Parallel minimization via Geneva, ScaLAPACK and MPI on the GSI cluster

J. Knedlik[1], *M.F.M. Lutz*[1], *and K. Schwarz*[1]
[1]GSI, 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. In 2012's Annual Report [1], we discussed our motivation in depth and presented our choices of software for the underlying needs of our framework.

In this report, we want to explore the framework's near final form and the workflow we are using for code developement.

We use Mathematica as the primary physic based developement environment, to ensure easy mathematical correctness and consistency by using Mathematica's internal checks. MathCode C++ is then able to compile Mathematica's code into native C++ code. We further extended this code with our own functions and classes, to enable the usage of ScaLAPACK's scalable solving of linear systems in Mathematica.

The desired use case requires to minimize a function with respect to a specific parameter set. The problem we needed to solve, was to create that function by MathCode C++, which needs ScaLAPACK's cluster wide scale.

Geneva is a C++ library for clustered problem solving by using evolutionary algorithms. In Geneva, in order to minimize our function, a broker sends singular genotypes to clients (in our usecase: varying parameters of our function) to let the clients evaluate their phenotype's fitness (in our usecase: the chisquare of that set of parameters). After many iterations over a well-sized population a good quality minimum will be reached. In combination, we can let Geneva spawn n cluster jobs (Geneva Clients), with m nodes (ScaLAPACK worker nodes) using ScaLAPACK's distributed solving for the analysis of our function with one specific parameter set. We can then iterate over evolutionary generations of these parameter sets, in order to find the parameter set that realizes the global minimum of our function.

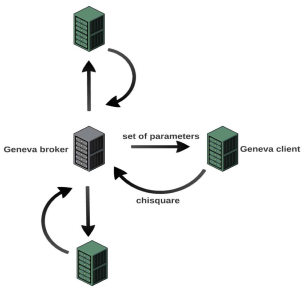

Figure 1: This figure shows the use case. Geneva's broker sends different sets of parameters to every Geneva client. The clients evaluate the functions' chisquare with that specific set of parameters and return them to the broker. The broker in return generates a new population of parameter sets with its evolutionary algorithm in order to find the minimum of our function.

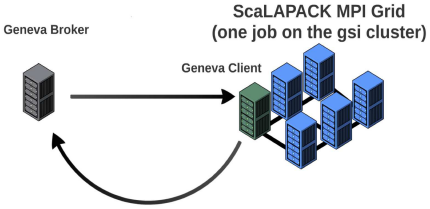

Figure 2: This figure shows how we integrated ScaLAPACK in Genevas workflow. A cluster job builds an MPI grid with all nodes. One node starts the Geneva client daemon and handles the communication of the parameter set to all mpi nodes. All other nodes wait in an application specific workloop. The Geneva client daemon can then utilize the computing power of the other nodes per mpi by using the functions of the mpi workloop(like ScaLAPACK).

In conclusion we have built a developement environment meeting the 2 requirements of both :

- easy usage by utilizing Mathematica and MathCode C++ including mathematical consistency checking
- solving and minimizing the developed Mathematica function on a cluster scale

So far, some problems have already been solved by utilizing Geneva on a singular machine with 64 cores. ScaLAPACK has been successfuly used to solve linear systems of an order of 200K on the GSI Prometheus cluster.

The important task, that is ongoing and well on its way right now, is to merge both of these aspects to show, that the combination of those two fully meet the requirements of our initial use case.

References

[1] J. Knedlik, M. Lutz, K. Schwarz GSI Report 2013-1, **504**, (2013).

DOI:10.15120/GR-2015-1-INFRASTRUCTURE-IT-11