TIRA Developer Manual

Tim Gollub Martin Potthast Martin Trenkmann

April 28, 2013

Contents

Chapter 1

The TIRA File System

The TIRA web service is a distributed application that employs several physical machines attached to a local network. The workflow of TIRA requires that all these machines have access to one common file system. For that reason we make use of the Network File System[1] (NFS).

Although NFS is called a distributed file system, all data is stored centralised on a dedicated machine which is known as the NFS server. The NFS server has direct access to the data and is able to supply it to the network via Remote Procedure Call[2] (RPC). On the other hand, a machine which accesses the data remotely is called NFS client.

The following sections describe the directory structure of the TIRA file system and where it is located on the NFS server and mounted on the client machines respectively. It is important to note that a virtual machine (VM) running on a NFS client does not see the complete TIRA file system at once but only the sub-tree which is associated with the user of the VM. Hence, we will show in detail when and how directories are made visible and readable to other users at runtime. The guide targets Ubuntu 12.04 which uses NFSv4 by default.

1.1 Setting Up the NFS Server

The NFS server is the central store for all BLOB[3] data associated with TIRA such as corpora, task descriptors and run results. The data is organized under a common directory tree which is shared among the TIRA infrastructure. We will refer to the root of this tree as the TIRA root directory. The following

[1] `http://en.wikipedia.org/wiki/Network_File_System`
[2] `http://en.wikipedia.org/wiki/Remote_procedure_call`
[3] `http://en.wikipedia.org/wiki/BLOB`

steps describe the installation of the NFS server and registration of the TIRA root directory.

1. Check if the NFS service is already installed on your system.

   ```
   # service nfs-kernel-server status
   ```

2. If the last command works skip this step, otherwise install NFS.

   ```
   # apt-get install nfs-kernel-server
   ```

3. Create the TIRA root directory under `/srv/tira` which contains the three directories `data`, `users` and `vms`.

   ```
   # mkdir -p /srv/tira/data
   # mkdir -p /srv/tira/users
   # mkdir -p /srv/tira/vms
   ```

4. Register the TIRA root directory in `/etc/exports` making it read-only for the network under `141.54.178.xxx`.

   ```
   # echo '/srv/tira 141.54.178.0/24(ro,sync,no_subtree_check)' \
     >> /etc/exports
   ```

5. If NFS was already installed, but not running start it.

   ```
   # service nfs-kernel-server start
   ```

6. Otherwise, force NFS to reload `/etc/exports`.

   ```
   # exportfs -ra
   ```

1.2 Setting Up an NFS Client

An NFS client is a machine that mounts the TIRA root directory into its local file system. By design a client has only read access to the shared data (option `ro` in `/etc/exports`). All write operations are executed directly on the NFS server. The following steps describe the installation of the NFS client and mounting of the TIRA root directory.

There is a need on client-side to write to the NFS when a user wants to publish the result of a run. We will tackle this issue later.

1. Install the NFS client package.

   ```
   # apt-get install nfs-common
   ```

2. Create the mountpoint for the TIRA root directory.

   ```
   # mkdir -p /mnt/nfs/tira
   ```

3. Mount the TIRA root directory. Replace `<host>` with the hostname of the NFS server.

   ```
   # mount <host>.medien.uni-weimar.de:/srv/tira /mnt/nfs/tira
   ```

4. Check if you can read the mounted directory.

   ```
   # ls /mnt/nfs/tira
   ```

1.3 Ownership and Visibility

The content of all directories is under control of the TIRA system. The only directory that is writable by the user is `output`. New data such as corpora and task descriptors will be provided via upload in the web frontend. To express ownership and visibility (read-access) we employ two concepts:

1. Ownership means to give one or more users permission to delete some data. For example, a user who uploads some corpus is automatically the owner of that corpus. However, he/she may grant ownership to other users of that corpus. The information about ownership is managed in a separate database system.

2. Visibility means to place a symbolic link pointing to some data into the `data` or `tasks` directory of the user. Of course ownership implies visibility.

1.4 User Activities

A user activity is any action that is initiated via the TIRA web frontend by some user. Most actions require updates in the database and/or the TIRA file system. It is important to note that the file system is under exclusive control of the TIRA system and that a user does never operate directly on it. The following sections mention some activities and the steps to perform on the file system.

1.4.1 Creating a New User

As a design goal there is no full Linux-like account for every TIRA user. All data is owned by `root` and TIRA users have only read access to the file system. However, the TIRA user's home directories are organized virtually in sub-directories under `<tiraroot>/users`. When adding a new user named `alice`, its home directory is created as follows on the NFS server.

```
# mkdir /srv/tira/users/alice
```

In addition, three sub-directories must be created. One of them, `output`, is a symbolic link that points to `/dev/null` by default. The purpose of this directory is described later.

```
# mkdir /srv/tira/users/alice/data
# mkdir /srv/tira/users/alice/tasks
# ln -s /dev/null /srv/tira/users/alice/output
```

- In the `data` directory there are all resources, such as corpora and run results, currently readable by the user. A resource is accessible via symbolic link.
- In the `tasks` directory there are descriptors of tasks the user currently participates in. A so called task descriptor is accessible via symbolic link.
- `output` is a symbolic link that acts as the output directory for all TIRA runs. It points to `<tiraroot>/data/<unique-run-dir>` if a run is in progress and to `/dev/null` otherwise.

1.4.2 Providing a Corpus

When `alice` wants to provide a new corpus, she do so via upload on the TIRA website. If the web server is on a different machine than the NFS server, the data needs to be copied. Finally, the corpus is placed under `/srv/tira/data` and a symbolic link is created in the home directory of the user pointing to the uploaded corpus.

```
# scp <webserver>:/tmp/uploads/<corpus> <nfsserver>:/srv/tira/data
# ln -s /srv/tira/data/<corpus> /srv/tira/users/alice/data/<corpus>
```

The user's home directory then looks like follows.

```
# tree --charset ascii /srv/tira/users/alice/
/srv/tira/users/alice/
|-- data
|   `-- corpus.txt -> /srv/tira/data/corpus.txt
|-- output -> /dev/null
`-- tasks
```

A corpus can comprise a single file or multiple files. We do not care about this detail at the moment.

1.4.3 Sharing a Corpus

Sharing a corpus means to give a user read-access to the corpus in question. To do so, the TIRA system creates a symbolic link in the user's `data` directory pointing to the corpus in the global data store under `<tiraroot>/data`. Let's say `alice` wants to share the corpus `corpus.txt` with `bob`, we execute:

```
# ln -s /srv/tira/data/corpus.txt \
  /srv/tira/users/bob/data/corpus.txt
```

The user's home directories then looks like follows.

```
# tree --charset ascii /srv/tira/
/srv/tira/
|-- data
|   `-- corpus.txt
|-- users
|   |-- alice
|   |   |-- data
|   |   |   `-- corpus.txt -> /srv/tira/data/corpus.txt
```

```
|   |   |-- output -> /dev/null
|   |   `-- tasks
|   `-- bob
|       |-- data
|       |   `-- corpus.txt -> /srv/tira/data/corpus.txt
|       |-- output -> /dev/null
|       `-- tasks
`-- vms
```

It is not clear if a user can have access to a corpus without participating a specific task. Maybe a user has to select a task first to be able to see associated corpora.

1.4.4 Providing a Task

At the moment this is analogous to *Providing a Corpus*.

1.4.5 Participate in a Task

At the moment this is analogous to *Sharing a Corpus*.

1.4.6 Running a Task

When `bob` wants to run a task he must provide a program that will be executed by the TIRA system. In general this program is developed and tested somewhere in `bob`'s virtual machine. Because he finally browses to the executable via the web frontend there is no need to copy it to any dedicated location. The input and output directories are passed as absolute paths. However, running a task means to do the following steps on the NFS server:

1. In `<tiraroot>/data` create a new directory that gather the run's output. The directory should be named after the name of the task, the user and some timestamp.

   ```
   # mkdir /srv/tira/data/<taskname-username-runid>
   ```

2. The link behind the user's `output` directory is updated to point to the newly created output directory.

   ```
   # rm /srv/tira/users/bob/output
   # ln -s /srv/tira/data/<taskname-username-runid> \
     /srv/tira/users/bob/output
   ```

Doing so for some imaginary task we end up with:

```
tree --charset ascii /srv/tira/
/srv/tira/
|-- data
|   |-- corpus.txt
|   `-- sometask-bob-12345
|-- users
|   |-- alice
|   |   |-- data
|   |   |   `-- corpus.txt -> /srv/tira/data/corpus.txt
|   |   |-- output -> /dev/null
|   |   `-- tasks
|   `-- bob
|       |-- data
|       |   `-- corpus.txt -> /srv/tira/data/corpus.txt
|       |-- output -> /srv/tira/data/sometask-bob-12345
|       `-- tasks
`-- vms
```

1.4.7 Creating a Virtual Machine

The user's virtual machine (VM) runs on a TIRA NFS client. The home directory of the user is made available via the *Shared Folders* feature of *VirtualBox*. From the user's points of view its home directory is mounted under `???` in its VM.

For command line instruction go to Anna's VirtualBox documentation.