FFT Libraries

Iain Bethune, Gavin Pringle, Joachim Hein
EPCC
The University of Edinburgh

Overview

- Use of Libraries
- FFT Libraries
  - General usage notes
- FFTW
  - Overview
  - Using the Library
  - Performance
- FFT Practical

Use of Libraries

- What is a library?
  - Collection of pre-implemented routines available in some sort of package (static archive, shared library, Framework, DLL…)
  - Defined (and hopefully documented) API
  - Often provided with operating system (e.g. Scientific Linux), available via package managers, or binary/source downloads
  - Can be combined with your code by linking (and perhaps #including header files)

Use of Libraries

- Why use libraries?
  - Easier - save effort and time, no duplicated code
  - Avoid bugs – libraries are (hopefully) well tested
  - Performance – usually contain efficient implementations, perhaps optimised for a given system (vendor-specific architectures), or auto-tuning

FFT Libraries

- FFTs do not normalise
  - Each FFT/Inverse FFT pair scales by a factor of N
  - Usually left as an exercise for the programmer.

- DFTs are complex-to-complex transforms, however, most applications require real-to-complex transforms
  - Simple solution: set imaginary part of input data to be zero
    - This will be relatively slow
  - May be better to pack and unpack data
    - Place all the real data into all slots of the input, complex array (of length (n/2) and then unpack the result on the other side (O(n))
    - Around twice as fast as the simple solution
    - Good details in Numerical Recipes
  - Some libraries have real-to-complex wrappers

FFT Libraries

- Multidimensional FFTs
  - simply successive FFTs over each dimension
    - order immaterial: linearly independent operations
  - pack data into 1D array – see practical
  - strided FFTs
  - some libraries have multidimensional FFT wrappers

- Parallel FFTs
  - performing FFT on distributed data
  - 1D FFTs are cumbersome to parallelise
    - Suitable only for huge N
  - parallel, array transpose operation
    - distributed data is collated on one processor before FFT
    - more in next lecture
  - some libraries have parallel FFT wrappers

FFTW

- Fastest Fourier Transform in the West
  - www.fftw.org
- The FFTW package was developed at MIT by Matteo Frigo and Steven G. Johnson
- Free under GNU General Public License
- Portable, self-optimising C code
  - Runs on a wide range of platforms
- Arbitrary sized FFTs of one or more dimensions
  - Fastest routines where extents are composed of powers of 2, 3, 5 and 7 (other sizes can be optimised for at configuration time)

FFTW: FFTW2 vs FFTW3

- Previous version: “FFTW2”
  - Many legacy codes employ FFTW2
  - Simple(r) C interface, with wrappers for many other languages
  - Supports MPI
  - Rest of this lecture assumes “FFTW2”
- New version: “FFTW3”
  - Different interface to FFTW2 – to allow planner more freedom to optimise
    - Users must rewrite code
  - Doesn’t support MPI (currently in beta release)
    - Most codes implement the parallel transpose, and perform the 1D FFTs using FFTW3
  - Somewhat faster than FFTW2 (~10% or more)

FFTW: Details

- Can perform FFTs on distributed data
  - MPI for distributed memory platforms
  - OpenMP or POSIX for SMPs
- If users rewrite their code to this FFT just once then the user is saved from
  - learning platform dependent, proprietary FFT routines
  - rewriting their code every time they port their code
    - No standard interface to FFTs
  - drastically rewriting their makefiles
    - location of FFTW libraries may vary
- FORTRAN wrappers for the majority of routines
  - Currently FORTRAN FFTs are not “in-place”
    - The input and output arrays must be separate and distinct
  - Nor are the strided FFT calls (in FFTW 2)
    - The input/output arrays must be contiguous

FFTW: Plans

- All FFT libraries pre-compute the *twiddle factors*
- FFTW 'plans' also generates the FFT code from *codelets*
  - Codelets compiled when FFTW configured
- Two forms of plans
  - Estimated
    - The best numerical routines are guessed, based on information gleaned from the configuration process.
  - Measured
    - Different numerical routines are actually run and timed with the fastest being used for all future FFTW calls using this plan.
- Old plans can be reused or even read from file: *wisdom*

FFTW: Installation

- Download library from the website and unpack (gzipped tar file)
- `./configure; make; make install`
  - Probes the local environment
  - Compiles many small C object codes called *codelets*
  - User can provide non-standard compiler optimisation flags
  - Libraries (both static and dynamic) are then installed along with online documentation and header files
- Includes test suite
  - Very important for any numerical library
- Pre-installed with Scientific Linux

FFTW: Compiling code

- **`gfortran fft_code.f -O3 -lfftw`**
  - If using C, FFTW must be linked with `-lfftw -lm`
  - If the FFTW library configured for both single and double precision, then link with **`-lsfftw`** and **`-lfftw`**, respectively.
- Example FORTRAN code:

```
integer plan
integer, parameter :: n = 1024
complex in(n), out(n)
! plan the computation
call fftw_f77_create_plan(…)
! execute the plan
call fftw_f77_one(…)
```

  - NB: actual incantations are not given here as reading documentation is integral to utilising any numerical library

FFTW: Performance

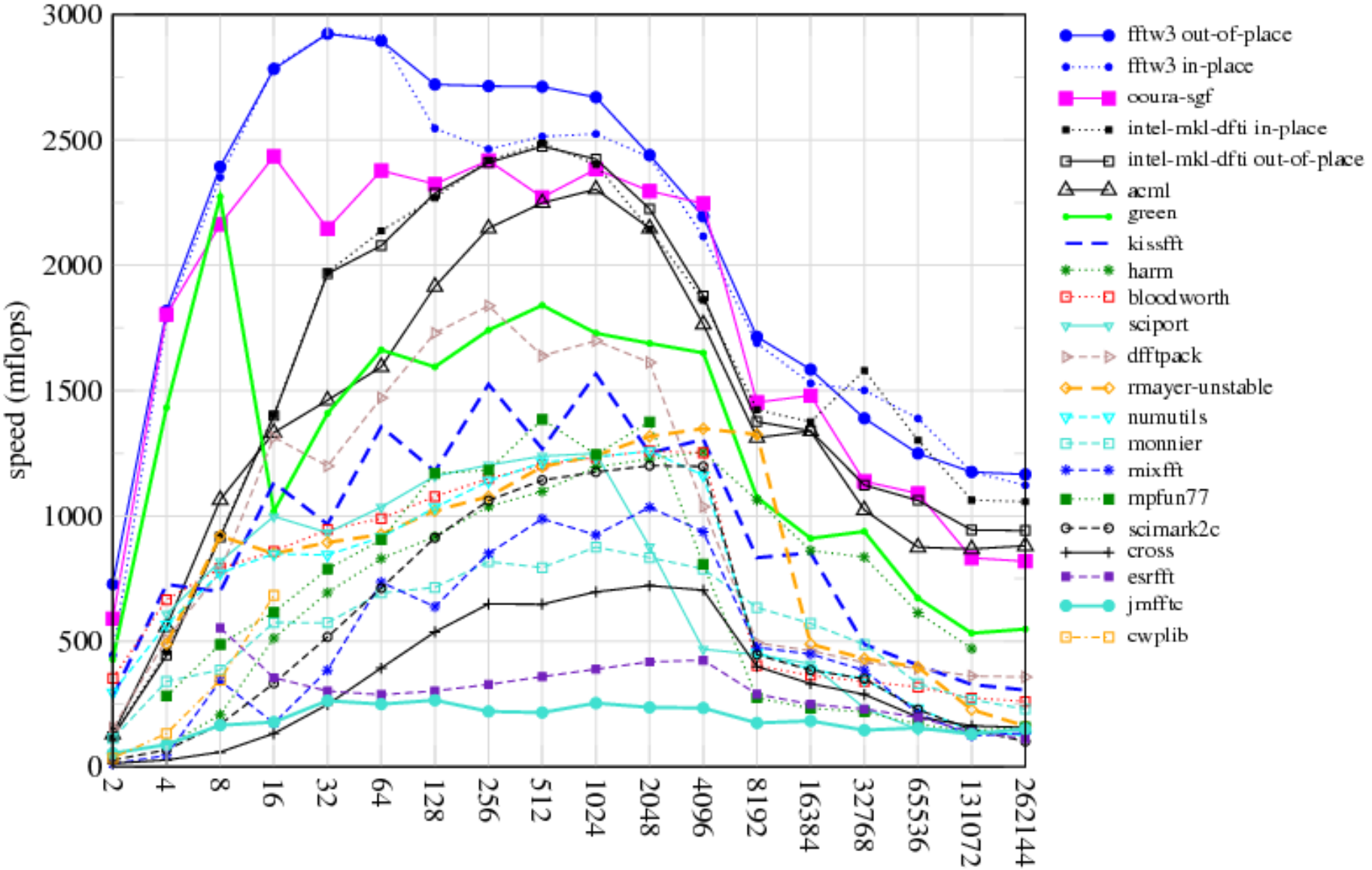
- The FFTW homepage, www.fftw.org, details the performance of the library compared to proprietary FFTs on a wide range of platforms.

- The FFTW library is faster than any other portable FFT library

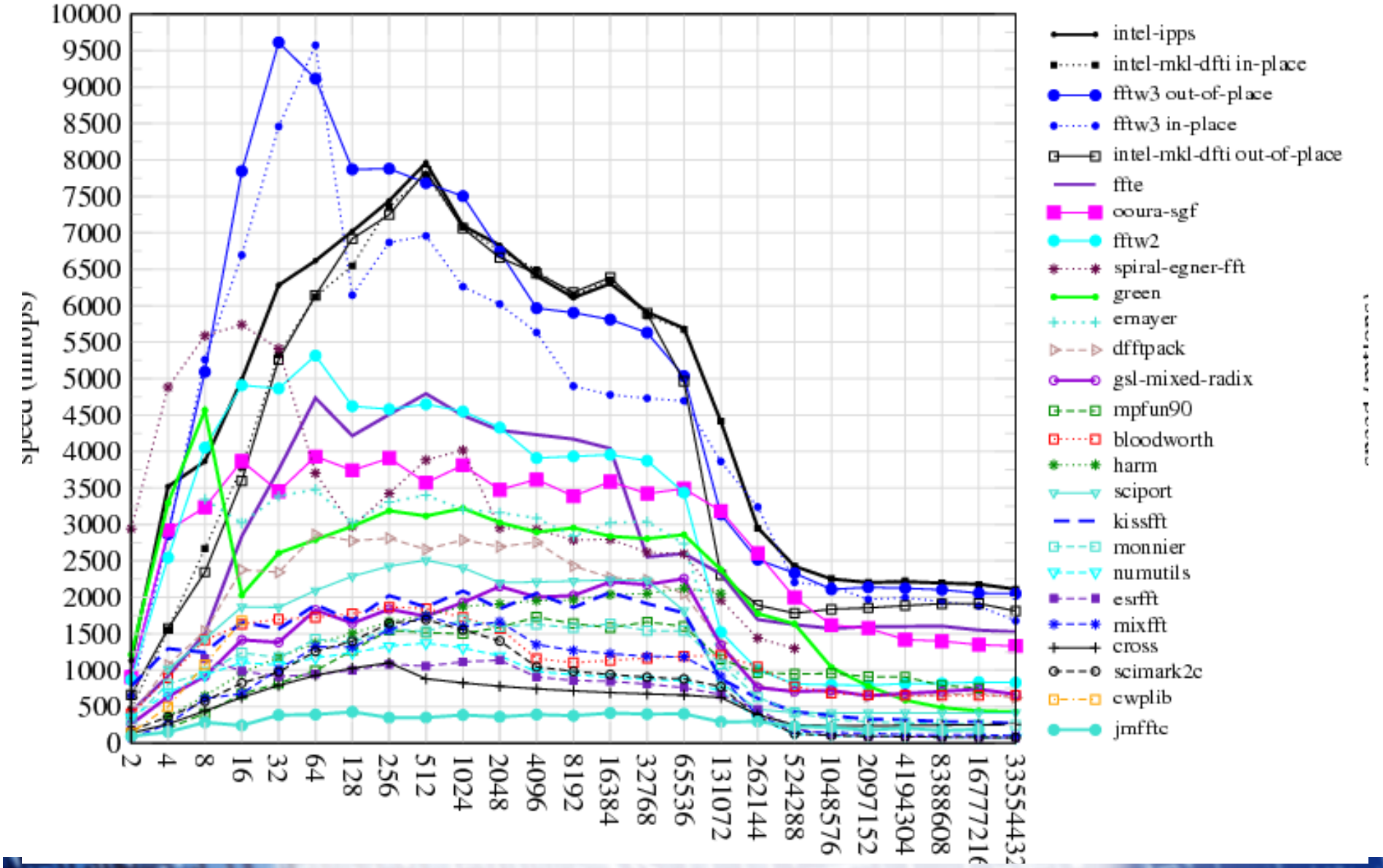
- Comparable with machine-specific libraries provided by vendors

- Performance results from http://www.fftw.org/speed/

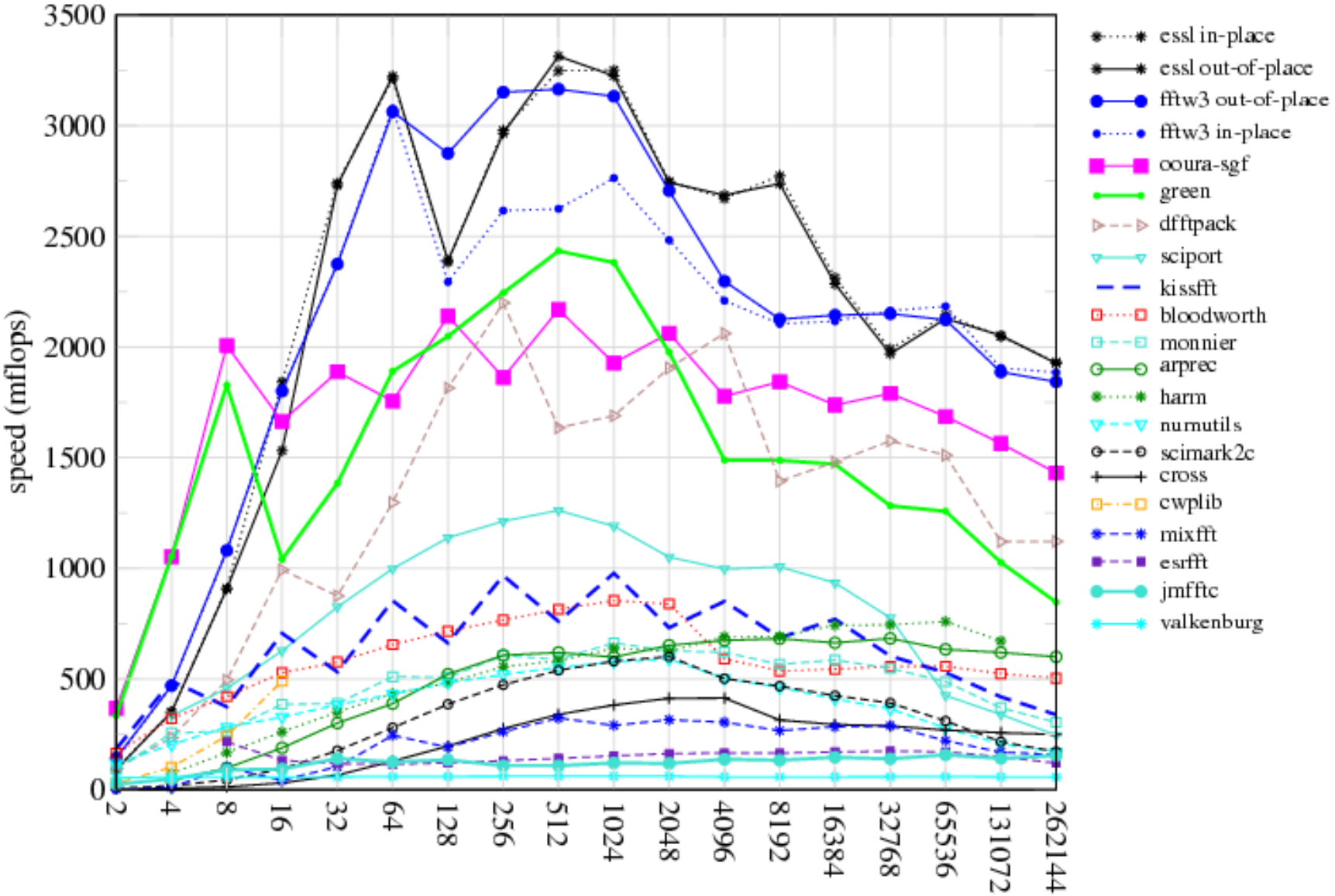
FFTW: AMD Opteron 275 2.2 GHz



FFTW: Intel Core Duo 3.0 GHz



FFTW: IBM POWER5 1.65 GHz



FFTW: Accolades

- Winner of the 1998 J.H. Wilkinson Prize for Numerical Software
  - awarded every four years to the software that "best addresses all phases of the preparation of high quality numerical software."
- Quotes from www.fftw.org.
  - “It’s the best FFT package I have ever seen”
  - “It performs my standard 256 iterations of 1024pt complex FFT about 20 times faster than the previous one I used.”
  - “FFTW is the best thing since microwave popcorn”
- Dr. Richard Field
  - Former Vice-principal of University of Edinburgh and Chairman of NAG
  - “I think FFTW is terrific. It’s the best piece of software I’ve seen written in a bunch of years. […] I give FFTW very high marks (probably as high marks as I would ever give).”

FFT Summary

- First lecture:
  - Introduced both the continuous and discrete forms of the Fourier Transform
  - Stated the translation theorems of the Fourier transform
    - Scaling, Shifting, Convolution and Correlation
  - Fast Fourier Transform
- Second lecture:
  - FFTW
    - Fast, robust and portable
    - FORTRAN and C, serial and parallel.
    - Simple to use
    - Recommended and used in major projects by EPCC

FFT Practical

- FFTs are also employed in JPEG image compression/ smoothing
  - MPEG also utilises FFTs along with other graphical techniques
- Examine how manipulating image data in Fourier space alters the transformed image using FFTW 2

FFT Practical

- Full instructions in practical handout sheet

- Two main image transformations to attempt:
  - How many higher frequencies can be removed, reducing the size of the image in memory, without visibly altering the image?
  - Performing edge detection by removing only lower frequencies.

- Hint: remember that Fourier space is periodic
- Begin with an estimated plan and a 1D, non-strided, complex-to-complex transform

FFT Practical

- F90, C, and Java makefiles and templates available

- For exercises
  - FORTRAN routines:
    - fftw_f77_create_plan, fftw_f77_one
  - C routines:
    - fftw_create_plan, fftw_one
  - Java
    - new Plan(), Plan.transform()
  - NB: Read the documentation for arguments, usage etc.

- Online documentation on FFTW
  - the `info fftw` command or www.fftw.org

FFT Practical : Further work

- How competitive is FFTW compared to other FFT libraries?
  - Alter previous code and time sections for plans and FFTs for FFTW and FFTPACK, say.
    - At present, FFTW is the only FFT library available
    - If interested, install FFTPACK in your home space
  - Extend data array to compare FFT libraries over a range of `N`
    - Including non-power-of-two `N`
    - fill array with zeros first

- Convert image manipulation code to use
  - 2D FFTW
  - Investigate the use of “wisdom”

FFT Practical: Mel Gibson in Fourier Space