Distributed matrix computations via MPI on the GSI cluster *

J. Knedlik[1,2], *M.F.M. Lutz*[1], *and K. Schwarz*[1]
[1]GSI, Darmstadt, Germany; [2]Hochschule Darmstadt, Germany

The theory group in collaboration with the scientific computing group at GSI is preparing a general framework to predict and analyze hadronic final-state interactions. This is of crucial importance for a successful interpretation of data to be taken by the PANDA collaboration at FAIR. The main challenge is to combine constraints from micro-causality and coupled-channel unitarity using the concepts of effective field theories. A general scheme was proposed recently and already tested successfully in various well studied sectors of hadronic interactions [1, 2, 3, 4]. An application to proton-antiproton annihilation requires an extension to a huge number of channels. Part of the corresponding framework has been prepared in [5, 6]. In this report we explore the computational needs of such a project.

The proper treatment of final state interaction requires the solution of large, dense linear systems, that cannot be solved by a single CPU. To meet the enormous memory requirements, a distribution of the problem on a computer cluster is required. Available software, like GSL or LAPACK, for solving linear equations is proven, but most of them are not designed to be scalable on a computer cluster. ScaLAPACK is an extension to LAPACK which fits this purpose by supplying functions to build and solve linear systems efficiently in a scalable distributed memory environment by utilizing 2D Block Cyclic Mapping. ScaLAPACK is capable to be configured with a customized BLAS, and to be tinkered to our clusters' hardware environment. As inter process communication protocol OpenMPI came into use since this is already supported in the GSI cluster. In order to minimize the usage of cluster resources ScaLAPACK must use an optimized BLAS. To find a good BLAS implementation for our hardware, Intels MKL- and AMD's ACML have been bench marked against the non optimized reference BLAS. ScaLAPACK can be further optimized, by setting the right size of blocks, in which the linear system is divided, that are evenly distributed onto the available CPU's. The best block size is determined by the hardware environment through networkpacket-, cachesize etc. In order to find the best block size another set of benchmarks have been conducted.

A C++-interface to ScaLAPACK, for building and solving linear systems, has been implemented for easy access to ScaLAPACK's routines. In conclusion a solution to the computational need by the theory group has been found by utilizing ScaLAPACK to distribute the computation in the cluster. Many linear systems of order 50000 have already been successfully solved on the cluster, but there are still crashes when scaled over 100 CPU's. Also no real data have been used in these computations so far. An important task for the near future is to solve the the scalability-issues on the cluster.

* Work supported by the KOSI program

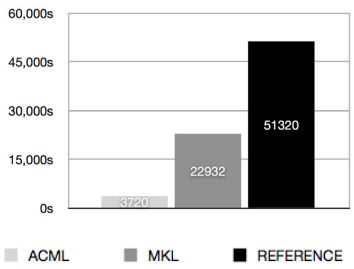

Figure 1: This figure shows the runtime of the different BLAS implementations solving a linear system of order 20000. The fastest result of 30 runs has been used. AMD's implementation of BLAS proves to be the most optimized.

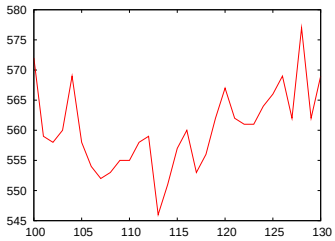

Figure 2: runtime with different block sizes solving a linear system of order 10000, filled with random values. The range of the best block size has been found in previous tests. The average of 50 runs has been taken. The best block size is 113.

References

[1] A. Gasparyan and M. F. M. Lutz, Nucl. Phys. A **848** (2010) 126.

[2] I. V. Danilkin, L. I. R. Gil and M. F. M. Lutz, Phys. Lett. B **703** (2011) 504.

[3] I. V. Danilkin, M. F. M. Lutz, S. Leupold and C. Terschlusen, arXiv:1211.1503 [hep-ph].

[4] A. M. Gasparyan, M. F. M. Lutz and E. Epelbaum, arXiv:1212.3057 [nucl-th].

[5] S. Stoica, M. F. M. Lutz and O. Scholten, Phys. Rev. D **84** (2011) 125001.

[6] M. F. M. Lutz and I. Vidana, Eur. Phys. J. A **48** (2012) 124.