Computing Strategy of the AMS Experiment

V. Choutko¹, A. Egorov¹, A. Eline¹, B. S. Shan²,³

¹ Massachusetts Institute of Technology (MIT), Cambridge, Massachusetts 02139, USA
² Beihang University (BUAA), Beijing, 100191, China
³ Corresponding author
E-mail: Baosong Shan@cern.ch

Abstract. The Alpha Magnetic Spectrometer (AMS) is a high energy physics experiment operating on board of the International Space Station (ISS). The detector was installed on ISS in May 2011 and is expected to continuously take cosmic rays data through Year 2024 and beyond. The computing strategy of the AMS experiment is discussed in the paper, including software design, data processing, data reconstruction and simulation, detector performance evaluation, and data production overview. In particular, parallelization of reconstruction and simulation of AMS data is discussed in detail.

1. Introduction
The Alpha Magnetic Spectrometer (AMS) [1] is a multi-purpose high energy physics experiment operating on board of the International Space Station (ISS). The detector has a large geometrical acceptance of \(0.4 \text{ m}^2 \text{ sr}\) and weight of 6 metric tons. The key elements of the AMS detector are: the permanent magnet, the four planes of time of flight hodoscope, the nine layers 10 \(\mu\)m precision silicon tracker, the gaseous Xe/CO\(_2\) transition radiation detector, the aerogel/NaF ring image Čerenkov detector and the lead/scintillator fibers 17 radiation lengths electromagnetic calorimeter.

The physics goals of AMS are to search for antimatter in the universe on the level of less than \(10^{-9}\), search for dark matter in various physics channels and to perform high statistics measurements of cosmic rays composition.

The AMS detector was launched to ISS on May 16, 2011, with the Space Shuttle Mission STS-134 and installed there on May 19, 2011. Since then, the detector has been steadily collecting the cosmic rays data with average live time of 85%. Up to now, more than 70 billion cosmic ray events have been registered and transmitted to the ground with the average transmission speed of 10 Mbit/sec.

2. AMS Data Processing Overview
Data taken on board of the ISS by AMS consist of raw events, where each event is the collection of the information read out from relevant AMS subdetectors electronics after the latter are triggered by particle(s) crossing the AMS sensitive volumes. Raw events are logically grouped in sequences called runs, each run having a unique 32bit identifier. Events inside one run are numbered in sequence. One run typically includes all events taken during a quarter of ISS Low Earth Orbit movements, i.e. \(\sim 23\) minutes, and contains in average 700,000 events. On orbits which are close to the Earth magnetic poles, the number of events in a run can exceed 2

Content from this work may be used under the terms of the Creative Commons Attribution 3.0 licence. Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.

Published under licence by IOP Publishing Ltd
The average size of raw event is 2 KB. Usually the AMS environmental characteristics are uniform during one run. On top of this data volume additional calibration information, like the AMS electronics parameters, as well as various temperatures, pressures and other health and status data is recorded regularly. This calibration related information corresponds to less than 1% of the total data volume.

Recorded data are packed to fixed length packets, or frames, and transmitted via the TDRS satellites to ground at White Sands, NM, and then relayed to NASA Marshall Space Flight Center (MSFC), AL. At MSFC data are firstly transferred to the AMS relay computers using the UDP protocol and finally to the AMS Science Operation Center (SOC) at CERN, Geneva, Switzerland with bbftp [2] software, the latter using multiple TCP streams between client and server to speed up data transfer. The sustained transferring speed from MSFC to SOC up to 50 Mbit/sec is achieved.

2.1. Data Validation and Preproduction
Data arrived at SOC are de-framed and checked for consistency. Then events are indexed and reassembled into the AMS raw files, one file per run. Resulted files are moved to permanent storage and registered in the AMS database, see Section 2.5.

2.2. AMS Offline Software
A dedicated AMS Offline Software was developed to fulfill the following two main objectives:

- **Data Reconstruction**: Convert the information stored in the raw events to the relevant physical quantities, which include all the necessary parameters of the particle(s) which crossed the detector and are ready for further physics analysis.
- **Data Simulation**: Provide the evaluation of the detector performance using simulated events, taking into account the cosmic ray fluxes, precise AMS geometry and the relevant physics processes.

The software contains more than 400,000 lines of source code, written mainly using C++ programming language.

The AMS reconstructed data format is based on ROOT 5.34 [3] package. Each reconstructed AMS event is represented by the ROOT tree object and contains the event header, with general information of the event (~ 100 bytes), 8 bytes status word and (referenced) arrays (C++ STL vectors) of reconstructed objects like particle(s), track(s), cluster(s), etc., of ~ 8 KB total event size. Reconstructed events are grouped together in event summary files, usually one file per run. For every run the relevant time based information, such as average geomagnetic cutoff, detector live time, ISS position parameters, temperatures, voltages of AMS electronics, etc. is also written to the event summary file. The timing precision of such data is about 100 milliseconds. In addition, the AMS conditional database, which contains the information of the time dependent properties of various AMS subdetectors and electronics, like pedestals, gains and detector alignments, is available through a collection of plain Unix files. The portions of the database which are relevant for a given run are also replicated in the event summary file.

2.3. Data Production
The AMS Data Production, i.e. massive data reconstruction is organized in two stages.

Firstly, freshly arrived data are put into first production. It runs in a fully automated manner and produces the data summary files for the quick detector performance evaluation. First production requires computing power of ~120 CPU cores (including 100% contingency) to cope with the data rate and to ensure nearly real-time processing. Usually the reconstructed data are available for the analysis in two hours after the raw data arrive in SOC. Figure 1 shows the overall first production computing organization.
The data summary files from the first production are used subsequently to produce detector calibrations, which are required for *second production*.

![Diagram](image)

**Figure 1:** First production organization and data processing flow. The available raw data are checked by Request Scheduler and/or by Operator via the web interface and transferred to Production Server as Job Descriptions by DB Server. Production Server starts the jobs execution on available hosts in the Production Farm. The successfully finished jobs meta data are stored in DB Server. In case of job execution failure one more execution of the job on a different host is attempted. The failed rate after second attempt is around 1 per mille. Such jobs are signalled to Operator and the latter takes appropriate action by analyzing the log files of the failed jobs.

Second production uses all the available calibrations, alignments and ancillary data from the ISS as well as monitoring values (temperatures, pressures, and voltages) to produce physics analysis ready dataset. Second production usually runs every 6 months incrementally. However, in case there are significant changes in reconstruction software, a full reproduction may be needed. Second production uses the computing power of both CERN and remote computing centers, shown in Table 1.

| Center                | #Cores | For Production | For Simulation | For Analysis |
|-----------------------|--------|----------------|----------------|--------------|
| CERN, Geneva          | 4000   | 800            | 0              | 3200         |
| CNAF, Bologna         | 1500   | 500            | 500            | 500          |
| JUROPA, Juelich       | 4000   | 2000           | 2000           | 0            |
| IN2P3, Lyon           | 700    | 0              | 700            | 0            |
| Acad. SINICA, Taipei  | 3000   | 1000           | 1000           | 1000         |
| SEU, Nanjing          | 2016   | 0              | 2016           | 0            |
| NLAA, Beijing         | 1024   | 0              | 1024           | 0            |
| **Total**             | 16240  | 4300           | 7240           | 4700         |

**Table 1:** List of AMS computing centers with the activities breakdown.

The latest full reproduction was done in February 2015 for the 3.5 years of raw data.
Production was completed in 2 weeks, with the average data reconstruction rate reaching 100 times the data collecting rate.

This performance was achieved due to parallel event processing in the reconstruction software and a light-weight production platform, which will be described in Chapter 3 and Chapter 5, respectively.

2.4. Monte Carlo Simulation
Monte Carlo simulated events are produced using a dedicated program developed by the AMS collaboration based on the GEANT4 package [4]. This program simulates electromagnetic and hadronic interactions of particles in the material of AMS and generates detector responses. The digitization of the signals is simulated precisely according to the measured characteristics of the electronics. The trigger logic is equally simulated and digitized. The simulated events then undergo the same reconstruction procedure as used for the data.

On top of the usual output data file with the reconstructed events an additional file per simulation job with format equivalent to AMS raw data is produced. These files can be used subsequently to rerun of the simulated data with different AMS offline software versions in line with the data reconstruction if needed.

2.5. Data Management
All the AMS data, including framed data, de-framed data, reconstructed and simulated data are stored at CERN and replicated at AMS remote computing centers. CERN EOS [5] is used as a primary storage, while CASTOR [6] is used to back up all the data. Data on EOS has two replicas, and framed data on CASTOR are backed up with two tape copies. Up to July 2015, AMS has 1.5 PB raw and reconstructed data and 0.8 PB simulated data stored on EOS.

The meta data, including the description of all the raw data, reconstructed data, and simulated data, is designed to facilitate the user access to AMS data. It is stored in the AMS Oracle database. As an example, for raw data, the database provides the following meta data for each file: run time and path of the file, the sequence numbers and timestamps of the first and last events of the run, the status and the total number of events in the file, the file size and the Cyclic Redundancy Code (CRC) checksum of the file. The meta data is also visualized and published via the web interfaces on the collaboration web page.

3. Parallelization of Reconstruction
Parallel event processing [7] in AMS was first introduced in 2009, and the main purpose was to shorten the elapsed time for the reconstruction of data runs and calibration runs, thus to get the physics data soon after the readiness of the raw data. Also, it reduces the host memory requirement, and eases the production job management since less jobs are running concurrently.

OPENMP [8] has been chosen because it allows parallelization of the code without explicit thread programming, by adding specific pragmas. The Intel C++/Fortran compiler version 15 was used to compile and bind the code. Third party frameworks and libraries such as GEANT4, ROOT, CERNLIB [9] were also rebuilt using this compiler version.

Fig. 2 shows the flowchart of AMS parallel event reconstruction. Parallelization is done on the event loop level, using omp parallel pragma. During the reconstruction, I/O operations use omp critical pragma, static variables use omp threadprivate pragma, histogram filling uses omp atomic pragma, and thread synchronization on database read uses omp barrier pragma. All the threads always write to a unique output file, which means no post merging is needed. The file can be written in ordered or unordered modes, controlled by a specific data card. In any case the reconstructed events can be directly accessed using recorded event map containing the relation between the event sequence and file entry numbers.
Fig. 3 shows the scalability of AMS offline reconstruction software running on a dual Intel Xeon E5-2699 processor host, which has 36 physical cores with 2-way hyper-threading. As seen from the figure, parallelization performance of the AMS reconstruction software scales well up to 36 threads (the number of physical cores), and allows to additionally increase the productivity up to 45% using the hyper-threading capability of the processors. The parallelization can shorten per run reconstruction time and calibration jobs execution time by up to factor 50.

AMS software has also been built on Intel Many Integrated Core Architecture (MIC) [10], using the same source code. Fig. 4 shows the scalability of the AMS offline reconstruction software running on a host with Intel Xeon Phi Coprocessor 7120P, which has 61 cores with 4-way hyper-threading. The scalability is similar to Xeon E5-2699 as long as the number of threads has not exceeded the number of physical cores. After 2-way hyper-threading turned on, additional 30% performance boost can be achieved. We hardly see any performance gain by turning on 4-way hyper-threading.

4. Parallelization and Memory Optimization of Simulation

Conventional single thread model of execution of simulation jobs often requires heavy memory load per CPU core, which may prevent effective multi-core hosts usage. This is especially critical for AMS for very high energy or high charge cosmic rays simulation. It was expected, that using the multithreaded capabilities of recently released GEANT4.10.1 [11] package, the amount of per thread memory can be reduced by factor two or more. Another attractive feature of the simulation parallelization is to shorten the simulation testing phase, where different MC models and parameters are being tuned to the data.

4.1. Parallelization Model

To minimize the AMS software changes the OPENMP parallelization platform was chosen as for the reconstruction part of the software. To make it work together with the original GEANT4.10.1 multithreaded model the Thread id, total number of threads and number of cores available are taken from GEANT4 model, omp barrier pragma has been reimplemented using C++ code, as the OPENMP one was found not to work properly. All other OPENMP pragmas were found to work correctly together with the GEANT4 multithreaded model. In this way it became possible to adapt the DPMJET 2.5 [12] model, written in FORTRAN, to work seamlessly in the GEANT4 multithreaded model.

The results of multithreaded performance evaluation, similar to obtained for the
reconstruction software, are shown in Fig. 5 and Fig. 6.

Figure 5: Multithreading simulation running on Intel Xeon E5-2699 processors shows good performance scalability with the number of threads. The line shows the fit with taking into account 0.5% inefficiency per thread introduced by multithreading. In addition, the productivity increases up to 35% by using hyper-threading capability of the processors, see data point at 72 threads.

Figure 6: Multithreading simulation running on Intel Phi 7120P shows good performance scalability with the number of threads. The line shows the fit with taking into account 0.1% inefficiency per thread introduced by multithreading. In addition, the productivity increases up to 35% by using 2-way hyper-threading capability of the coprocessors, see data points at 120 threads and above.

The same simulation software has been successfully ported to Intel Itanium architecture using Intel C++/Fortran 11.1.080 compiler. At NLAA (see Table 1), the simulation jobs with as many as 256 threads were able to run on SGI Altix 4700 supercomputers equipped with Intel Itanium 2
9140 processors. We found that the performance scales well with the number of threads up to 32 per job, and above this number, significant inefficiency was observed.

4.2. Additional Memory Optimization
In the original GEANT4.10.1 model, the memory allocation is always increasing with the number of simulated events, as no garbage collection mechanism exists. We introduced simple garbage collection in the class G4AllocatorPool, and as shown in Fig. 7 the average memory allocation of the same simulation job can be decreased this way by factor two. The performance penalty due to garbage collection is $\sim 2\%$.

Figure 7: The dependence of the allocated memory on the number of simulated events with the original and modified G4AllocatorPool for TeV electrons simulation in the AMS detector using 16 threads. As seen from the figure the garbage collection allows to reduce average allocated memory by factor two.

5. Light-weight Production Platform
The light-weight production platform was designed to automate the processes of reconstruction and simulation production in AMS computing centers. The platform manages all the production stages, including job acquiring, submission, monitoring, validation, transferring, etc. The platform is based on script languages, Perl [13] and Python [14], and the built-in sqlite3 [15] database on Linux operating systems. It can be easily customized, according to different batch management systems, file system storage, and transferring protocols. This platform has been deployed and is running in most of the remote computing centers listed in Table 1.

This platform is designed based on Deterministic Finite Automaton [16], which has three major states according to different status of job pool and job slots, as shown in Fig. 8.

6. Conclusions
The computing strategy of the AMS experiment allowed reliable running of the AMS data reconstruction and simulation during over 4 years of operations of AMS on ISS, resulting in more than 2 PB of reconstructed data and simulated events available for the AMS Collaboration.

Parallelization of reconstruction shows good performance scalability with the number of physical cores and allows for additional 45% performance increase by using the hyper-threading capability of Intel Xeon processors.
Figure 8: Deterministic Finite Automaton based light-weight production platform. State I: there are available job slots and there are jobs in job pool, to continue submitting jobs to available job slots. State II: there are available job slots but no jobs in job pool, to update status, validate finished jobs, and reset failed jobs. State III: there are no available job slots, to keep updating the status of running jobs, and validate if any finished jobs are found.

Parallelization of the simulation shows similar performance scalability with the number of physical cores and allows for additional 35% performance increase by using hyper-threading capability of Intel Xeon processors. In addition, the parallelization together with the optimization on G4AllocatorPool allows to decrease the memory consumption per core by factor five, which makes it possible to effectively use available multi-core hosts for simulation.

With parallelization and the production platform, the data reconstruction rate reached 100 times of the data collection rate.

References
[1] Ting S 2013 Nuclear Physics B-Proceedings Supplements 243 12–24
[2] Farrache G Bbftp URL http://ccweb.in2p3.fr/bbftp
[3] Antcheva I, Ballintijn M, Bellenot B, Biskup M, Brun R, Bunic N, Canal P, Casadei D, Coutet O, Fine V et al. 2011 Computer Physics Communications 182 1384–1385
[4] Allison J, Amako K, Apostolakis J, Araujo H, Dubois P A, Asai M, Barrand G, Capra G, Chauvie S, Chytracek R et al. 2006 Nuclear Science, IEEE Transactions on 53 270–278
[5] Peters A 2011 ACAT conference proceedings
[6] Presti G L, Barring O, Earl A, Rioja R M G, Ponce S, Taurelli G, Waldron D and Dos Santos M C 2007 MSST vol 7 (Citeseer) pp 275–280
[7] Choutko V et al. 2010 Journal of Physics: Conference Series vol 219 (IOP Publishing) p 032032
[8] Dagum L and Menon R 1998 Computational Science & Engineering, IEEE 5 46–55
[9] Shiers J et al. 1996 CN Division, CERN (Geneva)
[10] Duran A and Klemm M 2012 High Performance Computing and Simulation (HPCS), 2012 International Conference on (IEEE) pp 365–366
[11] Cosmo G, Collaboration G et al. 2014 Journal of Physics: Conference Series vol 513 (IOP Publishing) p 022005
[12] Roesler S, Engel R and Ranft J 2001 Advanced Monte Carlo for Radiation Physics, Particle Transport Simulation and Applications (Springer) pp 1033–1038
[13] Wall L et al. 1994 The perl programming language
[14] Van Rossum G et al. 2007 USENIX Annual Technical Conference vol 41
[15] Owens M and Allen G 2010 SQLite (Springer)
[16] Cohen D I and Chibnik M 1991 Introduction to computer theory vol 2 (Wiley)