

# A Framework for Check-pointed Fault-Tolerant Out-of-Core ScaLAPACK LU Factorization\*

*E.F. D'Azevedo<sup>†</sup>, P. Luszczyk<sup>‡</sup>*

## 1 Introduction

This paper presents a framework for breaking down time consuming large scale out-of-core ScaLAPACK LU factorization into a sequence of simple micro-instructions. These simple instructions encode subroutine arguments and ScaLAPACK array descriptors and can be easily mapped to subroutines in the ScaLAPACK library or to perform disk I/O operations. A simple driver examines the complete list of micro-instructions to determine the computations and I/O required to check-point or restart the computation. Updates are first written to a temporary location and then into the restart disk file. Writing the data in two steps provides extra protection against failures. Numerical experiments on a Linux cluster demonstrated that the check-point and restart capability incurred about 3% additional overhead above the version without check-point capability. This approach of using simple micro-

---

\*The submitted manuscript has been authored by a contractor of the U.S. Government under Contract No. DE-AC05-00OR22725. Accordingly, the U.S. Government retains a non-exclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes.

<sup>†</sup>Computer Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, TN 37831, e6d@ornl.gov.

<sup>‡</sup>Department of Computer Science, University of Tennessee, Knoxville, TN 37996, luszczyk@cs.utk.edu.

instructions can be generalized for check-pointing other time consuming computations such as QR or Cholesky factorization.

Large dense linear problems arise from many scientific areas such as modeling the effect of radio-frequency heating of plasmas in fusion applications [4], modeling high-resolution three-dimensional wave scattering using boundary element formulation [1, 3], in the cosmic microwave background data analysis<sup>1</sup>, and modeling neutrino scattering on lattices as part of the Terascale Simulations of Supernovae (TSI)<sup>2</sup>.

This work builds upon the ScaLAPACK out-of-core<sup>3</sup> extension [2] for LU factorization by adding a check-point and restart capability for long running computations. The out-of-core extension of ScaLAPACK enables the solution of large problems several times larger than available physical memory by storing the factors on disk. The computational cost for dense LU factorization increases as  $O(N^3)$  whereas the required storage increases as  $O(N^2)$ . Thus a ten-fold increase in storage amplifies the computational work by  $10^{3/2} = 31.6$ . Although most in-core LU factorizations complete within an hour, some large scale out-of-core computations can require several days. Most high performance computing facilities limit the runtime of individual jobs (say 12 or 24 hours) and typical mean time between failure (MTBF) for large clusters is also about several days. Therefore, a robust check-point and restart capability is of great value for long running computations.

Section 2 briefly describes the out-of-core ScaLAPACK software and the portable software layer to perform disk I/O. Section 3 describes the micro-instructions in more detail. The two-step process for updating the restart file is described in Section 4. Numerical experiments on a Linux cluster and described in Section 5. We summarize our findings in Section 6.

## 2 Out-of-core ScaLAPACK

The out-of-core ScaLAPACK LU factorization uses a ‘left-look’ algorithm to reduce the volume of I/O required for write operations. The full matrix is stored on disk and overwritten by the LU factors. Since pivoting is required, the algorithm uses two in-core column panels, called X and Y panels. Panel X acts as a buffer to hold and apply previously computed factors to panel Y using Parallel BLAS (PBLAS)<sup>4</sup>. Once all updates are performed, panel Y is factored using the in-core ScaLAPACK `PxGETRF` routine. The results in panel Y are then written back to disk. Panel X is chosen to be wide enough to achieve good I/O performance and the remaining bulk of available memory should be allocated to panel Y. This strategy reduces the volume of I/O by minimizing the number of passes over the previously computed factors. The volume of I/O data transferred is usually dominated by reads instead of writes.

The I/O to disk is performed by a portable high level software layer. Disk

---

<sup>1</sup>ScaLAPACK is used by the Microwave Anisotropy Dataset Computational Analysis Package (MAPCAP), for details see [www.nersc.gov/~borrill/cmb/madcap/](http://www.nersc.gov/~borrill/cmb/madcap/).

<sup>2</sup>ScaLAPACK is used in OAK3D, for details see [www.phy.ornl.gov/theory-astro/scidac/](http://www.phy.ornl.gov/theory-astro/scidac/).

<sup>3</sup>The software is available at [www.netlib.org/scalapack/prototype/index.html](http://www.netlib.org/scalapack/prototype/index.html).

<sup>4</sup>For details see [www.netlib.org/scalapack/pblas\\_gref.html](http://www.netlib.org/scalapack/pblas_gref.html).



If this first write is successful, the out-of-core matrix is then updated with panel Y. If a failure occurs during this time, the intact data in the temporary location can be used for recovery. Other zero length empty files are used to mark I/O progress. For example, the existence of file 00143a.dat (00143b.dat) indicates the first (second) write is successful for instruction 143.

However, there are still limitations in the recovery process. If the data is written to a local file system (commonly /tmp on a Linux cluster), the same set of processors must be used for restarting the computation. The correct assignment of MPI tasks to the processor grid is determined at runtime and reconfigured using BLACS\_GRIDMAP. The software assumes data written to disk is automatically 'flushed' or 'sync-ed' to disk after the file is closed. Delayed writes may cause problems during a machine crash if the data is still cached in memory.

## 5 Numerical Experiment

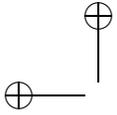
The check-point and restart capability imposes a very low overhead on the original ScaLAPACK out-of-core solver. The software was tested on the TORC Linux cluster<sup>5</sup>. Each compute node was configured as a 2 Ghz Pentium 4 with 768 MBytes of memory, 30 GBytes IDE local disk. Although both 100 Mbit/s ethernet and Gigabit ethernet connections were available, the software used LAM/MPI over 100 Mbits ethernet. ScaLAPACK version 1.7 and ATLAS BLAS (non SSE2) were used. About 288 MBytes of memory was allocated as work space for the out-of-core LU solver. Matrix size was  $56,000 \times 56,000$  (REAL\*8) with block size ( $MB = NB = 50$ ).

On a  $2 \times 2$  processor grid, the original version without check-point capability took about 31274 sec (8.69 hr), whereas the check-point version took about 32210 sec (8.95 hr), or about 3% overhead for processing the micro-instruction and updating the disk file in two steps. Overall performance was about 913 Mflop/s per cpu in the version using check-pointing. The same matrix was solved using in-core ScaLAPACK routines on a  $7 \times 7$  processor grid, which required about 512 MBytes per cpu, in about 3012 sec (0.84 hr). Overall performance was about 793 Mflop/s per cpu. Time to read in the matrix across all processors was about 30 sec or about 17 MBytes/s per cpu. Time for writing out the matrix was about 44 sec or about 11.6 MBytes/s per cpu. As observed in [2] and [5], the performance of the out-of-core solver was higher than the in-core solver since most of the computation was performed in large blocks, whereas the remaining submatrix size decreases rapidly for the in-core solver.

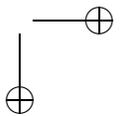
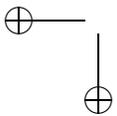
## 6 Summary

We have presented a check-point and restart enhancement to the ScaLAPACK out-of-core LU factorization. The approach was to generate simple micro-instructions that can be easily mapped to high level I/O and ScaLAPACK subroutines. A simple driver can then easily examine the entire instruction list to determine the I/O and computation needed for check-point and restart. Writing the data in two steps

<sup>5</sup>Tennessee Oak Ridge Cluster Project (TORC), for details see [www.csm.ornl.gov/TORC/](http://www.csm.ornl.gov/TORC/).



provides extra protection against unexpected failures. Performance on a Linux cluster suggests the new version imposes very low overhead. The same approach can be easily extended to other time consuming computations such as Cholesky or QR factorization.



# Bibliography

- [1] T. CWIK, R. VAN DE GEIJN, AND J. PATTERSON, *The application of parallel computation to integral equation models of electromagnetic scattering*, Journal of Optical Society of America A, 11 (1994), p. 1538.
- [2] E. D'AZEVEDO AND J. DONGARRA, *The design and implementation of the parallel out-of-core scalapack LU, QR, and Cholesky factorization routines*, Concurrency:Practice and Experience, 12 (2000), pp. 1481–1493.
- [3] P. GENG, J. T. ODEN, AND R. VAN DE GEIJN, *Massively parallel computation for acoustical scattering problems using boundary element methods*, Journal of Sound and Vibration, 191 (1996), p. 145.
- [4] E. F. JAEGER, L. A. BERRY, E. D'AZEVEDO, D. B. BATCHELOR, AND M. D. CARTER, *All-orders spectral calculation of radio-frequency heating in two-dimensional toroidal plasmas*, Physics of Plasmas, 8 (2001), pp. 1573–1583.
- [5] W. C. REILEY AND R. A. VAN DE GEIJN, *POOCLAPACK: Parallel out-of-core linear algebra package*, Tech. Rep. 99-33, Department of Computer Science, The University of Texas, Austin, Texas, 1999. (also available as PLAPACK Working Note #10).