Вестник РУДН. Серия: Инженерные исследования
RUDN Journal of Engineering Research

2026;27(2):193–202

ISSN 2312-8143 (Print); ISSN 2312-8151 (Online)
journals.rudn.ru/engineering-researches

DOI: 10.22363/2312-8143-2026-27-2-193-202
EDN: KYMATZ

Научная статья / Research article

Сравнительный анализ объектных систем хранения данных с открытым исходным кодом

С.А. Тast✉, **М.С. Мосева**

Московский технический университет связи и информатики, *Москва, Российская Федерация*
✉ s.a.tast@edu.mtuci.ru

История статьи

Поступила в редакцию: 12 декабря 2025 г.
Доработана: 13 февраля 2026 г.
Принята к публикации: 15 февраля 2026 г.

Заявление о конфликте интересов

Авторы заявляют об отсутствии конфликта интересов.

Заявление об использовании технологий искусственного интеллекта

При создании настоящей статьи технологии генеративного искусственного интеллекта не использовались.

Заявление о доступности данных

Все данные, полученные в ходе этого исследования, включены в опубликованную статью.

Аннотация. Объектом исследования в работе выступают открытые объектные системы хранения данных — Ceph, OpenStack Swift и MinIO — рассматриваемые в контексте их применимости для безопасного и эффективного хранения мультимедиа данных в корпоративных коммуникационных системах. Метод исследования основан на сравнительном анализе ключевых характеристик: архитектурных особенностей, производительности при работе с различными типами файлов (включая мелкие и крупные мультимедиа объекты), уровня отказоустойчивости, простоты развертывания и совместимости с распространенными протоколами, такими как S3 API. Результаты анализа показали, что Ceph обеспечивает высокую отказоустойчивость и универсальность за счет поддержки блочных, объектных и файловых интерфейсов, что делает его оптимальным для масштабных инфраструктур и контейнерных сред. OpenStack Swift отличается простотой интеграции в экосистему OpenStack, однако демонстрирует сниженную производительность при обработке множества мелких файлов. MinIO выделяется как легковесное и высокопроизводительное решение с нативной поддержкой S3 API, что определяет его как наиболее подходящее для рассматриваемой задачи. Подчеркивается также необходимость предварительного анализа доминирующих типов мультимедийного контента при проектировании хранилища для достижения максимальной эффективности.

Ключевые слова: хранение данных, файловая система, объектная структура, протокол s3, Ceph, MinIO, Openstack Swift

Вклад авторов

Тast С.А. — реализация исследования, сбор и обработка материалов, написание текста; *Мосева М.С.* — концепция и дизайн исследования, экспертная поддержка. Оба автора ознакомлены с окончательной версией статьи и одобрили ее.

Для цитирования

Тast С.А., Мосева М.С. Сравнительный анализ объектных систем хранения данных с открытым исходным кодом // Вестник Российского университета дружбы народов. Серия: Инженерные исследования. 2026. Т. 27. № 2. С. 193–202. http://doi.org/10.22363/2312-8143-2026-27-2-193-202 EDN: KYMATZ

Comparative Analysis of Open-Source Object Data Storage Systems

Sergey A. Tast✉, **Marina S. Moseva**

Moscow Technical University of Communications and Informatics, *Moscow, Russian Federation*
✉ s.a.tast@edu.mtuci.ru

Article history
Received: December 12, 2025
Revised: February 13, 2026
Accepted: February 15, 2026

Conflicts of interest
The authors declare that there is no conflict of interest.

Statement on the use of artificial intelligence technologies.
No generative artificial intelligence technologies were used in the creation of this article.

Data availability
All data obtained during this study are included in the published article.

Abstract. This article examines open-source object storage systems, namely Ceph, OpenStack Swift, and MinIO, with respect to their applicability to *secure and efficient storage* of multimedia data in enterprise communication systems. The study is based on a comparative analysis of key characteristics: architectural features, performance when working with various file types, including small and large multimedia objects, fault tolerance, ease of deployment, and compatibility with common protocols such as the S3 API. The results of the analysis that Ceph provides high fault tolerance and versatility through support for block, object, and file interfaces, making it well suited for large-scale infrastructures and container environments. OpenStack Swift is characterized by ease of integration into the OpenStack ecosystem, however; it demonstrates reduced performance when processing large numbers of small files. MinIO is a lightweight and high-performance solution with native S3 API support, which makes it the most suitable for this task. The study also emphasizes the need for a preliminary analysis of dominant multimedia content types when designing a storage system to achieve maximum efficiency.

Keywords: data storage, file system, object storage structure, S3 protocol, Ceph, MinIO, Openstack Swift

Authors' contribution
Tast S.A. — research conduction, collection and processing of materials, text writing; *Moseva M.S.* — research concept and design, expert support. Both of the authors read and approved the final version of the article.

For citation
Tast SA, Moseva MS. Comparative analysis of open-source object data storage systems. *RUDN Journal of Engineering Research.* 2026;27(2):193–202. (In Russ.) http://doi.org/10.22363/2312-8143-2026-27-2-193-202 EDN: KYMATZ

Введение

В рамках предоставления услуг корпоративного мессенджера неизбежно возникает потребность хранения больших объемов пользовательских файлов. В настоящее время в рассматриваемом авторами мессенджере объем потребности насчитывает в среднем 24 терабайта хранимых файлов в год, однако этот показатель имеет тенденцию к увеличению. Период хранения составляет минимум 1 год (с возможностью продления срока). При удалении учетной записи пользователя файлы, владельцем которых являлся этот пользователь, необходимо сохранять в течение 1 календарного месяца.

Цель исследования — выбор оптимального решения для предоставления пользователям возможности удаленного хранения не ограниченных по размеру данных произвольного формата. Наибольший интерес для реализации запроса представляют открытые решения.

Согласно Национальному стандарту Российской Федерации ГОСТ Р 7.0.95-2015[1], электронный объект — это файл (или совокупность файлов), формируемый в компьютерной программе пользователя или в автоматизиро-

[1] ГОСТ Р 7.0.95-2015. Электронные документы. Основные виды, выходные сведения, технологические характеристики. Москва : Стандартинформ, 2015. 23 с.

ванной системе и содержащий в зафиксированном виде данные, предназначенные для восприятия с помощью средств вычислительной техники [1].

Файл — это именованная часть диска [2].

Ниже перечислим основные существующие типы систем хранения данных [3].

1. *Файловое хранилище* — традиционная модель хранения данных, где информация организована в иерархическую структуру [4]. Это наиболее распространенный и простой тип структурированного хранения данных, который чаще всего можно найти на персональных компьютерах, серверах и мобильных устройствах. В данной модели каждый файл имеет уникальное имя, сопровождается метаданными (информацией о файле) и занимает строго определенное место в иерархии [5].

Преимущества файловой системы в иерархическом формате заключаются в ее простоте и интуитивной понятности для большинства конечных пользователей, что не требует от них особых навыков для работы и настройки, а также возможность кроссплатформенной совместимости по причине высокой популярности. Кроме того, такой тип хранения данных является относительно экономически эффективным и безопасным [6].

К недостаткам файлового хранилища можно отнести:

– ограничения на размеры файлов в хранилище,

– ограничения на количество файлов,

– сложности с масштабированием системы.

Данная форма хранения предпочтительна для относительно небольших объемов данных и локального использования обычными пользователями. Организация общего доступа к содержимому файлового хранилища может быть реализована с использованием специализированных расширений, таких как Network File System (NFS) [7].

2. *Блочное хранилище* — система хранения информации, которая напоминает иерархическую модель, но при этом файлы разбиваются на одинаковые по объему, непересекающиеся фрагменты или блоки [8].В этой системе блоками являются отдельные кластеры, способные вмещать большие объемы данных.

Преимущества блочной системы заключаются в возможности разделения большого объема информации на блоки и обеспечении отдельного доступа к каждому из них [9].

Кроме того, блочные хранилища используют набор инструментов, которые обеспечивают высокую производительность системы: хост-адаптер шины разгружает процессор, освобождая его ресурсы для выполнения других задач [10]. Отсутствие сложных абстракций также способствует увеличению производительности и упрощает процесс масштабирования. Поэтому блочные системы хранения часто применяются в контексте гипервизоров и виртуализации, а также удобны для работы с реляционными базами данных [11].

К недостаткам блочной системы относятся сложность управления и необходимость узкой специализации для настройки, а также ограниченность объема хранения. Эти факторы также влияют на стоимость такой структуры, делая ее использование достаточно дорогим. Ограниченность метаданных[2] приводит к необходимости обработки вспомогательной информации на уровне приложений и баз данных.

3. *Объектная система хранения данных* — это тип хранилища, в котором данные различного формата и объема хранятся в виде объектов с метаданными, как правило, на облачном сервисе [12]. Оптимизация хранения метаданных достигается за счет соответствующей настройки.

К основным преимуществам объектного хранилища относятся [13]:

– легкая масштабируемость и практически неограниченный объем;

– отсутствие громоздкой структуры;

– высокая скорость поиска объектов;

[2] Amazon Web Services. What is block storage? URL: https://aws.amazon.com/ru/what-is/block-storage/ (accessed: 25.11.2025).

– отсутствие строгих ограничений по формату и предобработке файлов;

– внутренние механизмы проверки целостности данных и другие функциональные возможности.

Недостатками объектного хранилища являются сложности при работе из-за частого отсутствия интерфейса или необходимости установки дополнительного программного обеспечения в правильной конфигурации, а также более высокая задержка доступа к данным из-за значительного объема метаданных по сравнению с блочной структурой [14].

Таким образом, объектная структура хранения лучше всего подходит для больших объемов неструктурированных данных, в то время как хранение реляционных баз данных в таком формате представляется крайне сложным и нецелесообразным.

1. Методы

Для реализации объектного доступа было разработано множество протоколов, среди которых наиболее популярными являются SOAP, S3 и OpenStack Swift [15]. Рассмотрим их особенности.

SOAP (Simple Object Access Protocol) — это протокол обмена сообщениями, используемый для передачи структурированной информации между веб-службами. Данный протокол предоставляет стандартизированный формат для коммуникации, включая использование XML для инкапсуляции сообщений. Он разработан с целью обеспечения взаимодействия между приложениями различных платформ и языков программирования через интернет. Для передачи документов SOAP может использовать различные протоколы, такие как FTP, HTTP, SMTP и POP3[3].

S3 (Simple Storage Service) — протокол, предоставляющий механизмы управления доступом к данным, позволяя назначить различные разрешения для отдельных объектов. Кроме того, S3 поддерживает управление жизненным циклом данных [16]. Этот протокол был представлен компанией Amazon в 2006 г. API (интерфейс прикладного программирования) доступен для всех сторонних разработчиков [17].

Некоторые уникальные особенности протокола S3[4]:

– элементы управления на уровне контейнера для управления версиями и истечения срока действия, которые применяются ко всем объектам в контейнере;

– копирование объектов — позволяет делать копии объектов на стороне сервера.

– анонимный доступ — возможность установить публичный доступ к объекту и обслуживать его по HTTP/HTTPS без аутентификации.

OpenStack Swift. OpenStack — это бесплатная платформа облачных вычислений с открытым исходным кодом, совместная разработка NASA и облачного провайдера Rackspace [18]. Swift — компонент объектного хранилища в OpenStack, действующий по одноименному протоколу[5].

Протокол Swift управляется OpenStack Foundation, некоммерческой корпоративной организацией, созданной в сентябре 2012 г. для продвижения программного обеспечения OpenStack и его сообщества. К проекту присоединились более 500 компаний.

Уникальная особенность Swift API: создание неразмерного объекта. Swift — единственный протокол, в котором можно использовать кодирование «Chunked» (фрагментация) для загрузки объекта, размер которого заранее неизвестен. S3 требует для этого несколько запросов [19].

Аутентификация в Swift осуществляется через отдельный механизм, создающий «токен», который может передаваться для аутентификации запросов. OpenStack Swift в ECS поддерживает две версии аутентификации.

[3] DreamFactory. Understanding SOAP security. URL: https://blog.dreamfactory.com/understanding-soap-security (accessed: 05.11.2025)

[4] Old Hen Hut. S3 vs Swift. URL: https://oldhenhut.wordpress.com/2016/05/31/s3-vs-swift/ (accessed: 05.11.2025).

[5] OpenStack Foundation. The most widely deployed open source cloud software in the world. URL: https://www.openstack.org/ (accessed: 01.12.2025).

На сегодняшний день протоколы S3 от Amazon и Swift от OpenStack — две наиболее популярные версии протоколов для облачных объектных хранилищ [20]. Они представлены в различных продуктах, предоставляющих интерфейсы взаимодействия с облачными сервисами[6].

Для реализации сравнительной характеристики взаимодействия существующих хранилищ с имеющимися данными мы будем изучать протоколы S3 от Amazon и Swift от OpenStack для объектного типа хранения данных и предлагаемые в рамках этих протоколов продукты для облачного хранения данных.

В качестве продуктов для сравнения выберем три предложения с открытым исходным кодом: Ceph [21], MinIO [22], Openstack Swift[7].

Ceph — комплексная система, поддерживает блочные, объектные и файловые данные. Основной отличительной чертой является высокая отказоустойчивость, которая хорошо зарекомендовала себя в применении для крупных инфраструктур [23].

MinIO — легкое, высокопроизводительное решение, оптимизировано под работу с данными через S3 API. Отлично подходит для контейнерных сред [24].

В отличие от других open-source решений MinIO изначально разработан как высокоскоростная, легковесная и масштабируемая система, оптимизированная именно для работы с большими объемами неструктурированных данных, что идеально соответствует природе медиаконтента, передаваемого в мессенджерах: изображений, видео-, аудиофайлов и документов [25].

OpenStack Swift — часть экосистемы OpenStack. Простая в развертывании, но менее производительна на мелких файлах. Лучше подходит для холодного хранения данных, то есть при редком использовании хранящихся файлов [26].

2. Результаты и обсуждение

2.1. Результаты анализа

Результаты сравнения после реализации представлены в таблице.

Опираясь на результаты представленной таблицы сравнения, можно сделать некоторые ключевые выводы о описанных продуктах хранения данных.

Для заявленной цели, поставленной в данной работе, наиболее приемлемым вариантом будет являться решение MinIO. Одним из главных преимуществ MinIO является его полная совместимость с API Amazon S3, который стал де-факто стандартом для объектного хранения. Это означает, что любое приложение, рассчитанное на работу с AWS S3, включая большинство современных платформ корпоративных коммуникаций, шлюзов безопасности и систем анализа данных, может быть интегрировано с MinIO без необходимости изменения кода. Такая совместимость значительно упрощает миграцию с облачных решений на локальную или гибридную инфраструктуру и позволяет организациям сохранять контроль над своими данными, соблюдая требования к их локализации и защите.

Архитектура MinIO построена на принципах горизонтального масштабирования и отказоустойчивости. Система легко формирует кластеры из множества узлов, автоматически распределяя данные с использованием технологии erasure coding, которая обеспечивает защиту от потери информации даже при выходе из строя нескольких дисков или серверов одновременно. При этом MinIO демонстрирует одну из самых высоких скоростей чтения и записи среди open-source решений, что критично для мессенджеров, где пользователи ожидают мгновенной загрузки медиафайлов. По сравнению с такими альтернативами, как OpenStack Swift или Ceph, MinIO требует меньше ресурсов для запуска, проще в настройке и не нуждается в сложной инфраструктуре управления, такой как мониторинговые узлы или специальные базы метаданных. Его можно быстро развернуть в виде контейнеров в Kubernetes, что делает его естественным выбором для организаций, использующих микросервисную архитектуру.

[6] OpenSourceAlternative.to. Open source alternatives to Amazon S3. URL: https://opensourcealternative.to/alternativesto/amazon-s3 (accessed: 03.11.2025).

[7] MSP360. Open-source object storage solutions review. URL: https://www.msp360.com/resources/blog/open-source-object-storage-vendors-comparison/ (accessed: 08.11.2025).

Сравнительный анализ предложений хранилищ объектного типа

Параметр	Ceph	MinIO	OpenStack Swift
1	2	3	4
Протокол	S3, Swift, RESTful	S3 API (полная совместимость)	OpenStack Swift API, S3-совместимый доступ
Производительность: • Скорость загрузки (большие файлы) • Скорость загрузки (мелкие объекты) • Скорость скачивания (большие файлы) • Скорость скачивания (мелкие объекты)	• 250–300 МБ/с (без аппаратного ускорения) • 100–120 МБ/сек (ограничено оверхедом CRUSH и репликацией) • 280–320 МБ/с (с кэшированием) • 120–140 МБ/с	• 300–400+ МБ/с (в распред. режиме) • 120–150 МБ/с • 350–450+ МБ/с • 160–200 МБ/с	• 180–220 МБ/с • 70–100 МБ/с • 200–250 МБ/с • 80–110 МБ/с
Максимальный объем хранимых данных	Не ограничен (масштабируется горизонтально)	Не ограничен (распределенное хранение)	Не ограничен (масштабируется горизонтально)
Возможность работы кластера с опциями отказоустойчивости	Высокая отказоустойчивость: механизм репликации и авт. восстановление данных. Различные уровни отказоустойчивости	Поддержка отказоустойчивости через репликацию данных и механизм распределенного хранения	Обеспечивает отказоустойчивость через репликацию и возможные настройки различных уровней защиты данных
Репликация	Разные стратегии репликации (в т.ч. 3-way replication) и erasure coding для защиты данных	Стандартная репликация, erasure coding для защиты данных	Реализует репликацию объектов и поддерживает erasure coding для оптимизации хранения
Работа в контейнере	Поддерживает работу в контейнеризованных средах (в том числе Kubernetes) с помощью Rook и других интеграций	Оптимизирован для работы в контейнерах, легко разворачивается в Kubernetes и Docker	Может работать в контейнерах, требует больше усилий для интеграции по сравнению с MinIO
Клиентские приложения и монтирование в популярные ОС	CLI (radosgw), S3cmd, Rclone, FUSE-монтирование в Linux	S3cmd, MinIO Client (mc), Rclone, FUSE, Web UI, поддержка S3 SDK для всех языков	CLI, Swift CLI, Rclone, поддержка FUSE (через проекты сообщества)
Трудозатраты разработчика при подготовке проекта (в раб. ч.)	20	2,5	12

И с т о ч н и к: выполнено С.А. Тастом.

Comparative Analysis of Object Storage Solutions

Parameter	Ceph	MinIO	Open Stack Swift
1	2	3	4
Protocol	S3, Swift, RESTful	S3-compatible API	OpenStack Swift API, S3-compatible access
Performance: • Upload speed (large files) • Upload speed (small objects) • Download speed (large files) • Download speed (small objects)	• 250–300 MB/s (without hardware acceleration) • 100–120 MB/s (limited by CRUSH and replication overhead) • 280–320 MB/s (with caching) • 120–140 MB/s	• 300–400+ MB/s (in distributed mode) • 120–150 MB/s • 350–450+ MB/s • 160–200 MB/s	• 180–220 MB/s • 70–100 MB/s • 200–250 MB/s • 80–110 MB/s
Maximum storage capacity	Not limited (horizontally scalable)	Not limited (distributed storage)	Not limited (horizontally scalable)
Cluster fault-tolerance capabilities	High fault tolerance: replication mechanism and automatic data recovery; various levels of fault tolerance	Fault-tolerance support through data replication and distributed storage mechanisms	Provides fault tolerance through replication and the ability to configure different levels of data protection
Replication	Various replication strategies, including 3-way replication, and erasure coding for data protection	Standard replication and erasure coding for data protection	Implements object replication and supports erasure coding for storage optimization
Containerized deployment	Supports operation in containerized environments, including Kubernetes, using Rook and other integrations	Optimized for containerized deployment; easily deployed in Kubernetes and Docker	Can run in containers, but requires more integration effort compared with MinIO
Client applications and mounting in popular OS	CLI (radosgw), s3cmd, Rclone, FUSE mounting in Linux	S3cmd, MinIO Client (mc), Rclone, FUSE, Web UI, S3 SDK support for major programming languages	CLI, Swift CLI, Rclone, FUSE support through community projects
Developer effort for project preparation, person-hours	20	2.5	12

S o u r c e: by S.A. Tast.

Кроме того, MinIO активно развивается сообществом и компанией-разработчиком, регулярно получает обновления и поддерживает современные функции, такие как уведомления о контейнерах, управление жизненным циклом, встроенная антивирусная проверка через интеграцию с ClamAV и поддержка машинного обучения через MinIO ML Gateway. Это позволяет не только хранить медиаданные, но и организовать их автоматическую обработку — например, анализ содержимого, тегирование на основе метаданных или удаление устаревших файлов согласно политикам хранения.

2.2. Обсуждение результатов

Проведенное исследование направлено на сравнительный анализ современных систем хранения данных с точки зрения их применимости в качестве хранилища медиафайлов для систем обмена сообщениями, с акцентом на требования к производительности, масштабируемости, отказоустойчивости и простоте эксплуатации. На основе анализа функциональных характеристик и архитектурных особенностей различных решений авторами было определено, что объектно-ориентированное хранилище, в частности MinIO, в наибольшей степени удовлетворяет совокупности предъявляемых требований и может рассматриваться как приоритетный вариант для внедрения в инфраструктуру корпоративных и отраслевых мессенджеров.

Ключевой результат работы заключается в обосновании выбора MinIO как оптимального решения, которое обеспечивает необходимую производительность, а также бесшовную интеграцию с широким спектром существующих приложений и сервисов без модификации их кода. Такая совместимость снижает порог внедрения для организаций, уже использующих облачные платформы, и открывает возможности для построения гибридных архитектур, в которых локальное объектное хранилище интегрируется с публичными облаками. В отличие от более тяжелых по инфраструктурным требованиям решений, таких как OpenStack Swift или Ceph, MinIO демонстрирует относительно низкую ресурсозатратность и упрощенную конфигурацию, что делает его особенно привлекательным для микросервисных систем.

Научная новизна работы для отечественной и зарубежной науки проявляется, прежде всего, в комплексной оценке MinIO как элемента инфраструктуры систем обмена сообщениями с учетом специфики их нагрузочного профиля. Если в существующих исследованиях объектные хранилища преимущественно рассматриваются в контексте высокопроизводительных вычислений, аналитики больших данных или классических облачных платформ, то в данной работе акцент сделан на сценариях, где критичны одновременно высокая скорость работы с медиафайлами, простота масштабирования и соответствие требованиям по локализации и защите данных.

Представляется целесообразным проведение экспериментальных нагрузочных испытаний MinIO в реальных (или близких к реальным) условиях эксплуатации мессенджеров с различными профилями трафика (корпоративный, образовательный, массовый публичный сервис) для количественной оценки показателей задержек, пропускной способности и устойчивости к отказам. Отдельным перспективным направлением может стать разработка методик и рекомендаций по миграции существующих систем хранения, основанных на файловой или блочной модели к объектной архитектуре с учетом организационных, правовых и эксплуатационных ограничений конкретных отраслей.

Авторы видят также возможности для дальнейших исследований в области совмещения функций хранения и интеллектуальной обработки данных на уровне инфраструктуры. Также для изучения интересной представляется сфера различных возможностей, которые не включены в текущую конфигурацию хранилища: логирование операций с данными, ограниченные возможности условий фильтрации, отсутствие автоматического отслеживания доступа и прочее. Эти особенности могут быть необходимы для отдельных областей применения хранилища, поэтому необходим их тщательный анализ.

Заключение

Используя представленные результаты, можно сделать вывод о несомненных преимуществах объектного типа хранилища для решения поставленной задачи — безопасного и эффективного хранения мультимедийных данных в системах корпоративных коммуникаций. При анализе конкретных протоколов и решений с открытым исходным кодом были определены ключевые характеристики, влияющие на выбор заказчика. Среди хранилищ Ceph, MinIO и OpenStack Swift имеются различия, которые могут быть преимуществами и недостатками, в зависимости от особенностей файлов, инфраструктуры и прочих параметров. Выбранное нами решение — MinIO — сочетает в себе производительность, безопасность, масштабируемость и простоту внедрения, что делает его наиболее подходящим для хранения мультимедийных данных в системах корпоративных коммуникаций.

Список литературы

1. *Майстрович Т.В.* Электронный документ: основные характеристики и его место в системе обязательного экземпляра // Библиотековедение. 2012. Т. 1. С. 43–46. https://doi.org/10.25281/0869-608X-2012-0-1-43-46

2. *Кварацхелия Л.Д.* Программа обеспечения автоматизации управления режимом доступа к ресурсам файловой системы // Евразийский союз учёных (ЕСУ). 2019. № 7 (64), ч. 6. С. 34. https://doi.org/10.31618/ESU.2413-9335.2019.6.64

3. *Matick R.E.* Computer storage systems and technology. New York : Wiley, 1977. p. 549–551. ISBN 9780471576297

4. *Смирнов А.А., Бабкин А.А., Галеев В.В.* S3-хранилище против блочного и файлового хранилищ // Научный лидер. 2024. № 34 (184). С. 11–12. URL: https://scilead.ru/media/journal_pdf_184.pdf#page=11 (дата обращения: 27.11.2025).

5. *Kampfner R.R.* A hierarchical model of organizational control for the analysis of information systems requirements // Information Systems. 1987. Vol. 12. No. 3. P. 243–254. https://doi.org/10.1016/0306-4379(87)90003-2

6. *Семеночкин Д.О., Марков А.К., Кравец А.Г.* Анализ иерархической модели организации файловых систем // Инновационные, информационные и коммуникационные технологии : сб. трудов XX Международной научно-практической конференции. 01–10 окт. 2023, Махачкала. Москва : Ассоциация выпускников и сот-рудников ВВИА им. Н.Е. Жуковского, 2023. С. 240–243. EDN: BBGPCE

7. *Walker E.* A distributed file system for a wide-area high performance computing infrastructure // Proc. of the 2009 ACM/IEEE Conference on High Performance Computing, Networking, Storage and Analysis (SC'09). 2009. P. 1–12. URL: http://arxiv.org/pdf/1001.0196.pdf (дата обращения: 01.11.2025).

8. *Алуев Е.А.* Преодоление проблем Big Data с помощью Data Lake // Big Data и анализ высокого уровня : сб. науч. статей XI Междунар. научно-практической конференции. Минск, 23–24 апреля. Минск : БГУИР, 2025. С. 66–73. ISBN 978-985-543-814-5

9. *Борискина А.Д., Ванисов Р.С.* Облачное хранилище // Информационно-телекоммуникационные системы и технологии, материалы Всероссийской научно-практической конференции Кемерово, 26 ноября 2021, Кемерово : КузГТУ, 2021. С. 60–61. ISBN 978-5-00137-271-4 EDN: ZZEEUW

10. *Захаров Д.С., Шпилевой Б.Д.* Файловое онлайн-хранилище для небольшой проектной организации // Инновационное развитие строительного комплекса региона, материалы II Всерос. научно-практической конференции, 15 окт. 2019, Михайловка — Волгоград. Волгоград : ВолгГТУ, 2020. С. 225–228. EDN: JBWWQH

11. *Кенжаев С.С., Рашидов А.Э.У.* Методы и алгоритмы хранения файлов для оптимального управления различными типами данных // Al-Farg'oniy avlodlari. 2024. № 3. С. 82–92. https://doi.org/10.5281/zenodo.13954911

12. *Самойлов Ю.Д.* Разработка серверной части, связанной с функциональными возможностями пользователя, для системы Request Helper: выпускная бакалаврская работа по направлению подготовки: 09.03. 04 Программная инженерия. 2023. URL: https://vital.lib.tsu.ru/vital/access/services/Download/vital:18526/SOURCE01 (дата обращения: 08.10.2025)

13. *Factor M., Meth K., Naor D., Rodeh O., Satran J.* Object storage: The future building block for storage systems // 2005 IEEE International Symposium on Mass Storage Systems and Technology. IEEE, 2005. P. 119–123. https://doi.org/10.1109/LGDI.2005.1612479

14. *Хомоненко А.Д., Абу-Хасан Р.* О надежности и доступности объектных хранилищ данных // Интеллектуальные технологии на транспорте. 2023. № S1 (35-1). С. 123–128. URL: https://cyberleninka.ru/article/n/o-nadezhnosti-i-dostupnosti-obektnyh-hranilisch-dannyh/viewer (accessed: 10.11.2025)

15. *Scope N., Rasin A., Lenard B., Wagner J.* Compliance and data lifecycle management in databases and backups // International Conference on Database and Expert Systems Applications. Cham : Springer Nature Switzerland, 2023. P. 281–297. https://doi.org/10.1007/978-3-031-39847-6_20

16. *Akshay M.S., Mohan S., Kuri V., Sitaram D., Phalachandra H.L.* Efficient support of big data storage systems on the cloud // Proc. of the 7th IEEE/ACM International Conference on Cloud Computing. 2014. P. 1–8. URL: http://arxiv.org/pdf/1411.7507.pdf (accessed: 01.11.2025).

17. *Зензин А.С.* Проектирование и разработка горизонтально масштабируемой распределенной файловой системы: дипломный проект. Томск : НИ ТПУ, Институт кибернетики, кафедра ИПС, 2016. 105 с. URL: http://earchive.tpu.ru/handle/11683/28558 (дата обращения: 15.11.2025).

18. *Teixeira J.A., Mian S.Q., Hytti U.* Cooperation among competitors in the open-source arena: The case of OpenStack // Journal of Internet Services and Applications. 2016. Vol. 7. Article no. 8. URL: http://arxiv.org/pdf/1612.09462.pdf (accessed: 01.10.2025).

19. *Rupprecht L., Zhang R., Owen B., Pietzuch P., Hildebrand D.* SwiftAnalytics: Optimizing object storage for big data analytics // Proc. of IEEE International Conference on Cloud Engineering (IC2E). Vancouver, 2017. P. 245–251. https://doi.org/10.1109/IC2E.2017.19

20. *Wang Y.* An analysis of performance and potential of cloud computing and object storage : Doctoral dissertation, University of Auckland. Auckland, 2022. URL: https://hdl.handle.net/2292/62217 (accessed: 08.10.2025).

21. *Ильин В.Г., Цынгалёв П.С., Боронников А.С.* Применение Ceph в современных облачных инсталляциях // International Journal of Open Information Technologies. 2023. Т. 11. № 2. С. 44–50. ISSN 2307-8162

22. *Ghemawat S., Gobioff H., Leung S.T.* The Google File System // ACM SIGOPS Operating Systems Review. 2003. Vol. 37. No. 5. P. 29–43. https://doi.org/10.1145/1165389.945450

23. *Weil S., Brandm S.A., Miller E.L., Long D.D.E., Maltzahn C.* Ceph: A scalable, high-performance distributed file system // Proceedings of the 7th USENIX OSDI Conference. November 6–8, Seattle, WA, USA, 2006. P. 307–320. URL: https://www.usenix.org/events/osdi06/tech/full_papers/weil/weil_html (accessed: 10.11.2025).

24. *Gadban F., Kunkel J.* Analyzing the performance of the S3 object storage API for HPC workloads // Applied Sciences. 2021. Vol. 11. No. 18. Article no. 8540. https://doi.org/10.3390/app11188540

25. *Dimakis A.G., Godfrey P.B., Wu Y., Wainwright M. J., Ramchandran K.* Network coding for distributed storage systems // IEEE Trans. Inf. Theory. 2010. Vol. 56. No. 9. P. 4539–4551. https://doi.org/10.1109/TIT.2010.2054295 ISSN 0018-9448 / 1557-9654

26. *Маркелов А.* Openstack. Практическое знакомство с облачной операционной системой. Москва : ДМК Пресс, 2018. 306 с. ISBN 978-5-97060-652-0

References

1. Maistrovich TV. Electronic document: Main Characteristics and its Place in the System of Legal Deposit. *Russian Journal of Library Science.* 2012;(1): 43–46. (In Russ.) https://doi.org/10.25281/0869-608X-2012-0-1-43-46

2. Kvaratskhelia LD. Program for ensuring automation of access control to file system re-sources. *Evraziyskiy Soyuz Uchenykh (ESU)*. 2019;7(64);34–38. (In Russ.) https://doi.org/10.31618/ESU.2413-9335.2019.6.64

3. Matick RE. *Computer storage systems and technology*. New York: Wiley, 1977. p. 549–551. ISBN 9780471576297

4. Smirnov AA, Babkin AA, Galeev VV. S3 storage versus block and file storage. *Nauchnyy Lider*. 2024;34: 11–12. (In Russ.) Available from: https://scilead.ru/media/journal_pdf_184.pdf#page=11 (accessed: 27.11.2025)

5. Kampfner RR. A hierarchical model of organizational control for the analysis of information systems requirements. *Information Systems.* 1987;12(3):243–254. (In Russ.) https://doi.org/10.1016/0306-4379(87)90003-2

6. Semenochkin DO, Markov AK, Kravets AG. Analysis of hierarchical model of file systems organization. *Innovative, Information and Communication Technologies: Proceedings of the XX International Scientific and Practical Conference;* 2023; Moscow: Association of Alumni and Staff of the VVIA named after Professor N.E. Zhukovsky for the Preser-vation of Historical and Scientific Heritage of the VVIA named after Professor N.E. Zhukovsky; 2023. p. 240–243. (In Russ.) EDN: BBGPCE

7. Walker E. A distributed file system for a wide-area high performance computing infrastructure. *In Proceedings of the 2009 ACM/IEEE Conference on High Performance Computing, Networking, Storage and Analysis (SC'09)*. 2009; Seattle. p. 1–12. Available from: http://arxiv.org/pdf/1001.0196.pdf (accessed: 01.11.2025).

8. Aluev EA. Overcoming big data problems using data lake. *The Eleventh International Scientific and Practical Conference on BIG DATA and Advanced Analytics,* April, 23–24, Minsk; BGUIR. 2025. p. 66–73. (In Russ.) ISBN 978-985-543-814-5

9. Boriskina AD, Vanisov RS. Cloud storage. *Proceedings of the All-Russian Scientific and Practical Conference "Information and Telecommunication systems and Technologies."* Kemerovo, November 26, 2021, Kemerovo: KuzGTU; 2021. p. 60–61. (In Russ.) ISBN 978-5-00137-271-4 EDN: ZZEEUW

10. Zakharov DS, Shpilevoy BD. File online storage for small project organization. Information and Telecommunication Systems and Technologies: Proceedings of the All-Russian Scientific and Practical Conference. Volgograd: VolgGTU; 2020. p. 225–228. (In Russ.) ISBN 978-5-9669-1981-8

11. Kenjaev S, Rashidov A. Methods and algorithms for file storage for optimal management of different types of data. *Al-Farg'oniy avlodlari.* 2024;1(3):82–89. (In Russ.) https://doi.org/10.5281/zenodo.13954911

12. Samoylov YuD. *Development of server part related to user functionality for Request Helper system.*

[Bachelor's thesis in Software Engineering]. 2023. (In Russ.). Available from: https://vital.lib.tsu.ru/vital/access/services/Download/vital:18526/SOURCE01 (accessed: 08.10.2025)

13. Factor M, Meth K, Naor D, Rodeh O, Satran J. Object storage: The future building block for storage systems. *2005 IEEE International Symposium on Mass Storage Systems and Technology*. 2005 20–24 Jun; Sardinia, Italy. IEEE; 2005. p. 119–123. https://doi.org/10.1109/LGDI.2005.1612479

14. Khomonenko AD, Abu-Khasan R. About the reliability and availability of object data storages. *Intellectual Technologies on Transrort.* 2023;(S1):123–128. (In Russ.). Available from: https://cyberleninka.ru/article/n/o-nadezhnosti-i-dostupnosti-obektnyh-hranilisch-dannyh/viewer (accessed: 10.11.2025)

15. Scope N, Rasin A, Lenard B, Wagner J. Compliance and Data Lifecycle Management in Databases and Backups. In: Strauss C, Amagasa T, Kotsis G, Tjoa AM, Khalil I, editors. *Database and Expert Systems Applications. DEXA 2023. Lecture Notes in Computer Science.* Cham: Springer; 2023. p. 281–297. https://doi.org/10.1007/978-3-031-39847-6_20

16. Akshay MS, Mohan S, Kuri V, Sitaram D, Phalachandra HL. Efficient support of big data storage systems on the cloud. *Proceedings of the 7th IEEE/ACM International Conference on Cloud Computing.* 2014. p. 1–8. (In Russ.) Available from: http://arxiv.org/pdf/1411.7507.pdf (accessed: 01.11.2025).

17. Zenzin AS. *Design and development of horizontally scalable distributed file system.* Diploma project. Tomsk: NI TPU, Institute of Cybernetics, Department IPS; 2016. (In Russ.) Available from: http://earchive.tpu.ru/handle/11683/28558 (accessed: 15.11.2025).

18. Teixeira JA, Mian SQ, Hytti U. Cooperation among competitors in the open-source arena: The case of OpenStack. *Journal of Internet Services and Applications.* 2016;7:8. https://doi.org/10.48550/arXiv.1612.09462

19. Rupprecht L, Zhang R, Owen B, Pietzuch P, Hildebrand D. SwiftAnalytics: Optimizing object storage for big data analytics. In: *Proceedings of IEEE International Conference on Cloud Engineering (IC2E);* 2017 Apr 1–7; Vancouver, BC, Canada. IEEE; 2017. p. 245–251. https://doi.org/10.1109/IC2E.2017.19

20. Wang Y. *An analysis of performance and potential of cloud computing and object storage.* Doctoral dissertation, University of Auckland. Auckland, 2022. Available from: https://hdl.handle.net/2292/62217 (accessed: 08.10.2025)

21. Ilyn VG, Tsyngalev PS, Boronnikov AS. Using Ceph in modern cloud installations. *International Journal of Open Information Technologies*. 2023;11(2):44–50. (In Russ.) ISSN 2307-8162 EDN: ZOMMUR

22. Ghemawat S, Gobioff H, Leung ST. The Google file system. *ACM SIGOPS Operating Systems Review*. 2003;37(5):29–43. https://doi.org/10.1145/1165389.945450

23. Weil S, Brandt SA, Miller EL, Long DDE, Maltzahn C. Ceph: A scalable, high-performance distributed file system. In: *Proceedings of the 7th USENIX OSDI Conference.* November 6-8, Seattle, WA, USA, 2006. p. 307–320. Available from: https://www.usenix.org/events/osdi06/tech/full_papers/weil/weil_html (accessed: 10.11.2025)

24. Gadban F, Kunkel J. Analyzing the performance of the S3 object storage API for HPC workloads. *Applied Sciences,* 2021;11(18):8540. https://doi.org/10.3390/app11188540

25. Dimakis AG, Godfrey PB, Wu Y, Wainwright MJ, Ramchandran K. Network coding for distributed storage systems. *IEEE Transactions on Information Theory* 2010; 56(9):4539–4551. https://doi.org/10.1109/TIT.2010.2054295

26. Markelov A. *OpenStack. Practical introduction to the cloud operating system.* Moscow: DMK Press; 2018. (In Russ.) ISBN 978-5-97060-652-0

Сведения об авторах

Тast Сергей Артурович, магистрант Центра заочного обучения по программам магистратуры, Московский технический университет связи и информатики, Российская Федерация, г. Москва, ул. Авиамоторная, д. 8А; ORCID: 0009-0004-6244-654X; e-mail: s.a.tast@edu.mtuci.ru

Мосева Марина Сергеевна, кандидат технических наук, доцент кафедры программной инженерии, факультет информационных технологий, Московский технический университет связи и информатики, Российская Федерация, г. Москва, ул. Авиамоторная, д. 8А; eLIBRARI SPIN-код: 1313-0436; ORCID: 0000-0002-9778-124X; e-mail: m.s.moseva@mtuci.ru

About the authors

Sergey A. Tast, Master's student at the Center for Distance Learning for Master's Degree Programs at the Moscow Technical University of Communications and Informatics, 8A Aviamotornaya St, Moscow, Russian Federation; ORCID: 0009-0004-6244-654X; e-mail: s.a.tast@edu.mtuci.ru

Marina S. Moseva, PhD in Technical Sciences; Associate Professor of the Department of Software Engineering, Faculty of Information Technology, Moscow Technical University of Communications and Informatics, 8A Aviamotornaya St, Moscow, Russian Federation; eLIBRARI SPIN-code: 1313-0436; ORCID: 0000-0002-9778-124X; e-mail: m.s.moseva@mtuci.ru