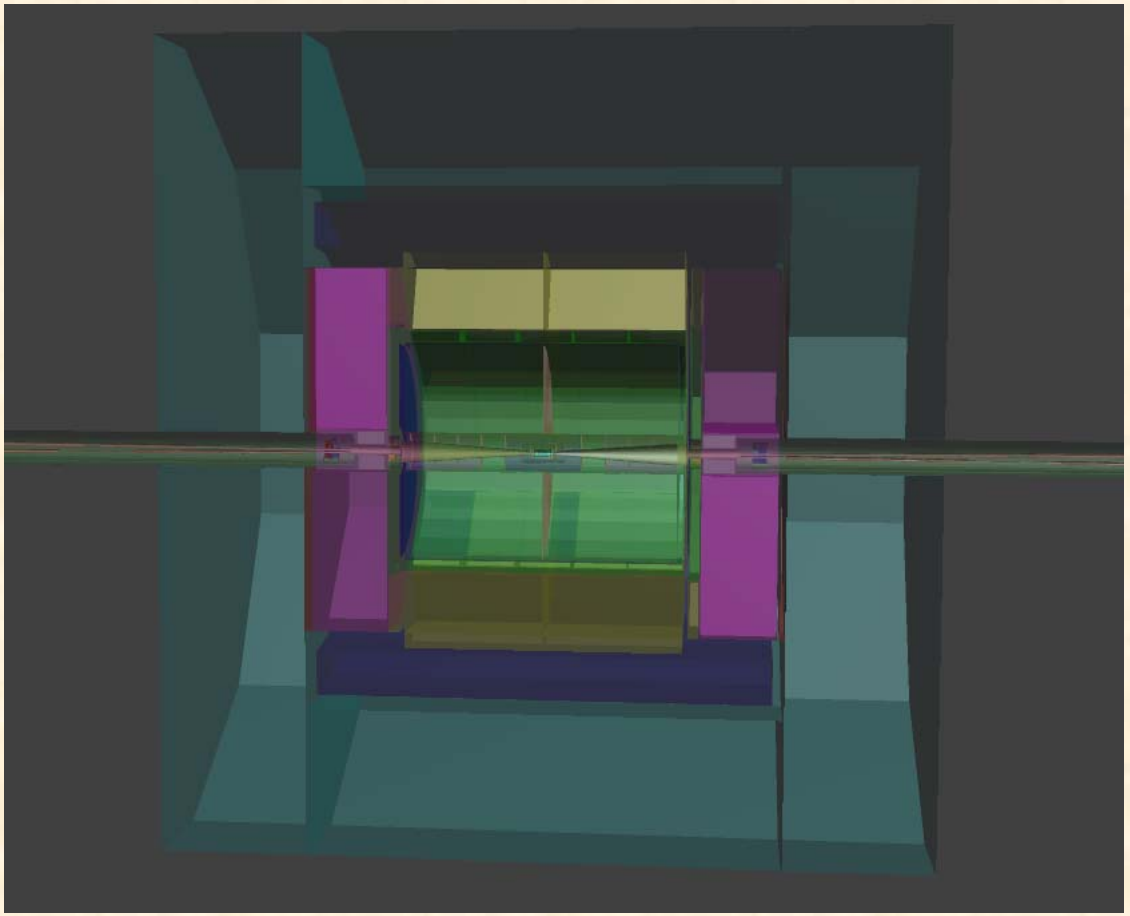
Mokka *Since 1999* *: Status and Plans*



ILD Software and Integration Workshop 2010, DESY
Paulo Mora de Freitas - L.L.R. – Ecole polytechnique

Since January 2010

- Plan to dump automatically the geometry parameters into GEAR file : aborted
  - Replaced by GearCGA or GearTGeo
- Plan to dump automatically, for each new model, the detailed pdf 3D per sub-detector
  - First step done (to dump in VRML)
  - Useful also for EDMS CAD interface
- To test fork strategy : postponed
- Geometry dump in GDML : done.

Current issues - I

- Beam pipe and masks
  - No improvements in design, so neither in simulation
  - Currently the masks should be place inside the Hcal end caps: problem not yet addressed
- Dead material in the central tracking area and between end caps and barrels
  - Work in progress in each sub-detector group to evaluate material, quantity, localization, distribution in space…
  - Not yet enough information to be able to simulate it

Current issues - II

- B Field
  - No progress
  - A 'realistic' field map exists, but we don’t use it except for background simulations, as this is where it is needed (close to the beam pipe in the fwd region)

Current issues - III

- Physics List
  - LOI studies have used LCPhys created by Dennis Wright (SLAC)
  - Calice people have used LHEP, QGSP_BERT or Q6SC_CHIPS depending on the prototype and/or the test beam year
  - A choice has to be done before starting MC mass production (before end 2011?)
  - Should be discussed by the LC community at the last LCWS

News about it ?

New plans - I

- To be able to read HepMC files:
  - Tony Johnson has already a Java reader for HepMC ascii and Les Houches formats, the next is to consider how to store the HepMC info in (extended) LCIO structures

    (Then it could easily be read by Mokka)
  - Mark Terwort, student at DESY, is interested in reading HepMC from Herwig++ into Mokka (no news about progress)

New plans - II

- To improve GEAR API, extending the interface for each sub detector as needed
  - Implies to review the current GEAR implementation in Mokka
- To formalize and propose a clear interface and protocol for scaling sub-detectors, to detect overlaps on the fly when propagating user modifications on a model

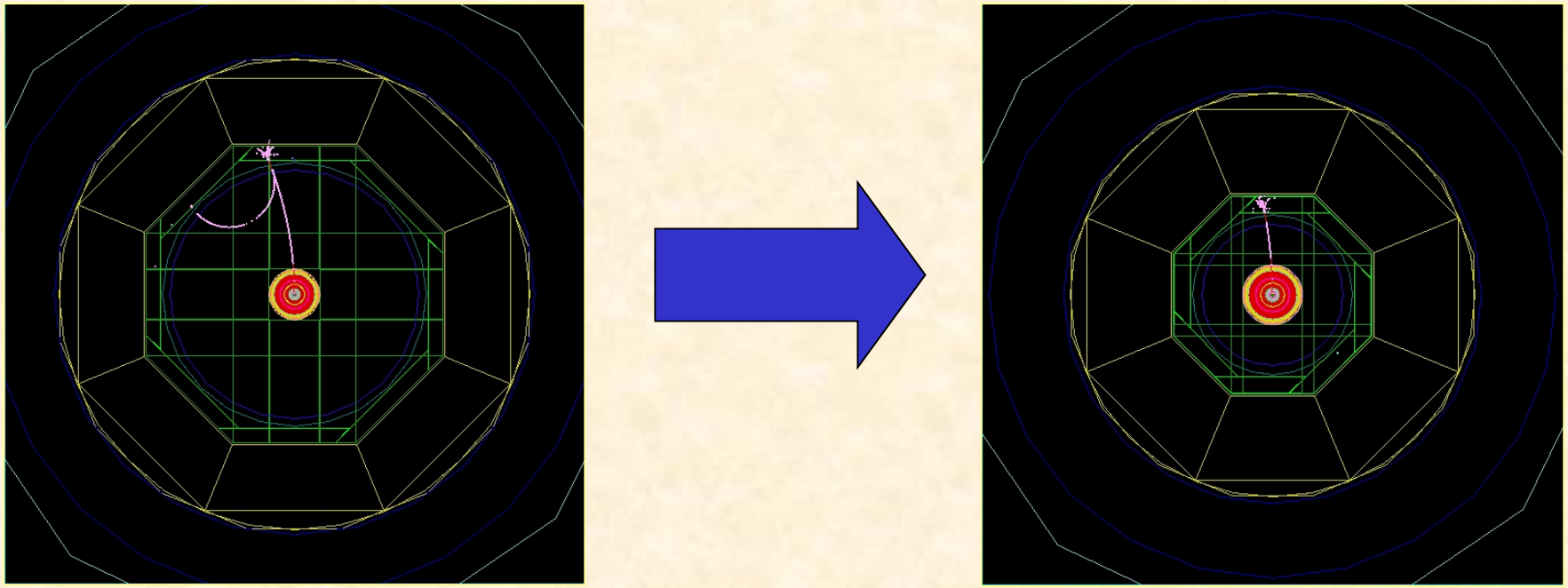
Acknowledgements

- Many thanks for all people which have collaborated with us to develop and maintain Mokka

BACKUPS

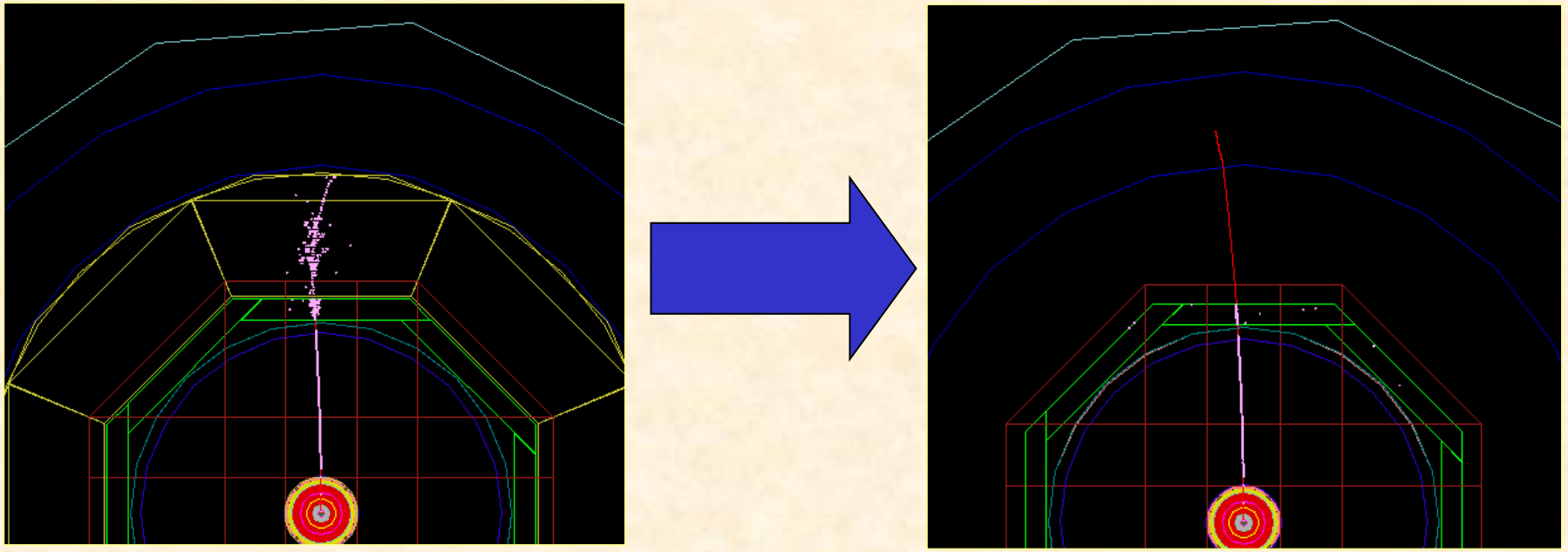
Some good features… (1)

- “Scaling”, the user is able to modify the model's main parameters at launch time, ex :
  - /Mokka/init/globalModelParameter TPC_outer_radius 800



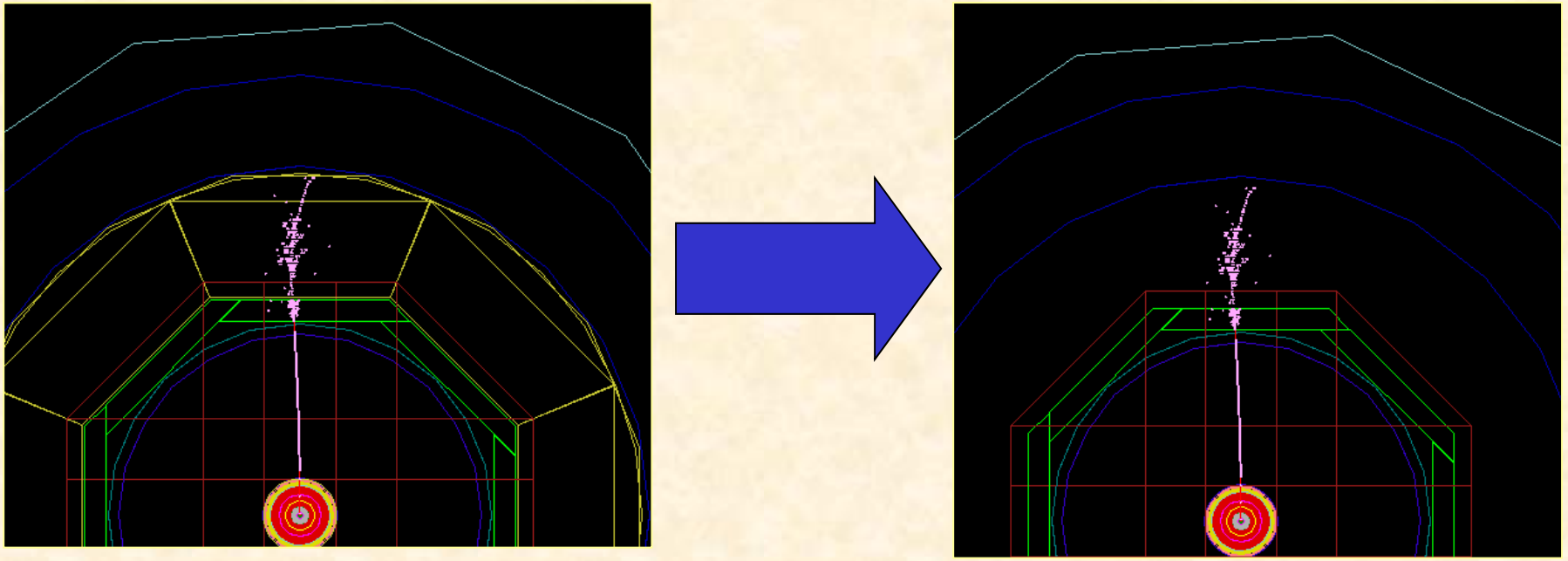
Some good features…(2)

- “Cooking”, the user is able to modify the model ingredients at launch time, ex :
  - /Mokka/init/EditGeometry/rmSubDetector SHcal01



Some good features…(3)

- “Visioning models”, the user is able to interactively modify the model rendering, ex :
  - Idle> /Mokka/Visu/Detector/Visibility hcal false

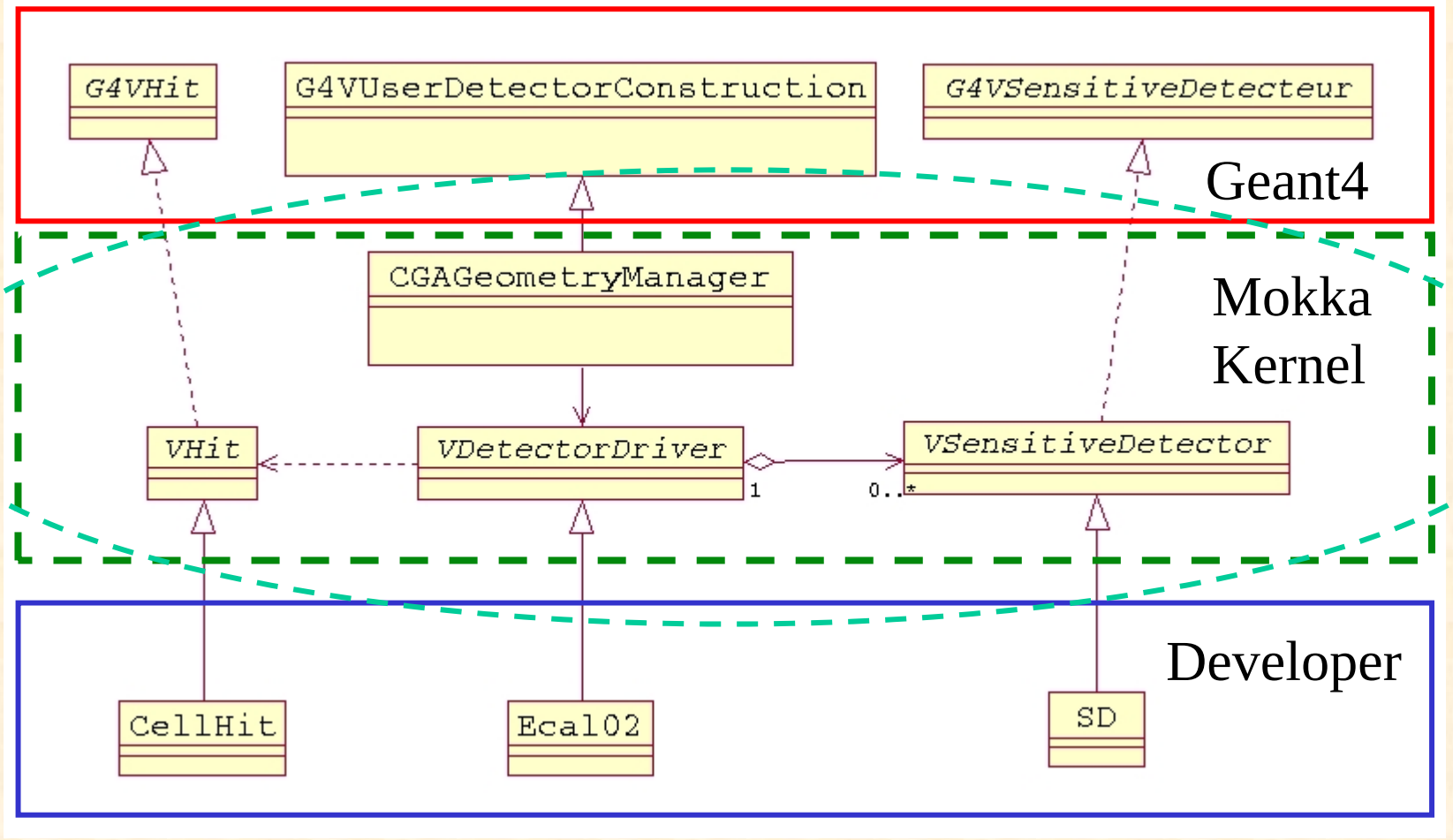


Some good features…(4)

- Plug-in user actions, the possibility to define several run time « user actions » via plug-ins:
  - virtual void *BeginOfRunAction* (const G4Run *)
  - virtual void *EndOfRunAction* (const G4Run *)
  - virtual void *BeginOfEventAction* (const G4Event *)
  - virtual void *EndOfEventAction* (const G4Event *)
  - virtual void *PreUserTrackingAction* (const G4Track *)
  - virtual void *PostUserTrackingAction* (const G4Track *)
  - virtual void *UserSteppingAction* (const G4Step *)

The strategy:

- Improving the Mokka Kernel:

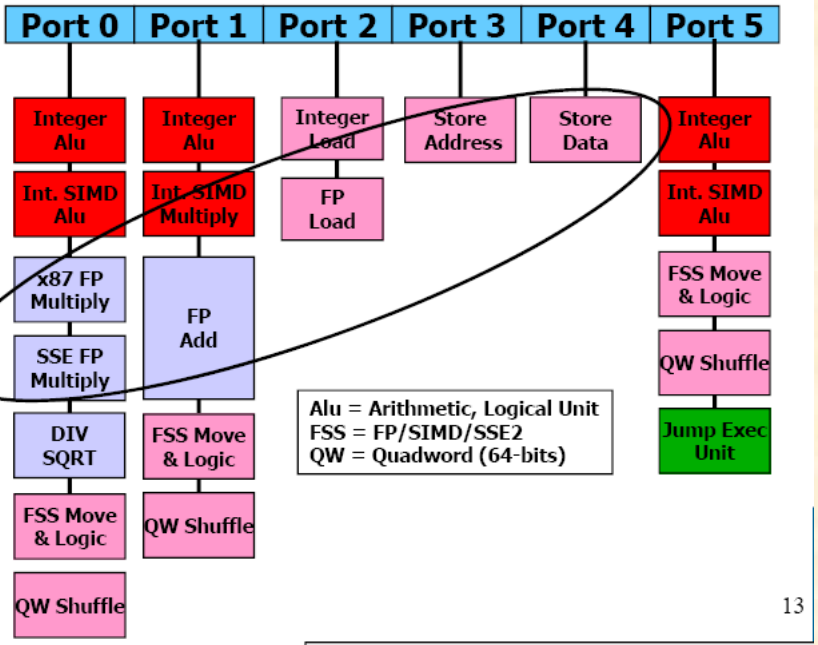


Improving performance (1)

- Optimizing CPU use:

(thanks to Sverre Jarp / Cern)

- **Intel's Core microarchitecture can handle:**
  - Four instructions in parallel:
  - Every cycle
  - Data width of 128 bits



Issue ports in the Core 2 micro-architecture (from Intel Manual No. 248966-016)

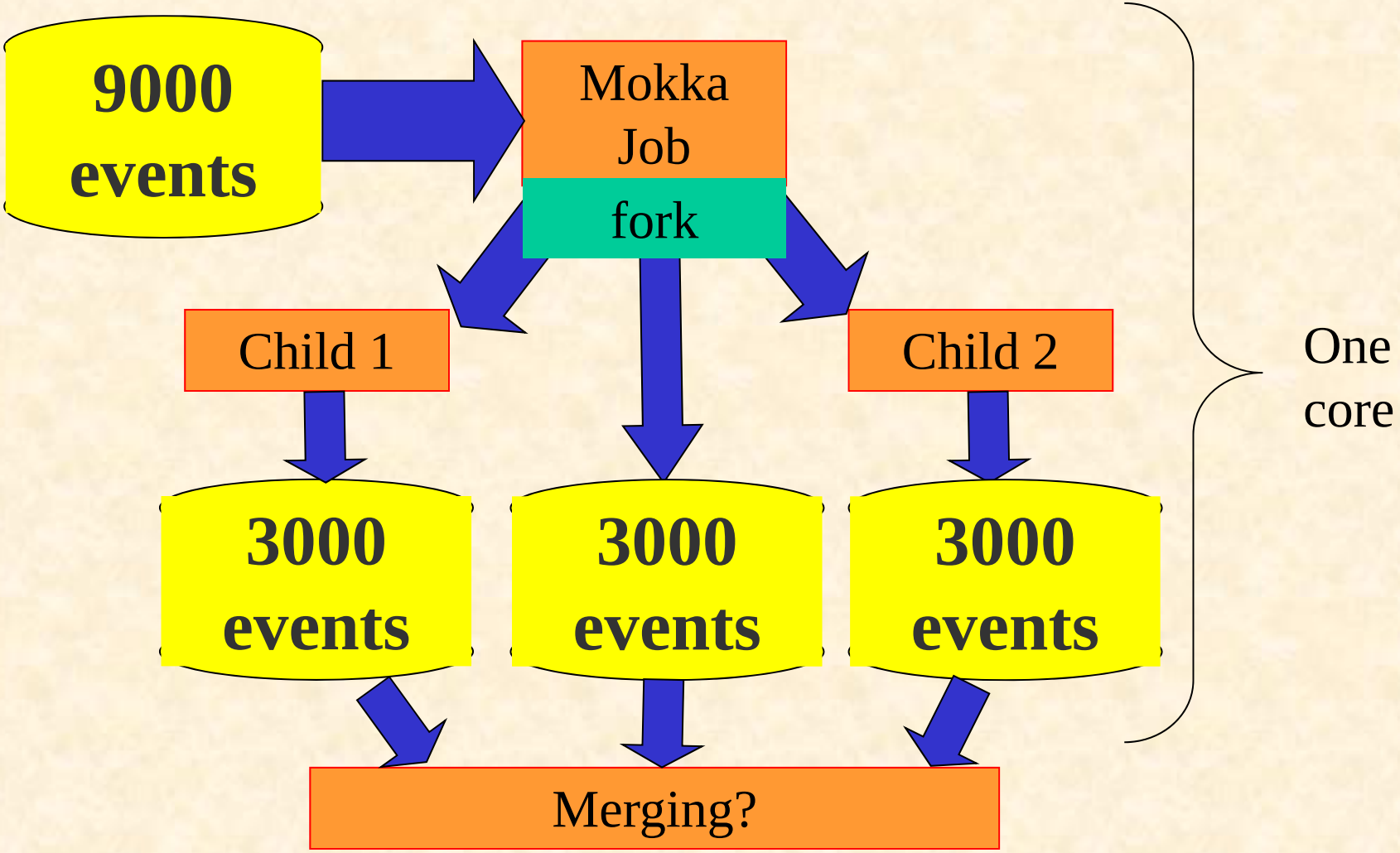
13

Sverre Jarp - CE

"Like having a Ferrari, but using only…

the first gear."

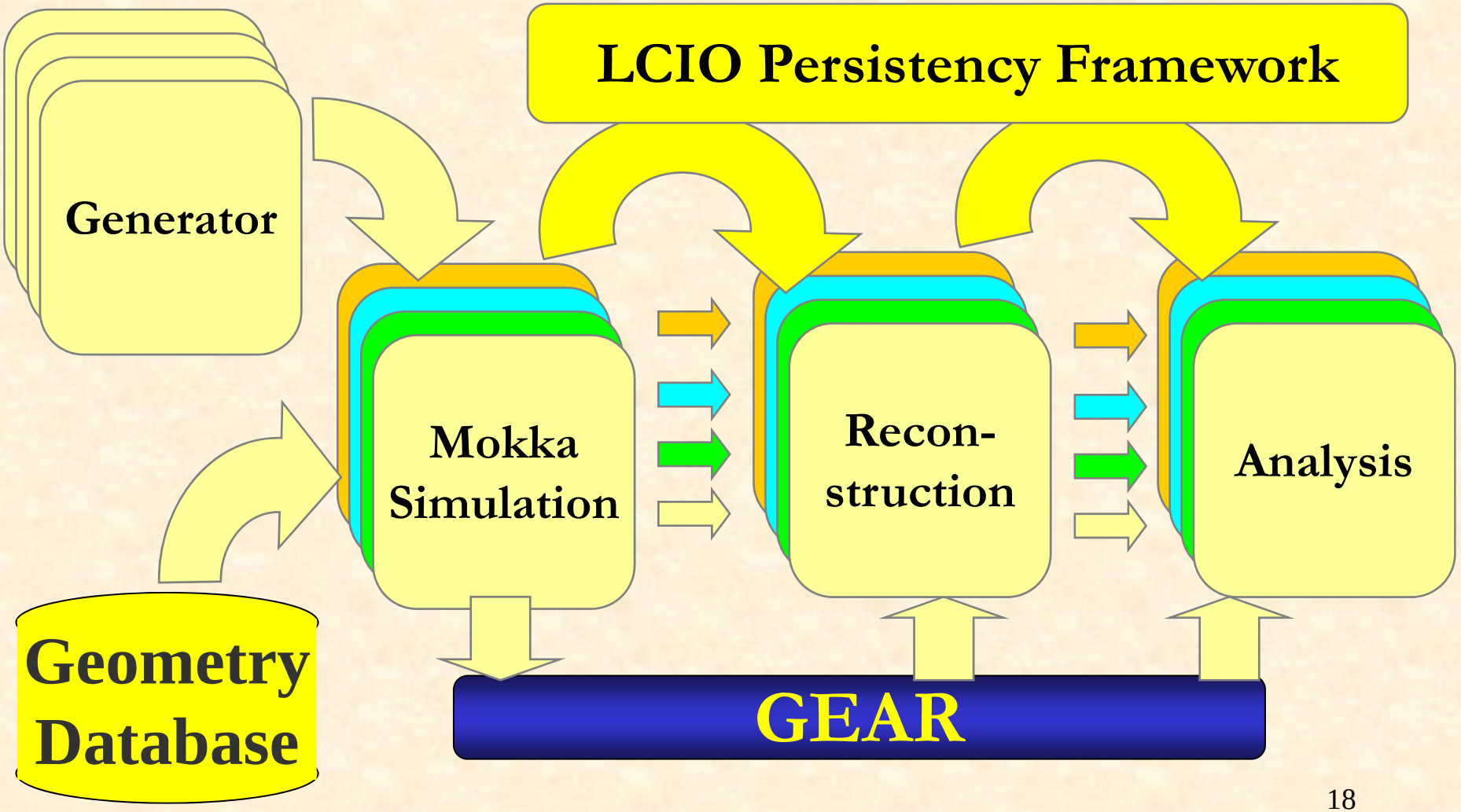
Improving performance (1)



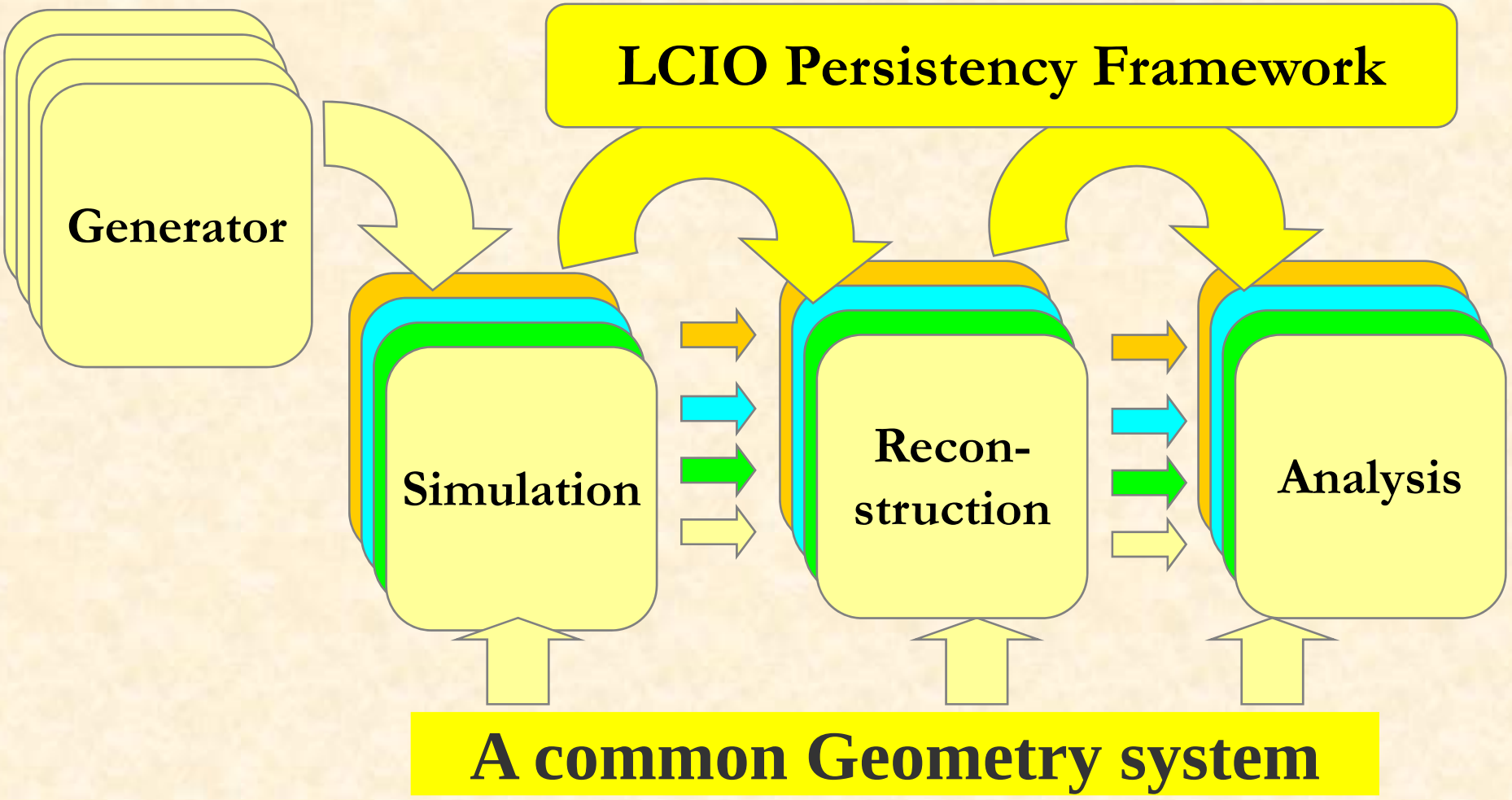
Improving performance (2)

- Switching fast/detailed simulation in Mokka :
  - It’s a standard Geant4 feature
  - It can take the control on a detector region, thanks to the “ghost volumes”
  - Can be generic (ex: Gflash for e-/e+ showers)
  - Could generate hits or reco-like objects directly written into the lcio output file
  - Has to be implemented by specialists, but should be driven by end users depending on their needs
  - Could help providing quick answers, depending on the studies being done

Sharing geometry today



Sharing geometry in the future?



While waiting for the future…

- The model parameters for each sub detector are there :

Base de données models03 - table parameters

requête SQL

		name	description	default_value
Modifier	Effacer	Ecal_Barrel_halfZ	The half Z size of Ecal barrel. It's a master parameter for LDCxx_02yy.	2206.25

Insérer un nouvel enregistrement

…at the geometry data base

While waiting for the future…

– But also there :

…

Building sub_detector SEcal03, geometry db VOID, driver SEcal03:

A scalable LDC Ecal driver without database, just parameters.

Current parameters for the SEcal03 detector :

- Ecal_Alveolus_Air_Gap = 0.5
- Ecal_Barrel_halfZ = 2350
- Ecal_EC_Ring_gap = 10

…

… at the Geometry Manager level, at run time.

While waiting for the future…

– And a few ones are also there :

…

<detector name="TPC" geartype="TPCParameters">

…

<parameter name="tpcInnerRadius" type="double“ value="3.290000000e+02" />

<parameter name="tpcOuterRadius" type="double" value="1.808000000e+03" />

…

…at the GearOutput.xml file.

(But depending on the detector driver code) 

Proposal to improve the geometry sharing (while waiting for the future):

- Exporting the model parameters for each sub detector into the GearOutput.xml file should be:
  - A Mokka Kernel responsibility
  - Done automatically for all parameters & for all sub detectors
- Reconstruction / analysis developers will be able to:
  - Access to all model parameters really used by simulation
  - Reliable information and for free (it's automatic)
  - Introduce in the Mokka DB specific reconstruction parameters, providing defaults in an elegant way (as already done for the Hcal_virtual_cell_size parameter)

Improving Documentation

- Doing it automatically, with Mokka & scripts.
  For example,
  - Gave a new model:
    - To create automatically 3D* pdf files, one per sub detector and fully detailed for deep inspection
    - To put it together the parameter list, description and values per sub detector in the Mokka Web page which describes the new model
  - Gave a new Mokka release:
    - Automatically indexing the release notes per subject and adding it to the Mokka Web site, in a indexed / searchable Web page

(*) concerning pdf 3D, many thanks to Norman Graf!

And don’t forget the user’s requests…

- New or better detector studies ask for new features or improvements. For example:
  - “Low energy particles can stay a long time looping inside the field”. So we should improve the event time structure:
    - to deal correctly with the detector response. Now probably it includes hits that are no more read by the readout system
    - to provide a way to implement and study correctly events with pile-ups (end of the previous event)
- “Frozen models could be built in stand alone (to avoid DB accesses), helping people using grid”
- (see MokkaDB tool)