The Abdus Salam International Centre for Theoretical Physics

6th Workshop on Collaborative Scientific Software Development and Management of Open Source Scientific Packages

Ivan Girotto – igirotto@ictp.it

High-Performance Computing Applications Specialist

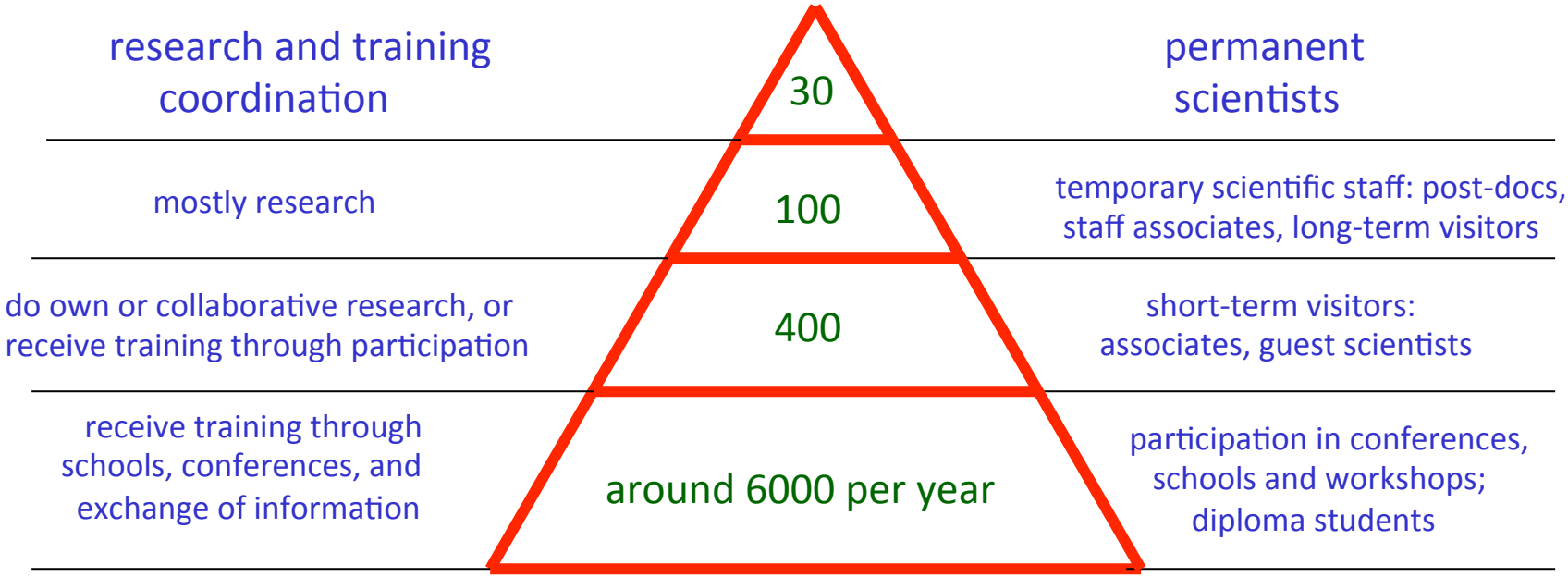
International Centre for Theoretical Physics (ICTP)

The Abdus Salam
International Centre
for Theoretical Physics

United Nations
Educational, Scientific and
Cultural Organization

IAEA
International Atomic Energy Agency

ICTP from Trieste to the World



Over 200.000 visit/year to the ICTP media (see www.ICTP.TV) for remote training!

Mission - An institute run by scientists for scientists

- Foster the growth of advanced studies and research in physical and mathematical sciences, especially in support of excellence in developing countries.
- Develop high-level scientific programmes keeping in mind the needs of developing countries, and provide an international forum of scientific contact for scientists from all countries.
- Conduct research at the highest international standards and maintain a conducive environment of scientific inquiry for the entire ICTP community.
- Thanks to the generous funding from the Italian Government, UNESCO and the IAEA, ICTP has been able to initiate and implement various schemes of support and assistance to scientists from developing countries.

ICTP Scientific Calendar

- Schools, Conferences, Workshops around the year
- Half of them on subjects related to main research areas (core)
- The rest on many subjects:
  medical physics, optics, nano physics, plasma physics, electronics, high-performance scientific computing, biophysics, satellite navigation, science dissemination and e-learning, m-science, entrepreneurship, nuclear physics (IAEA), teacher training, 3-D Printing, etc...
- http://www.ictp.it/scientific-calendar.aspx

ICTP IN NUMBERS 2016

ICTP VISITORS 2016

5827 VISITORS [25% FEMALE] FROM **135** NATIONS; **56** TRAINING ACTIVITIES ON CAMPUS, **21** IN DEVELOPING COUNTRIES

8 DAYS AVERAGE LENGTH OF VISIT FOR CONFERENCE PARTICIPANTS

57 DAYS AVERAGE FOR RESEARCH VISITORS

59 POSTDOCS ON CAMPUS [47% FROM DEVELOPING COUNTRIES]

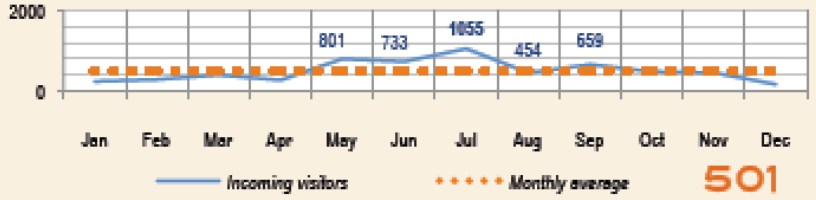
232 STUDENTS ENROLLED IN PRE-PHD EDUCATIONAL PROGRAMMES

367 SCIENTISTS ENGAGED IN CAREER DEVELOPMENT PROGRAMMES

TOP 10 DEVELOPING COUNTRIES, BY REGION

Africa	N. of visitors	Asia	N. of visitors	Latin America	N. of visitors
South Africa	85	India	378	Brazil	170
Ghana	79	Iran	324	Argentina	160
Nigeria	78	China	175	Mexico	77
Egypt	69	Pakistan	72	Colombia	51
Cameroon	48	Malaysia	45	Cuba	49
Morocco	41	Singapore	44	Chile	33
Algeria	35	Turkey	43	Venezuela	18
Tunisia	32	Korea Rep.	39	Costa Rica	17
Kenya	30	Philippines	28	Peru	17
Senegal	30	Viet Nam	24	Guatemala	10

FLOW OF INCOMING VISITORS BY MONTH AT ICTP



501

22 REGIONAL TRAINING ACTIVITIES

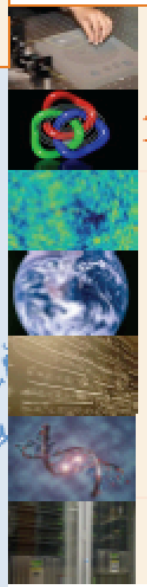
6 - MATHEMATICS (MATH)
5- APPLIED PHYSICS (AP)
5 - EARTH SYSTEM PHYSICS (ESP)
3 - CONDENSED MATTER PHYSICS (CMSP)
2 - HIGH ENERGY PHYSICS (HECAP)
1- QUANTITATIVE LIFE SCIENCES (QLS)

DISTRIBUTION OF FEMALE VISITORS

60% OF FEMALE VISITORS ARE FROM DEVELOPING REGIONS:

Asia-Pacific	416
Africa	184
Latin America	168
Eastern Europe	119

COURSE PARTICIPANTS BY RESEARCH AREA



1324 AP

1168 CMSP

814 HECAP

697 ESP

575 Math

359 QLS

281 HPC (High Performance Computing)

1,500 MONTHS OF TRAINING TO COURSE PARTICIPANTS LECTURED BY MORE THAN 1,300 EXPERTS

COUNTRIES REPRESENTED

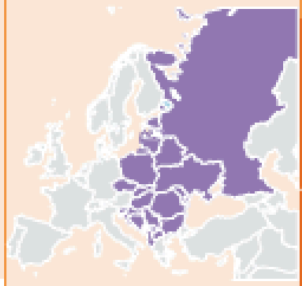
20 LDCs in Africa

17 from rest of Africa

6 LDCs in Asia

30 from rest of Asia

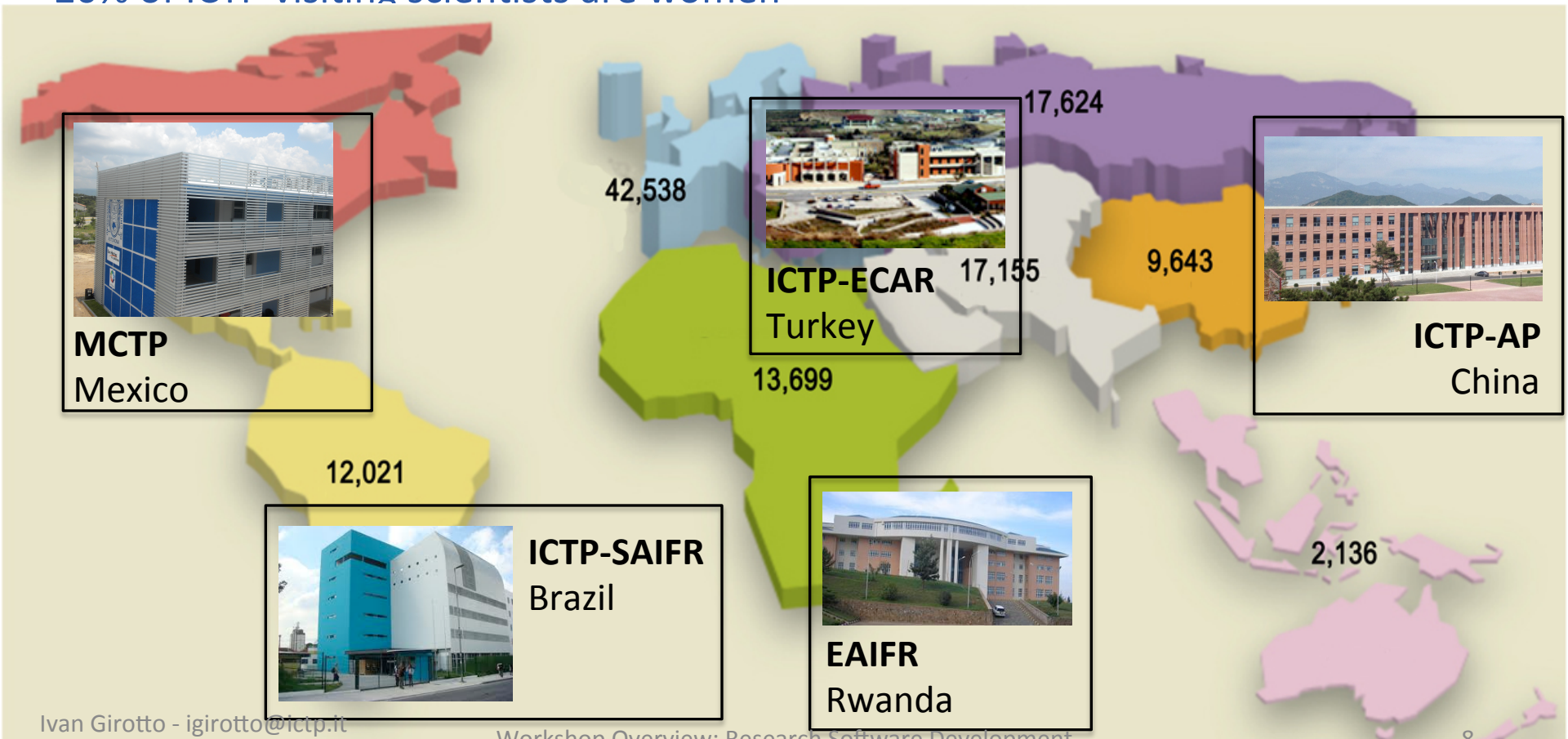
19 from Latin America



19 from Eastern Europe

- More than 140,000 visits since 1970
- 190 countries represented
- 20% of ICTP visiting scientists are women

++NEW++
The ICTP Partner Institutes

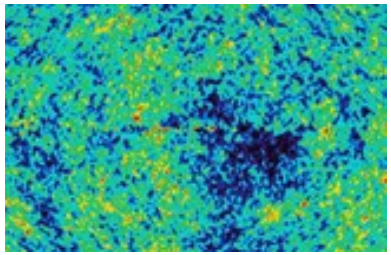


The Abdus Salam International Centre for Theoretical Physics

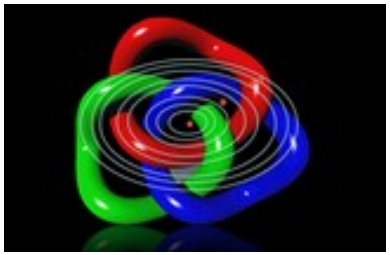
PRE-PHD PROGRAMMES	DEGREE PROGRAMMES	CAREER DEVELOPMENT	LABORATORY OPPORTUNITIES	SCIENTIFIC OUTREACH
ICTP Postgraduate Diploma Programme	Joint ICTP/SISSA PhD Programme in Physics and Mathematics	Conferences, workshops and schools	Training and Research in Italian Laboratories	Office of External Activities
ICTP/IAEA Sandwich Training Education Programme	Joint PhD Programme, Earth Science and Fluid Mechanics	Junior Associates	ICTP-ELETTRA Users Programme	ICTP Partner Institutes
	Physics PhD Program	Regular Associates	ICTP Laboratories	Science Dissemination Unit
	Joint Masters in Physics	Senior Associates		African Review of Physics
	Joint ICTP/Collegio Carlo Alberto Program in Economics	Federated Institutes		ICTP in East Africa
	International Master, Physics of Complex Systems	OFID Postgraduate Fellowship		Physics Without Frontiers
	Master of Advanced Studies in Medical Physics	The Kuwait Programme at ICTP		
	Masters in High Performance Computing			

Scientific Sections

High Energy Cosmology and Astroparticle Physics

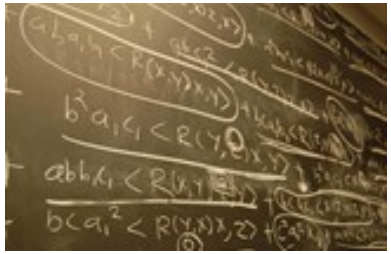


Condensed Matters and Statistical Physics



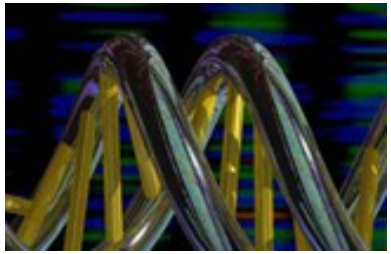
Earth System Physics

Mathematics



Applied Physics

New areas



HPC Staff and Collaborators

Dr. David Grellsheid
Herwig Software Manager
@ Durham University

Dr. Axel Kohlmeyer
Full Professor of Research
@ Temple University

Dr. Clement Onime
Responsible IT/HPC Infrastructure
@ ICTP

me
HPC Application Specialist
@ ICTP

High-Performance & Scientific Computing activities at the ICTP

- HPC service and HPC application consulting
  - in house HPC facility (Argo)
  - research enablement on massively parallel systems for HPC on both national service (CINECA) and EU infrastructures (PRACE)
- Dissemination & Training on HPC and Scientific Programming to the ICTP Scientific Community

MHPC in pills: www.mhpc.it

- High-level educational program: not an Ms.C. program!
- Intensive training aimed to build knowledge in solving complex problems with an HPC approach
- Innovative, hands-on based training
- Aimed to people with strong interest in:
  - advanced programming for scientific computing
  - software optimization
  - management of computing platforms
  - data management and data analytics

MHPC
Master in High Performance Computing

Background Requirements

- Candidates must have some experience in programming and a competence in at least one of the languages between C, C++ and/or Fortran
  - Python knowledge is a plus
- A sound knowledge of Linux operating system
- Master level of a scientific degree is required
- No prior HPC knowledge is assumed
- Enthusiasm is a must

1 year program divided in 6-8 months courses and 6 month project (some overlap)

Mandatory

- Scientific Programming Environment
- Introduction to Computer Architectures for HPC
- Object Oriented Programming
- Parallel Programming
- Introduction to Numerical Analysis
- Advanced Computer Architectures and Optimizations
- Parallel Data Management and Data Exchange
- High Performance Computing Technology
- Best Practices in Scientific Computing

Optional Choice

- Data structures, sorting and searching algorithms in serial and parallel
- Lookup tables, cell lists and neighbor lists
- Domain decomposition techniques
- Parallel FFT techniques
- Parallel Linear Algebra
- Multipole expansion, multi-grid methods
- Adaptive Meshes
- Maximum likelihood techniques
- Cluster or network or graph analysis
- Monte Carlo methods
- Agent-based models
- Automatic differentiation
- DFT from source to code

Search

Search in Conferences:

Overview

Programme

Speakers

Apply here

6th Workshop on Collaborative Scientific Software Development and Management of Open Source Scientific Packages | (smr 3199)

Starts 28 Apr 2018
Ends 9 May 2018
Central European Time

Sharif University of Technology
Physics Department
Azadi St. - Tehran - Islamic Republic of Iran

Writing software has become central to research in many fields of science. This school aims to give early-career scientists an introduction to a variety of topics that help them to write efficient, clean, maintainable and long-lived code that is useful beyond solving an immediate problem. In a mixture of talks and many hands-on sessions, the focus lies on showing best practices and building fundamental skills in creating, extending and collaborating on modular and reusable software.

TOPICS:

- Python / shell scripts as glue code
- Mixing programming languages
- Introduction to computer architectures and software optimization
- Modular, reusable software design
- Effective collaborative development with multiple co-authors
- Version control and release cycles
- Automated testing frameworks
- Structured documentation
- Systematic debugging

Organizers

D. Grellscheid (Durham University / ICTP), S. Baghram (Sharif University), M.R. Ejtehadi (Sharif University), A. Langari (Sharif University), S. Moghimi-Araghi (Sharif University), ICTP Scientific Contact: I. Girotto

Co-sponsors

Sharif University of Technology

Workshop Overview

Ivan Girotto – igirotto@ictp.it

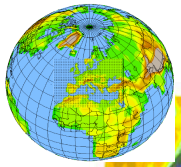
Information & Communication Technology Section (ICTS)

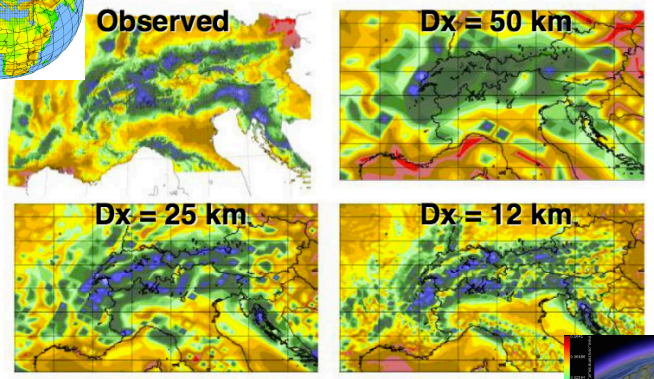
International Centre for Theoretical Physics (ICTP)

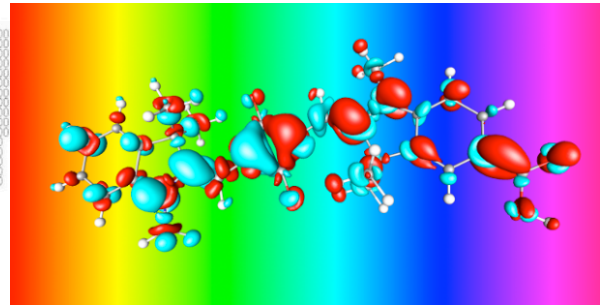
Why use Computers in Science?

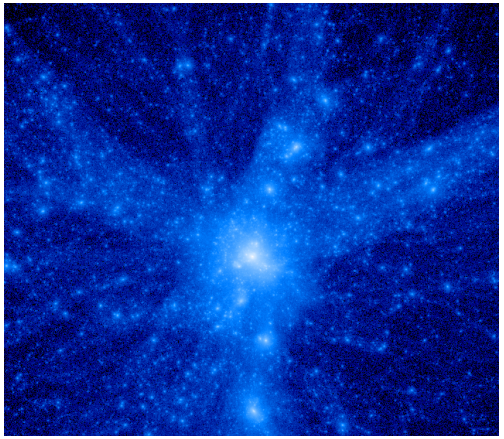
- Use complex theories without a closed solution: solve equations or problems that can only be solved numerically, i.e. by inserting numbers into expressions and analyzing the results
- Do “impossible” experiments: study (virtual) experiments, where the boundary conditions are inaccessible or not controllable
- Benchmark correctness of models and theories: the better a model/theory reproduces known experimental results, the better its predictions

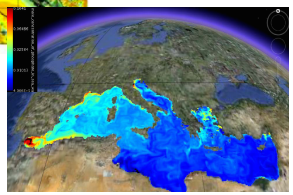
SW in Science

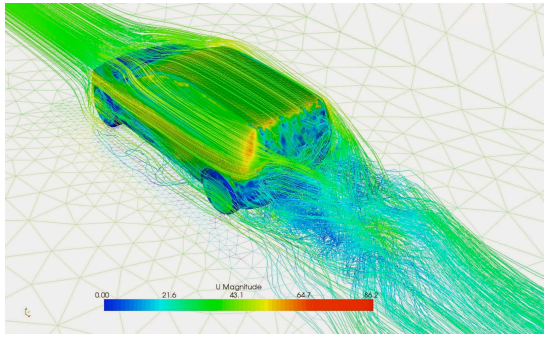




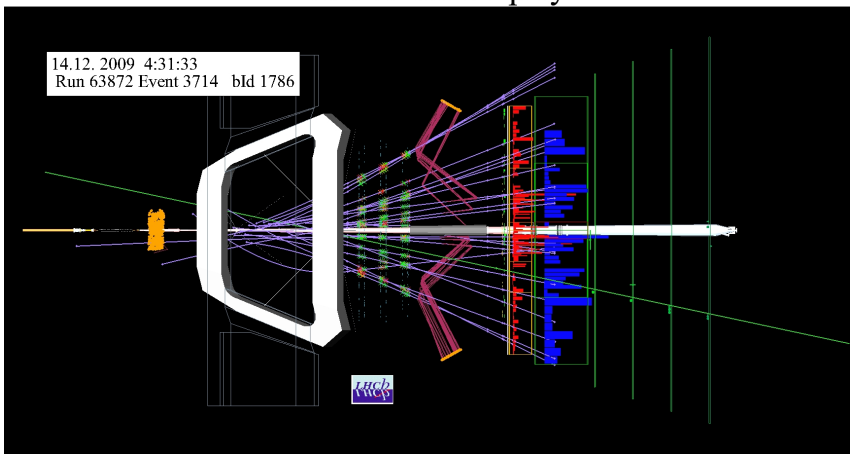




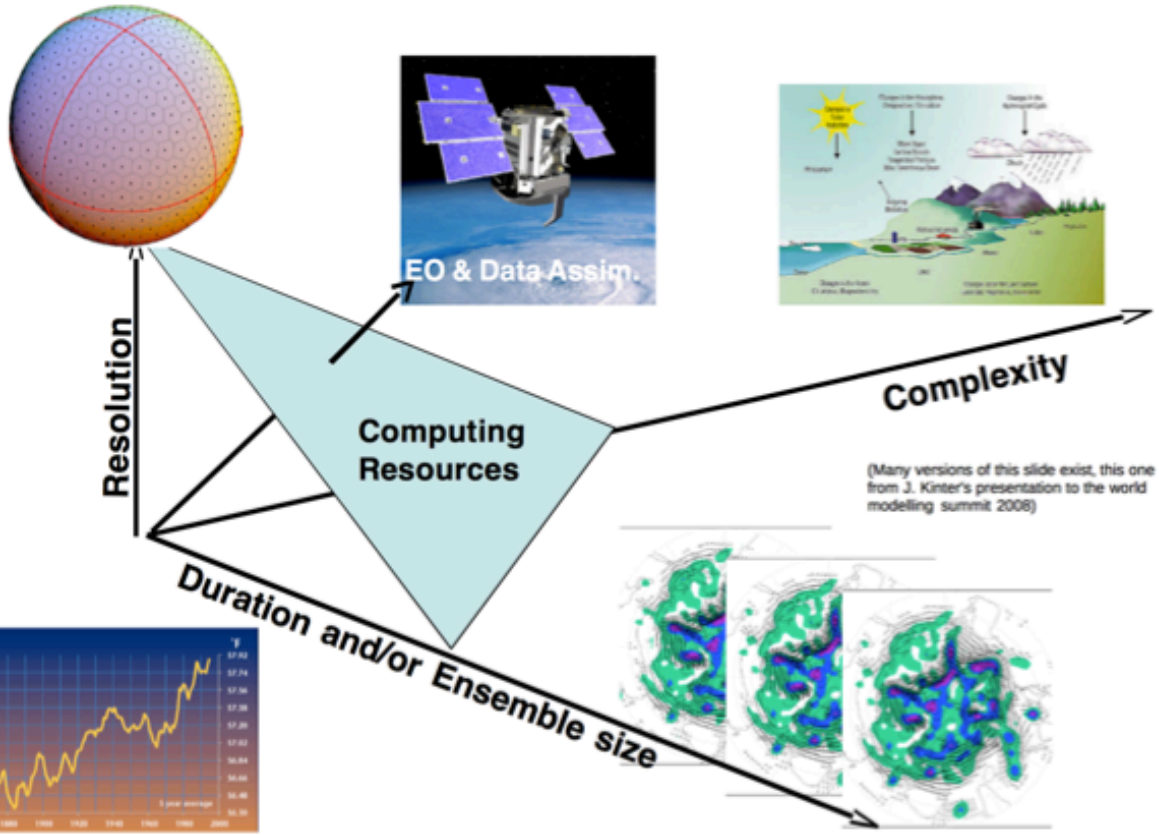




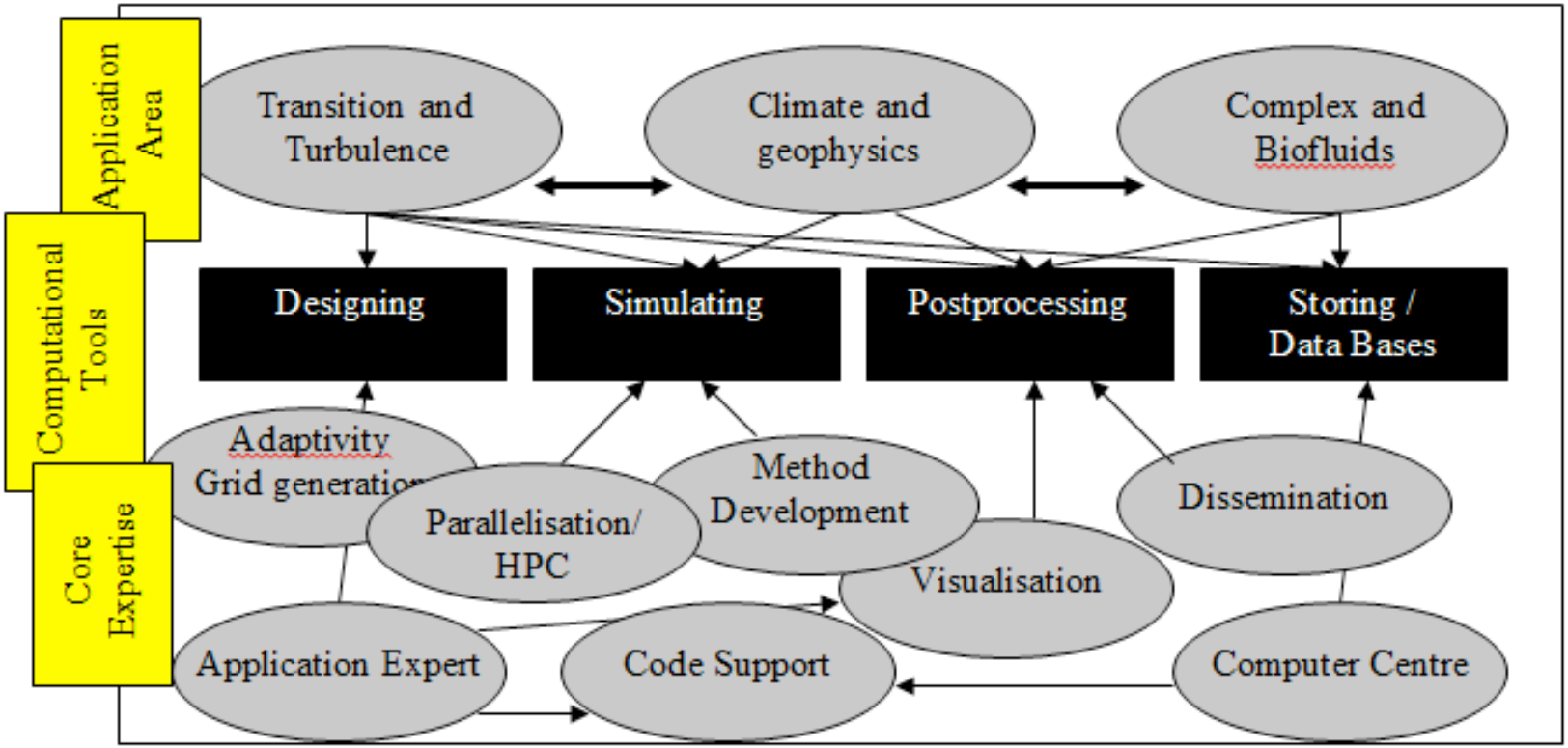
LHCb Event Display



More & More Computing …



SW flow in science



How can we make all of this happen in a single code?

Not a question of feasibility but of how we develop software:

- Is every student developing their own software?
- Or are we re-using what others have done?
- Do we insist on implementing everything from scratch?
- Or do we build our software on existing libraries?

There has been a major shift on how we approach the second question in scientific computing over the past 10-15 years!

Complexity of software

Many scientific applications are several orders of magnitude larger than everything you have probably ever seen!

- For example, a crude measure of complexity is the number of lines of code in a package (as of 2018):
  - Deal.II has 1.1M
  - PETSc has 720k
  - Trilinos has 3.3M
- At this scale, software development does not work the same as for small projects:
  - No single person has a global overview
  - There are many years of work in such packages
  - No person can remember even the code they wrote

Conventional Software Development Process

- Start with set of requirements defined by customer (or management):
  - features, properties, boundary conditions
- Typical Strategy:
  - Decide on overall approach on implementation
  - Translate requirements into individual subtasks
  - Use project management methodology to enforce timeline for implementation, validation and delivery
- Close project when requirements are met

What is Different in the Scientific Software Development Process?

- Requirements often are not that well defined
- Floating-point math limitations and the chaotic nature of some solutions complicate validation
- An application may only be needed once
- Few scientists are programmers (or managers)
- Often projects are implemented by students (inexperienced in science and programming)
- Correctness of results is a primary concern, less so the quality of the implementation

About Software (Observations)

- Most research software is not of high quality
  - Typically written by graduate students:
    - without a good overview of existing software
    - with little software experience
    - with little incentive to write high quality code
  - Often maintained by postdocs:
    - with little time
    - need to consider software a tool to write papers
  - Advised by faculty
    - with no time
    - oftentimes also with little software experience
- How does this affect our field (Reproducibility? Archival? “Standing on the shoulders of giants”?)
- There is a complexity limit to what we can get out of a PhD student.

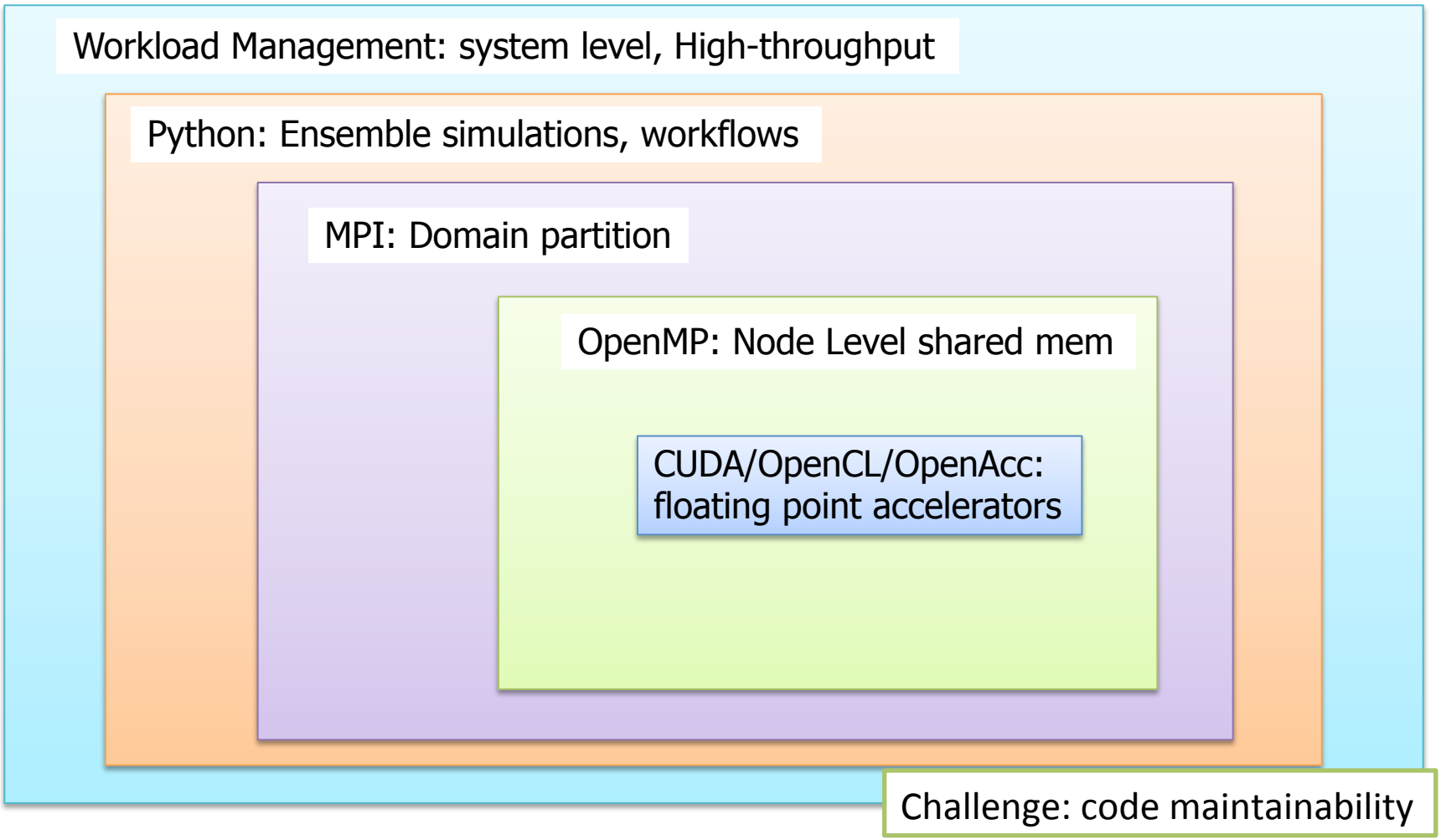
Complexity of software

The only way to deal with the complexity of such software is to:

- Modularize: different people are responsible for different parts of the project.
- Define interfaces: only a small fraction of functions in a module is available to other modules
- Document: for users, for developers, for authors, and at different levels
- Test, test, test: on proper software packages testing requires same development effort of writing the software

What Else?

- Computers become more powerful all the time and more complex problems can be addressed
- Solving complex problems requires combining expertise from multiple domains or disciplines
- Use of computational tools becomes common among non-developers and non-theorists
  - many users could not implement the whole applications that they are using by themselves
- Current hardware trends (SIMD, NUMA, GPU) make writing efficient software complicated


Workload Management: system level, High-throughput
Python: Ensemble simulations, workflows
MPI: Domain partition
OpenMP: Node Level shared mem
CUDA/OpenCL/OpenAcc:
floating point accelerators
Challenge: code maintainability

Modular Programming & Libraries

- Many tasks in scientific computing are similar
  - Tasks differ only in some subset of the calculation
  - Calculations use common operations like fast Fourier transforms (FFT), basic linear algebra, etc.
  - Data can be represented in a structured file format supported by generic analysis & visualization tools
- There is a large potential for code reuse
- Independent modules can be better validated
- Reusable code is better target for optimization

Source Code Management

- Not only a way to archive sources, but a tool for communication between developers
- Distributed source code management makes concurrent development easier
- Work with feature branches and merge often
- Commit changes in small increments and do not combine unrelated changes in on commit
- Have consistent, documented “whitespace rules” and best enforce them before committing

What makes such projects successful?

- Success or failure of scientific software projects is not decided on technical merit alone
- The true factors are beyond the code! It is not enough to be a good programmer! In particular, what counts:
  - Utility and quality
  - Documentation
  - Community
- All of the big libraries/packages provide this for their users.

The Bottom Line

- Many of these concepts and methods can help improve scientific software development
- Important: it is not the tools by themselves, but how they are used that makes the difference
- Fight the urge to take shortcuts and see the restrictions that modular and object oriented programming imposes as opportunities
- Finding the right balance is key to success
- Never underestimate the longevity of your code

Conclusions

- Computational science has spent too much time where everyone writes their own software.
- By building on existing, well written and well tested, software packages:
  - We build codes much faster
  - We build better codes
  - We can solve more realistic problems
- Scientific software development has to be recognized as a task requiring trained specialists and dedication of time and resources to produce dependable results
- Contribute to scientific software means to be part of a community, including social interactions, advantages and rules to be respected

A Roadmap to the Workshop

- Focus on software development concepts
- Introduce tools and processes for organizing development and maintenance
- Discuss strategies and best practices
- Explore methodology that encourages collaborative software development
- Favor writing reusable software frameworks
- Work in groups with complementary expertise

Thanks for your attention!!