Decoding Those Inscrutable RCU CPU Stall Warnings

“They are for your own good! Honest!!!”
Overview

- Why does RCU emit CPU stall warnings?
- Decoding CPU stall warnings
- What causes CPU stalls?
- Obligatory war stories
- Summary
Why Does RCU Emit CPU Stall Warnings?
Decoding Those Inscrutable RCU CPU Stall Warnings, September 12, 2017

RCU Mostly Doesn't Emit CPU Stall Warnings!!!
When Does RCU Emit CPU Stall Warnings? When RCU Believes That a CPU is in Deep Trouble!

- CPU 0
  - RCU Reader (Including Interrupts Disabled)
  - At Least 21 Seconds

- CPU 1
  - RCU Reader
  - Userspace Execution

- RCU
  - RCU CPU Stall Warning

Time
But What if I Don't Want CPU Stall Warnings???
But What if I Don't Want CPU Stall Warnings???

- Boot with `rcupdate.rcu_cpu_stall_suppress=1`
- Boot with `rcupdate.rcu_cpu_stall_timeout=NN` (in seconds)
  - Or build with `CONFIG_RCU_CPU_STALL_TIMEOUT=NN`
  - $3 \leq N \leq 300$, in seconds

Why would you want to suppress CPU stall warnings?
- Slow embedded system tested on faster development system
  - Increase timeout on slow embedded system (or decrease during test)
- Throughput-based rip-and-replace cloud-computing environment
- Embedded production environment where console output is ignored
  - Suppress warnings entirely

But if response time matters, you care about CPU stalls
- Especially during development and testing
Decoding CPU Stall Warnings
INFO: rcu_sched detected stalls on CPUs/tasks:
  0-...0: (1 GPs behind) idle=bf2/1400000000000000/0 softirq=554/555 fqs=6754
  (detected by 1, t=21003 jiffies, g=-154, c=-155, q=120339)
Sending NMI from CPU 1 to CPUs 0:
NMI backtrace for cpu 0
CPU: 0 PID: 773 Comm: rcu_torture_sta Not tainted 4.13.0-rc2+ #1
Hardware name: QEMU Standard PC (i440FX + PIIX, 1996), BIOS Bochs 01/01/2011
task: ffff93f7ddd172c0 task.stack: ffff95a3417f4000
RIP: 0010:get_seconds+0xc/0x10
RSP: 0000:ffff95a3417f7ef0 EFLAGS: 00000097
RAX: 00000000059a853e2 RBX: 00000000059a853e6 RCX: fffffffff8bc45d98
RDX: 0000000000000001 RSI: 0000000000000092 RDI: ffffffff8cf7f34c
RBP: ffff95a3417f7f00 R08: 00000000fffffffffe R09: 000000000000060e
R10: 000000000000005 R11: 000000000000000a R12: ffff93f7ddefae100
R13: ffff95a3400d7cf0 R14: 0000000000000000 R15: ffff93f7ddd172c0
FS: 0000000000000000(0000) GS:fffff93f7dfc00000(0000) knlGS:0000000000000000
CS: 0010 DS: 0000 ES: 0000 CR0: 0000000008005003
CR2: 0000000000000000 CR3: 0000000000000000 CR4: 0000000000000000
Call Trace:
  ? rcu_torture_stall+0xcb/0x140
  kthread+0x104/0x140
  ? rcu_torture_stats+0x70/0x70
  ? kthread_park+0x60/0x60
  ret_from_fork+0x22/0x30
Identifying an RCU CPU Stall Warning (First Format)

INFO: rcu_sched detected stalls on CPUs/tasks:
    0-...0: (1 GPs behind) idle=bf2/140000000000000000/0 softirq=554/555 fqs=6754
    (detected by 1, t=21003 jiffies, g=-154, c=-155, q=120339)
Sending NMI from CPU 1 to CPUs 0:
NMI backtrace for cpu 0
CPU: 0 PID: 773 Comm: rcu_torture_sta Not tainted 4.13.0-rc2+ #1
Hardware name: QEMU Standard PC (i440FX + PIIX, 1996), BIOS Bochs 01/01/2011
task: ffff93f7ddd172c0 task.stack: ffff95a3417f4000
RIP: 0010:get_seconds+0xc/0x10
RSP: 0000:ffff95a3417f7ef0 EFLAGS: 00000097
RAX: 0000000059a853e2 RBX: 0000000059a853e6 RCX: ffffffff8bc45d98
RDX: 0000000000000001 RSI: 0000000000000092 RDI: ffffffff8cf7f34c
RBP: ffff95a3417f7f00 R08: 00000000fffffffffe R09: 0000000000000060e
R10: 0000000000000005 R11: 000000000000000a R12: ffff93f7ddf9e100
R13: ffff95a3400d7cf0 R14: 0000000000000000 R15: ffff93f7ddd172c0
FS: 0000000000000000 GS: ffff93f7dfc00000(0000) knlGS: 0000000000000000
CS: 0010 DS: 0000 ES: 0000 CR0: 0000000000000000
CR2: 0000000000000000 CR3: 0000000019a0a000 CR4: 0000000000000060
Call Trace:
  ? rcu_torture_stall+0xcb/0x140
  kthread+0x104/0x140
  ? rcu_torture_stats+0x70/0x70
  ? kthread_park+0x60/0x60
  ret_from_fork+0x22/0x30
**Which CPU is Stalled? (First Format)**

INFO: rcu_sched detected stalls on CPUs/tasks:

₀₋₋₋₀: (1 GPs behind) idle=bf2/1400000000000000/₀ softirq=554/555 fqs=6754
(detected by 1, t=21003 jiffies, g=-154, c=-155, q=120339)

Sending NMI from CPU 1 to CPUs 0:

NMI backtrace for cpu 0
CPU: 0 PID: 773 Comm: rcu_torture_sta Not tainted 4.13.0-rc2+ #1
Hardware name: QEMU Standard PC (i440FX + PIIX, 1996), BIOS Bochs 01/01/2011

Task: ffff93f7ddd172c0 Task stack: ffff95a3417f4000

₀₋₋₋₀: (1 GPs behind)

CPU was aware of last GP
Interrupts disabled for awhile
CPU online for next GP begin
CPU was online at GP begin
CPU is online now
Stalled CPU

Dyntick counter
Process/irq nesting
NMI nesting

Idle/offline scans

GP start/now

Softirq=554/555 fqs=6754
Which CPU Detected the Stall? (First Format)

INFO: rcu_sched detected stalls on CPUs/tasks:
  0-...0: (1 GPs behind) idle=bf2/1400000000000000/0 softirq=554/555 fqs=6754
  (detected by 1, t=21003 jiffies, g=-154, c=-155, q=120339)

Sending NMI from CPU 1 to CPUs 0:
NMI backtrace for cpu 0
CPU: 0 PID: 773 Comm: rcu_torture_stab Not tainted 4.13.0-rc2+ #1
Hardware name: QEMU Standard PC (i440FX + PIIX, 1996), BIOS Bochs 01/01/2011
  task: ffff93f7ddd172c0 task.stack: ffff95a3417f4000
  RIP: 0010:get_seconds+0xc/0x10
  RSP: 0000:ffff95a3417f7ef0 EFLAGS: 00000097
  RAX: 0000000059a853e2 RBX: 0000000059a853e6 RCX: ffffffff8bc45d98
  RDX: 0000000000000001 RSI: 000000000000092 RDI: ffffffff8cf7f34c
  RBP: ffff95a3417f7f00 R08: 0000000000000000 R09: 000000000000060e
  R10: 0000000000000005 R11: 000000000000000a R12: ffff93f7ddfae100
  R13: ffff95a3400d7cf0 R14: 0000000000000000 R15: ffff93f7ddd172c0
  FS:  0000000000000000(0000) GS:fffffffff8fc00000(0000) knlGS:0000000000000000
  CS:  0010 DS: 0000 ES: 0000 CR0: 0000000080050033
  CR2: 00000000  CR3: 00000000000050033
  CR4: 0000000000000000
Call Trace:
  ? rcu_torture_stall
  kthread+0x104/0x140
  ? rcu_torture_stats+0x70/0x70
  ? kthread_park+0x60/0x60
  ret_from_fork+0x22/0x30

Which CPU Detected the Stall? (First Format)

Which CPU Detected the Stall? (First Format)

Which CPU Detected the Stall? (First Format)

Which CPU Detected the Stall? (First Format)
INFO: rcu_sched detected stalls on CPUs/tasks:
    0-...0: (1 GPs behind) idle=bf2/1400000000000000000/0 softirq=554/555 fqs=6754
    (detected by 1, t=21003 jiffies, g=-154, c=-155, q=120339)

Sending NMI from CPU 1 to CPUs 0:

NMI backtrace for cpu 0
CPU: 0 PID: 773 Comm: rcu_torture_stall Not tainted 4.13.0-rc2+ #1
Hardware name: QEMU Standard PC (i440FX + PIIX, 1996), BIOS Bochs 01/01/2011
  task: ffff93f7ddd172c0 task.stack: ffff95a3417f4000
  RIP: 0010:get_seconds+0xc/0x10
  RSP: 0000:ffff95a3417f7ef0 EFLAGS: 00000097
  RAX: 00000000059a853e2 RBX: 00000000059a853e6 RCX: ffffffff8bc45d98
  RDX: 0000000000000001 RSI: 0000000000000092 RDI: ffffffff8cf7f34c
  RBP: ffff95a3417f7f00 R08: 00000000fffffff6 R09: 000000000000060e
  R10: 0000000000000005 R11: 000000000000000a R12: ffff93f7ddf9e100
  R13: ffff95a340d7ef0 R14: 0000000000000000 R15: ffff93f7ddd172c0
  FS: 0000000000000000 GS:0000000000000000 kn1GS:000000000000000000000000
CS: 0010 DS: 0000 ES: 0000 CR0: 0000000000050033
CR2: 0000000000000000 CR3: 00000000019a0a000 CR4: 0000000000000600

Call Trace:
? rcu_torture_stall+0xcb/0x140
  kthread+0x104/0x140
? rcu_torture_stats+0x70/0x70
? kthread_park+0x60/0x60
ret_from_fork+0x22/0x30
INFO: rcu_sched detected stalls on CPUs/tasks:
  0-...0: (1 GPs behind) idle=bf2/1400000000000000/0 softirq=554/555 fqs=6754
  (detected by 1, t=21003 jiffies, g=-154, c=-155, q=120339)

Sending NMI from CPU 1 to CPUs 0:
NMI backtrace for cpu 0
CPU: 0 PID: 773 Comm: rcu_torture_sta Not tainted 4.13.0-rc2+ #1
Hardware name: QEMU Standard PC (i440FX + PIIX, 1996), BIOS Bochs 01/01/2011
task: ffff93f7ddd172c0 task.stack: ffff95a3417f4000
RIP: 0010:get_seconds+0xc/0x10
RSP: 0000:ffff95a3417f7ef0 EFLAGS: 00000097
RAX: 0000000059a853e2 RBX: 0000000059a853e6 RCX: ffffffff8bc45d98
RDX: 0000000000000001 RSI: 0000000000000092 RDI: ffffffff8cf7f34c
RBP: ffff95a3417f7f00 R08: 00000000fffffffefe R09: 000000000000060e
R10: 0000000000000005 R11: 0000000000000092a R12: ffff93f7ddfae100
R13: ffff95a3400d7cf0 R14: 0000000000000000 R15: ffff93f7ddd172c0
FS: 0000000000000000 GS:fffffff7dfe00000(0000) knlGS:0000000000000000
CS: 0010 DS: 0000 ES: 0000 CR0: 0000000000000000
CR2: 0000000000000000 CR3: 0000000019a0a000 CR4: 0000000000000600

NMI Backtrace Stack

rcu_torture_stall

Call Trace:
  ? rcu_torture_stall+0xcb/0x140
  kthread+0x104/0x140
  ? rcu_torture_stats+0x70/0x70
  ? kthread_park+0x60/0x60
  ret_from_fork+0x22/0x30
Example RCU CPU Stall Warning Splat (2\textsuperscript{nd} Format)

INFO: rcu_sched self-detected stall on CPU 0-....: (20937 ticks this GP) idle=b5e/14000000000000001/0 softirq=258/258 fqs=51764 (t=21000 jiffies g=-159 c=-160 q=98)

NMI backtrace for cpu 0
CPU: 0 PID: 713 Comm: rcu_torture_sta Not tainted 4.13.0-rc2+ #1
Hardware name: QEMU Standard PC (i440FX + PIIX, 1996), BIOS Bochs 01/01/2011
Call Trace:
  <IRQ>
  dump_stack+0x4d/0x6e
  nmi_cpu_backtrace+0xc5/0xd0
  ...
  smp_apic_timer_interrupt+0x33/0x50
  apic_timer_interrupt+0x86/0x90
RIP: 0010:get_seconds+0x0/0x10
RSP: 0000:ffffffa446813ebef0 EFLAGS: 00000297 ORIG_RAX: ffffffffffffffff10
RAX: 00000000599827f5 RBX: 00000000599827f9 RCX: fffffff84a45cd8
RDX: 0000000000000001 RSI: 0000000000000092 RDI: fffffff85d7320c
RBX: fffffffa446813ebef00 R08: 00000000fffffffe R09: 00000000000005fd
R10: 0000000000000005 R11: 0000000000000000a R12: fffffff810de08f0c0
R13: fffffffa446800d3cf0 R14: 0000000000000000 R15: fffffff810de0ea580
</IRQ>
? rcu_torture_stall+0xcb/0x140
kthread+0x104/0x140
? rcu_torture_stats+0x70/0x70
? kthread_park+0x60/0x60
ret_from_fork+0x22/0x30
Example RCU CPU Stall Warning Splat (2nd Format)

INFO: rcu_sched self-detected stall on CPU

0-....: (20937 ticks this GP) idle=b5e/14000000000000001/0 softirq=258/258 fqs=5176
(t=21000 jiffies g=-159 c=-160 q=98)

NMI backtrace for cpu 0
CPU: 0 PID: 713 Comm: rcu_torture_sta Not tainted 4.13.0-rc2+ #1
Hardware name: QEMU Standard PC (i440FX + PIIX, 1996), BIOS Bochs 01/01/2011
Call Trace:
<IRQ>
dump_stack+0x4d/0x6e
nmi_cpu_backtrace+0xc5/0xd0
...
smp_apic_timer_interrupt+0x33/0x50
apic_timer_interrupt+0x86/0x90
RIP: 0010:get_seconds+0x0/0x10
RSP: 0000:ffffa446813ebef0 EFLAGS: 00000297 ORIG_RAX: ffffffffffffffff10
RAX: 00000000599827f5 RBX: 00000000599827f9 RCX: ffffffff84a45cd8
RDX: 0000000000000001 RSI: 0000000000000092 RDI: ffffffff85d7320c
RBP: ffffa446813ebf00 R08: 00000000fffffffe R09: 00000000000005fd
R10: 0000000000000005 R11: 000000000000000a R12: fff8810de08f0c0
R13: ffffa446800d3cf0 R14: 0000000000000000 R15: fff8810de0ea580
</IRQ>
? rcu_torture_stall+0xc8/0x140
kthread+0x104/0x140
? rcu_torture_stats+0x70/0x70
? kthread_park+0xd/0x70
ret_from_fork+0xe/0xe0

rcu_sched self-detected stall on CPU
Which CPU is Stalled? (2nd Format)

INFO: rcu_sched self-detected stall on CPU
theta----: (20937 ticks this GP) idle=b5e/14000000000000001/0 softirq=258/258 fqs=5176
(t=21000 jiffies g=-159 c=-160 q=98)

NMI backtrace for cpu 0
CPU: 0/PID: 713 Comm: rcu_torture_sta Not tainted 4.13.0-rc2+ #1
Hardware name: QEMU Standard PC (i440FX + PIIX, 1996), BIOS Bochs 01/01/2011
Call Trace:
<IRQ>
dump_stack+0x4d/0x6e

theta----: (20937 ticks this GP)

CPU was aware of last GP
Interrupts enabled
CPU online for next GP begin
CPU was online at GP begin
CPU is online now
Stalled CPU
INFO: rcu_sched self-detected stall on CPU
  0-....: (20937 ticks this GP) idle=b5e/1400000000000001/0 softirq=258/258 fqs=5176
  (t=21000 jiffies g=-159 c=-160 q=98)
NMI backtrace for cpu 0
CPU: 0 PID: 713 Comm: rcu_torture_stas Not tainted 4.13.0-rc2+ #1
Hardware name: QEMU Standard PC (i440FX + PIIX, 1996), BIOS Bochs 01/01/2011
Call Trace:
  <IRQ>
dump_stack+0x4d/0x6e
  nmi_cpu_backtrace+0xc5/0xd0
...
  smp_apic_timer_interrupt+0x33/0x50
  apic_timer_interrupt+0x86/0x90
RIP: 0010:get_seconds+0x0/0x10
RSP: 0000:ffffa446813ebef0 EFLAGS: 00000297 ORIG_RAX: ffffffffffffffff10
RAX: 00000000599827f5 RBX: 00000000599827f9 RCX: ffffffff84a45cd8
RDX: 0000000000000001 RSI: 0000000000000092 RDI: ffffffff85d7320c
RBP: ffffa446813ebf00 R08: 00000000fffffffffe R09: 00000000000005fd
R10: 0000000000000005 R11: 000000000000000a R12: ffffa8810de08f0c0
R13: ffffa446800d3cf0 R14: 0000000000000000 R15: ffffa8810de0ea580
</IRQ>
? rcu_torture_stall+0xcb/0x140
  kthread+0x104/0x140
? rcu_torture_stats+0x70/0x70
? kthread_park+0x60/0x60
  ret_from_fork+0x22/0x30

NMI Backtrace Start Plus Interrupt Stack (2nd Format)
Repeated Stall Warnings: Compare Stack Traces!

- **CPU 0**: RCU Reader (Including Interrupts Disabled)
- **CPU 1**: RCU Reader → Userspace Execution

- **Stall Warning #1**
- **Stall Warning #2**

**Time**

- At Least 21 Seconds
- At Least 63 Seconds
What Causes CPU Stalls?
What Causes CPU Stalls?

rcu_read_lock();
for (;;)
    do_something;
rcu_read_unlock();

- How to fix this?
What Causes CPU Stalls?

```c
rcu_read_lock();
for (;;) {
    do_something;
rcu_read_unlock();
```

- How to fix this?
  - Break out of RCU read-side critical section occasionally
    - Preferably every few milliseconds
  - Ensure that your loops are finite
What Causes CPU Stalls?

```c
local_irq_disable();
for (;;)
    do_something;
local_irq_enable();
```

- How to fix this?
What Causes CPU Stalls?

```c
local_irq_disable();
for (;;)
    do_something;
local_irq_enable();
```

- **How to fix this?**
  - Break out of interrupt-disable regions occasionally
    * Increase break-out frequency until tglx stops throwing frozen sharks
  - Ensure that your loops are finite
    * Decrease loop length until tglx stops throwing frozen sharks
Variations on Long-Reader and IRQ-Disable Themes

- Looping with preemption disabled
  - You should expect incoming frozen sharks in this case as well

- Looping with bottom-half execution disabled
  - Ditto

- Long-running interrupt within RCU reader

- PREEMPT=n: Looping without invoking schedule() or cond_resched_rcu_qs()

- PREEMPT=y: Indefinitely preempting an RCU reader
  - Can also try RCU_BOOST=y

- In virtualized environments, vCPU preemption
  - (Working on it, see Aravinda Prasad et al. 2017 USENIX paper)
What Causes CPU Stalls?

Embedded Computer System

115Kbaud High-Speed Serial Line!!!
What Causes CPU Stalls?

Embedded Computer System

115Kbaud High-Speed Serial Line!!!
Unless you have 500K console output...

What do you do about this?
What Causes CPU Stalls?

▪ Develop on fast system that just barely avoids CPU stalls
  – Then deploy on slow system

▪ Interrupt overload

▪ Turning on super-high-overhead debugging
  – https://marc.info/?l=linux-kernel&m=150176048506696
  – So adjust/disable the CPU stall timeout in this case!!!

▪ Prevent RCU_SOFTIRQ from running
  – For example, CPU-bound high-priority real-time process

▪ Completely shut off CPU's scheduler-clock interrupt

▪ Hardware failure
  – In one case, a fail-stop CPU!
  – Timer issues are a recurrent theme (see later war story)
RCU Bugs Can Also Cause CPU Stalls

- When things are stuck for 21 seconds, no need to be dainty
  - *False!!!* As I spent a couple years learning...

- Stall-warning messages can cause the stall to end
  - After part of the message is printed... This case now flagged

- Extremely quiet embedded systems have their own issues
  - They can enter states noisy systems avoid!!!

- RCU kthread wakeup failures
Evolution of RCU Grace-Period Handling

Pre-v3.8: RCU state machine driven via softirq
First Clue of Large-System RT-Response Importance
Evolution of RCU Grace-Period Handling

Pre-v3.8: RCU state machine driven via softirq

v3.8: RCU state machine driven via kthread
Evolution of RCU Grace-Period Handling

Pre-v3.8: RCU state machine driven via softirq

v3.8: RCU state machine driven via kthread
RCU Grace-Period Kernel Thread Wakeup Failures

rcu_bh kthread starved for 21134 jiffies! g18446744073709551396
c18446744073709551395 f0x0 RCU_GP_WAIT_FQS(3) ->state=0x0 ->cpu=0
c18446744073709551395 RCU_GP_WAIT_FQS(3) ->state=0x0 ->cpu=0
rcu_bh R running task 14968 9 2 0x00080000

Call Trace:
__schedule+0x20b/0x6c0
schedule+0x31/0x80
schedule_timeout+0x170/0x2f0
? call_timer_fn+0x130/0x130
rcu_gp_kthread+0x4be/0xd90
kthread+0x104/0x140
? rcu_oom_notify+0xf0/0xf0
? kthread_park+0x60/0x60
ret_from_fork+0x22/0x30
Decoding Those Inscrutable RCU CPU Stall Warnings, September 12, 2017

RCU Grace-Period Kernel Thread Wakeup Failures

rcu_bh kthread starved for 21134 jiffies!  
g18446744073709551396  
c18446744073709551395 f0x0  
RCU_GP_WAIT_FQS(3) ->state=0x0 ->cpu=0  
rcu_bh R running task 14968 9 2 0x00080000

Call Trace:
__schedule+0x20b/0x6c0
schedule+0x31/0x80
schedule_timeout+0x170/0x2f0
? call_timer_fn+0x130/0x130
rcu_gp_kthread+0x4be/0xd90
kthread+0x104/0x140
? rcu_oom_notify+0xf0/0xf0
? kthread_park+0x60/0x60
ret_from_fork+0x22/0x30

RCU cannot do much for you if you don't let its kthreads run!!!
Obligatory War Stories
My Favorite? “CPU-0 Standard Time”
“CPU-0 Standard Time”
“CPU-0 Standard Time”

- **GP Start, t="0"**
- **CPU stall, t=30**

![Diagram showing CPU activity over time, with CPU 0, CPU 1, and CPU 2, and labeled time points 0 to 60.]
Time Waits For No One, But...
High-Level RCU Grace-Period Processing

- Initialize Grace Period
  - Wait For New Grace-Period Request
  - Grace Period Done?
    - Yes: Clean Up Grace Period
    - No: Wait A Few Jiffies
      - Check Idle CPUs

- Initialize Grace Period
  - Wait For New Grace-Period Request
  - Grace Period Done?
    - Yes: Clean Up Grace Period
    - No: Wait A Few Jiffies
      - Check Idle CPUs
High-Level RCU Grace-Period Processing

1. Initialize Grace Period
2. Check Grace Period Done?
   - Yes: Clean Up Grace Period
   - No: Wait For New Grace-Period Request
3. Wait A Few Jiffies
4. Check Idle CPUs
High-Level Timer Processing

RCU GP kthread: post timer on timer wheel

In the fullness of time...

raise_softirq of TIMER_SOFTIRQ

Wake up requesting process
High-Level Timer Processing, CPU Offline

RCU GP kthread: post timer on CPU 5 timer wheel

CPU 5 goes offline

In the fullness of time...

Timer migrated to surviving CPU

raise_softirq of TIMER_SOFTIRQ

Wake up requesting process
High-Level Timer Processing, CPU Offline, RCU

- **RCU GP kthread**: post timer on CPU 5 timer wheel
- **CPU 5 goes offline**
- **In the fullness of time...**
  - **raise_softirq of TIMER_SOFTIRQ**
  - **Wake up requesting process**
- **Hotplug notifier waits for RCU grace period**
- **Timer migrated to surviving CPU**
High-Level Timer Processing, CPU Offline, RCU

RCU GP kthread: post timer on CPU 5 timer wheel

In the fullness of time...

raise_softirq of TIMER_SOFTIRQ

Wake up requesting process

CPU 5 goes offline

Hotplug notifier waits for RCU grace period

Timer migrated to surviving CPU
Time Waits For No One, But It Can Deadlock With CPU-Hotplug Offline and RCU Grace Periods!!!
Summary

- **RCU CPU stall warnings are a valuable diagnostic tool**
  - CPUs stuck in various unhelpful states
  - Extreme overload
  - Priority issues
  - Temporal anomalies
  - Low-level software issues
  - Hardware problems
  - RCU bugs

- **Prevention:**
  - Pause points in unbounded loops
  - Test on deployment-class systems (or adjust CPU-stall timeout)
  - Assign priorities carefully
  - Respect the passage of time
Summary

- **RCU CPU stall warnings are a valuable diagnostic tool**
  - CPUs stuck in various unhelpful states
  - Extreme overload
  - Priority issues
  - Temporal anomalies
  - Low-level software issues
  - Hardware problems
  - RCU bugs

- **Prevention:**
  - Pause points in unbounded loops
  - Test on deployment-class systems (or adjust CPU-stall timeout)
  - Assign priorities carefully
  - Respect the passage of time
  - **Make sure this McKenney fellow doesn't mess up RCU!**
Legal Statement

- This work represents the view of the author and does not necessarily represent the view of IBM.
- IBM and IBM (logo) are trademarks or registered trademarks of International Business Machines Corporation in the United States and/or other countries.
- Linux is a registered trademark of Linus Torvalds.
- Other company, product, and service names may be trademarks or service marks of others.
Questions?