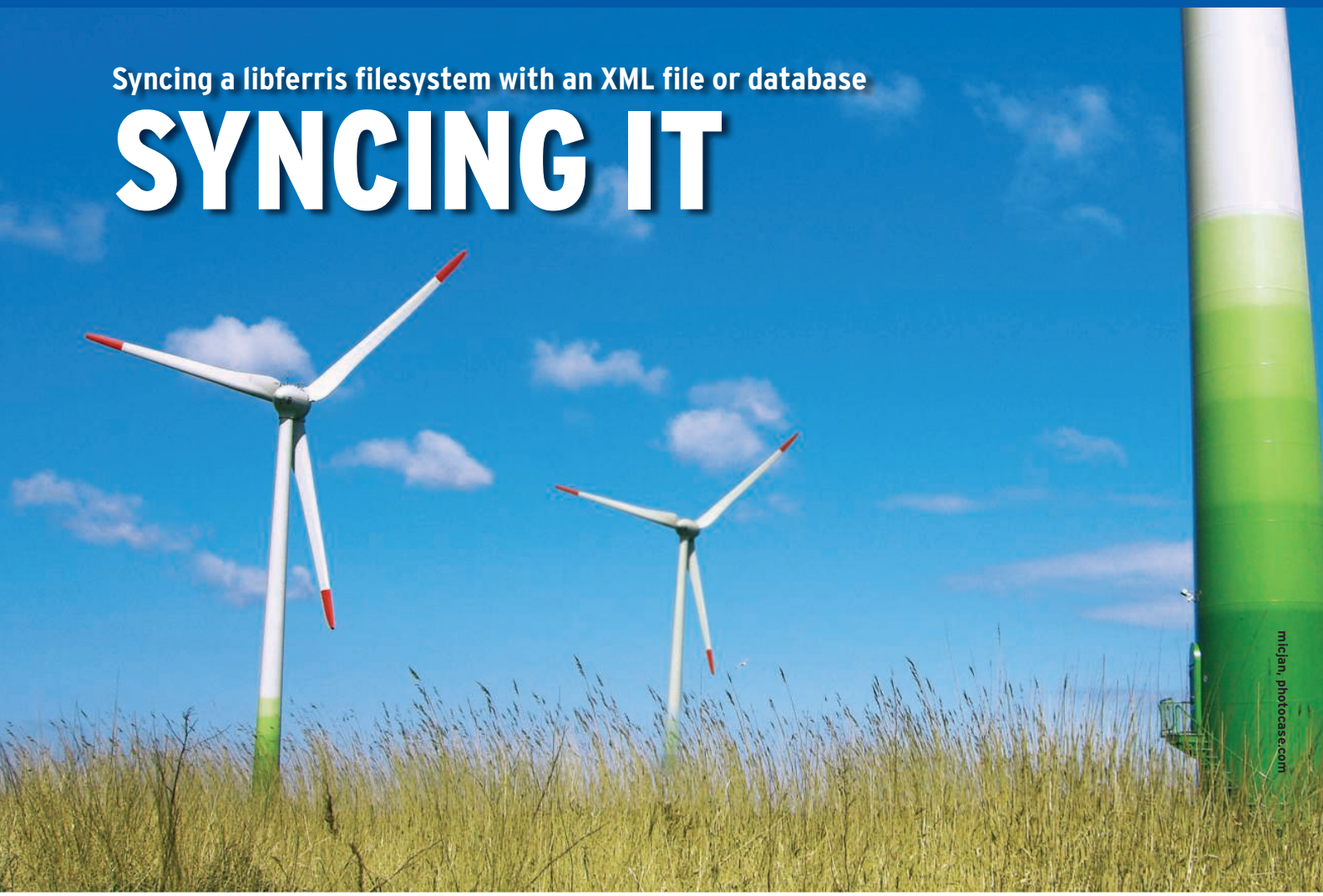
Syncing a libferris filesystem with an XML file or database

SYNCING IT



With libferris, FUSE, and rsync, you can synchronize a filesystem with a dissimilar data source. **BY BEN MARTIN**

Admins use rsync to snchronize two filesystem trees. With a few tricks, you can use FUSE and libferris with rsync [1][2][3] to synchronize a filesystem with another data source such as an XML file or a PostgreSQL database. Libferris is a user address space Virtual FileSystem (VFS) that lets you mount almost any data source as a filesystem. Examples of data sources libferris can mount include XML files, Berkeley db4 files, rpm packages, relational databases, LDAP servers, web servers, and applications like XWindow, Emacs, xmms, Amarok, and Firefox.

Libferris also includes evolving support for mounting web services. For example, you can interface a libferris directory with a photo-sharing website like 23hq or Flickr. In this article, I will discuss some of the possibilities for using rsync to synchronize a libferris filesystem with an XML file or database.

The ferrisfs application lets you expose libferris filesystems through FUSE. In the most basic form, ferrisfs requires two arguments. First, you can pass the URL of a libferris filesystem using *--url*. The last argument is where you want the FUSE filesystem to appear in your Linux kernel filesystem tree. Normally, I create a *fuse* subdirectory in my home directory where all my FUSE mount points appear.

Steps

Listing 1 shows some of the steps for setting up an interaction with a libferris-

Metadata and Search

Apart from mounting miscellaneous data sources, the other two goals of libferris are metadata handling and filesystem search.

Libferris comes with support for automatic metadata extraction and lets you add explicit metadata to any file on any filesystem regardless of the user's write permission.

As an example of libferris' metadata capability, consider adding a handy tag to a file on an FTP server in libferris for later identification. Even if the user does not have write access to the FTP server, libferris will store the metadata in Resource Description Framework (RDF) to associate the tag with the file. On the other hand, for a file in a home directory, if you add a metadata tag, libferrris will store the metadata in a kernel extended attribute to give non-libferris applications access via the attr(1) interface.

Metadata extraction in libferris covers simple cases such as extracting the dimensions and Exif data of image files, as well as more advanced cases. For example, if you tag files in the F-Spot photo management tool, you can then access those tags using libferris.

Filesystem search support in libferris allows you to create multiple filesystem indexes.

Plugins are used to let you build indexes using PostgreSQL, Lucene, Xapian, and other tools. You can even link indexes together to create a federation.

Recent versions support using libferris through FUSE, giving unmodified applications direct access to anything libferris sees as a filesystem.

Listing 1: FUSE Interaction on a Mounted XML File

```
$ cat simple-xml.xml
<simple-xml>
  <something/>
</simple-xml>
$ mkdir simple-xml
$ ferrisfs --url ~/fuse/
simple-xml.xml/simple-xml \
    simple-xml
$ ll simple-xml
total 0
-rwx------ 0 ferristester
ferristester 0 Jan  1  1970
something*
$ date >| simple-xml/something
$ cat simple-xml.xml
<?xml version="1.0"
encoding="UTF-8" standalone="no"
?>
<simple-xml>
  <something
mtime="1179838137">Tue May 22
22:48:57 EST 2007
</something>
</simple-xml>
```

backed FUSE filesystem. First a very basic XML file is created and mounted at *~/fuse/simple-xml*.

Notice that the *--url* parameter selects the first element in the XML file as the libferris filesystem (instead of the XML file itself).

XML files must have a single root element; by mounting that root element instead of the XML file, you avoid exposing this detail to the applications using the FUSE filesystem.

Normal filesystem metadata is mirrored in the XML file using XML attributes. By updating the contents of a file under the FUSE mount point, libferris both updates the contents of the XML element and records the modification time in an XML attribute.

Listing 2 shows rsync on a libferris-backed FUSE filesystem. First, the *source-native-fs* directory is created and populated with some simple test files. Other than the use of the *--temp-dir* command-line option, the command looks like any other invocation of rsync.

The final rsync invocation uses the *--delete-after* option to remove the *something* file, which was originally part of the XML file but is not part of the source filesystem passed to rsync.

The *grep* command checks that *something* is no longer part of the XML file after the sync.

Listing 2: Rsync to XML

```
$ mkdir source-native-fs
$ cd source-native-fs
$ date >datefile1.txt
$ date >datefile2.txt
$ touch emptyA
$ echo -n "hi there" > main
$ cd ~/fuse
$ mkdir ~/fuse/rsync-junk
$ rsync -avz -T ~/fuse/rsync-junk \
   source-native-fs/ simple-xml/
$ cat simple-xml.xml
<?xml version="1.0"
encoding="UTF-8" ...?>
<simple-xml atime="1179838274"
mode="40775"...
   mtime="1179838199">
  <something mtime="1179838137"
  >Tue May 22 22:48:57 EST 2007
</something>
  <datefile1.txt
atime="1179838337"
mode="100664"...
    mtime="1179838179">Tue May
22 22:49:39 EST 2007
</datefile1.txt>
...
  <main atime="1179838338"
mode="100664"...
    mtime="1179838199">hi
there</main>
</simple-xml>
$ rsync -avz  --delete-after \
  -T ~/fuse/rsync-junk \
   source-native-fs/ simple-xml/
building file list ... done
deleting something
sent 159 bytes  received 20 bytes
358.00 bytes/sec
total size is 66  speedup is 0.37
$ grep something simple-xml.xml
0
$ fusermount -u simple-xml
```

The previous section showed data being synced between a native kernel filesystem (ext3 in this case) and a subtree in an XML file.

Sync Across Filesystem Types

The libferris and FUSE combination allows you to convert between different data formats while you are performing the sync. By exposing part of an XML file through libferris and FUSE, you can keep various parts of an XML file in sync with other data – perhaps involving many different rsync invocations covering different parts of a single XML file.

The ability to rsync between different filesystems like this can be very convenient when both filesystems provide different features and you want a combination of these features. For example, many tools make editing XML simple, though accessing a single element (file) in XML is much slower than accessing a single file in a db4 file.

The commands shown in Listing 3 keep a db4 file in sync with the contents of an XML file. The simple-xml FUSE filesystem, which is based on the *sim-*

Listing 3: Rsyncing an XML File into a db4 File

```
$ fcreate `pwd` --create-type=db4
name=db4.db
$ mkdir db4
$ ferrisfs -u ~/fuse/db4.db db4
$ rsync -avz --delete-after -T ~/
fuse/rsync-junk  simple-xml/ db4/
$ db_dump -p db4.db
VERSION=3
format=print
type=btree
db_pagesize=4096
HEADER=END
 /atime
 1179840317
 /datefile1.txt/atime
 1179840317
 /datefile1.txt/mode
 100664
 /datefile1.txt/mtime
 1179838179
...
 datefile1.txt
 Tue May 22 22:49:39 EST 2007\0a
```

Listing 4: Using Rsync to Sync XML Attributes

```
$ fcreate `pwd` --create-type=db4 name=target.db
$ mkdir target
$ ferrisfs -u `pwd`/target.db target
$ cat attributes-in-xml.xml
<main>
  <sub1 attr1="hello" second="world"/>
  <gaw  another="value"/>
</main>
$ mkdir attributes-in-xml
$ ferrisfs -u `pwd`/attributes-in-xml.xml/main \
    attributes-in-xml
$ rsync -avz --delete-after -T ~/fuse/rsync-junk \
    attributes-in-xml/ target/
$ db_dump -p target.db
VERSION=3
...
HEADER=END
 gaw
 sub1
DATA=END
$ rsync -X -avz --delete-after -T ~/fuse/rsync-junk \
    attributes-in-xml/ target/
...building file list ...
rsync: rsync_xal_get: lgetxattr(".","as-xml",37199)
failed: Input/output error (5)
...
$ db_dump -p target.db
VERSION=3
...
HEADER=END
 gaw
 sub1
DATA=END
$ fusermount -u attributes-in-xml
$ ferrisfs -u `pwd`/attributes-in-xml.xml/main   \
   --show-ea-regex="(attr1|another|second)"      \
   --prepend-user-dot-prefix-to-ea-regex=".*"    \
    attributes-in-xml
$ rsync -X -avz --delete-after -T ~/fuse/rsync-junk \
    attributes-in-xml/ target/
$ db_dump -p target.db
...
HEADER=END
 /gaw/user.another
 value
 /sub1/user.attr1
 hello
 /sub1/user.second
 world
 gaw
 sub1
DATA=END
```

ple-xml.xml file in Listing 1, is reused here. If there are attributes in the XML file that are not the standard *lstat(2)* attributes, they are exposed by the libferris FUSE filesystem as extended attributes.

The rsync command has support for syncing extended attributes across filesystems using the *-X* (*--xattrs*) command-line option. In syncing extended attributes, libferris creates many virtual attributes to expose extra metadata about the filesystem.

To get around this extra metadata libferris wants to offer, the *ferrisfs* command has the option to limit what attri-

Listing 5: Accessing a PostgreSQL Database

```
$ psql ferristester
ferristester=> \d foobar
           Table "public.foobar"
 Column  |         Type          | Modifiers
---------+-----------------------+-----------
 fooid   | integer               | not null
 fooname | character varying(100) |
 e       | character varying(100) |
Indexes:
    "foobar_pkey" PRIMARY KEY, btree (fooid)
ferristester=> select * from foobar;
 fooid | fooname |          e
-------+---------+-----------------------
    10 | William |
    45 | Rick    | 15 credibility street
  3002 | Satou   | Tokyo
   101 | John    | Some data
(4 rows)
ferristester=> \q
$ ferrisls --xml pg://localhost/ferristester/foobar
<?xml version="1.0" encoding="UTF-8" ... ?>
<ferrisls>
  <ferrisls e="" fooid="" fooname="" ...
    name="foobar" primary-key="fooid" ...
   url="pg:///localhost/ferristester/foobar">
    <context e="" fooid="10"
      fooname="William" name="10".../>
    <context e="Tokyo" fooid="3002"
      fooname="Satou" name="3002".../>
...
  </ferrisls>
</ferrisls>
```

Listing 6: Rsyncing Data Out of a Table

```
$ mkdir pg
$ ferrisfs --show-ea=user.fooid,user.fooname,user.e \
  --prepend-user-dot-prefix-to-ea-regex=".*"  \
  --force-empty-file-contents-regex=".*" \
  -u pg://localhost/ferristester/foobar pg
$ ls -l pg
total 0
-rwx------ 0 ferristester ferristester 50 Jan  1
1970 10
-rwx------ 0 ferristester ferristester 57 Jan  1
1970 101
-rwx------ 0 ferristester ferristester 55 Jan  1
1970 3002
-rwx------ 0 ferristester ferristester 68 Jan  1
1970 45
$ cd pg
$ attr -l 101
Attribute "fooid" has a 3 byte value for 101
Attribute "fooname" has a 4 byte value for 101
Attribute "e" has a 9 byte value for 101
$ attr -g fooname 101
Attribute "fooname" had a 4 byte value for 101:
John
$ cd ..
$ mkdir target
$ rsync -Cavz -X -T ~/fuse/rsync-junk pg/ target/
building file list ... done
./
10
101
3002
45
7
sent 762 bytes  received 136 bytes  1796.00 bytes/
sec
total size is 0  speedup is 0.00
$ cd target
$ attr -l 3002
Attribute "e" has a 5 byte value for 3002
Attribute "fooid" has a 4 byte value for 3002
Attribute "fooname" has a 5 byte value for 3002
$ attr -g e 3002
Attribute "e" had a 5 byte value for 3002:
Tokyo
```

butes are reported from the FUSE filesystem. For example, using *--show-ea = user.dislikes* will make the FUSE filesystem report only the *user.dislikes* extended attribute. The result is that rsync will only try to sync that one extended attribute instead of a lot of other metadata that libferris makes available.

Another complication of syncing extended attributes is that filesystems report attributes that can be user modified with the *user.* prefix, so the attribute *dis-*

Listing 7: Rsyncing into a PostgreSQL Table

```
$ ferrisfs --show-ea=user.fooname,user.e,user.fooid \
  --prepend-user-dot-prefix-to-ea-regex=".*"  \
  --throw-away-write-to-file-contents-regex=".*" \
  --delay-commit-path=pg:///localhost/ferristester/
foobar \
  --delay-commit-path-trigger-ea=user.fooname \
  --throw-away-write-to-ea-regex=".*foobar" \
  -u pg://localhost/ferristester/foobar pg
$ rsync -avz -X -T ~/fuse/rsync-junk target/ pg/
building file list ... done
10
101
3002
45
7
sent 756 bytes  received 130 bytes  590.67 bytes/sec
total size is 0  speedup is 0.00
$ cd target
$ ll
total 28K
-rwx------ 1 ferristester ferristester 50 Jan  1
1970 10*
-rwx------ 1 ferristester ferristester 68 Jan  1
1970 45*
-rwx------ 1 ferristester ferristester 57 Jan  1
1970 101*
-rwx------ 1 ferristester ferristester 55 Jan  1
1970 3002*
$ attr -g fooname 10
Attribute "fooname" had a 7 byte value for 10:
William
$ attr -s fooname -V "Willie" 10
Attribute "fooname" set to a 6 byte value for 10:
Willie
$ touch 7
$ attr -s fooid -V 7 7
Attribute "fooid" set to a 1 byte value for 7:
7
$ attr -s fooname -V new-item 7
Attribute "fooname" set to a 8 byte value for 7:
new-item
$ cd ..
$ rsync -avz -X -T ~/fuse/rsync-junk target/ pg/
```

likes will only be readable by *getxattr(2)* using the name *user.dislikes*. As many XML files are not likely to have the *user.* prefix in their XML attributes, there is the ferrisfs - *-prepend-user-dot-prefix-to-ea-regex* command-line option to explicitly add *user.* to any attributes that match the given regular expression.

Listing 4 shows a first attempt to sync XML attributes as well as file content with ferrisfs and rsync. The first *db_dump* execution shows that none of the XML attributes have been written to the Berkeley db4 file. Using the rsync *-X* (*--xattrs*) command-line option to try to correct this gives the error message about "as-xml" not being available through *getxattr()*.

The trick is to use the ferrisfs - *-show-ea-regex* and - *-prepend-user-dot-prefix-to-ea-regex* options to only show the extended attributes you are interested in. If an attribute that matches *show-ea-regex* is available for a virtual libferris file, ferrisfs will export that attribute to FUSE as an extended attribute. As the final *db_dump* shows, the XML attributes are now available in the db4 file as well.

Listing 5 is a simple table in a PostgreSQL database. The table can be mounted by using the *postgresql://* or *pg://* URL in libferris, as the *ferrisls* command shows. Using a PostgreSQL table as the source for rsync presents no new issues with how to invoke ferrisfs, as shown in Listing 6. Each column in the table becomes an extended attribute in the target filesystem.

When the file contents of a tuple is read by libferris, it gives an XML serialized version of the data. As the extended attributes also give the same information in broken down format, you don't really care about the tuple's file content. Listing 6 solves this issue by reporting that all the tuples are zero-byte files.

Synching into PostgreSQL

Synchronizing information into a PostgreSQL database with rsync presents extra issues because a database table does not behave exactly like a filesystem. For example, as shown in Listing 5, the primary key of the table is *fooid*. Without specifying at least the primary key of the tuple to create, you cannot make a new file in a mounted PostgreSQL table.

Also, when the file contents of a tuple is read by libferris, it gives an XML serialized version of the tuple itself. Updating both the XML serialized version of a tuple and each individual table column through the extended attributes would be twice the effort. The *--throw-away-write-to-file-contents-regex* command-line option to ferrisfs solves the latter problem by ignoring anything that is written to the file's contents for files that have a URL matching the given regular expression. Updates must happen via the extended attributes interface.

The *--delay-commit-path* ferrisfs command-line option was added to solve the primary key issue. The nominated path allows new files to be created and extended attributes written on those new files without immediately trying to update the database. Listing 7 shows how to rsync into a PostgreSQL table.

The commands shown in Listing 8 create a second table and then populate it from *foobar* using rsync. If the commands from the *mkdir* command down are run again at a later time, then foo2 is updated using rsync with changes from the foobar table.

Future Directions

Support for rsync with PostgreSQL currently revolves around single tables. In the future, this support should expand to allow rsync to operate on an entire database at once.

Also, adding support for other syncing solutions like Unison [5] and Harmony [6] will be very interesting. ■

Listing 8: Keeping a Copy of a PostgreSQL Table

```
$ psql ferristester
ferristester=> create table foo2
  ( fooid serial primary key,
    fooname varchar(100),
    e varchar(100));
ferristester=> \q
$ mkdir -p foo2
$ ferrisfs --show-ea=user.fooname,user.e,user.fooid \
  --prepend-user-dot-prefix-to-ea-regex=".*"  \
  --force-empty-file-contents-regex=".*" \
  --force-empty-read-from-ea-regex=".*foobar" \
  -u pg://localhost/ferristester/foobar pg
$ ferrisfs --show-ea=user.fooname,user.e,user.fooid \
  --prepend-user-dot-prefix-to-ea-regex=".*"  \
  --throw-away-write-to-file-contents-regex=".*" \
  --delay-commit-path=pg:///localhost/ferristester/foo2 \
  --delay-commit-path-trigger-ea=user.fooname \
  --throw-away-write-to-ea-regex=".*foo2" \
  -u pg://localhost/ferristester/foo2 foo2
$ rsync -avz -X -T ~/fuse/rsync-junk pg/ foo2/
$ fusermount -u pg
$ fusermount -u foo2
```

INFO

[1] libferris: *http://witme.sourceforge.net/libferris.web/*

[2] rsync: *http://rsync.samba.org/*

[3] Filesystem in Userspace: *http://fuse.sourceforge.net/*

[4] fuselagefs and delegatefs: *http://sourceforge.net/project/showfiles.php?group_id=16036&package_id=225200*

[5] Unison bidirectional sync: *http://www.cis.upenn.edu/~bcpierce/unison/*

[6] Harmony bidirectional sync: *http://www.seas.upenn.edu/~harmony/*

THE AUTHOR

Ben Martin has been working on filesystems for more than 10 years. He is currently working toward a PhD. His research focuses on combining semantic filesystems with formal concept analysis to improve human-filesystem interaction.