

FFTW

The discrete Fourier transform in one or more dimensions, MPI parallel

FFTW is a C subroutine library for computing the discrete Fourier transform in one or more dimensions, of arbitrary input size, and of both real and complex data (as well as of even/odd data, i.e. the discrete cosine/sine transforms or DCT/DST). The FFTW library allows for MPI parallel, in-place discrete Fourier transform, with data distributed over number of nodes.

Two versions, **3.3.3** and **2.1.5** of FFTW are available on Anselm, each compiled for **Intel MPI** and **OpenMPI** using **intel** and **gnu**** compilers. These are available via modules:

Version	Parallelization
---------	-----------------

FFTW3.3.3	pthread, openMP
-----------	-----------------

FFTW3.3.3	pthread, openMP
-----------	-----------------

FFTW2.1.5	pthread
FFTW2.1.5	pthread
FFTW2.1.5	pthread

FFTW3.3.3	openMPI
FFTW3.3.3	openMPI

FFTW3.3.3	intel MPI
FFTW3.3.3	intel MPI

FFTW2.1.5	openMPI
FFTW2.1.5	openMPI

FFTW2.1.5	intel MPI
FFTW2.1.5	intel MPI

```
$ module load fftw3
```

The module sets up environment variables, required for linking and running fftw enabled applications. Make sure that the choice of fftw module is consistent with your choice of MPI library. Mixing MPI of different implementations may have

unpredictable results.

Example

```
#include <fftw3-mpi.h>
int main(int argc, char **argv)
{
    const ptrdiff_t N0 = 100, N1 = 1000;
    fftw_plan plan;
    fftw_complex *data;
    ptrdiff_t alloc_local, local_n0, local_0_start, i, j;

    MPI_Init(&argc, &argv);
    fftw_mpi_init();

    /* get local data size and allocate */
    alloc_local = fftw_mpi_local_size_2d(N0, N1, MPI_COMM_WORLD,
                                          &local_n0, &local_0_start);
    data = fftw_alloc_complex(alloc_local);

    /* create plan for in-place forward DFT */
    plan = fftw_mpi_plan_dft_2d(N0, N1, data, data, MPI_COMM_WORLD,
                                 FFTW_FORWARD, FFTW_ESTIMATE);

    /* initialize data */
    for (i = 0; i < local_n0; ++i) for (j = 0; j < N1; ++j)
    {   data[i*N1 + j][0] = i;
        data[i*N1 + j][1] = j; }

    /* compute transforms, in-place, as many times as desired */
    fftw_execute(plan);

    fftw_destroy_plan(plan);

    MPI_Finalize();
}
```

Load modules and compile:

```
$ module load impi intel
$ module load fftw3-mpi
```

```
$ mpicc testfftw3mpi.c -o testfftw3mpi.x -Wl,-rpath=$LIBRARY_PATH -lfftw3_mpi
```

Run the example as Intel MPI program.

Read more on FFTW usage on the FFTW website.