Performance Measurements of the 3D FFT on the Blue Gene/L Supercomputer

Maria Eleftheriou, Blake Fitch, Aleksandr Rayshubskiy,
T.J. Christopher Ward, and Robert Germain

IBM Thomas J. Watson Research Center
Yorktown Heights, NY 10598-0218
{mariae,bgf,arayshu,tjcw,rgermain}@us.ibm.com

Abstract. This paper presents performance characteristics of a communications-intensive kernel, the complex data 3D FFT, running on the Blue Gene/L architecture. Two implementations of the volumetric FFT algorithm were characterized, one built on the MPI library using an optimized collective `all-to-all` operation [2] and another built on a low-level System Programming Interface (SPI) of the Blue Gene/L Advanced Diagnostics Environment (BG/L ADE) [17]. We compare the current results to those obtained using a reference MPI implementation (MPICH2 ported to BG/L with unoptimized collectives) and to a port of version 2.1.5 the FFTW library [14]. Performance experiments on the Blue Gene/L prototype indicate that both of our implementations scale well and the current MPI-based implementation shows a speedup of 730 on 2048 nodes for 3D FFTs of size $128 \times 128 \times 128$. Moreover, the volumetric FFT outperforms FFTW port by a factor 8 for a $128 \times 128 \times 128$ complex FFT on 2048 nodes.

1 Introduction

The primary goal of the computational science portion of the Blue Gene project is to advance our understanding of biological phenomena such as protein folding via large scale computer simulations[10] and as part of this effort the Blue Gene application group has developed a Molecular Dynamics framework, *Blue Matter* [12]. Molecular dynamics (MD) is a well-established computational method used to study complex bimolecular systems. MD permits the computation of thermodynamic and kinetic properties of biomolecular systems[19]. Such studies can enhance our understanding of biological functions and provide insights into processes related to the action of potential new medications.

Blue Gene/L[15] is a massively parallel supercomputer developed at the IBM T.J. Watson Research Center in collaboration with Lawrence Livermore National Laboratory. BG/L has five networks, two of which are of particular interest to the application developers: the torus and the collective network. The three-dimensional torus has links between each node and its six neighbors while the collective network enables low latency broadcast and reduction operations as well as providing the path for i/o to external devices. Each node operates at a relatively low clock frequency of 700 MHz and has two PowerPC 440 CPUs on the same chip with associated dual floating point units. Two modes of operation are supported by the system software: (1) "coprocessor mode" which runs a single MPI task, and “virtual node mode” which runs two MPI-tasks on each node.

The operating assumption of the Blue Gene/L architecture is that performance gains will come from massive parallelism rather than increases in processor clock speed. This implies that algorithms must achieve good scalability across many thousands of processors in order to fully exploit the power of this massively parallel supercomputer. One of the major factors limiting the scalability of parallel molecular dynamics algorithms is the calculation of full electrostatic forces using Particle-Particle-Particle-Mesh-Ewald (P3ME)[5] techniques. P3ME methods involve the calculation of a convolution using two 3D-FFTs for each time step. Typical simulations require 10^6 to 10^9 time steps for 10,000-100,000 atoms system. For this reason, efficient computation of the 3D-FFT algorithm is neccessary. However, efficient implementation of the 3D-FFT presents a considerable challenge for the communications infrastructure of a parallel machine because of the all-to-all nature of the distributed transposes required.

The parallel computation of three-dimensional FFTs has been carried out using two approaches. In the first approach one-dimensional FFTs are computed in distributed fashion [7, 23, 24] and in the second approach, the transpose method, uses successive evaluations of independent one-dimensional local FFTs along each direction to evaluate the 3D-FFT [4, 6, 13, 14, 18]. A detailed description of our implementation of the volumetric FFT along with early performance data on the BG/L architecture has recently appeared[8]. A description of an earlier version of the volumetric FFT implementation and its performance on a more conventional (Power4/full bisectional bandwidth switch) machine were reported previously[9].

In this paper we present the performance characteristics of the volumetric FFT on BG/L as implemented on two different communication layers: (1) MPI with unoptimized and optimized all-to-all collective implementations and (2) a low-level System Programming Interface (SPI) of the Blue Gene/L Advanced Diagnostics Environment (BG/L ADE) [17]. The measured performance is compared with limits inherent to the hardware capabilities of the machine such as the bisectional bandwidth of the BG/L torus communications network. On BG/L the 3D-FFT of size of 128^3 continues to speed up through 16,384 nodes for the SPI-based version and up to 8,192 nodes for the optimized MPI-based vesion. On a single node, the volumetric FFT implementations are within 50% of the performance of the ported FFTW version 2.1.5, while outperforming FFTW at larger node counts. It is important to note that the MPI implementation of the volumetric FFT uses BGL/FFTW-GEL [21] 1D serial FFT as a building block. In the sections that follow, we describe the hardware limits to ultimate performance of the 3D FFT on a Blue Gene/L machine based on the hardware “speeds and feeds”[11], present the experimental measurements and discuss the corresponding results.

2 Hardware Limits on 3D FFT Performance

Lower bounds can be placed on the communication time for a 3D FFT implemented using the transpose technique by assuming that three all-to-all communications (along a row or within a plane of the processor mesh) are required. Detailed simulations of the BG/L torus network indicate that the time required for an all-to-all communication involving a set of nodes in a line, plane, or volume can be estimated using the expression[16]:

$$T_{all-to-all} = \frac{V_{received}\,\overline{N}_{hops}}{N_{links}\,BW\,f}$$

where $V_{received}$ is the volume of data received by each node, $\overline{N}_{hops}$ is the average number of hops required (for a three dimensional torus where each dimension is p, $\overline{N}_{hops} = p/4$ for all-to-all in a line, $\overline{N}_{hops} = p/2$ for all-to-all in a plane, and $\overline{N}_{hops} = 3p/4$ for all-to-all in a volume), N_{links} is the number of links available to each node (2 for linear communication, 4 for planar communication, and 6 for volumetric communication), BW is the raw bandwidth of the torus per link (2 bits per processor clock cycle), and f is the link utilization (simulations indicate that this should be about 80%). This expression indicates that the time required for all-to-all communication is independent of the dimensionality of the communication because of the increase of the average hop count with dimensionality is balanced by the increase in the number of links available. At the limits of scalability, where messages consist of a single complex number, the above expression based on bandwidth considerations will become inadequate because the hardware and software latencies associated with sending a packet will become significant.

For an idealized bound on the computation time required to compute a single 1D FFT of length N, we assume $8N\log_2 N$ cycles for a fused multiply-add machine (although the floating point operation count for a 1D FFT is $5N\log_2 N$, data dependencies force a fused multiply-add machine to use 8 cycles). Highly optimized FFT implementations, such as the library developed by the team at the Technical University of Vienna[20] can approach this idealized value. However, the 3D-FFT is dominated by the communication cost of the transposes for even small node counts and the performance of the 1D-FFT building block is not critical.

3 Parallel Performance Analysis

In this section we present performance measurements of the 3D-FFT kernel. Most of the benchmarks reported in this paper were performed on the 20480 node Blue Gene/L system at the IBM T.J. Watson Research Center. Each rack consists of two 512-node mid-planes and each mid-plane comprises 16 node cards. The Blue Gene/L system can be configured to complete a torus in all three dimension only when it is partitioned as 512 nodes and higher, where the number of nodes is power of 2. All of the benchmarks executed in 512-node and larger partitions used the torus network, while benchmarks executed on 32 and 128-node partitions had to use the mesh topology. The performance metric used here is the “total time to solution” for the problem. Our use cases for the 3D FFT from molecular simulations require strong scaling for relatively small data sizes such as 32^3, 64^3 and 128^3.

We compare the performance of two MPI implementations. The first is a port of MPICH2[22],[1] on BG/L, while the second is the optimized version for the BG/L architecture [3] and [2]. The MPI collective of interest to us is `MPI_Alltoallv`. The default MPICH2 `MPI_Alltoallv` implementation is based on non-blocking communication calls `MPI_irecv` and `MPI_isend` and thus requires message matching. However, the optimized implementation runs on a lower level messaging protocol that avoids the overheads of MPI point-to-point messages. Additional optimizations include avoiding sending zero size messages across the network and minimizing the number of cache misses. All the performance numbers reported here were obtained using co-processor mode, where both Power PC 440 processors on each node execute a single MPI task. The MPI tasks were organized in a three-dimensional grid using the MPI Cartesian topology constructs. By mapping the MPI tasks in our domain decomposition "naturally" to the physical machine topology, we can achieve substantial performance improvements since locality plays an important factor in the communication performance.

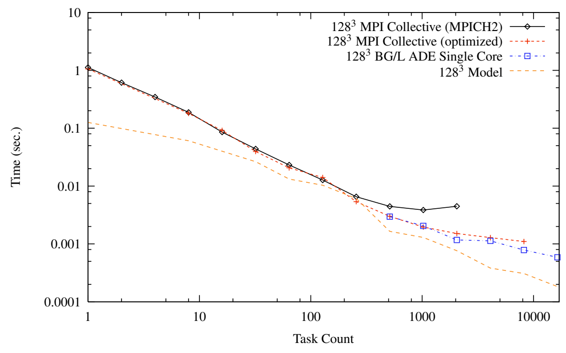
Figure 1 shows the measured execution time for the volumetric 3D-FFT as implemented using MPI collective communications (both unoptimized and optimized) on the 20480-node BG/L system. Speedup in excess of 700 is observed for 128^3 FFT at 2048 nodes and reaches over 900 at 8192 nodes for the MPI-based implementation. The optimized MPI outperforms the ported MPICH2 version when the number of tasks is 256 or more.

Although optimizations of the MPI collectives for the Blue Gene/L architecture give improved performance, it may not yet reflect the full capabilities of the hardware. To investigate whether even better performance is possible, we have implemented the `all-to-allv` collective required by the FFT via the low-level System Programming Interface (SPI) of the Blue Gene/L (ADE) [17]. The BG/L ADE was developed and is utilized by the BG/L hardware group for running diagnostics and manufacturing tests. The SPI packet-level interface provides direct access to the BG/L network-hardware. In our `all-to-allv` collective the packets were directly injected into the send FIFOs and then directly read by the receive FIFOs by using all six links concurrently. Moreover, we prepare the destination lists for the packet-headers in the plan phase, while in the MPI-based implementation the packet-headers have to be evaluated at each MPI call.

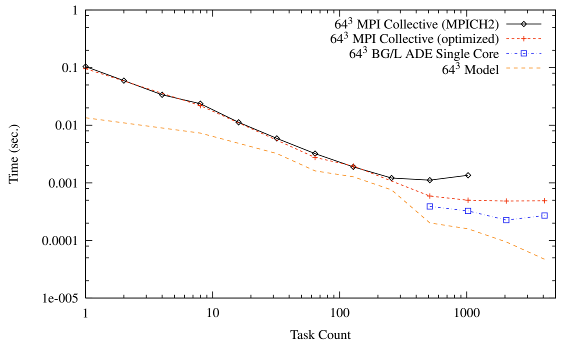
In an attempt to confine the differences in the implementations to the communications layer, the FFT code is written using C++ templates and takes a communications class that encapsulates the details of the communications implementation as a template parameter.

In Figure 2, we characterize the scalability behavior of the 3D-FFT algorithm on both MPI and BlueGene/L Advanced Diagnostics Environment SPI. The FFT scales well on both communications layers. However, as the limits of scalability are approached (where the node count $P = N^2$ for a $N \times N \times N$ FFT and only a single 1D-FFT is performed on each node between transposes) the performance of the SPI implementation still exceeds that of the optimized MPI version. The difference in the performace is probably due to the MPI overhead, related to cash misses on the user's data and dynamically creating destination lists during every MPI collective calls. Whether further improvements are possible in the MPI and/or SPI implementations is a subject for further investigation.

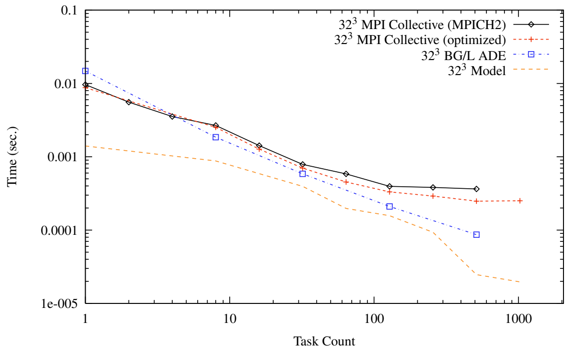
Comparisons of the volumetric 3D FFT on two different architectures, SP and early BG/L prototype, to the FFTW[14] library have been reported previously[8]. In this paper the authors repeat the benchmark on BG/L using the optimized MPI-version and the new SPI implementation of the `all-to-all` collective operation.



(a) $128 \times 128 \times 128$



(b) $64 \times 64 \times 64$



(c) $32 \times 32 \times 32$

Fig. 1. Measured execution times for the volumetric FFT, for a series of problem sizes (32^3, 64^3, 128^3) as a function of number nodes. Two sets of MPI data are shown. The default MPICH2 implementation data were taken in July 2004 while the optimized MPI data were taken in May 2005 after significant optimization of the all-to-all collectives used by the FFT as were the measurements of the implementation using the BG/L Advanced Diagnostic Environment (BG/L ADE). The limits to execution time using simple estimates for computation and communication costs are also shown (the limits shown assume mesh bandwidths which are half of those available on a torus for node count below 512 and torus bandwidths for larger node counts).

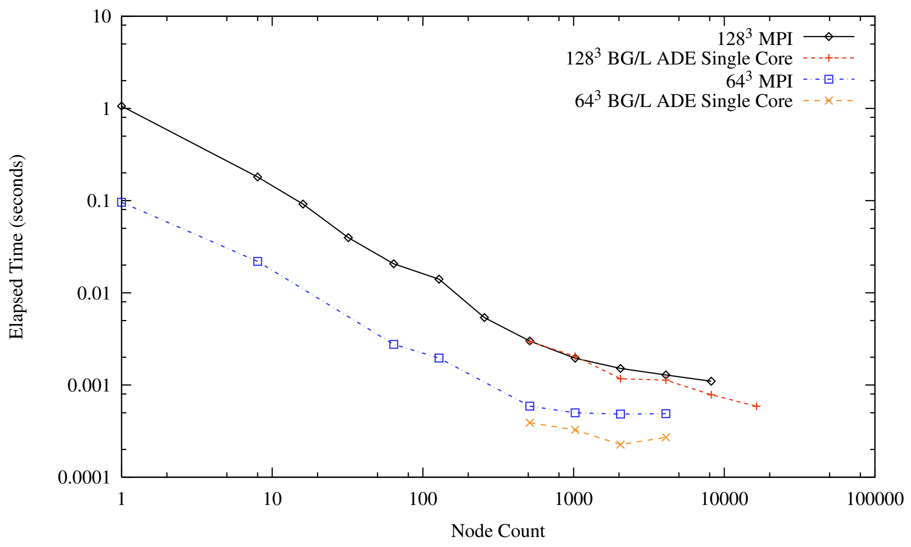


Fig. 2. Performance measurements of the execution time for the volumetric 3D-FFT[8] running on MPI and on low level communications interfaces derived from the BG/L Advanced Diagnostics Environment[17] environment.

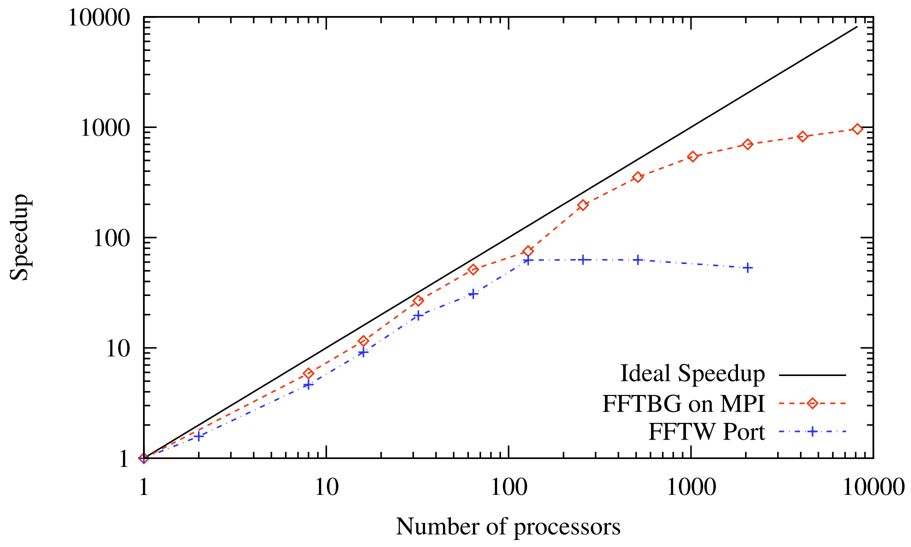


Fig. 3. This plot shows a comparison of the speedups of the two different FFT libraries, volumetric FFT (FFTBG) using the optimized MPI implementation and a straightforward port of version 2.1.5 of FFTW library on Blue Gene/L, as a function of number of processors. The 3D-FFT has a grid size of $(128 \times 128 \times 128)$.

Figure 3 compares the speedup of the volumetric FFT on both SPI and MPI, and a port of the FFTW library. The two versions of the volumetric code and the FFTW library exhibit similar scalability up to 128 nodes. Since the FFTW implementation is based on the slab decomposition, the speedup flattens after 128 nodes. However, the volumetric FFT continues to exhibit good speedups through 2048 nodes for the $128 \times 128 \times 128$ size FFT. Both volumetric FFT versions scale up to 8,192 nodes with the SPI-based version performing about 20% better on 8,192 nodes.

The comparison of the bounds on execution time based on hardware bandwidths and idealized 1D serial FFT execution times with the measured total times to solution in Figure 1 shows significant divergences at small node counts. We conjecture that this divergence is caused by memory hierarchy effects since floating point efficiencies of the 1D-FFTs used as building blocks for the 3D-FFT implementation are fairly high, e.g. for a 64-point FFT, the efficiency of the FFT from the Blue Gene/L FFT library supplied by the Technical University of Vienna is over 60%.

The memory access pattern of the in-memory transpose required for the 3D-FFT is presumably very unfavorable for pre-fetching and at low node counts the memory footprint per node of the larger size 3D-FFTs will certainly spill out of the 4MB L3 cache. No effort has been made to tile the implementation of the memory transposes in the volumetric FFT. Of course, at the high node counts that represent the limits to scalability for the FFT, the data will sit in cache and the measured performance more closely approaches the bandwidth limited constraints on performance. Note that even without any efforts at tiling, the uniprocessor performance of the volumetric FFT implementation is within 50% of that of the FFTW library implementation. We intend to use instrumentation that can access the hardware performance counters available on the Blue Gene/L chip to eventually measure the memory hierarchy effects

4 Summary

We have compared the performance measurements of the volumetric 3D-FFT algorithm, on two communications communication layers, MPI and SPI. Our measurements shows that the volumetric algorithm performs impressively well on both the optimized MPI and SPI communication layer. Moreover, we found that the volumetric FFT outperforms a port of the widely used FFTW library (based on a slab decomposition) by a significant margin on large numbers of nodes.

At the limits of scalability, approached by the 128^3 FFT on 16,384 nodes, where each node sends packets of the size of single complex number the code still scales, with the BG/L ADE SPI implementation still being faster than the MPI-based FFT using the optimized MPI collectives. Future work will involve instrumenting the code to understand the role of memory access patterns in the performance at small node counts and continuing optimization of the implementations on both communications layers. Finally, we plan to reproduce and report the MPI based FFT benchmarks in the virtual node mode in which each PPC440 core on the chip executes a separate MPI task.

Acknowledgments

We would like to acknowledge the contributions of Georghe Almasi, Charles Archer, and Mark Giampappa for their work on the high performance communications layers implemented on Blue Gene/L. We would like to acknowledge the proof-reading of this paper by Frank Suits and Alan Grossfield from the Blue Gene/L application and science team.

References

1. G. Almaśi, C. Archer, J. Castanos, M. Gupta, X. Martorell, J. E. Moreira, Gropp W, S. Rus, and B. Toonen. MPI on Blue Gene/L:Designing an Efficient General Purpose Messaging Solution for a Large Cellular System. In *Proceedings of the 10th EuroPVM/MPI conference, Lecture Notes in Computer Science, Klagenfurt, Austria*, 2003.
2. G. Almaśi, C. Archer, C. Chris Eway, Philip Heidelberger, X. Martorell, J. E. Moreira, B. D. Steinmacher-Burow, and Yili Zheng. Optimization of MPI collective operations on Blue-Genełsystems. 2005. To appear at ICS05.
3. G. Almasi et al. Design and implementation of message-passing services for the Blue Gene/L supercomputer. *IBM Journal of Research and Development*, 49(2/3):393–406, 2005.
4. C. E. Cramer and J. A. Board. The Development and Integration of a Distributed 3D FFT for a cluster of workstations. In *4th Annual Linux Showcase and Conference*, pages 121–128, Atlanta, GA, October 2000.
5. Markus Deserno and Christian Holm. How to mesh up ewald sums. i. a theoretical and numerical comparison of various particle mesh routines. *J. Chem. Phys.*, 109(18):7678–7693, 1998.
6. H. Q. Ding, R. D. Ferraro, and D. B. Gennery. A portable 3D FFT Package for Distributed-Memory Parallel Architecture. In *SIAM Conference on Parallel Processing for Scientific Computing*, 1995.
7. A. Edelman, P. McCorquodale, and S. Toledo. The future fast Fourier transform? In *SIAM J. Sci. Comput.*, volume 20, pages 1094–1114, 1999.
8. M. Eleftheriou, B.G Fitch, A. Rayshubskiy, T.J.C. Ward, and R.S. Germain. Scalable framework for 3d FFTs on the Blue Gene/L supercomputer: Implementation and early performance measurements. *IBM Journal of Research and Development*, 49(2/3):457–464, 2005.
9. Maria Eleftheriou, José E. Moreira, Blake G. Fitch, and Robert S. Germain. A Volumetric FFT for BlueGene/L. In *High Performance Computing – HiPC 2003*, pages 194–203, 2003.
10. F. Allen *et al.* Blue Gene: a vision for protein science using a petaflop supercomputer. *IBM Systems Journal*, 40(2):310–327, 2001.
11. N.R. Adiga *et al.* An overview of the Blue Gene/L supercomputer. In *Supercomputing 2002 Proceedings*, November 2002. `http://www.sc-2002.org/paperpdfs/pap.pap207.pdf`.
12. B.G. Fitch, R.S. Germain, M. Mendell, J. Pitera, M. Pitman, A. Rayshubskiy, Y. Sham, F. Suits, W. Swope, T.J.C. Ward, Y. Zhestkov, and R. Zhou. Blue Matter, an application framework for molecular simulation on Blue Gene. *Journal of Parallel and Distributed Computing*, 63:759–773, 2003.
13. M. Frigo and S. G. Johnson. The Fastest Fourier Transform in the West. Technical Report MIT-LCS-TR-728, Laboratory for Computing Sciences,MIT, Cambridge, MA, 1997.
14. M. Frigo and S. G. Johnson. FFTW: An Adaptive Software Architecture for the FFT. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 3, pages 1381–1384, 1998.
15. A. Gara et al. Overview of the Blue Gene/L system architecture. *IBM Journal of Research and Development*, 49(2/3):195–212, 2005.
16. A. Gara, P. Heidelberger, and B. Steinmacher-burow. private communication.

17. M.E. Giampapa et al. Blue Gene/L advanced diagnostics environment. *IBM Journal of Research and Development*, 49(2/3):319–332, 2005.
18. P. D. Haynes and M. Cote. Parallel Fast Fourier Transforms for electronic structure calculations. *Comp. Phys. Comm.*, 130:121, 2000.
19. M. Karplus and J.A. McCammon. Molecular dynamics simulations of biomolecules. *Nature Structural Biology*, 9(9):646–652, September 2002.
20. Stefan Kral, Franz Franchetti, Juergen Lorenz, Christoph W. Ueberhuber, and Peter Wurzinger. FFT Compiler Techniques. In *Compiler Construction: 13th International Conference, CC 2004, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2004, Barcelona, Spain, March 29 - April 2, 2004. Proceedings*, pages 217–231, 2004.
21. J. Lorenz, S. Kral, F. Franchett, and C. W. Ueberhuber. Vectorization techniques for the Blue Gene/L double FPU. *IBM Journal of Research and Development*, 49(2/3), 2005.
22. The MPICH and MPICH2 homepage. {`http://www-unix.mcs.anl.gov/mpi/mpich`}, January 2004.
23. Mohammad Zubair Ramesh C. Agarwal, Fred G. Gustavson. A high performance parallel algorithm for 1D-FFT. 1994.
24. E. L. Zapata, F. F. Rivera, J. Benavides, J. M. Garazo, and R. Peskin. Multidimensional Fast Fourier Transform into fixed size hypercubes. *IEE Proceedings*, 137(4):253–260, July 1990.