ДИАМАНТ
СОХРАНЯЯ БЕСЦЕННОЕ

Сравнение «ДИАМАНТ» «АМУР» vs. MinIO и Cloudian

Оглавление

ДИАМАНТ
СОХРАНЯЯ БЕСЦЕННОЕ

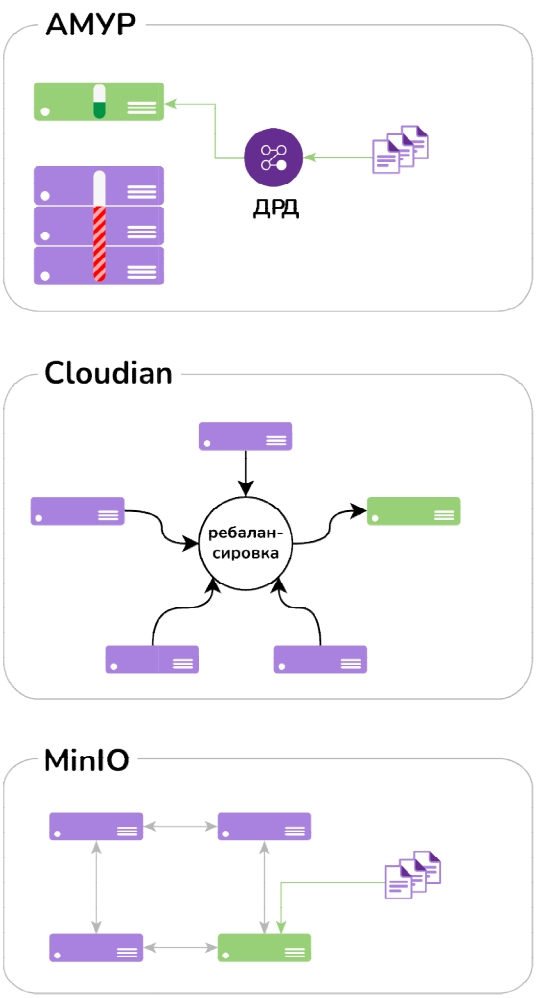
Производительность при смешанных нагрузках (мелкие и крупные объекты, параллельные запросы)

1. **Высокая скорость работы с метаданными и мелкими объектами. «АМУР»** хранит всю информацию о метаданных объектов на быстрых твёрдотельных накопителях NVMe, что минимизирует задержки операций получения списка объектов, поиска и доступа к маленьким файлам. В **«АМУРЕ»** реализована оптимизация для мелких объектов: система сохраняет полноценную копию каждого небольшого объекта на одном диске и параллельно распределяет его фрагменты по остальным дискам. При чтении небольшого объекта сначала возвращается целая копия, и только в случае недоступности — собираются фрагменты с нескольких дисков. Это ускоряет обработку множества мелких файлов, снижая вычислительную нагрузку на кодирование/декодирование. **MinIO** и **Cloudian** исторически испытывали трудности с высокими задержками доступа к малым файлам и добавили схожие решения (**MinIO** начал встраивать маленькие объекты в метаданные, **Cloudian** ввёл гибридные политики хранения с репликацией для мелких файлов), однако **«АМУР»** изначально спроектирован для таких нагрузок и демонстрирует более высокую производительность **без** сложной настройки.

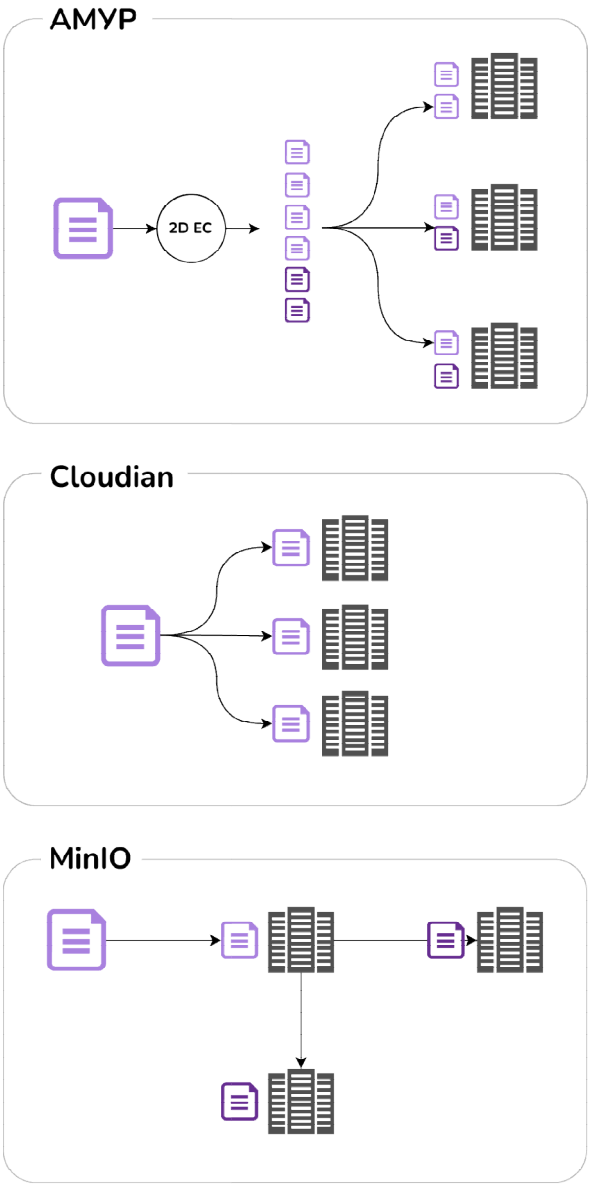
2. **Линейное масштабирование и высокая пропускная способность. «АМУР»** обладает масштабируемой параллельной архитектурой: производительность растёт линейно при добавлении узлов, без узких мест и без необходимости ребалансировки данных. В новой флеш-конфигурации **«АМУР Z»** компания «ДИАМАНТ» заявляет удвоенную производительность на ядро CPU по сравнению с конкурентами при реальных нагрузках и полную открытость и максимальную приближенность к реальным условиям во всех тестах производительности. Это значит, что **«АМУР»** достигает высоких скоростей ввода-вывода в реальных условиях (например, для больших объектов — гигабайты в секунду), обслуживая параллельно множество запросов. **MinIO** также известен высокой пропускной способностью, но её достижение сильно зависит от ручной оптимизации и топологии кластера. В **Cloudian** пропускная способность обычно ниже на аналогичном оборудовании, так как дополнительный уровень (NoSQL-база Cassandra для метаданных) добавляет накладные расходы на каждую операцию. **«АМУР»**, благодаря размещению метаданных во флеш-памяти и широкополосному распределению данных по всем дискам, обеспечивает стабильно высокую скорость как на потоковых операциях с крупными объектами, так и на транзакционных операциях с мелкими файлами.

Масштабируемость (горизонтальное расширение, георазпределение, отказоустойчивость)

3. **Бесшовное горизонтальное масштабирование без ребалансировки. «АМУР»** спроектирован для неограниченного роста кластера — можно добавлять новые узлы и диски, не останавливая систему и **без трудоёмкого перераспределения старых данных**. Механизм динамического размещения данных (ДРД) автоматически равномерно распределяет новые данные по всем доступным ресурсам, предотвращая появление «горячих зон», и при расширении кластера не требует перестраивать уже записанные объекты. В других системах обычно требуется ребалансировка: например, в **Cloudian** добавление узлов приводит к перемещению частей данных между узлами (поскольку используется статическая разбивка по хешу), что временно нагружает систему. В **MinIO** при наращивании количества серверов администратору зачастую приходится формировать новый пул и перемещать данные, либо принимать временную диспропорцию заполнения. **«АМУР»** избавляет от этой сложности — новые ресурсы сразу используются, а существующие данные остаются защищёнными без необходимости перемещения.



4. **Георазпределение с единым пространством имён и устойчивостью к отказу площадок. «АМУР»** поддерживает несколько моделей георазпределения. В режиме **3GEO** данные автоматически рассеиваются (с использованием восстанавливающего кодирования) по трём географически разнесённым площадкам в рамках единого кластера. При выходе из строя целого дата-центра система продолжает работу: оставшиеся две площадки содержат достаточное количество фрагментов данных и фрагментов чётности для восстановления объектов на лету. Это означа-



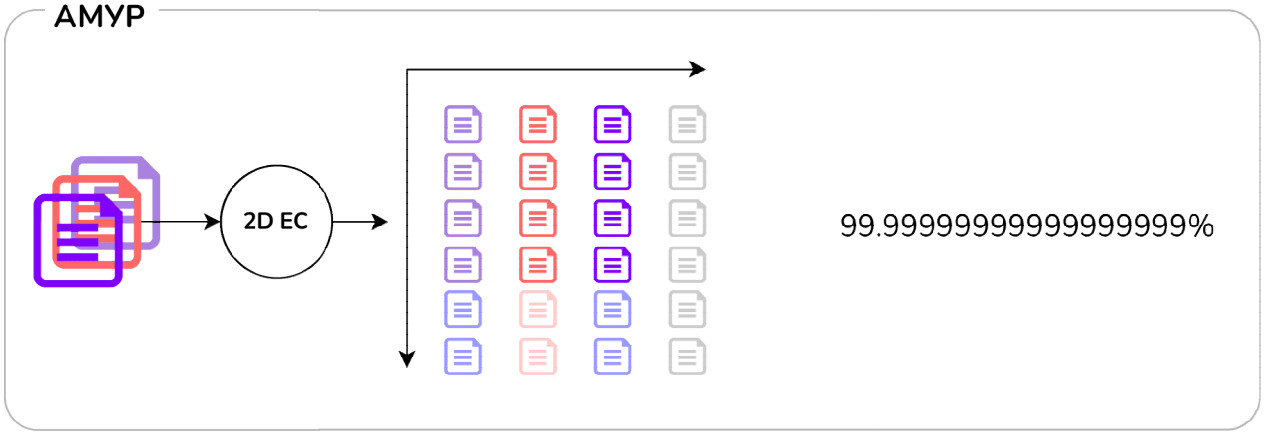
ет непрерывную доступность данных даже при катастрофе в одном центре обработки данных (ЦОД), *без* ручного переключения — уровень надёжности, которого по умолчанию нет у **MinIO** и **Cloudian**. (**Cloudian** обычно достигает геозащиты через асинхронную репликацию между кластерами или через зеркальные копии, что требует тройного объёма хранения для трёх ЦОД, либо запускается как георазпределённый кластер Cassandra с повышенными накладными расходами. **MinIO** предлагает только active-active репликацию между отдельными кластерами, но не имеет встроенного **восстанавливающего кодирования** через несколько площадок.) Помимо 3GEO, **«АМУР»** поддерживает 2-ЦОДовые схемы: двунаправленную асинхронную репликацию между кластерами (2GEO) или репликацию на публичные облака для резервирования. Горизонтальная масштабируемость **«АМУРА»** также проявляется внутри одного ЦОД — можно начинать с 3 узлов и наращивать до десятков и сотен, объединяя их в единое хранилище с общим пространством имён. **Cloudian** также масштабируется до сотен узлов и петабайт, однако за счёт архитектуры (спроектированной для мультиарендных окружений с Cassandra/Redis) имеет более сложное управление при росте. **«АМУР»** предлагает более простое управление масштабом: единая консоль видит весь кластер, а сильная согласованность данных упрощает приложениям работу в распределённой среде.

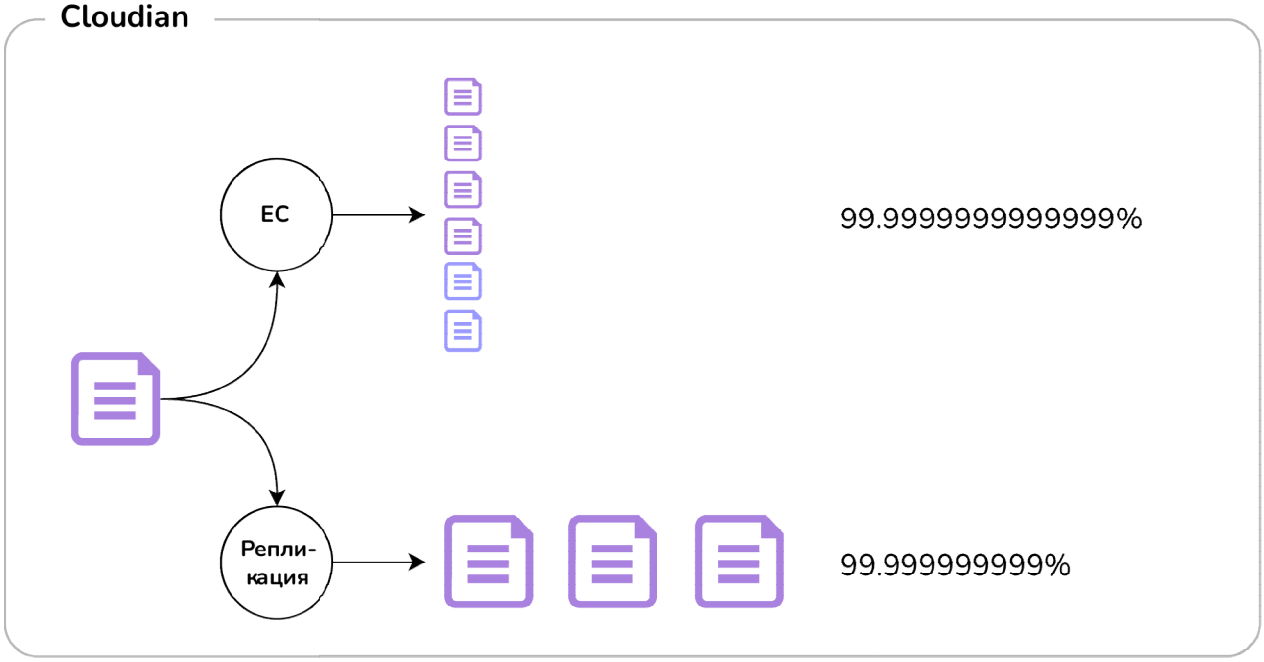
5. **Высокая отказоустойчивость на уровне компонентов.** В пределах одного ЦОД **«АМУР»** автоматически распределяет **фрагменты объектов и блоки чётности** по всем узлам и дискам кластера таким образом, чтобы выдерживать одновременный выход из строя нескольких узлов или накопителей без потери данных. Типичные политики кодирования **«АМУРА»** позволяют пережить отказ до 5 дисков одновременно в одном кластере, сохраняя доступность данных. При этом **«АМУР»** обеспечивает строгую согласованность: все узлы сразу видят актуальные данные, никаких «окон недоступности» или конфликтов при отказах узлов не возникает. Для сравнения, **MinIO** в распределённом режиме требует, как минимум, кворума узлов для операций записи/чтения; при отказе диска данные остаются **деградированными** (с пониженной избыточностью) до замены диска и запуска процедуры исправления, что увеличивает окно уязвимости. **Cloudian** при использовании восстанавливающего кодирования тоже способен выдерживать потерю нескольких устройств, но его схема восстановления задействует Cassandra и может быть более ресурсозатратной, особенно если требуется ребалансировка данных на новые узлы для восстановления уровня защиты. В **«АМУРЕ»** же расширение и восстановление — прозрачные и не требуют ручного вмешательства, что повышает общую надёжность при масштабировании.

Надёжность хранения данных (защита от потерь, автоматическое восстановление)

6. **Уникальное двухмерное восстанавливающее кодирование (2D EC) для максимальной устойчивости. «АМУР»** применяет собственный алгоритм двухмерного кодирования на основе кодов Рида-Соломона, который распределяет фрагменты данных и паритетные фрагменты как **внутри** площадки, так и **между площадками** (в многоузловых или геоконфигурациях). Это позволяет достичь экстремально высокой надёжности без чрезмерного расхода ёмкости. Например, в конфигурации с интегрированным холодным хранилищем на лентах 2D EC даёт долговечность данных до *19 девяток* (99.9999999999999999% сохранности) при совокупных накладных расходах ~15% сверх объёма данных. Для понимания: 19 девяток означают вероятность потери 1 объекта из 10 миллиардов лишь раз в миллиарды лет — практически абсолютная защита. Даже в чисто дисковом развертывании **«АМУРА»** стандартные политики обеспечивают долговечность вплоть до 19 девяток благодаря сочетанию широкого «размазывания» данных по многим дискам и интеллектуального восстановления. **Cloudian** HyperStore при типичной конфигу-

рации (например, кодирование 6+3 или 8+2) достигает 14-15 девяток, а при репликации 3 копий — около 11 девяток, то есть **«АМУР»** может предоставить более высокую гарантию сохранности. **MinIO** как программное решение сам по себе не сертифицирует уровень «девяток», всё зависит от выбранной схемы EC и резервирования; для сравнимой надёжности администраторы **MinIO** зачастую вынуждены делать большее число фрагментов четности или копий, что снижает эффективность хранения.





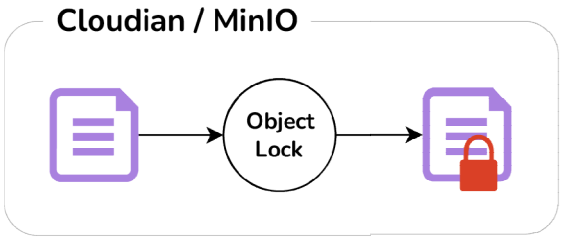
7. **Автоматическое самовосстановление без вмешательства. «АМУР»** постоянно контролирует целостность данных с помощью фоновых процессов динамического восстановления данных (**ДВД**). Система сканирует хранилище, вычисляет и проверяет контрольные суммы фрагментов, предсказывает сбои накопителей и **проактивно** восстанавливает утраченную избыточность. При обнаружении сбоя или повреждения блока, **«АМУР»** немедленно реконструирует недостающий фрагмент на исправных дисках, не дожидаясь физической замены диска. Восстановление происходит параллельно, задействуя максимальную степень параллелизма по всем узлам, благодаря чему **сокращается время реконструкции** и минимизируется окно уязвимости. Для сравнения, в ряде других решений восстановление данных начинается лишь после замены диска (особенно если нет зарезервированных «горячих» свободных дисков). Например, в классическом RAID или старых версиях **Cloudian** простой диска мог означать многочасовое восстановление на новый диск, что опасно при больших объёмах. **«АМУР»** же способен перестроить утраченные фрагменты в свободном пространстве существующих дисков, поддерживая уровень защиты. **MinIO** имеет механизм фонового сканирования и исправления, но администратору нужно его инициировать или настроить, — **«АМУР»** делает всё автоматически и непрерывно.

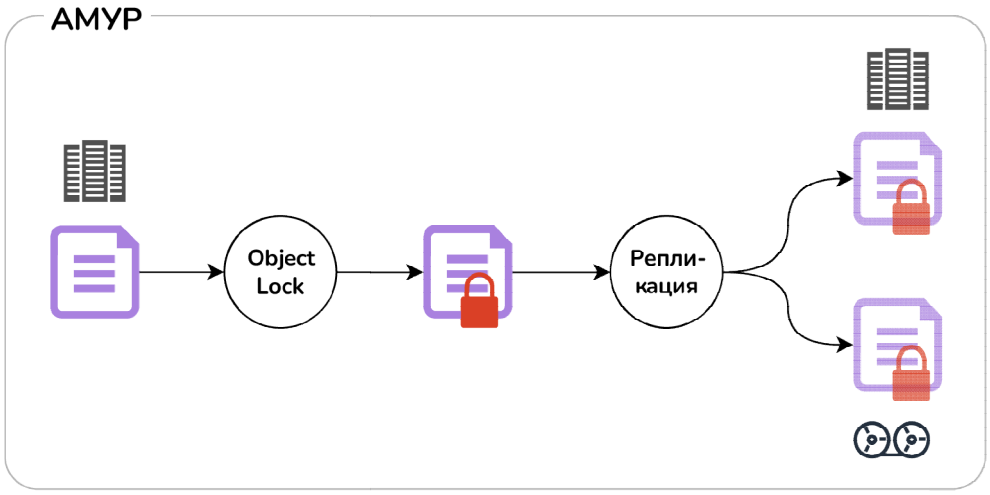
8. **Защита от битовых ошибок и тихой порчи данных.** Каждая сегментированная часть объекта в **«АМУРЕ»** содержит CRC-коды для проверки целостности. При каждом чтении данные валидируются, и, если обнаружено даже **незаметное** повреждение (битовый сбой, сбой блока на диске), объект автоматически восстанавливается с использованием остальных фрагментов, а повреждённая копия заменяется исправленной (процесс самовосстановления). Такая внимательность к целостности означает, что **«АМУР»** не только защищает от катастрофических потерь, но и сохраняет данные неизменными десятилетиями, предотвращая постепенную деградацию (в частности, bit rot). Аналогичные механизмы в той или иной степени есть у конкурентов (**Cloudian** при чтении проверяет хэши в Cassandra, **MinIO** хранит хэши объектов), но реализация **«АМУРА»** более комплексная: проверки выполняются вне основного пути данных и непрерывно, что обеспечивает длительную **непрерывную** проверку здоровья данных (особенно важную на архивных «холодных» данных). В итоге, **«АМУР»** может похвастаться тем, что данные защищены как от внезапных отказов, так и от накопительных ошибок хранения.

Безопасность и соответствие требованиям (неизменяемость, шифрование, контроль доступа)

9. **Полное шифрование данных и метаданных без дополнительной доработки.** **«АМУР»** обеспечивает шифрование данных **всех** уровней: передача шифруется по SSL/TLS, а хранение — с применением шифрования без необходимости ручного управления ключами (keyless encryption). Важно, что шифруются не только содержимое объектов, но и их пользовательские метаданные, — это защищает **всю** информацию от несанкционированного доступа (многие системы конкурентов шифруют только данные, оставляя служебные метаданные в открытом виде). **Cloudian** HyperStore также поддерживает серверное шифрование SSE с интеграцией внешних KMS (например, через KMIP) и клиентское шифрование; **MinIO** предлагает кодирование объекта с помощью Master Key или внешнего KMS. Однако **«АМУР»** выделяется тем, что шифрование встроено по умолчанию на уровне платформы — администратор получает максимальную безопасность без сложной интеграции.

10. Гарантии неизменяемости данных и соответствие регуляторным требованиям (WORM). **«АМУР»** полностью совместим с *S3 Object Lock* — функцией неизменяемости объектов. Можно пометить данные как не подлежащие удалению или перезаписи в течение заданного времени (режимы Governance или Compliance), что реализует требования правил типа SEC 17a-4(f) и защищает резервные копии от программ-вымогателей. **«АМУР»** предотвращает даже администраторское удаление защищённых копий, обеспечивая истинный WORM-режим. MinIO в режимах Enterprise тоже поддерживает Object Lock (в т.ч. для интеграции с Veeam), но в открытой версии это требует сборки с флагом «worm» или использования физической неизменяемости носителей. Cloudian предлагает функцию Object Lock начиная с версии 7 и её активно используют финансовые организации для соответствия регуляторам — по возможностям WORM Cloudian и **«АМУРА»** примерно равны. Однако, **«АМУР»** имеет преимущество в том, что объекты могут оставаться неизменяемыми даже при межкластерной репликации: при копировании на другой **«АМУР»** или в облако, система сохраняет флаги неизменности и единый срок хранения. Также **«АМУР»** — единственное решение, предлагающее Glacier-to-Glacier репликацию (репликация неизменяемых ленточных архивов между ЦОД) для многократного резервирования холодных данных, что полезно для строгих требований защиты.





11. **Многоуровневая система доступа, интеграция с службами каталогов и аудит.** **«АМУР»** поддерживает многопользовательский доступ с разграничением прав: через веб-интерфейс администратор может создавать учетные записи, **генерировать ключи доступа S3** для разных пользователей или приложений, устанавливать квоты и политики для них. Существует интеграция с внешними службами каталогов — система может аутентифицировать запросы через LDAP/Active Directory или RADIUS, что облегчает включение **«АМУРА»** в существующую инфраструктуру предприятия. Например, можно подключить **«АМУР»** к домену AD и управлять доступом на основе групп безопасности, а для NFS-доступа реализована поддержка Kerberos/AD для удостоверений пользователей (критично при работе с файловым доступом в корпоративных окружениях). Кроме того, **«АМУР»** поддерживает протоколы RADIUS, SAML (для доступа к веб-интерфейсу), и обладает собственным API для управления пользователями и ключами — то есть возможна интеграция с внешними IAM-системами. **Cloudian** изначально ориентирован на мультиарендность и тоже имеет богатые возможности интеграции с AD/LDAP, а также собственные механизмы контроля (группы, ACL, Buckets Policies). **MinIO** же предоставляет базовую IAM с политиками и может интегрироваться с Identity Provider (например, через OpenID Connect) для управления консолью, но *прямой*

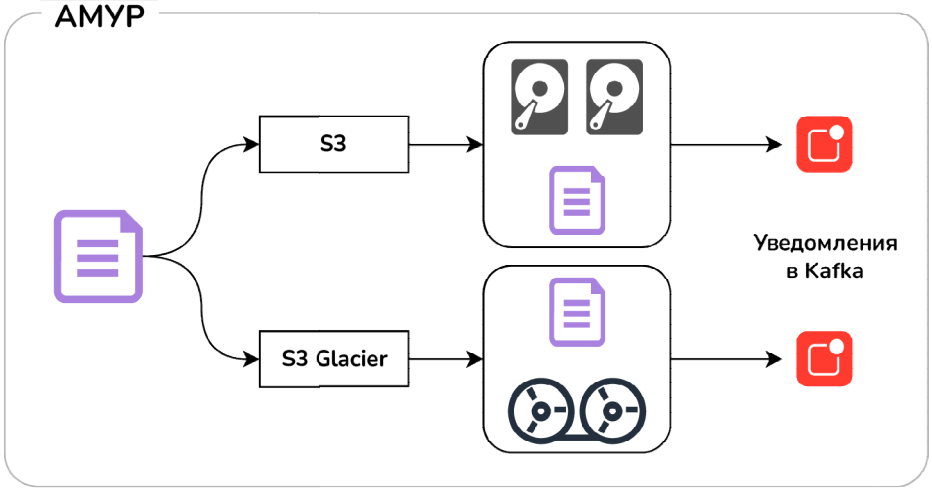
LDAP-авторизации запросов S3 в оригинальной поставке нет — обычно приходится использовать внешний прокси. Преимущество **«АМУРА»** — **готовность к использованию в сетях крупных и государственных предприятий**: учётные записи, группы, квоты, интеграция с AD/LDAP в оригинальной поставке — всё настраивается через Web UI, что снижает трудозатраты на подключение хранилища к существующей системе. Помимо разграничения прав, «**АМУР**» ведёт детальный аудит: фиксируются в журнал все операции чтения, записи, удаления и администрирования (включая метаданные), что важно для соответствия требованиям безопасности и расследований инцидентов.

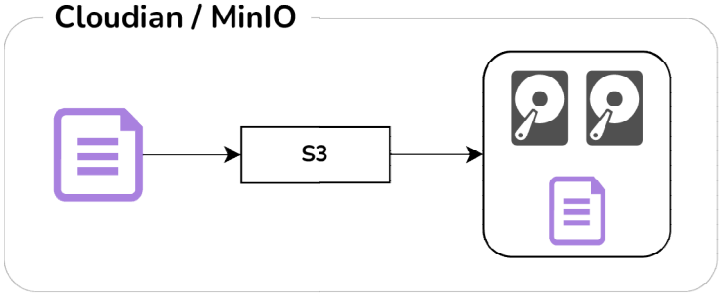
12. **Соответствие корпоративным стандартам и простота управления безопасностью. «АМУР»** обеспечивает строгую согласованность данных — все узлы кластера сразу видят подтверждённые записи. Это устраняет целый класс проблем безопасности, связанных с **устаревшими копиями** (например, когда один узел ещё возвращает старую версию объекта). В **«АМУРЕ»** нет компромисса между согласованностью и производительностью, даже в 3GEO-конфигурации данные согласованы без потери скорости. Это важно для приложений, требующих уверенности в том, что прочитанные данные — самые актуальные (банковские, медицинские системы и т. д.). Кроме того, **«АМУР»** поддерживает шифрование на уровне контейнера, версионирование объектов (для восстановления случайно удалённых или повреждённых данных), нотификации событий в Kafka (что позволяет строить системы SIEM/SOC, реагирующие на определённые операции). В совокупности всё это делает **«АМУР»** сильным в плане соответствия политикам безопасности предприятия и требованиям нормативов по хранению данных.

Интеграция с внешними системами (каталоги пользователей, S3 API, резервное копирование, NFS/SMB)

13. **Полная совместимость с Amazon S3 API (включая Glacier) + расширения. «АМУР»** реализует стандартный S3-протокол для основной работы (хранение в «Standard» классе) и **поддерживает S3 Glacier API** для работы с холодным архивом. Это означает, что приложения могут использовать знакомые SDK и REST-запросы для интеграции — **«АМУР»** без проблем работает как целевой S3-совместимый контейнер для систем резервного копирования (Veeam, Commvault, Veritas и т. д.) и аналитических платформ (Splunk, Hadoop — через S3A). Более того, **«АМУР»** дополнительно предлагает уведомления о событиях (аналог AWS

SNS/SQS) — при операциях S3 генерируются сообщения в Kafka, позволяя связывать хранилище с системами потоковой обработки данных, бессерверными триггерами и т. п., чего нет в оригинальной поставке у **Cloudian** и **MinIO**. **Cloudian** обеспечивает практически полную совместимость с S3 (включая большинство API функций, версионирование, Object Lock), **MinIO** также целенаправленно совместим с базовыми операциями протокола S3. Но **«АМУР»** выгодно отличается тем, что поддерживает **одновременно стандарт S3 и Glacier**, позволяя приложениям архивировать данные по тому же протоколу S3 — конкуренты обычно предлагают отдельные многоуровневые механизмы вместо непосредственной поддержки

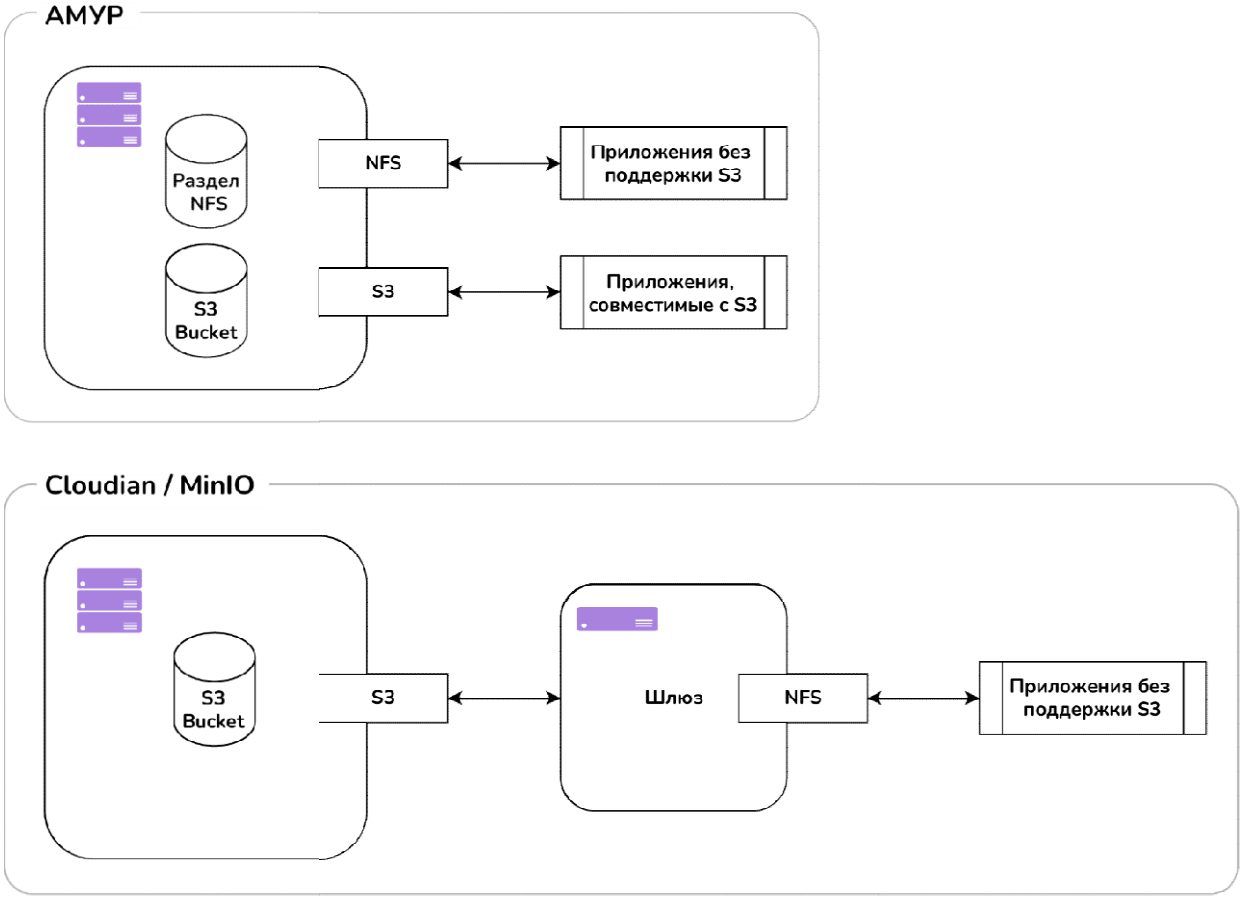




Glacier API.

14. **Встроенный файловый интерфейс (NFS) для работы с неглубоко интегрированными приложениями. «АМУР»** умеет предоставлять данные не только по протоколу S3, но и в качестве каталога файловой системы NFS: система поддерживает монтирование разделов через NFS v3. Это упрощает подключение **«АМУРА»** к существующим корпоративным приложениям, которые ожидают файловую систему (например, для аппаратных средств, научных приборов, систем видеонаблюдения

или резервного копирования по NFS). В **«АМУРЕ»** можно создать том NFS, данные в нём будут лежать в том же объектном хранилище (с теми же преимуществами защиты), но будут доступны по протоколу файлового доступа. **Cloudian** предоставляет файловый доступ только с помощью дополнительного модуля **HyperFile** (дополнительный программный шлюз, лицензируется отдельно), что усложняет инфраструктуру. **MinIO** не имеет встроенного SMB/NFS-сервиса — обычно применяется внешний шлюз (например, **MinIO** NAS Gateway или s3fs) с ограниченной производительностью и без сохранения всех свойств S3. Таким образом, **«АМУР»** выигрывает простотой: в нем доступна возможность организации как контейнеров с доступом через S3, так и файловых систем NFS, без необходимости использования внешних сервисов и дополнительных компонентов.



15. **Совместимость с ПО резервного копирования и корпоративными приложениями. «АМУР»** сертифицирован и протестирован с различными системами резервного копирования и архивирования. Благодаря поддержке S3 API и Object Lock,

«АМУР» отлично подходит как целевое хранилище для **резервных копий с неизменяемостью** (например, Veeam Backup & Replication — режим неизменяемых резервных копий для защиты от вредоносных программ). Аналогично, **Cloudian** активно продвигается как целевое хранилище резервных копий (у них есть интеграции с Veeam, Rubrik и др.), **MinIO** часто используется для резервирования в открытом программном обеспечении. Но **«АМУР»** даёт дополнительные бонусы: например, прямая интеграция с продуктами «ДИАМАНТ» — система может выступать архивным слоем для файловой системы «ДИАМАНТ» **«ВОЛГА»**, принимая данные по S3 и автоматически перемещая их на ленты. Кроме того, **«АМУР»** поддерживает *двунаправленную репликацию* между локальным хранилищем и публичными облаками. Это позволяет, к примеру, синхронизировать резервные копии в **«АМУРЕ»** с копиями в Amazon S3, Azure Blob или Яндекс.Облаке, или наоборот — использовать **«АМУР»** как локальный промежуточный буфер (кэш) к облачному архиву. Такие сценарии гибридного облака также возможны в **Cloudian** (есть функция Auto-Tiering и Cloud Mirror для выгрузки в AWS/Azure), **MinIO** предлагает процесс перемещения между уровнями хранения на внешний S3. Но **«АМУР»** предоставляет эти возможности в единой платформе, управляемой централизованно.

Мониторинг и открытые стандарты. «АМУР» интегрируется с системами мониторинга через Prometheus и Grafana — метрики кластера доступны локально или глобально и могут экспортироваться в Prometheus для построения графических интерфейсов (дашбордов). Также поддерживаются уведомления по e-mail, уведомления SNMP TRAP и rsyslog — всё для встраивания в существующие процессы IT-операций. **Cloudian** имеет собственную консоль (CMC) и может отправлять уведомления SNMP, но Prometheus/Grafana напрямую не встроены. **MinIO** предоставляет Prometheus-метрики, однако их детализация значительно ниже, чем у **«АМУРА»**. Преимущество **«АМУРА»** — готовые средства отслеживания состояния и производительности системы **со встроенными предупреждающими аналитическими функциями**. Это облегчает сопровождение, особенно когда **«АМУР»** интегрирован в большой парк корпоративных систем.

	Prometheus + Grafana	Email	SNMP TRAP	Remote Syslog
АМУР	✓ Расширенные метрики	✓	✓	✓
Cloudian		✓	✓	
MinIO	✓ Малое количество метрик			✓

Архитектурные особенности (двумерное восстанавливающее кодирование, NVMe для метаданных, политика жизненного цикла и пр.)

16. **Инновационное двумерное кодирование и широкополосное распределение данных.** Как уже отмечалось, **двумерное восстанавливающее кодирование** (2D EC) — фирменная особенность **«АМУРА».** На практике это означает более **умное распределение данных**: данные защищаются как внутри отдельного сайта (размещением по десяткам дисков), так и между ЦОД или библиотеками хранения. Такая многоуровневая избыточность даёт сочетание высокой эффективности (меньше лишних копий) и устойчивости к разнородным сбоям — например, система 3GEO с 2D EC может восстановить объект, даже если полностью потеряна одна площадка *и одновременно* несколько дисков на другой площадке. Конкуренты, как правило, используют одномерное кодирование: **Cloudian** — либо локальное EC по дискам, либо репликацию между дата-центрами. **MinIO** — классическое EC внутри своего набора узлов. **Двухмерная схема реализованная в «АМУРЕ» позволяет достичь тех самых «19 девяток» надёжности** и уникальной способности работать как распределённое хранилище с одним именем во множестве локаций. Для пользователя архитектура 2D EC прозрачна, но именно она отличает **«АМУР»** как решение класса **корпоративный архив (enterprise archive)**, рассчитанное сохранение данных на протяжении десятков лет.

17. **Двухуровневая архитектура: разделение слоя доступа и хранения. «АМУР»** построен по принципу двух основных слоёв — слой доступа и слой хранения данных. **Слой доступа** отвечает за интерфейсы (S3/NFS), аутентификацию, шифрование и хранение метаданных объектов. Он представляет собой набор контроллеров (каждый узел **«АМУРА»** совмещает функции доступа и хранения, но логически слои разделены), где и размещена высокопроизводительная база метаданных на NVMe. **Слой хранения** занимается непосредственно сохранением пользовательских данных на носителях — жёстких дисках для “горячего” класса Active и лентах для «холодного» класса. Такое разделение даёт важные преимущества: можно **масштабировать слои независимо** под нужды нагрузки. Если требуется больше производительности (больше операций в секунду, больше одновременных соединений, рост количества мелких объектов) — расширяют Слой доступа, добавляя узлы-контроллеры (CPU, NVMe, сеть). Если нужен объём — добавляют ёмкость в Слой хранения (новые дисковые полки, ленточные библиотеки) без необходимости увеличивать количество ресурсов CPU. **Cloudian** и **MinIO** не имеют чётко обособленного слоя доступа: каждый узел у них выполняет и функции API, и хранит данные. Масштабирование, соответственно, «связано» — добавляя узел в **Cloudian/MinIO**, вы наращиваете и CPU, и емкость хранения одновременно, но не можете увеличить одно без другого. **«АМУР»** же позволяет, например, подключить дополнительный сервер доступа (с NVMe) для разгрузки операций, не трогая уже заполненные диски. Это повышает гибкость архитектуры и оптимизирует затраты на масштабирование.

    NVMe для метаданных и ускорение операций управления. В **«АМУРЕ» вся база метаданных объектов хранится на NVMe-накопителях** высокого класса производительности. Метаданные включают не только имена, размеры, атрибуты объектов, но и сведения о том, на каких фрагментах/дисках лежат части объекта, версии, политики жизненного цикла и пр. За счёт размещения на NVMe **«АМУР»** достигает очень быстрого отклика на операции, связанные с метаданными: перечисление миллионов объектов в контейнере, получение списков, частые операции создания/удаления небольших файлов — всё это происходит значительно быстрее, чем на системах, где метаданные хранятся на HDD или в сторонней базе. **Cloudian** хранит метаданные распределённо в Cassandra (которую обычно размещают на SSD, но не всегда NVMe, и с дополнительными накладными расходами на индексы). **MinIO** вовсе избегает внешних баз — оно сохраняет информацию внутри файловой структуры на дисках, что на больших масштабах может замедлять поиск и просмотр списка файлов в папке. В **«АМУРЕ»** же

метаданные расположены централизованно в NVMe-пуле и **реплицированы между контроллерами** для устойчивости. Это позволяет выдавать согласованные результаты мгновенно в ответ на запросы HEAD/GET и прочие управляющие операции. Например, при сценарии с миллиардами объектов **«АМУР»** способен отвечать на запросы списков значительно стабильнее и быстрее, чем **MinIO** (которому пришлось внедрять встроенное хранение маленьких объектов, чтобы немного уменьшить количество операций по файловой системе). Преимущество NVMe-метаданных особенно заметно под смешанной нагрузкой: операции записи/чтения больших файлов идут на HDD, а параллельно десятки тысяч мелких операций обслуживаются флеш-контроллерами, не блокируя друг друга.

Политики жизненного цикла и автоматическое двухуровневое хранение. «АМУР» — единственная платформа, где в рамках одного хранилища существуют два класса: **Активный (диск/флеш)** и **Холодный (лента)**. Архитектурно это реализовано прозрачно: приложения видят единое пространство имён, а администратор может настроить политики жизненного цикла, которые автоматически переводят объекты из активного слоя в холодный и обратно Например, можно задать политику: все объекты старше 90 дней в контейнере X перемещать на ленточные носители (в класс хранения Glacier) — и **«АМУР»** сам упакует данные, перенесёт их в ленточное хранилище и пометит их в каталоге как **архивные**. При обращении к таким данным **«АМУР»** действует в соответствии с протоколом S3 и по команде приложения восстановит их во временный кэш на дисковом уровне хранения, после чего может отдать их по запросу (время восстановления измеряется в минутах, как и у AWS Glacier). Ключевое преимущество — пользователь платит только за один продукт и управляет политиками из одной консоли, получая экономию хранения холодных данных до 80% по затратам, не жертвуя доступностью (все классы доступны через S3 API). **Cloudian** не имеет собственного ленточного уровня; хотя возможна интеграция с внешними архивами (например, **Cloudian** может автоматически перемещать данные между уровнями хранения в AWS Glacier или в систему типа BlackPearl), это всегда надстройка, требующая соединения с внешним сервисом и часто дополнительных лицензий. **MinIO** тоже может перемещать данные в другое хранилище (через механизм Tiering rules), но у него нет понятия внутреннего холодного уровня хранения — это просто копирование на S3-совместимый архив и удаление локального объекта (что влечёт риск, например, ошибки в целевом хранилище трудно отслеживать). **«АМУР»** же даёт **сквозной контроль**: единый кластер сам управляет двумя уровнями и знает, где находится объект, обеспечивая

при этом **нулевую плату за выход/вход** (в отличие от публичных сервисов с поддержкой уровня хранения Glacier, где взимается плата за каждое извлечение). Таким образом, архитектурно **«АМУР»** предоставляет максимальную гибкость по месту хранения данных в зависимости от их актуальности.

18. **Бесшовные обновления и расширения без простоев.** Архитектура **«АМУРА»** предусматривает **скользящее обновление** — программное обеспечение и прошивки узлов обновляются последовательно, без остановки всего кластера. Данные и сервисы остаются доступными благодаря тому, что компоненты кластера распределены и дублированы (метаданные на нескольких узлах, избыточность данных). **Cloudian** также поддерживает поузловое обновление, но из-за зависимости от Cassandra обновления могут требовать больше координации. **MinIO**, будучи более простым, обновляется быстро, но в крупных кластерах тоже рекомендуется поэтапное развёртывание. Преимущество **«АМУРА»** — протестованные производителем процедуры обновления, минимизирующие риски («ДИАМАНТ» регулярно обновляет **ПО «АМУР»** и обеспечивает совместимость новых версий с данными предыдущих). Кроме того, **расширение ёмкости** архитектурно не требует каких-либо специальных действий — просто добавляются новые узлы или дисковые полки, и система автоматически включает их в работу. Нет необходимости, к примеру, переразбивать «зоны» хранения вручную (как в некотором открытом программном обеспечении) или планировать сегментирование базы данных (шардирование) — **«АМУР»** делает масштабирование действительно горизонтальным и бесшовным.

Стоимость владения (CAPEX/OPEX, эффективность хранения, гибкость архитектуры)

19. **Высокая эффективность хранения и низкие накладные расходы.** Благодаря мощному восстанавливающему кодированию, **«АМУР»** достигает очень высокого коэффициента полезного использования дисков. В режимах для одной площадки система даёт до ~80% полезной ёмкости от физической (например, политика 20+4 — 83%, 18+5 — ~78%). Для сравнения, традиционная тройная репликация использует лишь ~33% ёмкости, а даже EC 6+3 (**Cloudian**) — ~66%. То есть при равной защищённости **«АМУР»** хранит больше данных на том же объёме носителей, снижая **CAPEX** на диски. Отдельно стоит отметить, что **холодный класс «АМУРА»** (ленты) экономичнее на порядки: стоимость за терабайт на ленте в разы

ниже дисковой, плюс ленточные библиотеки практически не потребляют электроэнергию вне операции чтения/записи. «ДИАМАНТ» заявляет, что **«АМУР.ЛЕДНИК»** снижает стоимость хранения холодных данных до 80% по сравнению с хранением их на дисках. Ни **MinIO**, ни **Cloudian** не предлагают встроенного решения на ленточных носителях — достигнуть таких низких затрат с ними можно только выгружая данные во внешние дешёвые хранилища (что добавляет операционные расходы и сложность).

20. **Сокращение OPEX за счёт автоматизации и простоты управления. «АМУР»** спроектирован по принципу «упростить масштабирование, не увеличивая сложность» — управление кластером осуществляется из единого веб-интерфейса администратора или через API, где доступно всё: от создания пользователей и контейнеров до мониторинга и обновлений. Система сама выполняет рутинные задачи (балансировка, самовосстановление, расширение), требуя минимального вмешательства администраторов. Это означает меньше человеко-часов на поддержку, меньшие риски ошибок и, как следствие, снижение операционных затрат. **Cloudian**, хотя и предоставляет удобную консоль, в крупномасштабных развертываниях может требовать настройки Cassandra (например, управлять размерами кластеров, следить за временем GC и т.д.), что подразумевает наличие компетенции DBA/NoSQL у команды — это дополнительный OPEX. **MinIO** за счёт простоты требует мало администрирования на малых масштабах, но при росте до нескольких петабайт пользователи отмечают необходимость вручную разбираться с производительностью, возможными несоответствиями конфигураций, отсутствием «из коробки» некоторых инструментов, применяемых в окружениях корпоративного уровня (например, встроенного аудита, уведомлений) — всё это время специалистов. **«АМУР»** же включает все необходимые функции (мониторинг, предупреждения, аналитику) **изначально**, к тому же поставляется с поддержкой от «ДИАМАНТ». Таким образом, стоимость поддержки **«АМУРА»** предсказуема и ниже, особенно при длительной эксплуатации.

21. **Оптимизация капитальных затрат через гибкость приобретения и эффективное использование ресурсов. «АМУР»** доступен не только как традиционное ПО/аппаратный комплекс, но и в виде подписки (as-a-Service) на мощностях «ДИАМАНТ» — что позволяет превратить CAPEX в OPEX при необходимости. Но даже в классическом сценарии покупки аппаратного обеспечения **«АМУР»** экономит капитал: благодаря высокой производительности на узел требуется меньше

серверов для достижения заданной пропускной способности или IOPS. Например, флеш-узел **«АМУР Z»** с 32 ядрами выдаёт на ядро вдвое больше операций, чем типовые 64-ядерные узлы альтернативных продуктов, то есть для той же нагрузки может потребоваться меньше узлов (меньше расходы на оборудование, лицензии и поддержку). Кроме того, отсутствие необходимости отдельных шлюзов для NFS/SMB или отдельных узлов для метаданных означает, что все приобретённые ресурсы используются максимально эффективно. В **Cloudian** часть ресурсов тратится на работу Cassandra (RAM/CPU на хранение индексов), а при необходимости файлового доступа надо закупать дополнительные серверы HyperFile. В **MinIO** при росте объёма хранения часто приходится разбивать данные по отдельным кластерам (федерация) из-за ограничений масштабируемости, что ведёт к дублированию инфраструктуры. **«АМУР»** же масштабируется как единое целое, что позволяет избегать избыточных затрат.

22. **Гибкость архитектуры снижает риск дорогостоящих «переделок».** При проектировании хранения на годы вперёд трудно предугадать, как изменятся профили данных (горячие/холодные) и требования. **«АМУР»** обеспечивает уникальную гибкость: он одинаково эффективен для активных данных (можно добавить NVMe-флеш узлы для ускорения работы) и для архивных (возможность подключения дополнительной ленточной библиотеки для дешёвого роста ёмкости). Кластер **«АМУРА»** можно легко преобразовать или дооснастить под новые задачи без замены всей системы. Например, если через несколько лет доля «горячих» данных у организации вырастет — достаточно добавить несколько all-flash узлов в существующий кластер, и **«АМУР»** начнёт использовать их для ускорения доступа, сохраняя остальные данные на дешёвых HDD. В альтернативных решениях подобный апгрейд потребовал бы внедрения нового класса хранилищ и миграции данных (что дорого и рискованно). С **«АМУРОМ»** же инвестиции защищены: архитектура масштабируется **вверх и вглубь** по всем измерениям (производительность, объём, география) без «форс-мажоров». «ДИАМАНТ» подчёркивает, что **«АМУР»** обладает самой низкой совокупной стоимостью владения в отрасли среди масштабируемых объектных хранилищ именно благодаря сочетанию факторов: минимум резервных копий за счёт EC, автоматизация, интеграция холодного уровня хранения, эффективное использование оборудования и долговременная адаптивность платформы под задачи пользователя. Все эти преимущества дают **«АМУРУ»** заметный выигрыш в полной стоимости владения по сравнению с разворачиванием эквивалентных решений на базе **MinIO** или **Cloudian**.