gflash

руководство пользователя

Version 14

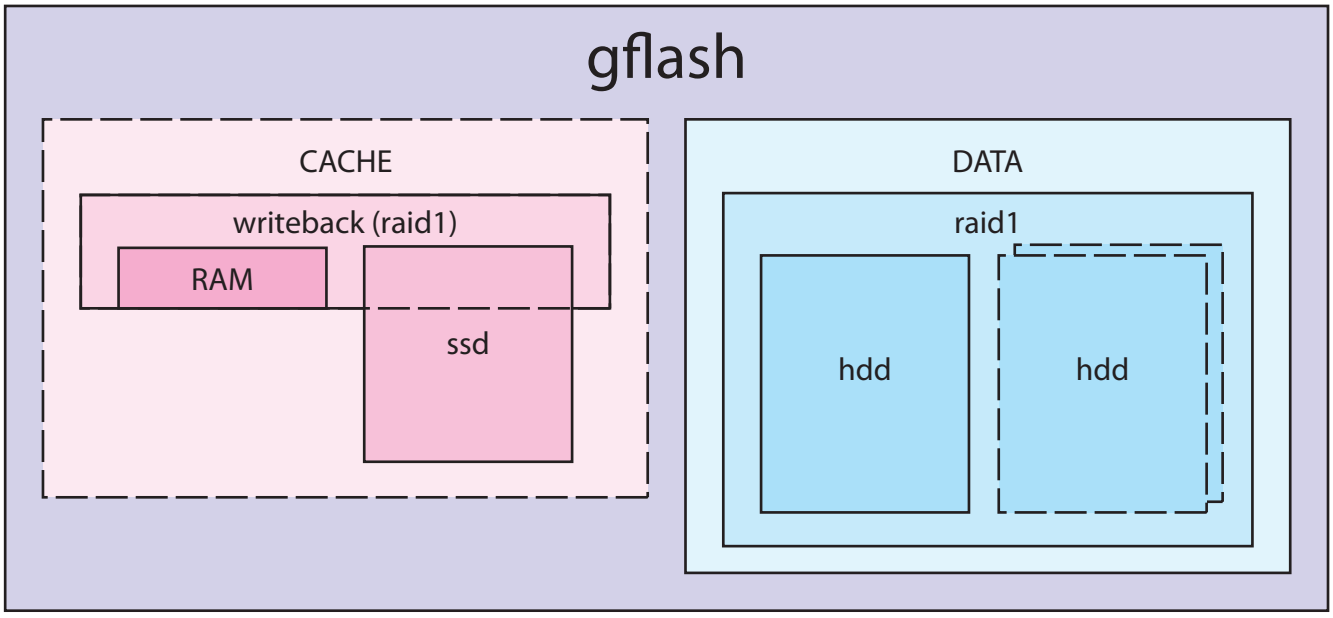
май 2015

Содержание

Общие сведения

gflash - класс **GEOM**, предназначенный для создания и обслуживания надежных, производительных и удобных в эксплуатации блочных устройств.

Структура **gflash**:



DATA - основное хранилище данных. Максимальный размер 16TB. Представляет из себя одно или несколько блочных устройств (обычно **hdd**), объединенных в **raid1**. Использование встроенного **raid1** обусловлено необходимостью тесной интеграции **DATA** и **CACHE.** Данная интеграция позволяет во многих случаях избежать процесса синхронизации при аварийном завершении работы. Также встроенный **raid1** во многих случаях имеет более высокую производительность чтения данных по сравнению с другими реализациями **raid1** (**gmirror, graid, gvinum**). Это связано с более производительным алгоритмом балансировки, реализацией возможности чтения с синхронизируемого компонента и отсутствием задержек, возникающих при включении дополнительного **GEOM-**а в цепочку ввода/вывода.

CACHE - объединение оперативной памяти (**RAM**) и блочного устройства с быстрым доступом (обычно **ssd**). Максимальный размер 4TB. Предназначен для увеличения производительности основного хранилища данных. Операции чтения данных кэшируются при повторном чтении. Операции записи данных кэшируются в режиме отложенной записи (**writeback**). Для обеспечения отказоустойчивости и повышения производительности **writeback** кэш организован в виде **raid1** из **RAM** и части блочного устройства с быстрым доступом. Размер **writeback** кэша может быть изменен в любое время.

RAM - оперативная память, выделенная для организации отказоустойчивого **writeback** кэша. Минимальный размер 128MB. Максимальный размер 16GB. Добавление/уменьшение происходит с шагом 128MB. При увеличении размера **RAM** в операционной системе всегда останется минимум 128MB свободной оперативной памяти. Данная защита предотвращает захват модулем **gflash** всей оперативной памяти в операционной системе.

Созданное **gflash** блочное устройство может работать в режиме **CACHE_OFF** (без кэша) и **CACHE_ON** (с кэшем). Переход из одного режима в другой может осуществляться в любое время.

Режим **CACHE_OFF** представляет из себя **raid1** и является аналогом **gmirror**. Данный режим в полном объеме поддерживает выполнение запросов **READ**, **WRITE**, **FLUSH**, **DELETE**.

Режим **CACHE_ON** представляет из себя **raid1**, дополненный кэшем для увеличения производительности. Данный режим в полном объеме поддерживает выполнение команд **READ**, **WRITE**, **FLUSH**. Внешняя команда **DELETE** не поддерживается, так как модуль **gflash** самостоятельно обслуживает кэш и все команды очистки флэш памяти формируются в зависимости от внутренних алгоритмов освобождения блоков кэша. В данном режиме отложенная запись из **CACHE** в **DATA** считается выполненной только после завершения внутренней команды **FLUSH**, поэтому использование в **DATA** дисков с включенной отложенной записью (**writeback**) не снижает надежность хранения данных. При аварийном завершении работы возможна потеря до четырех последних записей в блочное устройство вне зависимости от типа **ssd** используемого в **CACHE** (с защитой целостности данных при сбоях питания или без неё). Также возможна потеря данных, находящихся в кэше отложенной записи **ssd**, если он не оборудован защитой данных при сбоях питания. В случае планового выключения, **gflash** записывает все **DIRTY** блоки в **DATA**. Поэтому время завершения работы при плановом выключении может увеличиться. При старте блочного устройства выполняется обнаружение всех компонентов, проверяется целостность данных кэша, определяется корректность завершения работы и в случае аварийного завершения проверяется наличие **DIRTY** блоков для их последующей записи в **DATA**. Поэтому при старте может возникнуть временная задержка появления блочного устройства в системе. Во время старта, при обнаружении некоторых повреждений данных в **CACHE**, модуль **gflash** определит их критичность и попытается самостоятельно восстановить работоспособность блочного устройства. Для обслуживания **CACHE** требуется дополнительная оперативная память. Количество дополнительной памяти напрямую зависит от размера **CACHE** и косвенно от размера **DATA**. На каждый 4KB-блок **CACHE** приблизительно расходуется около 32-40 байт оперативной памяти. Столь значительное потребление оперативной памяти связано с реализацией кэширования по второму чтению и реализацией дополнительной возможности - журнала блочного устройства на базе **writeback** кэша. Данная возможность позволяет избежать запуска процесса синхронизации **raid1 DATA** при аварийном завершении работы. Общее потребление оперативной памяти будет равно размеру **RAM** плюс дополнительная память для обслуживания **CACHE**.

Блочное устройство обеспечивает работоспособность при выходе из строя любого компонента (кроме **RAM** и последнего синхронизированного компонента **DATA**). Также гарантируется сохранение работоспособности при добавлении/удалении любых компонентов (**RAM**, **ssd**, **hdd**) в любое время и при любой нагрузке. При возникновении ошибки ввода/вывода неисправный компонент автоматически удаляется из конфигурации блочного устройства. Все события связанные с изменением конфигурации блочного устройства, а также сообщения об ошибках заносятся в системный журнал. Также в системном журнале фиксируется состояние компонентов при старте/стопе блочного устройства. При старте блочного устройства для определения компонентов входящих в **DATA** установлен таймаут в 3 секунды, для **CACHE -** максимум 6 секунд. По истечении таймаута блочное устройство стартует, если состояние зарегистрировавшихся компонентов позволяет без потерь данных начать работу. Все запоздавшие компоненты будут добавлены в блочное устройство как новые (для **DATA** стартует процесс синхронизации, для **CACHE** произойдет полная очистка **ssd**). При старте всегда проверяется целостность данных, находящихся в **CACHE**. В некоторых случаях при частичном повреждении данных, возможна ликвидация повреждений и восстановление работоспособности **CACHE**.

Размер блока (сектора), созданного **gflash** блочного устройства, равен 4KB.

Установка

На **github.com/geomflash/geomflash** доступны два модуля **gflash**.

Модуль использующий стандартный механизм работы **GEOM**, подходит для **FreeBSD amd64** версий **9.x** - **10.x**.

```
su
tar -xPf gflash-14_FreeBSD-9.x-10.x-amd64.tar.gz
```

Более производительный модуль, использующий новые схемы прохождения запросов ввода-вывода, подходит для **FreeBSD amd64** начиная с версии **10.1**.

```
su
tar -xPf gflash-14_FreeBSD-10.1-amd64.tar.gz
```

После распаковки в операционную систему будет добавлено четыре файла:
`/boot/kernel/geom_flash.ko` - модуль ядра;
`/sbin/gflash` - программа администрирования;
`/lib/geom/geom_flash.so` - разделяемая библиотека;
`/usr/share/man/man8/gflash.8.gz` - справочное руководство.

При создании блочного устройства модуль **gflash** автоматически загружается в систему. Для ручной загрузки модуля используется команда:

`gflash load` или `kldload geom_flash`

Для автоматической загрузки модуля **gflash** при старте операционной системы необходимо в файл `/boot/loader.conf` добавить строку `geom_flash_load="YES"`.

Описание команд

Для обслуживания **gflash** используется предельно простой набор команд. Три команды конфигурирования (`create, insert, remove`), одна эксплуатационная команда (`stop`) и семь универсальных команд (`dump, clear, help, list, status, load, unload`). Системный интерфейс `sysctl` не используется.

Для описания команд введем обозначения:

`f0` - имя блочного устройства;
`ada1` - **ssd**, используется для создания **CACHE**;
`ada2 ada3` - **hdd**, используются для создания **DATA**.

`create`

Создаёт блочное устройство в `/dev/flash`.

```
gflash create [-r ram_size] [-c cache_prov] name data_prov ...
```

Примеры:

`gflash create f0 ada2` создает блочное устройство состоящее только из **DATA** (один **hdd**);

`gflash create f0 ada2 ada3` создает блочное устройство состоящее только из **DATA** (два **hdd**, объединенных в **raid1**);

`gflash create -c ada1 f0 ada2 ada3` создает блочное устройство состоящее из **DATA** (два **hdd**, объединенных в **raid1**) и **CACHE** (**ssd** и **RAM** по умолчанию 128MB);

`gflash create -r 4g -c ada1 f0 ada2 ada3` создает блочное устройство состоящее из **DATA** (два **hdd**, объединенных в **raid1**) и **CACHE** (**ssd** и **RAM** 4GB);

insert

Добавляет компонент в блочное устройство.

```
gflash insert [-r ram_size] [-c cache_prov] name data_prov
```

Примеры:

`gflash insert f0 ada3` добавляет **hdd** в **raid1 DATA**;

`gflash insert -c ada1 f0` добавляет **CACHE** (**ssd** и **RAM** по умолчанию 128MB);

`gflash insert -r 2g f0` увеличивает **RAM** на 2GB.

remove

Удаляет компонент из блочного устройства. Не позволяет удалить последний полностью синхронизированный (`Synchronized: 100%`) компонент **DATA**.

```
gflash remove [-r ram_size] [-c cache_prov] name data_prov
```

Примеры:

`gflash remove f0 ada3` удаляет **hdd** из **raid1 DATA**;

`gflash remove -c ada1 f0` удаляет **CACHE** блочного устройства, оперативная память возвращается в систему после записи всех **DIRTY** блоков.

ВНИМАНИЕ – во время записи **DIRTY** блоков возможна потеря данных при аварийном завершении работы, поэтому желательно удалять **CACHE** при отсутствии **DIRTY** блоков;

`gflash remove -r 256m f0` уменьшает **RAM** на 256MB, оперативная память возвращается в систему после записи всех **DIRTY** блоков. Данная операция безопасна.

stop

Останавливает работу блочного устройства.

```
gflash stop [-f] name ...
```

Примеры:

```
gflash stop f0
```

`gflash stop -f f0` форсированно останавливает занятое блочное устройство.

Возобновление работы блочного устройства возможно двумя способами.
Загрузить модуль **gflash**:

`gflash load` или `kldload geom_flash`

Если модуль **gflash** уже загружен и выполняет работу, то послать команду `true` одновременно всем компонентам блочного устройства:

```
true > /dev/ada1 & true > /dev/ada2 & true > /dev/ada3
```

dump, clear, help, list, status, load, unload

Универсальные команды для всех классов **GEOM**.

```
gflash dump prov ...
gflash clear prov ...
gflash help
gflash list [-a] [name ...]
gflash status [-ags] [name ...]
gflash load [-v]
gflash unload [-v]
```

Для удаления блочного устройства необходимо его остановить и затем очистить метаданные у каждого компонента:

```
gflash stop f0
gflash clear ada1 ada2 ada3
```

В вывод команды `list` включена статистика по использованию блочного устройства с момента старта и количество незаписанных (**DIRTY)** блоков.
Для каждого компонента (**Consumers**) блочного устройства отображается:
– состояние `State: START` | `STOP` | `FAIL` | `TASTE`;
– назначение `Role: DATA` | `CACHE`;
– дополнительная информация, в зависимости от назначения:
– для **DATA** `Synchronized:` от `0` до `100%`;
– для **CACHE** `RAM:` от `128` до `16384MB`

Рекомендации

Выбор размера writeback кэша

Размер **writeback** кэша определяется размером **RAM**. **Writeback** кэш может находиться в одном из режимов: режим заполнения или форсированный режим.

В режиме заполнения скорость записи приблизительно равна скорости записи в **ssd**. Отложенная запись производится при отсутствии чтения данных из **DATA**.

Форсированный режим возникает во время записи данных в блочное устройство после полного заполнения **writeback** кэша. В данном режиме скорость записи приблизительно равна скорости записи в **hdd**. Отложенная запись выполняется форсированно, что негативно сказывается на общей производительности блочного устройства.

Исходя из вышеизложенного, чем больше размер **writeback** кэша, тем дольше длится режим заполнения и соответственно выше производительность. При выполнении отложенной записи для повышения производительности оптимизируется порядок записи **DIRTY** блоков, поэтому чем больше размер **writeback** кэша, тем лучше работает оптимизация отложенной записи. Чем больше размер **RAM**, тем выше вероятность чтения данных из оперативной памяти, что также положительно сказывается на производительности блочного устройства.

Единственным побочным эффектом большого размера **writeback** кэша является возможное увеличение времени выключения системы и возможное увеличение времени старта блочного устройства после аварийного завершения работы. Изменять размер **writeback** кэша (**RAM**) можно безопасно в любое время.

Выбор ssd для CACHE

При выборе **ssd** следует обращать внимание на скорость последовательной записи (скорость случайной записи и скорость чтения не так критичны). Для сбалансированной производительности блочного устройства желательно, чтобы скорость последовательной записи **ssd** в два-три раза превосходила скорость последовательной записи **hdd**.

При работе с **ssd**, для поддержания высокой скорости записи во флэш память, модуль **gflash** использует внутреннюю команду очистки флэш памяти **DELETE** (**TRIM**). Поэтому при выборе **ssd** предпочтение необходимо отдавать моделям у которых применение данной команды действительно позволяет сохранять высокую скорость записи (например у **ssd,** построенных на контроллерах SandForce и использующих сжатие данных при записи, даже при выполнении команд очистки флэш памяти скорость записи падает). Также не следует для поддержания высокой скорости записи резервировать на **ssd** свободный объем флэш памяти (Over-Provisioning). **gflash** сам об этом позаботится.

Практически все **ssd** имеют внутренний **writeback** кэш, поэтому при аварийном завершении работы есть вероятность потери незаписанных данных. Если приложение (обычно файловая система) не умеет работать с блочными устройствами с **writeback** кэшем (то есть фиксировать команды записи командой **FLUSH**), то рекомендуется выбирать модели **ssd** с защитой целостности данных при сбоях питания (самый распространенный признак - наличие у **ssd** суперконденсатора для аварийного сохранения незаписанных данных).

Выбор размера **ssd** зависит от характера выполняемых задач. Обычно “чем больше, тем лучше”, но следует учитывать зависимость расхода дополнительной оперативной памяти на обслуживание кэша от размера **ssd**.

Использование в комбинированных RAID массивах

Созданное **gflash** блочное устройство может использовать в качестве компонентов любые составные блочные устройства, а также участвовать как внутренний компонент любого составного блочного устройства. Рассмотрим несколько возможных типичных применений.

Использование в **DATA** отдельного **raid1** (**gmirror, graid, gvinum**) вместо встроенного, рассмотренно ранее. Применение в **DATA** других типов **raid** (**raid10**, **raid5**, **raid6**) для повышения производительности требует некоторых пояснений. Современные массовые **ssd** имеют максимальную скорость чтения/записи около 500MB/s, а массовые **hdd** около 150MB/s, то есть разница в три-четыре раза. Даже в **raid1** из двух **hdd** возможно получить максимальную скорость чтения данных в два потока около 300MB/s. Если учесть, что при кэшировании кроме данных записываются еще метаданные, а также может происходить одновременное чтение данных из кэша, то производительность **ssd** может оказаться недостаточной. В случае применения в **DATA** еще более производительных **raid-**массивов, может получиться несбалансированная система. Поэтому для построения производительных составных блочных устройств предлагается объединять созданные **gflash** блочные устройства в **raid0**. Получается аналог **raid10**, с сопоставимой надежностью и лучшими эксплуатационными свойствами, за счет меньшего количества синхронизаций при аварийных завершениях работы. Для получения максимальной производительности рекомендуется выбирать **stripesize** равным 512KB (4 x **MAXPHYS**).

Пример:
`ada0 ada1` - **ssd**; `ada2 ada3 ada4 ada5` - **hdd**;

```
gflash create -r 2g -c ada0 f0 ada2 ada3
gflash create -r 2g -c ada1 f1 ada4 ada5
gstripe create -s 512k s0 flash/f0 flash/f1
```

Существует несколько вариантов потери данных при двойной аварии:

– вышел со строя **ssd**, в **RAM** находились **DIRTY** блоки и во время их записи в **DATA** произошло аварийное завершение работы;

– произошло аварийное завершение работы, в момент аварии в **RAM** находились **DIRTY** блоки, во время следующего старта блочного устройства **ssd** оказался неисправным.

Для защиты от подобных аварий возможно использование в **CACHE** вместо одиночного **ssd** несколько **ssd,** объединенных в **raid1**. Для высокой производительности кэша необходима поддержка команды **DELETE** (**TRIM**) в **raid1**. Рекомендуется использовать **gflash** в качестве **raid1** для **CACHE**.

Пример:
`ada0, ada1` – **ssd**; `ada2, ada3` – **hdd**;

```
gflash create cache0 ada0 ada1
gflash create -r 4g -c flash/cache0 f0 ada2 ada3
```

Недостатки данного построения:

– увеличение стоимости блочного устройства;

– после аварийного завершения работы, в **raid1** кэша всегда будет запускаться процесс синхронизации. Во время синхронизации производительность кэша резко снижается. После завершения процесса синхронизации один **ssd** будет полностью перезаписан, что на некоторое время отрицательно скажется на его скорости записи и соответственно производительности всего кэша;

– введение дополнительного **GEOM**-а, реализующего **raid1** для кэша увеличивает временные задержки в работе кэша;

– при старте, в случае отсутствия некоторых компонентов, любой **raid1** имеет предельное время ожидания (**timeout**). Поэтому в некоторых случаях возможно значительное несовпадение времени старта массивов **raid1 CACHE** и **DATA**, что может привести к старту блочного устройства без **CACHE**. Это не повлечёт за собой потерю данных, но при последующем обнаружении **CACHE** будет очищен и заново добавлен к блочному устройству.

Настройка файловых систем и приложений

Для настройки необходимо учитывать свойства **gflash** в режиме **CACHE_ON**:
– высокая скорость записи;
– реализация функции блочного журнала;
– реализация кэширования повторного чтения;
– возможная временная задержка при старте блочного устройства в случае аварийного завершения работы.

Очень эффективно использование **gflash** для создания экспортируемых блочных устройств (например по **iscsi**).

Настройка UPS

При настройке времени выключения **UPS** после подачи команды **shutdown** следует учитывать дополнительное время, которое может понадобится для записи **DIRTY** блоков. Время записи зависит от количества **DIRTY** блоков (максимальное количество определяется размером **RAM**) и характером распределения незаписанных блоков (случайное, последовательное, смешанное), поэтому однозначно определить временной интервал записи невозможно. Предлагается добавлять 5-10 мин, если это возможно, к времени выключения **UPS** после подачи команды **shutdown**. В случае, если выключение **UPS** произойдет раньше окончания записи всех **DIRTY** блоков, потеря данных не произойдет, но при следующем старте блочного устройства понадобится дополнительное время для повторной записи всех **DIRTY** блоков.

Восстановление после двойной аварии

После аварийного завершения работы операционной системы и одновременного выхода из строя **ssd**, входящего в **CACHE**, возникает ситуация неопределенности. При старте блочного устройства неизвестно все ли **DIRTY** блоки были записаны в **DATA** в момент аварийного выключения. Поэтому старт провайдера блочного устройства невозможен. Если не удаётся восстановить работоспособность **ssd**, возникает задача получения доступа к данным и определения их корректности. Для примера предположим, что в **DATA** входили `ada2` и `ada3` и в данный момент эти диски доступны в операционной системе. Имя поврежденного блочного устройства `f0`. Проверить состав и состояние блочного устройства можно командой `gflash list f0`. В нашем случае в блочном устройстве будет отсутствовать провайдер и присутствовать два компонента (**Consumers**) `ada2` и `ada3`. Для удаления компонетов необходимо остановить блочное устройство командой `gflash stop f0` и очистить метаданные командой `gflash clear ada2 ada3`. Заново создать блочное устройство из полностью синхронизированного компонента, допустим `gflash create f0 ada2`. Проверить корректность данных. Если данные повреждены, то удалить блочное устройство и создать для проверки блочное устройство из второго компонента. Все манипуляции с очисткой и записью метаданных не влияют на данные находящиеся в блочном устройстве, также гарантируется, что при старте после аварийного выключения данные находящиеся в блочном устройстве не изменяются.

Перенос больших объёмов данных

При переносе большого объема данных на вновь создаваемое **gflash** блочное устройство можно вначале не включать **CACHE** в конфигурацию. Перенести все данные и затем добавить **CACHE**. Этими действиями можно избежать многократной и бесполезной перезаписи **ssd**.

Зеркалирование разделов

При создании блочного устройства не рекомендуется использовать в **DATA** дисковые разделы как компоненты **raid1**. Лучше использовать в **DATA** диски без разметки, а затем создавать необходимые разделы на блочном устройстве **gflash**. Данная рекомендация позволит сохранить высокую производительность алгоритма балансировки, применяемого в **raid1 DATA**.

Сообщения в системном журнале

Все события необходимые для анализа работы блочного устройства, созданного **gflash**, фиксируются в системном журнале.

Формат сообщения:
– **GEOM_FLASH** - признак принадлежности сообщения блочному устройству **gflash**;
– имя блочного устройства;
– описание события.
Примеры команд просмотра только сообщений блочных устройств **gflash**:

`dmesg | grep GEOM_FLASH` - отображение всех сообщений всех блочных устройств **gflash**;
`dmesg | grep 'GEOM_FLASH f0'` - отображение всех сообщений блочного устройства `f0`.

Обозначения применяемые в сообщениях:
– `Role` - показывает назначение компонента в блочном устройстве, возможные варианты:
  – `DATA`;
  – `CACHE`;
– `State` - показывает состояние устройства, возможные варианты:
  – `TASTE` - устройство находится в режиме самоопределения;
  – `START` - активно, при старте указывает на аварийное завершение работы;
  – `STOP` - остановлено, при старте указывает на корректное завершение работы;
  – `FAIL` - компонент **DATA** ранее выполнял запросы **READ**/**WRITE** с ошибками, но не был удален из блочного устройства, потому что на момент возникновения ошибки он был единственным полностью синхронизированным компонентом, данное состояние не изменяется при старте/стопе;
– `Sync` - показывает в процентах количество синхронизированных данных;
– `CACHE` - показывает режим работы блочного устройства, возможные варианты:
  – `ON` - кэш включен;
  – `OFF` - кэш выключен;
– `Lsnw` - показывает значение глобального счетчика выполненных операций записи с момента создания блочного устройства, используется для определения компонентов, содержащих устаревшие данные.

Примеры сообщений:

– успешный старт блочного устройства `f0`:

```
GEOM_FLASH f0: start ada2:[Role=DATA Lsnw=2015 State=STOP Sync=100% CACHE=ON]
GEOM_FLASH f0: start ada3:[Role=DATA Lsnw=2015 State=STOP Sync=100% CACHE=ON]
GEOM_FLASH f0: start ada1:[Role=CACHE Lsnw=2015 RAM=128M DIRTY=NO]
GEOM_FLASH f0: start provider
```

– удаление диска `ada2` из **raid1 DATA** блочного устройства `f0`:

```
GEOM_FLASH f0: remove ada2
```

– добавление диска `ada2` в **raid1 DATA** блочного устройства `f0`:

```
GEOM_FLASH f0: start synchronization ada2:[Sync=0%]
GEOM_FLASH f0: insert ada2:[Role=DATA Lsnw=2015 State=START Sync=0% CACHE=ON]
GEOM_FLASH f0: stop synchronization ada2:[Sync=100%]
```

– удаление **CACHE** из блочного устройства `f0`:

```
GEOM_FLASH f0: remove ada1
```

– добавление **CACHE** в блочное устройство `f0`:

```
GEOM_FLASH f0: start cleaning ada1
GEOM_FLASH f0: finish cleaning ada1
GEOM_FLASH f0: insert ada1:[Role=CACHE Lsnw=2015 RAM=512M DIRTY=NO]
```

– добавление 128MB в **RAM** блочного устройства `f0`:

```
GEOM_FLASH f0: start inserting 128MB RAM:[512MB]
GEOM_FLASH f0: finish inserting RAM:[640MB]
```

– уменьшение на 256MB **RAM** блочного устройства `f0`:

```
GEOM_FLASH f0: start removing 256MB RAM:[640MB]
GEOM_FLASH f0: finish removing RAM:[384MB]
```

– ошибка чтения с диска `ada2` и его последующее удаление из блочного устройства `f0`:

```
GEOM_FLASH f0: request READ failed ada2:[Error=5 Offset=1048576 Length=4096]
GEOM_FLASH f0: remove ada2
```

– остановка по команде `stop` блочного устройства `f0`:

```
GEOM_FLASH f0: stop provider
GEOM_FLASH f0: stop ada3:[Role=DATA Lsnw=2015 State=STOP Sync=100% CACHE=ON]
GEOM_FLASH f0: stop ada1:[Role=CACHE Lsnw=2015 RAM=128M DIRTY=NO]
```

При завершении работы операционной системы по команде **shutdown** системный журнал прекращает свою работу до начала завершения работы блочных устройств **gflash**. Поэтому сообщения об остановке не заносятся в системный журнал, а отображаются только на консоли.