

Shenfun/mmpi4py-fft

Mikael Mortensen, University of Oslo, Department of Mathematics
and

Lisandro Dalcin, King Abdullah University of Science and Technology

<https://github.com/spectralDNS/shenfun>

<https://bitbucket.org/mmpi4py/mmpi4py-fft>

mpi4py-fft

<https://bitbucket.org/mpi4py/mpi4py-fft>



- Python module for pencil or slab decompositions
- Highly configurable – any dimensions (2,3,4,...) – any transforms over any axis in any order
- Wrapping FFTW using (serial) pyFFTW
- Transpose-free transforms using **Alltoallw**

Simplicity



- Once initialized and work arrays set up the code is basically:

for all axes (any dimensionality):
do FFT along aligned axis
Alltoallw communicate to new alignment
FFT along last aligned direction

Nothing more, nothing less.
No intermediate copying, no transposes
~500 lines of code

Shenfun

<https://github.com/spectralDNS/shenfun>



- Python module for automating the spectral Galerkin method on tensor product domains
 - Mixing Fourier and Legendre/Chebyshev discretizations
 - Uses mpi4py-fft to do transforms (pencil or slab)
 - High-level coding – similar to FEniCS (www.fenicsproject.org)
 - https://rawgit.com/spectralDNS/shenfun/master/docs/src/KleinGordon/kleingordon_bootstrap.html

```
from shenfun import *
from mpi4py import MPI
comm = MPI.COMM_WORLD
N = (512, 1024, 1033)
D0 = chebyshev.Basis(N[0])
F0 = fourier.C2CBasis(N[1])
F1 = fourier.R2CBasis(N[2])
T = TensorProductSpace(comm, (D0, F0, F1))
```

Performs the decompositions

An arrow points from the text 'Performs the decompositions' to the `TensorProductSpace` line in the code block above.

Questions?



1) Why did you write your own FFT?

I needed to do parallel FFT in Python. At the time (2014) there was nothing but serial alternatives.

2) What considerations are important for you in an FFT implementation?

Right now what's most important is flexibility/configurability. I need transforms of all sorts in the **shenfun** module. No limitations: 2D, 3D, 4D, Fourier, Cosine, Sine, Chebyshev or no transforms. In any order. To be able to solve any kind of partial differential equation on tensor product domains.

3) What might you look for if there were to be a unified FFT interface (similar to BLAS, LAPACK and SCALAPACK interfaces)?

Configurability. I like the FFTW interface, with planning and execution.

Questions?



4) How important are performance, portability, and scalability for you?

Performance is important, but less so than configurability. Portability is a must. Scalability is nice, but often outside the hands of the software developer (bandwidth etc.)

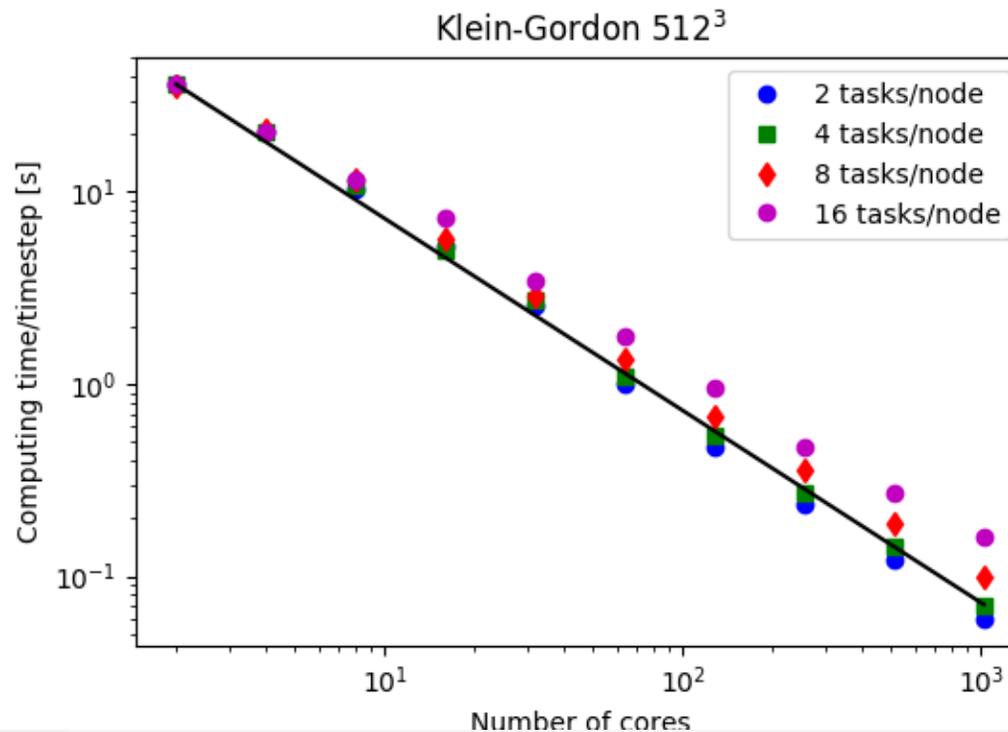
5) Will FFT be needed in exascale computing and if so how will it be achieved?

Yes, but not a major concern of mine, so I have no further comments

6) What would be a good FFT benchmark or a good way to include the FFT in a high-performance computer benchmark?

Forward and backward transforms of random numbers (real or complex), output should equal input.

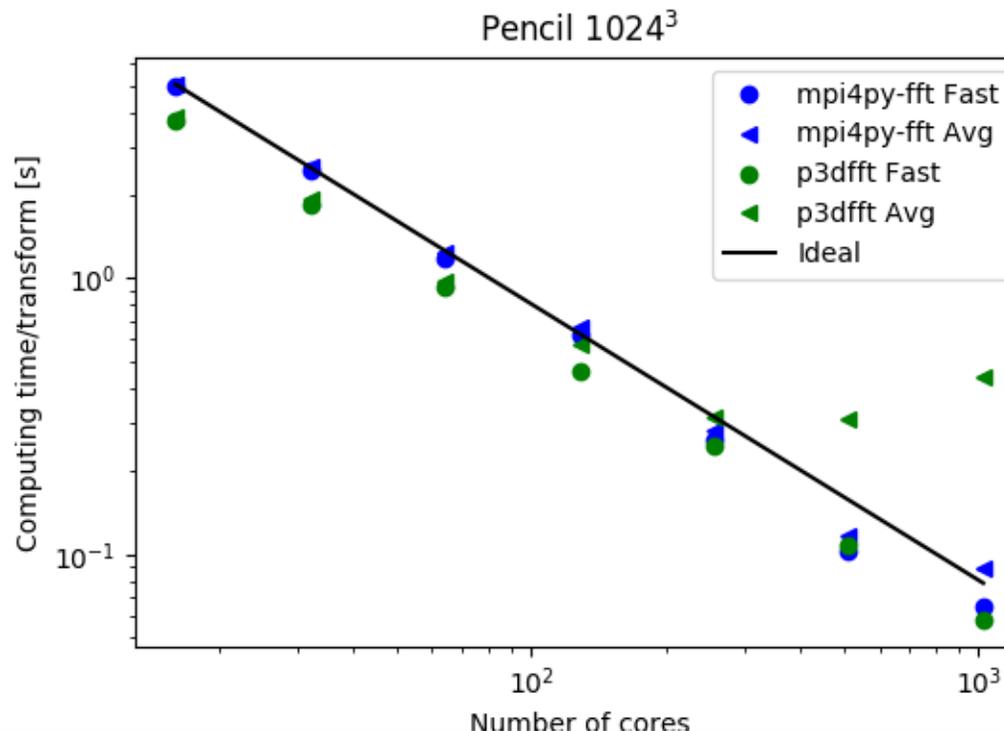
Strong scaling Sheheen II



The tasks per node is a setting on Sheheen II
Less tasks per node gives more memory per task.
With sufficient memory scaling is excellent

Klein-Gordon test case
Better than linear scaling because of memory issues with too large mesh per processor

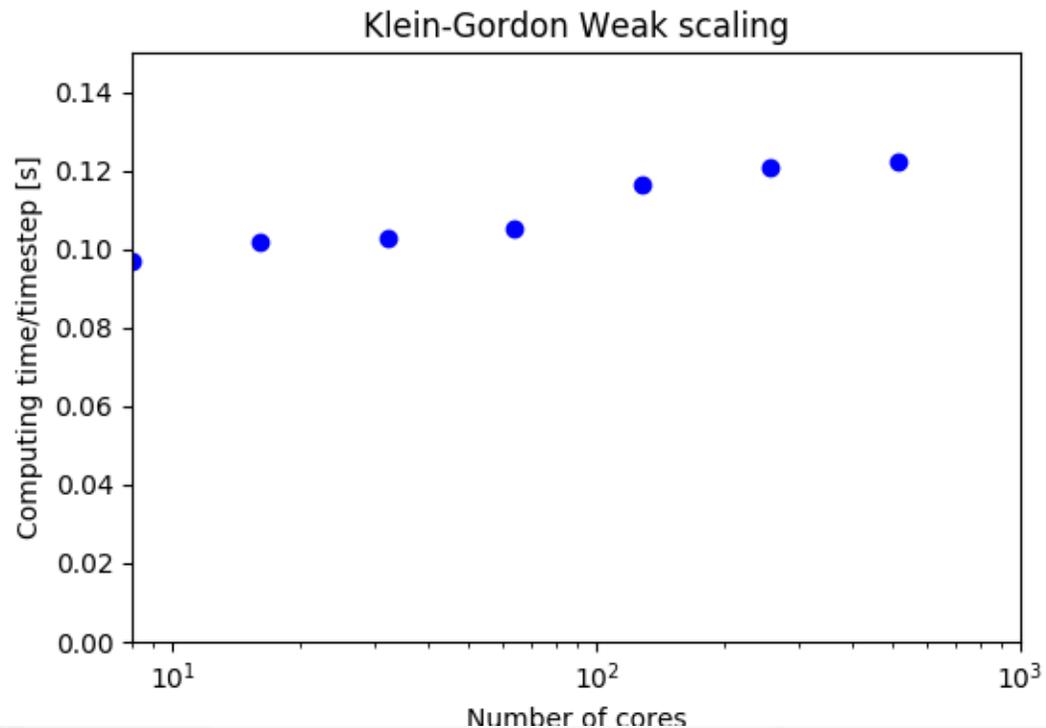
Forward/backward



P3DFFT:
driver_sine.c
modified to
store fastest
+ average
Includes normali-
zation time.
Default settings
otherwise.

5 consecutive forward/backward transforms.
Fastest transform and average shown. Reproduced twice
with more or less exactly the same results
On low counts P3DFFT probably faster because Alltoall is
faster than Alltoallw. On high core counts there may be
delays in transpose operations not required by Alltoallw

Weak scaling Shaheen II



Mesh of size
262144 per task
Corresponds to
 64^3 per task

Not accounting for $N \log N$ factor in transforms because element-wise operations are just as time consuming. Scaling by $N \log N$ and the curve is more or less flat (giving the wrong impression of perfect weak scaling)