Exceptional service in the national interest

Sandia National Laboratories

Performance Portable Line Smoother for Multiphysics Problems using Compact Batched BLAS

Kyungjoo Kim Mehmet Deveci Andrew M. Bradley
Simon D. Hammond Sivasankaran Rajamanickam

Center for Computing Research, Sandia National Labs

BLIS Retreat, Sep 18 - 19, 2017

U.S. DEPARTMENT OF ENERGY NNSA National Nuclear Security Administration CCR Center for Computing Research

Introduction: Line Smoother

Compact Data Layout

Numerical Experiments

Summary

Sandia National Laboratories

Line Smoother

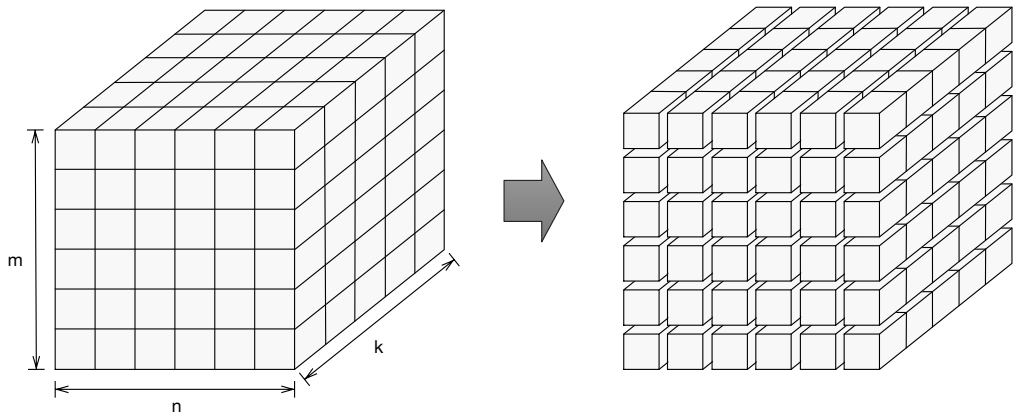
Line Smoother

- Consider a block sparse system of equations $Ax = b$ arising from coupled PDEs.
- By splitting $A = M - S$, we obtain

$$\begin{aligned}(M-S)x &= b \\ Mx &= b + Sx \\ x &= M^{-1}(b+Sx) = x + M^{-1}(b-Ax),\end{aligned}$$

  where M is a set of block tridiagonal matrices corresponding to the extracted lines of elements.
- Solve this iteratively

$$x^{k+1} = M^{-1}(b + Sx^k).$$



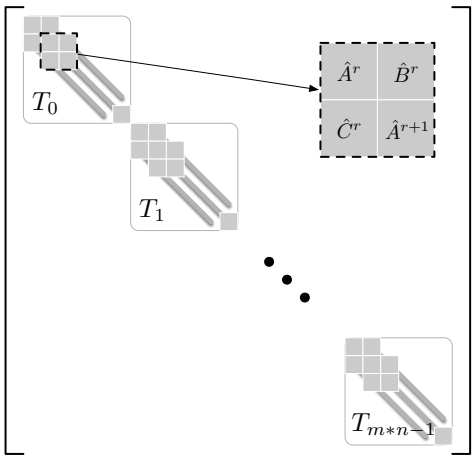
A set of line elements extracted along the stream line direction.

Considerations for Line Smoother Implementation

- A fixed number of this iteration is used as preconditioner.

$$x^{k+1} = M^{-1}(b + Sx^k).$$

- M is factorized once per solution (or each non-linear iteration) and applied multiple times.
- Typical blocksizes b are 3, 5, 9 and 15, which are related to scientific applications e.g., elasticity, ideal gas and multiphysics fluid problems.
- Limit memory usage up to 16 GB i.e., MCDRAM on KNL and GPU device memory.
- With this memory constraint, typical local problem sizes ($m \times n \times k$) are selected as $128 \times 128 \times 128$ for $b = 3, 5$ and $64 \times 64 \times 128$ for $b = 10, 15$.
- Batch parallelism: running a sequential algorithm within `parallel for`.



1 **for** T *in* $\{T_0, T_1, \cdots, T_{m\times n-1}\}$ **do in parallel**
2 **for** $r \leftarrow 0$ **to** $k-2$ **do**
3 $\hat{A}^r := LU(\hat{A}^r)$;
4 $\hat{B}^r := L^{-1}\hat{B}^r$;
5 $\hat{C}^r := \hat{C}^r U^{-1}$;
6 $\hat{A}^{r+1} := \hat{A}^{r+1} - \hat{C}^r\hat{B}^r$;
7 $\hat{A}^{k-1} := LU(\hat{A}^{k-1})$;

Initialization of the line smoother

Objective: Develop Performance Portable Line Preconditioner

Sandia National Laboratories

Architecture Constraints:

- Wide vector length (AVX512) of KNL for small problem sizes.
- A large number of CUDA threads (including vector lanes, 132K) on GPUs.
- Hierarchical parallelism to improve concurrency for solving each tridiagonal system.
- Different data access patterns for different architectures; core-affinity access on KNL vs coalesced access on GPUs.

Problems Using Vendor DLA Libraries

Sandia National Laboratories

Batched BLAS

- Batched BLAS computes many dense problems in parallel.
- Conceptually, similar to BLAS calls within `parallel for`.
- On the line smoother context: batched tridiagonal factor/solve using a sequence of batch operations is **not** efficient.
  - No data locality is exploited in the sequence of batch calls.
  - In each `parallel for`, synchronization is implicitly imposed.

BLAS within OpenMP

- BLAS is not optimized for 3×3 matrix computations.
- cuBLAS provides batch interface only.

Sandia National Laboratories

Compact Data Layout

Compact Data Layout (SIMD type)

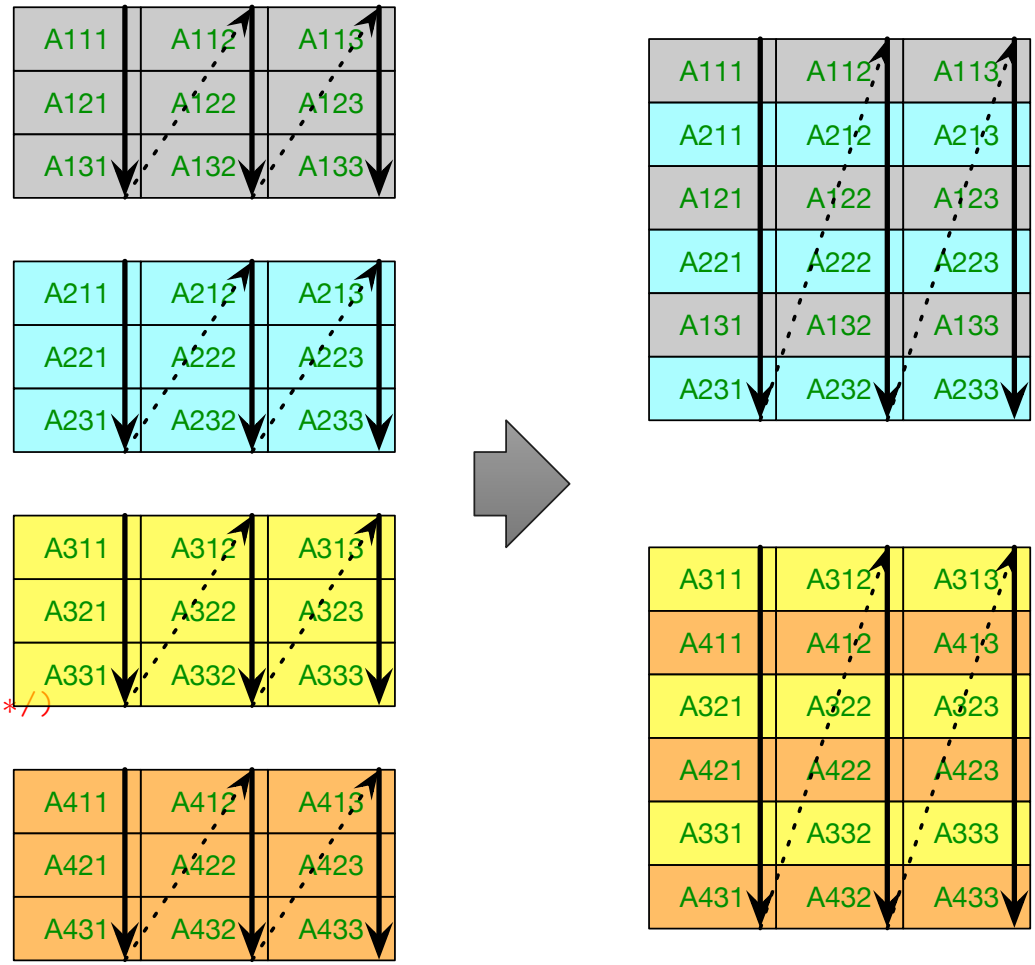
- Blocks are too small to use a wide vector units.

→ Vectorize across batch instances.

- Compact BLAS uses packed vectors as computing unit instead scalar.
- By overloading arithmatic operators, the scalar BLAS are reused and naturally vectorized.

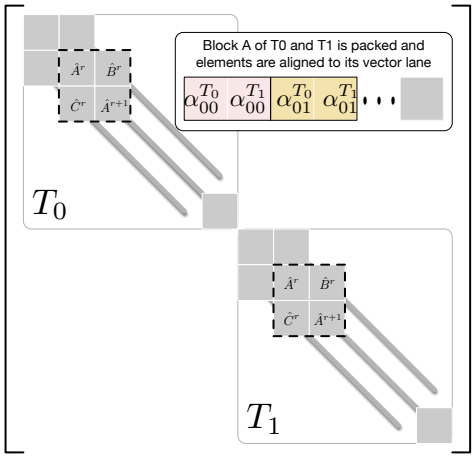
```
// computing unit
struct VectorAVX512D {
  union {
    __m512d v;
    double s[8];
  };
};

// overload arithmatic operators (+-*/)
VectorAVX512D
operator+(VectorAVX512D const &a,
          VectorAVX512D const &b) {
  return __mm512_add_pd(a, b);
}
```



In collaboration with Intel MKL team, Compact Batched BLAS standard is adopted as new batch interface.

Line Smoother Impl. using Compact Data Layout



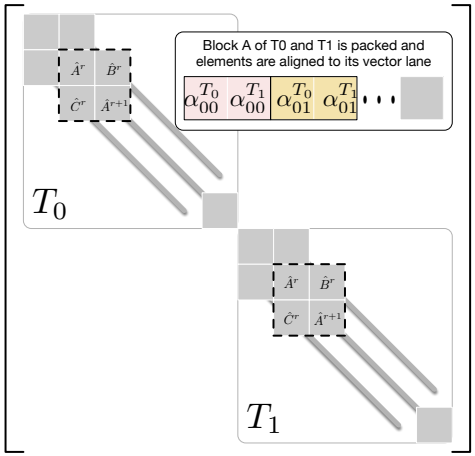
1 **for** *a pair T in* $\{(T_0,T_1),(T_2,T_3),\cdots,(T_{m\times n-2},T_{m\times n-1})\}$ **do in parallel**
2 **for** $r \leftarrow 0$ **to** $k-2$ **do**
3 $\hat{A}^r := LU(\hat{A}^r)$;
4 $\hat{B}^r := L^{-1}\hat{B}^r$;
5 $\hat{C}^r := \hat{C}^r U^{-1}$;
6 $\hat{A}^{r+1} := \hat{A}^{r+1} - \hat{C}^r\hat{B}^r$;
7 $\hat{A}^{k-1} := LU(\hat{A}^{k-1})$;

Layered APIs for Kokkos Hierarchical Parallelism

- *Serial (Vector)*: a single thread is mapped to a single instance of parallel batch.
- *Team*: a team of threads is mapped to a single instance of parallel batch.
- *Device*: the entire device used as like current batch interface provided by vendors.

Serial and Team APIs are used to compose batched block tridiagonal factorization/solve.

Line Smoother Impl. using Compact Data Layout



1 **for** *a pair* T *in* $\{(T_0,T_1),(T_2,T_3),\cdots,(T_{m\times n-2},T_{m\times n-1})\}$ **do in parallel**
2 **for** $r \leftarrow 0$ **to** $k-2$ **do**
3 $\hat{A}^r := LU(\hat{A}^r)$;
4 $\hat{B}^r := L^{-1}\hat{B}^r$;
5 $\hat{C}^r := \hat{C}^r U^{-1}$;
6 $\hat{A}^{r+1} := \hat{A}^{r+1} - \hat{C}^r\hat{B}^r$;
7 $\hat{A}^{k-1} := LU(\hat{A}^{k-1})$;

Some Issues

- As it perform cross-matrix vectorization, pivoting in LU is not feasible.

→ For preconditioning purpose, this does not matter.

- There is repacking overhead when the standard format is used.

→ Block tridiagonal matrices are extracted and repacked at the same time.

Sandia National Laboratories

Numerical Experiments

Test Setup

Sandia National Laboratories

Intel Knight Landing

- 34 Tiles, 2 Cores (4 Threads/core), 2x AVX512 units/core, 1MB L2
- 3+ TFLOPs in double precision, 400+ GB/s (MCDRAM)

NVIDIA Tesla P100

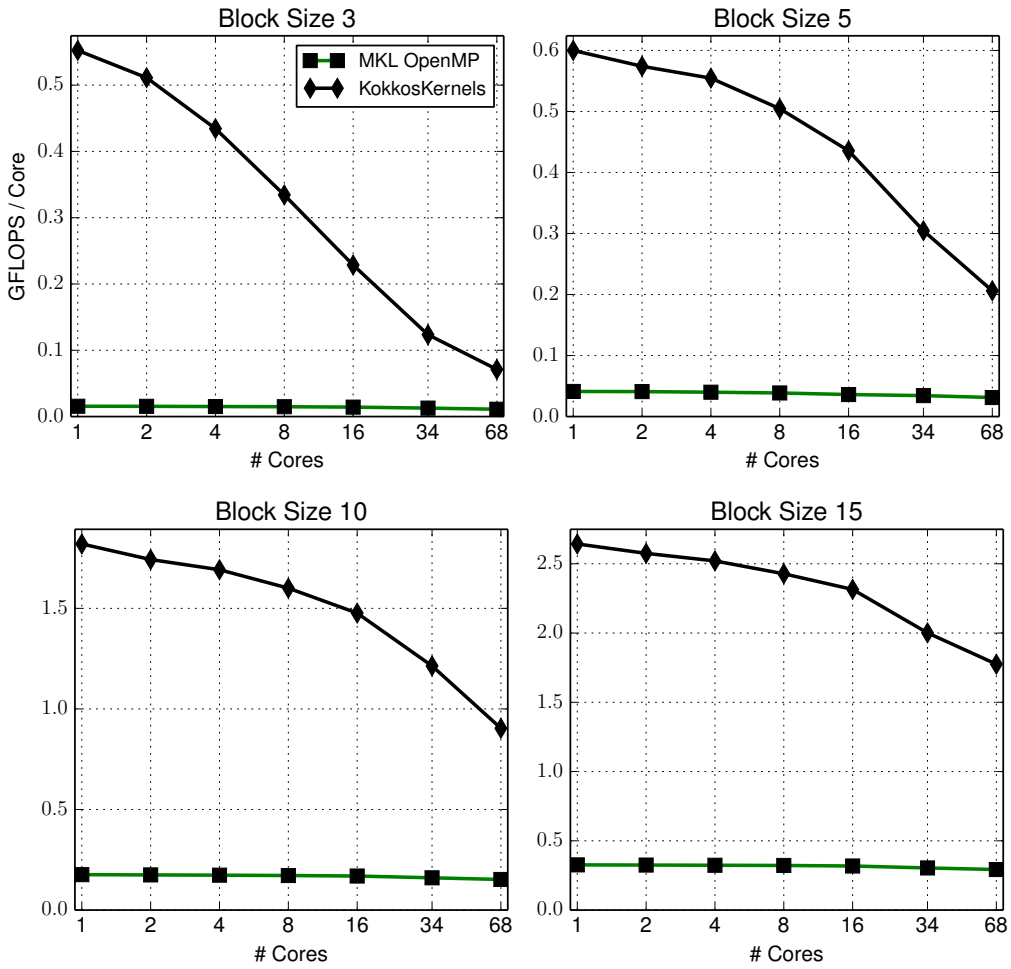
- 56 SMs, 32 FP64 CUDA Cores/SM
- 4.7 TFLOPs in double precision, 732+ GB/s (HBM)

Bechmark

- Compact batched BLAS is implemented in KokkosKernels.
- We evaluate our compact batched BLAS against vendor optimized BLAS implementations; 1) MKL with OpenMP, 2) MKL batched APIs and 3) cuBLAS batched APIs.
- Line smoother is implemented using our compact batched BLAS serial and team interfaces and compared with SPARC.
- SPARC is Sandia production simulation code for solving Navier-Stokes equations for compressible and reacting flows.

Intel Knight Landing: Batched BLAS

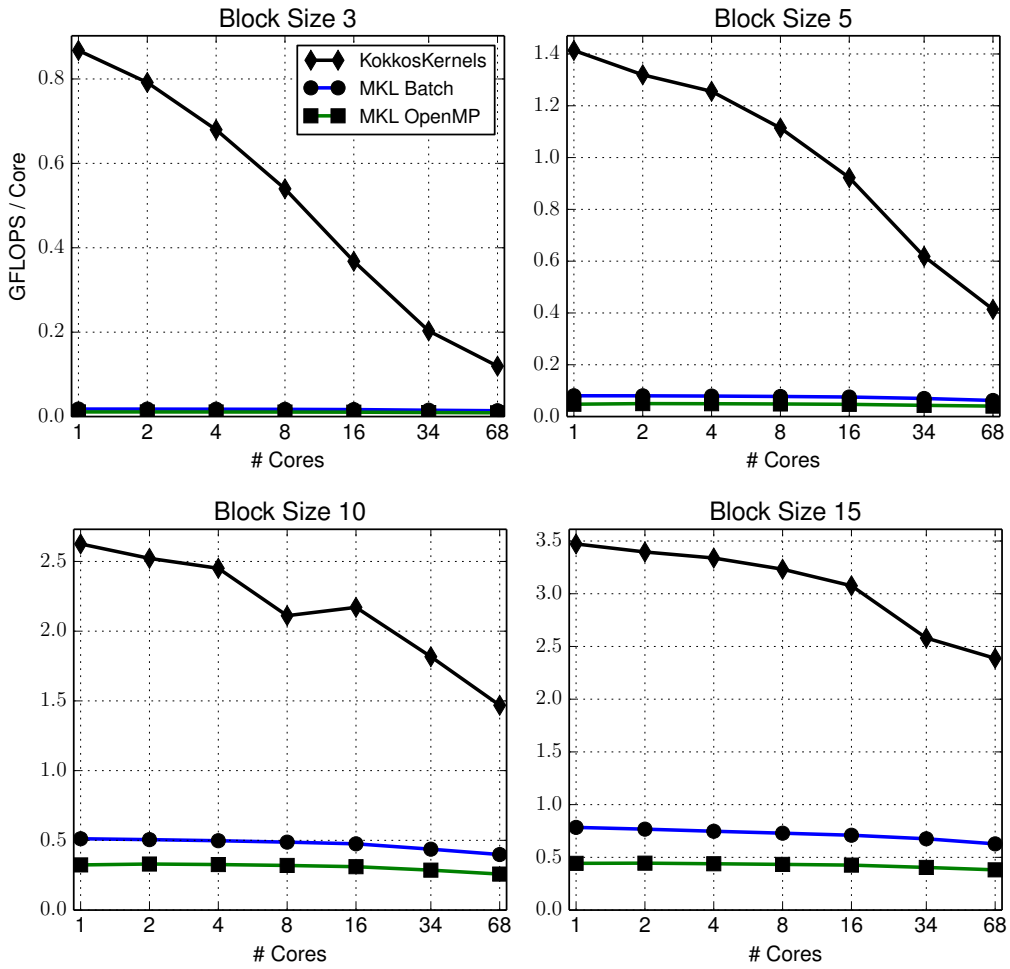
- SIMD data layout allows vectorization for small blocksizes.



Batched LU

Intel Knight Landing: Batched BLAS

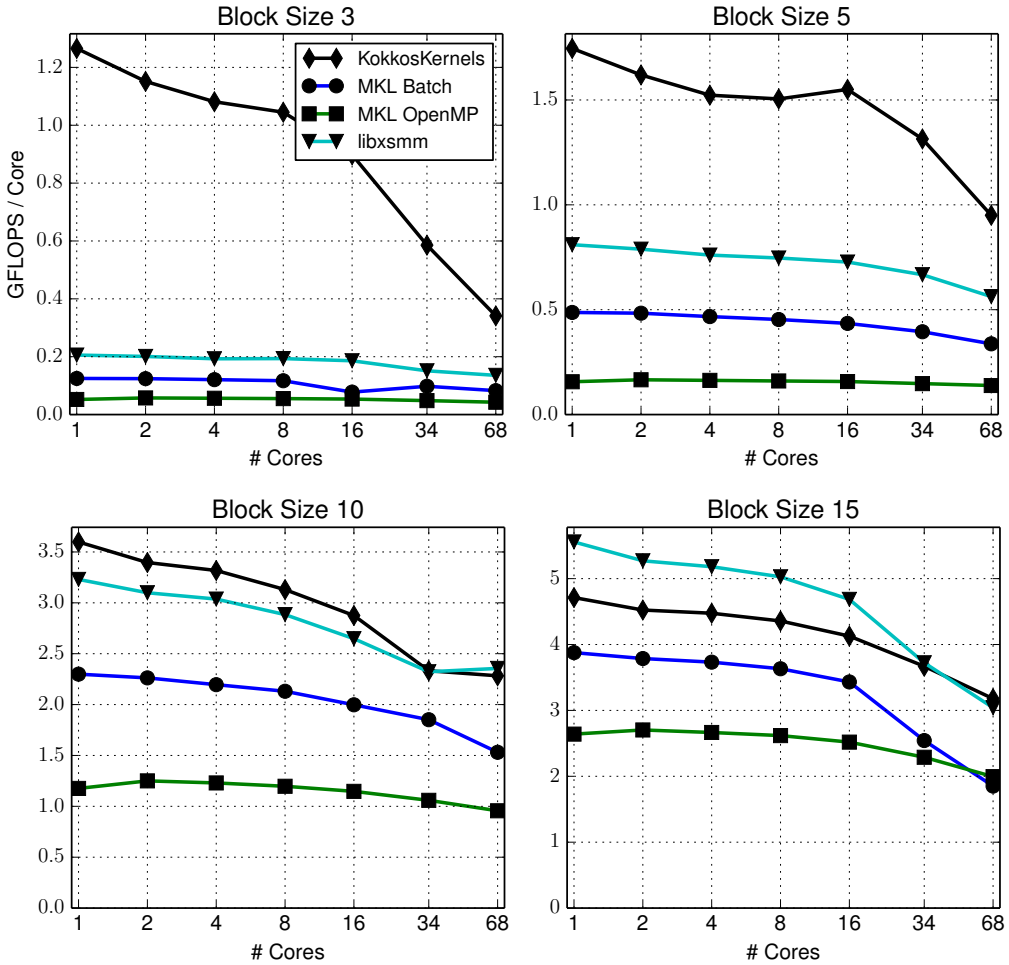
- SIMD data layout allows vectorization for small blocksizes.



Batched Trsm

Intel Knight Landing: Batched BLAS

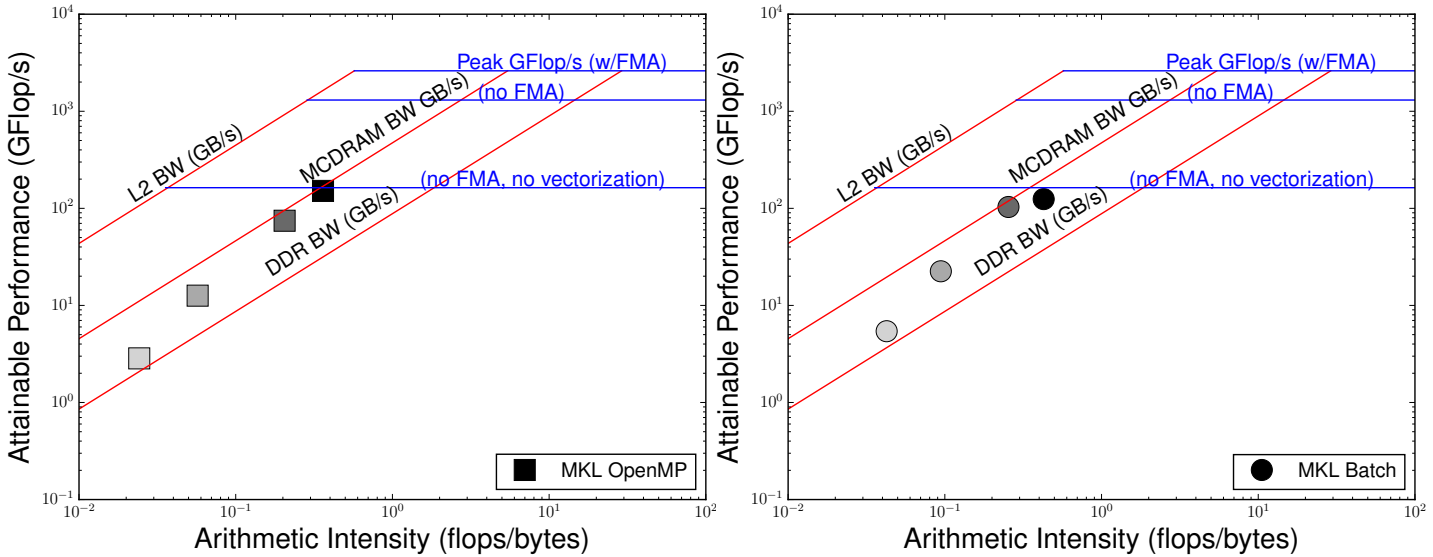
- SIMD data layout allows vectorization for small blocksizes.



Batched Gemm

Intel Knight Landing: Roofline Analysis

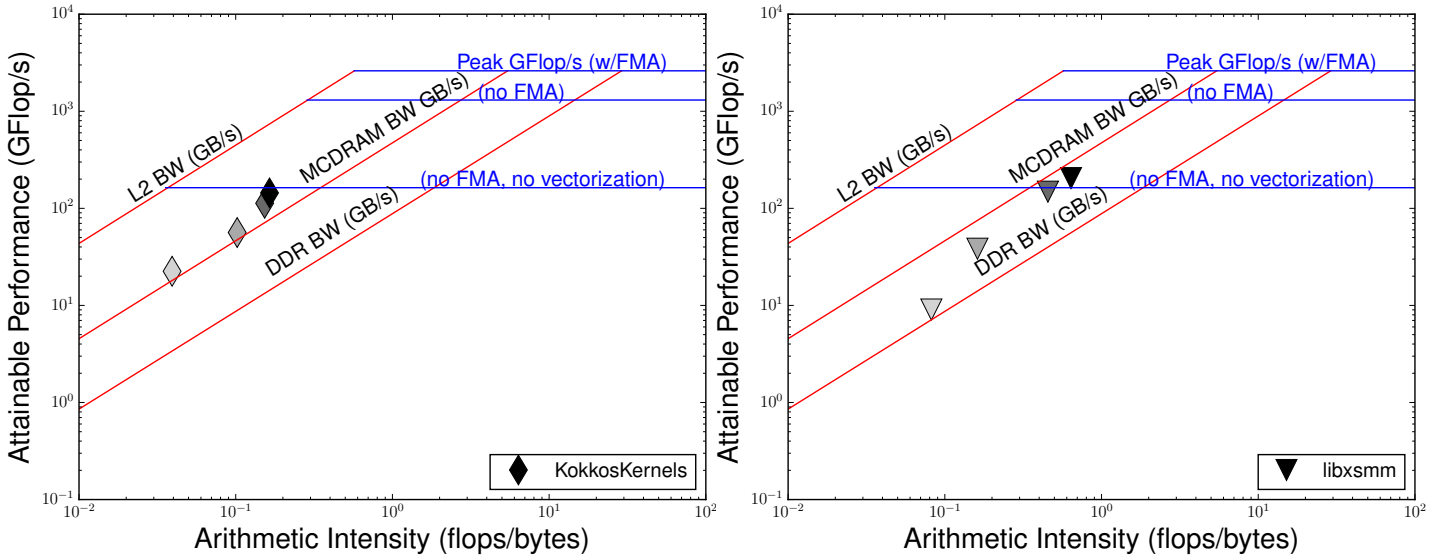
- APEX toolkit (developed in Sandia) is used for performance analysis.
- Memory bandwidth bounded compute characteristics.
- Our implementation fully exploits MCDRAM memory bandwidth.



Batched Gemm

Intel Knight Landing: Roofline Analysis

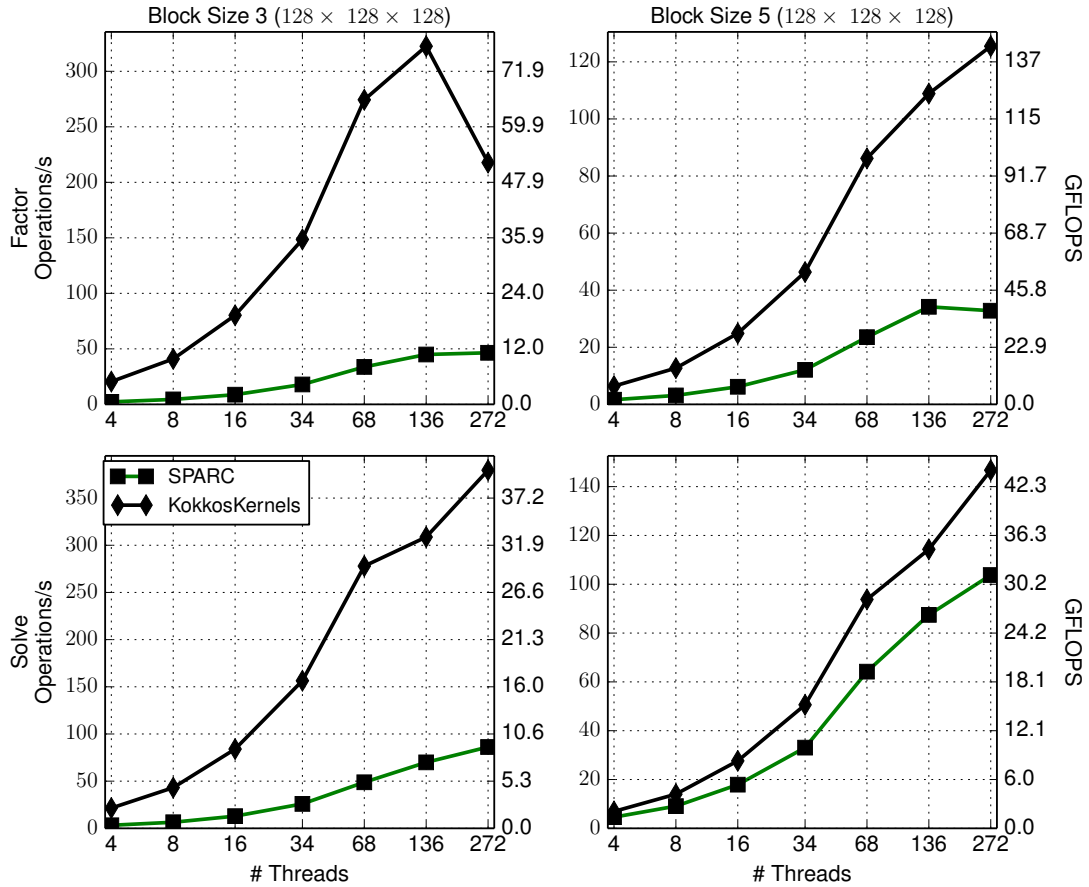
- APEX toolkit (developed in Sandia) is used for performance analysis.
- Memory bandwidth bounded compute characteristics.
- Our implementation fully exploits MCDRAM memory bandwidth.



Batched Gemm

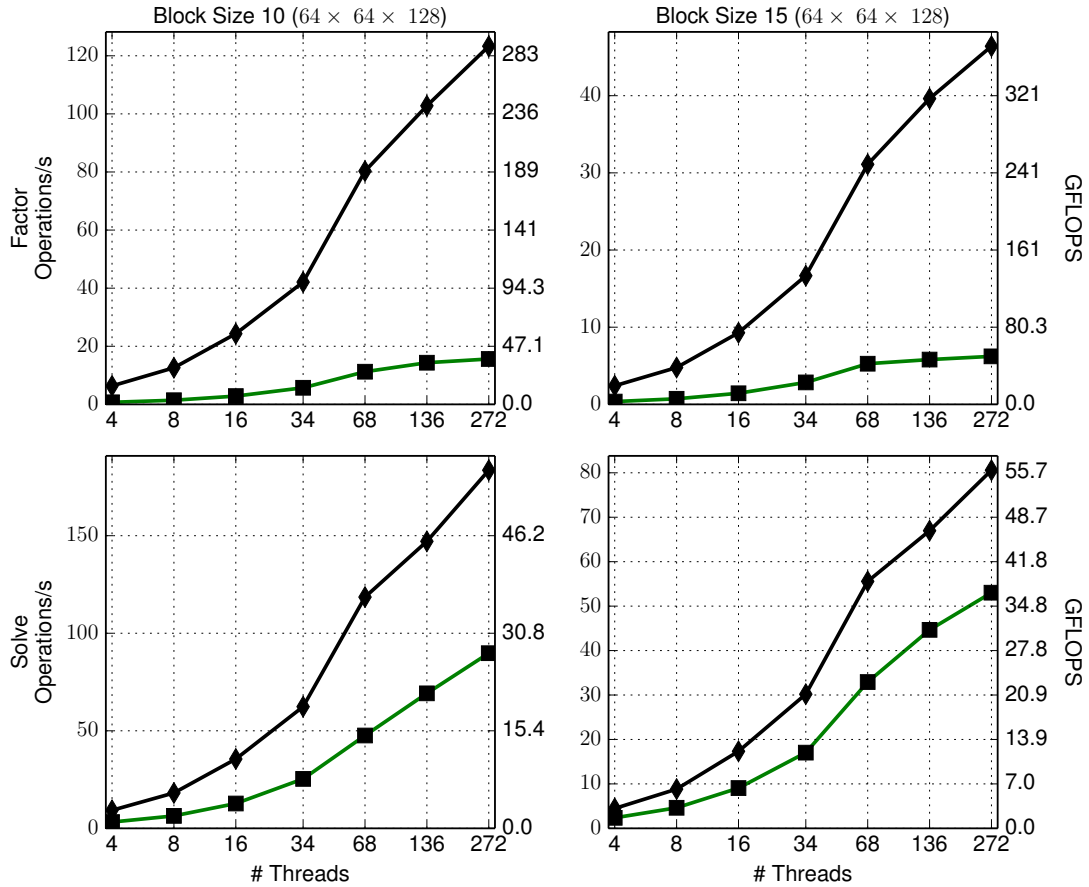
Intel Knight Landing: Line Smoother

- Our batch block tridiagonal solver with compact data layout is purely vectorized regardless of block sizes.
- # of batch size ($m \times n$) is 16,384 and the length of a block tridiagonal matrix (k) is 128.



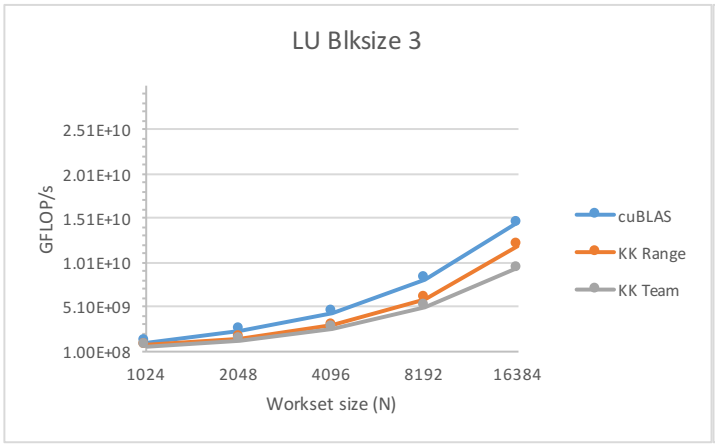
Intel Knight Landing: Line Smoother

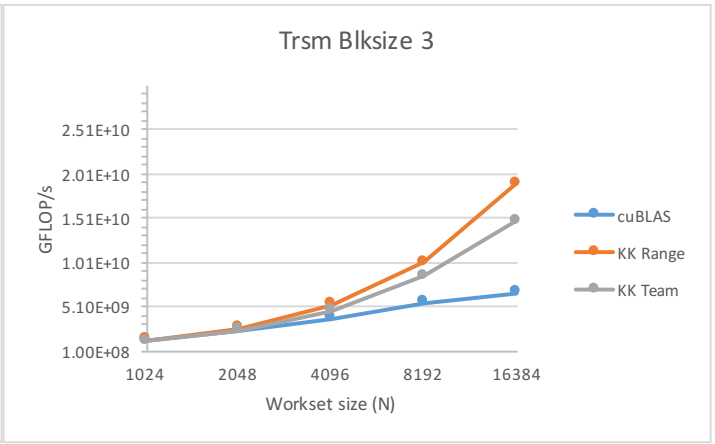
- Our batch block tridiagonal solver with compact data layout is purely vectorized regardless of block sizes.
- # of batch size ($m \times n$) is 4096 and the length of a block tridiagonal matrix (k) is 128.

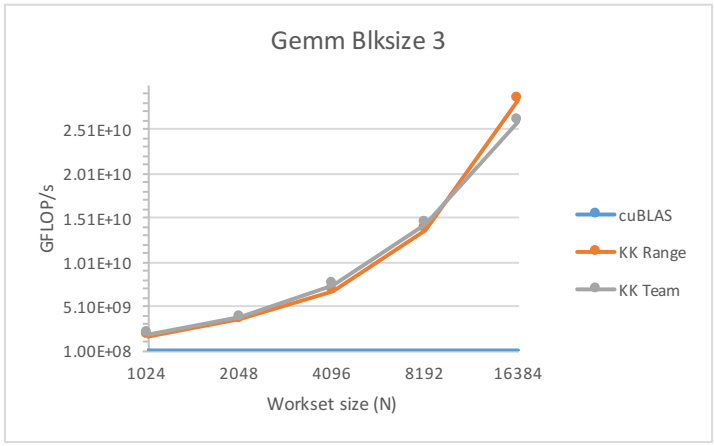


NVIDIA Tesla P100: Batched BLAS

- Device is not fully occupied with small workset sizes: 130k threads vs 16k batch size.
- Flat parallelism (called `KK Range`) vs Hierarchical parallelism (called `KK Team`).
- Team parallelism is essential for performance; however, identifying right team size is challenging.

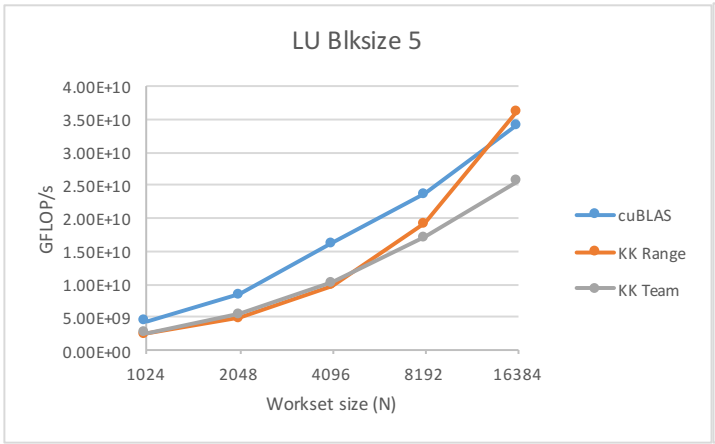


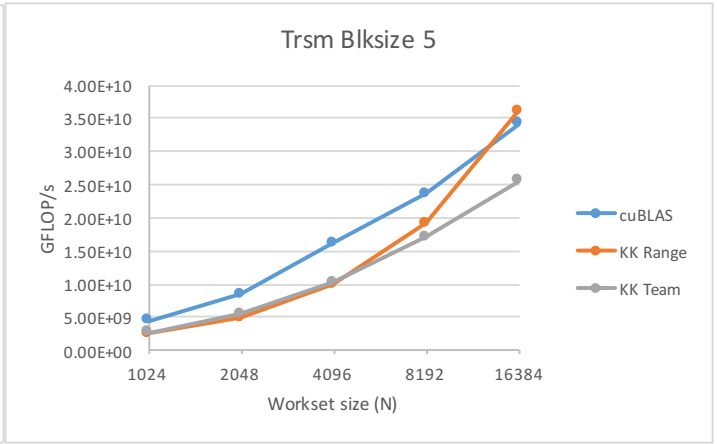


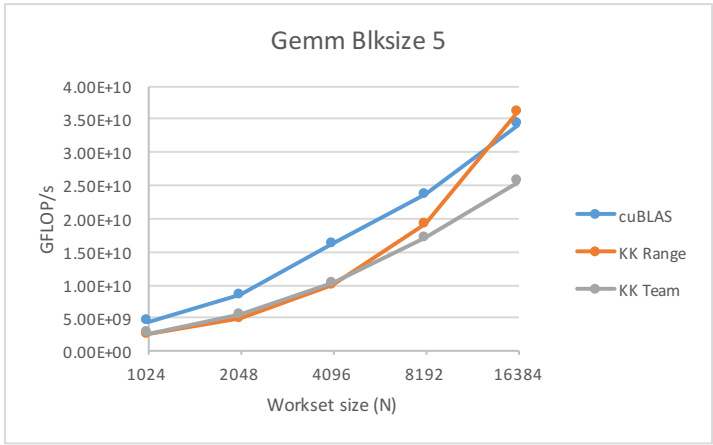


NVIDIA Tesla P100: Batched BLAS

- Device is not fully occupied with small workset sizes: 130k threads vs 16k batch size.
- Flat parallelism (called `KK Range`) vs Hierarchical parallelism (called `KK Team`).
- Team parallelism is essential for performance; however, identifying right team size is challenging.

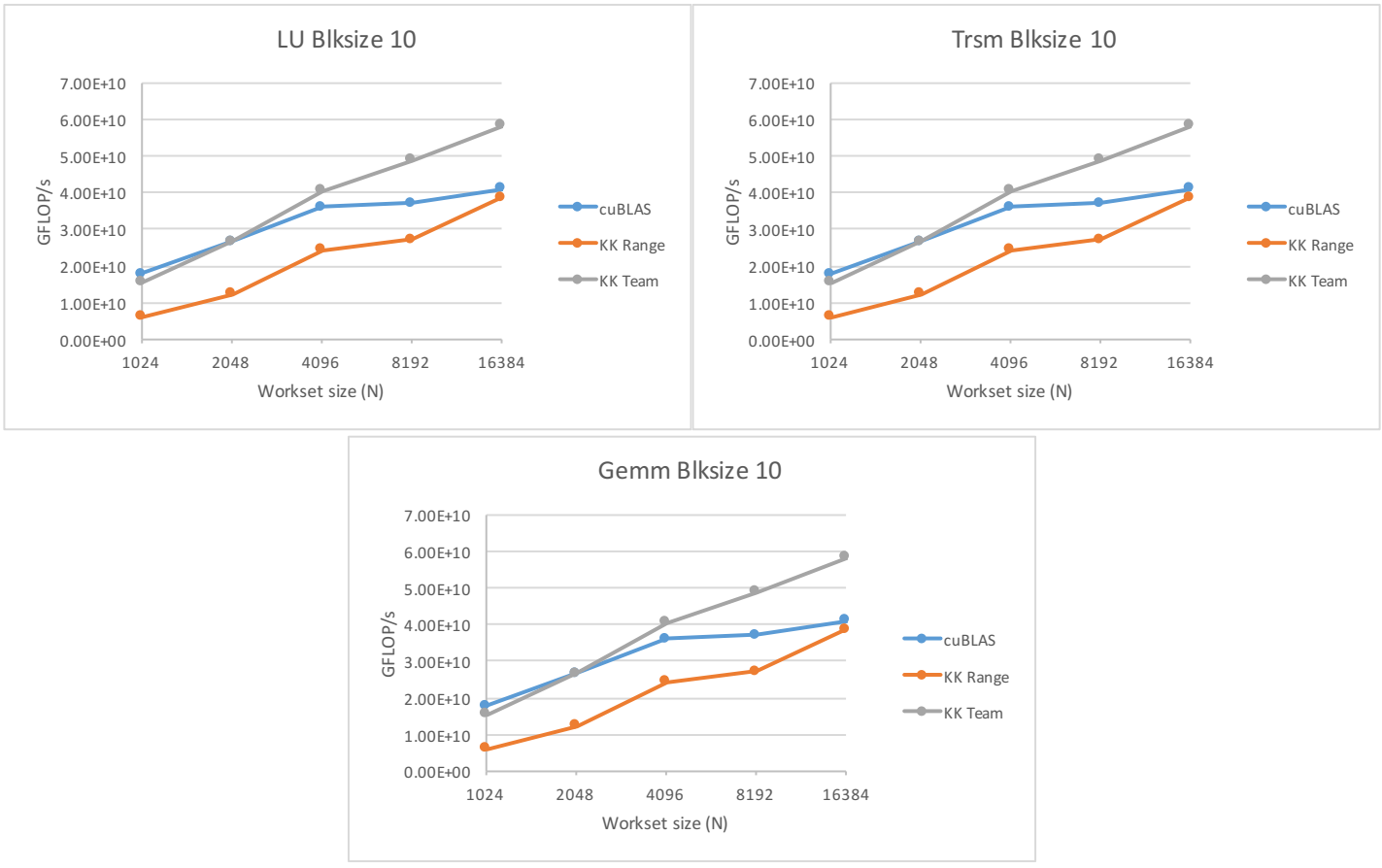






NVIDIA Tesla P100: Batched BLAS

- Device is not fully occupied with small workset sizes: 130k threads vs 16k batch size.
- Flat parallelism (called `KK Range`) vs Hierarchical parallelism (called `KK Team`).
- Team parallelism is essential for performance; however, identifying right team size is challenging.

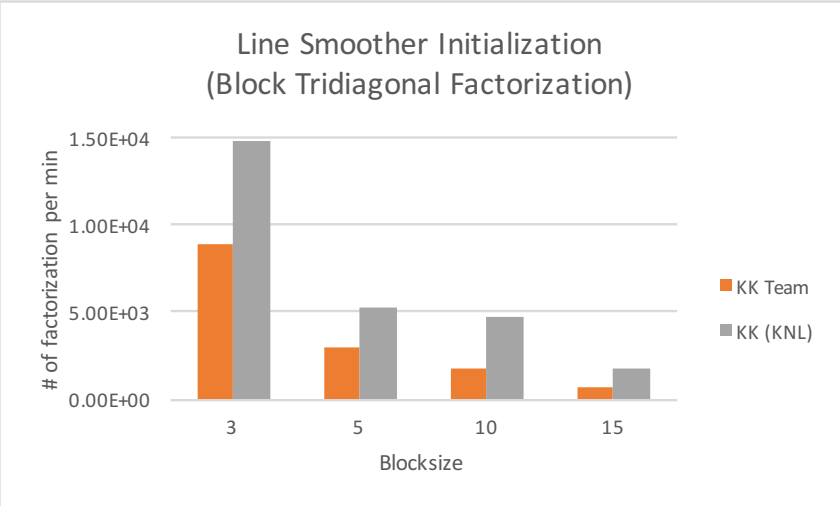


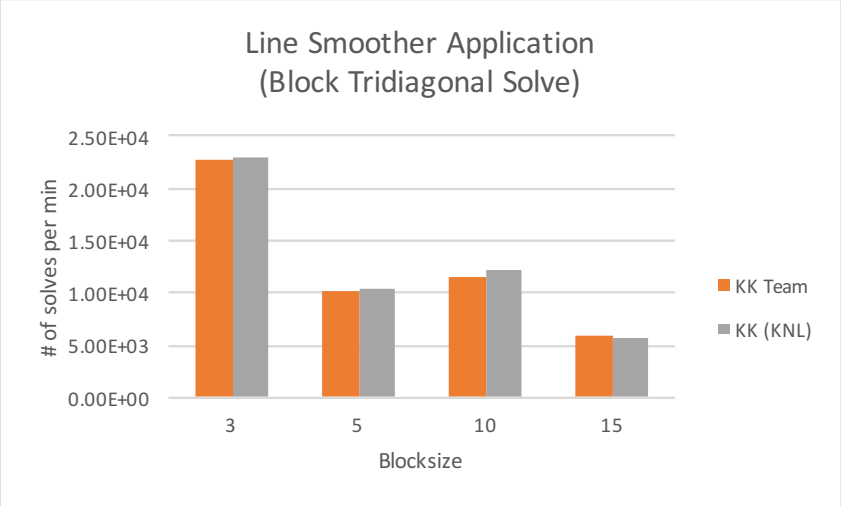
NVIDIA Tesla P100: Batched BLAS

- Device is not fully occupied with small workset sizes: 130k threads vs 16k batch size.
- Flat parallelism (called `KK Range`) vs Hierarchical parallelism (called `KK Team`).
- Team parallelism is essential for performance; however, identifying right team size is challenging.

NVIDIA Tesla P100: Line Smoother

- Native version of SPARC does not include GPU implementation of line smoother.
- KokkosKernels provide portable implementation for two architectures i.e., KNL and P100, and demonstrate comparable performance each other.





Summary

- Demonstrate performance portable block line smoother implementation.
- Numerical experiments are presented for Intel Knight Landing and NVIDIA P100.
- Functor-level interfaces (serial and team) are used as building blocks for implementing the line smoother.
- By packing data aligned with hardware specific vector length, codes are easily vectorized.