An Efficient Implementation of the Parallel Prime Edge-length Crystallographic FFT

Daniel Alberto Burbano
Advisor: Jaime Seguel

Electrical and Computer Engineering Department
University of Puerto Rico, Mayaguez Campus
Mayagüez, Puerto Rico
dburbano@ece.uprm.edu

Abstract

This paper implements a parallel multidimensional crystallographic FFT of prime edge-length, and reports its run times on a cluster of eight SMPs.

1. Introduction

The present works implements the Parallel prime edge-length Crystallographic FFT (PCFFT). This implementation is organized in four phases (Figure 1). The first and last are computation phase which are implemented using Fastest Fourier Transform in the West (FFTW). The second phase develops the Hadamard product and it uses bit permutation to rearrange data. The third phase develops a hypercube communication based on Karnaugh map using Message Passing Interface (MPI).

This article is organized as follows: Section 2 provides a detailed description of PCFFT and implementation phases of PCFFT. Section 3 summarizes the results of the experiment of PCFFT for hypercube model and for MPI collective communication.

2. Background

The core computations of PCFFT are represented in [Seguel02b] as $\overrightarrow{U}$. The vector $\overrightarrow{U}$ is composed by Q-point segments denoted $\overrightarrow{U}_{\vec{a}}, \vec{a} \in I_s$. This is represented by Equation 1 from [Seguel02b]. We assume that the processors are power of two that divides Λ. We denote the ratio by $\rho = \Lambda / P$. We decompose V in P processor groups of size ρQ and denote the set of indices of the i-th group by P_i. Thus, processor i receives the collection of input vector segments $\overrightarrow{V_{\vec{b}}}$ with $\vec{b} \in P_i$. A pre-computed array of size ΛQ contains the diagonal entries of $\Delta\left(H_{(\vec{a},\vec{b})}\right)$, it is stored in the processor i. This array is denoted by $D_{\vec{b}}$. For a Q-point vector Y, we define $D_{\vec{b}} \cdot Y$ as the Hadamard product of the vector formed by concatenating Λ copies of Y and $D_{\vec{b}}$.

$$\overrightarrow{U}_{\vec{a}} = F_Q^{-1}\left[\sum_{\vec{b}\in I_{s*}} \Delta\left(H_{(\vec{a},\vec{b})}\right) F_Q^{-1}\overrightarrow{V_{\vec{b}}}\right] \quad \text{Equation 1}$$

Using these conventions, the PCFFT is represented by the Figure 1, where the computation set is represented by the FFTW phase (phase 1) that is denoted as $Y_{\vec{a}} = F_Q^{-1} V_{\vec{a}}$, Hadamard product (phase 2) that is denoted as $Z_{\vec{a}} = D_{\vec{a}} \cdot Y_{\vec{a}}$, and a sub-phase inside of communication phase that is denoted as $X_i = \sum_{\vec{a}\in P_i} Z_{\vec{a}}$. The previous phases do not require inter-processor communication.

In the communication phase, for each processor peer ***(i,j)***, the processor ***i*** sends the upper segment of X_i to processor ***j***, and processor ***j*** sends the lower segment of X_j (Figure 2). Then the processor ***i*** adds the lower segment of X_j with its lower segment X_i and stores the result in a new half size vector X_i; while processor ***j*** adds the upper segment of X_i with its upper segment of X_j and stores the result in a new half size vector X_j.

At the end of the communication phase, the core computation is completed by performing $U_{\vec{a}} = F_Q^{-1} X_{\vec{a}}^{(i)}$ for each $\vec{a} \in P_i$ (Phase 4).

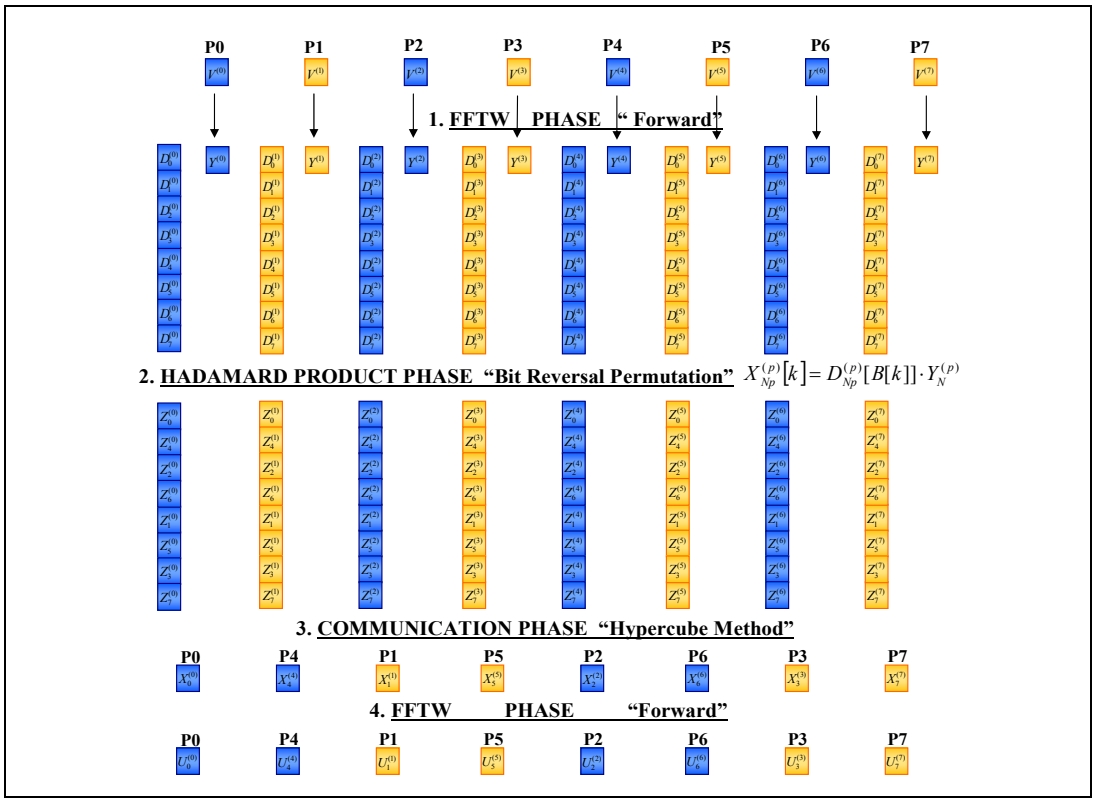

Figure 1. PCFFT structure for 8 processes.

The parallel communication of PCFFT is implemented by our hypercube topology; this topology performs the communication in $\log P$ steps (Figure 3). Also, the parallel prime edge-length symmetric FFT introduced in "A new Prime Edge Length Crystallographic FFT" [Seguel02] involves O(n²) communications. By using a hypercube method, we have managed to reduce the communication to O(n).

Where the problem size is given by $M = N.p$; p is the number of processes and N is the number of elements in the Vector V. The number of communication stages in the butterfly structure is given by $\log p$, and each processes transfers the half of information on each stage (Figure 2). Based on these characteristics, we conclude that the communications with the hypercube method in the parallel Prime Edge Length Crystalographic FFT are less than Np (Equation 3).

$$\sum_{i=1}^{\log p} \frac{Np}{2^i} = Np \sum_{i=1}^{\log p} \frac{1}{2^i} = Np\left(1 - \frac{1}{2^{\log p}}\right) \leq Np \quad \text{Equation 3}$$

The hypercube method is conformed by two strategies: First, hypercube communication which is implemented by MPI; and second, data rearrangement which uses bit reversal permutation.

First strategy, the hypercube method bases on Karnaugh map [Karnaugh53] for selects the process peers (Figure 3). This method selects the processes whose id changes from one to another in a bit.

Second strategy, bit reversal permutation rearranges the data to reduce the communications and auxiliary buffers (Figure 2).

The parallel program developed in this research implements the Equation 1. The Equation 2 is a matrix notation representation of Equation 1. In the Equation 2, the expression $V_N^{(p)}$ defines a column vector V with N elements belonging at the process p. The notation $D_{Np}^{(p)}$ defines a column vector D with $N*p$ elements belonging at the process p. $B[k]$ is the bit reversal permutation. And $Z_{Np}^{(p)}$ is the Hadamard product of D and V belonging at the process p with $N*p$ elements. To simplify the equation, the size of the vector Z depends on number of processes, and the size of the vectors V are homogenous in every processes. Each process has a different vector V and D created randomly.

$$X_{Np}^{(p)}[k] = D_{Np}^{(p)}[B[k]] \cdot V_{N}^{(p)} \quad \textbf{Equation 2}$$

In the hypercube method, the Hadamard product is executed before the communications, which it is the product of a bit-reversal permutation of *D* with *V*. The bit-reversal permutation rearranges the data (Figure 2) to allow process communicates data without jumps or auxiliary buffers. That information is storages in *Z*.

As soon as it is storages, each process sends the half of *Z* vector to its peer process. Depend on specified flag, the process sends the upper half or the lower half of the *Z* vector. When the information is received by peer entity, the peer process sums it with the its half section that was not send and storages in *X*.

The process pairs for each communication epoch are identified by its rank binary number, from Less Significant Bit (LSB) to the Most Significant Bit (MSB). That means, the LSB identifies the process peers of the first communication epoch, the second bit identifies the process peers of the second epoch, and so on (Figure 3).

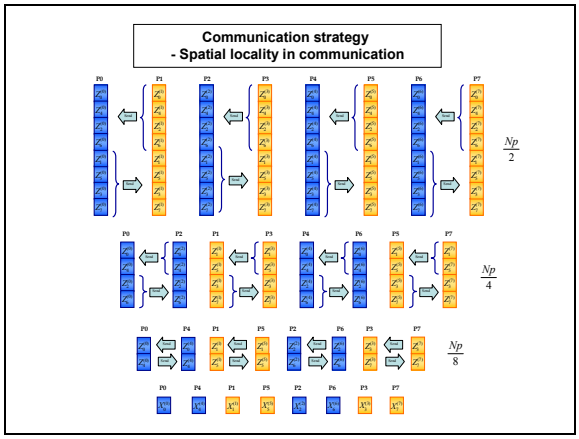

Figure 2. Communication Strategy, spatial locality in communication, using butterfly and bit-reversal models.

3. Results and conclusions

The programs were executed in a cluster of 8 SMPs. The SMPs are conformed by Pentium III 650 MHz dual, Cache 256 KB, RAM 256 MB. The nodes and server are interconnected by a switch CentreCOM FS716 with 16 ports. The software used on these tests are OSCAR 1.2.1, LAM 6.5.6.

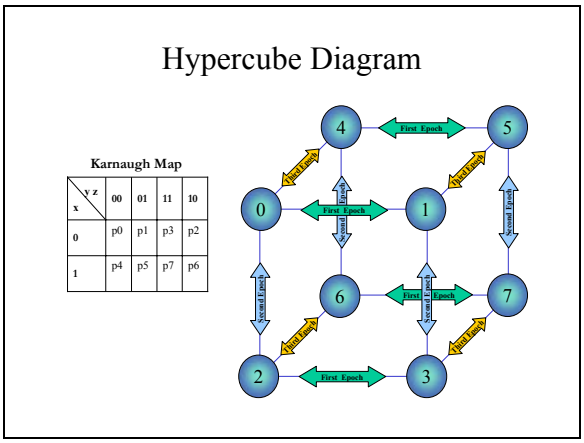

Figure 3. Structure of a hypercube of 8 processes with its Karnaugh map.

The program was evaluated for problem sizes from 12167 (Λ=24, Q=263) to 30080231 (Λ=312, 48359) elements, and processes from 2 to 8, such as of them are in the Table 1. The parallel program was executed with the following line: "`mpirun -np p filename Λ Q -nolocal`", where `p` is the number of processes, `filename` is the name of the parallel program to be executed, Λ is the number of segments of the problem, and *Q* is the number of elements of each segment, and `nolocal` is a conditional flag which executes the program out of the server node. The execution time is taken at the beginning and at the final of the programs.

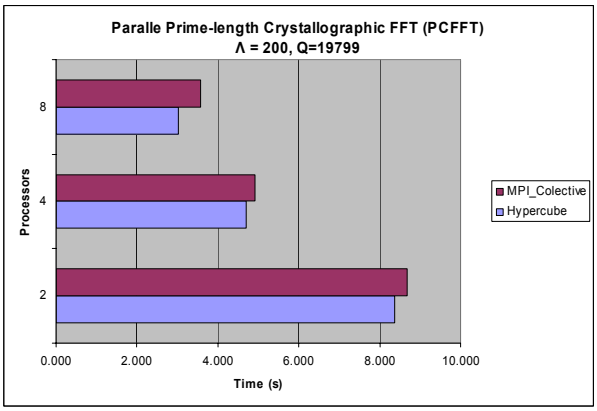

Figure 4. Execution times of PCFFT (hypercube method), PCFFT (MPI_Reduce_scatter) for 2, 4, 8 processors, Λ=200.

We compare the PCFFT program implemented with our hypercube method with the PCFFT program implemented with MPI collective communications (MPI_Reduce_scatter). Part of the information is in the Table 1.

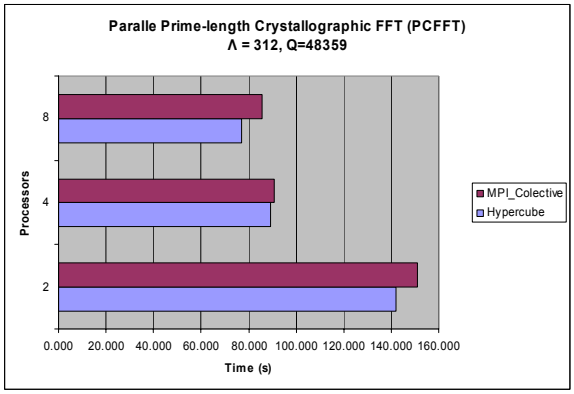

Figure 5. Execution times of PCFFT (hypercube method), PCFFT (MPI_Reduce_scatter) for 2, 4, 8 processors, Λ=312.

In many cases the PCFFTW with our hypercube method is faster than PCFFTW with MPI collective communication (Figure 4, Figure 5, Table 1). This difference increases when the problem size increases.

Table 1. Execution times of PCFFT (hypercube method), PCFFT (MPI Native function).

Λ	Q	P	Hypercube	MPI_Collective
200	19799	2	8.379	8.692
		4	4.700	4.909
		8	3.018	3.572
224	24863	2	9.306	9.386
		4	5.360	5.546
		8	3.572	4.223
240	28559	2	14.840	14.742
		4	8.183	8.385
		8	5.184	5.924
264	34583	2	44.403	39.043
		4	17.332	17.757
		8	10.336	10.939
272	36719	2	42.715	29.612
		4	8.767	9.526
		8	5.803	6.762
312	48359	2	141.870	150.935
		4	89.265	90.565
		8	77.259	85.756

The implementation of PCFFT with our hypercube method allows us to overlap computation and communication using additional MPI functions. In future work, we will study the computation and communication overlapping and evaluates the program in a cluster of 64 dual SMPs.

4. References

[Foster95] Ian Foster, "Design and Building Parallel Programs", Addison-Wesley, 1995.

[Frigo98] M. Frigo and S. G. Johnson, "FFTW: An Adaptive Software Architecture for the FFT", in proceedings of the International Conference on Acoustic Speech and Signal Processing, ICASSP 1998, vol. 3, pp. 1381-1384.

[Karnaugh53] Karnaugh, M, "A Map Method for Synthesis of Combinational Logic Circuits" Trans. AIEE, Comm. And Electronics, Vol. 72, Part I, November 1953.

[NCSA02] National Computational Science Alliance, "Introduction to MPI", http://foxtrot.ncsa.uiuc.edu:8900/webct/public/home.pl, December 2002.

[Seguel02] Jaime Seguel, D. Bollman, and E. Orozco, "A new prime Edge Length Crystallographic FFT", Lecture Notes of Computer Science, Vol. 2330, pp. 548-557, 2002.

[Seguel02b] Jaime Seguel and Daniel Burbano, "A Parallel Prime Edge-length Crystallographic FFT", in proceedings of the International Conference on Computational Science and Its Applications, ICCSA 2003, May 2003.

[Snir98] Marc Snir, Steve Otto, Steven Huss-Lederman, David Walker, Jack Dongarra, "MPI –The Complete Reference Volume 1, The MPI Core", 1998.