Contribution ID: **358** Type: **Poster**

Optimizing Frameworks' Performance Using C++ Modules-Aware ROOT

Tuesday 10 July 2018 16:40 (20 minutes)

The LLVM community advances its C++ Modules technology providing an io-efficient, on-disk code representation capable of reducing build times and peak memory usage. Significant amount of efforts were invested in teaching ROOT and its toolchain to operate with clang's implementation of the C++ Modules. Currently, C++ Modules files are used by: cling to avoid header re-parsing; rootcling to store io information in a cross-platform way; and ROOT's reflection layer to implement implicit and explicit runtime shared library loading.

ROOT builds and works with C++ Modules in its dictionary system showing promising performance. The conducted work paved our way for optimizing experiments' software stacks to achieve performance even beyond the observed in ROOT. The new technology adds three new requirements. First, every dictionary header needs to be parsable on its own. Secondly, it should not depend on outside macro definitions which can alter its content. Thirdly, all its direct or indirect includes should be also in a C++ Module. This talk shows the status of the C++ Modularization in ROOT and CMSSW. It argues for a bottom-up adoption approach and outlines a set of tools to aid the process. The authors share performance results and implementation experience gained when migrating CMSSW and its dependencies such as HepMC, Geant and boost.

Authors: TAKAHASHI, Yuka (University of Cincinnati (US)); VASILEV, Vasil Georgiev (Princeton University (US)); ISEMANN, Raphael (Chalmers University of Technology (SE))

Presenter: TAKAHASHI, Yuka (University of Cincinnati (US))

Session Classification: Posters

Track Classification: Track 2 –Offline computing