DLMFS

A minimal DLM userspace interface implemented via a virtual file system.

dlmfs is built with OCFS2 as it requires most of its infrastructure.

Project web page: http://ocfs2.wiki.kernel.org
Tools web page: https://github.com/markfasheh/ocfs2-tools
OCFS2 mailing lists: https://oss.oracle.com/projects/ocfs2/mailman/

Credits

Some code taken from ramfs which is Copyright © 2000 Linus Torvalds and Transmeta Corp.

Mark Fasheh <mark.fasheh@oracle.com>

Caveats

- Right now it only works with the OCFS2 DLM, though support for other DLM implementations should not be a major issue.

Mount options

None

Usage

If you're just interested in OCFS2, then please see ocfs2.txt. The rest of this document will be geared towards those who want to use dlmfs for easy to setup and easy to use clustered locking in userspace.

Setup

dlmfs requires that the OCFS2 cluster infrastructure be in place. Please download ocfs2-tools from the above url and configure a cluster.

You'll want to start heartbeating on a volume which all the nodes in your lockspace can access. The easiest way to do this is via ocfs2_hb_ctl (distributed with ocfs2-tools). Right now it requires that an OCFS2 file system be in place so that it can automatically find its heartbeat area, though it will eventually support heartbeat against raw disks.

Please see the ocfs2_hb_ctl and mkfs.ocfs2 manual pages distributed with ocfs2-tools.

Once you're heartbeating, DLM lock 'domains' can be easily created / destroyed and locks within them accessed.

Locking

Users may access dlmfs via standard file system calls, or they can use 'libo2dlm' (distributed with ocfs2-tools) which abstracts the file system calls and presents a more traditional locking api.

dlmfs handles lock caching automatically for the user, so a lock request for an already acquired lock will not generate another DLM call. Userspace programs are assumed to handle their own local locking.

Two levels of locks are supported - Shared Read, and Exclusive. Also supported is a Trylock operation.

For information on the libo2dlm interface, please see o2dlm.h, distributed with ocfs2-tools.

Lock value blocks can be read and written to a resource via read(2) and write(2) against the fd obtained via your open(2) call. The maximum currently supported LVB length is 64 bytes (though that is an OCFS2 DLM limitation). Through this mechanism, users of dlmfs can share small amounts of data amongst their nodes.

mkdir(2) signals dlmfs to join a domain (which will have the same name as the resulting directory)

rmdir(2) signals dlmfs to leave the domain

Locks for a given domain are represented by regular inodes inside the domain directory. Locking against them is done via the open(2) system call.

The open(2) call will not return until your lock has been granted or an error has occurred, unless it has been instructed to do a trylock operation. If the lock succeeds, you'll get an fd.

open(2) with O_CREAT to ensure the resource inode is created - dlmfs does not automatically create inodes for existing lock resources.

Open Flag	Lock Request Type
O_RDONLY	Shared Read
O_RDWR	Exclusive

Open Flag	Resulting Locking Behavior
O_NONBLOCK	Trylock operation

You must provide exactly one of O_RDONLY or O_RDWR.

If O_NONBLOCK is also provided and the trylock operation was valid but could not lock the resource then open(2) will return ETXTBUSY.

close(2) drops the lock associated with your fd.

Modes passed to mkdir(2) or open(2) are adhered to locally. Chown is supported locally as well. This means you can use them to restrict access to the resources via dlmfs on your local node only.

The resource LVB may be read from the fd in either Shared Read or Exclusive modes via the read(2) system call. It can be written via write(2) only when open in Exclusive mode.

Once written, an LVB will be visible to other nodes who obtain Read Only or higher level locks on the resource.

See Also

http://opendlm.sourceforge.net/cvsmirror/opendlm/docs/dlmbook_final.pdf

For more information on the VMS distributed locking API.