Пространство пользователя и файловые системы в пространстве пользователя

Изай Д. Ю., Алексеев Н. А.

Национальный технический университет Украины «Киевский политехнический институт»
Институт телекоммуникационных систем
пер. Индустриальный, 2, Киев, 03056, Украина

dimmu4@mail.ru, alexeyev@its.kpi.ua

Аннотация. *Рассмотрены особенности работы в пространстве пользователя (User-Space), рассмотрена концепция работы FUSE (Filesystem in Userspace), приведены особенности построения и работы файловых систем в пространстве пользователя, на примере показано построение простейшей файловой системы, работающей в пространстве пользователя и обращение к ней, определены преимущества и недостатки данной технологии. Также на примере GlusterFS и Hadoop Distributed File System (HDFS) проведен сравнительный анализ распределенных файловых систем - построенной на FUSE и не использующей данную технологию.*

Ключевые слова

Файловая система, пространство пользователя, FUSE (Filesystem in Userspace).

1 Введение

Для большинства крупных предприятий актуальны проблемы хранения большого количества информации. Зачастую объемы хранимой информации и требования к организации хранилища не позволяют использовать стандартные файловые системы (ФС) или облачные технологии. Одним из подходов к построению файлового хранилища является использование модуля ядра семейства операционных систем UNIX, FUSE (Filesystem in Userspace).

Технической документации, позволяющей судить об эффективности использования FUSE в зависимости от различных условий не опубликовано. Таким образом, вопросы, связанные с построением и эффективным использованием файловых хранилищ с использованием этой технологии, рассмотренные в данной работе, являются актуальными.

2 Основная часть

В современных операционных системах виртуальная память разделяется на пространство ядра и пространство пользователя [1]. К пространству ядра относятся все, что происходит в коде самого ядра операционной системы или в "пространстве" кода ядра и ресурсов, как правило, связанных с привилегированным пользователем, тогда как к пользовательскому пространству относится все, что происходит за пределами ядра, где любой пользователь может создать и запустить приложение и использовать системные ресурсы. При написании кода в пространстве ядра группа разработчиков должна уделять существенное внимание вопросам безопасности кода, защиты от возможных сбоев и восстановления после них, эффективности работы, взаимодействия с другими частями кода и даже о стиле кодирования, так как любая ошибка может привести к отказу всей системы. Кроме того, они должны думать о том, когда код отвечает на эти проблемы и принадлежит ядру, так как добавление кода сделает ядро больше и приведет к ряду нежелательных побочных эффектов.
Одним из существующих решений по снижению издержек, связанных с работой в пространстве ядра, является перемещение кода в пространство пользователя. Файловые системы в пространстве пользователя имеют некоторые преимущества: относительную легкость совместной разработки кода с обновлениями последующих версий, изолированность от ядра позволяет избежать отказов всей вычислительной системы в случае ошибок в ФС.

Одной из трудностей в написании кода в пользовательском пространстве является организация взаимодействия кода с ядром. В частности, файловой системе необходимо будет взаимодействовать с ядром VFS (Virtual File System) в какой-то момент для получения доступа к физическим устройствам хранения данных. Для упрощения организации такого взаимодействия был разработан FUSE. Это модуль для ядер UNIX-

подобных операционных систем, с открытым исходным кодом, который позволяет пользователям, не обладающими привилегиями, создавать собственные файловые системы без необходимости переписывать код ядра. Концепция FUSE заключается в создании простого модуля ядра, которое взаимодействует с ядром, в частности VFS, от имени непривилегированных приложений пользователя и имеет API, которые могут быть доступны из пользовательского пространства. На рисунке 1 показан упрощенный пример работы FUSE.

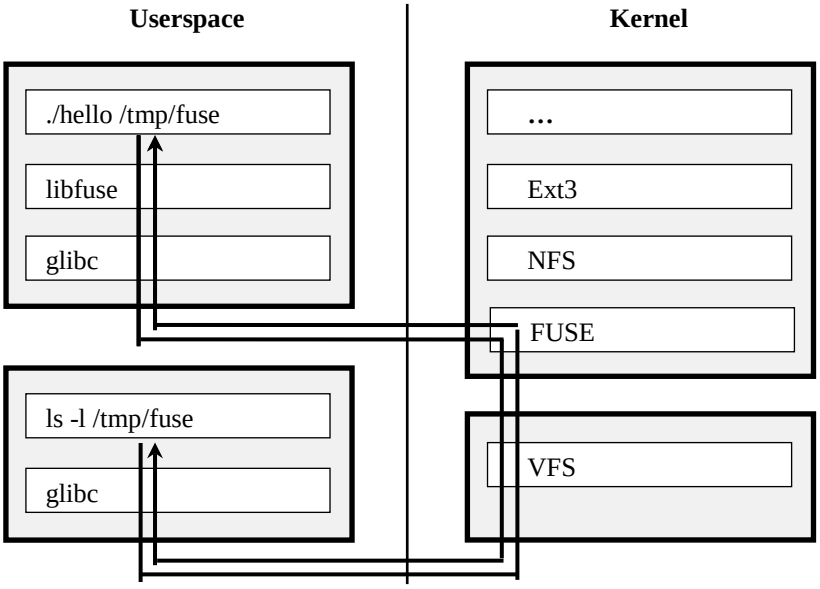

Рис. 1. Схема обращения файловой системы созданной на FUSE

На верхнем уровне тестовая файловая система компилируется для создания бинарного вызова "Hello". Этот бинарный вызов изображен в правом верхнем углу рис.1 в файловой системе с точкой монтирования /tmp/fuse. Затем пользователь выполняет команду ls -l к точке монтирования (ls –l /tmp/fuse). Эта команда транслируется через glibc (библиотеку языка Си, которая обеспечивает системные вызовы и основные функции) в VFS. VFS затем обращается к модулю FUSE с точкой монтирования, которая соответствует файловой системе построенной на FUSE. Модуль ядра FUSE затем проходит через glibc и libfuse (libfuse является библиотекой FUSE в пользовательском пространстве) и связывается с бинарным файлом ("Hello"). Двоичный файл система возвращает как результат обратно вниз по стеку к модели ядра FUSE, обратно через VFS в команду ls -l.

Таким образом, используя FUSE API можно создать практически любой тип файловой системы с практически любой функцией, как это было показано на тестовом примере выше. Более того, для этих целей возможно использовать различные языки программирования: C, C++, JAVA, Haskell, Erlang, TCL, Python, Perl, OCamlv, Ruby, Lua.

На данный момент существует много реализаций файловых систем построенных на FUSE. Иногда FUSE используется для создания прототипа или тестирования файловой системы. Ниже приведен список самых известных файловых систем, построенных на FUSE: SSHFS; GmailFS; EncFS; NTFS-3G; archivemount; ZFS-Fuse; CloudStore; MountableHDFS; GlusterFS; MooseFS; s3fs [2,3].

Для того, чтобы оценить преимущества и недостатки использования FUSE рассмотрим и проанализируем 2 файловые системы: GlusterFS, которая использует рассматриваемую технологию, и HDFS, которая не использует ее.

Благодаря FUSE в GlusterFS используется технология трансляторов, которые являются бинарными динамическими объектами (.so), загружаемыми в момент выполнения, исходя из параметров в файле томов. Использование необходимых трансляторов и их настройка позволяет гибко конфигурировать режим работы системы и подстраивать систему под определенные требования или условия [6,7]. В отличие от этого HDFS, которая построена по классической схеме, более производительна, хотя и не такая гибкая.

В HDFS есть сервер метаданных, который является единой точкой отказа. То есть при его отказе откажет вся файловая система [4,5]. В GlusterFS метаданные хранятся вместе с данными в расширенных атрибутах файлов, что делает сервер метаданных ненужным. Соответственно при поломке одной из машин пользователь может продолжать работать на остальных. Однако так как сервера метаданных нет, то значительно увеличены расходы на поиск и передачу нужной информации, так как системе нужно опросить все машины до того пока не найдет необходимую.

Использование FUSE и трансляторов дает возможность изменять размер блоков хранимых файлов, что дает возможность снизить расходы на хранение данных [6,7]. В HDFS блоки имеют фиксированный размер в 64 МБ, что снижает эффективность ее использования как файлового хранилища на основе облачного подхода и подразумевает использование для хранения преимущественно файлов большого размера. Однако, использование блоков постоянного размера повышает производительность и скорость работы файловой системы такого типа [4,5].

Все файловые системы, построенные на FUSE, работают на таких операционных системах, как Linux, Mac OS X, FreeBSD, OpenSolaris.

3 Заключение

Таким образом, к преимуществам подхода к построению ФС на основе FUSE можно отнести гибкость построения и конфигурации, что дает возможность максимально адаптировать их к конкретным аппаратным и программным требованиям существующей информационной инфраструктуры. К преимуществам также можно отнести то, что ошибки в коде файловой системы не повлекут за собой выход из строя всей информационной системы. Также использование FUSE означает что будет реализована POSIX – совместимость.

К недостаткам относится то, что файловые системы построенные с помощью FUSE менее производительны по сравнению с остальными и могут использоваться только с операционными системами, соответствующими стандартам POSIX (т.е., фактически не могут быть в полной мере реализованы в распространенном семействе операционных систем Windows).

Литература

[1] Userspace Wiki (08.03.2013): http://en.wikipedia.org/wiki/Userspace
[2] Linux magazine, Jeffrey B. Layton (22.06.2010): http://www.linux-mag.com/id/7814/
[3] FUSE Wiki (13.03.2013): http://ru.wikipedia.org/wiki/Filesystem_in_Userspace
[4] Yahoo Developer Network Module 2: The Hadoop Distributed File System (03.04.2013): http://developer.yahoo.com/hadoop/tutorial/module2.html
[5] Apache Hadoop официальный сайт (22.02.2013): http://hadoop.apache.org/docs/stable/hdfs_design.html
[6] GlusterFS Wiki (22.02.2012): http://ru.wikipedia.org/wiki/GlusterFS
[7] Официальный сайт GlusterFS (31.08.2011) http://gluster.org/community/documentation/index.php/Community/Ru:GlusterFS#.D0.A2.D1.80.D0.B0.D0.BD.D1.81.D0.BB.D1.8F.D1.82.D0.BE.D1.80.D1.8B