gfs2-glocks.txt

Glock internal locking rules

This documents the basic principles of the glock state machine internals. Each glock (struct gfs2_glock in fs/gfs2/incore.h) has two main (internal) locks:

1. A spinlock (gl_spin) which protects the internal state such as gl_state, gl_target and the list of holders (gl_holders)
2. A non-blocking bit lock, GLF_LOCK, which is used to prevent other threads from making calls to the DLM, etc. at the same time. If a thread takes this lock, it must then call run_queue (usually via the workqueue) when it releases it in order to ensure any pending tasks are completed.

The gl_holders list contains all the queued lock requests (not just the holders) associated with the glock. If there are any held locks, then they will be contiguous entries at the head of the list. Locks are granted in strictly the order that they are queued, except for those marked LM_FLAG_PRIORITY which are used only during recovery, and even then only for journal locks.

There are three lock states that users of the glock layer can request, namely shared (SH), deferred (DF) and exclusive (EX). Those translate to the following DLM lock modes:

Glock mode	DLM lock mode	
UN	IV/NL	Unlocked (no DLM lock associated with glock) or NL
SH	PR	(Protected read)
DF	CW	(Concurrent write)
EX	EX	(Exclusive)

Thus DF is basically a shared mode which is incompatible with the "normal" shared lock mode, SH. In GFS2 the DF mode is used exclusively for direct I/O operations. The glocks are basically a lock plus some routines which deal with cache management. The following rules apply for the cache:

Glock mode	Cache data	Cache Metadata	Dirty Data	Dirty Metadata
UN	No	No	No	No
SH	Yes	Yes	No	No
DF	No	Yes	No	No
EX	Yes	Yes	Yes	Yes

These rules are implemented using the various glock operations which are defined for each type of glock. Not all types of glocks use all the modes. Only inode glocks use the DF mode for example.

Table of glock operations and per type constants:

Field	Purpose
go_xmote_th	Called before remote state change (e.g. to sync dirty data)
go_xmote_bh	Called after remote state change (e.g. to refill cache)
go_inval	Called if remote state change requires invalidating the cache

go_demote_ok	Returns boolean value of whether its ok to demote a glock (e.g. checks timeout, and that there is no cached data)
go_lock	Called for the first local holder of a lock
go_unlock	Called on the final local unlock of a lock
go_dump	Called to print content of object for debugfs file, or on error to dump glock to the log.
go_type	The type of the glock, LM_TYPE_.....
go_min_hold_time	The minimum hold time

The minimum hold time for each lock is the time after a remote lock grant for which we ignore remote demote requests. This is in order to prevent a situation where locks are being bounced around the cluster from node to node with none of the nodes making any progress. This tends to show up most with shared mmaped files which are being written to by multiple nodes. By delaying the demotion in response to a remote callback, that gives the userspace program time to make some progress before the pages are unmapped.

There is a plan to try and remove the go_lock and go_unlock callbacks if possible, in order to try and speed up the fast path though the locking. Also, eventually we hope to make the glock "EX" mode locally shared such that any local locking will be done with the i_mutex as required rather than via the glock.

Locking rules for glock operations:

Operation	GLF_LOCK bit lock held	gl_spin spinlock held
go_xmote_th	Yes	No
go_xmote_bh	Yes	No
go_inval	Yes	No
go_demote_ok	Sometimes	Yes
go_lock	Yes	No
go_unlock	Yes	No
go_dump	Sometimes	Yes

N.B. Operations must not drop either the bit lock or the spinlock if its held on entry. go_dump and do_demote_ok must never block. Note that go_dump will only be called if the glock's state indicates that it is caching uptodate data.

Glock locking order within GFS2:

1. i_mutex (if required)
2. Rename glock (for rename only)
3. Inode glock(s)
   (Parents before children, inodes at "same level" with same parent in lock number order)
4. Rgrp glock(s) (for (de)allocation operations)
5. Transaction glock (via gfs2_trans_begin) for non-read operations
6. Page lock (always last, very important!)

There are two glocks per inode. One deals with access to the inode itself (locking order as above), and the other, known as the iopen glock is used in conjunction with the i_nlink field in the inode to determine the lifetime of the inode in question. Locking of inodes

is on a per-inode basis. Locking of rgrps is on a per rgrp basis.