NORDUGRID-MANUAL-10
20/10/2009

Chelonia Administrator's manual

Zsombor Nagy*
Jon Nilsen†
Salman Zubair Toor ‡

*zsombor@niif.hu
†j.k.nilsen@usit.uio.no
‡salman.toor@it.uu.se

Contents

1 Installing a centralized system

The Chelonia storage system can be installed from binary packages, or from compiling the source packages. Here we assume that it is installed on every machine we want to use. But we can do a quick check if the ARC client python libraries are installed properly. It is recommended to set the PYTHONPATH variable, e.g.:

```
$ export PYTHONPATH=/usr/local/lib/python2.5/site-packages
```

Then we can check if the packages can be imported:

```
$ python
Python 2.5.2 (r252:60911, Jan  4 2009, 17:40:26)
[GCC 4.3.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import arc
>>> import storage
>>> import arcom
>>>
```

If there is no error on importing any of the packages, everything is OK.

It is very important to syncronize the time of the machines (at least within a few seconds) - the storage system will make decisions based on differences between timestamps. When deploying multiple machines, different times will cause problems.

In this document we will use a debian lenny system, and we will run multiple versions of the ARC HED hosting environment to simulate multiple machines. We will use three different hostnames , which will in this case point to the same machine, but in real deployment they would of course point to different machines (`chelonia[123].knowarc.eu`).

Host (and user) certificates are required to run the system. For testing purposes the NorduGrid InstantCA solution can be used. The InstantCA is capable of creating a demo-CA with short lifetime. The URL of the InstantCA service is `https://vls.grid.upjs.sk/CA/instantCA`. Now, create 3 user and 3 host certificates. (for this example, we set the Organization Name to 'knowarc', the Common name of the CA to 'chelonia', the name of the users to 'penny', 'billy' and 'hammer' and the name of the hosts to 'chelonia1', 'chelonia2' and 'chelonia3'. A password for the CA and passwords for the user certificates are needed as well, these should be at least 4 characters long).

Now, download the generated certificates and untar the archive file to see its contents:

```
$ ls -lR
.:
total 28
drwxr-xr-x 2 zsombor zsombor 4096 2009-10-16 13:31 CA
-rw-r--r-- 1 zsombor zsombor  951 2009-10-16 13:31 ca.key
-rw-r--r-- 1 zsombor zsombor  778 2009-10-16 13:31 ca.pem
drwxr-xr-x 2 zsombor zsombor 4096 2009-10-16 13:31 hostCerts
-rw-r--r-- 1 zsombor zsombor 1044 2009-10-16 13:31 readme
-rw-r--r-- 1 zsombor zsombor    3 2009-10-16 13:31 serial.srl
drwxr-xr-x 2 zsombor zsombor 4096 2009-10-16 13:31 userCerts

./CA:
total 8
-rw-r--r-- 1 zsombor zsombor 778 2009-10-16 13:31 afc5f10f.0
-rw-r--r-- 1 zsombor zsombor 135 2009-10-16 13:31 afc5f10f.signing_policy

./hostCerts:
total 24
-rw-r--r-- 1 zsombor zsombor 782 2009-10-16 13:31 hostcert-chelonia1.pem
```

```
-rw-r--r-- 1 zsombor zsombor 782 2009-10-16 13:31 hostcert-chelonia2.pem
-rw-r--r-- 1 zsombor zsombor 782 2009-10-16 13:31 hostcert-chelonia3.pem
-rw-r--r-- 1 zsombor zsombor 887 2009-10-16 13:31 hostkey-chelonia1.pem
-rw-r--r-- 1 zsombor zsombor 887 2009-10-16 13:31 hostkey-chelonia2.pem
-rw-r--r-- 1 zsombor zsombor 887 2009-10-16 13:31 hostkey-chelonia3.pem

./userCerts:
total 24
-rw-r--r-- 1 zsombor zsombor 769 2009-10-16 13:31 usercert-billy.pem
-rw-r--r-- 1 zsombor zsombor 774 2009-10-16 13:31 usercert-hammer.pem
-rw-r--r-- 1 zsombor zsombor 769 2009-10-16 13:31 usercert-penny.pem
-rw-r--r-- 1 zsombor zsombor 963 2009-10-16 13:31 userkey-billy.pem
-rw-r--r-- 1 zsombor zsombor 963 2009-10-16 13:31 userkey-hammer.pem
-rw-r--r-- 1 zsombor zsombor 963 2009-10-16 13:31 userkey-penny.pem
```

Here can be seen the certificate and key files for all the hosts and users, and the CA file with the proper hashed name. There is no rule about where to put these certificate, but it is common to put them in `/etc/grid-security`.

In these examples we assume that the CA certificate is put into `/etc/grid-security/certificates`,

and the host's certificate and key is at `/etc/grid-security/hostcert-chelonia[123].pem`

and `/etc/grid-security/hostkey-chelonia[123].pem`.

```
$ ls -l /etc/grid-security/certificates/
total 8
-rw-r--r-- 1 root root 778 2009-10-16 14:55 afc5f10f.0
-rw-r--r-- 1 root root 135 2009-10-16 14:55 afc5f10f.signing_policy
```

Choose a user certificate and put it into the `~/.arc` directory, removing the password from the key file for convenience:

```
$ mkdir ~/.arc
$ cp userCerts/usercert-billy.pem ~/.arc
$ openssl rsa -in userCerts/userkey-billy.pem -out ~/.arc/userkey-billy.pem
$ chmod 600 ~/.arc/userkey-billy.pem
$ ls -l ~/.arc
total 8
-rw-r--r-- 1 zsombor zsombor 778 2009-10-16 14:55 usercert-billy.pem
-rw------- 1 zsombor zsombor 891 2009-10-16 14:55 userkey-billy.pem
```

The storage system runs within the `arched` hosting environment daemon, which needs a configuration file describing which services to be run, on which ports do to listen, etc. There are several profiles for different deployment scenarios, these profiles are installed by default in the `/usr/local/share/arc/profiles` directory[4]. First we will use the profile called `CheloniaAllServicesCentralizedAHashWithISIS`. After we have chosen a profile, we need to create a config file which contains the deployment-specific attributes. Let's create a file in `/etc/arc` called `chelonia1.ini`:

```
profile=/usr/local/share/arc/profiles/CheloniaAllServicesCentralizedAHashWithISIS.xml

pidfile=/tmp/chelonia1.pid
logfile=/var/log/chelonia1.log
debug=ERROR

port=60001
```

[4]or they can be acquired from http://svn.nordugrid.org/trac/nordugrid/browser/arc1/trunk/src/hed/profiles

```
cacert=/etc/grid-security/certificates
host_cert=/etc/grid-security/hostcert-chelonia1.pem
host_key=/etc/grid-security/hostkey-chelonia1.pem
client_cert=/etc/grid-security/hostcert-chelonia1.pem
client_key=/etc/grid-security/hostkey-chelonia1.pem

[register]
index_server1=https://chelonia1.knowarc.eu:60001/ISIS

[ahash]
endpoint=https://chelonia1.knowarc.eu:60001/AHash
datadir=/var/spool/arc/chelonia1_ahash_data

[bartender]
endpoint=https://chelonia1.knowarc.eu:60001/Bartender

[librarian]
endpoint=https://chelonia1.knowarc.eu:60001/Librarian

[shepherd]
endpoint=https://chelonia1.knowarc.eu:60001/Shepherd
turlprefix=https://chelonia1.knowarc.eu:60001/Hopi
datadir=/var/spool/arc/chelonia1_shepherd_data
storedir=/var/spool/arc/chelonia1_shepherd_store
transferdir=/var/spool/arc/chelonia1_shepherd_transfer

[isis]
endpoint=https://chelonia1.knowarc.eu:60001/ISIS
dbpath=/var/spool/arc/chelonia1_isis_data
peerurl1=https://chelonia1.knowarc.eu:60001/ISIS
```

In this config we configure several services, and we have some common parts as well. The first line points to the profile which we want to use (please make sure that it is there or specify the correct path). The common parts contains the logfile, the level of debugging, the port to listen, the path to the certificates to be used on the service side, and the certificates to be used when a service wants to connect to an other service (in current case they are the same). Then we have an ISIS service which stores information about the running services and it is needed for the services to find eachother. The register section tells all the other services where to find the ISIS service. And then we have all the storage services: the Bartender, the Librarian, the Shepherd and the A-Hash. The endpoints of the services should be specified for all services. There is one additional service which doesn't need extra configuration: the Hopi service is a lightweight HTTP server. The Shepherd service is connected to the Hopi service, that's why it is needed to specify the Hopi's URL in the Shepherd's section (`turlprefix`). Some of these services need to have directories to store data. We will use subdirectories in `/var/spool/arc`, so we have to create this directory first:

```
$ sudo mkdir /var/spool/arc
```

Now, run the `arched` daemon with this INI config which is specified with the `-i` option. (In this document we will run the ARC HED daemon as root.) We can first run it in the foreground (with the `-f` flag), to see the possible initial misconfiguration errors immediately:

```
$ sudo /usr/local/sbin/arched -c /etc/arc/storage_service.xml -f
```

When we start the services, they are trying hard to find eachother, which means lots of connection related error messages in the beginning, but after a minute or so, the system should stabilize. (This is the case with shutdown as well, it is normal to get lots of connection related error messages). The `arched` daemon should now listen no port 60001:

```
$ netstat -at | grep 60001
tcp        0      0 *:60001                 *:*                     LISTEN
```

Now it is time to set up the client tool to access the system. It is called `chelonia` and it is by default installed to `/usr/local/bin`. The CLI tool needs to be told where it can find the user credentials, and a URL of one or more Bartender services or preferably the URL of one or more ISIS services, where Bartender services are registered. The Bartender service is the front-end of the storage system. These can be specified with a config file, which is called `~/.arc/client.conf` by default:

```
defaultservices=index:ARC1:https://chelonia1.knowarc.eu:60001/ISIS
keypath=/home/zsombor/.arc/userkey-billy.pem
certificatepath=/home/zsombor/.arc/usercert-billy.pem
cacertificatesdirectory=/etc/grid-security/certificates
```

Now we can use the tool `chelonia` to access the local deployment. If our client.conf is not at the default directory, then we can specify the path with the `-z` flag. First let's try to do a 'list' on the root collection ('/'):

```
zsombor@chelonia:~$ chelonia list /
ERROR: No Bartender URL found.
(use the '-v' flag for more details, e.g. 'chelonia -v list /')
```

If we get this error message, we should call it again with the `-v` verbose flag:

```
zsombor@chelonia:~$ chelonia -v list /
- Using default user config.
- The key file: /home/zsombor/.arc/userkey-billy.pem
- The cert file: /home/zsombor/.arc/usercert-billy.pem
- The CA dir: /etc/grid-security/certificates
- No Bartender URL found in the config or in the environment, try to get one from ISIS:
[2009-10-20 14:11:34] [Arc.Loader] [ERROR] [19139/149632560]
    Component tcp.client(tcp) could not be created
[2009-10-20 14:11:34] [Arc.Loader] [ERROR] [19139/149632560]
    Component tls.client(tls) could not be created
[2009-10-20 14:11:34] [Arc.Loader] [ERROR] [19139/149632560]
    Component http.client(http) could not be created
[2009-10-20 14:11:34] [Arc.Loader] [ERROR] [19139/149632560]
    Component soap.client(soap) could not be created
  - https://chelonia1.knowarc.eu:60001/ISIS - Failed to connect to ISIS.
ERROR: No Bartender URL found.
```

If we get the error 'Component [...] could not be created', it is usually because python somehow doesn't find the ARC clien libraries, so let's check if we have the PYTHONPATH variable set properly, and set it, if not:

```
$ echo $PYTHONPATH

$ export PYTHONPATH=/usr/local/lib/python2.5/site-packages/
```

Without arguments, the `chelonia` tool prints the list of available methods. Specifying only a method name prints the syntax of the given method. The `list` method requires Logical Names (LN). In the storage system, every file and collection has a Logical Name, which is a path within the namespace.

Now a (centralized) Chelonia is running, including a Hopi service, which is a basic HTTP server, for transferring files. The Bartender service is the front-end of the system to the users, so the clients always connect to the Bartender service. There are several directories configured in the config file where the services store their data. These directories can always be emptied, which resets the whole system to a clean state. If Chelonia is running with clean data directories, then it is completely empty, meaning that there is not even a root collection:

```
$ chelonia list /
'/': not found
```

The root collection has the Logical Name of '/', and it's role to provide a starting point for all the other Logical Names. This root collection needs to be created first:

```
$ chelonia mkdir /
Creating collection '/': done

$ chelonia list /
'/': collection
    empty.
```

Now there is an empty root collection.

2 Using the chelonia client tool

We assume that Chelonia is running and can be accessed with the `chelonia` tool. The `~/.arc/client.conf` is set properly, the contents of the root collection can be listed:

```
$ chelonia list /
'/': collection
    empty.
```

Now, create a small file and upload it:

```
$ echo ORANGE > orange
$ cat orange
ORANGE

$ chelonia put orange /
'orange' (7 bytes) uploaded as '/orange'.

$ chelonia list /
'/': collection
  orange <file>
```

Additional information about the file can be found with the stat method:

```
$ chelonia stat /orange
'/orange': found
  states
    checksumType: md5
    neededReplicas: 3
    size: 7
    checksum: a7992bd22ce079983da12c0172983ff4
  entry
    owner: /DC=eu/DC=KnowARC/O=knowarc/CN=billy
    GUID: eb973cb4-7924-0663-2b3e-13c01203fcd1
    type: file
  parents
    0/orange: parent
  locations
    https://chelonia1.knowarc.eu:60001/Shepherd 630b9bf0-7b70-8e04-1098-01769af3f8ea: alive
  timestamps
    created: 1256041232.28
```

Here, the owner of the file (Billy), the location of the replicas (currently only one), the number of needed replicas (which is three), the checksum and size of the file, and the timestamp of the creation is shown.

This file can be downloaded:

```
$ chelonia get /orange /tmp
'/orange' (7 bytes) downloaded as '/tmp/orange'.

$ cat /tmp/orange
ORANGE
```

We can create collections which can contain files and other collections forming a tree-hierarchy. We can move files and collection around within this namespace.

```
$ chelonia mkdir /fruits
Creating collection '/fruits': done

$ chelonia mkdir /fruits/apple
Creating collection '/fruits/apple': done

$ chelonia mv /orange /fruits
Moving '/orange' to '/fruits': targetexists
```

Note that if we want to move something into a colleciton, we have to end the collection's logical name with a slash:

```
$ chelonia mv /orange /fruits/
Moving '/orange' to '/fruits/': moved

$ chelonia ls /fruits
'/fruits': collection
  orange <file>
  apple  <collection>
```

We can remove files, and remove collections (but only if the collection is empty):

```
$ chelonia rmdir /fruits
Removing collection '/fruits': collection is not empty

$ chelonia rmdir /fruits/apple
Removing collection '/fruits/apple': removed

$ chelonia rm /fruits/orange
/fruits/orange: deleted

$ chelonia rmdir /fruits
Removing collection '/fruits': removed
```

Now, let's introduce a new user, Penny, into the system. The easiest would be to create a new client.conf for Penny:

```
$ cat /home/zsombor/.arc/penny.conf
defaultservices=index:ARC1:https://chelonia1.knowarc.eu:60001/ISIS
keypath=/home/zsombor/.arc/userkey-penny.pem
certificatepath=/home/zsombor/.arc/usercert-penny.pem
cacertificatesdirectory=/etc/grid-security/certificates
```

We should copy her certificate and key the same way as we did with Billy's certificate and key. Then we can specify Penny's config with the `-z` flag (and we can use the `-v` flag to see what is happening:

```
$ chelonia -v -z ~/.arc/penny.conf list /
```

```
- Config file specified: /home/zsombor/.arc/penny.conf
- The key file: /home/zsombor/.arc/userkey-penny.pem
- The cert file: /home/zsombor/.arc/usercert-penny.pem
- The CA dir: /etc/grid-security/certificates
- No Bartender URL found in the config or in the environment, try to get one from ISIS:
  - https://chelonia1.knowarc.eu:60001/ISIS - Got Bartender URL from ISIS: https://chelonia1.knowarc
- The URL of the Bartender(s): https://chelonia1.knowarc.eu:60001/Bartender
- Calling the Bartender's list method...
- Trying https://chelonia1.knowarc.eu:60001/Bartender...
- done in 0.15 seconds.
'/': denied

$ chelonia -z ~/.arc/penny.conf put orange /pennys-orange
/pennysorange: failed to add child to parent
```

Listing the root collection or uploading files into it for Penny is denied, because the root collection is owned by Billy, and nobody else has permissions to it. Now, Billy allows for everyone to access the root collection:

```
$ chelonia policy / change ALL +read +addEntry
Setting action list of '/' for user ALL to +read +addEntry: set.
```

The syntax of the `policy` method is the following:

```
$ chelonia pol
$ chelonia policy
Usage: policy <LN> <changeType> <identity> <action list>
  <changeType> could be 'set', 'change' or 'clear'
    'set': sets the action list to the given user overwriting the old one
    'change': modify the current action list with adding and removing actions
    'clear': clear the action list of the given user
  <identity> could be a '<UserDN>' or a 'VOMS:<VO name>'
  <action list> is a list actions prefixed with '+' or '-'
    e.g. '+read +addEntry -delete'
    possible actions:  read addEntry removeEntry delete
                       modifyPolicy modifyStates modifyMetadata
```

So Billy has changed the action list for user 'ALL' to '+read +addEntry', which means that everybody can list the contents of this collection, create subcollections in it, upload files into it, or move existing entries into it.

Now, Penny can upload a file:

```
$ chelonia -z ~/.arc/penny.conf put orange /pennys-orange
'orange' (7 bytes) uploaded as '/pennys-orange'.
```

This file now only can be downloaded by Penny, so she should allow for Billy to download it as well:

```
$ chelonia get /pennys-orange /tmp
/pennys-orange: denied

$ chelonia -z ~/.arc/penny.conf pol /pennys-orange
    change /DC=eu/DC=KnowARC/O=knowarc/CN=billy +read
Setting action list of '/pennys-orange' for user /DC=eu/DC=KnowARC/O=knowarc/CN=billy
    to +read: set.

$ chelonia stat /pennys-orange
'/pennys-orange': found
```

```
  locations
    https://chelonia1.knowarc.eu:60001/Shepherd 22be0347-ab35-42c8-12a0-07a50160399a: alive
  states
    checksumType: md5
    neededReplicas: 3
    size: 7
    checksum: a7992bd22ce079983da12c0172983ff4
  parents
    0/pennys-orange: parent
  timestamps
    created: 1256044171.04
  policy
    /DC=eu/DC=KnowARC/O=knowarc/CN=billy: +read
  entry
    owner: /DC=eu/DC=KnowARC/O=knowarc/CN=penny
    GUID: ea4fdb99-6a13-e926-0ec5-093e9fac524e
    type: file
```

The metadata of the file shows that Billy has 'read' rights. Billy can now download this file:

```
$ chelonia get /pennys-orange /tmp
'/pennys-orange' (7 bytes) downloaded as '/tmp/pennys-orange'.

$ cat /tmp/pennys-orange
ORANGE
```

Currently we only have one storage element (a Shepherd service and a Hopi service together), that's why the file has only one replica, even though it should have 3. However the number of needed replicas of an existing file can be modified with the 'modify' method:

```
$ chelonia stat /pennys-orange | grep neededReplicas
    neededReplicas: 3

$ chelonia -z ~/.arc/penny.conf modify /pennys-orange set states neededReplicas 2
set

$ chelonia stat /pennys-orange | grep neededReplicas
    neededReplicas: 2
```

It should be noted that the current version of the `chelonia` tool does not support recursive methods or working with multiple files at once.

3 Adding more storage elements

The following shows how to add a new storage element to the system. A new storage element should be on a different physical machine, preferably in a different building to minimize the possibility of both storage elements being offline at the same time. But for testing purposes it is of course possible to run it on the same machine using different ports. A storage element consists of a Shepherd service and a storage element service, which will be our lightweight HTTP server called Hopi. (But there is support for using apache as well.)

For our second machine we will use the profile called `CheloniaShepherdWithHopi`, which only contains the Shepherd service and the Hopi service. Here is the INI config for our chelonia2 server:

```
profile=/usr/local/share/arc/profiles/CheloniaShepherdWithHopi.xml
```

```
pidfile=/tmp/chelonia2.pid
logfile=/var/log/chelonia2.log
debug=ERROR

port=60002

cacert=/etc/grid-security/certificates
host_cert=/etc/grid-security/hostcert-chelonia2.pem
host_key=/etc/grid-security/hostkey-chelonia2.pem
client_cert=/etc/grid-security/hostcert-chelonia2.pem
client_key=/etc/grid-security/hostkey-chelonia2.pem

[register]
index_server1=https://chelonia1.knowarc.eu:60001/ISIS

[shepherd]
endpoint=https://chelonia2.knowarc.eu:60002/Shepherd
turlprefix=https://chelonia2.knowarc.eu:60002/Hopi
datadir=/var/spool/arc/chelonia2_shepherd_data
storedir=/var/spool/arc/chelonia2_shepherd_store
transferdir=/var/spool/arc/chelonia2_shepherd_transfer
```

The Hopi service doesn't need additional configuration, it is tied to the Shepherd here. We have to specify the ISIS running on the `chelonia1` server in order to the services in the two container could find eachother.

While our first server (`chelonia1`) is still running, start this new server:

```
$ sudo arched -i /etc/arc/chelonia2.ini
$ cat /tmp/chelonia2.pid
21477
```

Now in a few minutes the file should have a second replica:

```
$ chelonia stat /pennys-orange
'/pennys-orange': found
  locations
    https://chelonia1.knowarc.eu:60001/Shepherd 22be0347-ab35-42c8-12a0-07a50160399a: alive
    https://chelonia2.knowarc.eu:60002/Shepherd 386aec8c-174c-c83c-e5ea-f0784f5ff79f: alive
[...]
```

Let's start a third server called `chelonia3` with this config:

```
profile=/usr/local/share/arc/profiles/CheloniaShepherdWithHopi.xml

pidfile=/tmp/chelonia3.pid
logfile=/var/log/chelonia3.log
debug=ERROR

port=60003

cacert=/etc/grid-security/certificates
host_cert=/etc/grid-security/hostcert-chelonia3.pem
host_key=/etc/grid-security/hostkey-chelonia3.pem
client_cert=/etc/grid-security/hostcert-chelonia3.pem
client_key=/etc/grid-security/hostkey-chelonia3.pem

[register]
index_server1=https://chelonia1.knowarc.eu:60001/ISIS
```

```
[shepherd]
endpoint=https://chelonia3.knowarc.eu:60003/Shepherd
turlprefix=https://chelonia3.knowarc.eu:60003/Hopi
datadir=/var/spool/arc/chelonia3_shepherd_data
storedir=/var/spool/arc/chelonia3_shepherd_store
transferdir=/var/spool/arc/chelonia3_shepherd_transfer
```

It again has the same ISIS configured. Let's start it:

```
$ sudo arched -i /etc/arc/chelonia3.ini
```

Now when checking the file again there should be no change at all:

```
$ chelonia stat /pennys-orange
'/pennys-orange': found
  locations
    https://chelonia1.knowarc.eu:60001/Shepherd 22be0347-ab35-42c8-12a0-07a50160399a: alive
    https://chelonia2.knowarc.eu:60002/Shepherd 386aec8c-174c-c83c-e5ea-f0784f5ff79f: alive
[...]
```

Let's stop `chelonia2` to see what is happening with the two replicas:

```
$ chelonia stat /pennys-orange
'/pennys-orange': found
  locations
    https://chelonia1.knowarc.eu:60001/Shepherd 22be0347-ab35-42c8-12a0-07a50160399a: alive
    https://chelonia2.knowarc.eu:60002/Shepherd 386aec8c-174c-c83c-e5ea-f0784f5ff79f: offline
[...]

$ chelonia stat /pennys-orange
'/pennys-orange': found
  locations
    https://chelonia1.knowarc.eu:60001/Shepherd 22be0347-ab35-42c8-12a0-07a50160399a: alive
    https://chelonia3.knowarc.eu:60003/Shepherd 4a5cb7d5-bcec-0c84-e37f-3d2658ed3d4c: alive
    https://chelonia2.knowarc.eu:60002/Shepherd 386aec8c-174c-c83c-e5ea-f0784f5ff79f: offline
[...]
```

A new replica has been created on the third server to maintain the needed number of 2.

Let's start `chelonia2` again, and check the state of the replicas:

```
$ chelonia stat /pennys-orange
'/pennys-orange': found
  locations
    https://chelonia1.knowarc.eu:60001/Shepherd 22be0347-ab35-42c8-12a0-07a50160399a: alive
    https://chelonia3.knowarc.eu:60003/Shepherd 4a5cb7d5-bcec-0c84-e37f-3d2658ed3d4c: alive
    https://chelonia2.knowarc.eu:60002/Shepherd 386aec8c-174c-c83c-e5ea-f0784f5ff79f: alive

$ chelonia stat /pennys-orange
'/pennys-orange': found
  locations
    https://chelonia1.knowarc.eu:60001/Shepherd 22be0347-ab35-42c8-12a0-07a50160399a: alive
    https://chelonia3.knowarc.eu:60003/Shepherd 4a5cb7d5-bcec-0c84-e37f-3d2658ed3d4c: alive
    https://chelonia2.knowarc.eu:60002/Shepherd 386aec8c-174c-c83c-e5ea-f0784f5ff79f: thirdwheel
```

```
$ chelonia stat /pennys-orange
'/pennys-orange': found
  locations
    https://chelonia1.knowarc.eu:60001/Shepherd 22be0347-ab35-42c8-12a0-07a50160399a: alive
    https://chelonia3.knowarc.eu:60003/Shepherd 4a5cb7d5-bcec-0c84-e37f-3d2658ed3d4c: alive
```

One of the replicas has been removed to maintain the needed number of 2. Sometimes it takes a while while the Shepherds figure out how should remove the replica to make sure we don't loose the file accidentaly.

4 Make the full system replicated

Now the deployment is as following: One machine has the A-Hash, Librarian and Bartender services, and all three machines have a Shepherd and a storage element service. The A-Hash service stores all the metadata of all the files and collections, so it is critical. If the first machine goes offline then the whole system dies. However, it is possible to deploy the A-Hash service on all the machines and still have one consistent metadata database. The replicated version of the A-Hash uses the Berkeley DB to provide a distributed database. But the A-Hash is just the database backend of the Librarian, so we need more than one Librarian as well. And the Bartender service is critical for the user to access the system, so we should have more than one Bartender services. And because of all the services are registered in the ISIS service, and they find eachother by querying the ISIS, that's why we need more than one ISIS as well. So let's deploy every service on each machine.

While the Librarian, Bartender and ISIS services could be easily deployed on multiple sites (the ISIS services should have eachother URLs in the config), the A-Hash services should form a consistent replicated database together, and this is implemented in a different version of the A-Hash which has different configuration as well. So if we want to use the replicated A-Hash, we need a different configuration profile on each servers, called `CheloniaAllServicesReplicatedAHashWithISIS`.

Let's stop all the arched daemons, and remove everything from the /var/spool/arc directory, which clears all the metadata and files stored in the system (because the replicated A-Hash and the centralized A-Hash cannot be mixed together, and currently cannot be converted).

We need to install at least the 4.6 version of Berkeley DB, and the python bindings as well. Maybe the easiest way to install the python bindings is to use `easy_install` which is part of `python-setuptools`:

```
$ sudo easy_install bsddb3
```

If we managed to install it, we should be able to run the unittests without a problem:

```
$ sudo python /usr/lib/python2.5/site-packages/\
    bsddb3-4.8.0-py2.5-linux-i686.egg/bsddb3/tests/test_all.py
```

Let's see the configuration of the first server:

```
profile=/usr/local/share/arc/profiles/CheloniaAllServicesReplicatedAHashWithISIS.xml

pidfile=/tmp/chelonia1.pid
logfile=/var/log/chelonia1.log
debug=ERROR

port=60001

cacert=/etc/grid-security/certificates
host_cert=/etc/grid-security/hostcert-chelonia1.pem
host_key=/etc/grid-security/hostkey-chelonia1.pem
client_cert=/etc/grid-security/hostcert-chelonia1.pem
client_key=/etc/grid-security/hostkey-chelonia1.pem
```

```
[register]
index_server1=https://chelonia1.knowarc.eu:60001/ISIS

[ahash]
endpoint=https://chelonia1.knowarc.eu:60001/AHash
datadir=/var/spool/arc/chelonia1_ahash_data
peerurl1=https://chelonia1.knowarc.eu:60002/AHash

[bartender]
endpoint=https://chelonia1.knowarc.eu:60001/Bartender

[librarian]
endpoint=https://chelonia1.knowarc.eu:60001/Librarian

[shepherd]
endpoint=https://chelonia1.knowarc.eu:60001/Shepherd
turlprefix=https://chelonia1.knowarc.eu:60001/Hopi
datadir=/var/spool/arc/chelonia1_shepherd_data
storedir=/var/spool/arc/chelonia1_shepherd_store
transferdir=/var/spool/arc/chelonia1_shepherd_transfer

[isis]
endpoint=https://chelonia1.knowarc.eu:60001/ISIS
dbpath=/var/spool/arc/chelonia1_isis_data
peerurl1=https://chelonia1.knowarc.eu:60001/ISIS
peerurl2=https://chelonia2.knowarc.eu:60002/ISIS
peerurl3=https://chelonia3.knowarc.eu:60003/ISIS
```

The main difference in the config is that there is a peer URL given for the A-Hash, and three peer URL is given for the ISIS. The A-Hash is replicated not because we set a peer URL here, but because of the different profile. This profile contains a replicated A-Hash, while the previous profile contained a centralized one.

The config for `chelonia2`:

```
profile=/usr/local/share/arc/profiles/CheloniaAllServicesReplicatedAHashWithISIS.xml

pidfile=/tmp/chelonia2.pid
logfile=/var/log/chelonia2.log
debug=ERROR

port=60002

cacert=/etc/grid-security/certificates
host_cert=/etc/grid-security/hostcert-chelonia2.pem
host_key=/etc/grid-security/hostkey-chelonia2.pem
client_cert=/etc/grid-security/hostcert-chelonia2.pem
client_key=/etc/grid-security/hostkey-chelonia2.pem

[register]
index_server1=https://chelonia2.knowarc.eu:60002/ISIS

[ahash]
endpoint=https://chelonia2.knowarc.eu:60002/AHash
datadir=/var/spool/arc/chelonia2_ahash_data
peerurl1=https://chelonia1.knowarc.eu:60001/AHash

[bartender]
endpoint=https://chelonia2.knowarc.eu:60002/Bartender
```

```
[librarian]
endpoint=https://chelonia2.knowarc.eu:60002/Librarian

[shepherd]
endpoint=https://chelonia2.knowarc.eu:60002/Shepherd
turlprefix=https://chelonia2.knowarc.eu:60002/Hopi
datadir=/var/spool/arc/chelonia2_shepherd_data
storedir=/var/spool/arc/chelonia2_shepherd_store
transferdir=/var/spool/arc/chelonia2_shepherd_transfer

[isis]
endpoint=https://chelonia2.knowarc.eu:60002/ISIS
dbpath=/var/spool/arc/chelonia2_isis_data
peerurl1=https://chelonia1.knowarc.eu:60001/ISIS
peerurl2=https://chelonia2.knowarc.eu:60002/ISIS
peerurl3=https://chelonia3.knowarc.eu:60003/ISIS
```

And for `chelonia3`:

```
profile=/usr/local/share/arc/profiles/CheloniaAllServicesReplicatedAHashWithISIS.xml

pidfile=/tmp/chelonia3.pid
logfile=/var/log/chelonia3.log
debug=ERROR

port=60003

cacert=/etc/grid-security/certificates
host_cert=/etc/grid-security/hostcert-chelonia3.pem
host_key=/etc/grid-security/hostkey-chelonia3.pem
client_cert=/etc/grid-security/hostcert-chelonia3.pem
client_key=/etc/grid-security/hostkey-chelonia3.pem

[register]
index_server1=https://chelonia3.knowarc.eu:60003/ISIS

[ahash]
endpoint=https://chelonia3.knowarc.eu:60003/AHash
datadir=/var/spool/arc/chelonia3_ahash_data
peerurl1=https://chelonia1.knowarc.eu:60001/AHash
peerurl2=https://chelonia2.knowarc.eu:60002/AHash

[bartender]
endpoint=https://chelonia3.knowarc.eu:60003/Bartender

[librarian]
endpoint=https://chelonia3.knowarc.eu:60003/Librarian

[shepherd]
endpoint=https://chelonia3.knowarc.eu:60003/Shepherd
turlprefix=https://chelonia3.knowarc.eu:60003/Hopi
datadir=/var/spool/arc/chelonia3_shepherd_data
storedir=/var/spool/arc/chelonia3_shepherd_store
transferdir=/var/spool/arc/chelonia3_shepherd_transfer

[isis]
endpoint=https://chelonia3.knowarc.eu:60003/ISIS
dbpath=/var/spool/arc/chelonia3_isis_data
```

```
peerurl1=https://chelonia1.knowarc.eu:60001/ISIS
peerurl2=https://chelonia2.knowarc.eu:60002/ISIS
peerurl3=https://chelonia3.knowarc.eu:60003/ISIS
```

Now we can start all three. It may take a while for the system to stabilize, but then we will see a clean system, with no root collection. We should create one, then upload a file:

```
$ chelonia list /
'/': not found

$ chelonia mkdir /
Creating collection '/': done

$ chelonia put orange /
'orange' (7 bytes) uploaded as '/orange'.
```

After a while the file would have 3 replicas:

```
$ chelonia stat /orange
'/orange': found
  locations
    https://chelonia3.knowarc.eu:60003/Shepherd 46831763-8162-06c5-8c61-81ab8a910e28: alive
    https://chelonia1.knowarc.eu:60001/Shepherd 69e08316-093a-a788-6aea-6592c769c1de: alive
    https://chelonia2.knowarc.eu:60002/Shepherd 1332408b-55e9-54b9-f56f-0ea5d77a069b: alive
[...]
```

Now we can stop `chelonia1`, because all the services are replicated.

```
$ sudo kill `cat /tmp/chelonia1.pid`
```

Let's check the file's replicas again:

```
$ chelonia stat /orange
ERROR: No Bartender URL found.
(use the '-v' flag for more details, e.g. 'chelonia -v list /')
```

As always, let's use the `-v` flag to get more details:

```
$ chelonia -v stat /orange
- Using default user config.
- The key file: /home/zsombor/.arc/userkey-billy.pem
- The cert file: /home/zsombor/.arc/usercert-billy.pem
- The CA dir: /etc/grid-security/certificates
- No Bartender URL found in the config or in the environment, try to get one from ISIS:
[2009-10-20 16:23:44] [Arc.MCC.TLS] [ERROR] [25243/148071368] Failed to establish SSL connection
[2009-10-20 16:23:44] [Arc.MCC.TLS] [ERROR] [25243/148071368] SSL error: -1 - (empty):(empty):(empty
[2009-10-20 16:23:44] [Arc.MCC.TLS] [ERROR] [25243/148071368] Failed to send content of buffer
  - https://chelonia1.knowarc.eu:60001/ISIS - Failed to connect to ISIS.
ERROR: No Bartender URL found.
```

It says failed to connect to ISIS on the first server, which is true, because we stopped it. But in the current client configuration there is only this one ISIS specified, so let's add the others (the first three line should be in one line, separated with space)

```
defaultservices=index:ARC1:https://chelonia1.knowarc.eu:60001/ISIS
    index:ARC1:https://chelonia2.knowarc.eu:60002/ISIS
```

```
    index:ARC1:https://chelonia3.knowarc.eu:60003/ISIS
keypath=/home/zsombor/.arc/userkey-billy.pem
certificatepath=/home/zsombor/.arc/usercert-billy.pem
cacertificatesdirectory=/etc/grid-security/certificates
```

And try again:

```
$ chelonia stat /orange
'/orange': found
  locations
    https://chelonia3.knowarc.eu:60003/Shepherd 46831763-8162-06c5-8c61-81ab8a910e28: alive
    https://chelonia1.knowarc.eu:60001/Shepherd 69e08316-093a-a788-6aea-6592c769c1de: offline
    https://chelonia2.knowarc.eu:60002/Shepherd 1332408b-55e9-54b9-f56f-0ea5d77a069b: alive
[...]
```

So it is working, we have two replicas and we can download the file:

```
$ chelonia get /orange /tmp
'/orange' (7 bytes) downloaded as '/tmp/orange'.

$ cat /tmp/orange
ORANGE
```

And if we restart `chelonia1` all three replicas should be online again.

5 Accessing external storage solutions

This section was not updated.

If you have permissions to access some third-party storage solutions (e.g. dCache via the GridFTP protocol), you can create a mount point within the namespace of the ARC storage system which points to a third-party URL, and then you can use the client interface of the ARC storage to access the third-party storage as well. This is achieved by a module of the Bartender, called the 'Gateway'. Currently only GridFTP access is supported, and for that the Globus libraries needs to be installed. Next, ARC needs to be recompiled for GridFTP support. The output of the `configure` script should look like this:

```
    Available third-party features:

    RLS:                yes
    GridFTP:            yes
    LFC:                no
    RSL:                yes
    SAML:               yes
    MYSQL CLIENT LIB:   no
    gSOAP:              yes

    Included components:
    A-Rex service:      no
    ISI service:        no
    CHARON service:     no
    HOPI service:       yes
    SCHED service:      no
    STORAGE service:    yes
    PAUL service:       no
    SRM client (DMC):   yes
    GSI channel (MCC):  yes
```

After compilation and installation the `arcls` tool can be used to list the content of a third-party storage. The user Billy can see if has access to this directory:

```
$ arcls gsiftp://sal1.uppmax.uu.se:2811/pnfs/uppmax.uu.se/data
data1
data2
other
```

Since he can see some files in there, so he have permissions to list that directory.

Now, configure the Bartenders on all three machines to use the Gateway module:

```
<Service name="pythonservice" id="bartender">
    <ClassName>storage.bartender.bartender.BartenderService</ClassName>
    <LibrarianURL>[...]</LibrarianURL>
    <ProxyStore>/var/spool/arc/proxy_store</ProxyStore>
    <GatewayClass>storage.bartender.gateway.gateway.Gateway</GatewayClass>
    <GatewayCfg>
        <ProxyStore>/var/spool/arc/proxy_store</ProxyStore>
        <CACertificatesDir>/etc/grid-security/certificates</CACertificatesDir>
    </GatewayCfg>
    <ClientSSLConfig FromFile="/etc/arc/clientsslconfig.xml" />
</Service>
```

Next, start all three servers. Then create a mount point with the Logical Name `/salmount` which points to this URL:

```
$ chelonia makemount /salmount \
    gsiftp://sal1.uppmax.uu.se:2811/pnfs/uppmax.uu.se/data
- Calling the Bartender's makeMountpoint method...
- done in 3.07 seconds.
Creating mountpoint '/salmount': done

$ chelonia stat /salmount
- Calling the Bartender's stat method...
- done in 0.47 seconds.
'/salmount': found
  states
    closed: 0
  entry
    owner: /DC=eu/DC=KnowARC/O=knowarc/CN=billy
    type: mountpoint
    GUID: b21d9d22-e885-502a-ee0e-bd0a7b967543
  parents
    0/salmount: parent
  mountpoint
    externalURL: gsiftp://sal1.uppmax.uu.se:2811/pnfs/uppmax.uu.se/data
  timestamps
    created: 1237990168.09
```

List the content of this mountpoint:

```
$ chelonia list /salmount- Calling the Bartender's list method...
- done in 0.49 seconds.
'/salmount': Your proxy cannot be found. Please delegate your credentials!
```

OK, First you need to delegate your credentials with the `credentialsDelegation` method.

```
$ chelonia cre
- Calling the Bartender's credentialsDelegation method...
[...]
- done in 0.42 seconds.
Successful delegation.
Proxy ID: eNCMDmi8i6YnPSAtDmVmuSEmABFKDmABFKDm0zKKDmFBFKDmeOzdBo
```

Then try to list the mount point again:

```
$ chelonia list /salmount
- Calling the Bartender's list method...
- done in 3.19 seconds.
'/salmount': collection
  data1  <unknown>
  data2  <unknown>
  other  <unknown>
```

Now, download a file:

```
$ chelonia get /salmount/data1 /tmp
- Calling the Bartender's getFile method...
- done in 3.56 seconds.
- Got transfer URL: gsiftp://sal1.uppmax.uu.se:2811/pnfs/uppmax.uu.se/data/data1
- Downloading from 'gsiftp://sal1.uppmax.uu.se:2811/pnfs/uppmax.uu.se/data/data1'
    to '/tmp/data1' with gridftp...
   2 s:        0.0 kB       0.0 kB/s       0.0 kB/s    . . .
   3 s:        1.3 kB       0.4 kB/s       0.4 kB/s    . . .
- done in 2.7828 seconds.
'/salmount/data1' (1290 bytes) downloaded as '/tmp/data1'.
```

6 Using the ARC DMC

This section was not updated.

ARC has its own multi-protocol storage client tools which provide a unified way to access different kind of storage. For all the supported types there is a DMC module. For Chelonia, the ARC DMC module is used. If it is compiled and installed, then you can use the `arcls`, `arccp`, `arcrm`, etc. tools with the `arc://` protocol to access the ARC storage system. You can specify the URL of the Bartender with the `BartenderURL` URL option, or you can have a Bartender URL configured in the `client.xml` user config.

```
$ arcls arc:///
$ arccp orange arc:///orange
$ arcls arc:///
orange
$ arcls arc:///?BartenderURL=https://storage2:60006/Bartender
orange
$ chelonia list /
- Calling the Bartender's list method...
- done in 0.56 seconds.
'/': collection
  orange <file>
$ chelonia makeCollection /fruits
- Calling the Bartender's makeCollection method...
- done in 2.38 seconds.
Creating collection '/fruits': done
$ arccp arc:///orange arc:///fruits/orange
$ arcls arc:///
```

```
orange
fruits
$ arcls arc:///fruits
orange
$ arccp arc:///orange /tmp/orange
$ cat /tmp/orange
orange
$ arcrm arc:///orange
$ arcls arc:///
fruits
```

7 Using the FUSE module

This section was not updated.

Besides the `chelonia` tool you can access the storage system with the use of a FUSE module. The following steps installs the ARC storage FUSE module:

- Install ARC1 storage system.
- Install `fuse >= 2.7.3` and `fuse-python >= 0.2`:

  ```
  $ sudo aptitude install fuse-python
  ```

- Set up `~/.arc/client.xml`:

  ```
  $ cat ~/.arc/client.xml
  <ArcConfig>
    <KeyPath>/home/<username>/.arc/userkey.pem</KeyPath>
    <CertificatePath>/home/<username>/.arc/usercert.pem</CertificatePath>
    <CACertificatesDir>/etc/grid-security/certificates</CACertificatesDir>
    <BartenderURL>https://localhost:60000/Bartender</BartenderURL>
  </ArcConfig>
  ```

- Create a mountpoint and mount arcfs.py:

  ```
  $ pwd
  /home/me/arc
  $ ln -s arc1/src/services/storage/fuse/arcfs.py arcfs.py
  $ mkdir mnt
  $ python arcfs.py ./mnt
  ```

  This mounts ARCFS in `/home/me/arc/mnt`, creates directory `/home/me/arc/fuse_transfer` and a log file `/home/me/arc/arcfsmessages`.

- Make a collection:

  ```
  $ mkdir -p mnt/home/me
  $ stat mnt/home/me
    File: ‘mnt/home/me’
    Size: 0               Blocks: 0          IO Block: 4096   directory
  Device: 19h/25d Inode: 3           Links: 2
  Access: (0755/drwxr-xr-x)  Uid: (  500/  me)   Gid: (  100/   users)
  Access: 1970-01-01 01:00:00.000000000 +0100
  Modify: 1970-01-01 01:00:00.000000000 +0100
  Change: 1970-01-01 01:00:00.000000000 +0100
  ```

  Note that so far there is no security handling implemented, so that all files have mode 0644, all collections have mode 0755, everything is owned by the user that mounts the system.

- Create some entries in your collection:

```
$ cd mnt/home/me
$ emacs -nw fish
catfish
$ cat fish
catfish
$ cp fish fish2
$ ls -la
drwxr-xr-x 5 me users  0 1970-01-01 01:00 .
drwxr-xr-x 3 me users  0 1970-01-01 01:00 ..
-rw-r--r-- 1 me users  8 1970-01-01 01:00 fish
-rw-r--r-- 1 me users 28 1970-01-01 01:00 fish~
-rw-r--r-- 1 me users  8 1970-01-01 01:00 fish2
```

- You probably want to remove that anoying emacs backup file

```
$ rm fish~
$ ls
fish  fish2
```

- Maybe you want fish2 in a separate collection

```
$ mkdir sea_creatures
$ mv fish2 sea_creatures
```

- Maybe you're not happy with the collection name

```
$ mv sea_creatures creatures
$ ls creatures
fish2
```

- Moving the collection out of ARC storage:

```
$ cd ../../..
$ mv mnt/home/me/creatures .
```

- The collection is no longer in the ARC storage:

```
$ find mnt/home/me/
mnt/home/me/
mnt/home/me/fish
```

- Unmounting ARCFS; To unmount, refer to fuse documentation. Usually you can do

```
$ fusermount -u mnt
```

  If this for some reason this doesn't work, you can do

```
$ sudo umount -f mnt
```

  or even

```
$ sudo umount -l mnt
```

  Now, remounting the system, everything should still be there:

```
$ python arcfs.py ./mnt

$ find mnt/home/me/
mnt/home/me/
mnt/home/me/fish

$ fusermount -u mnt
```