PROFILING DATA STRUCTURES

perf + pahole

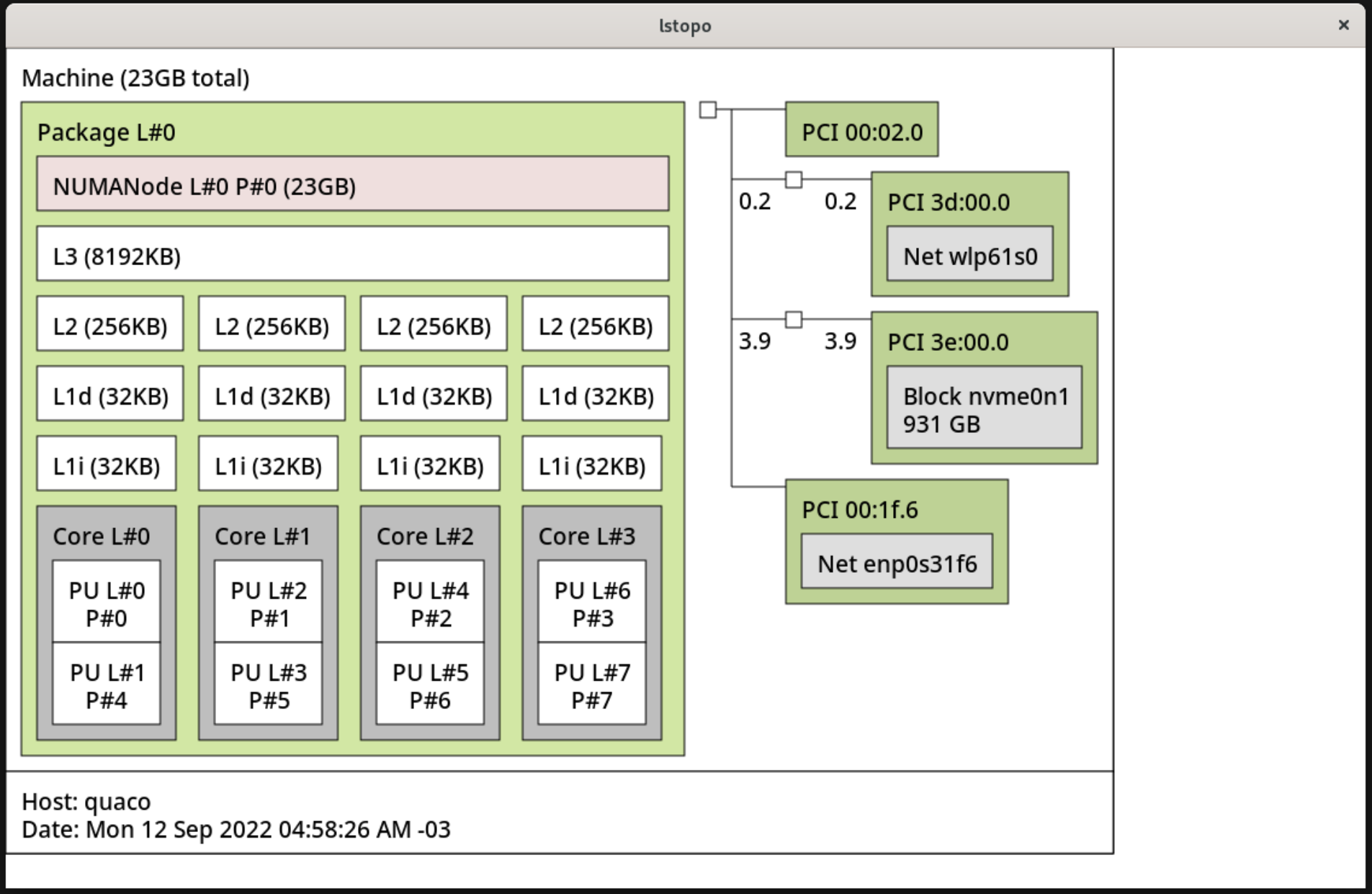
Arnaldo Carvalho de Melo
acme@redhat.com

WHAT IS THIS ABOUT?

- Data structures
- Optimal field ordering
- CPU Cache usage
- False Sharing
- Avoiding unintendend outcomes

WHY DO WE CARE?

- Cachelines
- Cache hierarchy
- Cache coherency
- L1, L2, LLC
- NUMA: Remote memory
- Load latencies


lstopo
Machine (23GB total)
Package L#0
NUMANode L#0 P#0 (23GB)
L3 (8192KB)
L2 (256KB)
L2 (256KB)
L2 (256KB)
L2 (256KB)
L1d (32KB)
L1d (32KB)
L1d (32KB)
L1d (32KB)
L1i (32KB)
L1i (32KB)
L1i (32KB)
L1i (32KB)
Core L#0
PU L#0 P#0
PU L#1 P#4
Core L#1
PU L#2 P#1
PU L#3 P#5
Core L#2
PU L#4 P#2
PU L#5 P#6
Core L#3
PU L#6 P#3
PU L#7 P#7
PCI 00:02.0
0.2
0.2
PCI 3d:00.0
Net wlp61s0
3.9
3.9
PCI 3e:00.0
Block nvme0n1
931 GB
PCI 00:1f.6
Net enp0s31f6
Host: quaco
Date: Mon 12 Sep 2022 04:58:26 AM -03

PROBLEMS?

- Non cache aligned arrays
- Close by read-mostly and write-mostly variables
- Lots fixed over the years
- Learn from it!

TEDIOUS

- Being done already
- Manually
- Tinkering
- Move, rebuild, perf stat it

MANUALLY DOING IT

- Look for "false sharing" on git log
- Eric Dumazet does it a lot

FIRST EXAMPLE

```
commit 91b6d325635617540b6a1646ddb138bb17cbd569
Author: Eric Dumazet
Date:   Mon Nov 15 11:02:39 2021 -0800

    net: cache align tcp_memory_allocated, tcp_sockets_allocated

    tcp_memory_allocated and tcp_sockets_allocated often share
    a common cache line, source of false sharing.
```

THE CHANGE

```
+++ b/net/ipv4/tcp.c
-atomic_long_t tcp_memory_allocated; // Current allocated memory
+atomic_long_t tcp_memory_allocated ____cacheline_aligned_in_smp; /

-struct percpu_counter tcp_sockets_allocated;
+struct percpu_counter tcp_sockets_allocated ____cacheline_aligned_
```

BCACHE_DEV_SECTORS_DIRTY_ADD()

- Always set_bit(stripe, full_dirty_stripes)
- Cacheline being invalidated constantly
- 100G dirty data with 20 threads
- 50 times slower
- Change to test_bit() first
- Mingzhe Zou
- 7b1002f7cfe581930f63787a0b3de0144e61ed55

THE CHANGE

```
+++ b/drivers/md/bcache/writeback.c
                sectors_dirty = atomic_add_return(s,
                                                  d->stripe_sectors_dirty + s
-               if (sectors_dirty == d->stripe_size)
-                       set_bit(stripe, d->full_dirty_stripes);
-               else
-                       clear_bit(stripe, d->full_dirty_stripes);
+               if (sectors_dirty == d->stripe_size) {
+                       if (!test_bit(stripe, d->full_dirty_stripes
+                               set_bit(stripe, d->full_dirty_stripe
+               } else {
+                       if (test_bit(stripe, d->full_dirty_stripes)
+                               clear_bit(stripe, d->full_dirty_stri
+               }
```

PAGE_COUNTER

- cgroup perf regression
- 'usage' and 'parent' on the same cacheline
- 'parent' is mostly read
- 'usage' is mostly written
- False sharing
- Feng Tang used 'perf c2c' to detect it
- https://lore.kernel.org/lkml/20201102091543.GM3109 debian
- 802f1d522d5fdaefc2b935141bc8fe03d43a99ab

THE CHANGE

```
+++ b/include/linux/page_counter.h
@@ -12,7 +12,6 @@ struct page_counter {
        unsigned long low;
        unsigned long high;
        unsigned long max;
-       struct page_counter *parent;

        /* effective memory.min and memory.min usage tracking */
        unsigned long emin;
@@ -26,6 +25,12 @@ struct page_counter {
        unsigned long watermark;
        unsigned long failcnt;
+       /*
+        * 'parent' is placed here to be far from 'usage' to reduce
+        * false sharing, as 'usage' is written mostly while parent
+        * frequently read for cgroup's hierarchical counting natur
+        */
+       struct page_counter *parent;
 };
```

THE NEW STRUCT

```
$ pahole page_counter
struct page_counter {
        atomic_long_t              usage;                /*   0   8 */
        long unsigned int          min;                  /*   8   8 */
        long unsigned int          low;                  /*  16   8 */
        long unsigned int          high;                 /*  24   8 */
        long unsigned int          max;                  /*  32   8 */
        long unsigned int          emin;                 /*  40   8 */
        atomic_long_t              min_usage;            /*  48   8 */
        atomic_long_t              children_min_usage;   /*  56   8 */
        /* --- cacheline 1 boundary (64 bytes) --- */
        long unsigned int          elow;                 /*  64   8 */
        atomic_long_t              low_usage;            /*  72   8 */
        atomic_long_t              children_low_usage;   /*  80   8 */
        long unsigned int          watermark;            /*  88   8 */
        long unsigned int          failcnt;              /*  96   8 */
        struct page_counter *parent;                     /* 104   8 */

        /* size: 112, cachelines: 2, members: 14 */
        /* last cacheline: 48 bytes */
};
$
```

LAYOUT

- Type info: DWARF, BTF
- BTF now always available
- per-cpu variable types

BTF

- Compact
- Yeah, BPF stuff
- Used with BPF's CO-RE
- And other BPF features
- /sys/kernel/btf/

SMALL

```
$ cd /sys/kernel/btf
$ ls -lh vmlinux
-r--r--r--. 1 root root 5.1M Sep  8 20:38 vmlinux
$
$ ls -lh i915
-r--r--r--. 1 root root 556K Sep 12 09:29 i915
$
$ ls -1 | wc -l
204
$ lsmod | wc -l
204
$ lsmod | head -2
Module                  Size  Used by
sctp                  434176  28
$
```

pahole

```
$ pahole page_pool
struct page_pool {
        struct page_pool_params  p;                    /* 0  56 */
        struct delayed_work      release_dw;           /* 56  88 */
        /* XXX last struct has 4 bytes of padding */

        /* --- cacheline 2 boundary (128 bytes) was 16 bytes ago --- */
        void                     (*disconnect)(void *); /* 144  8 */
        long unsigned int        defer_start;          /* 152  8 */
        long unsigned int        defer_warn;           /* 160  8 */
        u32                      pages_state_hold_cnt;  /* 168  4 */
        unsigned int             frag_offset;          /* 172  4 */
        struct page *            frag_page;            /* 176  8 */
        long int                 frag_users;           /* 184  8 */
        /* --- cacheline 3 boundary (192 bytes) --- */
        u32                      xdp_mem_id;          /*   192     4 */
        /* XXX 60 bytes hole, try to pack */

        /* --- cacheline 4 boundary (256 bytes) --- */
        struct pp_alloc_cache    alloc __attribute__((__aligned__(64))); /*   256  1032 */
        /* XXX 56 bytes hole, try to pack */

        /* --- cacheline 21 boundary (1344 bytes) --- */
        struct ptr_ring          ring __attribute__((__aligned__(64))); /*  1344   192 */
        /* XXX last struct has 48 bytes of padding */
        /* --- cacheline 24 boundary (1536 bytes) --- */
        atomic_t                 pages_state_release_cnt; /*  1536     4 */
        refcount_t               user_cnt;             /*  1540     4 */
        u64                      destroy_cnt;          /*  1544     8 */
        /* size: 1600, cachelines: 25, members: 15 */
        /* sum members: 1436, holes: 2, sum holes: 116 */
        /* padding: 48 paddings: 2, sum paddings: 52 */
        /* forced alignments: 2, forced holes: 2, sum forced holes: 116 */
} __attribute__((__aligned__(64)));
$
```

CPU HELPS

- Intel PEBS
- AMD IBS
- ARM CoreSight

EXAMPLE PEBS DOC

18.8.1.2 Load Latency Performance Monitoring Facility

The load latency facility provides software a means to charaterize the average load latency to different levels of cache/memory hierarchy. This facility requires processor supporting enhanced PEBS record format in the PEBS buffer, see Table 18-23.

This field measures the load latency from load's first dispatch of till final data writeback from the memory subsystem. The latency is reported for retired demand load operations and in core cycles (it accounts for re-dispatches).

WHAT DO WE HAVE IN PERF?

- perf stat
- perf mem
- perf c2c
- Several articles about using it
- Improve upon this foundation

PERF MEM

- loads, stores
- load latency
- Stephane Eranian
- Jiri Olsa, Kan Liang, others

GLOBAL STATS

```
# perf mem record -a sleep 1
#
# perf mem -t load report --sort=mem --stdio
# Total Lost Samples: 0
#
# Samples: 51K of event 'cpu/mem-loads,ldlat=30/P'
# Total weight : 4819902
# Sort order   : mem
#
# Overhead       Samples  Memory access
# ........  ............  ........................
    44.87%         20217  LFB or LFB hit
    27.30%         18618  L3 or L3 hit
    22.53%         11712  L1 or L1 hit
     4.85%           637  Local RAM or RAM hit
     0.25%             1  Uncached or N/A hit
     0.20%           188  L2 or L2 hit
     0.00%            35  L3 miss
```

WORKLOAD STATS

```
# perf mem record sleep 1
#
# perf mem -t load report --sort=mem --stdio
# Total Lost Samples: 0
#
# Samples: 16  of event 'cpu/mem-loads,ldlat=30/P'
# Total weight : 1556
# Sort order   : mem
# Overhead       Samples  Memory access
# ........  ............  ......................
#
    64.52%             8  LFB or LFB hit
    14.07%             4  L1 or L1 hit
    11.05%             3  L3 or L3 hit
    10.35%             1  Local RAM or RAM hit
```

A FAMILIAR WORKLOAD

```
$ make -j8 O=../build/allmodconfig/
make[1]: Entering directory '/home/acme/git/build/allmodconfig'

# perf mem record sleep 1m
[ perf record: Woken up 1 times to write data ]
[ perf record: Captured and wrote 0.037 MB perf.data (20 samples) ]
#
```

PERF MEM REPORT

```
# perf mem report --stdio
# Total Lost Samples: 0
#
# Samples: 11  of event 'cpu/mem-loads,ldlat=30/P'
# Total weight : 2155
# Sort order   : local_weight,mem,sym,dso,symbol_daddr,dso_daddr
#
#              Local Mem
#Overhead Weig Access   Symbol              Sh Object   Data Symbol        Data Obj
#........ .... ........ ..................... .........   .................   ..........
   23.94% 516  LocalRAM copy_page           [kernel]    0xffff8d42228ea900 [unknown]
   15.31% 330  LFB      flush_signal_handle [kernel]    0xffff8d3f976020a0 [unknown]
   14.66% 316  LFB      strlen              [kernel]    0xffffffff9b5f4cd3 [kernel].ro
   13.36% 288  LFB      _dl_relocate_object ld-linux.so 0x00007f6ccdc23068 libc.so.6
   11.46% 247  LFB      next_uptodate_page  [kernel]    0xffffe401957e4df4 [unknown]
    7.33% 158  LFB      copy_page           [kernel]    0xffff8d41f2dae920 [unknown]
    4.04% 87   LFB      unlock_page_memcg   [kernel]    0xffffe4019333d8b8 [unknown]
    3.06% 66   L1       check_preemption_di [kernel]    0xffffa8e8622ffc80 [unknown]
    2.69% 58   LFB      perf_output_begin   [kernel]    0xffff8d3f52a1b01c [unknown]
    2.13% 46   L3       task_work_run       [kernel]    0xffff8d3f4a9c802c [unknown]
    2.00% 43   L1       kmem_cache_alloc_tr [kernel]    0xffffa8e8622ffbc8 [unknown]
```

PERF C2C

- Cache to Cache
- False Sharing
- Cachelines tugged
- Resolves global variables
- Should resolve its type and field
- Using BTF

ORIGINS

- Dick Fowles
- HP-UX had something similar
- Joe Mario
- Don Zickus
- Jiri Olsa

C2C OUTPUT

- Cachelines where false sharing was detected
- Readers and writers and cacheline offsets
- pid, tid, instruction addr, function name, binary object names
- Source file and line number
- Average load latency
- NUMA nodes and CPUs involved

PERF C2C

```
# perf c2c record -a sleep 1
[ perf record: Woken up 1 times to write data ]
[ perf record: Captured and wrote 7.787 MB perf.data (2450 samples)
# perf evlist
cpu/mem-loads,ldlat=30/P
cpu/mem-stores/P
dummy:HG
#
```

WHATS IN THERE?

```
# perf script --cpu 4 --pid 0 | head
 swapper 0 [4] 319242.043904:    58 cpu/mem-loads,ldlat=30/P: ffff8d3e49c0e688 11868100242 |OP LOAD|LVL LFB or LFB hit|SNP None|TLB L1 or L2 hit|LCK N
 swapper 0 [4] 319242.142295:    39 cpu/mem-loads,ldlat=30/P: ffff8d44865f2408 10268100142 |OP LOAD|LVL L1 or L1 hit|SNP None|TLB L1 or L2 hit|LCK No|
 swapper 0 [4] 319242.143587: 99614         cpu/mem-stores/P: ffff8d4486500028     5080184 |OP STORE|LVL L1 miss|SNP N/A|TLB N/A|LCK N/A|BLK  N/A
 swapper 0 [4] 319242.174494:    33 cpu/mem-loads,ldlat=30/P: ffff8d3f595ddc38 11a68201042 |OP LOAD|LVL Local RAM or RAM hit|SNP Hit|TLB L1 or L2 hit|
 swapper 0 [4] 319242.178002:    27 cpu/mem-loads,ldlat=30/P: ffff8d44865312c0 10668100842 |OP LOAD|LVL L3 or L3 hit|SNP None|TLB L1 or L2 hit|LCK No|
 swapper 0 [4] 319242.212148:    23 cpu/mem-loads,ldlat=30/P: ffff8d44865322e8 10668100842 |OP LOAD|LVL L3 or L3 hit|SNP None|TLB L1 or L2 hit|LCK No|
 swapper 0 [4] 319242.217357:    18 cpu/mem-loads,ldlat=30/P: ffff8d4486532490 10268100142 |OP LOAD|LVL L1 or L1 hit|SNP None|TLB L1 or L2 hit|LCK No|
 swapper 0 [4] 319242.220573:    15 cpu/mem-loads,ldlat=30/P: ffff8d3f4f35f218 11868100242 |OP LOAD|LVL LFB or LFB hit|SNP None|TLB L1 or L2 hit|LCK N
 swapper 0 [4] 319242.240176:    15 cpu/mem-loads,ldlat=30/P: ffff8d3f6b617be0 10650100842 |OP LOAD|LVL L3 or L3 hit|SNP None|TLB L2 miss|LCK No|BLK
 swapper 0 [4] 319242.243441:  8849         cpu/mem-stores/P: ffff8d3f40c2b1a4     5080144 |OP STORE|LVL L1 hit|SNP N/A|TLB N/A|LCK N/A|BLK  N/A
#
```

WHAT WAS REALLY ASKED?

```
# perf evlist -v | head -1
cpu/mem-loads,ldlat=30/P: type: 4, size: 128, config: 0x1cd, \
{ sample_period, sample_freq }: 4000, \
sample_type: IP|TID|TIME|ADDR|ID|CPU|PERIOD|DATA_SRC|PHYS_ADDR| \
             WEIGHT_STRUCT, \
read_format: ID, disabled: 1, inherit: 1, freq: 1, precise_ip: 3, \
sample_id_all: 1, { bp_addr, config1 }: 0x1f
#
```

PERF_SAMPLE_DATA_SRC

- PEBS Load Latency
- opcode: load, store, prefetch, exec
- mem level: l1, l2, l3, l4, lfb, ram, pmem, etc
- tlb: not available, hit, miss, l1, l2, hw walker, OS fault handler
- snoop: no snoop, hit, miss, hit modified
- lock: not available, locked instruction

PERF_SAMPLE_WEIGHT_STRUCT

- hardware provided number
- How expensive the sampled action represents
- For profiler to scale samples

C2C PERF.DATA HEADER

```
# perf report --header-only
# captured on    : Fri Sep  9 11:10:04 2022
# hostname : quaco
# os release : 5.18.17-200.fc36.x86_64
# perf version : 6.0.rc3.gfaf59ec8c3c3
# arch : x86_64
# nrcpus online : 8
# nrcpus avail : 8
# cpudesc : Intel(R) Core(TM) i7-8650U CPU @ 1.90GHz
# total memory : 24487388 kB
# cmdline : /home/acme/bin/perf c2c record -a sleep 1m
# event : name = cpu/mem-loads,ldlat=30/P, freq = 4000,
  sample_type = IP|TID|TIME|ADDR|ID|CPU|PERIOD|DATA_SRC|\
                PHYS_ADDR|WEIGHT_STRUCT
# event : name = cpu/mem-stores/P, freq = 4000,
  sample_type = IP|TID|TIME|ADDR|ID|CPU|PERIOD|DATA_SRC|\
                PHYS_ADDR|WEIGHT_STRUCT
```

KEY METRICS

```
# perf c2c report --stats
  Total records                     :    3223429
  Locked Load/Store Operations      :     112673
  Load Operations                   :    1387118
  Loads - uncacheable               :          1
  Loads - IO                        :          4
  Loads - Miss                      :        142
  Loads - no mapping                :       2350
  Load Fill Buffer Hit              :     455747
  Load L1D hit                      :     264355
  Load L2D hit                      :      29304
  Load LLC hit                      :     534642
  Load Local HITM                   :        629
  Load Remote HITM                  :          0
  Load Remote HIT                   :          0
```

PERF C2C CONTENDED CACHELINES VIEW

```
# perf c2c report --stdio
=================================================
           Shared Data Cache Line Table
=================================================
#
#      -- Cacheline --    -- Load Hitm  ----   Tot  Total   Total  -- Stores --
# Idx           Address    Hitm  Tot LclHitm   rec  Loads  Stores  L1Hit L1Miss
# ...  ................   .....  ... .......   ...  .....  ......  .............
#
    0  ffff8d449e7d6380   8.43%   53      53   510    499      11     11      0
    1  ffff8d4058209340   6.20%   39      39   371    135     236    223     13
    2  ffff8d449e7ff400   5.88%   37      37   501    479      22     22      0
    3  ffffffff9bf53980   4.93%   31      31   233    208      25     24      1
    4  ffff8d3f49ebd280   3.18%   20      20   162    153       9      9      0
    5  ffff8d3f420d4880   2.86%   18      18   126    121       5      5      0
```

TUGGED CACHELINE

```
Cacheline 0xffff8d449e7ff400
-HITM-     CL                    --- cycles ---  Tot  cpu
LclHitm   Off      Code address  lcl hitm  load  rec  cnt   Symbol Object      Source:Line
 97.30%   0x0 0xffffffff9a2d293b      113    44  454    8 __mod_node_page_state [kernel] vmstat.c:379
  0.00%   0x8 0xffffffff9a2d29bb        0   112  40     8 __mod_node_page_state [kernel] atomic64_64.h:46
  2.70%  0x18 0xffffffff9a2d2be5      959   103   2     2 refresh_cpu_vm_stats  [kernel] atomic64_64.h:46
```

NEXT STEPS

- Look at the CL Off (Cacheline Offset)
- Three fields being accessed
- Two are with local HITM
- Find out what is the data structure
- By looking at the functions, source:line

VMSTAT.C:379

```
$ perf probe -L vmstat.c:379 | head

    379         struct per_cpu_nodestat __percpu *pcp = pgdat->per_cpu_nodestats;
    380         s8 __percpu *p = pcp->vm_node_stat_diff + item;
    381         long x;
                long t;

                if (vmstat_item_in_bytes(item)) {
                        /*
                         * Only cgroups use subpage accounting right now; at
                         * the global level, these items still change in
$
```

LOOKING AT PGDAT

```
$ pfunct __mod_node_page_state
void __mod_node_page_state(struct pglist_data * pgdat,
                           enum node_stat_item item, long int delta);
$
$ pahole pglist_data | grep -B2 -A6 per_cpu_nodestats
    /* --- cacheline 2704 boundary (173056 bytes) --- */
    struct zone_padding      _pad2_;             /* 173056   0 */
    struct per_cpu_nodestat *per_cpu_nodestats
                __attribute__((__aligned__(64))); /* 173056   8 */
    atomic_long_t            vm_stat[41];        /* 173064 328 */

    /* size: 173440, cachelines: 2710, members: 32 */
    /* sum members: 173309, holes: 6, sum holes: 83 */
    /* padding: 48 */
    /* forced alignments: 2 */
$
```

CODE INSPECTION

- Manual steps
- Streamlining the current UI:
- Should go from the c2c TUI
- To the 'perf annotate' browser
- Positioning at that source:line
- ENTER on type: go to a pahole browser

REFRESH_CPU_VM_STATS

```
# perf annotate --stdio2 refresh_cpu_vm_stats
<SNIP>
refresh_cpu_vm_stats() /usr/lib/debug/lib/modules/5.18.17-200.fc36.x86_64/vmlinux
             ffffffff812d29e0 :
             static int refresh_cpu_vm_stats(bool do_pagesets)
             struct pglist_data *pgdat;
             struct zone *zone;
<SNIP>
             for_each_online_pgdat(pgdat) {
<SNIP>
        1f3:   cmpxchg %r8b,%gs:(%rdx)
             ↑ jne     1f3
               movsbl  %al,%edi
             if (v) {
               test    %edi,%edi
             ↓ je      20b
             atomic_long_add(v, &pgdat->vm_stat[i])
```

BEING MINDFUL

```
* Update the zone counters for the current cpu.

* Note that refresh_cpu_vm_stats strives to only access node local
* memory.  The per cpu pagesets on remote zones are placed in the
* memory local to the processor using that pageset. So the loop over
* all zones will access a series of cachelines local to the processor.

* The call to zone_page_state_add updates the cachelines with the stats
* in the remote zone struct as well as the global cachelines with the
* global counters. These could cause remote node cache line bouncing
* and will have to be only done when necessary.

* The function returns the number of global counters updated.
```

PAHOLE TODO

- We have --hex, from struct start
- Implement per-cacheline offsets in pahole output
- For multi cacheline structs
- Like pgdat in the previous example

WHAT DO WE SEE?

- Code
- Source code/line
- All focused where things happened
- Not on what data
- But looking at where it happens
- Helps figuring out the data structure accessed

NO RMAP FOR DATA SYMBOLS?

- Global vars: ok
- SLAB: can we do it?
- From code?
- Going back to args
- DWARF location expressions
- LBR?

PERF LOCK CONTENTION -B

- Gets some backtraces
- Identifies locks by backtrace sigs
- Good enough?

A DETOUR

- perf record + report is too costly
- In kernel aggregation should be supported
- perf kmem, kwork, bcounters do it
- BPF way of doing things
- perf using it

PERF BPF SKELS

```
$ wc -l tools/perf/util/bpf_skel/*.bpf.c
  191 tools/perf/util/bpf_skel/bperf_cgroup.bpf.c
   78 tools/perf/util/bpf_skel/bperf_follower.bpf.c
   55 tools/perf/util/bpf_skel/bperf_leader.bpf.c
   92 tools/perf/util/bpf_skel/bpf_prog_profiler.bpf.c
  116 tools/perf/util/bpf_skel/func_latency.bpf.c
  383 tools/perf/util/bpf_skel/kwork_trace.bpf.c
  175 tools/perf/util/bpf_skel/lock_contention.bpf.c
  273 tools/perf/util/bpf_skel/off_cpu.bpf.c
 1363 total
$
```

PERF LOCK CONTENTION -B

```
 $ sudo perf lock contention -b
^C
contended  total wait   max wait  avg wait      type  caller

        42   192.67 us  13.64 us   4.59 us  spinlock  queue_work_on+0x20
        23    85.54 us  10.28 us   3.72 us  spinlock  worker_thread+0x14a
         6    13.92 us   6.51 us   2.32 us     mutex  kernfs_iop_permission+0x30
         3    11.59 us  10.04 us   3.86 us     mutex  kernfs_dop_revalidate+0x3c
         1     7.52 us   7.52 us   7.52 us  spinlock  kthread+0x115
         1     7.24 us   7.24 us   7.24 us  rwlock:W  sys_epoll_wait+0x148
         2     7.08 us   3.99 us   3.54 us  spinlock  delayed_work_timer_fn+0x1b
         1     6.41 us   6.41 us   6.41 us  spinlock  idle_balance+0xa06
         2     2.50 us   1.83 us   1.25 us     mutex  kernfs_iop_lookup+0x2f
         1     1.71 us   1.71 us   1.71 us     mutex  kernfs_iop_getattr+0x2c
...
```

PERF SAMPLING IN BPF

- We have BPF programs counting events
- That perf then treats as a counter (evsel)
- Now for sampling
- Do the c2c aggregation in a BPF program
- Until control+C
- Or some other existing perf stop method
- End of a control workload (sleep), etc
- Recent discussion: Namhyung Kim and Song Liu

BTF

- per-cpu variables
- patch to record all variables
- And have it in a kernel module
- Load it and get access to type info
- For all variables

CODE ANNOTATION

- perf annotate
- Source code
- Machine code
- Where events take place

perf annotate

```
Samples: 6M of events 'anon group { cycles:P, L1-icache-load-misses }', 4000 Hz, Event count (approx.): 42093638433
bpf_prog_0bc3fc9d11754ba1_sys_enter  bpf_prog_0bc3fc9d11754ba1_sys_enter [Percent: local period]
Percent
 54.68   2.34       int sys_enter(struct syscall_enter_args *args)
  0.71  68.27         push   %rbp
  0.23   0.20         mov    %rsp,%rbp
  6.46   0.11         sub    $0x200,%rsp
  0.55  13.91         push   %rbx
  0.46   1.14         push   %r13
  0.33   1.12         push   %r14
  0.55   0.55         push   %r15
  0.03   1.38         pushq  $0x0
                      mov    %rdi,%rbx
                         return bpf_get_current_pid_tgid();
  0.54   0.15       → callq  *ffffffffe017a907
  1.41   0.04         mov    %eax,-0x8(%rbp)
  0.02   3.20         mov    %rbp,%rsi

                      add    $0xfffffffffffffff8,%rsi
                         return bpf_map_lookup_elem(pids, &pid) != NULL;
  0.11   0.18         movabs $0xffff8b6b8418c800,%rdi
  1.28   0.02       → callq  *ffffffffe017c2a7
  0.02   0.21         cmp    $0x0,%rax
  0.25   0.00       ↓ je     3f
  0.10   0.05         add    $0x38,%rax
  0.54   0.87  3f:   mov    %rax,%rdi
  0.02   0.43         xor    %eax,%eax
                         if (pid_filter__has(&pids_filtered, getpid()))
  0.05   0.00         cmp    $0x0,%rdi
                    ↓ jne    db
  0.55   0.12         xor    %edi,%edi
Press 'h' for help on key bindings
```

nospectre_v1 + nospectre_v2

```
Samples: 954K of events 'anon group { cycles:P, L1-icache-load-misses }', 4000 Hz, Event count (approx.): 77077306917
bpf_prog_0bc3fc9d11754ba1_sys_enter  bpf_prog_0bc3fc9d11754ba1_sys_enter [Percent: local period]
Percent                 int sys_enter(struct syscall_enter_args *args)
 12.07   14.45          push   %rbp
  0.06   14.93          mov    %rsp,%rbp
  6.19    0.00          sub    $0x200,%rsp
  1.72    3.90          push   %rbx
  0.97    3.70          push   %r13
  0.00    0.21          push   %r14
  2.08    0.76          push   %r15
  0.00    4.80          pushq  $0x0
  0.00    1.03          mov    %rdi,%rbx
                           return bpf_get_current_pid_tgid();
  8.76    0.00        → callq  *ffffffffd7d26567
  8.13    3.65          mov    %eax,-0x8(%rbp)
  0.00   11.17          mov    %rbp,%rsi

                        add    $0xfffffffffffffff8,%rsi
                           return bpf_map_lookup_elem(pids, &pid) != NULL;
  2.08    0.00          movabs $0xffff9a6f1c0d9400,%rdi
  1.18    0.78        → callq  *ffffffffd7d27f07
  0.16    0.00          cmp    $0x0,%rax
  0.27    0.00        ↓ je     3f
  0.00    0.65          add    $0x38,%rax
  2.76    1.18  3f:     mov    %rax,%rdi
  0.00    9.61          xor    %eax,%eax
                           if (pid_filter__has(&pids_filtered, getpid()))
  0.21    0.31          cmp    $0x0,%rdi
  0.66    0.00        ↓ jne    db
  0.62    0.84          xor    %edi,%edi
                           int key = 0;
  0.57    1.02          mov    %edi,-0x4(%rbp)
  0.00    0.69          mov    %rbp,%rsi

                        add    $0xfffffffffffffffc,%rsi
                           return bpf_map_lookup_elem(&augmented_args_tmp, &key);
Press 'h' for help on key bindings
```

PERF + PAHOLE

- Data structure annotation
- Show HITM causing fields
- Mapping data memory to a variable
- Finding its type
- BTF info for variables: types
- Function signatures: arg types

THE END

- https://www.linaro.org/blog/using-the-arm-statistical-profiling-extension-to-detect-false-cache-line-sharing/
- https://joemario.github.io/blog/2016/09/01/c2c-blog/
- https://frankdenneman.nl/2016/07/11/numa-deep-dive-part-3-cache-coherency/
- https://perf.wiki.kernel.org/index.php/Useful_Links
- acme@kernel.org
- https://twitter.com/acmel