Linompss – A Linear Algebra Library on OMPSs

Sicong Zhuang
Computer Science Department - BSC
sicong.zhuang@bsc.es

Abstract— Exascale performance will require massive parallelism and asynchronous execution (DARPA, DOE, EESI2). The former pertains to the design choice to increase hardware performance through growing core counts. The latter ensures that the associated software scales well. As a result, traditional, bulk-synchronous parallel programming models like OpenMP or MPI will likely fall out of grace in favor of more amenable variants. At the time of writing, the general consensus inclines towards task-based programming models that support dynamic scheduling and use graph-based models to record task dependencies.

In this context, we present LINOMPSs, a parallel linear algebra library built on top of OMPSs. Currently, we offer most of the BLAS-3 functionality and the three main factorizations (QR, LU, Cholesky). There is limited support for sparse matrices. For systems of equations, LINOMPSs disposes of implementations of both direct methods as well as iterative solvers.

At the lowest level, LINOMPSs can be linked with most BLAS or LAPACK libraries (MKL, ESSL, Netlib, ATLAS...). These functions are used, in turn, to implement the OMPSs tasks for the parallel implementations of the BLAS, or the Blocked BLAS. The Blocked BLAS are used to write blocked versions of the factorizations in OMPSs, which can then be combined to write solvers or full-fledged numerical simulations with finite element methods, multigrid methods, etc. These applications benefit from the asynchronous parallelism developed by OMPSs at run time. Moreover, the layered design allows for the concurrent and out-of-order execution of tasks that stem from previously distant regions of computation.

We not only consider LINOMPSs as an environment for the development of linear algebra applications and benchmarks for the Computer Science Department. In our own department for example, we use LINOMPSs to experiment with mixed-precision and incomplete/inaccurate computations. With the expertise of the CASE and Earth Science Department at BSC, we envision LINOMPSs as a tool for cross-disciplinary collaboration and the development of industrial-strength engineering applications.

A. INTRODUCTION

LINOMPSs (LINear algebra on OMPSs) is a numerical linear algebra library developed in the Computer Science department of BSC. It strives to provide high performance and scalability on heterogeneous multi-core architectures, as well as to offer a clear interface as a math kernel for building large scientific/engineering applications.

B. GENERAL STRUCTURE

At its core LINOMPSs relies on an optimized implementation of the BLAS and LAPACK (ATLAS, Intel MKL, etc.). Each of the functions in those libraries has a corresponding wrapper in LINOMPSs which is compiled with OMPSs task support. A parallelized version of the BLAS functions (GEMM(), SYRK(), TRSM()) is then realized by blocking the input matrices and passing the blocks to the aforementioned OMPSs tasks. The OMPSs runtime handles the task dependencies and dynamically schedules the tasks in an asynchronous fashion.

Along the same lines, we have implemented several LAPACK functions (Cholesky factorization, LU factorization) and solvers (Jacobi, Gauss-Seidel, Conjugate Gradient).

C. FUNCTIONALITY

The very basic level exposed to the user is a set of wrappers of the BLAS and LAPACK functions compiled with OMPSs tasks as a dynamic library. This set of wrappers also serves as the building block for the other LINOMPSs functions.

Going one level above are three blocked parallel BLAS level 3 functions (GEMM(), SYRK(), TRSM()). From this level on we also build a standalone program for benchmarking along with the dynamic libraries.

With these blocked functions at hand plus a few wrappers we are able to build two blocked parallel matrix factorization functions LU and Cholesky.

Subsequently, two blocked parallel linear equation solvers (one for general matrix and the other for a positive-definite matrix respectively) are built on top of that. Using the LU/Cholesky factorization to decompose the input matrix and applying forward-, backward-substitution to solve the system.

The current version of Linompss also ships with some iterative solvers (Conjugate Gradient, Iterative Refinement, Jacobi and Gauss-Seidel).

D. PERFORMANCE

Experimenting on the standalone benchmark programs from Linompss we were able to extract some speedup plots (on MareNostrum III – BSC).
E. COMPARISON AND FUTURE WORK

We are aware of another state-of-the-art numerical linear algebra library PLASMA that takes a rather similar approach (blocked/tiled algorithm and a dynamic scheduler).

We thus drew a comparison on two functions dpotrf() from both libraries with the same set of parameters. As is shown in the figure below both libraries demonstrate a fairly good scalability trend.

As of now Linompss is under active development. We are planning to further extend its supported functions (QR factorization, various sparse functions to name a few) and to exploit the OMPSs programming model to a wider extent (priority, support for heterogeneous devices etc.).