис. 3. Пример web-приложения управления региональной системой здравоохранения

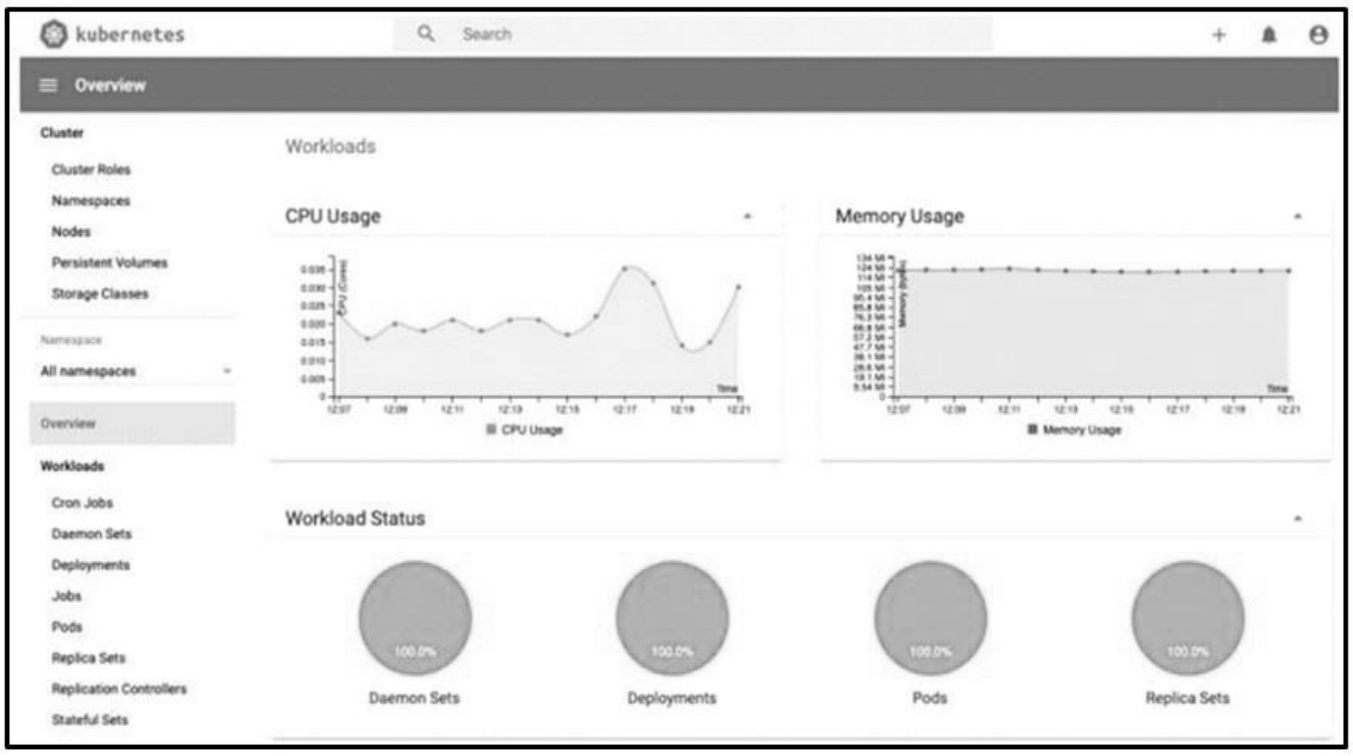

Рис. 4. Панель управления кластером Kubernetes

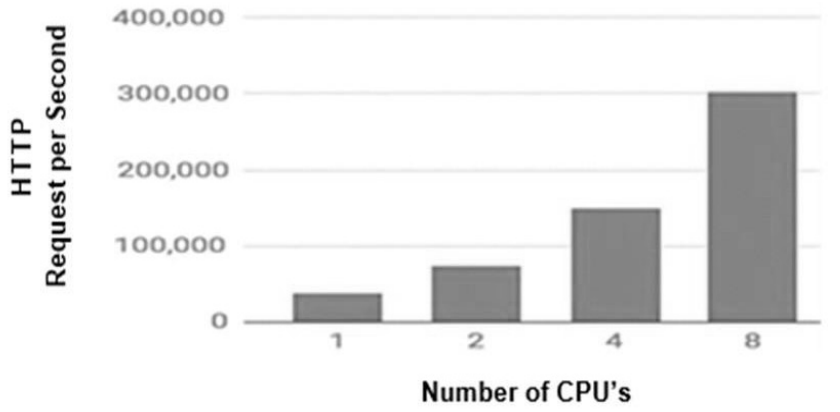

Рис. 5. Сводка Kubernetes о числе запросов в секунду в зависимости от числа ядер процессора

На рисунке 4 показана панель управления кластером Kubernetes. Отображена структура организации контейнеров, потребляемые мощности, состояние узлов.

На рисунке 5 показана сводка Kubernetes о числе запросов в секунду в зависимости от числа ядер процессора, что позволяет динамически подбирать оптимальные параметры контейнера в зависимости от планируемой нагрузки.

Асинхронное вычисление множества функциональных блоков с использованием платформы оркестровки Kubernetes повышает эффективность решения любых задач (в том числе управления организационными процессами), повышает производительность системы и дает возможность ее горизонтального расширения.

Благодаря использованию Kubernetes получена гибкая архитектура, апробированная в рамках приложения для управления региональной системой здравоохранения [7, 8]. Полученное решение позволяет отслеживать изменения в реальном масштабе времени, прогнозировать потребности в выделяемых вычислительных мощностях, а также перераспределять нагрузку на наименее загруженные модули с высоконагруженных.

ЛИТЕРАТУРА

1. Коновалов А.А., Ромашкова О.Н. Модели бизнес-процессов по осуществлению рейтингового оценивания деятельности организаций медико-социального профиля // Современная наука: актуальные проблемы теории и практики. Серия: Естественные и технические науки. 2021. № 1. С. 83–96.
2. Заболотникова В.С., Ромашкова О.Н. Информационная управленческая система для налоговой службы// Современная наука: актуальные проблемы теории и практики. Серия: Естественные и технические науки. 2017. № 6. С. 27–32.
3. Ромашкова О.Н., Самойлов В.Е. К определению качества пакетной передачи речи в сетях подвижной связи //Наукоемкие технологии в космических исследованиях Земли. 2017. Т. 9. № 3. С. 39–44.
4. Ромашкова О.Н., Пономарева Л.А., Василюк И.П. Применение инфокоммуникационных технологий для анализа показателей рейтинговой оценки вуза // В книге: Информационно-телекоммуникационные технологии и математическое моделирование высокотехнологичных систем. Материалы Всероссийской конференции с международным участием. 2018. С. 65–68.
5. Ромашкова О.А., Моргунов А.И. Информационная система для оценки результатов деятельности общеобразовательных организаций г. Москвы// Вестник Российского университета дружбы народов. Серия: Информатизация образования. 2015. № 3. С. 88–95.
6. Пономарева Л.А., Ромашкова О.Н., Василюк И.П. Алгоритм оценки эффективности работы кафедр университета для управления его рейтинговыми показателями // Вестник Рязанского государственного радиотехнического университета. 2018. № 64. С. 102–108.
7. Ромашкова О.Н., Федин Ф.О., Фролов П.А. Применение нейросетевых технологий для проверки благонадежности контрагентов сетевой торговой компании // Современная наука: актуальные проблемы теории и практики. Серия: Экономика и право. 2018. № 7. С. 126.
8. Ромашкова О.Н., Чискидов С.В. Методологии и технологии проектирования информационных систем // Учебно-методическое пособие / Москва, 2020. Часть 1.

Журнал «Современная наука: актуальные проблемы теории и практики»