Contribution ID: **105** Type: **oral presentation**

Optimizations in Python-based HEP Analysis

Thursday, September 6, 2007 3:00 PM (20 minutes)

Python does not, as a rule, allow many optimizations, because there are too many things that can change dynamically. However, a lot of HEP analysis work consists of logically immutable blocks of code that are executed many times: looping over events, fitting data samples, making plots. In fact, most parallelization relies on this. There is therefore room for optimizations.

There are many open source tools available to optimize python code. However, all of them stop short of dealing with calls into extension libraries, which is a major part of any Python-based HEP analysis.

The typical extension library in HEP is written in object-oriented C++ code, and used through interface pointers. In the analysis code, these pointers are then used to call specific functionality (e.g. to retrieve data), as well as simply passed around (e.g. to call a fit on selected data). In both cases, the Python part exists mostly for the convenience to the user in wiring the needed functionality together; it does not add functional code that the C++ extension library needs to be aware of.

The natural division in blocks, and the usage of Python as a conduit from C++ to C++, makes HEP analysis code particularly suited to the kind of partial evaluation and specialization techniques used in Psyco (psyco.sourceforge.net). In this paper, I will how this is used to achieve automatic optimizations for HEP libraries bound with PyROOT.

Submitted on behalf of Collaboration (ex, BaBar, ATLAS)

ATLAS

Primary authors: Dr BINET, Sebastien (LBNL); LAVRIJSEN, Wim (LBNL)

Presenter: Dr BINET, Sebastien (LBNL)

Session Classification: Software components, tools and databases

Track Classification: Software components, tools and databases