PARALLEL COMPUTING STUDIES FOR THE ALIGNMENT OF THE ATLAS SILICON TRACKER

M. Karagöz Ünel*, P. Brückman de Renström, the University of Oxford, Oxford, OX1 3RH, UK
K. Bernardet, Centre de Physique des Particules de Marseille, France
A. Hicheur, Rutherford Appleton Laboratory, Didcot, UK

Abstract

The ATLAS Inner Detector consists of about 6000 modules in its Silicon Tracker. Therefore, the offline global fit algorithm for alignment has to deal with solving a problem of up to 35000 degrees of freedom. 32-bit single-CPU platforms were foreseen to be unable to handle such large-size problems. The proposed solution is to utilize a parallel cluster with a 64-bit architecture. We performed the initial studies on performance of such a system, compared with earlier predictions, obtained the first promising results on parallel computing for the ATLAS tracker alignment. After a brief introduction with the motivation, we describe the hardware and software used and present the results of the studies using ATLAS simulated data samples.

INTRODUCTION

The silicon tracker of the ATLAS Inner Detector consists of 5832 modules in its SemiConductor Tracker (SCT) and Pixel Detector (PIX) (3568 in the Barrel region and 2264 in the EndCap/Forward region) [1]. With three translational and three rotational movements of the modules, the alignment of the tracker involves dealing with up to 34992 degrees of freedom (DoF). The global χ^2 algorithm [2] is aimed at achieving high precision alignment using the offline tracking information. It consists of minimizing a giant χ^2 resulting from a simultaneous fit of all particle trajectories and alignment parameters. The current implementation of the algorithm within the ATLAS software framework (Athena) is described somewhere else [3].

The global χ^2 algorithm requires solving large number of linear equations and handling large matrices in its full scope due to the large number of DoF in the system. Using conventional 32-bit single-CPU platforms is a limiting factor in the performance since they are unable to handle such large operations [4].

The solution proposed was the utilization of a parallel (Beowulf) cluster with a 64-bit architecture and extensive studies were performed earlier using a 32-bit parallel-cluster (Asgard [5]), from which predictions for 64-bit cluster were drawn. The focus of the present study is to initiate the performance tests of such a cluster for the first time, compare with earlier results and predictions, comment on the advantages of 64-bit parallel computing for the ATLAS silicon tracker alignment for further usage.

The three main advantages of 64-bit parallel computing over conventional platforms are the improvements in memory handling, precision and execution time. A parallel-processing system allows the distribution of a large size of data among various nodes. For example, a matrix with type double entries of size N=35k will require about 10GB disk space and memory allocation. This is more than a 32-bit single platform can handle as a 32-bit program can only address 4GB of virtual memory. On the other hand, a 64-bit architecture allows up to 16GB. Furthermore, if the matrix is distributed among 16 processors, then the load on a single node is reduced down to just 0.6GB. The matrices of alignment problems can have large condition numbers which may compete with the machine precision. Conventional 32-bit libraries are not adequate for computational accuracy for full size problem (35k DoF), as we will discuss later on. Making use of enlarged data types and extended-precision libraries in a 64-bit platform is a first step to improve the precision. Parallel processing allows faster execution times compared with single CPUs by distributing the CPU load.

COMPUTING PLATFORM

The parallel cluster used for the tests is resident at the Rutherford Appleton Laboratory (RAL) and is a 128-node dual CPU 64-bit AMD Opteron 248 2.2.GHz cluster. It runs Red Hat Enterprise (ES) Linux 3.0 and x86 code natively. The cluster is shared by various disciplines and we are allocated about 450GB disk space. Table 1 is a comparison of the cluster (SCARF) with Asgard as well as with the suggested platform. SCARF [6] matches and exceeds the suggested requirements.

The available compilers on SCARF are PGI (Portland Group, v5.2) and GNU (v3.2.3-53), for C, C++ and FORTRAN. GNU is publicly available, whereas PGI is commercial. GNU long double data type size is 16B on 64-bit (12B on 32-bit). PGI does not support quadrupole precision and the size of long double is the same as double on both platforms (8B). To obtain quadrupole precision, software emulators are possible to use, however, with an additional overload in time performance[1]. Simple time and precision performance tests were carried out by calculating expressions, $a = b \times 3 - 1, b = 1/3$, for various data types. The time performances are slightly better for optimized (O3) GNU, compared with PGI. A comparison of precision was also performed and shows about 3 orders of magnitude improvement from double to long double with GNU (as expected). A 32-bit optimized GNU compilation

* muge.karagoz.unel@cern.ch

[1]Private communication with a QA Engineer of Portland Group.

also yields same results on the 64-bit architecture, as with optimization, floats in the intermediate steps will be kept in the FPU. We should note that AMD Opteron holds 80-bit floating point registers and a long double floating point is 80-bit extended type. We chose to use GNU compiler for our studies.

Table 1: Comparison of parallel-computing clusters.

	Asgard	**Suggested**	**SCARF**
Archit.	32-bit	64-bit	64-bit
N_{CPU}	dual, 240	large N	dual, 128
CPU	PIII 0.5/	AMD Opteron	AMD Opteron
	0.6 GHz	2 GHz	248 2.2 GHz
RAM	1 G	4G	4G, 122 nodes
			8G, 16 nodes
Persistent	–	40 GB IDE	2x120 GB IDE
Network	–	1x Gigabit	2x Gigabit

TESTS ON THE BEOWULF CLUSTER

In all the following measurements, the AMD ScaLAPACK (v1.7), as an extension to the AMD Core Math Library (ACML) (v2.5.0), is used as the scalable version of LAPACK for distributed-memory parallel computing via BLACS(v1.1) [7]. AMD ScaLAPACK is optimized for high performance on AMD Opteron, its unoptimized version is a public domain software and can be downloaded from the web. The communication between the processors of the cluster were obtained by MPI libraries (v1.2.6..14a) for 64-bit architecture [8]. The queuing used was LSF (v6.1). We compiled and linked using GNU gcc compiler.

Matrix Inversion Tests Using ScaLAPACK

We tested the time and precision performance of SCARF using various-sized large matrices. Our test code inputs a randomly generated symmetric, double precision matrix kept in binary format. All measurements are performed for the function *pdgesvx* of ScaLAPACK, an expert driver routine. For 16 CPUs on a 4x4 grid, tests were performed for various matrix sizes with fixed square block size for the grid (SB=64). The dependency of walltime for the routine to matrix size is given in Figure 1, starting from N=64 up to N=16384. The dependence is found to be to the 2.3 power from a linear fit to the data. It should be noted that walltime per processor may include overhead for network communication as well, on top of the CPU time. In all the cases, the walltimes were very close to the total CPU times.

Subsequently, we provide a table of time performance in a 2D scan of number of CPUs and grid/block size (SB) on SCARF. N_{CPU} and SB has three measurements for the matrix inversion time: Wall and CPU times were measured for the inversion routine only, and realtimes measured for the whole MPI job submitted. The results are listed in a URL [9] to be used as a lookup table for the performance dependence on the grid structure given a matrix size. SCARF is a facility shared among various groups, therefore, the reproducibility of the results are variable, as the load on the cluster effects the queue waiting times, node availability and I/O operation load. This is in particular true for those problem sizes which required large number of CPUs. Nevertheless, we observe that 36 CPUs for SB=64 is an optimal choice for almost all our test points. The optimal block size agrees with the value suggested in ScaLAPACK's user guide [7].

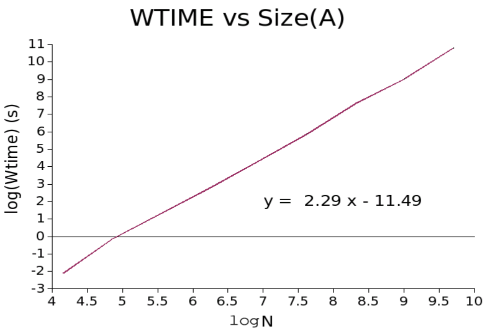

Figure 1: Measured wall time for inversion as a function of the size of the matrix, N.

The number of accurate decimal digits in the solution is related to the exponent of the condition number of the matrix, cond(A)[2]. We define the accuracy to be the ratio of infinity norm of the deviation of AA^{-1} from the identity matrix (I) to the reciprocal condition number of the matrix. This quantity gives us the number of digits preserved in the final solution and its matrix size dependence is shown in Figure 2. In our tests, digits are lost for a matrix size of about 16k. During previous tests with the 32-bit cluster, this point was at about 14k. Therefore, the results are improved over earlier results, however, not substantially, as the current ScaLAPACK libraries are not able to provide actual quadrupole precision. Earlier tests of quadrupole precision with MAPLE for N=200 matrix[3] its extrapolations to large N values showed that the accuracy will only be lost for sizes much larger than our full size problem.

A matrix of N=8000 was used to test the time performance of SCARF in comparison with another AMD Opteron cluster at 2.4 GHz with 32-bit compilation, using 16 CPUs and SB=256 on both platforms. 64-bit AMD showed an improvement of about a factor of 2.5 in time performance. The same-size matrix was also used to study the dependence of processing time on N_{CPU} used for the inversion. An $N^{-0.2}$ dependence was found for the walltime, from a fit to somewhat limited data points[4].

[2] The reader may refer to any of the packages referred to here, for a reminder about various definitions.

[3] Processing times were painfully long.

[4] It should be noted that this is, in general, a 3-parameter problem, which involves the matrix size, grid block size and number of processors in the grid.

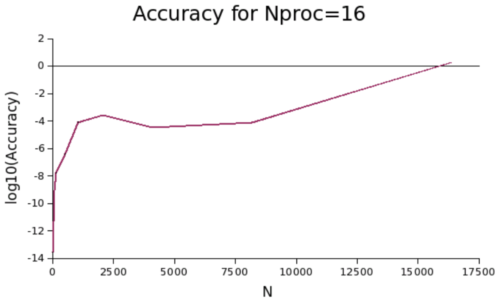

Figure 2: Accuracy as a function of the size of the matrix. For very small matrix sizes, the curve follows a faster slope.

Solving Eigenvalues of Global χ^2 Matrices

We have performed various tests for a performance comparison on eigenvalue solutions of matrices that are generated using simulated ATLAS data. Without additional constraints, Global χ^2 algorithm produces a singular matrix, where the singularities correspond to global motions of the detector, therefore, a few first modes of the system should be either completely singular or very close to zero (depending on the precision). For this study, we have not attempted to remove these modes.

A single muon (μ) sample was generated, simulated and reconstructed in the ATLAS inner detector, using a recent release of Athena (v11.0.3). Production was carried out on the Oxford Particle Physics Linux SLC3 cluster, using PBS. One full chain per event took 5-10s with an integrated time of $\approx$1000 hours. The muon η-range is chosen to cover [-2,2] uniformly; ϕ coverage was full and uniform. The transverse momenta distribution was flat and wide-ranged to prevent biases: $20 < p_{\mathrm{T}} < 100$ GeV/c. The sample consists of 825 thousand events, with almost equal share between the two μ charges. It was generated such that it has high detector coverage, while retaining reasonable enough track statistics. This is important even though our aim is to discuss solution performances rather than demonstrating the ultimate achievable precision. With larger statistics, the solution should be more reliable due to smaller fluctuations. The full setup studied consists of barrel (SCT+PIX) region. One region used is the special "cone" configuration, which covers only [0,1] in η, and corresponds to 1030 modules. This limited coverage is convenient for many reasons: it has a small problem size and has been studied earlier. The SCT coverage was obtained by about 550k and cone coverage was obtained by 250k tracks.

After track processing, the algorithm generates a binary matrix (basically the alignment inverse covariant matrix) to be solved for the eigenvalues and the eigenvectors of the problem. For this study, we used a standalone version of the algorithm, ported it also to SCARF. The solution was then used to extract alignment corrections for the silicon modules. We compared two platforms for the solver: SCARF vs Intel Pentium IV machines of the Oxford cluster. The 32-bit PIV had 2GB RAM, 2GB swap and 2.8/2 GHz CPU depending on the node. The software was compiled with GCC 3.2.3-49.

On SCARF, the systems were solved by a ScaLAPACK parallel eigenvalue/eigenvector solver routine (*pdsyevd*). The routine is a simple "Divide and Conquer" one, using recursive reduction of an instance of the problem into one or more smaller instances. On single CPU, the standalone code used a LAPACK routine (*dspev*) for eigencomputation of the matrices.

The cone problem was used to compare the single-CPU and parallel cluster eigensolutions. The agreement was found to be very good. For the SCT problem, one module remained unpopulated resulting in 6 real spurious modes and not taken into account to be solved. The situation arises because of limited statistics available for this study as well as a limited coverage. The SCT problem can be "solved" on both platforms. However, the solution on the parallel platform was found to be 50 times faster. Figure 3 shows the pull values of the SCT modes for alignment corrections. For the full barrel problem, number of unpopulated modules were 37. Figure 4 shows the eigenvalue spectrum for barrel's 21k modes. This size problem can only be solved (in a reasonable amount of time) using the parallel cluster. The matrix size is above the limits of a 32-bit machine as there is also a limitation due to the fact that ATLAS software by default imposes a 2GB file-size limit.

The timing results are summarized in Table 2. The repeatability of the parallel job performances were found very good. Here, we present rather conservative results of the subsequent jobs. Excluding the binary file I/O, the total running time for the process on SCARF was a little over a minute, for the cone problem size. For results from obsolete data samples, a fit to the processing time versus matrix size shows an $\mathrm{N}^{2.3}$ dependence and a fit to the job processing time versus number of processors for N=6180 matrix and using SB=64 yields an $\mathrm{N}^{-0.5}$ dependence (Figure 5). We also attempted to invert the matrices. As expected, the direct inversion problem is unsolved as the reciprocal condition number (RCOND) of the matrix gets smaller than the machine precision (ϵ) as the matrix becomes singular to working precision (for 6180 $\mathrm{N_{DoF}}$). Larger matrices were found to be completely singular with RCOND=0.0 and no solution was provided.

CONCLUSIONS

A 64-bit parallel cluster was utilized for matrix manipulations and various tests were performed. A number of case studies with global χ^2 outputs using distributed-memory parallel computing were studied for the first time. For matrix handling and solving, we demonstrated that 64-bit platforms are much advantageous over the 32-bit ones. Parallel-computing allowed for large problem solutions in a very efficient way and the performances greatly improved compared to single-CPU solvers. Now that large system sizes can be examined and alignment constants can be obtained, the alignment algorithm and its numerical issues

Table 2: Performance comparison grand table. The binary file size is given for the matrix in square format. Measurements include the binary file I/O time. The grid block size were set to 256 for all jobs.

$\mathbf{N}_{\mathrm{module}}$	$\mathbf{N}_{\mathrm{DoF}}$	**File Size (GB)**	**Processing Time**		
		(N×N×8B)	SCARF	Intel PIV	SCARF
			N_{CPU} = 16 / 4	Single CPU	
1030	6180	0.3	1m53s / 3m	35m (2.8 GHz)	39m
2112	12672	1.2	10m30s / 26m58s	7h34m (2 GHz)	5h28m
3568	21408	3.4	43m43s / 121m51s	—	> 20 h

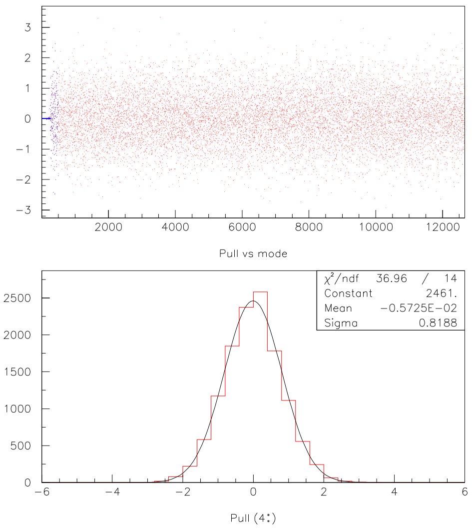

Figure 3: Pulls as a function of modes for the barrel SCT.

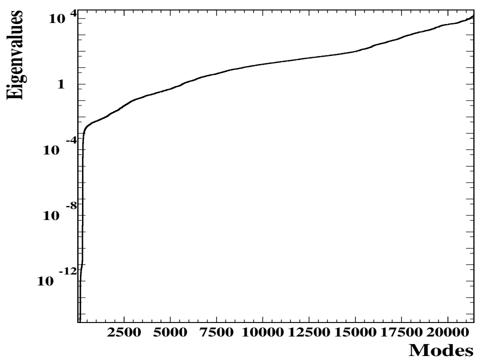

Figure 4: Eigenvalue spectrum for barrel tracker. First few modes are removed from the plot as they were close to zero.

will be debugged and understood better.

We find improvement in the precision due to the increase in the machine hardware precision. However, the predicted increase cannot be achieved by hardware solutions. To obtain further precision, emulations and iterative solutions libraries may be considered. A variety of possible library options exists: ScaLAPACK or other parallel eigen-

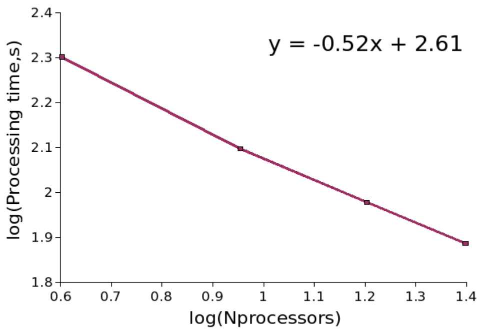

Figure 5: Processing time vs. N_{CPU} for the cone size problem, excluding the binary file output time.

solvers, such as PLAPACK[5], and Numerical Algorithms Group (NAG) libraries which are licensed software [10].

Unfortunately, no general quadrupole precision library exists. ScaLAPACK is working on implementation of extended precision, especially using iterative refinement using XBLAS. Few emails were iterated with the ScaLAPACK team who showed interest in our requests. NAG has iterative refinement procedures that can be used with a preconditioner for faster solution convergence yielding extended precision. Our understanding is that currently none of these libraries are guaranteed to extend to full quadrupole precision, and even if so, a true quadrupole precision solution may not be computed in time fast enough for the problem sizes of the global χ^2 algorithm. The exercise with 21k DoF indicates that it may be possible to use double precision, as the numerical precision limit seems to effect the weakest modes that the global χ^2 algorithm is not aiming to solve for. Further precision tests will prove useful using available libraries, especially with the option of dividing the problems into subspaces.

64-bit AMD platform is planned to be supported by ATLAS. Athena has been reported to run successfully on these machines, in 32-bit mode. A 64-bit port is expected to be available in conjunction with gcc 3.4.4 porting of the software, in summer 2006 [12]. This will also allow to exceed the 2GB per core memory size limit currently adapted in

[5]ScaLAPACK's *pdsyevd* was found to be faster than PLAPACK's solver [11].

the ATLAS software, on dedicated sites. When this happens, a better infrastructure will be in place for global χ^2 alignment processing.

ACKNOWLEDGEMENTS

Computing resources for parallel processing was provided by the CCLRC's e-Science facility at RAL. We thank the support group of these facilities. We also thank the Oxford Particle Physics computing support group for maintaining the cluster used for event simulation.

REFERENCES

[1] ATLAS Technical Design Reports 4 and 5, CERN/LHCC/97-16, CERN/LHCC/97-17, 1997.

[2] P. Brückman de Renström and S. Haywood, Phystat05 proceedings, Eds L. Lyons and M. Karagöz Ünel, 2006; `http://www.physics.ox.ac.uk/phystat05` .

[3] A. Hicheur, "Implementation of a Global Fit Method for the Alignment of the Silicon Tracker in ATLAS Framework", these proceedings.

[4] K. Bernardet, *et al.*, ATLAS-Marseille team, "Conclusions on Marseille Alignment Activities" and "Computing Requirements for the ID Alignment", 2004; `http://atlas.web.cern.ch/Atlas/GROUPS/INNER_DETECTOR/PERFORMANCE/ALIGN/docs` .

[5] `http://www.asgard.ethz.ch` .

[6] `http://hpcsg.esc.rl.ac.uk/scarf` .

[7] See `http://www.netlib.org/scalapack` and `http://developer.amd.com/assets/amdscalapack.pdf` .

[8] `http://www-unix.mcs.anl.gov/mpi` .

[9] `http://www-pnp.physics.ox.ac.uk/~karagozm/Oxford/scarf` .

[10] `http://www.nag.co.uk`.

[11] E. Breitmoser and A.G. Sunderland, "A Performance Study of the PLAPACK and ScaLAPACK", HPCx Tech. Report 406, 2003; `http://www.hpcx.ac.uk/research/hpc/technical_reports/HPCxTR0406.pdf` .

[12] `https://uimon.cern.ch/twiki/bin/view/Atlas/ComputingSystemCommissioning/` .