Неограниченно расширяемое хранилище iNc

Вычисления в памяти: новое понимание хранения данных

LC-INC-003 1.0

Основная концепция

Основная концепция

Преобразуйте устройства хранения данных из топологии «централизованного управления (контроллер + пассивное NAND)» в «децентрализованную/распределенную вычислительную сеть (ячеистую сеть вычислений NAND)».

Используя характеристики RLNC (случайное линейное сетевое кодирование) без сохранения состояния в сочетании с возможностями вычислений внутри Nand (iNc), теоретически возможно преодолеть физические ограничения традиционного CE (Chip Enable) традиционного управления SSD (Chip Enable) и количества каналов и достичь архитектуры бесконечного расширения, аналогичной «Интернету хранения».

Если эта архитектура будет успешно реализована, Lingcun получит преимущество, с которым не смогут сравниться конкурирующие продукты: надежность на уровне системы.

Это решение выходит за рамки ограничений физического уровня и использует математику (RLNC) и логику (INC + MRD) для создания срока службы хранилища, при котором теоретически никогда не происходит потери данных и который физически бесконечно расширяем.

Это самое передовое пересечение в глобальной сфере хранения данных. Если Lingcun сможет возглавить внедрение этой «самобалансирующейся системы хранения данных без мастера на основе RLNC» на FPGA, она изменит определение того, что такое «жесткий диск».

1. Анализ осуществимости

Анализ технической логики

Техническая логика установлена. Трудность аппаратной реализации связана не с хранилищем, а с топологией межсоединения.

- Традиционный SSD: основным элементом управления является «мозг», а NAND — «склад». Если склад слишком велик, мозг с ним не справится (ограничение CE).

- Текущая идея: Каждый NAND становится «интеллектуальным узлом», который считается существующим. Данные больше не хранятся в фиксированной точке, а передаются и размещаются в сети в виде фрагментов, закодированных RLNC.

Основные проблемы: Стандартная флэш-память NAND (даже с возможностями вычислений в памяти) физически является ведомым устройством (Slave Device). Он не имеет источника синхронизации и не может активно инициировать передачу (Master).

- Суть NAND: Стандарт NAND — это ведомое устройство (ведомое устройство). Он не имеет «автономного сознания» и будет действовать только тогда, когда получает инструкции (Чтение/Программирование). Он не знает «кто я» или «кто мой сосед».

- Требования к сети RLNC: Каждый узел должен быть автономным агентом.

- Необходимо активно инициировать рукопожатие.

- Тут нужно судить о ТТЛ (время жить).

- Необходимо управлять кредитом (управление потоками).

- Ему необходимо активно «проталкивать» пакеты данных соседям (Пересылка).

Чистые схемы массивов хранения не могут обеспечить такое «активное поведение». Должно быть логическое ядро (конечный автомат или микроконтроллер), чтобы придать ему «интеллект», этот мозг должен находиться на логическом кубике.

Физический уровень высокоскоростного соединения (SerDes PHY)

Поскольку межчиповое соединение требует высокоскоростного SerDes (дифференциального последовательного интерфейса).

- Физическая реальность: Схема SerDes содержит PLL (ФАПЧ), CDR (Восстановление тактовой частоты), Equalizer (Эквалайзер). Это аналоговые схемы смешанных сигналов.

- Ограничения NAND: Процесс производства флэш-памяти NAND очень затрудняет создание высокопроизводительных аналоговых схем. Если вы принудительно создадите SerDes 16 Гбит/с на кристалле NAND, джиттер сигнала будет настолько велик, что связь будет невозможна.

- Преимущества логического кристалла: Стандартные процессы CMOS очень хорошо справляются с SerDes PHY.

Чтобы реализовать «магистраль» между чипами, физический интерфейс должен быть предоставлен Logic Die.

Энергоэффективность вычислений

Хотя можно реализовать простые битовые операции (например, XOR) путем изменения логики фиксации буфера страниц, это расчет «наилучших усилий».

- Требования к RLNC: включает умножение конечных полей GF(2^8), что требует большого количества справочных таблиц (LUT) или операций сдвига XOR.

- Сравнение эффективности:

- Делаем это рядом с массивом NAND: выполнение матричной операции может занять сотни микросекунд (нас), поскольку транзисторы не могут работать быстро.

- Сделайте это на логическом кубике: это займет всего несколько наносекунд (нс).

Без логического кристалла скорость чтения и записи «бесконечного жесткого диска» может быть такой же медленной, как у дискеты, и не сможет удовлетворить потребности искусственного интеллекта или центров обработки данных.

Smart NAND: интеллектуальное объединение кристаллов

Smart NAND (интеллектуальное стекирование) = кристалл хранения (чистый массив хранения) + связный логический кристалл (вычисления в памяти + логика маршрутизации)

В этой форме:

1. Нет необходимости в большом контроллере (главном контроллере) на уровне печатной платы в традиционном смысле этого слова.

2. Не требуется дополнительных процессоров или микросхем DRAM.

3. Но необходимо соединить тонкий логический слой CMOS через 3D-упаковку снизу (или сверху) каждой частицы NAND.

Таким образом, Logic Die все еще существует, но превратился из «большого чипа на материнской плате» в «невидимую прослойку в частице NAND».

Это также является ключом к снижению затрат:

- Пластины для хранения ориентированы на хранение (более высокий выход, более низкая стоимость).

- Платы логики ориентированы на вычисления (используется зрелый 40-нм процесс, чрезвычайно маленькая площадь, чрезвычайно дешевая).

- Как только они объединяются, они образуют «бесконечно масштабируемый узел с возможностями вычислений и маршрутизации».

Решение: Необходимо добавить минималистичный микромаршрутизатор (микроконтроллер/маршрутизатор) рядом с каждым NAND или каждой группой NAND (например, стек кристаллов) или использовать технологию вычислений в памяти, чтобы дать NAND некоторую форму возможности «драйвера шины» (это чрезвычайно сложно и обычно требует взаимодействия внешней FPGA/ASIC).

2. Подробный технический подход

Возможно ли неограниченное расширение хранилища?

Да, но физическая форма изменится.

- Логика: При использовании RLNC данные разбиваются на n закодированных пакетов. Данные существуют до тех пор, пока в сети достаточно узлов для хранения этих пакетов. Чтобы увеличить емкость, просто «подключите новый узел», и новый узел автоматически участвует в расчетах и перекодировании, без необходимости переназначения логического адреса (LBA) центральным мастером.

- Реализация: Это больше не стандартный твердотельный накопитель, а гипермасштабируемый массив. Его масштабируемость аналогична Интернету. По мере добавления узлов (NAND + микроконтроллерная логика) емкость и вычислительная мощность будут увеличиваться линейно.

Совместная коррекция ошибок MRD и LDPC

Это отличное сочетание многоуровневого управления.

- Нижний уровень (физический уровень): LDPC. Из-за переворота битов и износа оксидного слоя самой флэш-памяти NAND. Это «строительство дорог» внутри NAND.

- Верхний уровень (системный уровень): MRD (код расстояния максимального ранга). MRD превосходит традиционные коды RS при обработке ошибок в системах сетевого кодирования (таких как потеря ранга, вредоносное заражение узла, потеря пакетов).

Рабочий процесс: При расчете в памяти сначала используется LDPC для калибровки считанных данных -> уровень вычислений RLNC/MRD обрабатывает линейную комбинацию пакетов данных -> вывод/пересылка. Если частица NAND полностью повреждена (отказ узла), MRD в сочетании с функциями RLNC может восстановить данные, просто собирая пакеты кодирования достаточных размеров из окружающих узлов, не зная, какое местоположение повреждено.

Может ли NAND Flash заменить маршрутизаторы?

Техническая сложность: Как упоминалось ранее, стандартная NAND является ведомой.

Путь:

- Вариант A (вычисления в памяти +): Если технология «памяти» позволяет NAND генерировать сигналы на шине ввода-вывода, это достижимо.

- Вариант B (связанная логика): Более реалистичным подходом является архитектура NoC (сеть на кристалле). Очень маленький логический кристалл прикреплен рядом с каждой частицей NAND (или встроен в корпус). Этот логический кристалл отвечает за квитирование связи, маршрутизацию и управление шиной, тогда как ядро NAND отвечает только за хранение и матричные вычисления.

Логика пересылки: Пакет данных поступает -> вычислительный блок NAND определяет, принадлежит ли он самому себе -> если нет, он использует логический вентиль вычислений в памяти для прозрачной передачи его непосредственно на следующий порт.

Проектирование контроллера

Централизованное основное управление исчезает и развивается в «Шлюз» + «распределенное микроуправление».

- Предварительная разработка: необходимо использовать FPGA, поскольку необходимо проверить логику кодирования и декодирования RLNC и протокол связи между NAND.

- Окончательная форма:

- Сторона интерфейса (шлюз): обрабатывает протокол NVMe и притворяется стандартным жестким диском для хоста. Он отвечает только за нарезку и «внедрение» пользовательских данных в сеть NAND и не отвечает за управление конкретными физическими адресами.

- Сторона узла: Полностью децентрализована. Каждый узел NAND выполняет набор микрокода, ответственного за: прием пакетов, выполнение RLNC XOR/умножения матриц, сохранение или пересылку.

Изменения прошивки

Полный переход от FTL (Flash Translation Layer) к «стеку протоколов маршрутизации и кодирования».

- Традиционное встроенное ПО: Логический адрес -> сопоставление физических адресов, сбор мусора, выравнивание износа.

- Новая прошивка (распределена):

- Разбор заголовка: Идентифицирует вектор коэффициентов RLNC.

- Протокол сплетен: Обменивайтесь информацией о загрузке между узлами («Я сыт, не отправляйте ее мне»).

- Частичное восстановление: Если соседний узел окажется недействительным, автоматически запустится расчет реконструкции.

Проектирование времени жизни (TTL)

При пересылке сети RLNC TTL относится к «пределу переходов».

- Дизайн: Предотвращает бесконечное зацикливание пакетов в массиве NAND.

- Механизм: Каждый заголовок пакета содержит счетчик TTL. Каждый раз, когда он проходит через узел NAND для расчета/пересылки, TTL уменьшается на 1. Когда TTL = 0 и не сохраняется, пакет отбрасывается и запрашивает повторную передачу от источника (или запускает коррекцию ошибок MRD).

- Срок жизни хранилища: для «холодных» данных установите «цикл перекодирования» и используйте вычисления в памяти для регулярного считывания и обновления коэффициентов кодирования, чтобы предотвратить потерю данных, вызванную утечкой заряда.

Проектирование аппаратных схем

Топология: 2D/3D-сетка или структура тора, а не структура шины.

- Традиционный SSD: Несколько параллельных каналов (каналов), в каждом из которых установлено несколько CE.

- Массив RLNC: Аналогичен внутреннему соединению FPGA.

- Взаимодействие «точка-точка»: Чип A соединяет чип B и чип C.

- Интерфейс с высокой пропускной способностью. Традиционной 8-битной шины ONFI может быть недостаточно. Рекомендуется использовать высокоскоростной последовательный интерфейс (SerDes), аналогичный HBF, для уменьшения количества контактов и увеличения плотности межсоединений.

Формат продукта: массив или отдельный SSD?

Пройдите в два этапа.

- Форма 1 (уровень суперкомпьютеров): Массив/шкаф хранения данных, обеспечивающий внешний интерфейс Ethernet/Infiniband и большое количество внутренних узлов NAND. Подходит для проверки «бесконечного расширения».

- Форма 2 (потребительский уровень): Одна карта PCIe, которая выглядит как видеокарта и покрыта NAND. Внешний интерфейс представляет собой интерфейс NVMe. Он использует высокую одновременную пропускную способность RLNC для обеспечения максимального количества операций ввода-вывода в секунду.

Планирование соединений кристаллов HBF

Если используется HBF, его характеристики высокой пропускной способности идеально соответствуют вычислительным потребностям RLNC.

- Планирование схемы: Используйте технологию стекирования TSV (через кремний) или соединение промежуточного устройства (промежуточного устройства).

- Конструкция конвейера: Данные проходят по цепочке HBF как по конвейеру: чип 1 выполняет $A \times K_1$, чип 2 выполняет $B \times K_2$, а результаты накапливаются во время передачи.

Стратегия хранения и чтения данных

Равномерно распределяется по всей сети (аналогично DHT, распределенной хеш-таблице).

Запись (сохранение): Не заполняйте линейно (не заполняйте сначала чип 1, а затем чип 2).

- Шлюз разделяет и RLNC кодирует данные.

- Используйте многопутевую маршрутизацию для «распыления» закодированных пакетов фонтаном по всей сети.

- Передний чип действует как узел маршрутизации и сначала передает данные в глубокие узлы. После полной загрузки глубоких узлов мелкие узлы заполняются обратно.

Читать: Хост запрашивает данные -> Шлюз выдает запрос -> Все k узлов с соответствующими пакетами кодирования отвечают -> Агрегировать на шлюз для декодирования. Используя вычисления в памяти, можно выполнить частичное декодирование на пути передачи, чтобы уменьшить трафик по шине.

Логика реализации: Данные больше существуют не в виде «файлов», а в виде «решений

математических уравнений».

Процесс написания:

1. Хост записывает данные размером 4 КБ.

2. Шлюз разбивается на k символов и генерирует n векторов случайных коэффициентов (n > k).

3. Шлюз отправляет кортеж (фрагмент данных, коэффициент) в сеть NAND.

4. Узел NAND получает, использует вычисления в памяти для линейного объединения новых данных со старыми (если требуется агрегирование) и записывает их на страницу Flash.

Процесс чтения:

1. Шлюз собирает любые k комбинаций пакетов.

2. Исключение по Гауссу (на шлюзе или ускоренное с помощью FPGA) восстанавливает исходные 4 КБ.

Сборка мусора и выравнивание износа

Выравнивание износа: Естественное преимущество. RLNC разбивает данные, что само по себе является своего рода рандомизированной записью. Никакого сложного централизованного алгоритма WL не требуется. Узел просто проверяет свой собственный номер PE при записи, и если он слишком высок, новые записи отклоняются через протокол маршрутизации, заставляя данные передаваться на узлы с низким износом.

Сборка мусора: при удалении файла шлюз передает «Идентификатор фактора истечения срока действия». Узел использует вычисления в памяти в фоновом режиме для сканирования заголовка и пометки недействительных блоков. Никакой передачи данных не требуется, просто удалите их напрямую.

Предотвращение потери данных

Математическая гарантия: Если количество линейно независимых пакетов кодирования во всей сети ≥ количества исходных фрагментов данных, данные можно восстановить.

Физическая охрана:

- Контроль пульса: соседние узлы контролируют состояние выживания друг друга с помощью сигналов малой мощности.

- Автоматическая регенерация: как только узел в определенной области оказывается отключенным, окружающие узлы используют вычисления в памяти для создания новых пакетов кодирования на основе существующих данных для заполнения избыточности (без чтения исходных данных, непосредственное перекодирование закодированных данных).

Предотвращение глобальных задержек при сборке мусора

В децентрализованной архитектуре RLNC + вычисления в памяти (INC) сбор мусора (GC) больше не является тяжелой задачей «глобальной синхронизации», а превратился в локальный процесс самовосстановления, аналогичный метаболизму биологических систем.

Поскольку не существует централизованного мастера для ведения глобальной таблицы

физических сопоставлений, необходимо решить: когда узел стирает блок (Стереть), как предотвратить задержку всей Mesh-сети от сильных колебаний?

Основное нововведение: перекодирование в виде GC

В традиционном твердотельном накопителе сборщику мусора необходимо переместить действительные страницы в новые блоки, что приводит к значительному усилению внутреннего чтения и записи. В новой архитектуре, поскольку узел INC обладает вычислительными возможностями, мы можем реализовать GC посредством «перекодирования»:

1. Частичное сжатие: Когда недопустимые страницы в блоке достигают порогового значения, логика INC не просто перемещается, а считывает все действительные пакеты кодирования, принадлежащие различным GID в блоке.

2. Слияние в памяти: Линейное объединение этих пакетов с вновь прибывшим потоком записи в буфере.

3. Однократная обратная запись. Создайте совершенно новый пакет кодирования и запишите новые фрагменты.

- Преимущества: Процесс сборки мусора стал процессом «повышения избыточности данных», и «усиление» данных завершается при перемещении данных.

Три механизма предотвращения глобальных задержек

1. Механизм стабилизации кредитного лимита (регулирование GC на основе кредита)

Мы упомянули Credit (кредитный лимит) в конструкции протокола связи.

- Логика: Когда узел A необходимо физически стереть (это самая трудоемкая операция NAND, около 3–5 мс), он активно устанавливает внешне объявленный кредит на 0.

- Эффект: Соседние узлы будут автоматически обходить этот узел для маршрутизации или временно кэшировать пакет данных в своем собственном буфере. Поскольку это Mesh-сеть, путь является многомерным, и трафик будет автоматически обходить, как вода, встречающаяся с камнем, что позволяет избежать глобальных остановок ввода-вывода.

2. Стратегия «Drop & Reconstruct» (Drop & Reconstruct)

Это уникальный «доминирующий» подход решения RLNC:

- Традиционное решение: Каждая допустимая страница должна сохраняться во время сборки мусора, иначе данные будут потеряны.

- Решение RLNC: Если узел имеет чрезвычайно высокую нагрузку и ему необходимо срочно стереть блок, чтобы освободить место, он может напрямую «отбросить» некоторые действительные пакеты кодирования в блоке, а затем выдать RECODE_REQ (запрос на реконструкцию).

- Самовосстановление: после получения запроса шлюз или соседний узел использует функцию избыточности RLNC для генерации новых пакетов кодирования на других простаивающих узлах, чтобы восполнить недостающий ранг (Rank).

- Вывод: Замените пространство (избыточность) на время (нулевая задержка).

3. Асинхронное прогнозируемое стирание

Используйте логику вычислений в памяти для анализа шаблонов записи:

- Если логика INC предсказывает, что в следующие 100 мс будет большой объем трафика (на основе прогноза трафика внешнего шлюза), она упреждающе выполнит инструкцию Erase во время периода простоя.

- Поскольку 3D NAND имеет несколько плоскостей, можно реализовать независимую работу плоскостей: плоскость 0 выполняет стирание GC, а плоскость 1 по-прежнему нормально реагирует на чтение, не мешая друг другу.

Естественное выравнивание износа

В соответствии с концепцией «бесконечной производительности» выравнивание износа осуществляется автоматически:

- Пространство для жизни: поскольку емкость можно бесконечно расширять, данные распределяются в сети случайным образом.

- Автоматический сдвиг точки доступа: Если узел записывает слишком много данных и количество PE увеличивается, скорость его обработки замедлится за несколько микросекунд. Протокол маршрутизации RLNC автоматически определит это увеличение задержки, тем самым уменьшая вес, отправляемый узлу.

- Алгоритм без алгоритмов: Это динамический баланс, основанный на физической обратной связи, который более эффективен, чем любой сложный программный алгоритм WL.

3. Общие принципы проектирования

Отображение пакетов на страницы NAND

Физически сопоставьте пакеты со структурами страниц NAND.

Традиционная структура страницы SSD обычно следующая: пользовательские данные (4 КБ/8 КБ/16 КБ) + OOB/резервные (несколько сотен байт).

Прорывной дизайн:

- OOB/Резервная область → Заголовок маршрутизации и кодирования.

- Причина: Вычислительные схемы в памяти часто могут быстро считывать области OOB без необходимости активировать передачу данных всей страницы. Таким образом, узел может сначала прочитать заголовок, чтобы определить, требуется ли вычисление или пересылка, что значительно снижает энергопотребление и задержку.

- Область пользовательских данных → Закодированная полезная нагрузка.

- Причина: Сохраняет объекты данных после линейной комбинации.

Подробная структура пакета

Предположим, мы определяем логическую атомарную единицу (атомарную единицу), соответствующую физической странице размером 4 КБ (плюс 128 байт OOB в качестве примера).

Контейнер физического уровня

Этот уровень в основном занимается коррекцией ошибок мультимедиа, которая автоматически выполняется базовым оборудованием.

Регион	Имя поля	Длина	Описание
Хвост	LDPC Паритет	Переменная длина	Код физического исправления ошибок (защита от переворота битов) от ошибок носителя NAND. Базовая схема автоматически удаляется, а верхняя логика становится невидимой.

A. Заголовок маршрутизации и управления

Функция: Отвечает за управление потоком данных между чипами и каналами. Расположен перед зоной OOB.

Смещение	Имя поля (Поле)	Разрядность (биты)	Подробное определение и цель
0x00	Тип пакета	4	Идентифицирует тип пакета: 0001: исходный пакет данных (систематический), 0010: кодированный пакет данных (кодированный), 0100: команда управления (управление/GC), 1000:

Смещение	Имя поля (Поле)	Разрядность (биты)	Подробное определение и цель
			запрос повторной передачи (запрос)
0x04	TTL (Время жить)	4	Предел прыжков. Уменьшайте на 1 для каждого переданного узла NAND. Защитите от циклических штормов.
0x08	Идентификатор целевого узла	16/32	Идентификатор целевой логической области. Если это режим широковещания (полное сетевое хранилище), это конкретная маска.
0x20	Код операции INC	8	Инструкция расчета в памяти. Сообщите NAND, что делать с этим пакетом: 0x01: только пересылка (только маршрутизация), 0x02: XOR и сохранение, 0x03: умножение и накопление (MAC и вперед)
0x28	Контрольная сумма MRD	32	Контрольная сумма кода максимального рангового расстояния. Для быстрого обнаружения загрязнения линейных подпространств (обнаружение ошибок без декодирования).

B. Заголовок кодирования RLNC

Функция: Описывает математические свойства данных, используемые для декодирования и перекодирования. Расположен в задней части OOB.

Смещение	Имя поля (Поле)	Разрядность (биты)	Подробное определение и цель
0x00	Идентификатор поколения (GID)	64	Идентификатор поколения/пакета. Все фрагменты, принадлежащие одному и тому же исходному файлу (или сегменту LBA), имеют один и тот же идентификатор. Это необходимое условие для комбинированных расчетов.
0x40	Индекс символов	16	Если это исходный пакет данных, он указывает, какой это фрагмент. Если это полностью закодированный пакет, этот элемент можно игнорировать или установить все 1.

Смещение	Имя поля (Поле)	Разрядность (биты)	Подробное определение и цель
0x50	Кодирование вектора / начального числа	128~256	Базовое поле: вектор коэффициентов кодирования. Решение по оптимизации: для экономии пространства OOB полный вектор коэффициентов не сохраняется напрямую, а сохраняется начальное значение PRNG (начальное число случайного числа) и маска. Приемник NAND использует логику INC для локальной генерации вектора коэффициентов на основе начального числа.
0xПеременная	Текущий рейтинг	8	Количество независимых уравнений, содержащихся в текущей линейной комбинации (используется для определения того, достаточно ли собрано данных для восстановления исходной информации).

C. Поле полезной нагрузки

Функция: Перенос информационных объектов. Расположен в области «Основные данные».

Имя поля	Размер	Описание
Символы закодированных данных	4 КБ (типично)	Это вектор $P = \Sigma(c_i \cdot D_i)$. Здесь хранится не исходный текст файла, а результат исходного текста после арифметических операций в поле GF(2^q).
Прокладка	Переменная длина	Если исходные данные меньше 4 КБ, для выравнивания страницы NAND используется заполнение нулями, чтобы облегчить выравнивание матричных операций.

Основные технические оптимизации

Чтобы максимизировать производительность в сценариях «неограниченной емкости» и «вычислений в памяти», рекомендуется принять следующие две стратегии оптимизации:

Технология сжатия семян

В RLNC по мере увеличения размера генерации k вектор коэффициентов становится очень большим (например, k=128, с 1 байтом на коэффициент, вектор требует 128 байт, занимая OOB).

- Как это сделать: Не передавайте c_1, c_2, ..., c_k в Заголовке.

- Улучшение: передавайте только одно случайное начальное число (32 бита).

- Расчет: Схема INC узла NAND имеет встроенный фиксированный генератор псевдослучайных чисел (например, LFSR).

- Коэффициенты = LFSR(Seed)

- Преимущества: Размер заголовка значительно уменьшен, оставляя больше места для управления маршрутизацией и проверки MRD.

Разреженное сетевое кодирование (разреженное RLNC)

Чтобы снизить энергопотребление вычислений NAND в памяти, не допускайте, чтобы каждый пакет состоял из всех k исходных пакетов.

- Дизайн: Ограничьте плотность векторов кодирования.

- Реализация заголовка: Добавьте поле «Карта плотности» в заголовок.

- Преимущества: сокращает количество операций умножения чисел с плавающей запятой и целых чисел во время вычислений в памяти и повышает производительность.

Пример: прохождение пакета между устройствами NAND

Предполагая, что NAND-A получает пакет данных, процесс выглядит следующим образом:

1. Анализ заголовка (1–3 такта):

- Логика интерфейса NAND-A считывает только область OOB.

- Проверьте идентификатор целевого узла → обнаружено, что он отправляется в эту область.

- Проверьте TTL → TTL > 0, допустимо.

- Проверьте контрольную сумму MRD → пакет не загрязнен.

Оценка в памяти (логика INC):

- Прочтите идентификатор поколения (GID).

- Локальный массив хранения INC с логическим поиском: «Есть ли у меня другие фрагменты данных для этого GID?»

- Случай 1 (Хранилище): Если локальная область пуста, запишите полезную нагрузку непосредственно во флэш-накопитель.

- Случай 2 (агрегирование вычислений): Если данные для GID имеются локально, запустите сложение матрицы в памяти:

- New_Payload = Payload_in ⊕ Payload_stored

- New_Vector = Vector_in ⊕ Vector_stored

- Случай 3 (пересылка): Если локальная область заполнена, измените следующий переход в заголовке и передайте его в NAND-B с помощью шины.

Выполнить действие:

- Весь процесс не требует вмешательства главного процессора управления и выполняется в граничной логике NAND и массиве ячеек.

Для этого продукта дизайн заголовка определяет верхний предел архитектуры.

- Предложение 1. Используйте схему «Заголовок фиксированной длины + вектор

переменной длины», чтобы использовать пространство OOB.

- Рекомендация 2: INC OpCode — это основная инновация, которая превращает NAND из памяти в исполнитель инструкций. Вам необходимо определить упрощенный набор команд (ISA), встроенный в заголовок.

«iNc OpCode (набор инструкций для вычислений в памяти)» относится к конкретной логической конструкции аппаратной схемы. Это очень критическая ссылка. Преобразование NAND Flash из «пассивного хранилища» в «активный вычислительный узел» требует оптимизированного, эффективного и специального набора инструкций для операций RLNC. Этот набор инструкций не обязательно должен быть таким же общим, как x86 или ARM. Это больше похоже на ASIP (процессор с набором инструкций для конкретного приложения), оптимизированный для линейной алгебры и потока данных.

4. Набор инструкций INC OpCode

Обзор набора инструкций INC OpCode

Код операции разделен на три основные категории: Поток, Вычисления и Обслуживание.

Рекомендуемая разрядность: 8 бит (поддерживает 256 инструкций, в настоящее время определяет около 10 основных инструкций, оставляя много места для расширения)

Инструкции управления потоком

Используется для транспортировки и маршрутизации данных без изменения содержимого данных.

Код операции (шестнадцатеричный)	Мнемоника	Полное название инструкции	Логика поведения оборудования
0x00	НОП	Нет операции	Никакой операции. Используется для выравнивания часов или ожидания арбитража шины.
0x01	ПРОЙДИТЕ	Пройти	Переадресация маршрута. Без чтения массива Flash данные из входного порта напрямую передаются в выходной порт (следующий переход). Срабатывает, когда TTL > 0 и локальное хранилище заполнено.
0x02	МАГАЗИН	Прямой магазин	Прямое хранение. Запишите полезную нагрузку непосредственно на Flash-страницу без вычислений. Используется для хранения системных кодов (копий исходных данных).
0x03	БЦСТ	Трансляция	Транслировать. Копирует и отправляет пакет данных всем соседним узлам, что часто используется для распределения запросов на глобальную настройку или срочного исправления ошибок.

Инструкции вычислений и RLNC

Это ядро вычислений в памяти, выполняющее математические операции в области GF(2^q).

Код операции (шестнадцатеричный)	Мнемоника	Полное название инструкции	Логика поведения оборудования	Математическое выражение
0x10	XOR_WR	XOR и запись	Считайте целевую страницу, выполните XOR с входными данными и запишите результат обратно. Это самая основная операция кодирования RLNC.	D_new = D_in ⊕ D_old

Код операции (шестнадцатеричный)	Мнемоника	Полное название инструкции	Логика поведения оборудования	Математическое выражение
0x11	MAC_WR	Умножение-накопление и запись	Основные инструкции. Входные данные умножаются на коэффициент α, сохраненные данные умножаются на коэффициент β, и результат записывается обратно.	D_new = α · D_in ⊕ β · D_stored
0x12	ПЕРЕКОДИРОВАТЬ	Восстановить код	Перегенерировать код. Не зависит от внешних данных. Прочитайте локальные k страниц, сгенерируйте новый пакет случайной линейной комбинации и передайте его. Используется для восстановления потерянных данных в сети.	P_out = Σ r_i · P_stored_i
0x13	ЧК_МРД	Проверьте синдром MRD	Проверка целостности. Рассчитайте синдром данных, чтобы определить, есть ли потеря ранга или переворот битов. Если проверка не удалась, автоматически запускается RECODE.	S знак равно ЧАС · Д^Т

Инструкции обслуживания

Используется для сбора мусора и управления состоянием.

Код операции (шестнадцатеричный)	Мнемоника	Полное название инструкции	Логика поведения оборудования
0x20	ИНВ_ГИД	Недействительный GID	Пометить данные для определенного идентификатора поколения как «недействительные/мусорные». Физический блок не стирается, отмечается только область метаданных, ожидающая фонового GC.
0x21	РПТ_СТАТ	Статус отчета	Сообщайте о своем собственном статусе (счетчик износа PE, оставшаяся емкость, текущая нагрузка) шлюзу или соседу.

Кодирование инструкций

Для быстрого анализа логики INC NAND формат инструкций должен быть фиксированным и оптимизированным.

Рекомендуемый формат (1-байтовый код операции + необязательные операнды):

битовое поле	имя	описание
[7:5]	Категория	Категория инструкции: 000=Поток, 001=Вычисление, 010=Обслуживание
[4:0]	Подоперационная	Подкод операции (32 исходные инструкции)

Источник операндов:

- Неявные операнды. Большинство операндов берутся из других полей заголовка (например, GID, вектор кодирования), и их не нужно добавлять после кода операции.

- Явный операнд: Только если необходимо указать конкретный адрес страницы (например, RECODE с указанием исходной страницы), адрес размером 2–4 байта добавляется к коду операции.

Рекомендации по аппаратной реализации

1. Микро АЛУ (арифметико-логическое устройство):

Реализуйте ALU, оптимизированный для GF(2^8) на логическом кристалле.

- Базовый модуль: 8-битный массив XOR + множитель справочной таблицы (множитель на основе LUT).

- Площадь: Очень маленькая, около 0,01 мм² (процесс 40 нм).

2. Декодер инструкций:

Простая комбинационная логическая схема считывает байты кода операции и выводит управляющие сигналы на интерфейсы ALU и Flash.

3. Государственный автомат (FSM):

Поскольку инструкции выполняются потоковым способом (прибытие пакета → декодирование → выполнение → эмиссия), сложный конвейер не требуется. Достаточно 3-5 состояний FSM: IDLE → FETCH_HEADER → DECODE → EXECUTE → EMIT.

5. Проектирование аппаратных схем

Обзор проектирования аппаратных схем

Чтобы реализовать концепцию «памяти бесконечной емкости», необходимо разработать новую аппаратную архитектуру. Суть этой архитектуры заключается в преобразовании традиционной модели «главный + пассивный NAND» в «ячеистую сеть, состоящую из интеллектуальных узлов NAND».

Архитектура узла Smart NAND

Каждый узел Smart NAND состоит из двух частей:

1. Storage Die (уровень хранения):

- Стандартный флэш-массив 3D NAND

- Отвечает за физическое хранение данных.

- Обеспечить возможности исправления ошибок LDPC

2. Логический кубик:

- Склеивание с помощью Storage Die посредством 3D-упаковки

- Содержит вычислительный блок INC, логику маршрутизации, SerDes PHY.

- Отвечает за расчет, пересылку и обработку протоколов данных.

Метод подключения:

- Использование технологии TSV (Through-Silicon Via) или гибридного соединения.

- Пропускная способность между логическим кристаллом и кристаллом хранения может достигать уровня ТБ/с.

Основные модули логического кристалла

А. Вычислительная система INC

Модули	Функции	Оценка площади
ГФ(2^8) АЛУ	Умножение и сложение конечных полей	0,01 мм²
XOR-массив	Пакетная операция XOR	0,005 мм²
ЛФСР	Генерация псевдослучайных чисел	0,001 мм²
Декодер инструкций	Анализ кода операции	0,002 мм²

Б. Механизм маршрутизации

Модули	Функции	Оценка площади
Парсер заголовков	Разбираем территорию ООО	0,003 мм²
Таблица маршрутизации	Таблица адресов соседних узлов	0,01 мм²
Кредитный менеджер	Управление потоками	0,005 мм²

Модули	Функции	Оценка площади
ТТЛ-процессор	Управление количеством переходов	0,001 мм²

С. Физический интерфейс

Модули	Функции	Оценка площади
СерДес PHY	Высокоскоростной последовательный интерфейс	0,1 мм²
ФАПЧ	Генерация часов	0,02 мм²
ЦДР	Восстановление часов	0,03 мм²

Топология сети Mesh

Выбор топологии: 2D-тор

- Каждый узел подключен к 4 соседям (сверху, снизу, слева, справа)

- Граничные узлы соединены друг с другом, образуя тор.

- Преимущества: резервирование путей, балансировка нагрузки, отсутствие единой точки отказа.

Пример физического макета (массив 4x4):

[N00]--[N01]--[N02]--[N03]--[N00]

| | | | |

[N10]--[N11]--[N12]--[N13]--[N10]

| | | | |

[N20]--[N21]--[N22]--[N23]--[N20]

| | | | |

[N30]--[N31]--[N32]--[N33]--[N30]

| | | | |

[N00]--[N01]--[N02]--[N03]--[N00]

Метод расширения:

- Горизонтальное расширение: увеличьте количество столбцов.

- Вертикальное расширение: увеличьте количество строк.

- 3D-расширение: складывайте несколько слоев 2D-тора.

Проектирование шлюза

Шлюз — это интерфейс между всей системой и внешним миром, отвечающий за преобразование протоколов и ввод/извлечение данных.

Основные функции:

1. Обработка протокола NVMe

- Анализ команд NVMe, отправленных с хоста

- Преобразование логического адреса в идентификатор поколения.

2. Кодер RLNC

- Разделить пользовательские данные на k символов

- Сгенерировать n пакетов кодирования (n > k)

- Назначить вектор случайных коэффициентов

3. Декодер RLNC

- Собрать k линейно независимых пакетов кодирования.

- Метод исключения Гаусса восстанавливает исходные данные

4. Планировщик трафика

- Распределение потока данных в зависимости от нагрузки сети.

- Реализуйте балансировку нагрузки.

Аппаратная реализация:

- Для проверки прототипа рекомендуется использовать FPGA.

- Версия массового производства может быть интегрирована как ASIC

6. Протокол физического уровня

Обзор протокола физического уровня

Для достижения эффективной связи между узлами Smart NAND необходимо разработать специальный протокол физического уровня. Данное соглашение должно отвечать следующим требованиям:

- Высокая пропускная способность: поддерживает скорость передачи данных более 16 Гбит/с.

- Низкая задержка: Сквозная задержка < 100 нс.

- Надежность: Частота битовых ошибок < 10^-15

- Расширяемость: поддерживает динамическое увеличение и уменьшение узлов.

Формат кадра

Физическая структура фрейма:

Поле	Длина	Описание
Преамбула	8 байт	Последовательность синхронизации для восстановления тактовой частоты
ЮФО	1 байт	Разделитель начала кадра
Заголовок	16 байт	Маршрутизация и кодирование заголовка
Полезная нагрузка	4 КБ	Полезная нагрузка данных
КПР	4 байта	Последовательность проверки кадров
ЭФД	1 байт	Разделитель конца кадра

Эффективность трансмиссии: Коэффициент полезной нагрузки > 99 %.

Механизм управления потоком

Внедрить механизм управления потоками на основе кредитов:

1. Фаза инициализации:

- Каждый узел объявляет свою глубину буфера своим соседям.

- Соседи записывают начальный кредит узла.

2. Передача данных:

- Отправка пакета данных требует 1 кредита.

- После того, как получатель завершит обработку, вернитесь к обновлению кредита.

3. Обработка перегрузок:

- Когда Credit = 0, приостановить отправку этому соседу

- Пакеты автоматически маршрутизируются другим соседям

Преимущества: отсутствие потери пакетов, отсутствие повторной передачи, предсказуемая задержка.

Протокол установления соединения

Процесс присоединения узла:

1. Фаза обнаружения: новый узел передает сообщение HELLO.

2. Фаза аутентификации: Соседние узлы проверяют личность нового узла.

3. Этап настройки: назначьте идентификатор узла и синхронизируйте таблицу маршрутизации.

4. Этап готовности: новые узлы начинают участвовать в хранении и пересылке данных.

Процесс выхода узла:

1. Фаза уведомления: Узел передает сообщение GOODBYE.

2. Фаза миграции: Соседние узлы запускают перекодирование данных.

3. Этап очистки: обновите таблицу маршрутизации и удалите узел.

7. Динамическое расширение

Обзор динамического расширения

Суть «неограниченной емкости» заключается в том, что система может динамически добавлять узлы хранения во время работы без простоев или миграции данных. Это фундаментальное преимущество архитектуры RLNC перед традиционным RAID.

Процедура горячего добавления узла

1. Физическое подключение:

- Новые узлы подключаются к Mesh-сети.

- Автоматически обнаруживать соседние узлы

2. Регистрация мощности:

- Новые узлы сообщают о своей мощности шлюзу.

- Шлюз обновляет глобальное представление емкости.

3. Балансировка данных:

- Система автоматически переносит некоторые пакеты кодирования на новые узлы.

- Используя функцию RLNC, вам нужно только создать новый пакет кодирования без перемещения исходных данных.

4. Завершение расширения:

- Новый узел начинает нормально участвовать в чтении и записи

- Производительность системы увеличивается линейно

Основные преимущества: Весь процесс прозрачен для приложений верхнего уровня и не требует простоев.

Процедура горячего удаления узла

1. Отметка о выходе на пенсию:

- Администратор помечает узел как подлежащий удалению.

- Система прекращает запись новых данных на узел

2. Перекодирование данных:

- Соседний узел читает закодированный пакет узла

- Генерировать новые пакеты кодирования и хранить их на других узлах.

- Обеспечить, чтобы избыточность не уменьшалась

3. Безопасное удаление:

- Подтвердите, что данные были полностью перекодированы.

- Физически отключить узлы

Отказоустойчивость: даже если узел внезапно выходит из строя (без тега), система может автоматически восстановиться.

Расчёт ёмкости

Эффективная емкость = Общая физическая емкость × (k/n)

Среди них:

- k = количество исходных фрагментов данных

- n = общее количество закодированных пакетов (n > k)

- (n-k)/n = накладные расходы на избыточность

Пример:

- Физическая емкость: 100 ТБ (100 узлов по 1 ТБ)

- k=8, n=12 (1,5-кратное резервирование)

- Эффективная емкость = 100 ТБ × (8/12) = 66,7 ТБ.

После расширения:

- Физическая емкость: 150 ТБ (добавлено 50 новых узлов)

- Эффективная емкость = 150 ТБ × (8/12) = 100 ТБ.

- Емкость растет линейно без реорганизации данных

8. Применение кодов MRD

Обзор кодов MRD

Код MRD (максимальное ранговое расстояние) — это код исправления ошибок, специально разработанный для систем сетевого кодирования. В отличие от традиционных кодов RS, коды MRD могут эффективно обнаруживать и исправлять ошибки типа «недостаточный ранг».

Сравнение кодов MRD и Рида — Соломона

Особенности	Код PC	Код MRD
Модель ошибки	Ошибка входа	Потеря ранга
Применимые сценарии	Традиционное хранение	Сетевое кодирование
Возможности обнаружения	Поиск мест ошибок	Обнаружение загрязнения подпространства
Возможность исправления ошибок	Правильные символы t	Исправление t-мерной потери ранга
Вычислительная сложность	$O(n^2)$	$O(n^2)$

Основные преимущества: Коду MRD не нужно знать, какой узел неисправен, ему нужно лишь определить, достаточен ли общий ранг.

MRD в неограниченно расширяемом хранилище

1. Проверка целостности данных:

- Каждый закодированный пакет содержит контрольную сумму MRD.

- Узлы могут быстро обнаружить, загрязнены ли данные.

- Определить достоверность данных без декодирования

2. Обнаружение вредоносного узла:

- Если узел возвращает неверные данные

- Проверка MRD немедленно обнаружит отклонения

- Система автоматически изолирует узел

3. Эффективное исправление ошибок:

- При обнаружении потери ранга

- Просто получите дополнительные пакеты кодирования с других узлов.

- Нет необходимости повторно передавать все данные

4. В сотрудничестве с ЛДПК:

- LDPC обрабатывает переключение физического уровня.

- MRD обрабатывает потерю ранга на системном уровне.

-Двойная защита, данные никогда не будут потеряны

Аппаратная реализация MRD

Расчётная единица:

- Матричные операции на основе GF(2^8)

- Делитесь ресурсами ALU с RLNC.

Процесс проверки:

1. Считайте вектор коэффициентов пакета кодирования.

2. Вычислить сопряженную формулу S = H × V^T

3. Если S ≠ 0, запустить процесс исправления ошибок.

Накладные расходы:

- Дополнительная логика < 0,01 мм²

- Многоразовый вычислительный блок RLNC

9. Состав модулей INC-Core

Обзор модулей INC-Core

INC-Core — это основной логический блок узла Smart NAND, объединяющий все необходимые функции вычислений, маршрутизации и управления. Ниже приведен полный список модулей.

Вычислительные модули

Имя модуля	Описание функции	Интерфейс	Площадь
ГФ_АЛУ	GF(2^8) Арифметико-логический блок	8-битный вход/выход	0,01 мм²
XOR_ARRAY	Пакетное исключающее ИЛИ 4 КБ	4 КБ на входе × 2, 4 КБ на выходе	0,005 мм²
ЛФСР_GEN	Генерация псевдослучайных коэффициентов	32-битное начальное значение, 8-битное выходное	0,001 мм²
МАТРИКС_MUL	Ускоритель умножения матриц	Интерфейс конфигурации + поток данных	0,02 мм²
ГАУСС_ЕЛИМ	Блок исключения Гаусса	Ввод матрицы, вывод вектора решения	0,03 мм²

Модули маршрутизации

Имя модуля	Описание функции	Интерфейс	Площадь
HDR_PARSER	Парсер заголовков	Вход OOB, выход управляющего сигнала	0,003 мм²
МАРШРУТ_ТАБЛИЦА	Хранение таблицы маршрутизации	Интерфейс запроса адреса	0,01 мм²
КРЕДИТ_МГР	Управление потоками кредитного управления	Интерфейс статуса соседа	0,005 мм²
ТТЛ_ПРОК	ТТЛ-процессор	Интерфейс счетчика	0,001 мм²
ПКТ_ПЕРЕКЛЮЧАТЕЛЬ	Коммутатор пакетов	4-портовый ввод-вывод	0,015 мм²

Интерфейсные модули

Имя модуля	Описание функции	Технические характеристики	Площадь
СЕРДЕС_PHY	Высокоскоростной последовательный трансивер	16 Гбит/с × 4 полосы	0,1 мм²
ФАПЧ	Фазовая автоподстройка частоты	100 МГц – 8 ГГц	0,02 мм²
ЦДР	Восстановление данных часов	Адаптивное выравнивание	0,03 мм²
FLASH_IF	Интерфейс флэш-массива	Совместимость с ONFI 4.0	0,01 мм²

Модули управления

Имя модуля	Описание функции	Интерфейс	Площадь
ФСМ_CTRL	Главная государственная машина	Глобальный сигнал управления	0,005 мм²
INSTR_DEC	Декодер инструкций	Ввод кода операции, выход управления	0,002 мм²
INT_CTRL	Контроллер прерываний	Интерфейс обработки исключений	0,002 мм²
PWR_MGR	Управление питанием	Контроль сна/пробуждения	0,003 мм²

Оценочная общая площадь: < 0,3 мм² (процесс 40 нм)

Оценка потребляемой мощности: < 50 мВт (типичное рабочее состояние)

10. Применения и преимущества

Обзор областей применения

Система хранения неограниченной емкости на основе RLNC + INC имеет революционную прикладную ценность во многих областях.

Центры обработки данных ИИ

Сценарий применения:

- Крупномасштабное хранилище данных для обучения модели искусственного интеллекта.

- Хранение параметров модели для службы вывода

- Мультимодальное озеро данных

Основные преимущества:

- Неограниченная масштабируемость: По мере роста объема данных просто добавляйте узлы.

- Высокая пропускная способность: параллельное чтение RLNC для удовлетворения требований к пропускной способности графического процессора.

- Высокая надежность: Двойная коррекция ошибок: данные обучения никогда не будут потеряны.

Показатели эффективности:

- Последовательное чтение: 100+ ГБ/с (кластер из 1000 узлов)

- Случайные IOPS: 10M+ (благодаря распределенной архитектуре)

Периферийные вычисления

Сценарий применения:

- Автономная запись данных вождения

- Сбор данных промышленного Интернета вещей

- Пограничный кэш 5G

Основные преимущества:

- Эластичное развертывание: Динамически увеличивайте или уменьшайте количество узлов в зависимости от спроса.

– Локальные вычисления. Вычисления в памяти сокращают объем обработки данных.

- Высокая надежность: отказ узла не влияет на качество обслуживания в целом.

Типичная конфигурация:

- Установлен на транспортном средстве: 16 узлов, эффективная емкость 16 ТБ.

- Завод: 64 узла, эффективная емкость 64 ТБ.

Архивирование редко используемых данных

Сценарий применения:

- Архивирование корпоративных данных в соответствии с требованиями

- Долговременное хранение медицинских изображений

- Постоянное хранение данных научных исследований

Основные преимущества:

- Сверхдолгий срок службы: RLNC автоматически устраняет старение носителя

- Низкая стоимость: использование QLC/PLC NAND, высокая плотность емкости.

- Нулевое обслуживание: Система восстанавливает себя без ручного вмешательства.

Экономика:

- Стоимость хранения: <0,01 доллара США/ГБ/год.

- Срок хранения данных: более 100 лет (теоретическое значение)

Конкурентные преимущества

Размеры	Традиционный SSD	RAID-массив	Неограниченная емкость хранилища
Масштабируемость	Фиксированная мощность	Ограниченное расширение	Неограниченное расширение
Надежность	Единая точка отказа	Восстановление требует много времени	Автоматическое самовосстановление
Производительность	Ограничено мастером	Ограничено контроллером	Линейное расширение
Стоимость	Высокий (требуется главный контроль)	Высокий (требуется карта RAID)	Низкий (децентрализованный)
Техническое обслуживание	Требует замены	Требуется восстановление	Нулевое обслуживание

Основной вывод: Система хранения данных неограниченной емкости – это новый сдвиг в парадигме технологий хранения данных.

11. Тестовые сценарии прототипа

Обзор тестовых сценариев

Чтобы проверить осуществимость системы хранения неограниченной емкости, необходимо выполнить следующие функциональные тесты на прототипе FPGA. Тест разделен на четыре этапа: базовые функции, кодирование и декодирование RLNC, Mesh-сеть и системная интеграция.

Этап 1: базовые функции

Номер теста	Название теста	Цель теста	Критерии прохождения
T1.1	Flash чтение и запись	Проверьте корректность интерфейса NAND	Запись данных = Чтение данных
T1.2	СерДес шлейф	Проверьте высокоскоростной интерфейс	Коэффициент битовых ошибок < 10^-12
T1.3	ГФ(2^8) АЛУ	Проверка конечных полевых операций	Все результаты операций верны
T1.4	поколение LFSR	Проверьте качество случайных чисел	Пройти тест NIST

Этап 2: кодирование и декодирование RLNC

Номер теста	Название теста	Цель теста	Критерии прохождения
T2.1	Однопоколенческое кодирование	Проверка правильности кодирования RLNC	k пакетов могут восстановить исходные данные
T2.2	Перекодирование	Проверьте восстановленное кодирование	Новые пакеты кодирования линейно независимы
T2.3	Метод исключения Гаусса	Проверьте правильность декодирования	Декодированные данные = исходные данные
T2.4	Проверка MRD	Обнаружение ошибок валидации	100% обнаружение ошибок впрыска

Этап 3: сеть Mesh

Номер теста	Название теста	Цель теста	Критерии прохождения
T3.1	Обнаружение узлов	Проверьте протокол рукопожатия	Новый узел <1 с для присоединения
T3.2	Переадресация маршрутов	Проверка пересылки пакетов	TTL уменьшился правильно
T3.3	Механизм управления потоком	Проверка Кредитный менеджмент	Нет потери пакетов, нет взаимоблокировок
T3.4	Восстановление неисправностей	Обработка сбоев узла проверки	Автоматическое перекодирование данных

Этап 4: системная интеграция

Номер теста	Название теста	Цель теста	Критерии прохождения
T4.1	NVMe-интерфейс	Проверьте совместимость хоста	Идентифицируется как стандартное устройство NVMe
T4.2	Последовательная запись	Проверка производительности записи	> 1 ГБ/с (4 узла)
T4.3	Последовательное чтение	Проверенная производительность чтения	> 2 ГБ/с (4 узла)
T4.4	Случайное чтение и запись	Проверка IOPS	> 100 тыс. операций ввода-вывода в секунду
T4.5	Горячее расширение	Проверка динамического расширения	Узел успешно добавлен онлайн
T4.6	Сохранение данных	Проверка защиты от отключения электроэнергии	Целостность данных после отключения электроэнергии

Этап: после завершения всех испытаний вы можете перейти к этапу подготовки к серийному производству.