`Chat for the Microconference:
https://chat.2020.linuxplumbersconf.org/channel/real-time-mc

irc.oftc.net - #linux-rt Regular IRC channel

== A Realtime Tour Through BPF ==
- Focusing on use-cases
- 4k instructions limit at first
  - but then wanted to increase to 1m! (because it is useful)
- new limit of 1M instructions means that potential latency for BPF program is > 1ms
- want to be more flexible, like access to the user-space memory (that might cause page faults)
- current BPF environment allows for sleepable programs
- 4k was enough for a long time, now 1M is enough. Who knows if 100M might be needed in the future? RT shows the problem earlier. 1M (with a 1ns time per instruction) is a problem for RT, abut 100M is going to be a problem for most workloads.
- Low Bandwidth monitoring?
  - monitoring of events with low b/w requrements (i.e. trying to keep impact of monitoring low)
- BPF for in-kernel data reduction of a high-bandwidth in-kernel data source
- Histograms in BPF? Merging with Tracing capabilities? There's a lot of shared code.
- Wonder if we need to work with the BPF community to increase preemption in BPF programs?

== futex2: A New Interface ==
- last futex feature added was RT clock in 2008
- Many features to be added, but futex is tricky: to many things to avoid breaking.
- lack of Numa awareness
- determinism is tough due to single point of access for internal reousrces such as the hash table
- ? will the new i/f be replacement or will it be phased in?
  - not sure, debating how best to move forward
    - - could use the PREEMPT_RT queue as an implementation sandbox?
- tglx cautions that futextes are a dangerous place to develop

PoC: https://gitlab.collabora.com/tonyk/linux/-/commits/futex2-lpc

- Then discussions about how to deal with multiple interfaces, which may take 10 years to deprecate the old one.

== How do we kick our RT habit? ==

- sounds like he's talking about the "scheduling policy" not the "kernel".

- They are "working around" CFS "limitations" with RT scheduler.
    - - sounds familiar [clark] :) o/

- Affining the nice-20 critical processes to a small set of CPUs improves the throughput to bring it in the same ballpark with RT.

- The number of migrations in the affined case also goes down significantly when compared to the non-affined case.

- wonder if SCHED_DEADLINE would be an appropriate choice?

- Maybe the deadline server might help

- the discussion in the chat went to the trace side: what is tracing showing? what is the difference in the scheduling/migration?

- Mentions to the latency nice

== Handling stable releases once RT is merged ==

- Once RT is mainline, how do we handle? role of stable?
  - Will regressions in RT be taken as stable fixes?
  - How will issues in backporting happen?
  - Testing of RT on stable kernels?

- Automate downloading of each preview of the stable, and starts to run the RT tests; possibly trigger?

- - trigger on RC, email, just let Greg know.
- - How set up RT infrastructure to talk to Greg's. What have to setup and build?
- - Why not in normal CI today? As it gets merged in, go from there?
- - How become part of kernel CI?
- - RT testing is more benchmarky. Performance criteria? Stress tests?
- - go off the git push, get notification. Otherwise let know email to be notified.
-

- Should we maintain an "rt-staging"/"rt-next" tree, for testing things like futex, new features, and should these be backported to stable?

- - "come back into the fold, not be out in the cold"?
- - Good API stabilization, might be a role, but say we don't backport.
-

-Testing of RT - we'll need a team to make sure it doesn't break in the stable releases. Just need to do it.

- - Owners of bug reports? Nothing really special, will be RT maintainers. Onus is on them to fix it.
- - Why would it be broken in stable, and not mainline. Bug compatible.
- - Greg notes that this is no real different from the Graphic tree.
- - Old stables not up to date with what is upstream.
- - *** Add "Tested-by:" once we're added to mainline, 48 hours window, once in mainline.
- - Sasha recommends add automation before it gets to mainline, even if ignored.
- - RT Maintainers meeting - track down Kernel CI... done. Mark Brown also works in Kernel CI team. RT stable trees were automated merge.

- - maybe start testing right now?
    - - If it fails the merge, note for manual intervention.
    - - For stable, need to get the earlier notification (the rc candidate), start narrowing the gap. Stable release and RT release more in sync.

- Continue discussion in next RT stable meeting. RT mainlining - test farms sync as well.

-

== Continuous Integration for mainline Real-Time Linux ==

- Contact ci@linutronix.de if you want access to ci-rt.linutronix.de.
- Notifications mailed to the person who triggered test. Tests triggered by pushing to git repo.

- Additonal trees to test:
  - Current RC
  - linux-next
  - stable trees (once RT is in a stable release)

What other branches, and for how long?

Mel Gorman FWIW, my battery of tests for a potentiall RT kernel is
functional-ltp-lite-xfs
functional-ltp-cve-xfs
workload-cyclictest-none
io-dbench4-async-xfs
network-netperf-unbound
network-netperf-cross-socket
scheduler-unbound
workload-cyclictest-none
workload-cyclictest-hackbench
workload-cyclictest-kernbench
workload-cyclictest-fine-none
workload-cyclictest-fine-hackbench
workload-cyclictest-fine-kernbench
workload-futexbench
workload-futexwait
workload-rt-migration
workload-rt-migration-check
workload-sembench-futex
workload-jitter_detect
workload-stockfish
functional-ltp-lite-xfs
functional-ltp-cve-xfs
functional-pistress

Nikolai - common reporting? Worth pushing our CI results, so we can have it summarized through the

Kernel CI.
Then we'd have particular test info for release of RT kernel.

Decouple running of test.

Start small and send something, and then build up.

Kernel CI willing to work to evolve this into something formal in the UI.

See
https://foundation.kernelci.org/blog/2020/08/21/introducing-common-reporting/

Also
https://linuxplumbersconf.org/event/7/contributions/730/
https://linuxplumbersconf.org/event/7/contributions/731/

Veronika notes Red Hat (CKI) is doing some of this already.
- already testing rt-devel && internal RT kernels, not much work to adjust to testing a different tree

Mel - Would a different partitioning of test make sense that what's emerged? Client side continuous integration. Testing on a non-RT config, by accident has happens.

Where can we collaboarate?
#kernelci - IRC freenode
Monthly Linaro meeting
mailing list where folks are talking about this: https://lists.yoctoproject.org/g/automated-testing

automated-testing@lists.yoctoproject.org

== The usage of PREEMPT_RT in safety-critical systems: what do we need to do? ==

Question: Is this system, clear what is does, and how its designed. 34 yes/4 no.

What is recommendation on how system should be set up?
- Challenge is providing pieces of evidences, that system is working correctly in logical and timing. What are the kinds of evidence we need to show?

When we turn on the CONFIGU_PREEMPT_RT are we going to see more bugs, etc.? What's the gut feeling? Why?

What information would be needed beyond the Arcade game.

Quality comparison, but fail to hit deadlines?

== **Identifying Sources of OS Noise** ==

Double digit microsecond requirements today, some use cases wanting to see single digit microsecond.

Why do we care? Deploy a system and grab all CPUs you want - static deployement, can do fairly good job of walling it off from OS noise. Today...
- Kubernetes deploying dozens of containers.
- Community wants to specify real time its a real-time container, work just like on a more static config'd system.

Guidelines to provide to new BIOS vendors? "please don't do this ... " like ECC example. SMI to SMM. Being able to manage these types of interrupts a bit more intelligently.

Infinite number of configurations to work with, how do we start to pull some sense out of it.

CPUset, Hotplug, ...

Moving timer to different node?
- CPUset does not know, so timer will just where it is.
- Regular timers or HR timers? Both
- Telco uses HRtimers, but others using tic based stuff.

- Not always clear where the timer belongs to, will be requiring some separate infrastructure.

- Starvation Monitor running, seeing 40ms of sys jitter.
- Can tune the period and runtime (sched deadline paramenter).

How many customers who want to run real time in containers?
- Customer reps having to do hand holding, was surprised at how many folks wanting to deploy RT apps as containers (hearing it from OpenShift)
- Need to do tuning to get latency out, its just a fact of life.

== **PREEMPT_RT: status and Q&A** ==

- PrintK - didn't make it into 5.9

  - - on to 15th version of enable/disable.

- memory management - allocation from truly atomic regions
  - Paul, reworking patches using this. RCU tagging is fall back if this current approach doesn't work.

- Cleaning up locking, will be hitting LKML soon.
- Scheduler core code, still WIP in progress, waiting to get input from Peter.

To release a 5.8 RT would need to do some significant amount of backporting. We'll be aiming at 5.9-rc2 based, with merged goodies, and and also current WIP.

Printk should not be blocking out anything else, per Peter M. Wanting to have timestamping, with printk dmsg changes. Timekeeping infrastructure now permits, and reserve the fields at least. Combining with timestamp infrastructure, so hoping it doesn't hold up anything.

Reduce tree close to zero, big flood of patches should be showing up soon.