Parallel Fourier Transform

A Practical Guide

Dhrubaditya Mitra

Indian Institute of Science, Bangalore, 560012

Outline

- Motivation
- Serial FFT
  - Serial FFT : Basic Algorithm
  - FFT of Real Data
  - FFT in d>1
  - Limitations
  - Advertising FFTW
- Parallel FFT
- Applications : Spectral Methods

Motivation

"If you speed up any non-trivial algorithm by a factor of million or so, the world beat a path towards finding useful application for it " -Numreical Recepies.

- Convolution and Correlation
- Optimal Filtering
- Power Spectrum Estimation
- Integral Transforms (Wavelet and Fourier)
- Spectral Method for solving PDE

(Continuous) Fourier Transform

- (Inverse) Fourier Transform

$$h(t) = \int_{-\infty}^{+\infty} H(f) e^{-2\pi i f t} df$$

- (Forward) Fourier Transform

$$\int_{-\infty}^{+\infty} h(t) e^{+2\pi i f t} dt = H(f)$$

Discrete Fourier Transform

- Finite time series, sampled at an interval Δ

$$h_k = h(t_k) = h(k\Delta) \qquad k = 0, \ldots N-1$$

- Discrete (Forward) Fourier Transform

$$\sum_{k=0}^{N-1} h_k e^{2\pi ikn/N} = H_n; \qquad \Delta H_n = H(f_n)$$

with $f_n = -f_c, .. + f_c$, where $f_c = \frac{1}{2\Delta}$

- Discrete (Inverse) Fourier Transform

$$h_k = \frac{1}{N} \sum_{n=0}^{N-1} H_n e^{-2\pi ikn/N}$$

Discrete Fourier Transform(contd)

Discrete Fourier Transform

$$h_k = \frac{1}{N} \sum_{n=0}^{N-1} H_n e^{-2\pi i k n / N}$$

-

$$H_{-n} = H_{N-n}$$

- Output in "wrap around order"
- Naively Order N^2 algorithm !

$$H_n = \sum_{k=0}^{N-1} W^{nk} h_k; \qquad W = e^{2\pi i/N}$$

Fast Fourier Transform

$$
\begin{aligned}
H_n &= \sum_{k=0}^{N-1} (W^n)^k h_k \\
&= \sum_{k=0}^{\frac{N}{2}-1} W^{2kn} h_{2k} + \sum_{k=0}^{\frac{N}{2}-1} W^{(2k+1)n} h_{2k+1} \\
&= \sum_{k=0}^{\frac{N}{2}-1} (W^n)^k h_{2k} + W^n \sum_{k=0}^{\frac{N}{2}-1} (W^n)^k h_{2k+1} \\
&= H_n^e + W^n H_n^o
\end{aligned}
$$

Fast Fourier Transform (contd)

$$H_n = H_n^e + W^n H_n^o$$

Fast Fourier Transform (contd)

$$H_n = H_n^{ee} + W^n H_n^{eo} + W^n H_n^{oe} + W^{2n} H_n^{oo}$$

Fast Fourier Transform (contd)

$$H_n = H_n^{eee} + W^n H_n^{eeo} + W^{2n} H_n^{eoe} + W^{2n} H_n^{eoo} + \ldots$$

Fast Fourier Transform (contd)

$H_n^{eoe..eeoo} = h_k$ for some k

To find Corresponding k : Bit Reversal

Reverse e and o

$e = 0, o = 1$, gives k in binary

Algorithmic Complexity

Let $C(p)$ denote the amount of computation needed for an array of size $N = 2^p$. Then,

$$C(0) = 0$$

$$C(p+1) = 2C(p) + 2^p$$

Hence $C(p) = 2^{p-1}p$, and

$$\boxed{C'(N) = \frac{N}{2}log_2(N)}$$

Serial FFT:Improvements

- Twidle Factor Trick : Evaluation of W^n involves time consuming evaluation of sines and cosines.

Serial FFT:Improvements

- Twidle Factor Trick : Evaluation of W^n involves time consuming evaluation of sines and cosines.
  - Make a look up table

Serial FFT:Improvements

- Twidle Factor Trick : Evaluation of W^n involves time consuming evaluation of sines and cosines.
  - Make a look up table
  - Lot of W^n are $1(n = 0)$, or$-i(n = N/4)$ (about $3/log_2(N))$

Serial FFT:Improvements

- Twidle Factor Trick : Evaluation of W^n involves time consuming evaluation of sines and cosines.
  - Make a look up table
  - Lot of W^n are $1(n = 0)$, or$-i(n = N/4)$ (about $3/log_2(N)$)
- Cache Efficiency

Serial FFT:Improvements

- Twidle Factor Trick : Evaluation of W^n involves time consuming evaluation of sines and cosines.
  - Make a look up table
  - Lot of W^n are $1(n = 0)$, or$-i(n = N/4)$ (about $3/log_2(N))$
- Cache Efficiency
  - one FLOP is faster than Memory Access

Serial FFT:Improvements

- Twidle Factor Trick : Evaluation of W^n involves time consuming evaluation of sines and cosines.
  - Make a look up table
  - Lot of W^n are $1(n = 0)$, or$-i(n = N/4)$ (about $3/log_2(N))$
- Cache Efficiency
  - one FLOP is faster than Memory Access
  - Cache Overflow

Serial FFT: Improvements(contd.)

IISc

Efficiency crucially depends on particular system architecture

- USE SYSTEM LIBRARIES (ESSL in IBM, DXML in Compaq etc)
- Disadvantage : No Portability
- Execption : FFTW

Serial FFT: Real Data

- $$H_{N-n} = (H_n)^*$$
  Calculate only half the output !
- Two for the price of one
- A different algorithm for Real Data.
- In-Place Transform.

Serial FFT : Data Storage

0	1	2		N-1	N	N+1

Array of Real Data, of size N. Last two elements (N,N+1) are zero.

Serial FFT : Data Storage

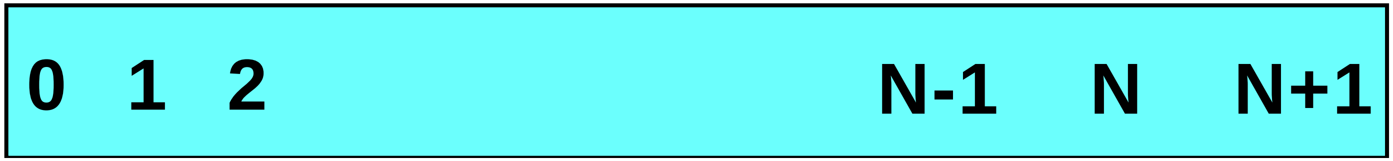
0	1	2	N-1	N	N+1

Array of Real Data, of size N. Last two elements (N,N+1) are zero.

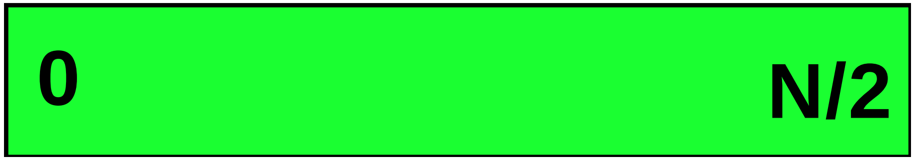
-N/2	0	N/2

Fourier Transform of Real Data. Complex data No array element is zero. But $H(-n) = H(n)^*$

Serial FFT : Data Storage

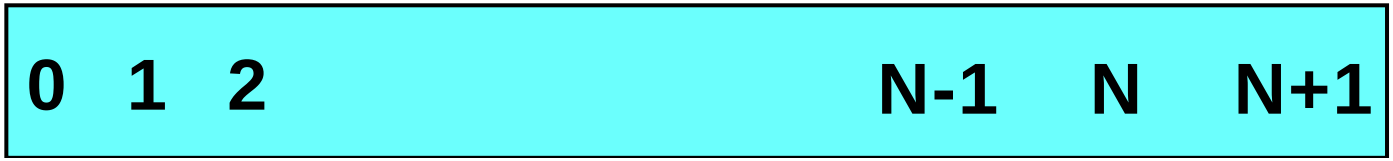


Array of Real Data, of size N. Last two elements (N,N+1) are zero.



Only the positive frequency part of complex data.

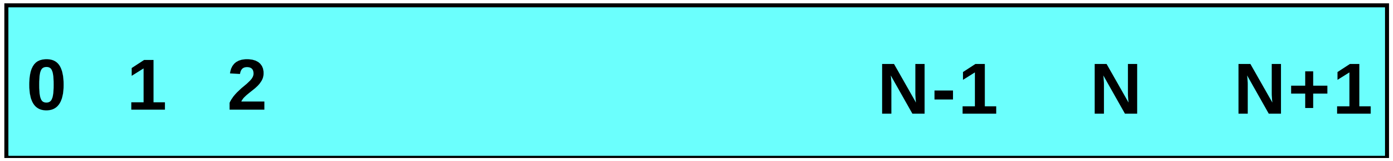
Serial FFT : Data Storage



Array of Real Data, of size N. Last two elements (N,N+1) are zero.

Only the positive frequency part of complex data, with real and imaginary part explicitly shown.

Serial FFT : Data Storage



Array of Real Data, of size N. Last two elements (N,N+1) are zero.

Only the positive frequency part of complex data, with real and imaginary part explicitly shown.

- zero-padding
- Imaginary part $H(0)$ and $H(N/2)$ is zero.

Serial FFT : 2-d

- essentially two transforms in two directions.
- Computational complexity : $N^2 log(N)$
- output in 'wrap around order'
- Real Input Data
  - zero-padding in the first direction. (for Fortran)
  - Output in the second direction in wrap around order.

Fast Fourier Transform : Limitations

- N should be power of small primes.
- For prime N algorithms are slower.
- Aliasing Error.

Advertising FFTW (www.fftw.org)

- Portable but efficient
- Callable from both C and Fortran.
- Free !! under GPL
- Parallel transform of both Shared and Distributed memory machines

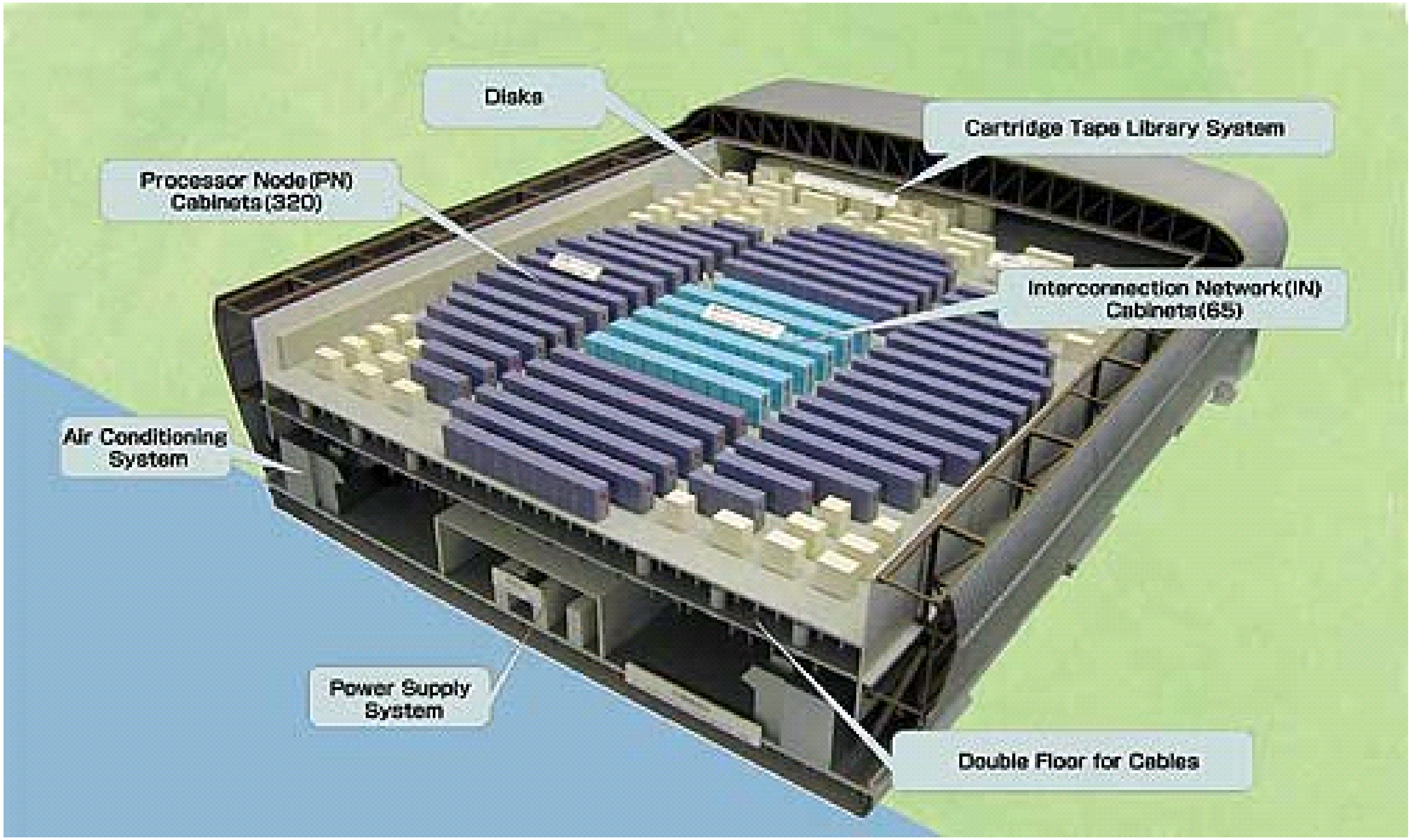
Is FFT Parallelisable ?

- Naive approach to parallelising FFT
  - Divide the data equally among two processors.
  - FFT now involves evaluating bit-reversing which will imply lots of communication between the processors.
  - communication is much slower than FLOPS (or local memory access)
- Lessons :
  - multidimensional FFT will be better parallelisable then 1-d
  - parallelisation of 1-d FFT essentially need new approach.

Why Parallise FFT ?

- Fluid Dyanmics : Turbulence
  Direct Numerical Simulation of Navier-Stokes equation, in Spectral Method with 1024^3 grid points requires about 34 GB of memory.
- Weather Prediction
- Complex Fluids
- The Earth Simulator

The Earth Simulator



The Earth Simulator

- Nodes : 640 processor nodes, each with 8 vector processors.
- Each node has 16 GB shared memory. i.e. Each node is a shared memory machine.
- Communication : 16GB/s full duplex mode.
- Total Peak Performance : 40 TFLOPS which is of course only theoritical.
- computing paradigm : Both shared and distributed memory coding.

2-d Parallel FFT

- Divide the array equally among 2 processors, along the second direction (fortran).
  (word of caution : This is non-standard)
- Do FFT along the local direction.
- Transpose (The time consuming communication step)
- Do FFT along the local direction (again).
- Transpose Back (Optional step)
- Transposing Trick
- 3-d fft paralleises better than 2-d

Parallel FFT (contd)

- FFT of Real data :
  - Do NOT divide along the first direction(fortran)
  - PESSL data storage is really wierd
  - FFTW is much easier to use, but not so fast.
- Bandwidth is more important than latency.
- Does NOT scale well.
- Code must be tuned to the architecture.

Fast Fourier Transform in ES

- Divide the array equally among Processor Nodes(PN), along 3rd direction(fortran 90).
- Use threading (*p*arallel do loops) with 8 arithmatic processors(AP) in each PN, to FFT in 1-d. (Automatic Parallelisation was not effective enough !)
- Use vectorization and microtasking in each AP.
- Peak performance of each AP is 8Gflops. Bandwidth between AP and (shared) memory is 32GB/s, i.e. processor get only one (double precision) number for two flops.
  But in Radix-2 FFT, memory access:flops = 1:1.
  i.e. if Radix-2 FFT is used processor will be kept waiting for data from memory.
- Solution : Use Radix-4 FFT.

Is ES big enough?

- single precision run of 4096^3 DNS. (world record)
- double precision run of 2048^3 DNS. (world record)
- Is this persual of huge size physically meaningfuly? YES
- Even world record simulations gives Reynolds Number as small as 800, whereas experimental data gives about 10 times larger.
- **The Earth (simulator) is Not Enough**

Spectral Method:Burgers Equation

$$\partial_t u(x,t) = -u\partial_x u(x,t) + \nu\partial_x^2 u(x,t)$$

- 1 dimensional, nonlinear PDE
- periodic boundary condition
- time stepping not spectral (obvious)
- Evaluate Derivatives in Fourier Space
- Products in Real Space.

Spectral Method : Virtues

- Best spatial derivative possible. Bettern than any finite difference.
- Quite fast (thanks to FFT)
- Often you are interested in fourier space quantities for physical reasons. (e.g. energy spectrum)

Spectral Method : Vices

- Difficult to work with anything other than periodic boundary condition.
- FFT is not very parallelisable.

Conclusions

- Parallel FFT holds the key to huge simulations in almost any field of research.
- Performance of Parallel FFT depends crucially on tuning the code to the architecture of the parallel machines.
- FFTW seems to be the best choice for single node FFT.
- Even the Earth Simulator is not the limit.
- The story of beowulf.

BEOWULF

Acknowledgements

- CSIR, India
- Indo-French Centre of Advanced Scientific Research.
- Rahul Pandit, Dept of Physics, IISc
- Chinmay Das,Pinaki Chaudhuri.
- Jasjeet Singh Bagla, MRI, Allahabad
- Yanik Ponty, Observatorie of Nice, France.