Summary Track 5

Software Engineering, Parallelism & Multi-Core

Solveig Albrand
Francesco Giacomini
Benedikt Hegner
Liz Sexton
(Simon Patton,
Jim Kowalkowski)

Some Statistics

- 58 submissions to the track – less than half (26) could be ORALS.
  - Difficult decisions!
- Came from “big” and “small” experiments.
- Significant number were “common” to several experiments
- 3 were remote
  - Thanks to the local organization for allowing them
- 1 was withdrawn
  - Thanks to the USA government
- Thanks to all participants for very interesting talks – and excellent time-keeping.

Main Themes

- Beyond x86 servers
- Vectorization
- Concurrency
- C++11
- Software Engineering

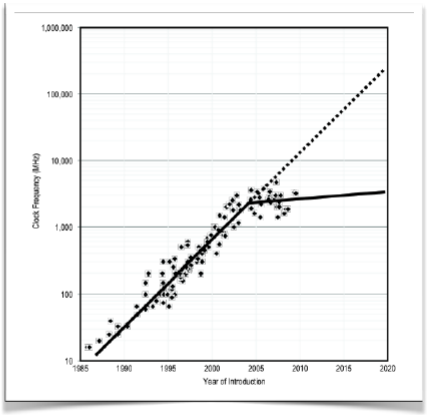
Beyond x86 servers

- To improve performance per watt
- Alternative platforms & co-processors
  - ARM
  - Xeon Phi
  - GPU

New Architectures

- Even multi-core, implemented with large "aggressive" cores is just a stop-gap. The power limitations remain. The focus is shifting to performance/watt, not just performance/price.

Explorations of the viability of ARM and Xeon Phi for physics processing
Peter Elmer et al.



From: "The Future of Computing Performance: Game Over or Next Level?"

In practice

The computer-farm of the future

Not really...

- Different approaches adopted
  - Porting existing code
  - Re-implementing algorithms

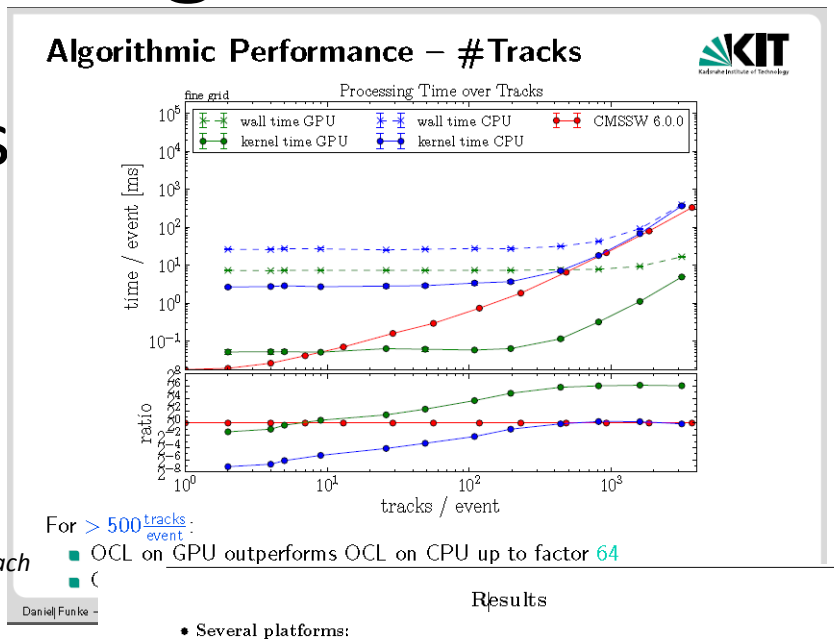
LHCb software stack on ARM,
Niko Neufeld

Porting Existing Code ?

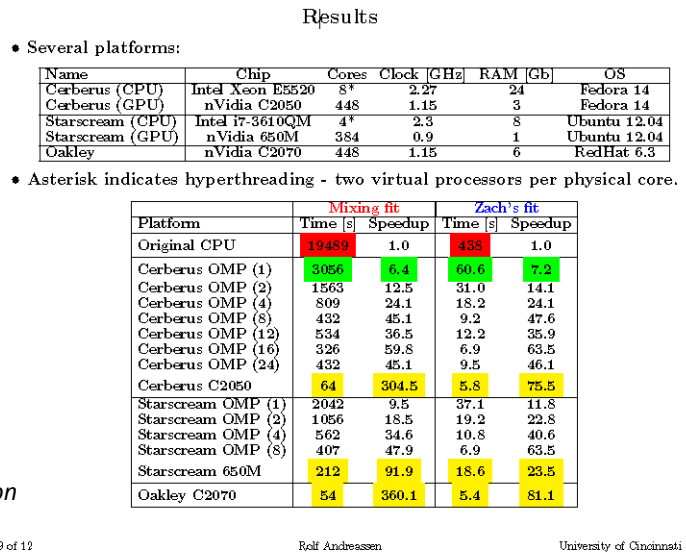
- Examples from 2 LHC experiments.
  - LHCb stack
  - CMS SW
    - Almost everything could be ported to ARM
    - Only a subset on Xeon-Phi (due to compilation issues)
- Importance of OPEN SOURCE
  - Pre-condition for any experimental work on new platforms.

Or re-implement algorithms?

- Many examples for GPUs from every processing step!
  - Track Fitting
  - GooFit
  - GEANT
  - …



Parallel Track Reconstruction in CMS Using the Cellular Automaton Approach Daniel Funke et al.



This approach gets better speed-up but is much more work.
(must revalidate physics performance)

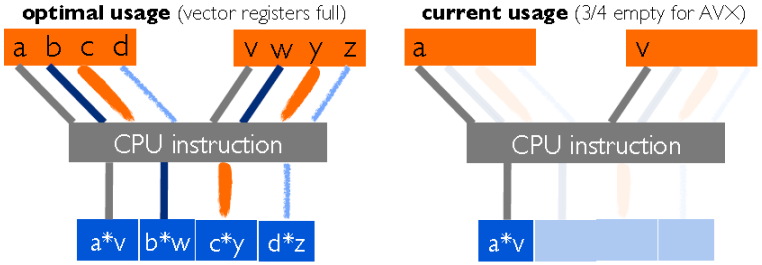
GooFit: Massively parallel function evaluation, Rolf Andreassen

VECTORIZATION

(hard, but not always impossible)

Does the Intel Xeon Phi processor fit HEP workloads? , Andrzej Nowak

- A necessary (but not sufficient) condition for performance on modern architectures.

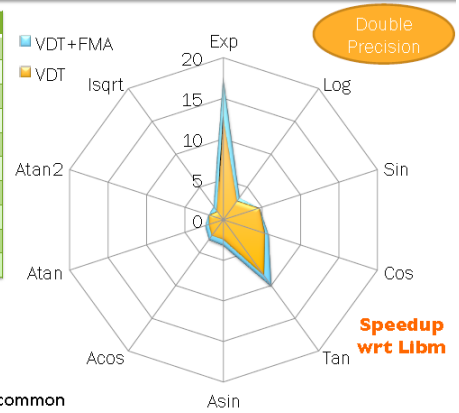


*Vectorizing the detector Geometry to op1mize Par1cle transport, **A.Gheata et al.***

Speed: VDT Vs Libm

Fnc.	Libm	VDT	VDT-FMA
Exp	102	8	5.8
Log	33.3	11.5	9.8
Sin	77.8	16.5	16.5
Cos	77.6	14.4	13.2
Tan	89.7	10.6	8.9
Asin	21.3	8.9	6.9
Acos	21.6	9.1	7.3
Atan	15.6	8.4	6.7
Atan2	36.4	19.9	18.9
Isqrt	5.7	4.3	2.8

Time in **ns** per value calculated

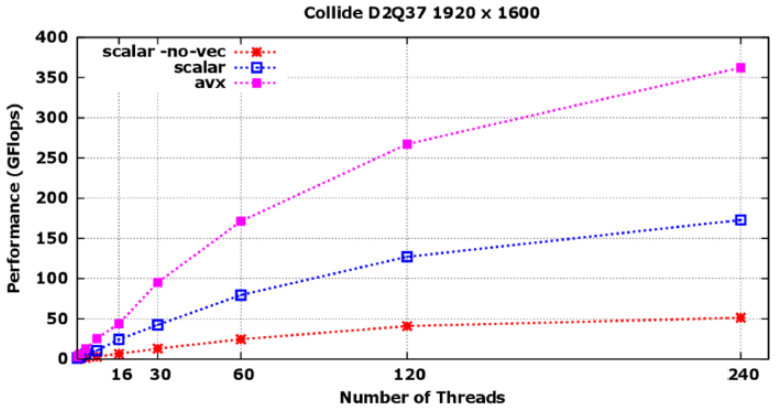


FMA: Fused Multiply Add $d = a + b \times c$

- Operative input range: [-5k, 5k]
- **Speedup factors of >5** not uncommon

Computing on Knights and Kepler Architectures, Sebastiano Fabio Schifano

Collide



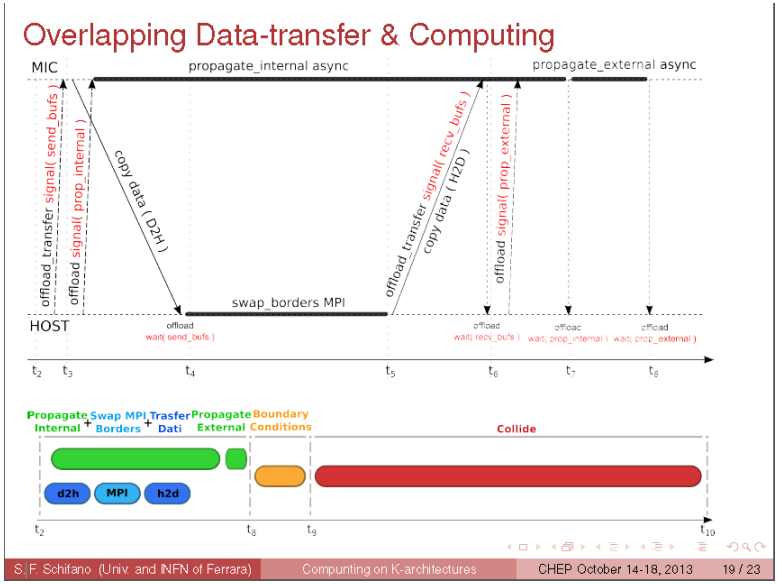
- scalar: `icc -mmic -O3 -openmp -novec`, 240 threads, $\epsilon \approx 5\%$
- scalar: `icc -mmic -O3 -openmp`, 240 threads, $\epsilon \approx 15\%$
- vector: intrinsic, openmp, 240 threads, $\epsilon \approx 30\%$

S. F. Schifano (Univ. and INFN of Ferrara) | Computing on K-architectures | CHEP October 14-18, 2013 | 15 / 23

Speeding up HEP experiments' software with a library of fast and autovectorisable mathematical functions, D. Piparo et al

Think about…

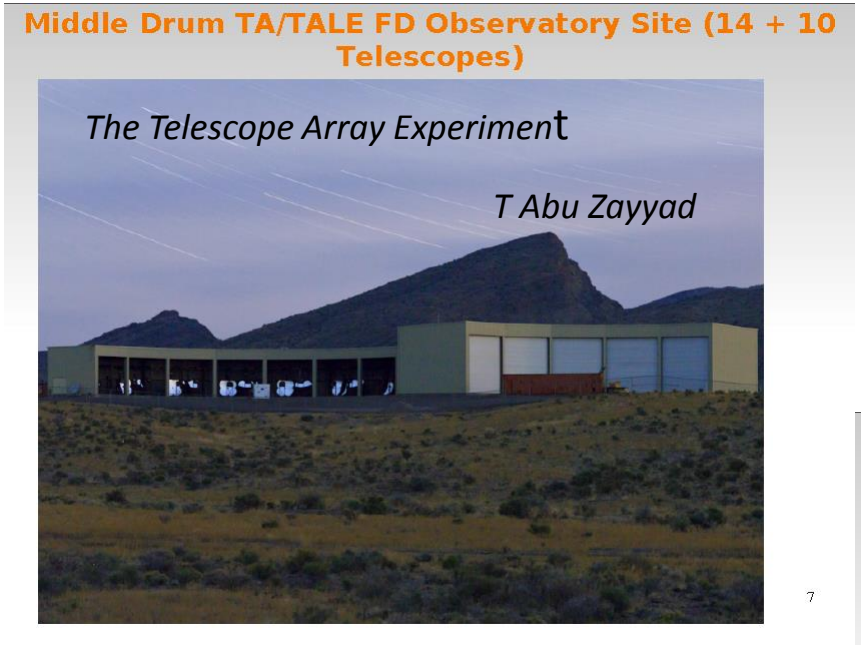
- Communication overhead with accelerators
  - Overlap it with computation
  - Batch more tasks
- Precision
- Memory Layout



Pros:

+ Scheduling intrinsically immune to deadlocks
+ Possible to lump data from different events for computations on accelerators
+ Possibility to run same algorithm repeatedly on the same core: cache friendly

Precision



Excellent speed-up on GPU
Tested single & double precision
"If single is OK – then don't use double"

Performance comparison – running time

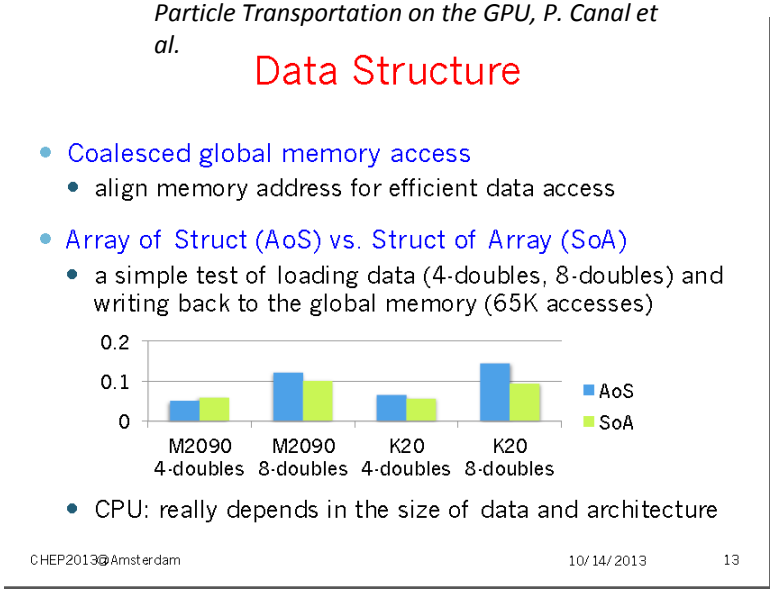
- Desktop CPU vs. Desktop GPU

E (eV)	100 evt (cpu) t in sec.	100 evt (gpu) t in sec.	1000 evt (cpu) t in sec.	1000 evt (gpu) t in sec.
10^{17}	13.91	0.759 (18x)	140.97	2.860 (49x)
10^{18}	18.76	0.875 (21x)	164.59	3.348 (49x)
10^{19}	24.64	1.006 (24x)	196.99	4.230 (47x)
10^{20}	40.72	1.297 (31x)	364.44	6.834 (53x)

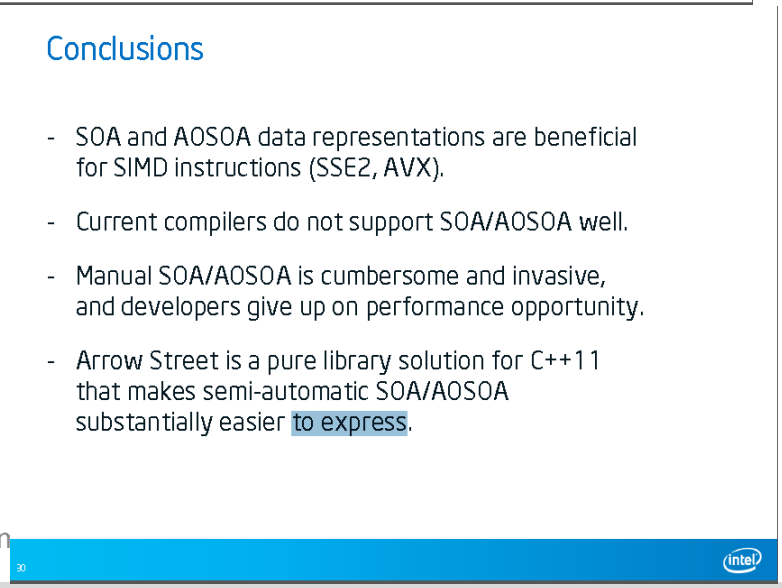
Memory Layout

- Alignment
- Array of Structures vs. Structure of Arrays
- Possible solution
  Arrow Street from INTEL

High Energy Electromagnetic Particle Transportation on the GPU, P. Canal et al.

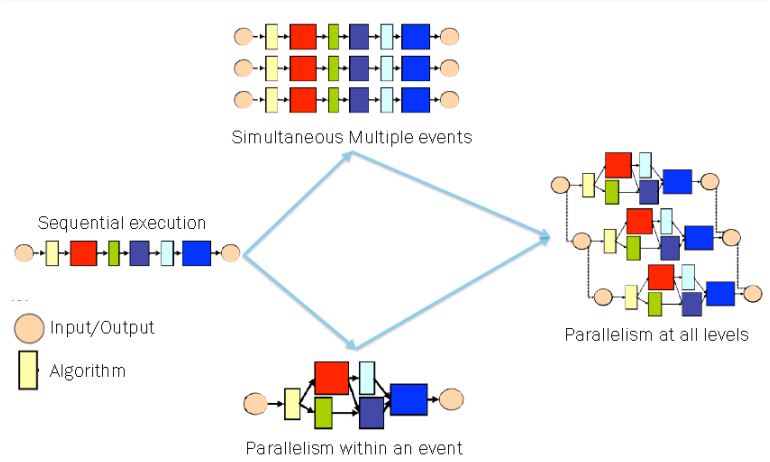


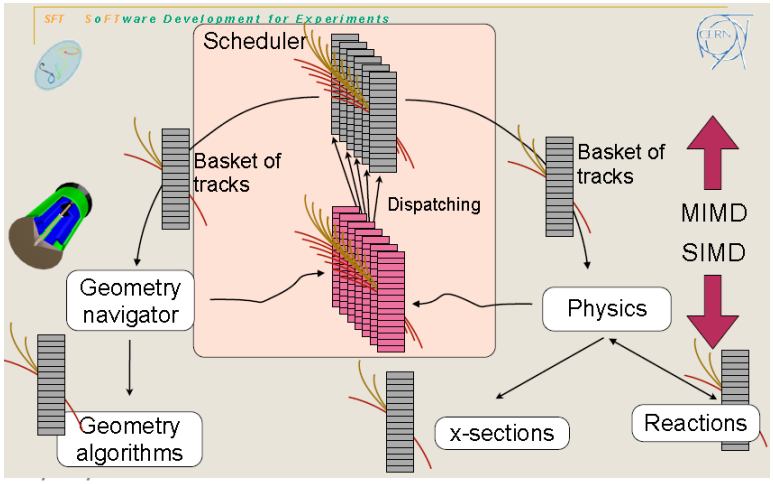
Arrow Street: Semi-automatic SOA / AOSOA , Pascal Costanza



Concurrency

- Adapt applications and application frameworks to many-core systems to exploit different sources of parallelism.
  - Multiple events
  - Within an event
  - Within an algorithm





A well-separated pairs decomposition algorithm for kd-trees implemented on multi-core architectures

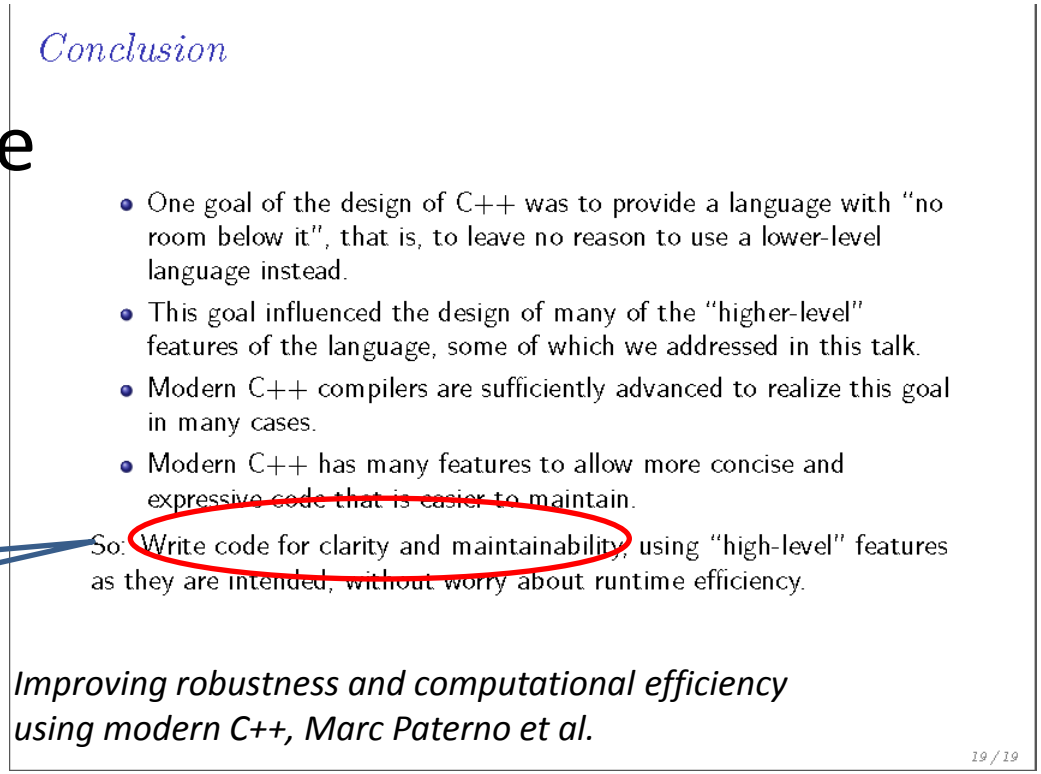
Raul H. C. Lopes
Ivan D. Reid
Peter R. Hobson

Particle Physics Group, School of Engineering and Design, Brunel University

C++ ++

- Finally we got C++11
- Easier to use
- Can we trust the compiler to generate optimal code?
- Yes!

"Write code for clarity and maintainability"



Conclusion

- One goal of the design of C++ was to provide a language with "no room below it", that is, to leave no reason to use a lower-level language instead.
- This goal influenced the design of many of the "higher-level" features of the language, some of which we addressed in this talk.
- Modern C++ compilers are sufficiently advanced to realize this goal in many cases.
- Modern C++ has many features to allow more concise and expressive code that is easier to maintain.

So: Write code for clarity and maintainability, using "high-level" features as they are intended, without worry about runtime efficiency.

Improving robustness and computational efficiency using modern C++, Marc Paterno et al.

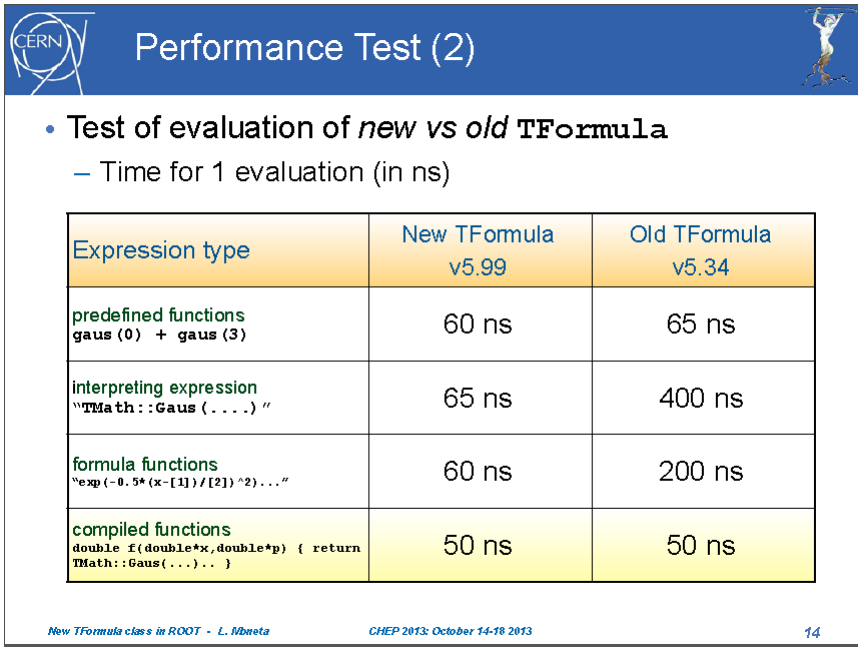
19 / 19

Software Engineering

- LHC experiments are using the Long Shutdown to optimize code and infrastructure.
  - Optimizing
  - Profiling
  - Quality Assurance
  - Integrated tools.
    - Particular attention to release and build management.
    - LHCb, ATLAS & CMS

Making code better

- An example from ROOT
- Improving TFormula



Performance Test (2)

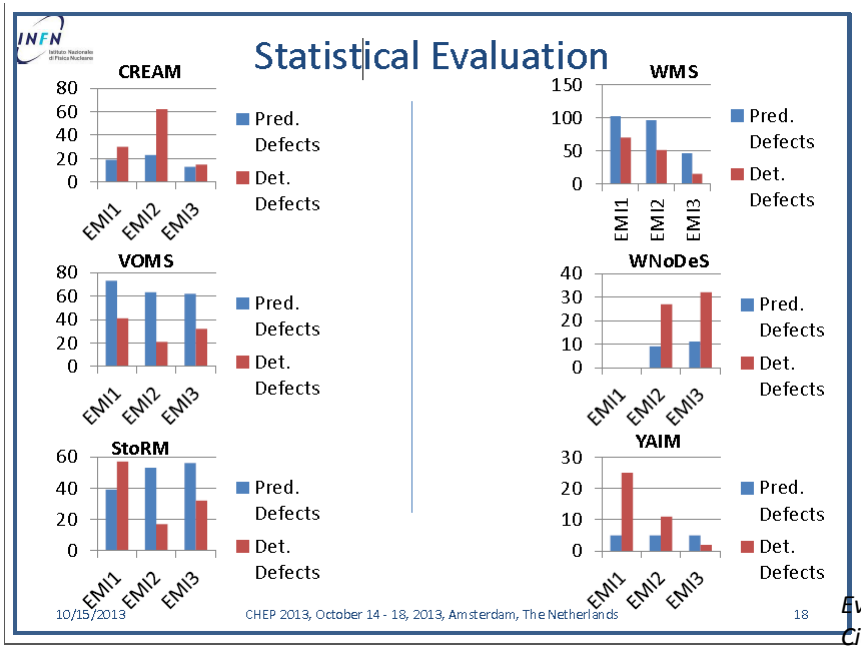
- Test of evaluation of *new vs old* **TFormula**
  - Time for 1 evaluation (in ns)

Expression type	New TFormula v5.99	Old TFormula v5.34
predefined functions `gaus(0) + gaus(3)`	60 ns	65 ns
interpreting expression `"TMath::Gaus(....)"`	65 ns	400 ns
formula functions `"exp(-0.5*(x-[1])/[2])^2)..."`	60 ns	200 ns
compiled functions `double f(double*x,double*p) { return TMath::Gaus(...).. }`	50 ns	50 ns

New ROOT TFormula class, *Fons Rademakers et al.*

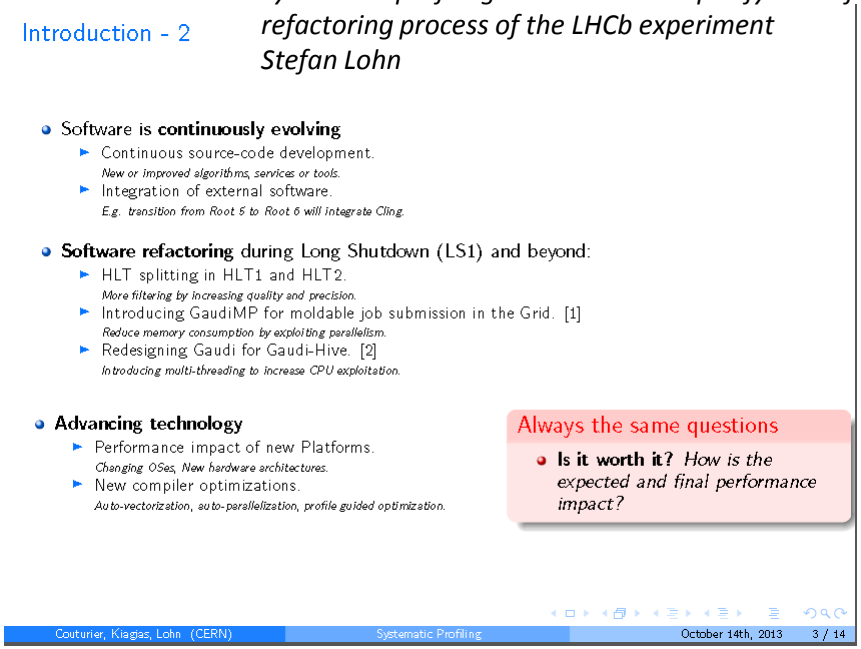
How bad are you? (And are you getting better?)

- Need performance metrics, and need to find the most profitable places to intervene ➔ profiling.



Evaluating Predictive Models of Software Quality, Vincenzo Ciaschini,

Systematic profiling to monitor and specify the software refactoring process of the LHCb experiment Stefan Lohn



Quality Assurance

- The easiest bugs to cure are the ones that aren't in your code.

- Rucio is the new data management system of ATLAS
- Its development follows a modified waterfall process
  - From conceptual design
  - Via architectural design
  - To test-driven and peer-reviewed implementation
- Initial inhibition threshold well overcome
  - Social uncertainties almost negligible
  - Conduct enforced where possible by software
- Resulted in
  - High throughput of essentially error free code
  - Easy injection of new engineers into the team

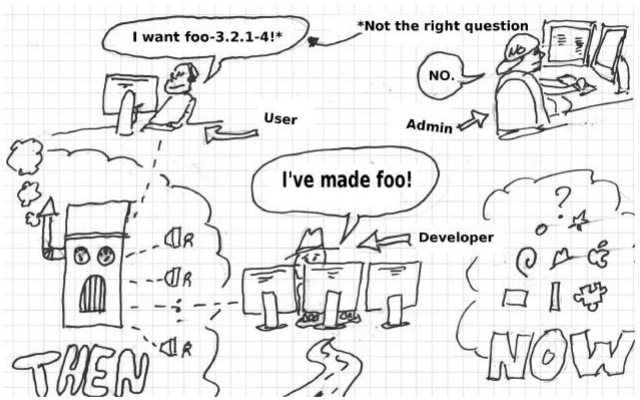
The ATLAS data management software engineering process
Mario Lassnig

Release Build & Packaging

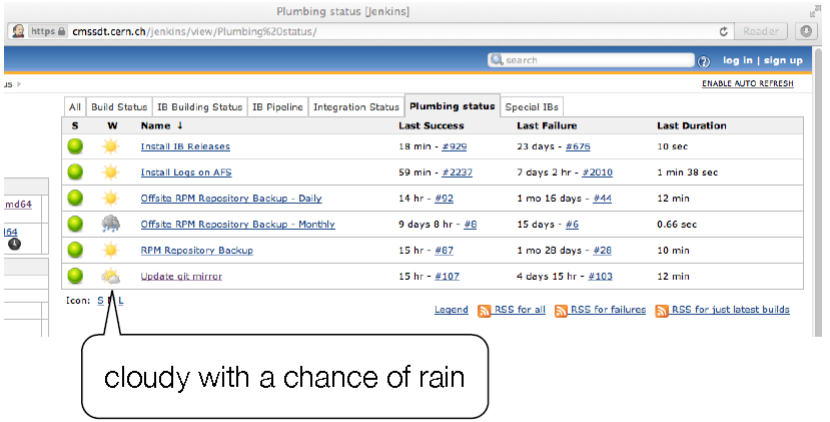
- Many examples!
- Move to open source tools
- Jenkins /Github etc.

Experiences with moving to open source standards for building and packaging
Dennis van Dok,

Software midwifery



acrontab on steroids



The Rise of the Build Infrastructure Guilio Ulisse

What was missing?

Software Engineering, Parallelism & Multi-Core

CPU/GPU architectures; tightly-coupled systems; GPGPU; concurrency; vectorization and parallelization; mathematical libraries; foundation and utility libraries; programming techniques and tools; software testing and quality assurance; configuration management; software build, release and distribution tools; documentation.

No progress to report!

Robert Lupton may be right!
''My current theory is that we should give up on scientists writing introductory and how-to documents and instead employ professionals working in close collaboration with the development team.''

Software engineering for science at the LSST, Robert Lupton Monday Morning Plenary