

# Harnessing multicores: strategies and implementations in ATLAS

*Tuesday 24 March 2009 15:20 (20 minutes)*

Computers are no longer getting faster: instead, they are growing more and more CPUs, each of which is no faster than the previous generation.

This increase in the number of cores evidently calls for more parallelism in HENP software.

If end-users' stand-alone analysis applications are relatively easy to modify, LHC experiments frameworks, being mostly written with a single 'thread' of execution in mind and consequent code bases, are on the other hand more challenging to parallelize. Widespread and inconsiderate changes so close to data taking are out of the equation: we need clear strategies and guidelines to reap the benefits out of the multicore/manycore era while minimizing the code changes.

Exploiting parallelism is usually achieved via a) multithreading or b) multiple processes (or a combination of both) but each option has its own set of tradeoffs in terms of code changes (and code complication) and possible speed-ups.

This paper describes the different strategies the Offline and Online communities from the ATLAS collaboration investigated, which have been implemented and integrated into the Athena framework, and finally how well they perform in terms of speed-ups, memory usage, I/O and code development.

We present the work integrated in AthenaMT to harness the High Level Trigger farms' computing power via multithreading, the needed improvements and modifications applied to Athena/Gaudi in order to raise its thread awareness and the impact on common design patterns to preserve thread safety across release cycles.

Threads sharing the same address space, AthenaMT is the most promising option in terms of speed-ups and memory usage efficiency at the cost of an increased development load for the code writer needing to worry about locks, data races and the like.

AthenaMP leverages the 'fork()' system call and the 'Copy On Write' mechanism through the 'multiprocessing' python module, to free oneself from the usual concurrency problems that stem from sharing state, while still retaining part of the memory usage efficiency at the cost of a diminished flexibility (compared to threading.)

We'll detail the AthenaMP implementation, highlighting the minimized code changes, how they seamlessly blend into the Athena framework and their interplay with I/O and OS resources.

**Authors:** Dr WINKLMEIER, Frank (CERN); Dr BINET, Sebastien (LBNL)

**Co-authors:** Dr VON DER SCHMITT, Hans (Max Planck Institut für Physik); Dr CALAFIURA, Paolo (LBNL); Dr SNYDER, Scott (BNL); Dr WIEDENMANN, Werner (University of Wisconsin)

**Presenter:** Dr BINET, Sebastien (LBNL)

**Session Classification:** Software Components, Tools and Databases

**Track Classification:** Software Components, Tools and Databases