УДК 004.451.51

РЕАЛИЗАЦИЯ ТРАНСЛЯТОРА RAID-5 ДЛЯ РАСПРЕДЕЛЕННОЙ ФАЙЛОВОЙ СИСТЕМЫ GLUSTERFS[1]

А.С. Игумнов, А.Ю. Берсенев

Статья посвящена реализации алгоритма RAID-5 в распределенной файловой системе GlusterFS. Анализ требований предъявляемых к масштабируемой файловой системе (ФС), способной задействовать в дисковые ресурсы узлов вычислительного кластера, показывает, что реализация распределенной версии алгоритма RAID-5 позволяет существенно повысить устойчивость ФС к сбоям отдельных узлов и даже стоек кластера. В статье дается краткий обзор принципов функционирования распределенной файловой системы GlusterFS и описывается способ встраивания алгоритма RAID-5 в эту систему. Описываются основные алгоритмы и структуры данных, реализованные для адаптации RAID-5 в распределенную ФС. Делаются выводы об устойчивости и производительности разработанной ФС. Показано, что реализованный алгоритм позволяет наращивать пропускную способность ФС до пропускной способности нижележащей сетевой системы, незначительно теряя в производительности при наличии отказавших узлов.

Ключевые слова: распределенная файловая система, отказоустойчивость, RAID-5, GlusterFS.

Введение

Развитие вычислительных кластеров современного типа начиналось в 90-х годах прошлого века с объединения относительно недорогих персональных компьютеров с помощью ставшей в те годы коммерчески доступной сети 100 Мбит/с Ethernet. Для хранения данных в состав кластера включался дополнительный модуль, игравший роль файлового сервера, как правило, поддерживающего протокол NFS. Повышение производительности вычислительных узлов и увеличение их количества в составе единого суперкомпьютера сделали классический файловый сервер узким местом в архитектуре кластера.

Техническим решением этой проблемы стало увеличение пропускной способности каналов связывающих вычислительные узлы и файловый сервер (40 Гбит/с Infiniband или связки из двух-четырех каналов 1 Гбит/с Ethernet), а так же использование RAID для преодоления ограничения на пропускную способность единичного диска.

Алгоритмическим решением проблемы стало распараллеливание подсистемы ввода/вывода (В/В) — частичное (использование нескольких узлов В/В, подключенных к одному многоканальному RAID контроллеру), тотальное (использование множества относительно недорогих узлов В/В, оборудованных стандартными дисками), а также различные промежуточные варианты.

К сожалению, ни российский список TOP50 [1], ни международный TOP500 [2] не предоставляют информацию о подсистеме В/В суперкомпьютеров. На основе изучения публикаций можно сделать вывод, что компьютеры с максимальной производительностью, такие как кластер «Ломоносов» (Московский государственный университет имени

[1] Статья рекомендована к публикации программным комитетом Международной суперкомпьютерной конференции «Научный сервис в сети Интернет: поиск новых решений – 2012».

М.В. Ломоносова) используют бездисковые вычислительные узлы и высокопроизводительные системы хранения, объединенные с помощью параллельной файловой системы (ФС) Lustre. В то же время, кластеры меньшего масштаба (кластер «Уран», Екатеринбург, Институт математики и механики УрО РАН) используют вычислительные узлы, снабженные локальными дисками, и высокопроизводительные RAID системы, доступные на узлах по NFS.

Данная статья посвящена разработке распределенной высокопроизводительной и надежной ФС, способной задействовать локальные диски узлов кластера. Разработанная система расширяет реализацию известной ФС с массовым параллелизмом — GlusterFS [3].

При разработке ФС ставились следующие задачи:

- возможность задействовать для хранения данных диски на узлах кластера, участвующих в вычислениях или выделенных;
- обеспечение целостности данных при выходе из строя единичного узла хранения или нескольких узлов из заранее известного домена отказа;
- масштабируемость пропускной способности системы В/В до скорости канала связи соединяющего узлы;
- возможность работы с большими наборами данных, существенно превышающих по объему емкость одного локального диска;
- минимальные накладные расходы на хранение служебной информации ФС.

Для реализации этих задач было принято решение реализовать в распределенной ФС алгоритм RAID-5, используя транспортный уровень и вспомогательные библиотеки ФС GlusterFS. Отдельной целью разработки стала оценка стабильности существующей версии GlusterFS и оценка гибкости, заложенных в нее механизмов расширения.

Поскольку в терминологии GlusterFS отдельные модули, расширяющие функциональность называются «трансляторами» в дальнейшем мы будем обозначать нашу ФС как RAID-5 Translator (R5T).

1. Описание структуры GlusterFS

GlusterFS – модульная распределенная сетевая ФС. Ее отличительной особенностью является отсутствие выделенного сервера метаданных, что позволяет осуществлять практически линейное масштабирование ФС на однородных узлах В/В. Первая версия GlusterFS вышла в 2006 году. В конце 2011 года GlusterFS была приобретена компанией Red Hat Inc. и в настоящий момент находится в стадии активной разработки и оптимизации.

И серверная и клиентская часть GlusterFS работают в пространстве пользователя через интерфейс FUSE (Filesystem in Userspace). Это облегчает разработку и отладку ФС, а так же позволяет защитить ядро ОС от ошибок в реализации драйверов ФС.

Ключевым элементом архитектуры GlusterFS являются так называемые «трансляторы» – программные модули, обрабатывающие и модифицирующие запросы на файловые операции. По своему функциональному назначению трансляторы делятся на следующие группы:

- трансляторы хранения данных, которые отвечают за взаимодействие с базовой ФС на узлах В/В;
- трансляторы передачи данных – сетевое взаимодействие;

- трансляторы, повышающие производительность – кэширование;
- трансляторы, добавляющие функциональность блокировки, права доступа;
- трансляторы кластеризации – наиболее важный класс трансляторов, реализующий алгоритмы распределения данных по узлам В/В.

Трансляторы в GlusterFS могут объединяться в сложную древовидную структуру. Когда GlusterFS получает вызов от файловой системы, он передает его транслятору, находящемуся в корне дерева трансляторов. Корневой транслятор, в свою очередь, передает вызов дальше подмножеству своих трансляторов-детей, те – своим детям и т.д. Результат вызова передается в обратном порядке – от листьев к корню, и затем – к приложению. На рис. 1 изображен пример объединения нескольких трансляторов, обеспечивающих локальное кэширование (io-cache, read-ahead), распределенную по сетевым узлам систему хранения файлов (unify) и взаимодействие с ФС на узлах В/В (POSIX).

2. Реализация транслятора RAID5 (R5T)

GlusterFS оперирует данными не на уровне отдельных блоков, а на уровне файлов. Для того, чтобы реализовать алгоритмы из RAID 5 каждый файл будем считать состоящим из блоков определенной длины. Блоки разных файлов независимы друг от друга. Размер блока задается при конфигурировании тома. По умолчанию он равен 128 КБ. Разбиение на блоки служит для распределения данных и контрольных сумм по узлам В/В и не накладывает на минимальный и максимальный размер операций В/В. Размещение блоков проще всего показать на примере RAID-5, собранного на трех узлах. Обозначим Дn – n-ный блок данных, а КС(i-k) — блок контрольной суммы, вычисленный на основе блоков Дi...Дk с помощью побайтовой операции XOR. Структура размещения фрагментов файлов будет следующая:

Первый узел: Д1, Д3, К(5-6)...
Второй узел: Д2, К(3-4), Д5...
Третий узел: К(1-2), Д4, Д6...

При выполнении операции чтения R5T вычисляет положение нужного блока на основе смещения искомого фрагмента от начала файла и отправляет через сетевой транслятор запрос на нужный узел. Контрольные суммы при операции чтения не используются.

При выполнении операции записи, R5T вычисляет положение нужного блока и положение соответствующей ему контрольной суммы, считывает их, производит вычисление новой контрольной суммы, выполняет две операции записи.

Операции работы с метаданными, такие как, chmod, mv и rm, будут выполняться на каждом узле В/В без изменений.

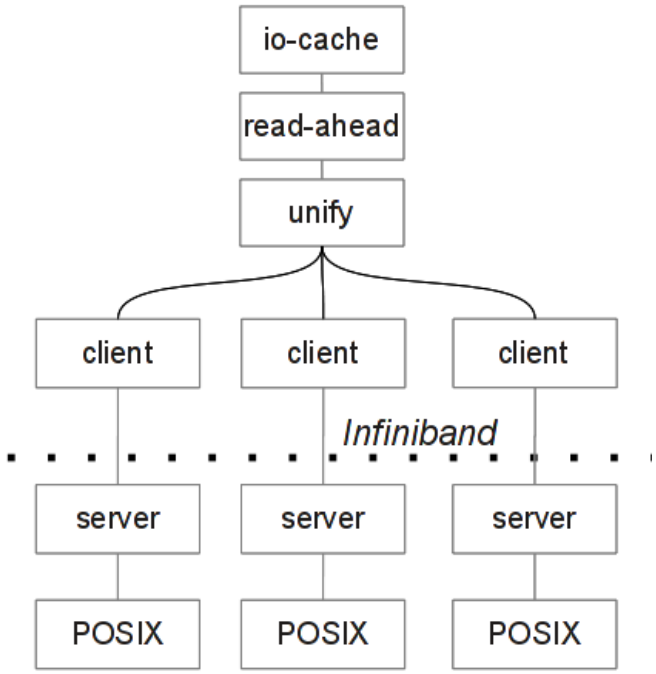

Рис. 1. Объединение трансляторов в GlusterFS

Хранение метаданных

В соответствии с идеологией GlusterFS все метаданные файла хранятся либо в метаданных фрагментов файла на узлах хранения, либо в расширенных атрибутах этих фрагментов. В случае с R5T все стандартные метаданные (uid, gid, права доступа и др.) хранятся в стандартных метаданных фрагментов файла, а в расширенных атрибутах хранятся метаданные, описывающие структуру RAID-5, а также метаданные, необходимые для восстановления после сбоя. Структура RAID-5 задается общим количеством узлов, а так же номером узла на котором размещается данный фрагмент файла. Для целей восстановления в расширенных атрибутах хранятся номер версии файла и номер версии метаданных файла. Если фрагмент файла восстанавливается после сбоя, то в его атрибутах дополнительно хранятся следующие величины: размер файла на момент начала восстановления, смещение от начала файла уже восстановленной области, номер версии файла, до которой производится восстановление (см. раздел 3, подраздел «Восстановление после сбоев»).

Отдельно стоит упомянуть про хранение размера файла. С одной стороны, общий размер хранимых в R5T данных превышает фактический размер файла за счет добавления контрольных сумм. С другой стороны, фрагмент файла, размещаемый на одном узле, существенно меньше файла в целом. Тем не менее, размер каждого из фрагментов можно сделать в точности равным размеру файла за счет использования «разреженных» (sparce) файлов.

Основные алгоритмы транслятора

Рассмотрим алгоритм размещения данных по узлам на примере вновь созданного файла нулевой длинны, размещенного на томе R5T, состоящем из N узлов В/В.

При записи первого байта файла:
на узле У(1) действительно записывается один байт данных,
на узлах У(2)-У(N-1) выполняется вызов seek, смещающий позицию записи, без выделения реального дискового пространства;
на узле У(N) выполняется запись КС (0 xor У(1)).

После заполнения 1го блока хранения порядок выполнения операций меняется:
У(1) — seek; У(2) — запись; У(3)-У(N-1) — seek;
на узле У(N) выполняется запись КС (У(1) xor У(2)), после чего выполняется seek, для придания файлу нужного размера.

Таким образом, размер фрагментов файла на всех узлах становится равным текущему размеру файла.

Алгоритмы транслятора, реализующие основные операции над файлами представлены на рис. 2–5.

1. Проверить, что недоступно не более одного узла.
2. Скорректировать флаги создания файла:
2.1. Убрать флаг O_APPEND, поскольку при использовании этого флага нельзя перемещаться по файлу с помощью вызова seek, а это необходимо для записи контрольной суммы.
2.2. Если установлен O_WRONLY, изменить его на O_RDWR, поскольку для обновления контрольной суммы, необходимо иметь возможность читать файл.
3. Передать вызов создания файла первому доступному подтому, указывая дополнительные расширенные атрибуты при создании: stripe-size, stripe-count, stripe-index, real-size и bad-node-index.
4. Если операция создания файла на первом подтоме завершилась неудачно, то удалить созданный фрагмент файла (если он был создан). Если удачно – передать вызов остальным дочерним трансляторам.
5. Установить файловый контекст и вернуть агрегированный результат работы.

Рис. 2. Алгоритм создания файла

1. Проверить, что недоступно не более одного узла.
2. Скорректировать флаги открытия файла:
2.1. Убрать флаг O_APPEND, т.к. при использовании этого флага нельзя перемещаться по файлу с помощью вызова seek, а это необходимо для записи контрольной суммы.
2.2. Если установлен O_WRONLY, изменить его на O_RDWR, потому что для обновления контрольной суммы необходимо иметь возможность читать файл.
3. Передать вызов открытия файла доступным узлам В/В.
4. Установить файловый контекст и вернуть агрегированный результат работы.

Рис. 3. Алгоритм открытия файла

1. Проверить, что недоступно не более одного узла.
2. Получить файловый контекст.
3. Если в процессе какой-либо записи не был доступен один из узлов, а сейчас недоступен другой, то читать файл нельзя, вернуть ошибку.
4. Вычислить первый и последний блоки, которые необходимо прочитать.
5. От первого до последнего блока, исключая блоки с контрольной суммой, сформировать запросы на чтение данных на узлах В/В. Если узел, с которого пытаемся читать недоступен, сформировать запросы на чтение ко всем другим узлам, чтобы восстановить информацию, которая была на недоступном узле.

Рис. 4. Алгоритм чтения из файла

1. Проверить, что недоступно не более одного узла.
2. Прочитать данные, которые находятся на месте записываемых.
3. Вычислить значение XOR между прочитанными данными и записываемыми.
4. Для каждой группы блоков(группа блоков — группа, имеющая общий блок с контрольной суммой) сформировать и выполнить запрос на запись блоков с данными, затем прочитать данные из блоков с контрольной суммой и выполнить операцию XOR с соответствующей частью данных, полученными на шаге 3.
5. Записать блоки с контрольной суммой.
6. Обновить реальный размер файла и данные файлового контекста, вернуть агрегированный результат работы. Если один из узлов был недоступен при записи, внести отметку об этом в расширенные атрибута файла на каждом доступном узле.

Рис. 5. Алгоритм записи в файл

Домены отказов

Узлы В/В GlusterFS размещаются на различных узлах кластера. Иногда возникают ситуации, которые приводят к недоступности целой группы узлов, находящихся в одном домене отказа, например, узлов, находящихся в одном здании, в одной серверной стойке или в сдвоенных узлах, в которых ремонт одного сервера требует отключения второго. R5T позволяет учитывать эту особенность. Тома рекомендуется создавать из узлов, находящихся в разных доменах отказа. Затем эти тома можно объединить в один, используя транслятор cluster/unify. На рис. 6 изображена файловая система, которая сохраняет работоспособность даже в случае отключения всех узлов в одном домене отказа. В этом случае пользователю доступно 2/3 от суммарной емкости дисков на узлах.

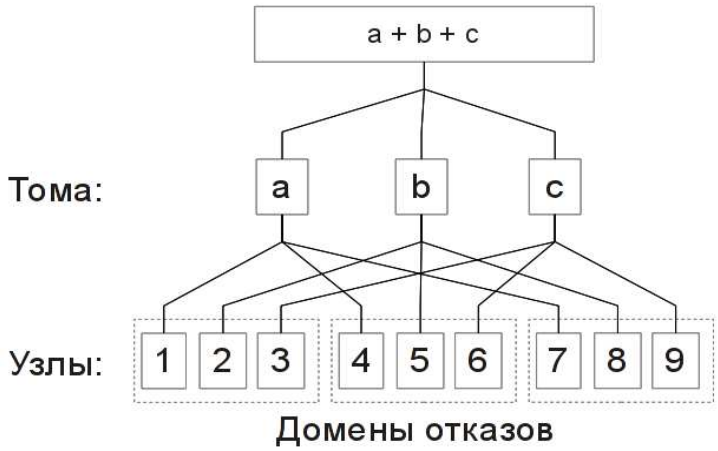

Рис. 6. Распределение узлов хранения по доменам отказов

3. Оценка производительности ФС

На операциях чтения транслятор R5T обеспечивает максимально возможную производительность. Такую же производительность показывает входящий в стандартную поставку GlusterFS транслятор cluster/stripe, но он не обеспечивает устойчивости к сбоям. Один клиент, читающий порциями меньше размера блока, будет ограничен пропускной способностью диска на отдельном узле. При параллельном чтении с нескольких клиентов, а так же при чтении большими порциями теоретическая пропускная способность R5T возрастает прямо пропорционально числу узлов В/В и ограничивается пропускной способностью использованной сети.

На операциях записи происходит двух-четырехкратное падение скорости по сравнению с теоретически возможным максимумом (достигается в трансляторе cluster/stripe). Падение, с одной стороны, определяется необходимостью выполнять операцию чтения перед каждой записью, а с другой стороны, при приближении к пределу пропускной способности сети, еще и необходимостью выполнения двух операции записи — данных и контрольной суммы. При увеличении количества узлов, теоретическая пропускная способность R5T растет и ограничивается сверху половиной пропускной способностью использованной сети.

Производительности ФС при сбоях

На операциях записи, а так же на операциях, манипулирующих метаданными, производительность ФС не изменяется. Все вычисления и обмены производятся в том же порядке и в тех же объемах, что и до появления сбоя кроме операции передачи данных на сбойный узел. Отсутствие передачи данных на сбойный узел может даже немного ускорить процесс записи, но при достаточно большом количестве узлов этот прирост станет практически незаметным.

Операция чтения замедлится за счет необходимости восстанавливать недостающие данные, хранящиеся на сбойном узле, по данным, хранящимся на остальных узлах В/В.

Рассмотрим набор из N узлов и операцию последовательного чтения блоками, соответствующими по размеру единичному блоку хранения. В такой ситуации $N-2$ операций чтения (пропускаем сбойный узел и контрольную сумму) пройдут в штатном режиме, а

N-ная операция потребует повторного считывания N−2 блоков, одного блока с контрольной суммой и вычисления результата XOR по всем полученным данным. Если учесть, что время выполнения XOR пренебрежимо мало по сравнению со скоростью передачи данных, то можно считать, что временные затраты на чтение N блоков будут равны 2N−3 единичных операций чтения. То есть, при достаточно большом числе узлов В/В мы всегда получаем фиксированное двукратное снижение производительности.

Данный алгоритм можно улучшить, разработав специализированный кэш, отслеживающий порядок чтения блоков. В этом случае, значение N-ого блока рассчитывается по мере считывания предыдущих N−1 блоков. Если данные из файла читались последовательно и файл не был изменен, то восстановление данных сбойного узла будет происходить практически без накладных расходов.

Существует «патологический» сценарий чтения из файла — небольшими фрагментами, размещенными исключительно на сбойном узле. В этом случае каждое считывание приводит к необходимости чтения данных со всех узлов, что может привести к падению скорости чтения прямо пропорциональному количеству задействованных узлов.

Восстановление данных после сбоя

С каждым фалом в R5T связан номер версии, который увеличивается при каждой записи в файл. В случае временной недоступности одного из узлов В/В и последующего его включения возможны две ситуации: файл не был изменен за время недоступности узла и его номер версии одинаков на всех узлах; номер версии на восстанавливаемом узле меньше, чем на остальных и, соответственно, данные хранящиеся на узле не соответствуют актуальному состоянию файла.

В первом случае узел В/В может быть задействован немедленно, без каких-либо вспомогательных операций. Во втором случае в фоновом режиме запускается процесс восстановления, заключающийся в считывании данных файла со всех узлов R5T, вычисление данных для восстанавливаемого узла и обновление их на диске.

Следует учесть, что запись в R5T после сбоя автоматически восстанавливает корректную структуру данных. Поэтому в процессе восстановления узел открывается для записи, но без обновления номера версии. В метаданные узла заносятся три величины: размер файла в момент начала восстановления, смещение от начала файла уже восстановленной области, номер версии файла, до которой производится восстановление. Первая величина позволяет обнаружить запись в конец файла новых — корректных данных, а две другие величины позволяют продолжить процедуру восстановления в случае повторного сбоя или перезапуска R5T. Каждая запись в файл обновляет номер версии, до которого идет восстановление. Если операция записи пересекается с уже восстановленной областью, то граница восстановленной области смещается до границы операции записи.

4. Оценка устойчивости GlusterFS

Несмотря на активную поддержку со стороны компании RedHat и довольно продолжительные сроки разработки, ФС GlusterFS содержит заметное количество ошибок и не до конца устоявшийся API.

В процессе тестирования ФС R5T, в базовых функциях GlusterFS была выявлена ошибка, которая, при определенном сочетании длинны запроса на запись и размера блока хранения, приводила к потере данных. Отчет об обнаруженной ошибке был отправлен

основной команде разработчиков. По результатам отчета в код GlusterFS были внесены изменения, устраняющие целый класс потенциальных ошибок. Было выявлено несколько утечек памяти. Исправляющий ошибки патч внесен в основную ветвь GlusterFS. Во время работы над R5T в API GlusterFS было внесено как минимум одно серьезное изменение, потребовавшее правки всего написанного кода, — добавлен новый параметр в целый ряд основных функций.

Несмотря на обнаруженные ошибки, активная работа, ведущаяся по совершенствованию GlusterFS фирмой RedHat, позволяет предположить, что эта ФС вскоре станет стабильной платформой, на которой смогут базироваться разнообразные распределенные ФС.

Заключение

Разработана масштабируемая файловая система с устойчивостью к одиночным сбоям узлов хранения данных. Показано, что на основе ФС Glusterfs возможно создание новых типов ФС со структурами данных, разнесенными по нескольким узлам хранения. Выявлена некоторая неустойчивость существующей версии ФС Glusterfs, которая ограничивает ее применение в полномасштабных промышленных приложениях.

Исходные тексты R5T доступны на сервере GitHub [4]

Работа выполнена в рамках программы Президиума РАН № 18 «Алгоритмы и математическое обеспечение для вычислительных систем сверхвысокой производительности» при поддержке УрО РАН (проект 12-П-1-1034).

Литература

1. Суперкомпьютеры. TOP50
   URL: http://top50.supercomputers.ru (дата обращения: 24.12.2012)
2. The Top500 List
   URL: http://www.top500.org (дата обращения: 24.12.2012)
3. Babu A. Gluster – «The GNU Cluster Distribution»
   URL: https://github.com/gluster/historic (дата обращения: 24.12.2012)
4. Исходные тексты R5T
   URL: https://github.com/alexbers/glusterfs_experiments/ (дата обращения: 24.12.2012)

Игумнов Александр Станиславович, Федеральное государственное бюджетное учреждение науки Институт математики и механики им. Н.Н.Красовского Уральского отделения Российской академии наук, igumnov@imm.uran.ru

Берсенев Александр Юрьевич, Федеральное государственное бюджетное учреждение науки Институт математики и механики им. Н.Н.Красовского Уральского отделения Российской академии наук, bay@hackerdom.ru

IMPLEMENTATION OF RAID-5 TRANSLATOR FOR GLUSTERFS DISTRIBUTED FILE SYSTEM

A.S. Igumnov, IMM UB RAS (Yekaterinburg, Russian Federation),
A.Yu. Bersenev, IMM UB RAS (Yekaterinburg, Russian Federation)

The article is devoted to the implementation of the RAID-5 algorithm as a part GlusterFS distributed file system. The article provides a brief overview of the principles of functioning of GlusterFS and describes how to embed RAID-5 into this file system. The article describes data structures and algorithms of RAID-5 translator. The conclusions of the stability and performance of translator are made. The article shows that implemented RAID-5 algorithm allows to achieve file system resiliency and to increase the throughput capacity of the file system. It is shown that damage of some storage nodes does not affect on write speed and slightly affect read speed of the translator.

Keywords: distributed file system, file system resiliency, RAID-5, GlusterFS.

References

1. Supercomputers. TOP50
   URL http://top50.supercomputers.ru (accessed: 24.12.2012)
2. The Top500 List
   URL http://www.top500.org (accessed: 24.12.2012)
3. Babu A. Gluster – «The GNU Cluster Distribution»
   URL https://github.com/gluster/historic (accessed: 24.12.2012)
4. R5T source code
   URL https://github.com/alexbers/glusterfs (accessed: 24.12.2012)

Поступила в редакцию 9 января 2013 г.