Leveraging Optimized FFT on Intel Platforms

Jeongnim Kim, DCG/E&G/**H**PC **E**cosystem and **A**pplications **T**eam
SSG/DPD/MKL team
May/30/2018

Legal Disclaimers

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

A "Mission Critical Application" is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL'S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS' FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL PRODUCT OR ANY OF ITS PARTS.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.
Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: http://www.intel.com/design/literature.htm

Knights Landing and other code names featured are used internally within Intel to identify products that are in development and not yet publicly announced for release. Customers, licensees and other third parties are not authorized by Intel to use code names in advertising, promotion or marketing of any product or services and any such use of Intel's internal code names is at the sole risk of the user

Intel, Look Inside, Xeon, Intel Xeon Phi, Pentium, Cilk, VTune and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.

*Other names and brands may be claimed as the property of others.

Legal Disclaimers

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel.

Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

Legal Disclaimers

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark* and MobileMark*, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more information go to http://www.intel.com/performance.

Intel® Advanced Vector Extensions (Intel® AVX)* provides higher throughput to certain processor operations. Due to varying processor power characteristics, utilizing AVX instructions may cause a) some parts to operate at less than the rated frequency and b) some parts with Intel® Turbo Boost Technology 2.0 to not achieve any or maximum turbo frequencies. Performance varies depending on hardware, software, and system configuration and you can learn more athttp://www.intel.com/go/turbo.

Estimated Results Benchmark Disclaimer:
Results have been estimated based on internal Intel analysis and are provided for informational purposes only. Any difference in system hardware or software design or configuration may affect actual performance.

Software Source Code Disclaimer:
Any software source code reprinted in this document is furnished under a software license and may only be used or copied in accordance with the terms of that license.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Legal Disclaimers

The above statements and any others in this document that refer to plans and expectations for the third quarter, the year and the future are forward-looking statements that involve a number of risks and uncertainties. Words such as “anticipates,” “expects,” “intends,” “plans,” “believes,” “seeks,” “estimates,” “may,” “will,” “should” and their variations identify forward-looking statements. Statements that refer to or are based on projections, uncertain events or assumptions also identify forward-looking statements. Many factors could affect Intel’s actual results, and variances from Intel’s current expectations regarding such factors could cause actual results to differ materially from those expressed in these forward-looking statements. Intel presently considers the following to be the important factors that could cause actual results to differ materially from the company’s expectations. Demand could be different from Intel's expectations due to factors including changes in business and economic conditions; customer acceptance of Intel’s and competitors’ products; supply constraints and other disruptions affecting customers; changes in customer order patterns including order cancellations; and changes in the level of inventory at customers. Uncertainty in global economic and financial conditions poses a risk that consumers and businesses may defer purchases in response to negative financial events, which could negatively affect product demand and other related matters. Intel operates in intensely competitive industries that are characterized by a high percentage of costs that are fixed or difficult to reduce in the short term and product demand that is highly variable and difficult to forecast. Revenue and the gross margin percentage are affected by the timing of Intel product introductions and the demand for and market acceptance of Intel's products; actions taken by Intel's competitors, including product offerings and introductions, marketing programs and pricing pressures and Intel’s response to such actions; and Intel’s ability to respond quickly to technological developments and to incorporate new features into its products. The gross margin percentage could vary significantly from expectations based on capacity utilization; variations in inventory valuation, including variations related to the timing of qualifying products for sale; changes in revenue levels; segment product mix; the timing and execution of the manufacturing ramp and associated costs; start-up costs; excess or obsolete inventory; changes in unit costs; defects or disruptions in the supply of materials or resources; product manufacturing quality/yields; and impairments of long-lived assets, including manufacturing, assembly/test and intangible assets. Intel's results could be affected by adverse economic, social, political and physical/infrastructure conditions in countries where Intel, its customers or its suppliers operate, including military conflict and other security risks, natural disasters, infrastructure disruptions, health concerns and fluctuations in currency exchange rates. Expenses, particularly certain marketing and compensation expenses, as well as restructuring and asset impairment charges, vary depending on the level of demand for Intel's products and the level of revenue and profits. Intel’s results could be affected by the timing of closing of acquisitions and divestitures. Intel's results could be affected by adverse effects associated with product defects and errata (deviations from published specifications), and by litigation or regulatory matters involving intellectual property, stockholder, consumer, antitrust, disclosure and other issues, such as the litigation and regulatory matters described in Intel's SEC reports. An unfavorable ruling could include monetary damages or an injunction prohibiting Intel from manufacturing or selling one or more products, precluding particular business practices, impacting Intel’s ability to design its products, or requiring other remedies such as compulsory licensing of intellectual property. A detailed discussion of these and other factors that could affect Intel’s results is included in Intel’s SEC filings, including the company’s most recent reports on Form 10-Q, Form 10-K and earnings release.

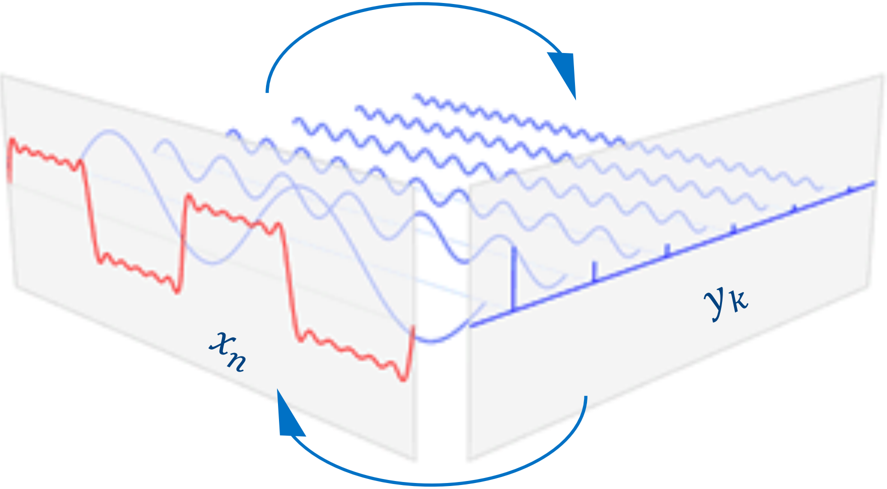
Rev. 7/17/13

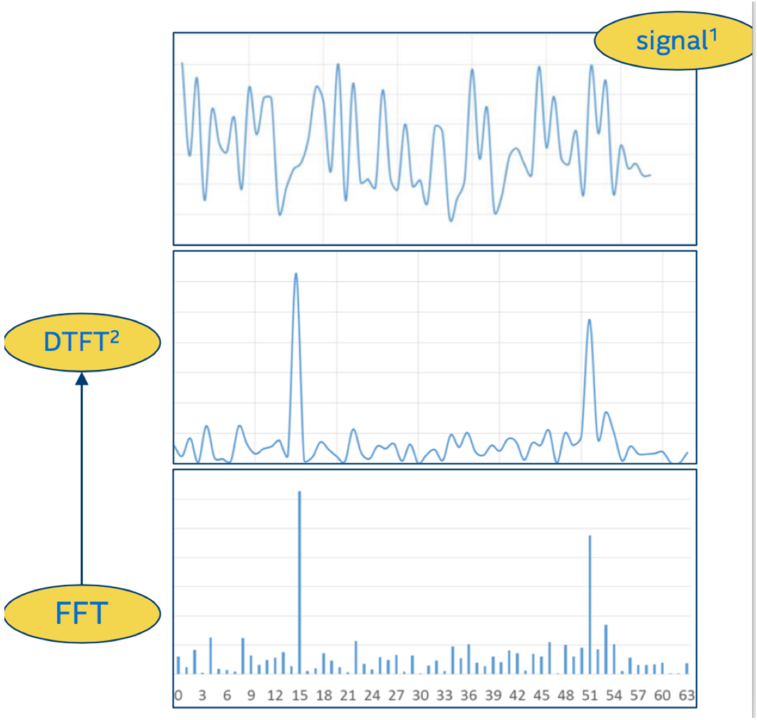
Agenda

- Discrete Fourier Transform in applications
- Fast Fourier Transform (FFT) in a nutshell
- FFTW & MKL DFT
- 3D FFT in applications
- Summary

Discrete Fourier transforms (DFT)

$$y_k = \sum_{n=0}^{N-1} e^{-2\pi i k n/N}\, x_n$$

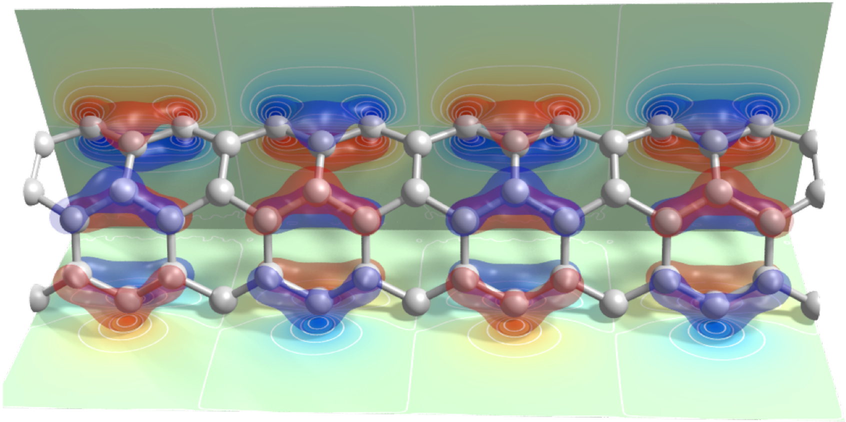


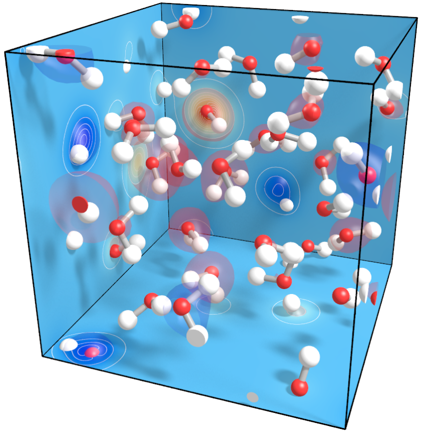


$$x_k = (1/N) \sum_{n=0}^{N-1} e^{+2\pi i k n/N}\, y_n$$

DFT in Applications

- Signal processing, spectroscopy, magnetic resonance imaging, ….
- Speech and image recognition
- Simulations of periodic systems
- Quantum computing





"It's hard to explain what sort of impact Fourier's had," because the Fourier transform is such a fundamental concept that by now, "it's part of the language," Demanet, http://news.mit.edu/2009/explained-fourier

Fourier transform as matrix-vector product: $O(N^2)$

$$x_k = (1/N)\sum_{n=0}^{N-1} e^{+2\pi ikn/N}\, y_n$$

$$\omega = e^{-2\pi i/N} = \cos\frac{2\pi}{N} - i\sin\frac{2\pi}{N}$$

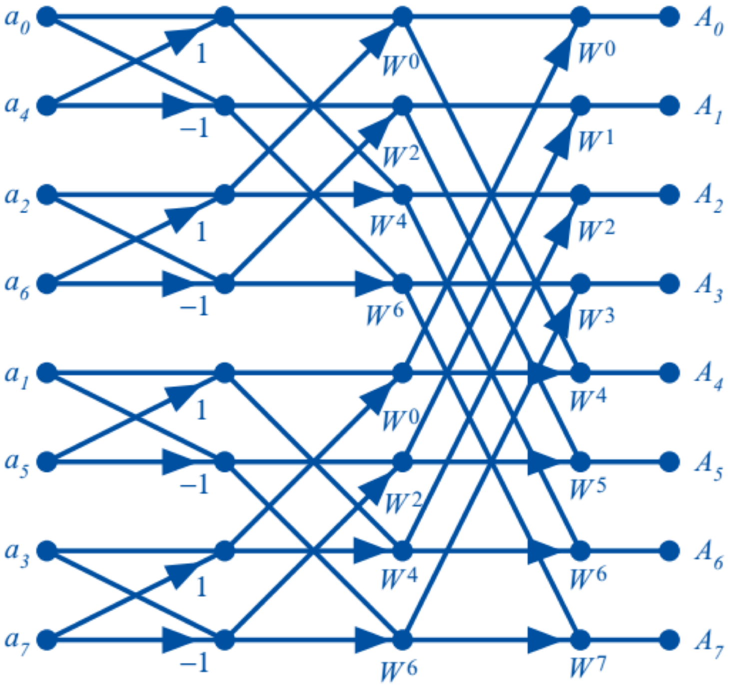
$$y = \left(\quad \omega^{jk} \quad\right) x \quad \Longleftrightarrow \quad y_k = \sum_{n=0}^{N-1} e^{-2\pi ikn/N}\, x_n$$

Fast Fourier Transform (FFT): a fast way to compute DFT

Fast Fourier Transformation (FFT), *O(N log(N))*
The Cooley-Tukey algorithm

8-point FFT

$$FFT8 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega & -i & -i\omega & -1 & -\omega & i & i\omega \\ 1 & -i & -1 & i & 1 & -i & -1 & i \\ 1 & -i\omega & i & \omega & -1 & i\omega & -i & -\omega \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & -\omega & -i & i\omega & -1 & \omega & i & -i\omega \\ 1 & i & -1 & -i & 1 & i & -1 & -i \\ 1 & i\omega & i & -\omega & -1 & -i\omega & -i & \omega \end{bmatrix}$$



$$W = \omega = e^{-2\pi i/8} = (1-i)/\sqrt{2}$$

Twiddle factors: $W^0, W^1, \ldots, W^7$

Typical Ways to Compute FFT

Cooley-Tukey

- Recursively split the FFT into smaller equal sizes

Split-Radix

- Radix 4:2:2

Stockham

- Eliminates the need for rearranging the inputs/outputs that is specific to Cooley-Tukey

Prime-factor

- Decompose into **relatively** prime numbers

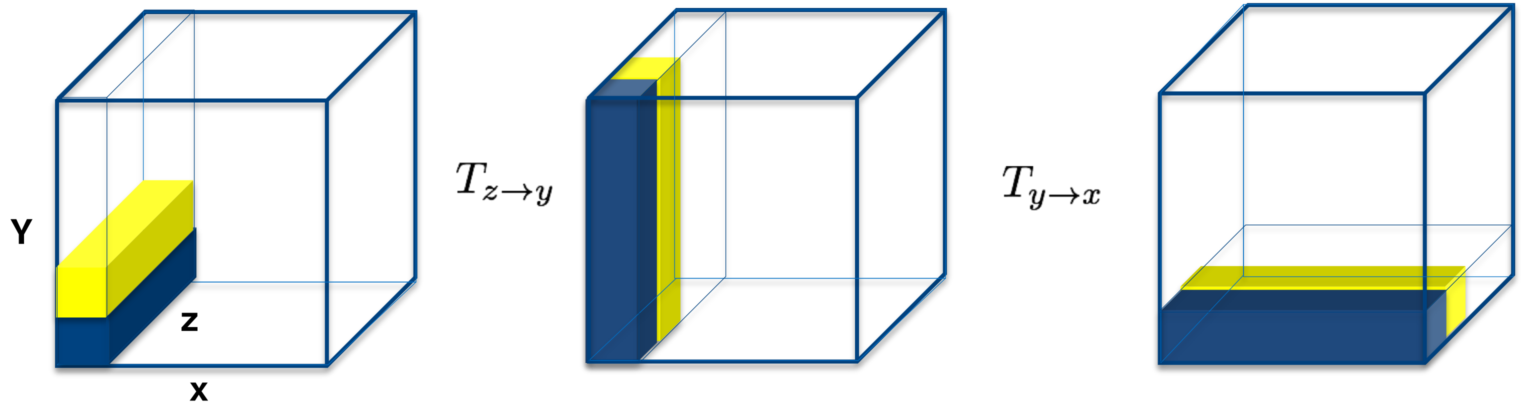
Bluestein (Chirp-Z)

- For arbitrary sizes, uses Cooley-Tukey and convolution theorem

many others…

Basic 3D FFT using stride-1 1D FFT

$$\hat{f}(k_x, k_y, k_z) = \sum_x \sum_y \sum_z e^{i2\pi(k_x x + k_y y + k_z z)} f(x, y, z) = \sum_x e^{i2\pi k_x x} \left[\sum_y e^{i2\pi k_y y} \left[\sum_z e^{i2\pi k_z z} f(x, y, z) \right] \right]$$



- Optimized 1D FFT on stride-1 for each direction
- Transpose to transform the data for 1D FFT

We have libraries for FFT

- MKL-FFT, FFTW …
- Highly optimized 1D FFT $2^p 3^q 5^r \cdots P^z$
- Optimized N-dim FFT and transposes
- Building blocks for DIY FFT

How to use FFT libraries to maximize the productivity!

FFTW and MKL DFT

- Both implement 1 to multi-dimensional FFT
  - Complex-to-Complex
  - Complex-to-Real and Real-to-Complex
  - Double & single precision
  - Threaded on SMP nodes and distributed FFT on multi SMP nodes
- FFTW www.fftw.org
  - Introduced FFT plan concepts
  - Novel code-generation and runtime self-optimization techniques
- Intel® MKL Fast Fourier Transforms (next)
  - Highly optimized on Intel platforms
  - Provide FFTW Wrapper

Intel® MKL Fast Fourier Transforms (FFTs)

FFTW Interfaces support	C, C++ and FORTRAN source code wrappers provided for FFTW2 and FFTW3. FFTW3 wrappers are already built into the library
Cluster FFT	Perform Fast Fourier Transforms on a cluster Interface similar to DFTI Multiple MPIs supported
Parallelization	Thread safe with automatic thread selection
Storage Formats	Multiple storage formats such as CCS, PACK and Perm supported
Batch support	Perform multiple transforms in a single call
Additional Features	Perform FFTs on partial images Padding added for better performance Transform combined with transposition mixed-language usage supported

Intel® MKL 2018 New FFT Features & Optimizations

- MKL_VERBOSE support for FFT starting from MKL 2018 Update 1

  set MKL_VERBOSE=1. The verbose information in the program will be shown up:

  MKL_VERBOSE Intel(R) MKL 2018.0 Update 1 Product build 20170712 for Intel(R) 64 architecture Intel(R) Advanced Vector Extensions 2 (Intel(R) AVX2) enabled processors, Lnx 2.30GHz gnu_thread NMICDev:0
  MKL_VERBOSE FFT: MAIN_DESC | sro4:5:3x5:1:1 | THR_LIMIT = 1 | 0.00s CNR:OFF Dyn:1 FastMM:1 TID:0 NThr:36 WDiv:HOST:+0.000

- Significantly improved 1D and 3D FFT performance for Intel® Xeon® Processor supporting Intel® Advanced Vector Extensions 512 (Intel® AVX-512)

  https://software.intel.com/en-us/mkl/features/benchmarks

Typical FFT use in applications

FFTW APIs

```
void do_fftw(){

  const int N=3;
  int ngrid[]={64,64,64};
  const int howmany=4;
  const size_t distance=64*64*64;
  fftw_complex in[howmany*distance];
  fftw_complex out[howmany*distance];

  fftw_plan for_plan=fftw_plan_many_dft(N,ngrid,howmany,
      in,ngrid,1,distance,out,ngrid,1,distance,
      FFTW_FORWARD, FFTW_MEASURE);

  fftw_plan back_plan=fftw_plan_many_dft(N,ngrid,howmany,
      out,ngrid,1,distance,in,ngrid,1,distance,
      FFTW_BACKWARD, FFTW_MEASURE);

  //FFT
  fftw_execute_dft(for_plan,in,out);

  //inverse FFT
  fftw_execute_dft(back_plan,out,in);

  fftw_destroy_plan(back_plan);
  fftw_destroy_plan(for_plan);
}
```

1. Create plans for forward FFT and inverse FFT
   - Require addresses of In/Out
   - Select (FFTW_ESTIMATE, FFTW_MEASURE, FFTW_ PATIENT,…)
   - Alternative APIs
2. Execute Forward FFT
3. Execute backward (Inverse) FFT
4. Cleanup

Using MKL DFT APIs

```
void do_mkl_dfti(){

  const int N=3;
  int ngrid[]={64,64,64};
  const int howmany=4;
  const size_t distance=64*64*64;
  _MKL_Complex16 in[howmany*distance];
  _MKL_Complex16 out[howmany*distance];

  DFTI_DESCRIPTOR_HANDLE mkl_plan;

  MKL_LONG status = DftiCreateDescriptor(&mkl_plan,
      DFTI_DOUBLE,DFTI_COMPLEX,N,ngrid);
  status=DftiSetValue(mkl_plan,DFTI_NUMBER_OF_TRANSFORMS,howmany);
  status=DftiSetValue(mkl_plan,DFTI_INPUT_DISTANCE,distance);
  DftiCommitDescriptor(mkl_plan);

  //FFT
  DftiComputeForward(mkl_plan,in,out);

  //inverse FFT
  DftiComputeBackward(mkl_plan,out,in);

  DftiFreeDescriptor(&mkl_plan);

}
```

1. Create a descriptor
2. Set configuration parameters (optional)
3. Commit a descriptor
4. Execute Forward FFT
5. Execute backward (Inverse) FFT
6. Cleanup

FFTW Wrapper in MKL

- No code modification necessary.
- Include the header file: `-I${MKLROOT}/include/fftw`
- Link MKL as usual, e.g., -mkl

Mapping between FFTW and Intel MKL Interfaces

- https://software.intel.com/en-us/articles/the-intel-math-kernel-library-and-its-fast-fourier-transform-routines

3D FFT in electronic structure codes

PW DFT on Parallel Computers

Clarke, Stich & Payne, CPC, 72, 14 (1992)

Computer Physics Communications 72 (1992) 14–28
North-Holland

Computer Physics Communications

Large-scale ab initio total energy calculations on parallel computers

L.J. Clarke
Edinburgh Parallel Computer Centre, University of Edinburgh, Mayfield Road, Edinburgh, EH9 3JZ, UK

I. Štich [1] and M.C. Payne
Cavendish Laboratory (TCM), University of Cambridge, Madingley Road, Cambridge, CB3 0HE, UK

Received 30 March 1992

An implementation of a set of total energy plane-wave pseudopotential codes on a parallel computer is described which allows calculations to be performed for systems containing many hundreds of atoms in the unit cell. Possible parallelisation strategies are discussed and it is shown that assigning parts of real and Fourier space across the processors is the least restricted approach. The performance of our parallel codes is demonstrated by timing tests carried out on several i860-based parallel machines and these are compared with tests performed on conventional sequential supercomputers. Ab initio computations on systems which are beyond the power of conventional supercomputers as well as new perspectives for first-principles molecular dynamics are discussed.

Abinit
BigDFT
CASTEP
CPMD
GPAW
NWChem-PW
Qbox
Quantum Espresso
VASP
……

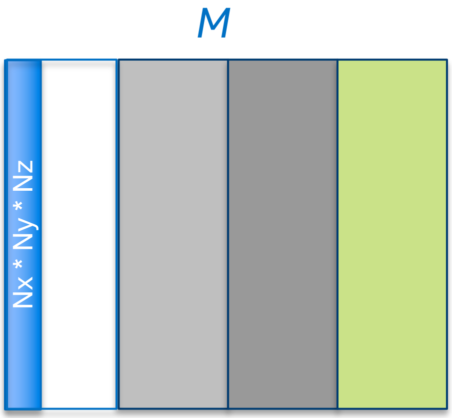
Algorithm 8: A pseudo PW KS-DFT code

```
C(:,:) = random
Orthonormalize all the columns:  C^H C=I  // Gram-Scmidth

for step=1,...,Nstep
{
  //compute density
  rhoR=0                              //memet
  for n=1,..,N
    Cr(:,n)=iFFT(C(:,n))              //iFFT
    rhoR(:) += f*conj(Cr(:,n))*Cr(:,n) //DOT
  endfor
  rhoG=FFT(rhoR)                      //FFT

  rhoG_old=rhoG

  do {

    B=0                          //memset
    for n=1,...,M
      B(:,n) += Gsq(:) C(:n)     //DOT
      B(:,n) += Vh(:) C(:,n)     //DOT

      Vtmp(:)=Vxc(rhoR(:))*Cr(:,n) //interpolation,DOT
      B(:,n) += FFT(Vtmp)(:)     //FFT

      B(:,n) += Vloc+Vnl         //small GEMM
    endfor

    E=C^H B                      //zgemm
    Diagonalize(E)               //zheev family
    Update C <- EC               //zgemm

    Compute rhoR and rhoG        //DOT,FFT

  } while(||rhoG - rhoG_old||>eps)

  Compute forces on ions: C^H F C
  Move ions
}
```

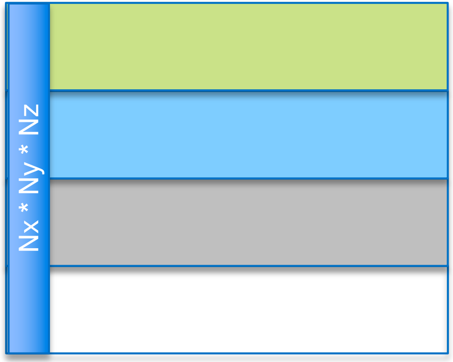
C(Ng, N), the solution
Ng=Nx * Ny * Nz, FFT grid
N= number of KS orbials

Data parallelization (FORTRAN convention)

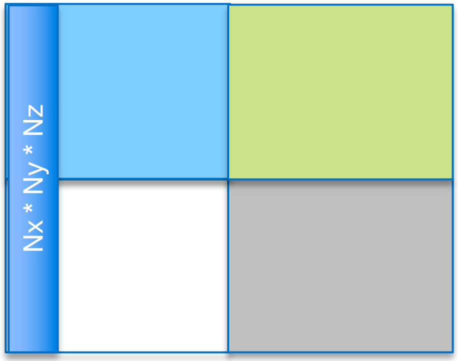
Band distribution
- Distribute M
- FFT within a task; exploit optimized FFT library
- Choice for HF, GW & RPA



FFT distribution
- Distribute Ng
- Cluster FFT
- Choice for DFT in many codes

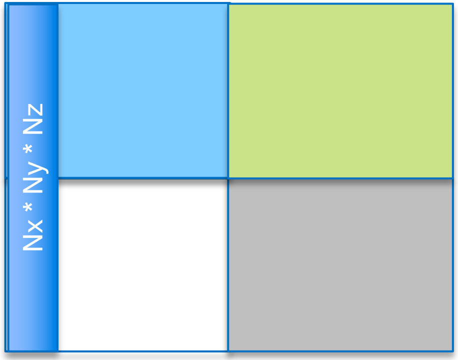


Mixed
- FFT groups
- Band groups



Colors=MPI tasks

Band & FFT distribution



Distribute M and Ng
Distribute FFT
Parallel GEMM (aka Scalapack)

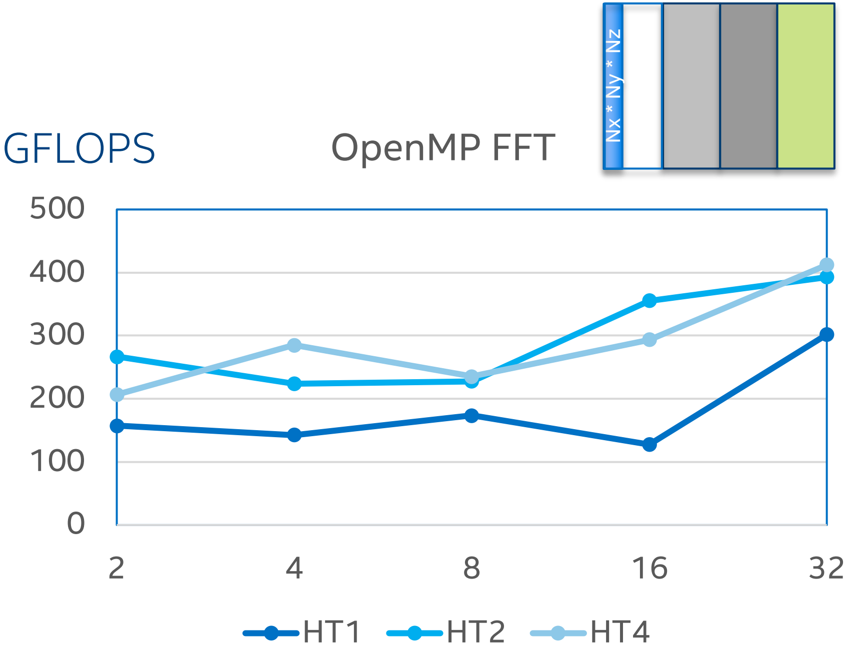
- Implemented in VASP, Qbox, NWCHEM-PW
- Necessary for small-memory systems (e.g., Blue Gene L/P/Q): Qbox 2006 Gordon Bell
- Maybe needed for big problems
- May provide the best compromise between FFT and GEMM efficiency
- Can leverage **TSGEMM** and **TSQR**

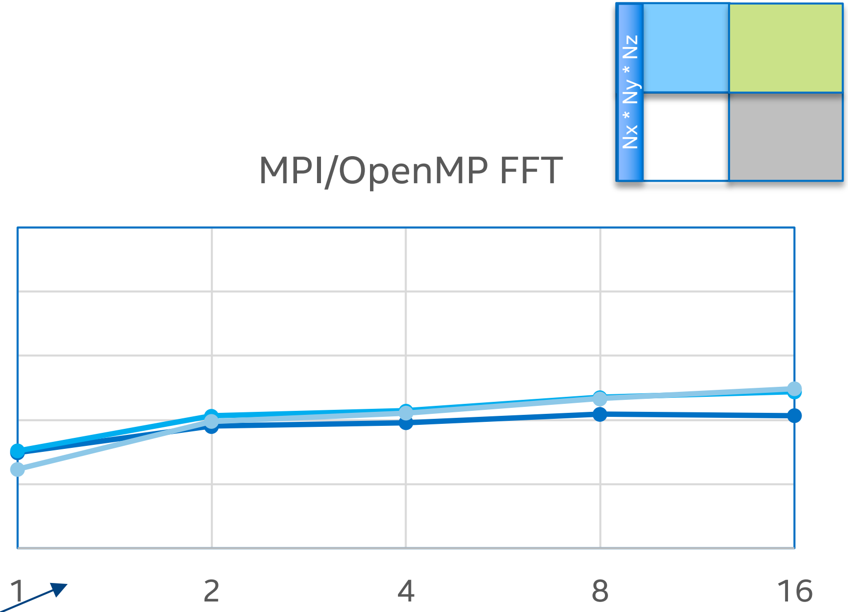
Optimizing 3D FFT use

- Many concurrent FFTs of *Nx*Ny*Nz*
  - M=10-1000
  - FFT grid Nx=10-200 for typical problems of today
- Memory bandwidth or network bandwidth limited
- Cache blocking, SIMD and overlap of the data movement and computation: critical for high performance
- Leverage “many (batched)” FFT
- Moving data efficiently via optimized transpose
- Overlap with other computations

FFT 3D on SMP: OpenMP vs MPI

GLOPS=Theoretical Cooley-Tukey



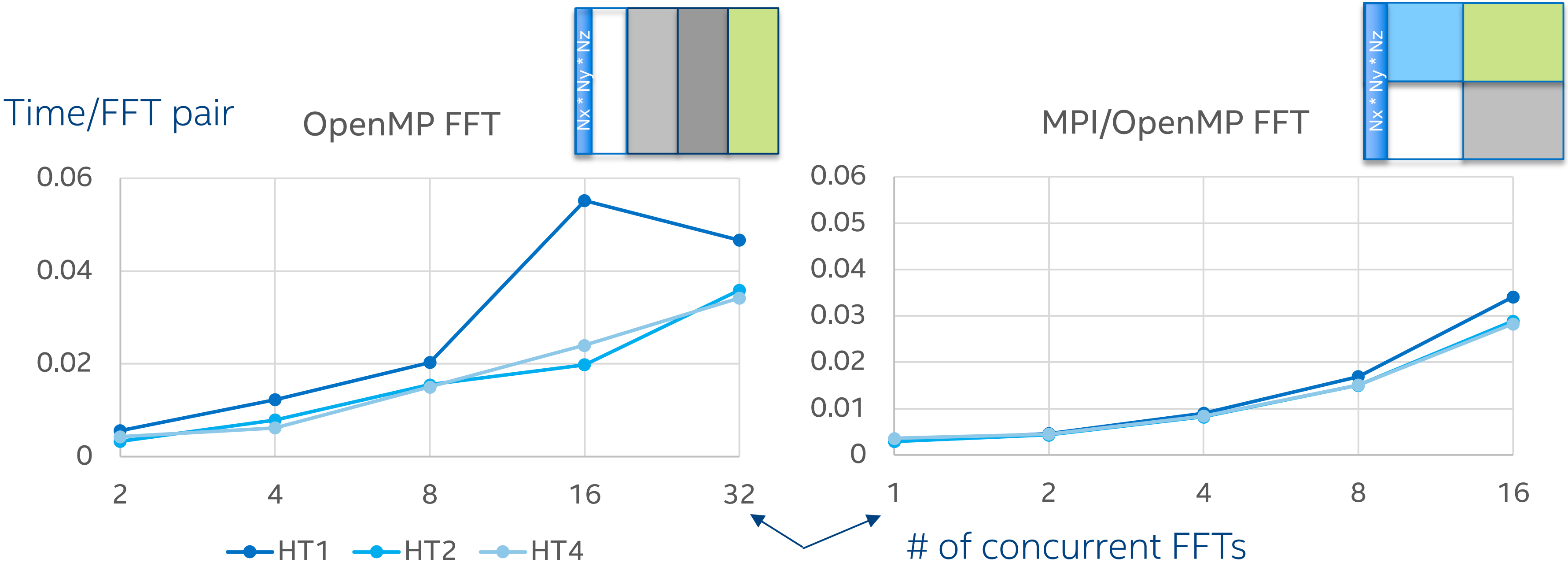


# of concurrent FFTs

- Using 64 cores of 72-core Intel(R) Xeon Phi(TM) CPU 7250 @ 1.40GHz
- I_MPI_PIN_DOMAIN= 64*4/MPI
- KMP_AFFINITY=(scatter,balanced,compact), granularity=fine

FFT 3D on SMP: OpenMP vs MPI

$128 \times 128 \times 128$

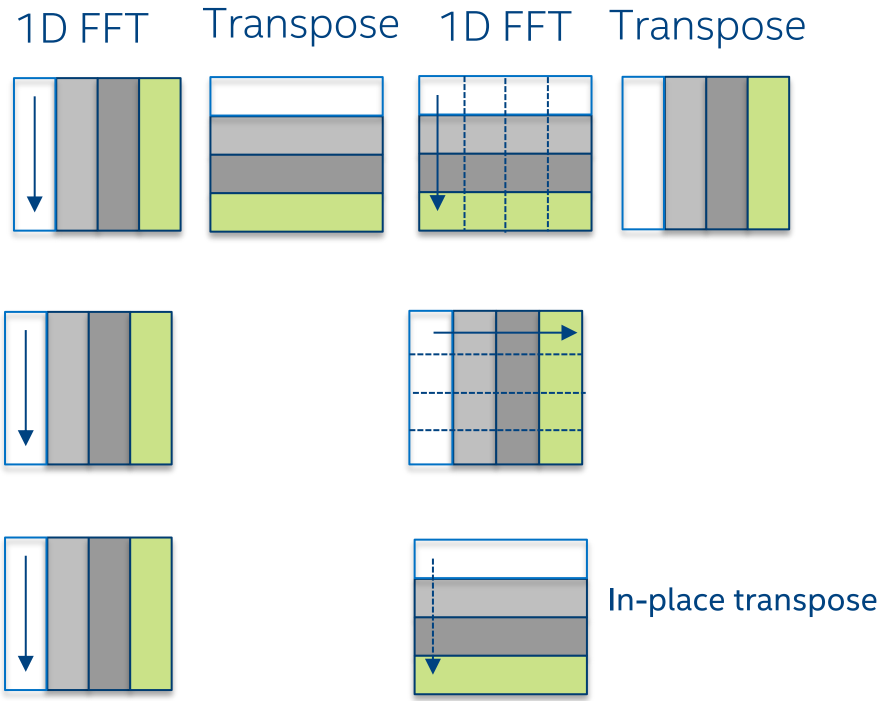


- Optimize either throughputs (how many FFT/s) or time (strong scaling)
- Using 64 cores of 72-core Intel(R) Xeon Phi(TM) CPU 7250 @ 1.40GHz
- I_MPI_PIN_DOMAIN= 64*4/MPI; KMP_AFFINITY=(scatter,balanced,compact), granularity=fine

Application-tailored FFT

- Data structure and distribution set by the entire application
- Can interleave FFT and other computations
- **Can reduce data movement**
- Can overlap communications/computations

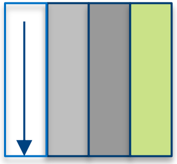
2D FFT example



Using MKL FFT configuration parameters for customized FFT: composed 3D FFT=2D + 1D

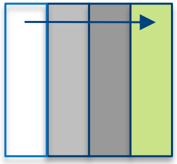
Create 2D FFT plan

```
MKL_LONG ngrid_2d[]={ngrid[1],ngrid[2]};

DFTI_DESCRIPTOR_HANDLE plan_yz, plan_x;
MKL_LONG status = DftiCreateDescriptor(&plan_yz,dfti_get_precision(real_type()),DFTI_COMPLEX,2,ngrid_2d);
status=DftiSetValue(plan_yz,DFTI_NUMBER_OF_TRANSFORMS,ngrid[0]);
status=DftiSetValue(plan_yz,DFTI_INPUT_DISTANCE,ngrid[1]*ngrid[2]);
DftiCommitDescriptor(plan_yz);
```



Create 1D FFT plan

```
MKL_LONG strides[2]={0,ngrid[1]*ngrid[2]};
status=DftiCreateDescriptor(&plan_x,dfti_get_precision(real_type()),DFTI_COMPLEX,1,ngrid[0]);
status=DftiSetValue(plan_x,DFTI_NUMBER_OF_TRANSFORMS,ngrid[1]*ngrid[2]);
status=DftiSetValue(plan_x,DFTI_INPUT_DISTANCE,1);
status=DftiSetValue(plan_x,DFTI_OUTPUT_DISTANCE,1);
status=DftiSetValue(plan_x,DFTI_INPUT_STRIDES,strides);
status=DftiSetValue(plan_x,DFTI_OUTPUT_STRIDES,strides);
DftiCommitDescriptor(plan_x);
```



Forward 3D FFT

```
DftiComputeForward(plan_yz,g[0]);
DftiComputeForward(plan_x,g[0]);
```

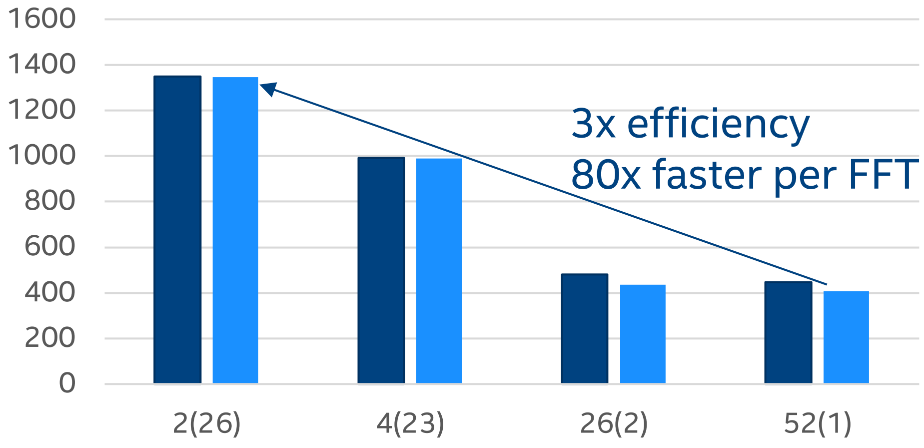
Backward 3D FFT

```
DftiComputeBackward(plan_x,g[0]);
DftiComputeBackward(plan_yz,g[0]);
```

https://software.intel.com/en-us/mkl-developer-reference-c-configuration-settings

Performance of “many” 3D FFT of 128^3

GFLOPS, higher the better



# of FFT (threads)

# of FFT = # of MPI ranks
Theoretical Cooley-Tukey GFLOPS

Intel® Xeon® Platinum 8170 CPU, 2S, 26C/S

Reducing BW needs

- Multiple threads for each FFT : SMP optimization and no MPI communications
- Cache/register optimization via in-place, on-the-fly transpose and SIMD optimization in MKL
- A FFT is 80 times faster using Composed methods using 26 threads than the baseline using 1 thread.

Summary

- Overview of FFT capabilities in MKL
- Considerations to maximize FFT performance in applications
  - Allocation
  - Computation
  - Data movement
- Adapt FFT use for application-specific data structures and algorithms based on FFT primitives

intel®