

Software for FCC Physics and Experiments

B. Hegner, CERN

for the FCC Experiment Software Team

FCC Week 2015

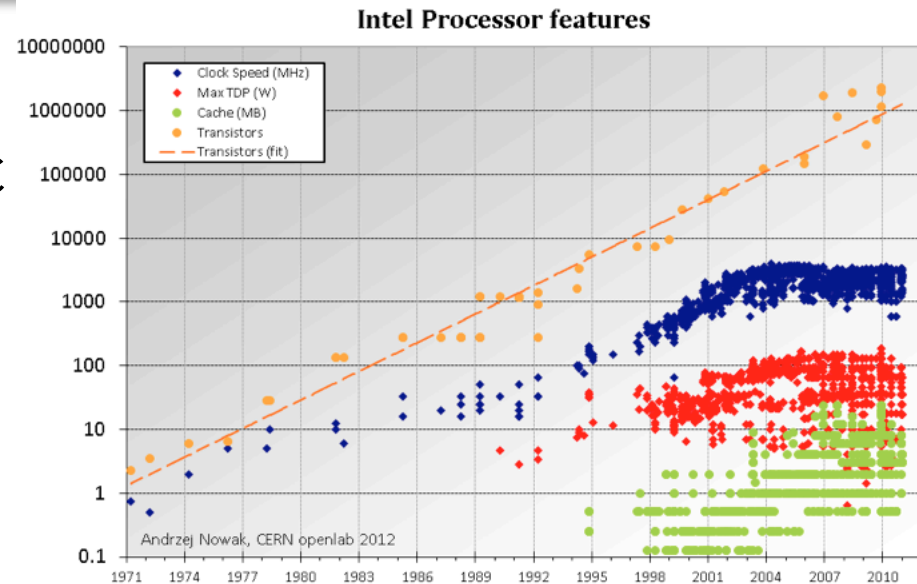
Washington DC



- Provide robust software to allow physics studies for CDR in 2018
- Support all **FCC-ee, -eh, and -hh** communities at the same time
 - Requires flexibility for Geometry and Simulation
- Start pragmatically
- As studies progress move to more sophisticated solutions
 - Allow components to be replaced later on
- FCC software effort relies on effort of other people
 - There is a give and take
 - Aim for, but don't blindly force, synergy with other communities

The Power Wall - Importance of Parallelism

- In the past speed increase happened automatically - just wait for your next PC
- That is over now!
- New CPU improvements go into parallelization
- Clock speed will not increase because of power consumption:

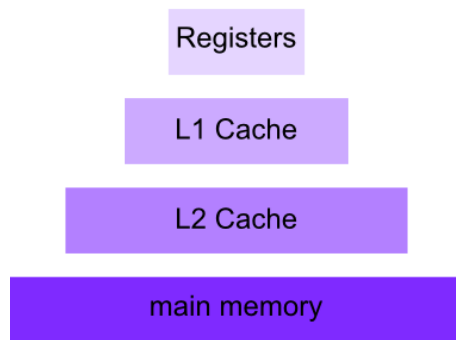
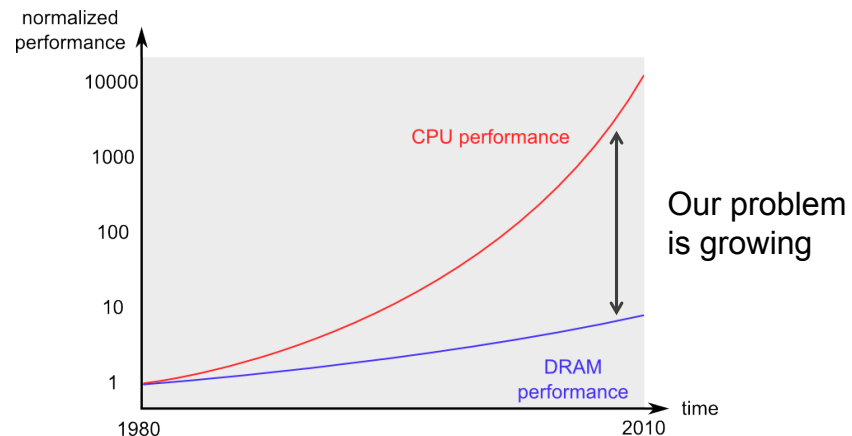


$$Power \propto Frequency^3$$

- **Need to adapt our software to parallel HW architectures right from the start**
 - Otherwise we waste the computing resources we urgently need

The Memory Wall - Getting data is slow

- Back in the 70's memory used to be faster than CPUs.
- Nowadays, **memory** compared to CPUs is **incredibly slow!**
- Mitigated by introducing more and more levels of caches in hardware



Technology	Capacity	Latency
SRAM	bytes	< 1 ns
SRAM	kilobytes	1 ns
SRAM	megabytes	< 10 ns
DRAM	gigabytes	70-100ns

source: Jens Teubner "Data processing on modern hardware"

- **Need to adapt our SW design to the physical memory hierarchy**

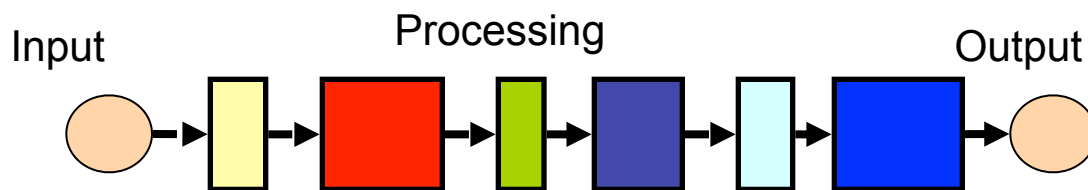
- Architecture landscape is changing heavily
 - Dominance of x86 coming to an end
 - ARM, GPUs, ...
- Software technology itself progresses
 - New programming languages and paradigms
 - Even C++ going through rapid evolution (C++11, 14, 17, ...)
- We cannot know what will be there by the time of FCC
 - but we know that the relative stability of the last two decades is over

- Adapt existing solutions from LHC
 - Gaudi as underlying framework
 - ROOT for I/O
 - Geant4 for simulation
 - C++ and Python for user analysis
- Adapt software developments from ILC/CLIC
 - DD4Hep for detector description
- Invest in **better fast vs. full sim integration**
 - Geant4 fastsim, Atlfast (ATLAS)
- Invest in **proper future-proof data model**
 - The LHC experiments' ones are over-engineered
 - The ILC/CLIC model (LCIO) was designed before power and memory wall



H,A → $\tau\tau$ → two jets + X, 60 fb⁻¹

- Gaudi is an event-independent data processing framework
 - Used by LHCb, ATLAS, and a few smaller experiments
- Based on the concept of a software bus
- Work is split up in interdependent “algorithms”



- Parallelization effort with “GaudiHive” to take advantage of ever increasing hardware parallelization

- FCC Software needs to support the studies of multiple detectors
- At different stages different level of detail required
 - Smearing vs. fast sim vs. full sim
- FCC choices are
 - Delphes (*) and HepSim (**)
 - Fast simulation in Python
 - Integrated fast/full simulation with Geant4
- Should all be accessible from within the same framework

(*) <http://delphes.hepforge.org>

(**) <http://atlaswww.hep.anl.gov/hepsim/>

Detector Description in LHC experiments is a not-well organized environment

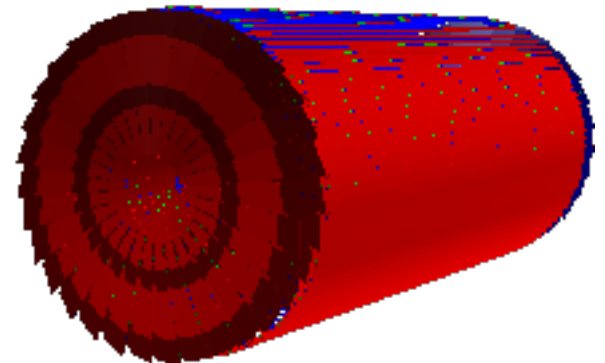
- Detectors modeled long ago and expertise largely gone
- Work on upgrade reveals weaknesses
- Heterogeneous setups even within experiments

ILC/CLIC efforts triggered the project DD4hep (*)

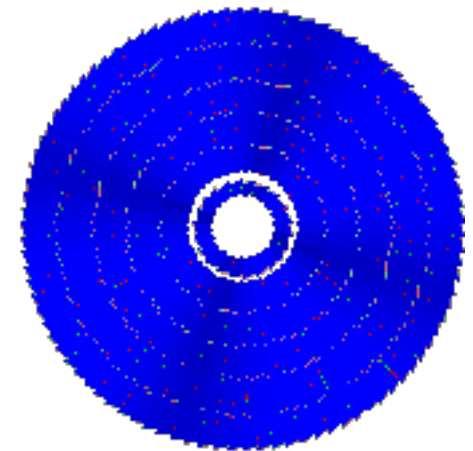
- Covering simulation, display, alignment in a consistent way

FCC joined these efforts of DD4hep

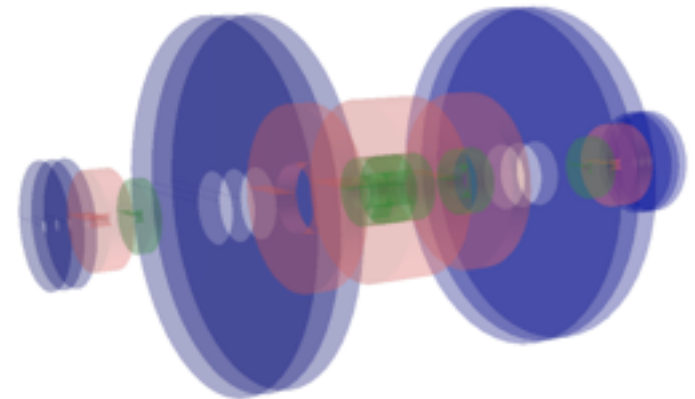
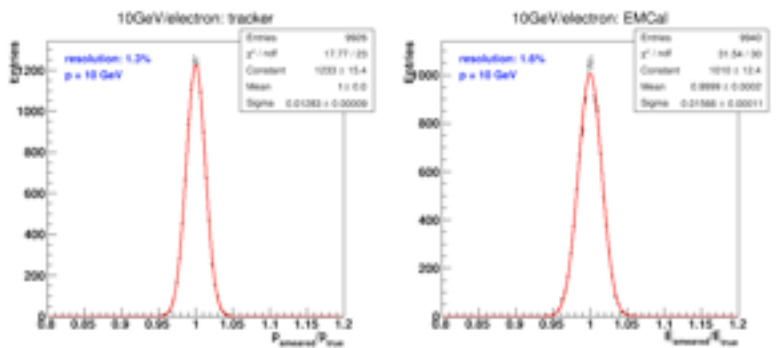
- Good support by developers!
- Working on first test-detector



- Goal is to have a **combined fast and full simulation**
 - Decide at the config level where to do what
- (Semi-) automatic extraction of fast simulation parameters from full simulation
 - To be able to do fast-sim for any detector design
- Though not re-inventing the wheel, we are heavily re-designing it



- First development phase was focussed on producing a demonstrator
 - Using expertise from ATLAS and Geant4 developers
 - Chosen approach worked out nicely
 - Results now being integrated into Geant4 and Gaudi



The FCC requirements for a good data model are not special at all:

- Simplicity
- Flexibility
- Completeness

Data Models of LHC experiments are proven to work

- Fairly complex, and very detector specific beasts

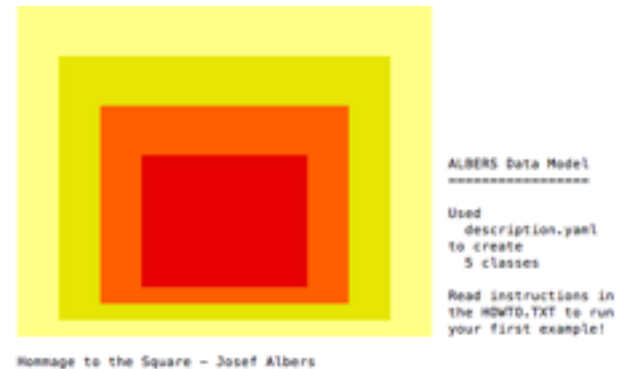
The ILC community has a simple, but complete data model (LCIO)

- Needs adaption to allow direct ROOT access outside FWK
- Parallelism and memory optimization not part of the core design
- Developers interested in extension and one should take advantage of it

The proper data model is **essential** for allowing good results

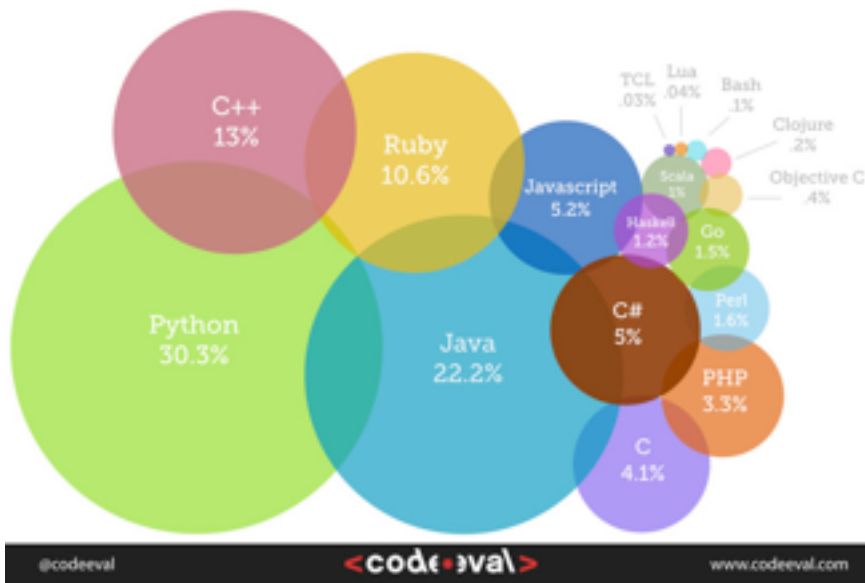
Thus it is worth investing here with a new project!

- ROOT as first choice for I/O
- No deep object hierarchies
 - Wherever possible concrete types
- Simple memory layout
 - Employ simple structs instead of fat objects
 - Helps with parallelization
- Allow access from Python and C++
 - Only loose coupling with event processing framework
- Quick turnaround for improvements
 - Employ code generation
- Wrote a demonstrator data model
 - Used throughout all developments now



- Analysis should be easy and powerful
- Lesson from **LHC experiments** and **ILC/CLIC**
 - If setup is too complex, physicists stop using common software and create their own mini-frameworks
- Physicists will join from different experiments and we need to make the transition as easy as possible
- Need to allow **multiple paradigms** to do analysis
 - **C++ and Python**

Most Popular Coding Languages of 2014

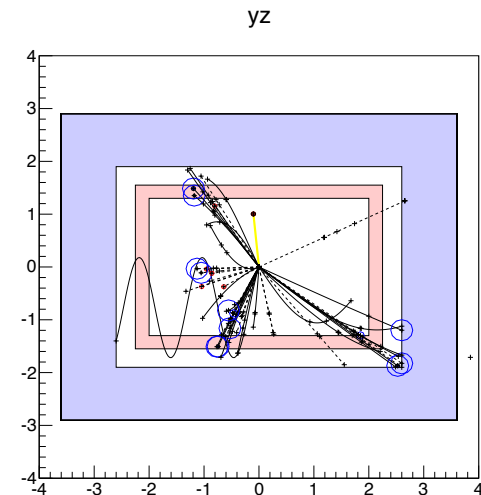
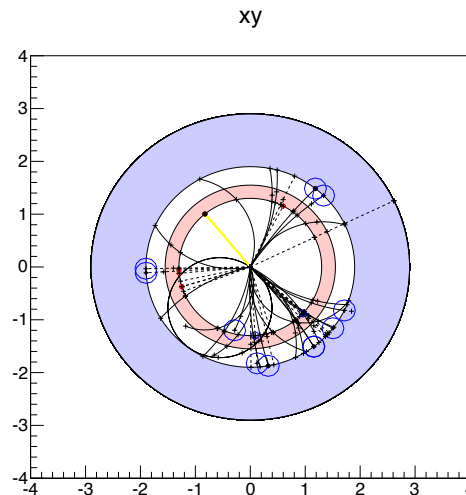


- Very large user base
- Super easy to learn
- Light & short code
- Good performance
 - usually wraps C or C++ modules
- « Batteries included »
 - massive and easy-to-use standard library
- Dynamic typing
 - good for multichannel analyses
 - code highly reusable
- Dynamic object modification
 - Can attach new attributes (or methods) to an existing object
- Productivity x 5-10 w/r C++
- A lot of fun!

- Supporting this with the **heppy** package originating from CMS

- Fast simulation for prototyping of high-level algorithms
 - Written in Python it allows quick turn-around
- Uses the same data model as the C++ framework
 - Lowers the bar when moving to production code

**FCC
Particle Flow
Prototyping**



- Established common FCC experiment software project
- Base software environment in place
- First phase of pick & chose is finished
- Integrated fast/full sim design validated
- Data Model demonstrator finished
- C++ and Python based analysis environment provided

- Framework
 - Core event data model, Gaudi integration, Software stack
- Generators
 - Integration
- Simulation infrastructure
 - Geant4 (fast & full)
 - Delphes integration
- Reconstruction
- Analysis tools
 - python & C++ framework
- Validation
 - testing and performance
- Computing
 - sample production and management

Bernet, Hegner



Pilicer, **People needed**



Carminati, Dell'aqua, Hrdinka, Salzburger, Zaborowska



De Gruttola, Hegner



People needed



Many thanks to our hard working team!

Bernet



Hegner, **People needed**



People needed

