Modification of Crystallographic Codes for Parallel Architectures

M. Ramanadham, B.S. Jagadeesh & R. Chidambaram

Solid State Physics Division & Computer Division
Bhabha Atomic Research Centre, Trombay, Mumbai 400085, India
ramu@magnum.barc.ernet.in

Abstract

The use of high-speed computers having large memories and storage capacities is an essential component of many present-day scientific and engineering applications. In recent years, parallel computers have emerged as viable alternatives to supercomputers, at least for those applications having inherent parallelism in them, such as crystal-structure analysis. Two such applications pertaining to the field of protein crystallography, viz., structure optimization and summation of three-dimensional Fourier series, carried out on ANUPAM, the BARC-built parallel computing system, are described in this article.

1 Introduction

Crystal-structure analysis by the method of single-crystal x-ray or neutron diffraction is one of the most computer intensive branches of modern science. Computers and computations are used at every stage, from data acquisition to structure presentation. Comprehensive and accurate studies on the structures of large proteins and other biological macromolecules by this method have become possible only with the advent of modern computers. It is not uncommon to find the use of even supercomputers for some of the high-end applications pertaining to the field of macromolecular crystallography.

Almost all crystallographic computations have inherent parallelism in them. In many cases, the same series of calculation steps can be carried out simultaneously on different portions of the data set. In some other cases, different calculations, possibly on different data sets can be carried out simultaneously when the calculations are independent of one another. Thus, the use of parallel computers for macromolecular crystallographic applications seems to be a very attractive proposition. A variety of parallel architectures have been designed and built in recent years. ANUPAM, the parallel computing system [1] built at Bhabha Atomic Research Centre (BARC), is one such system that belongs to MIMD (*Multiple Instruction, Multiple Data*) type of parallel architecture. During the past few years, it has been successfully used for calculations pertaining to various fields of research, such as computational fluid dynamics, molecular dynamics and Monte Carlo simulations, electronic structure calculations, weather forecasting and protein crystallography.

2 Parallelization of Codes on ANUPAM

ANUPAM (A *Sanskrit* word, meaning incomparable), designed and built [1] at Bhabha Atomic Research Centre (BARC), India, is a loosely coupled, message passing parallel computing system. It uses powerful RISC processors, interconnected through Multibus II. Each processor has its own memory. One of the processors, known as the master or the host, runs the UNIX operating system, while all the other processors, known as slaves or nodes, run the monitor (control) programs. Only the host processor communicates with the external devices. All the processors, including the host exchange data with one another with the help of library calls introduced in the user code at appropriate places. A number of ANUPAM systems having 8 to 64 processors are operational at BARC and elsewhere during the past few years.

The sequential program of the user has to be modified to make it run on the ANUPM system in the parallel mode. The first step in this direction is to copy all the portions of the source code that can run concurrently on the slave processors into another source file. Hereafter, the original source code is referred to as the master file, while the copied code is referred to as the slave file. Then, the following two lines ,

include '*mincl.inc*'
include '*sinlcl.inc*'

should be inserted in the beginning of the master and slave files respectively. The following line,

call m_init (*no_of_*slave-*cpus, slave_file_name*)

should be inserted as the first executable statement, and another line,

call m_end ( )

should be inserted before the **stop** statement in the master file. The *slave_file_name* is the name of the slave source file with the extension 'dump'. Correspondingly, the following two lines,

call s_init ( )
call s_end ( )

should be inserted in the slave file. The following communication calls are to be inserted in the master and slave source files for exchanging data among the processors during the program execution. A single data element, or variable is communicated using the calls

call send_element (*var, dest_cpu_id, data_type*)
call receive_element (*var, source_cpu_id, data_type*).

To broadcast a data element from one cpu to all the other cpus, the following statement is used.

call b_send_element (*var, data_type*)

Here, *var* is the variable name, *dest_cpu_id* and *source_cpu_id* are the ID numbers of the cpus, and *data_type* is the data type. In ANUPAM, the master processor is identified either by the integer constant, *0*, or by the symbolic variable, *CPU0*. The slave processors are identified either by the integer constants, *1, 2,* etc., or by the symbolic variables, *CPU1, CPU2,* etc. The cpu ID number of any processor during execution can be extracted by the call

id = **get_cpu_id** ( ).

Various data types accepted by ANUPAM are; *char, int, s_real, d_real, s_complex, d_complex and user_defined.* If more than one variables are to be communicated, the following statements,

call send_els (*dest_cpu_id, no_of_vars, var1, data_type, var2, data_type, ...*)
call receive_els (*source_cpu_id, no_ov_vars, var1, data_type, var2, data_type, . . .*)
call b_send_els (*no_of_vars, var1, data_type, var2, data_type, . . .*)

are to be used. The corresponding calls for communicating dimensioned variables, or arrays are,

call send_data (*array_name, array_size, dest_cpu_id, data_type*)
call receive_data (*array_name, array_size, source_cpu_id, sz_chk, data_type*)
call b_send_data (*array_name, array_size, data_type*)

The variable, *sz_chk* returns the actual size of the data received. This should be the same as the value of *array_size*. The new variable is introduced for debugging purposes. There are a few more calls available, but they are not described here.

Once the sequential source file is used to create two separate source files, namely, the master and the slave files, and all the required calls, explained above, are inserted in these files at the appropriate places, the code is ready for compilation, linking, loading and execution.

3 Parallelization of PROLSQ

The function to be minimized in PROLSQ, the stereochemically restrained least squares refinement method of macromolecular structures [2] is,

$$\begin{aligned}\Phi = &\sum \left(1/\sigma^2(\mathrm{F})\right)\left(\left|\mathrm{F}^{\mathrm{obs}}\right| - \left|\mathrm{F}^{\mathrm{cal}}\right|\right)^2 \\ &+ \sum \left(1/\sigma^2(\mathrm{D})\right)\left(\mathrm{d}^{\mathrm{ideal}} - \mathrm{d}^{\mathrm{model}}\right)^2 \\ &+ \sum\sum \left(1/\sigma^2(\mathrm{P})\right)\left(\mathbf{m}\cdot\mathbf{r} - \mathrm{d}\right)^2 \\ &+ \sum \left(1/\sigma^2(\mathrm{C})\right)\left(\mathrm{V}^{\mathrm{ideal}} - \mathrm{V}^{\mathrm{model}}\right)^2 \\ &+ \sum \left(1/\sigma^4(\mathrm{N})\right)\left(\mathrm{d}^{\mathrm{min}} - \mathrm{d}^{\mathrm{model}}\right)^2 \\ &+ \sum \left(1/\sigma^2(\mathrm{T})\right)\left(\chi^{\mathrm{ideal}} - \chi^{\mathrm{model}}\right)^2 \\ &+ \sum \left(1/\sigma^2(\mathrm{B})\right)\left(\mathrm{B}^{\mathrm{origin}} - \mathrm{B}^{\mathrm{target}}\right)^2 \\ &+ \cdot\ \cdot\ \cdot\ \cdot\ \cdot\ \cdot\ \cdot\ \cdot\ \cdot\ \cdot\end{aligned} \qquad (1)$$

The first term in the expression above is the usual crystallographic term of structure amplitudes, and all the other terms are due to various stereochemical and other restraints, directly incorporated into the optimization algorithm. The summations are over, x-ray structure amplitudes (First term), inter-atomic distances (second term), planar groups and atoms in each plane (third term), chiral volumes (fourth term), non-bonded contact distances including the hydrogen bonds (fifth term), conformational torsion angles (sixth term), individual isotropic thermal vibration parameters (seventh term), etc. The choice of the necessary and sufficient number of restraints to be imposed on the model, relative weights

assigned through the σ values to various classes of restraints, and to various restraints in each class, relative weights of the experimental observations among themselves, and scaling of the experimental (the first) term with respect to all the other terms in the above expression, are not elaborated here. Differentiating Φ with respect to each of the parameters to be optimized, and equating the results to zero, gives rise to the normal equations of the type

$$\mathbf{BP} = \mathbf{Q} \qquad (2)$$

where, **B** is the matrix of normal equations, a symmetric matrix of rank *m* (number of parameters to be optimized), **P** is the column vector of *m* parameter shifts to be estimated, and **Q** is a vector of *m* known coefficients. Typical elements of **B** and **Q** are,

$$B(i,j) = \sum \omega_n \left(\partial Y_n / \partial p_i\right) \left(\partial Y_n / \partial p_j\right) \qquad (3)$$

$$\mathbf{Q}(j) = \sum \omega_n \left(Z_n - Y_n\right) \left(\partial Y_n / \partial p_j\right) \qquad (4)$$

where, Z is one of the observed or ideal quantities, and Y is the corresponding calculated or model parameter in the PROLSQ cost function. The summations are over all the observations. About 97% of the execution time of PROLSQ is taken up by the calculation of structure amplitudes and their derivatives with respect to each of the refinable parameters, and the augmentation of the elements of **B** and **Q**, eventhough only those elements of **B** for which restraints make contributions are computed.

In the first phase of the parallelization of PROLSQ on ANUPAM, only the structure factor part of the code is parallelized [3]. First, all the data pertaining to the atomic parameters and restraints are read by the master processor, model parameters and their derivatives are computed, and the elements of **B** and **Q** are augmented by the results in the main processor itself. Then, atomic parameters and all the other variables necessary for the structure factor computations are transmitted to all the slave processors. The structure amplitude data are read and devided into (*k*+1) groups, where *k* is the number of slave processors used. Any leftovers (at most, *k*-1 observations) are added to the data in the master processor. Then, each group of data is sent to one of the slaves, and the computations are simultaneously carried out in each processor, including the master. The partial sums of the elements of **B** and **Q** , accumulated in each of the slave processors are received in the master processor at the end of the computations, and all the corresponding data are augmented in it to obtain the final values of **B** and **Q**. Finally, the system of linear simultaneous equations is solved to obtain the parameter shifts. The time gain was almost linear (93%) as the number of processors was increased up to eight processors. Beyond this the communication time became quite significant, thus leading to less and less time gain. With the use of vary large data sets for refining large models, the time response is expected to remain linear as more and more processors used.

This method of parallelization is called the data parallelization, as the same set of instructions operate simultaneously on different data sets of the same kind in different processors. Algorithm parallelisation was implemented during the next phase of PROLSQ parallelisation. Atomic parameters and other relevant data are broadcast to all the slave processors. The data corresponding to each of the terms in the expression (1), from the second term onwards, are read in the main processor, and sent to one of the processors. In each slave processor, computations pertaining to only one type of restraints are carried out. Eventhough, the same code resides in all the slave processors, portions of it can be skipped using the cpu ID of the processor, extracted as explained in section 2. This amounts to the algorithm parallelization, as different sets of instructions of the same code are activated in different processors, possibly working on different kinds of data. While the slave processors are busy with the computations, simultaneously carried out on each type of restraints in individual processors, the structure factor data are read in the master processor, and grouped for the sake of data parallelization. Once the slave processors are free, the structure factor data are sent to them, and the computations are carried out as described earlier.

the third phase of parallelisation is currently underway, during which the process of solving the normal equations by the method of conjugate gradients is being parallelized.

4 Prallelization of Fourier Summation

The summation of a three-dimensional Fourier series in space group P1 is taken as an example of the parallelization of a time consuming and frequently used crystallographic calculation on ANUPAM. Breaking up of the three-dimensional series into three one-dimensional series [4], which itself speeds up the calculations quite considerably, and its parallelization are discussed in detail. The use of FFT [5] at this stage , and its parallelization are expected to speed up the calculations still further. The methodology employed while developing a parallel FFT algorithm for a different application [6] has been of great help while carrying out the work described here. The electron density in a crystal structure can be expressed as a three-dimensional Fouried series

$$\rho(xyz) = (1/V) \sum\sum\sum F(hkl) \exp[-2\pi i(hx+ky+lz)] \quad (5)$$

where, the triple summation is over the entire accessible reciprocal space. Under the validity of the Friedel law, one can combine Friedel pairs in the summation above, as a result of which the expression (5) reduces to,

$$\rho(xyz)=\big(F(000)/V\big) + (2/V) \sum\sum\sum (A\cos\varphi + B\sin\varphi) \quad (6)$$

where,

$$\varphi=2\pi(hx + ky + lz)$$
$$F(hkl) = A + iB$$

The triple summation in the expression (6) is over half of the reciprocal space only. As a result, one of the three indices can have only non-negative values. Let l be the index with non-negative values. This choice is purely arbitrary.

If the unit cell is divided into N (= $Nx \cdot N_y \cdot N_z$) grid points, and if there are M sets of unique Fourier coefficients, it takes enormous time to carry out the three-dimensional Fourier summation as expressed in (6). However, it can easily be broken down into three one-dimensional series using the so-called Beever-Lipson factorization [4]. One can write,

$$\cos2\pi(hx+ky+lz) = C_hC_kC_l - S_h S_kC_l - S_hC_kS_l - C_hS_kS_l$$
$$\sin2\pi(hx+ky+lz) = S_hC_kC_l + C_hS_kC_l + C_hC_kS_l - S_h S_kS_l$$

where, $C_h = \cos2\pi hx$, etc., and $S_h = \sin2\pi hx$, etc. Then,

$$A\cos\varphi + B\sin\varphi$$
$$= A(C_hC_kC_l - S_hS_kC_l - S_hC_kS_l - C_hS_kS_l)$$
$$+ B(S_hC_kC_l + C_hS_kC_l + C_hC_kS_l - S_hC_kS_l)$$

$$= (AC_hC_k - AS_hS_k + BS_hC_k + BC_hS_k)C_l$$
$$- (AS_hC_k + AC_hS_k - BC_hC_k + BS_hS_k)S_l$$

$$=\big[(AC_h + BS_h)C_k + (BC_h - AS_h)S_k\big]C_l$$
$$+\big[(BC_h - AS_h)Ck - (AC_h + BS_h)S_k\big]S_l$$

On substituting this result in (6), and using the following definitions, with h as the summation index,

$$P(xkl) = \sum\big[A(hkl) \cos2\pi hx + B(hkl) \sin2\pi hx\big] \quad (7a)$$
$$Q(xkl) = \sum\big[B(hkl) \cos2\pi hx - A(hkl) \sin2\pi hx\big] \quad (7b)$$

the triple summation of (6) reduces to a double summation,

$$= \sum\sum\big[(PC_k + QS_k)C_l + (QC_k - PS_k)S_l\big]$$

with k and l as the double summation indices. Further, using the following definitions, with k as the summation index,

$$U(xyl) = \sum\big[P(xkl) \cos2\pi ky + Q(xkl) \sin2\pi ky\big] \quad (8a)$$
$$W(xyl) = \sum\big[Q(xkl) \cos2\pi ky - P(xkl) \sin2\pi ky\big] \quad (8b)$$

the original triple summation of (6) reduces to a single summation, with l as the summation index, and finally, the Fourier summation can be written as

$$\rho(xyz) = \big(F(000)/V\big) + (2/V) \sum\big[U(xyl) \cos2\pi lz + W(xyl) \sin2\pi lz\big] \quad (9)$$

Thus, the three-dimensional Fourier summation (6) is split into three one-dimensional Fourier summations, (7), (8) and (9). This itself reduces the computation time by about two orders of magnitude, or more. Introduction of FFT at this stage will reduce the execution time still further [5].

The trigonometric functions are precomputed and stored in a table, and the required values are extracted by the standard table lookup procedures. In a sequential run of the program, the first one-dimensional transform is carried out by using (7) on rows of constant k and l indices. Then, the second one-dimensional transform is carried out by using (8) on rows of constant x and l values. Finally, the third one dimensional transform is carried out by using (9) on rows of constant x and y values.

In a MIMD type of parallel computing system with no shared memory, it is important to keep the communication time to the minimum. The grid for electron density calculations is chosen so as to be exactly distributed among the processors used. First, the 3-D data were divided into n planes perpendicular to one of the summation directions, where n is the total number of processors used, including the master. Each of these blocks of data is transferred to one slave processor, retaining one block for the master. All the processors work on the two dimensional blocks of data, and compute the transforms. At the end, the results are transferred to the master. In the next step, the 3-D data, with one of the directions already transformed, is divided equally among all the processors along the second direction of summation. Upon completion of the second transform by all the processors, the data are received again in the master. Finally, the 3-D data, now transformed in two directions, is equally divided among all the processors along the third and final direction of summation, and the

final results of the 3-D fourier map are available in the master processor.

5 Conclusions

Parallelization of the analytical method of computation of structure factors and their derivatives with respect to the refinable parameters, as described here can be incorporated into any other code for the protein structure optimization. The method of algorithm parallelization of stereochemical restraints can be easily extended to the energy based restraints used in some other optimization codes. Knowledge acquired by parallelizing the FFT code for electron density calculation can easily be extended to the inverse Fourier transformation for the computation of structure factors and their derivatives. Efforts to parallelize the method of conjugate gradients, which have already been tried out in one of the medium-range weather forecasting codes [7] , will soon be extended to crystallographic computations. One of the authors of this article has already worked on the parallelization of the molecular dynamics simulations. Parallelization of the code that generates a calculated Fourier map using the atomic positions will soon be taken up. Thus, with the tasks completed so far, and the remaining tasks lined up for the near future, most of the major crystallographic computations can, in principle, be carried out on ANUPAM in the future, by integrating these parallelized modules into the crystallographic packages acquired from other sources. However, this task may be very difficult, because, modifying large packages of highly optimized sequential codes, written by other people is not easy. It would, perhaps be better if all the parallel codes are written as parts of a new package of crystallographic software for ANUPAM.

Acknowledgments

Help received at various stages of this work from our colleagues in Computer Division, Solid State Physics Division and High Pressure Physics Division is gratefully acknowledged. One of the authors (MR) is thankful to M. Rajgopal and M. Vishwas for helping with the manuscript preparation using MS-Word.

References

[1] P. S. Dhekne, K. Ramesh, K. Rajesh, S. M. Mahajan & H.K. Kaura, “ANUPAM Parallel Computer”, in: Supercomputing in Scientific Visualization (Edited by S. M. Mahajan, H. K. Mani, K. Guruvayurappan & P. S. Dhekne), Tata McGraw-Hill Publishing Company, New Delhi, pp.3-12, 1994.

[2] W. A. Hendrickson & J. H. Konnert, “Incorporation of Stereochemical Restraints into Crystallographic Refinement”, in: Computing in Crystallography (Edited by R. Diamond, S. Ramaseshan & K. Venkatesan), Indian Academy of Sciences, Bangalore, pp. 13.01-13.23, 1980.

[3] M. Ramanadham & R. Chidambaram, “Protein Structure Optimization using Parallel Computers”, in: Supercomputing in Scientific Visualization (Edited by S. M. Mahajan, H. K. Mani, K. Guruvayurappan & P. S. Dhekne), Tata McGraw-Hill Publishing Company, New Delhi, pp. 142-150, 1994.

[4] G. H. Stout & L. H. Jensen, X-Ray Structure Determination. A Practical Guide, Macmillan Company, New York, 1968.

[5] A. Immirzi, “ Fast Fourier Transform in Crystallography”, in: Crystallographic Computing Techniques (Edited by F. R. Ahmed), Munksgaard, Copenhagen, pp.399-412, 1976.

[6] B. S. Jagadeesh, R. S. Rao & B. K. Godwal, “Normal and High Pressure Simulations by *ab initio* Molecular Dynamics with Parallel Processors”, in: High Performance Computing (Edited by S. Sahni, V. K. Prasanna & V. P. Bhatkar), Tata McGraw-Hill Publishing Company, New Delhi, pp. 175-180, 1995.

[7] M. Ramanadham, M. Gaurav, B. S. Jagadeesh, Phool Chand, S. R. H. Rizvi & R. K. Bansal, “Statistical Spectral Interpolation: Analysis Code for Medium Range Weather Forecasting”, Presented at the Second International Workshop on Parallel Processing and Supercomputing Applications in Science and Engineering”, ICTP, Trieste, Italy, September 9-27, 1996.