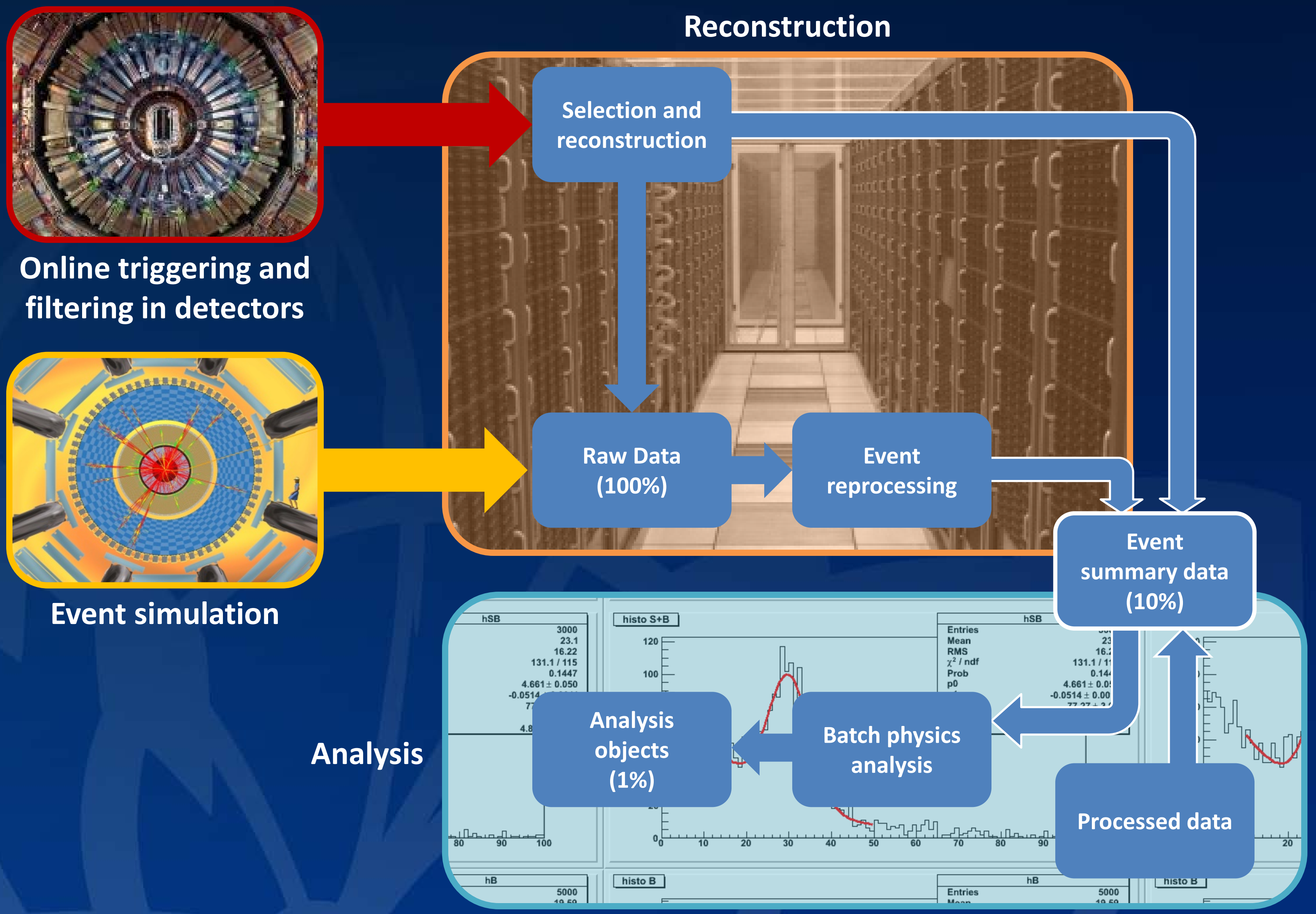
Performance monitoring at CERN openlab

July 20th 2012

Andrzej Nowak, CERN openlab

Data flow


Reconstruction
Online triggering and filtering in detectors
Event simulation
Selection and reconstruction
Raw Data (100%)
Event reprocessing
Event summary data (10%)
Analysis
Analysis objects (1%)
Batch physics analysis
Processed data

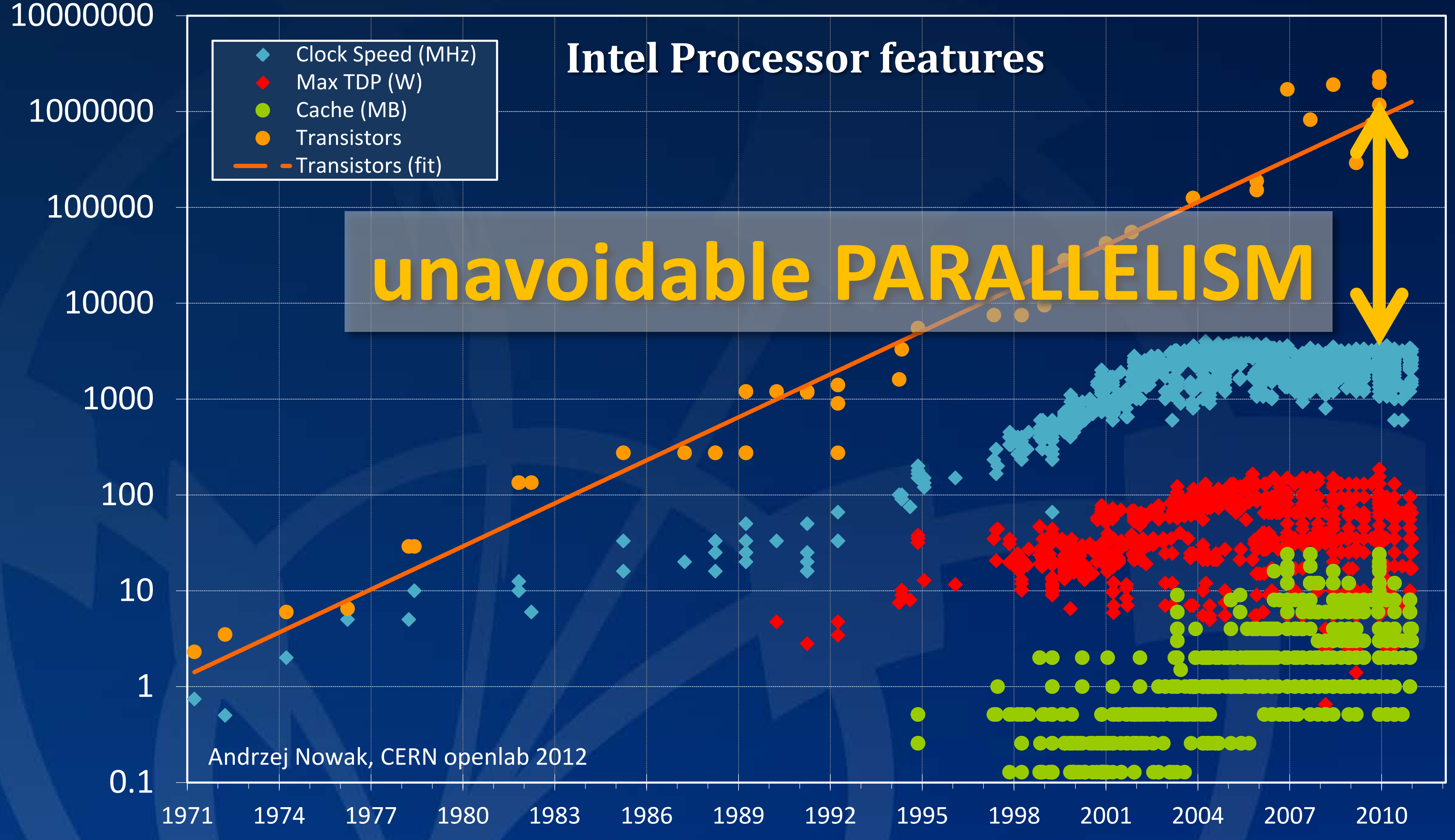
Characteristics of CERN code (1)

- **2 major toolkits and 4 major frameworks built on top – one per LHC experiment**
- **Large C++ frameworks with millions of lines of code**
  - Thousands of shared libraries in a distribution, gigabytes of binaries
  - Low number of key players but high number of brief contributors
- **Large regions of memory read only or accessed infrequently**
- **Characteristics:**
  - Significant portion of double precision floating point (10%+)
  - Loads/stores up to 60% of instructions
  - Unfavorable for the x86 microarchitecture (even worse for others)
    - Low number of instructions between jumps (<10)
    - Low number of instructions between calls (several dozen)
- **For the most part, code not fit for accelerators at all in its current shape**

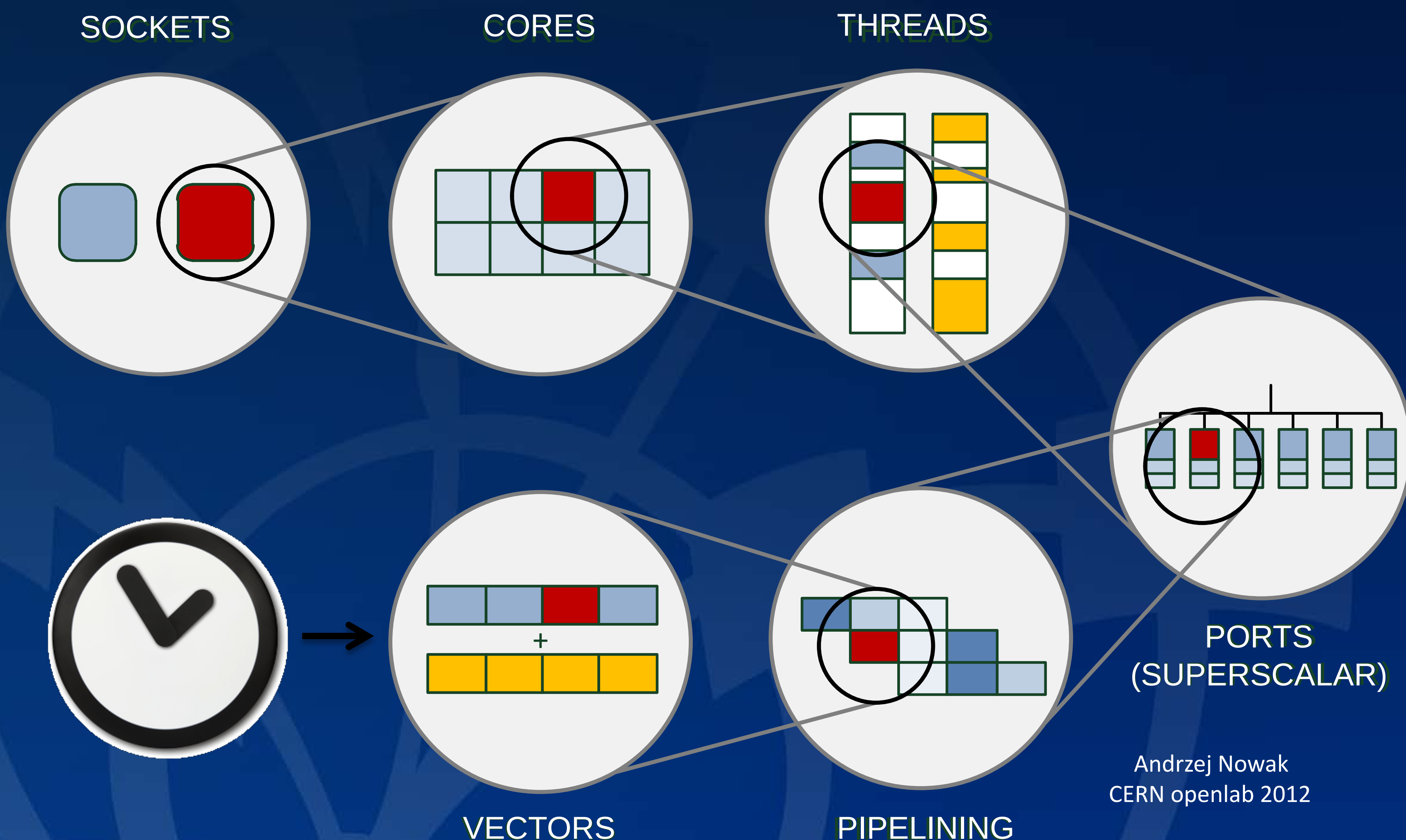
Characteristics of CERN code (2)

- **Memory footprint: 2-4GB per process**
  - Most of that read only
  - Very sparse writes
  - KSM/forking claimed to save 50% of memory (even though these are crude schemes)
  - More advanced schemes – thread-private variables – save even more, >95% (per thread)
- **Cache footprint**
  - key code fits in L2
  - Key code and data fit in L3
  - Heavy C++ virtualization and other C++ mechanism footprint
  - War on TLB misses successfully waged in 2008, but still lots of references
- **CPU intensive workloads have very sparse IO**
  - 1 average disk per dozen processes is enough + 1 gbit ethernet
- **Again: For the most part, code not fit for accelerators at all in its current shape**

Hardware landscape



Inside a modern PC



Omnipresent parallelism - where are we now?

	SIMD	ILP	HW THREADS	CORES	SOCKETS
MAX	4	4	1.35	8	4
OPT	2.5	1.43	1.25	8	2
US	1	0.80	1	6	2

	SIMD	ILP	HW THREADS	CORES	SOCKETS
MAX	4	16	21.6	172.8	691.2
OPT	2.5	3.57	4.46	35.71	71.43
US	1	0.80	0.80	4.80	9.60

Legacy applications use a low single digit percentage of raw machine power available today

_%

Write your percentage here

Techniques

- Event Counting
  – Black-box studies and regression
- EBS IP Sampling
  – Wide range of tuning activities
  – Low precision on our code
- Time based sampling of counts
  – Phase monitoring
- Instrumentation

Tools for performance monitoring

- **PMU based**
  - perfmon2
  - perf
    - Badly designed, painful to use
    - De facto standard
    - Gooda coming up from Google
  - Intel tools (Amplifier, SEP, PTU)
- **Instrumentation**
  - PIN (slow)
  - Intel Amplifier
  - Intel Inspector (low success rate)
- **Own tools**
  - Scripts, analyzers parsing raw data

Intel Software tool usage at CERN

- **Pool of licenses within the openlab agreement**
  - Parallel Studio for Linux and Windows
  - C++ and Fortran compilers for Mac
  - Cluster Toolkit
- **Alpha and Beta testing**
- **Non-standard software**
- **Newly purchased licenses for CERN-wide usage**
  - Suggestion, expertise and setup came out of openlab
  - Prompted by growing demand
  - Linux, Windows, Mac covered

Internal activities focused on performance monitoring

- **Building home-grown tools for analysis and batch reporting**
  - Two separate non-openlab collaborations on PTU front-ends (CMS experiment)
  - Numerous other smaller performance monitoring tools
  - Recent efforts focused on most recent CPU features (some of the original concepts and support came from Intel, HP and Google)
  - Looking forward to performance tuning API/SDK toolkits
  - Planning to employ performance monitoring in parallel to OSS Linux solutions

Case 1: test40

- **Simple electromagnetic test in the Geant4 framework – an electron traveling through a detector**
- **Similar workload to a real physics app but with a tiny footprint**
- **Control flow driven to a large extent**
- **Collection:**
  1. Sampled with PDIR on Sandy Bridge
  2. Instrumented with PIN
- **Expecting:**
  – The sampling profile to match the instrumented profile as closely as possible

Case 1: EBS view (sampling)

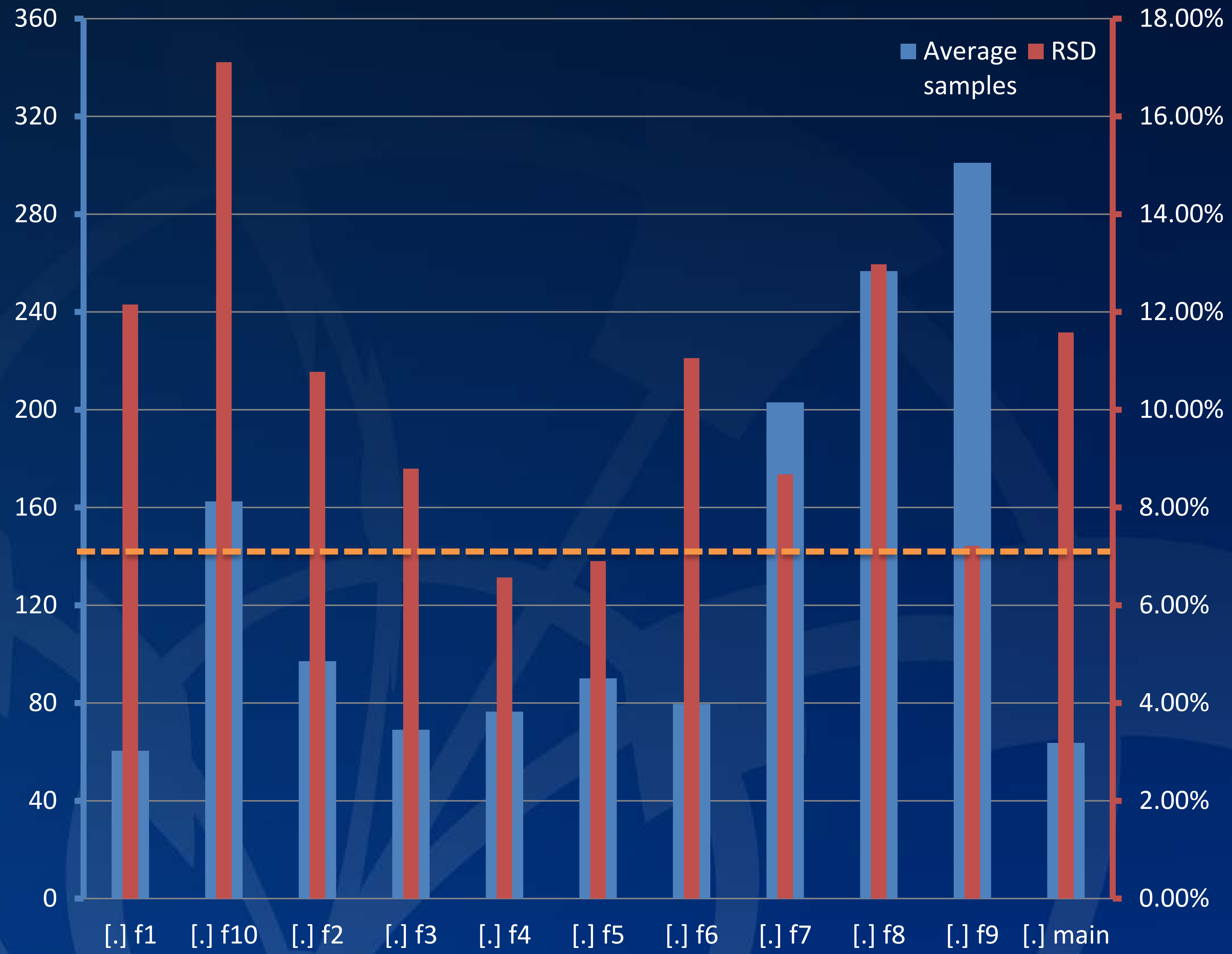
PDIR measurement	Samples	Percentage	RSD
[.] __ieee754_log	25453	21.09%	0.51%
[.] RanecuEngine::flat()	9014	7.47%	0.88%
[.] G4SteppingManager::DefinePhysicalStepLength()	6254	5.18%	0.78%
[.] G4Tubs::Inside(Hep3Vector const&) const	6095	5.05%	1.72%
[.] __ieee754_exp	4783	3.96%	1.61%
[.] G4SteppingManager::InvokePSDIP(unsigned long)	4599	3.81%	1.18%
[.] G4SteppingManager::Stepping()	4191	3.47%	1.22%
[.] _int_malloc	3579	2.97%	1.26%
[.] _int_free	3547	2.94%	1.13%
[.] G4SteppingManager::InvokeAlongStepDoItProcs()	3196	2.65%	1.54%
[.] __log	3235	2.68%	1.58%
[.] G4Track::GetVelocity() const	2673	2.21%	1.72%

Case 1: Reference (real) counts

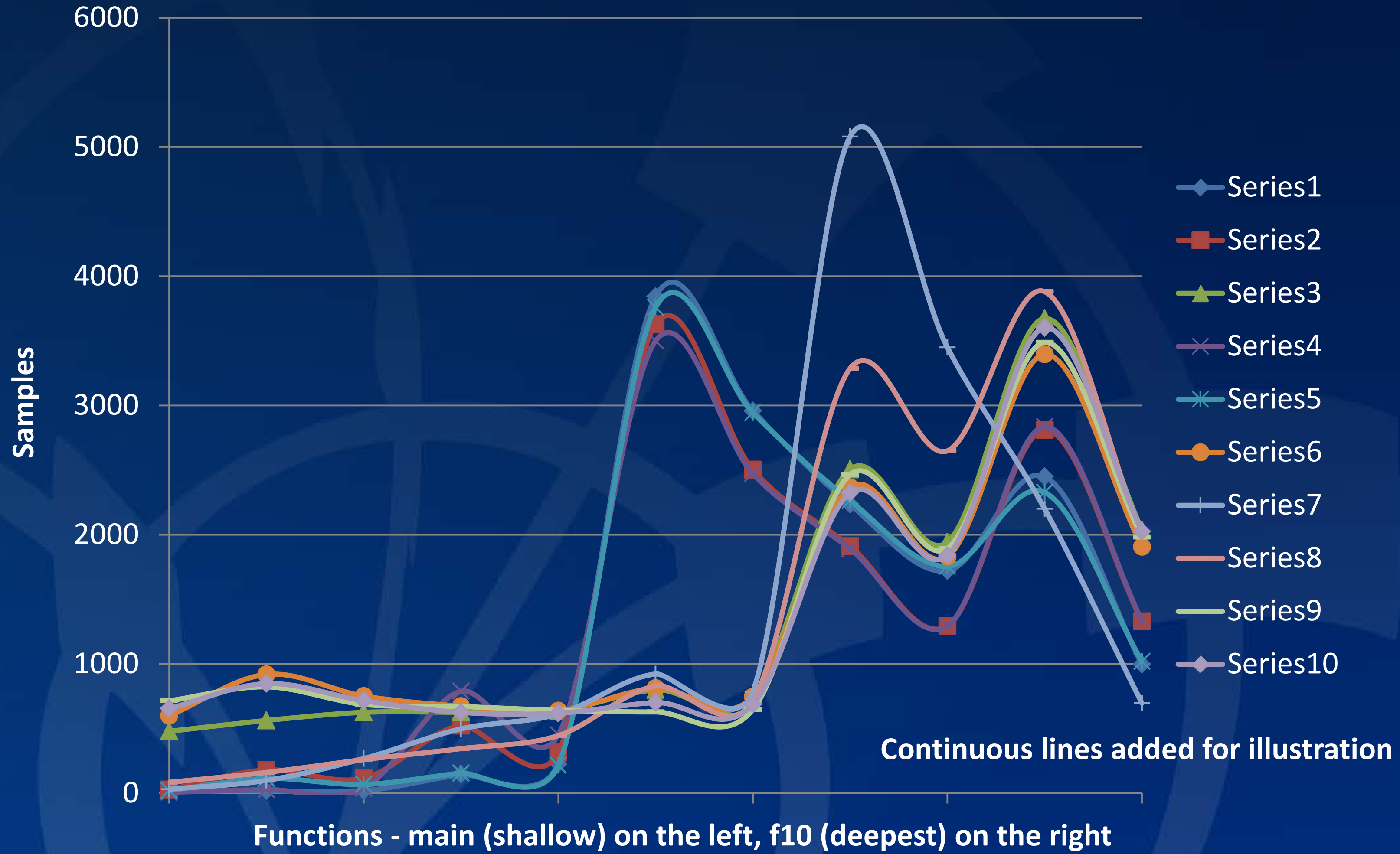
Instrumented reference measurement	Samples	Percentage
G4Navigator::LocateGlobalPointAndSetup(…)	39928	17.43%
__ieee754_log	25736	11.23%
RanecuEngine::flat()	8593	3.75%
G4SteppingManager::DefinePhysicalStepLength()	6229	2.72%
G4Tubs::Inside(Hep3Vector const&) const	5997	2.62%
__ieee754_exp	4874	2.13%
G4ClassicalRK4::DumbStepper(…)	4796	2.09%
G4SteppingManager::InvokePSDIP(unsigned long)	4565	1.99%
G4ReplicaNavigation::ComputeStep(…)	4366	1.91%
G4Transportation::AlongStepGetPhysicalInteractionLength(…)	4259	1.86%
G4SteppingManager::Stepping()	4121	1.80%
G4Navigator::ComputeStep(…)	3718	1.62%
_int_malloc	3576	1.56%
_int_free	3518	1.54%

Case 2: DTB

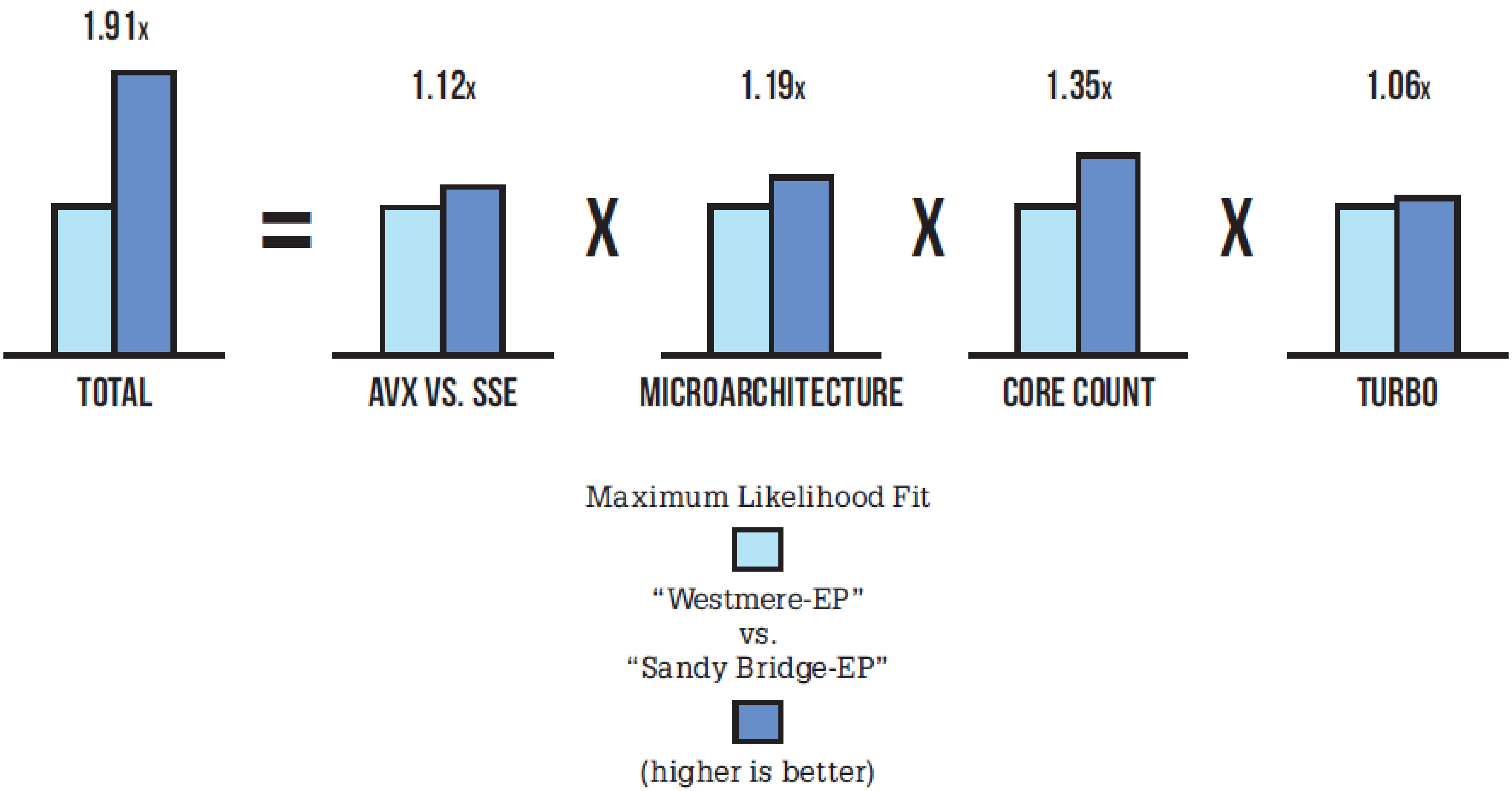
- **Microbenchmark with deep call stack emulation**
  - Representative of OO apps, especially C++
- **F1 calls F2, F2 calls F3, etc until F10**
- **Each function multiplies a var by a constant**
- **Expecting**
  1. Uniform # instructions per function within one experiment
  2. Uniform # instructions per function across all 10 experiments run


360
320
280
240
200
160
120
80
40
0
18.00%
16.00%
14.00%
12.00%
10.00%
8.00%
6.00%
4.00%
2.00%
0.00%
Average samples
RSD
[.] f1
[.] f10
[.] f2
[.] f3
[.] f4
[.] f5
[.] f6
[.] f7
[.] f8
[.] f9
[.] main

Deep call stack profiling (PDIR)



ROOT minimization
(there are some success stories)



Work with the community

- **Regular consultancy with physicists on a range of applications**
  - Assistance with porting, vectorization etc
  - Pathfinding
- **Work with the IT department on platform tuning and debugging**
- **Entering the online domain, joint EU projects**

Teaching efforts

- 2 dedicated workshops on performance monitoring per year (in addition to 2 other workshops on multi-threading)
- Advanced performance tuning sessions w/ Intel experts – 1-2 per year
- New activity: floating point workshops
  - Very successful, lots of interest
- International computing schools, conferences

Our MIC experience

- One of the first Intel customers to be engaged – started with ISA reviews in 2008
- In-depth feedback on the OS, drivers and Xeon/KNF/KNC toolchains
- Ported and optimized 3 large representative benchmarks
  - Ongoing activities
- Looking forward to dissemination of KNC results

THANK YOU

Q & A

BACKUP

About openlab

- CERN openlab is a framework for evaluating and integrating cutting-edge IT technologies or services in partnership with industry: http://cern.ch/openlab
- We are the Platform Competence Center (PCC) of the CERN openlab, working closely with Intel since 11 years ago and addressing:
  - many-core scalability
  - performance tuning and optimization
  - benchmarking and thermal optimization
  - teaching