Санкт-Петербургский государственный университет

Жереб Вера Вадимовна

Выпускная квалификационная работа

Исследование и внедрение RAM-кэша для RAID-массива, оптимизированного под Flash-накопители

Уровень образования: бакалавриат

Направление *02.03.03 "Математическое обеспечение и администрирование информационных систем"*

Основная образовательная программа *СВ.5006.2017 "Математическое обеспечение и администрирование информационных систем"*

Профиль *Системное программирование*

Научный руководитель:
зав. каф. системного программирования
д-р физ.-мат. наук А. Н. Терехов

Консультант:
Технический директор ООО "Рэйдикс"
канд. техн. наук С. В. Лазарева

Рецензент:
Разработчик исследовательской лаборатории ООО "Рэйдикс"
А. И. Васенина

Санкт-Петербург
2021

Saint Petersburg State University

Zhereb Vera

Bachelor's Thesis

Investigation and implementation of RAM-cache for RAID optimised for Flashdrives

Education level: bachelor

Speciality *02.03.03 "Software and Administration of Information Systems"*

Programme *CB.5006.2017 "Software and Administration of Information Systems"*

Educational profile *System Engineering*

Scientific supervisor:
chair head, D.Sc, prof. Andrey Terekhov

Scientific consultant:
Head of RAIDIX R&D department, Ph. D. Svetlana Lazareva

Reviewer:
RAIDIX R&D Software Engineer Vasenina Anna

Saint Petersburg
2021

Оглавление

Введение

Объёмы хранимой человеком информации многократно увеличиваются с каждым годом, в связи с этим возрастают затраты на хранение. Поэтому организации стремятся получить продукт, обеспечивающий хорошую производительность всех компонентов электронной системы, а также гарантирующий защиту и целостность данных. Для безопасного хранения данных и предоставления гарантированного доступа к ним используются системы хранения данных (СХД). Одним из основных подходов организации СХД является технология RAID (Redundant Array of Independent/Inexpensive Disks) — массив из нескольких дисков, управляемый контроллером, «отказоустойчивый массив из независимых дисков» [4].

Отказоустойчивость СХД обеспечивается с помощью информационной избыточности: в системе используются дополнительные диски (называемые синдромами), информация которых позволяет восстанавливать данные в случае их частичной утраты.

Существует множество модификаций RAID, но наиболее широкое распространение получили перечисленные ниже.

- RAID 0: данные делятся на столько частей, сколько дисков в массиве, и равномерно распределяются по дискам. Используется для ускорения обработки запросов, отказоустойчивость не предусмотрена: выход из строя любого из дисков, входящего в массив, приводит к невосстановимой потере хранимых данных.

- RAID 1 (зеркалирование/дублирование): массив из двух или более дисков, являющихся полными копиями друг друга. Такой вариант хранения данных надёжен, но влечёт большую избыточность — 50%.

- RAID 5: допускает параллельную запись, поскольку блоки данных и контрольные суммы циклически записываются на все диски массива. Модификация позволяет восстановить до одного утраченного диска, если его номер заранее известен.

- RAID 6: этот способ также позволяет параллельную запись, однако избыточность в данной модификации на один диск больше, чем в RAID 5, в связи с этим возможно восстановление одного или двух утраченных дисков.

СХД может взаимодействовать с разными устройствами хранения данных (УХД): как с жесткими дисками — HDD (в которых присутствуют механические детали — магнитные диски, из-за чего при поиске данных в них возникают задержки вследствие вращения дисков), так и с твердотельными накопителями (SSD/NVMe). Основными критериями при выборе УХД являются объем, надежность хранения данных, производительность и стоимость. На многие из этих критериев существенное влияние оказывает интерфейс — протокол взаимодействия накопителя и вычислительных ресурсов системы [5].

Класс	HDD			SSD		
Интерфейс	SCSI	SATA	SAS	SATA	SAS	PCIe
Накопитель	SCSI	SATA	SAS	SATA	SAS	NVMe
Надежность	Средняя	Низкая	Высокая	Средняя	Высокая	Высокая
Производительность	Низкая	Низкая	Средняя	Высокая	Высокая	Очень высокая
Стоимость	Низкая	Низкая	Средняя	Средняя	Высокая	Очень высокая

Рис. 1: Сравнение характеристик HDD и SSD накопителей

В результате расширения рынка твердотельных накопителей и появления новых УХД возникла необходимость в создании технологий, позволяющих оптимально решать высокопроизводительные задачи. В 2018 году российской компанией RAIDIX была представлена RAIDIX ERA — программный RAID, который, благодаря продуманной внутренней архитектуре и параллелизации вычислений, эффективно работает с SSD и NVMe накопителями [6]. Однако при определенных сценариях использования наблюдается ухудшение скорости работы RAIDIX ERA по сравнению с известными продуктами.

Одним из основных способов увеличения производительности без особых вложений в модернизацию оборудования является добавление к RAID-контроллеру относительно небольшой, но обычно более быстрой области памяти, называемой RAID-кэшем, использующейся для про-

межуточного хранения записываемых или считываемых данных. Это позволяет эффективнее управлять операциями ввода-вывода.

Зачастую современные компьютеры оснащены большим объемом оперативной памяти (RAM), что позволяет использовать в качестве кэша RAM-диск — технология, реализующая хранение данных в оперативной памяти как на блочном устройстве. Кроме того, использование RAM-диска может продлить срок службы твердотельных накопителей, поскольку SSD имеют ограниченное количество циклов записи.

В рамках данной работы планируется изучение различных технологий кэширования, сравнение, выявление и внедрение наиболее подходящей реализации для системы RAIDIX ERA.

1. Цели и задачи

1.1. Цель работы

Целью работы является повышение производительности системы RAIDIX ERA посредством внедрения RAM-кэширования.

1.2. Поставленные задачи

Для достижения обозначенной цели необходимо выполнить задачи, перечисленные ниже.

- Изучение различных технологий кэширования и выявление наиболее подходящей.
- Внедрение технологии, ее улучшение и адаптация под RAID компании RAIDIX.
- Выполнение тестирования внедренной технологии.

2. Обзор

Обзор включает в себя ознакомление с продуктом RAIDIX ERA, рассмотрение теории кэширования, описание основных особенностей RAID-кэша, а также приводятся примеры других решений компании RAIDIX, связанных с кэшированием. В последней секции обзора рассмотрены основные инструменты и подходы к тестированию производительности систем хранения данных.

2.1. RAIDIX ERA

RAIDIX ERA [6] — это программный RAID, представленный в виде модуля ядра Linux и управляющей утилиты, которые собраны и сконфигурированы для наиболее популярных дистрибутивов операционной системы Linux (CentOS 7.9, Oracle Linux 7.9, Ubuntu 16.04 LTS, Ubuntu 18.04 LTS, Ubuntu 20.04.1 LTS, Debian 10.6, Proxmox 6.3). Первый релиз системы состоялся в сентябре 2018, в настоящий момент актуальной версией является 3.2, в которой сохранены все достоинства предыдущих версий (реализация до 97% заявленной производительности NVMe-накопителей, поддержка различных уровней RAID, обеспечение высокой скорости чтения и записи данных в режиме смешанной нагрузки), а также добавлен новый функционал: поддержка RAID N+M, возможность увеличения объёма или изменения уровня уже созданного RAID при добавлении к нему новых дисков, поддержка фреймворка DKMS для автоматической пересборки драйвера RAIDIX ERA.

Дистрибутив RAIDIX ERA 3.2 представляет собой два программных пакета для ОС Linux:

- eraraid — модуль ядра Linux, предоставляющий основную функциональность: обеспечивает создание и управление RAID-массивами, которые доступны в виде локальных блочных устройств;

- eraraid-util — управляющая утилита, написанная на языке python версии 3.4. Она обеспечивает удобное взаимодействие с пользователями: утилита выполняет преобразование введенных пользова-

телем команд управления RAID-массивами в команды, которые модуль ядра Linux использует для дальнейшей обработки, а также проверяет правильность введенных аргументов.

Несмотря на высокие показатели эффективности данного продукта, иногда возникает потребность применять дополнительные технологии кэширования для обеспечения оптимальной производительности.

2.2. Кэширование

Кэширование применяется многими устройствами: центральным процессором, жесткими дисками и т.д. Кэш состоит из набора записей. Каждая запись ассоциирована с небольшой частью данных, являющейся копией некоторой информации внутреннего хранилища. У каждой записи также есть тег, который определяет соответствие между элементами данных в кэше и их копиями в основной памяти.

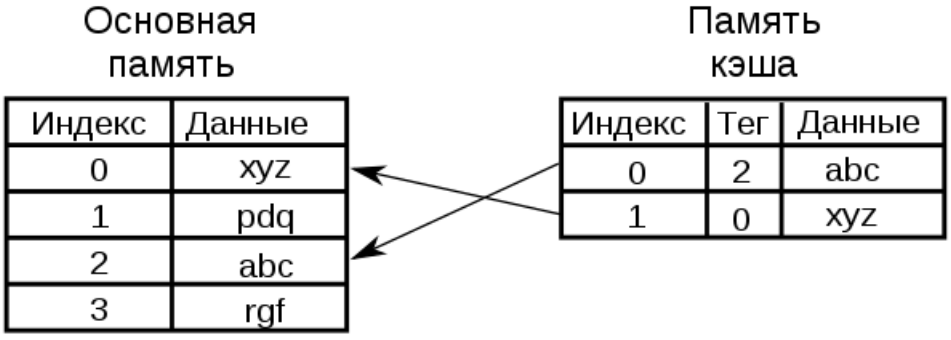

Рис. 2: Отображение кэша в основной памяти. Источник [13]

При выполнении операции ввода-вывода сначала проверяется, есть ли запрашиваемые данные в кэше. Если запись может быть найдена с тегом, совпадающим с тегом требуемых данных, то вместо информации из основной памяти используются данные из кэша. Такой случай называется попаданием кэша. Противоположная ситуация, когда в кэше нет записей с желаемым тегом, называется промахом кэша. При недостаточном количестве места в хранилище кэша при промахе может быть отброшена некоторая запись для освобождения пространства. Какую

именно запись помещать в основную память выбирается в зависимости от политики вытеснения [13].

Использование RAID-кэша позволяет увеличить производительность во многих ситуациях:

- **При чтении**

  "Горячие данные" (hot data) — это данные, к которым необходимо часто обращаться. Примером "горячих данных" могут служить метаданные и частые пользовательские обращения к одному ресурсу. Горячие данные обычно хранятся в самом быстром хранилище. "Холодные данные" (cool data) — информация, к которой организация обращается реже. Эти данные часто находятся в менее производительных и менее дорогих средах хранения. Существует множество алгоритмов, позволяющих выделять "горячие данные" и помещать их в кэш.

- **При записи**

  Стоит заметить, что при использовании в RAID-массивах контрольных сумм для одной операции записи необходимо выполнить следующую последовательность действий: чтение информации из адресуемого блока и соответствующих ему синдромов, пересчет контрольных сумм, запись адресуемого блока и новых контрольных сумм.

  1. Часто возникает ситуация, когда информация поступает в виде большого количества малых по объему блоков данных. Одним из вариантов повышения производительности в данном случае является накопление достаточного количества информации в отдельном месте с последующей записью ее на диск. Этот подход реализуется политикой write-back: данные записываются сначала в кэш, и только потом (либо по мере заполнения кэша, либо в моменты минимальной загрузки дисковой системы) из кэша на диски.

2. Иногда необходимо записывать данные на медленные диски последовательно, а не в произвольные его блоки (второй подход сильно замедляет работу).

Что может выступать в качестве RAID-кэша:

1. NVDIMM — пример совместного использования двух технологий: энергонезависимой и оперативной памяти. Это позволяет избежать потери данных в кэше при незапланированном отключении питания или аппаратных отказах компонентов системы хранения данных.

2. Дополнительный диск (HDD/SSD) — существенно дешевле, чем использование NVDIMM, но медленнее.

3. RAM-диск — программная технология, позволяющая хранить данные в оперативной памяти как на блочном устройстве.

NVDIMM и RAM-диски хороши также и тем, что повышают долговечность и надежность твердотельных накопителей, уменьшая количество обрабатываемых с помощью SSD запросов (твердотельные накопители имеют ограниченное количество циклов записи).

2.3. Реализованные продукты компании RAIDIX, связанные с кэшированием

SSD-cache RAIDIX 5.X [8]

Одной их технологий компании RAIDIX является SSD-кэш, имеющий две особенности: разделение с помощью детектора входящих запросов на категории: для случайных запросов на чтение — RRC (Random Read Cache) и для случайных запросов на запись — RWC (Random Write Cache); а также использование log-структурированной записи (механизм последовательной записи блоков данных без учета их логической адресации) для собственных алгоритмов вытеснения.

SSD-кэширование используется, когда HDD накопители не имеют физической возможности обеспечить желаемый результат, также оно

позволяет ускорить работу всей системы и продлить срок службы твердотельных накопителей благодаря алгоритмам перезаписи и специальному детектору нагрузки. Данный продукт существенно повышает производительность СХД при работе со смешанным типом нагрузки. Например, при случайном чтении (80% попадание в кэш, 20% попадание на HDD), количество операций ввода-вывода увеличивается в 6.5 раз, а при 100% попадании в кэш — в 34 раза.

NVDIMM-N-cache RAIDIX 5.X [7]

В RAIDIX реализована поддержка защиты кэша на запись при помощи энергонезависимой памяти NVDIMM-N. Этот вид памяти применяется для сохранения целостности данных в непредвиденных ситуациях при использовании механизма write-back. При таком способе записи данных они сначала помещаются в кэш, а затем в упорядоченном виде подаются непосредственно на накопитель. В таком варианте взаимодействия с данными существует промежуток времени, когда информация уже передана в кэш, но еще не записана на диск. В связи с этим и возникает потребность в использовании энергонезависимой памяти. В стандартном режиме NVDIMM работает как обычный модуль оперативной памяти, поэтому скорость обработки операций ввода-вывода данного продукта очень высока.

Представленные в RAIDIX технологии разработаны с расчетом либо на большие денежные затраты на реализацию (NVDIMM-N-cache), либо наоборот обходятся дешевле, но не обеспечивают достаточную производительность (SSD-cache). Также эти технологии не оптимизированы под RAIDIX ERA. В связи с этим реализация RAM-кэша для RAIDIX ERA является актуальной задачей, в рамках которой планируется увеличить производительность при меньших материальных расходах.

2.4. Тестирование производительности систем хранения данных

В большинстве случаев тестирование производительности систем хранения данных осуществляется двумя различными способами: нагрузкой непосредственно на блочное устройство или с использованием файловой системы. В данной работе рассматривается первый вариант, так как будучи промежуточным слоем между прикладным/системным ПО и дисковым пространством, файловая система должна более негативно влиять на показатели производительности системы и не давать "чистых" результатов.

Для тестирования блочных устройств в операционной системе Linux используются распространенные бенчмарки (инструменты для оценки производительности): Flexible I/O Tester (FIO) [1] и Oracle VDbench [3]. Они позволяют генерировать различные типы нагрузки в соответствии с заданным конфигурационным файлом. Утилиты считают IOPS (количество операций ввода-вывода, выполняемых СХД, за одну секунду) и пропускную способность системы, а также позволяют оценить глубину очереди операций ввода-вывода. Кроме того FIO имеет возможность последовательного запуска сценариев тестирования. Обычно Fio используется для генерации однотипной нагрузки, VDbench же предоставляет инструменты для проведения более сложных тестов и эмуляции паттернов нагрузки реальных приложений.

3. Анализ технологий кэширования

В первой секции данной главы описана методика тестирования с помощью FIO. В следующих секциях рассмотрены технологии кэширования Open CAS, bcache и dm-cache и приведены результаты сравнения производительности данных технологий на основе описанной в первой секции методики.

3.1. Описание тестового сервера и методики тестирования с помощью FIO

Характеристики тестового сервера: Oracle Linux Server 8.3, версия ядра 3.11, Intel(R) Xeon(R) CPU E5-2620 v4 (16x 2.10GHz), 32G RAM.

Тестирование осуществлялось для трех технологий кэширования: Open CAS, dm-cache и bcache при одинаковых параметрах. Диски, входящие в RAID-массив – это HDD-диски, уровень RAID – 6, количество дисков в СХД – 6. Объем кэша – 3 ГБ, объем основного устройства 767 ГБ. Размер линии кэша (размер блока данных, которым оперирует кэш) – 64 КБ. Тестировалось 20% попадание в кэш, 80% попадание на основное устройство.

Рассматривалось **пять различных шаблонов ввода-вывода:**

- read – последовательное чтение,
- write – последовательная запись,
- randread – случайное чтение,
- randwrite – случайная запись,
- randrw – случайные смешанные чтение и запись (соотношение 50/50).

С помощью параметра numjobs в конфигурационном файле создается указанное количество клонов задания. Каждый клон создается как

независимый поток. Данный параметр настраивает определенное количество потоков, выполняющих одно и то же. Каждый поток сообщается отдельно; статистика выводится по клонам в целом с помощью параметра group_reporting. В этой главе представлены результаты для 1, 4 и 16 потоков.

Тестировалась также глубина очереди операций ввода-вывода (параметр iodepth в конфигурационном файле). Этот параметр указывает, сколько операций ввода-вывода могут одновременно обрабатываться. Обычно iodepth=1 указывает, что невозможно обработать операцию до тех пор, пока предыдущая операция не будет завершена. Использование низкой глубины очереди помогает имитировать приложение с высокой степенью параллелизма. В ином случае происходит увеличение глубины очереди до достижения требуемой производительности. В данном разделе представлены результаты для глубины 1, 4 и 16.

Рассматрвалось три варианта кэширующего устройства – SSD-диск, NVMe-диск, RAM-диск.

Результаты представлены для режима кэширования Write-Back, так как это один из основных режимов кэширования [13].

3.2. Open Cache Acceleration (Open CAS)

Open Cache Acceleration (Open CAS) [2] – это проект с открытым исходным кодом, охватывающий программные библиотеки блочного кэширования, адаптеры, инструменты и многое другое. Основная цель этого ПО — повышение производительности центров обработки данных за счет интеллектуального кэширования, а не чрезмерных затрат. Open CAS создает многоуровневый кэш, который оптимизирует использование системной памяти и автоматически определяет наиболее подходящий уровень кэша для ”горячих данных”.

Это программное обеспечение поддерживет шесть режимов кэширования, из которых в данной работе рассматриваются три:

1. **Write-Through/wt:** (используется по умолчанию) в режиме Write-Through механизм кэширования записывает данные в кэш-па-

мять и одновременно записывает те же данные в основную память. Этот режим ускоряет только операции чтения, так как записи должны выполняться как в основную память, так и в кэш.

2. **Write-Back/wb:** в режиме Write-Back данные записываются сначала в кэш, и только потом (в зависимости от политики вытеснения) из кэша в основную память. При этом сигнал о завершении операции записи передается управляющей ОС сразу же по получении данных кэшем контроллера. Этот режим ускоряет и операции чтения, и операции записи, но существует риск потери данных, если с кэшем произошла аварийная ситуация, и данные не успели записаться в основное хранилище.

3. **Write-Only/wo:** в режиме Write-Only операции записи обрабатываются также, как и в Write-Back режиме, однако операции чтения не продвигают данные в кэш-память. Write-Only режим ускоряет только интенсивные операции записи, так как при чтении информация напрявляется напрямую в основное хранилище. Аналогично wb-режиму существует риск потери данных.

Ниже представлены результаты в IOPS, полученные с помощью FIO для Open CAS.

поток	глубина	read	write	randread	randwrite	randrw
1	1	11707	885	76	93	60
1	4	37129	1483	224	143	161
1	16	68404	1618	462	228	139
8	1	19708	6442	309	146	114
8	4	22194	6703	584	226	177
8	16	28335	2636	885	263	208
16	1	19476	4704	605	162	123
16	4	21449	2842	852	242	190
16	16	26203	2311	1054	265	222

Таблица 1: Результаты в IOPS для Open CAS: wb-режим для ERA RAID как основного устройства и SSD в качестве кэша

поток	глубина	read	write	randread	randwrite	randrw
1	1	12463	934	96	96	67
1	4	39627	1561	251	148	176
1	16	73139	1776	514	248	145
8	1	21106	7065	399	156	123
8	4	23726	6871	650	247	192
8	16	31776	2947	930	277	225
16	1	20993	4961	659	173	132
16	4	23039	3060	904	254	197
16	16	27855	2385	1177	289	240

Таблица 2: Результаты в IOPS для Open CAS: wb-режим для ERA RAID как основного устройства и NVMe в качестве кэша

поток	глубина	read	write	randread	randwrite	randrw
1	1	13700	1015	106	114	75
1	4	43400	1713	280	174	199
1	16	77900	1900	562	279	172
8	1	22800	7537	432	177	142
8	4	25800	7712	738	275	221
8	16	33200	3081	1045	317	263
16	1	22200	5372	715	202	151
16	4	24500	3234	1005	294	230
16	16	30100	2653	1324	333	271

Таблица 3: Результаты в IOPS для Open CAS: wb-режим для ERA RAID как основного устройства и RAM в качестве кэша

Из результатов видно, что наибольшая производительность достигается при использовании RAM-диска в качестве кэширующего устройства. Именно на основе Open CAS планируется реализация RAM cache для ERA RAIDIX.

3.3. bcache

bcache [11] — это компонент ядра Linux. Он позволяет одному или нескольким быстрым устройствам хранения выступать в качестве кэша для одного или нескольких более медленных устройств хранения. Т.к. bcache разработан с учетом характеристик производительности твердотельных накопителей, он избегает случайные записи, превращая их в последовательные. Это объединение операций ввода-вывода помога-

ет продлить срок службы устройств, используемых в качестве кэшей, а также повышает производительность чувствительного к записи основного хранилища. Последовательный ввод-вывод обнаруживается и обходится с настраиваемыми пороговыми значениями. Обход также может быть отключен. Особенностями bcache являются высокоэффективная реализация write-back — грязные данные всегда записываются в отсортированном порядке, и, возможно, фоновая wb-запись плавно снижается, чтобы сохранить сконфигурированный процент кэша грязным; использование высокопроизводительных деревьев B+ — bcache способно значительно повышать скорость операций ввода-вывода в секунду при произвольном чтении, если оборудование достаточно быстрое.

bcache — это кэш на уровне блоков ядра Linux. По умолчанию установлен в ядрах 3.11 и выше. Поставляется вместе с bcache-tools — инструментами пользовательского пространства для bcache. В отличие от dm-cache, требует только указания основного и кэширующего устройства (как и в Open CAS, раздел метаданных создается автоматически).

Это программное обеспечение поддерживает четыре режима кэширования: write-back, write-through, pass-through и write-around (режим кэширования write-around ускоряет только операции с интенсивным чтением. Write-Around дополнительно оптимизирует кэш, чтобы избежать загрязнение кэша в случаях, когда записанные данные впоследствии не читаются повторно).

Кэш-устройство обязательно должно быть отформатировано перед созданием.

Ниже представлены результаты в IOPS, полученные с помощью FIO для bcache. Из результатов видно, что наибольшая производительность достигается при использовании RAM-диска в качестве кэширующего устройства.

поток	глубина	read	write	randread	randwrite	randrw
1	1	14333	829	98	37	34
1	4	40250	1325	202	90	91
1	16	88815	1499	471	131	131
8	1	19998	3021	350	119	100
8	4	71278	3208	625	138	131
8	16	93819	2170	933	176	150
16	1	19868	3183	459	131	106
16	4	57337	2135	674	167	138
16	16	57516	1113	951	180	156

Таблица 4: Результаты в IOPS для bcache: wb-режим для ERA RAID как основного устройства и SSD в качестве кэша

поток	глубина	read	write	randread	randwrite	randrw
1	1	15566	880	125	39	38
1	4	42460	1463	234	95	94
1	16	92414	1599	480	147	146
8	1	21928	3299	382	128	107
8	4	75222	3456	701	150	137
8	16	99699	2290	1043	191	167
16	1	20666	3307	516	143	115
16	4	62222	2246	803	176	149
16	16	62258	1243	1123	184	168

Таблица 5: Результаты в IOPS для bcache: wb-режим для ERA RAID как основного устройства и NVMe в качестве кэша

поток	глубина	read	write	randread	randwrite	randrw
1	1	16800	940	135	44	43
1	4	47100	1513	264	113	112
1	16	101000	1708	551	164	166
8	1	23400	3494	429	147	124
8	4	81200	3681	794	167	158
8	16	107000	2525	1133	215	186
16	1	22600	3704	573	163	130
16	4	67000	2459	898	205	174
16	16	66000	1298	1220	219	191

Таблица 6: Результаты в IOPS для bcache: wb-режим для ERA RAID как основного устройства и RAM в качестве кэша

3.4. dm-cache

dm-cache [12] является компонентом ядра Linux device mapper (dm).

Он, как и bcache, направлен на повышение производительности блочного устройства за счет динамического переноса некоторых данных на более быстрое и компактное устройство.

Конструкция dm-cache требует подробной настройки и четкого указания трех разделов: основное устройство, устройство кэширования и устройство под метаданные, а также требует точного подсчета количества секторов, что усложняет работу.

Это программное обеспечение поддерживает три режима кэширования: write-back, write-through and pass-through (В режиме pass-through механизм кэширования будет обходить кэш для всех операций. Если кэш содержал грязные данные до переключения в режим pass-through, то попадания кэша при чтении будут обрабатываться путем чтения данных из хранилища кэша, пока все строки кэша не будут очищены).

Ниже представлены результаты в IOPS, полученные с помощью FIO для dm-cache. Из результатов видно, что наибольшая производительность достигается при использовании RAM-диска в качестве кэширующего устройства.

поток	глубина	read	write	randread	randwrite	randrw
1	1	17949	1108	62	36	22
1	4	44527	1853	226	93	61
1	16	88610	2197	402	132	108
8	1	21385	2490	320	116	86
8	4	71402	3101	583	145	116
8	16	94695	2624	726	170	135
16	1	20810	2462	455	132	103
16	4	59866	2443	655	154	129
16	16	57556	1511	889	175	144

Таблица 7: Результаты в IOPS для dm-cache: wb-режим для ERA RAID как основного устройства и SSD в качестве кэша

поток	глубина	read	write	randread	randwrite	randrw
1	1	18993	1204	76	39	24
1	4	49170	2029	276	102	63
1	16	90374	2437	519	142	113
8	1	22801	2593	376	125	98
8	4	74376	3429	638	150	124
8	16	99563	2699	928	176	148
16	1	22669	2608	522	144	110
16	4	63523	2575	782	167	136
16	16	63739	1559	1108	191	149

Таблица 8: Результаты в IOPS для dm-cache: wb-режим для ERA RAID как основного устройства и NVMe в качестве кэша

поток	глубина	read	write	randread	randwrite	randrw
1	1	21100	1278	85	44	28
1	4	51400	2175	298	113	74
1	16	101000	2557	561	165	132
8	1	24400	2860	409	143	109
8	4	82100	3599	716	177	142
8	16	111000	3021	1022	205	167
16	1	24100	2864	570	164	129
16	4	69400	2872	863	186	158
16	16	67100	1729	1201	214	176

Таблица 9: Результаты в IOPS для dm-cache: wb-режим для ERA RAID как основного устройства и RAM в качестве кэша

3.5. Сравнение технологий

В приложении 1 представлены результаты сравнения технологии Open CAS с bcache и dm-cache, полученные на основе результатов из предыдущих пунктов.

Open CAS в некоторых случаях проигрывает dm-cache при последовательных чтении/записи и bcache при случайном чтении. Это связано с внутренней структурой dm-cache и bcache: наличие дополнительных оптимизаций для определенных сценариев использования. Но при смешанных и случайных нагрузках (а также при некоторых последовательных) Open CAS демонстрирует существенный выигрыш в производительности.

Open CAS обладает подробной документацией, легок в настройке,

предоставляет шесть режимов кэширования (в dm-cache — три, в bcache — четыре), а также поддерживает легко настраиваемое многоуровневое кэширование. Проект активно развивается и демонстрирует в большинстве случаев наилучшую производительность среди представленных технологий, поэтому в дальнейшем будет рассматриваться именно Open CAS.

4. Адаптация Open CAS для RAIDIX ERA

Раздел включает в себя рассмотрение архитектуры Open CAS и Open CAS Framework, описание результатов проведенного подробного тестирования RAM-кэша Open CAS для RAIDIX ERA и разработанного на основе этих результатов оффлайн-анализатора входящих запросов, а также описание внедренных в Open CAS политик вытеснения.

4.1. Описание работы Open CAS

В основе Open CAS лежит Open CAS Framework (OCF) — это высокопроизводительная мета-библиотека кэширования блочного хранилища, написанная на C. Она полностью независима от платформы и системы, получает доступ к системному API через слой оболочек среды, предоставляемый пользователем. OCF тесно интегрируется с остальным программным стеком, реализуя оптимальную утилиту кэширования с малой задержкой. Open CAS Framework — это основа, с помощью которой Open CAS Linux предоставляет полные и удобные решения для кэширования.

В первую очередь Open CAS был разработан для кэширования данных с жестких дисков на твердотельных накопители, но также может использоваться для кэширования данных с помощью накопителей Optane, оперативной памяти или любой комбинации вышеперечисленного, включая всевозможные многоуровневые конфигурации.

В OCF существует несколько типов абстракций для успешного взаимодействия с элементами, участвующими в кэшировании [2]:

1. volume — это общее представление хранилища, которое позволяет OCF получать доступ к различным типам хранилищ (кэш-хранилище или основное хранилище), используя общий абстрактный интерфейс. Хранилище, представленное volume, может быть любым устройством, допускающим произвольный блочный доступ;

2. core object (core) — это абстракция, которая позволяет приложению получить доступ к кэшированному основному хранили-

щу. Core object предоставляет интерфейс для отправки запросов ввода-вывода, которые обрабатываются в соответствии с текущими параметрами конфигурации. Во время нормальной работы кэша основное хранилище принадлежит исключительно core object, и приложение никогда не должно обращаться к нему напрямую;

3. cache object (cache) — в OCF предоставляет интерфейс для взаимодействия с кэш-устройством. Этот интерфейс позволяет подключать кэш-хранилище, добавлять и удалять core object, изменять различные параметры конфигурации и получать статистику. Каждый кэш работает с одним хранилищем кэша (представленным cache volume), где хранятся данные и метаданные. Cache object позволяет эффективно работать с многоуровневым кэшированием, а также обрабатывать данные от многих core object-ов.

В данной главе рассматривается только одно основное устройство – RAID-массив ERA RAID и RAM-диск в качестве кэширующего устройства.

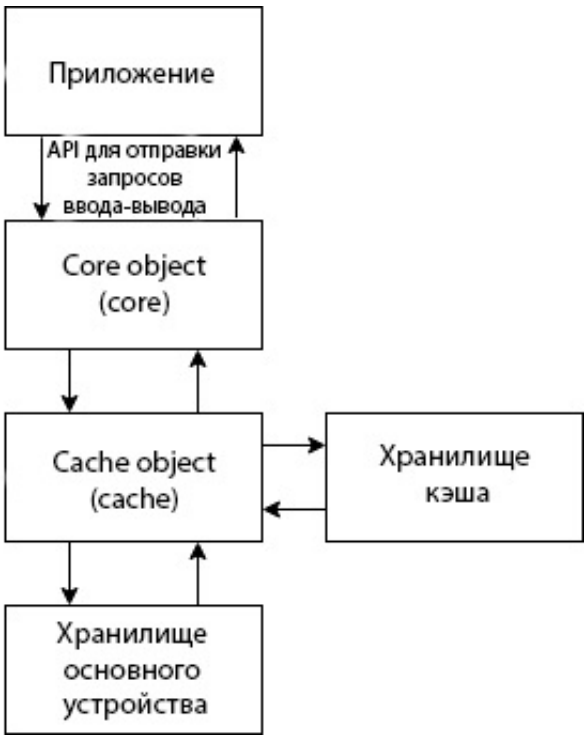

Рис. 3: Основные объекты OCF и их взаимодействие. Источник [2]

Общая схема, описывающая программный стек, предоставляющий кэш Open CAS Linux, показана ниже:

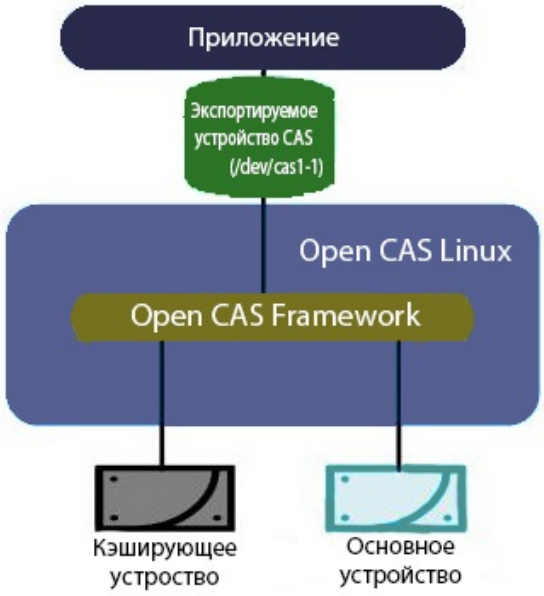

Рис. 4: Общая схема Open CAS Linux. Источник [2]

4.2. Внедрение и тестирование Open CAS для RAIDIX ERA

С помощью бенчмарка FIO было проведено подробное тестирование RAM-кэша в зависимости от накопителей и сценариев использования. Исследовались случайные и последовательные нагрузки при разном проценте чтения (rwmixread): 0%, 30%, 50%, 70%, 100%; различной глубине очереди операций ввода-вывода (iodepth) 1, 8, 16, а также количестве потоков (numjobs) равном 1, 4, 8, 16. Подробный отчет, описание используемых тестовых серверов и другие дополнительные файлы представлены в репозитории [14].

Несмотря на то, что RAIDIX ERA оптимизирована под NVMe, SAS, SATA накопители, наилучшие показатели производительности при работе с ними достигаются при большом количестве потоков и глубине очереди, при малых значениях этих параметров скорость работы RAIDIX ERA может быть не так велика в сравнении с производительностью оборудования. В связи с этим было исследовано применение Open

CAS с RAM-кэшом для SSD и NVMe накопителей.

Основной вывод по каждому из накопителей представлен ниже. Стоит отметить, что при малых параметрах iodepth и numjobs и большом размере блока, используемого для операций ввода-вывода (в таблицах для 1 мегабайта), Open CAS не показывает существенного прироста производительности (а зачастую даже замедляет работу) для любого из дисков. Это связано с размером строки кэша (единица передачи данных между кэшем и основной памятью): CAS поддерживает 4 (по умолчанию), 8, 16, 32, 64 килобайтные строки кэш-памяти. Также Open CAS лучше всего оптимизирует работу при смешанных нагрузках. Чем больше параметр rwmixread (т.е. чем больше операций чтения), тем меньшее наблюдается ускорение. При rwmixread=100% использование данной технологии кэширования не является целесообразным. Продукт показывает наибольшую эффективность при малых значениях numjobs и iodepth.

Дополнительно для SSD и NVMe было проведено тестирование бэнчмарком VDbench на основе паттернов нагрузки SNIA (Storage Networking Industry Association) [9]. Данный набор тестов позволяет эмулировать некоторые стратегии поведения при нагрузке реальных систем. Open CAS продемонстрировал существенное увеличение производительности для всех сценариев использования SSD, а также для NVMe при iodepth не более 10 и numjobs не более 8.

Тестировалось 20% попадание в кэш, 80% попадание на основное устройство. В таблицах % — на сколько процентов Open CAS производительнее (в IOPS), чем RAIDIX ERA без использования кэширующего устройства. Для SSD и NVMe приведены результаты для паттернов VDbench, подробное описание этих паттернов представлено в [14]. Зеленым помечены те строки, в которых использование Open CAS дает выигрыш в производительности более 15%, красным — в которых использование кэша снижает количество операций ввода-вывода более, чем на 15%, желтым — все остальные случаи.

- **HDD:** для всех представленных паттернов, кроме случайного и последовательного чтения, Open CAS демонстрирует увеличение

количества операций ввода-вывода в секунду (IOPS). При смешанных чтении/записи наблюдается увеличение IOPS в несколько раз.

паттерн	блок	поток	глубина	IOPS CAS	IOPS ERA	%
randwrite	4k	1	1	299	47	536
write	4k	1	1	721	657	10
randrw	4k	1	1	71	31	129
rw	4k	1	1	2144	1268	267
randread	4k	1	1	86	82	5
read	4k	1	1	19100	23700	-19
randwrite	4k	4	8	368	241	53
write	4k	4	8	11100	6908	61
randrw	4k	4	8	293	199	47
rw	4k	4	8	7655	3695	107
randread	4k	4	8	869	867	0
read	4k	4	8	27500	110000	-75
randwrite	4k	8	16	460	306	50
write	4k	8	16	5632	5518	2
randrw	4k	8	16	420	262	60
rw	4k	8	16	7699	5468	41
randread	4k	8	16	1319	1315	0
read	4k	8	16	34600	109000	-68

Таблица 10: Результаты Open CAS, диски — HDD, FIO

- **SSD:** существенное повышение производительности наблюдается при последовательных паттернах, на случайных же только при numjobs, iodepth равных 1. При последовательных смешанных операциях при небольших значениях глубины очереди и количества потоков наблюдается увеличение IOPS в несколько раз.

паттерн	блок	поток	глубина	IOPS CAS	IOPS ERA	%
randwrite	4k	1	1	8165	4684	74
write	4k	1	1	2170	2403	-9
randrw	4k	1	1	3550	2933	21
rw	4k	1	1	7592	1768	329
randread	4k	1	1	6535	8109	-19
read	4k	1	1	10100	11700	-13
randwrite	4k	4	8	37900	67300	-44
write	4k	4	8	81700	215000	-62
randrw	4k	4	8	22700	49100	-54
rw	4k	4	8	53900	16400	228
randread	4k	4	8	85100	220000	-61
read	4k	4	8	60100	149000	-59

Таблица 11: Результаты Open CAS, диски — SSD, FIO

паттерн	поток	глубина	IOPS CAS	IOPS ERA	%
Four_Neoseq	1	1	7844	2942	166
Neoclasssic	1	1	5793	2924	98
Four_Neoseq	1	4	18028	8826	104
Neoclasssic	1	4	16716	8176	104
Four_Neoseq	1	8	28725	13220	117
Neoclasssic	1	8	23277	12298	89
Four_Neoseq	1	16	30569	18950	61
Neoclasssic	1	16	28551	17800	60

Таблица 12: Результаты Open CAS, диски — SSD, VDbench

- **NVMe:** наибольшее повышение производительности демонстрируется для последовательных операций, а также для write-паттернов при небольшом значении numjobs и iodepth.

паттерн	блок	поток	глубина	IOPS CAS	IOPS ERA	%
randwrite	4k	1	1	8100	5919	37
write	4k	1	1	7734	6076	27
randrw	4k	1	1	3340	3518	-5
rw	4k	1	1	5286	2717	95
randread	4k	1	1	7670	9636	-20
read	4k	1	1	8737	9839	-11
randwrite	4k	4	8	35300	53700	-34
write	4k	4	8	82100	314000	-73
randrw	4k	4	8	28300	50600	-44
rw	4k	4	8	39700	48800	-18
randread	4k	4	8	96000	301000	-68
read	4k	4	8	62600	230000	-72

Таблица 13: Результаты Open CAS, диски — NVMe, FIO

паттерн	поток	глубина	IOPS CAS	IOPS ERA	%
Four_Neoseq	1	1	8202	5079	61
Neoclasssic	1	1	7022	4623	51
Four_Neoseq	1	4	37929	18185	108
Neoclasssic	1	4	31380	16869	86
Four_Neoseq	1	8	49621	32086	54
Neoclasssic	1	8	45168	29502	53
Four_Neoseq	1	16	59548	58089	2
Neoclasssic	1	16	41369	41618	-1

Таблица 14: Результаты Open CAS, диски — NVMe, VDbench

На основе проведенного анализа работы Open CAS с RAIDIX ERA был написан оффлайн-детектор, анализирующий тип входящих запросов ввода вывода (тип дисков, из которых состоит RAID-массив; средний размер поступающих блоков данных; глубина очереди; количество потоков; тип нагрузки) и подбирающий оптимальные параметры кэширующего устройства (размер строки кэша, режим кэширования) либо рекомендующий не использовать Open CAS в описываемой пользователем ситуации.

4.3. Добавление новой функциональности в Open CAS

Open CAS поддерживает только одну политику вытеснения — LRU (least recently used), т.е. в первую очередь вытесняется неиспользованный дольше всех элемент. Open CAS Linux может анализировать каждый запрос ввода-вывода, чтобы определить, является ли запрошенный блок метаданными файловой системы или данными, и, если это данные, размер файла назначения. Используя эту информацию, Open CAS может определить наилучшие параметры конфигурации для типичной рабочей нагрузки установить, какие запросы следует кэшировать, а какие нет, а также установить уровень приоритета для каждого запроса ввода-вывода. В результате, когда возникает необходимость удалить строку кэша, программное обеспечение сначала вытесняет строки кэша

с самым низким приоритетом (значительное улучшение по сравнению с традиционным LRU).

Но существует множество других алгоритмов вытеснения, которые также распространены в сегменте СХД. В связи с разнообразием задач, выполняемых RAIDIX ERA, было принято решение о внедрении дополнительных алгоритмов вытеснения:

- First-In First-Out (FIFO) — последовательно вытесняются наиболее старые блоки, замещаясь наиболее свежими;
- Least Frequently Used (LFU) — первыми вытесняются те блоки данных, к которым было меньше всего обращений, как с использованием анализатора входящих запросов, так и без него.

Выбор данных политик основан на опыте компании RAIDIX, полученном при разработке SSD-cache RAIDIX (в RAIDIX используется FIFO и LRFU — комбинация LRU и LFU), а также связан с небольшими затратами при их реализации.

Изменения производились как в kernel space (добавление новой функциональности), так и в user space (предоставление возможности выбора определенных политик вытеснения пользователем).

Также было рассмотрено влияние анализатора запросов, реализованного разработчиками Open CAS Framework, на производительность (тестирование проводилось как для LRU с анализатором, так и без него). При однотипных нагрузках влияние на производительность незначительное (0-11%); при использовании паттернов, основанных на паттернах SNIA, наблюдается улучшение до 19%.

Реализация LFU оказалась наиболее эффективна, когда есть область данных, которая используется очень часто (в соответствии с принципом работы алгоритма такие данные будут храниться в кэше). При использовании LFU на таких паттернах прирост достигает 25%.

Заключение

В ходе данной работы были получены следующие результаты.

1. Выполнено изучение и сравнение технологий кэширования Open CAS, bcache, dm-cache. Выявлено, что наилучшей производительностью обладает Open CAS RAM-кэш.

2. Было протестировано и проанализировано взаимодействие Open CAS RAM-кэша с RAIDIX ERA. Описаны подходящие сценарии использования Open CAS для разных видов накопителей: HDD, SSD, NVMe.

3. В Open CAS добавлена новая функциональность: дополнительные политики вытеснения (FIFO, LFU), исследовано влияние анализатора запросов Open CAS ввода-вывода на производительность. Исследована производительность внедренной технологии: при определенных шаблонах наблюдается улучшение в среднем на 16%.

4. На основе проведенного тестирования разработан оффлайн детектор, анализирующий входящие запросы ввода-вывода и рекомендующий оптимальные параметры работы Open CAS.

В дальнейшем планируется выполнить задачи, перечисленные ниже.

1. Разработать детектор, который будет способен в онлайн-режиме анализировать входящие запросы ввода-вывода и подбирать оптимальные параметры для работы с Open CAS.

2. Известно, что в RAID-массивах существует проблема write hole — это повреждения данных в RAID-массивах, вызванные прерыванием записи на диск [10]. Проблема write hole может быть устранена с помощью ведения упреждающей журнализации (информация об изменениях вносится и фиксируется в специальный журнал перед записью в RAID-массив). В связи с наличием описанной проблемы планируется произвести внедрение технологии

Open CAS, когда в качестве кэширующего устройства используется энергонезависимая память NVDIMM/Optane, и использовать NVDIMM/Optane в качестве журнала с целью устранения проблемы write hole.

Приложение 1

На пересечении шаблона ввода-вывода с одним из девяти исследуемых сценариев стоит величина, показывающая, на сколько процентов технология Open CAS производительнее (в IOPS), чем bcache или dm-cache.

поток	глубина	read	write	randread	randwrite	randrw
Сравнение Open CAS с bcache, кэш — SSD						
1	1	-18	7	-22	151	76
1	4	-8	12	11	59	77
1	16	-23	8	-2	74	6
8	1	-1	113	-12	23	14
8	4	-69	109	-7	64	35
8	16	-70	21	-5	49	39
16	1	-2	48	32	24	16
16	4	-63	33	26	45	38
16	16	-54	108	11	47	42
Сравнение Open CAS с bcache, кэш — NVMe						
1	1	-20	6	-23	146	76
1	4	-7	7	7	56	87
1	16	-21	11	7	69	-1
8	1	-4	114	4	22	15
8	4	-68	99	-7	65	40
8	16	-68	29	-11	45	35
16	1	2	50	28	21	15
16	4	-63	36	13	44	32
16	16	-55	92	5	57	43
Сравнение Open CAS с bcache, кэш — RAM						
1	1	-18	8	-21	159	74
1	4	-8	13	6	54	78
1	16	-23	11	2	70	4
8	1	-3	116	1	20	15
8	4	-68	110	-7	65	40
8	16	-69	22	-8	47	41
16	1	-2	45	25	24	16
16	4	-63	32	12	43	32
16	16	-54	104	9	52	42

Таблица 15: Сравнение Open CAS и bcache

поток	глубина	read	write	randread	randwrite	randrw
Сравнение Open CAS с dm-cache, кэш — SSD						
1	1	-35	-21	23	158	173
1	4	-17	-20	-1	54	164
1	16	-23	-26	15	73	29
8	1	-8	159	-3	26	33
8	4	-69	116	0	56	53
8	16	-70	0	22	55	54
16	1	-6	91	33	23	19
16	4	-64	16	30	57	47
16	16	-54	53	19	51	54
Сравнение Open CAS с dm-cache, кэш — NVMe						
1	1	-34	-22	26	146	179
1	4	-19	-23	-9	45	179
1	16	-19	-27	-1	75	28
8	1	-7	172	6	25	26
8	4	-68	100	2	65	55
8	16	-68	9	0	57	52
16	1	-7	90	26	20	20
16	4	-64	19	16	52	45
16	16	-56	53	6	51	61
Сравнение Open CAS с dm-cache, кэш — RAM						
1	1	-35	-21	25	159	168
1	4	-16	-21	-6	54	169
1	16	-23	-26	0	69	30
8	1	-7	164	6	24	30
8	4	-69	114	3	55	56
8	16	-70	2	2	55	57
16	1	-8	88	25	23	17
16	4	-65	13	16	58	46
16	16	-55	53	10	56	54

Таблица 16: Сравнение Open CAS и dm-cache

Список литературы

[1] Flexible I/O Tester. — URL: https://github.com/axboe/fio (дата обращения: 28.05.2021).

[2] Open CAS. — URL: https://www.intel.ru/content/www/ru/ru/software/intel-cache-acceleration-software-performance.html (дата обращения: 28.05.2021).

[3] Oracle Vdbench. — URL: https://www.oracle.com/downloads/server-storage/vdbench-downloads.html (дата обращения: 28.05.2021).

[4] RAID Levels. — URL: https://www.ixbt.com/storage/raids.html (дата обращения: 28.05.2021).

[5] RAIDIX. Оборудование СХД: основные компоненты и их назначение. — URL: https://www.raidix.ru/blog/storage-hardware-components/ (дата обращения: 28.05.2021).

[6] RAIDIX ERA. — URL: https://www.raidix.ru/products/era/ (дата обращения: 28.05.2021).

[7] RAIDIX NVDIMM-cache. — URL: https://www.raidix.ru/products/about/nvdimm-support/ (дата обращения: 28.05.2021).

[8] RAIDIX SSD-cache. — URL: https://www.raidix.ru/products/about/ssd-cache/ (дата обращения: 28.05.2021).

[9] SNIA Emerald Power Efficiency Measurement Specification. — URL: https://www.snia.org/sites/default/files/technical_work/Emerald/SNIA_Emerald_Power_Efficiency_Measurement_Specification_V3_0_3.pdf (дата обращения: 28.05.2021).

[10] Write hole phenomenon. — URL: http://www.raid-recovery-guide.com/raid5-write-hole.aspx (дата обращения: 28.05.2021).

[11] bcache. — URL: https://www.kernel.org/doc/html/latest/admin-guide/bcache.html (дата обращения: 28.05.2021).

[12] dmcache. — URL: https://www.kernel.org/doc/html/latest/admin-guide/device-mapper/cache.html (дата обращения: 28.05.2021).

[13] Кэш-память. — URL: https://dic.academic.ru/dic.nsf/ruwiki/997199 (дата обращения: 28.05.2021).

[14] Результаты сравнения для RAM-cache. — URL: https://gitlab.com/vzhereb9/tables-for-diploma (дата обращения: 28.05.2021).