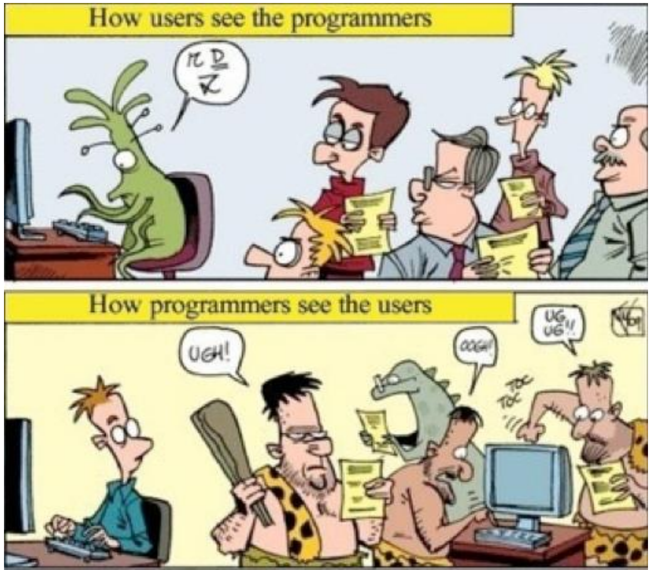
FAIRROOT
PAST, PRESENT AND FUTURE

The Past

- Once upon a time there was an experiment which needs to proof its feasibility.
- In the year 2003 two brave physicists appeared and searched in the HEP kingdoms for some existing framework which could be used for this task.
- After a long journey they came back empty-handed but with many good ideas.
- CbmRoot mainly based on ideas/design from AliRoot and the simulation and analysis software of the Hades experiment.
- After some time also other experiments interested in the development.
  - Decouple experiment independent code (FairRoot) from experiment dependent code (CbmRoot).

Framework for Simulation, Reconstruction and Analysis

- Let the physicist concentrate on their tasks
  - Don’t bother them with IO, parameters …
  - Should allow fast development cycles
  - Flexible to easily change experimental setup
  - Easy to install
  - Easy to use
  - Extensible for new developments
- Reuse existing software and tools
  - Don’t reinvent the wheel if not necessary
- Use standards as much as possible
  - Here ROOT comes into play
  - IO; Geometry; Virtual MC; …



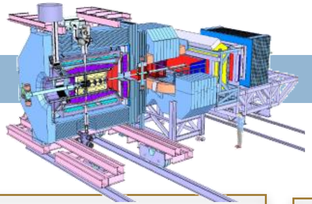
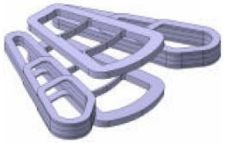
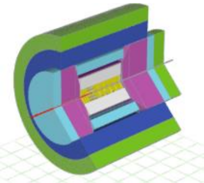
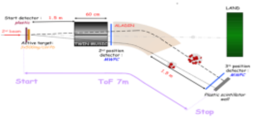
The Presence

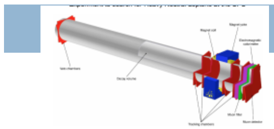
- Try to make users life as easy as possible
  - Provide scripts which install all the needed dependencies
  - Modern build system using CMake
    - Continuous Integration
    - Coverage Analysis
    - Dashboard
- Code runs on Mac OSX and (all) Linux flavors
  - Tested on
    - Mac OSX 10.6 – 10.10; OpenSuse 13.1 and 13.2; Fedora 19, 20 and 21; Debian 6, 7, and 8; SLC 6, and 7; Ubuntu 14.04, and 14.10 (all 64bit)
    - Ubuntu 14.04 (32bit)
    - Probably many more different flavors on user systems

The Presence (cont.)

- Open Source project (LGPL v3) for Simulation, Reconstruction, and Analysis of HEP experiments
- Hosted at Github
  - https://github.com/FairRootGroup/FairRoot
- Used by ~ 10 experiments
  - Mostly at GSI
  - Some at CERN
  - Some more are evaluating FairRoot
- Core Development team at GSI
  - Each experiment has one developer with a shared position experiment/core team
    - Improves communication between core team and experiments
  - Many developments from the experiments went into FairRoot

FairRoot Universe



Start testing the VMC concept for CBM

2004

First Release of CbmRoot

Panda decided to join->

FairRoot: same Base package for different experiments

2006

MPD (NICA) start also using FairRoot

R3B joined

2010

ASYEOS joined (ASYEOSRoot)

EIC (Electron Ion Collider BNL)

EICRoot

2011

GEM-TPC seperated from PANDA branch (FOPIRoot)

SOFIA (Studies On Fission with Aladin)

2012

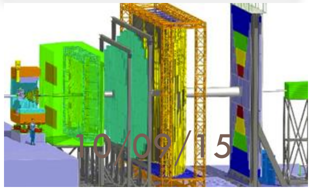
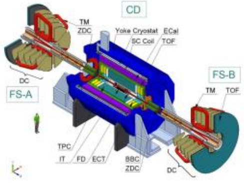
CALIFA (CALorimeter for the In Flight detection of γ rays and light charged pArticles)

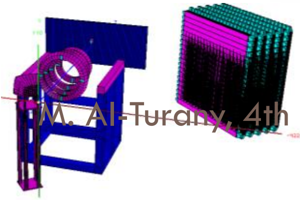
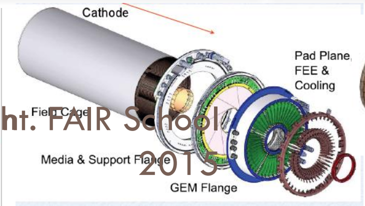
2013

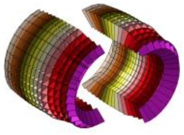
SHIP - Search for Hidden Particles

2014

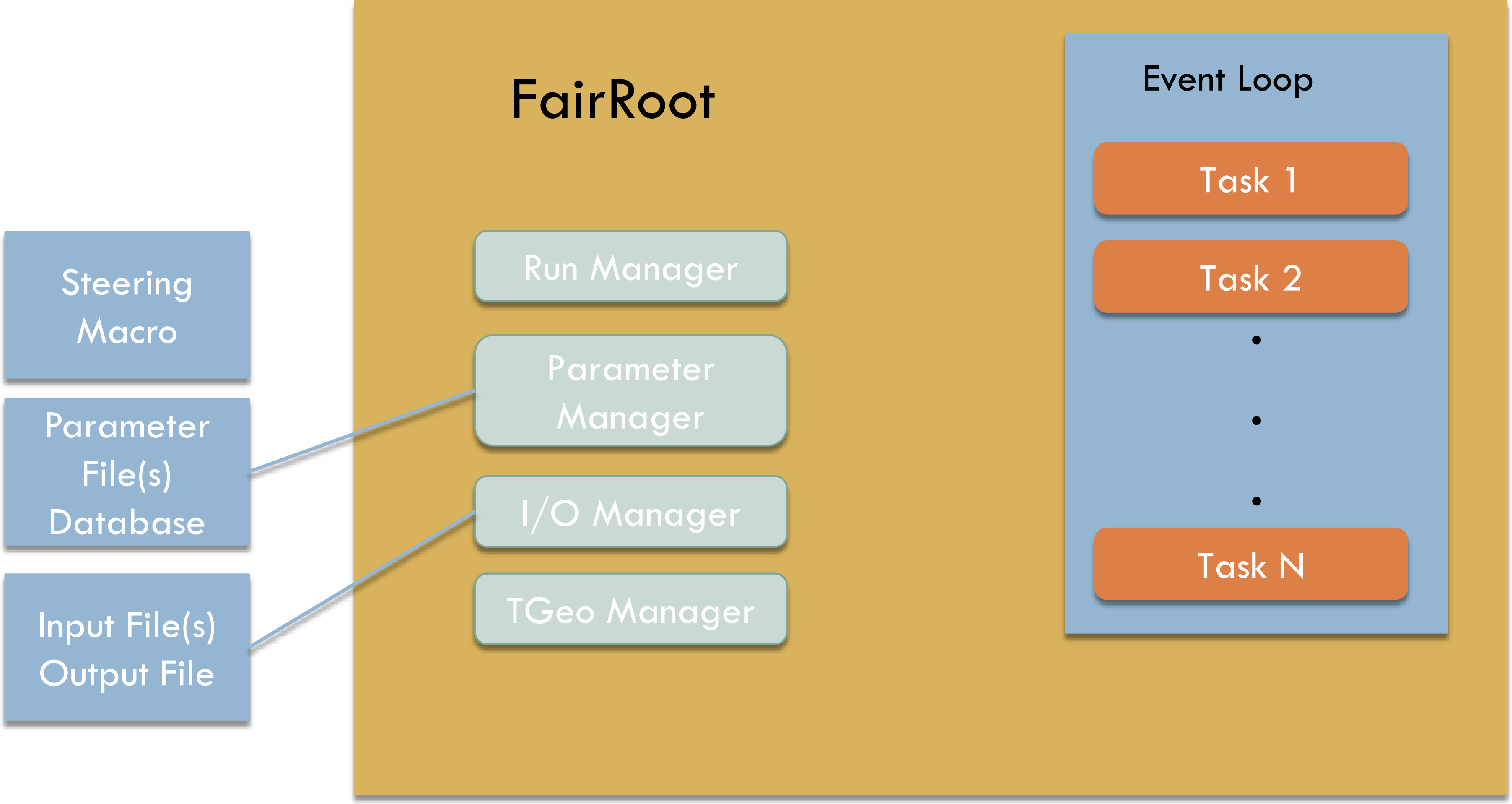
ENSAR-ROOT Collection of modules used by structural nuclear phsyics exp.

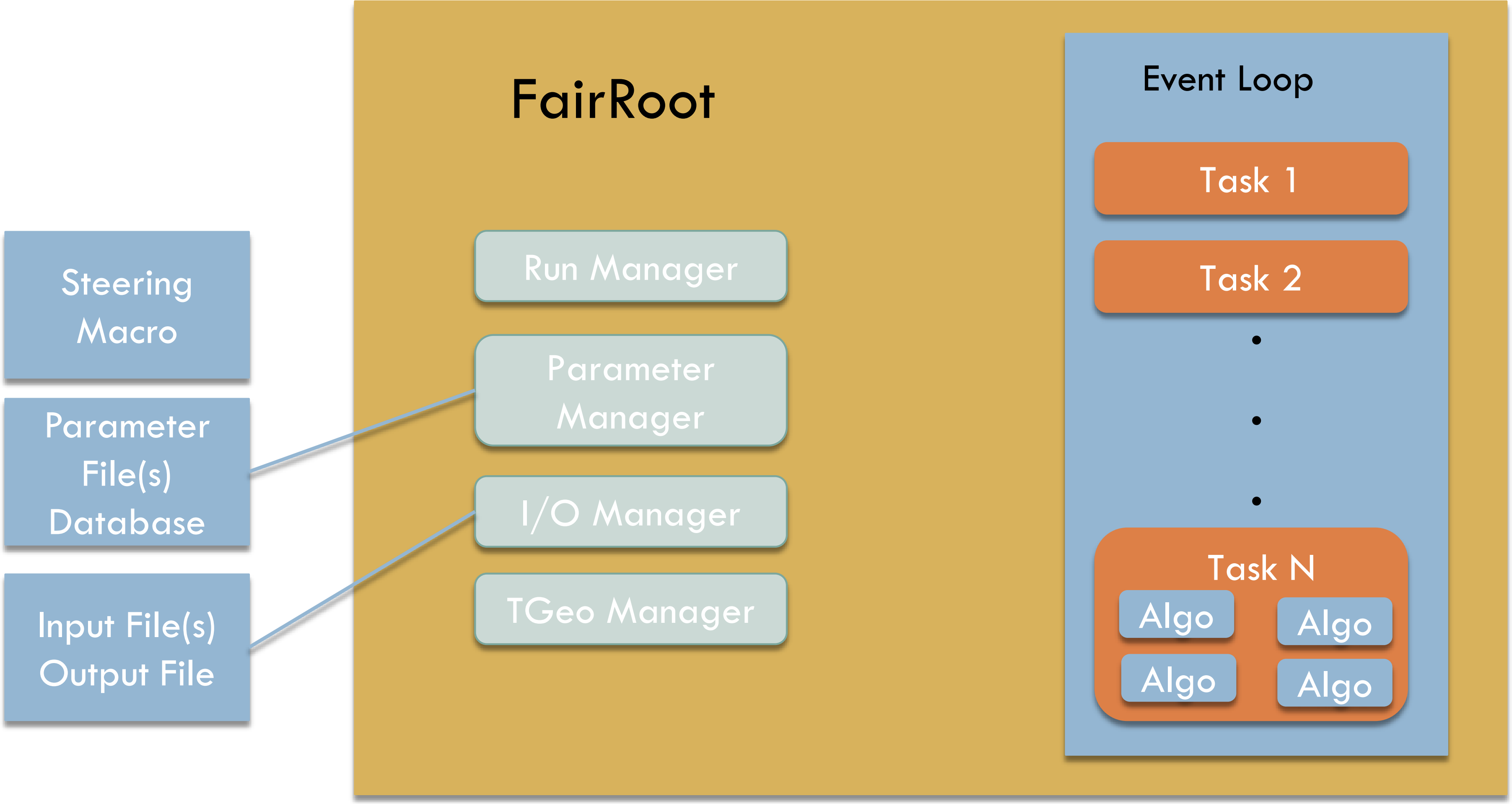





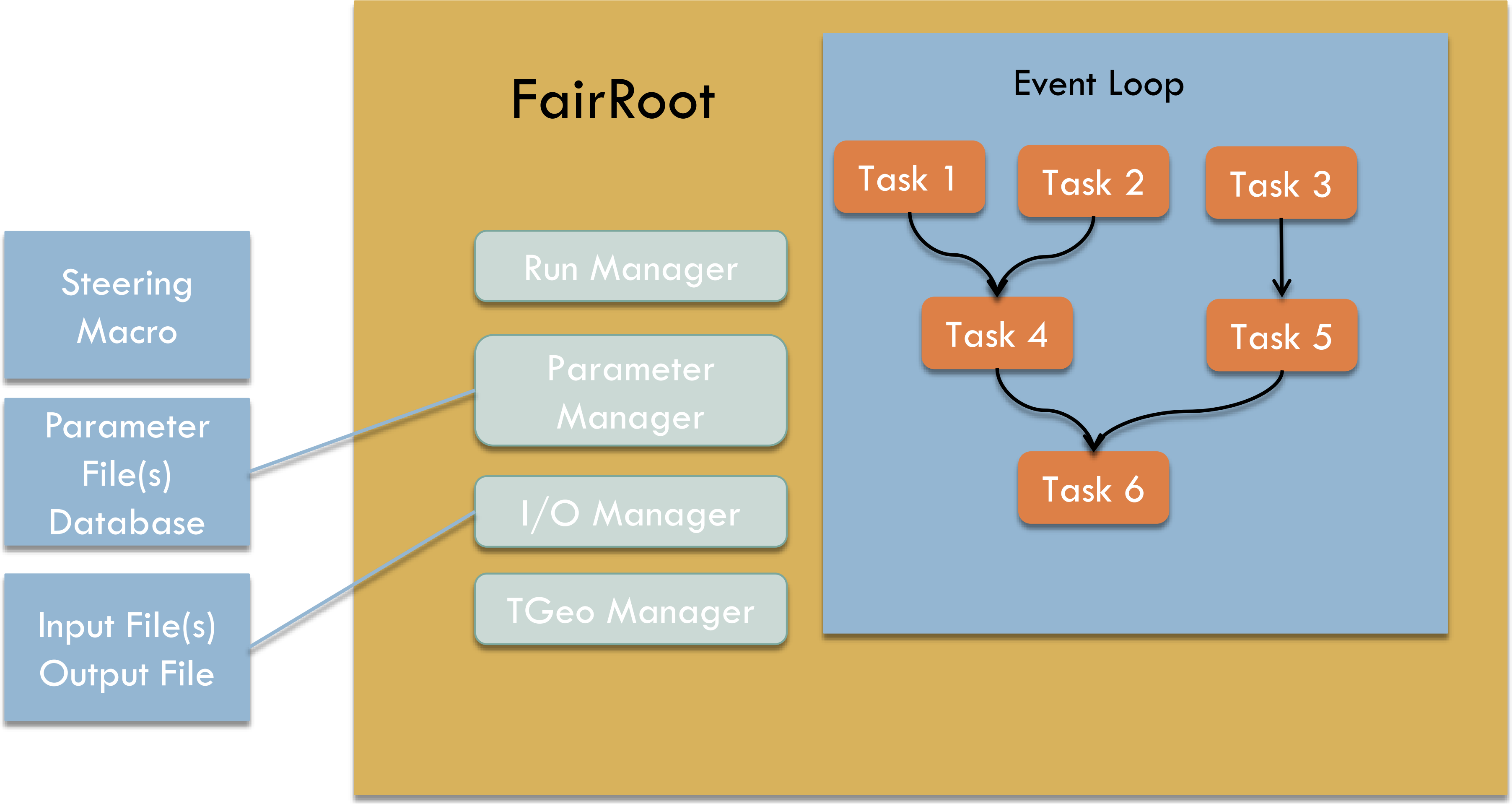
Current Layout: Reconstruction



FairRoot: parallelized Tasks



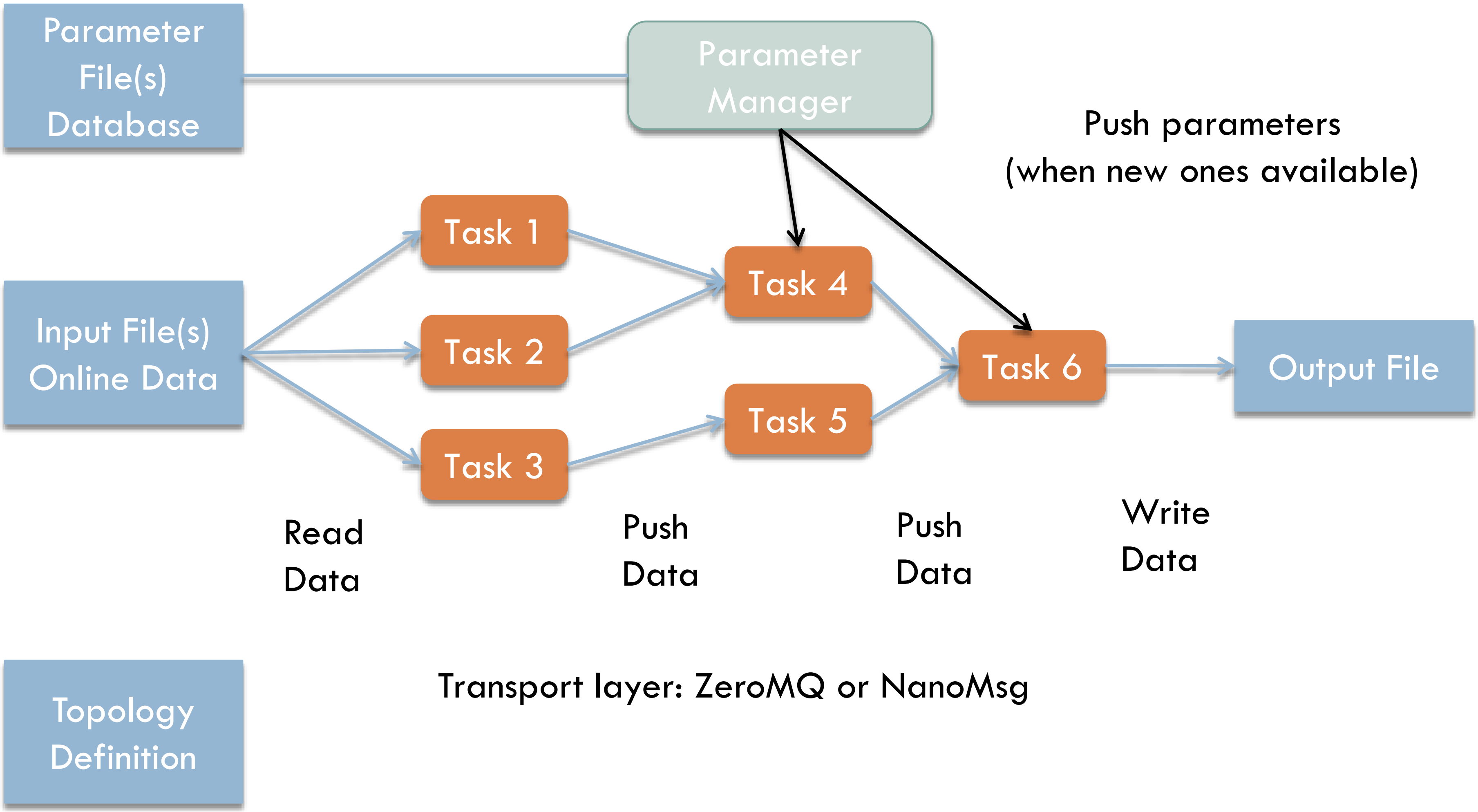
FairRoot: parallel Tasks



FairRoot and Concurrency

- Start Tasks in different independent threads.
  - Huge changes in the framework needed
  - Management overhead?
    - How to handle concurrent data access?
  - Program is still monolithic
    - If one tasks crashes the whole program may crash
  - Program has still one (ROOT) event loop
    - Execution of next event can only start when the execution of the previous event has finished

Multi Process FairRoot: FairMQ



Parameter File(s) Database

Parameter Manager

Push parameters (when new ones available)

Task 1

Task 2

Task 3

Task 4

Task 5

Task 6

Input File(s) Online Data

Output File

Read Data

Push Data

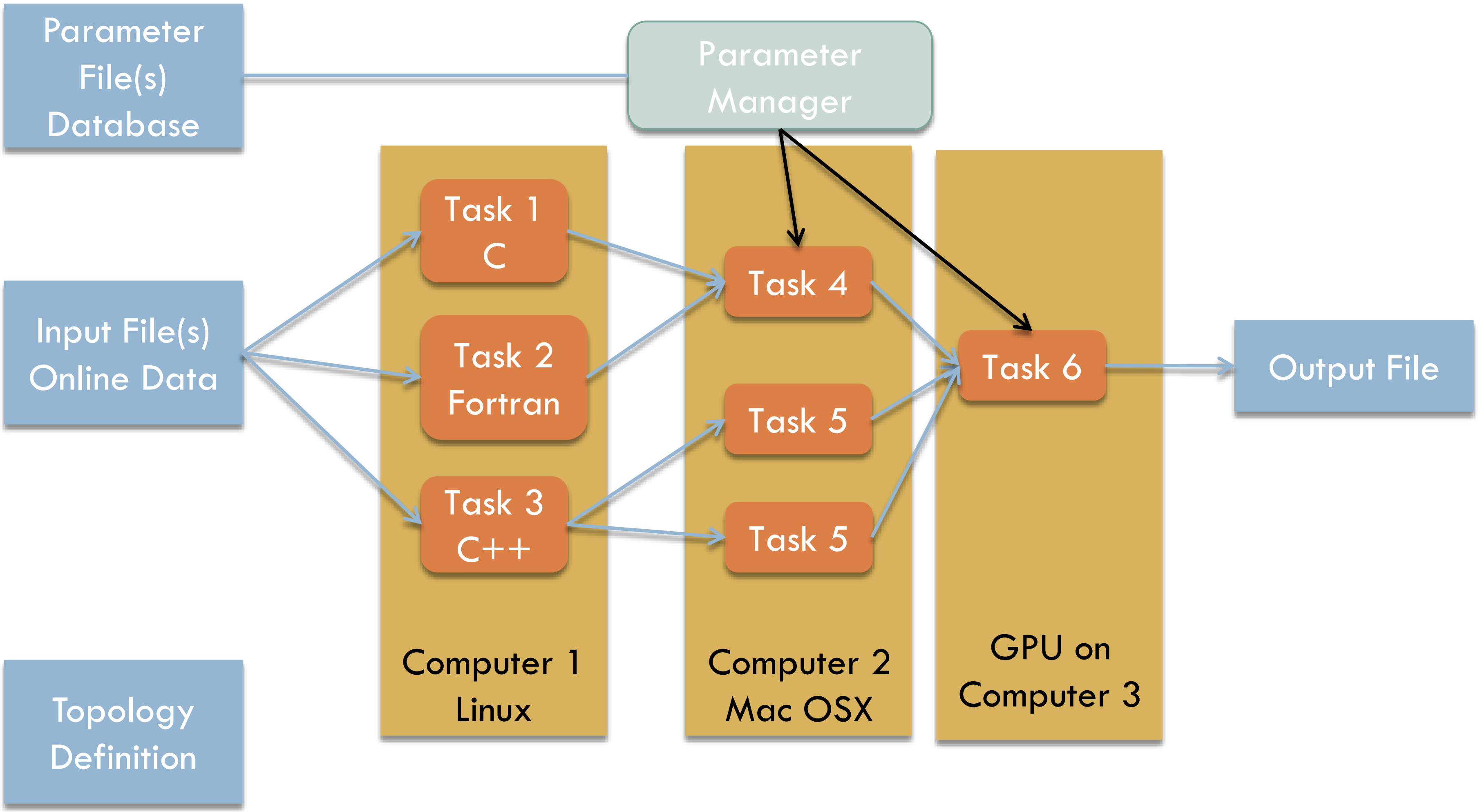
Push Data

Write Data

Transport layer: ZeroMQ or NanoMsg

Topology Definition

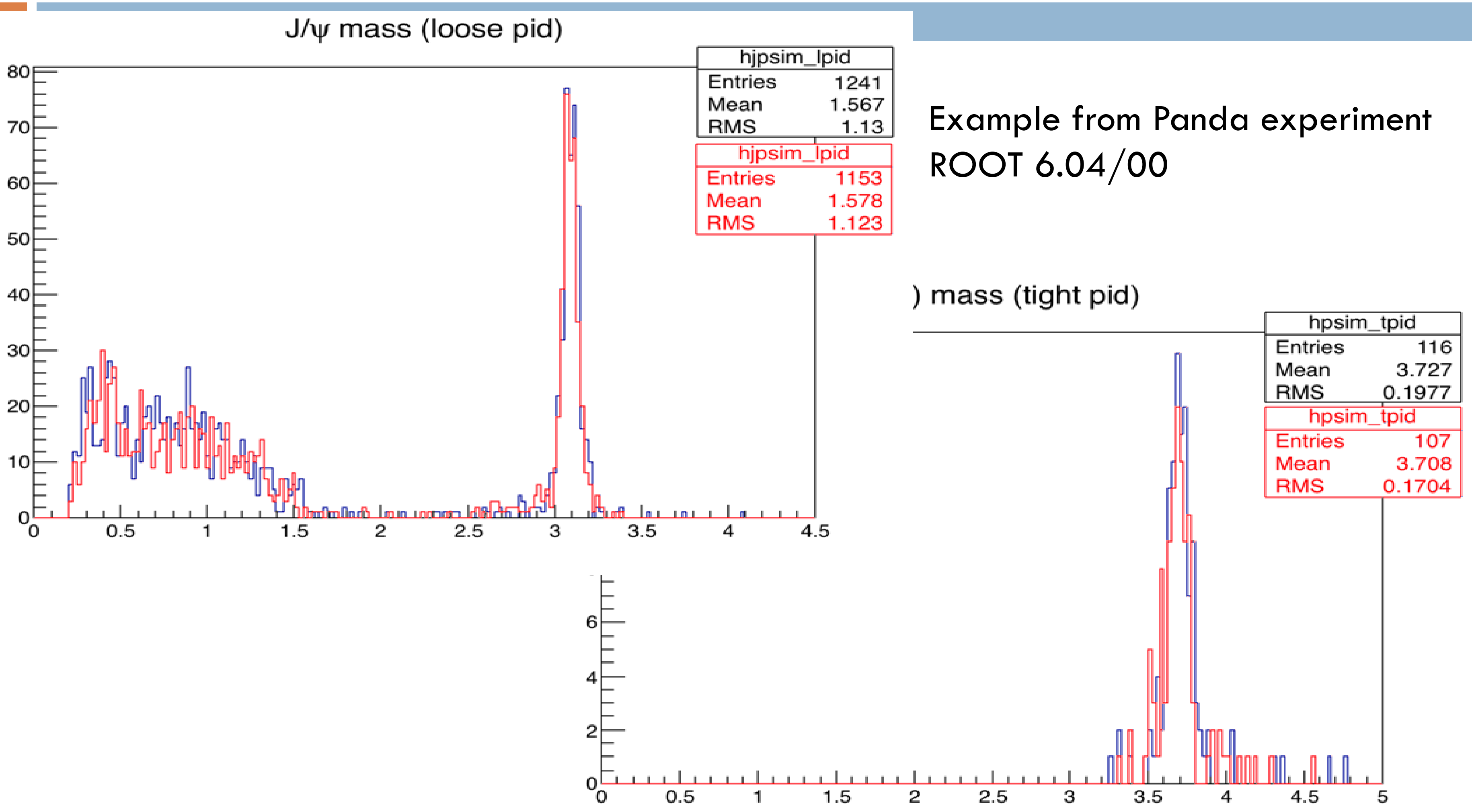
FairMQ


Parameter File(s) Database
Parameter Manager
Input File(s) Online Data
Task 1 C
Task 2 Fortran
Task 3 C++
Task 4
Task 5
Task 5
Task 6
Output File
Computer 1 Linux
Computer 2 Mac OSX
GPU on Computer 3
Topology Definition

Near Future

- C++98 -> C++11
  - Went very smooth
  - Also first experiments went to C++11
- Root5 -> Root6
  - API didn’t change, but some class(es) have disappeared
    - Find out how to implement the features using new class(es)
      - TFitterMinuit
  - Much more work to convert the macros
    - Incorrect C++ in the macros which was accepted by Cint.
    - Sometimes hard to find problems in the code
      - Error messages only states ***error in input_line***

From ROOT5 to ROOT6



Example from Panda experiment
ROOT 6.04/00

Performance

Example from Panda experiment
ROOT 6.04/00

	sim_complete.C		digi_complete.C		reco_complete.C		pid_complete.C	
Geant3	ROOT5	ROOT6	ROOT5	ROOT6	ROOT5	ROOT6	ROOT5	ROOT6
1 event	44s	25s	18s	11s	26s	15s	8s	5s
10 events	47s	28s	20s	13s	30s	17s	12s	7s
100 events	82s	42s	33s	24s	81s	45s	62s	35s
1000 events	464s	214s	147s	140s	642s	309s	616s	286s

Performance

Example from Panda experiment
ROOT 6.04/00

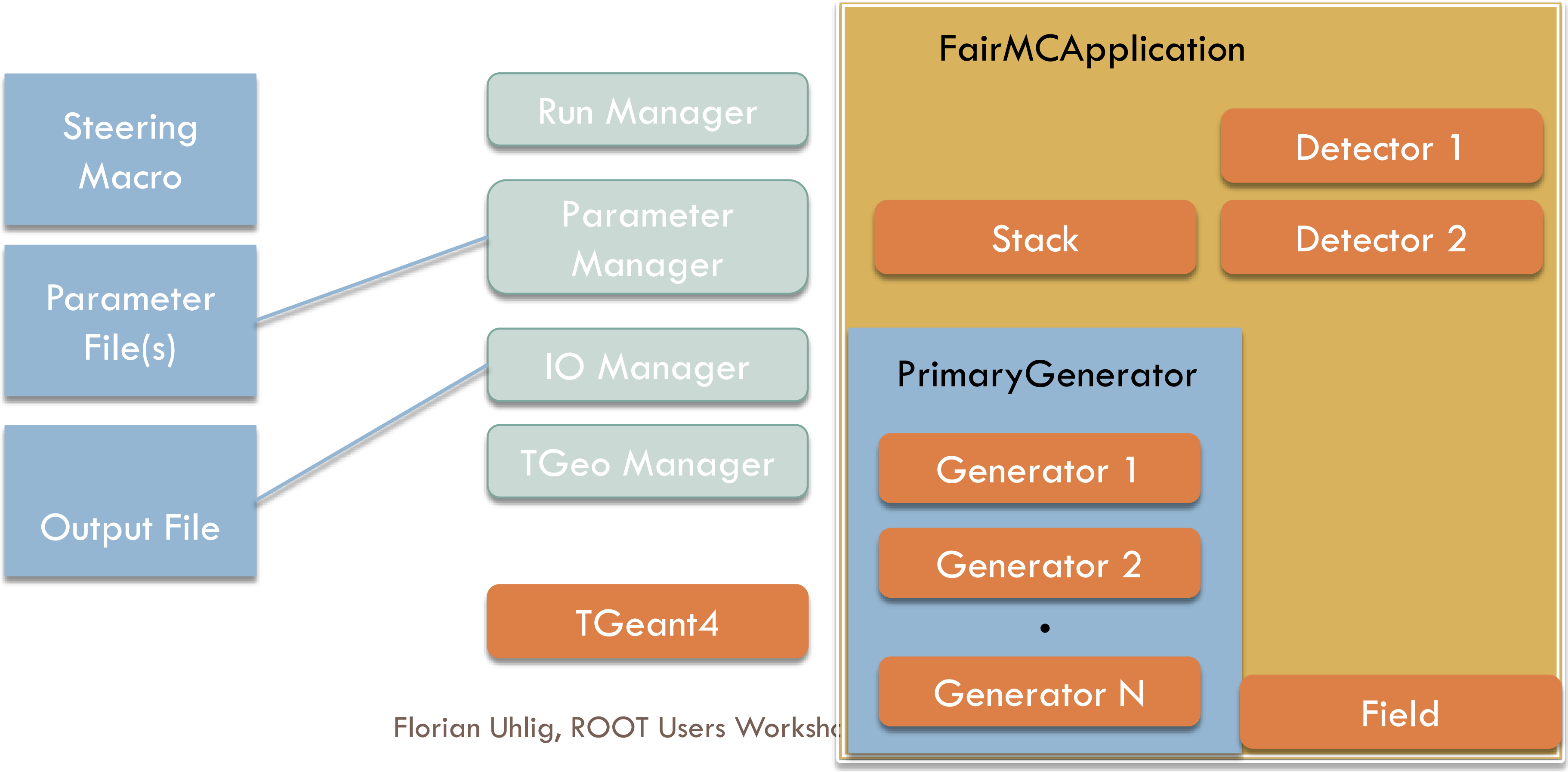
	sim_complete.C		digi_complete.C		reco_com		id_complete.C	
Geant3	ROOT5	ROOT6	ROOT5	ROOT6	R	OT6	ROOT5	ROOT6
1 event	44s	25s	18s	11s	[illegible]	15s	8s	5s
10 events	47s	28s	20s	[illegible]	30s	17s	12s	7s
100 events	82s	42s	[illegible]	24s	81s	45s	62s	35s
1000 events	464s	21	147s	140s	642s	309s	616s	286s

ROOT6 up to 2x faster

Summary and Outlook

- FairRoot meanwhile used by many experiments
- Transition from C++98 to C++11 (done)
  - Still don’t use C++11 features
- Transition from ROOT5 to ROOT6 (done)
  - Wait for experiments
- What comes next
  - C++14 or C++17 ?
  - Root7 ?
- For more information about message queue based system don’t miss the talk of Mohammad Al-Turany (18.09.15, 15.00h)

Current Layout: Simulation



Advantages of FairMQ

- Do not use an event loop
  - No Run Manager is needed
  - Work asynchronously
- Each task is a single independent process
  - If one task fails only restart this task
- Each Task sends its results to the next task in the row
  - No central I/O Manager is needed
  - Different possible connection scenarios
  - No task has to wait, except for input data
- Allow to setup exactly the needed chain

Setup of Topologies

- Setup of small number of processes easily possible with scripts
- Does not scale for large and complex topologies
- Solution: Dynamic Deployment System