High-Performance Linear Algebra on the Intel KNL

Samuel D. Relton


s.d.relton@leeds.ac.uk


@sdrelton


www.samrelton.com


blog.samrelton.com

UNIVERSITY OF LEEDS

Who am I?

- Research Fellow in LIHS since July
  - Worsley building, Room 10.23
- High Performance Computing, Manchester Math Dept.
  - Optimising libraries for GPUs and Xeon Phi
  - Task-based programming models
  - Close collaboration with Jack Dongarra and Intel MKL team
- PhD in Numerical Analysis, Manchester Math Dept.
  - Algorithm design
  - Linear algebra
  - Error analysis

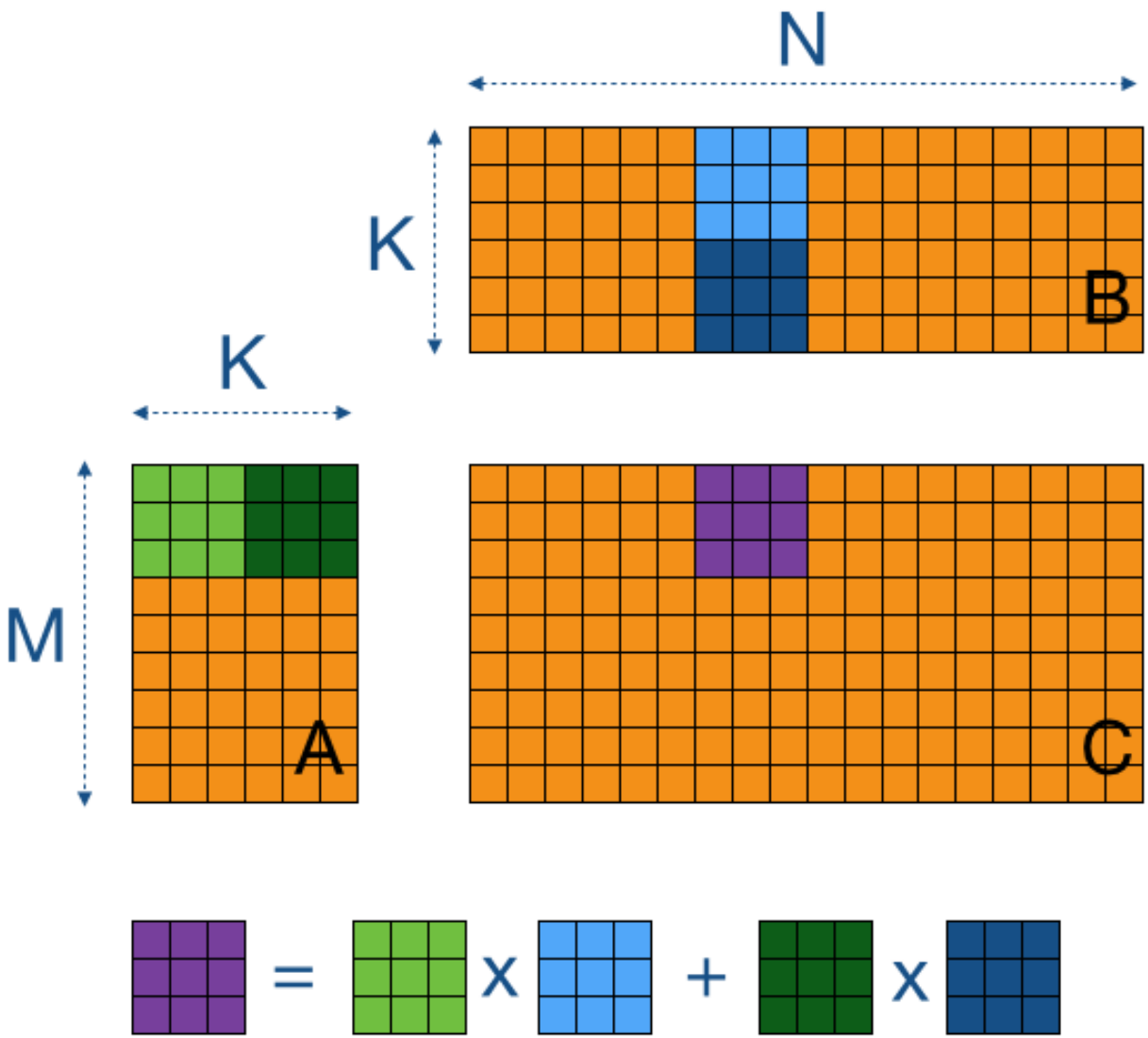
UNIVERSITY OF LEEDS

Contents

- Tile-based linear algebra
- Task-based linear algebra
- Batched BLAS

UNIVERSITY OF LEEDS

Tile-based linear algebra



UNIVERSITY OF LEEDS

Tile-based linear algebra

- Tile size chosen to maximize cache utilisation.
- Tiles stored in contiguous memory (not C/Fortran format).
- Tiled factorizations (e.g. LU, Cholesky, QR) available.
- Implementations available in PLASMA, for example.
- Intel MKL uses this strategy in some routines (increasingly so).

UNIVERSITY OF LEEDS

Task-based programming - DGEMM

Recently added to the OpenMP 4.0 standard (improved in 4.5)

Traditional loop

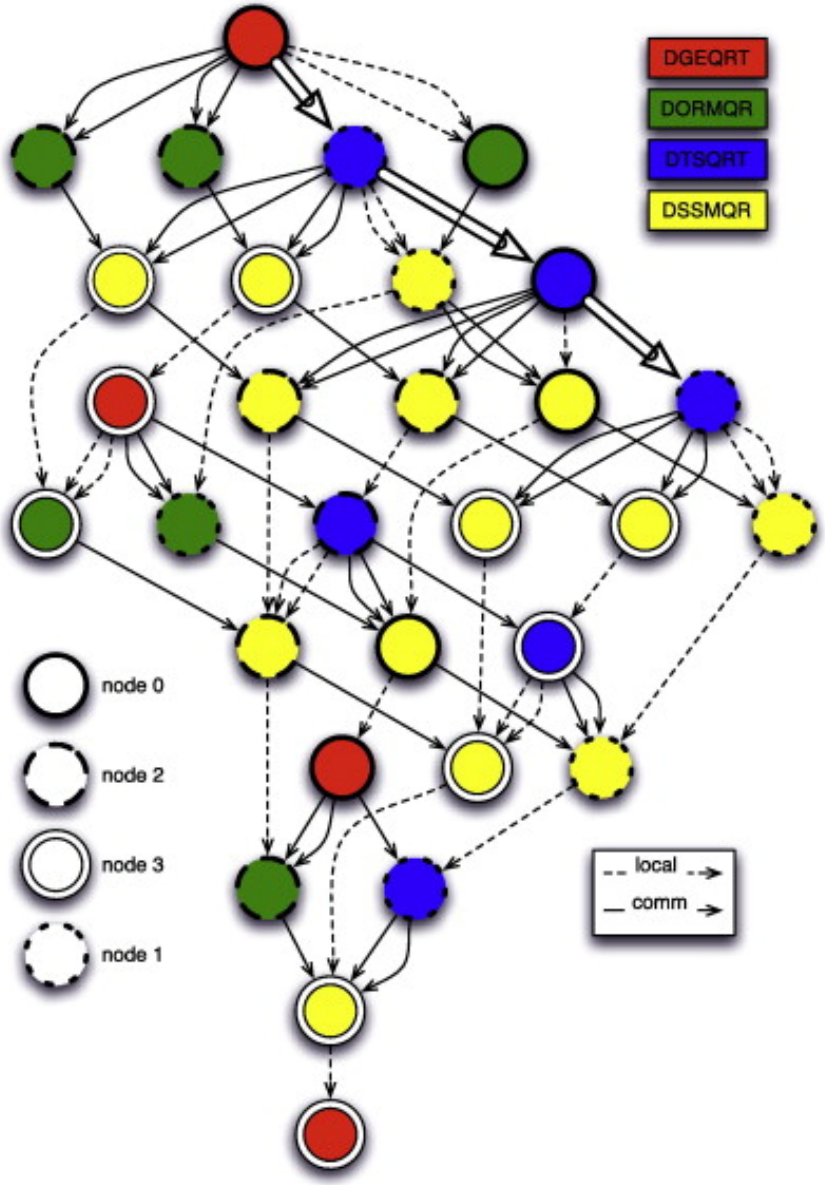
```
for i = 1:m_t
  for j = 1:n_t
    #pragma omp parallel for
    for k = 1:p_t
      dgemm(A(i,k), B(k,j), C(i,j))
    end
  end
end
```

Task-based

```
for i = 1:m_t
  for j = 1:n_t
    for k = 1:p_t
      #pragma omp task
          depend(in:A(i,k),
                 in:B(k,j),
                 inout:C(i,j))
      {
      dgemm(A(i,k), B(k,j), C(i,j))
      }
    end
  end
end
#pragma omp taskwait
```

UNIVERSITY OF LEEDS

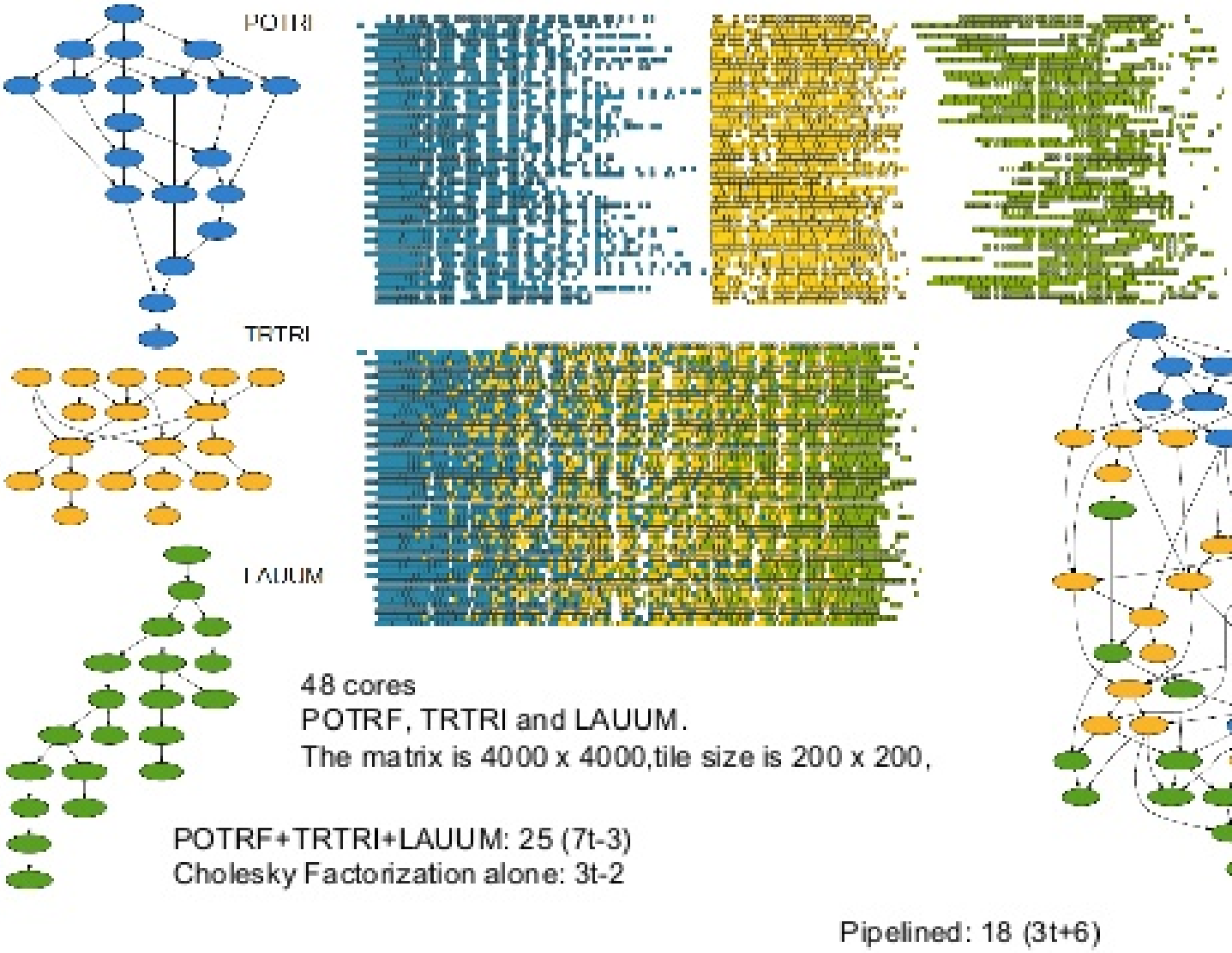
Task-based scheduling



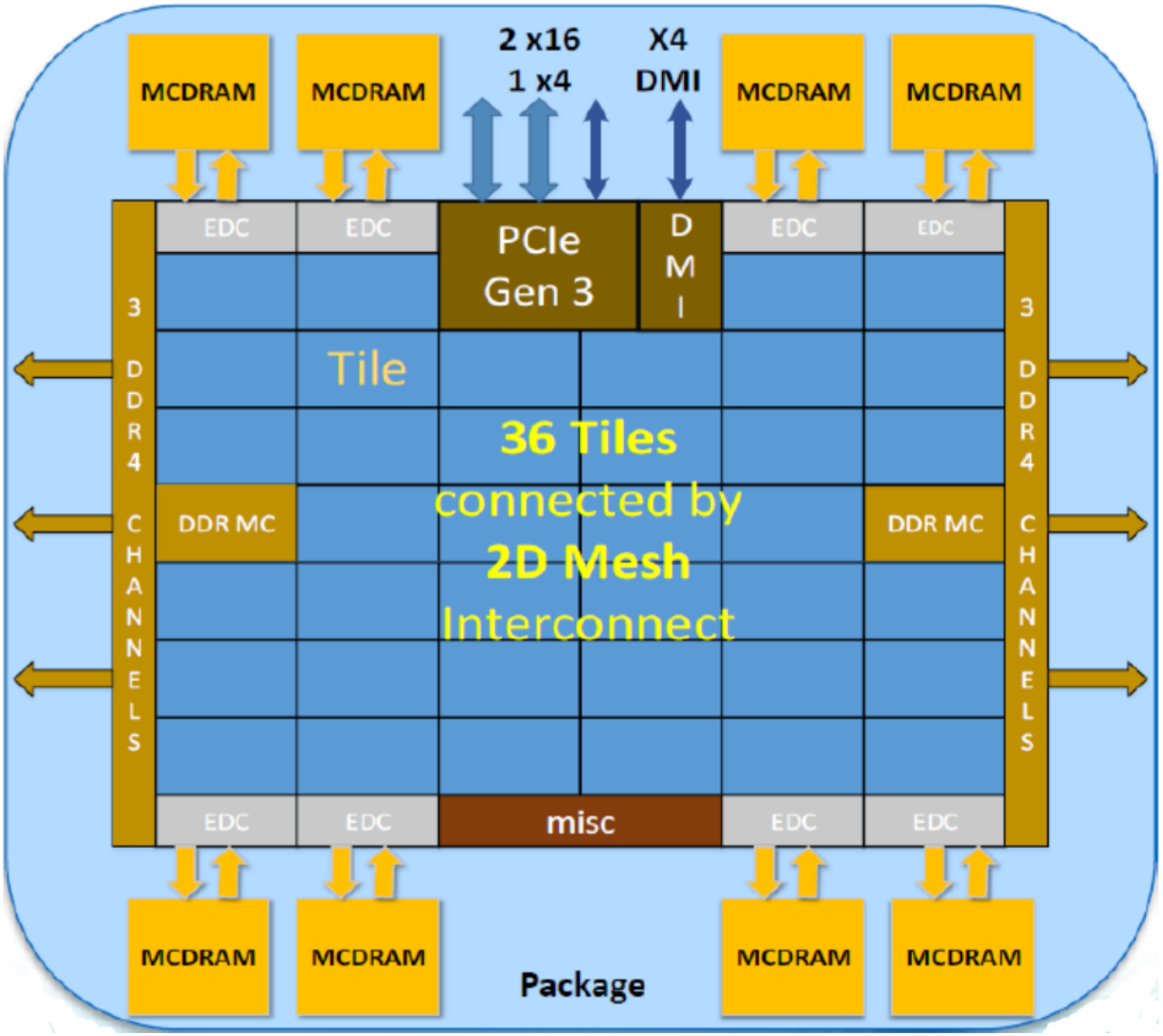
- DAG of 4×4 tile QR factorization.
- Allows tasks to be executed as soon as their dependencies are ready (asynchronously).
- Can move onto next phase of algorithm before previous completes.
- Need a scheduler to map tasks to cores.

UNIVERSITY OF LEEDS

Pipelining: Cholesky Inversion
3 Steps: Factor, Invert L, Multiply L's



Intel Knights Landing



- 72 cores (each tile contains 2)
- 512-bit SIMD vectors (8 doubles)
- MCDRAM: 4x speed of RAM but only 16GB capacity
- Runs linux natively, can compile and run on device
- Peak performance 6 TFlop/s (SP) and 3 TFlop/s (DP)

UNIVERSITY OF LEEDS

Experimental setup

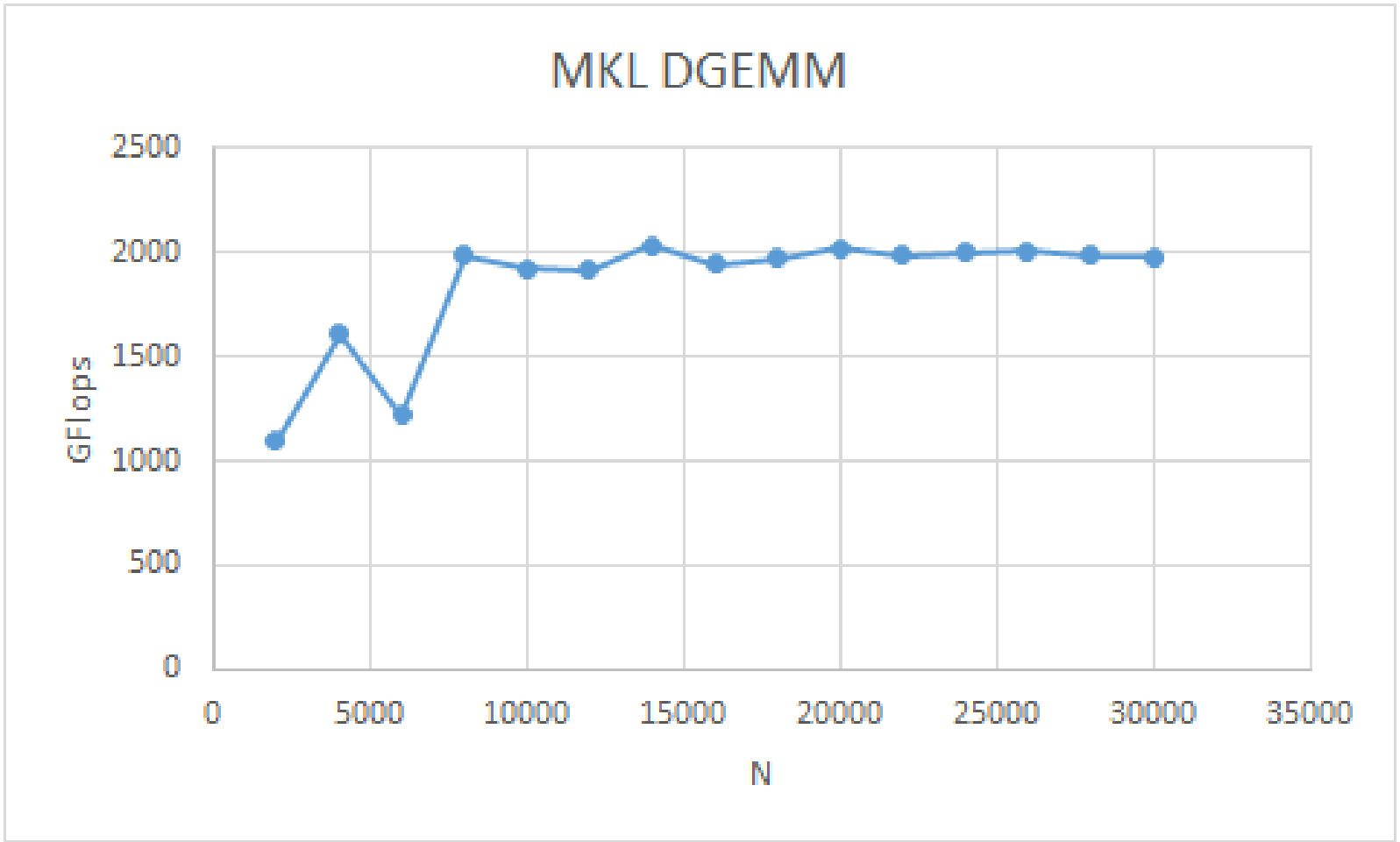
- Algorithms – DGEMM, SPOINV, DPOINV
- Libraries – Intel MKL 17.1, PLASMA 17
- Matrices – Random from $n = 2000 : 30000$.
- Hardware – 68 core KNL in ARC.

Note: PLASMA and MKL both call the same library underneath, only difference is scheduling.

Compiler/runtime flags – gcc-7.2 -O3, numactl -m 1 (to force use of MCDRAM), OMP_NUM_THREADS=68, OMP_PROC_BIND=true, OMP_MAX_TASK_PRIORITY=100.

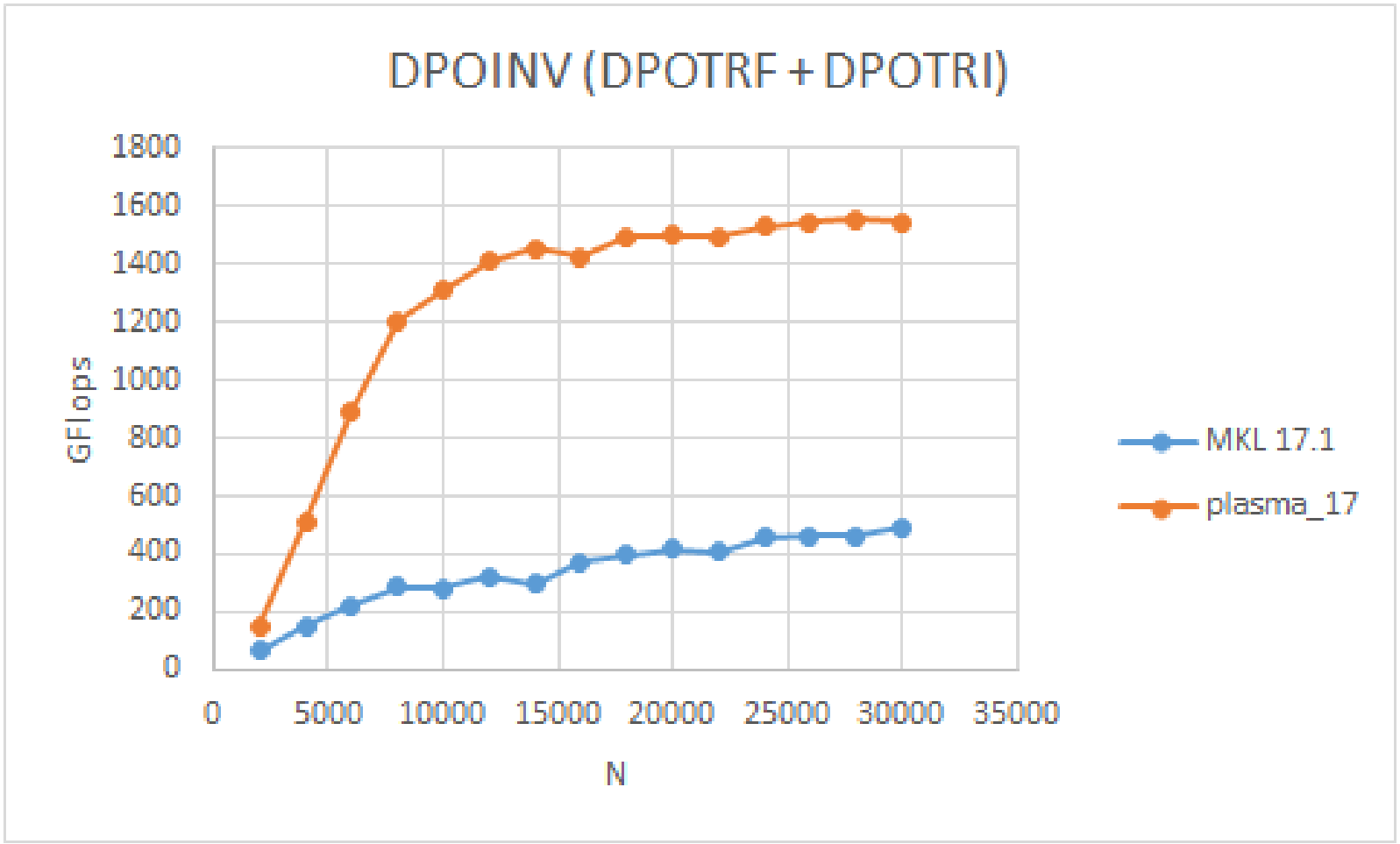
UNIVERSITY OF LEEDS

DGEMM



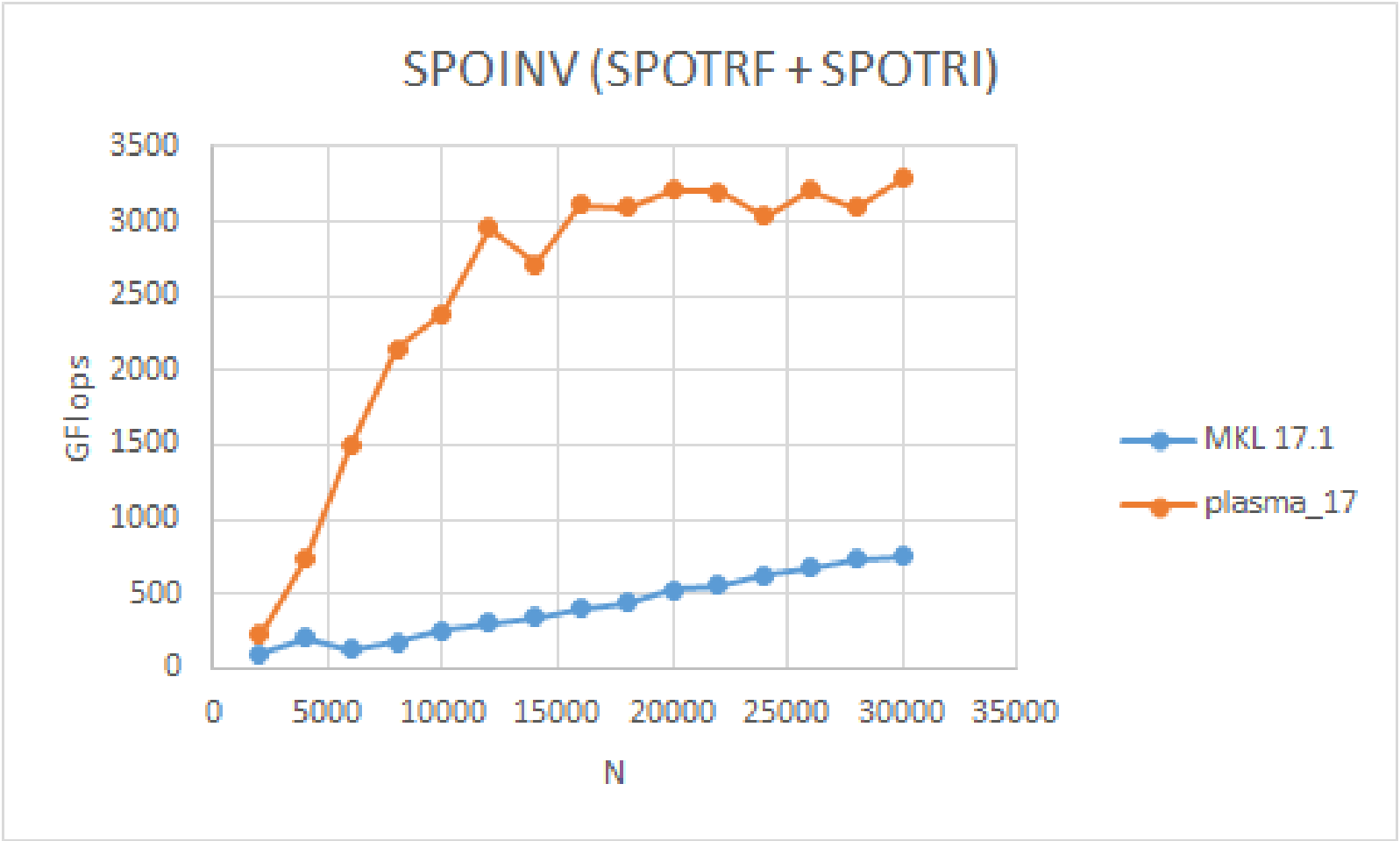
UNIVERSITY OF LEEDS

DPOINV



UNIVERSITY OF LEEDS

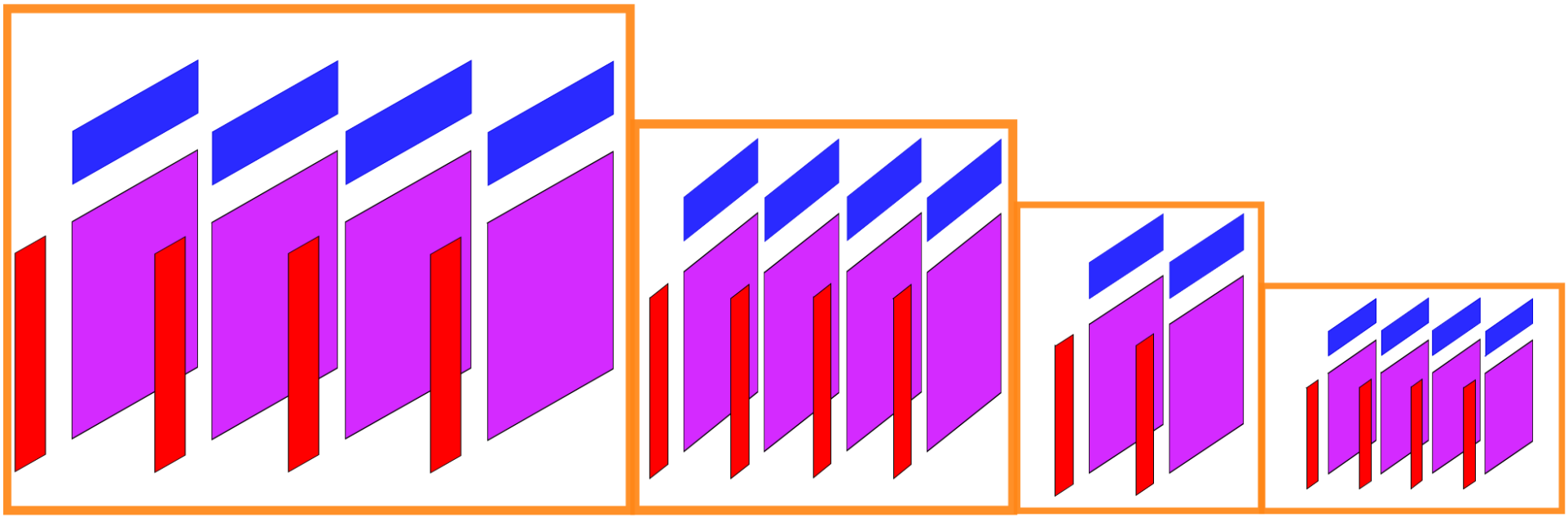
SPOINV



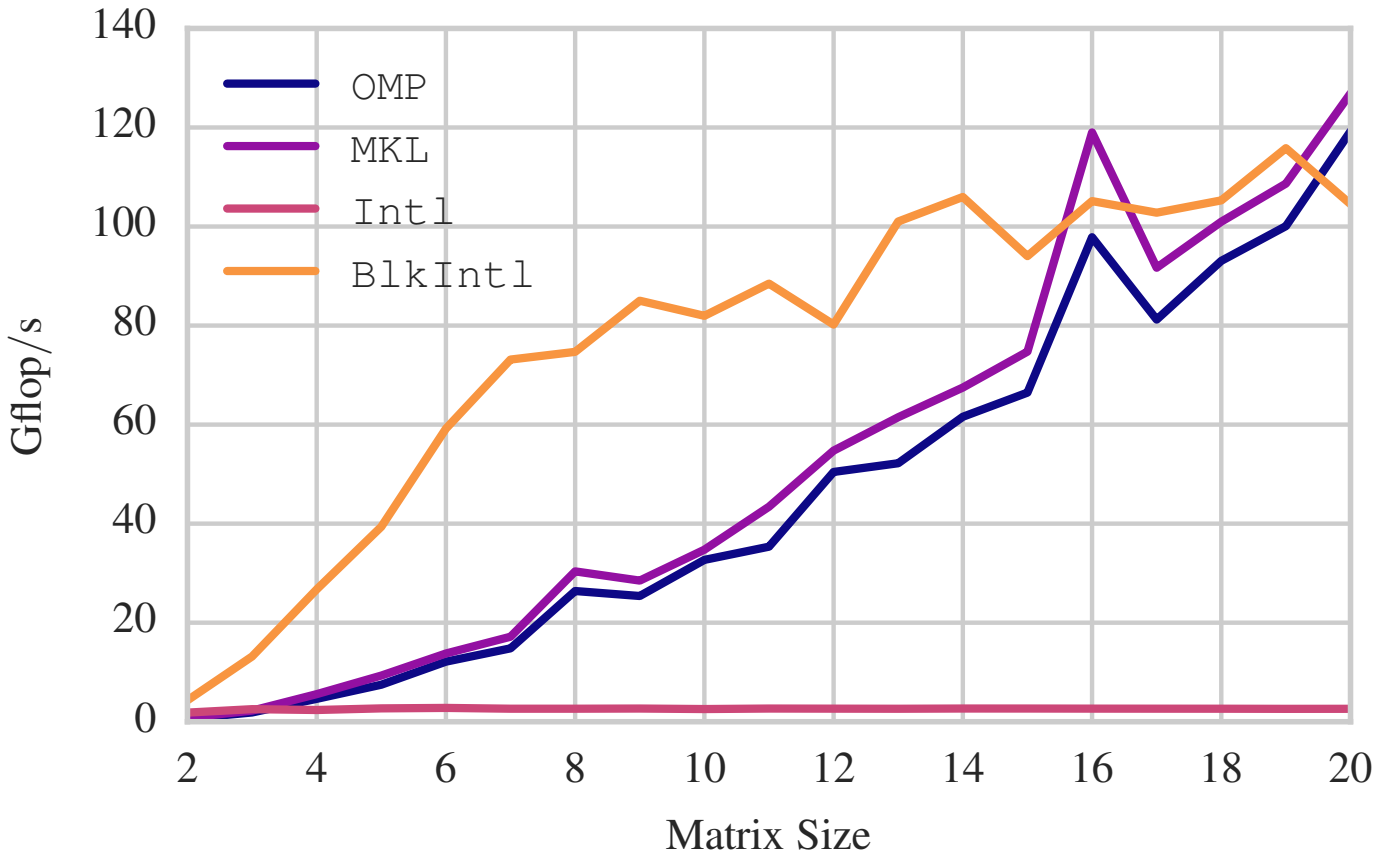
UNIVERSITY OF LEEDS

Batched BLAS

- Compute lots of small matrix problems in parallel (machine learning).
- Standardised API being developed between us, Intel, and NVIDIA.
- Uses a "group" approach
  - Create multiple groups, each group has matrices of same size.
  - Send all groups to be computed in one function call.
- Lots of discussion about best way to reorder memory etc.



Batched BLAS



- 10'000 DGEMMs of varying size vs. `mkl_dgemm_batch`
- BlkIntl strategy includes time to convert to and from our new memory layout.

UNIVERSITY OF LEEDS

Further information

- PLASMA library – `https://bitbucket.org/icl/plasma`
- StarPU – Task-based programming including use of GPUs
- Tile algorithms – Papers by Jack Dongarra (Tennessee), Julian Languo (U. C. Denver), Emmanuel Agullo (INRIA) et. al.
- Batched BLAS – `http://bit.ly/Batch-BLAS-2017`

UNIVERSITY OF LEEDS

Thanks for listening!

UNIVERSITY OF LEEDS