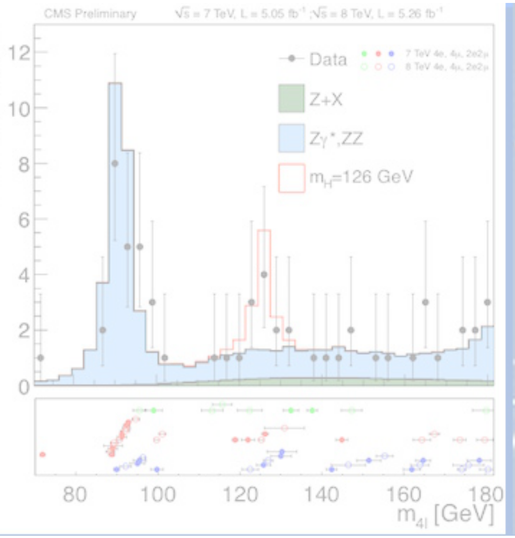

IPCC ROOT

Princeton/Intel Parallel Computing Center

Progress Report

Vassil Vassilev, PhD

16.01.2018

Status 16.01.2018—29.01.2018

- We worked mostly on finalizing our statement of work for 2018

IPCC-ROOT Statement of Work 2018

Component in ROOT	Deliverable	Success Criteria	Period
Infrastructure	**Enable Continuous Performance Integration:** In Y1 we implemented various microbenchmarks which test code scalability (esp with respect to threading and vectorisation). We would like to continue extending them and running them on a nightly basis. Automatizing the process would allow us to find performance regressions. Another direct benefit would be that we can provide more detailed comparisons between compilers, compiler versions, compiler switches, libraries and operating systems. Currently the process is very laborious and takes a lot of developer's time which can be replaced by this automatic infrastructure making it a matter of setting up a configuration matrix.	Run ROOT's benchmarks nightly on Intel hardware	Q1

IPCC-ROOT Statement of Work 2018

Component in ROOT	Deliverable	Success Criteria	Period
Math	**Modernize ROOT's Math packages by integrating clad:** Y1, Q4 delivers clad: a tool to speed up the production of derivatives. RooFit and TMVA are one of the major places where clad can be used. Currently, the only foreseen derivation backend is employing the numerical differentiation. Clad can be implemented as another backend which delivers derivatives.	Enable a clad-based derivative backend	Q2

IPCC-ROOT Statement of Work 2018

Component in ROOT	Deliverable	Success Criteria	Period
I/O and Reflection	**Optimize ROOT's I/O and dictionary format employing C++ Modules**: ROOT's I/O and reflection layers performs an essential role in the overall performance of ROOT. Currently, ROOT uses its C++ interpreter, cling, to learn about memory layout and other important properties of C++ entities in order to perform correct and efficient on-disk serialization or deserialization. Cling, parses source code to understand the object layouts. In many cases the parsing slows down the overall system performance. We can reduce the amounts of parsing by introducing C++ modules. This in turn will reduce the locking times in the reflection layer, making ROOT more robust when used in multithreaded	Enable C++ Modules as a reflection dictionary provider	Q3

IPCC-ROOT Statement of Work 2018

Component in ROOT	Deliverable	Success Criteria	Period
I/O and Reflection	**Optimize ROOT's reflection layer:** In a few places ROOT asks for reflection information eagerly which causes the interpreter to activate locks and reduce the parallel execution. Instead, ROOT's reflection layer should request only the minimal amount of type information lazily. This in turn will reduce the locking times in the reflection layer, making ROOT more robust when used in multithreaded environments.	Reduce ROOT's locking times	Q4

Thank you!