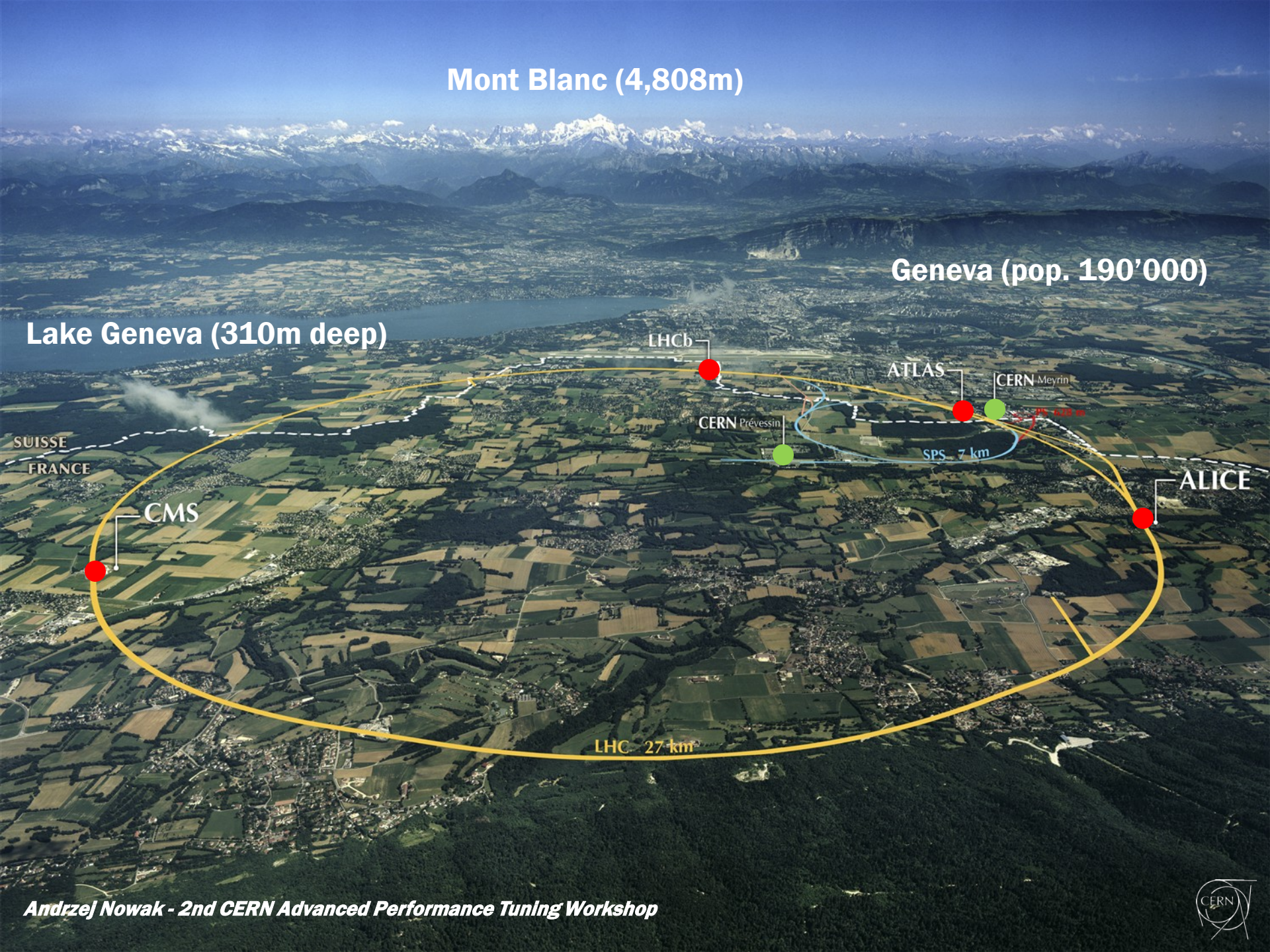
2nd CERN Advanced Performance Tuning Workshop - introduction

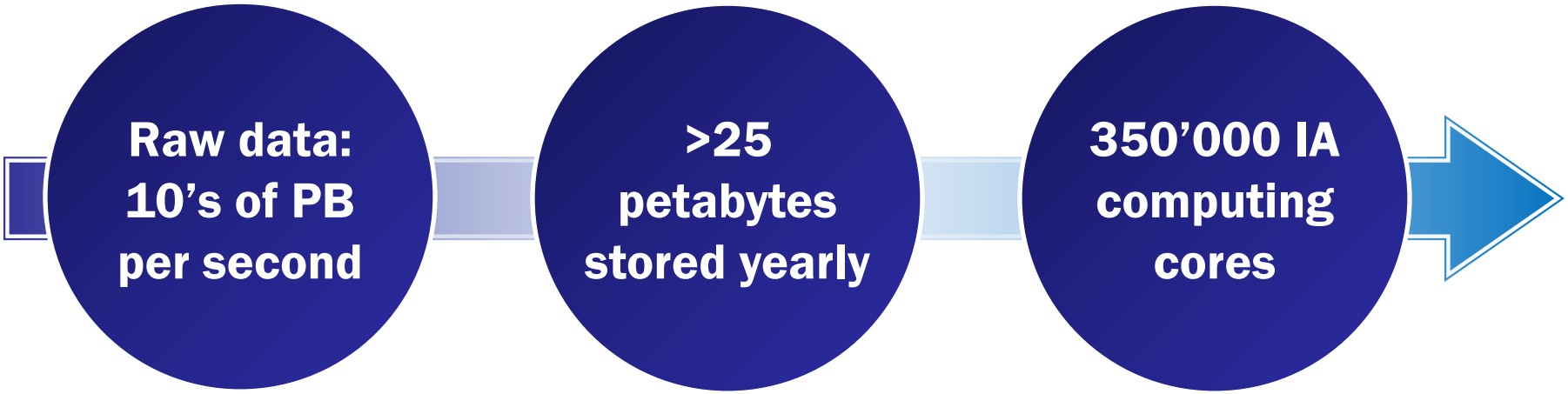
Andrzej Nowak (CERN openlab)

November 21st 2013


Mont Blanc (4,808m)
Geneva (pop. 190’000)
Lake Geneva (310m deep)
LHCb
ATLAS
CERN Meyrin
CERN Prévessin
SPS 7 km
SUISSE
FRANCE
CMS
ALICE
LHC 27 km

Worldwide LHC Computing

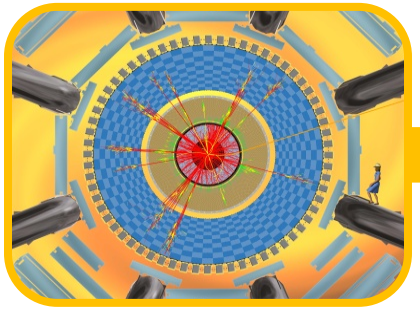
Intense data pressure creates strong demand for computing



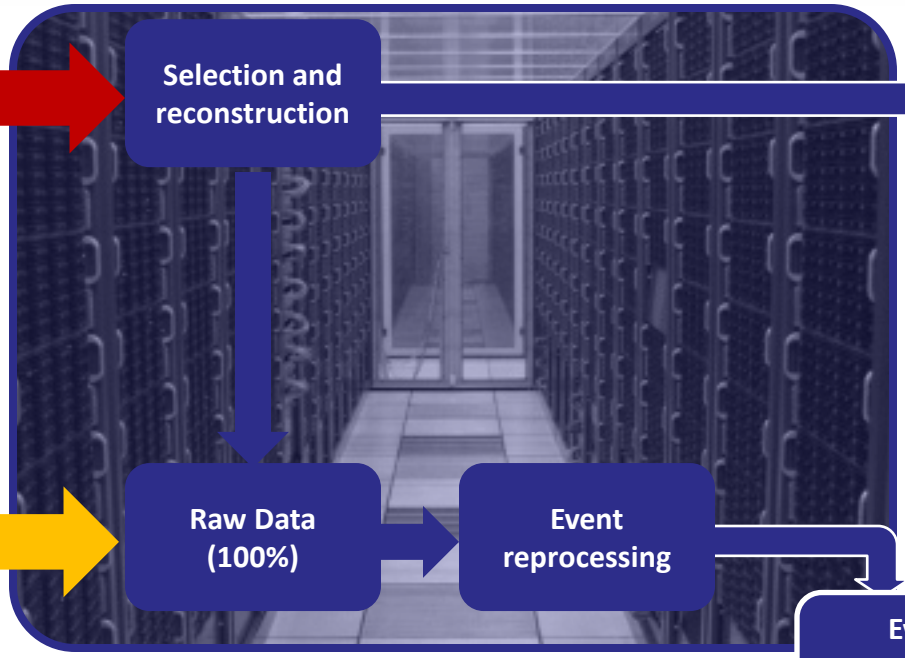
A rigorous selection process enables us to find that one interesting event in 10 trillion (10^{13})

Data flow from the LHC detectors

Online triggering and filtering in detectors



Event simulation



Analysis

uArch level

Pattern	Load, load, do something, multiply, add, store
FP	Scalar double, 10-15%
CPI	>1.0
Load/store	60% of instructions
Inst/jump	<10
Inst/call	<30-60
Memory	Largely read-only

- **Conclusions:**
  - Unfavorable for the x86 microarchitecture (even worse for others)
  - For the most part, code not fit for accelerators at all in its current shape

Workload classes

	CPU time on the Grid	CPU usage	Disk IO	Net IO (bw & lat)
Simulation	High	High	Minimal	Minimal
Reconstruction	Medium	High	Minimal	Minimal
Digitization	Low	High	Varying	Low
Generation	Low	Med-High	Low-Med	Low
Client/IT	None	Low	Low	Low
Client/Analysis	Varying	Varying	Varying	Varying

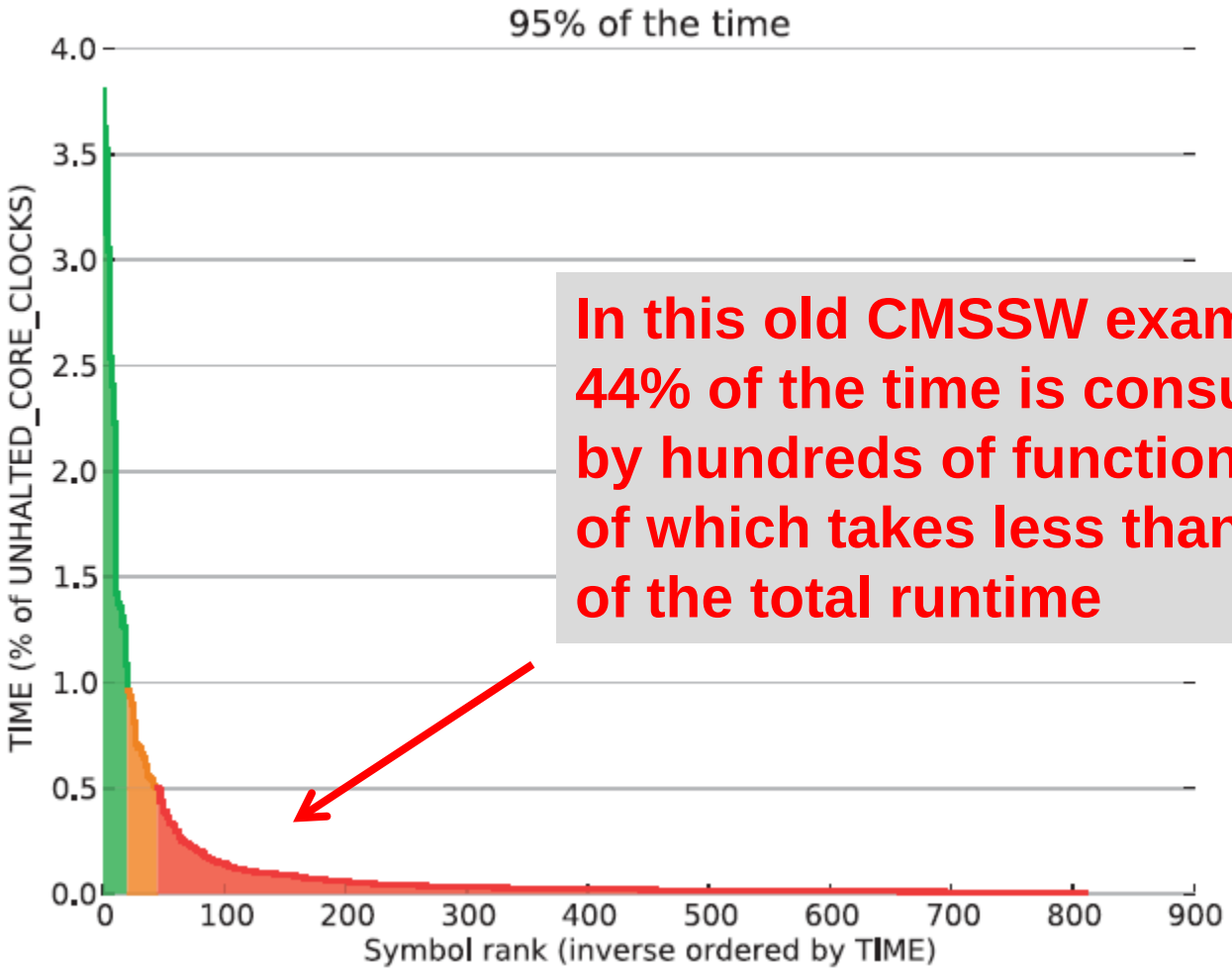
Performance tuning processes in 2010

- **Surveyed 6 major offline collaborations (20 MLOC)**
  - ROOT, Geant4
  - ALICE, ATLAS, CMS, LHCb
- **Software performance is not a priority, but the quality of science is**
  - Memory layout and usage patterns
  - Fragmentation, leaks, allocation leads to pressure and non-locality
  - Microartchitectural issues secondary and not well explored
- **Opportunistic optimization prevailed**
  - Regression based - maintain constant overall performance rather than improve
  - All parties run nightly regression checks
  - 2 out of 6 had dedicated „performance people”
  - 3 out of 6 depended exclusively on best effort

Extracting benchmarks

- **Extracting a meaningful benchmark from several million lines of code is hard**
  - There are loopy parts, but many of them
  - High fragmentation and large code base
  - Too many code paths – the outer layer/loop might be the same in many cases but the contents can vary wildly per „physics situation” and „per experiment”
  - Making it self-contained and independent
- **Two realistic options**
  - Extract „snippets” – a single method + friends
  - Copy full frameworks

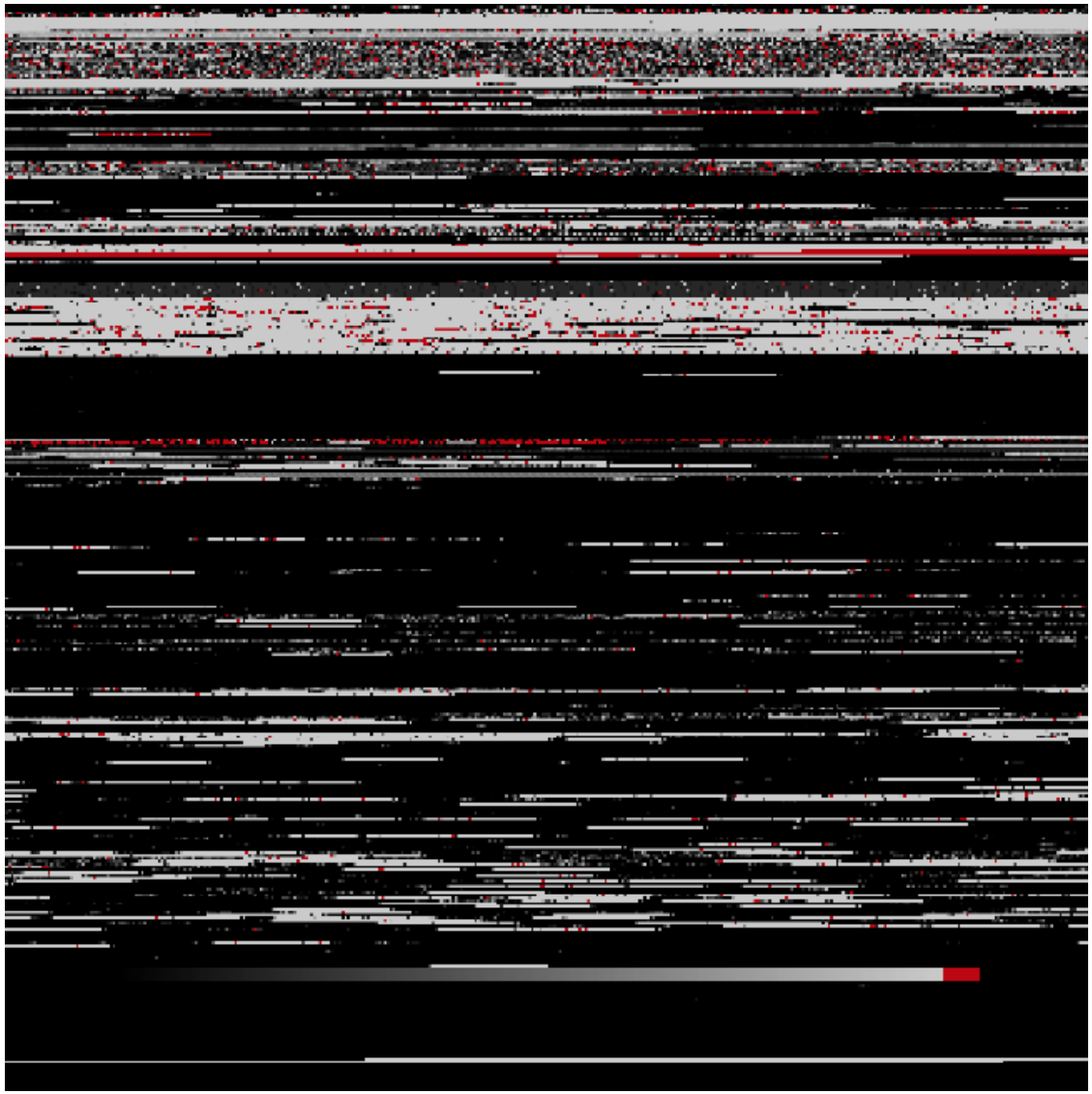
Fragmentation



In this old CMSSW example, 44% of the time is consumed by hundreds of functions, each of which takes less than 0.5% of the total runtime

From G. Eulisse

Fragmentation



From G. Eulisse

Compilers

- **The best tuning aid we could possibly imagine**
  - Very conservative options: -O2, -fPIC
  - Value safety very important
- **GCC base (recent GCC) + old system GLIBC**
- **ICC and LLVM slowly picked up**
  - ICC for performance
    - O3 very rarely used, -fast: never
  - LLVM for analysis and introspection
- **PGO produces penalties (code paths hard or impossible to predict)**

Tools – functional requirements

Memory related statistics	Track IO bottlenecks easily
	Layout on heap, page sharing, usage histograms
	Allocations and deallocations (usage patterns, allocation patterns, pressure, layout)
	Categorize by calling stack
	Tracking down leaks
Event based sampling	Per-function
	Per-module
	With stack traces
Non-technical guidelines:	Understandable by non-experts
	OSS, work in RHEL, without ROOT access
	Stable and reliable on large code
Call graph building	

Performance tools used

> PMU based

- earlier: perfmon2
- perf
  - Badly designed, painful to use
  - De facto standard
  - Gooda from Google
- Intel tools (Amplifier – worked on the alpha, SEP, PTU)
- Some PAPI adoption

> Instrumentation

- IgProf, Valgrind + friends (very popular)
- PIN (slow)
- Intel Amplifier
- Intel Inspector (low success rate)

> Own tools

- Not many tools work with large applications
- Scripts, analyzers parsing raw data

PMU techniques employed

- **Event Counting**
  - Black-box studies and regression
  - Good for fragmentation
- **EBS IP Sampling**
  - Wide range of tuning activities
  - Low precision on our code
  - Bad in a fragmented scenario
- **Time based sampling and time based displays of counts**
  - Phase monitoring
  - Provides added value for discovery
- **Experience: high level brings most value since localized optimization is hard**

Our issues with the PMU in a nutshell

“I have 100’000 cache misses more because of this choice of data structure – so what?”

(actual quote from a senior developer)

CERN/High Energy Physics Needs

- **See next talk**
- **Ultimate goal: a simplified performance optimization process**
  - It can only be achieved by striking a good balance between relieving the users of some of the burden and educating them about the microarchitecture at the same time
- **Access to advanced information and data**
  - Much of this is inaccessible today but the hardware is there
- **Easier access to information**
  - Visual reports; high level, composed reports based on advanced data
- **Easier access to the right optimization directions**
  - Extra data allows to give extra advice
- **More intelligent tuning enabled by higher-level conclusions**

Workshop structure

- **Lectures and interactive discussions with optional hands-on**
- **Topics**
  - Monitoring and tuning facilities (here: x86 and ARM)
  - Methodologies
  - Tools – open source and proprietary
  - Workloads: CERN needs, large workload specifics

Speakers

- **ARM**
  - Al Grant
  - Michael Williams
- **Calxeda**
  - Robert Richter (also an AMD expert)
- **CERN**
  - Vincenzo Innocente
- **Google**
  - Maria Dimakopoulou
  - Stephane Eranian
  - David Levinthal
- **Intel**
  - Stanislav Bratanov
  - Michael Chynoweth
  - Ahmad Yasin
- **Versailles Exascale Lab**
  - Andres S. Charif-Rubial

Thank you

Other questions? Andrzej.Nowak@cern.ch