clMAGMA

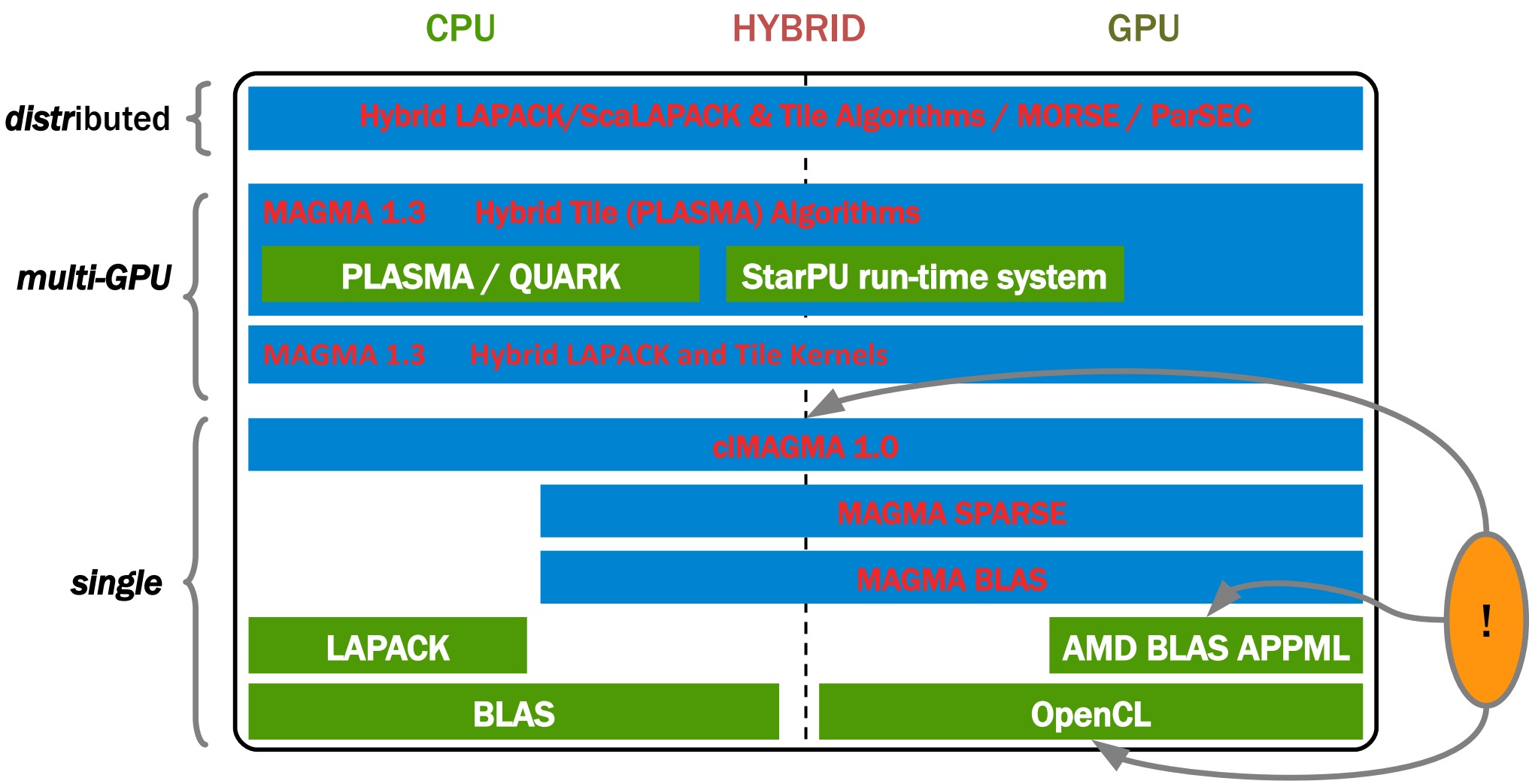
MAGMA on top of OpenCL

Innovative Computing Laboratory
Electrical Engineering and Computer Science
University of Tennessee

Piotr Luszczek (presenter)

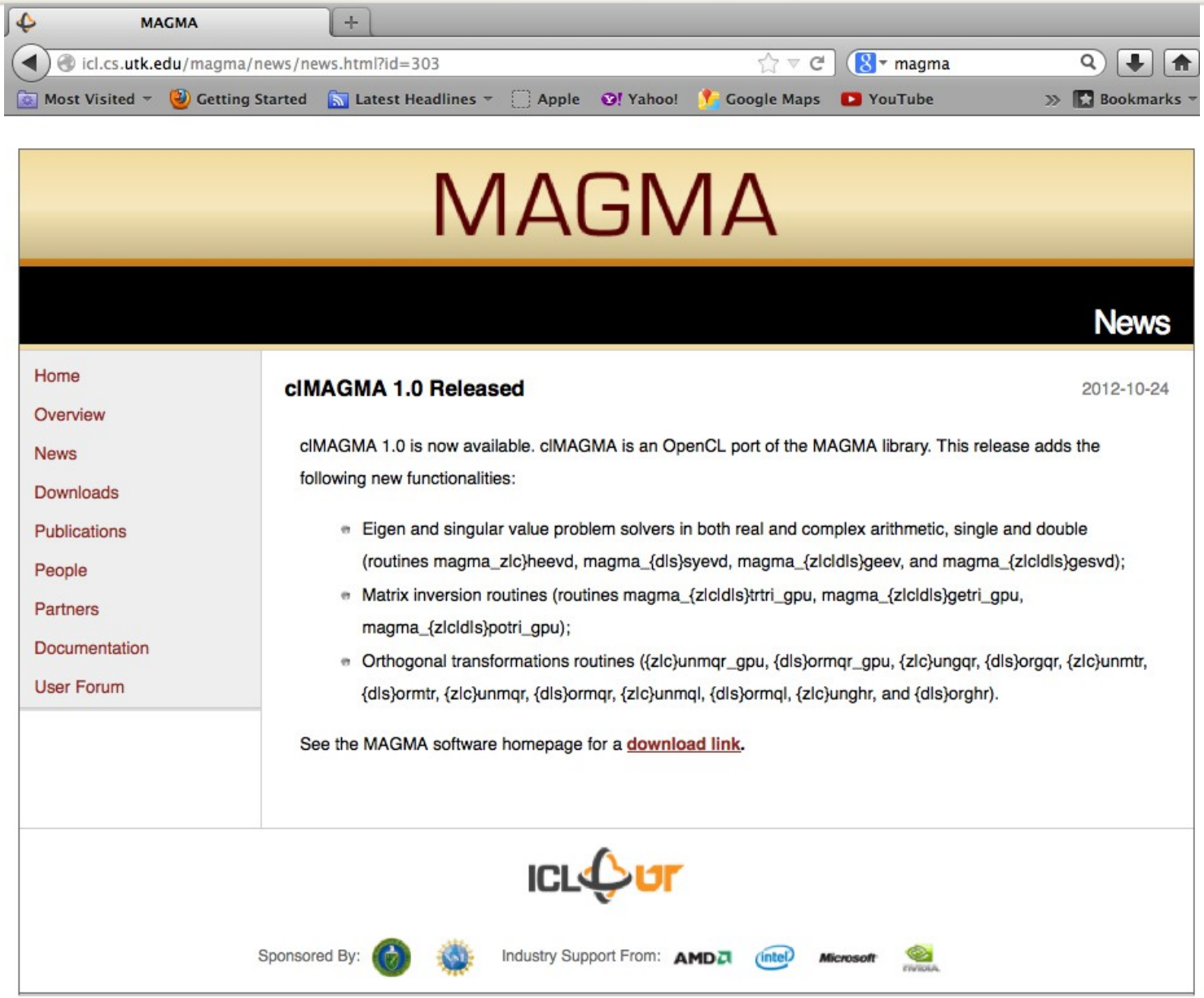
web.eecs.utk.edu/~luszczek/conf/

MAGMA Software Stack



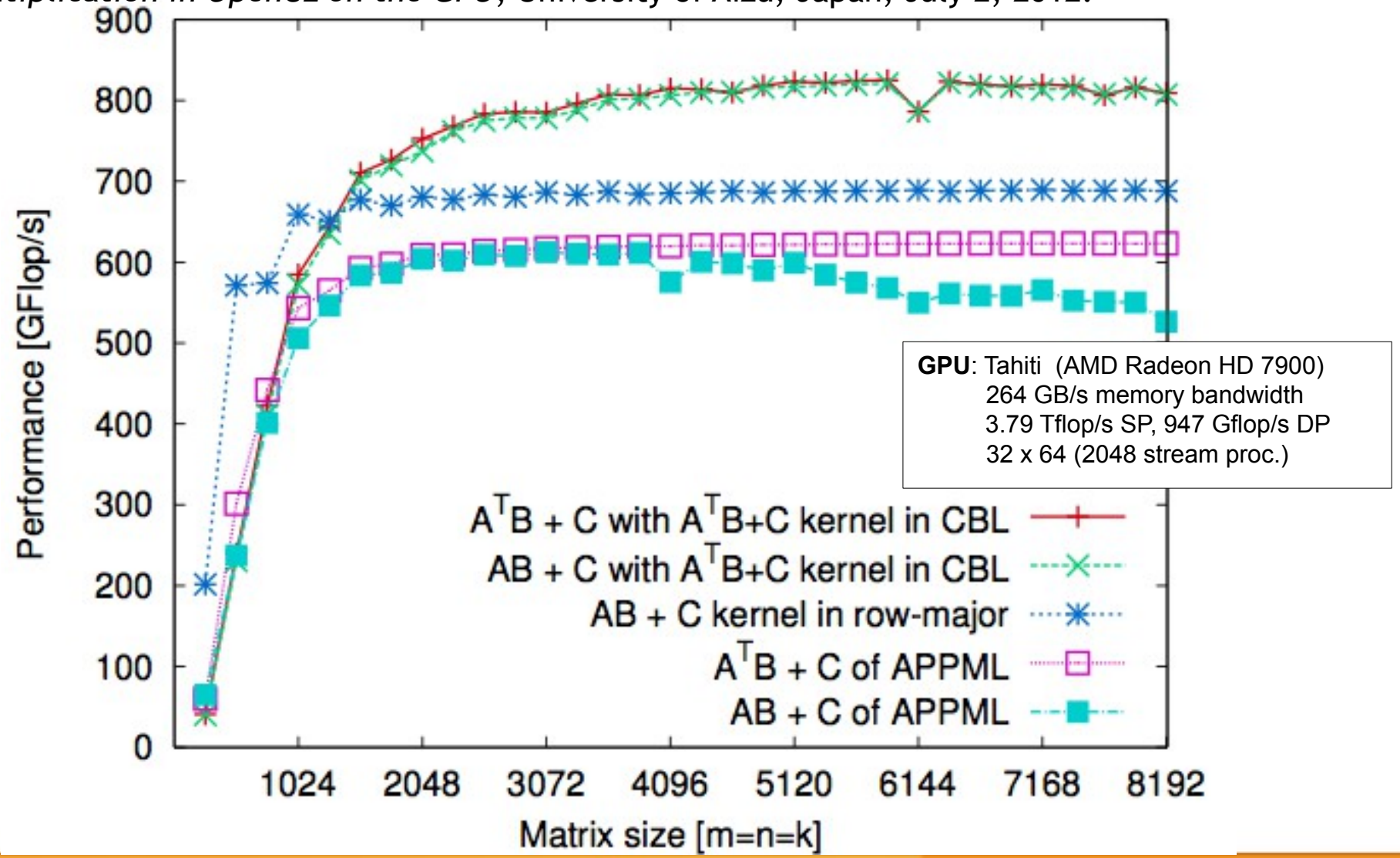
- *Linux, ~~Windows, Mac OS X~~*
- *C/C++, ~~Fortran~~*
- *~~Matlab, Python~~*

clMAGMA 1.0


MAGMA
icl.cs.utk.edu/magma/news/news.html?id=303
magma
Most Visited
Getting Started
Latest Headlines
Apple
Yahoo!
Google Maps
YouTube
Bookmarks
MAGMA
News
Home
Overview
News
Downloads
Publications
People
Partners
Documentation
User Forum
clMAGMA 1.0 Released
2012-10-24
clMAGMA 1.0 is now available. clMAGMA is an OpenCL port of the MAGMA library. This release adds the following new functionalities:
Eigen and singular value problem solvers in both real and complex arithmetic, single and double (routines magma_zlc}heevd, magma_{dls}syevd, magma_{zlcldls}geev, and magma_{zlcldls}gesvd);
Matrix inversion routines (routines magma_{zlcldls}trtri_gpu, magma_{zlcldls}getri_gpu, magma_{zlcldls}potri_gpu);
Orthogonal transformations routines ({zlc}unmqr_gpu, {dls}ormqr_gpu, {zlc}ungqr, {dls}orgqr, {zlc}unmtr, {dls}ormtr, {zlc}unmqr, {dls}ormqr, {zlc}unmql, {dls}ormql, {zlc}unghr, and {dls}orghr).
See the MAGMA software homepage for a download link.
ICL UT
Sponsored By:
Industry Support From:
AMD
intel
Microsoft
NVIDIA

DGEMM in OpenCL

Kazuya Matsumoto, Naohito Nakasato, Stanislav G.Sedukhin, *Implementing a Code Generator for Fast Matrix Multiplication in OpenCL on the GPU*, University of Aizu, Japan, July 2, 2012.



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, queue );
      magma_zgetmatrix_async( jb, jb, dA(j,j), ldda, work, 0, jb, queue, &event );
      if ( j+jb < n )
          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 );
      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, &event );
      if ( j+jb < n ) {
          magma_event_sync( event );
          magma_ztrsm( MagmaLeft, MagmaUpper, MagmaConjTrans, MagmaNonUnit,
                               jb, n-j-jb, z_one, dA(j, j), ldda, dA(j, j+jb), ldda, queue );
      }
}
```

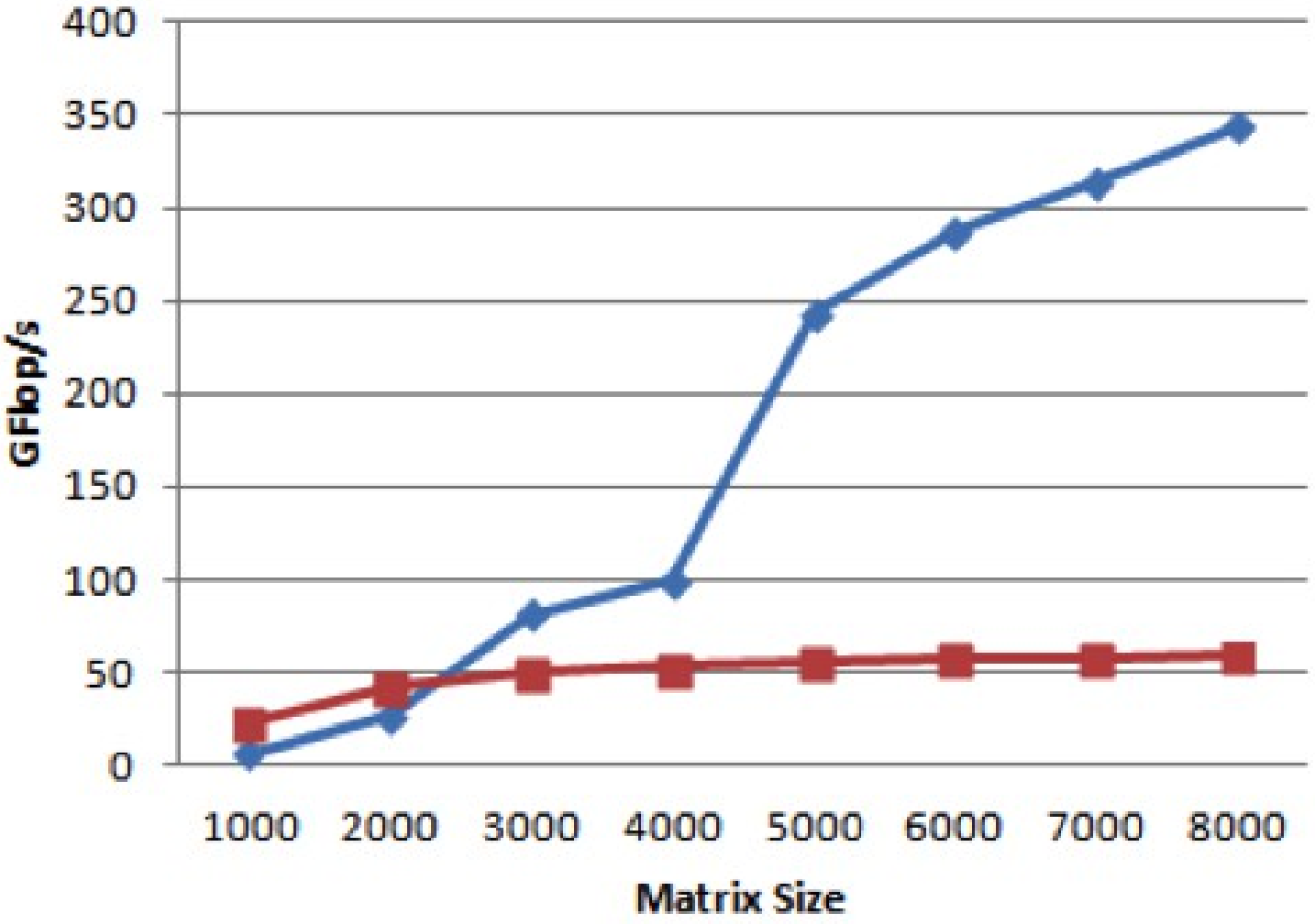
OpenCL interface - communications

```
magma_err_t
magma_zgetmatrix_async(
    magma_int_t m, magma_int_t n,
    magmaDoubleComplex_const_ptr dA_src, size_t dA_offset, ma
    magmaDoubleComplex*          hA_dst, size_t hA_offset, ma
    magma_queue_t queue, magma_event_t *event )
{
    size_t buffer_origin[3] = { dA_offset*sizeof(magmaDoubleC
    size_t host_orig[3]     = { 0, 0, 0 };
    size_t region[3]        = { m*sizeof(magmaDoubleComplex),
    cl_int err = clEnqueueReadBufferRect(
        queue, dA_src, CL_FALSE,  // non-blocking
        buffer_origin, host_orig, region,
        ldda*sizeof(magmaDoubleComplex), 0,
        ldha*sizeof(magmaDoubleComplex), 0,
        hA_dst, 0, NULL, event );
```

OpenCL interface - AMD APPML BLAS

```
mag
magma_zherk(
    magma_uplo_t uplo, magma_trans_t trans,
    magma_int_t n, magma_int_t k,
    double alpha, magmaDoubleComplex_const_ptr dA, size_t dA_
    double beta,  magmaDoubleComplex_ptr       dC, size_t dC_
    magma_queue_t queue )
{
    cl_int err = clAmdBlasZherk(
        clAmdBlasColumnMajor,
        amdblas_uplo_const( uplo ),
        amdblas_trans_const( trans ),
        n, k,
        alpha, dA, dA_offset, lda,
        beta,  dC, dC_offset, ldc,
        1, &queue, 0, NULL, NULL );
    return err;
}
```

clMAGMA Cholesky Factorization (real64)



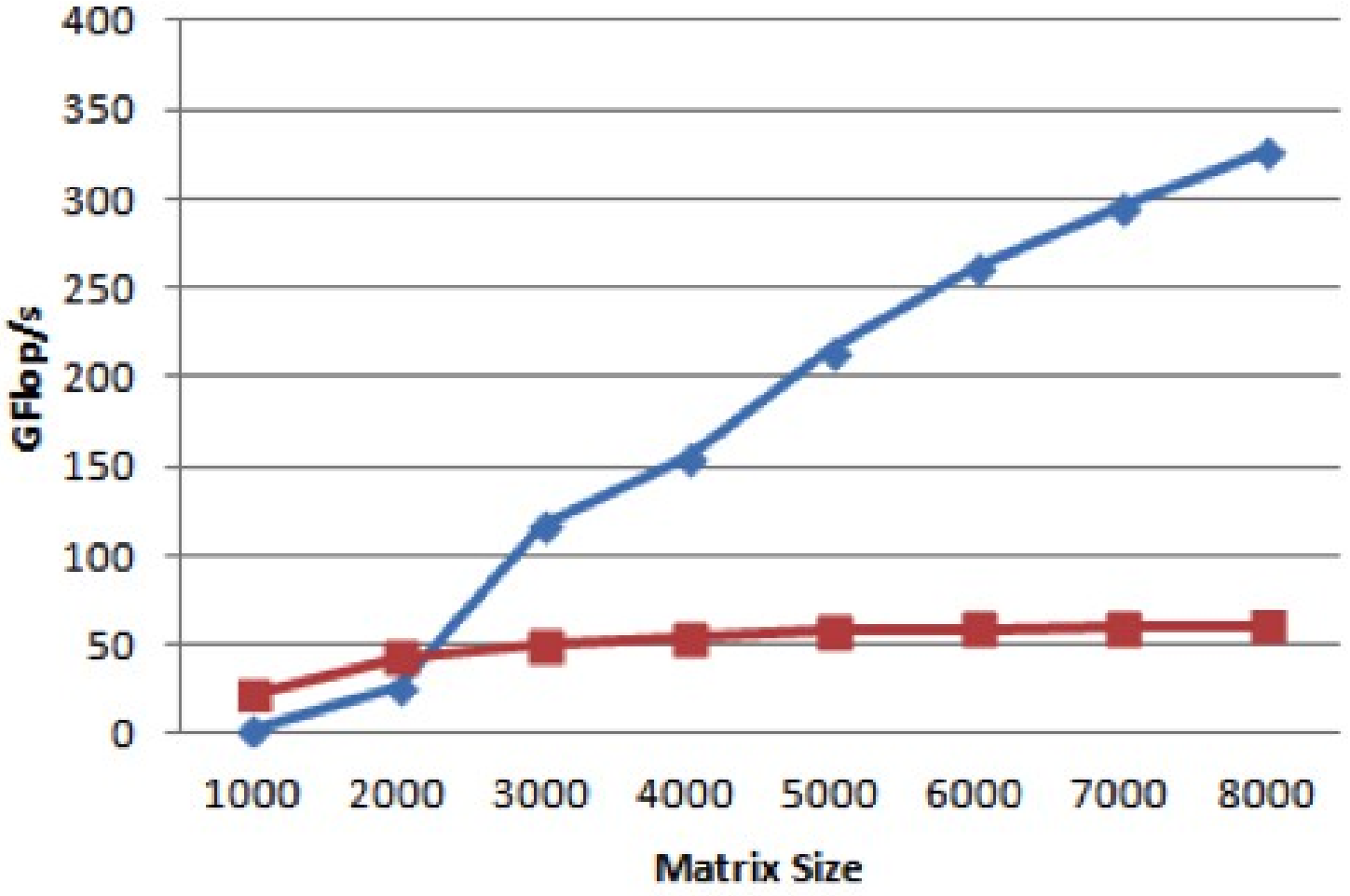
clMAGMA

MKL11.1

GPU:
Tahiti (AMD Radeon 7970)
947 GFlop/s DP

CPU:
One socket six-core AMD Phenom II X6 1100T @3.71GHz

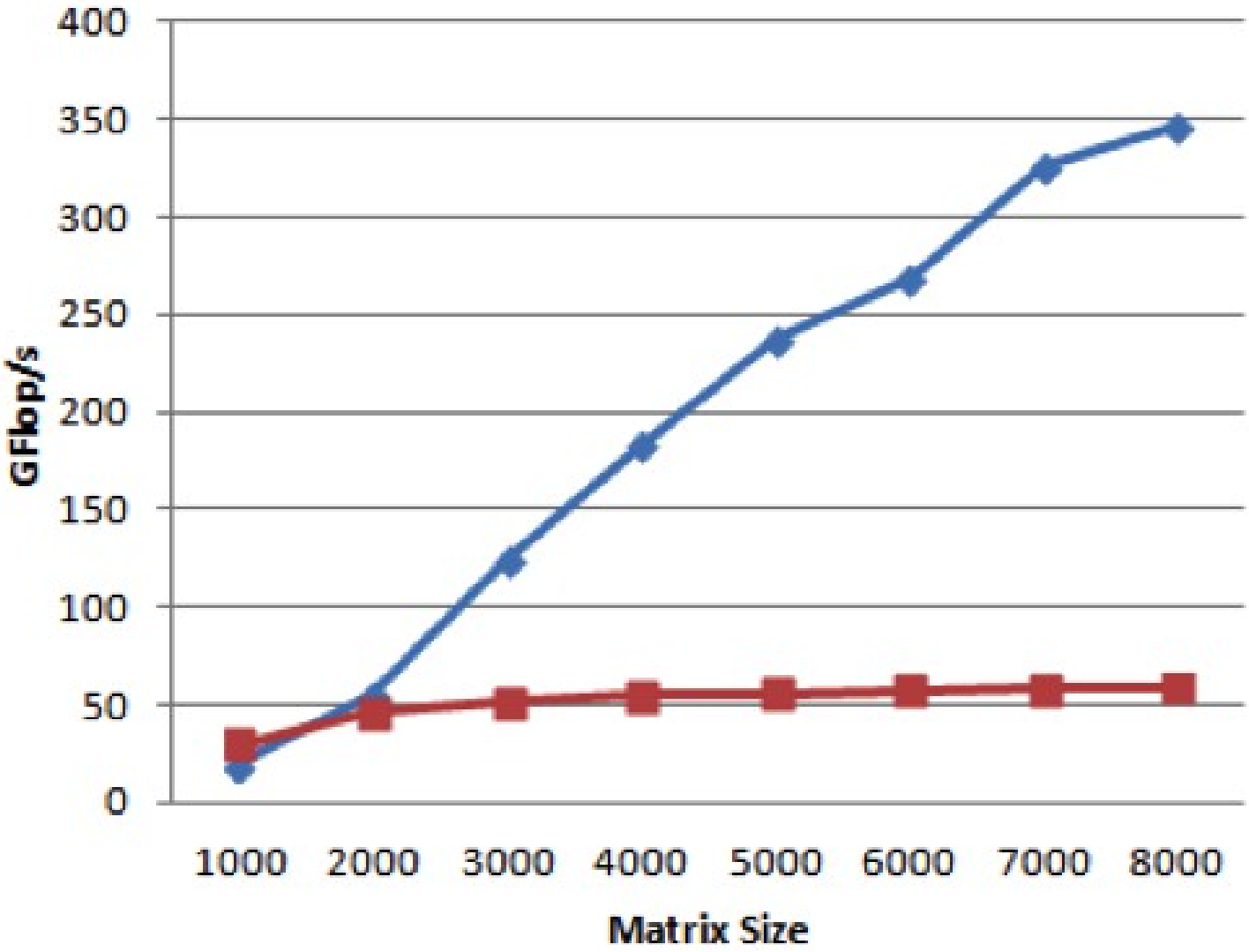
clMAGMA LU Factorization (real64)



GPU:
Tahiti (AMD Radeon 7970)
947 GFlop/s DP

CPU:
One socket six-core AMD Phenom II X6 1100T @3.71GHz

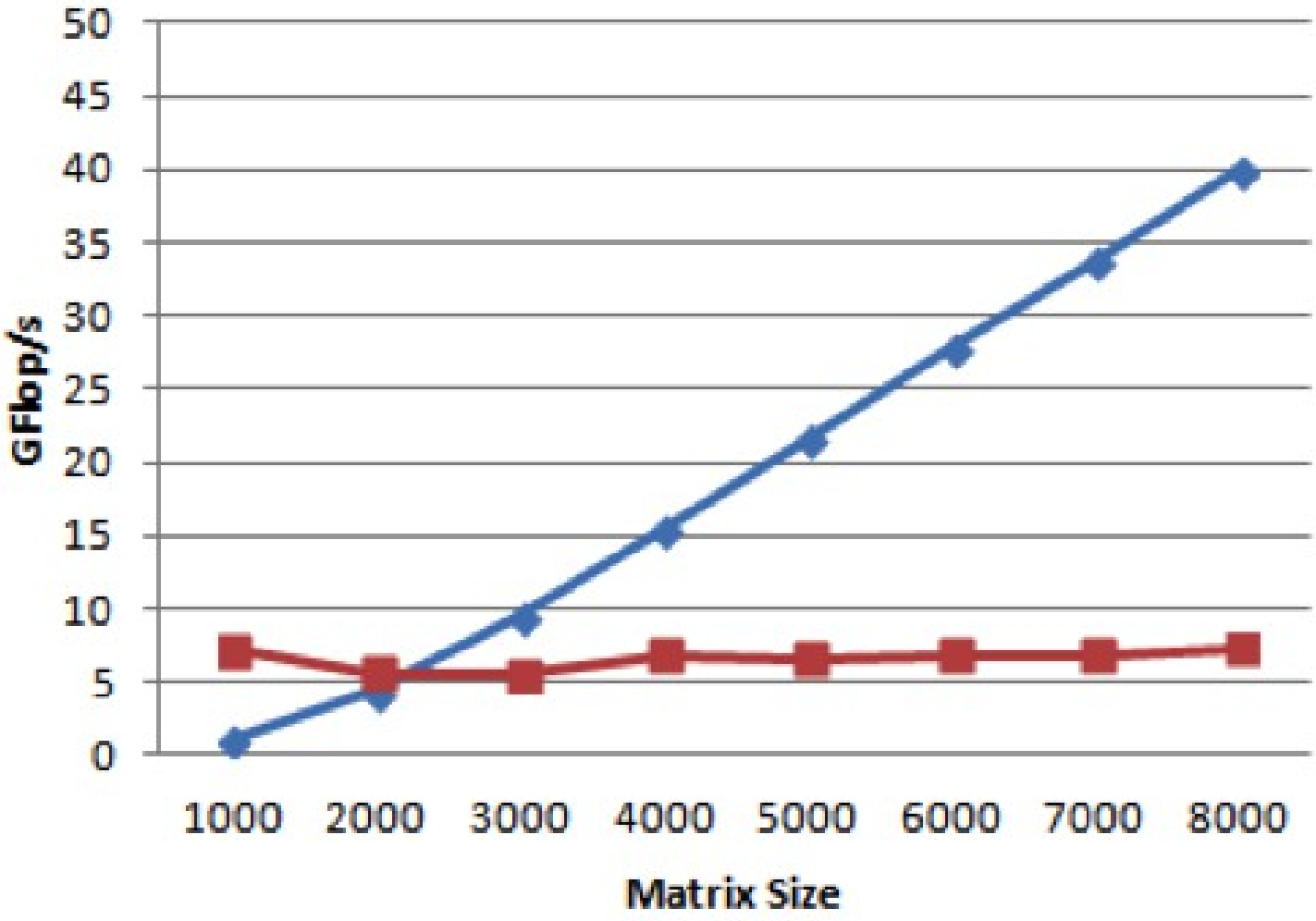
clMAGMA QR Factorization (real64)



GPU:
Tahiti (AMD Radeon 7970)
947 GFlop/s DP

CPU:
One socket six-core AMD Phenom II X6 1100T @3.71GHz

clMAGMA Hessenberg Reduction (real64)

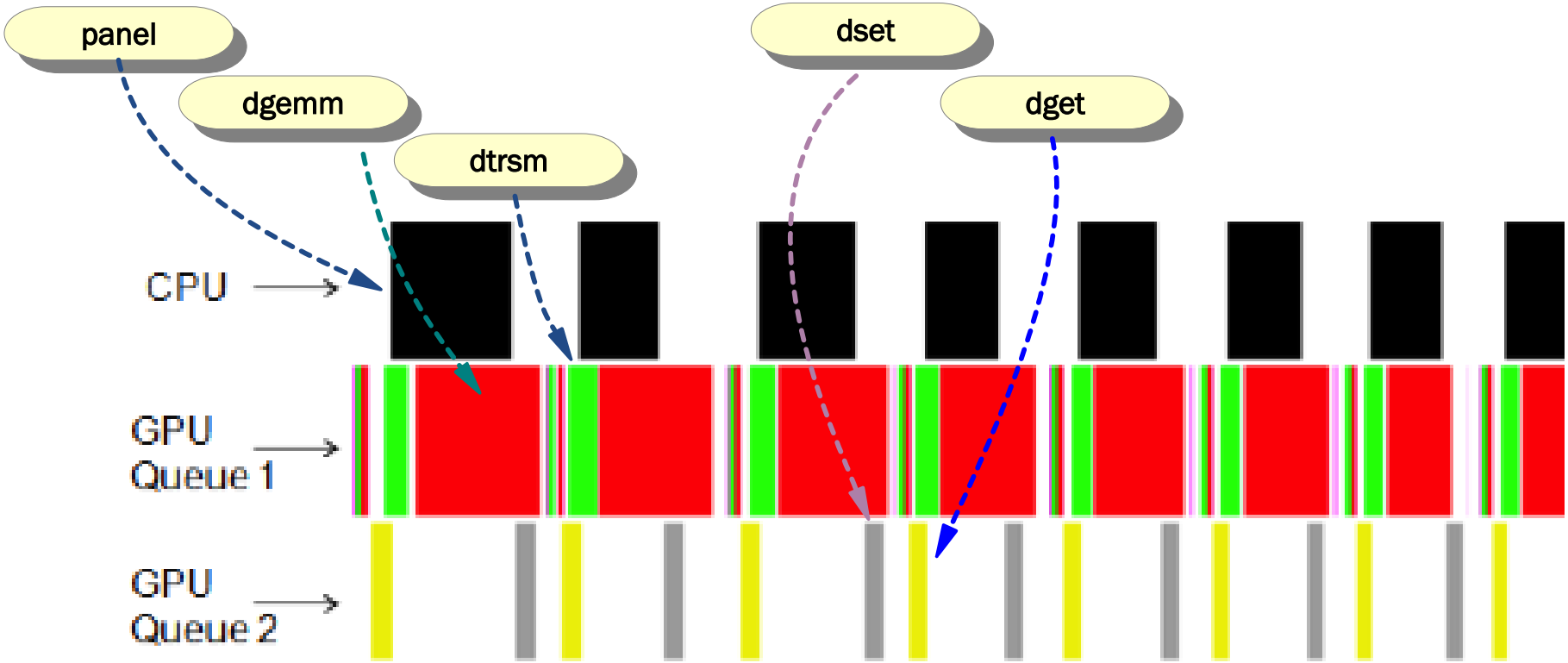


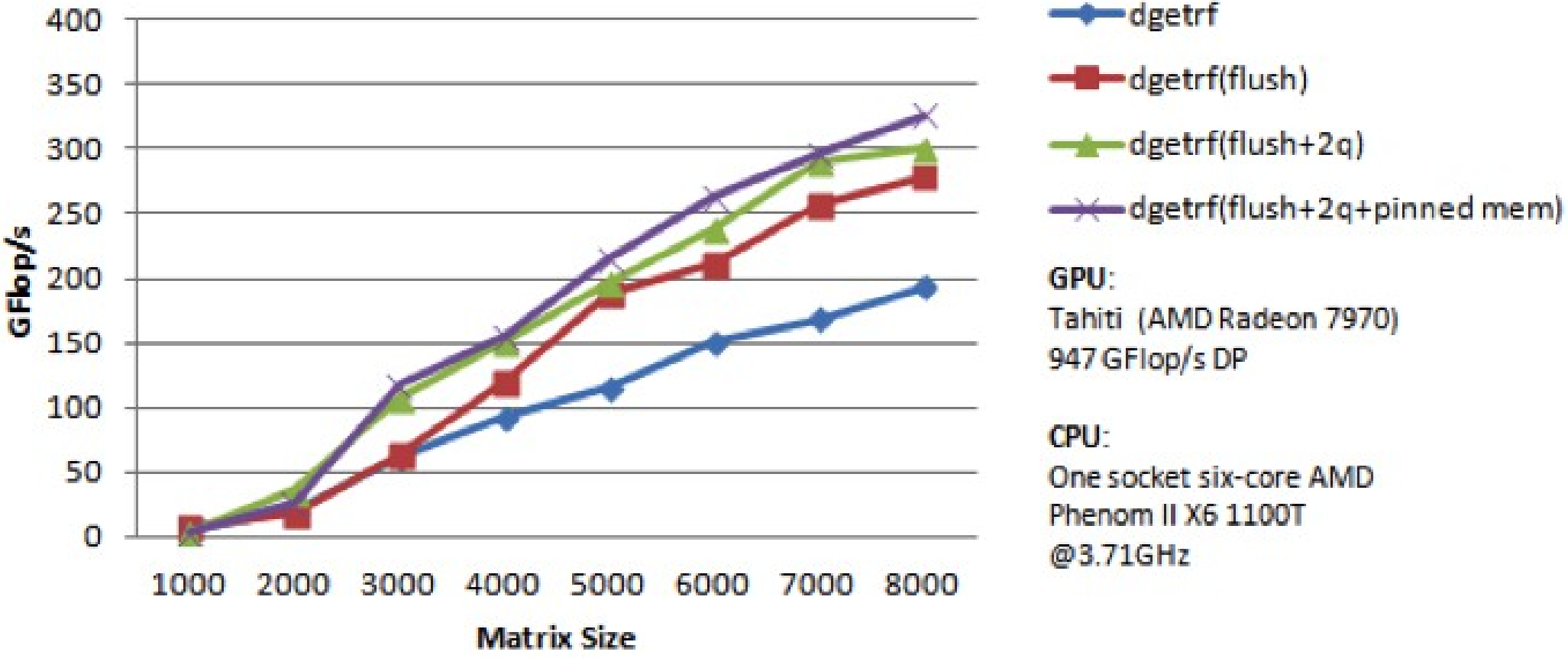
GPU:
Tahiti (AMD Radeon 7970)
947 GFlop/s DP

CPU:
One socket six-core AMD Phenom II X6 1100T @3.71GHz

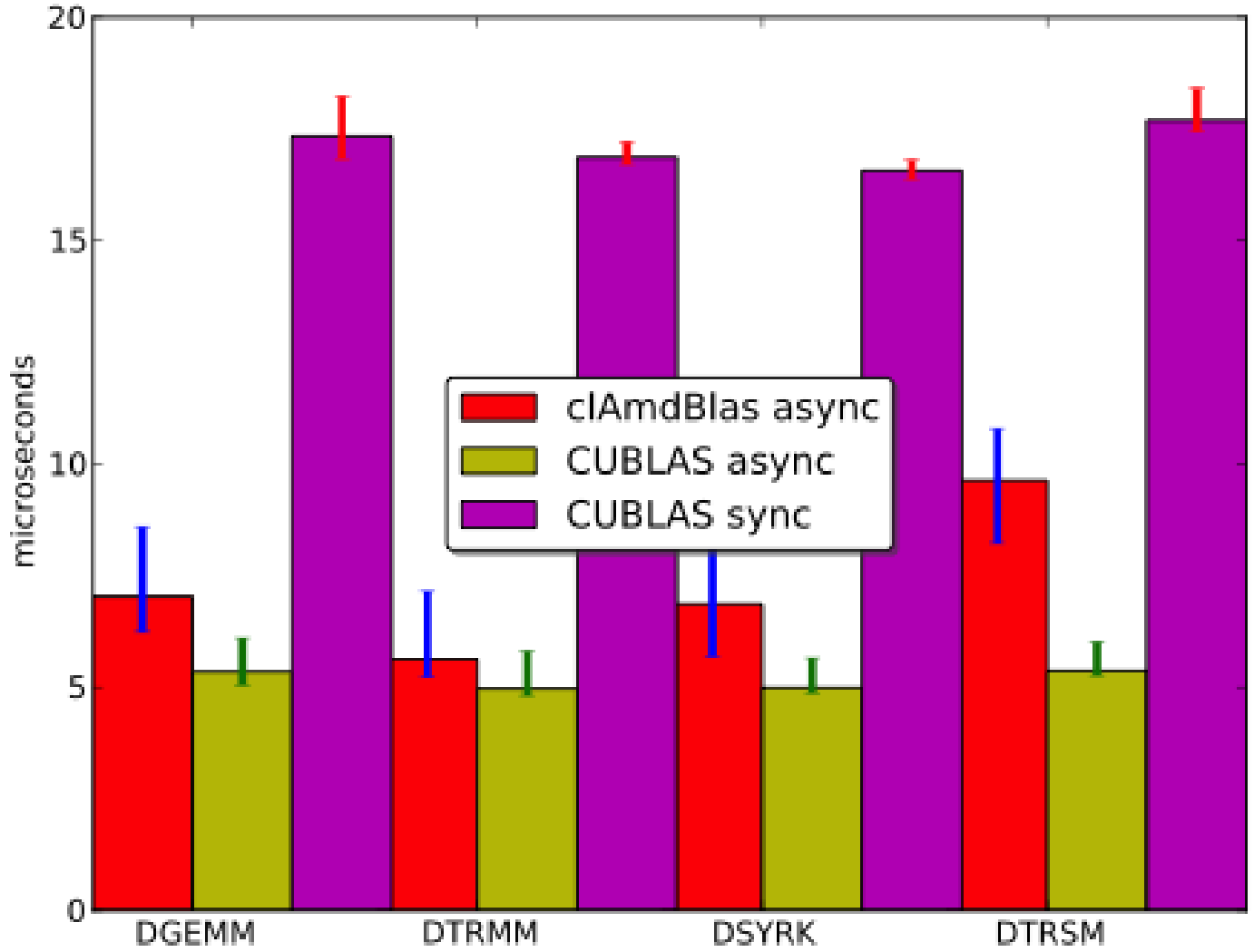
Performance Optimization Based on Traces

- CPU-GPU communications
- A dgetrf trace example:




400
350
300
250
200
150
100
50
0
GFlop/s
1000
2000
3000
4000
5000
6000
7000
8000
Matrix Size
dgetrf
dgetrf(flush)
dgetrf(flush+2q)
dgetrf(flush+2q+pinned mem)
GPU:
Tahiti (AMD Radeon 7970)
947 GFlop/s DP
CPU:
One socket six-core AMD
Phenom II X6 1100T
@3.71GHz

OpenCL-specific optimizations



Benchmarks to discover OpenCL specifics

Compute Panels entirely on GPU?

- Important to have for both
  - Dense, certain sparse linear system, and
  - eigen-problem solvers
- Can we factor panels faster on GPU?
  - Panels are memory bound
- Latencies may be a bottleneck
  - 64-column panel requires the invocation of about 400 kernels
- Performance of QR panels in double precision on:
  - Kepler (in CUDA)
  - Tahiti (in OpenCL), and
  - 16 Intel Sandy Bridge cores

M	N	CUDA	OpenCL	Sandy
10^3	64	5	94	9
10^4	64	7	104	17
10^5	64	36	131	89
10^6	64	365	528	1431

Difference is due to latencies (in our software and/or hardware configuration) as shown by increasing the problem size.