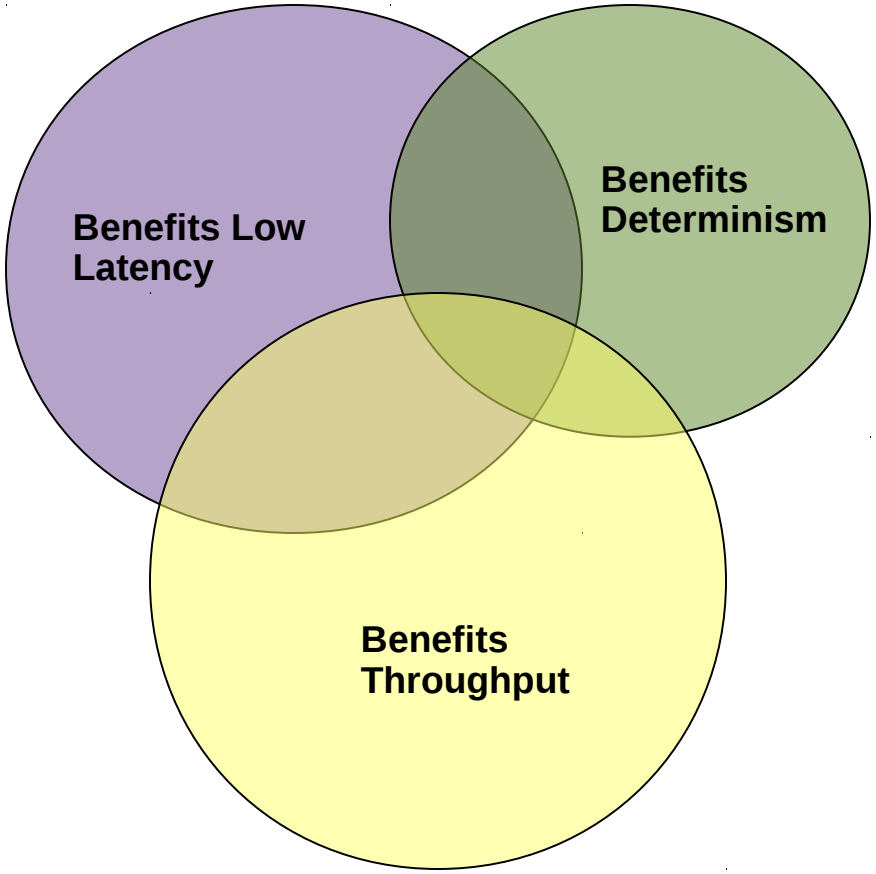
Where Are You Slow And Why Are You Slow?

– Tools and Techniques for Low Latency Evaluation

Nivedita Singhvi
April 30, 2012

Low Latency, throughput, determinism – a trade-off



Sometimes, design and configuration choices benefit the goals of low latency, throughput, and even determinism.

More often, these are conflicting goals and a choice needs to be made to priotitize what's more important for the workload and environment.

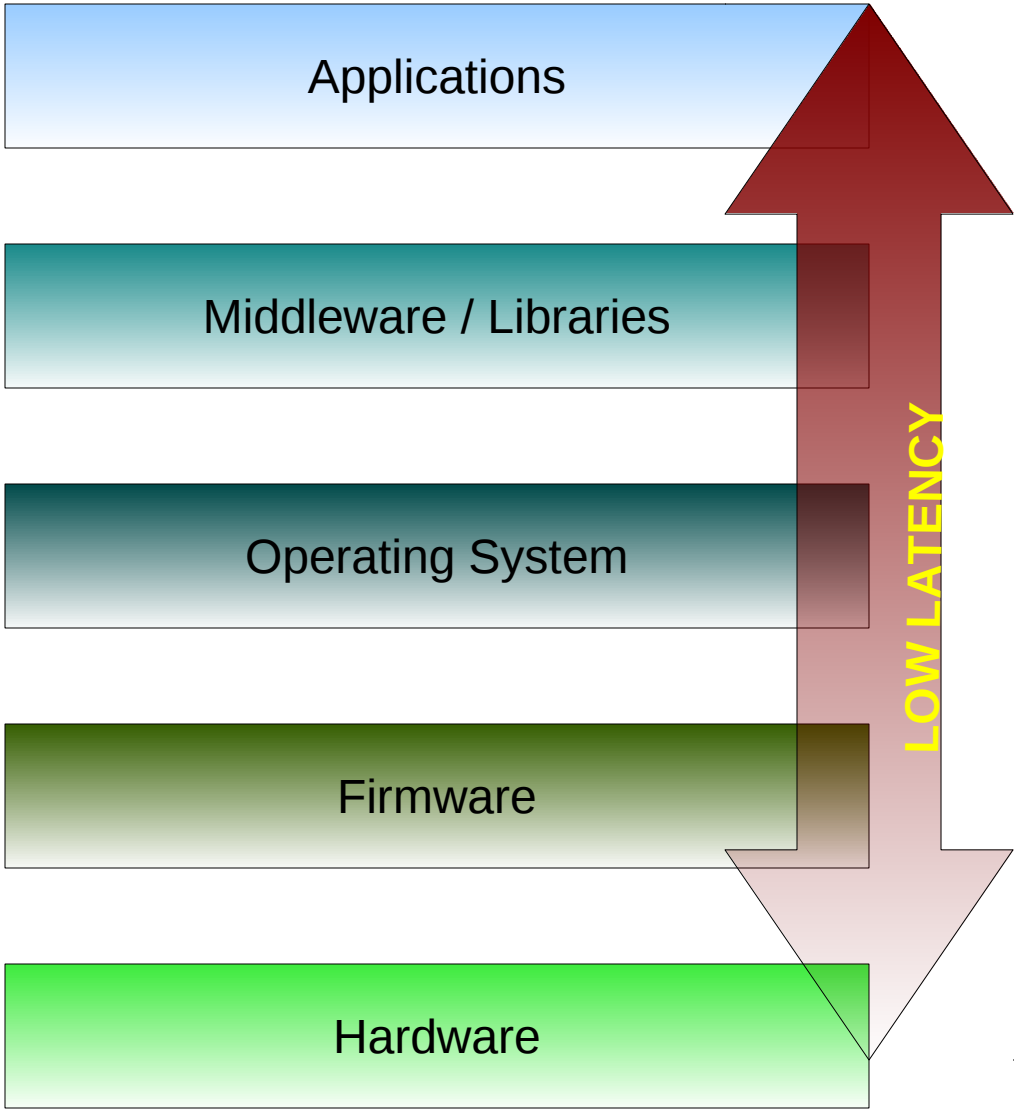
And don't forget...

All very load dependent!

Need Holistic Stack Tuning

Consolidation can create conflicts between workloads with different needs

Workload Prioritization...

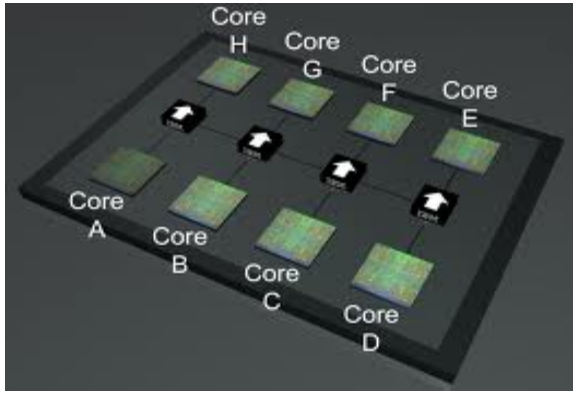


Hardware provisioning considerations

No amount of tuning will help if lack of adequate provisioning for peak loads, hardware limitations

of Cores

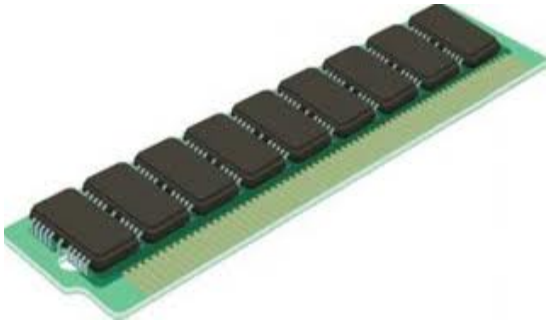
Scheduling overhead mitigated by adequate core count



Memory

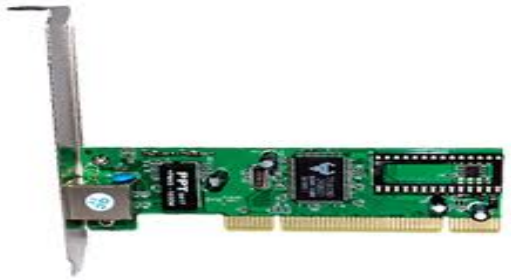
Memory management latencies mitigated by excess memory

(page faults, swapping, fragmentation, ...)



Interconnect technology

Infiniband
RDMA
Ethernet
....



Is your hardware / firmware a source of latency spikes?

FIRMWARE

HARDWARE

Latency Source?

"hwlatdetect" test *

rt-eval pkg: hwlatdetect output

```
hwlatdetect:  test duration 172800 seconds
   parameters:
        Latency threshold: 1us
        Sample window:     1000000us
        Sample width:      999999us
     Non-sampling period:  1us
        Output File:       None

Starting test
test finished
Max Latency: 2us
Samples recorded: 1
Samples exceeding threshold: 1
```

SMI Triggers

(very few, recent IBM platforms)

- Memory Errors
- PCI Errors
- CPU QPI Link Errors

* Real-Time Kernel

More about hwlatdetect and rt-tests (maintained by Red Hat)

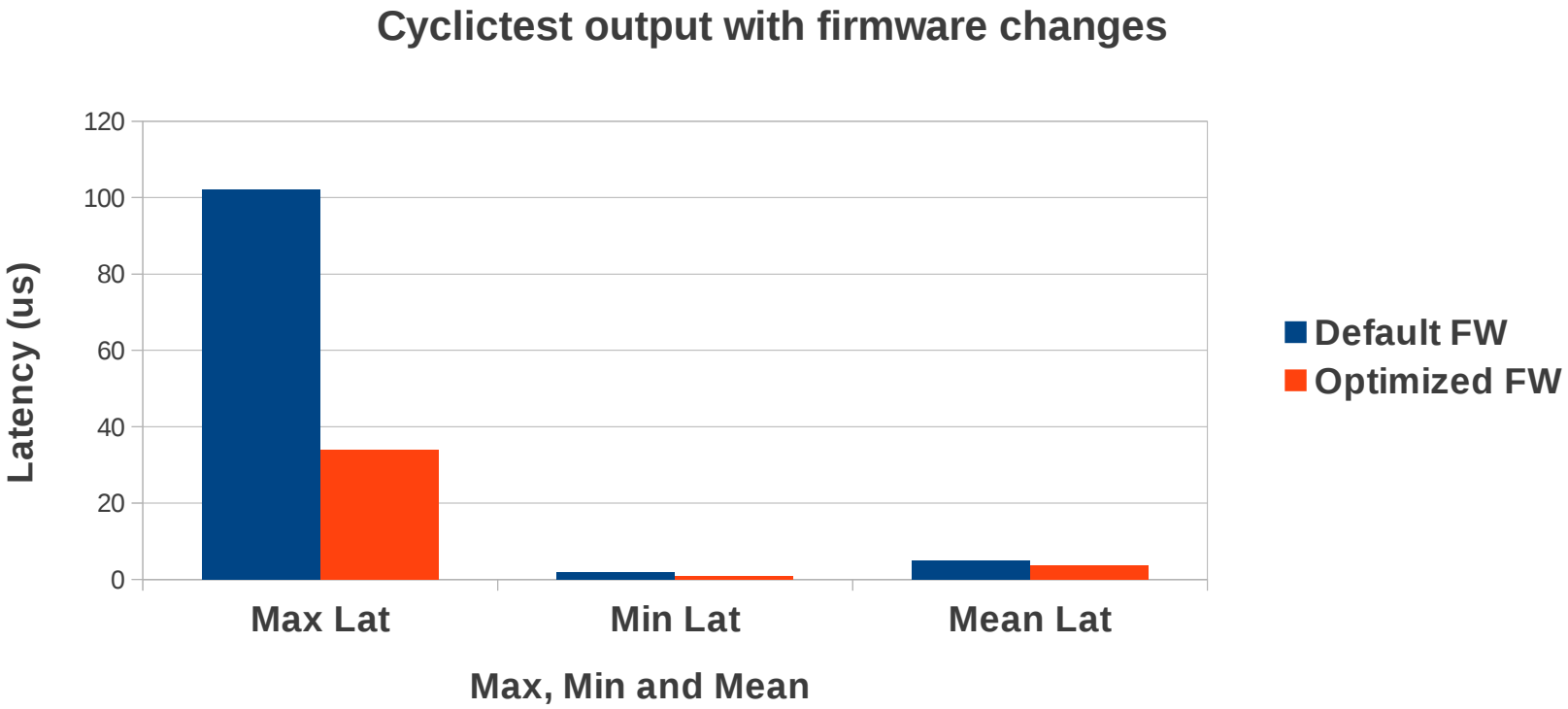
- hwlatdetect
  - output in /debugfs/hwlat
  - smi_count, max_sample_us, ...

  kernel module (hwlat.ko), python script (hwlatdetect.py)
  - stop_machine(), polls the TSC counter repeatedly
  - requires port to standard kernel

- Where to get rt-tests

  # git clone https://github.com/clrkwllms/rt-tests
  - Meant for the real-time kernel from Red Hat
  - several other microbenchmarks for real-time for latency use

Firmware tuning impact – a drastic picture (real-time kernel)



cyclictest with load – optimized and non-optimized firmware settings
cyclictest -i100 -qm -d0 -h 2000 -p95 –smp
your mileage will vary!

More about cyclictest (maintained by Clark Williams, Red Hat)

- Cyclictest
  - scheduling microbenchmark
  - sleep 5s; how accurately am I scheduled on CPU?
    load, time/timer granularity, ...
  - very, very large number of options, new features
  - what latencies might a thread see at different real-time priorities?
- Where to get cyclictest
  # git clone https://github.com/clrkwllms/rt-tests
  - was meant for the real-time kernel
  - original from Thomas Gleixner

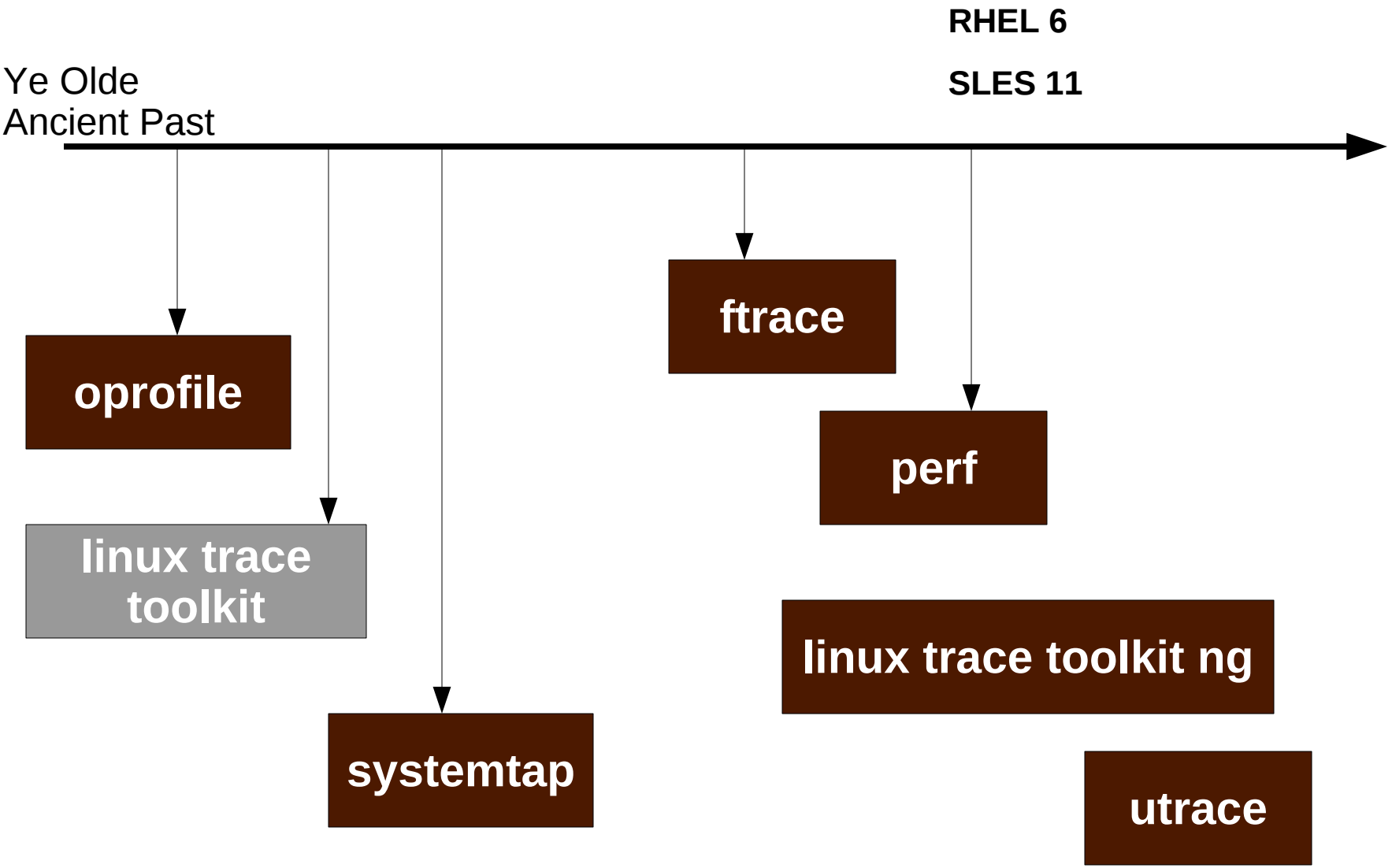
Key firmware settings for low latency – x3650 M4 example

Variable	Description
processors.c-states	disable
processors.hyper-threading	disable
processors.C1EnhanceMode	disable
processors.TurboMode	enable
imm.thermalmodepolicy	performance
processors.intelvirtualizationtechnology	disable
processors.apilinkfrequency	max performance
memory.memoryspeed	max performance
memory.patrolscrub	disable
memory.powermanagement	disable

Key firmware settings for low latency – x3650 M4 example

Variable	Description
memory.pagepolicy	adaptive
memory.ckethrottling	disable
power.activeenergymanager	capping disabled
power.platformcontrolledtype	maximum performance
power.workloadconfiguration	i/o sensitive
...	

Performance diagnostic tools – no shortage



Using SystemTap for common low latency workload tasks

- Complex, rich, powerful
- Investment of time for full exploitation
- Very kernel-specific (breaks with changes in kernel)

- http://sourceware.org/systemtap/examples/
  - quick use
  - to use systemtap scripts/tapsets

    systemtap-runtime
    kernel-debuginfo
    kernel-debuginfo-common
    kernel-devel

systemtap – cyclethief.stp (what's interrupting you?)

```
 # stap cycle_thief.stp
^C
task 0 migrated: 43
task 0 on processor (us):
 value |-------------------------------------------------- count
     0 |                                                    0
     1 |                                                    0
     2 |                                                   19
     4 |                                                    5
     8 |                                                    2
    16 |                                                    1
    32 |                                                   7
    64 |                                                    4
   128 |                                                    3
   256 |                                                    5
   512 |                                                36
  1024 |                                                30
  2048 |                                                    274
  4096 |                                                        117
  8192 |                                                        181
 16384 |                                                         272
 32768 |                                                        236
 65536 |                                                        104
131072 |                                                      0
262144 |                                                      0
```

systemtap – cyclethief.stp (what's interrupting you?)

```
 other pids taking processor from task 0
  4215        285
    40        202
  3279        154
  3277        124
   753         62
    25         51
    35         35
    38         33
    36         31
    37         31
...

irq taking processor from task 0
    irq      count    min(us)    avg(us)    max(us)
     16         34          3          4         19
    284         20          0          0
```

The **perf** utility and toolset

Perf Command	Description
perf help $cmd	Displays help (also perf-$cmd help)
perf record	Records perf data to a file, minimal output
perf report	Generates human-readable analysis and summary or recorded data
perf list	Lists HW events available on system
perf stat	Displays statistics
perf sched	Scheduler related data
perf top	...
perf diff	...
many more....	development rapid

Using perf for common low latency workload tasks

```
# perf stat ls

 [cut ls output]

 Performance counter stats for 'ls':

         0.892883 task-clock                #    0.723 CPUs utilized
                0 context-switches          #    0.000 M/sec
                0 CPU-migrations            #    0.000 M/sec
              237 page-faults               #    0.265 M/sec
        1,657,323 cycles                    #    1.856 GHz
   <not counted> stalled-cycles-frontend
   <not counted> stalled-cycles-backend
        1,032,854 instructions              #    0.62  insns per cycle
          211,223 branches                  #  236.563 M/sec
           11,535 branch-misses             #    5.46% of all branches        [33.66%]

      0.001234528 seconds time elapsed
```

Using perf for common low latency workload tasks

```
# perf stat -a -A stacsingle

# grep CPU3 report.out

CPU3        600013.767376 task-clock              #    1.000 CPUs utilized
CPU3               3,617 context-switches
CPU3                  74 CPU-migrations
CPU3              13,288 page-faults
CPU3         8,525,426,157 cycles                 #    0.000 GHz                      (50.65%)
CPU3         <not counted> stalled-cycles-frontend
CPU3         <not counted> stalled-cycles-backend
CPU3        10,566,806,531 instructions           #    0.00  insns per cycle          (75.58%)
CPU3         1,868,519,074 branches                                                   (75.46%)
CPU3            13,702,540 branch-misses                                              (73.89%)
```

Using perf for common low latency workload tasks

```
# perf list

List of pre-defined events (to be used in -e):
  ...
  cpu-cycles OR cycles   [Hardware event]
  cache-misses           [Hardware event]
  branch-misses          [Hardware event]
  page-faults OR faults  [Software event]
  context-switches OR cs  [Software event]
  cpu-migrations OR migrations  [Software event]
  dTLB-store-misses      [Hardware cache event]
  power:cpu_idle         [Tracepoint event]
  ...
```

Using perf for common low latency workload tasks

```
# perf sched record udp_rr_test

# perf sched latency
-----------------------------------------------------------------------------------------------------------
 Task                  |   Runtime ms  | Switches | Average delay ms | Maximum delay ms | Maximum delay at     |
-----------------------------------------------------------------------------------------------------------
 :5922:5922            |      0.174 ms |        2 | avg: 1439.576 ms | max: 2879.152 ms | max at: 424504.924203 s
 ksoftirqd/2:13        |   2752.101 ms |       15 | avg:  183.460 ms | max:  359.945 ms | max at: 424237.514072 s
 ksoftirqd/3:17        |   4534.673 ms |       29 | avg:  156.357 ms | max:  362.408 ms | max at: 424312.559072 s
 ksoftirqd/6:29        |   5528.435 ms |       37 | avg:  149.406 ms | max:  395.016 ms | max at: 424408.082072 s
 ksoftirqd/4:21        |   4102.912 ms |       33 | avg:  124.319 ms | max:  391.436 ms | max at: 424478.487139 s
 ksoftirqd/0:4         |  24244.636 ms |      433 | avg:   55.982 ms | max:  391.014 ms | max at: 424196.893071 s
 ksoftirqd/5:25        |    515.151 ms |       11 | avg:   46.819 ms | max:  117.041 ms | max at: 424167.629071 s
 ksoftirqd/1:9         |      8.339 ms |       12 | avg:    0.688 ms | max:    4.004 ms | max at: 424142.016011 s
 netperf:5800          |     82.805 ms |     6357 | avg:    0.154 ms | max:  950.059 ms | max at: 424143.701117 s
 events/2:37           |     10.125 ms |      527 | avg:    0.065 ms | max:   12.024 ms | max at: 424166.710154 s
 events/0:35           |     15.716 ms |      764 | avg:    0.019 ms | max:    6.045 ms | max at: 424860.710050 s
 events/4:39           |     14.066 ms |      373 | avg:    0.014 ms | max:    0.111 ms | max at: 424510.692156 s
 jbd2/dm-0-8:458       |      1.236 ms |       33 | avg:    0.014 ms | max:    0.033 ms | max at: 424209.715275 s
 events/3:38           |      9.832 ms |      379 | avg:    0.013 ms | max:    0.036 ms | max at: 424745.372236 s
 events/6:41           |      9.953 ms |      569 | avg:    0.011 ms | max:    0.049 ms | max at: 424547.640082 s
 udp_rr_script:5799    |      2.135 ms |        2 | avg:    0.011 ms | max:    0.014 ms | max at: 424142.031395 s
 jbd2/dm-2-8:1090      |      0.123 ms |        2 | avg:    0.010 ms | max:    0.010 ms | max at: 424147.696460 s
 perf:5798             |     29.299 ms |        3 | avg:    0.010 ms | max:    0.012 ms | max at: 424142.028029 s
```

Using perf for common low latency workload tasks

 # perf record finder

```
[ perf record: Woken up 1 times to write data ]
[ perf record: Captured and wrote 0.033 MB perf.data (~1451 samples) ]
```

 # perf report > report.out

```
Events: 635  cycles
    8.50%    find  find                [.] 0x149d9
    3.27%    find  [kernel.kallsyms]   [k] _spin_lock
    2.12%    find  [kernel.kallsyms]   [k] __d_lookup
    2.05%    find  [kernel.kallsyms]   [k] filldir
    2.01%    find  [kernel.kallsyms]   [k] kmem_cache_alloc
    1.87%    find  libc-2.12.so        [.] __gconv_transform_utf8_internal
    1.83%    find  libc-2.12.so        [.] __GI_memmove
    1.82%    find  libc-2.12.so        [.] _int_malloc
    1.70%    find  [kernel.kallsyms]   [k] copy_user_generic_string
    1.66%    find  libc-2.12.so        [.] internal_fnwmatch
    1.66%    find  [kernel.kallsyms]   [k] _atomic_dec_and_lock
    1.66%    find  [ext4]              [k] ext4_readdir
    1.62%    find  libc-2.12.so        [.] _int_free
    1.43%    find  [kernel.kallsyms]   [k] kfree
    1.35%    find  libc-2.12.so        [.] __mbsrtowcs_l
    1.27%  finder  [kernel.kallsyms]   [k] page_remove_rmap
```

You have a profile – now what?

```
# perf report
  16.67%     81743      producer  [unknown]             [.] 0xffffffffff60014c
  12.12%     59407      producer  [vdso]                [.] 0x7fff25f648a4
  11.49%     56312      producer  libstacm2.so          [.] dequeueMessageNow
   5.73%     28074      producer  libstacm2.so          [.] dequeueMessageNow@plt
   5.58%     27345      producer  libstacm2.so          [.] ptrDequeue
   5.41%     26515      producer  libstacm2.so          [.] dequeueMessageSpinForever
   4.38%     21487      producer  libstacm2.so          [.] ptrDequeue@plt
   3.05%     14975      producer  libstacm2.so          [.] ptrQueueLength@plt
   2.33%     11101      producer  [kernel.kallsyms]     [k] system_call
   1.51%      7389      producer  libstacm2.so          [.] ptrQueueLength
   1.30%      6352      producer  libstacm2.so          [.] waitOnQueue
   1.28%      6303      producer  libpthread-2.12.so    [.] __recvmsg
   1.17%      5393
[kernel.kallsyms]       [k] schedule
   1.15%      5647      producer  libc-2.12.so          [.] gettimeofday
   1.12%      5472      producer  libstacm2.so          [.] getMicrosNow
   1.04%      4983      producer  [kernel.kallsyms]     [k] copy_user_generic_string
   1.01%      4796      producer  [kernel.kallsyms]     [k] __audit_syscall_exit
   1.01%      4656      producer  libc-2.12.so          [.] __GI_sched_yield
   0.95%      4664      producer  libstacm2.so          [.] dequeueMessageSpinTime
   0.95%      4351      producer  [kernel.kallsyms]     [k] _spin_lock
   0.81%      3950      producer  libstacm2.so          [.] getTimeOfDayNow
```

STAC M2 SINGLE, unoptimized everything

You have a profile – now what?

0.74%	3424	producer	[kernel.kallsyms]	[k] sched_clock_local
0.70%	3297	producer	[kernel.kallsyms]	[k] system_call_after_swapgs
0.69%	3055	producer	[kernel.kallsyms]	[k] update_curr_rt
0.58%	2854	producer	[kernel.kallsyms]	[k] fget_light
0.58%	2842	producer	libstacm2.so	[.] supplier
0.57%	2760	producer	[kernel.kallsyms]	[k] _spin_lock_irqsave
0.56%	2748	producer	libstacm2.so	[.] getMicrosNow@plt
0.55%	2712	producer	libstacm2.so	[.] gettimeofday@plt
0.54%	2484	producer	[kernel.kallsyms]	[k] native_read_tsc
0.53%	2593	producer	[kernel.kallsyms]	[k] __sys_recvmsg
0.53%	2412	producer	[kernel.kallsyms]	[k] _spin_lock_irq
0.48%	2187	producer	[kernel.kallsyms]	[k] thread_return
0.45%	2211	producer	[kernel.kallsyms]	[k] sock_recvmsg
0.45%	2148	producer	[kernel.kallsyms]	[k] sysret_check
0.42%	2059	producer	libpthread-2.12.so	[.] __pthread_enable_asynccancel
0.41%	2004	producer	[kernel.kallsyms]	[k] fput
0.39%	1882	producer	libpthread-2.12.so	[.] __pthread_disable_asynccancel
0.38%	1819	producer	[kernel.kallsyms]	[k] sysret_signal
0.34%	1675	producer	[kernel.kallsyms]	[k] udp_recvmsg
0.33%	1643	producer	[kernel.kallsyms]	[k] sock_update_classid

STAC M2 SINGLE, unoptimized everything

The typical checklist for low latency

- Disable all unneeded services

- Minimize memory latencies (cache misses, non-local memory accesses, page faults, swapping...)
  - numactl -membind
  - mlockall()
  - taskset

- Minimize scheduling latencies (context switches, migration, unnecessary pre-emption...)
  - taskset, cgroups
  - irq binding
  - real-time scheduling attributes

- Offload to hardware
  - network offload

- High resolution APIs, clocksources, synchronization...

- Non-blocking, non-batching operations

REFERENCE

RESOURCES

Firmware Tuning (IBM Platforms)

- my contact : niv at us.ibm.com for docs

Latest rt-tests package

- git clone https://github.com/clrkwllms/rt-tests
- hwlatdetect (python module; kernel driver
- cyclictest

perf utility

- RHEL/SLES/Fedora : “perf” pkg; Ubuntu : “linux-tools-common” pkg
- https://perf.wiki.kernel.org/

ftrace utility

- kernel/Documentation/ftrace.txt (many, many others)

systemtap

- http://sourceware.org/systemtap/wiki
- http://sourceware.org/systemtap/examples/

Performance Goals - They're Different Things! Really!

- **Throughput**
  - Optimize for **best average**
  - Default design criteria for most operating systems
  - "how much can you do at a time?"
- **Low Latency**
  - Optmize for **best minimum**
  - Minimize execution times for certain paths
  - "what's the fastest we can push a packet out?"
- **Determinism**
  - Optimize for **best (lowest) maximum**
  - Fewest/lowest outliers
  - "what's the maximum time it will take?"