Joe Werne, CoRA/NWRA
Chris Bizon CoRA/NWRA
Michael Gourlay CoRA/NWRA

GWD-I
CoRA/NWRA
September 12, 2001

`archive`: A facility-independent tool for dealing with archival data systems.

September 12, 2001

Status of this memo: Informational

Abstract

With the increased availability of multiple, scalable, high-performance computing systems at U. S. national supercomputer facilities, end users now face increasingly complex computing tasks that have little or nothing to do with the scientific applications that motivate their work. One such task is the real-time data migration of large data sets to and from archival storage, which is further complicated by syntax for common tasks that varies from facility to facility.

As part of our work on the Practical Supercomputing Toolkit (see www.pstoolkit.org), we are implementing a utility to remove some of this burden from the user by providing a script, `archive`, that provides a facility-independent interface method for the user to access archival storage.

Contents

1 The Problem: Management of archival data is difficult for end users. **2**

2 The Solution: Provide a facility independent method of dealing with data archiving. **2**
2.1 The Practical Supercomputing Toolkit (PST) 2
2.2 `archive` 3

3 Author Contact Information **4**

4 Acknowledgments **4**

5 Glossary **4**

A `archive` man page **5**

1 The Problem: Management of archival data is difficult for end users.

Difficulties in the use of massively parallel supercomputers result from the size of the resultant data sets combined with the persistent lagging of system- and archival-storage-disk and network-infrastructure capabilities behind machine flop rates. As a result, instead of waiting for a massively parallel application to complete before migrating data to archival storage (a strategy which would result in filled system disks), I/O-intensive applications must archive data in real time. In some cases data must then be intentionally migrated off-line to slow media in the archival system, lest the main archival storage disks fill. This complicates even the simplest post-processing operations (i.e., operations not anticipated during code execution) as data must be migrated back to the supercomputer system disks for subsequent interrogation.

Data archiving and migration must be performed as data are computed, and jobs are controlled by scripts which may run at any time, based on a queuing system. Therefore, data archiving and migration must also be script-controllable. This aspect of scripting is made unnecessarily complicated by the fact that each facility has its own method of data archiving, and its own interface to the data archive. For instance, putting a file on the data archive may be performed through remote shell commands (rsh, rcp), through an ftp-style interface (pftp), or through other facility-specific commands (far put, hsi put, msfput). This means either that scripts that deal with data migration are often facility-dependent, and therefore costly to create and maintain.

Add to this the fact that many end users are graduate-student or post-doctoral researchers in the mathematical or physical sciences who are often neither trained nor proficient at elaborate script development, and one can clearly see that massively parallel supercomputing can become uninteresting drudgery that robs researchers of the time needed to answer scientific questions.

2 The Solution: Provide a facility independent method of dealing with data archiving.

2.1 The Practical Supercomputing Toolkit (PST)

In response to similar problems, we have developed a set of utilities that simplifies the tasks of developing and maintaining complex job-preparation, execution, and post-processing scripts at all of the centers we use. The Practical Supercomputing Toolkit (www.pstoolkit.org) distills our experience in using supercomputing facilities into open-source cross-facility utilities. PST is broken into several layers; the lowest layer of PST is the unification layer, Tools for Uniform Supercomputing (TUSC). Its intention is to conceal differences between facilities for common tasks such as job submission and data archiving. Essentially, TUSC applications provide a uniform command line interface for performing similar

tasks at different facilities. Scripts can now be written using TUSC applications rather than facility-dependent commands, making the scripts portable, and increasing the productivity of the end user. Further, because TUSC commands can be run interactively, users are relieved from ever learning facility-specific commands.

2.2 `archive`

`archive` is a TUSC-level utility of PST that provides an interface between the end-user and a facility's particular method for archiving and migrating data. It allows users to add, retrieve, and remove files from the archive, migrate or unmigrate them to slower media, list files in the archive, and change an archived file's mode. In other words, it performs all the tasks that are usually allowed through a particular facility's archiving utilities, but through a facility-independent interface. See the appendix for the man page to `archive`, which lists its full functionality.

We have an experimental version of `archive` that is usable at several Department of Defense centers (AHPCRC, ERDC, NAVO, and ASC) as well as one National Science Foundation center (PSC).

Archive is implemented in Perl and has two layers. The top layer is facility independent and handles command line parsing, multiple retries, and any other logic. In addition it makes calls to the second layer, which contains a number of very short Perl functions whose implementations are facility-dependent. These local functions must be implemented for each center, but usually call a single facility-dependent system command, and therefore typically comprise only a single line of actual code. Implementation and maintenance of the local functions, then, are meant to simple, with any complex logic residing in the facility-independent layer of `archive`.

The implementation of local functions is further aided by the inclusion of user-submitted local functions to the `archive` distribution for use as examples. Current implementations of the local layer deal with archives that are NFS mountable, those that use samFS commands via remote shell, and some that use commands developed at a particular facility.

With `archive` and the other TUSC-level applications, users will be able to write job scripts that are completely facility independent. This should allow large productivity increases by first removing the initial learning curve encountered when beginning work on a new center, and second, replacing the maintenance of many facility-dependent scripts with a single facility-independent script.

`archive` abstracts the process of dealing with a particular center's data archive while at that center, but does not directly address the need to access one facility's archive from another facility. It would be, however, a natural expansion to `archive`; such functionality would most likely be implemented using the GridFTP protocol (see www.globus.org/datagrid/gridftp.html).

3 Author Contact Information

Joe Werne
Northwest Research Associates / Colorado Research Associates Division
Boulder, CO 80301
werne@co-ra.com

www.pstoolkit.org contains the man pages for both `archive` and a tool for similarly abstracting the process of job submission (`qprep`), as well as the implementation of `archive` and pointers to mailing lists for PST users and developers.

4 Acknowledgments

Our work on PST is currently funded by the Department of Defense High Performance Computing Modernization Office. (?). In addition to those people listed on this document, significant contributions have been made to PST by Joseph Robichaux, Ray Sheppard, and John Skinner of NAVO and David Sanders of ERDC.

5 Glossary

- AHPCRC: Army High Performance Computer Research Center.
- ASC: Aeronautical Systems Center.
- `archive`: A utility that provides a common command line interface for archiving data at supercomputing facilities. `archive`is part of the TUSC layer of PST.
- data archive: A long-term storage system, usually used to hold the typically large results of computations on a supercomputer.
- ERDC: U.S. Army Engineer Research Development Center.
- local: Related to a particular facility.
- migration: The act of moving data that is has already been archived to a still-slower form of media.
- NAVO: Naval Oceanographic Office.
- PSC: Pittsburgh Supercomputer Center.
- PST: The Practical Supercomputing Toolkit. A collections of utilities that allow users to more productively manage supercomputing resources.
- TUSC: Tools for Uniform Supercomputing. A layer of PST that focuses on hiding facility-dependent details of common tasks such a job submission or data archiving.

A `archive` man page

NAME

archive — perform basic file-handling operations on the local archival storage system.

SYNOPSIS

S<**archive put** [putopts] *file1* [*file2 ...*] >
S<**archive get** [getopts] *file1* [*file2 ...*] >
S<**archive mv**∗ [mvopts] *file1* [*file2 ...*] *target* >
S<**archive rm**∗ [rmopts] *file1* [*file2 ...*] >
S<**archive ls** [lsopts] [*file1 ...*] >
S<**archive stat** [statopts] [*name*] >
S<**archive mkdir** [mkdiropts] *dir1* [*dir2 ...*] >
S<**archive rmdir**∗ [rmdiropts] *dir1* [*dir2 ...*] >
S<**archive chmod**∗ [chmodopts] *mode file1* [*file2 ...*] >
S<**archive chgrp**∗ [chgrpopts] group *file1* [*file2 ...*] >
S<**archive chown**∗ [chownopts] owner[:group] [*file1 ...*] >
S<**archive migrate**∗ -(M|u|q) [migrateopts] *file1 file2 ...*] >

DESCRIPTION

archive provides users with a uniform, site-independent command interface to the local archival storage system, eliminating the need to learn the myriad syntactical variations that appear at different institutions to perform essentially identical archival file management tasks.

COMMANDS

Each instance of **archive** performs the file-management command identified by **archive**'s first argument: **put**, **get**, **mv**∗, **rm**∗, **ls**, **stat**, **mkdir**, **rmdir**∗, **chmod**∗, **chgrp**∗, or **chown**∗. Each command accepts a set of arguments as well as a set of command-line options described in the **OPTIONS** section below. If multiple archival storage facilities are available at a particular site, all of the commands act on the the default system only, unless another system is specified with the S<**-asf** *host*> option. The detailed behavior of the different commands is as follows:

∗ The current version of **archive** does not support the commands indicated with asterisks (∗). Later versions will incorporate these additional commands.

S<archive put [putopts] *file1* [*file2 ...*]>

Copy files *file1* [*file2 ...*] from the current working directory to the archival storage system. Options *putopts* include **-retry N**, S<**-asf** *host*>,

S<**-C** *path*>, **-s**, **-f**, **-i**, **-p**, **-M**, **-D**, **-n**, S<**-t** *tarfile*>, **-S**, and **-z**. *file1* S<[*file2 ...*]> may contain wildcards ∗ and ?.

The destination directory on the archival storage system may be specified using the S<**-C** *path*> option. If S<**-C** *path*> is not present, the user's archival home directory will be used.

Multiple files are combined with *tar*(1) before copying to the archival storage system if the S<**-t** *tarfile*> option is specified. Files are tarred by first executing

tar cf — run10 run10.001, ..., run10.025 ¿ *tarfile*

then *tarfile* is transferred to the archival storage system.

S<archive get [getopts] *file1* [*file2 ...*] >

Copy files *file1* [*file2 ...*] in the archival storage system to the current working directory. Options *getopts* include **-retry N**, S<**-asf** *host*>, S<**-C** *path*>, **-s**, **-f**, **-i**, **-p**, **-M**, **-n**, **-S**, **-x**, and **-z**. If wildcards ∗ and ? are used in specifying *file1* [*file2 ...*], then double quotes must be used:

S<archive get "*file1* [*file2 ...*]">

A source directory on the archival storage system may be specified using the S<**-C** *path*> option. If S<**-C** *path*> is not specified, the user's archival home directory will be used.

Archival storage files with a .tar, .tar.gz, or .tgz extension will be extracted with the **gzip** and **tar** utilities if the **-x** option is present. For example, if *filea* is named run10.tgz, then **archive get -x** will copy run10.tgz from the archival storage system, then execute

gunzip **-f** ¡ run10.tgz | tar xf -

Specifying **-x** has no effect on files with extensions different from .tar, .tar.gz, or .tgz.

S<archive mv∗ [mvopts] *file1* [*file2 ...*] *target*>

Move *file1* [*file2 ...*] to the target location *target*; *file1* [*file2 ...*] and *target* are all located on the archival storage system. Options *mvopts* include **-retry N**, S<**-asf** *host*>, S<**-C** *path*>, **-s**, **-f**, **-i**, **-M**, and **-n**. If *target* is an existing directory, then *file1* [*file2 ...*] are moved into *target*. If *target* does not exist, S<**archive mv**> will attempt to rename *file1* as *target*. If in this case [*file2 ...*] is present or the names *file1* and *target* are identical, **archive mv** will fail and write an error message to standard error. If wildcards ∗ and ? are used to specify *file1* [*file2 ...*] and *target*, then double quotes must be used:

S<archive mv "*file1* [*file2 ...*]" "*target*">

Note: do not include both *file1* [*file2 ...*] and *target* in the same set of quotes. If wildcards are used to specify *file1* [*file2 ...*] or *target*, but not both, then quotes need only delineate the argument containing wildcards.

S<archive rm∗ [rmopts] *file1* [*file2 ...*] >

Remove *file1* [*file2 ...*] on the archival storage system. Options *rmopts* include **-retry N**, S<**-asf** *host*>, S<**-C** *path*>, **-s**, **-f**, **-i**, **-r**, and **-n**. If wildcards ∗ and ? are used to specify *file1* [*file2 ...*], then double quotes must be used:

S<archive rm "*file1* [*file2 ...*]">

When **-r** is specified, directories are emptied of files and removed. The user is prompted for removal of any write-protected files, except when the **-f** option is used.

S<archive ls [lsopts] [*file1 ...*] >

List [*file1 ...*] on the archival storage system. Options *lsopts* include **-retry N**, S<**-asf** *host*>, and other options identified by the manual page for **ls** on the archival storage system. (See **man archive ls(3)** for a full list of *lsopts*.) If wildcards ∗ and ? are used to specify [*file1 ...*], then double quotes must be used:

S<archive ls "[*file1 ...*]">

If [*file1 ...*] is not specified, then the contents of the user's home directory on the archival storage system are listed.

S<archive stat [statopts] >

Check and report status of archival storage system. Options *statopts* include **-retry N**, S<**-asf** *host*>, and **-s**. An exit status of 0 is returned when the system is operational; a non-zero exit status is returned otherwise. A message of the form

hh:mm:ss mm/dd/yyyy *name status* [*note*]

is printed to standard out, where hh:mm:ss mm/dd/yyyy is the current date, *name* is the name of the archival storage system, *status* is either "on-line" or "unavailable", and *note* is an optional text message that may or may not be returned.

S<archive mkdir [mkdiropts] *dir1* [*dir2 ...*] >

Create directories *dir1* [*dir2 ...*] on the archival storage system. Options *mkdiropts* include **-retry N**, S<**-asf** *host*>, S<**-C** *path*>, **-s**, **-p**, S<**-m** *mode*>, and **-n**.

S<archive rmdir∗ [rmdiropts] *dir1* [*dir2 ...*] >

Remove empty directories *dir1* [*dir2 ...*] from the archival storage system. Options *rmdiropts* include **-retry N**, S<**-asf** *host*>, S<**-C** *path*>, **-s**, **-p**, and **-n**. If *dir1* [*dir2 ...*] are not empty, **archive rmdir** will fail and print an error message to standard error. If wildcards ∗ and ? are used to specify *dir1* [*dir2 ...*], then double quotes must be used:

S<archive rmdir "*dir1* [*dir2 ...*]">

S<archive chmod∗ [chmodopts] *mode file1* [*file2 ...*]>

Change permissions for *file1* [*file2 ...*] according to *mode*. Options *chmodopts* include **-retry N**, S<**-asf** *host*>, **-C** *path*, **-s**, **-R**, and **-n**. *mode* may be either symbolic or absolute. See discussion of *mode* syntax in the **OPTIONS** section under S<**-m** *mode*>.

S<archive chgrp∗ [chgrpopts] group *file1* [*file2 ...*]>

Change the group ID of *file1* [*file2 ...*] to *group*, which may be a decimal group ID or a group name found in the group file. Options *chgrpopts* include **-retry N**, S<**-asf** *host*>, **-C** *path*, **-s**, **-f**, **-h**, **-R**, and **-n**. Only an appropriately authorized user may change the group of a file that is owned by another user. Unless users are appropriately authorized, they must be a member of the specified group to change the group of a file. Unless the user is appropriately authorized, **archive chgrp** clears the set-user-ID and set-group-ID file mode bits.

S<archive chown∗ [chownopts] owner[:group] *file1* [*file2 ...*]>

Change the owner of *file1* [*file2 ...*] to *owner*; The optional *group* argument changes the group. Options *chownopts* include **-retry N**, S<**-asf** *host*>, **-C** *path*, **-s**, **-f**, **-h**, **-R**, and **-n**. Only an appropriately authorized user may change the owner of a file. Unless the user is appropriately authorized, **archive chown** clears the set-user-ID and set-group-ID file mode bits.

S<archive migrate∗ -(M|u|q) [migrateopts] *file1* [*file2 ...*] >

Control file migration of *file1* [*file2 ...*]; Options *migrateopts* include **-retry N**, S<**-asf** *host*>, **-C** *path*, **-s**, **-h**, and **-n**. archive migrate must receive one and only one of the options -M, -u, and -q. -M migrates a remote file or set of files. -u unmigrates a remote file or set of files, and -q prints a message indicating whether or not the *file*(s) are migrated.

OPTIONS

By default, **archive** assumes all arguments beginning with a dash (−) are command-line options. Therefore, when **archive** encounters an input item of the form -?, where ? does not match any of **archive**'s valid options, **archive** will print a message to standard error and return a non-zero exit status. Because this default behavior does not allow the user to specify file or directory names that begin with a dash, you can override this default behavior by including a dash-dash (−−) in the argument list. When **archive** encounters a dash-dash, it stops parsing command options and assumes all following input items are command arguments.

∗ archive currently does not support the options identified below with asterisks (∗). There are two acceptions: **-f** is assumed, but sufficient error checking to enforce **-f** for all possible errors is not currently enabled. Also, S<**-m** *mode*> currently only handles asolute (as opposed to sybmolic) mode specification. Later versions of archive will include all of the options and behavior listed below.

-C ***path***
Manage files in the archival storage directory *path*. *path* is the user's archival storage home directory by default.

-M Once a file has been written or accessed on the archival storage system, migrate the file to offline media. Some centers do not support this operation. This option may prove useful for users with numerous large data files to move, which, if moved all at once, would threaten to fill the first tier of the archival storage system. By specifying **-M** the user immediately makes room for subsequent archival data transfers.

-R∗
Execute command recursively in all directories and subdirectories in the argument list. Same as **-r**. **archive chmod** only takes **-R**, since **-r** can have a different meaning for this command.

-asf ***host***
Archival Storage Facility: execute command on the archival storage facility *host*. This option overrides the default specified by the $ARCHIVE_HOST environment variable.

-D Delete the local copy of source files involved in data transfer. If not set, all local source files are retained. Note, "source files" refers to files before they have been tarred with the **-t** option or local files that are not tarred before transfer. "Source files" are not the intermediate tarfiles generated when **-t** is envoked. Intermediate tarfiles generated with **-t** are deleted by default upon successful transfer. To override this default behavior and retain intermediate tarfiles, use **-S**.

-S Save local copies of transferred tarfiles.

-f*

Force creation, deletion, or modification of files/directories, possibly overriding permissions set on existing files/directories owned by the user. When **-f** is not present, the user is prompted interactively to override permission settings on files owned by the user. Regardless the presence of **-f**, **archive** will not attempt to change permissions on files or directories not owned by the user. When used in conjunction with **archive chgrp** or **archive chown**, the **-f** option is identical to **-s**.

-h*

If the file is a symbolic link, change attributes of the symbolic link. If **-h** is not used, attributes of the file referenced by the symbolic link are changed.

-i*

Force interactive confirmation by the user before overwriting or removing an existing file or directory, both on the archival storage system and local disks. If **-i** is not set and permission settings permit, **archive** will overwrite or remove existing files/directories.

-m ***mode****

Specify the permissions for the directory or file according to *mode*. The syntax for *mode* described here is also used by the **archive chmod** command.

File *mode*s may be either symbolic or absolute. Absolute changes to modes are specified with octal numbers, with *mode* of the form *nnn*. Here *n* is a number from 0 to 7.

An absolute mode is given as an octal number constructed from the "OR" of the following modes:

Code

Description

400

Sets read permission for owner.

200

Sets write permission for owner.

100

Sets execute (search in directory) permission for owner.

070

Sets read, write, and execute (search in directory) permission for group.

007

Sets read, write, and execute (search in directory) permission for others.

Symbolic changes to modes are specified with mnemonic characters, with *mode* of the form

[*who*] *operator* [*permissions*]

Note: if *who* is specified, there should be no space between *who* and *operator*; similarly, if *permissions* is specified, there should be no space between *operator* and *permissions*. Here *who* is one or more characters that corresponds to user, group, other, or all (**u**, **g**, **o**, or **a**); *operator* is +, -, or =, signifying the addition, removal, or absolute assignment of permissions; and *permissions* is one or more characters that correspond to the type of permission, e.g., read (**r**), write (**w**), execution (**x**) for a given file. For example, to add read and write permission for group, *mode* should be g+rw.

Multiple symbolic modes separated by commas can be specified, although no spaces may intervene between these modes. Operations are performed in the order given. Multiple symbolic letters following a single operator cause the corresponding operations to be performed simultaneously.

By default *mode* is 777 (a+rwx) minus the bits set in the umask on the archival storage system.

-n*
Execute **archive** in background mode and return zero exit status. If **-i** is set, **-n** is ignored.

-p Create all non-existent parent directories in a specified archival storage system path. By default, if the parent directory of a target pathname does not exist, **archive** will fail and write an output error message to standard error.

-q Query the migration status of a remote file. Used with **archive migrate**.

-r Execute command recursively in all directories and subdirectories in the argument list. Same as **-R**. **archive chmod** only takes **-R**, since **-r** has a different meaning for this command.

-retry N
If **archive** fails, retry **N** times, with increasing wait time between repeated attempts. Currently **archive** sleeps for 5 seconds plus two seconds for every iteration attempt. Default is **-retry 1000**.

-s Suppress printing of any output messages to standard error or standard out, except for usage errors.

-t *tarfile*
This option handles archival storage files created with *tar*(1). See **DESCRIPTION** section for **archive put** above for discussion of the **-t** option.

-u Unmigrate a remote file. Used with **archive migrate**.

-x Extract (possibly gzipped) tarfiles after transfer from archival storage with **archive get**.

-z Use *gzip*(1) to compress/uncompress files transfered to/from archival storage. The compressed file will have a .gz extension. If **-t** is also set, then the output from *tar*(1) will be piped to *gzip*(1), and the resulting compressed tar file will have a .tgz extension. See **EXAMPLES** section below.

DIAGNOSTICS

EXAMPLES

ENVIRONMENT

The following environment variables are checked by **archive**: $ARCHIVE_HOST, $ARCHIVE_HOME, and $ARCHIVE_PROBE. If set, **archive** overrides default settings established during installation.

Each environment variable is a comma-separated list (with no white space), representing multiple archival storage facilities. Currenlty all **archive** commands operate on only one system and use the first element of each list, which is interpretted by **archive** as the default. A different system may be specified with the **-asf** option. In the future, **archive get**, **stat**, and **ls** will be modified to operate on multiple systems, sequencing through the entries in the lists for these environment variables.

$ARCHIVE_HOST
List of archival storage systems available at this site. S<**-asf** *host*> overrides this environment variable.

$ARCHIVE_HOME
List of archival storage system home directories associated with $ARCHIVE_HOST. $ARCHIVE_HOME must contain either only one element, which is used for all elements of $ARCHIVE_HOME, or a separate element for each system specified by $ARCHIVE_HOST.

$ARCHIVE_PROBE
List of flags ("yes" or "no") that indicate whether or not **archive** will attempt to check the existence, type, and size of files on the archival storage system before and after transfer so that error checking is possible. A reason for setting $ARCHIVE_PROBE="no" could be failure in accurate or complete reporting of file information by the archival storage system.

FILES

NOTES

archive is a command-line translation tool in the TUSC layer of the Practical

Supercomputing Toolkit (PST). By using **archive** (and other elements of PST), users develop scripts and techniques that work, without modification, at all institutions supported by PST.

Centers currently supported include ERDC, NAVO, ASC, AHPCRC.

SEE ALSO

archive put(3), *archive get*(3), *archive mv*(3)∗, *archive rm*(3)∗, *archive ls*(3), *archive stat*(3), *archive mkdir*(3), *archive rmdir*(3)∗, *archive chmod*(3)∗, *archive chgrp*(3)∗, *archive chown*(3)∗.

AUTHOR

Written and Copyright (C) 2001 by Joseph Werne and Michael Gourlay. www.pstoolkit.org.

HISTORY

Documentation v1.0 assembled Friday, July 13, 2001.