BLUE WATERS

SUSTAINED PETASCALE COMPUTING

November 23, 2010

Toward petaflop 3D FFT on clusters of SMPs

Jeongnim Kim
NCSA

GREAT LAKES CONSORTIUM FOR PETASCALE COMPUTATION

Petaflop parallel 3D FFT, why?

NSF petascale turbulence benchmark required petaflop performance of 3D FFT of 12288^3 on BW

- Demands 10% of BW's peak including communication

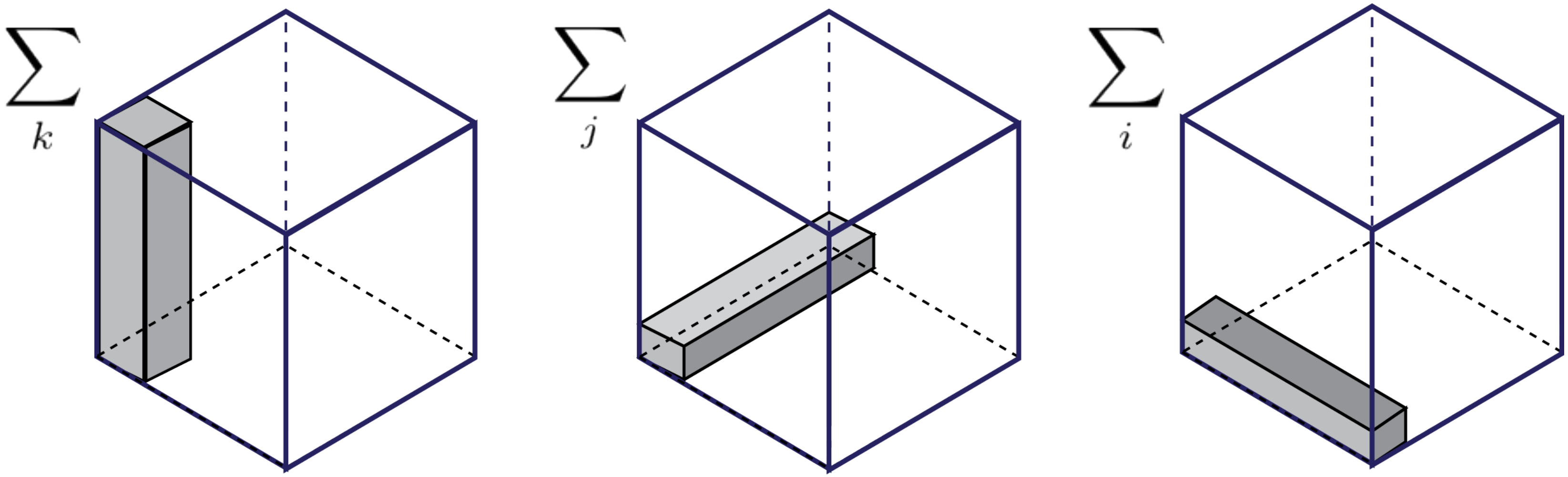
Disclaimers

- The results on P7 were obtained on early hardware and software and do not represent the true performance of BW; the conclusions are subject to change.
- All the test codes are available upon request and distributed under UIUC/NCSA open-source (a BSD) license.
- Opinions are solely mine and not NCSA's.

Outline

- 3D FFT, basic algorithm and use cases
- Slab distribution on clusters of SMPs
- Performance Analysis of 2D FFT on P7
  - PDCFT2 in PESSL, MPI
  - DCFT2 in ESSLSMP using OpenMP
  - OpenMP/TLS: transpose with get operation (TLS=thread-local storage)
  - OpenMP/Mix: no transpose
- Further improvements
- Requirements of 3D FFT library

3D FFT $u^x_{\mathbf{k}} \leftarrow \sum_i \sum_j \sum_k \exp^{i\mathbf{k}\cdot\mathbf{r}_{i,j,k}} u^x(\mathbf{r}_{i,j,k})$



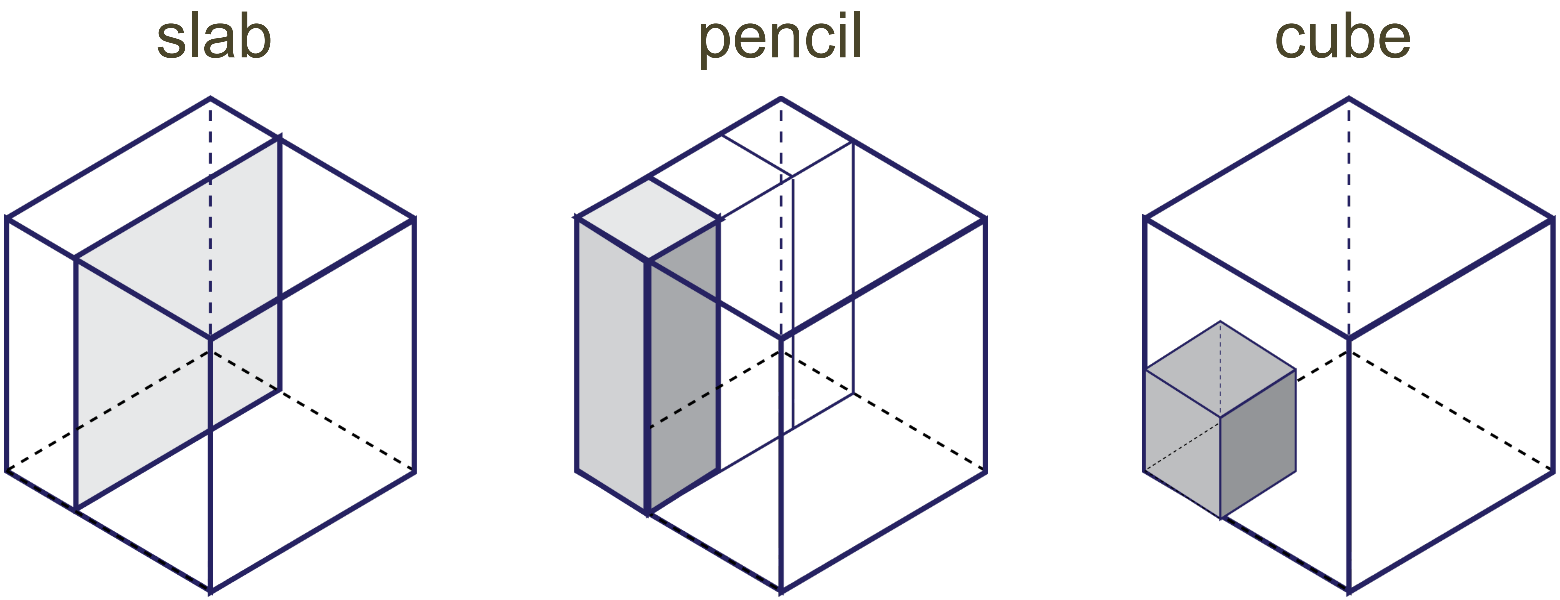
- Commonly using 1D FFT, variants of Cooley–Tukey algorithm
- Applications using 3D FFT
  - Coulomb potential for MD, e.g., NAMD, LAMMPS …
  - CFD: Turbulence with direct numerical solver (DNS)
  - Electronic structure methods, e.g.,Qbox, OpenAtom …

Status of parallel 3D FFT

Parallel 3D FFT libraries including P3DFFT use 1D FFTs and perform transposes. Why?

- 1D FFTs can be and are highly tuned on a target architecture.
  - No special data structure: regular 1D arrays.
- Requirements of applications widely vary and supporting many special cases at **high** efficiency is hard.
  - How many variables to be transformed: 1, few and many?
  - Computations on each domain and their relative operations counts dictate how to distribute the data.
    - E.g., real-space computations are much less critical than the rest for turbulence or DFT and are ignored.
- HPC architectures are evolving and so are the optimal data distribution for a target problem: BG, XT, and PERCS

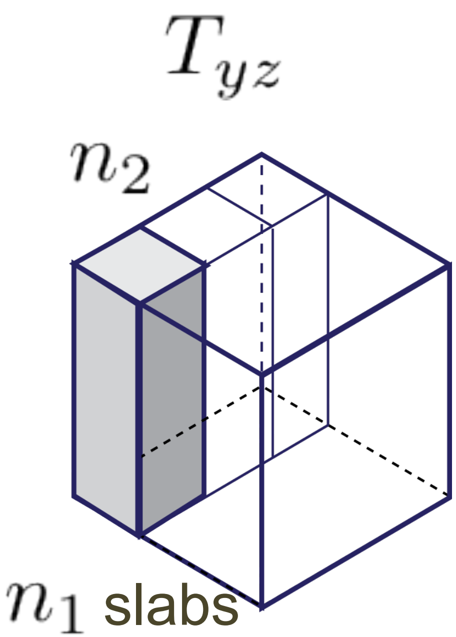
Parallel 3D FFT: data distribution over n_p PE



$n_p < N$ | $N < n_p < N^2$ | $N^2 < n_p$

What is n_p on BW? → more than 300 K cores; → about 10K SMP nodes

Pencil distribution using MPI: P3DFFT*



n_p MPI tasks with $n_p = n_1 n_2$

- Can exploit efficient 1D FFT on *N* elements of stride 1 by FFT libraries, e.g., ESSL, FFTW
- But, need to transpose the pencils twice

$$u(i,j,\underline{k}) \xrightarrow[T_{yz}]{} u(i,k,\underline{j}) \xrightarrow[T_{xy}]{} u(j,k,\underline{i})$$

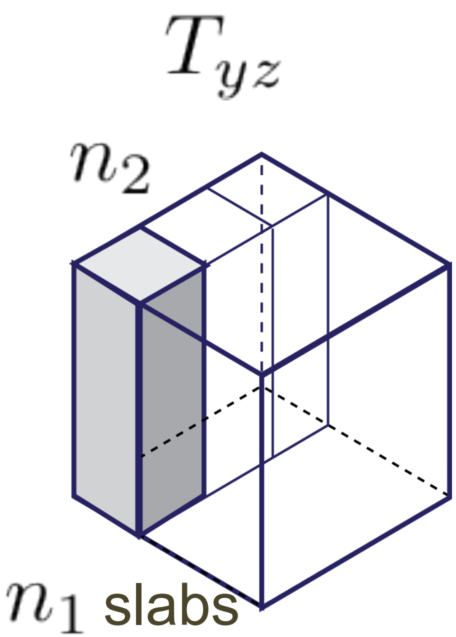
n_1 communicator groups (YZ slabs) of n_2 tasks

n_2 communicator groups (XY slabs) of n_1 tasks

Array syntax in C convention.

* P3DFFT library, http://code.google.com/p/p3dfft/, D. Pekurovsky, SDSC

Optimal distribution for clusters of SMPs: back to slabs



2D FFT on a SMP

$$u(i,j,\underline{k}) \underset{T_{yz}}{\longrightarrow} u(i,k,\underline{j}) \underset{T_{xy}}{\longrightarrow} u(j,k,\underline{i})$$

- BW's SMP node is powerful
  - 32 cores, > 64 GB memory
  - High memory bandwidth
  - A lot of threads : 128=4x32 threads
- 2D FFT on a SMP
  - MPI can be optimized to exploit SMPs
  - OpenMP (any threading) will work

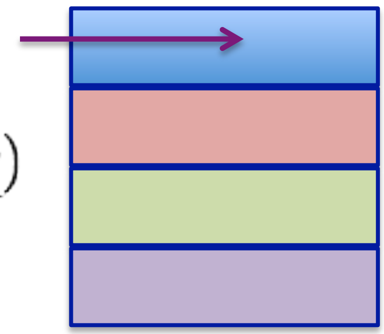
2DFFT on a SMP

- Use optimized 1DFFT: ESSL, FFTW, MKL
  - Multiple 1DFFTs: e.g., guru interface of FFTW
  - 20-50% of peak for N>1000
- On P7, analyze the performance of
  - PESSL: MPI reference implementation
  - ESSLSMP
  - MPI: alltoall implementation
  - OpenMP/TLS
  - OpenMP/Mix

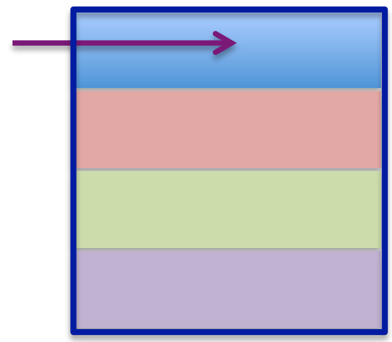
Memory access pattern of 2D FFT: no reordering

Distributed memory, OpenMP/TLS

$FFT(x,\underline{y})$



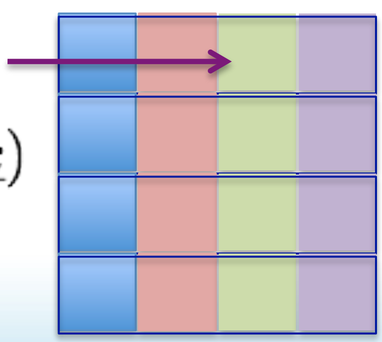
OpenMP/Mix

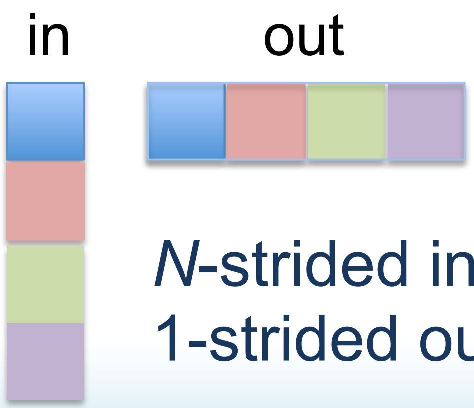


ESSPSMP

Transpose

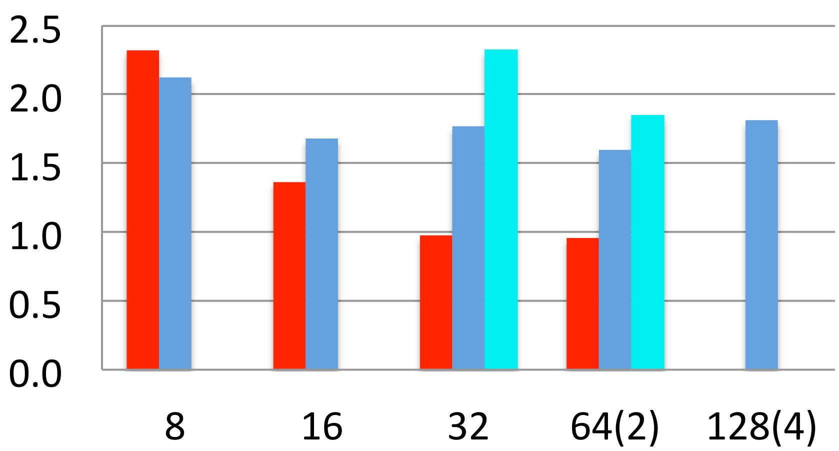
$FFT(y,\underline{x})$



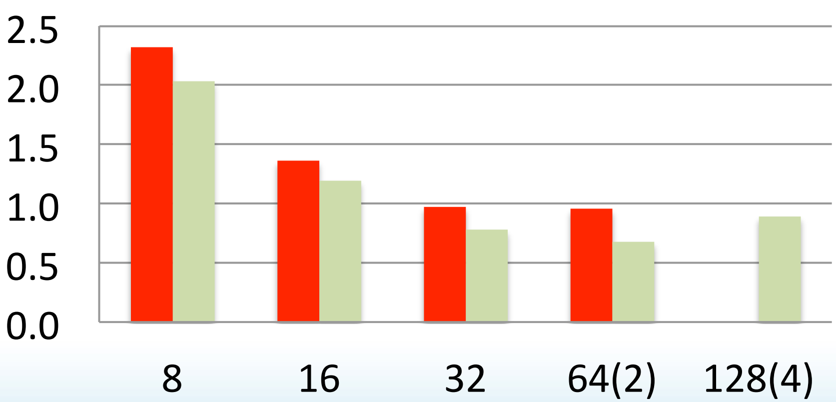


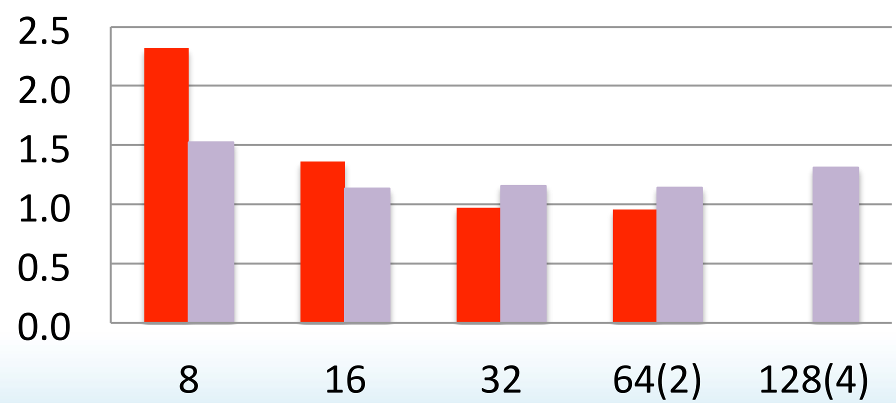
N-strided in,
1-strided out

Timing on BlueDrop: *N*=12288

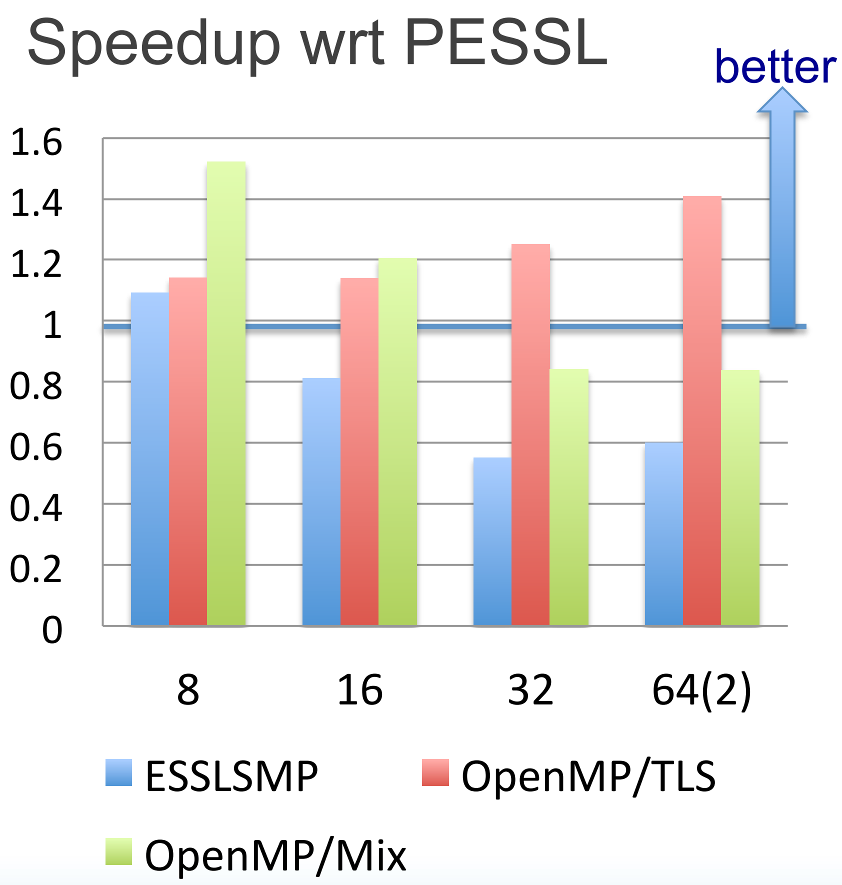


PESSL
ESSLSMP
MPI (not showing 8 and 16)
OpenMP/TLS Best
OpenMP/Mix





Effects of memory hierarchy of BlueDrop



- BD node: not a pure SMP but NUMA with two-level memory hierarchy

8 core – 16 core – 32 core
chip enclosure

- OpenMP/TLS & OpenMP/Mix outperform PESSL on a16-core enclosure
- OpenMP/TLS wins as the threads increase
- Similar trends with N=4096,6144,and 8192

Will OpenMP 2D FFT work on BW?

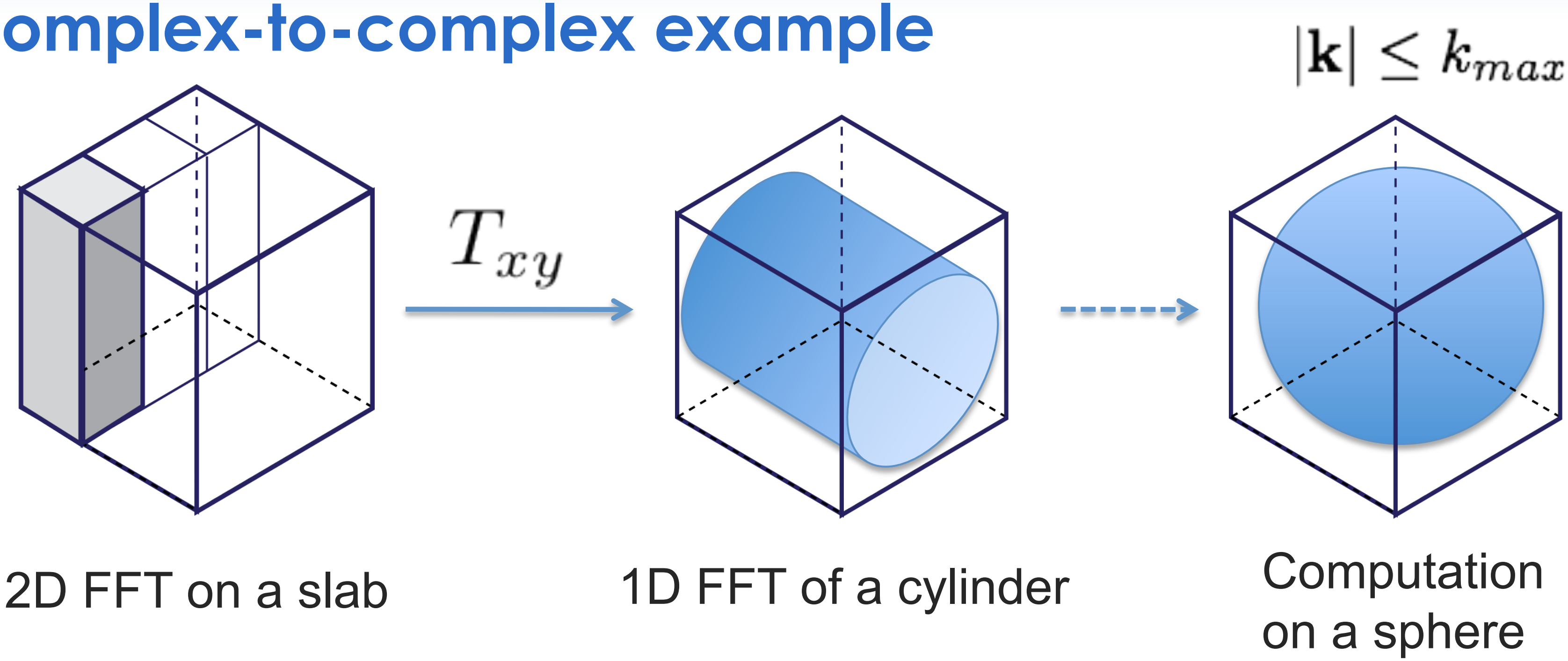
- Parallel execution of *N* 1DFFTs on *N* elements for *N* >> 1
- Some or all explicit memory copies can be eliminated.
  - 4 copies of naïve MPI : one or no copy with OpenMP
  - Possible to eliminate 2 copies with MPI with MPI_Datatype
- Memory and thread locality can be managed: OpenMP/TLS
- SMT 2/4 may further hide memory latency
- Synchronization not necessary

Yes, any threading model that exploits shared memory would work well on BW or clusters of SMPs.

2D FFT on IH-Drawer:
really early results and need NDA

- Nearly flat memory on a P7 node (*almost* perfect SMP)
  - OpenMP methods all outperform PESSL
  - OpenMP/Mix works best *so far*
  - ESSLSMP and OpenMP/TLS similar
  - PESSL scales beyond a node but 64-task on 64 cores is not better than OpenMP implementations on 32 cores
- OpenMP/TLS, most likely the method of choice
  - Can improve transpose using VSX/SSE and thread scheduling
  - Can hide NUMAness
  - Can perform multiple 2D FFTs simultaneously
  - Especially useful as a component of 3DFFT

Improvement of 3D FFT algorithm:
complex-to-complex example



Physics dictates and always de-aliasing imposed in applications

- Depending upon k_{max} , optimal data in the spectral space varies
- Often, real-to-complex and only half of a sphere needed

Common usecases of 3D FFT

Apps	N_v	N	Datatype	Spectral data
NAMD, LAMMPS	1	10^2-10^3	real	Cubic
DNS	~3	> 10^3	real	Cylinder
Qbox, OpenAtom	>> 1	> 10^2	real/complex	Sphere

for i $= 1 \cdots N_v$ **do**
    1D FFT $\mathbf{v}_i$ on z
    transpose on yz slab
    1D FFT $\mathbf{v}_i$ on y
    transpose on xy slab
    1D FFT $\mathbf{v}_i$ on x
end for

for i $= 1 \cdots N_v$ **do**
    2D FFT $\mathbf{v}_i$ on yz
    transpose on xy slab
    1D FFT $\mathbf{v}_i$ on x
end for

Blocking

2D FFT $\mathbf{V}$ on yz
transpose on xy slab
1D FFT $\mathbf{V}$ on x

???

My wish list for a parallel 3D FFT library

- Hybrid: threads on SMP and MPI-like over SMPs
- Allow maximum overlap
  - Non-blocking alltoall(v) on a team of SMPs
  - One-sided get from *M*-strided to *1*-strided data
- Portable: use optimized 1D FFT and maybe 2D FFT
- Allocators!!!
- Efficient APIs to access the data and memory layout
- Learn from FFTW and MKL and provide
  - Basic and “guru” APIs
  - APIs to tune the run-time variables, BUT auto-tuning and compile-time optimization much preferred

Conclusions

- 3D FFT in the era of clusters of SMPs
  - Back to slabs
  - Exploit shared memory and threads
- But, global communication (a.k.a. alltoall) ultimately limits the performance.
- Other related works
  - Takahashi (previous talk)
  - Chen *et al* (PKUFFT, 3D FFT on clusters of GPUs)
  - …