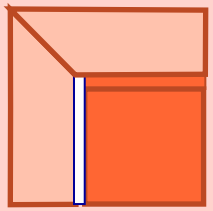
Redesigning Dense Linear Algebra Software for ManyCore and Beyond

Jack Dongarra
University of Tennessee
and
Oak Ridge National Laboratory

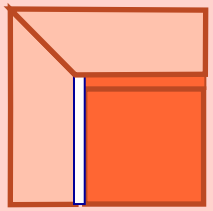
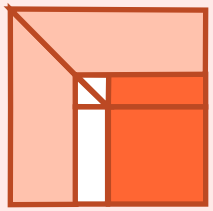
Major Changes to Software

- Must rethink the design of our software
  - Another disruptive technology
    - Similar to what happened with cluster computing and message passing
  - Rethink and rewrite the applications, algorithms, and software
- Numerical libraries for example will change
  - For example, both LAPACK and ScaLAPACK will undergo major changes to accommodate this

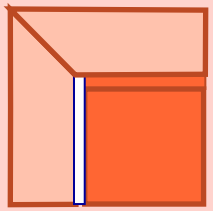
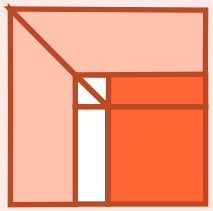
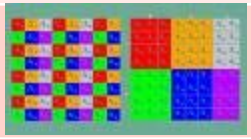
A New Generation of Software:

Software/Algorithms follow hardware evolution in time		
LINPACK (70's) (Vector operations)		Rely on - Level-1 BLAS operations

A New Generation of Software:

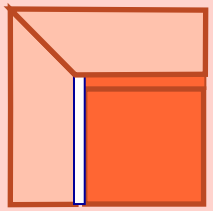
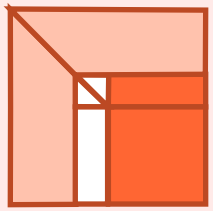
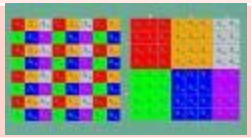
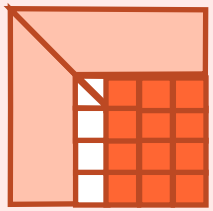
Software/Algorithms follow hardware evolution in time		
LINPACK (70's) (Vector operations)		Rely on - Level-1 BLAS operations
LAPACK (80's) (Blocking, cache friendly)		Rely on - Level-3 BLAS operations

A New Generation of 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

A New Generation of Software:

Parallel Linear Algebra Software for Multicore Architectures (PLASMA)

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

Those new algorithms
- have a very low granularity, they scale very well (multicore, petascale computing, …)
- removes a lots of dependencies among the tasks, (multicore, distributed computing)
- avoid latency (distributed computing, out-of-core)
- rely on fast kernels

Those new algorithms need new kernels and rely on efficient scheduling algorithms.

Steps in the LAPACK LU

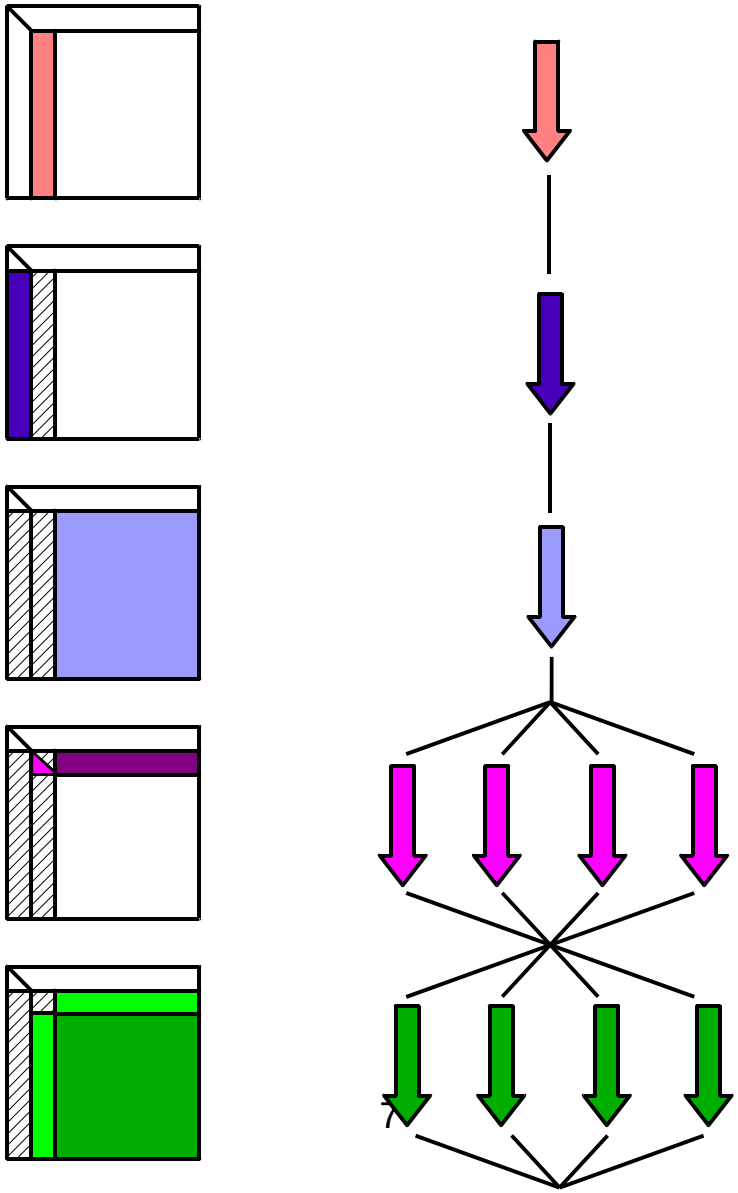
DGETF2
(Factor a panel)

DLSWP
(Backward swap)

DLSWP
(Forward swap)

DTRSM
(Triangular solve)

DGEMM
(Matrix multiply)



LAPACK

LAPACK

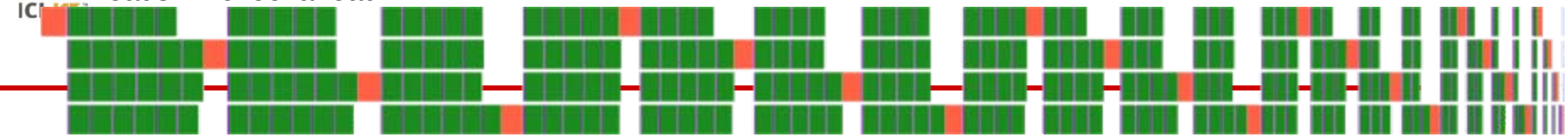
LAPACK

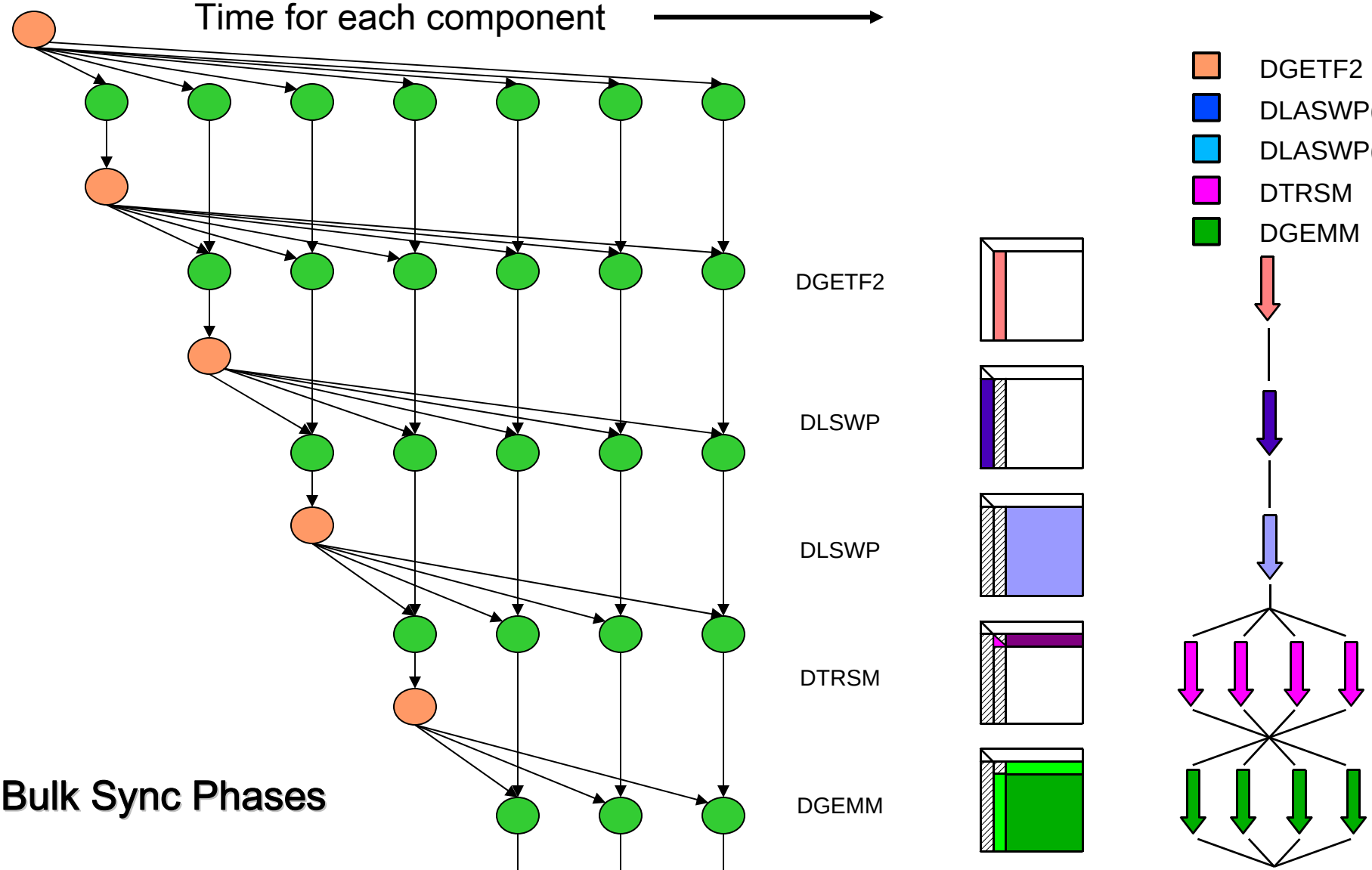
BLAS

BLAS

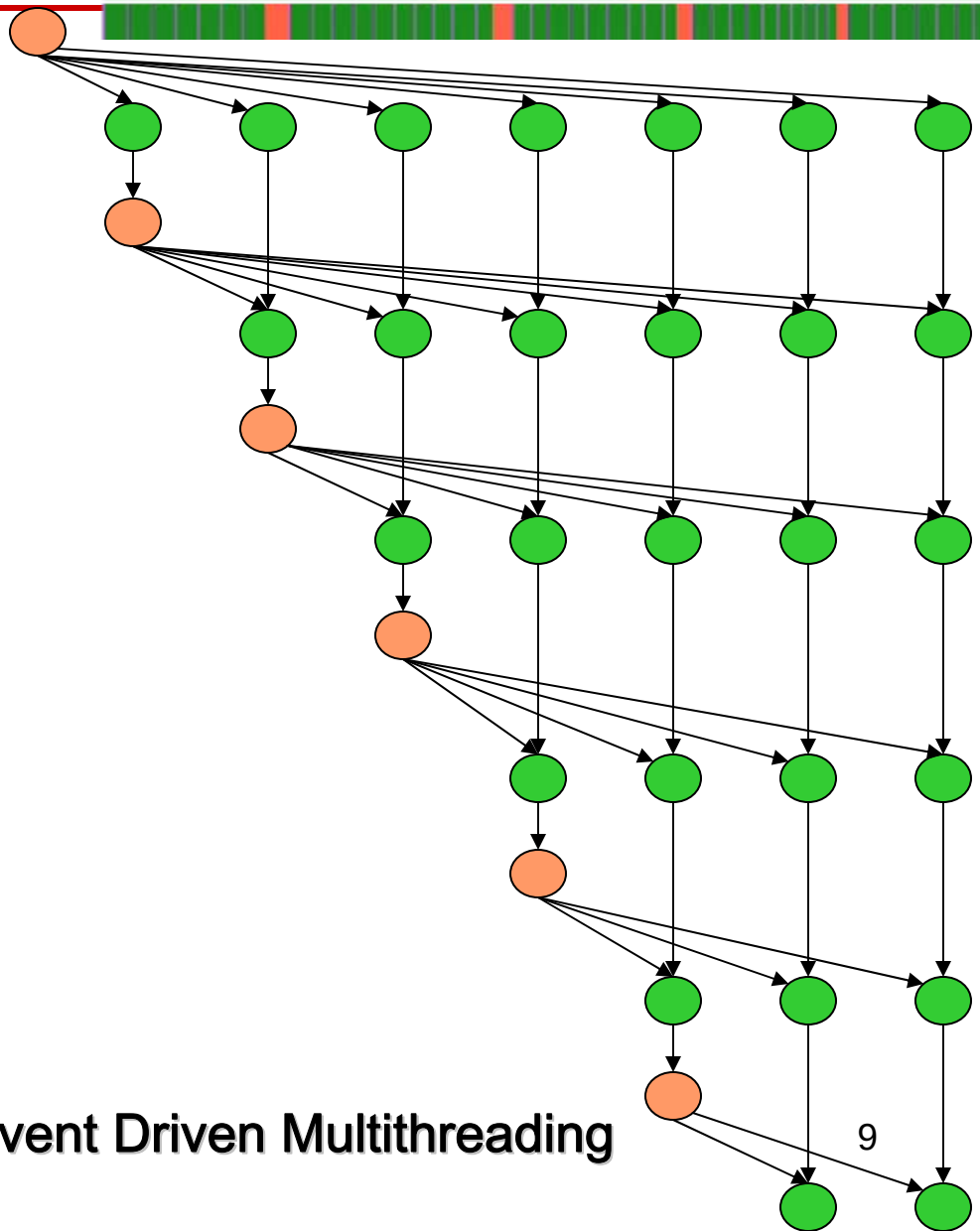
LU Timing Profile (4 processor system)

Threads – no lookahead





Adaptive Lookahead - Dynamic



```
while(1)
    fetch_task();
    switch(task.type) {
        case PANEL:
            dgetf2();
            update_progress();
        case COLUMN:
            dlaswp();
            dtrsm();
            dgemm();
            update_progress();
        case END:
            for()
                dlaswp();
            return;
    }
}
```

Reorganizing algorithms to use this approach

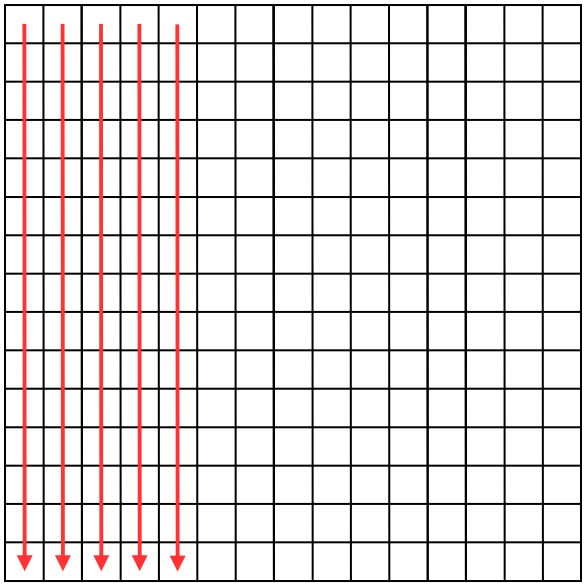
Redesign

- Asychronicity
  - Avoid fork-join (Bulk sync design)
- Dynamic Scheduling
  - Out of order execution
- Fine Granularity
  - Independent block operations
- Locality of Reference
  - Data storage – Block Data Layout

Achieving Fine Granularity

Fine granularity may require novel data formats to overcome the limitations of BLAS on small chunks of data.

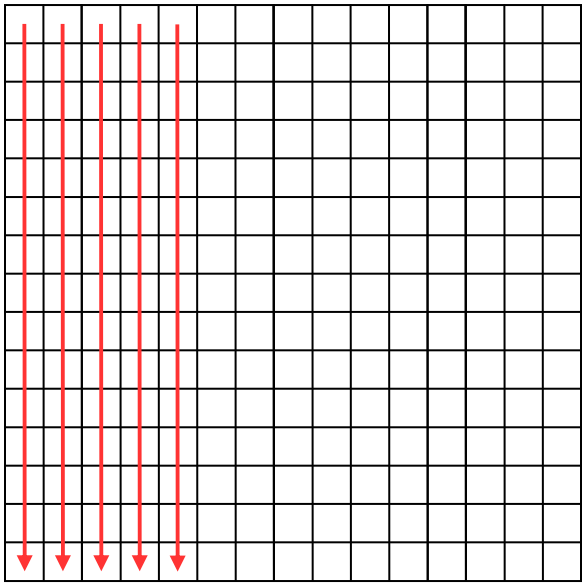
Column-Major



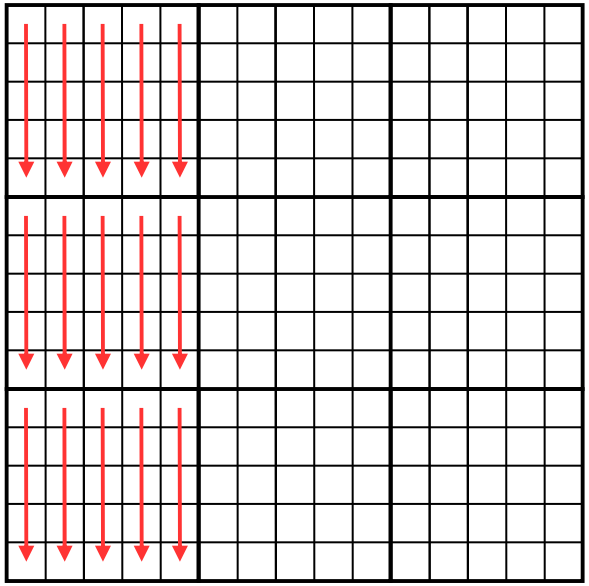
Achieving Fine Granularity

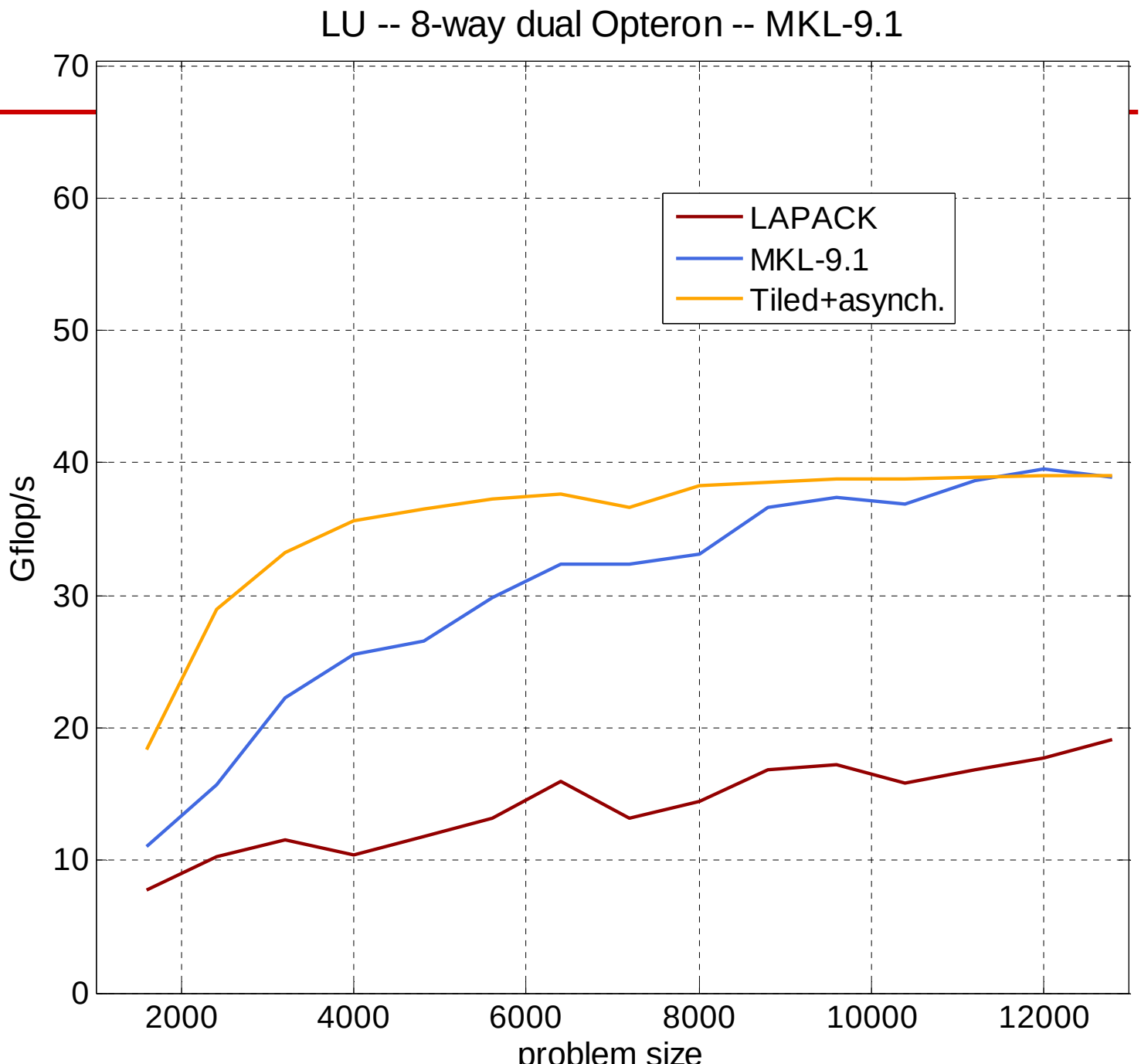
Fine granularity may require novel data formats to overcome the limitations of BLAS on small chunks of data.

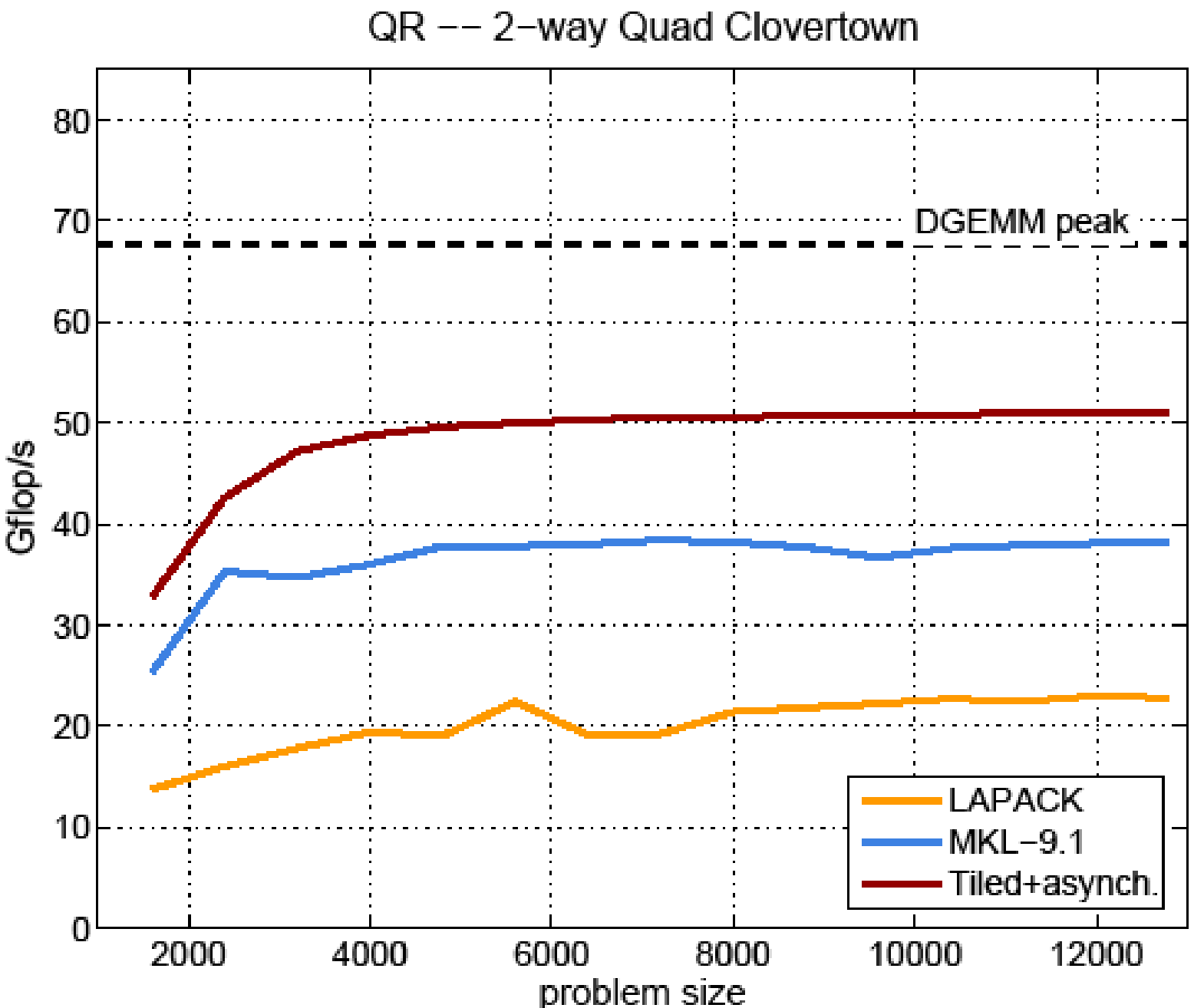
Column-Major



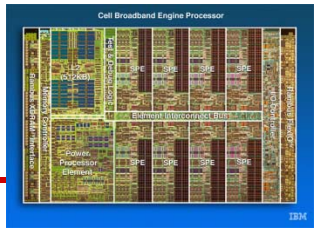
Block data layout

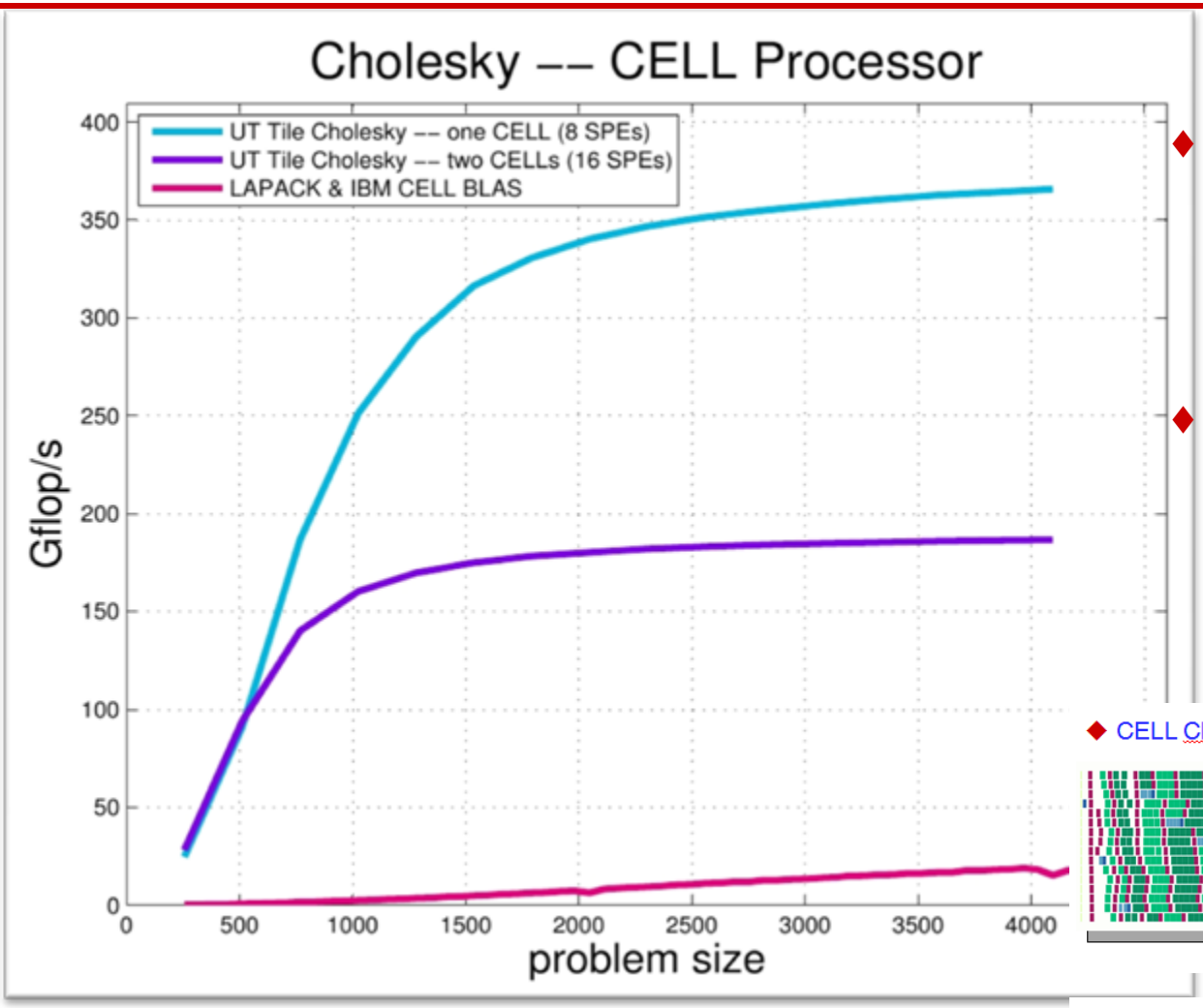



LU -- 8-way dual Opteron -- MKL-9.1
Gflop/s
problem size
0
10
20
30
40
50
60
70
2000
4000
6000
8000
10000
12000
LAPACK
MKL-9.1
Tiled+asynch.


QR -- 2-way Quad Clovertown
Gflop/s
problem size
DGEMM peak
LAPACK
MKL-9.1
Tiled+asynch.
0
10
20
30
40
50
60
70
80
2000
4000
6000
8000
10000
12000

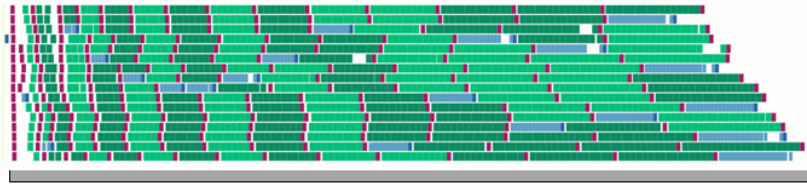
Cholesky on the CELL



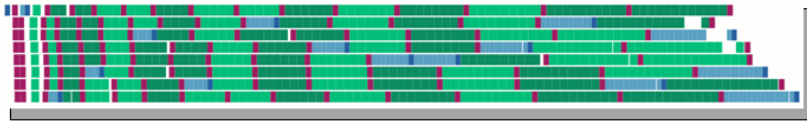


- 1 CELL (8 SPEs)
  - 186 Gflop/s
  - 91 % peak
  - 97 % SGEMM peak
- 2 CELLs (16 SPEs)
  - 365 Gflop/s
  - 89 % peak
  - 95 % SGEMM peak

- CELL Cholesky – 16 cores



- CELL Cholesky – 8 cores



Single precision results on the Cell

Next Steps

- Looking at implementations of the algorithms using:
  - Cilk, SMP SuperScalar, Intel Threading Building Blocks, UPC,…
- Distributed memory implementations will follow

Collaborators / Support

- U Tennessee, Knoxville
  - Julien Langou, Julie Langou, Piotr Luszczek, Jakub Kurzak, Stan Tomov, Remi Delmas, Peng Du
- UC Berkeley:
  - Jim Demmel, Ming Gu, W. Kahan, Beresford Parlett, Xiaoye Li, Osni Marques, Christof Voemel, David Bindel, Yozo Hida, Jason Riedy, Jianlin Xia, Jiang Zhu

Microsoft

Office of Science
U.S. DEPARTMENT OF ENERGY

The MathWorks