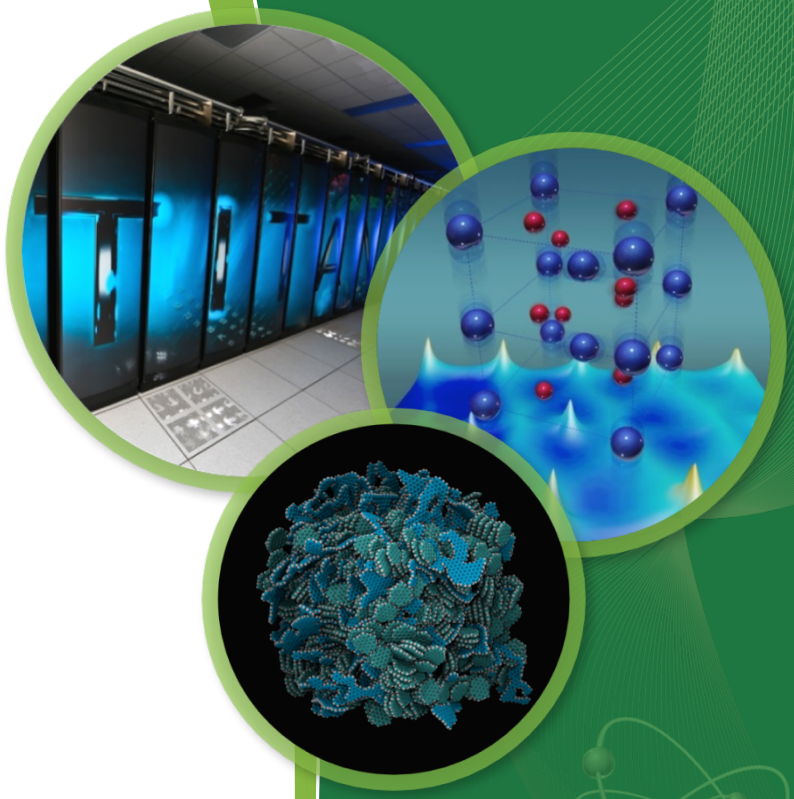
How to use modern tracing tools with Lustre

New debugging powers



Tracing is magical

Setup needed

- Rebuild perf
- libunwind or libdw for newer platforms
- pahole and friends in dwarves package (newer distros)
  - pahole -C sk_buff vmlinux | less (see C structure)
- trace-cmd for ftrace use

General profiling

- perf top
  - Default setting : -e cycles:pp (PEBS skids level 2)
- perf mem record -- touch /lustre/lustre/foobar
  - /sys/devices/cpu/events/*
- perf kmem record – touch /lustre/lustre/foobar
- perf sched record – touch /lustre/lustre/foobar
- perf stat –e ‘syscalls:sys_enter_write’ dd if=/dev/zero of=/lustre/lustre/foobar bs=512 count=100K
  - strace –c dd if=/dev/zero of=/lustre/lustre/foobar bs=512 count=100K
- perf lock (needs lockdep enabled)
- pmu-tools by Andi Kleen (raw counters)
  - Does not work in VMs (look to MSR registers)
  - New hardware specific events i.e for MESI protocol
  - ocperf.py record -e l2_lines_in.all -e l2_lines_in.e – touch /lustre/lustre/foobar

Lustre's new trace event example

- works with lctl set_param debug=+…
  - Sets by category
- /sys/kernel/debug/tracing/events/libcfs/*
- perf list libcfs*
- cat /sys/kernel/debug/tracing/events/libcfs/libcfs_info_fail_loc/format
- Filtering
  - perf record -F 99 -a –g --call-graph dwarf -e libcfs:libcfs_info_fail_loc --filter 'cfs_fail_loc == 0x319'
  - ONLY="74" ./sanity.sh
  - perf script

Lustre tracing implementation example

- Libcfs tracepoint patch - https://review.whamcloud.com/#/c/24554
- Local xxx_trace.[ch] files for each module subsystem
- TRACE_EVENT(libcfs_info_fail_loc, …
  - trace_info_fail_loc(…) …
- TRACE_EVENT_CONDITION(libcfs_warning_invalid_hash_algo,
  - Trace_cwarn_invalid_hash_algo(…)...
- pr_info(..) and friends???
- Avoid inline functions and debugging in headers

Dynamic tracing – when static tracepoints are not enough

- Need debuginfo installed
- perf probe –l
- perf probe -F
- perf probe -F --filter dev*xmit*
- perf probe -m /lib/modules/$(uname -r)/extra/lustre/fs/lustre.ko -F
- perf probe -m /lib/modules/$(uname -r)/extra/lustre/fs/lustre.ko -L ll_dir_getstripe
- perf probe -m /lib/modules/$(uname -r)/extra/lustre/fs/lustre.ko -L ll_dir_getstripe:31
- perf probe -m /lib/modules/$(uname -r)/extra/lustre/fs/lustre.ko -V ll_dir_getstripe
- perf probe -m /lib/modules/$(uname -r)/extra/lustre/fs/lustre.ko -V ll_dir_getstripe -externs

Dynamic tracing necromancy

- Example use:
  - perf probe -m /lib/modules/$(uname -r)/extra/lustre/fs/lustre.ko -a 'how_big_lmm=ll_dir_getstripe:33 lmm_size max_mdsize=body->mbo_max_mdsize:u32'
  - perf record -e probe:how_big_lmm –ag --call-graph dwarf lfs getstripe -c /lustre/lustre
  - perf script
  - perf probe –d how_big_lmm
- Break points at return
  - perf probe -m /lib/modules/$(uname -r)/extra/lustre/fs/lustre.ko -a 'getstripe_return=ll_dir_getstripe%return’
- Probing return values
  - perf probe -m /lib/modules/$(uname -r)/extra/lustre/fs/lustre.ko -a 'll_getname_return=ll_getname%return +0($retval):string’
  - perf record -e probe:ll_getname_return -ag --call-graph dwarf lfs getstripe -c /lustre/lustre/foobar
  - No longer need RETURN() macros

Replacing lctl set_param debug=+trace

- Can remove ENTER and EXIT in lustre kernel code
- trace-cmd record -p function_graph touch /lustre/lustre/foobar
  - trace-cmd report
- trace-cmd record -p function_graph -l ll_* touch /lustre/lustre/foobar

Replacing lctl set_param debug=+malloc

- perf kmem record -- touch /lustre/lustre/foobar
- Limit kernel memory scope to scan
  - lfs path2fid /lustre/lustre/foobar
  - perf record -a -g --call-graph dwarf -e kmem:kmalloc -- lfs fid2path /lustre/lustre [0x200000401:0x1f:0x0]
- Trace_cmd
  - trace-cmd record -p function_graph -e kmem:kmalloc -e kmem:kfree -l *:mod:lustre touch /lustre/lustre/foobar
    - Other filters include “function_name:traceon” or “function_name:traceoff”
    - Enable/disable_event