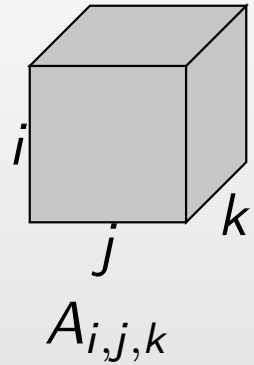
TTC: A Compiler for Tensor Transpositions

Paul Springer and Paolo Bientinesi

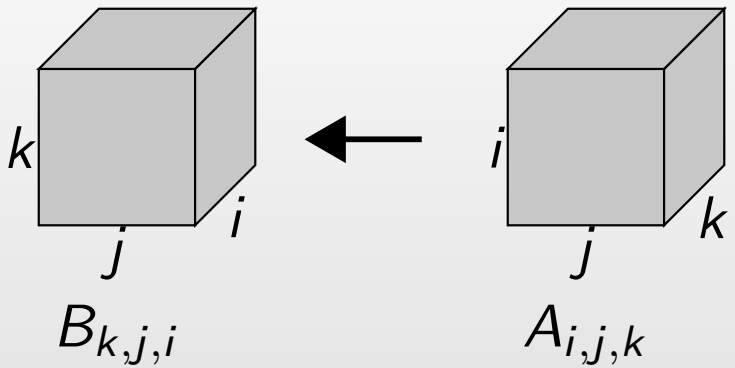
Aachen Institute for Advanced Study in
Computational Engineering Science

Paris, 14.04.16 — SIAMPP16

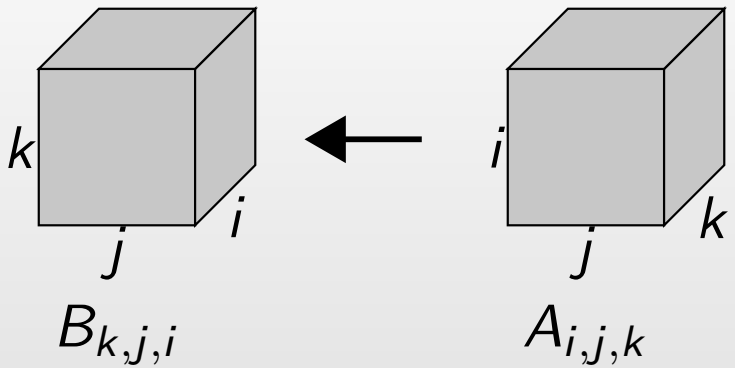
Tensor Transpositions



Tensor Transpositions

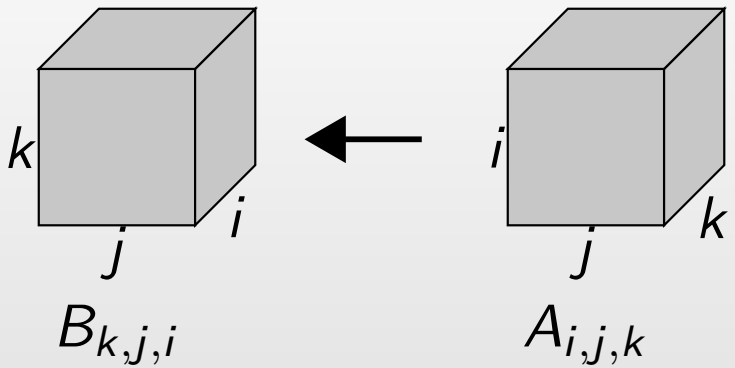


Tensor Transpositions



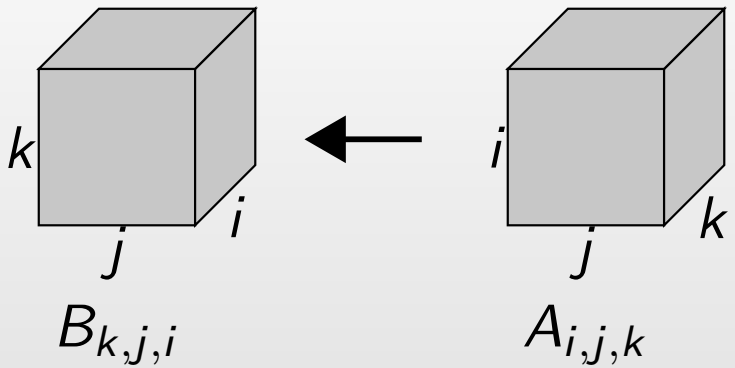
- Building block for tensor contractions
  - Map tensor contraction to GEMM

Tensor Transpositions



- Building block for tensor contractions
  - Map tensor contraction to GEMM
- Packing routines for dense linear algebra kernels
  - e.g., GEMM

Tensor Transpositions



- Building block for tensor contractions
  - Map tensor contraction to GEMM
- Packing routines for dense linear algebra kernels
  - e.g., GEMM
- Huge search space of viable implementations
  - Implementations vary greatly in performance

TTC: Tensor Transpose Compiler

RWTH AACHEN UNIVERSITY

- Domain-specific compiler
  - Multidimensional tensor transpositions

AICES

TTC: Tensor Transpose Compiler

RWTH AACHEN UNIVERSITY

- Domain-specific compiler
  - Multidimensional tensor transpositions
- Generates high-performance C++/CUDA C code
  - Parallelized
  - Vectorized

AICES

TTC: Tensor Transpose Compiler

RWTH AACHEN UNIVERSITY

- Domain-specific compiler
  - Multidimensional tensor transpositions
- Generates high-performance C++/CUDA C code
  - Parallelized
  - Vectorized
- Support for multiple architectures
  - NVIDIA GPUs, KNC (Xeon Phi), AVX-enabled CPUs

AICES

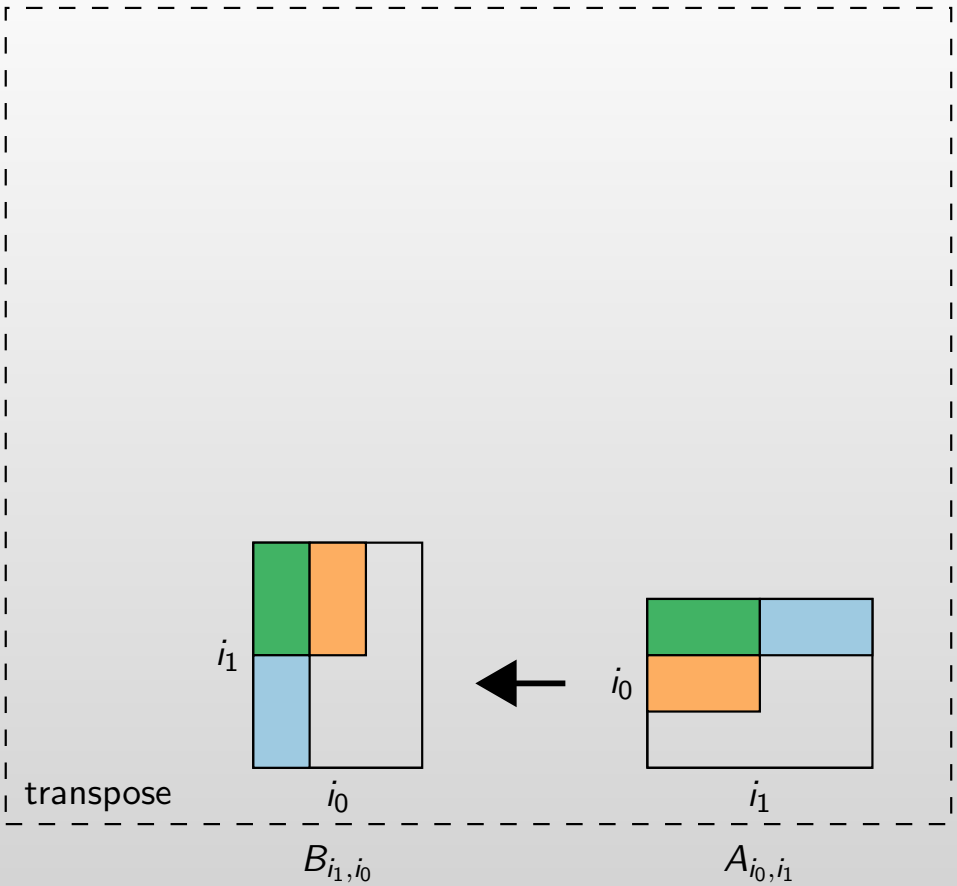
TTC: Tensor Transpose Compiler

RWTH AACHEN UNIVERSITY

- Domain-specific compiler
  - Multidimensional tensor transpositions
- Generates high-performance C++/CUDA C code
  - Parallelized
  - Vectorized
- Support for multiple architectures
  - NVIDIA GPUs, KNC (Xeon Phi), AVX-enabled CPUs
- Support for common numerical data types
  - single, double, single-complex, double-complex
  - mixed precision

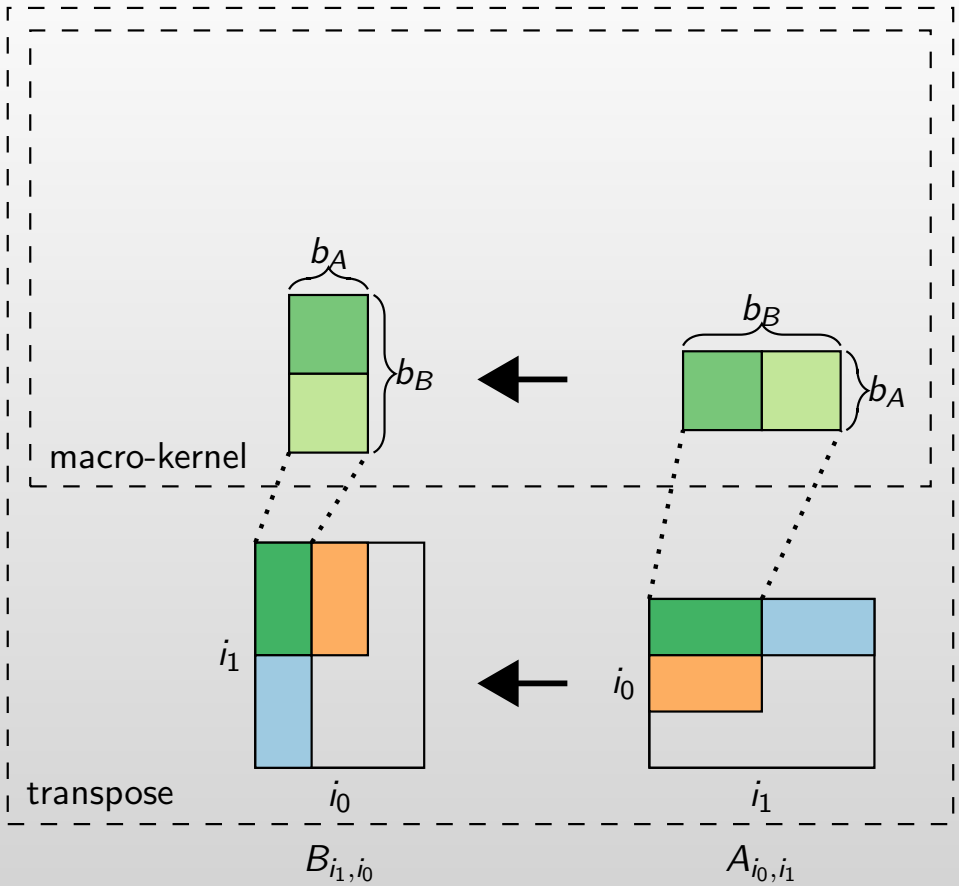
AICES

Design overview



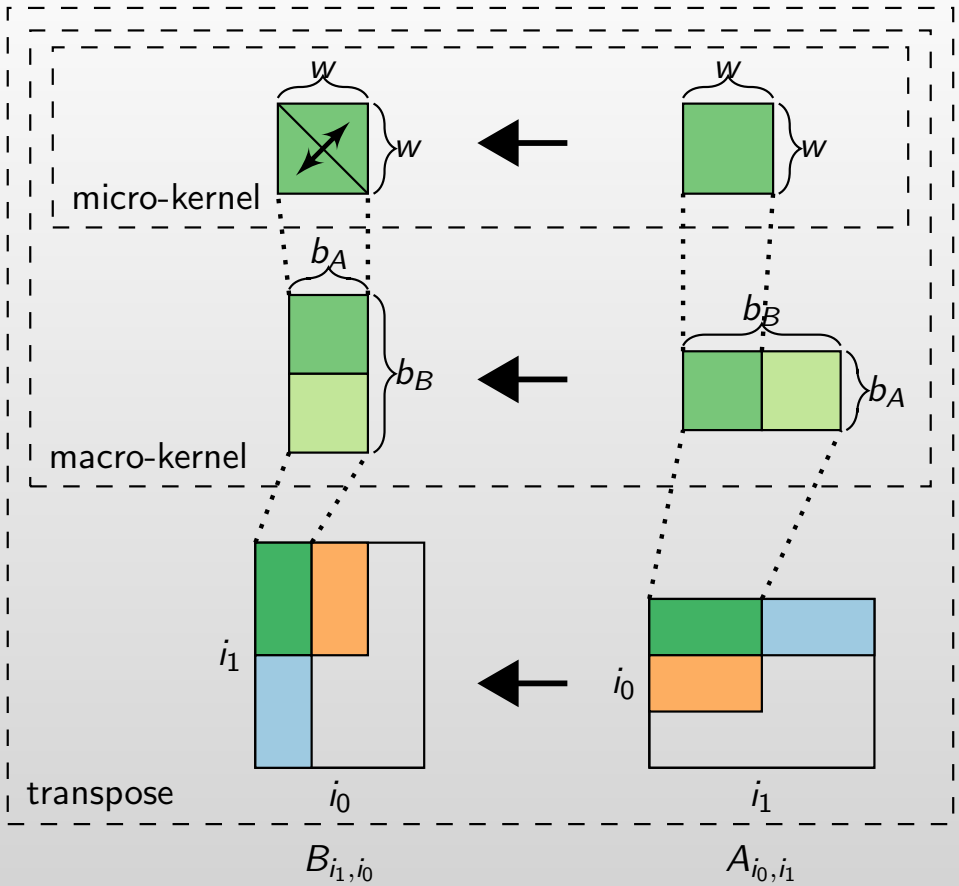
- Decompose a transposition into macro-tiles
  - Parallel over macro-tiles

Design overview



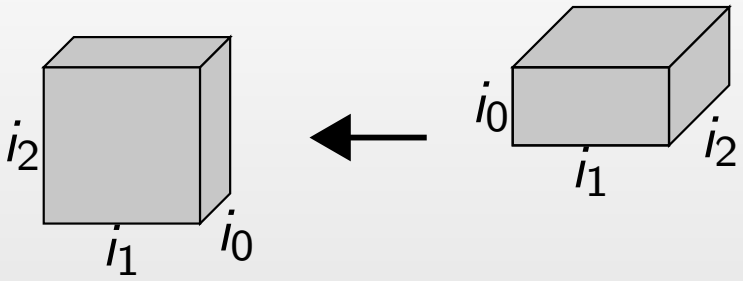
- Decompose a transposition into macro-tiles
  - Parallel over macro-tiles
- Decompose a macro-tile into micro-tiles
  - Vectorized micro-tiles

Design overview



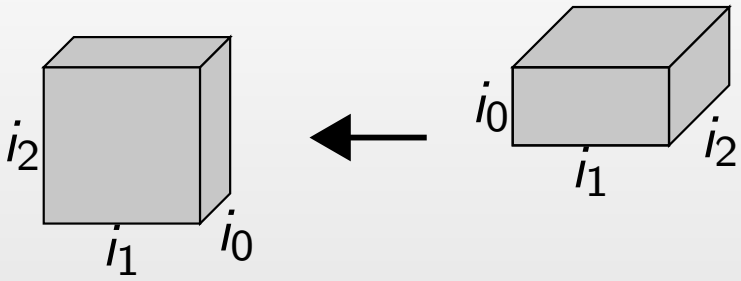
- Decompose a transposition into macro-tiles
  - Parallel over macro-tiles
- Decompose a macro-tile into micro-tiles
  - Vectorized micro-tiles

Example 1



B_{i_2,i_1,i_0} A_{i_0,i_1,i_2}

Example 1

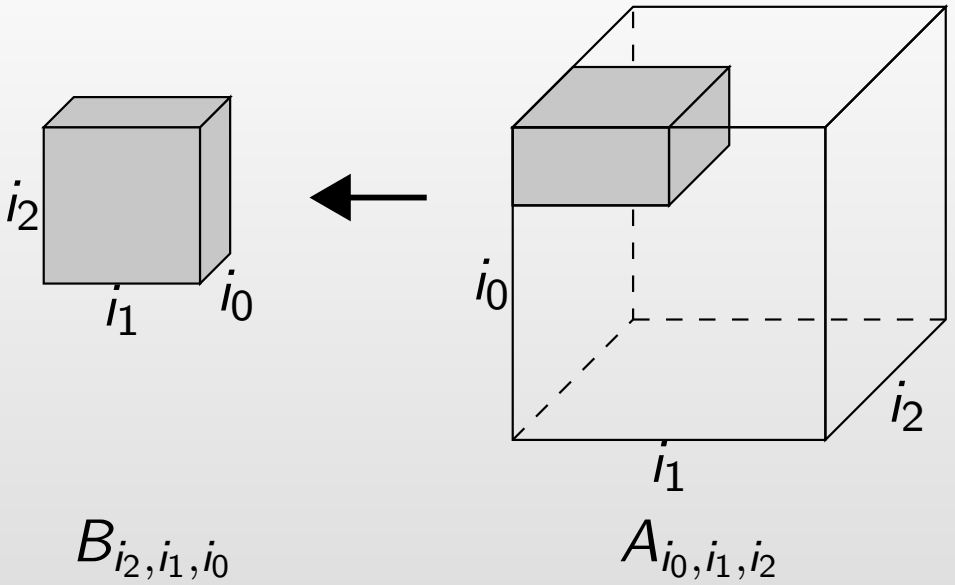


B_{i_2,i_1,i_0} A_{i_0,i_1,i_2}

- Input to TTC:

```
ttc --perm=2,1,0 --size=8,16,16
```

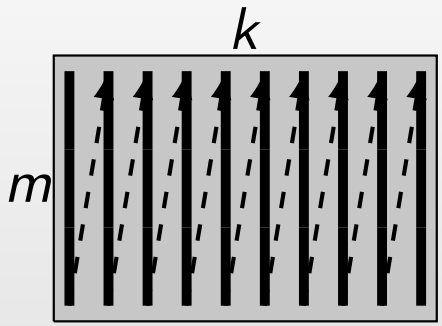
Example 1



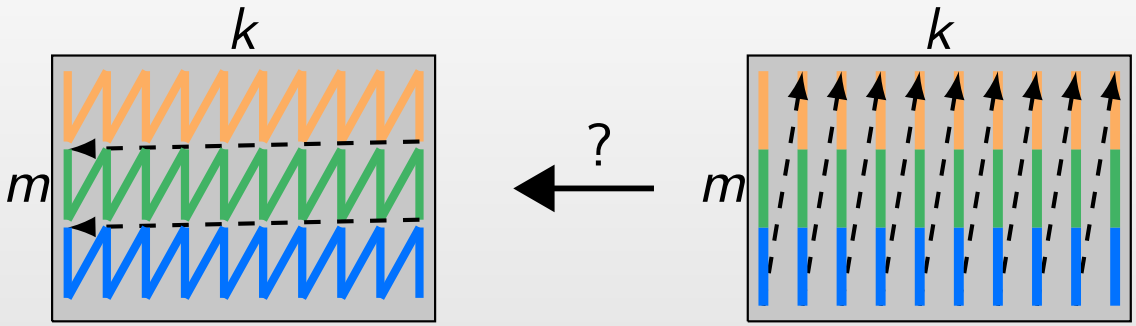
- Input to TTC:

```
ttc --perm=2,1,0 --size=8,16,16 --lda=32,32,32
```

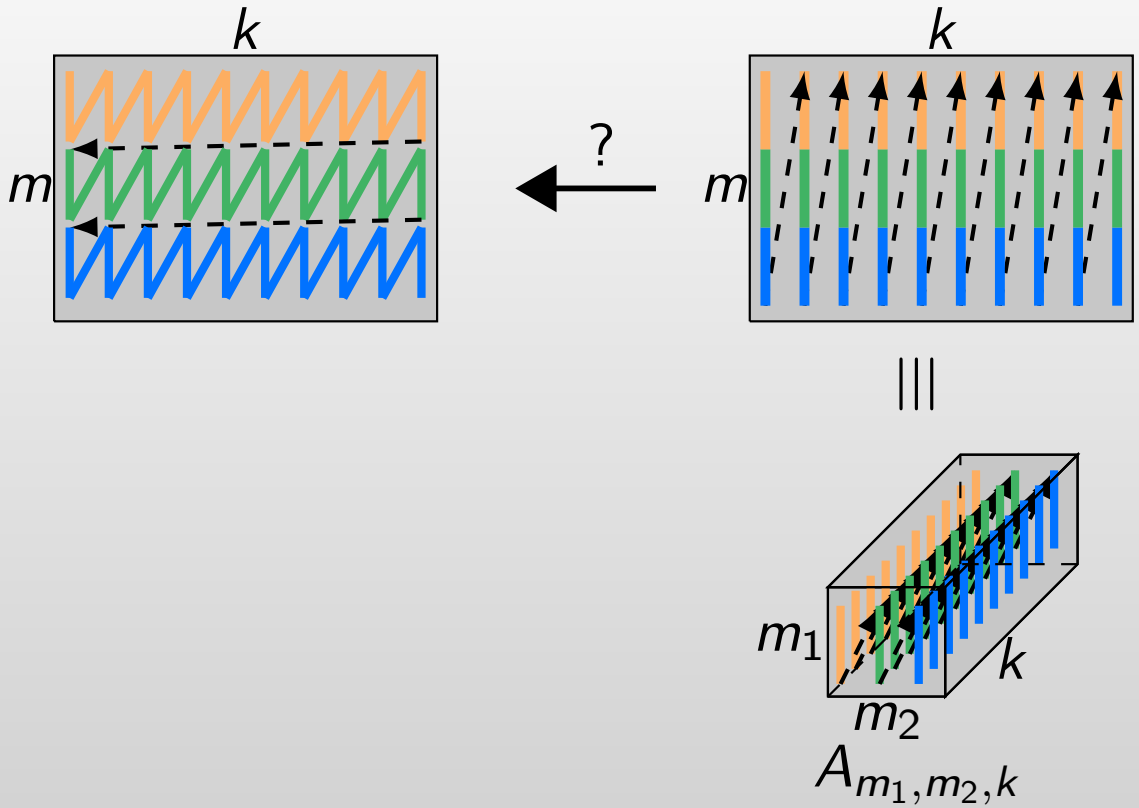
Example 2



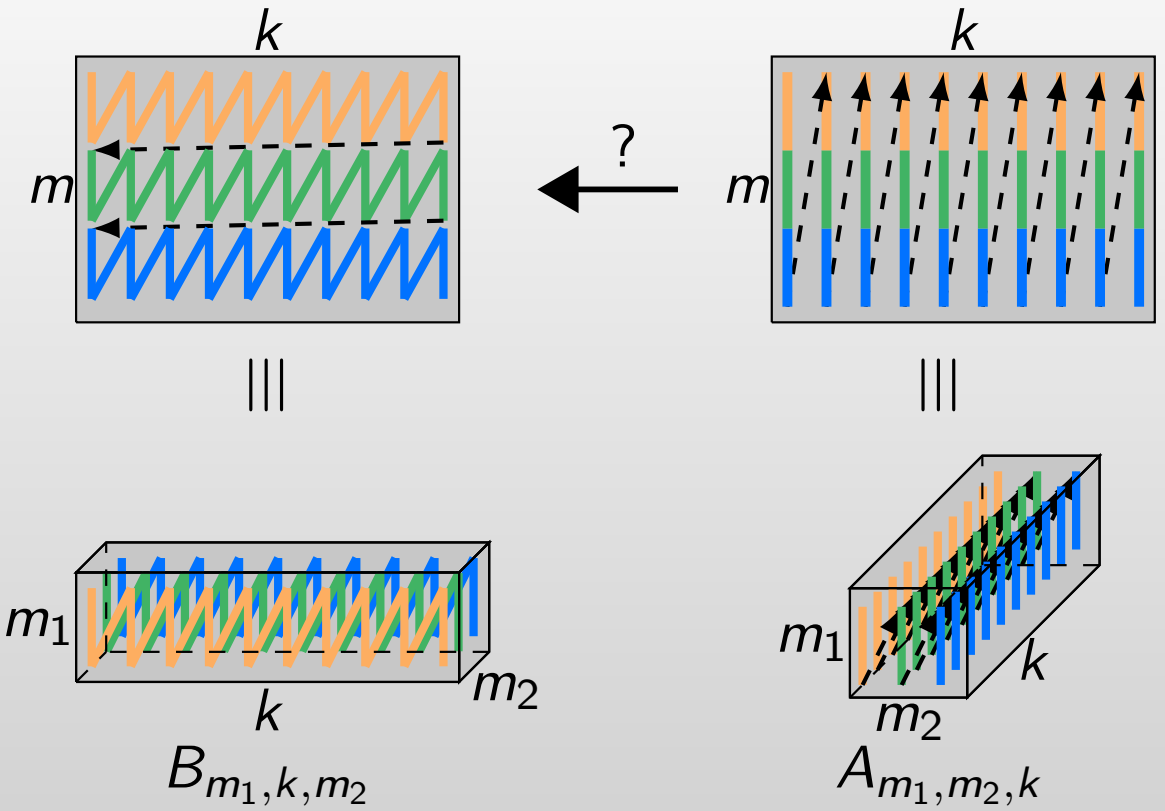
Example 2



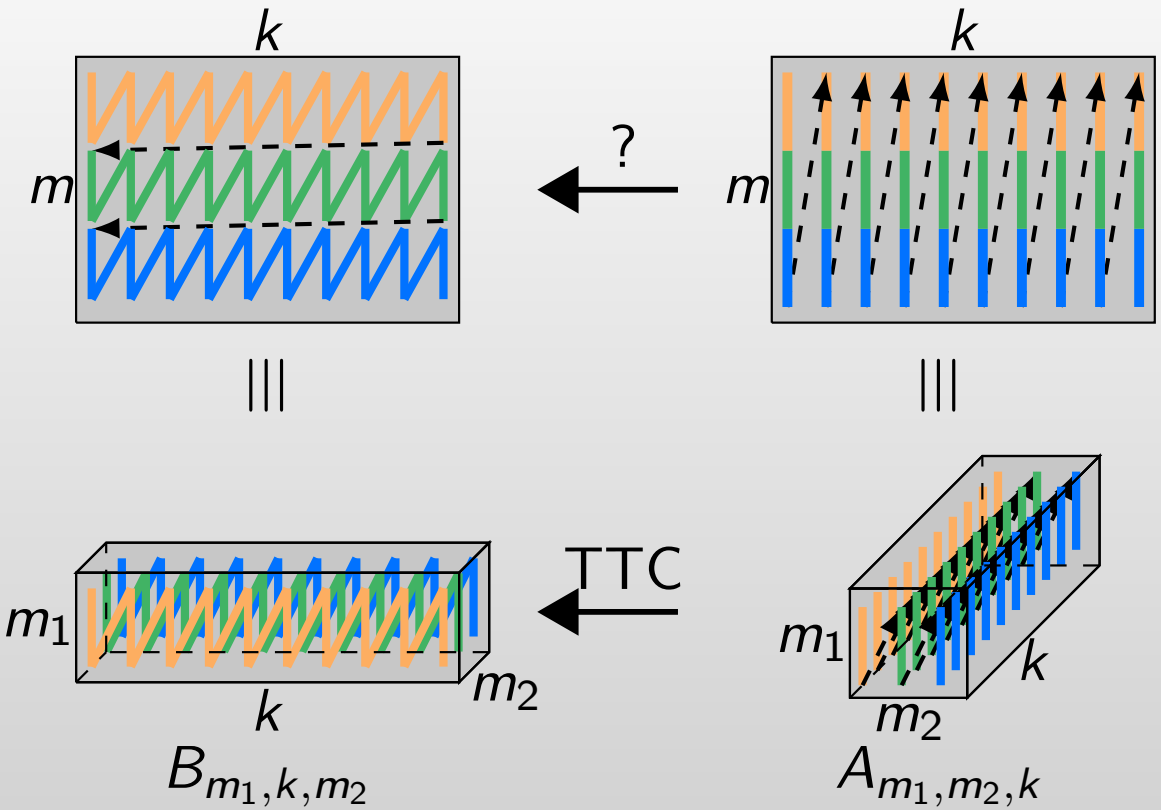
Example 2



Example 2



Example 2



- Input to TTC:

```
ttc --perm=0,2,1 --size=S_{m1},S_{m2},S_k
```

Speedup

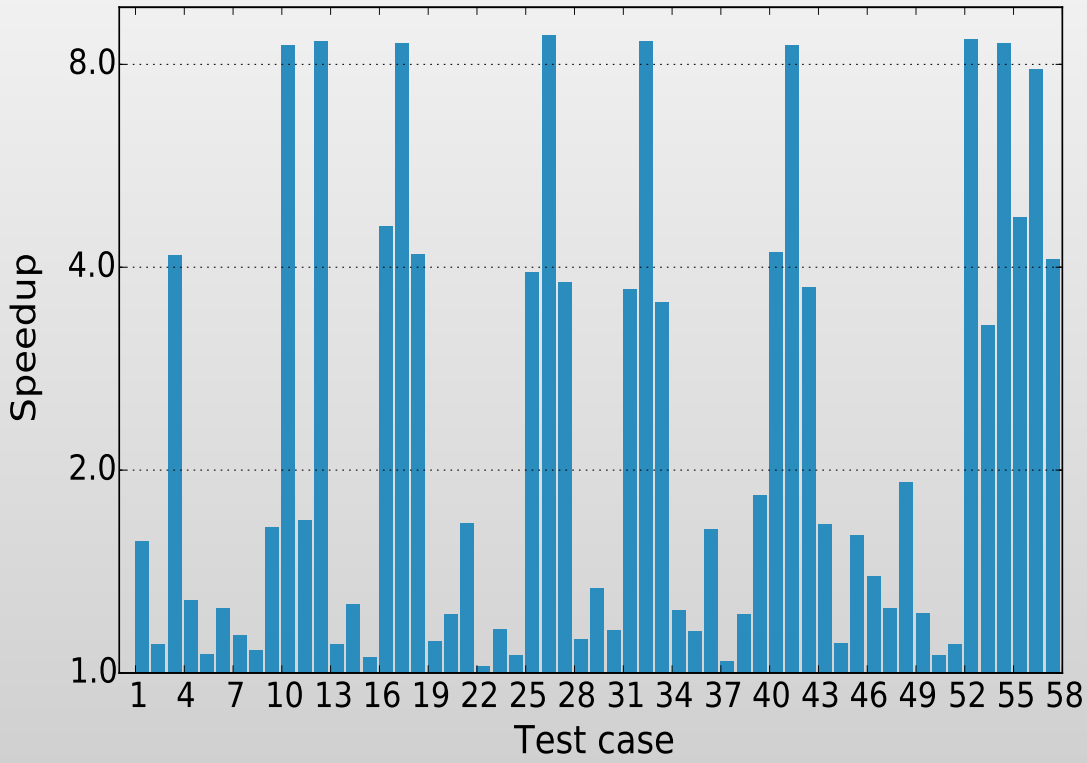


Figure: Intel Haswell architecture. 2x Intel Xeon E5-2680v3 @ 24 threads, ICPC 16.0.1.

Bandwidth

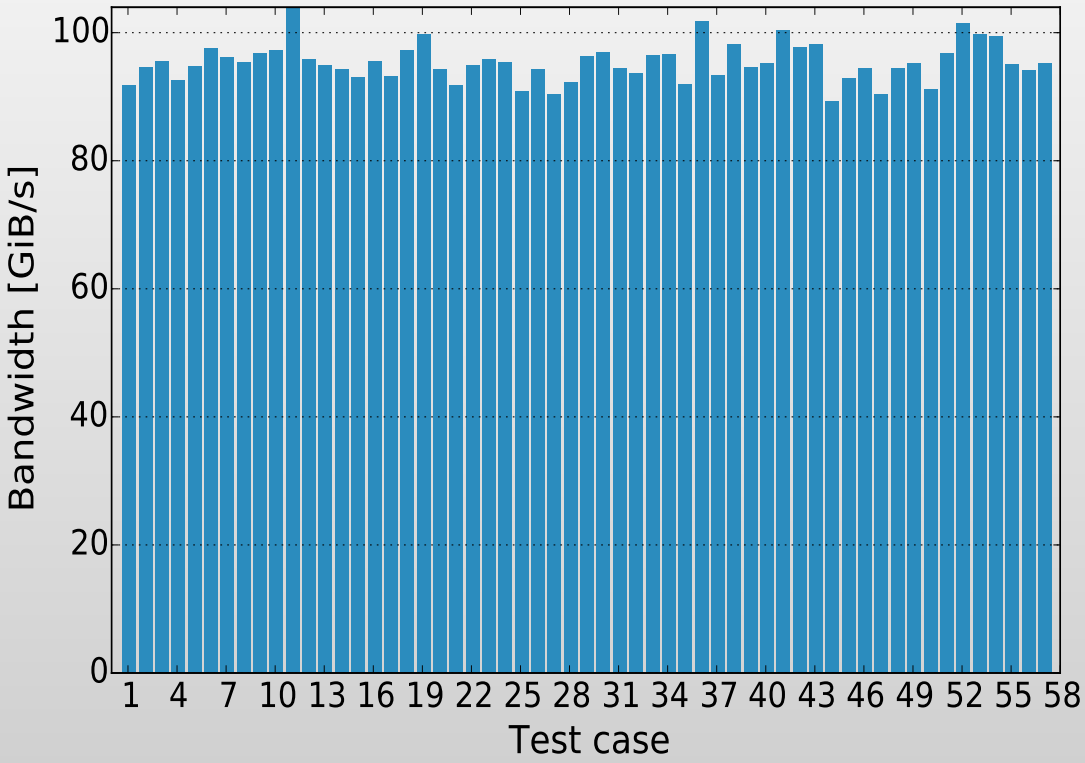


Figure: Intel Haswell architecture. 2x Intel Xeon E5-2680v3 @ 24 threads, ICPC 16.0.1.

Bandwidth

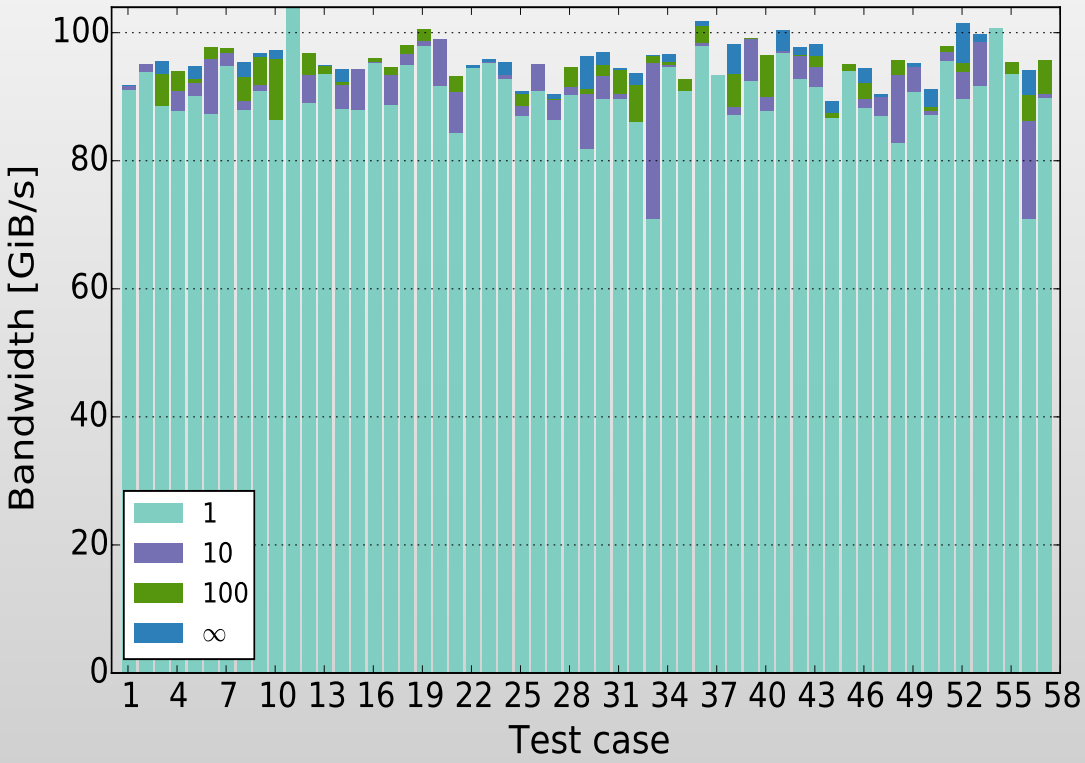
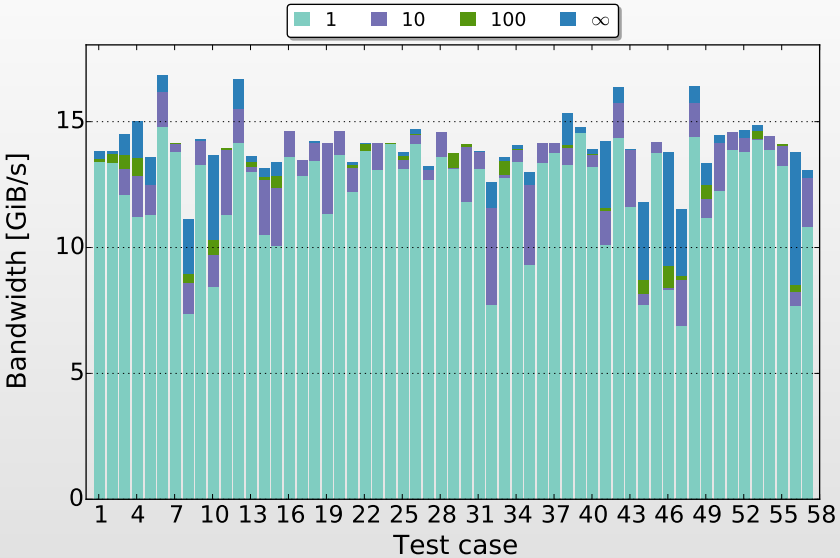
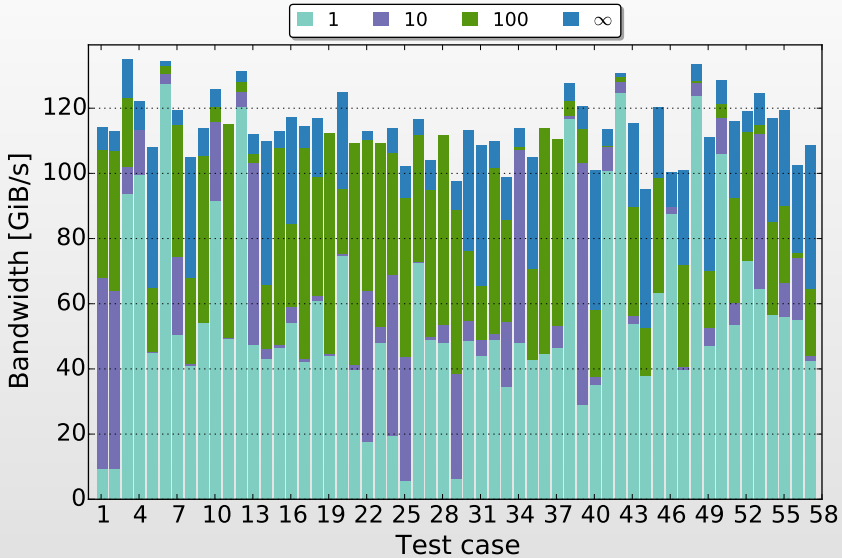


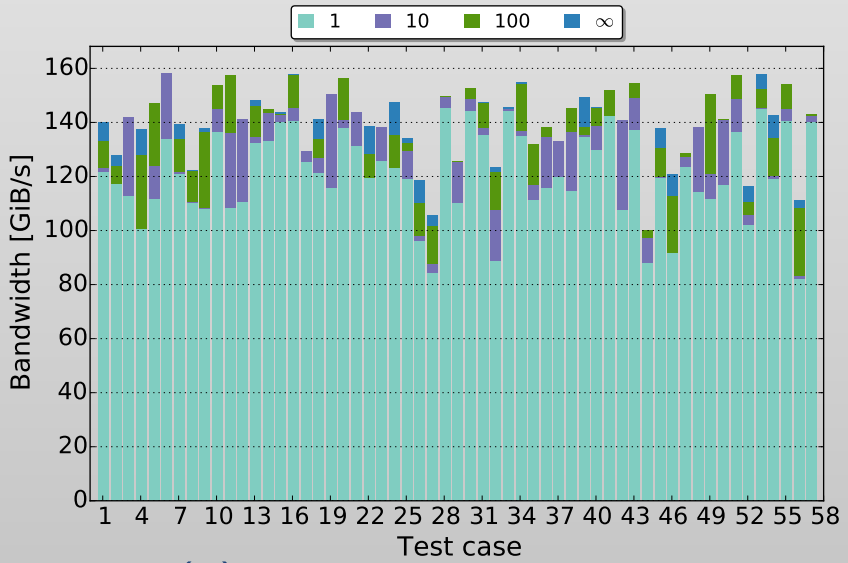
Figure: Intel Haswell architecture. 2x Intel Xeon E5-2680v3 @ 24 threads, ICPC 16.0.1.



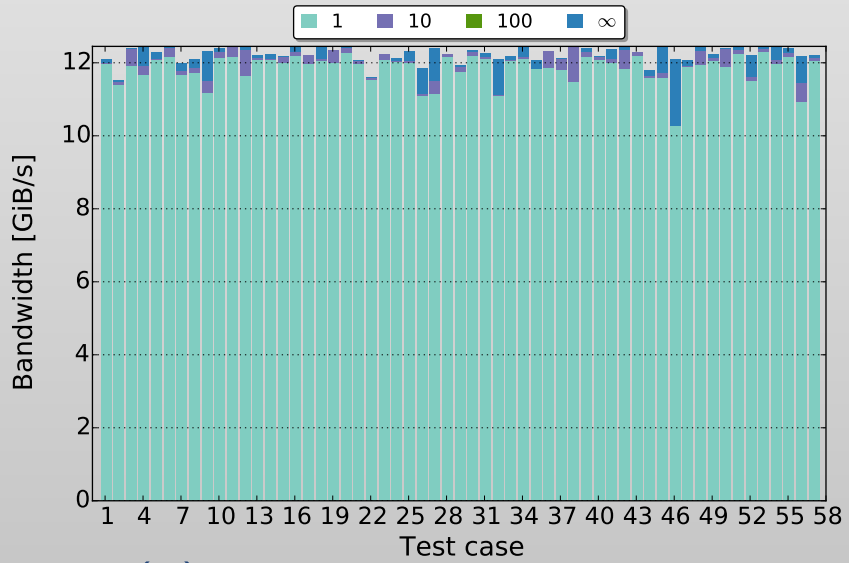
(a) AMD Steamroller, AMD A10-7850K.



(b) Intel KNC, Xeon Phi 5110p.



(c) NVIDIA Kepler, Tesla K40c.



(d) NVIDIA Maxwell, GeForce 840m .

Summary

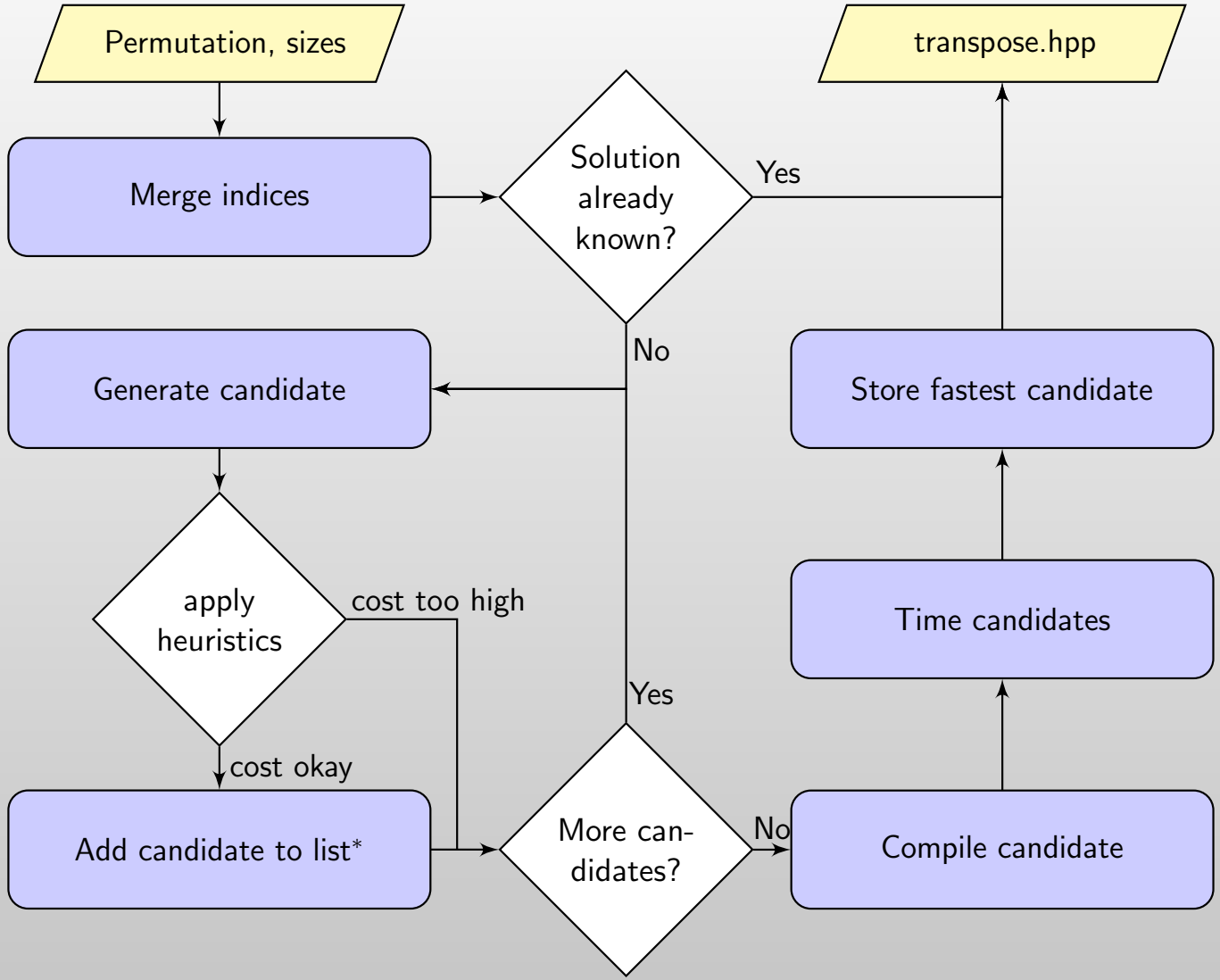
- Domain-specific compiler for multidimensional transpositions
  - Generates high-performance C++/CUDA C code
- Features
  - multiple architectures
  - all numerical data types
  - mixed precision
  - multiple leading dimensions
- Open Source
  - `https://github.com/hpac/ttc`
- Preprint
  - `http://arxiv.org/pdf/1603.02297v1`

Summary

- Domain-specific compiler for multidimensional transpositions
  - Generates high-performance C++/CUDA C code
- Features
  - multiple architectures
  - all numerical data types
  - mixed precision
  - multiple leading dimensions
- Open Source
  - `https://github.com/hpac/ttc`
- Preprint
  - `http://arxiv.org/pdf/1603.02297v1`

Thank you for your attention.

Program Flow



*capacity of the list can be influenced by the user, default capacity: 200.

Generated C++ interface

```
/**
  * B(i2,i1,i0) <- alpha * A(i0,i1,i2)
  *
  * \param[in] A Input tensor
  * \param[out] B Output tensor
  * \param[in] alpha scalar factor of A
  * \param[in] lda leading dimensions of A, can be NULL
  * \param[in] ldb leading dimensions of B, can be NULL
  */
template<int size0, int size1, int size2>
void sTranspose210_8x16x16(const float* A,
                           float* B,
                           float alpha,
                           const int *lda,
                           const int *ldb);
```

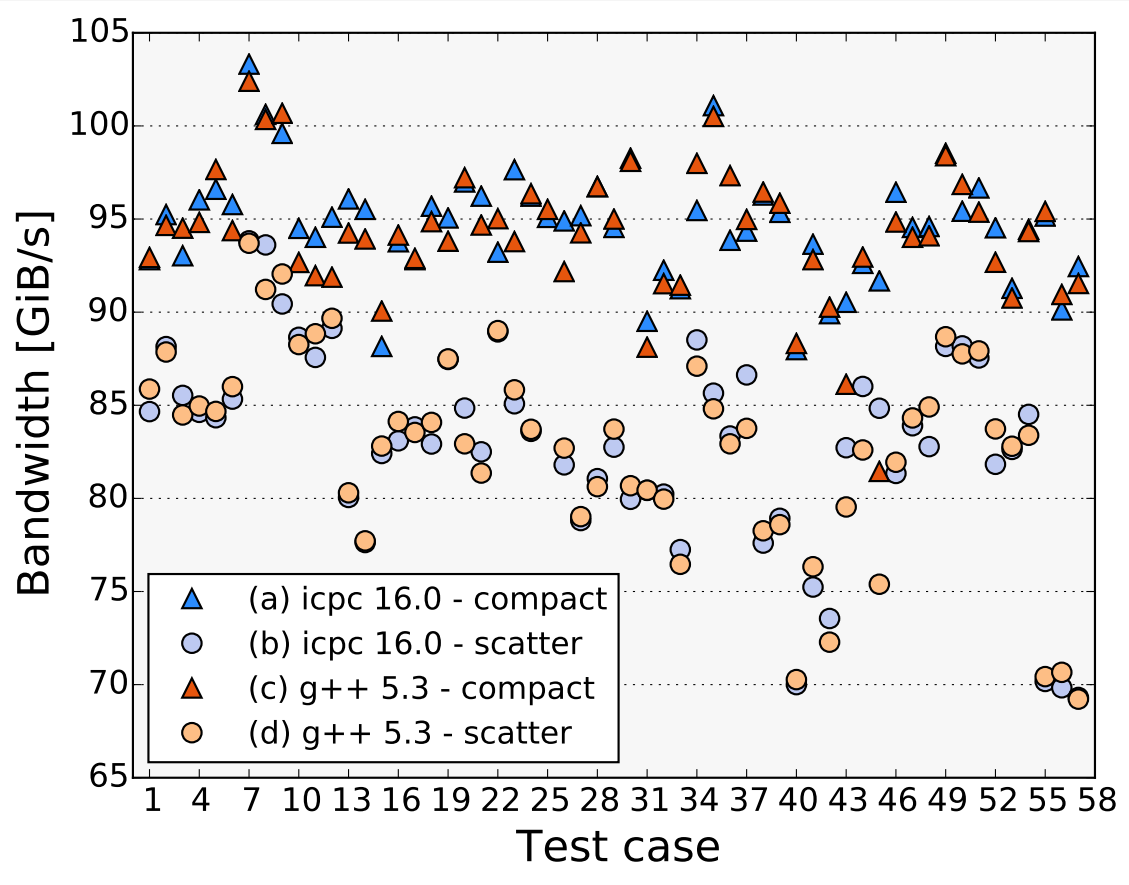


Figure: Influence of thread affinity and compiler on performance.