intel Software

PARALLELIZING MPI USING TASKS FOR HYBRID PROGRAMMING MODELS

Presenter: Surabhi Jain

Contributors: Surabhi Jain, Gengbin Zheng, Maria Garzaran, Jim Cownie, Taru Doodi, and Terry L. Wilmarth

May 25, 2018
ROME workshop (in conjunction with IPDPS 2018), Vancouver, Canada

Notices

Acknowledgment: This material is based upon work supported by the U.S. Department of Energy and Argonne National Laboratory and its Leadership Computing Facility under Award Number(s) DE-AC02-06CH11357.

Disclosure Notice: This report/presentation was prepared as an account of work sponsored by an agency and/or National Laboratory of the United States Government, in. Neither the United States Government nor any agency or National Laboratory thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency or National Laboratory thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency or National Laboratory thereof.

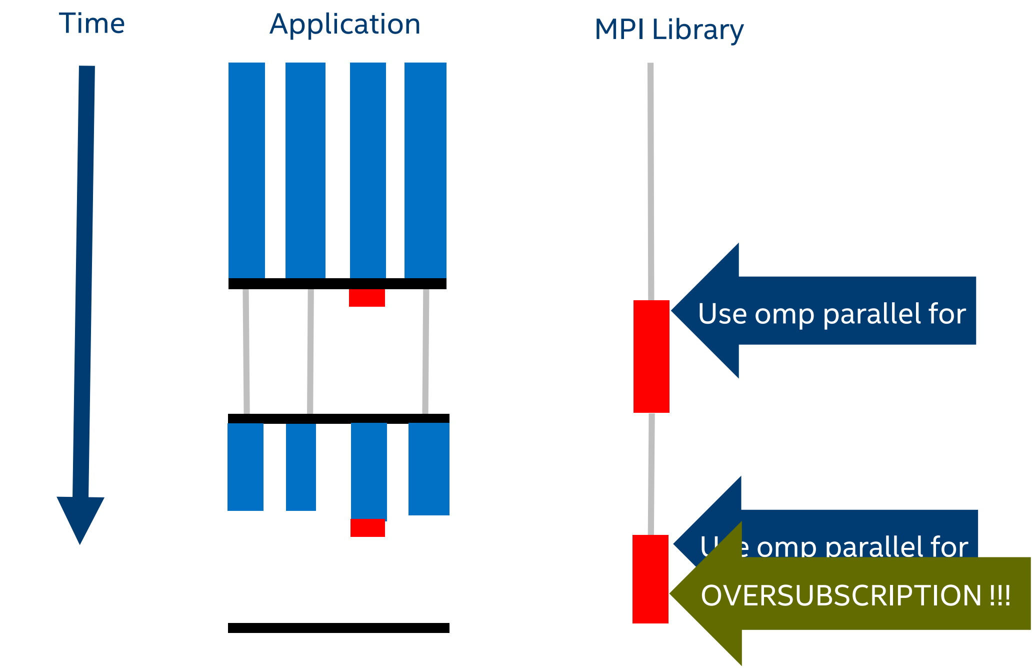
Access to this document is with the understanding that Intel is not engaged in rendering advice or other professional services. Information in this document may be changed or updated without notice by Intel.

This document contains copyright information, the terms of which must be observed and followed.

Reference herein to any specific commercial product, process or service does not constitute or imply endorsement, recommendation, or favoring by Intel or the US Government.

Intel makes no representations whatsoever about this document or the information contained herein. IN NO EVENT SHALL INTEL BE LIABLE TO ANY PARTY FOR ANY DIRECT, INDIRECT, SPECIAL OR OTHER CONSEQUENTIAL DAMAGES FOR ANY USE OF THIS DOCUMENT, INCLUDING, WITHOUT LIMITATION, ANY LOST PROFITS, BUSINESS INTERRUPTION, OR OTHERWISE, EVEN IF INTEL IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Example: Hybrid (MPI+OpenMP*) Application



- Machine supports 4 threads (including hyperthreading)
- 1 MPI rank, 4 threads per rank

* Other names and brands may be claimed as the property of others.

Motivation and Goal

Motivation:

Applications run using hybrid programming models (MPI+X)

- X=OpenMP, TBB, Pthreads
- Application threads run the computation code in parallel
- Usually only one thread calls the MPI library

Goal:

To have an MPI library that runs using multiple threads which do not compete with the application threads (avoid oversubscription)

* Other names and brands may be claimed as the property of others.

Approaches (MPI+X, X = OpenMP)

Thread Partitioning

- Partition the threads. Dedicate n threads to MPI and rest to OpenMP

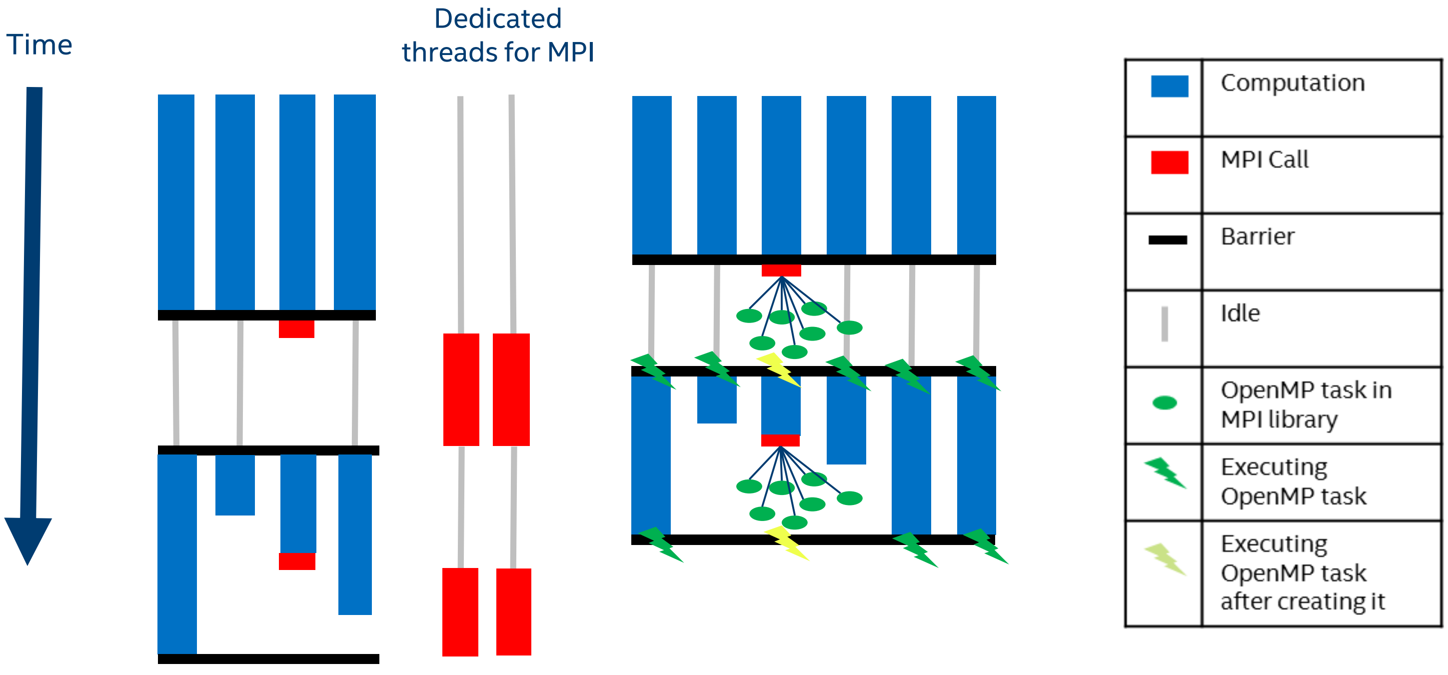
Modify OpenMP library

- OpenMP tells MPI the number of idle threads
- Spawn “number of idle threads” threads in MPI
- MT-MPI: Multithreaded MPI for many-core Environments, ICS’14 by Si et al.

Create tasks in MPI (Our approach)

- Tasks can be executed by idle application threads
- Does not spawn additional threads in MPI, no oversubscription
- No modifications required in OpenMP
- Maps well to any library which supports a tasking model e.g. OpenMP, TBB

Ways to share threads between MPI and OpenMP



Thread partitioning

Our approach (using Tasks)

Our approach orthogonal to MPI_THREAD_MULTIPLE

Do not expect much benefit with MPI_THREAD_MULTIPLE

- Parallelism comes from several threads concurrently calling MPI
- Fewer threads are idle to execute MPI tasks

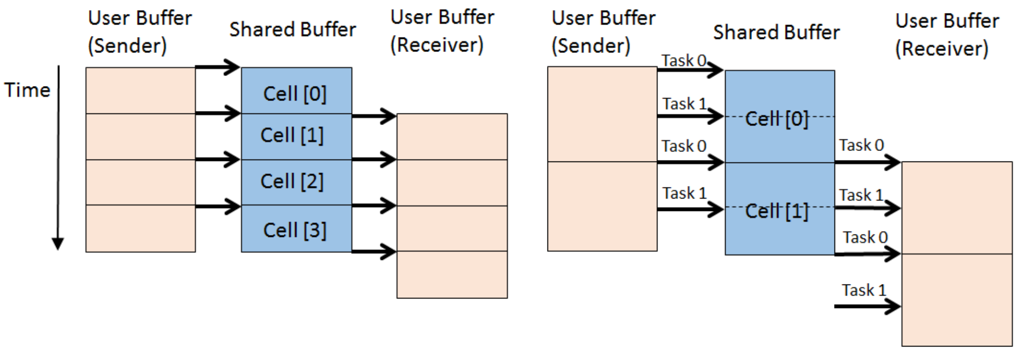
Creating OpenMP tasks in MPI

```
if(omp_in_parallel()) {
     //Create tasks for what MPI wants to do in parallel
     //which will run on idle pre-existing OpenMP threads
     #pragma omp taskwait
     /* All tasks we created have completed when we get here */
} else {
     /* No pre-existing parallelism so create some */
     #pragma omp parallel
     {
          #pragma omp single nowait
          {
               //Create tasks for what MPI wants to do in parallel
          }
     }
     /* All tasks we created have completed when we get here */
}
```

Where to create tasks inside MPI?

- Shared memory communication
- Packing/unpacking of non-contiguous data

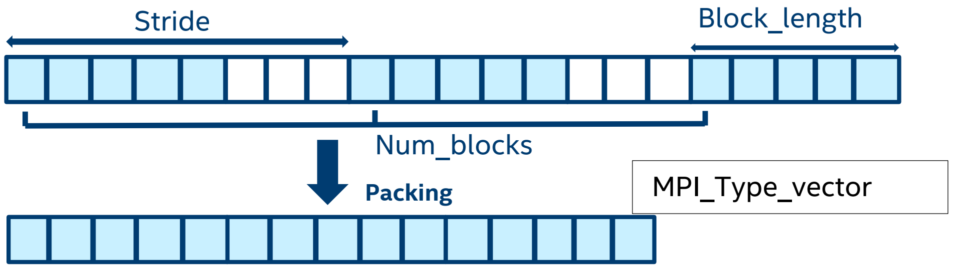
Shared Memory Communication

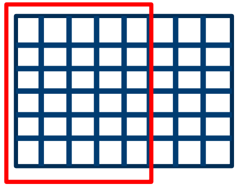


(a) Sequential Data Copying (b) Parallel Data Copying

- Sender and receiver rank on the same node, use intermediate shared buffer for large messages
- Pipelined Double Copy approach- sender can copy to next cell(s), while receiver is copying from previous cell(s)
- Find balance between pipeline parallelism and task based parallelism

Pack/Unpack non-contiguous data





Stride = 8
Block_length = 5
Num_blocks = 6

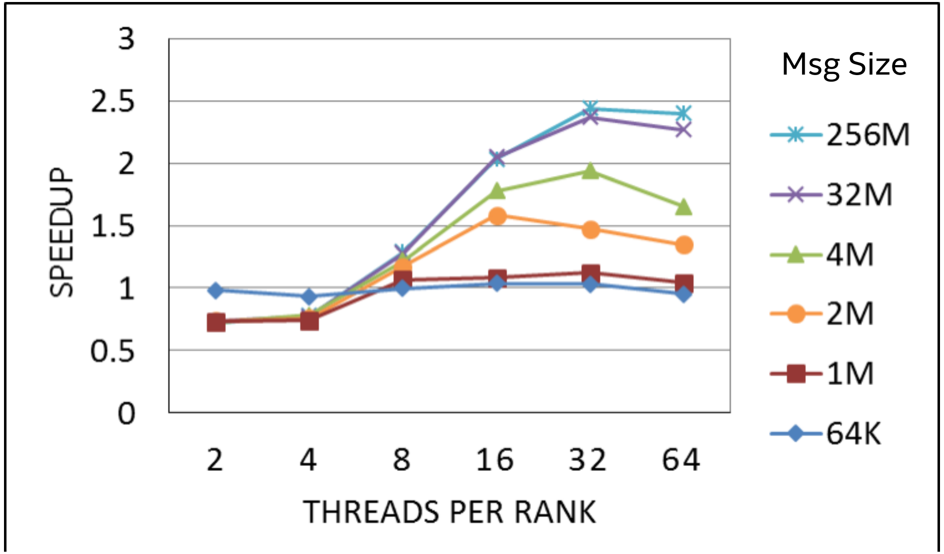
Derived types

- Constructed from existing types (basic and derived). E.g. MPI_Type_indexed, MPI_Type_vector, MPI_Type_struct
- Each task can pack/unpack one or more blocks

Experimental Setup

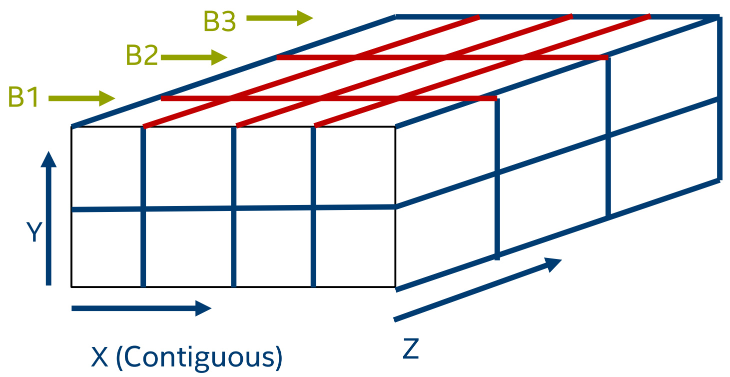
- Intel® Xeon Phi Processor 7210 (1.3 GHz, 64 cores, 4 threads/core) (Knights Landing)
- 32KB L1 data and instruction cache, 1MB L2 cache
- 96GB DDR, 16GB MCDRAM
- KNL memory mode – Flat, cluster mode – Quadrant, No SNC
- Data placed on MCDRAM (using numactl –m 1)
- Compiler from Intel® Parallel Studio XE Composer Edition for C++ (version 2016.0.109)
- MPICH v3.2b4-98-g4551de1 as the baseline

Parallel memcpy – OSU latency benchmarks



	MPICH (Original)	MPICH (Modified) [Baseline]	MPICH (Modified) with OpenMP tasks
Task Size			32KB
Cell Size	32KB	256KB	#threads * Task Size
Total Size	256KB	4MB	4MB
Num Cells	8	16	Total Size/Cell Size

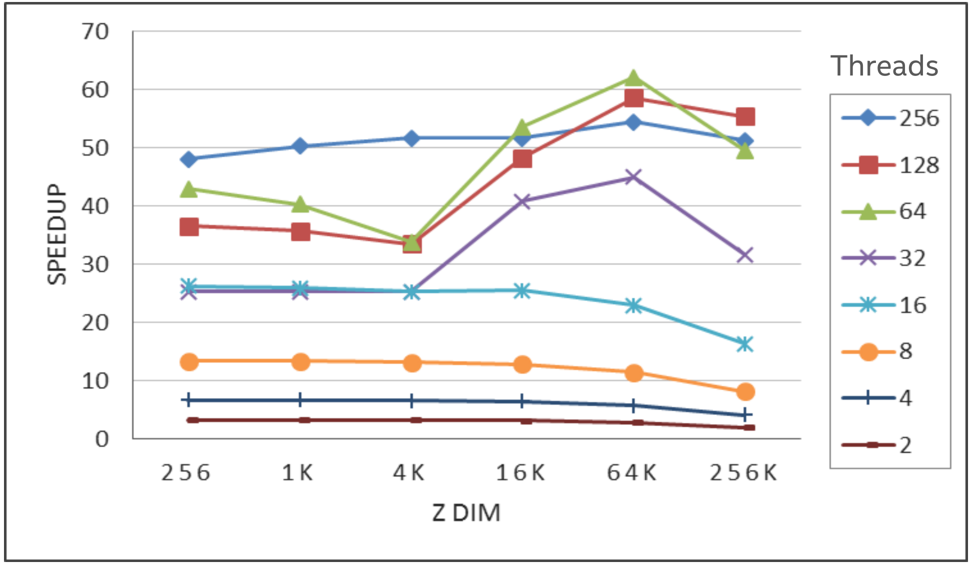
Top Pack Benchmark (from MT-MPI* paper)



- Pack the top surface (XZ plane) of a 3D matrix of doubles
- Matrix volume fixed to 1 GB and Y dimension to 2
- Represented using MPI_Type_vector

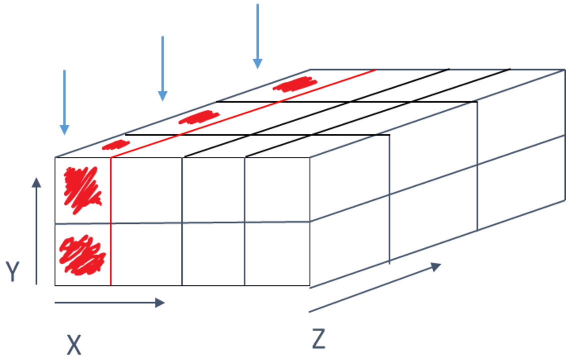
*Si et al . MT-MPI: Multithreaded MPI for many-core Environments . ICS'14

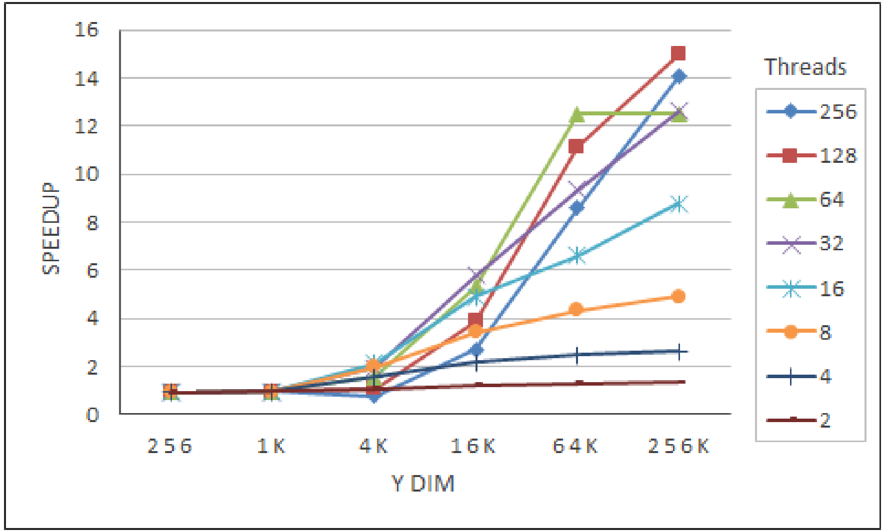
Results: Top Pack, MPI_Type_vector



- Packing called from a serial region in application
- 1 MPI rank
- Blk_size(X) decreases as num_blks(Z) increases

Left Pack, Nested MPI_Type_vector





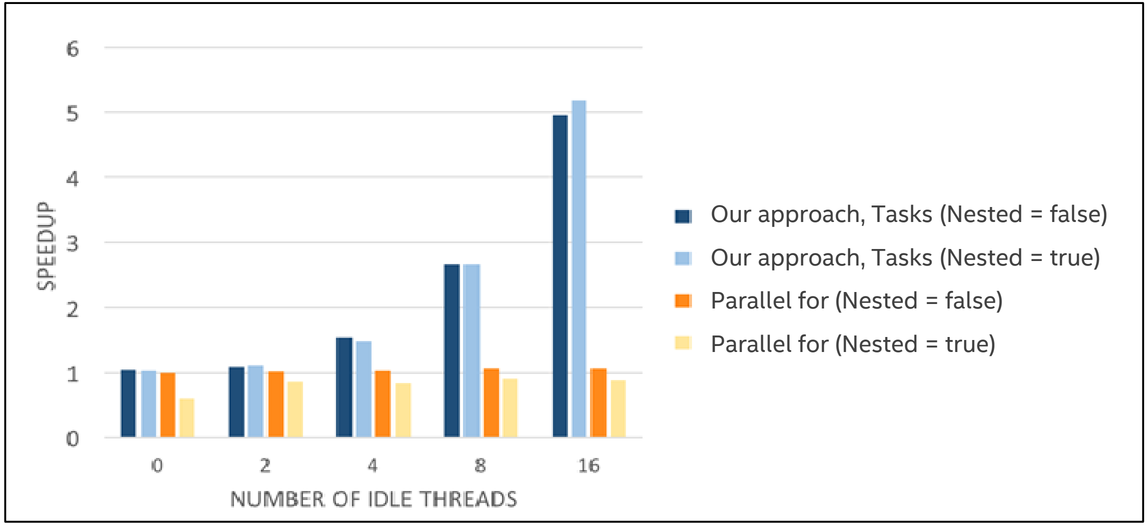
- Tasks at leaf level – Parallelize over Y dimension (vector datatype)
- Tasks at higher level – Parallelize over Z dimension (vector of vectors datatype)

MPI_Pack() called from a parallel region

```
#pragma omp parallel
{
      thread_id = omp_get_thread_num();
      if (thread_id < 4)
            call MPI_Pack();
      else if (thread_id < 4 + num_idle_threads)
            do_nothing
      else
            do_computation
}
```

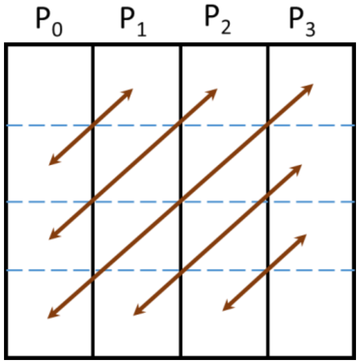
- Uses MPI_THREAD_MULTIPLE mode
- Threads are divided into 3 groups -
  - Threads calling MPI_Pack(). Create OpenMP task in MPICH
  - Idle threads. Wait at the barrier and execute tasks
  - Compute threads. Can help in executing tasks when reach the barrier

Results: MPI_Pack() called from a parallel region



- Total threads = 256, Packing threads = 4
- Significant benefits when threads are idle
- No penalty when no idle threads

Transpose from Parallel Research Kernels*

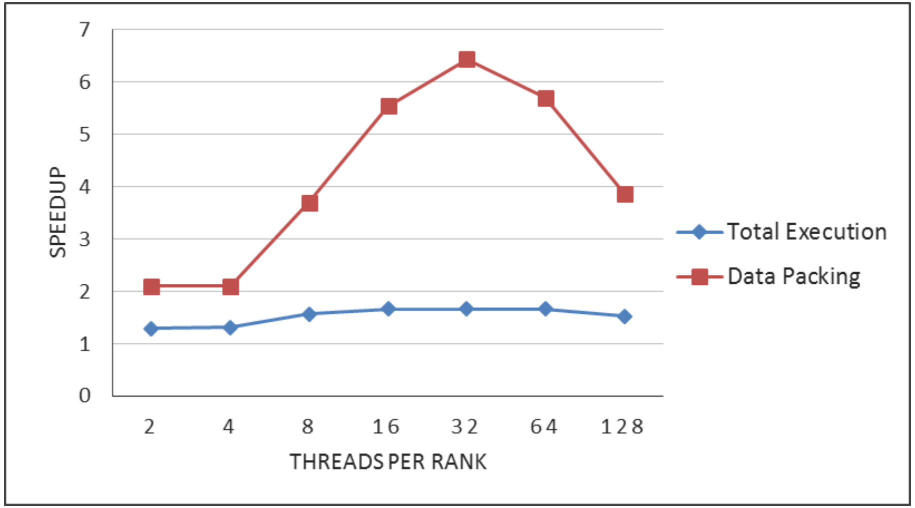


Steps to do the transpose

1. On each rank, local transpose. No data communicated
2. All-to-all communication
   1. Use nested MPI_Type_vector datatype
   2. Parallelize the pack/unpack

* https://github.com/ParRes/Kernels

Results : Transpose Kernel



- Matrix Order 8K doubles
- 2 MPI ranks on 1 KNL node
- Leaf vector – num_blks= 4K, blk_len=1

Conclusion

- Our task-based approach-
  - Opportunistic
  - No creation of additional threads, so no oversubscription
  - No modification made in the OpenMP library
- Speedup up to 62X on Top Pack, when all the threads are idle
- Speedup up to 6.5X in data packing and up to 1.5X reduction in overall execution time of transpose kernel
- Code is publicly available (link in the paper)

Questions ?

surabhi.jain@intel.com

https://www.linkedin.com/in/surabhi-jain-9066821a/

Legal Disclaimer & Optimization Notice

intel®

Software