THE UNIVERSITY of TENNESSEE UT
NATIONAL LEADERSHIP IN HPC

SC12
Salt Lake City, Utah

MAGMA MIC: Linear Algebra Library for Intel Xeon Phi Coprocessors

Jack Dongarra w/ M. Gates, Y. Jia, K. Kabir, P. Luszczek, and S. Tomov
University of Tennessee, Knoxville

11 / 13 / 2012

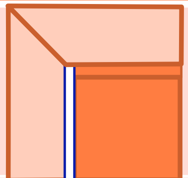
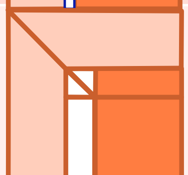
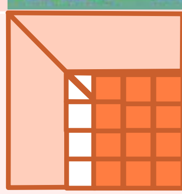
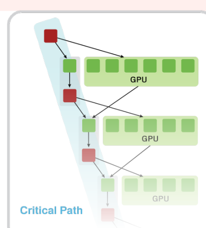
MAGMA: LAPACK for hybrid systems

- MAGMA
  - A new generation of HP Linear Algebra Libraries
  - To provide LAPACK/ScaLAPACK on hybrid architectures
  - http://icl.cs.utk.edu/magma/

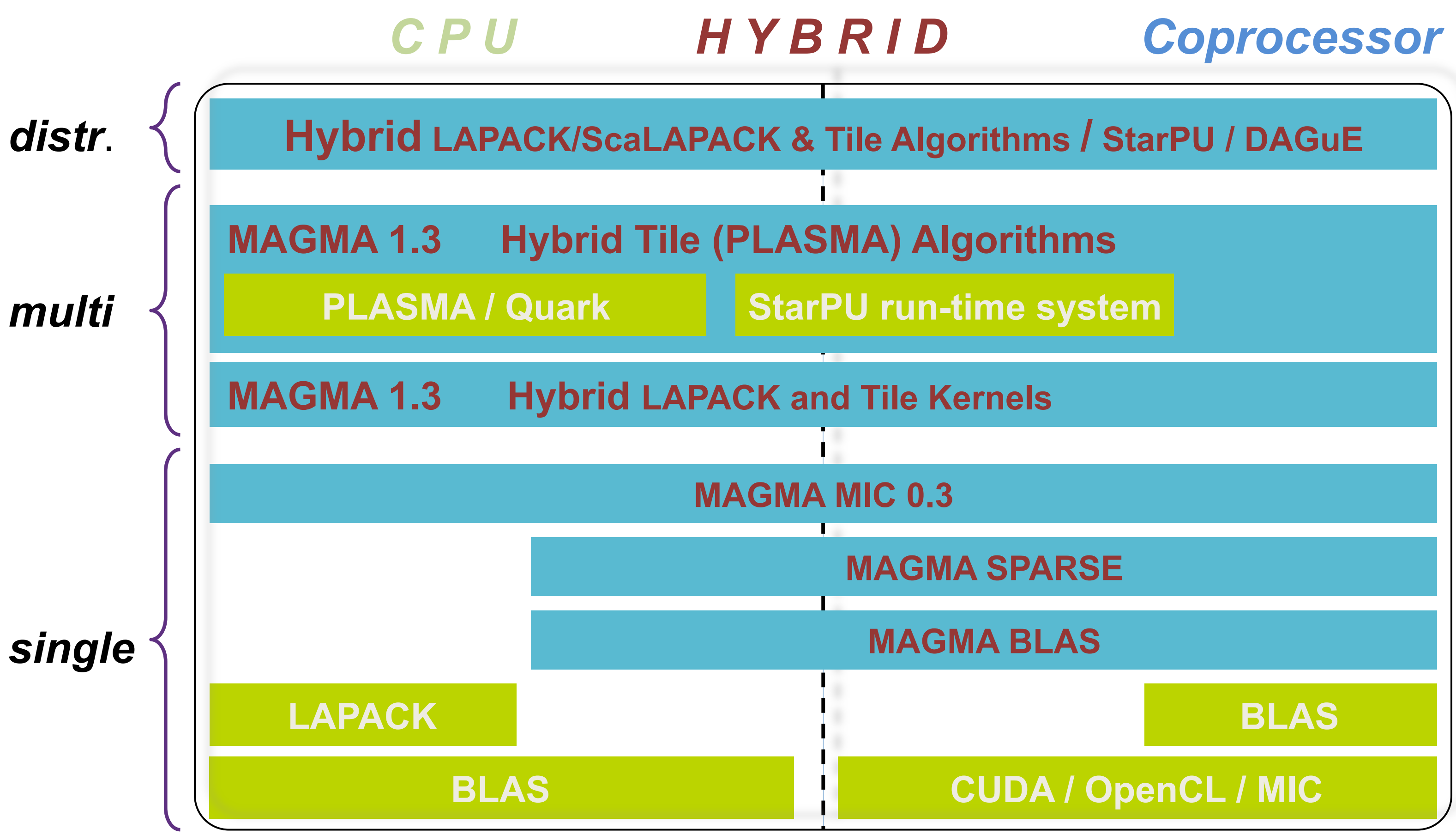
- MAGMA MIC 0.3
  - For hybrid, shared memory systems featuring **Intel Xeon Phi coprocessors**
  - Included are one-sided factorizations
  - Open Source Software (http://icl.cs.utk.edu/magma)

- MAGMA developers & collaborators
  - UTK, UC Berkeley, UC Denver, INRIA (France), KAUST (Saudi Arabia)
  - Community effort, similar to LAPACK/ScaLAPACK

A New Generation of DLA Software

Software/Algorithms follow hardware evolution in time		
LINPACK (70's) (Vector operations)		Rely on - Level-1 BLAS operations
LAPACK (80's) (Blocking, cache friendly)		Rely on - Level-3 BLAS operations
ScaLAPACK (90's) (Distributed Memory)		Rely on - PBLAS Mess Passing
PLASMA (00's) New Algorithms (many-core friendly)		Rely on - a DAG/scheduler - block data layout - some extra kernels
MAGMA Hybrid Algorithms (heterogeneity friendly)		Rely on - hybrid scheduler - hybrid kernels

MAGMA Software Stack



Support: ***Linux, Windows, Mac OS X*; *C/C++, Fortran*; *Matlab, Python***

MAGMA Functionality

- 80+ hybrid algorithms have been developed (total of 320+ routines)
  - Every algorithm is in 4 precisions (s/c/d/z)
  - There are 3 mixed precision algorithms (zc & ds)
  - These are hybrid algorithms, expressed in terms of BLAS
  - **MAGMA MIC** provides support for **Intel Xeon Phi Coprocessors**

MAGMA 1.3 ROUTINES & FUNCTIONALITIES	SINGLE GPU	MULTI-GPU STATIC	MULTI-GPU DYNAMIC
One-sided Factorizations (LU, QR, Cholesky)	✓	✓	✓
Linear System Solvers	✓		✓
Linear Least Squares (LLS) Solvers	✓		✓
Matrix Inversion	✓		✓
Singular Value Problem (SVP)	✓		
Non-symmetric Eigenvalue Problem	✓	✓	
Symmetric Eigenvalue Problem	✓	✓	
Generalized Symmetric Eigenvalue Problem	✓	✓	

SINGLE GPU
Hybrid LAPACK algorithms with static scheduling and LAPACK data layout

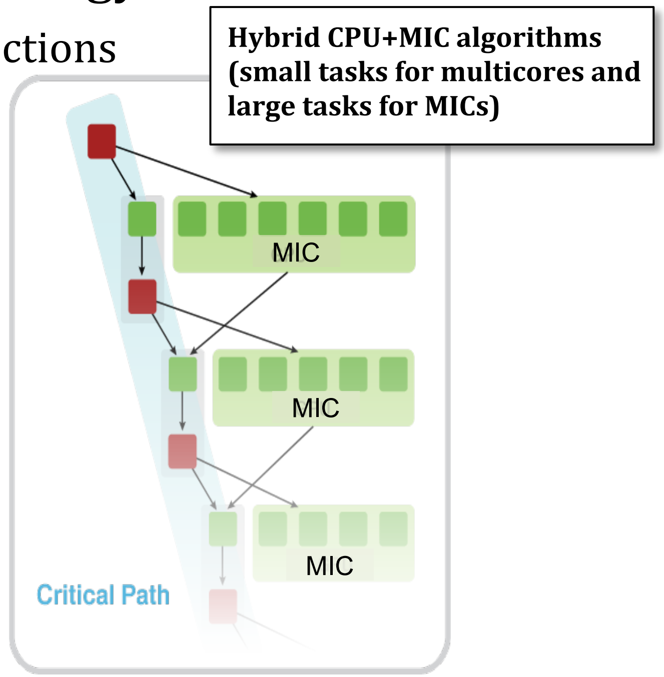
MULTI-GPU STATIC
Hybrid LAPACK algorithms with 1D block cyclic static scheduling and LAPACK data layout

MULTI-GPU DYNAMIC
Tile algorithms with StarPU scheduling and tile matrix layout

Methodology overview

A methodology to use all available resources:

- MAGMA MIC uses hybridization methodology based on
  - Representing linear algebra algorithms as collections of tasks and data dependencies among them
  - Properly scheduling tasks' execution over multicore CPUs and manycore coprocessors



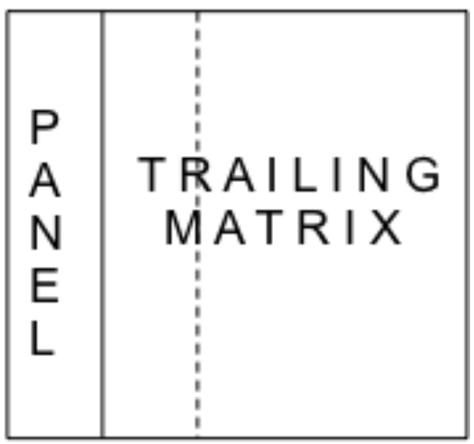
- Successfully applied to fundamental linear algebra algorithms
  - One- and two-sided factorizations and solvers
  - Iterative linear and eigensolvers

- Productivity
  1) High level;
  2) Leveraging prior developments;
  3) Exceeding in performance homogeneous solutions

Hybrid Algorithms

One-Sided Factorizations (LU, QR, and Cholesky)

- Hybridization
  - Panels (Level 2 BLAS) are factored on CPU using LAPACK
  - Trailing matrix updates (Level 3 BLAS) are done on the MIC using “look-ahead”



Programming LA on Hybrid Systems

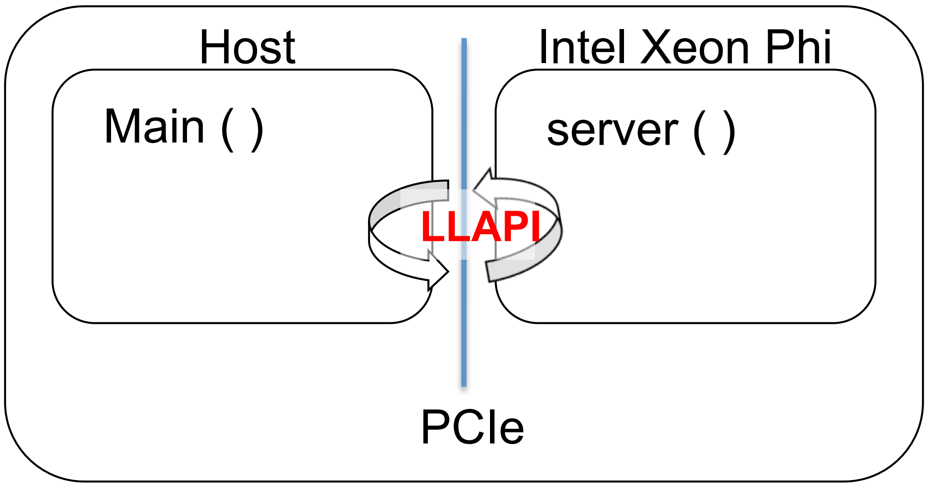
- Algorithms expressed in terms of BLAS
- Use vendor-optimized BLAS
- Algorithms expressed in terms of tasks
- Use some scheduling/run-time system

Intel Xeon Phi specific considerations

- Intel Xeon Phi coprocessors (vs GPUs) are less dependent on host
  [can login on the coprocessor, develop, and run programs in native mode]
- There is no high-level API similar to CUDA/OpenCL facilitating Intel Xeon Phi's use from the host *
- There is pragma API but it may be too high-level for HP numerical libraries
- We used Intel Xeon Phi's Low Level API (LLAPI) to develop MAGMA API
  [allows us to uniformly handle hybrid systems]

* OpenCL 1.2 support is available for Intel Xeon Phi as of Intel SDK XE 2013 Beta

MAGMA MIC programming model



- Intel Xeon Phi acts as coprocessor
- On the Intel Xeon Phi, MAGMA runs a "server"
- Communications are implemented using LLAPI

A Hybrid Algorithm Example

- Left-looking hybrid Cholesky factorization in MAGMA

```
1   for ( j=0; j<n; j += nb) {
2         jb = min(nb, n – j);
3         magma_zherk( MagmaUpper, MagmaConjTrans,
                            jb, j, m_one, dA(0, j), ldda, one, dA(j, j), ldda, queue );
4         magma_zgetmatrix_async( jb, jb, dA(j,j), ldda, work, 0, jb, queue, &event );
5         if ( j+jb < n )
6             magma_zgemm( MagmaConjTrans, MagmaNoTrans, jb, n-j-jb, j, mz_one,
                                  dA(0, j ), ldda, dA(0, j+jb), ldda, z_one,  dA(j, j+jb), ldda, queue );
7         magma_event_sync( event );
8         lapackf77_zpotrf( MagmaUpperStr, &jb, work, &jb, info );
9         if ( *info != 0 )
10             *info += j;
11        magma_zsetmatrix_async( jb, jb, work, 0, jb, dA(j,j), ldda, queue, &event );
12        if ( j+jb < n ) {
13            magma_event_sync( event );
14            magma_ztrsm( MagmaLeft, MagmaUpper, MagmaConjTrans, MagmaNonUnit,
                                jb, n-j-jb, z_one, dA(j, j), ldda, dA(j, j+jb), ldda, queue );
          }
    }
```

- The difference with LAPACK – the 4 additional lines in red
- Line 8 (done on CPU) is overlapped with work on the MIC (from line 6)

MAGMA MIC programming model

Host program

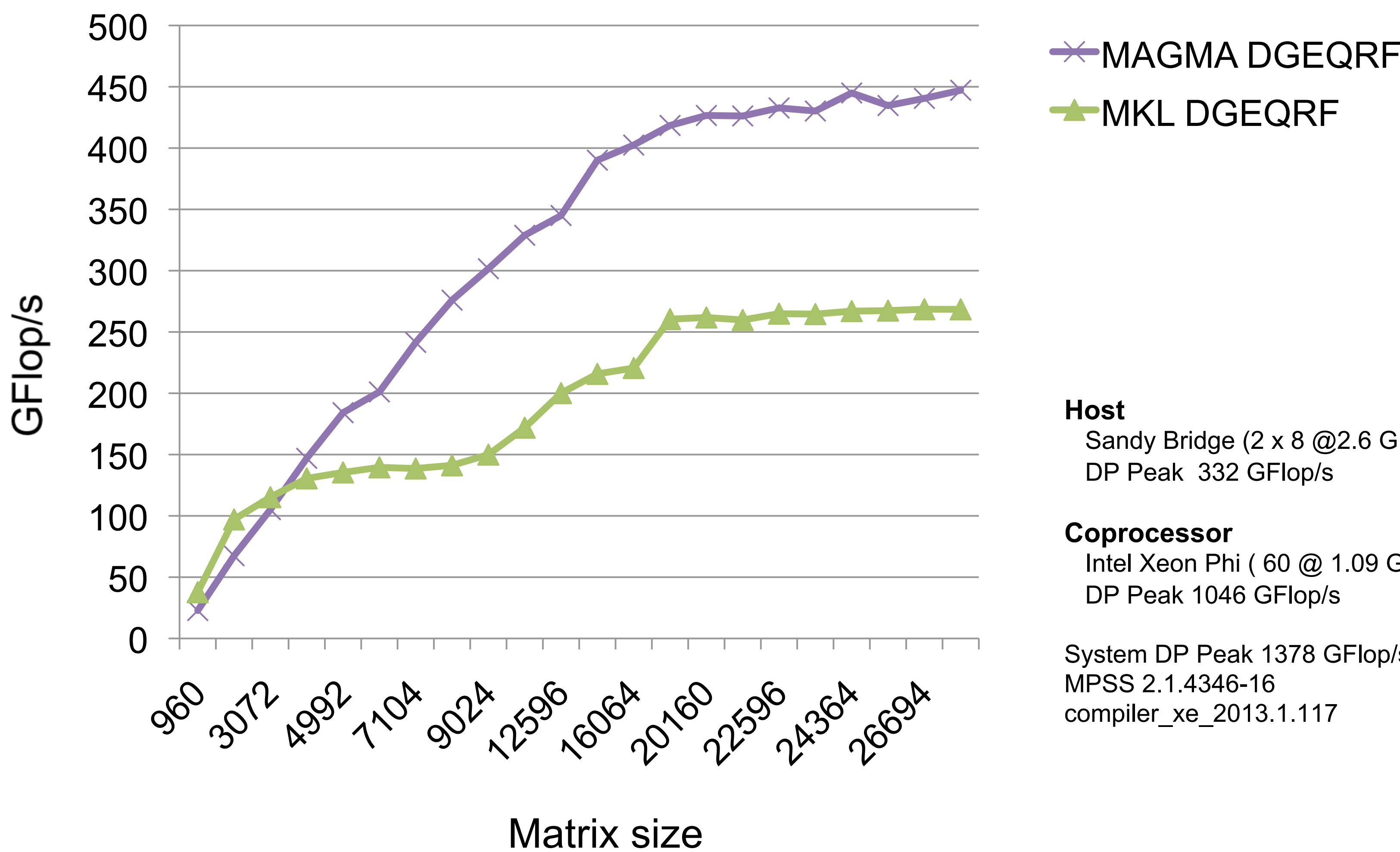
```
for ( j=0; j<n; j += nb)  {
      jb = min(nb, n – j);
      magma_zherk( MagmaUpper, MagmaConjTrans,
                          jb, j, m_one, dA(0, j), ldda, one, dA(j, j), ldda, qu
      magma_zgetmatrix_async( jb, jb, dA(j,j), ldda, work, 0, jb, queue
      if ( j+jb < n )
          magma_zgemm( MagmaConjTrans, MagmaNoTrans, jb, n-j-jb
                              dA(0, j ), ldda, dA(0, j+jb), ldda, z_one,  dA(j,
      magma_event_sync( event );
      lapackf77_zpotrf( MagmaUpperStr, &jb, work, &jb, info );
      if ( *info != 0 )
           *info += j;
      magma_zsetmatrix_async( jb, jb, work, 0, jb, dA(j,j), ldda, queue, &e
      if ( j+jb < n ) {
          magma_event_sync( event );
          magma_ztrsm( MagmaLeft, MagmaUpper, MagmaConjTrans, Mag
                                jb, n-j-jb, z_one, dA(j, j), ldda, dA(j, j+jb), ldda, queu
      }
}
```

Intel Xeon Phi interface

```
// ========================================
// BLAS functions
magma_err_t
magma_zgemm(
    magma_trans_t transA, magma_trans_t transB,
    magma_int_t m, magma_int_t n, magma_int_t k,
    magmaDoubleComplex alpha, magmaDoubleComplex_const_ptr dA, size_t dA_offset, m
                              magmaDoubleComplex_const_ptr dB, size_t dB_offset, m
    magmaDoubleComplex beta,  magmaDoubleComplex_ptr       dC, size_t dC_offset, m
    magma_queue_t handle )
{
    int err;
    magma_mic_zgemm_param gemm_param;

    gemm_param.transa = transA;
    gemm_param.transb = transB;
    gemm_param.m      = m;
    gemm_param.n      = n;
    gemm_param.k      = k;
    gemm_param.alpha  = alpha;
    gemm_param.a      = dA + dA_offset;
    gemm_param.lda    = lda;
    gemm_param.b      = dB + dB_offset;
    gemm_param.ldb    = ldb;
    gemm_param.beta   = beta;
    gemm_param.c      = dC + dC_offset;
    gemm_param.ldc    = ldc;

    int control_msg = magma_mic_ZGEMM;

    if ((err = scif_send(gEpd, &control_msg, sizeof(control_msg), 1)) <= 0) {
        err = er  o;
        printf("s   send failed with err %d\n", errno);
        fflush(st
    }

    if ((err = scif_          emm_param, sizeof(gemm_param), 1)) <= 0) {
        err = err
```

Send asynchronous requests to the MIC;
Queued & Executed on the MIC

MAGMA MIC Performance (QR)

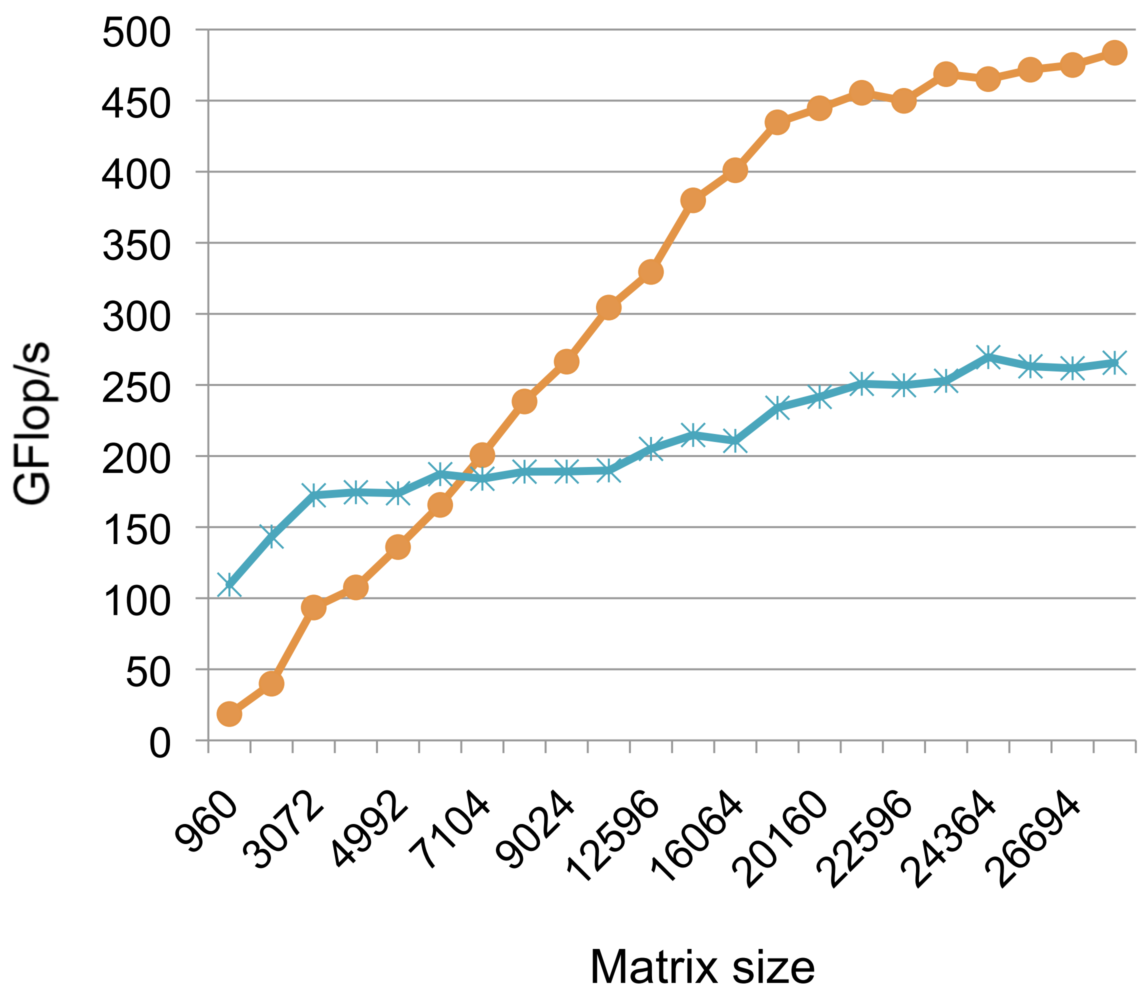


Host
Sandy Bridge (2 x 8 @2.6 GHz)
DP Peak 332 GFlop/s

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

MAGMA MIC Performance (Cholesky)

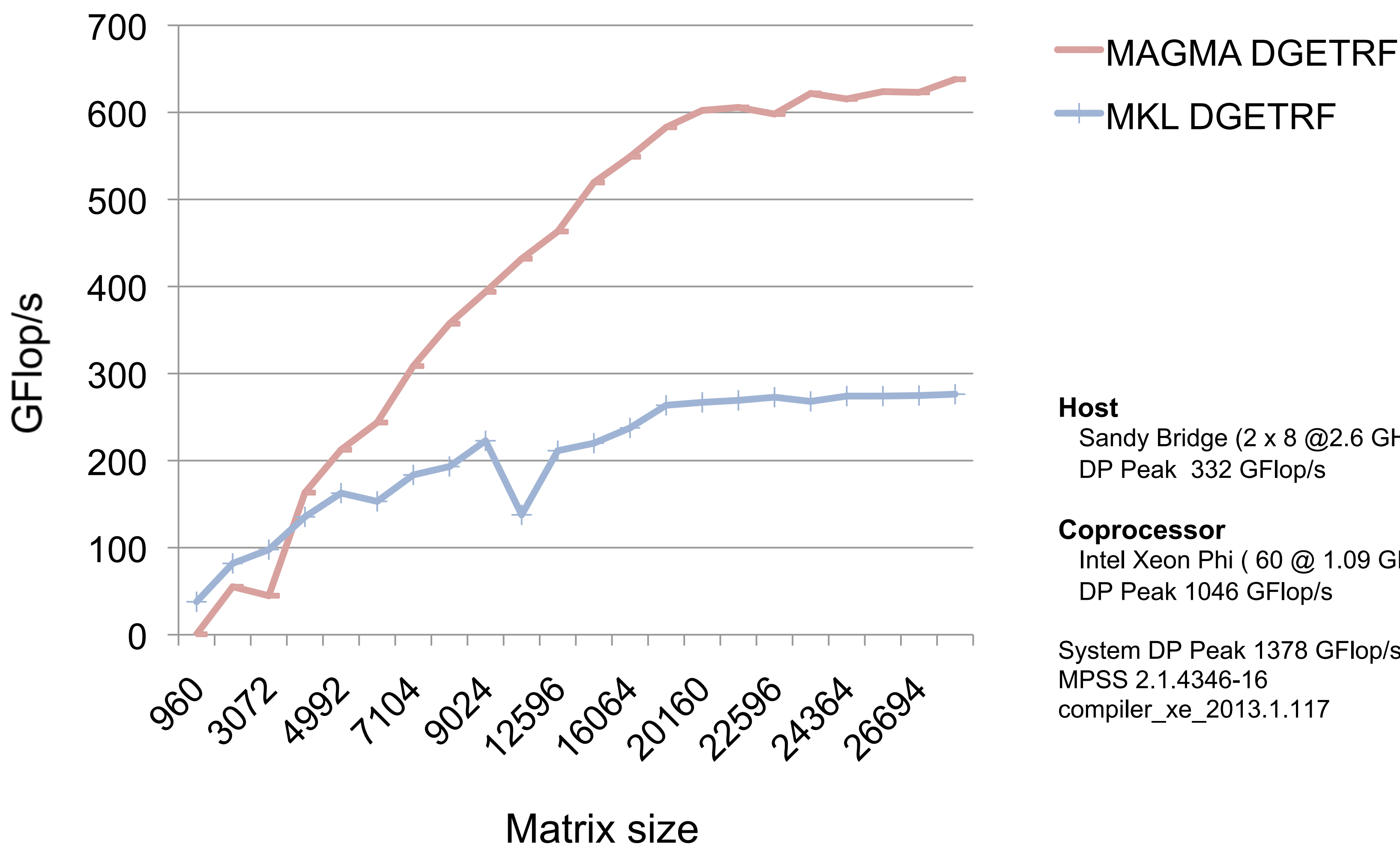


Host
Sandy Bridge (2 x 8 @2.6 GHz)
DP Peak 332 GFlop/s

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

MAGMA MIC Performance (LU)

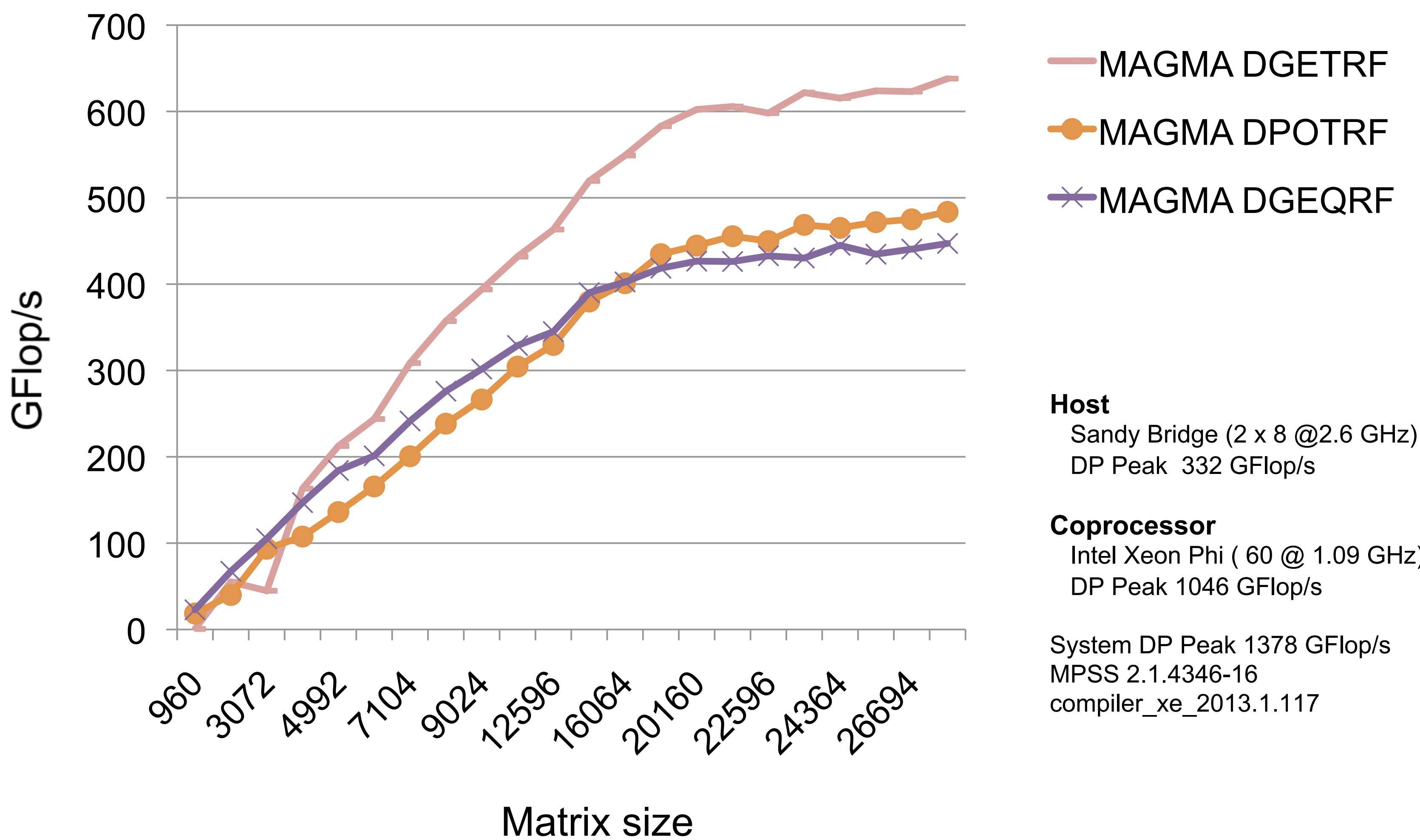


Host
Sandy Bridge (2 x 8 @2.6 GHz)
DP Peak 332 GFlop/s

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

MAGMA MIC Performance



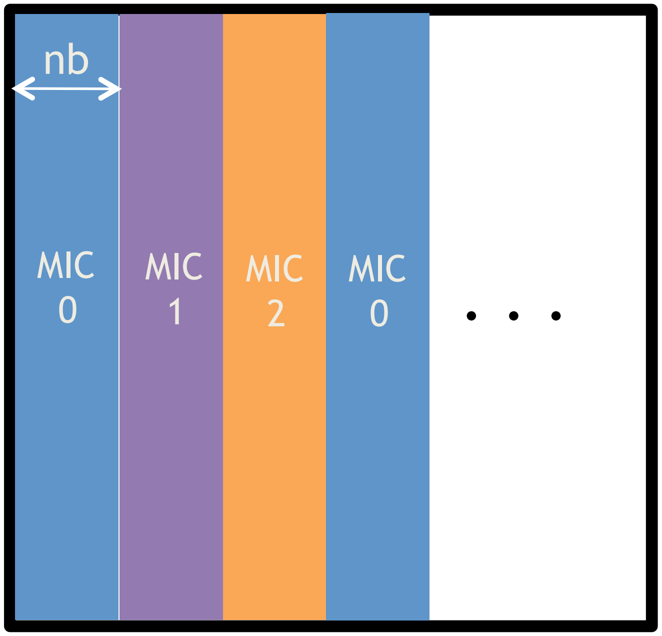
Host
Sandy Bridge (2 x 8 @2.6 GHz)
DP Peak 332 GFlop/s

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

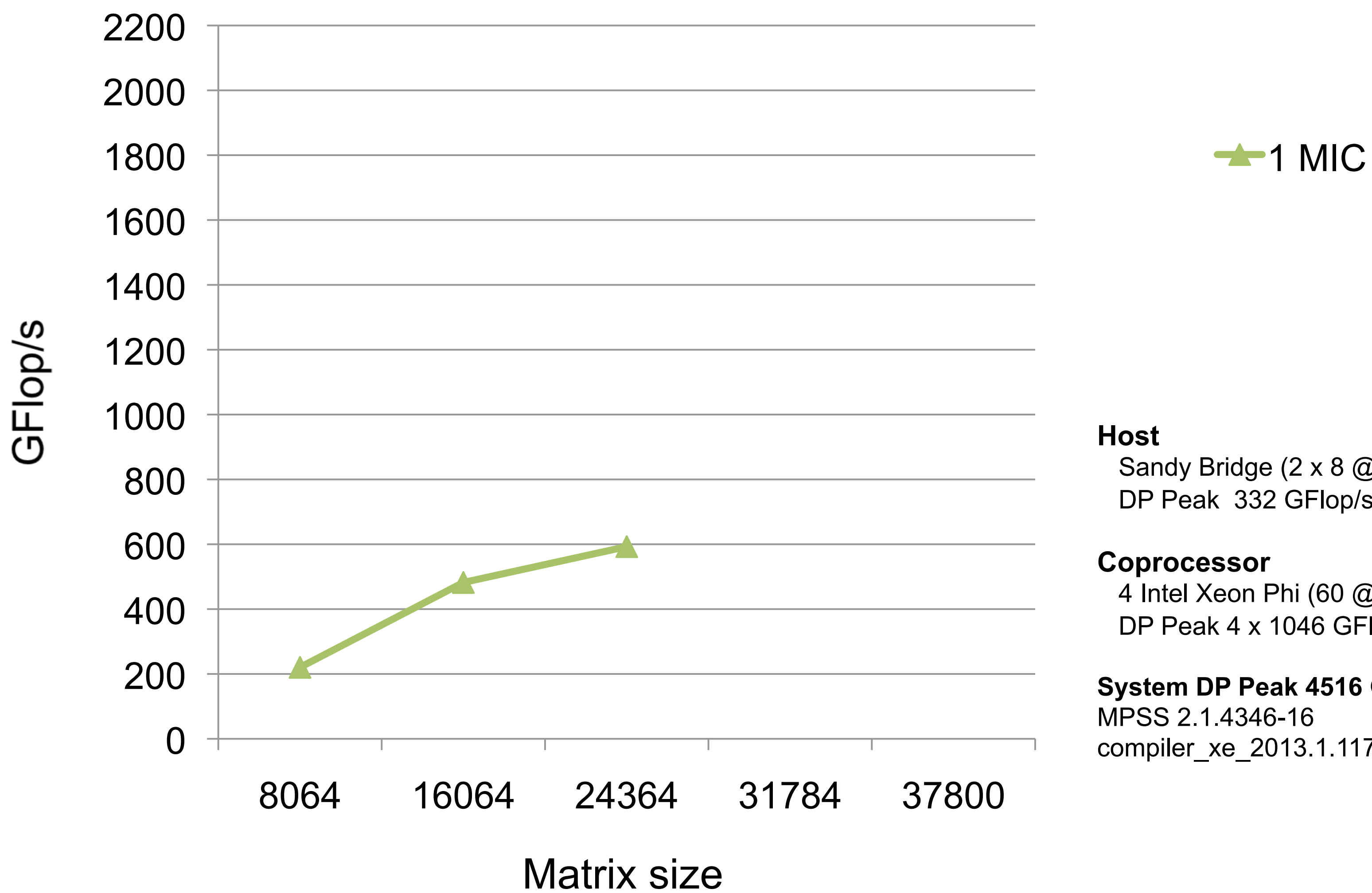
System DP Peak 1378 GFlop/s
MPSS 2.1.4346-16
compiler_xe_2013.1.117

From Single to MultiMIC Support

- Data distribution
  - 1-D block-cyclic distribution
- Algorithm
  - MIC holding current panel is sending it to CPU
  - All updates are done in parallel on the MICs
  - Look-ahead is done with MIC holding the next panel



MAGMA MIC Scalability
LU Factorization Performance in DP

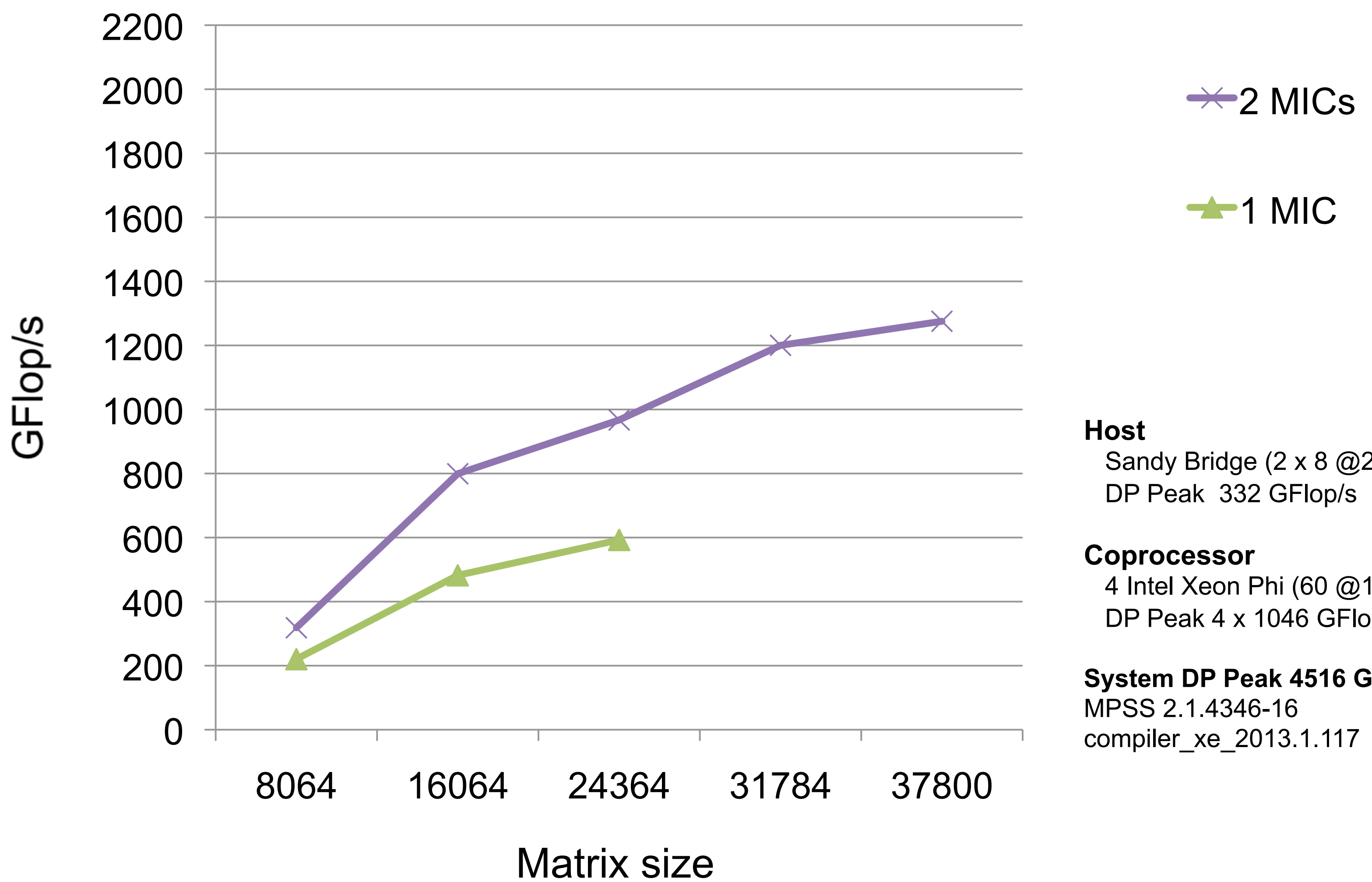


Host
Sandy Bridge (2 x 8 @2.6 GHz)
DP Peak 332 GFlop/s

Coprocessor
4 Intel Xeon Phi (60 @1.09 GHz)
DP Peak 4 x 1046 GFlop/s

System DP Peak 4516 GFlop/s
MPSS 2.1.4346-16
compiler_xe_2013.1.117

MAGMA MIC Scalability
LU Factorization Performance in DP

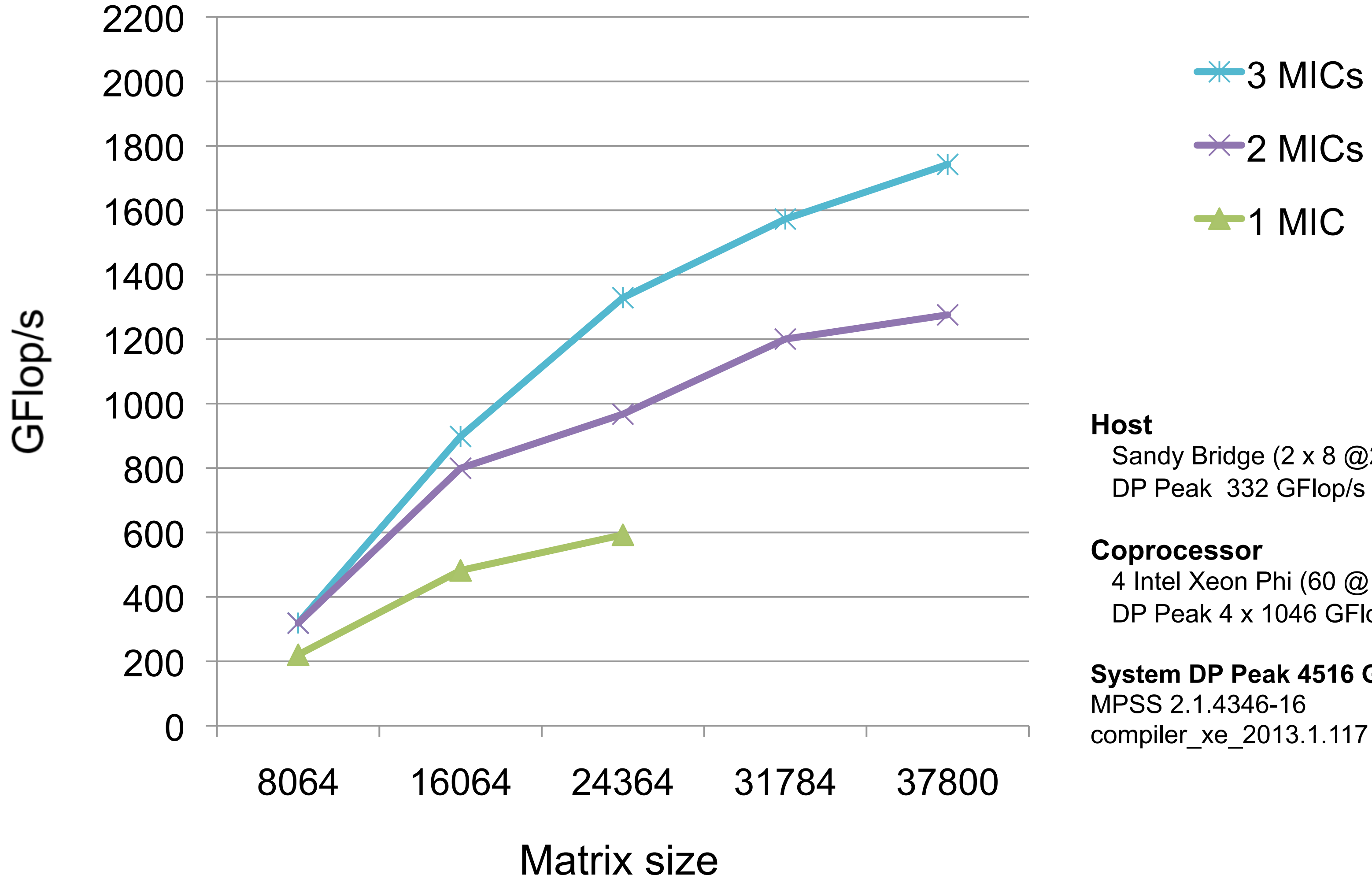


Host
Sandy Bridge (2 x 8 @2.6 GHz)
DP Peak 332 GFlop/s

Coprocessor
4 Intel Xeon Phi (60 @1.09 GHz)
DP Peak 4 x 1046 GFlop/s

System DP Peak 4516 GFlop/s
MPSS 2.1.4346-16
compiler_xe_2013.1.117

MAGMA MIC Scalability
LU Factorization Performance in DP

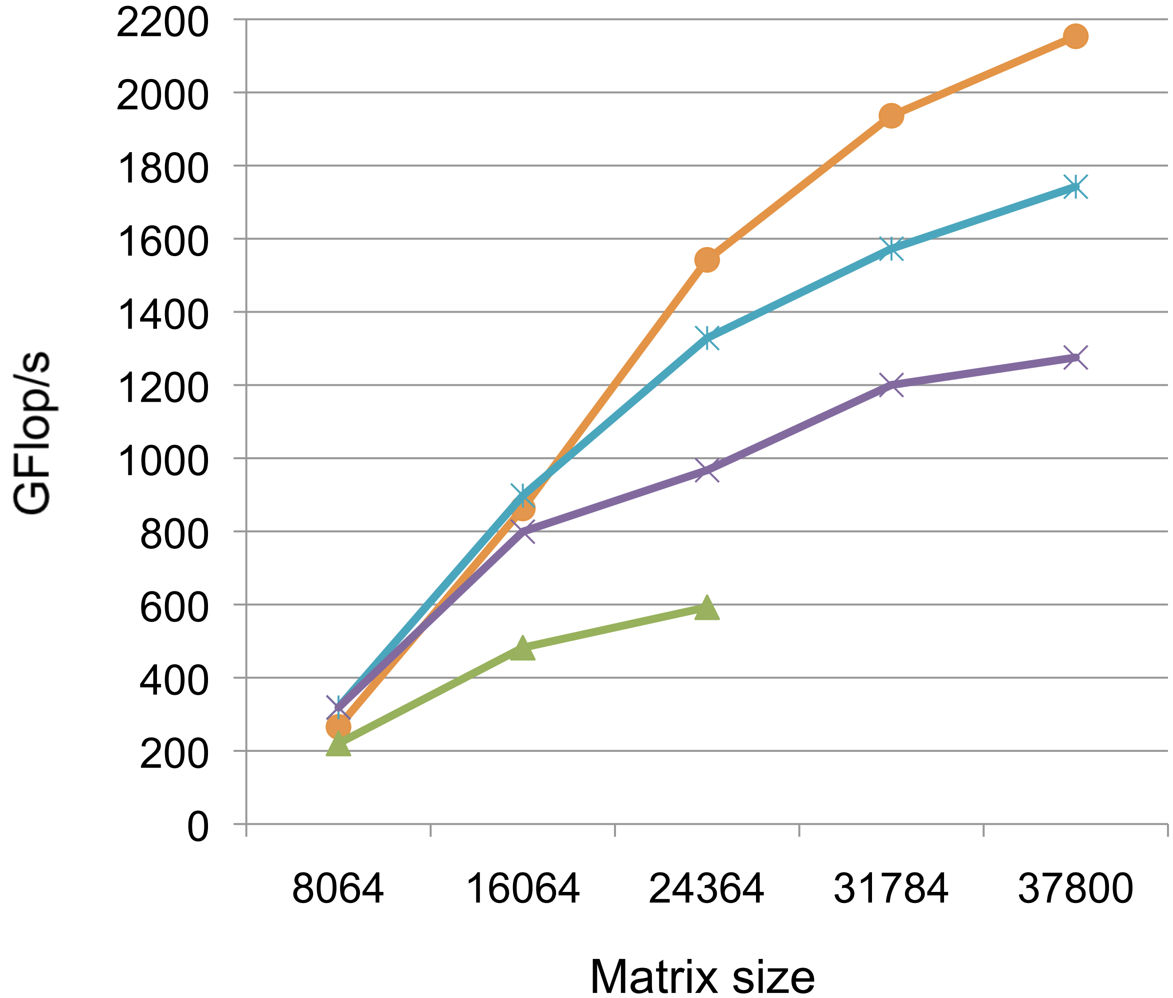


Host
Sandy Bridge (2 x 8 @2.6 GHz)
DP Peak 332 GFlop/s

Coprocessor
4 Intel Xeon Phi (60 @1.09 GHz)
DP Peak 4 x 1046 GFlop/s

System DP Peak 4516 GFlop/s
MPSS 2.1.4346-16
compiler_xe_2013.1.117

MAGMA MIC Scalability
LU Factorization Performance in DP



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

Coprocessor
4 Intel Xeon Phi (60 @1.09 GHz)
DP Peak 4 x 1046 GFlop/s

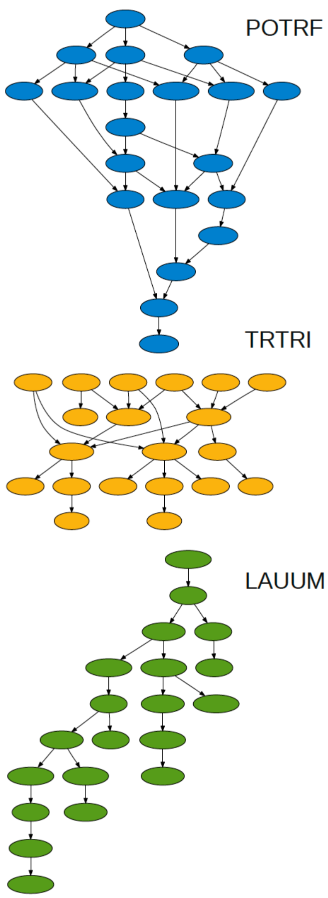
System DP Peak 4516 GFlop/s
MPSS 2.1.4346-16
compiler_xe_2013.1.117

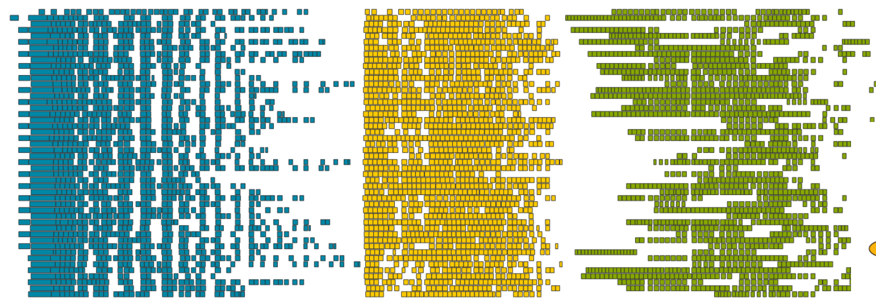
Current and Future MIC-specific directions

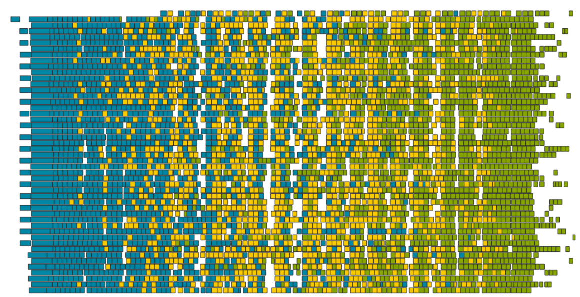
- Explore the use of tasks of lower granularity on MIC
  [GPU tasks in general are large, data parallel]
- Reduce synchronizations
  [less fork-join synchronizations]
- Scheduling of less parallel tasks on MIC
  [on GPUs, these tasks are typically offloaded to CPUs]
  [to reduce CPU-MIC communications]
- Develop more algorithms,
  porting newest developments in LA,
  while discovering further MIC-specific optimizations

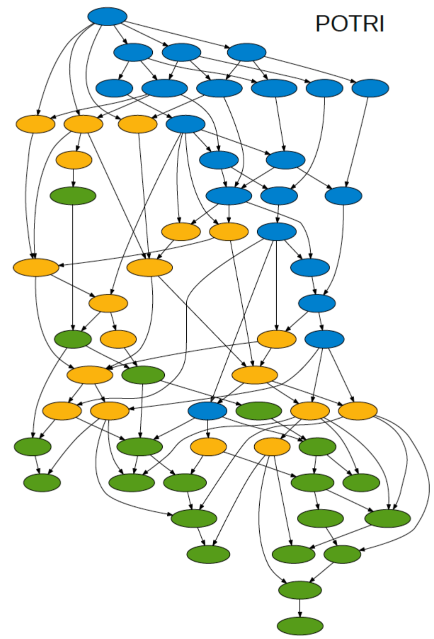
Current and Future MIC-specific directions

- Synchronization avoiding algorithms using **Dynamic Runtime Systems**









48 cores
POTRF, TRTRI and LAUUM.
The matrix is 4000 x 4000,tile size is 200 x 200

Current and Future MIC-specific directions

High-productivity w/ Dynamic Runtime Systems

From Sequential Nested-Loop Code to Parallel Execution

```
for (k = 0; k < min(MT, NT); k++){
   zgeqrt(A[k;k], ...);
   for (n = k+1; n < NT; n++)
    zunmqr(A[k;k], A[k;n], ...);
   for (m = k+1; m < MT; m++){
    ztsqrt(A[k;k],,A[m;k], ...);
    for (n = k+1; n < NT; n++)
        ztsmqr(A[m;k], A[k;n], A[m;n], ...);
   }
}
```

Current and Future MIC-specific directions

High-productivity w/ Dynamic Runtime Systems

From Sequential Nested-Loop Code to Parallel Execution

```
for (k = 0; k < min(MT, NT); k++){
   starpu_Insert_Task(&cl_zgeqrt, k , k, ...);
   for (n = k+1; n < NT; n++)
    starpu_Insert_Task(&cl_zunmqr, k, n, ...);
   for (m = k+1; m < MT; m++){
    starpu_Insert_Task(&cl_ztsqrt, m, k, ...);
    for (n = k+1; n < NT; n++)
        starpu_Insert_Task(&cl_ztsmqr, m, n, k, ...);
   }
}
```

Contact

Jack Dongarra
dongarra@eecs.utk.edu
Innovative Computing Laboratory
University of Tennessee, Knoxville

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

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

MAGMA

PLASMA

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