MS260
Next Generation FFT Algorithms in Theory and Practice: Parallel Implementations and Applications

• Organizers:
  – Daisuke Takahashi
    University of Tsukuba, Japan
  – Franz Franchetti
    Carnegie Mellon University, U.S.
  – Samar A. Aseeri
    King Abdullah University of Science & Technology (KAUST), Saudi Arabia
  – Benson Muite
    Kichakato Kizito, Kenya
Aim of this minisymposium

• The fast Fourier Transform (FFT) is an algorithm used in a wide variety of applications, yet does not make optimal use of many current hardware platforms.

• Hardware utilization performance on its own does not however imply optimal problem solving.

• The purpose of this minisymposium is to enable exchange of information between people working on alternative FFT algorithms, to those working on FFT implementations, in particular for parallel hardware.

• [http://www.fft.report](http://www.fft.report)
MS260

• 2:15-2:35 Automatic Tuning of Computation-Communication Overlap for Parallel 3-D FFT with 2-D Decomposition
  Daisuke Takahashi, University of Tsukuba, Japan

• 2:35-2:55 Updates on FFTX and Spectralpack
  Franz Franchetti, Carnegie Mellon University, U.S.

• 2:55-3:15 A Scheduling Policy to Improve 10% of Communication Time in Parallel FFT
  Samar A. Aseeri, King Abdullah University of Science & Technology (KAUST), Saudi Arabia

• 3:15-3:35 FFT for Magnetohydrodynamic Simulations
  Benson Muite, Kichakato Kizito, Kenya

• 3:35-3:55 The Crucial Role of Parallel FFT in a New Computational Algorithm of Electronic Structure
  Dietrich Foerster, Bordeaux University, France
Automatic Tuning of Computation-Communication Overlap for Parallel 3-D FFT with 2-D Decomposition

Daisuke Takahashi
Center for Computational Sciences
University of Tsukuba, Japan
Outline

• Background
• Objectives
• Parallel 3-D FFT with 2-D Decomposition
• Computation-Communication Overlap
• Automatic Tuning of Parallel 3-D FFT with 2-D decomposition
• Performance Results
• Conclusion
Background (1/2)

• The fast Fourier transform (FFT) is widely used in science and engineering.

• Several FFT libraries with automatic tuning have been proposed, including FFTW [Frigo and Johnson 05] and SPIRAL [Puschel et al. 2005, Franchetti et al. 2018].

• Parallel FFTs on distributed-memory parallel computers require intensive all-to-all communication, which affects their performance.

• How to overlap the computation and the all-to-all communication is an issue that needs to be addressed for parallel FFTs.
Background (2/2)

• A typical decomposition for performing a parallel 3-D FFT is slabwise.
  – A 3-D array $x(N_1, N_2, N_3)$ is distributed along the third dimension $N_3$.
  – $N_3$ must be greater than or equal to the number of MPI processes.
• This becomes an issue with very large MPI process counts for a massively parallel cluster of many-core processors.
• To solve this problem, parallel 3-D FFTs with 2-D decomposition have been proposed [Takahashi 2010, Pekurovsky 2012, Ayala and Wang 2013].
Related Works

- Overlapping methods of all-to-all communication and FFT algorithms for torus-connected massively parallel supercomputers [Doi and Negishi 2010].
  - None of their implementations optimize the computation-communication overlap automatically.

- Computation-communication overlap and parameter auto-tuning for scalable parallel 3-D FFT [Song and Hollingsworth 2016].
  - Their approach requires the non-blocking MPI_Ialltoall operation described in the MPI-3.0 standard.
  - The number of MPI_Test calls also needs to be tuned.
Objectives

• On the other hand, a computation-communication overlap method that introduces a communication thread with OpenMP has been presented [Idomura et al. 2014, Maeyama et al. 2015].
• This method does not require the MPI-3.0 standard non-blocking collective operations.
• We used this method for the computation-communication overlap.
• We propose a method for the automatic tuning of the computation-communication overlap for a parallel 3-D FFT with 2-D decomposition.
3-D DFT

- 3-D discrete Fourier transform (DFT) is given by

\[
y(k_1, k_2, k_3) = \sum_{j_1=0}^{n_1-1} \sum_{j_2=0}^{n_2-1} \sum_{j_3=0}^{n_3-1} x(j_1, j_2, j_3) \omega_{n_3}^{j_3k_3} \omega_{n_2}^{j_2k_2} \omega_{n_1}^{j_1k_1},
\]

\(0 \leq k_r \leq n_r - 1, \omega_{n_r} = e^{-2\pi i/n_r}, 1 \leq r \leq 3\)
1-D Decomposition along the z-axis

1. FFTs in x-axis    2. FFTs in y-axis    3. FFTs in z-axis

With a slab decomposition
2-D Decomposition along the y- and z-axes

1. FFTs in x-axis  
2. FFTs in y-axis  
3. FFTs in z-axis

With a pencil decomposition
Computation-Communication Overlap [Idomura et al. 2014]

```fortran
!$OMP PARALLEL
!$OMP MASTER

MPI communication

!$OMP END MASTER
!$OMP DO SCHEDULE(DYNAMIC)
DO I=1,N

Computation

END DO
!$OMP DO
DO I=1,N

Computation using the result of communication

END DO
!$OMP END PARALLEL
```

← MPI communication is performed on the master thread

← No barrier synchronization

← Computation is performed by a thread other than the master thread

← Computation is performed after completion of the MPI communication
Pipelined Computation-Communication Overlap

Without overlap

Overlap (NDIV=2)

Overlap (NDIV=4)
Automatic Tuning of Parallel 3-D FFT with 2-D Decomposition

• The automatic tuning process consists of three steps:
  – Selection of the MPI process grid \((P \times Q)\)
  – Selection of the number of divisions NDIV for the computation-communication overlap
  – Selection of the block size NB
Selection of MPI Process Grid

- Typically, $P$ and $Q$ such that the total number of MPI processes is $P \times Q$ are chosen to be $P \approx Q \approx \sqrt{PQ}$.
- By searching all combinations of $P$ and $Q$, optimal combinations of $P$ and $Q$ can be examined.
- When the number of MPI processes $P \times Q$ is a power of two, even if all combinations of $P$ and $Q$ have been examined, the search space is of size $\log_2(PQ) + 1$. 
Selection of Number of Divisions for Computation-Communication Overlap

• When the number of divisions for computation-communication overlap is increased, the overlap ratio also increases.
• On the other hand, the performance of all-to-all communication decreases due to reducing the message size.
• Thus, a tradeoff exists between the overlap ratio and the performance of all-to-all communication.
• The default overlapping parameter of the original FFTE 7.1alpha is NDIV=4.
• In our implementation, the overlapping parameter NDIV is varied between 1, 2, 4, 8, and 16.
Selection of Block Size

• The default blocking parameter of the original FFTE 7.1alpha is NB=32.

• Although the optimal block size may depend on the problem size, the block size NB can also be varied.

• In our implementation, the block size NB is varied between 8, 16, 32, and 64.
Performance Results

• To evaluate the parallel 3-D FFT with automatic tuning, we compared
  – FFTE 7.1alpha (without overlap)
  – FFTE 7.1alpha (with overlap, NDIV=4)
  – FFTE 7.1alpha with automatic tuning (AT)
  – FFTW 3.3.9

• Weak scaling \((N = 256 \times 512 \times 512 \times \text{MPI processes})\) and strong scaling \((N = 256 \times 512 \times 512)\) were measured.
Evaluation Environment

- Oakforest-PACS at Joint Center for Advanced HPC (JCAHPC).
  - 8208 nodes, Peak 25.008 PFlops
  - CPU: Intel Xeon Phi 7250 (68 cores, Knights Landing 1.4 GHz)
  - Interconnect: Intel Omni-Path Architecture
  - Compiler: Intel Fortran compiler 19.0.5.281 (for FFTE)
    Intel C compiler 19.0.5.281 (for FFTW)
  - Compiler option: “-O3 -xMIC-AVX512 -qopenmp”
  - MPI library: Intel MPI 2019.5.281
  - flat/quadrant, MCDRAM only, KMP_AFFINITY=balanced
  - Each MPI process has 17 cores and 17 threads, i.e. 4 MPI processes per node.
## Results of Automatic Tuning of Parallel 3-D FFTs (Oakforest-PACS, 8192 MPI processes)

<table>
<thead>
<tr>
<th>N^3</th>
<th>P</th>
<th>Q</th>
<th>NDIV</th>
<th>NBLK</th>
<th>GFlops</th>
<th>P</th>
<th>Q</th>
<th>NDIV</th>
<th>NBLK</th>
<th>GFlops</th>
</tr>
</thead>
<tbody>
<tr>
<td>512^3</td>
<td>64</td>
<td>128</td>
<td>4</td>
<td>32</td>
<td>701.5</td>
<td>64</td>
<td>128</td>
<td>1</td>
<td>64</td>
<td>2199.0</td>
</tr>
<tr>
<td>1024^3</td>
<td>64</td>
<td>128</td>
<td>4</td>
<td>32</td>
<td>3857.6</td>
<td>32</td>
<td>256</td>
<td>1</td>
<td>64</td>
<td>6928.0</td>
</tr>
<tr>
<td>2048^3</td>
<td>64</td>
<td>128</td>
<td>4</td>
<td>32</td>
<td>8941.3</td>
<td>64</td>
<td>128</td>
<td>1</td>
<td>16</td>
<td>12012.2</td>
</tr>
<tr>
<td>4096^3</td>
<td>64</td>
<td>128</td>
<td>4</td>
<td>32</td>
<td>7875.5</td>
<td>8</td>
<td>1024</td>
<td>1</td>
<td>16</td>
<td>15511.5</td>
</tr>
<tr>
<td>8192^3</td>
<td>64</td>
<td>128</td>
<td>4</td>
<td>32</td>
<td>7508.9</td>
<td>4</td>
<td>2048</td>
<td>2</td>
<td>8</td>
<td>22231.7</td>
</tr>
</tbody>
</table>

As the problem size increases, MPI processes PxQ with an elongated shape becomes optimal.
Performance of Parallel 3-D FFTs
($N = 256 \times 512 \times 512 \times \text{MPI processes}$)

![Graph showing the performance of parallel 3-D FFTs with different FFTE versions and overlap settings.](image)
Performance of Parallel 3-D FFTs
\((N = 256 \times 512 \times 512)\)
### Breakdown of Execution Time in FFTE 7.1alpha

\(N = 8192^3, 8192\) MPI processes

<table>
<thead>
<tr>
<th>Time (sec)</th>
<th>FFTE 7.1alpha (with overlap, NDIV=4)</th>
<th>FFTE 7.1alpha with AT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rearrangement4</td>
<td>12</td>
<td>4</td>
</tr>
<tr>
<td>Alltoall4</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>Rearrangement2+FFTz+Transpose3+Alltoall3</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>Rearrangement1+FFTy+Transpose2+Alltoall2</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>FFTx+Transpose1+Alltoall1</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>
Conclusion

• We proposed an automatic tuning of computation-communication overlap for parallel 3-D FFT with 2-D decomposition.
• We used a computation-communication overlap method that introduces a communication thread with OpenMP.
• An automatic tuning facility for selecting the optimal parameters of the MPI process grid, the computation-communication overlap, and the block size was implemented.
• The performance results demonstrate that the proposed implementation of a parallel 3-D FFT with 2-D decomposition and automatic tuning is efficient for improving the performance.