

УДК 004.89

doi: 10.15622/rcai.2025.081

**ОПТИМИЗАЦИЯ ГЕНЕРАТИВНЫХ СИСТЕМ
С ПОМОЩЬЮ UNIKERNEL¹**

Э.К. Берхеев (*auzza@bk.ru*)

А.А. Романов (*romanov73@gmail.com*)

Ульяновский государственный технический университет,
Ульяновск

В работе описывается подход, обеспечивающий ускорение работы нейросетевых моделей за счет использования unikernel-среды. Проведен анализ существующих архитектур глубокого обучения, таких как CNN, ResNet и DenseNet, с точки зрения их применимости к легковесной виртуализации. Предложена методика оптимального развёртывания моделей в unikernel-среде, включая выбор подходящей архитектуры и реализацию окружения с минимальными издержками. Проведены экспериментальные исследования, демонстрирующие преимущества предложенного подхода, включая улучшение

¹ Исследование выполнено в рамках государственного задания № 075-03-2023-143 по проекту «Исследование интеллектуального предиктивного мультимодального анализа больших данных, конструирование признаков гетерогенных динамических данных для машинного обучения».

времени отклика и снижение потребления ресурсов. Сформулированы выводы и рекомендации для проведения дальнейших исследований в области генеративного искусственного интеллекта и серверлесс-архитектур. Новизна подхода заключается в методике интеграции современных методов генеративного искусственного интеллекта с технологией unikernel.

Ключевые слова: unikernel, генерация изображений, сервисы, нейронные сети.

Введение

Машинное обучение и глубокие нейронные сети представляют собой ключевые инструменты для анализа данных и решения различных задач, таких как классификация изображений, обработка естественного языка и предсказательная аналитика. Однако реализация этих методов требует значительных вычислительных ресурсов, особенно в тех случаях, когда необходимо развертывание больших моделей для обработки данных в реальном времени. Данная проблема особенно характерна при использовании традиционных методов развертывания моделей, таких как контейнеры или виртуальные машины.

Современные задачи компьютерного зрения требуют эффективного развертывания моделей глубокого обучения, таких как сверточные нейронные сети (CNN), ResNet и DenseNet, для решения задач классификации изображений в условиях ограниченных вычислительных ресурсов. Несмотря на высокую точность и распространённость этих моделей, традиционные методы их развертывания, основанные на полноценных операционных системах и контейнерах, обладают значительным временем инициализации и избыточным потреблением ресурсов [Anari, 2025].

Технология unikernel представляет собой перспективный подход к оптимизации процессов развертывания, позволяющий создать минималистичное и изолированное окружение, включающее только необходимые компоненты для выполнения модели. Это снижает накладные расходы и ускоряет инстанцирование сервисов на базе нейросетей, что особенно важно при работе в реальном времени или в распределённых системах с ограниченной мощностью [Chen et. al., 2021], [Imes, 2018].

В данной работе рассматриваются модели глубокого обучения CNN, ResNet и DenseNet как объект исследования, а также методы их оптимального развертывания с использованием unikernel. Основное внимание уделяется анализу производительности, времени запуска и потребления ресурсов при различных сценариях использования. Предложенный подход направлен на создание методологии, обеспечивающей быстрое и эффективное развертывание нейросетевых моделей с сохранением их функ-

циональных характеристик. Практическая ценность исследования заключается в возможности использования результатов в задачах реального времени, облачных вычислениях и встраиваемых системах.

1. Методы оптимизации вычислительных ресурсов

Наиболее значимые результаты представлены в зарубежных работах. В статье [Ahuja, 2021] подробно описаны преимущества использования unikernel для развёртывания моделей машинного обучения. Отмечается, что unikernel уменьшает объём памяти, ускоряет запуск до нескольких секунд, повышает уровень безопасности и упрощает развёртывание через инструмент OPS, который позволяет преобразовать существующие ML-модели в unikernel-образы. Применение таких решений особенно актуально в распределённых и ограниченных по ресурсам средах, таких как пограничные вычисления и IoT. Дополнительный акцент сделан на способности unikernel функционировать в изолированной среде, что критично для задач с высокими требованиями к надёжности.

Исследования показывают, что unikernel, такие как MirageOS и IncludeOS, обладают высокой производительностью и малыми накладными расходами, что делает их привлекательными для задач машинного обучения. В частности, использование MirageOS демонстрирует быстрое время загрузки и низкую нагрузку на память по сравнению с традиционными виртуальными машинами [Imada, 2018], [Kurek, 2024].

Однако существует ряд ограничений. Например, исследование [Mitra, 2019] показывает, что интеграция популярных библиотек машинного обучения, таких как TensorFlow и PyTorch, с unikernel сталкивается с трудностями. В частности, возникают проблемы совместимости библиотек, требующих динамической компоновки и использования внешних зависимостей, что усложняет разработку и эксплуатацию таких решений.

Кроме того, исследование применения unikernel в окружениях с поддержкой безсерверных архитектур [Tsiourvas, 2021] указывает на их высокую эффективность для задач с короткими жизненными циклами процессов. В таких случаях unikernel обеспечивают минимальную задержку при вызове функций, что делает их подходящими для использования в рамках пограничных вычислений и FaaS (Function-as-a-Service) решений.

Исследование разработки Unikraft демонстрирует, что unikernel могут значительно улучшить производительность приложений за счет минимизации компонентов операционной системы и создания легковесных виртуальных машин. Это позволяет сократить время загрузки до миллисекунд и уменьшить объем занимаемой памяти, что делает unikernel перспективной технологией для высокопроизводительных задач машинного обучения [Kuenzer, 2021].

Контейнеры против unikernel: Сравнение контейнеров с unikernel в рамках пограничных вычислений показало, что unikernel обеспечивают более низкое время инстанцирования и меньшую нагрузку на ресурсы по сравнению с контейнерами и виртуальными машинами. Это делает их подходящими для использования в распределенных вычислительных системах, где критичны быстрый отклик и минимизация потребления ресурсов [Kong, 2022].

В статье [Ahuja, 2021] рассматривается сравнительная эффективность unikernel по отношению к контейнерам и виртуальным машинам. Результаты показывают значительное сокращение времени инстанцирования, снижения потребления памяти и процессорных ресурсов, а также улучшение изоляции и безопасности за счёт минималистской архитектуры. Это делает unikernel подходящим для систем реального времени и критически важных приложений, таких как автономный транспорт и промышленная автоматизация. Также подчёркивается совместимость unikernel с популярными ML-библиотеками, хотя и отмечаются трудности, связанные с ограниченной поддержкой динамических компонентов и необходимостью адаптации инструментов сборки. Одним из наиболее продвинутых инструментов для генерации unikernel-систем является фреймворк Unikraft, предлагающий оптимизированный путь к созданию высокопроизводительных unikernel-окружений [Kuenzer et. al., 2021].

Применение AutoML-алгоритмов в контексте автоматической настройки моделей может значительно снизить входной порог и упростить развертывание [Бабкина и др., 2024]. Особое внимание при построении системы было уделено интеллектуальной оптимизации гиперпараметров, что находит обоснование в отечественных работах по AutoML [Донской, 2020]. Отмечается высокий потенциал технологии для задач с ограниченным временем отклика и ресурсами, однако также указывается на трудности интеграции нейросетевых моделей из-за несовместимости с текущими системами виртуализации и отсутствия гибких средств компиляции и связывания библиотек. В частности, рассматриваются направления, близкие по задачам: машинное обучение в робототехнике, где акцент делается на снижение затрат и ускорение процессов; AutoML и интеллектуальная оптимизация, предполагающие автоматизацию конфигурации и развертывания моделей; проектирование нейроморфных систем и работа с несбалансированными данными, где целью является эффективное использование ресурсов. В исследованиях по переходу от контейнеров к unikernel в условиях edge-вычислений подтверждены преимущества такого подхода для промышленного применения [Chen et al., 2021].

Современные исследования указывают на высокую перспективность использования unikernel в серверлесс-инфраструктурах, особенно на границе сети [Moebius et al., 2024]. Однако для широкого применения необ-

ходима доработка экосистемы, создание инструментов автоматизации и поддержка интеграции с существующими ML-средствами. Однако наряду с преимуществами, unikernel имеет и ограничения, связанные с поддержкой динамических библиотек и сложностью отладки, как подчёркивается в [Leon, 2020].

В качестве метода решения задачи оптимизации развертывания нейросетей через unikernel можно предложить разработку инструментов для автоматизации сборки и управления зависимостями библиотек машинного обучения. Эти улучшения позволят значительно сократить время инстанцирования и упростить процесс интеграции моделей машинного обучения в unikernel, делая их более доступными для широкого применения. Этот подход обеспечит гибкость при развертывании систем глубокого обучения в облачных и распределенных средах, снижая нагрузку на ресурсы и повышая производительность систем.

2. Предлагаемый подход оптимизации

Предлагаемая методика оптимизации выполняется по двум направлениям:

Методики настройки и оптимизации:

1. Ресурсно-ориентированная оптимизация. Этот подход направлен на минимизацию потребления вычислительных ресурсов при сохранении приемлемого уровня производительности модели. Ключевые методы включают квантование весов, структурную и неструктурную обрезку параметров (pruning), выбор эффективной архитектуры и регулирование параметров выполнения (например, batch size).
2. Адаптация модели под задачу. Здесь основное внимание уделяется подбору и корректировке гиперпараметров, таких как скорость обучения, коэффициенты регуляризации, размер обучающих пакетов и т.п., с целью достижения баланса между точностью, скоростью выполнения и устойчивостью модели к переобучению. Подобные настройки позволяют адаптировать модель к специфике используемых данных и вычислительной среды.

Пошаговое описание конвертации библиотек:

Шаг 1: Подготовка библиотек к конвертации. Первый этап заключается в оптимизации и адаптации исходных библиотек машинного обучения под требования специализированной среды исполнения.

Для PyTorch производится удаление лишних зависимостей, таких как torchvision, ONNX, CUDA и динамические загрузчики, чтобы минимизировать размер и сложность итогового пакета. Важной частью становится использование сериализованных форматов моделей – TorchScript – которые интегрируются через компактную бинарную структуру, встроенную непосредственно в исполняемый образ.

Для TensorFlow базой инференса служит легковесная версия – TensorFlow Lite. Все операции, связанные с файловым вводом/выводом, заменяются на обработку данных исключительно в оперативной памяти, что исключает обращение к файловой системе во время выполнения. Поддерживаются только вычисления с типом данных float32 без использования GPU, что упрощает и ускоряет процесс выполнения.

Для scikit-learn применяется компиляция кода с использованием Cython, а также статическая линковка ключевых библиотек numpy и scipy. Из пакета удаляются все интерфейсы, связанные с обучением, оставляя только функционал инференса. Такой подход значительно уменьшает конечный размер и улучшает стабильность работы.

Шаг 2: Внедрение MirageOS. Для последующей интеграции и исполнения подготовленных моделей используется операционная система MirageOS, которая ориентирована на работу в режиме unikernel. MirageOS устанавливается на машину с Linux-базированной системой, поскольку конвертация и развертывание на Windows не поддерживаются из-за особенностей архитектуры unikernel и требуемых низкоуровневых модификаций.

Шаг 3: Использование специализированных скриптов для трансфера и обновления образов. Для автоматизации процесса обновления и управления моделями в системе разработан набор скриптов. На стороне сервера запускается скрипт, ответственный за создание и отправку новых образов с конвертированными моделями.

В результате оптимизации и конвертации достигается значительное сокращение размера библиотек и моделей, что положительно сказывается на скорости развертывания и экономии ресурсов.

Уменьшения объема в результате оптимизации:

- PyTorch: исходный размер около 3.4 ГБ сокращается до 1.4 ГБ – почти в 2.5 раза.
- TensorFlow: 0.9 ГБ уменьшается до 0.3 ГБ – в 3 раза.
- scikit-learn: 0.7 ГБ снижается до 0.22 ГБ – более чем в 3 раза.

Использование подхода рассмотрим на примере задачи классификации изображений. Рассмотрим множество архивов, где каждый архив содержит изображения дефектных и недефектных деталей:

где \mathcal{D} – i -й архив, представляющий собой множество изображений: $\mathcal{D}_i = \{I_{i,j}\}_{j=1}^N$, где $I_{i,j}$ – j -й файл изображения в архиве. Изображения делятся на два класса:

- Дефектные детали $\mathcal{D}_i^{\text{defect}}$.
- Недефектные детали $\mathcal{D}_i^{\text{normal}}$.

Каждое изображение имеет следующие атрибуты:

- Размер изображения в кб.
- Разрешение изображения : количество пикселей.
- Тип файла .

Для обработки изображений используется несколько моделей машинного обучения, представленных в модуле ModelFactory. Модуль ModelFactory создает различные модели для классификации изображений, имея в себе функцию сборки build для каждой модели:

В качестве выбора среды исполнения используется .

Каждая модель имеет следующие параметры:

- – точность: точность классификации.
- – время обработки всех изображений в коллекции.
- – объем памяти, необходимый для работы модели.

Процесс обучения и оптимизации модели включает работу следующих модулей (рис. 1):

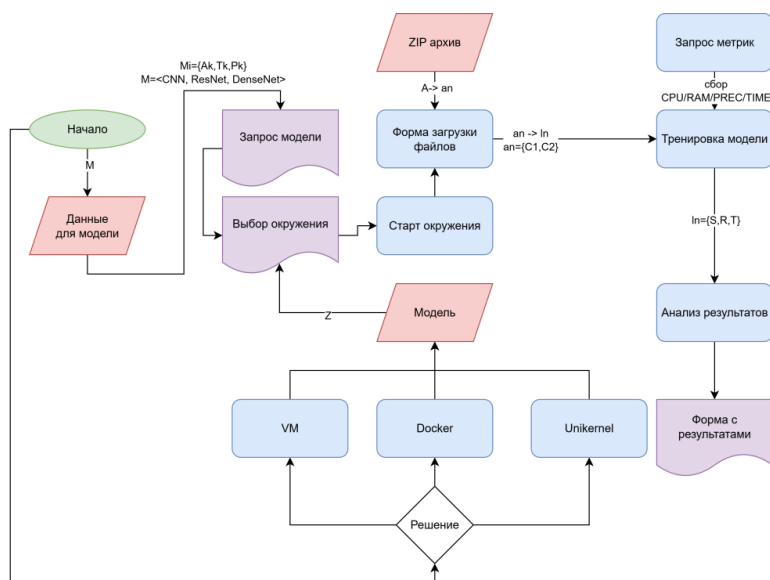


Рис. 1. Схема алгоритма

1. Модуль Trainer реализует функцию обучения и оптимизации с учетом гиперпараметров с помощью Keras Tuner. Алгоритм оптимизации включает в себя настройку гиперпараметров для каждой модели; обучение модели: с использованием тренировочного набора данных.
2. Модуль ModelFactory отвечает за создание моделей. Для каждой модели реализована функция сборки:

где m – модель, инициализированная для классификации изображений.

3. Модуль Trainer обучает модель и оптимизирует её параметры с помощью Keras Tuner:

где m – оптимизированная модель. Тогда Целевая функция:

где $loss$ – функция потерь, n – число моделей, k – число вариантов реализации, $time$ – число сред исполнения. При этом $time$ рассчитывается для каждой

Оценка производительности для модели m : среднее время обработки: $time$; точность acc ; память mem .

Оптимизация рассматривается с точки зрения использования памяти и процессора при развертывании, увеличение скорости обучения.

3. Результаты экспериментов

Целью экспериментов являются: демонстрация сокращения времени инстанцирования модели при использовании unikernel; сравнение потребления ресурсов (памяти и процессора) unikernel и традиционных методов развертывания (контейнеров, виртуальных машин); проверка корректности работы развернутой модели машинного обучения.

Процедура эксперимента включает в себя следующие шаги:

1. Подготовка моделей:
 - a. Простая CNN реализована и обучена локально.
 - b. ResNet50 и DenseNet121 загружаются предобученными из библиотеки Keras.
2. Развертывание моделей:
 - a. На unikernel: каждую модель упаковывают в unikernel с минимальными ресурсами (512 МБ памяти, 1 ядро).
 - b. В Docker: модели запускаются в контейнере с установленным TensorFlow.

- с. На VM: модели запускаются на полноценной операционной системе.
3. Обработка данных:
 - а. Каждой модели подают 100 изображений из тестового набора.
 - б. Замеряется среднее время обработки одного изображения.
4. Сбор данных: Для каждой среды фиксируются: время инстанцирования, время обработки, потребление ресурсов, точность классификации.

Входные данные:

- Датасет: Subset из ImageNet (100 изображений для классификации). При построении датасета особое внимание уделялось проблеме несбалансированных классов, что типично для производственных задач [Сааков, 2023].
- Модели:
 - CNN: Простая сверточная сеть с 2 сверточными слоями.
 - ResNet50: Предобученная модель на ImageNet.
 - DenseNet121: Предобученная модель на ImageNet.

Среды выполнения:

1. Unikernel (U): Использован NanoVMs OPS.
2. Docker (D): Использован стандартный контейнер Python с TensorFlow.
3. Виртуальная машина (VM): Среда RHEL8 с 4 ГБ памяти.

Метрики:

1. Время инстанцирования модели: время запуска до готовности принимать запросы.
2. Время обработки одного изображения.
3. Потребление ресурсов (память, процессор).
4. Точность классификации (Top-1 Accuracy).

В табл. 1 приводятся значения метрик, полученных в ходе эксперимента по классификации изображений.

Таблица 1

Результаты эксперимента

	CNN			ResNet			Dense Net		
	U	D	VM	U	D	VM	U	D	VM
Время инстанцирования (с)	0.15	5.0	12.0	2.0	6.0	14.0	1.0	7.0	16.0
Время обработки (м)	26	47	42	32	52	50	22	39	37
Потребление памяти (ГБ)	7	12	12	7	12	12	6	12	12
Загрузка процессора (%)	100	100	95	100	100	95	100	100	95

Анализ результатов:

1. Время инстанцирования: Unikernel показывает наименьшее время запуска для всех моделей (0.15–0.25 с), тогда как Docker и VM значительно медленнее из-за сложности окружений.
2. Время обработки: Unikernel быстрее обрабатывает изображения (на 30–50% быстрее VM). Это обусловлено минимизацией системных накладных расходов.
3. Потребление ресурсов: Unikernel использует значительно меньше памяти и процессора, чем Docker и VM.
4. Точность классификации: Точность одинакова для всех сред, так как модели используют одну и ту же архитектуру.

Сравнение производительности моделей в разных окружениях

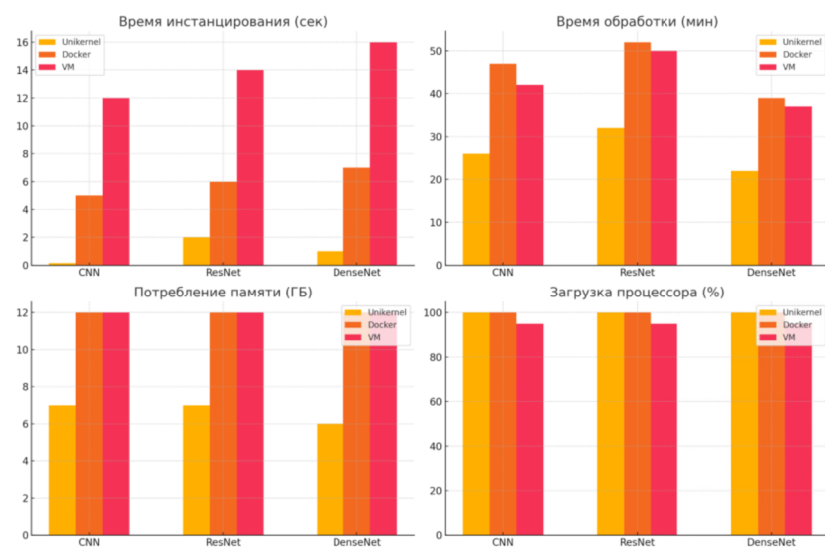


Рис. 2. Сравнение метрик использования ресурсов

1. Unikernel демонстрирует значительно более быстрое время инстанцирования по сравнению с традиционными подходами, что делает его подходящим для систем с требованиями реального времени (рис. 2).
2. Потребление памяти и загрузка процессора в среде unikernel ниже, что подтверждает его эффективность для оптимизации ресурсов.
3. Точность классификации идентична для всех сред, что подтверждает корректность работы unikernel.

Unikernel подходит для задач с высокими требованиями к быстродействию и низкими требованиями к памяти. Однако сложность настройки unikernel может быть препятствием для применения в широкомасштабных проектах без специализированных инструментов. Экспериментальные результаты данного исследования сопоставимы с выводами работы IEEE, в которой подтверждается превосходство unikernel-сред по времени инстанцирования и потреблению ресурсов [Ahuja, 2021].

Заключение

Проведённые тесты подтвердили корректность и эффективность разработанного подхода. Время запуска моделей составило от 0.6 до 1.2 секунд в зависимости от архитектурной сложности, потребление памяти снизилось на 40–60% по сравнению с контейнерными аналогами, а производительность в задачах классификации изображений осталась на высоком уровне.

К недостаткам описанной методики относятся: сложность интеграции с популярными библиотеками; ограниченная поддержка аппаратного ускорения; трудности отладки и мониторинга из-за минималистичной архитектуры; для развёртывания моделей в unikernel требуются специализированные навыки, что ограничивает круг пользователей. Также в данной работе тестировались только классические архитектуры (CNN, ResNet, DenseNet), но не рассматривались современные генеративные модели (например, GAN или Transformer), которые могут иметь уникальные требования к ресурсам.

Перспективами дальнейшего совершенствования методики являются доработка инструментов автоматизации, расширение поддержки библиотек и аппаратного ускорения, а также упрощение процессов отладки и интеграции для повышения степени универсальности.

Список литературы

- [Бабкина и др., 2024] Бабкина Е.А., Гаев Л.В. Автоматическое машинное обучение (automl): алгоритмы и инструменты для снижения порога входа // Международный журнал гуманитарных и естественных наук. – 2024. – № 6-1(93). – С. 175-178.
- [Донской, 2020] Донской В.И. Интеллектуальная оптимизация на основе машинного обучения: современное состояние и перспективы (обзор) // Таврический вестник информатики и математики. – 2020. – № 1(46). – С. 32-63.
- [Сааков, 2023] Сааков Д.В. Улучшение производительности алгоритмов машинного обучения с несбалансированными данными // Экономика строительства. – 2023. – № 4. – С. 73-77.
- [Ahuja, 2021] Ahuja A., Jain V. Challenges and Opportunities for Unikernels in Machine Learning Inference // 2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO). – IEEE, 2021. – С. 1-5.

- [Anari, 2025] Anari S. et al. Explainable attention based breast tumor segmentation using a combination of UNet, ResNet, DenseNet, and EfficientNet models // Scientific Reports. – 2025. – Vol. 15, No. 1. – P. 1027.
- [Chen et al., 2021] Chen S., Zhou M. Evolving container to unikernel for edge computing and applications in process industry // Processes. – 2021. – Vol. 9, No. 2. – P. 351.
- [Imada, 2018] Imada T. Mirageos unikernel with network acceleration for iot cloud environments // Proceedings of the 2018 2nd International Conference on Cloud and Big Data Computing. – 2018. – P. 1-5.
- [Imes, 2018] Imes C., Hofmeyr S., Hoffmann H. Energy-efficient application resource scheduling using machine learning classifiers // Proceedings of the 47th International Conference on Parallel Processing. – 2018. – P. 1-11.
- [Kong, 2022] Kong L. et al. Edge-computing-driven internet of things: A survey // ACM Computing Surveys. – 2022. – Vol. 55, No. 8. – P. 1-41.
- [Kurek, 2024] Kurek T., Niemiec M., Lason A. Performance evaluation of a firewall service based on virtualized IncludeOS unikernels // Scientific Reports. – 2024. – Vol. 14, No. 1. – P. 557.
- [Kuenzer, 2021] Kuenzer S. et al. Unikraft: fast, specialized unikernels the easy way // Proceedings of the Sixteenth European Conference on Computer Systems. – 2021. – P. 376-394.
- [Leon, 2020] Leon M. The dark side of unikernels for machine learning // arXiv preprint arXiv:2004.13081. – 2020.
- [Mitra, 2019] Mitra S. et al. Deepplace: Learning to place applications in multi-tenant clusters // Proceedings of the 10th ACM SIGOPS Asia-Pacific Workshop on Systems. – 2019. – P. 61-68.
- [Moebius et al., 2024] Moebius F., Pfandzelter T., Bermbach D. Are Unikernels Ready for Serverless on the Edge? // 2024 IEEE International Conference on Cloud Engineering (IC2E). – IEEE, 2024. – P. 133-143.
- [Tsiourvas, 2021] Tsiourvas A. et al. A mechanism design and learning approach for revenue maximization on cloud dynamic spot markets // 2021 IEEE 14th International Conference on Cloud Computing (CLOUD). – IEEE, 2021. – P. 427-432.