Parallelization of the Event Processing in the AMS Experiment

Vitali Choutko on behalf of the AMS Collaboration
Laboratory For Nuclear Science, MIT, 77 Massachusetts Av., Cambridge MA
E-mail: vitali.choutko@cern.ch

Abstract. The Alpha Magnetic Spectrometer (AMS) is a high energy physics experiment scheduled for a three year mission on board the ISS. The event model for the AMS experiment is presented. The OPENMP based parallelization of the AMS event processing both for the reconstruction and physics analysis is discussed. The performance of the multithreaded applications is shown.

1. Introduction
The Alpha Magnetic Spectrometer (AMS) [1] is a high energy physics experiment scheduled for a three year mission on board the ISS and featured following main components: Superconducting Magnet of 0.8 Tesla magnetic field; TimeOfFlight hodoscope; eight layers silicon tracker; gaseous Transition Radiation Detector; Ring Image Cherenkov Detector and electromagnetic calorimeter.

The physics goals of the AMS are to search for antimatter in the Universe on the level of less than $10^{-9}$, to search for dark matter and to make high statistics measurements of cosmic rays composition as well as $\gamma$ rays.

2. AMS Data Volumes and Data Model
The average AMS to Earth bandwidth transmission is 4 Mbit/s, which results to a about 16 Tbytes of raw data per year. Given the AMS raw event size of 2 kbytes this corresponds to the approximately 8 billion recorded events per year.

The AMS Data Model is based on ROOT package [2]. Each event is represented by ROOT tree object and contains the Header object, 64 bit status word and then arrays (STL vectors) of reconstructed objects. All (cross) references are done via STL vector indexes.

About 60 Tbytes of reconstructed data per year is expected, starting from Fall of 2010. On top of that about 100 Tbytes of simulated data per year of operation will be generated.

A cluster of Intel Xeon series 5500 dual blade servers with 16 (hyper)threads per node is supposed to reconstruct the data.

3. Parallelization Motivation
Main motivations for the parallelization of AMS Event reconstruction software are:

- Shorten the (elapsed) time reconstruction for data runs, allowing faster data quality assessment;
• Shorten the (elapsed) time for (potentially long) calibration jobs;
• Increase productivity by using HyperThreading available in the Intel Xeon 5500 processors;
• Ease the production jobs management.

4. Obtained Results
The OPENMP [3] parallelization tool was chosen as it is allowed the parallelization of the code with minimal changes and without explicit thread programming. The Intel C++/Fortran compilers version 11 were used to compile and bind the code.

4.1. Parallel Program Flowchart
Figure 1 shows the parallelized AMS reconstruction code flowchart. The following main modifications were made to the code:
• serialization of Input file reading by adding critical pragma section;
• serialization of Output file writing by adding an event map in memory. Each thread allows to write the file;
• Adding barrier pragma section while trying to read conditional database;
• serialization of governing the various statistical information by adding appropriate critical pragma sections;
• parallelization of the appropriate static variables by marking them threadprivate.

![Figure 1. The parallelized AMS code flowchart based on OPENMP.](image)

4.2. AMS Reconstruction Code Multithreading Performance
Figure 2 shows the scalability of the event code reconstruction for the Intel DualSocket X5482 (8 processor cores) server. The multithreads introduced inefficiency with respect to the singlethreaded application one turned out to be around 1.6% per thread (see Fig. 3). Results both for jobs without the possibility of thread number reallocation during the job execution and for jobs with such a possibility are shown.
4.3. HyperThreading Multithreading Performance

Figures 4 and 5 show the AMS code performance on the hyperthread enabled processors. The addition performance gain of 45% is seen for the Intel 5500 series processors by comparison the processing rate with two 8 thread jobs with the one 8 thread job.

4.4. Data Runs Processing Example

Figure 6 shows the elapsed reconstruction time distribution for the data runs assuming approximately 1 day of data taking with $65 \times 22.5$ minutes (a quarter of orbit) long data runs,
using the processing farm of 8 dual socket Intel 5500 blade servers. The shortening of the mean elapsed time reconstruction by factor of 8 is seen.

**Figure 6.** The distribution of run elapsed run processing time for sequential and parallel processing.

5. Parallelization of Physics Analysis software

By adding few pragmas (see Appendix) to the main ROOT code the parallel processing of the ROOT chains on the local multi-core machines become possible.

The scheme does not require any merging of the user defined output information (like histograms, etc).

Figure 7 shows the performance of the typical I/O bound parallelized physics analysis application for the given method together with TProofLite [2] based parallelization. One can see that the obtained performance gain between two methods is nearly the same.

**Figure 7.** The AMS physics analysis performance example using different parallelization schemes. Errors are obtained by running multiple samples.
6. Summary

Parallelization of the AMS reconstruction software allows to increase the productivity of the reconstruction program up to 45% using the hyperthreads enabled processors as well as to shorten per run reconstruction time and calibration jobs execution time by factor of 10, as shown on Figures 5 and 6. Parallelization of the ROOT reading package and/or using TProofLite allows to gain productivity by factor 2 to 4 depending of I/O bandwidth.

7. Appendix

The following pragmas were added into the ROOT code:

TDirectory.h:\#pragma omp threadprivate(gDirectory)

AMS Case:
TBranchElement.cxx:\#pragma omp critical (root)
info->ReadBuffer(b, (char**) &fObject, fID)

(More) General Case:

TBranch.cxx:\#pragma omp critical (root)
ReadLeaves(*buf);

References
[1] M. Aguilar et al 2002 Phys. Rep. 366/6 331-404.
[2] R. Brun and F. Rademakers. ROOT User’s Guide (version 5.21) in: http://root.cern.ch.
[3] Barbara Chapman, Gabriele Jost and Ruud van der Pas. Using OPENMP MIT Press, October 2007.