Auto-Tuning Distributed-Memory 3-Dimensional Fast Fourier Transforms on the Cray XT4

Manisha Gajbe (GATech),

Andrew Canning, John Shalf, Ling Wang, Harvey Wasserman (LBNL),

Richard Vuduc (GATech)

Outline

- Brief Introduction to Parallel 3D FFTs
- Data Layouts and communication structure
- Necessity of Auto tuning
- Motivation for Auto Tuning of 3F FFTs
- Approach for Auto Tuning
- Results
- Future Plans
- Conclusion

Parallel 3D FFT

- At the core of many real-world scientific and engineering applications is the necessity for computing 3D FFT.

- FFT is a very commonly used numerical technique in computational physics, engineering, chemistry, geosciences, and other areas of high performance computing.

Parallel 3D FFT

- Computation : *O(NlogN)*
- Communication : *O(N)*
- The communication costs rapidly overwhelmed the parallel computation savings even for small sizes like 64 x 64 x 64.
- Challenge : for the communications infrastructure of a parallel machine because of the all-to-all nature of the distributed transposes involved in 3D FFT

Typical Parallel 3D FFT today

Grid: N x N x N – how to do parallel FFT?

- Transpose strategy
  - That is, make sure all data in direction of 1D transform resides in one processor's memory.
  - Parallelize over remaining dimensions.
- 1D decomposition
- Each CPU has several planes (slabs)
  - Local geometry: N x N x L, where L=N/P
- Transform in 2 dimensions local to the planes first
  - Use an established library for 1D FFT, e.g. ESSL, FFTW

Typical Parallel 3D FFT today

Grid: N x N x N – how to do parallel FFT?

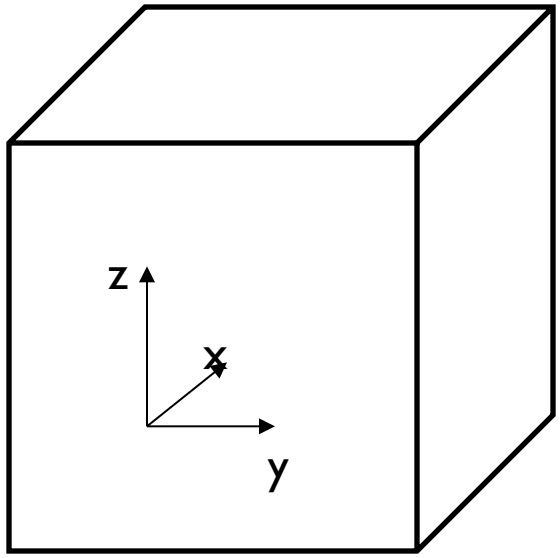
- Transpose data to localize third dimension
  - Local geometry: N x L x N
- Complete transform by operating along third dimension
- Most has been using this approach. Examples:
  - PESSL
  - NAS Parallel Benchmarks

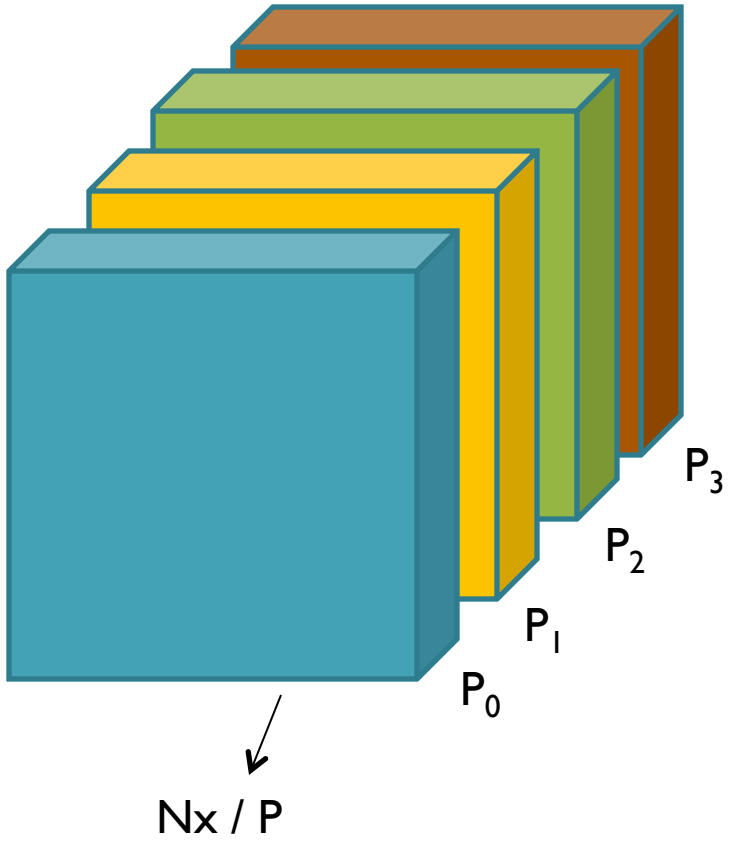
1D decomposition scaling

- Communication: Single call to MPI_Alltoall or equivalent all-to-all exchange.
  - Demanding operation. Performance depends on interconnect bisection bandwidth.
    - Cut the network in two. Count the links cut.
    - This parameter depends on bandwidth of each link, and also network topology (e.g. fat-tree, 3D torus)
- Can scale OK (depending on interconnect), but only up to a point:

  P <= N
- For 512^3 grid, no more than 512 processors can be used.

1D Decomposition





1-Dimensional Layout in X dimension

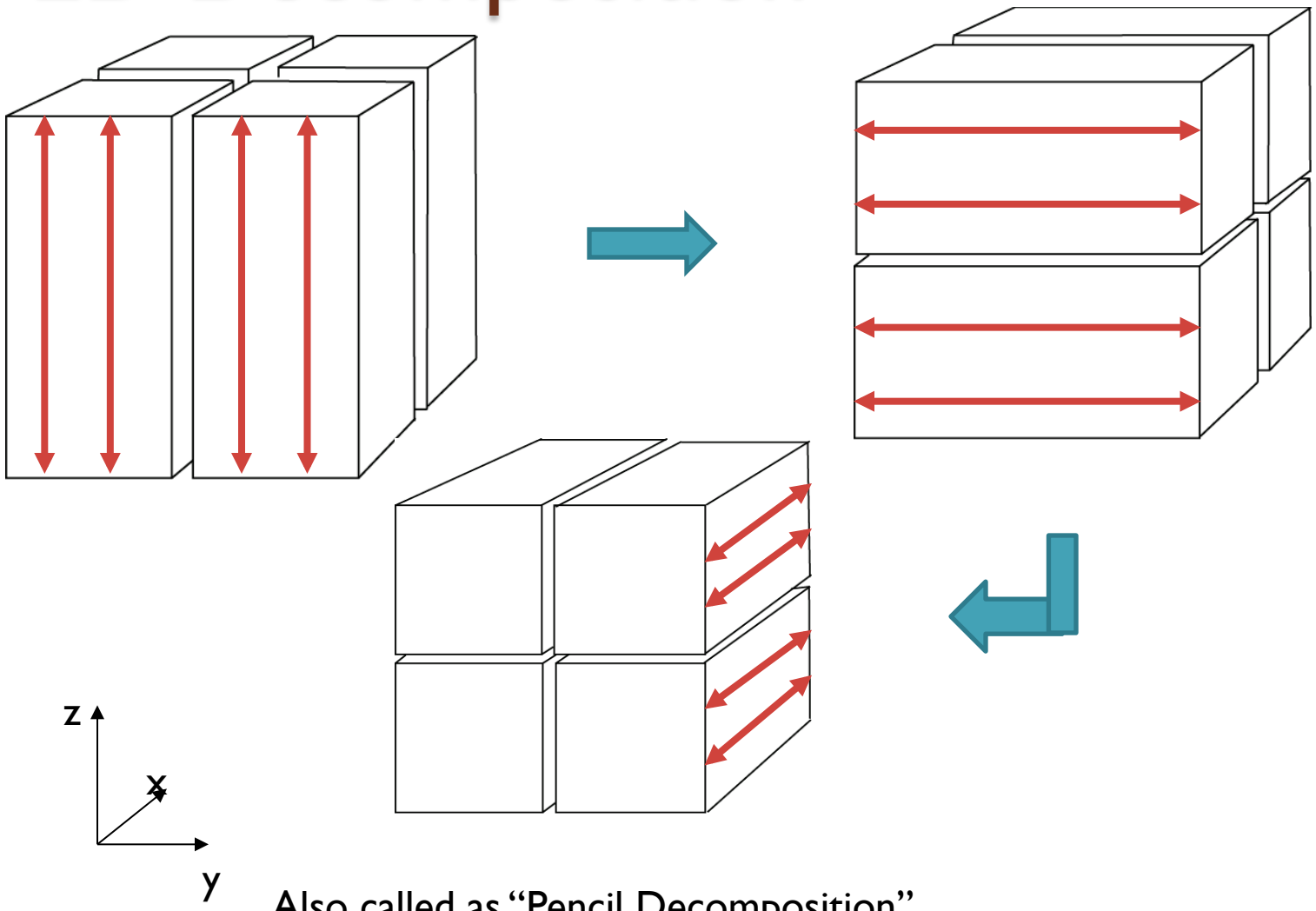
Each processor gets Nx / P "planes" of Nz x Ny elements per plane

1D Decomposition

- Step 1 : 1D FFTs on columns in Z dimension (all elements local)
- Step 2 : 1D FFTs on rows in Y dimension (all elements local)
- Step 3 : 1D FFTs in X dimension (Requires communication)

Also called as "Slab Decomposition"

2D Decomposition



Also called as "Pencil Decomposition"

2 D Decomposition

- Step 1 : 1D FFTs on columns in Z dimension (all elements local)
- Step 2 : Transpose (z,y,x -> x,y,z)(Requires communication)
- Step 3 : 1D FFTs on rows in Y dimension
- Step 4 : Transpose (y,z,x -> z,y,x) (Requires communication)
- Step 5 : 1D FFTs in X dimension

Also called as "Pencil Decomposition"

Scaling

- Scaling Issues (bandwidth and latency):
- computations/communications ~ $N^2 N \log N / N^3 = \log N \sim O(10)$
- message size ~ (num. of processors)$^{-2}$

Hence require to explore as efficient as possible implementation of the communication strategy for the transpose operations.

Necessity of Performance Tuning

- The performance is largely dependent upon a computation kernels such as dense or sparse matrix operations, Fast Fourier Transforms, Stencil computations etc.
- In general these kernels require machine-dependent tuning by hand or using highly optimized machine specific scientific and mathematical libraries and compilers.

Conventional Performance Tuning

- Vendor or user hand tunes kernels
- Drawbacks:
  - Very time consuming and tedious work
  - Even with intimate knowledge of architecture and compiler, performance hard to predict
  - Growing list of kernels to tune
    - Example: New BLAS (Basic Linear Algebra Subroutines) Standard
  - Must be redone for every architecture, compiler
  - Compiler technology often lags architecture
  - Not just a compiler problem:
    - Best algorithm may depend on input, so some tuning at run-time.
    - Not all algorithms semantically or mathematically equivalent

Auto Tuning

- Plays a vital role in optimizing 3D FFT kernels for a specific platform as it can make use of the system configuration
- A number of automatic tuning systems are developed that typically uses –
  - *Selection of an optimal implementation of* kernel
  - *Development of multiple machine specific implementations of kernels*
  - *Use of* optimized system configurations.

Auto Tuning

The best approach can be taken towards auto tuning is –

- Identification of a set of algorithms or methods
- Identify the fastest method by collecting the performance data

Uses -

- Modular design approach : Testing, portability and maintenance of a code
- Easy to focus on the performance of a key subroutine or kernel.
- Tuning these kernels automatically improves performance of an application that uses it.

Parallel 3D FFT

- Parallel 3 D FFT is achieved using highly optimized versions of 1D FFT that are available on most of the modern systems.
- No optimized autotuned3D FFT library or kernel available on today's modern supercomputers.
- Most successful attempt to auto tune FFT are
  - *FFTW*
  - *Spiral*

Motivation for Auto Tuning of Parallel 3D FFT

- Both FFTW and SPIRAL do not address automatic performance tuning of Parallel 3D FFTs.
- FFTW 3D MPI FFT subroutine is not scalable as the maximum number of processors or cores that can be used is *N for N x N x N problem size.*
- Spiral focuses on Auto tuning of 1D FFTs.

We thought of Auto tuning Parallel 3D FFTs

Approach for Auto Tuning

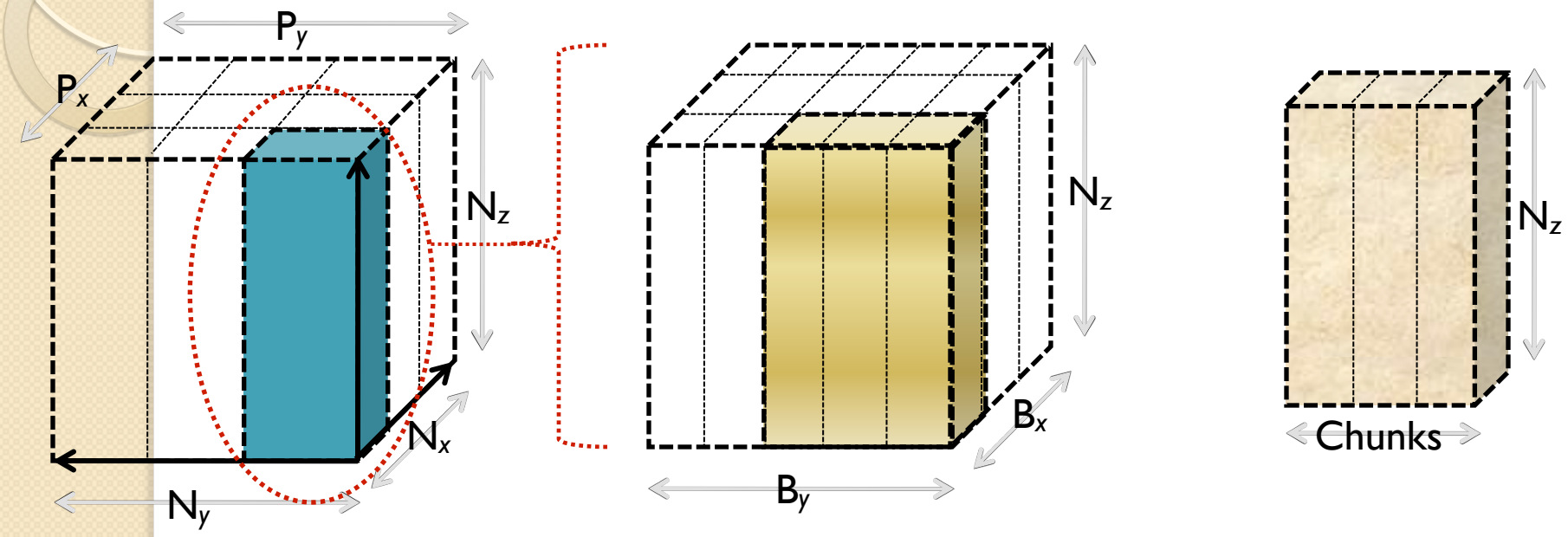
- Auto tune a kernel involves two steps
  - Identification of set of methods to perform 3D FFT.
  - Identification of the fastest method after analyzing the collected data.

- Methods :
  - FFTW 3D FFT library routine
  - Use of AlltoAllv
  - Use of ISend Receive
    - Transmission of Entire buffer
    - Transmission of a part of buffer (Overlapping of Communication and Computation)

Our Approach

- An optimal set of columns are packaged together and communicated.
- The number of columns that are packaged together depends upon the optimal bandwidth of the network during the computation and communication overlap.

Our Approach



(a)

Decomposition of 3D grid On ***N*** nodes (P_x *X* P_y) in blocks.

(b)

Decomposition of Block into ***M*** number of chunks.

$B_x = N_x / P_x$ $B_y = N_y / P_y$

(c)

C number of chunks packaged together for computation and transmission

Our Approach : Staggered Communication

for i = 1 to Number of Processors

Proc 0	Proc 1	Proc 2	Proc 3
0 → 1	1 → 2	2 → 3	3 → 0
0 → 2	1 → 3	2 → 0	3 → 1
0 → 3	1 → 0	2 → 1	3 → 2

Our Approach: Staggered Communication

- The communications are staggered rather than all just sending to the same processor at the same time.

For one block of "m" FFTs

for i = 1 to Number of Processors

- Process *i will receive data from mod(MyRank−i−1+NoNodes, NoNodes)*.
- Process i will send data to *mod(MyRank + i − 1, NoNodes)*.

System Configuration

	Franklin
Model	XT4
Cores / Comm. Channel	4
CPU Clock	2.3 GHz AMD Quad Core (Budapest)
Memory Speed	800.0 MHz
Comm. Channel B/W	7.6 GB/s
I/O Bandwidth	36.0 GB/s
Network Topology	3D Torus
Interconnect	Cray Sea Star2

Results of Different Methods

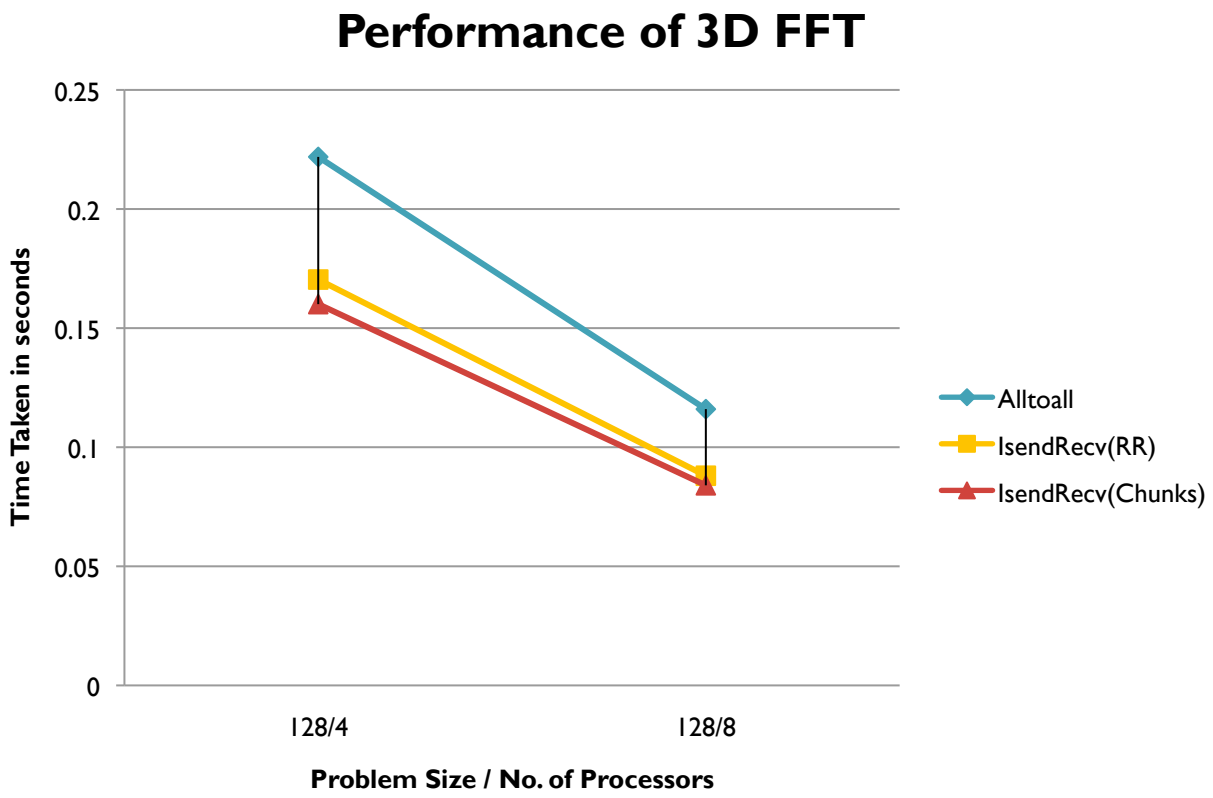
Procs	alltoallv	isenrec	Auto (4C)	Chunks
4	0.2218	0.1703	0.1600	64
8	0.1159	0.0880	0.0840	128

Grid Size : 128^3

Message size Transmitted : 32K

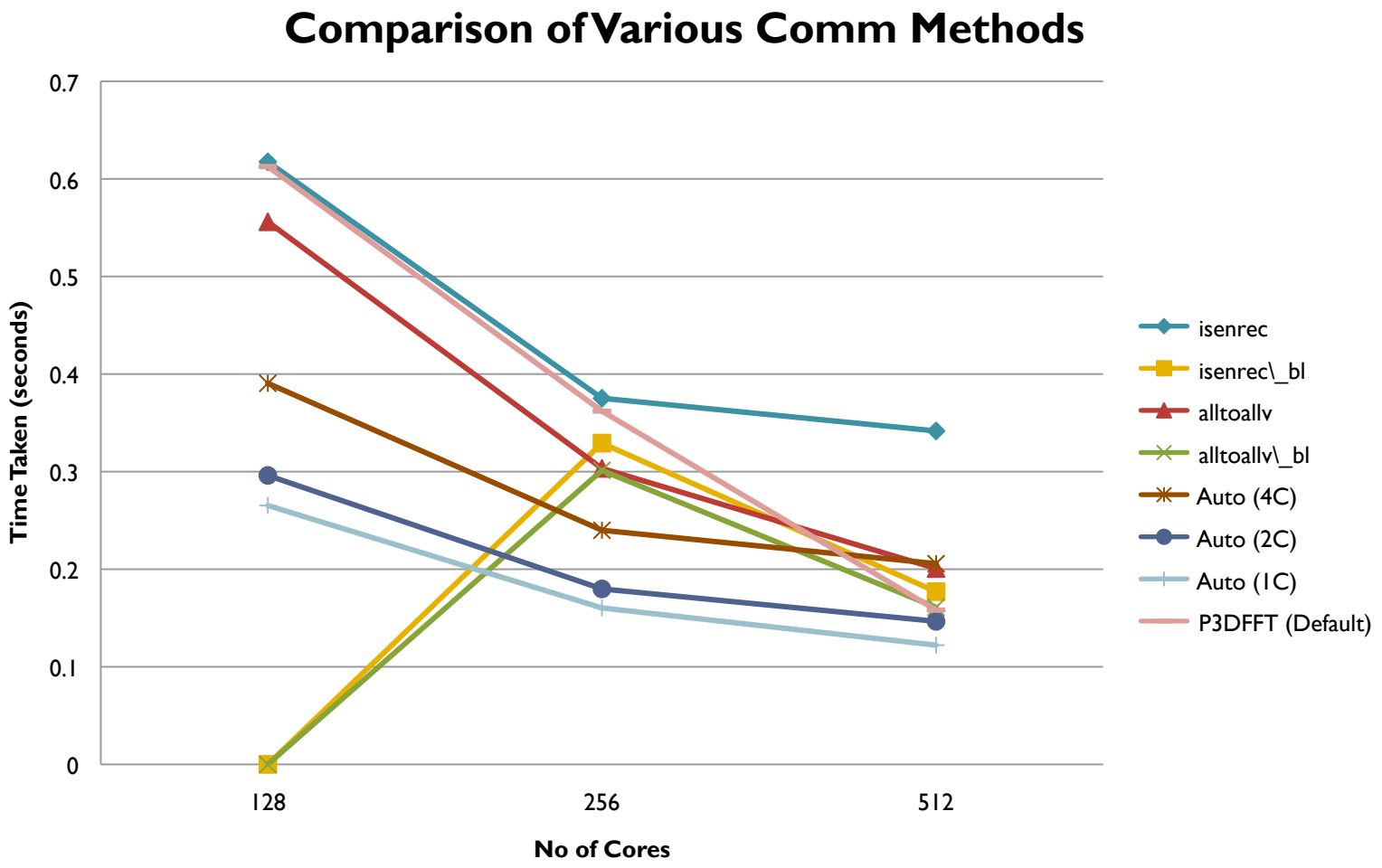
On Franklin (Cray XT4) at NERSC

Results of Different Methods



Grid Size : 128^3

Results of Different Methods



Results of Different Methods

Procs	isenrec	alltoallv	Auto (4C)	Auto (2C)	Auto (1C)	P3DFFT (Default)
128	0.6175	0.556	0.3905	0.2961	0.2653	0.6127
256	0.3752	0.3033	0.2399	0.1798	0.1603	0.362
512	0.3417	0.2007	0.2057	0.1466	0.1222	0.1582
1024		0.1992	1.65			0.3626

Grid Size : 512^3

P3DFFT : Parallel 3D FFT library developed by SDSC

On Franklin (Cray XT4) at NERSC

Challenges in Auto Tuning

- One major challenge is finding efficient ways to select methods at run-time when several known implementations are available.

- Aim is to discuss a possible framework, using sampling and statistical classification, for attacking this problem in the context of automatic tuning systems as well as development of performance model to predict the performance of scientific applications on the upcoming parallel systems.

Performance Model

$$T_{fft}(N, P, Chunks) = T_{comp}(N*Chunks, P) + T_{comm}(Chunks, P)$$

where

$T_{comp}(N*Chunks, P)$: Computation Time

$T_{comm}(Chunks, P)$: Communication Time

Future Work

Search Space

- Message Size :
  - Depends upon the optimal bandwidth of a particular system select the message size to be sent. This means select the number of chunks to be used for overlapping.
- Chunks :
  - How many 1 D FFT one can pack and use for overlapping of computation and communication.
- Pencil size :
  - Depends upon memory per core, optimal message size
- Data Decomposition :
  - 2D data decomposition based on the pencil size used or vice versa
  - 3D data decomposition : can use N x N x N processing elements

Future Work

Search Space

- Data Decomposition :
  - 2D data decomposition based on the pencil size used or vice versa
  - 3D data decomposition : can use N x N x N processing elements

Performance Model

Publications

- PhD Forum Technical Paper on "**Performance Modeling and Optimization of 3 Dimensional Fast Fourier Transforms**". Accepted in IPDPS 2009. Authors : Manisha Gajbe (GATech).

- A paper titled "**Optimizing and Auto Tuning of 3 Dimensional Fast Fourier Transform on Cray XT4**". Accepted in CUG 2009, Compute the Future, in Atlanta, Georgia, May 4-7, 2009. Authors : Manisha Gajbe (GATech), Andrew Canning, John Shalf, Ling Wang, Harvey Wasserman (LBNL), Richard Vuduc (GATech)

- A paper titled "**A Comparison of Different Communication Structures for Scalable Parallel Three Dimensional FFTs in First Principles Codes**" is submitted to ParCo 2009. Authors : Andrew Canning, John Shalf, Ling Wang, Harvey Wasserman (LBNL), Manisha Gajbe (GATech).

- A poster titled "**Scalable Parallel 3d FFTs for Electronic Structure Codes**" is presented at SIAM Conference on Computational Science and Engineering (CSE09) by LBNL team. Authors : Andrew Canning, John Shalf, Ling Wang, Harvey Wasserman (LBNL), Manisha Gajbe (GATech).

QUESTIONS ?