Summary of the Workshop
Discussion on Software Tools

Workshop on the circular Electron Positron Collider
Oxford 15 - 17 April 2019

B. Di Micco - Università degli Studi di Roma Tre e I.N.F.N.
References

Monday - Physics Object I Session

Full Silicon Tracking - Emilia Leogrande
Fast Simulation Tracking - Franco Bedeschi
Machine Learning Tracking - Jean-Roch Vilmant
Particle Identification for Higgs Physics at CEPC - Wei-Ming Yao
Particle Flow at IPNL - Bo Li
Shower simulation with ML - Sofia Vallecorsa

Monday - Physics Object II Session

Calorimeter Reconstruction with ML - Jan Kieseler
Jet Calibration Techniques - Rosa Simoniello
b/c flavour identification with ML techniques - Loukas Gouskos
Jet Flavour Identification - Gang Li
Tau identification - Trevor Vickey
Physics benchmark with jets at CEPC - Manqi Ruan

Tuesday - Software Session

Computing challenges and infrastructure - Xiaomei Zhang
LHC Computing Overview - Alessandro Di Girolamo
The FCCSW - Valentin Volkl
The CEPC Software and Simulation Framework - Tao Lin
Typical requirements for a SW framework

**From LHC experience and developments for HL-LHC**
1) one single framework, easily configurable, where all the needs of data simulation and reconstruction are handled

2) interfaces with Matrix Element generators and parton showers: Whizhard, Pythia8, Herwig++ [can profit from standardizations developed for LHC: Les Houches Event Format, HEPMC and so on];

3) typically needs interfaces to run and to interface with generators: generators will have their own developments, the SW needs to work plug and play with new available versions;

4) fully integrated with detector simulation and geometry [Geant4];

4) need to implement a fast simulation (above all at the Z peak running);

5) more and more developments in fast simulation and object reconstruction are going in the direction to use multivariate classifiers: Neural Networks, BDT’s and so on

6) ROOT is not more enough on the NN side: many tools available of the market, cannot be stuck by ROOT development (or start to think to put effort in ROOT development to implement newest algorithms)

7) ideally: able to run at trigger level (software trigger), avoid double chain developments
   In a competition among projects to be funded, having the best performance is relevant even now (far from the data taking)
The ILC experience

Linear Collider software

Used ILC, CLIC, CEPC, CALICE

ILCSoft

- ILC community has a long tradition of using and developing common software tools
  - started in 2003 with LCIO common EDM -
  - since 2010 common detector description DD4hep

- Used by ILC, CLIC, CEPC and test beams and others

SiD & ILD

- Both concepts use full simulation for detector performance studies and physics analyses
- Detailed simulation models with electronics, gaps and imperfections, cables and services
- Realistic detector and physics performance aided by common software
FCCSW implementation

Generators and Simulations well integrated in Gaudi framework

- pileup overlay
- HepMC and HepEVT Readers
- Pythia8
- many more generators available in LCG-releases!

Simulations

- Delphes - SimDelphes
- PaPas - SimPaPas
- Geant4 - SimG4Full / SimG4Fast

- Initial Idea: Avoid complex inheritance schemes and formulate everything as Plain Old Data
- Use PODIO, a template engine to create everything around PODs (containers, methods, ROOT dictionaries)
Conformal mapping is a geometry transformation that maps circles in the (x,y) plane passing through the origin into straight lines in the (u,v) plane.

\[
\begin{align*}
  u &= x / (x^2 + y^2) \\
  v &= y / (x^2 + y^2)
\end{align*}
\]

Standard tracking in conformally-mapped space:
search for peaks in the angular distributions in (u,v) space.

Pattern recognition

Cellular automation

Seed hit

Seed cell 1

Seed cell 2

Cellular track 1

Cellular track 2

Cellular track 3

Track fitting through Kalman Filter
DD4Hep - KalTest interface to provide surfaces

Tracking code repository

https://github.com/iLCSoft/ConformalTracking
Fast-Tracking for detector performance study

- no path recognition
- simple single-track fitting
- fast and useful to compare detector performances on single (or small occupancy) configurations

Examples (1)
- Initialize geometry and draw it

Examples (2)
- IDEA (baseline)/ CLD comparison

Conclusion
- Simple ROOT based classes to simulate tracking system resolution
  - Easy to change configuration
  - Perfect to compare options
- Validated on full simulation
- Use demonstrated on specific physics process
- Usable to feed fast simulation a realistic covariance
  - Can/Should be included inside DELPHES

Need to go beyond DELPHES
ML for particle tracking

substantial improvement of track performances thanks to NN

seed finding in jets

- Predict tracklets parameters from raw pixels using CNN
- Approaching the maximum performance

ATLAS Preliminary Simulation \( \rho \rightarrow \pi^+\pi^- \)

Track Reconstruction Efficiency

ATLAS Preliminary Simulation, \( s = 13 \text{ TeV}, Z(3 \text{ TeV}) \)
450 < \( p_T^\text{in} \) < 750 GeV, \( p_T^\text{fin} \) > 2 GeV
\( R_{\text{perpendicular}} < 100 \text{ mm}, R_{\text{deca}} > 600 \text{ mm} \)

TIDE
- Light-Jets
- B-Jets
Baseline
- Light-Jets
- B-Jets

CNN output:
- EM-like (blue) / track-like (red)

MC truth:
- EM-like (green) / track-like (red)
Calorimeter Fast Simulation with Adversarial Neural Network

S. Vallecorsa

The model

Condition training on input variables, Custom losses
Auxiliary regression tasks assigned to the discriminator

3D convolutional Generative Adversarial Networks

The data set

CLIC Electromagnetic calorimeter detector
Design 5 mm × 5 mm cell segmentation
25 tungsten absorber layers + silicon sensors
1M single particle samples (e, γ, π) with flat energy spectrum (10–500) GeV
α = ±30° random incident angle
Detector response as 3D images (51x51x25 pixels)
Large dynamic range

M. Pierini et al. NIPS 2017
http://cds.cern.ch/record/2254048# (updated April 2019)

Shower shapes

α = 60°

α = 90°

α = 120°

Time to create an electron shower

<table>
<thead>
<tr>
<th>Method</th>
<th>Machine</th>
<th>Time/Shower (msec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full Simulation (geant4)</td>
<td>Intel Xeon Platinum 8180</td>
<td>17000</td>
</tr>
<tr>
<td>3d GAN (batch size 128)</td>
<td>Intel Xeon Platinum 8160 (TF 1.12)</td>
<td>1</td>
</tr>
</tbody>
</table>
Calorimeter Reconstruction with Machine Learning
J. Kieseler

NN can be used to reconstruct the showers

- Showers are successfully reconstructed
  - Connecting **tracks** are identified
  - EM/hadronic components are **linked**
  - Fractions are separated

Graph NN to cope with non regular geometries
2 π shower identification
Particle FLOW

PFLOW: more than 90% of jet energy from charged particles and photons

clustering
- Arbor [3]: use it as the algorithm for clustering the hits in calorimeter with tree topology.

association
- Track-cluster association: position, direction and energy are considered.
- PID
  - $\gamma, \pi^\pm$, neutral hadron
  - Shower profile, energy deposition and track information are used.

other options: Pandora, almost same performance
Distributed computing needs at CEPC

- **Event size from simulation**
  - Size of signal event: ~500KB/event for Z, ~1MB/event for Higgs
  - Adding the background, likely increase to 5MB~10MB/event for Z and 10MB~20MB/event for Higgs

- **Estimated data rate output to disk:**
  - Higgs/W factory (8 years) with $10^8$ events: 1.5~3PB/year
  - Z factory (2 year) with $10^{11}$~$10^{12}$ events: 0.5~5EB/year

- **Data volume in LHC and HL-LHC**
  - LHC: 50PB/y in 2016
  - HL-LHC: ~600PB/y in 2027

- The challenging part would be in Z factory
  - EB scale, same data volume level as HL-LHC
  - But 12 years after

Core GRID based job scheduling and handling based on DIRAC
(from LHCb, now used by many projects)

**Ready to use from final user**
It is possible to send jobs from any country
Need to work to add more resources from international collaboration to start training the grid system

<table>
<thead>
<tr>
<th>Operation mode</th>
<th>Z factory</th>
<th>WW threshold scan</th>
<th>Higgs factory</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sqrt{s}$ (GeV)</td>
<td>91.2</td>
<td>158 – 172</td>
<td>240</td>
</tr>
<tr>
<td>Running time (years)</td>
<td>2</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>$L$ ($10^{24}$ cm$^{-2}$s$^{-1}$)</td>
<td>17 – 32</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>Integrated Luminosity ($ab^{-1}$)</td>
<td>8 – 16</td>
<td>2.6</td>
<td>5.6</td>
</tr>
<tr>
<td>Higgs yield</td>
<td>-</td>
<td>-</td>
<td>$10^6$</td>
</tr>
<tr>
<td>W yield</td>
<td>-</td>
<td>$10^7$</td>
<td>$10^8$</td>
</tr>
<tr>
<td>Z yield</td>
<td>$10^{11-12}$</td>
<td>$10^8$</td>
<td>$10^8$</td>
</tr>
</tbody>
</table>

- **CEPC simulation for detector design needs ~2K CPU cores and ~2PB each year**
  - Currently no enough funding to meet requirements

- **Distributed computing becomes the main way to collect and organize resources for R&D**
  - Dedicated resources from funding
  - Contributions from collaborators
  - Share IHEP resources from other experiments through HTCondor and IHEPCloud
  - Commercial Cloud, SuperComputing center...
Experience from LHC

Adiabatic evolution doesn’t seem to work anymore, we need to start thinking at cost effective data storage (tapes), full implementation of multi-threading algorithms, distributed storage

Multi Process: single-thread processes, accessing common memory objects (i.e. geometry database)

Full multi-thread

AthenaMP and beyond

- First big wall hit by ATLAS is the memory wall
- Multi-processing with copy on write (Athena MultiProcess, or AthenaMP) is serving ATLAS well in Run2, but we don’t expect this to scale for Run3
- Need a multi-threading solution — genuine memory sharing, with all its known advantages and problems

AthenaMT / Gaudi Hive

- AthenaMT: based on Gaudi Hive: multi-threaded, concurrent extension to Gaudi
- Data Flow driven
  - Algorithms declare their data dependencies
  - Scheduler automatically executes Algorithms as data becomes available.
  - optimal traversal of graph possible if avg. Algorithm runtimes known
- Multi-threaded
  - Algorithms process events in their own thread, from a shared Thread Pool.
- Pipelining: multiple algorithms and events can be executed concurrently
  - some Algorithms are long, and produce data that many others need (eg track fitting), instead of waiting for it to finish, and idling processor, start a new event.
- Algorithm Cloning
  - multiple Instances of the same Algorithm may exist, and be executed concurrently, each with different Event Context.
  - thread safe: one instance, non-concurrent
  - reusable: one or more Instances, in its own thread
  - re-entrant: once instance, executed concurrently by multiple threads
- Thread Safety
  - Only shared Services and re-entrant Algorithms need to be thread safe
  - Algorithms must avoid thread-hostile behaviour
  - global states, etc.
Grid based architecture has some costs: manpower to handle many sites, software development and integration, site updates and availability

- HPCs have been "always" around
  - But in the last years they have made themselves more available to HEP
  - Win-win situation for now:
    - HPC: experience from WLCG in complex workflows, filling the machines
    - WLCG: real workload

- Exaflop machines will be here in 2021.
  - WLCG experiments will have an early access.

Grid framework future development

- Commercial cloud would be a good potential resource for urgent and CPU-intensive tasks with its unlimited resource for CEPC
- Cloud can be well integrated in current distributed computing system and used in an elastic way
  - Cloud resource can be occupied and released in real time according to real CEPC job requirements

HPC federation is in plan to build a “grid” of HPC
- Integrate HTC and HPC resources as a whole
- Preliminary study has been done with GPU
  - With “tag” in DIRAC, GPU and CPU jobs can easily find their proper resources

Commercial Cloud integration
- Needs to build-up a team of people that works on interface implementation with several HPC (no interest for uniformity on their side)
Present wide-used frameworks and CEPC intention by T. Lin

Gaudi
- Developed by LHCb, became CERN standalone project.
- BESIII and Daya Bay experiments used Gaudi.
- Good design, but complicated.
- Requires a dedicated expert team to maintain.

Marlin
- Developed by ILC, used for Reconstruction & Analysis.
- A simple framework based on LCIO.
- Only used in R&D.

<table>
<thead>
<tr>
<th></th>
<th>Gaudi</th>
<th>Marlin</th>
<th>CEPC intended</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maintainer</td>
<td>CERN EP-SFT</td>
<td>DESY IT</td>
<td>CEPC dedicated team</td>
</tr>
<tr>
<td>Experiments</td>
<td>FCC, LHCb, etc.</td>
<td>ILC, CEPC</td>
<td>CEPC</td>
</tr>
<tr>
<td>Functionalities</td>
<td>Complex, Heavy</td>
<td>Simple</td>
<td>Lightweight, Extensible</td>
</tr>
<tr>
<td>Event Data Model</td>
<td>Gaudi::DataObject</td>
<td>LCIO</td>
<td>ROOT</td>
</tr>
<tr>
<td>External Libraries</td>
<td>~35 of LCG (~100 pkgs)</td>
<td>Lightweight (~10 pkgs)</td>
<td>Lightweight (~10 pkgs)</td>
</tr>
<tr>
<td>API &amp; Configuration</td>
<td>C++, Python</td>
<td>C++, XML</td>
<td>C++, Python</td>
</tr>
<tr>
<td>Parallel Computing</td>
<td>GaudiHive, Algorithm-level</td>
<td>Not supported</td>
<td>Multiple levels (node, event, intra-event)</td>
</tr>
</tbody>
</table>
### Planned development

<table>
<thead>
<tr>
<th>Date</th>
<th>Component</th>
<th>Task</th>
<th>Manpower</th>
</tr>
</thead>
<tbody>
<tr>
<td>2018.11</td>
<td>General</td>
<td>Design, requirements discussion</td>
<td>Jiaheng, Xingtao, Ziyan, Tao</td>
</tr>
<tr>
<td></td>
<td>Core+Sim</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2019.04</td>
<td>General</td>
<td>This workshop (prototype proposal)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Core</td>
<td>Event Data Model &amp; ROOT I/O</td>
<td>Jiaheng, Xingtao</td>
</tr>
<tr>
<td>2019.05</td>
<td>Sim</td>
<td>Integration with DD4hep</td>
<td>Ziyan, Tao</td>
</tr>
<tr>
<td>2019.06</td>
<td>Release</td>
<td>Initial prototype (Core+Sim)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>General</td>
<td>Tutorial &amp; Documentation</td>
<td></td>
</tr>
<tr>
<td>2019.07</td>
<td>Rec</td>
<td>Migration of algorithms</td>
<td>Chengdong, Yao</td>
</tr>
<tr>
<td>2019.11</td>
<td></td>
<td>Demonstration with sim+track rec</td>
<td></td>
</tr>
</tbody>
</table>
Discussion on the CEPC sw

1) Gaudi would seem a natural choice: widely developed at LHC, experience already in other chinese experiments: Daya Bay, Bes II, it is the baseline of FCC-software:

**pros:** could profit of future HL-LHC developments and debugging;
could help attracting international collaborators (interest fro software community to work on R&D projects);
**workshop:** planning a software dedicated workshop to join the CepC, FCC, CLIC, LHC communities to discuss a possible collaborative effort (probably mid. June in Italy)

2) on the other side: it is a quite heavy framework, it needs quite a bit of specialized manpower to develop, build and implement analysis algorithms; prefer to have a ROOT based EDM (no need for conversion for analysis)

3) the final software will need anyway to evolve (Gaudi already quite old at this point), full interface with Python (to have ML developments easily implemented);

4) It is felt by the team that the overhead of Gaudi doesn’t pay at this moment, the future software will change, coordinate algorithm developments for physics object and coordination, but keep the framework light and ready to go in production by next year;

5) the advise from LHC/FCC people is that a lightweight approach could quickly become difficult to handle, because not properly structured for the final needs.
Sniper software

SNiPER

- A lightweight framework, developed by IHEP & SDU (~2013)
- Use ROOT as transient and persistent data model.
- Scalable: multi-threading support (~2016)

A simple framework with key Gaudi design features.

- **Flexible workflow.** (Event Filter & Event Mixing)
- **Parallel computing**
  
  Task: similar to Gaudi application manager (event loop), manages algorithms, services and sub-tasks. Easy to be parallelized.

- **No Converters.** Straightforward mapping between transient ROOT objects and persistent data in ROOT files.
- **Thread-safe event data management**

Plan: develop a CEPCSFW prototype based on existing code.
Integration with DD4HEP

- Service providing **unified access to geometry** via DD4hep.
- Simulation algorithms produce sim events into ROOT files.

Notes:
- Invoke
- Data flow

Simulation

- Generator Alg
- DetSim Alg
- Random Service
- Event Data Service
- Geometry Service

DetSim Service

- Digitization Alg

ROOT Files

- ROOT I/O Service

CEPCSW Core

Frame

- DetSim Service

Notes:
- Invoke
- Data flow

Framework

- Geant4Kernel

User Actions

- Geant4 Converter

Generic Detector Description Model

- Geant4 & Geant4

- DDG4 & Geant4

- DD4hep

- Compact description (.xml)

- Detector Constructor (.cxx => .so)
Main points

1) software needs to be modular, scalable, multi-threading

2) efficient for GPU and HPC computing

3) full interfaced to external tools/algorithms like Machine-Learning algorithm (fast developments in the recent years, large gain in both detector simulation and reco)

4) huge benefit from integration collaboration with existing projects
Present status and what we need

After the CDR it is a good time to re-evaluate our software tools
Improve simplicity, flexibility, efficiency and collaborative nature

Simulation Software
Based on standard tools
- Root data format
  - DD4hep
  - Geant4
- New hit-based Fast Simulation
  - FATRAS
    (Fast ATLAS TRAck Simulation)

Reconstruction Software
Considering new tracking tool
- ACTS
  (A Common Tracking Software)
- Porting of PFA tools:
  - Pandora and Arbor
- Developing other algorithms:
  - vertex, long-lived charged particles,
  - particle identification in jets