PLASMA 17.1 Functionality Report

Parallel BLAS and Norms
Linear Systems and Least Squares
Mixed Precision and Matrix Inversion

Maksims Abalenkovs
Jack Dongarra
Mark Gates
Azzam Haidar
Jakub Kurzak
Piotr Luszczek
Mawussi Zounon
Samuel Relton
Jakub Sistek
David Stevens
Ichitaro Yamazaki
Asim YarKhan

Department of Electrical Engineering & Computer Science, University of Tennessee
School of Mathematics, The University of Manchester

June 7, 2017

Contents

1 Introduction 1

2 Parallel BLAS 3

3 Parallel Norms 4

4 Linear Systems 6

5 Mixed Precision 8

6 Matrix Inversion 9

7 Least Squares 10

8 Miscellaneous 12

9 Auxiliary 13

10 Performance 14

11 Traces 17

List of Tables

2.1 Level 3 BLAS routines. 3

3.1 Matrix norm routines. 4
3.2 Matrix norm types. 4

4.1 Linear systems routines. 7
4.2 Band linear systems routines. 7

5.1 Mixed precision routines. 8

6.1 Matrix inversion routines. 9

7.1 Least squares routines. 10

8.1 Miscellaneous routines. 12

9.1 Auxiliary routines. 13
9.2 Configurable parameters. 13

List of Figures

10.1 Performance of dpotrf on POWER8. 15
10.2 Performance of dsytrf on POWER8. 15
10.3 Performance of dgeinv on Haswell. 16
10.4 Performance of dpoinv on Haswell. 16

11.1 Trace of plasma_dgetrf() on Haswell 18
11.2 Trace of plasma_dpotrf() on Haswell 18
11.3 Trace of plasma_dgetrf() on KNL 19
11.4 Trace of plasma_dpotrf() on KNL 19

CHAPTER 1

Introduction

PLASMA (Parallel Linear Algebra for Multicore Architectures) is a dense linear algebra package at the forefront of multicore computing. PLASMA is designed to deliver the highest possible performance from a system with multiple sockets of multicore processors. PLASMA achieves this objective by combining state of the art solutions in parallel algorithms, scheduling, and software engineering. PLASMA currently offers a collection of routines for solving linear systems of equations and least square problems.

Tile Matrix Layout

PLASMA lays out matrices in square tiles of relatively small size, such that each tile occupies a continuous memory region. Tiles are loaded to the cache memory efficiently with little risk of eviction while being processed. The use of tile layout minimizes conflict cache misses, TLB misses, and false sharing, and maximizes the potential for prefetching. PLASMA contains parallel and cache efficient routines for converting between the conventional LAPACK layout and the tile layout.

Tile Algorithms

PLASMA introduces new algorithms redesigned to work on tiles, which maximize data reuse in the cache levels of multicore systems. Tiles are loaded to the cache and processed completely before being evicted back to the main memory. Operations on small tiles create fine grained parallelism providing enough work to keep a large number of cores busy.

Dynamic Scheduling

PLASMA relies on runtime scheduling of parallel tasks. Runtime scheduling is based on the idea of assigning work to cores based on the availability of data for processing at any given point in time, and thus is also referred to as data-driven scheduling. The concept is related closely to the idea of expressing computation through a task graph, often referred to as the DAG (Directed Acyclic Graph), and the flexibility of exploring the DAG at runtime. This is in direct opposition to the fork-and-join scheduling, where artificial synchronization points expose serial sections of the code and multiple cores are idle while sequential processing takes place.

CHAPTER 2

Parallel BLAS

Table 2.1 lists all the Level 3 BLAS routines implemented in PLASMA. PLASMA implements the complete set. Parallel BLAS routines in PLASMA are numerically equivalent to their reference Netlib implementations.

Name	**Description**
☐gemm	matrix matrix multiply
☐hemm	Hermitian matrix matrix multiply
☐her2k	Hermitian rank-2k update to a matrix
☐herk	Hermitian rank-k update to a matrix
☐symm	symmetric matrix matrix multiply
☐syr2k	symmetric rank-2k update to a matrix
☐syrk	symmetric rank-k update to a matrix
☐trmm	triangular matrix matrix multiply
☐trsm	triangular solve with multiple right hand sides

Table 2.1: Level 3 BLAS routines.

CHAPTER 3

Parallel Norms

Table 3.1 lists all the norm routines implemented in PLASMA, and Table 3.2 lists all the types of norms supported. PLASMA norms are mathematically equivalent to LAPACK, but only the *max* norm is equivalent numerically. LAPACK norm routines traverse entire rows or columns in a consecutive order of elements. PLASMA, on the other hand, employs tiling, for increased parallelism, which alters the numerical properties of the *one*, *infinity*, and *Frobenius* norms.

Name	**Description**
□lange	norm of a matrix
□lanhe	norm of a Hermitian matrix
□lansy	norm of a symmetric matrix
□lantr	norm of a triangular matrix

Table 3.1: Matrix norm routines.

Name	**Description**
PlasmaMaxNorm	max norm - maximum absolute value
PlasmaOneNorm	one norm - maximum column sum
PlasmaInfNorm	infinity norm - maximum row sum
PlasmaFrobeniusNorm	Frobenius norm - square root of sum of squares

Table 3.2: Matrix norm types.

In the *one* norm routine, first the vector of column sums is computed for each tile. Then the partial results are combined in a final reduction step. In the *infinity* norm routine, the same approach is applied row-wise. In the *Frobenius* norm routine, the sum of squares is computed for each tile, then the partial results are combined, and then the square root is taken. The *Frobenius* norm follows the LAPACK approach of scaling the results along the way, to minimize roundoff errors. Check the LAPACK □`lassq` routine for details.

In general, computing partial sums should be beneficial, rather than detrimental, to the numerical stability of the norm computations. Tiling has no effect on the *max* norm, as the operation is order invariant.

In addition, PLASMA contains the `[dz|sc|d|s]amax` routine, which computes the *max* norm for each column of a matrix, and returns the result as a vector. This routine is needed for checking the convergence of the solution in the iterative refinement process of the mixed precision solvers.

CHAPTER 4

Linear Systems

Dense

Table 4.1 lists all the linear systems routines implemented in PLASMA. Same as LAPACK, PLASMA provides a routine for solving a system of linear equations, as well as a routine for only factoring the matrix, and a routine for solving a system using a previously factored matrix. This allows for factoring a matrix once, and reusing the result for repeatedly solving different right hand sides.

PLASMA routines for solving general systems of linear equations are based on the LU factorization with partial (row) pivoting. Routines for solving symmetric positive definite systems of linear equations are based on the Cholesky factorization. Both LU and Cholesky factorization routines in PLASMA are numerically equivalent to their LAPACK counterparts.

The factorization routine for solving a symmetric (non positive definite) system first reduces the matrix into a band form based on the tiled Aasen's algorithm [?]. This is different from the blocked Aasen's algorithm [?] implemented in LAPACK, and its bound on the backward error depends linearly on the tile size. Then, the band matrix is factored using the PLASMA's band LU factorization routine in the next subsection.

Name	Description
□gesv	linear system solve
□getrf	triangular factorization
□getrs	linear system solve (previously factored)
[z|c] hesv	Hermitian linear system solve
[z|c] hetrf	Hermitian triangular factorization
[z|c] hetrs	Hermitian linear system solve (previously factored)
□posv	positive definite linear system solve
□potrf	positive definite triangular factorization
□potrs	positive definite linear system solve (previously factored)
[d|s] sysv	symmetric linear system solve
[d|s] sytrf	symmetric triangular factorization
[d|s] sytrs	symmetric linear system solve (previously factored)

Table 4.1: Linear systems routines.

Band

Table 4.2 lists all the band linear systems routines implemented in PLASMA. PLASMA routines for solving band systems of linear equations are based on a band version of the LU factorization with partial (row) pivoting. Routines for solving symmetric positive definite band systems of linear equations are based on a band version of the Cholesky factorization. Both the band LU and the band Cholesky factorization routines in PLASMA are numerically equivalent to their LAPACK counterparts. However, the band LU routine relies on the PLASMA's LU panel factorization routine that explicitly applies the pivots to the previous columns of the panel. Hence, though the matrix is stored in the LAPACK's band matrix format, its leading dimension must accommodate the additional $n_d - 1$ entries on the bottom, where n_d is the tile size.

Name	Description
□gbsv	band linear system solve
□gbtrf	band triangular factorization
□gbtrs	band linear system solve (previously factored)
□pbsv	band positive definite linear system solve
□pbtrf	band positive definite triangular factorization
□pbtrs	band positive definite linear system solve (previously factored)

Table 4.2: Band linear systems routines.

CHAPTER 5

Mixed Precision

Table 5.1 lists all the mixed precision routines implemented in PLASMA. PLASMA implements mixed precision routines for the solution of general linear systems of equations and symmetric positive definite systems of equations. PLASMA mixed precision routines are numerically equivalent to their LAPACK counterparts.

Name	**Description**
[zc|ds]gesv	linear system solve
[zc|ds]posv	positive definite linear system solve

Table 5.1: Mixed precision routines.

The algorithms is based on factoring the matrix in reduced precision (`float`) and recovering the full precision accuracy (`double`) in the process of iterative refinement. The approach is motivated by the performance advantage of single precision arithmetic over double precision arithmetic, which is typically $2\times$. The technique works if the condition number of the input matrix is smaller than the single precision *machine epsilon*. If this is not the case, the routine falls back on solving the system in full precision. The method incurs memory overhead due to the necessity of storing the input matrix both in the full precision and in the reduced precision.

CHAPTER 6

Matrix Inversion

Table 6.1 lists all the matrix inversion routines implemented in PLASMA. PLASMA implements routines for computing the inverses of general matrices and for computing inverses of symmetric positive definite matrices. PLASMA matrix inversion routines are numerically equivalent to their LAPACK counterparts.

Name	**Description**
□getri	matrix inversion
□potri	positive definite matrix inversion

Table 6.1: Matrix inversion routines.

Generally, matrix inversion should not be used for solving linear systems of equations. Instead, matrix factorization should be use, such as LU, LL^T or LDL^T factorization, followed by forward and backward substitution. Explicitly finding the inverse of a matrix is still required by some applications, such as computing the covariance matrix in statistics.

CHAPTER 7

Least Squares

Table 7.1 lists all the PLASMA routines related to solving overdetermined and underdetermined systems of linear equations. This includes routines for QR and LQ factorizations, generation of the Q matrices, as well as application of the transformations without explicit generation of the Q matrices.

Name	**Description**
☐gelqf	LQ factorization
☐gelqs	minimum norm solve using LQ factorization
☐gels	overdetermined or underdetermined linear systems solve
☐geqrf	QR factorization
☐geqrs	least squares solve using QR factorization
☐[un|or]glq	generate the Q matrix from LQ factorization
☐[un|or]gqr	generate the Q matrix from QR factorization
☐[un|or]mlq	apply the Q matrix from LQ factorization
☐[un|or]mqr	apply the Q matrix from QR factorization

Table 7.1: Least squares routines.

PLASMA routines have the same numerical stability as LAPACK, but are not numerically equivalent to LAPACK. This is because PLASMA reduces the input matrices by tiles, not by full columns, and generates sets of tile reflectors in the process. This makes no difference to the user, as long as PLASMA functions are used for operations involving the reflectors, such as generation of the Q matrix or application of the transformations to another matrix.

CHAPTER 8

Miscellaneous

Name	Description
□desc2ge	PLASMA to LAPACK layout conversion
□desc2pb	band PLASMA to LAPACK layout conversion
□ge2desc	LAPACK to PLASMA layout conversion
□pb2desc	band LAPACK to PLASMA layout conversion
□lacpy	matrix copy
□laset	matrix initialization
□lascl	matrix multiplication by a scalar
□geadd	element-wise matrix addition
□tradd	element-wise triangular matrix addition
zlag2c	double complex to single complex conversion
dlag2s	double real to single real conversion
□lauum	compute U×U' or L'×L
□trtri	triangular matrix inverse
[dz|sc|d|s]amax	find maximum absolute value for each column
□geswp	row or column pivoting

Table 8.1: Miscellaneous routines.

CHAPTER 9

Auxiliary

Name	Description
plasma_init	initialization
plasma_finalize	cleanup
plasma_set	set parameter value
plasma_get	get parameter value

Table 9.1: Auxiliary routines.

excludes functions for manipulating descriptors, because the naming convention is in flux

Parameter Name	Possible Values
PlasmaTuning	PlasmaEnabled, PlasmaDisabled
PlasmaNb	positive integer
PlasmaIb	positive integer
PlasmaNumPanelThreads	positive integer

Table 9.2: Configurable parameters.

excludes parameters for controlling QR/LQ reduction patterns, because the naming convention is in flux

CHAPTER 10

Performance

This chapter presents performance charts for a handful of PLASMA routines. More charts are available in the *PLASMA 17 Performance Report*.

Linear systems on POWER8

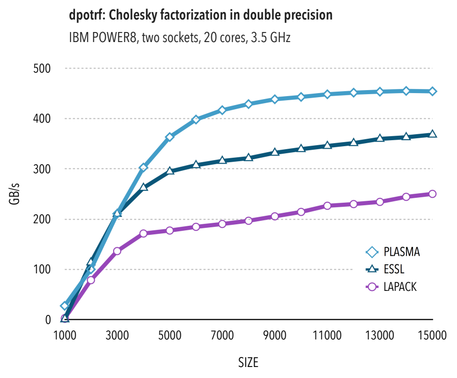

Figure 10.1: Performance of dpotrf on POWER8.

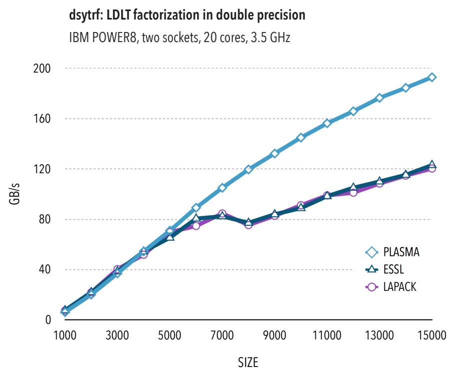

Figure 10.2: Performance of dsytrf on POWER8.

Matrix inversion on Haswell

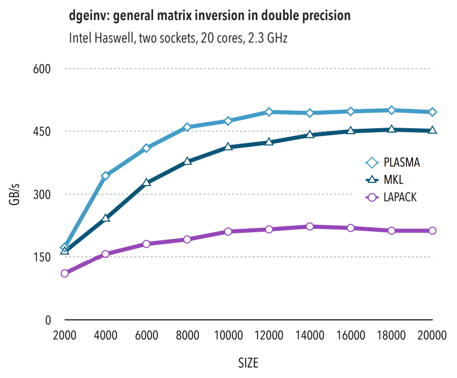

Figure 10.3: Performance of dgeinv on Haswell.

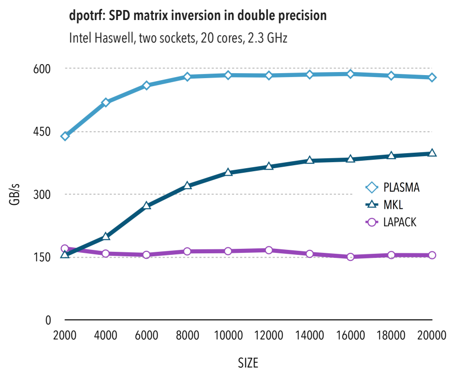

Figure 10.4: Performance of dpoinv on Haswell.

CHAPTER 11

Traces

This chapter presents execution traces of a handful of PLASMA routines. More traces are available in the *PLASMA 17 Performance Report*.

Linear Systems on Haswell

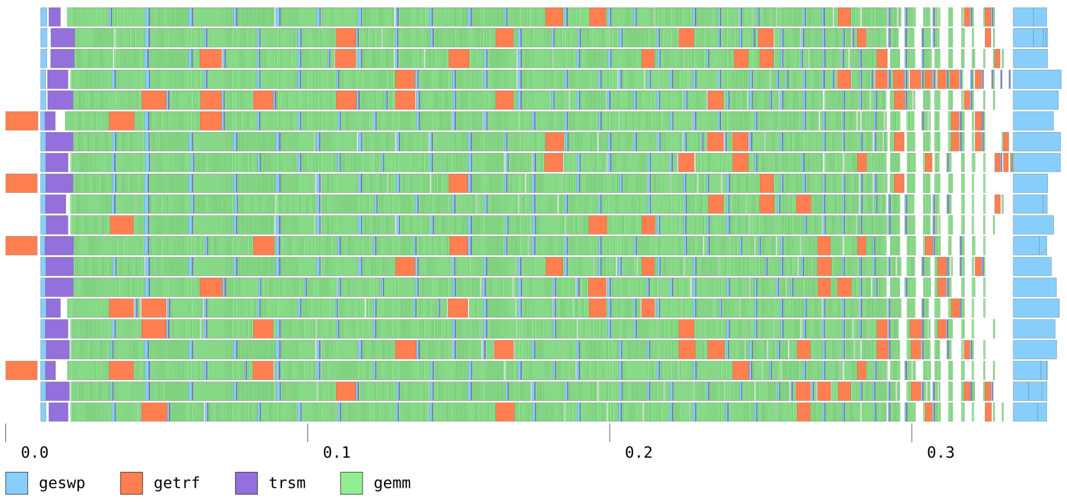

Figure 11.1: Trace of plasma_dgetrf() on Haswell:
`numactl --interleave=all ./test dgetrf --dim=6144 --nb=192 --ib=28 --mtpf=4`

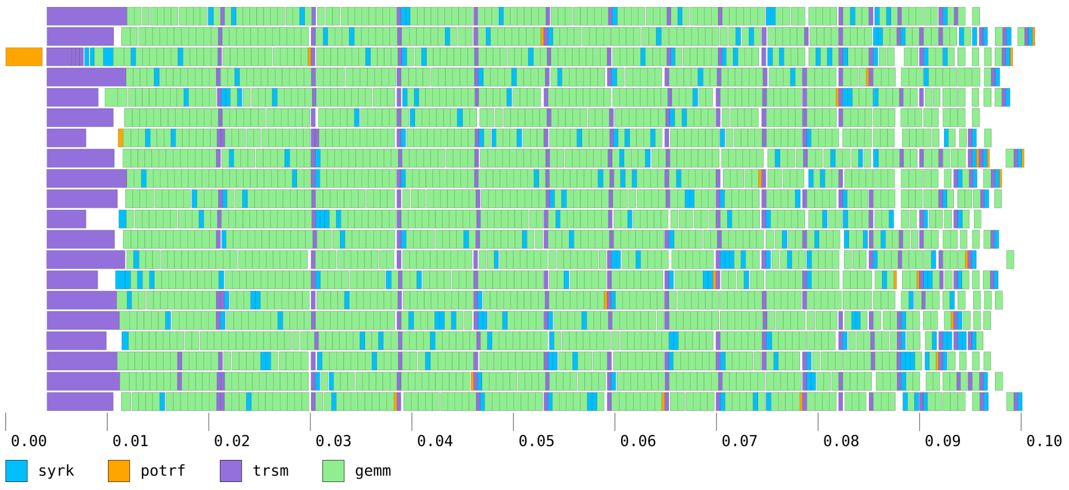

Figure 11.2: Trace of plasma_dpotrf() on Haswell:
`numactl --interleave=all ./test dpotrf --dim=5376 --nb=224`

Linear Systems on Knights Landing

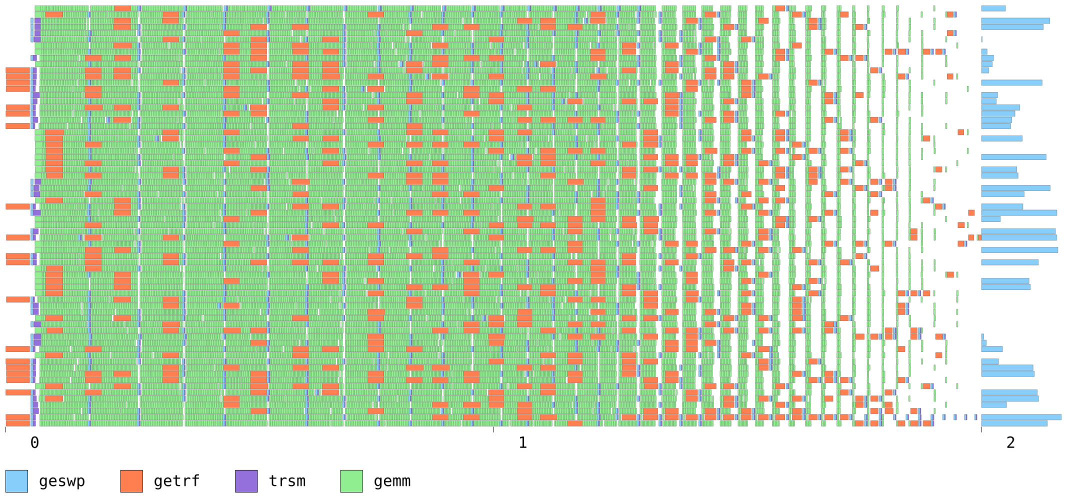

Figure 11.3: Trace of plasma_dgetrf() on KNL:
`numactl -m 1 ./test dgetrf --dim=14112 --nb=336 --ib=40 --mtpf=20`

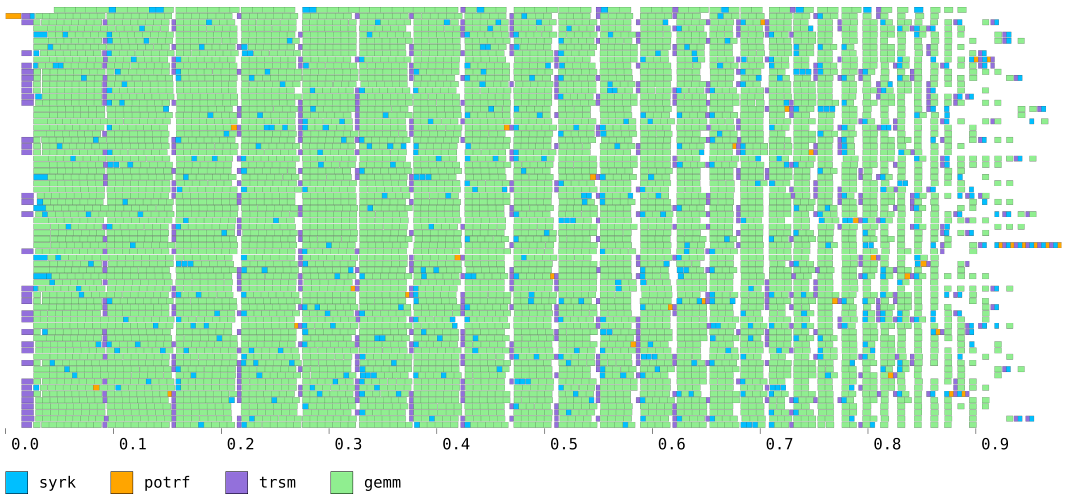

Figure 11.4: Trace of plasma_dpotrf() on KNL:
`numactl -m 1 ./test dpotrf --dim=15680 --nb=448`