



LINUX PITER

2019 OCTOBER 4-5
SAINT PETERSBURG





LINUX PITER

2019 OCTOBER 4-5
SAINT PETERSBURG

Linux Perf advancements for compute intensive and server systems



Summary

- Asynchronous record trace streaming
- NUMA awareness in record profiling
- Runtime record trace compression
- Event naming in stat and record modes
- Typed context switches
- Perf privileged users
- *Parallel record trace streaming*

Asynchronous record trace streaming

The kernel can lose profiling data when its rate and volume is high.

`--aio[=<n>]`

Use `<n>` control blocks in asynchronous trace writing mode
(default: 1, max: 4)

Example: matrix multiplication executing 128 threads on Intel Xeon Phi

```
$ perf record -a -N -B -T -R -g -F K --call-graph dwarf,1024 \  
--user-regs=IP,SP,BP \  
--switch-events -e cycles,instructions,ref-cycles,\  
software/period=1,name=cs,config=0x3/Duk -- matrix.gcc.O3
```

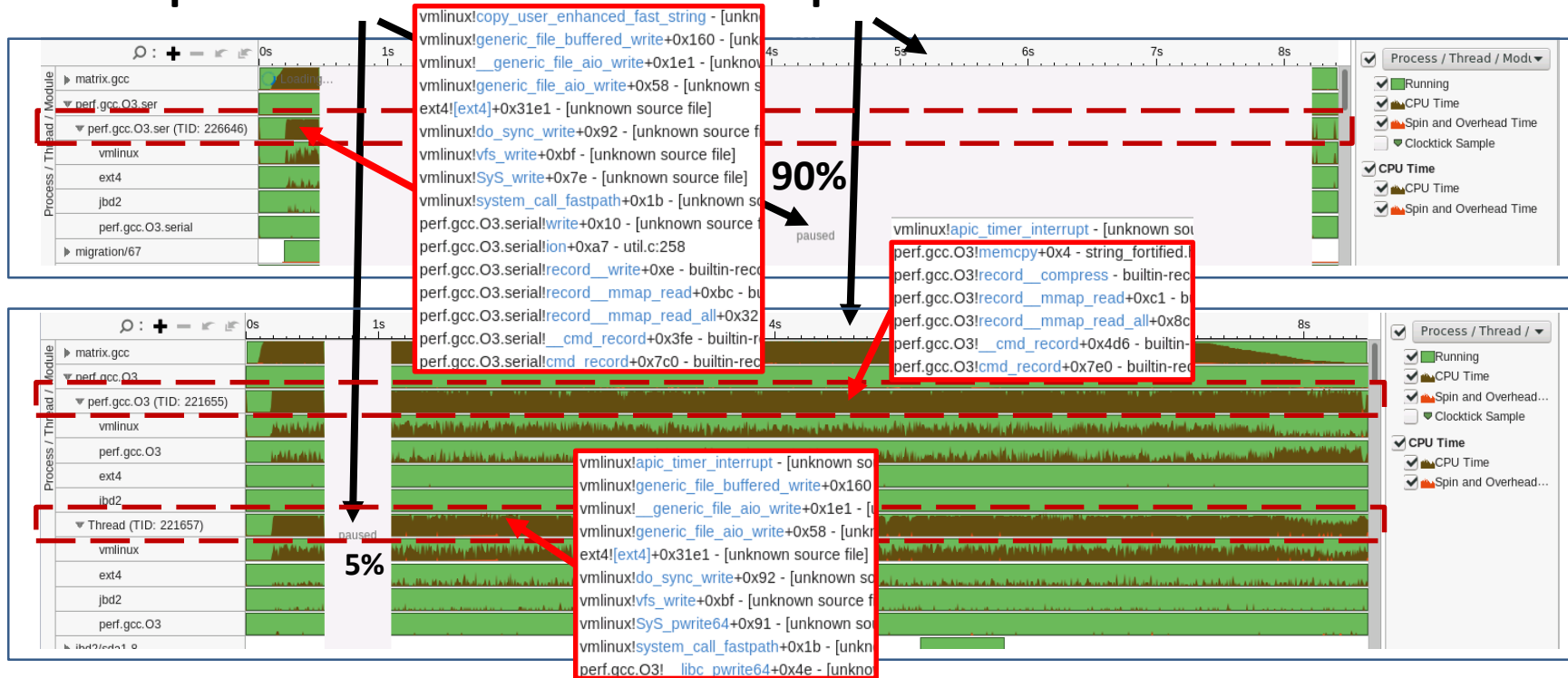
Metrics: **% data loss = paused_time / elapsed_time**

Asynchronous record trace streaming

paused time

elapsed time

~2x data loss decrease



NUMA awareness in record profiling

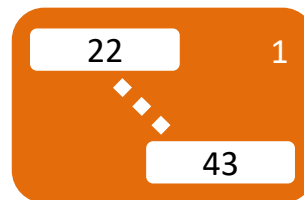
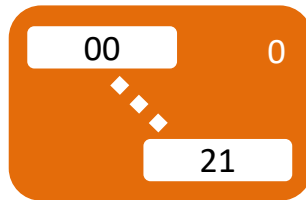
Up to 30% runtime overhead when a compute intensive workload fully utilizes a large server system with NUMA.

`--affinity <node|cpu>`

set affinity mask of trace reading thread to NUMA node
CPU mask or the CPU of processed event buffer

Example:

Intel Broadwell
2 sockets
44 cores



workload Perf
cpu mask [0-43] [0]

`--affinity node : cpu mask = [0-21] | [22-43]`

`--affinity cpu : cpu mask = [0] | [1] | [2] ... | [43]`

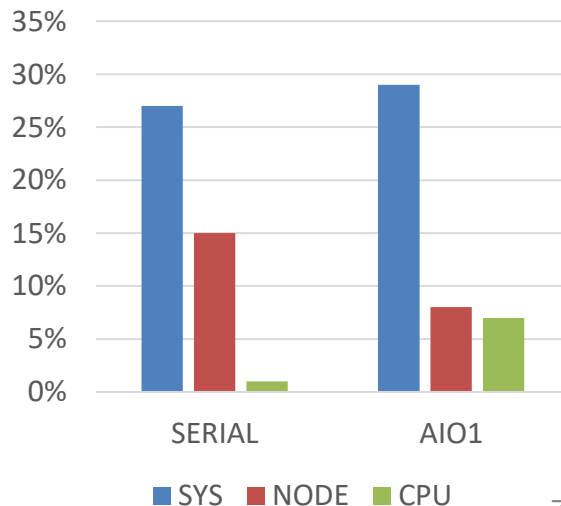
NUMA awareness in record profiling

NASA BT parallel benchmark

```
$ perf record -a -N -B -T -R -g -F K --call-graph dwarf,1024 --user-regs=IP,SP,BP \  
  --switch-events -e cycles,instructions,ref-cycles,\  
  software/period=1,name=cs,config=0x3/Duk \  
  [--aio] [--affinity=node|cpu] -- bt-mz.B
```

v4.20.0-rc5+	BENCH REPORT BASED	ELAPSED TIME BASED
SERIAL-SYS / BASE	1,27x (14,37/11,31)	1,29x (15,19/11,69)
SERIAL-NODE / BASE	1,15x (13,04/11,31)	1,17x (13,79/11,69)
SERIAL-CPU / BASE	1,00x (11,32/11,31)	1,01x (11,89/11,69)
AIO1-SYS / BASE	1,29x (14,58/11,31)	1,29x (15,26/11,69)
AIO1-NODE / BASE	1,08x (12,23/11,31)	1,11x (13,01/11,69)
AIO1-CPU / BASE	1,07x (12,14/11,31)	1,08x (12,83/11,69)

NUMA overhead



Runtime record trace compression

~8x (~5x avg.) trace size reduction when profiling a typical benchmark with a distinguished hotspot, thread stacks (dwarf) and context switches.

`-z, --compression-level[=n]`

Compress records using specified level (default:

1 - fastest compression,

22 - greatest compression)

`--mmap-flush <number>`

Minimal number of bytes that is extracted from mmap data pages (default: 1)

	SERIAL		AIO1	
-z	Ovh (x)	Ratio (x)	Ovh (x)	Ratio (x)
0	1,00	1,000	1,00	1,000
1	1,04	8,427	1,01	8,474
2	1,07	8,055	1,03	7,912
3	1,04	8,283	1,03	8,220
5	1,09	8,101	1,05	7,780
8	1,05	9,217	1,12	6,111

Extended event naming

```
$ perf stat -e cpu/name=\ 'CPU_CLK_UNHALTED.THREAD:cmask=0x1\ ',  
    period=0x3567e0,event=0x3c,cmask=0x1/Duk -- futex
```

...

```
$ perf record -e  
    cpu/name=\ 'OFFCORE_RESPONSE:request=DEMAND_RFO:response=L3_HIT.SNOOP_HITM\ ',  
    period=0x3567e0,event=0x3c,cmask=0x1/Duk -- futex
```

...

```
# Samples: 24K of event \  
\ 'OFFCORE_RESPONSE:request=DEMAND_RFO:response=L3_HIT.SNOOP_HITM' # Event count \  
(approx.): 8649200000 #  
# Overhead  Command  Shared Object      Symbol  
# .....  
#  
    14.75%  futex      [kernel.vmlinux]  [k] __entry_trampoline_start
```

Typed context switches

Grouping: Function / Call Stack

	CPU Time	Inactive Wait Time	Inactive Wait Count	
			Inactive Sync Wait Count	Preemption Wait Count
schedule	0.002s	33.885s	4	38,040
schedule	0s	33.885s	4	38,040
exit_to_usermode	0s	11.405s	0	38,040
prepare_exit_to_usermode ← retint_user ← multithread ← thread-function ← start_thread	0s	11.405s	0	38,038
do_syscall_64 ← entry_SYSCALL_64_after_dispatch			0	2
__write ← _IO_file_write ← new_do_write			0	1
__getdents ← readdir64 ← get_nprocs_conf			0	1
io_schedule ← __lock_page_killable ← filemap_fault			1	0
futex_wait_queue_me ← futex_wait ← do_futex			3	0

Inactive Sync Wait Count
 Inactive Sync Wait Count is the number of context switches a thread experiences when it is excluded from execution by the OS scheduler due to synchronization. Excessive number of thread context switches may negatively impact application performance. Apply optimization techniques to reduce synchronization contention and eliminate the problem.

Preemption Wait Count
 Preemption Wait Count is the number of context switches a thread experiences when it is excluded from execution by the OS scheduler due to thread preemption. Excessive number of thread context switches may negatively impact application performance. Apply optimization techniques to reduce synchronization contention and eliminate the problem. Explore Total Thread count and eliminate thread oversubscription changing the number of threads in the application.

OVERHEAD:

SW (System Wide) mode:
 Up to ~1,5x on desktop
 Up to ~2,0x on server

PP (Per Process) mode:
 Up to ~3,0x


```
$ perf record [-a] -N -B -T -R -F K -g --call-graph dwarf,1024 --switch-events \
  -e cycles,instructions,ref-cycles, \
  software/period=1,name=cs,config=0x3/Duk -- <futex|sleep0>
```

		futex		multithread sleep (0)	
	v4.17.0+	PP mode	SW mode	PP mode	SW mode
Intel Skylake 8C	Fedora 25	2,85x	1,96x	0,92x	0,87x
Intel Skylake EP 48C 2S	Ubuntu 17.04	2,80x	1,43x	1,30x	1,07x
Intel Broadwell 8C	Ubuntu 16.04 LTS	2,50x	1,81x	0,99x	0,91x
Intel Broadwell EP 44C 2S	Ubuntu server 17.04	5,50x	1,40x	1,22x	1,06x

Perf privileged users

<https://www.kernel.org/doc/html/latest/admin-guide/perf-security.html>

Documentation/admin-guide/perf-security.rst



Privileged
processes

euid == 0, CAP_SYS_ADMIN etc.



Perf privileged
processes

access to monitoring
bypass ***perf_event Paranoid***
and ***system limits***



Unprivileged
processes

access to monitoring
bound by ***perf_event Paranoid***
and ***system limits***

```
# ls -alhF
```

```
-rwxr-x--- 2 root perf_users 11M Oct 19 15:12 perf
```

```
# getcap perf
```

```
perf = cap_sys_ptrace,cap_sys_admin,cap_syslog+ep
```

Parallel record trace streaming

```
$ perf record --threads --dir --output-dir
```

- record prototype currently exists, no compression yet
- outperforms SERIAL and AIO in terms of runtime overhead:
 - NASA BT parallel benchmark
 - 14 events with sampling interval of 1ms (client) and 5ms (server)
 - stack size up to 32KiB for later dwarf based unwinding
- *report part is still at PoC stage:*
 - *mapping huge (~200GiB) Perf data directory content to memory is challenging*
 - *thus, distinct approach to loading of Perf data directory is required ...*

Thanks for attending!

Q/A

Linux Perf users: linux-perf-users@vger.kernel.org

Linux Perf developers: linux-kernel@vger.kernel.org

Contact: Alexey Budankov
[<alexey.budankov@linux.intel.com>](mailto:alexey.budankov@linux.intel.com)

Backup

Typed context switches

Grouping: Function / Call Stack

Function / Call Stack	CPU Time	Inactive Wait Time	Inactive Wait Count	
			Inactive Sync Wait Count	Preemption Wait Count
schedule			4	38,040
schedule			4	38,040
exit_to_usermode_loop			0	38,040
prepare_exit_to_usermode			0	38,038
do_syscall_64			0	2
_write	0s	0.000s	0	1
_getdents	0s	0.001s	0	1
io_schedule	0s	0.001s	1	0
futex_wait_queue_me	0s	22.478s	3	0

Inactive Sync Wait Count

Inactive Sync Wait Count is the number of context switches a thread experiences when it is excluded from execution by the OS scheduler due to synchronization. Excessive number of thread context switches may negatively impact application performance. Apply optimization techniques to reduce synchronization contention and eliminate the problem.

Preemption Wait Count

Preemption Wait Count is the number of context switches a thread experiences when it is excluded from execution by the OS scheduler due to thread preemption. Excessive number of thread context switches may negatively impact application performance. Apply optimization techniques to reduce synchronization contention and eliminate the problem. Explore Total Thread count and eliminate thread oversubscription changing the number of threads in the application.

Preemption Wait Time

Viewing 1 of 1 selected stack(s)

100.0% (1.679s)

vmlinux!schedule+0x29 - [unknown source file]

vmlinux!schedule+0x28 - [unknown source file]

vmlinux!exit_to_usermode_loop+0x56 - [unknown source file]

vmlinux!prepare_exit_to_usermode+0x74 - [unknown source file]

vmlinux!retint_user+0x7 - [unknown source file]

matrix.gcc.g.O3!multiply1+0x77 - multiply

matrix.gcc.g.O3!ThreadFunction+0x-fa9 - [unknown source file]

libpthread-2.28.so!start_thread+0xfd - [unknown source file]

libc-2.28.so!clone+0x-21fbc - [unknown source file]

Thread (TID: 31475)

Context Switches

Start: 8.338s Duration: 18.05ms

CPU: cpu_2

Reason: Preemption

CPU Time

5.2%

Spin and Overhead Time

0.0%

matrix.gcc.g.O3 (TID: 31459)

Context Switches

Start: 0.023s Duration: 22.062s

CPU: cpu_2

Reason: Synchronization

CPU Time

0.0%

Spin and Overhead Time

0.0%

Inactive Sync Wait Time

Viewing 1 of 1 selected stack(s)

100.0% (21.934s of 21.934s)

vmlinux!schedule+0x29 - [unknown source file]

vmlinux!schedule+0x28 - [unknown source file]

vmlinux!futex_wait_queue_me+0xb8 - [unknown source file]

vmlinux!futex_wait+0x13f - [unknown source file]

vmlinux!_x64_sys_futex+0x13b - [unknown source file]

vmlinux!do_syscall_64+0x5e - [unknown source file]

vmlinux!entry_SYSCALL_64_after_hwframe+0x43 - [unknown source file]

libpthread-2.28.so!_GI__pthread_timedjoin_ex+0x18c - [unknown source file]

matrix.gcc.g.O3!ParallelMultiply+0x13c - thrmodel.c:105

Parallel record trace streaming

```
$ perf record --threads --dir --output-dir ____bt.$stack_size.data \  
-N -B -T -R --call-graph dwarf,$stack_size --user-regs=ip,bp,sp \  
-e 'cpu/period=P,event=0x3c,name='\"CPU_CLK_UNHALTED.THREAD\"'/Duk,\ \  
cpu/period=P,umask=0x3,name='\"CPU_CLK_UNHALTED.REF_TSC\"'/Duk,\ \  
cpu/period=P,event=0xc0,name='\"INST_RETIRED.ANY\"'/Duk,\ \  
cpu/period=P,event=0x3c,umask=0x1,name='\"CPU_CLK_UNHALTED.REF_XCLK\"'/uk,\ \  
cpu/period=P,event=0x3c,umask=0x2,name='\"CPU_CLK_UNHALTED.ONE_THREAD_ACTIVE\"'/uk,\ \  
cpu/period=P,event=0xc2,umask=0x2,name='\"UOPS_RETIRED.RETIRE_SLOTS\"'/uk,\ \  
cpu/period=P,event=0xc7,umask=0x2,name='\"FP_ARITH_INST_RETIRED.SCALAR_SINGLE\"'/uk,\ \  
cpu/period=P,event=0xc7,umask=0x8,name='\"FP_ARITH_INST_RETIRED.128B_PACKED_SINGLE\"'/uk,\ \  
cpu/period=P,event=0xc7,umask=0x20,name='\"FP_ARITH_INST_RETIRED.256B_PACKED_SINGLE\"'/uk,\ \  
cpu/period=P,event=0xc7,umask=0x1,name='\"FP_ARITH_INST_RETIRED.SCALAR_DOUBLE\"'/uk,\ \  
cpu/period=P,event=0xc7,umask=0x4,name='\"FP_ARITH_INST_RETIRED.128B_PACKED_DOUBLE\"'/uk,\ \  
cpu/period=P,event=0xc7,umask=0x10,name='\"FP_ARITH_INST_RETIRED.256B_PACKED_DOUBLE\"'/uk,\ \  
cpu/period=P,event=0xb1,umask=0x10,name='\"UOPS_EXECUTED.X87\"'/uk,\ \  
cpu/period=P,event=0xb1,umask=0x1,name='\"UOPS_EXECUTED.THREAD\"'/uk\  
--clockid=monotonic_raw -- bt-mz.<B|C>
```


Patches

AIO: <https://marc.info/?l=linux-kernel&m=154149439404555&w=2>

NUMA: <https://marc.info/?l=linux-kernel&m=154817912621465&w=2>

Compression: <https://marc.info/?l=linux-kernel&m=155293062518459&w=2>

Event names: <https://marc.info/?l=linux-kernel&m=152809506802631&w=2>

Typed context switches: <https://marc.info/?l=linux-kernel&m=152325846408261&w=2>

Perf privileged users: <https://www.kernel.org/doc/html/latest/admin-guide/perf-security.html>

1ms uncore printing: <https://marc.info/?l=linux-kernel&m=152277952316450&w=2>

frame pointer on sample: <https://marc.info/?l=linux-kernel&m=152717113521558&w=2>

clockid frequency: <https://marc.info/?l=linux-kernel&m=153909580104443&w=2>

Optimized register set capture: <https://marc.info/?l=linux-kernel&m=155594387916976&w=2>

Augmented call stacks: <https://marc.info/?l=linux-kernel&m=156536377224480&w=2>

Decompression fix: <https://marc.info/?l=linux-kernel&m=156268372122638&w=2>

Overhead mitigation:

<https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=486adcea4a63bec206cba6f0d7f301fb945ae9d3>