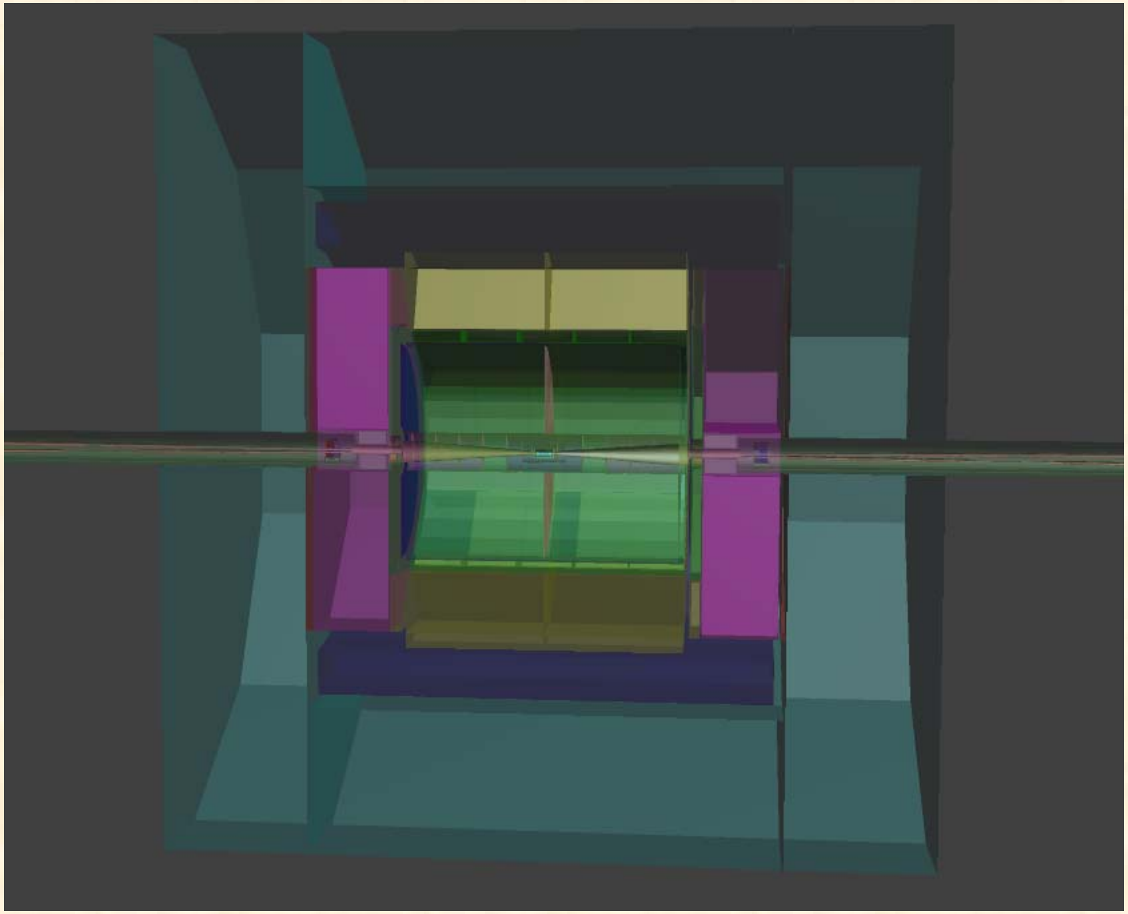
Going ahead with Mokka



International Linear Collider Workshop 2008 - LCWS08 and ILC08

November 16-20, 2008 - University of Illinois at Chicago

Paulo Mora de Freitas - L.L.R. – Ecole polytechnique

Talk plan

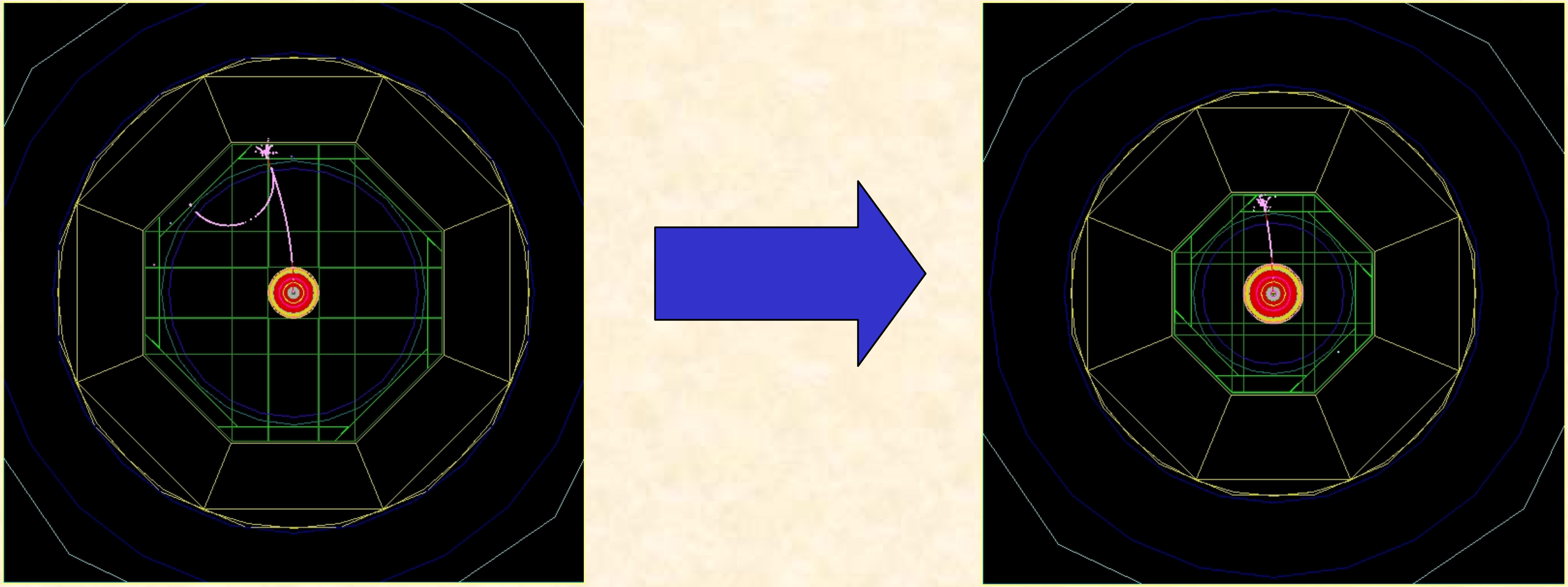
- Going ahead with Mokka:
  - Why
  - What
  - The strategy
  - How improving Mokka
  - Users requests
  - Conclusion

Why

- Mokka has been the framework for the ILD detailed simulations as well for the Calice prototypes.
- A lot of knowledge embedded, thanks to detector’s specialists writing drivers.
- Probably we'll still need to use and develop it for a while (at least while waiting for a new framework wrote from the scratch).

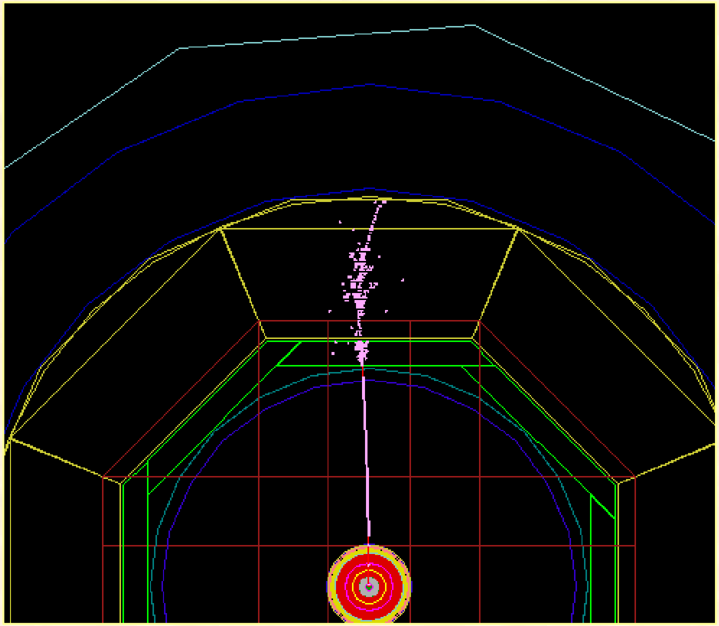
Some good features… (1)

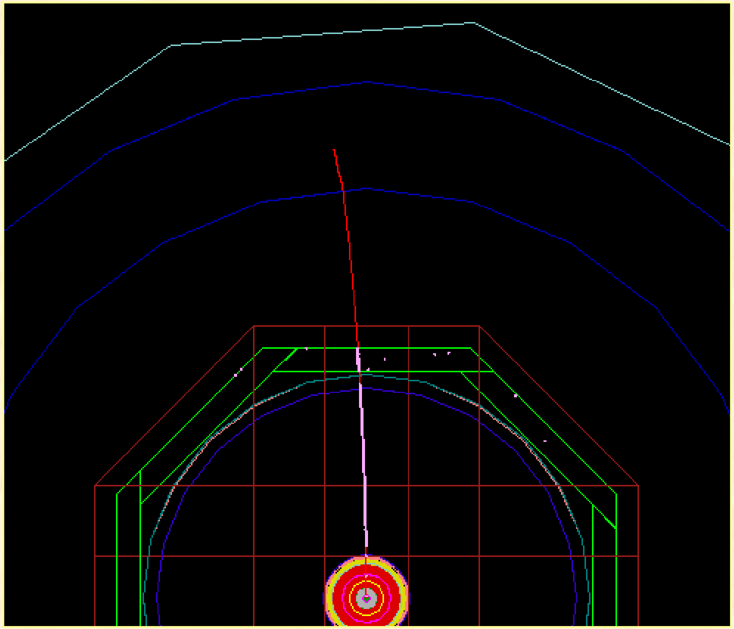
- “Scaling”, the user is able to modify the model's main parameters at launch time, ex :
  - /Mokka/init/globalModelParameter TPC_outer_radius 800



Some good features…(2)

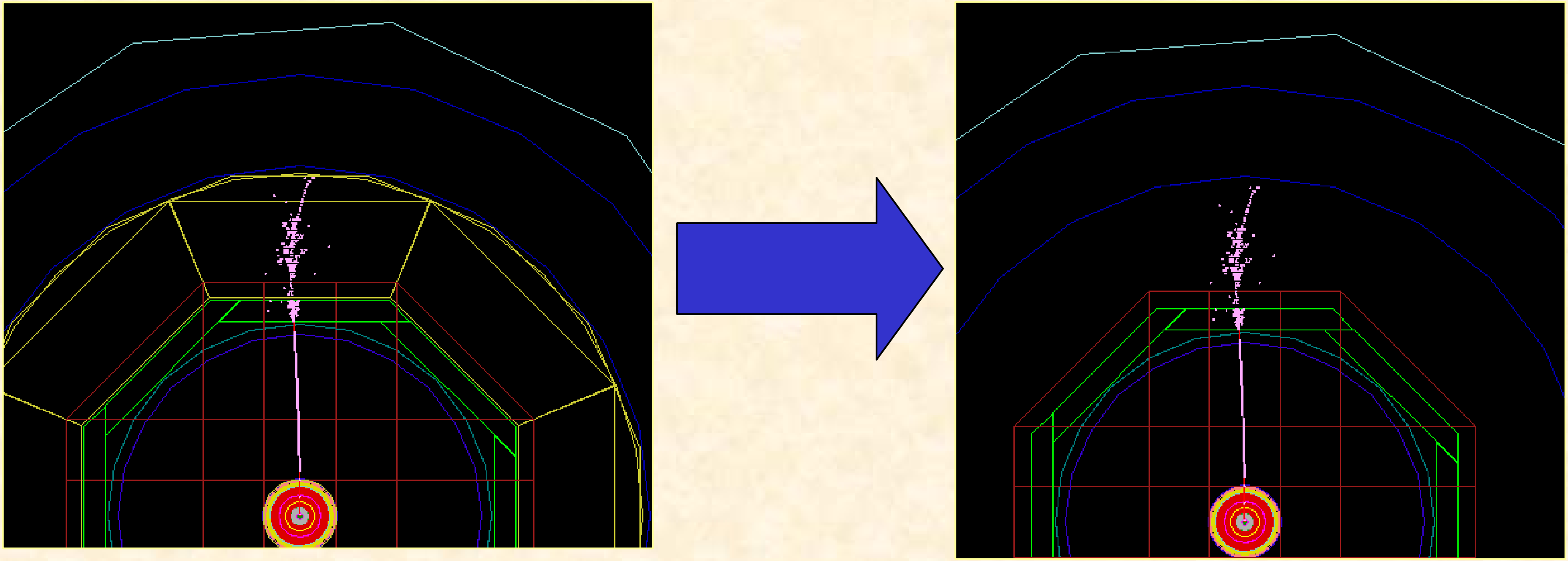
- “Cooking”, the user is able to modify the model ingredients at launch time, ex :
  - /Mokka/init/EditGeometry/rmSubDetector SHcal01





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 *)

But the real raison that people will continue to use Mokka for a while:

- Economics:
  - We don’t have enough people to quickly rewrite from the scratch a new simulation framework!

What is not so good with Mokka

- Performance (anyway it's a Geant4 application…)

- Sharing geometry

- Documentation

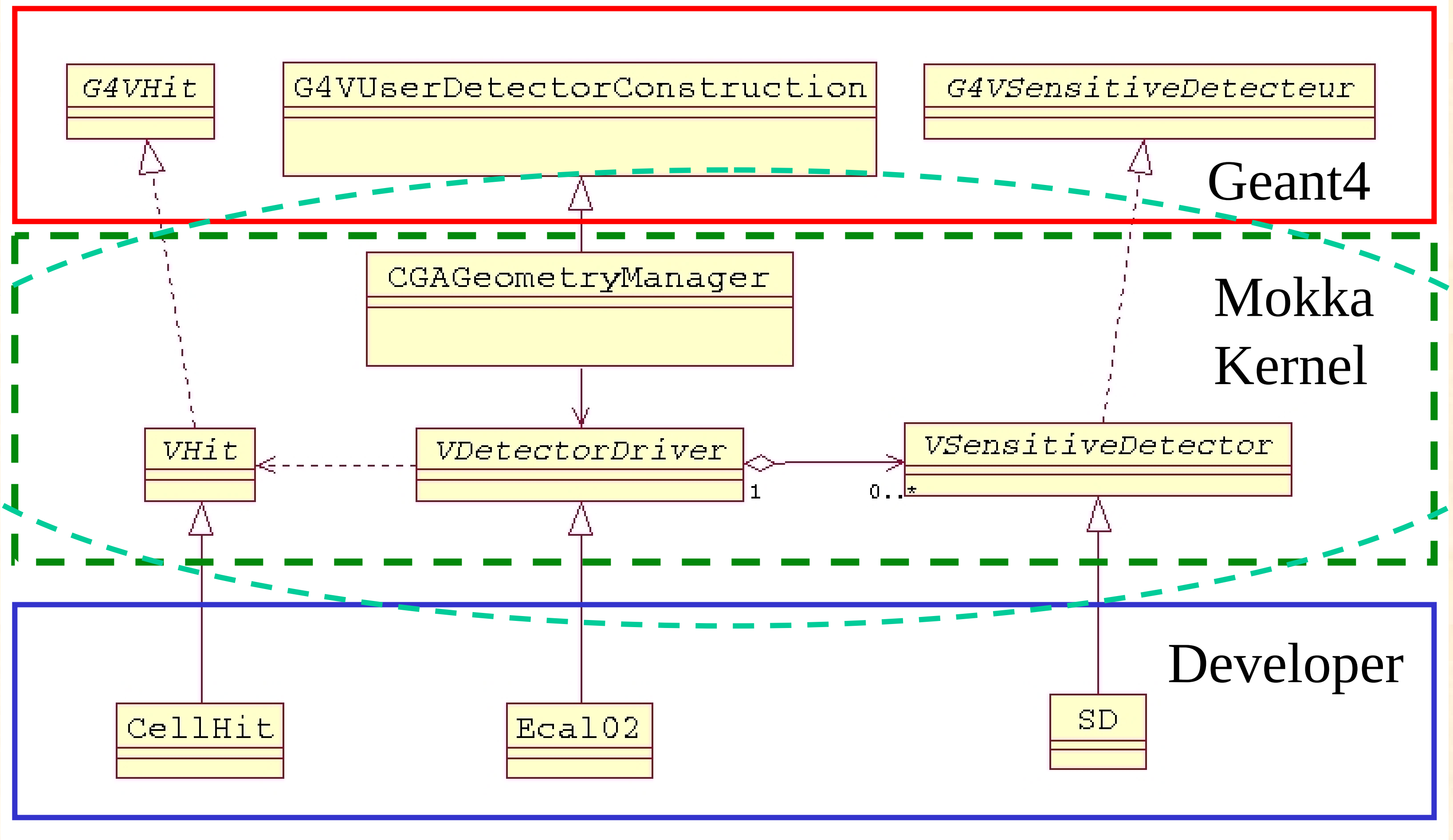
How?

- If we don’t have enough people …

…how to improve Mokka?

The strategy:

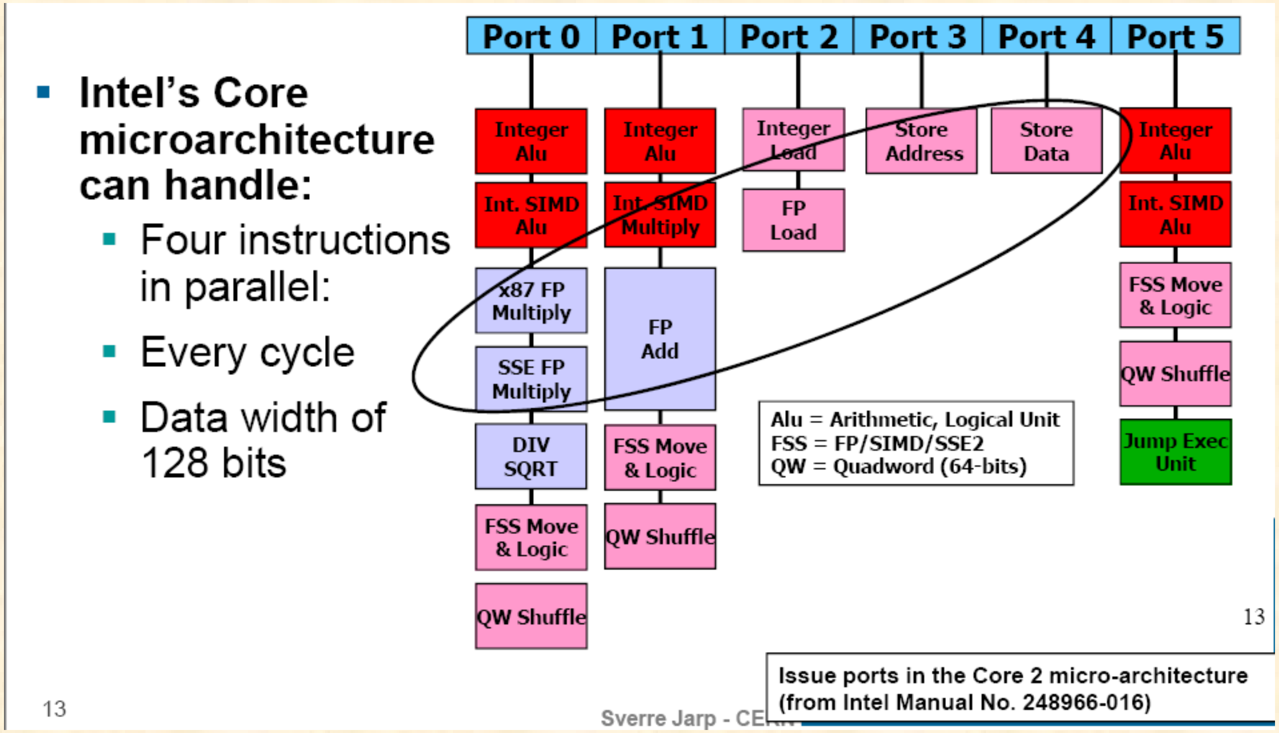
- Improving the Mokka Kernel:



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

Port 0: Integer Alu; Int. SIMD Alu; x87 FP Multiply; SSE FP Multiply; DIV SQRT; FSS Move & Logic; QW Shuffle

Port 1: Integer Alu; Int. SIMD Multiply; FP Add; FSS Move & Logic; QW Shuffle

Port 2: Integer Load; FP Load

Port 3: Store Address

Port 4: Store Data

Port 5: Integer Alu; Int. SIMD Alu; FSS Move & Logic; QW Shuffle; Jump Exec Unit

Alu = Arithmetic, Logical Unit
FSS = FP/SIMD/SSE2
QW = Quadword (64-bits)

13

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

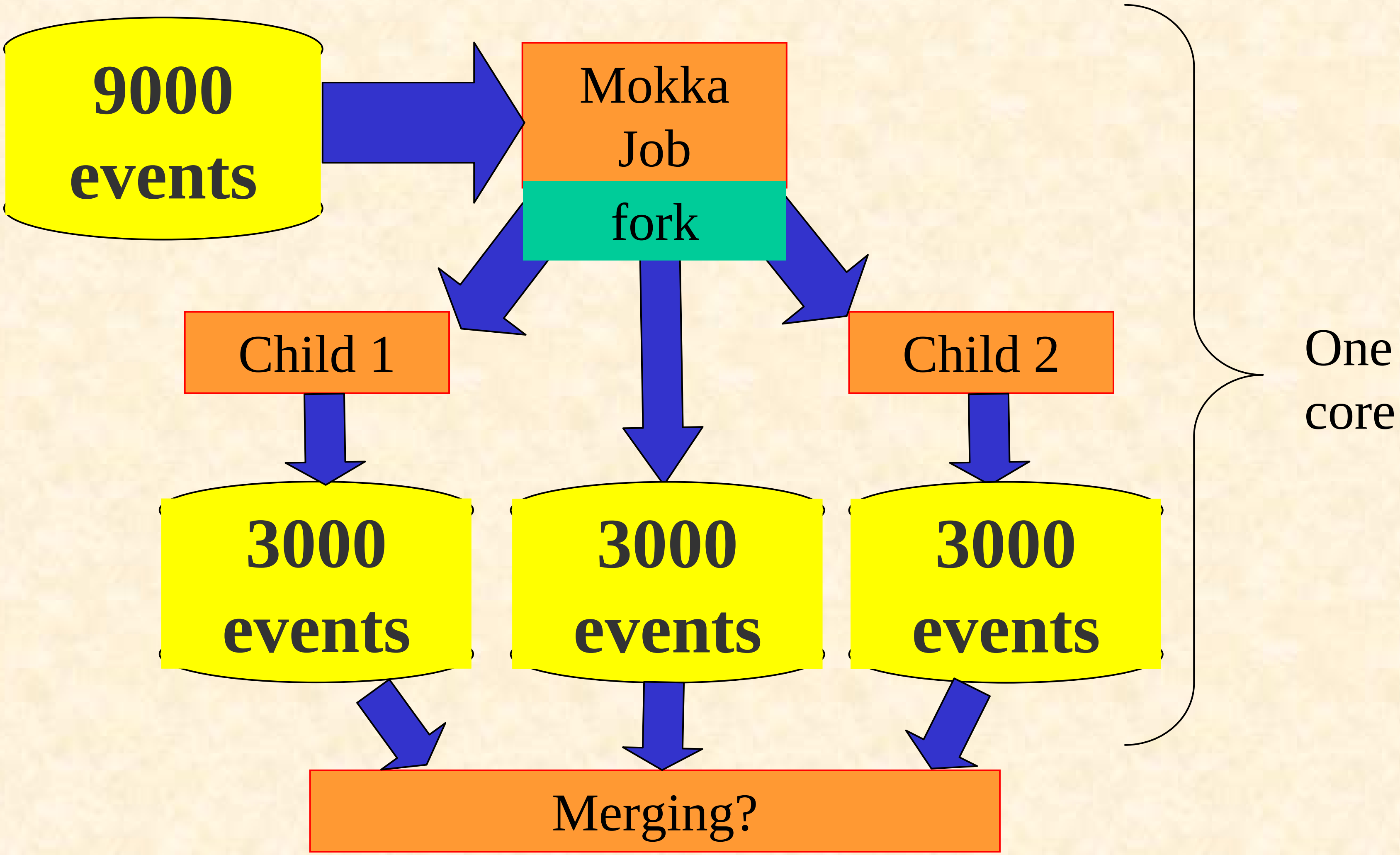
13

Sverre Jarp - CERN

"Like having a Ferrari, but using only…

the first gear."

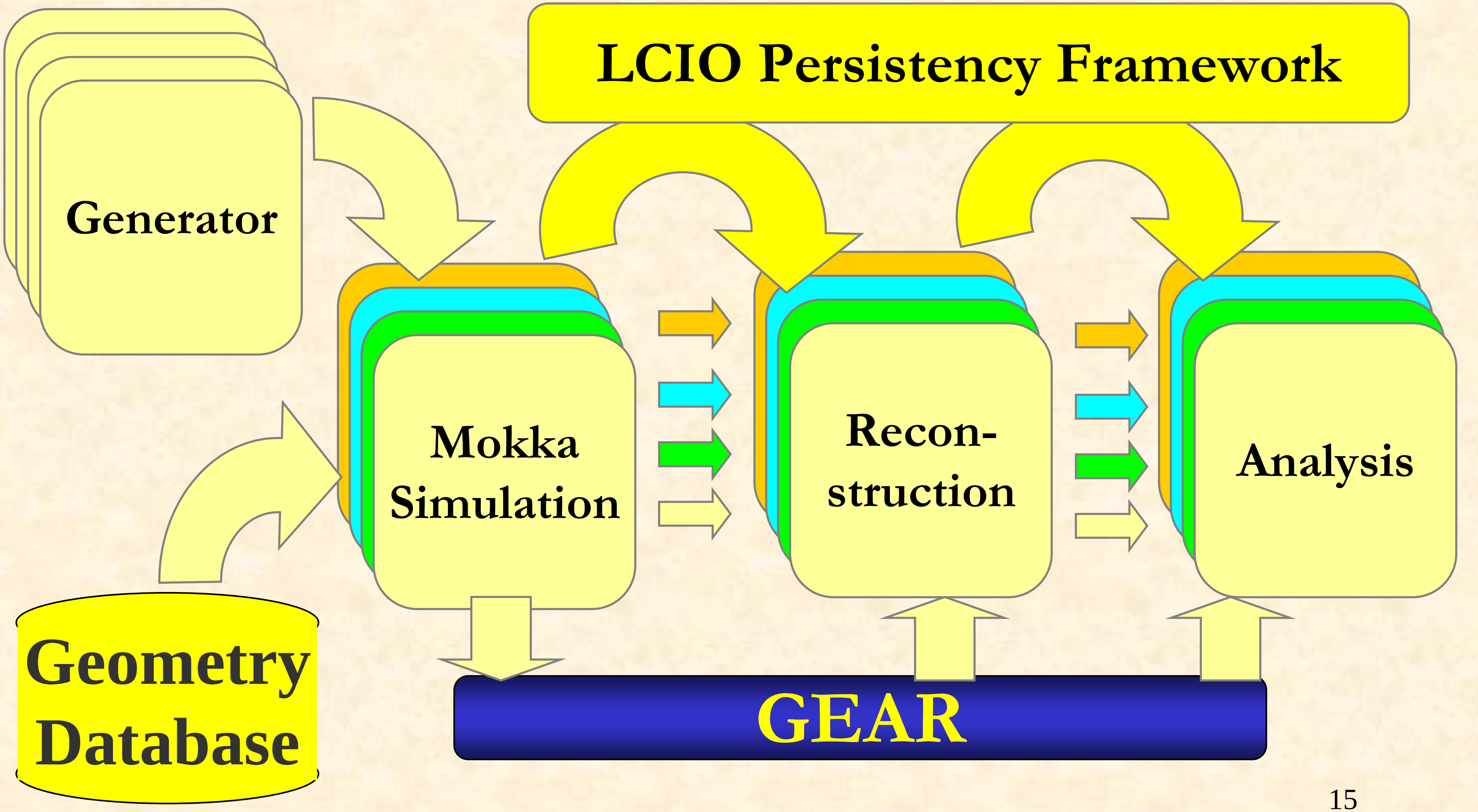
Improving performance (1)



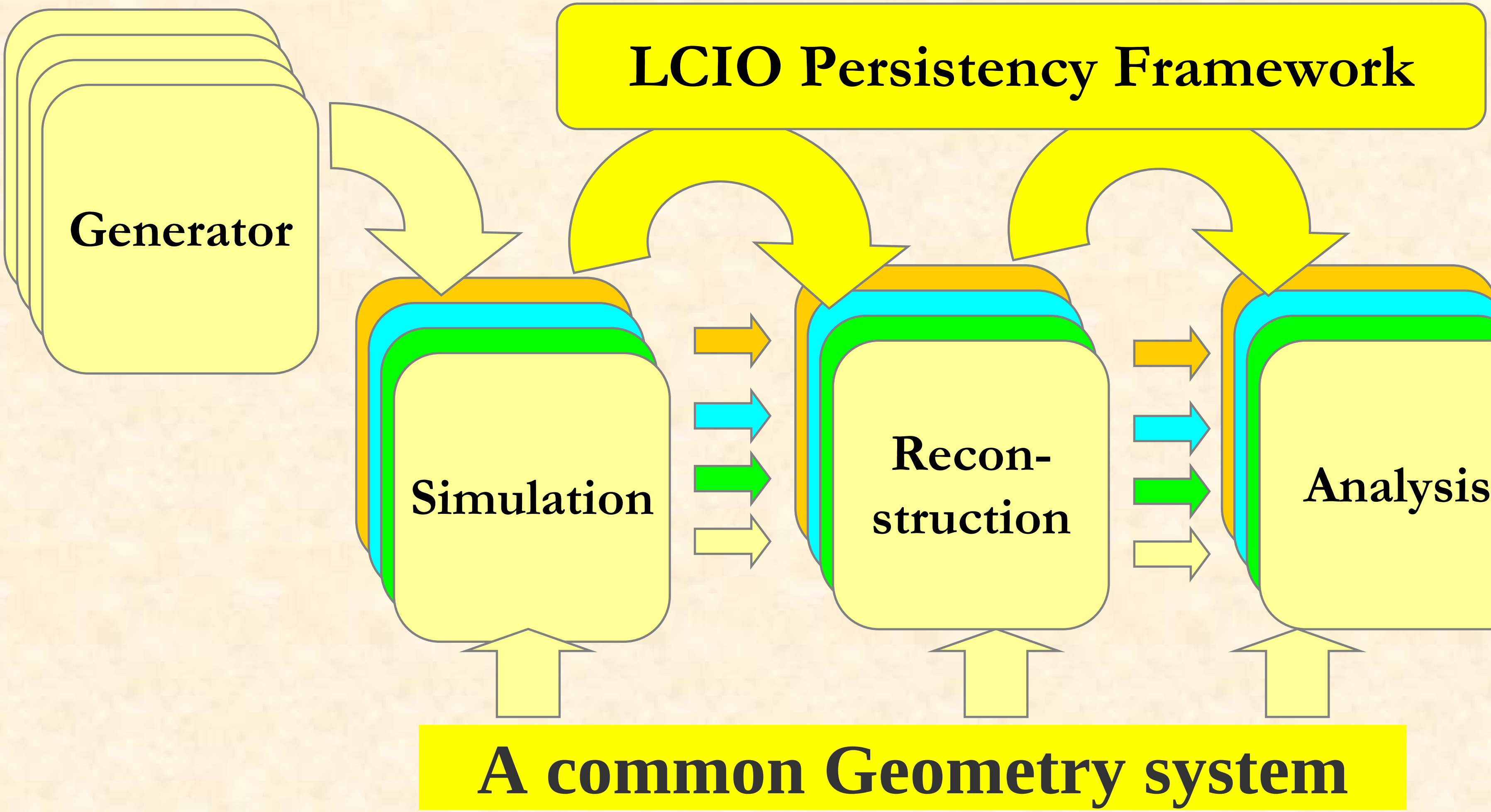
Improving performance (2)

- 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

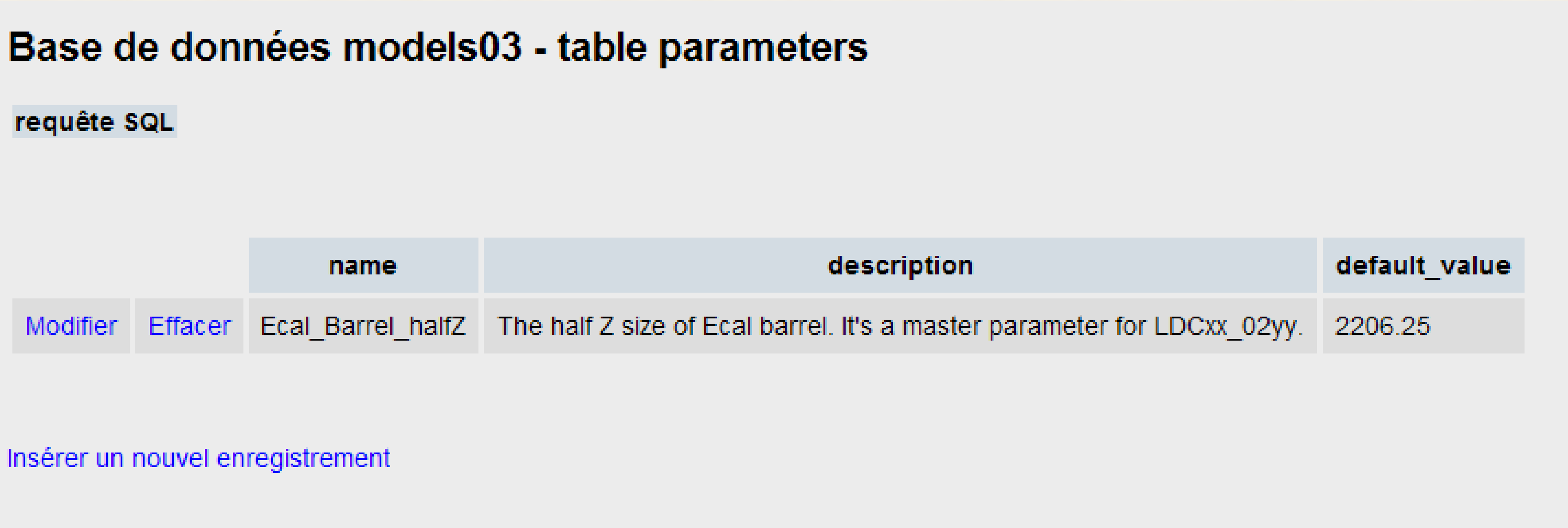


Sharing geometry in the future?



While waiting for the future…

- The model parameters for each sub detector are there :



Base de données models03 - table parameters

		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

…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”

Conclusions

- Probably we’ll continue developing and maintaining Mokka, always with a small team
- Some improvements could be implemented at Kernel level, optimizing the development throughput
- Anyway we have to define priorities, which should steer our efforts

Thanks a lot!

- Anyway many thanks for a lot of people helping to develop and to maintain Mokka

Please, fell free to join… and to enjoy it!

(volunteers are welcomed)