User's Guide

Mimic

The fast file synchronization tool

Version 1.1.0

Todd King – July 01, 2016

Introduction

Mimic is a set of tools to create mirrors of file collections. It uses a combination of techniques to optimize the synchronization of the collections. Mimic can use either the SSH or FTP protocol to copy files. It supports both pushing files to remote locations and pulling files from remote locations. This means that the Mimic tools only need to be installed on one of the systems in the set of mirror hosts.

Mimic was written to address the performance issues that exist in tools like bbcp, rsync, and rcp. While these tools are widely used they are not optimized for synchronizing archives. For example, bbcp achieves high speed copying of files, but the process is blind to the state of the destination file. While bbcp does support the use of check sums using this feature slows down the copy process as the check sum is calculated on demand. With rsync only those files (blocks) which are different are copied, but the process building the files list (block maps) occurs with each invocation. For an archive with thousands or millions of files the building of the file list is very time consuming. With rcp copies are completely blind, but secure. Mimic takes the best of what bbcp, rsync and rcp offer and combines the features into the a single tool designed specifically to synchronize mirrors of collections at distributed locations.

How it works

Mimic optimizes synchronization by maintaining a file size and check sum inventory for every file in a folder tree. This inventory file is maintained in a hidden folder in the folder at the root of the tree. When synchronizing Mimic retrieves the inventory from the remote system and compares it to the local inventory. Files which are different are transferred. To optimize the refreshing of the inventory both a file size and check sum for each file are maintained in the inventory. During a refresh, if a file name is not in the inventory it is added. If a file name exists in the inventory, first file size is checked. If its different a new check sum is calculated and the inventory is updated. If the file sizes are the same the check sum is calculated and compared to the one in the inventory. If the check sums are different the inventory is updated with the new check sum. A refresh will also detect files which no longer exist, but are in the inventory. These files can be removed from the inventory.

Installation

Mimic is written in Java and so it is highly portable. To install Mimic tools download the latest distribution from http://release.igpp.ucla.edu/igpp/mimic and unpack the files into any folder you like. The organization of files in the distribution is:

```
+ api    // Documentation on the Mimic classes.
+ bin    // Helper scripts for all tools
+ jar    // Jar files for the tool.
```

+ lib // All required supporting jar files.

On Linux systems it may be necessary to mark all scripts in “bin” to executable after installation. This can be done with the command:

```
chmod +x bin/*
```

Setting up SSH

To copy between systems using the SSH protocol you will need to exchange a public key with the remote system for the user you plan to have run the mimic tools. To do this, first create a key pair:

1. Create ".ssh" directory in the user's home directory (if it does not already exist) It must be owned by the user and readable/writeable only by the user.

```
mkdir .ssh
chmod 700 .ssh
```

2. Create an RSA public/private key pair. The private key must be in the ".ssh" directory and should have the name "id_rsa".

```
cd ~/.ssh
ssh-keygen -t rsa
```

Do not enter a passphrase. This will eliminate any prompting.

3. Set permission on generated files to readable by owner only

```
chmod 600 *
```

Then copy the “rsa_id.pub” file to the remote system and add it to the “authorized_keys” file:

```
cat .ssh/id_rsa.pub | ssh -l user remote_host 'cat >> .ssh/authorized_keys'
```

Mimic looks in the user .ssh folder for all personal keys.

Getting Started

Configure a folder

1. Initialize a folder for Mimic management

```
mimic-init
```

2. Add all files and folders

```
mimic-add -r
```

recursively (-r).

Copy Mimic managed folder to a another location

1. Pull all files (from PDS-PPI)

```
mimic-clone -s -t pds-ppi -u none -i http://ppi.pds.nasa.gov/data/CO-S_SW-CAPS-5-DDR-ION-MOMENTS-V1.0
```

where the tag (-t) for this copy is “pds-ppi”, the user (-u) is “none” and the URI (-i) is “http://ppi.pds.nasa.gov/data/CO-S_SW-CAPS-5-DDR-ION-MOMENTS-V1.0”

Updating a mimic folder (source)

1. Refresh info

```
mimic-refresh
```

2. Add any new files

```
mimic-add -r
```

recursively (-r).

Updating a mimic folder (copy)

1. Synchronize with remote

```
mimic-pull
```

Usage

Mimic commands can be invoked either with the syntax:

```
mimic task
```

or with the syntax

```
mimic-task
```

See the "Commands" section for details on available tasks.

Examples

Initialize mimic management for a folder:

```
mimic-init -v
```

Add all files in the folder and sub-folders

```
mimic-add -v -r .
```

Refresh mimic management files. Updates records and removes missing items.

```
mimic-refresh -v .
```

Refresh mimic management entry for a single file. (Execute in folder and use a ./xxx relative path.)

```
mimic-refresh -v -f {filename}
```

Configure a push to a remote system:

```
mimic-config -v -d Push -t pds-archive2 -u {user} -i scp://pds-
archive.igpp.ucla.edu/pds/archive1/VOLUME/
```

Push content to remote

```
mimic-push -v
```

Configure a pull from a remote system using scp:

```
mimic-config -v -d Pull -t pds-archive2 -u {user} -i scp://pds-
archive.igpp.ucla.edu/pds/archive1/VOLUME/
```

Configure a pull from a remote system using scp, cipher and person key file:

```
mimic-config -v -d Pull -t pds-archive2 -c -k {keyfile} -u {user} -i scp://pds-
archive.igpp.ucla.edu/pds/archive1/VOLUME/
```

Pull content from remote

```
mimic-pull -v
```

Check if remote connections are possible

```
mimic-info -v -u {user} -i
```

Clone a remote (combined “init” and “config”) using scp

```
mimic-clone -v -t pds-store -u {user} -i scp://pds-
archive.igpp.ucla.edu/tmp/test
```

Clone a remote (combined “init” and “config”) using scp with cipher and personal key file

```
mimic-clone -v -t pds-store –c –k {keyfile} -u {user} -i scp://pds-
archive.igpp.ucla.edu/tmp/test
```

Create a bundle of multiple Mimic managed folders (add all managed folders in the current directory)

```
mimic-init –v .
mimic-config -v -b .
```

Sample Script to initialize and define a push destination

```
#!/bin/bash

# Initialize a set of folders
for f in VGLE*; do
   echo "Processing $f ...";
   pushd $f
   /opt/igpp/java/igpp-mimic-0.0.1/bin/mimic-init -v
   /opt/igpp/java/igpp-mimic-0.0.1/bin/mimic-add -v -r .
   /opt/igpp/java/igpp-mimic-0.0.1/bin/mimic-config -v -d Push -t pds-archive -u
{user} -i scp://pds-archive.igpp.ucla.edu/pds/archive1/VOLUME/$f
   popd
done
```

Sample script to push a set of folders

```
#!/bin/bash

# Initialize a set of folders
for f in VGLE*; do
   echo "Processing $f ...";
   pushd $f
   /opt/igpp/java/igpp-mimic-0.0.1/bin/mimic-push -v
   popd
done
```

Common Scenarios

You already have mirrored collections which you want to put under Mimic management.

On the main collection:

1. Run “mimic-add -v -r .”
2. Run “mimic-conf” to point to the remote collection.

On the remote collection:

1. Run “mimic-add -v -r .”

This will create a check sum inventory on both collections and configure the main collection to push files to the remote. When you run “mimic-push” on the main collection it will copy only those files

which need updating. This will synchronize the two collections with a minimal amount of file transfers.

The reason this works so well is that, in most cases, the generation of the check sum inventory is much quicker than transferring the files.

Adding multiple mirrors

It is possible to add multiple mirrors for a collection. To add a mirror run:

"mimic-conf" to point to the remote collection.

for each desired mirror. Then when you run "mimic-push" or "mimic-pull" (depending on the direction of the mirror) all mirrors will be updated.

Commands

mimic-add

Tool to add any new files to the Mimic managed collection.

Usage: mimic-add [options] [file …]

If no file or folder is specified the current folder is used.

Options:

-h,--help		Display this text
-m,--message	<arg>	The type of messages to output. Types are: o: valid matches; f: failed matches; m: missing files; e: error. Default: ofme
-p,--progress		Show progress at each step.
-r,--recursive		Run on folder and all sub-folders.
-t, -test		Run in test mode, do not update any files.
-v,--verbose		Show status at each step.
-x,--exclude	<arg>	A comma separated list of the prefix of files or folder names to exclude from checksum generation or testing.

Java class

igpp.mimic.Add

mimic-check

Tool to check the consistency of a Mimic managed collection.

Usage: mimic-check [options]

Options:

-h,--help		Display this text

-l,--list	<arg>	The path to the checksum file list.
-q, --quick		Check only file size to determine if file has been modified. Disables checksum calculation.
-m,--message	<arg>	The type of messages to output. Types are: o: valid matches; f: failed matches; m: missing files; e: error. Default: ofme
-n,--new		Check only for new or missing files.
-s, --summary		Provide a summary of task upon completion.
-v, --verbose		Show status at each step.
-x, --exclude	<arg>	A comma separated list of the prefix of files or folder names to exclude from checksum generation

Java class
igpp.mimic.Check

mimic-clone

Tool to clone another Mimic managed collection.

Usage: mimic-clone[options]

Options:

-h,--help		Display this text
-i,--uri	<arg>	URI for the destination host. Include protocol, host and path.
-p,--progress		Show progress at each step.
-t,--tag <arg>		Name tag for the action.
-u,--username	<arg>	The username of the account to use at the destination host.
-c --cipher		Cipher. Keep cipher (encryption) active for data transfers.
-k --keyfile		Keyfile. The name of the file containing the SSH private key.
-v,--verbose		Show status at each step.

Java class
igpp.mimic.Clone

mimic-config

Tool to configure synchronization with remote hosts.

Usage: mimic-config [options]

Options:

-b,--bundle	<arg>	Bundle. Bundle one or more folders as a set. Names of folders is passed

		as a comma separated list. A folder name of "." adds all managed folders in the current directory.
-d,--direction	<arg>	Direction. Direction to synchronize files [Push | Pull].
-h,--help		Display this text
-i,--uri	<arg>	URI. URI for the destination host. Include protocol, host and path.
-l,--list		List. Display configuration information.
-c --cipher		Cipher. Keep cipher (encryption) active for data transfers.
-k --keyfile		Keyfile. The name of the file containing the SSH private key.
-t,--tag	<arg>	Tag. Name tag for the action.
-u,--username	<arg>	Username. The username of the account to use at the destination host.
-v,--verbose		Verbose. Show status at each step.

Java class
igpp.mimic.Config

mimic-info

Tool to display information about a Mimic managed collection.

Usage: mimic-info [options]

Options:

-h,--help		Display this text
-v,--verbose		Show status at each step.

Java class
igpp.mimic.Info

mimic-init

Tool to initialize a Mimic managed collection.

Usage: mimic-info [options] [folder]

If no folder is specified the current folder is initialized.

Options:

-h,--help		Display this text
-v,--verbose		Show status at each step.

Java class
igpp.mimic.Init

mimic-pull

Tool to pull content from another Mimic managed archive.

Usage: mimic-pull [options]

Options:

-a,--all	<arg>	Push all mimic managed folders at the given path.
-h,--help		Display this text
-p,--progress		Show progress at each step.
-t,--tag	<arg>	Name tag of the push task to perform.
-v,--verbose		Show status at each step.

Java class
    igpp.mimic.Pull

mimic-push

Tool to push content to another Mimic managed archive.

Usage: mimic-push [options]

Options:

-a,--all	<arg>	Push all mimic managed folders at the given path.
-h,--help		Display this text
-p,--progress		Show progress at each step.
-s, --summary		Provide a summary of task at completion
-t,--tag	<arg>	Name tag of the push task to perform.
-v,--verbose		Show status at each step.

Java class
    igpp.mimic.Push

mimic-refresh

Tool to refresh the Mimic files to reflect the current files in the folder.

Usage: mimic-refresh [options] [folder ...]

If no folder is specified the current folder is used.

Options:

-h,--help		Display this text
-m,--message	<arg>	Message. The type of messages to output. Types are: o: valid matches; f: failed matches; m: missing files; e: error. Default: ofme
-r,--recursive		Recursive. Run on folder and all sub-folders.
-s, --summary		Provide a summary of task at completion
-t,--test		Test. Run in test mode, do not update any files.
-v,--verbose		Verbose. Show status at each step.
-x,--exclude	<arg>	Exclude. A comma separated list of the prefix of files or folder names to exclude from checksum generation or testing.

Java class
igpp.mimic.Refresh

mimic-scan

Tool to count the number files in a branch of a file system.

Usage: java igpp.mimic.Scan [options] [file ...]

If no file or folder is specified the current folder is used.

Options:
usage: igpp.mimic.Scan

-h,--help		Display this text
-c,--checksum		Scan Mimic checksum file and report on entries.
-m,--message	<arg>	The type of messages to output. Types are: o: valid matches; f: failed matches; m: missing files; e:error. Default: ofme
-n, --new		Check for any files not included in the checksum file.
-r,--recursive		Run on folder and all sub-folders.
-v,--verbose		Verbose. Show status at each step.
-x,--exclude	<arg>	Exclude. A comma separated list of the prefix of files or folder names to exclude from checksum generation or testing.

mimic-status

Tool to report on the status of a Mimic managed collection

Same as running a “mimic-check” followed by a “mimic-refresh -t”

Usage: mimic-refresh [options] [folder ...]

mimic-test

Determine information about the system and supported SSH protocol.

Usage: mimic-test [options]

Options:

-h,--help		Display this text
-n,--name	<arg>	The name of the host to connect to.
-t,--test		Test logging into to remote host.
-u,--user	<arg>	The name of the user to connect as to host.
-c --cipher		Cipher. Keep cipher (encryption) active for data transfers.
-k --keyfile		Keyfile. The name of the file containing the SSH private key.
-v,--verbose		Show status at each step.

Java class
igpp.mimic.Test

Troubleshooting

FileNotFoundException
When connecting or testing for remote access you may receive the message:

```
Error: java.io.FileNotFoundException: /user/dda/.ssh/id_rsa (No such file or
directory)
```

This indicates that SSH certificates have not been created for the local account. To create a certificate do the following:

1. Create ".ssh" directory in the user's home directory (if it does not already exist) It must be owned by the user and readable/writeable only by the user.

```
mkdir .ssh
chmod 700 .ssh
```

2. Create an RSA public/private key pair. The private key must be in the ".ssh" directory and should have the name "id_rsa".

```
cd ~/.ssh
ssh-keygen -t rsa
```

Do not enter a passphrase. This will eliminate any prompting.

3. Set permission on generated files to readable by owner only

```
chmod 600 *
```

A push or pull fails
Make sure the public key for the account you are using is registered with the remote system. To add a public key to the remote systems “authorized_keys” file do the following

```
cat ~/.ssh/id_rsa.pub | ssh -l user2 computer2 'cat >> .ssh/authorized_keys'
```

You get the error "com.jcraft.jsch.JSchException: session is down"
One of the ways mimic optimizes transfers is to turn encryption off after authentication. This is fine for public data where encryption is not necessary. However, a systems may require that encryption always be active. To correct the problem turn encryption on using the "withCypher" (-c) option in config.

Revision History

First Draft – January 27, 2014; Todd King

Minor updates – April 16, 2014; Todd King

Add "session is down" to Troubleshooting – July 01, 2016; Todd King