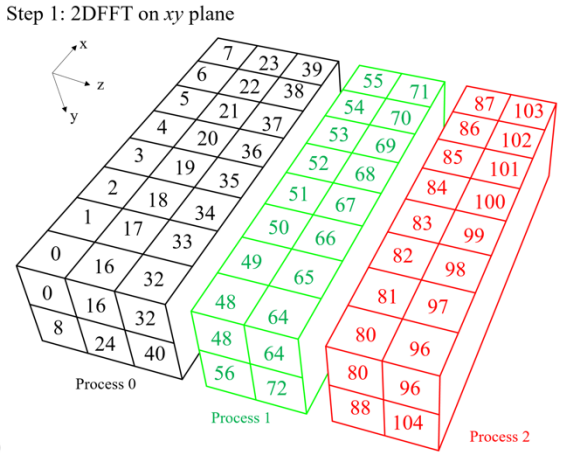
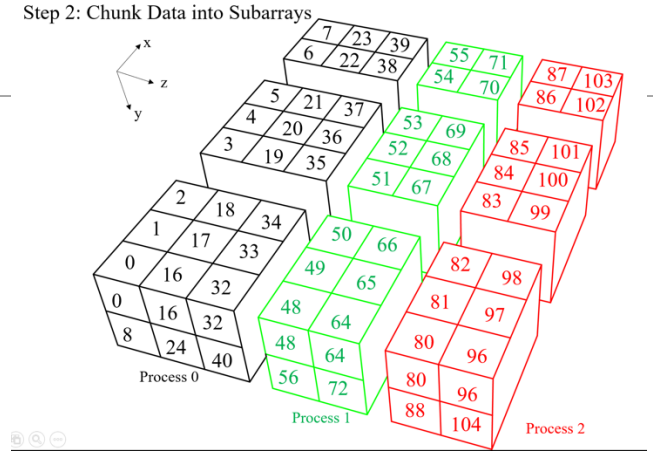
Improved MPI Collectives for 3D-FFT

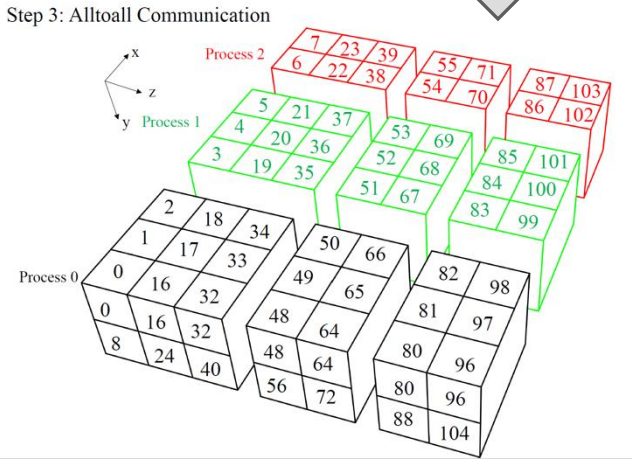
PRESENTER: YUANG YAN

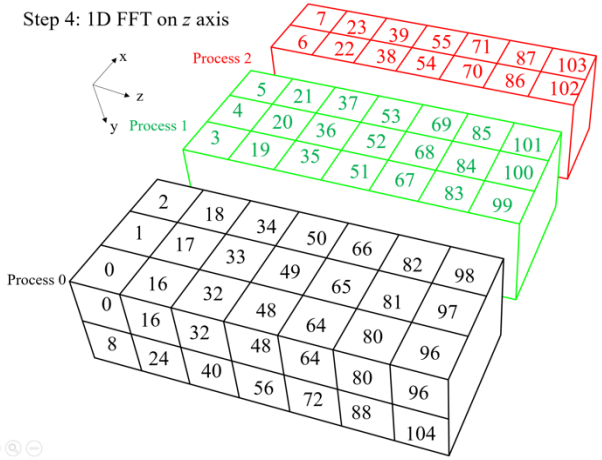
Queen's UNIVERSITY

Parallel multidimensional FFT(Slab Decomposition)





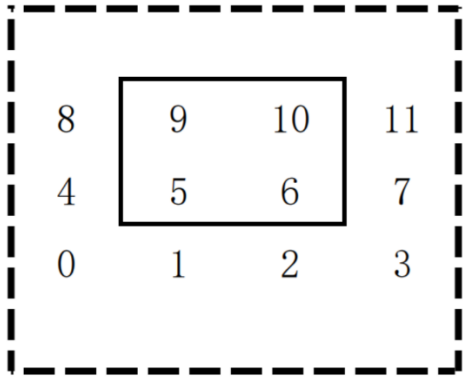




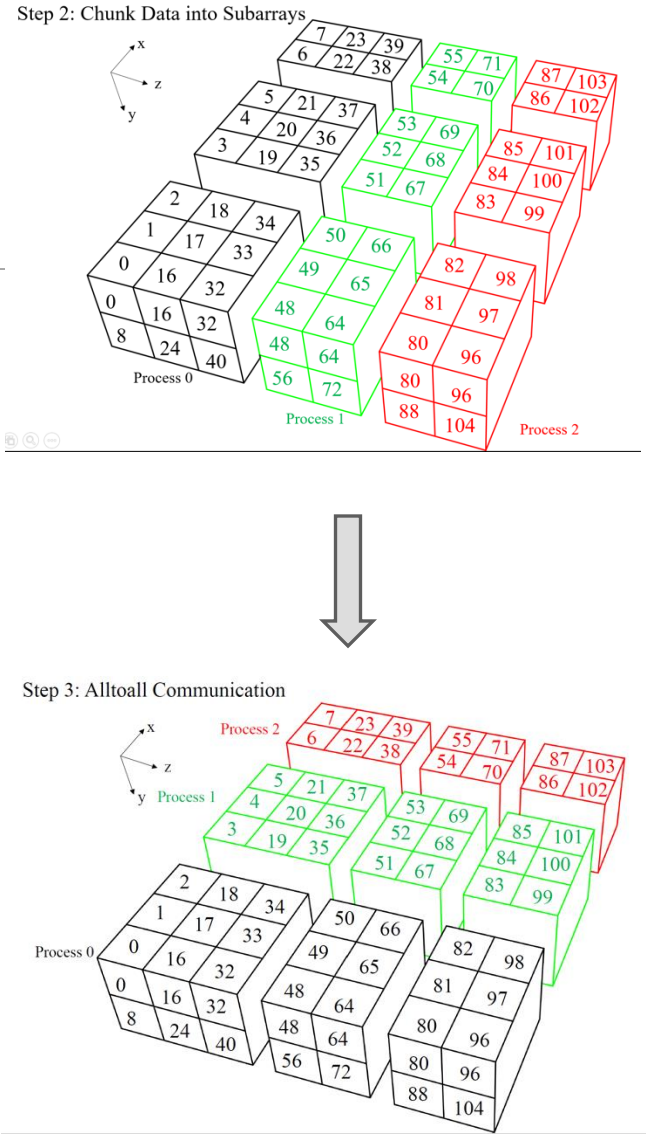
Time breakdown

- Individual serial FFT for the first phase and the last phase
- Data packing and communication time
- Data rearrangement time(if necessary)
- Data layout has impact on both serial FFT time and communication time

Use MPI subarray datatype for data packing




```
int MPI_Type_create_subarray(
int ndims,
const int array_of_sizes[],
const int array_of_subsizes[],
const int array_of_starts[],
int order,
MPI_Datatype oldtype,
MPI_Datatype *newtype)
```



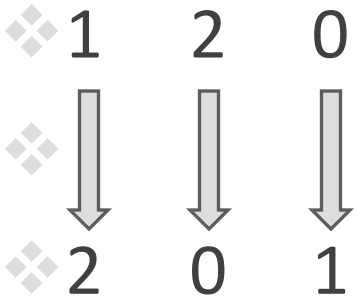
Dalcin, L., Mortensen, M., Keyes, D.E.: Fast parallel multidimensional fft us- ing advanced mpi. Journal of Parallel and Distributed Computing 128, 137–150 (2019).

Limit of subarray datatype

- Only two storage orders is supported(MPI_ORDER_C and MPI_ORDER_FORTRAN)
- Which means a strided FFT or a data rearrangement
- MPI does require exact match for datatype between send and recv
- Thinking of supporting subarray datatype with more flexible storage orders, implementing in-flight data arrangement
- For a 3 D array, there are 6 possible orders(x-y-z, x-z-y, y-z-x, y-x-z, z-x-y, z-y-x), represented by tuples(0,1,2), (0,2,1), (1,0,2), (1,2,0), (2,1,0), (2,0,1)

In-flight rearrangement

- Consider send type ordered (1, 2, 0) and recv type ordered (2, 0, 1)
- Sending side: The elements in axis 1(y axis) are packed first, then axis 2(z axis) and axis 0 (x axis)
- Recv side: The data is first unpacked to axis 2 (z axis), the axis 0(x axis) then axis 1(y axis)

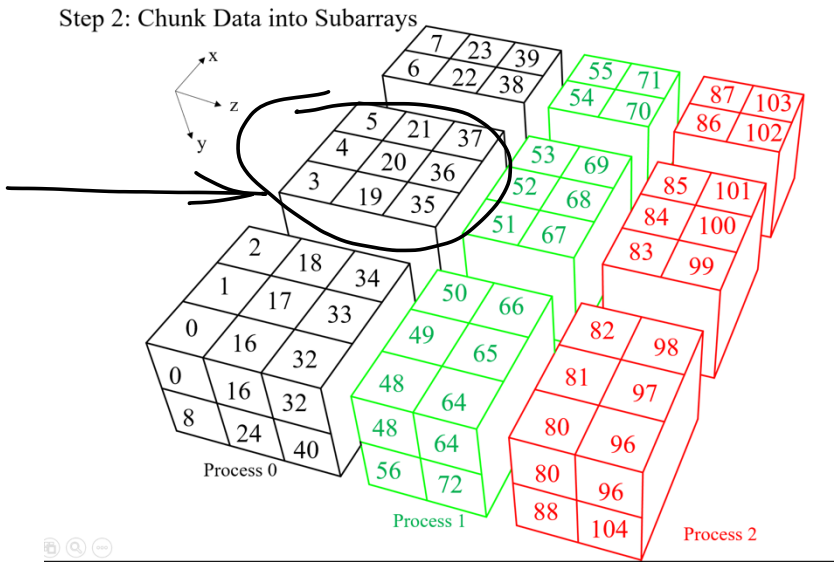


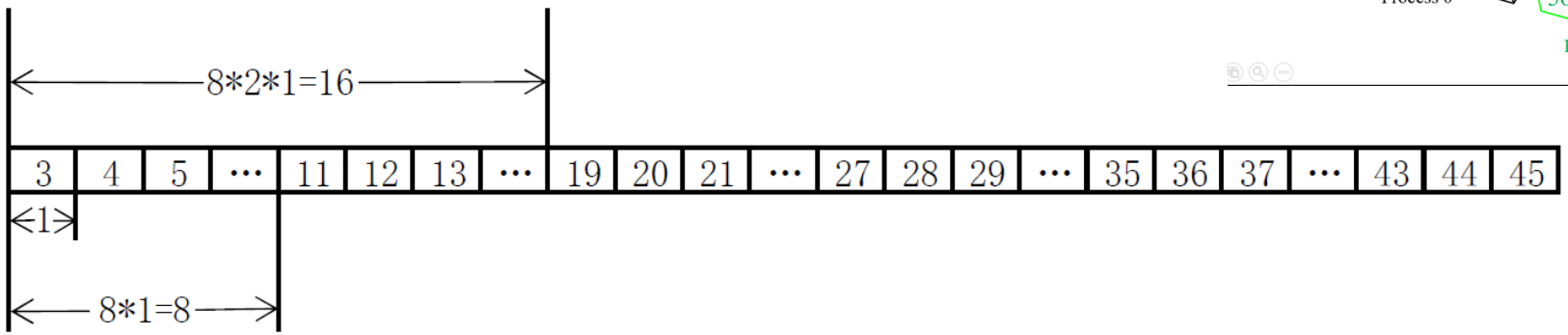
Which is xyz-yzx transpose

Creating a flexible subarray datatype

- Nest MPI_Type_create_hvector()

```
int MPI_Type_create_hvector(
        int count,
        int blocklength,
        MPI_Aint stride,
        MPI_Datatype oldtype,
        MPI_Datatype *newtype
)
```

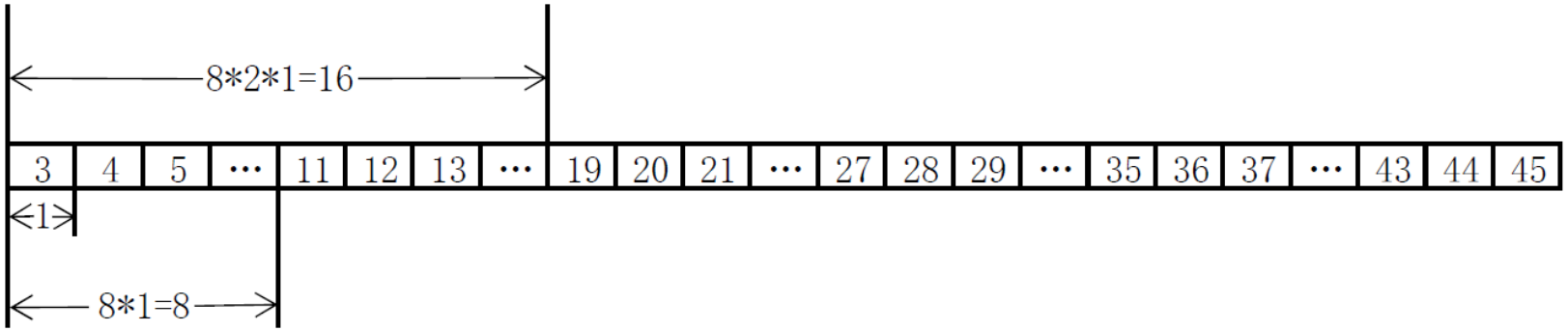


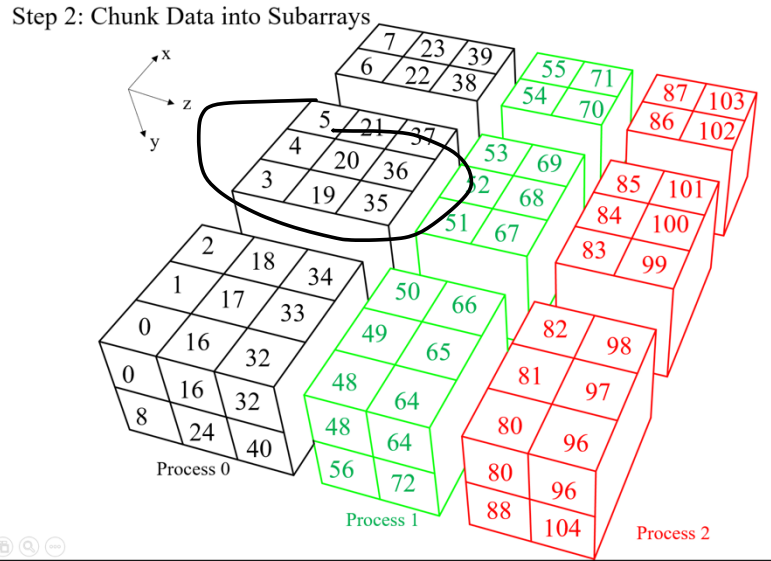


Creating a flexible subarray datatype

count	blocklen	stride	oldtype	newtype
3	1	1*sizeof(complex)	complex	DDT1
2	1	8*1*sizeof(complex)	DDT1	DDT2
3	1	8*2*1*sizeof(complex)	DDT2	DDT3

count	blocklen	stride	oldtype	newtype
3	1	8*2*1*sizeof(complex)	complex	DDT1’
2	1	8*1*sizeof(complex)	DDT1’	DDT2’
3	1	1*sizeof(complex)	DDT2’	DDT3’





Creating a flexible subarray datatype

- Step1: Calculating the stride

```
for(int i = 0; i < ndims; ++i) {
    counts[i] = subsizes[i];
    if(i == 0) {
        strides[i] = primitiveTsize;
    }
    else {
        strides[i] = strides[i-1] * sizes[i-1];
    }
}
```

Creating a flexible subarray datatype

- Step2: Create the block with shuffled order

```
for(int i = 0; i < ndims; ++i) {
int axis = order[i];
if(i == 0) {
        MPI_Type_create_hvector(counts[axis], blocklength, strides[axis], primitiveT, nestedT+i);
        MPI_Type_commit(nestedT+i);
}
else {
        MPI_Type_create_hvector(counts[axis], blocklength, strides[axis], nestedT[i-1], nestedT+i);
        MPI_Type_commit(nestedT+i);
        MPI_Type_free(nestedT+i-1);
        }
}
```

Creating a flexible subarray datatype

- Step3: append the offset

```
long offset = 0;
for(int i = ndims - 1; i >= 0; i --) {
        // int axis = order[i];
        int axis = i;
        if(i < ndims - 1) {
                offset *= sizes[axis];
        }
        offset += starts[axis];
}
```

Experiment setup

- System: Narval of Digital research alliance of Canada
- Infiniband Mellanox HDR network
- 2 AMD Rome 7532 CPUs, each with 32 cores
- 32KB L1 data cache, 512KB L2 cache, 256MB L3 cache
- Rocky Linux 8.9 and OpenMPI 4.1.5

Experiment setup

- For serial FFTs, use FFTW library with flag FFTW_MEASURE
- Balance between setup time and execution time
- For a 256*256*256 array

	FFTW_ESTIMATE	FFTW_MEASURE	FFTW_PATIENT
Computation time(s)	2.331	0.176	0.170
Set up time(s)	0.03	2.735	239

Minimizing Communication in the Multidimensional FFT, T. Koopman and R. Bisseling, SIAM Journal on Scientific computing, Vol,45, Iss.6(2023)

Performance Evaluation of subarray_x

- 12 valid send-receive order pairs for the matrix transpose
- Use a 2048*2048*2048 complex array
- Some performance degradation compared to original subarray datatype
- Dependent on certain system configuration

Id	Send/Recv Order	Cores	Time(s)
0	$(0,1,\mathbf{2}) \rightarrow (1,2,\mathbf{0})$	32	7.47
1	$(0,1,\mathbf{2}) \rightarrow (2,1,\mathbf{0})$	32	11.72
2	$(0,\mathbf{2},1) \rightarrow (1,\mathbf{0},2)$	32	**4.40**
3	$(0,\mathbf{2},1) \rightarrow (2,\mathbf{0},1)$	32	11.67
4	$(1,0,\mathbf{2}) \rightarrow (1,2,\mathbf{0})$	32	11.60
5	$(1,0,\mathbf{2}) \rightarrow (2,1,\mathbf{0})$	32	22.23
6	$(1,\mathbf{2},0) \rightarrow (1,\mathbf{0},2)$	32	11.57
7	$(1,\mathbf{2},0) \rightarrow (2,\mathbf{0},1)$	32	21.39
8	$(\mathbf{2},1,0) \rightarrow (\mathbf{0},1,2)$	32	10.48
9	$(\mathbf{2},1,0) \rightarrow (\mathbf{0},2,1)$	32	10.50
10	$(\mathbf{2},0,1) \rightarrow (\mathbf{0},1,2)$	32	9.62
11	$(\mathbf{2},0,1) \rightarrow (\mathbf{0},2,1)$	32	9.66

(a) Communication time using different sending/receiving storage orders

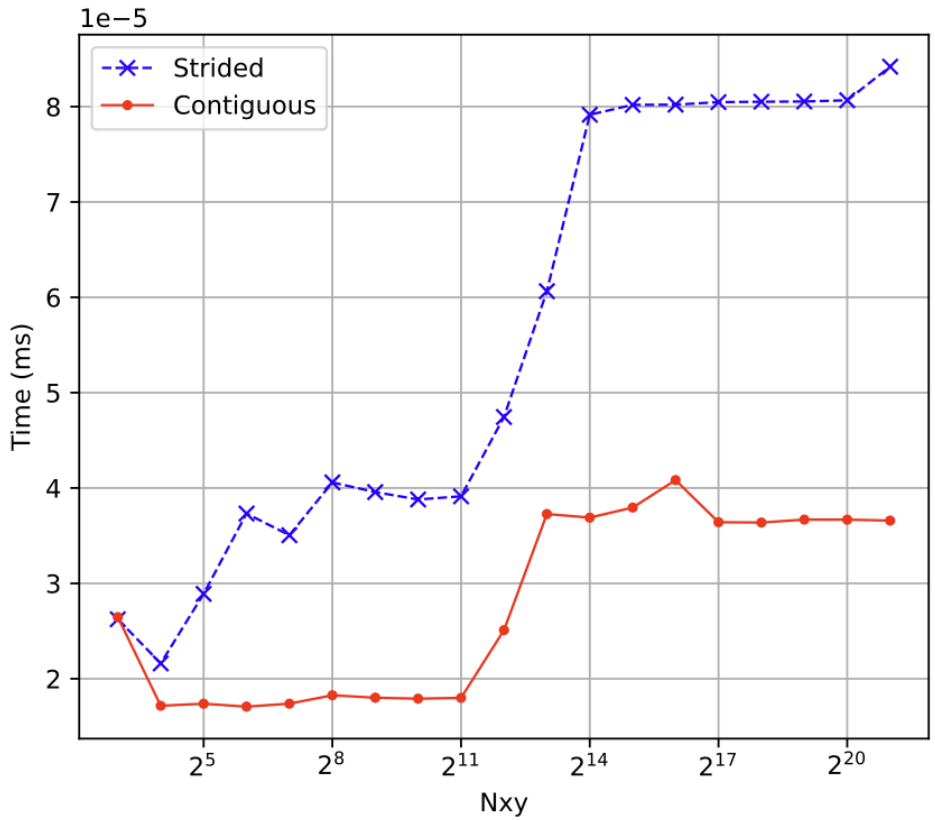
Cores	Size/Core	MPI default subarray(s)	Ours(s)	Increase
32	128GB	3.79	4.40	16.09%
64	64GB	1.76	1.93	9.66%
128	32GB	0.87	1.08	24.14%
256	16GB	0.65	0.68	4.62%
512	8GB	0.39	0.52	33.33%

(b) Communication performance comparison between MPI default subarray (MPI_ORDER_C) and our sending/receiving order pair $(0,2,1) \rightarrow (1,0,2)$ with different number of cores involved.

Table 1. Global Redistribution time using unconventional storage order pairs and comparison against MPI default subarray under a $2048 \times 2048 \times 2048$ complex 3D array on Narval.

Characterization of strided FFT penalty

- Use a single 1D dimensional FFT, L=128
- Compare between contiguous and strided FFT of z axis
- Contiguous: Stride=1
- Strided: Stride=N_xy ranges from 2^3 to 2^21

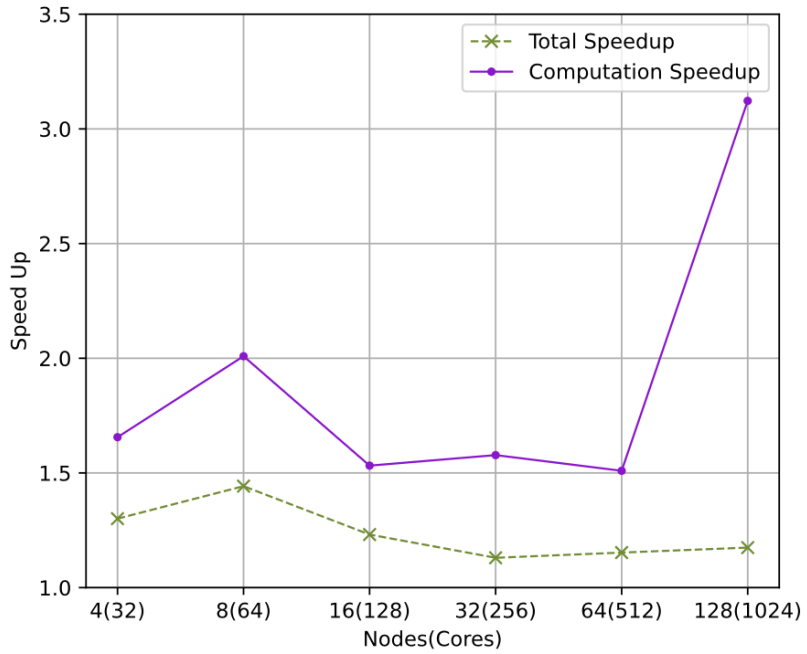


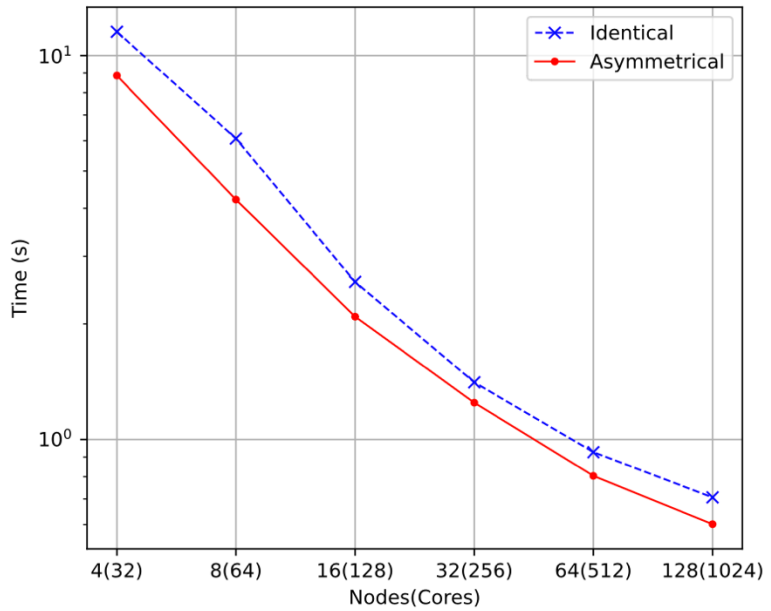
Performace evaluation of subarray_x in 3D FFT

- Strong scaling
- Use a 2048*2048*2048 complex array, cores ranges from 32 to 1024
- Due to the memory limit, not all cores are utilized in the experiment
- Computation time and communication time are recorded respectively

Performace evaluation of subarray_x in 3D FFT

- Strong scaling, 2048^3





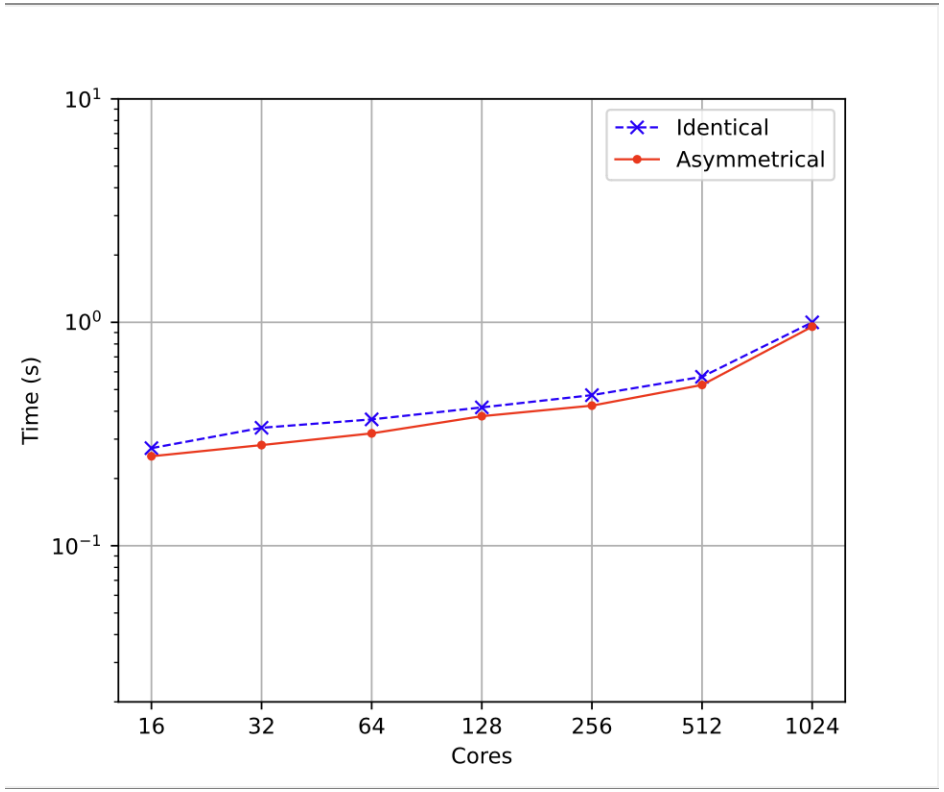
Performace evaluation of subarray_x in 3D FFT

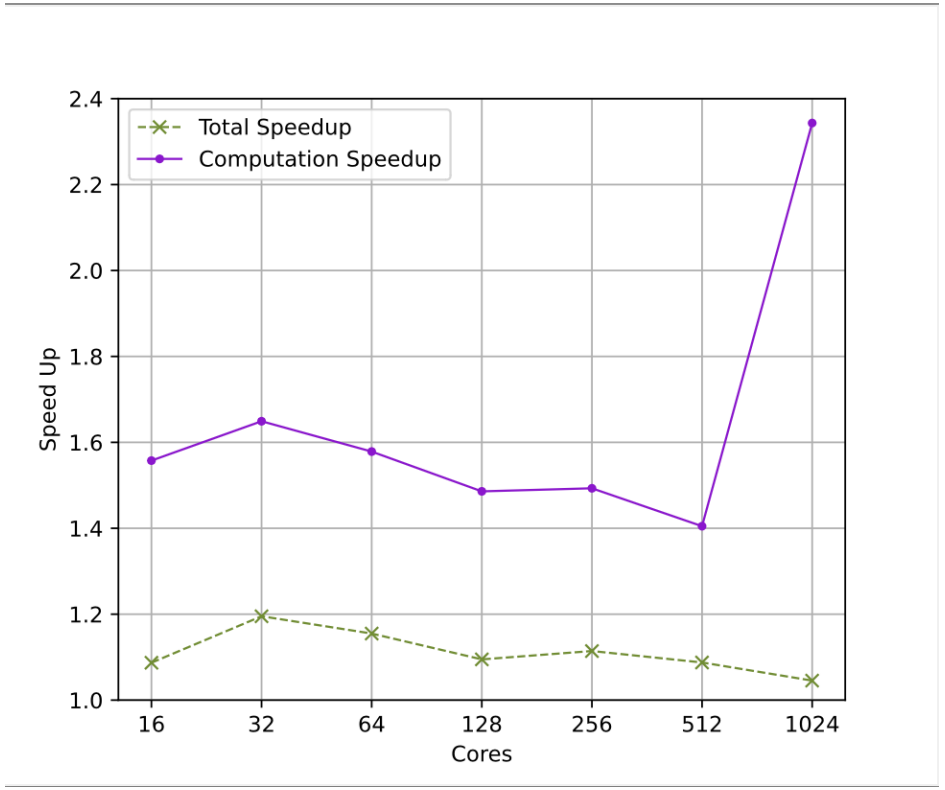
- Weak scaling

- Due to the memory limit, not all cores are utilized in the experiment
- Computation time and communication time are recorded respectively

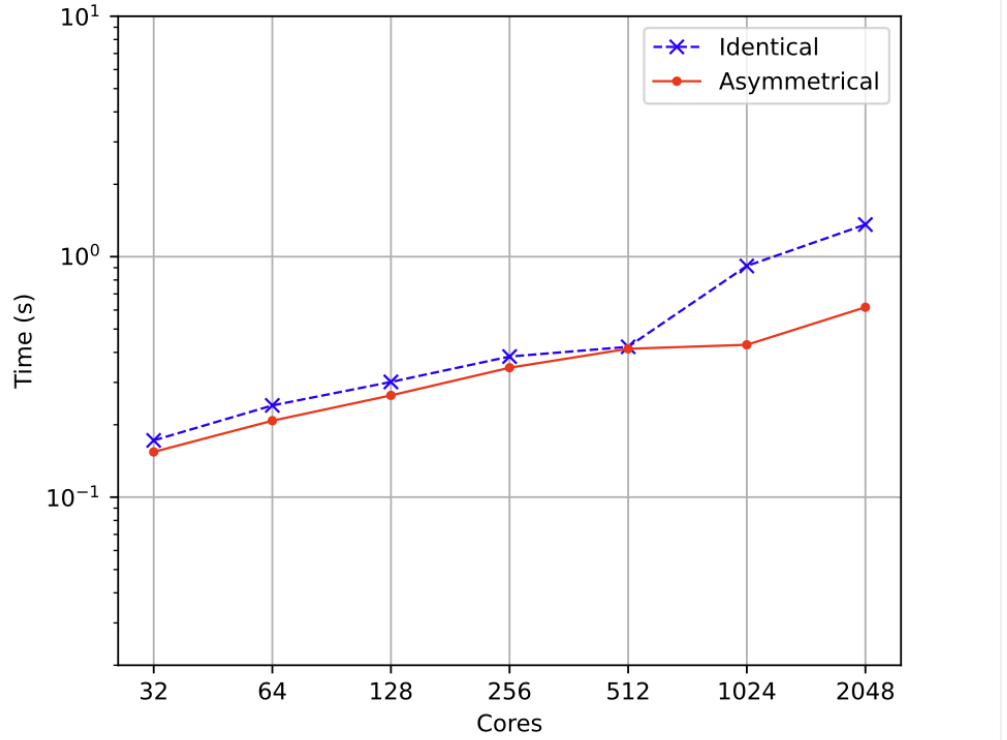
Performace evaluation of subarray_x in 3D FFT

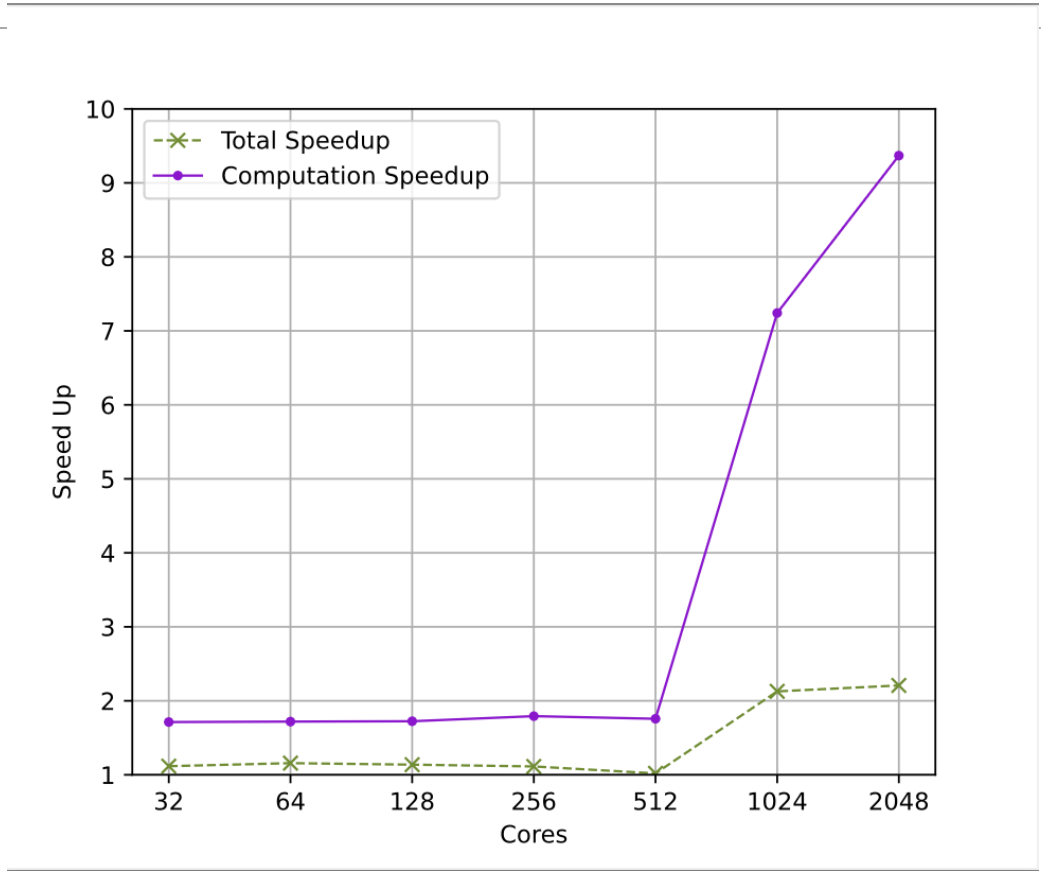
- Weak scaling





Performace evaluation of subarray_x in 3D FFT





Conclusions

- Developed MPI subarray datatype with arbitrary given storage order
- Use the datatype to implement flexible in-flight data rearrangement, performance tested
- Examined the strided access impact on FFT computation

- Potential in frequent data movement&rearrangement

Thank You!

Questions?

Natural Sciences and Engineering Research Council of Canada
Conseil de recherches en sciences naturelles et en génie du Canada
Canada

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC).