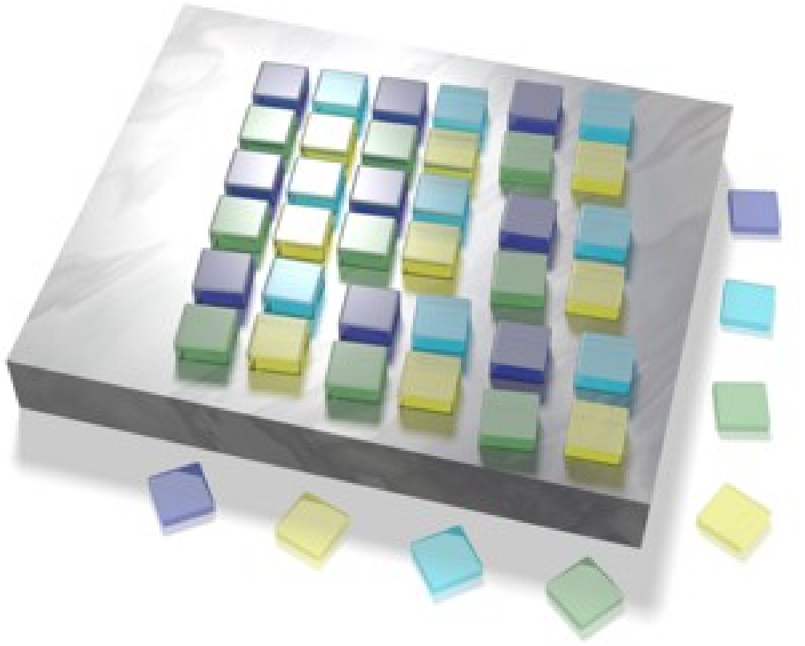
SLATE Project: Objectives, Design, and Results

SLATE: Software for Linear Algebra Targeting Exascale

This research was supported by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy's Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation's exascale computing imperative.



ScaLAPACK Legacy

1991 Linux

1992

1993 NCSA Mosaic

1994 MPI 1.0

1995 **ScaLAPACK**

1996

1997 OpenMP 1.0 for FORTRAN

1998 OpenMP 1.0 for C/C++

February 28, 1995:

- initial release of ScaLAPACK
- Denver International Airport opens

Fujitsu Numerical Wind Tunnel is the fastest machine on the TOP500 list, with **170 Gflop/s** of LINPACK performance.

My 2016 MacBook Pro gets **166 Gflop/s**

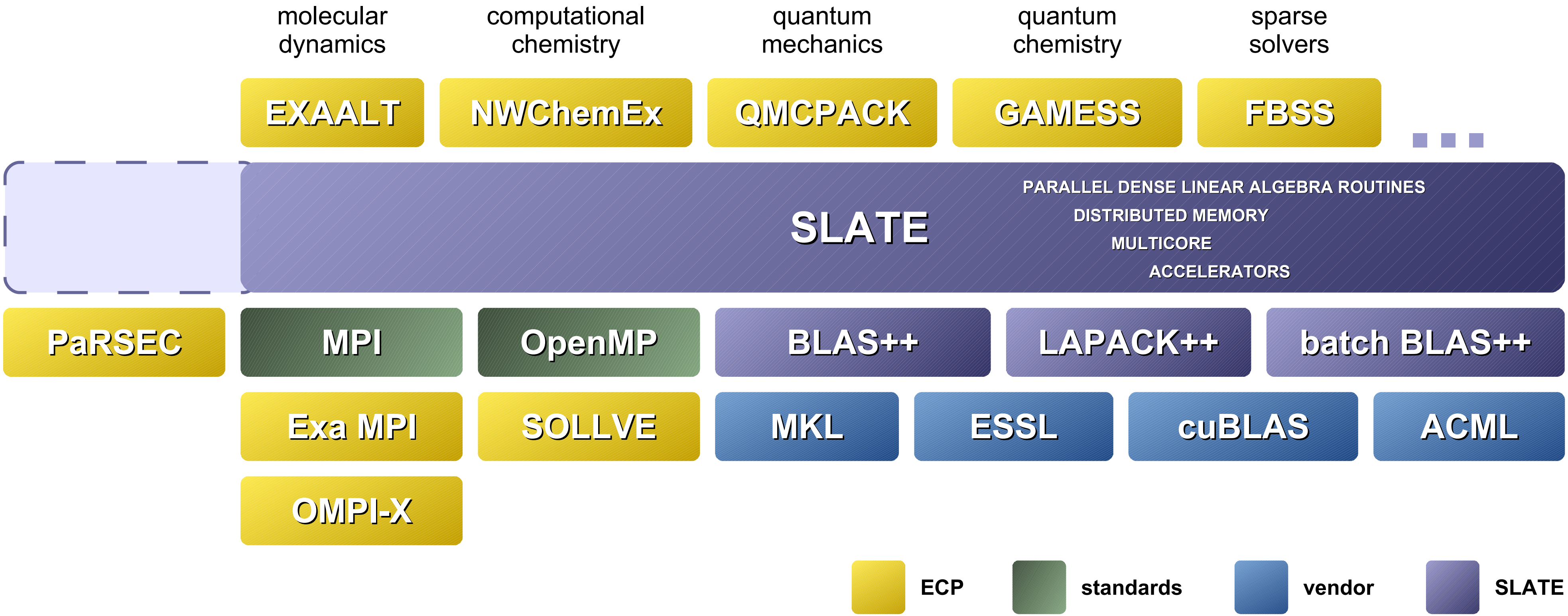
SLATE Objectives

can be built:

- serial
- OpenMP multithreading
- MPI message passing
- GPU acceleration

• Coverage	ScaLAPACK and beyond
• Modern Hardware	DOE CORAL (pre Exascale) → DOE Exascale
• Portability	Intel Xeon (&Phi), IBM POWER, ARM, NVIDIA, AMD, …
• Modern Language	C++11/14/17 (templates, STL, overloading, polymorphism, …)
• Modern Standards	MPI 3, OpenMP 4/5 (&omp target)
• Performance	80-90% of peak (asymptotic)
• Scalability	full machine (tens of thousands of nodes)
• Productivity	ca. 4 full time developers
• Maintainability	part time developers + community

SLATE Software Stack



SLATE Resources

• main ECP website:	https://www.exascaleproject.org/
• main SLATE website:	http://icl.utk.edu/slate/
• main SLATE repository:	https://bitbucket.org/icl/slate
• BLAS++ repository:	https://bitbucket.org/icl/blaspp
• LAPACK++ repositry:	https://bitbucket.org/icl/lapackpp
• SLATE Working Notes:	http://www.icl.utk.edu/publications/series/swans
• Research Gate project:	https://www.researchgate.net/project/ECP-SLATE
• SLATE User	https://groups.google.com/a/icl.utk.edu/forum/#!forum/slate-user

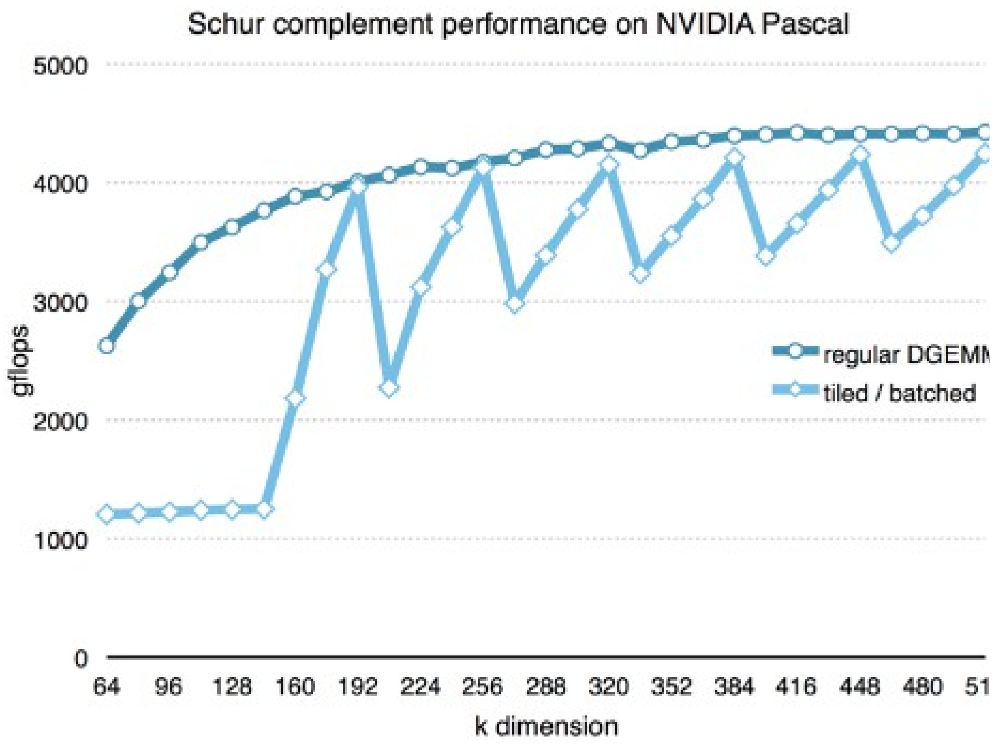
SWAN Catalog

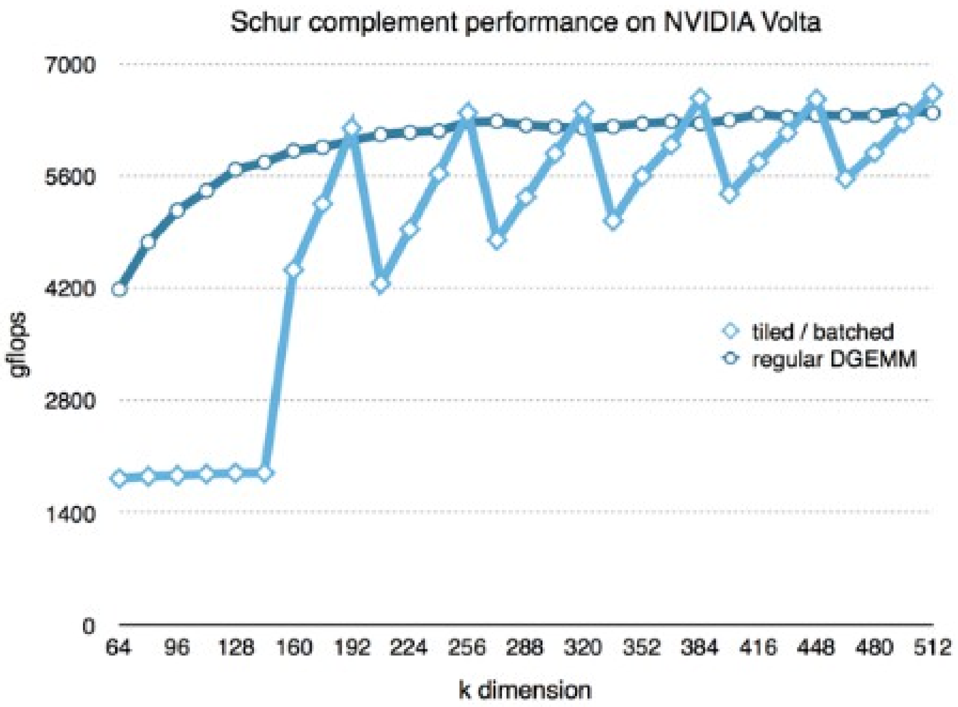
- SWAN 1 Prospectus
- SWAN 2 BLAS++ & LAPACK++
- SWAN 3 Design
- SWAN 4 Batched BLAS++ API
- SWAN 5 Parallel BLAS Performance
- SWAN 6 Parallel Norms Performance
- SWAN 7 Batched BLAS++ Implementation
- SWAN 8 Linear Systems Performance
- SWAN 9 Least Squares Performance

SLATE Working Notes

- Main SWANs page
  - http://www.icl.utk.edu/publications/series/swans
- Designing SLATE: Software for Linear Algebra Targeting Exascale
  - http://www.icl.utk.edu/publications/swan-003
- C++ API for BLAS and LAPACK
  - http://www.icl.utk.edu/publications/swan-002
  - https://bitbucket.org/icl/blaspp
  - https://bitbucket.org/icl/lapackpp
- Roadmap for the Development of a Linear Algebra Library for Exascale Computing:
  - SLATE: Software for Linear Algebra Targeting Exascale
  - http://www.icl.utk.edu/publications/swan-001

GEMM Efficiency

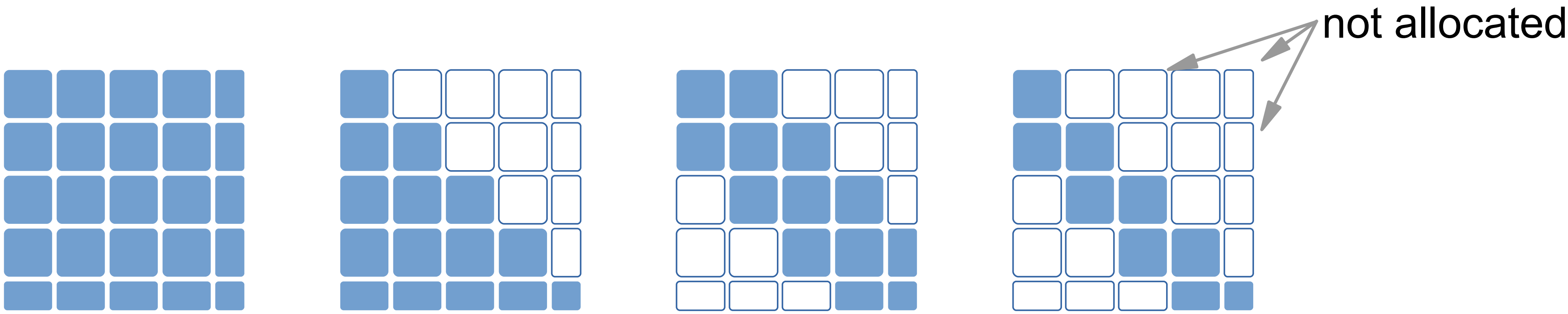




C = C – A × B with small k, i.e., the DGEMM called in LU factorization

The matrix fills out the GPU memory. The X axis shows the k dimension.

SLATE Matrix




```
std::map<std::tuple<int64_t, int64_t, int>, Tile<FloatType>*> *tiles_;
```

row

column

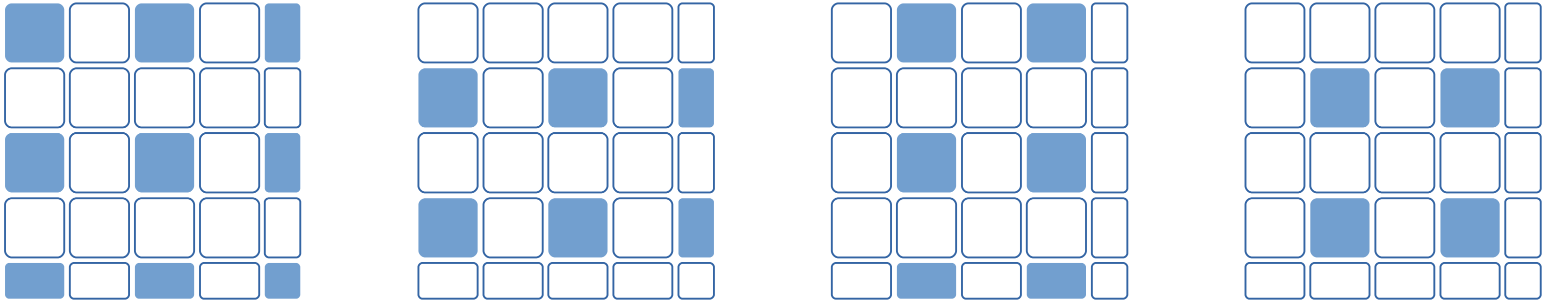
host & devices

- collection of tiles
- **individually allocated**
- only allocate what is needed
- accommodates: symmetric, triangular, band, …

While in the PLASMA library the matrix is also stored in tiles, the tiles are laid out contiguously in memory.

In contrast, in SLATE, the tiles are individually allocated, with no correlation of their locations in the matrix to their addresses in memory.

SLATE Distributed Matrix



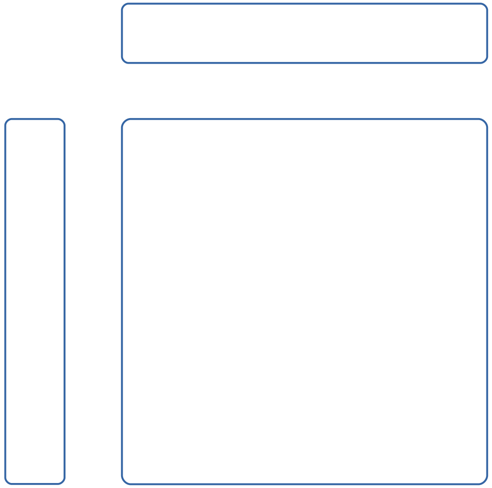
std::map<std::tuple<*int64_t*, *int64_t*, *int*>, Tile<FloatType>*> *tiles_;

- distributed matrix
- global indexing of tiles
- only allocate the local part
- any distribution is possible (2D block cyclic by default)

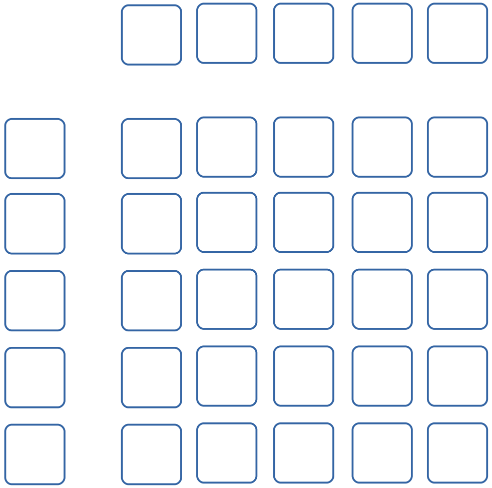
The same structure, used for single node representation, naturally supports distributed memory representation.

Data Storage Comparison

LAPACK
MAGMA



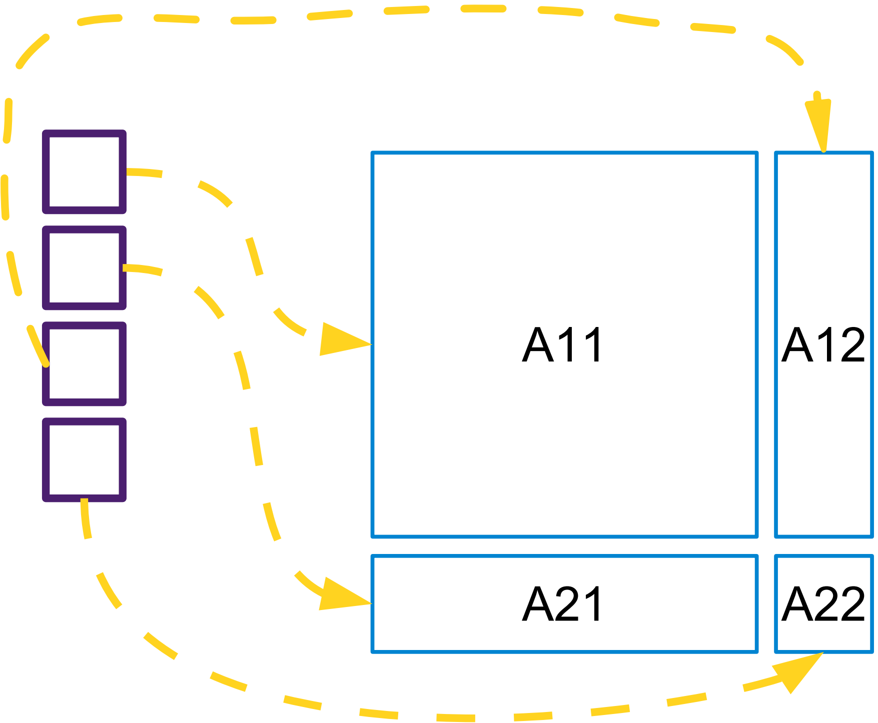
SLATE



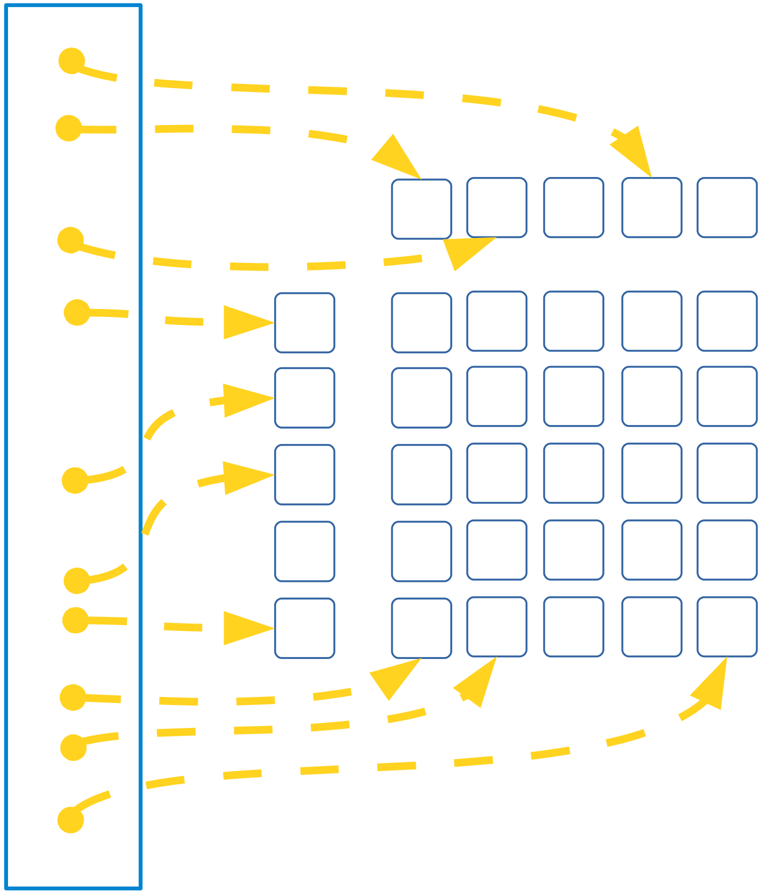
$$C = C - A \times B$$

Data Storage Comparison

PLASMA Descriptor

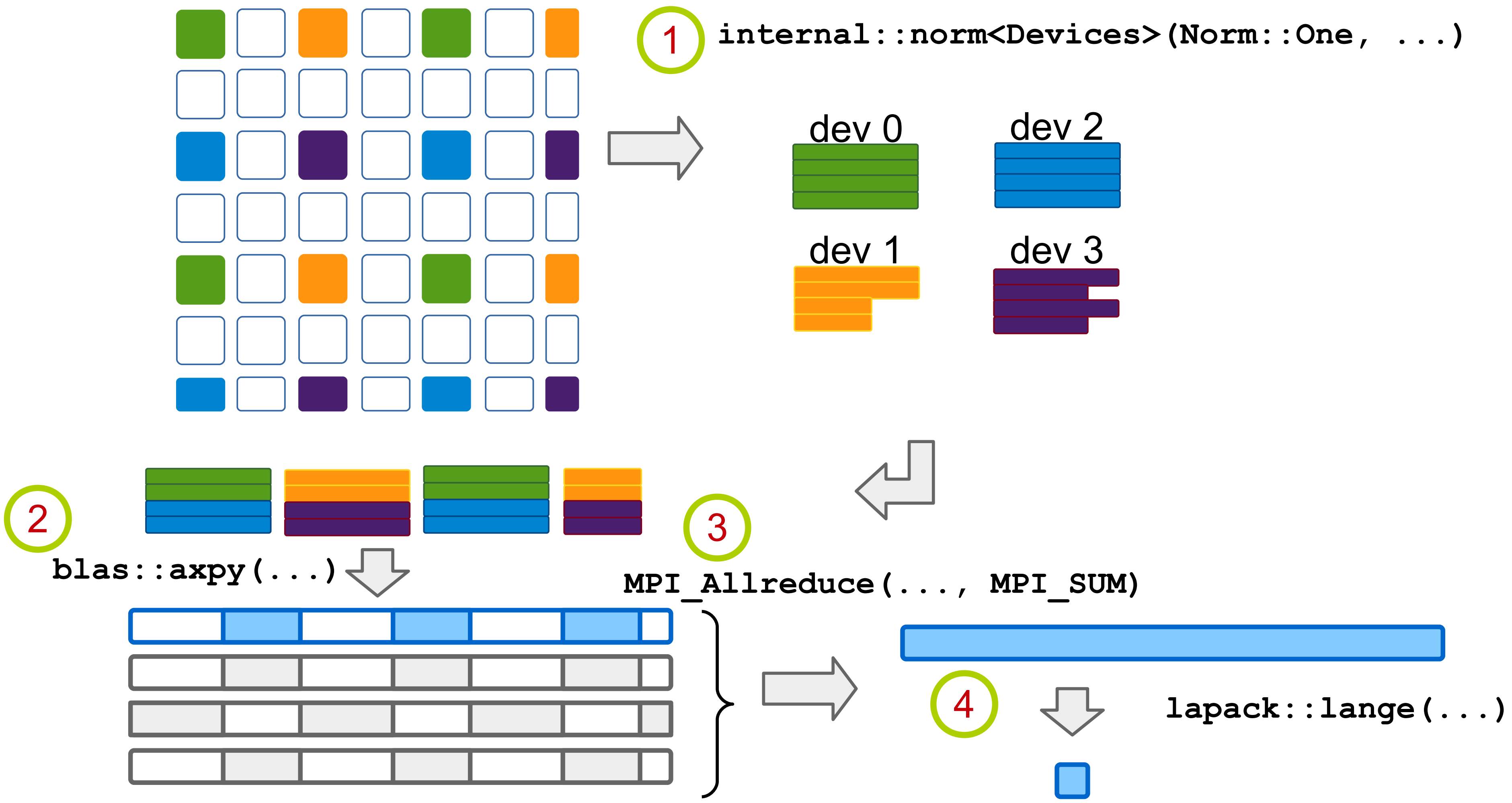


SLATE Tile Map<>

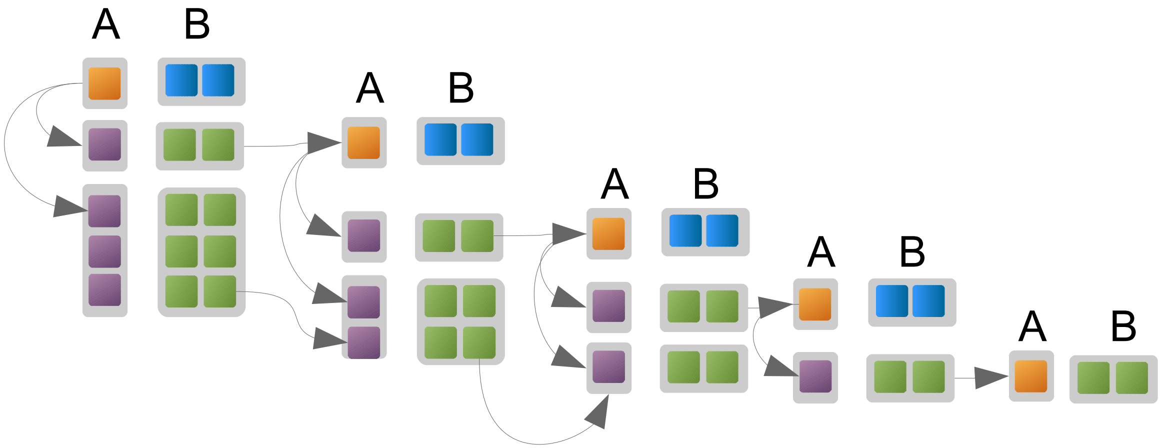


This layout allows in-place translation.

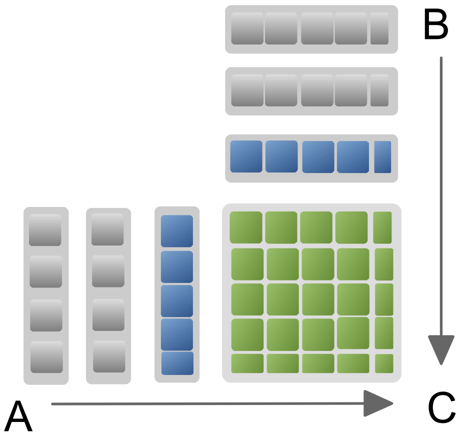
Computing Norms in Parallel in SLATE



SLATE Parallel BLAS Scheduling



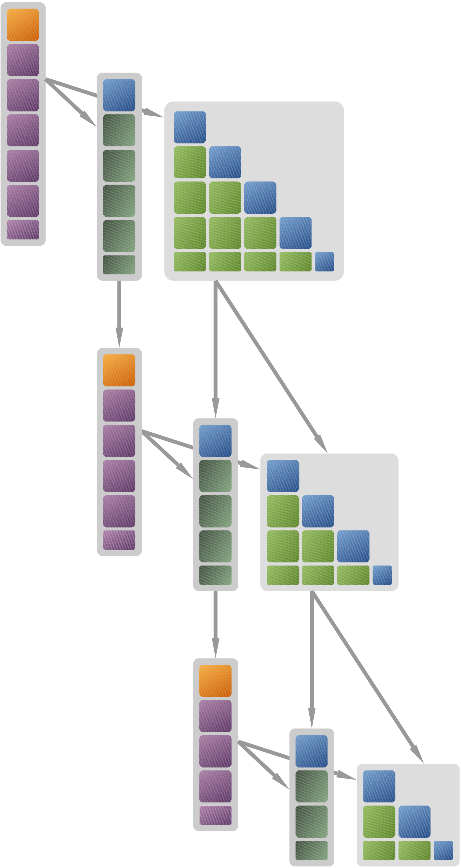
TRSM



GEMM

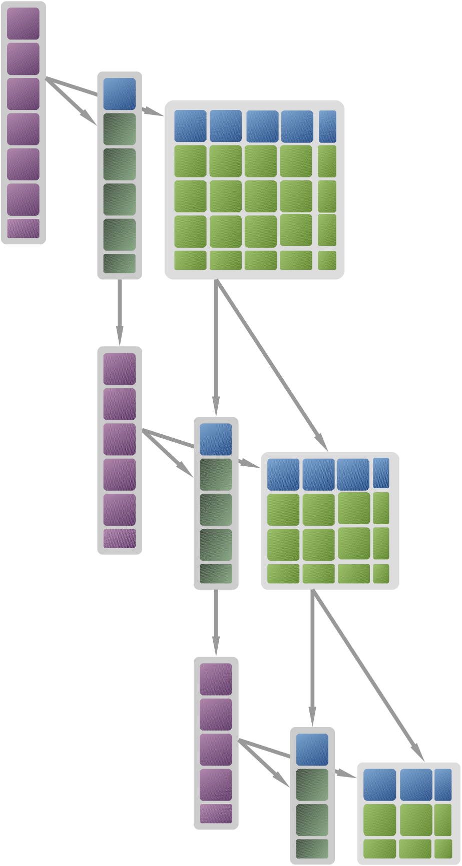
- nested parallelism
- top level:
  - #pragma omp task depend
- bottom level:
  - #pragma omp task
  - batch GEMM

SLATE Scheduling: Linear Solvers



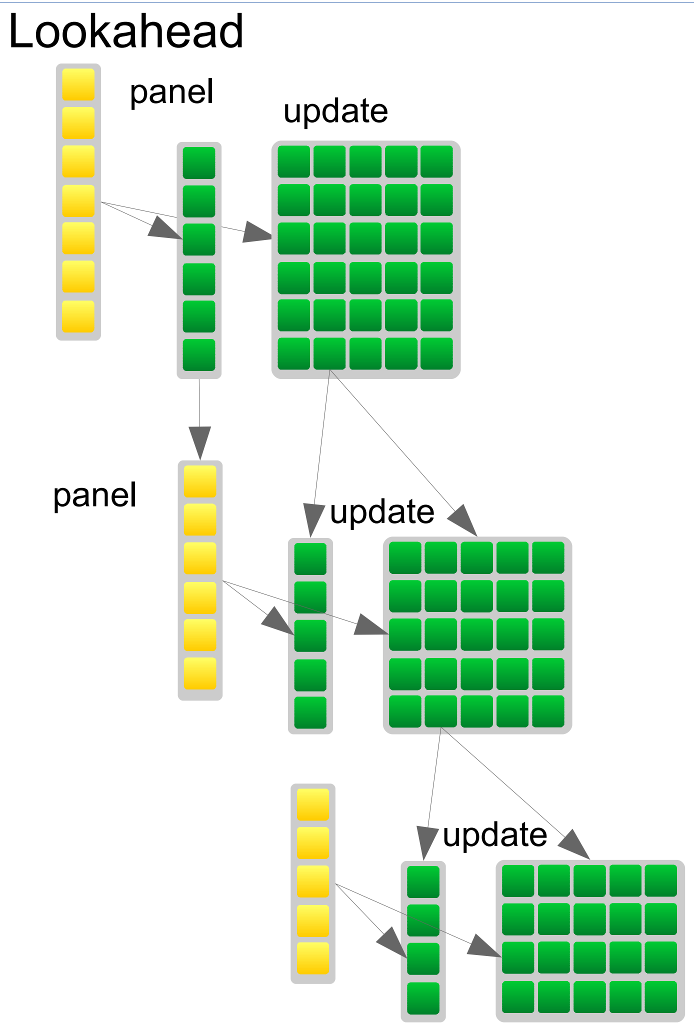
Cholesky factorization

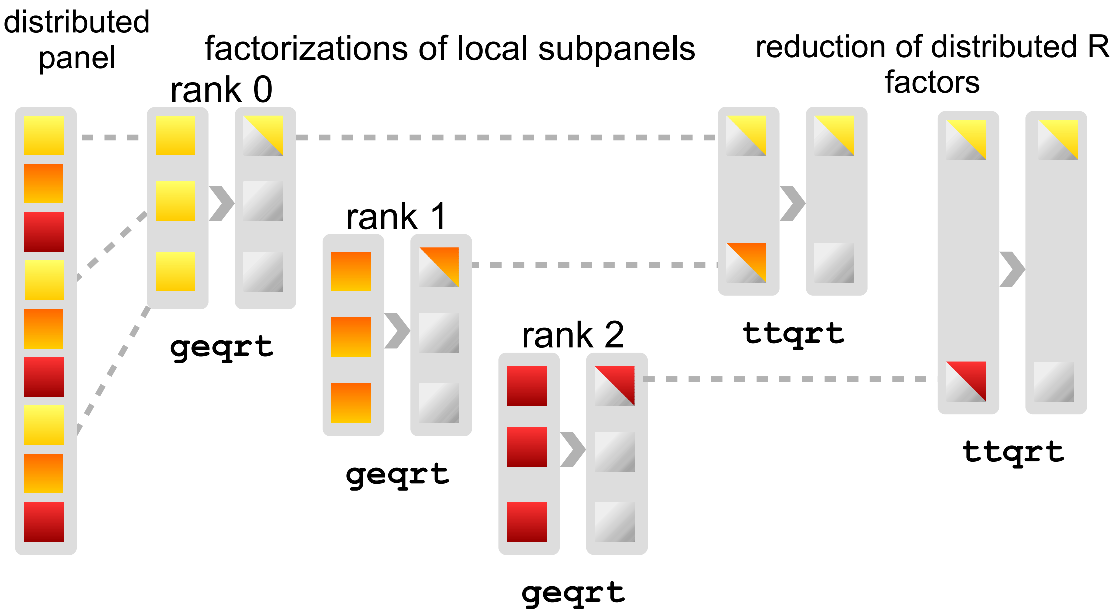
- nested parallelism
- top level:
  - #pragma omp task depend
- bottom level:
  - #pragma omp task
  - batch GEMM



LU factorization

SLATE QR Scheduling



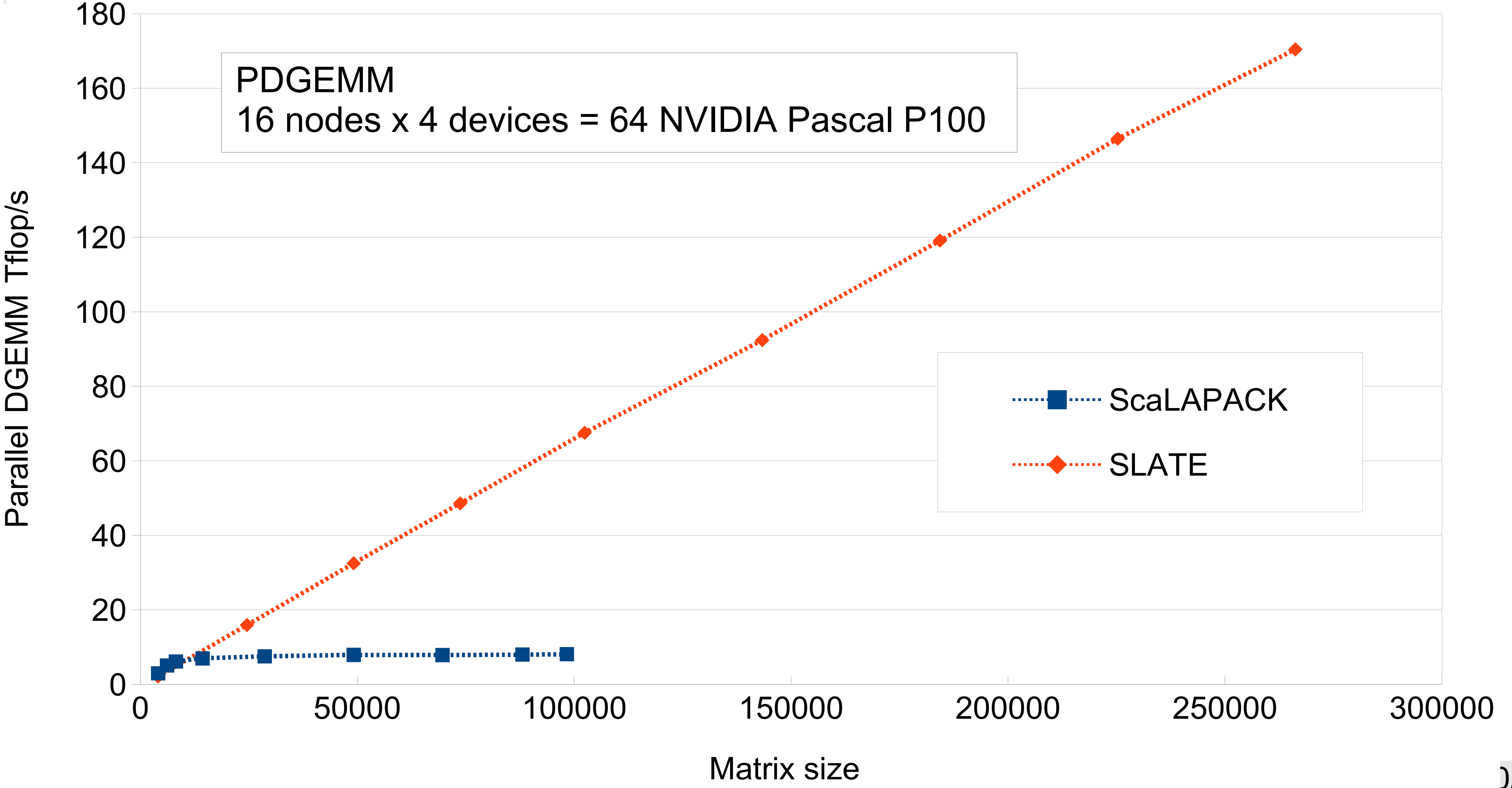


Preliminary SLATE Performance Results

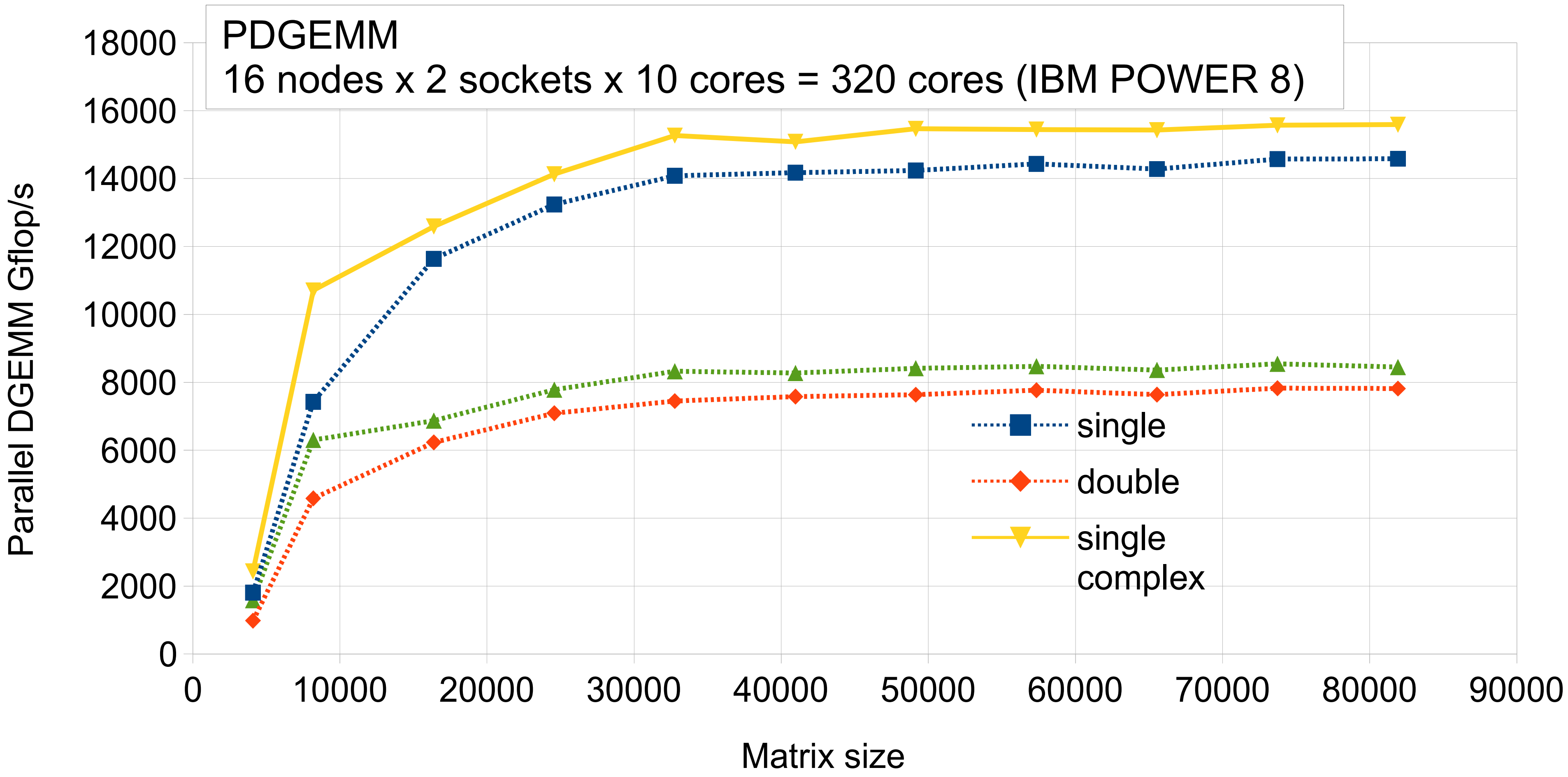
SummitDev @ OLCF

- 3×18 = **54** nodes (IBM S822LC)
- 2×10 = **20** cores (IBM POWER8) ca. 0.5 TFLOPS (2.5%)
- **4** GPUs (NVIDIA P100) ca. 20 TFLOPS (97.5%)
- **256** GB DDR4
- 4×16 = **64** GB **HBM2**
- NVLink 1.0 80 Gbps (advertised)
- NVLink 2.0 ~200 Gbps
- GCC 7.1.0
- ESSL 5.5.0
- CUDA 8.0.54
- Spectrum MPI 10.1.0.3.

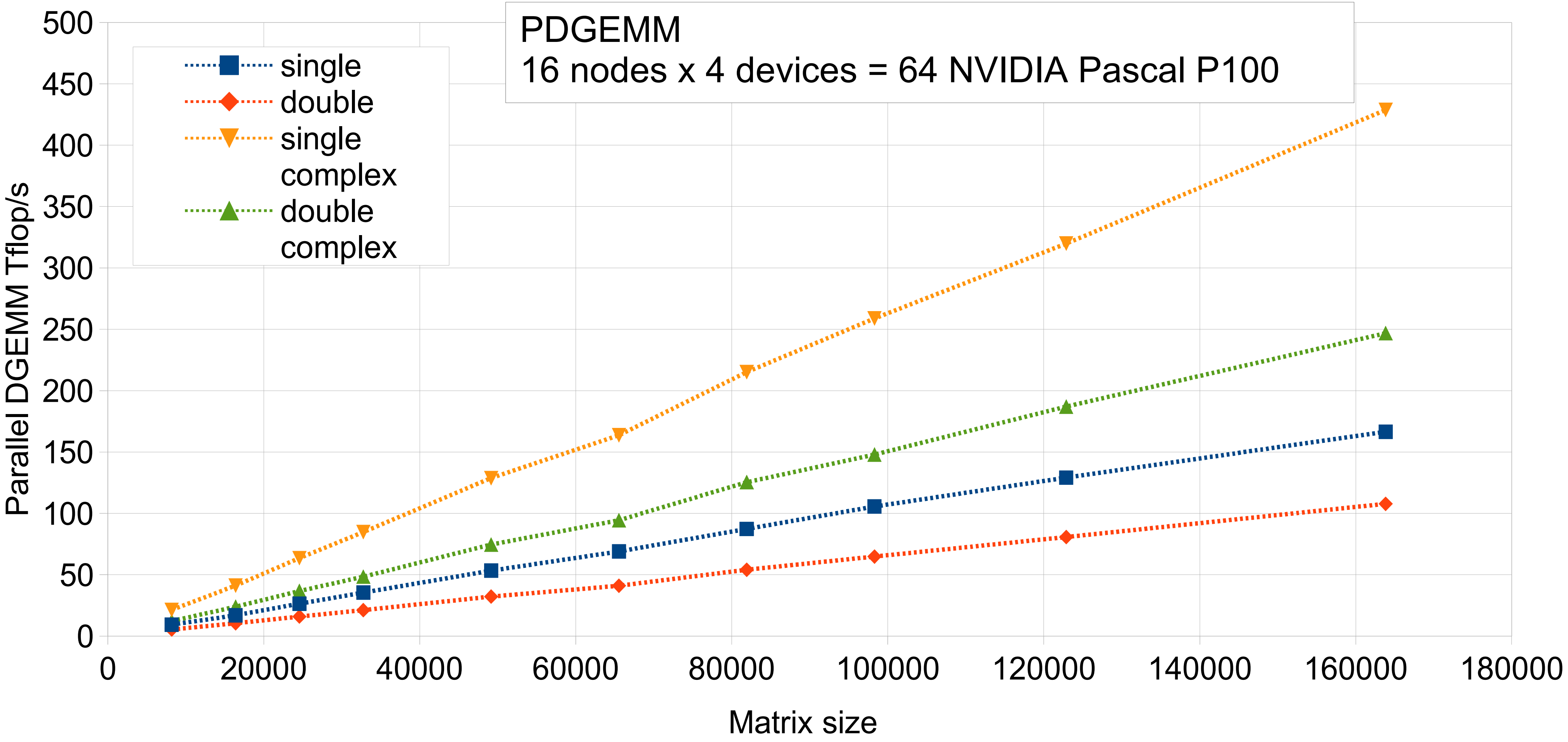
GPU-Accelerated Parallel Matrix Multiply: SLATE vs. ScaLAPACK



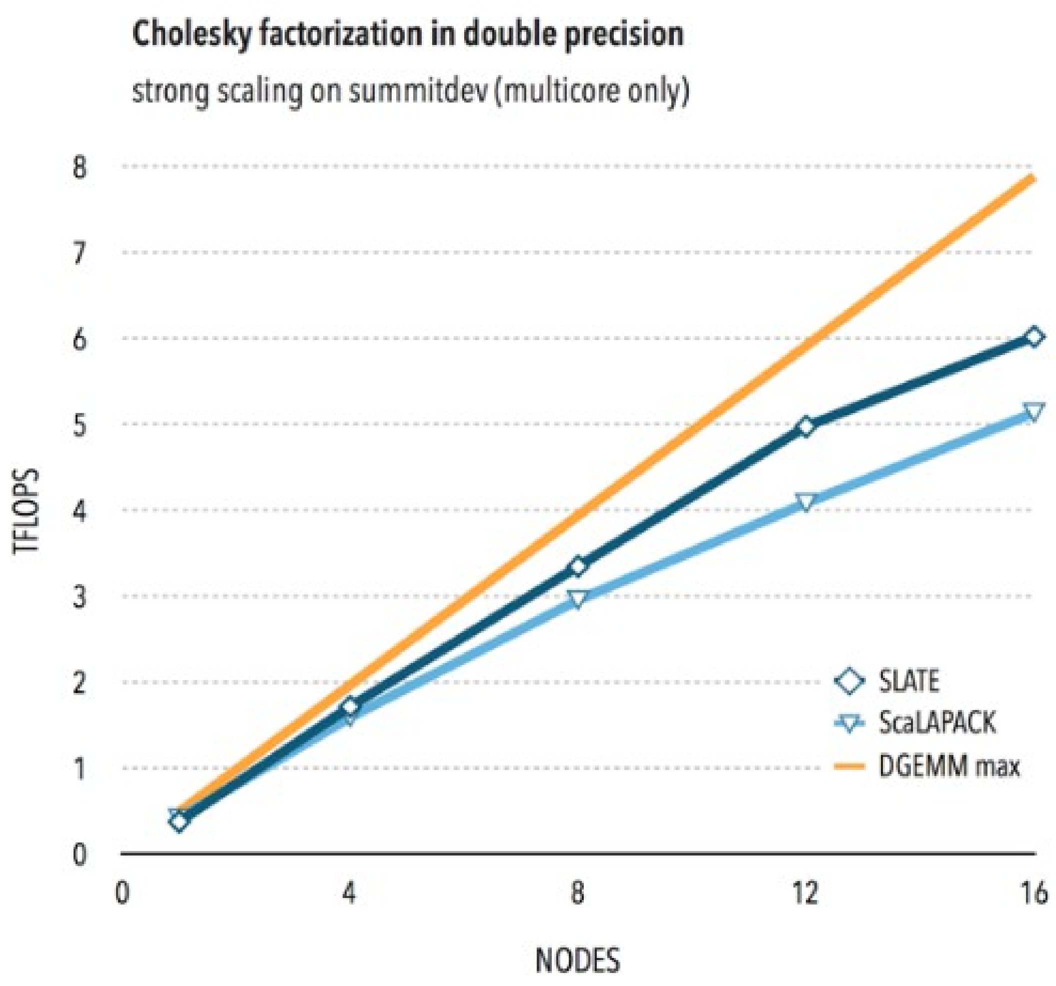
SLATE PDGEMM with Multiple Precisions: CPU only



SLATE PDGEMM with Multiple Precisions: GPUs



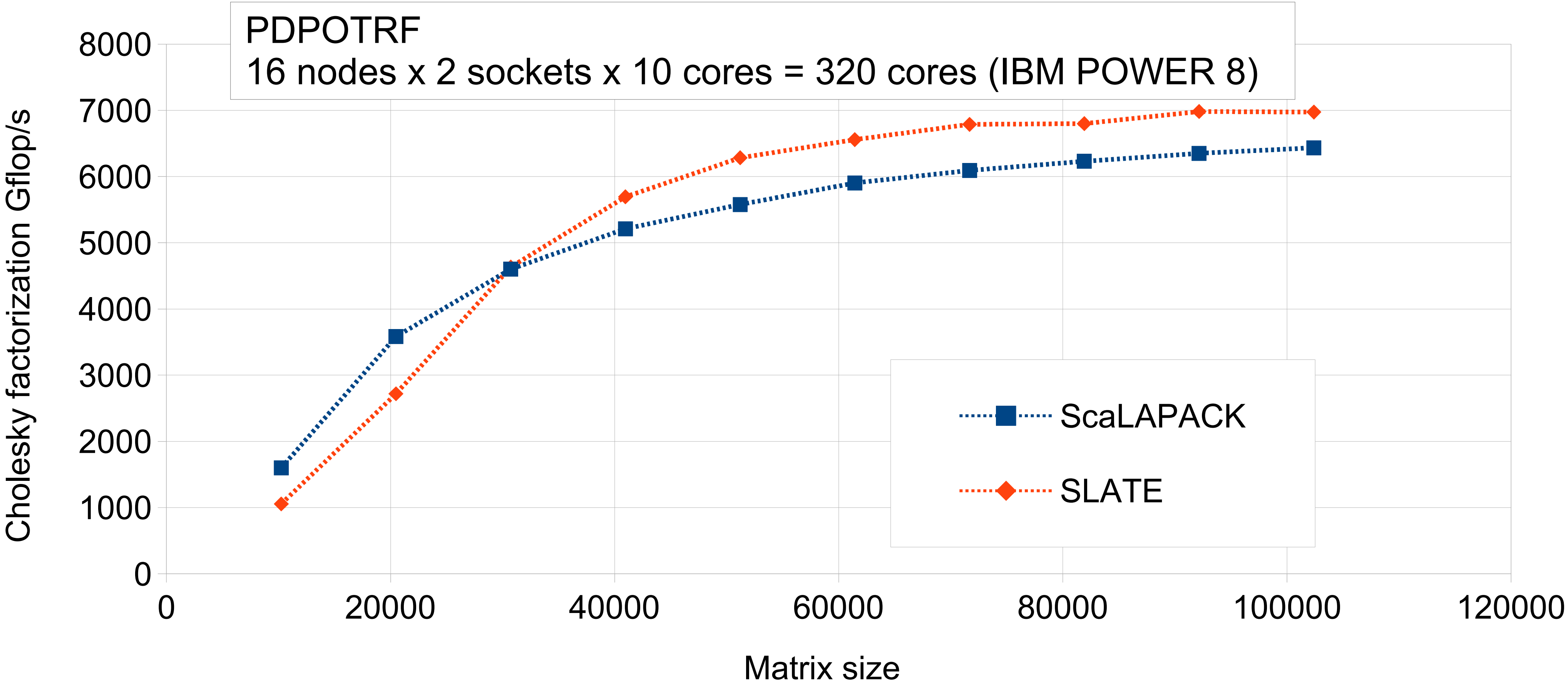
SLATE Cholesky Multicore Performance



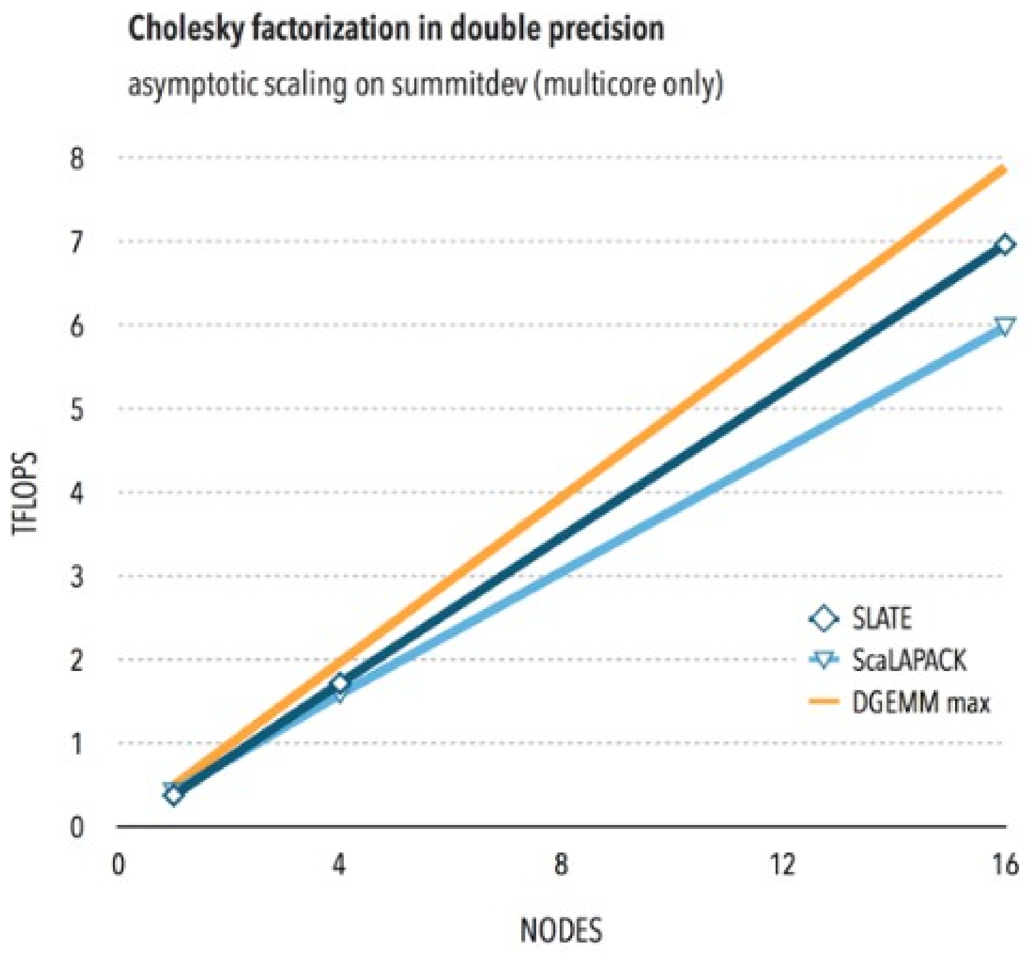
<u>strong scaling</u>

- 40 K × 40 K
- up to 16 nodes / 32 sockets / 320 cores

SLATE Cholesky Factorization Comparison



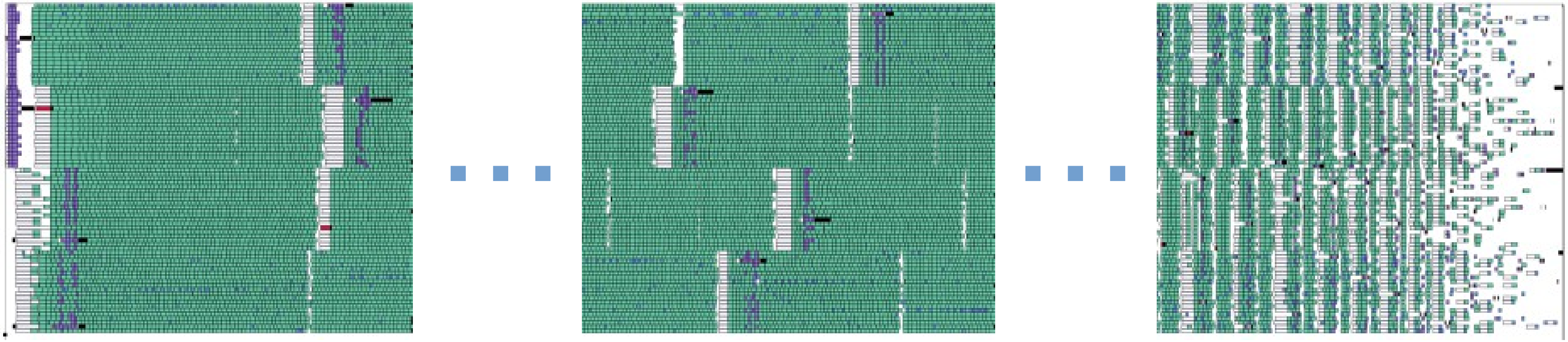
SLATE Cholesky Multicore Performance



<u>asymptotic scaling</u>

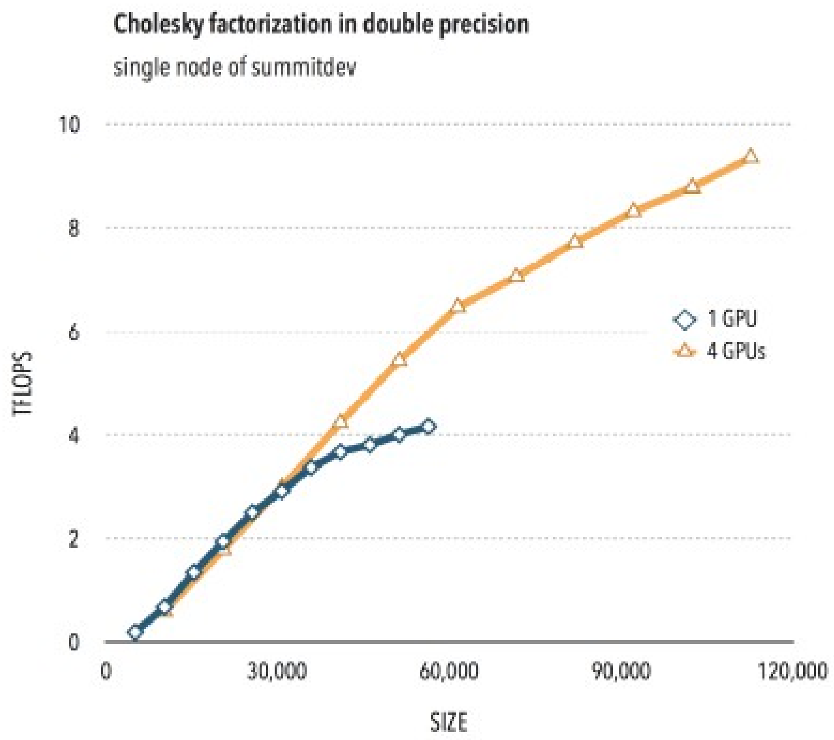
- 20 K × 20 K1 node 20 cores
- 40 K × 40 K4 nodes 80 cores
- 80 K × 80 K16 nodes 320 cores

SLATE Multicore Trace



- Cholesky factorization
- 4 nodes (80 cores) factoring a 25 K × 25 K matrix using a tile size of 256
- dynamic scheduling – no fork-and-join synchronization

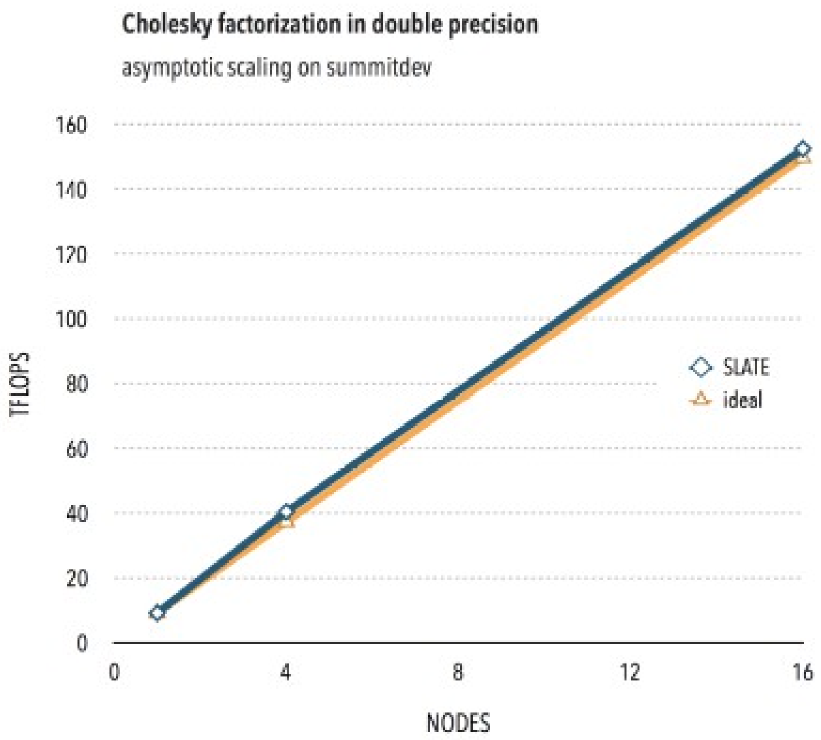
SLATE Cholesky GPU Performance



<u>GPU performance</u>

- up to 56 K × 56 K — 1 GPU
- up to 112 K × 112 K — 4 GPUs

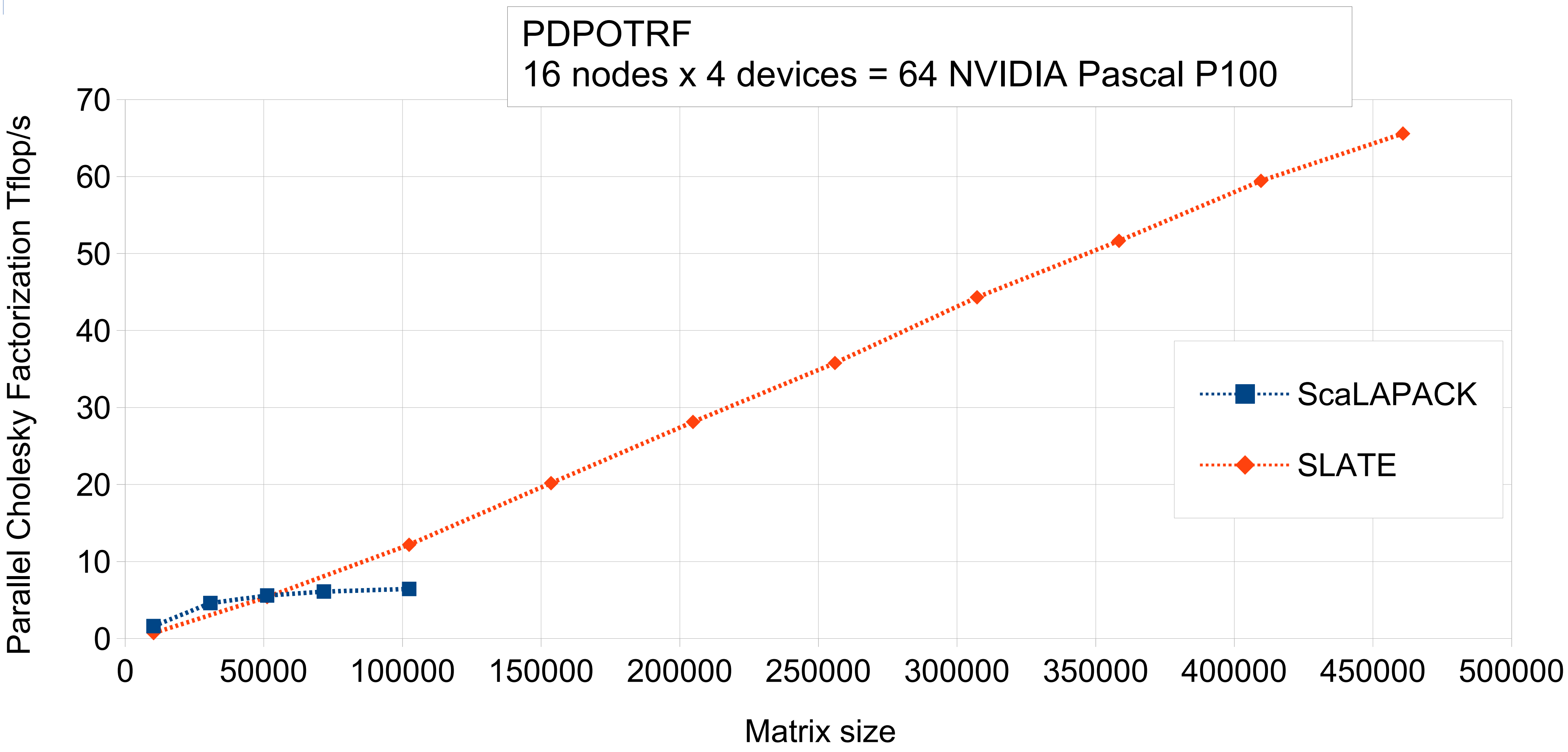
SLATE GPU Performance



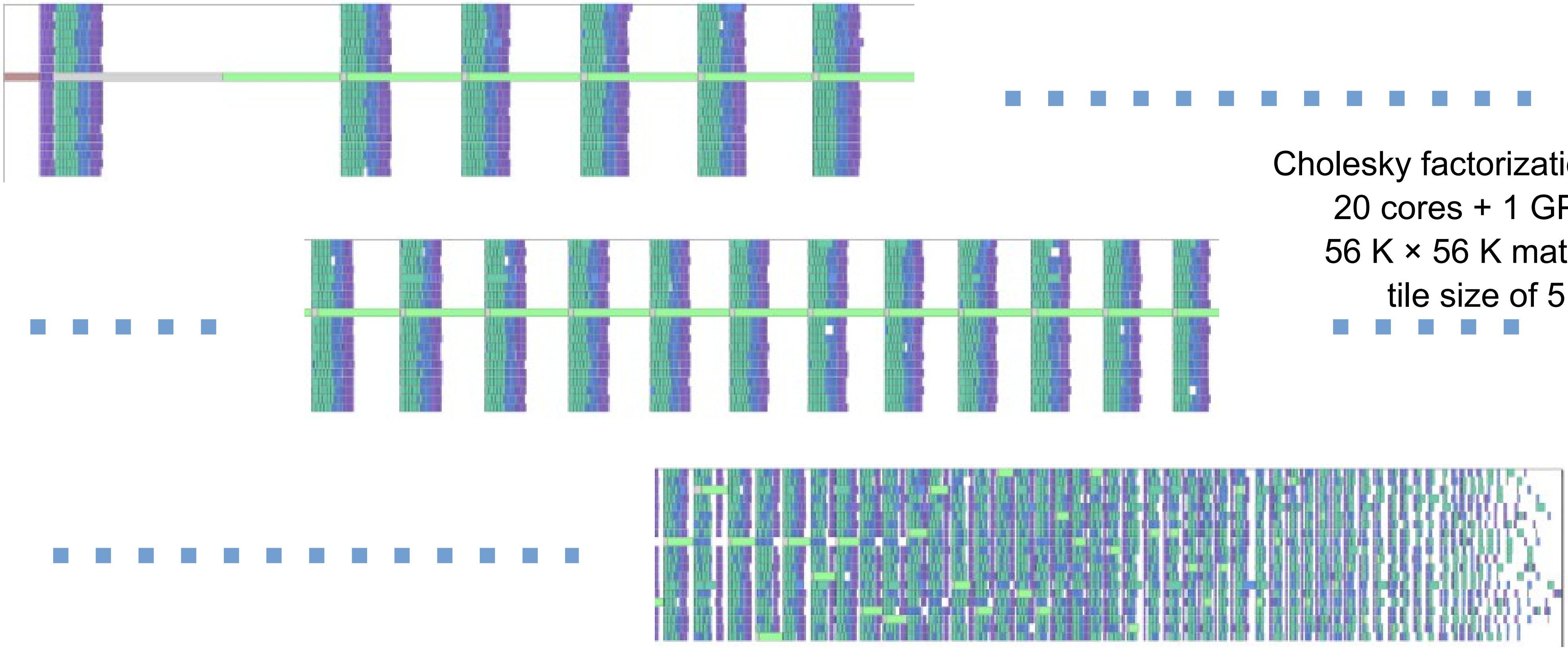
asymptotic scaling

- 112 K × 112 K 1 node 4 GPUs
- 225 K × 225 K 4 nodes 16 GPUs
- 450 K × 450 K 16 nodes 64 GPUs

SLATE Cholesky Factorization: GPUs

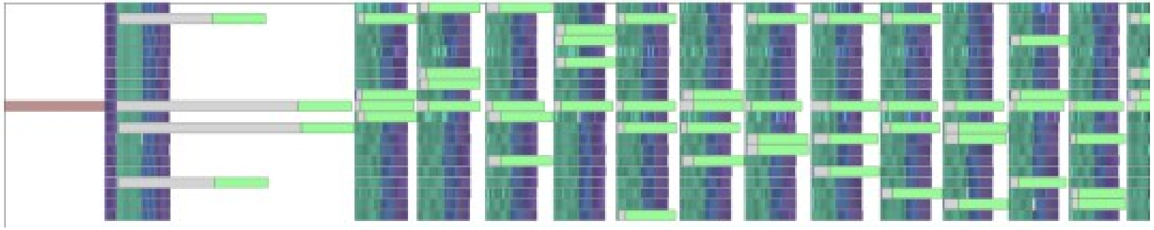


SLATE 1 GPU Trace

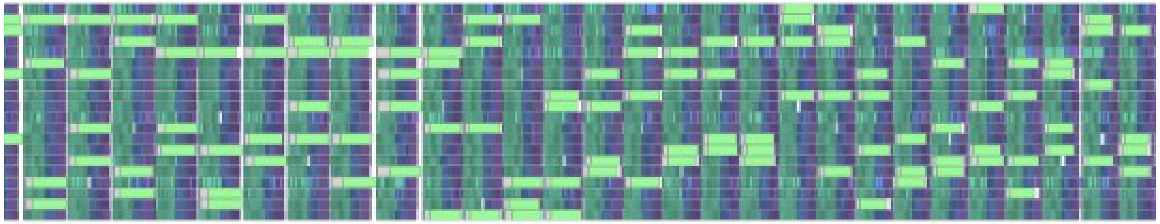


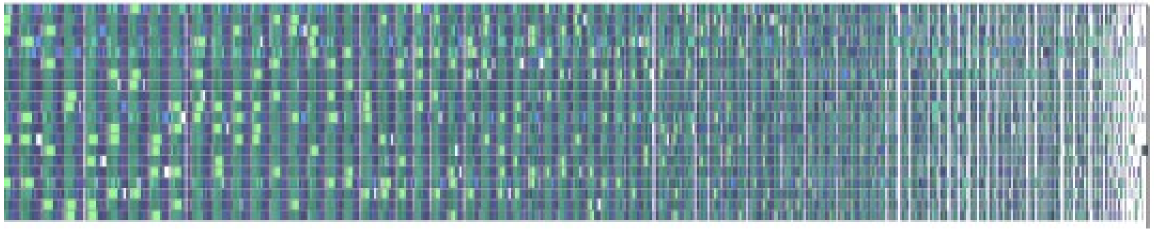
Cholesky factorization
20 cores + 1 GPU
56 K × 56 K matrix
tile size of 512

SLATE 4 GPU Trace

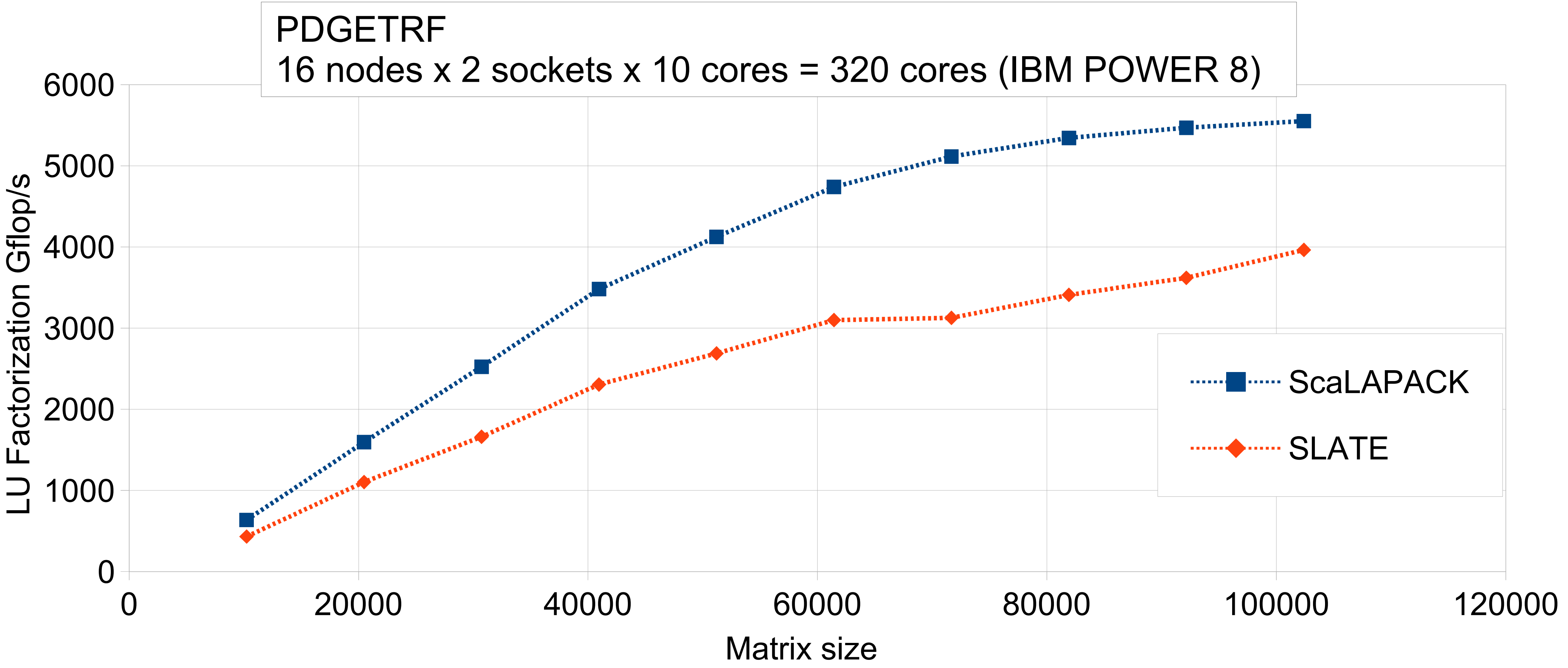


Cholesky factorization
20 cores + 4 GPUs
112 K × 112 K matrix
tile size of 512

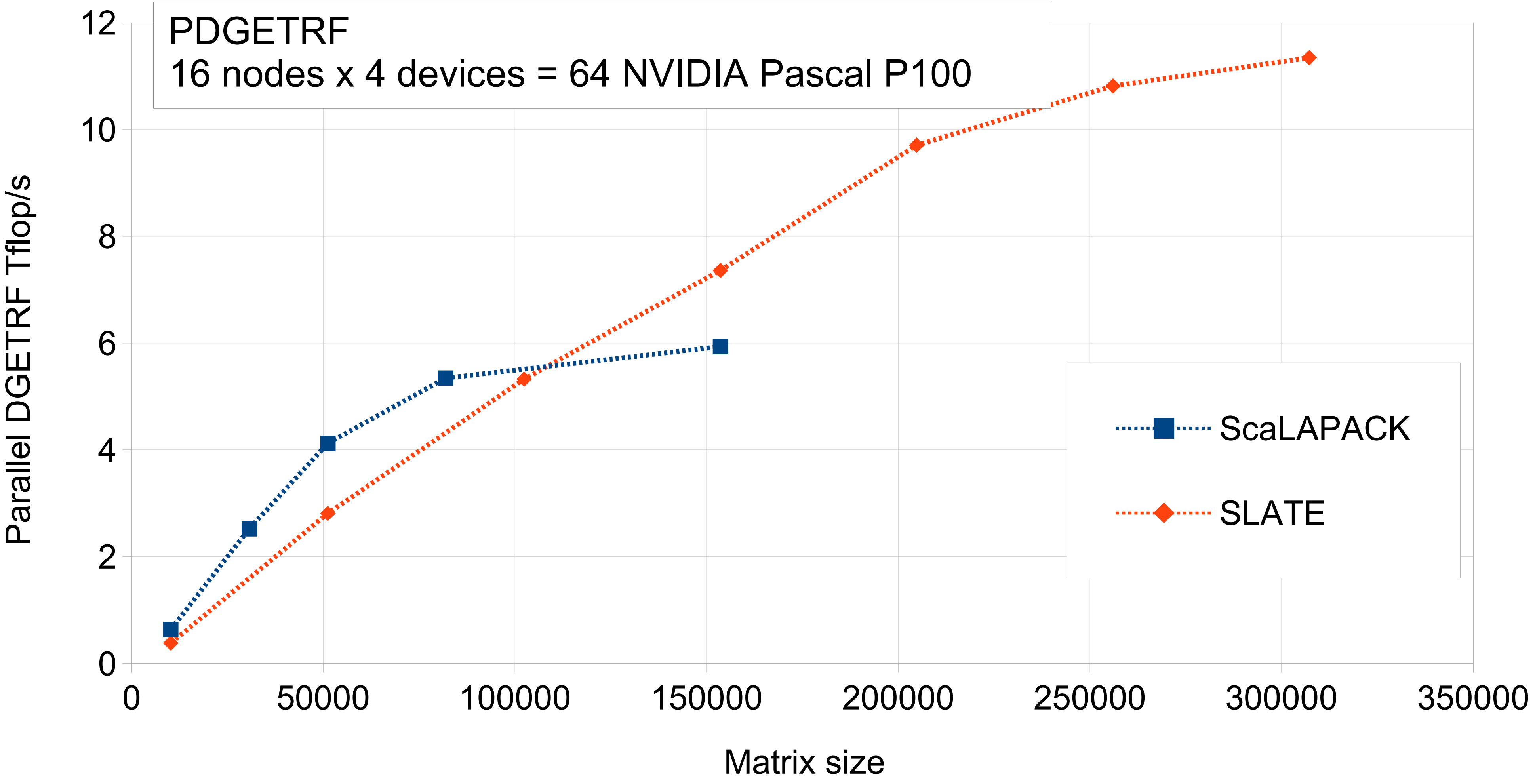




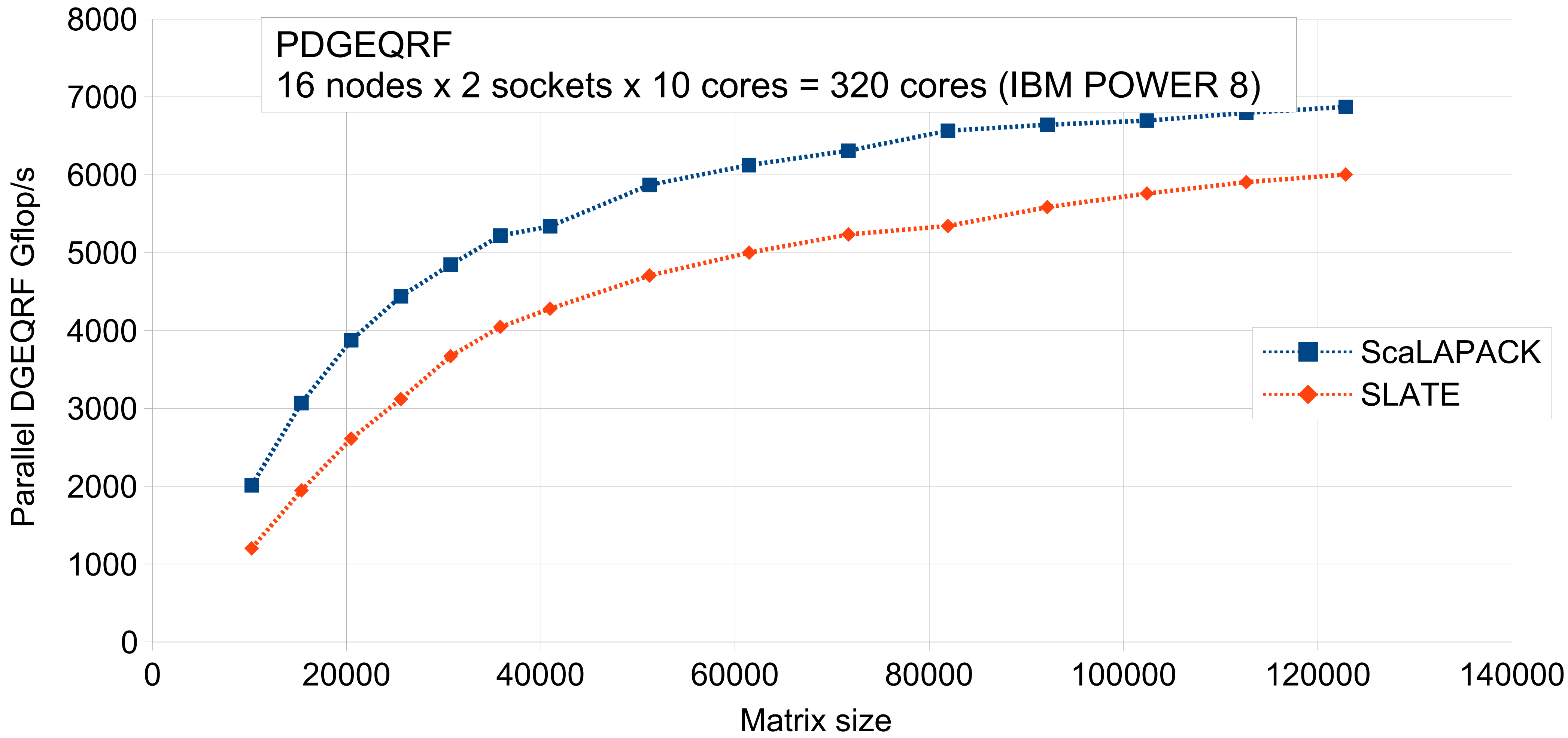
SLATE LU Performance: CPUs



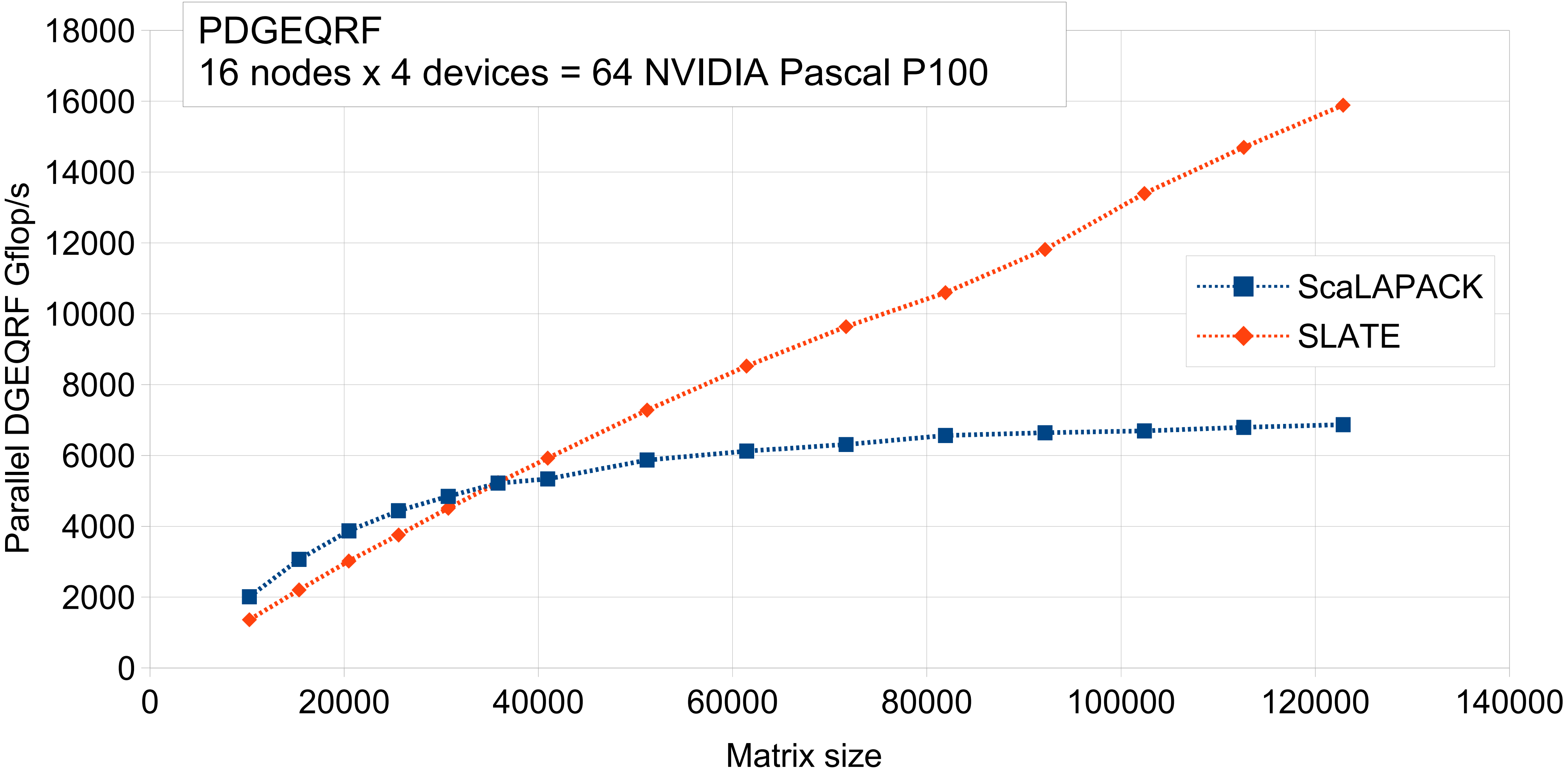
SLATE LU Performance: GPUs



SLATE QR Factorization Performance: CPUs



SLATE QR Factorization Performance: GPUs



SLATE Timeline: Past and Future

2016
Q1 *research*
Q2 *design*
Q3 *prototyping*

2017
Q4 C++ APIs for BLAS and LAPACK
Q1 parallel BLAS
Q2 parallel norms
Q3 linear systems (LU, LLT, LDLT)

2018
Q4 least squares (CA-QR/LQ)
Q1 mixed precision linear systems
Q2 matrix inversion
Q3 SVD

2019
Q4 EVP