RCU Torture Test Operation

CONFIG_RCU_TORTURE_TEST

The CONFIG_RCU_TORTURE_TEST config option is available for all RCU implementations. It creates an rcutorture kernel module that can be loaded to run a torture test. The test periodically outputs status messages via printk(), which can be examined via the dmesg command (perhaps grepping for "torture"). The test is started when the module is loaded, and stops when the module is unloaded.

CONFIG_RCU_TORTURE_TEST_RUNNABLE

It is also possible to specify CONFIG_RCU_TORTURE_TEST=y, which will result in the tests being loaded into the base kernel. In this case, the CONFIG_RCU_TORTURE_TEST_RUNNABLE config option is used to specify whether the RCU torture tests are to be started immediately during boot or whether the /proc/sys/kernel/rcutorture_runnable file is used to enable them. This /proc file can be used to repeatedly pause and restart the tests, regardless of the initial state specified by the CONFIG_RCU_TORTURE_TEST_RUNNABLE config option.

You will normally -not- want to start the RCU torture tests during boot (and thus the default is CONFIG_RCU_TORTURE_TEST_RUNNABLE=n), but doing this can sometimes be useful in finding boot-time bugs.

MODULE PARAMETERS

This module has the following parameters:

fqs_duration	Duration (in microseconds) of artificially induced bursts of force_quiescent_state() invocations. In RCU implementations having force_quiescent_state(), these bursts help force races between forcing a given grace period and that grace period ending on its own.
fqs_holdoff	Holdoff time (in microseconds) between consecutive calls to force_quiescent_state() within a burst.
fqs_stutter	Wait time (in seconds) between consecutive bursts of calls to force_quiescent_state().
irqreaders	Says to invoke RCU readers from irq level. This is currently done via timers. Defaults to "1" for variants of RCU that permit this. (Or, more accurately, variants of RCU that do -not- permit this know to ignore this variable.)
nfakewriters	This is the number of RCU fake writer threads to run. Fake writer threads repeatedly use the synchronous "wait for current readers" function of the interface selected by torture_type, with a delay between calls to allow for various different numbers of writers running in parallel. nfakewriters defaults to 4, which provides enough parallelism to trigger special cases caused by multiple writers, such as

```
                the synchronize_srcu() early return optimization.

nreaders        This is the number of RCU reading threads supported.
                The default is twice the number of CPUs.  Why twice?
                To properly exercise RCU implementations with preemptible
                read-side critical sections.

shuffle_interval
                The number of seconds to keep the test threads affinitied
                to a particular subset of the CPUs, defaults to 3 seconds.
                Used in conjunction with test_no_idle_hz.

stat_interval   The number of seconds between output of torture
                statistics (via printk()).  Regardless of the interval,
                statistics are printed when the module is unloaded.
                Setting the interval to zero causes the statistics to
                be printed -only- when the module is unloaded, and this
                is the default.

stutter         The length of time to run the test before pausing for this
                same period of time.  Defaults to "stutter=5", so as
                to run and pause for (roughly) five-second intervals.
                Specifying "stutter=0" causes the test to run continuously
                without pausing, which is the old default behavior.

test_no_idle_hz Whether or not to test the ability of RCU to operate in
                a kernel that disables the scheduling-clock interrupt to
                idle CPUs.  Boolean parameter, "1" to test, "0" otherwise.
                Defaults to omitting this test.

torture_type    The type of RCU to test: "rcu" for the rcu_read_lock() API,
                "rcu_sync" for rcu_read_lock() with synchronous reclamation,
                "rcu_bh" for the rcu_read_lock_bh() API, "rcu_bh_sync" for
                rcu_read_lock_bh() with synchronous reclamation, "srcu" for
                the "srcu_read_lock()" API, "sched" for the use of
                preempt_disable() together with synchronize_sched(),
                and "sched_expedited" for the use of preempt_disable()
                with synchronize_sched_expedited().

verbose         Enable debug printk()s.  Default is disabled.
```

OUTPUT

The statistics output is as follows:

```
        rcu-torture: --- Start of test: nreaders=16 stat_interval=0 verbose=0
        rcu-torture: rtc: 0000000000000000 ver: 1916 tfle: 0 rta: 1916 rtaf: 0
rtf: 1915
        rcu-torture: Reader Pipe:  1466408 9747 0 0 0 0 0 0 0 0 0
        rcu-torture: Reader Batch:  1464477 11678 0 0 0 0 0 0 0 0
        rcu-torture: Free-Block Circulation:  1915 1915 1915 1915 1915 1915 1915
1915 1915 1915 0
        rcu-torture: --- End of test
```

The command "dmesg | grep torture:" will extract this information on

most systems. On more esoteric configurations, it may be necessary to
use other commands to access the output of the printk()s used by
the RCU torture test. The printk()s use KERN_ALERT, so they should
be evident. ;-)

The entries are as follows:

o	"rtc": The hexadecimal address of the structure currently visible
	to readers.

o	"ver": The number of times since boot that the rcutw writer task
	has changed the structure visible to readers.

o	"tfle": If non-zero, indicates that the "torture freelist"
	containing structure to be placed into the "rtc" area is empty.
	This condition is important, since it can fool you into thinking
	that RCU is working when it is not. :-/

o	"rta": Number of structures allocated from the torture freelist.

o	"rtaf": Number of allocations from the torture freelist that have
	failed due to the list being empty.

o	"rtf": Number of frees into the torture freelist.

o	"Reader Pipe": Histogram of "ages" of structures seen by readers.
	If any entries past the first two are non-zero, RCU is broken.
	And rcutorture prints the error flag string "!!!" to make sure
	you notice. The age of a newly allocated structure is zero,
	it becomes one when removed from reader visibility, and is
	incremented once per grace period subsequently -- and is freed
	after passing through (RCU_TORTURE_PIPE_LEN-2) grace periods.

	The output displayed above was taken from a correctly working
	RCU. If you want to see what it looks like when broken, break
	it yourself. ;-)

o	"Reader Batch": Another histogram of "ages" of structures seen
	by readers, but in terms of counter flips (or batches) rather
	than in terms of grace periods. The legal number of non-zero
	entries is again two. The reason for this separate view is that
	it is sometimes easier to get the third entry to show up in the
	"Reader Batch" list than in the "Reader Pipe" list.

o	"Free-Block Circulation": Shows the number of torture structures
	that have reached a given point in the pipeline. The first element
	should closely correspond to the number of structures allocated,
	the second to the number that have been removed from reader view,
	and all but the last remaining to the corresponding number of
	passes through a grace period. The last entry should be zero,
	as it is only incremented if a torture structure's counter
	somehow gets incremented farther than it should.

Different implementations of RCU can provide implementation-specific
additional information. For example, SRCU provides the following:

```
        srcu-torture: rtc: f8cf46a8 ver: 355 tfle: 0 rta: 356 rtaf: 0 rtf: 346
rtmbe: 0
        srcu-torture: Reader Pipe:  559738 939 0 0 0 0 0 0 0 0 0
        srcu-torture: Reader Batch:  560434 243 0 0 0 0 0 0 0 0
        srcu-torture: Free-Block Circulation:  355 354 353 352 351 350 349 348
347 346 0
        srcu-torture: per-CPU(idx=1): 0(0,1) 1(0,1) 2(0,0) 3(0,1)
```

The first four lines are similar to those for RCU. The last line shows
the per-CPU counter state. The numbers in parentheses are the values
of the "old" and "current" counters for the corresponding CPU. The
"idx" value maps the "old" and "current" values to the underlying array,
and is useful for debugging.

Similarly, sched_expedited RCU provides the following:

```
        sched_expedited-torture: rtc: d0000000016c1880 ver: 1090796 tfle: 0 rta:
1090796 rtaf: 0 rtf: 1090787 rtmbe: 0 nt: 27713319
        sched_expedited-torture: Reader Pipe:  12660320201 95875 0 0 0 0 0 0 0 0
0
        sched_expedited-torture: Reader Batch:  12660424885 0 0 0 0 0 0 0 0 0 0
        sched_expedited-torture: Free-Block Circulation:  1090795 1090795
1090794 1090793 1090792 1090791 1090790 1090789 1090788 1090787 0
```

USAGE

The following script may be used to torture RCU:

```
        #!/bin/sh

        modprobe rcutorture
        sleep 100
        rmmod rcutorture
        dmesg | grep torture:
```

The output can be manually inspected for the error flag of "!!!".
One could of course create a more elaborate script that automatically
checked for such errors. The "rmmod" command forces a "SUCCESS" or
"FAILURE" indication to be printk()ed.