MAGMA MIC 1.3 Release
Optimizing Linear Algebra for Applications on Intel Xeon Phi Coprocessors

J. Dongarra, M. Gates, A. Haidar, K. Kabir, P. Luszczek, S. Tomov, and I. Yamazaki

Innovative Computing Laboratory
Department of Computer Science
University of Tennessee, Knoxville

November 15, 2014
IPCC at ICL

ICL Intel Parallel Computing Center

The University of Tennessee's Innovative Computing Laboratory is now an Intel Parallel Computing Center.

The Intel Parallel Computing Center (IPCC) program is composed of universities, institutions, and labs that are leaders in their field, focusing on modernizing applications to increase parallelism and scalability through optimizations that leverage cores, caches, threads, and vector capabilities of microprocessors and coprocessors.

The objective of the Innovative Computing Laboratory’s IPCC is to develop and optimize numerical linear algebra libraries and technologies for applications, while tackling current challenges in heterogeneous Intel® Xeon Phi™ coprocessor-based high-performance computing. In collaboration with Intel’s MKL team, the IPCC at ICL will modernize the popular LAPACK and ScaLAPACK libraries to run efficiently on current and future manycore architectures, and will disseminate the developments through the open source MAGMA MIC library.

https://software.intel.com/ipcc
IPCC at ICL

LAPACK and ScaLAPACK

• Standard dense linear algebra (DLA) libraries
• Many applications rely on DLA
• Designed in 80/90’s for cache-based architectures

Must be redesigned for modern heterogeneous systems with multi/many-core CPUs and coprocessors.
IPCC at ICL

• **Develop**
  – Next generation LAPACK / ScaLAPACK
  – Programming models, and
  – Technologies
  for heterogeneous
  Intel Xeon Phi-based platforms

• **Disseminate developments**
  through the MAGMA MIC library

• **High value proposition**
  MAGMA MIC enables ease of use and adoption of Intel Xeon Phi architectures
  in applications as linear algebra is fundamental to scientific computing
# A New Generation of Dense Linear Algebra Libraries

Software/Algorithms follow hardware evolution in time

<table>
<thead>
<tr>
<th>LINPACK (70’s)</th>
<th>Rely on</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Vector operations)</td>
<td>- Level-1 BLAS operations</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LAPACK (80’s)</th>
<th>Rely on</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Blocking, cache friendly)</td>
<td>- Level-3 BLAS operations</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ScaLAPACK (90’s)</th>
<th>Rely on</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Distributed Memory)</td>
<td>- PBLAS Mess Passing</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PLASMA (00’s)</th>
<th>Rely on</th>
</tr>
</thead>
<tbody>
<tr>
<td>New Algorithms</td>
<td>- a DAG/scheduler</td>
</tr>
<tr>
<td>(many-core friendly)</td>
<td>- block data layout</td>
</tr>
<tr>
<td></td>
<td>- some extra kernels</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MAGMA</th>
<th>Rely on</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hybrid Algorithms</td>
<td>- hybrid scheduler</td>
</tr>
<tr>
<td>(heterogeneity friendly)</td>
<td>- hybrid kernels</td>
</tr>
</tbody>
</table>
MAGMA MIC
LAPACK for heterogeneous systems

• MAGMA MIC
  – Project on the development of a new generation of HP Linear Algebra Libraries
  – To provide LAPACK/ScaLAPACK on heterogeneous Intel Xeon Phi-based systems
  – Well established project with product disseminated through the MAGMA MIC libraries:

<table>
<thead>
<tr>
<th>MAGMA MIC 0.3</th>
<th>(2012-11-13)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAGMA MIC 1.0</td>
<td>(2013-05-03)</td>
</tr>
<tr>
<td>MAGMA MIC 1.1</td>
<td>(2014-01-07)</td>
</tr>
<tr>
<td>MAGMA MIC 1.2</td>
<td>(2014-09-17)</td>
</tr>
<tr>
<td>MAGMA MIC 1.3</td>
<td>(2014-11-15)</td>
</tr>
</tbody>
</table>
  • For heterogeneous, shared memory systems
  • Included are the main factorizations, linear system and eigen-problem solvers
  • Open Source Software (http://icl.cs.utk.edu/magma)

• Collaborators
  – Intel MKL Team
  – UC Berkeley, UC Denver, INRIA (France), KAUST (Saudi Arabia)
  – Community effort, similar to LAPACK/ScaLAPACK
**Key Features of MAGMA MIC**

**HIBRID ALGORITHMS**

MAGMA MIC uses hybrid algorithms where the computation is split into tasks of varying granularity and their execution scheduled over the hardware components. Scheduling can be static or dynamic. In either case, small non-parallelizable tasks, often on the critical path, are scheduled on the CPU, and larger more parallelizable ones, often Level 3 BLAS, are scheduled on the MICs.

**PERFORMANCE & ENERGY EFFICIENCY**

**MAGMA MIC on KNC**

LU factorization in double precision arithmetic

- **MIC**
  - KNC 7120
  - 60 cores @ 1.23 GHz

- **CPU**
  - Intel Xeon ES-2670 (Sandy Bridge)
  - 2x6 cores @ 2.60 GHz

- **3 GFlop / Watt**
  - 3 x faster for
  - 3 x less energy

**FEATURES AND SUPPORT**

**MAGMA MIC 1.3**

- Linear system solvers
- Eigen-problem solvers
- SVD
- CPU/AO interface
- MIC/Native interface
- Multiple precision support
- Mixed-precision iter. refinement solvers
- Multicore and multi-MIC support
- LAPACK testing
- Linux
Methodology overview

A methodology to use all available resources:

- **MAGMA MIC uses hybrid algorithms**
  - Representing linear algebra algorithms as collections of tasks and data dependencies among them
  - Properly scheduling tasks’ execution over multicore CPUs and manycore coprocessors

- Successfully applied to fundamental linear algebra algorithms
  - One- and two-sided factorizations and solvers
  - Iterative linear and eigensolvers

- **Productivity**
  1) High level;
  2) Leveraging prior developments;
  3) Exceeding in performance homogeneous solutions
A Hybrid Algorithm Example

Left-looking hybrid Cholesky

to parallel hybrid

MAGMA

```c
for (j=0, j<n; j+=nb) {
    jb = min(nb, n-j);
    magma_zherk(MagmaUpper, MagmaConjTrans, jb, j, one, dA(0,j), ldda, one, dA(j,j), ldda, queue);
    magma_zgetmatrix_async(jb, jb, dA(j,j), ldda, work, jb, &event);
    if (j+jb < n) {
        magma_zgemm(MagmaConjTrans, MagmaNoTrans, jb, n-j-jb, j, one, dA(0,j), ldda, dA(0,j+jb), ldda, one, dA(j,j+jb), ldda, queue);}
    magma_event_sync(event);
    zpotrf(MagmaUpperStr, &jb, work, &jb, info);
    if (info != 0) {
        *info += j;
        magma_zsetmatrix_async(jb, jb, work, jb, dA(j,j), ldda, queue, &event);
    }
}
```

Note:
- MAGMA and LAPACK look similar
- Difference is lines in red, specifying data transfers and dependencies
- Differences can be hidden in a dynamic scheduler making the top level representation of MAGMA MIC algorithms almost identical to LAPACK
A Hybrid Algorithm Example

Left-looking hybrid Cholesky

```c
for (j=0; j<n; j+=nb) {
    jb = min(nb, n-j);
    magma_zherk(MagmaUpper, jb, one, dA(0,j), ldda, one, dA(j,j), ldda);
    if (j+jb < n) {
        magma_zgemm(MagmaConjTrans, MagmaNoTrans, jb, n-j-jb, one, dA(0,j), ldda, dA(0,j+jb), ldda); 
        magma_zpotrf(MagmaUpperStr, &jb, work, &jb, info);
        if (info != 0) *info += j;
        if (j+jb < n) {
            magma_ztrsm(MagmaLeft, MagmaUpper, MagmaConjTrans, MagmaNo, jb, n-j-jb, one, dA(j,j), ldda, dA(j,j+jb), ldda); 
        }
    }
}
```

Note:
- MAGMA and LAPACK look similar
- Difference is lines in red, specifying data transfers and dependencies
- Differences can be hidden in a dynamic scheduler making the top level representation of MAGMA MIC algorithms almost identical to LAPACK

MAGMA MIC runtime environment

- Scheduling can be static or dynamic
- Dynamic is based on QUARK
- Uses CUDA streams to offload computation to the GPU
A Hybrid Algorithm Example

Left-looking hybrid Cholesky

MAGMA

1  for (j=0, j<n; j+=nb) {
2      jb = min(nb, n-j);
3      magma_zherk( MagmaUpper, MagmaConjTrans,
4                       jb, j, one, dA(0,j), ldda, one, dA(j,j), ldda,
5                   queue);
6      magma_zgetmatrix_async( jb, jb, dA(j,j), ldda, work, jb, &event);
7      if (j+jb < n) {
8          magma_zgemm( MagmaConjTrans, MagmaNoTrans,
9                             jb, n-j-jb, one, dA(0,j), ldda, dA(0,j+jb), ldda,
10                queue);
11          magma_zpotrf( MagmaUpperStr, &jb, work, &jb, info);
12          if (info != 0) { info += j; }
13          magma_zsetmatrix_async( jb, jb, work, jb, dA(j,j), ldda,
14                    queue);
15          magma_ztrsm( MagmaLeft, MagmaUpper, MagmaConjTrans,
16                              jb, n-j-jb, one, dA(j,j), ldda,
17                         queue);
18      }
19  }

Note:
- MAGMA and LAPACK look similar
- Difference is lines in red, specifying data transfers and dependencies
- Differences can be hidden in a dynamic scheduler making the top level representation of MAGMA MIC algorithms almost identical to LAPACK

MAGMA MIC runtime environment

CPU

MIC

MIC Queue

Computed on the MIC

Offloaded to the MIC

on the CPU

Offloaded to the MIC

CPU task #8 and CPU-MIC communications are overlapped with MIC computations

From sequential LAPACK

for( j=0, j<n; j+=nb) {
  jb = min(nb, n-j);
  zherk( MagmaUpper, jb, j, one, dA(0,j), ldda, one, dA(j,j), ldda,
        queue);
  zgetmatrix( sjb, sjb, dA(sj, sj), ldda, work, sjb, &event);
  if (sj+sjb < n) {
    zgemm( MagmaConjTrans, sjb, sj, one, dA(0,j), ldda, dA(sj,sj), ldda,
           queue);
    zpotrf( MagmaUpperStr, &sjb, work, &sjb, info);
    if (info != 0) { info += j; }
    zsetmatrix( sjb, sjb, work, sjb, dA(sj, sj), ldda,
                queue);
    ztrsm( MagmaLeft, MagmaUpper, MagmaConjTrans, sjb, sj-sjb, one, dA(sj,sj), ldda,
           queue);
  }
}
Programming models

• We developed two APIs for offloading work to MIC:
  
  - **LLAPI based**
    - A server runs on the MIC
    - Communications are implemented through LLAPI using SCIF
  
  - **Compiler pragma offload based**
    - API is using Phi-specific offload directives
    - Enhancements for CPU-MIC communications

Both APIs have the same interface and abstract low level programming details.
Scheduling strategies

High-productivity with Dynamic Runtime Systems

From sequential code

\[
\text{for } (k = 0; k < \min(MT, NT); k++)\
\quad \text{zgeqrt}(A[k;k], ...); \\
\quad \text{for } (n = k+1; n < NT; n++) \\
\quad \quad \text{zunmqr}(A[k;k], A[k;n], ...); \\
\quad \text{for } (m = k+1; m < MT; m++) \\
\quad \quad \quad \text{ztsqrt}(A[k;k], A[m;k], ...); \\
\quad \quad \quad \text{for } (n = k+1; n < NT; n++) \\
\quad \quad \quad \quad \text{ztsmqr}(A[m;k], A[k;n], A[m;n], ...); \\
\quad } \\
\]
Performance on single MIC
QR AO with static and dynamic MAGMA

Algorithms are scalable using all available hardware, e.g. CPU cores

71% of KNC peak
90% of dgemm peak

Host
Ivytown (2 x 12 @2.7 GHz)
DP Peak  518 GFlop/s

Coprocessor
Intel Xeon Phi (60 @ 1.23 GHz)
DP Peak  1180 GFlop/s
Scalability on multiple MICs

Performance scales well in spite of PCI's bandwidth limitations.

MAGMA DGETRF Performance (Multiple Card)

- 4 MIC
- 3 MIC
- 2 MIC
- 1 MIC

Performance GFLOP/s vs. Matrix Size N x N

Host: Sandy Bridge (2 x 8 @ 2.6 GHz)
- DP Peak: 332 GFlop/s

Coprocessor: Intel Xeon Phi (60 @ 1.09 GHz)
- DP Peak: 1046 GFlop/s

System DP Peak: 1378 GFlop/s
- MPSS 2.1.4346-16
- compiler_xe_2013.1.117

76% of peak
Plans & Goals: Dense Linear Algebra

• Derive new methods and algorithmic improvements
  – Eigensolvers and SVD using two-stage reductions
    [ remove the memory-bound limitations of the LAPACK algorithms,
      and depending on hardware show an order of magnitude improvement]
  – Factorizations and solvers for symmetric indefinite problems

• Develop linear algebra on small matrices
  – Batched linear algebra operations to provide support for various applications
  – Batched LU, QR, and Cholesky
    [ for the simultaneous factorization of many very small dense matrices ]
Plans & Goals: Sparse Linear Algebra (SLA)

• While extremely important for applications, SLA is notorious for running only at a fraction of the peak of modern architectures.

• Develop a highly optimized MAGMA MIC Sparse package
  [ include the standard CG, BiCGSTAB, GMRES, and preconditioned versions ]

• Incorporate communication-avoiding algorithms to significantly exceed in performance the standard memory and latency bound algorithms.
  [ include s-step methods, CA-GMRES, and blocked eigensolvers, e.g., LOBPCG ]
Plans & Goals: Mixed-Precision Methods

- Develop numerical algorithms that recognize and exploit the presence of mixed-precision mathematics:
  - Show 2x acceleration using mixed-precision iterative refinement solvers for dense problems;
  - Mixed-precision orthogonalization schemes to accelerate applications, sparse iterative linear system and eigenproblem solvers:

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>Gram-matrix formation $B := V^T V$ on MICs in extended precision</td>
</tr>
<tr>
<td>Step 2</td>
<td>Cholesky factorization $R^T R := B$ on CPUs in extended precision</td>
</tr>
<tr>
<td>Step 3</td>
<td>Backward-substitution $Q := VR^{-1}$ on MICs in standard-precision.</td>
</tr>
</tbody>
</table>
Plans & Goals: Benchmarks

• Develop a set of benchmarks for both performance and energy consumption. Include the
  – Newly proposed HPCG, optimized for Intel Xeon Phi architectures
  – Benchmarks for main communication and computation patterns
    [ e.g., CPU-MIC communication, MIC copy, MIC broadcast, latencies, representative BLAS 1/2/3, SpMV, SpMV, LU, SVD, etc. ]

• Show essential communication and computation patterns in various applications

• Goal is to encourage the focus of both hardware and software developers on architecture features and application needs; incorporate in performance analysis tools
Collaborators and Support

MAGMA team
http://icl.cs.utk.edu/magma

PLASMA team
http://icl.cs.utk.edu/plasma

Intel MKL team

Collaborating partners
University of Tennessee, Knoxville
University of California, Berkeley
University of Colorado, Denver
INRIA, France (StarPU team)
KAUST, Saudi Arabia