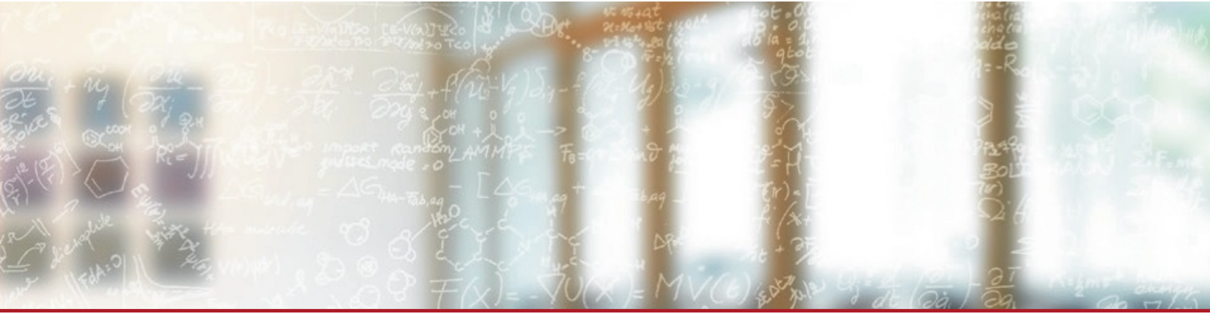
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich



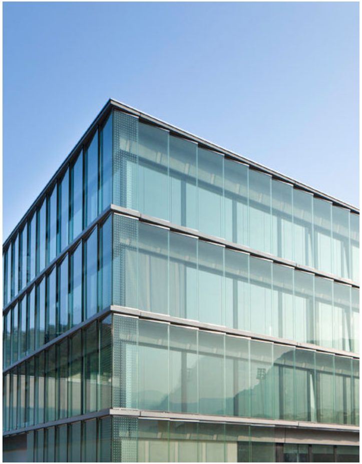
Improved scaling of massively parallel FFTs on modern supercomputers

Cray User Group Meeting (CUG) 2021

Andreas Jocksch (CSCS), Nikolaos Tselepidis (ETH Zurich), Vasileios Karakasis (CSCS)

11 May, 2021

Improved scaling of massively parallel FFTs on modern supercomputers



- Motivation
- Parallelisation approaches
- Implementation of the parallelisation
- Alternative decompositions
- More flexibility, resolution
- Distributed 1D FFT
- Benchmarks
- Conclusions and outlook

Motivation

- Fast Fourier Transform (FFT) algorithm part of many HPC applications
  - Plasma physics
  - Chemistry and material science
- Mostly 2D or 3D Fourier transforms
- Distributed computation is communication intensive, scaling is a challenge
- Optimised MPI collective routines are available – applied to FFTs

Parallelisation approaches

- Example of 3D FFT, applies also to 2D and 1D FFTs
- Slab decomposition (preferred decomposition)
- Pencil decomposition (more tasks than number of grid points in one direction)
- Higher order decompositions (rarely used)
- Binary exchange

Parallelisation approaches contd.

- Slab and pencil decomposition require all-to-all communication
- Binary exchange requires point-to-point communication
- Large message sizes: communication and computation overlap
- We focus on strong scaling – small message sizes

Implementation of the parallelisation

- Multiple processing units with shared memory on the node, data transfer within the node assumed to have zero cost
- High speed network between nodes assumed to be fully connected (good approximation for current Cray machines with Aries or Slingshot network), bandwidth-latency cost model
- Computer networks might support more than one port per node, message size dependent, on Aries network short messages many ports and long messages one port
- For pure MPI solutions latency is reduced by collecting all messages on the nodes before sending them and distributing them after receiving (presented at CUG 2018) $\longrightarrow$ reduction of number of messages
- Efficient `MPI_Alltoall` or `MPI_Alltoallv` required, persistent MPI well suited (initialisation phase for setup of the communication algorithms), no performance difference between these two collectives

FFTs with pencil decomposition

- Comparison between our `all-to-all` routine from “ext_mpi”
  https://github.com/eth-cscs/ext_mpi_collectives
  and standard MPI `all-to-all` as reference
- Strong scaling for pencil decomposed 3D FFT (FFTW) 600^3, 768^3 and 1200^3 grid-points double precision real numbers, 12 (3×4) tasks per node
- Benefits for small message sizes (large number of nodes) visible in the strong scaling experiments

---

- A totally revised version of the library will be published soon

FFTs with pencil decomposition contd.

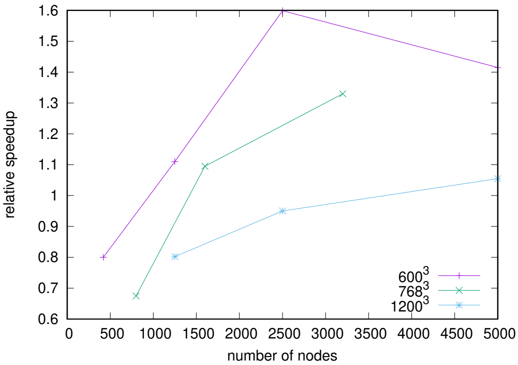
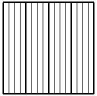
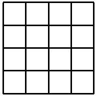


Figure: Relative speedup of our pencil decomposed 3D FFTs ("ext_mpi" `all-to-all` plus FFTW) with respect to standard MPI (FFTW), 12 (3×4) tasks per node, 600^3, 768^3 and 1200^3 grid-points, Cray XC50, presented at CUG 2018

Alternative decompositions

- In many cases decompositions enforced by constraints of the solver
- For particle in cell (PIC) method minimum communication volume for particles is obtained by division of domain (cube) in three directions

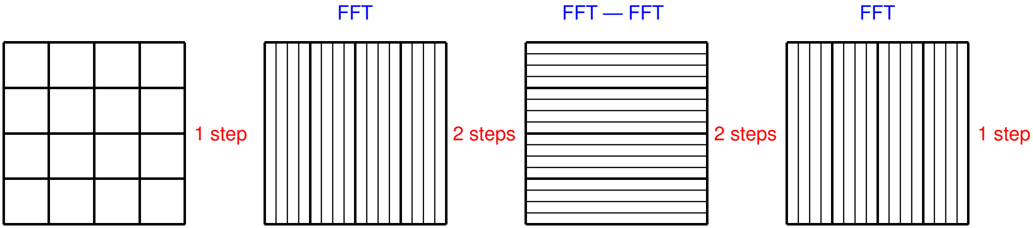
in 2D:  versus 

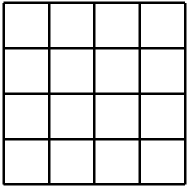
- Either additional communication steps are required for rearrangement of data or FFTs are applied directly to the decomposition $\longrightarrow$ parallelisation of the FFT algorithm in higher dimension than necessary for resolution and number of nodes given

More flexibility, resolution

- FFT kernels for large prime numbers solved in parallel (parallel discrete Fourier transform DFT)
- Parallel DFT solve with `MPI_Allgather` or `MPI_Reduce_scatter_block`
- Optimised routines exploiting multiple communication ports per node
- `allgather` might also be applied to non-prime numbers, if no proper decomposition can be made, or the prime factors are small and many ports are available
- Example: 4×4 distributed 2D mesh and 3 ports per node; Poisson equation

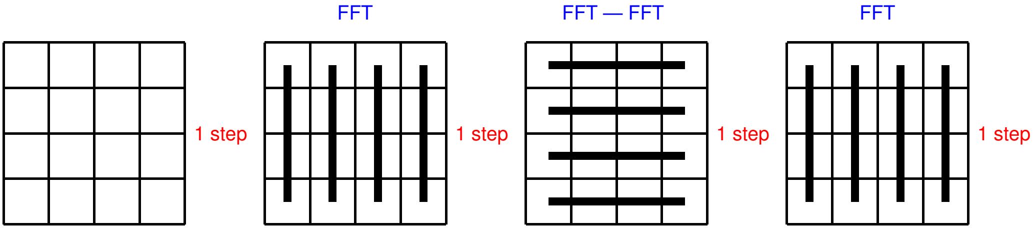
More flexibility, resolution contd.

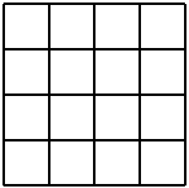




- 2 additional steps for data rearrangement (3 ports per node)
- Minimum computational cost

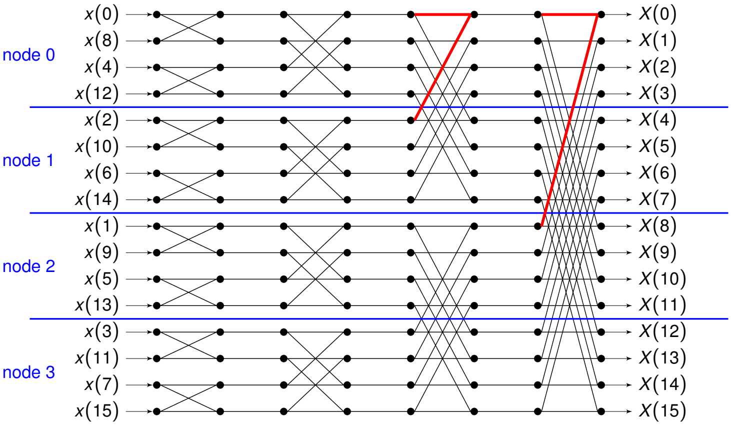
More flexibility, resolution contd.





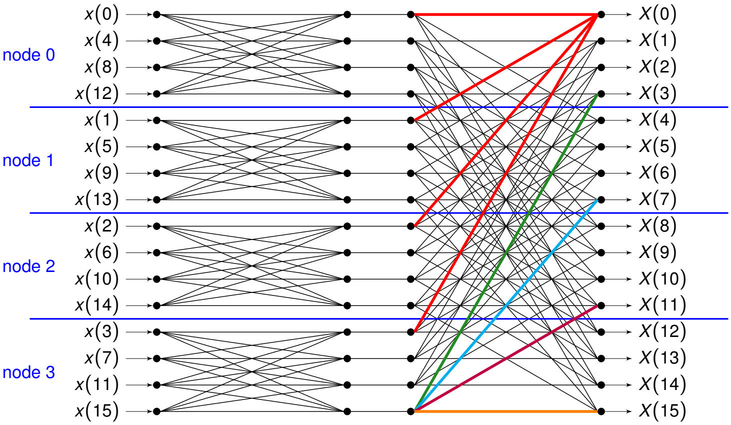
- 1 communication step per FFT (3 ports per node)
- Minimum computational cost for `alltoall` solution
- More restrictive with respect to resolutions
- `allgather` solution also with 1 communication step per FFT/DFT

Distributed 1D FFT



Radix 2, decimation in time

Distributed 1D FFT contd.

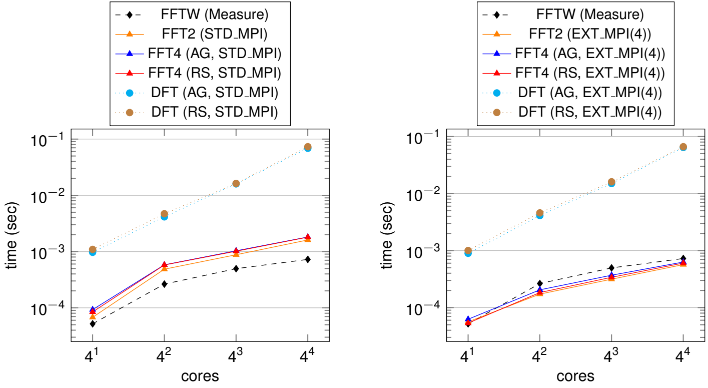


Radix 4, decimation in time

Benchmarks

- Distributed 1D FFT radix 2 and 4 and comparison with FFTW
- Reordering of the data before execution included in the benchmark
- 4 MPI tasks per node communicating with each other using shared memory ("ext_mpi" library)
- Both, `allgather` and `reduce_scatter_block` approach for small DFTs (FFT kernels)
- Local FFTs within the node computed with FFTW

Benchmarks contd.

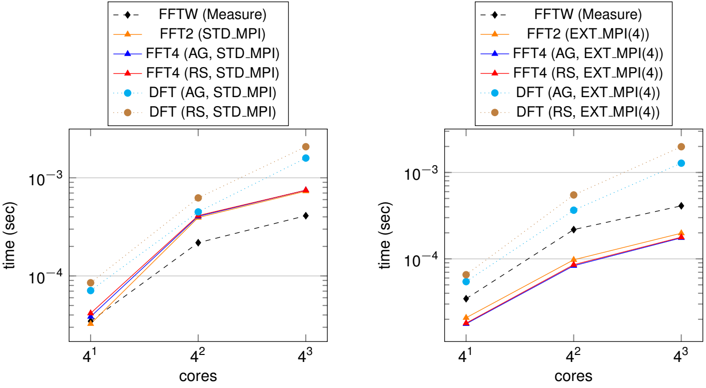


(a) 256 points per core (STD_MPI)

(b) 256 points per core (EXT_MPI(4))

Figure: Weak scalability of various FFT and DFT implementations on Cray XC40 KNL, using standard MPI (left), and ext_mpi with shared memory between every 4 cores (right) in the custom implementations.

Benchmarks contd.



(a) 64 points per core (STD_MPI)

(b) 64 points per core (EXT_MPI(4))

Figure: Weak scalability of various FFT and DFT implementations on Cray XC40 KNL, using standard MPI (left), and ext_mpi with shared memory between every 4 cores (right) in the custom implementations.

Benchmarks contd.

- 256 grid points per task (core) is the break even for the `allgather` / `reduce_scatter_block` (ext_mpi) approach versus the `alltoall` (standard MPI) method
- Large part of the speedup of “ext_mpi” compared to “standard MPI” comes form the data reordering before the execution of the FFTs, done with `all-to-all` (ext_mpi), not necessary for solution of the Poisson equation

Literature

- Andreas Jocksch, Matthias Kraushaar and David Daverio: Optimised all-to-all communication on multicore architectures applied to FFTs with pencil decomposition, Concurrency Computat Pract Exper., 2018
- Andreas Jocksch, Noe Ohana and Emmanuel Lanti and Vasileios Karakasis and Laurent Villard: Optimised allgatherv, reduce_scatter and allreduce communication in message-passing systems, arXiv, 2020

Conclusions and outlook

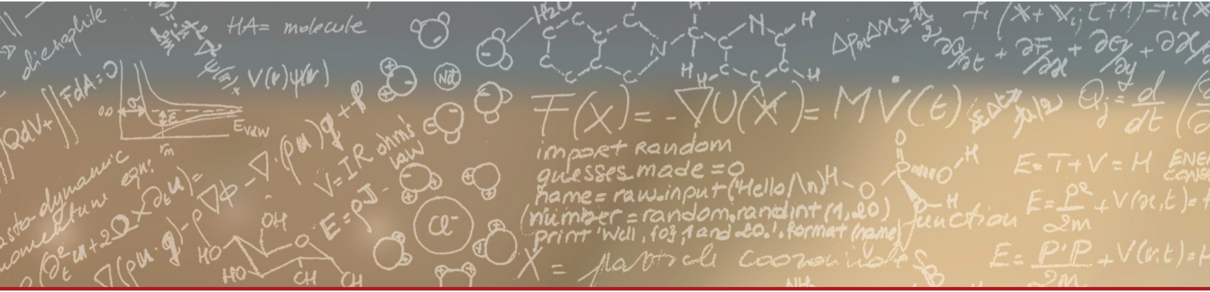
- For pure MPI solutions short messages all-to-all communication can be accelerated by using shared memory on the node
- Optimised `Allgather` and `Reduce_scatter_block` provide more degrees of freedom to parallelise FFTs (DFTs)
- Improved strong scaling properties of parallel FFTs
- Work in progress, library will be further developed

Acknowledgements

Noé Ohana, Emmanuel Lanti, Laurent Villard (Swiss Plasma Center, EPFL)

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich



Thank you for your attention.

https://github.com/eth-cscs/ext_mpi_collectives