Оптимизация ввода-вывода с помощью кеширующих блочных устройств в среде GNU/Linux

А. Г. Прилипко[1], С. Г. Романюк[2], Д. В. Самборский[3]

[1]НИЦ «Курчатовский институт» - НИИСИ, Москва, Россия, aleksey.prilipko@gmail.com;

[2]НИЦ «Курчатовский институт» - НИИСИ, Москва, Россия, sgrom@niisi.ras.ru;

[3]НИЦ «Курчатовский институт» - НИИСИ, Москва, Россия, samborsky_d@fastmail.com

Аннотация. Современные устройства хранения данных имеют широкий диапазон основных характеристик: объёма, пропускной способности, и скорости записи. В силу физических ограничений в одном устройстве невозможно достичь максимальных значений всех этих характеристик. Тем не менее, совместное использование разных типов накопителей часто позволяет оптимизировать производительность операционной системы для решения прикладных задач. Ядро ОС GNU/Linux позволяет создавать композитные устройства блочного ввода-вывода, такие как программные массивы накопителей данных (RAID-устройства) и кеширующие программные устройства. В данной статье выполнен анализ системных средств и ядра ОС GNU/Linux с целью поиска стратегии повышения производительности аппаратно-программной конфигурации подсистемы ввода-вывода.

Ключевые слова: ввод-вывод данных, Linux, RAID, dm-cache, dm-writecache

1. Введение

Выбор конфигурации системы хранение данных для вычислительных узлов серверного кластера в последние годы осложняется разнообразием устройств и дороговизной наиболее ёмких и производительных моделей SSD-накопителей. Вместе с тем, морально устаревшие механические HDD-диски также продолжают использоваться, когда требуются хранение и обработка больших массивов информации.

Операционная система GNU/Linux имеет сложную подсистему виртуальных блочных устройств, которая обеспечивает создание и работу программных RAID-массивов, кеширующих устройств, средств контроля целостности данных, и других виртуальных устройств, расширяющих функции системы Linux для работы с накопителями данных. Данная работа рассматривает задачу повышения скорости работы прикладных задач с устройствами хранения данных системными средствами ОС GNU/Linux. В статье рассмотрены три конфигурации использования разнородных накопителей данных и выполнен сравнительный анализ этих конфигураций.

2. Асимметричное RAID1-устройство

В современных версиях ядра Linux есть опции для тонкой настройки алгоритма репликации данных в RAID1-массивах [1]. При наличии условно быстрого устройства (с технологией ячеек памяти Z–NAND, XL– FLASH, или 3D XPoint) и условно медленного SSD- или HDD-устройства можно попробовать создать такое RAID1-устройство, в котором запись на медленное устройство откладывается и не замедляет работу прикладной программы. Это достигается с помощью опции write-mostly, указывающей на медленное устройство, и опции write-behind, задающей объем данных, запись которых на это устройство допускается задержать. Увеличение объема отложенных для записи данных вызывает эффект сглаживания периодов интенсивной нагрузки на подсистему ввода-вывода со стороны прикладных программ.

Опция write-mostly обозначает медленное устройство и подсказывает, что чтения данных лучше выполнять с другого, более быстрого устройства. Следует отметить, что действие опции write-behind также распространяется и на операции принудительного завершения операций записи (системные вызовы sync и datasync).

Другими словами, опция write-behind нарушает контракт, согласно которому работает

программный RAID1-массив, поскольку при её использовании в течение некоторого периода времени теряется гарантия наличия двух экземпляров записанных данных. При выходе из строя быстрого диска данные будут потеряны, что может привести к некорректному состоянию прикладных программ. В случае внезапного выключения сервера сразу после перезагрузки на этом RAID1-диске будет запущена процедура восстановления данных. Данная процедура использует битовую карту неполных блоков и для каждого такого блока сделает запись в правильном направлении — запишет данные на медленный диск, поскольку опция write-mostly заставляет алгоритм драйвера RAID1-массива избегать чтение данных с медленного диска. По этой же причине прикладные программы прочитают корректный блок данных, даже если эта операция чтения опередит процесс восстановления массива и прочитает блок, у которого еще различаются копии данных.

Таким образом, если имеется два быстрых и два медленных устройства, то есть два способа построения RAID -массивов: (1) - симметричные RAID1-массивы из быстрых и медленных дисков, и (2) - два асимметричных RAID1-массива. Во втором способе будет в 2 раза больше объем данных, доступ к которым будет осуществляться преимущественно через быстрый накопитель.

Суммируя, можно перечислить следующие достоинства и недостатки асимметричных RAID1-массивов:

- *Достоинства:* Простота конфигурации. Высокая скорость чтения; Высокая скорость записи, если объем записанных данных в течении короткого промежутка времени не превышает значения write-behind.
- *Недостатки:* Риск потери данных в случае выхода из строя быстрого устройства; Неполное использование объема «медленного» диска; Повышение нагрузки на быстрое устройство операциями чтения данных.

3. Универсальное кеширующее устройство (dm-cache)

Для кеширования чтения и записи данных на медленных накопителях в ОС GNU/Linux существует драйвер dm-cache [2]. Этот драйвер позволяет использовать быстрый накопитель в качестве кеш-устройства для ускорения операций ввода-вывода на медленном устройстве, базовом устройстве. Для обеспечения надежности хранения данных в качестве базового и кеш-устройства используются RAID-массивы.

Кеширование данных драйвером dm-cache может выполняться в двух основных режимах: writeback и writethrough. По умолчанию используется режим отложенной записи (writeback), в котором операция записи считается завершенной, когда данные записаны либо на обоих устройствах, либо только на кеш-устройстве. В последнем случае блок данных кеш-устройства помечается как несинхронизированный и его запись на базовое устройство откладывается на неопределенное время. Операции принудительной записи данных (системные вызовы sync и datasync) не влияют на алгоритм кеширования и не требуют ожидания записи на базовое устройство, поэтому прикладные программы, требующие синхронного ввода-вывода, также видят повышение скорости записи. В режиме сквозной записи (writethrough) данные записываются сразу на оба устройства, и только после этого операция считается выполненной. Такой режим ускоряет операции чтения недавно прочтённых или записанных данных.

В обоих режимах операции чтения блоков данных с базового устройства вызывают запись этих блоков в кеш-устройство для их возможного повторного использования. Если основное назначение кеш-устройства состоит в ускорении записи, то кеширования прочтённых данных следует избегать. До версии ядра Linux v.4.6 для этого было достаточно использовать настройки read-promote-adjustment и write-promote-adjustment, которые определяли баланс кеширования чтения и записи. Однако, начиная с версии v4.6 единственной политикой кеширования в драйвере dm-cache стала новая политика кеширования smq (stochastic multi-queue), в которой используется стохастический алгоритм, не имеющий никаких пользовательских настроек [4].

Таким образом, универсальное устройство кеширования дает наибольший эффект, когда кеширование чтения тоже необходимо – либо, когда быстрый накопитель имеет достаточно большой объём.

4. Устройство кеширования записи (dm-writecache)

Для кеширования только операций записи данных в ядро ОС Linux начиная с версии

v.4.18 был добавлен драйвер dm-writecache [3]. Основными мотивациями написания этого драйвера были сложность настройки универсального драйвера dm-cache и тот факт, что с удешевлением оперативной памяти задача кеширования чтения данных стала хорошо решаться общесистемным файловым кешем (page cache, [5]). Драйвер dm-writecache также способен использовать накопители в формате модулей памяти DIMM, называемых NVDIMM (non-volatile DIMM) или Pmem (persistent memory), которые в силу интерфейса шины DIMM имеют наименьшую задержку при передачи данных.

При создании данного кеширующего устройства допустимо указать параметры алгоритма работы кеша [3], которые влияют на интенсивность синхронизации данных кеш-устройства с базовым устройством. Параметры по умолчанию обеспечивают режим приоритетного обслуживания операций записи: алгоритм старается как можно быстрее сбросить кешированные данные на базовый диск, чтобы кеш был готов к будущим всплескам операций записи. Если после пиковой нагрузки операциями записи наблюдается замедление операций чтения, то следует уменьшить значения параметров writeback_jobs и pause_writeback.

Когда необходимо одновременно решать задачи хранения данных большого объёма и обеспечения высоких скоростей синхронной записи, то оказывается оправдано применение трехуровневого устройства: кеш записи dm-writecache подключен к кеш-диску dm-cache, работающему в режиме writethrough и подключенному к медленным устройствам, таким как RAID10-массив состоящих их механических дисковых накопителей (HDD). Например, сервер виртуализации требует хранения большого объема данных для виртуальных ОС, и, хотя большая часть этих данных является «холодными» и редко модифицируется, внутри виртуальных ОС встречаются приложения, дающие высокую нагрузку записи. В таком случае трудно разделить «горячие» и «холодные» данные по разным системам хранения, и требуется универсальное скоростное и объёмное хранилище. Если старт или возобновление работы некоторых прикладных систем замедляется чтением данных с самого медленного накопителя, то предварительный или периодический «прогрев» общесистемного файлового кеша и кеша чтения могут выполнить утилиты vmtouch [7] и vmprobe [8]. Ядро Linux также имеет системный вызов mincore, который сообщает список находящихся в файловом кеше блоков указанного файла. Применение этого системного вызова позволяет построить карту часто используемых данных и удерживать эти данные в кеше чтения внешним инструментом, без модификации прикладных систем или виртуальных ОС [6].

5. Использование RAM-диска

Предварительную оценку максимального теоретического ускорения операций ввода-вывода от применения кеширующего накопителя данных можно сделать с помощью виртуального блочного устройства, хранящего данные в оперативной памяти (RAM-диск). Такое устройство теряет данные при внезапном выключении питания и его недопустимо применять в рабочей конфигурации, но оно имеет заведомо более высокую скорость чтения и записи данных и поэтому при тестировании покажет максимально достижимый прирост производительности. Блочное устройство RAM-диска нужного размера создается с помощью загрузки модуля ядра brd [9] с опцией rd_size, задающей размер устройства в килобайтах. Далее созданное устройство /dev/ram0 готово к использованию и с точки зрения операционной системы неотличимо от обычного устройства хранения данных.

Такое тестирование производительности драйверов dm-cache и dm-writecache не совсем корректно, так как объем памяти, доступной для создания RAM-диска, обычно меньше размера даже наиболее быстрых SSD-накопителей. Однако, для задач с короткими периодами интенсивной записи данных необходимый объем кеша записи может оказаться меньше размера RAM-диска, и тогда оценка прироста производительности будет довольно точной.

6. Измерения производительности различных конфигураций дисковых подсистем

Для оценки роста производительности операций ввода-вывода при использовании быстрых SSD-накопителей в качестве кеша записи было выполнено тестирование утилитой FIO версии 3.33 [10]. В тесте использовались три накопителя данных: PCIe-плата Optane SSD 900P 280GB, использующая технологию 3D XPoint для ячеек памяти, NVMe-накопитель средней ценовой категории Apacer AS2280P4U

2TB, и механический HDD-накопитель Western Digital WD43PURZ-74BWPY0 4TB.

Запуск теста выполнялся командой

fio --bs=4k --ioengine=libaio --iodepth=32 --direct=1 --rw=randwrite --numjobs=1 ...,

которая инициирует операции записи коротких блоков данных однопотоковой нагрузкой. Этот режим соответствует наиболее требовательным прикладным программам, в которых узким местом становится ожидание подтверждения последней операции записи. Тест применялся к каждому из накопителей, а после этого к нескольким комбинациям кеш-устройства и базового устройства (асимметричный RAID1-массив не тестировался). Кеш-устройство создавалось драйвером dm-writecache в режиме writeback с параметрами по умолчанию. Длительность теста была выбрана равной 5 мин, что приблизительно соответствует среднему времени всплесков активности прикладных задач.

Результаты данного теста (Таблица 1) показывают, что кеширование записи позволяет почти полностью сохранить скорость кеш-устройства, и при этом использовать весь объем базового накопителя данных.

Таблица 1. Тесты записи блоков размером 4Kb на различные устройства

Режим тестирования	Объем устройства, Gb	IOPS	Медианная задержка, µs
Optane	261	180000	174
Apacer	1908	80000	388
HDD WD	3725	360	90000
Optane + Apacer	1908	131000	237
Apacer + HDD WD	3725	79000	388
Optane + Apacer + HDD WD	3725	123000	260

7. Заключение

Опыт использования накопительных устройств разного типа в серверах с операционной системой GNU/Linux показал, что многоуровневое кеширование позволяет увеличить производительность подсистемы хранения данных без применения наиболее дорогих накопителей категории high-end.

Работа выполнена в рамках темы государственного задания НИЦ «Курчатовский институт» - НИИСИ по теме № FNEF-2024-0001 (1023032100070-3-1.2.1).

Optimizing Input/Output using caching block devices in GNU/Linux environment

A. G. Prilipko, S. G. Romanyuk, D. V. Samborskiy

Abstract. Modern data storage devices vary widely in their key characteristics: capacity, throughput, and write latency. Due to physical limitations, no single device can maximize all of these characteristics simultaneously. However, combining different types of storage often makes it possible to optimize operating system performance for specific application workloads. The GNU/Linux kernel enables the creation of composite block I/O devices, including software RAID arrays and caching devices. This article presents an analysis of system tools and GNU/Linux kernel capabilities, with the goal of developing an algorithm for determining the optimal hardware-software configuration of the I/O subsystem.

Keywords: input-output, Linux, RAID, dm-cache, dm-writecache

Литература

1. Сайт документации ядра Linux, раздел «Device Mapper: RAID». https://www.kernel.org/doc/Documentation/device-mapper/dm-raid.txt (дата обращения 19.12.2025)

2. Сайт документации ядра Linux, раздел «Device Mapper: Cache». https://www.kernel.org/doc/Documentation/device-mapper/cache.txt (дата обращения 19.12.2025)

3. Сайт документации ядра Linux, раздел «Device Mapper: Write cache». https://www.kernel.org/doc/Documentation/device-mapper/writecache.txt (дата обращения 19.12.2025)

4. Сайт документации ядра Linux, раздел «Device Mapper: Cache policies». https://www.kernel.org/doc/Documentation/admin-guide/device-mapper/cache-policies.txt (дата обращения 19.12.2025)

5. Сайт документации ядра Linux, раздел «Memory Management». https://www.kernel.org/doc/html/latest/admin-guide/mm/index.html (дата обращения 19.12.2025)

6. А.Б. Бетелин, Г.А. Прилипко, А.Г. Прилипко, С.Г. Романюк, Д.В. Самборский. Динамический анализ и оптимизация ввода-вывода в среде виртуализации GNU Linux/QEMU/KVM. «Труды НИИСИ РАН», т.14 (2024), №1, 25-32

7. Сайт утилиты vmtouch. https://hoytech.com/vmtouch (дата обращения 19.12.2025)

8. Сайт утилиты vmprobe. https://vmprobe.com/intro (дата обращения 19.12.2025)

9. Сайт документации ядра Linux, раздел «RAM block device driver». https://www.kernel.org/doc/html/latest/admin-guide/blockdev/ramdisk.html (дата обращения 19.12.2025)

10. Сайт утилиты Flexible I/O tester (FIO). https://fio.readthedocs.io/en/latest/index.html (дата обращения 19.12.2025)