CG² TALK CODE

CHAOSGROUP

Better Profiling

An Intro to Event-based Sampled Profiling

Overview of the main approaches

- Instrumented profiling
- Non-instrumented profiling
  - Timer-based
  - PMU-based
  - Emu-based

Overview of the main approaches, cont'd

Instrumented profiling

Can produce precise call graphs and invocation counts (traditional use).
Though technically there's no limit to its applications, it's still intrusive!

Example: Intel's Pin
https://software.intel.com/en-us/articles/pin-a-dynamic-binary-instrumentation-tool

Overview of the main approaches, cont'd

Non-instrumented profiling

- Timer-based

  Can detect hotspots*, unreliable at call graphs

- PMU-based

  Can detect hotspots and their causes, potentially call graphs

- Emu-based

  Can produce precise call graphs and hotspots, for an imaginary CPU

* Very sensitive to timer/CPU clock skews.

PMU (Performance Monitoring Unit) Enables Event-based Sampling

- Performance counters of ‘events’ trigger a PMI when predefined counts are reached -- that constitutes the fundamental ‘profiling sample’.
- Found in x86 since the Pentium days, but gained traction with the Prescott (P4) and Yonah (Core Solo/Duo), which brought..
- Standardized query mechanism (via CPUID) for “**Architectural**” performance events, in addition to traditional..
- Model-specific “**Non-architectural**” events
- Can keep track of more conditions with each new gen, occurring at various spots in the pipeline (and beyond) at every core cycle.

Architectural performance events

“Performance monitoring events are architectural when they behave consistently across [x86] *microarchitectures.”*

- Query-able via CPUID functionality tree
- Versioning -- currently at 3; introduces SMT cores, also..
- Compliant cores can count (some subset of) these events:
  - UnHalted Core Cycles
  - Instruction Retired
  - UnHalted Reference Cycles
  - LLC References
  - LLC Misses
  - Branch Instruction Retired
  - Branch Misses Retired

So how ‘non-intrusive’ is PMU-based profiling?

Very much so, but let’s not fool ourselves

https://software.intel.com/en-us/articles/performance-impact-when-sampling-certain-llc-events-on-snb-ep-with-vtune

Today's focus will be on these tools..

- Linux Perf
  - A collection of APIs and tools, kernel- and user-side
  - Made by Linux kernel maintainers - quick dev cycle
  - Supports x86, x86_64, PPC64, ARM and MIPS
  - Linux-only
- Intel VTune
  - Linux and Windows
  - Targets Intel CPUs only
  - Does comprehensive derivative analysis

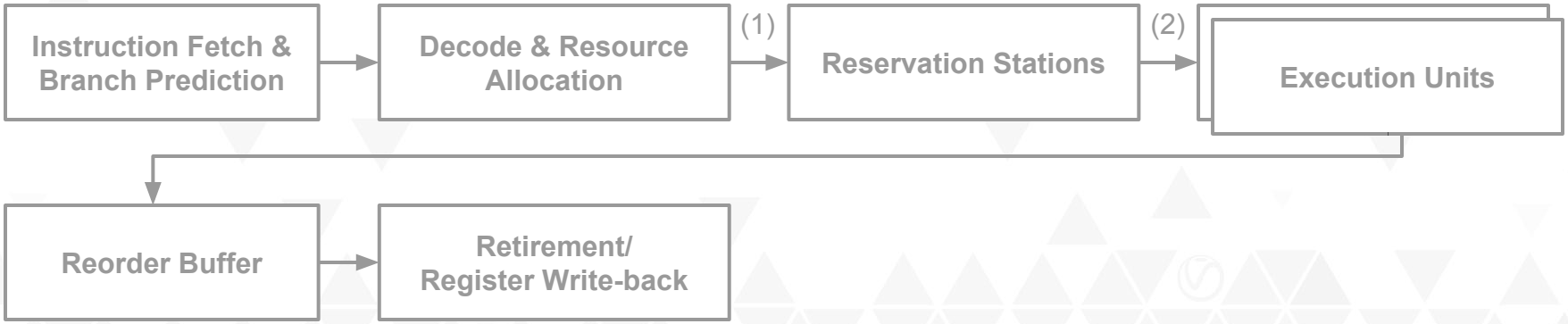
Today's focus will be on this CPU..

The "contemporary x86_64 CPU"

- Deep pipelines
- OOO (out-of-order) parallel dispatch & execution
- reconstruction of the original order at exit
- Multi-level cache hierarchy, shared & inclusive LLC, TLBs

A bird's view of an OOO x86_64 CPU

The Issue-Dispatch-Execute-Retire Pipeline Model



(1) Issue

(2) Dispatch

Perf userland tools at a glance

- perf list
- perf stat
- perf record
- perf annotate
- perf script
- perf report
- perf top

Hands-on with Perf and VTune

..

refs

- David Levinthal, *Performance Analysis Guide for Intel Core i7 Processor and Intel Xeon 5500 processors*, 2008-2009 Intel Corporation
- *Intel 64 and IA-32 Architectures Software Developer's Manual, Volume 3: System Programming, Chapter 18*

Sampling ‘raw’ events example (Xeon E5-2687W):

event	umask	mnemonic
0x0E	0x01	UOPS_ISSUED.ANY

“Increments each cycle the number of Uops issued by the RAT to RS.”

0xC2	0x01	UOPS_RETIRED.ALL

“Counts the number of micro-ops retired.”

```
$ perf stat -e task-clock:u,cycles:u,instructions:u,branches:u,branch-misses:u,r10e:u,r1c2:u -- ./test_bsearch space_size 16777216 alt 0
$ perf stat -e task-clock:u,cycles:u,instructions:u,branches:u,branch-misses:u,r10e:u,r1c2:u -- ./test_bsearch space_size 16777216 alt 3
```

Thank you for your attention. Enjoy your ticks!

martin.krastev@chaosgroup.com