MHPC
Master in High Performance Computing

Esssential Perf + valgrind

Stefano Cozzini

SISSA
Scuola Internazionale Superiore di Studi Avanzati

ICTP The Abdus Salam International Centre for Theoretical Physics

Perf

- Linux utility that interfaces its kernel-space layer (perf_events) to the user-space
- It allows to access, read and collect the performance counters during the run time of a process.
- Perf_events interacts with the model-specific registers (MSR) and the performance monitoring unit (PMU) of the CPUs through the msr kernel module.
- Data collected can be post-processed in order to perform a deeper analysis and extract derived metrics.

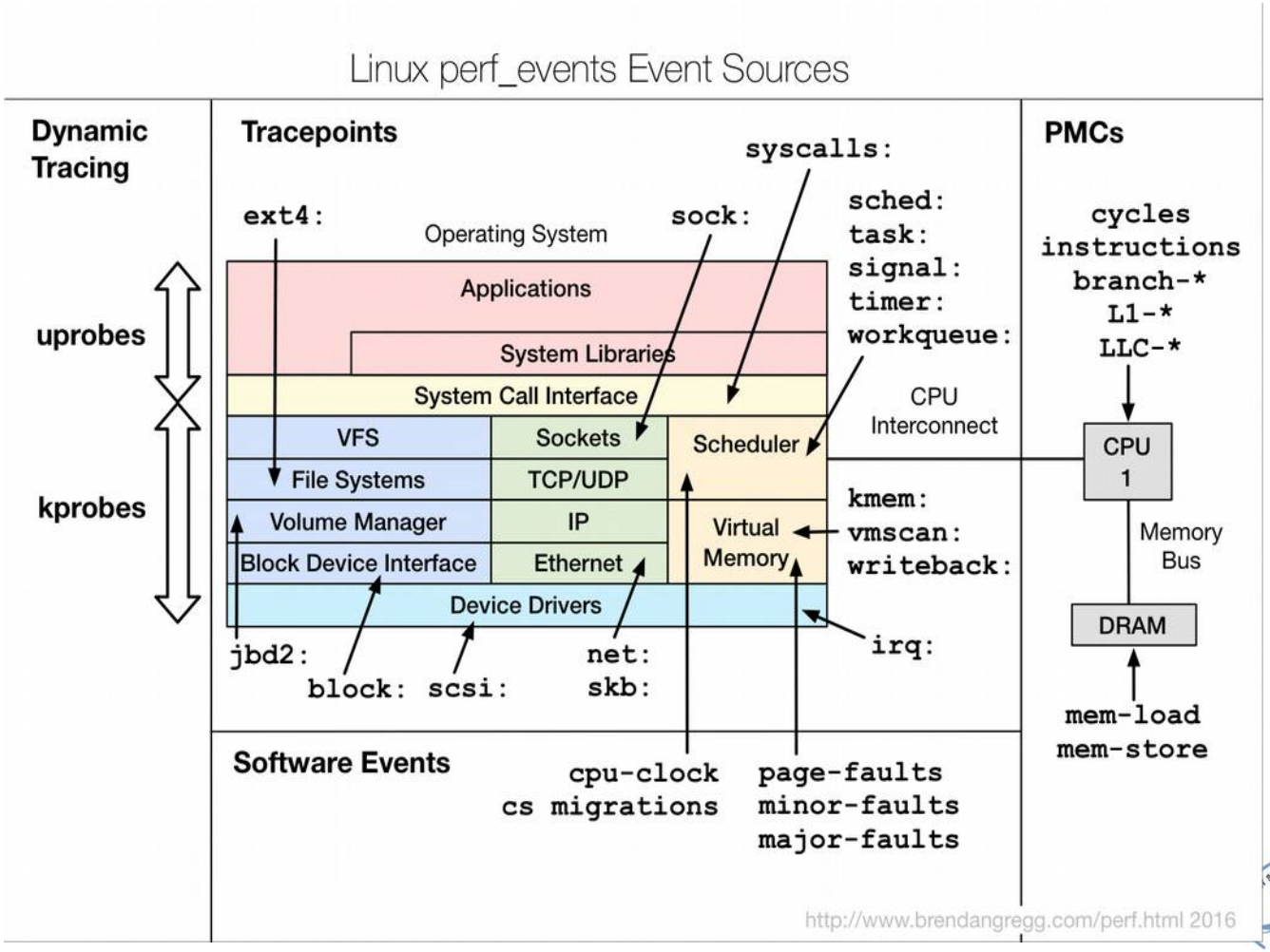
MHPC
Master in High Performance Computing

PMU= performance monitoring unit

- hardware counters,
  - Also called performance monitoring counters (PMCs),
  - erformance instrumentation counters (PICs).
  - These instrument low-level processor activity, for example, CPU cycles, instructions retired, memory stall cycles, level 2 cache misses, etc. Some will be listed as Hardware Cache Events.
  - PMCs are documented in the Intel 64 and IA-32 Architectures Software Developer's Manual Volume 3B: System Programming Guide, Part 2 and the BIOS and Kernel Developer's Guide (BKDG) For AMD Family 10h Processors.
  - There are thousands of different PMCs available.
- only a few or several can be recorded at the same time,
- Just a limited number of registers programmed to begin counting the selected events.

SISSA ICTP

Perf events


Linux perf_events Event Sources
Dynamic Tracing
Tracepoints
PMCs
uprobes
kprobes
Operating System
Applications
System Libraries
System Call Interface
VFS
Sockets
File Systems
TCP/UDP
Volume Manager
IP
Block Device Interface
Ethernet
Device Drivers
Scheduler
Virtual Memory
CPU Interconnect
CPU 1
Memory Bus
DRAM
ext4:
sock:
syscalls:
sched:
task:
signal:
timer:
workqueue:
kmem:
vmscan:
writeback:
irq:
jbd2:
block:
scsi:
net:
skb:
cycles
instructions
branch-*
L1-*
LLC-*
mem-load
mem-store
Software Events
cpu-clock
cs migrations
page-faults
minor-faults
major-faults
http://www.brendangregg.com/perf.html 2016

Usage examples

```
>perf stat sleep 1
>perf stat -e branch-instructions,branch-misses /bin/ls
>perf stat -o ./perf.log -x, -e r03c,r19c,r2c2,r10e,r30d /bin/ls
>perf stat -a -x, -o ./perf.log \
-e cpu/config=0x003C,name=CPU_CLOCK_UNHALTED_THREAD_P/ \
-e cpu/config=0x019C,name=IDQ_UOPS_NOT_DELIVERED_CORE/ \
-e cpu/config=0x02C2,name=UOPS_RETIRED_RETIRE_SLOTS/ \
-e cpu/config=0x010E,name=UOPS_ISSUED_ANY/ \
-e cpu/config=0x030D,name=INT_MISC_RECOVERY_CYCLES/ \
mpirun --np 24 xhpl
```

MHPC
Master in High Performance Computing

Perf Resources

- http://www.brendangregg.com/perf.html
- Moreno 's thesis: http://preprints.sissa.it/xmlui/handle/1963/35155

SISSA
ICTP

Profiling: tools - Valgrind

Memcheck helps you in highlighting some common memory errors:

- Unvalid memory access: overrunning/underrunning of heap blocks or top of stack, addressing freed blocks, …
- Use of variables with undefined values
- Incorrect freeing of heap memory
- Errors in moving memory (unwanted src/dst overlaps, …)
- Memory leaks

Cachegrind simulate the interaction of the code with a cache (you can model it).
It can report how many hits (L1, L2 and L3, I- and D-) and how many misses.
It can analyze CPU's branch prediction.

Ir, I1mr, LLmr, Dr, D1mr, DLmr, Bc, Bcm, Bi, Bim, …

Callgrind is a CPU profiler.
It collects the number of instructions executed, links them to source lines, records the caller/callee relationship between functions, and the numbers of such calls.
Optionally, it may collect data on cache simulation and/or branch as Cachegrind does

Profiling: tools - Valgrind

An instrumentation framework for building dynamic analysis tools.
Valgrind basically runs your code in a virtual "sandobx" where a synthetic CPU (the same you have) is simulated.
Actually it converts x86 instructions in cleaner RISC-like Ucode and executes it appropriately instrumented.

There are various Valgrind based tools for debugging and profiling purposes.

- **Memcheck** is a memory error detector → correctness
- **Cachegrind** is a cache and branch-prediction profiler → velocity
- **Callgrind** is a call-graph generating cache profiler. It has some overlap with Cachegrind
- **Helgrind** is a thread error detector → correctness
- **DRD** is also a thread error detector. Different analysis technique than Helgrind
- **Massif** is a heap profiler → memory efficiency using less memory
- **DHAT** is a different kind of heap profiler → memory layout inefficiencies
- **SGcheck** (experimental tool) that can detect overruns of stack and global arrays

KCacheGrind is a very useful GUI

MHPC
Master in High Performance Computing

Valgrind examples

- valgrind –tool=cachegrind program arguments

SISSA
ICTP