FFTE

Daisuke Takahashi
University of Tsukuba, Japan

About

- FFTE is a Fortran subroutine library for computing the Fast Fourier Transform (FFT) in one or more dimensions.
- It includes real, complex, mixed-radix and parallel transform.
- FFTE may be faster than other publically-available FFT implementations and vendor-tuned libraries.
- Available at http://www.ffte.jp/

Features

- Parallel transforms
  - Shared / Distributed memory parallel computers (OpenMP, MPI and OpenMP + MPI)
- High portability
  - Fortran + OpenMP + MPI
- Data layout
  - 1-D decomposition
  - 2-D decomposition (for parallel 3-D FFT)
- HPC Challenge Benchmark
  - FFTE's 1-D parallel FFT routine has been incorporated into the HPC Challenge (HPCC) benchmark.

Questions?

1) Why did you write your own FFT?

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

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

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

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

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

1) Why did you write your own FFT?

- FFTE is a byproduct developed when calculating pi to high precision.
  - In multiple-precision multiplication of more than several thousand decimal digits, FFT-based multiplication is the fastest method.
- In 1990's, there were no open source parallel FFT libraries.
- So I decided to develop a parallel FFT library by myself.

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

- Performance
  - Minimizing the number of cache misses
  - SIMD instructions
- Ease of use: routine interfaces
- Portability
  - Communication: MPI
  - Computation: Fortran + OpenMP

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

- Currently, there are two de facto standards for the interfaces of FFT libraries.
  - FFTW's interface
  - Intel MKL's Discrete Fourier Transforms Interface (DFTI)
- These interfaces support only one-dimensional block distribution.
- I hope these interfaces also support two-dimensional (or more) distributions.

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

- Performance is the most important.
- Scalability is also important for high performance.
- Portability should not be impaired for performance reasons.

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

- I think that FFT is still needed in exascale computing.
- However, it seems that FFT using whole exascale system is not realistic.
- The performance of parallel one-dimensional FFT in K computer (82944 nodes, 10.6 PFlops peak) was only 252 TFlops (approx. 2.4 % of peak).
  - More than 2/3 of the execution time is dominated by all-to-all communication.

An example of conditions necessary to achieve exaflops in FFT

- Assuming that computation and communication are completely overlapped, the performance of the FFT depends on the total communication bandwidth.
- The number of data points for FFT: $N = 2^{60}$
  - The number of arithmetic operations:
    $$5N \log_2 N \approx 3.46 \times 10^{20}$$
  - Memory usage: 16EB
- The number of nodes: $P = 2^{24} \approx 1.68 \times 10^7$
  - All-to-all message size: 64KB
- In this case, the communication bandwidth of approx. 15.9 GB/s per link is required.

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

- The FT benchmark of NAS Parallel Benchmarks (NPB) and the G-FFT of HPC Challenge Benchmark (HPCC) are good benchmarks.
- However, in recent supercomputers, these benchmarks are becoming a benchmark for all-to-all communication rather than measuring the performance of FFT.