Volume: 6 Issue: 2 Pages: 2244-2252 (2014) ISSN : 0975-0290

A Novel Technique in Software Engineering for Building Scalable Large Parallel Software: Demonstration on Dynamical Core of VARSHA - A Global Circulation Model Software

T. Venkatesh
Associate Professor, Dept. of Computer Science & Engineering, G.C.E, Ramanagaram.
Email: t_venk5@yahoo.co.in
U.N.Sinha
Former Distinguished Scientist, CSIR-NAL /CMMACS, Bangalore- 560017.
Email:uns@nal.res.in

---**ABSTRACT**--
Parallel processing is the only alternative for meeting computational demand of scientific and technological advancement. Yet first few parallelized versions of a large application code- in the present case-a meteorological Global Circulation Model- are not usually optimal or efficient. Large size and complexity of the code cause making changes for efficient parallelization and further validation difficult. The paper presents some novel techniques to enable change of parallelization strategy keeping the correctness of the code under control throughout the modification.

Keywords: **overlap between computation and communication, parallelization strategy, scalable parallel computing, software engineering.**

Date of Submission : August 26, 2014 Date of Acceptance : September 07,2014

1 Introduction

Scientific and technological advances have made ever increasing demands on computational processing power, and, parallel processing has emerged as the only paradigm to meet the challenge, yet, excepting those classes of problems where demands on communication is modest, for example, handling of Monte Carlo simulations or carrying out domain decomposition technique for solving equations resulting from finite discretisation of partial differential equations of mathematical physics where communication is largely confined to near neighbor, the situation has not matured for tightly coupled large scale parallel computation. This is most evident in the context of tightly coupled spectral computing of atmospheric models. Suffice to say that the field of tightly coupled large scale parallel software needs sufficient number of examples for its core substance and standing. The present paper focuses on one such example, that of, efficient parallelization of spectral version of GCM (Global Circulation Model) of Atmospheric motion.

GCM has three components:

(a) Dynamical
(b) Physical
(c) Radiation

Of these three components, Dynamical core is relatively difficult to parallelize efficiently, and the present paper, therefore, concentrates on the dynamical part to address the issue of scalability. The existing situation of parallelization is that the number of nodes for a fixed size of problem saturate quit rapidly. That this limitation can be overcome has been described in [1] where a new strategy has been proposed based on algorithmic considerations. Implementation of the strategy demands a critical overlap between computation and communication which yields dramatic gain in scalability. To achieve this overlap, many techniques of software engineering had to be stretched to a limit, some new tools have to be made and some delicate programming of computation and communication are needed.

In the next sections we first briefly describe the scientific and algorithmic content of the application software, followed by the existing parallelization scheme outlining the source of bottleneck and finally present the details of the new implementation along with sample results.

2 Domain Details

2.1 Brief description of the Problem

The application software whose scalable parallelization is under consideration deals with tightly coupled spectral version of NWP (Numerical Weather Prediction). The present section describes the domain knowledge in a brief but comprehensive format so as to enable a critical examination of the overall strategy and its alternatives.

NWP was pioneered by Richardson, Charney, Rossby, Philips and many others [3-12]. The governing equations describe the evolution of the state of the atmosphere. The initial state is the input data and using these equations final state is computed.

The interval between the initial instant and next instant is determined by the grid resolution and numerical strategy. The process is repeated till numerical and round off errors swamp the validity of the computations. As the vertical extent of atmosphere is much smaller than horizontal extent (50 km against 3000 km) hydrostatic approximation is made, i.e. acceleration in the vertical is ignored which has been a standard approximation for GCM. State of atmosphere is described by the quantities (u, v, w, p, ρ, T, q), where u is the velocity in the horizontal plane along the latitude(East-West); v is the velocity in the horizontal plane along the longitude(South-North); w is the vertical velocity; p is the pressure; ρ is the density; T is the temperature and q is the measure of moisture content and the independent variables are (x, y, z, t); x is measured along latitude, y is measured along longitude, z is the vertical coordinate and t is the time. In view of hydrostatic approximation, z can be replaced by pressure. Instead of z, $\sigma = \frac{p}{p^*}$ is introduced, where p^* is pressure on the surface of the earth which simplifies the computational scheme algorithms. The following equations correspond to VARSHA [2] and caters to operational forecasting. Many other GCMs operational in the world have similar framework and similar complexities.

Law of mass conservation

For air,

$$\frac{\partial}{\partial t}\ln p_* + \overrightarrow{V_H} \,.\, \nabla_H \ln p_* + \nabla_H \,.\, \overrightarrow{V_H} + \frac{\partial \dot{\sigma}}{\partial \sigma} = 0$$
(1)

For moisture,

$$\frac{\partial q}{\partial t} = S \qquad (2)$$

Here S represents the sources and sinks. Source comes from evaporation and sink comes from condensing of water vapour into rain etc. The expression for S comes from phenomenological part of physics.

Horizontal Momentum Equations

$$\frac{d\overrightarrow{V_H}}{dt} = -RT\nabla \ln p_* - \nabla\phi - f\,\overrightarrow{k} \times \overrightarrow{V_H} + \overrightarrow{F} \qquad (3)$$

Where ϕ is the geopotential, $\overrightarrow{F}$ represents dissipative process coming from phenomenological part of physics, f is the Coriolis component representing contribution from the earth's rotation, $\overrightarrow{V_H}$ is the horizontal velocity vector = (u, v), and **k** is unit vector in vertical.

Vertical Momentum Equation

$$\frac{\partial p}{\partial z} = -\rho g$$

Law of Energy Balance

In terms of potential temperature θ ($= \frac{T}{p^{\kappa}}$), $\kappa = (\frac{c_p - c_v}{c_p}$), the relation is:

$$\frac{\partial}{\partial t}\ln\theta = \frac{H}{C_p\, T} \qquad (4)$$

Where H is the heating rate per unit mass, C_p is the specific heat at constant pressure. H comes from 'physics' and includes solar radiation, heating due to latent heat released during rain or formation of ice etc.

Equation of state It is p=ρRT, where p represents pressure, T is the temperature, R is gas constant; it is taken appropriately for air and water vapour and their mixture as the situation demands.

2.2 Computational Strategy

For achieving better accuracy in numerical computation spectral technique is adopted, Moreover velocity variables are recast in terms of divergence and vorticity of horizontal velocity components. The equations for computation are as following:

(i) (a) **The continuity equation** is integrated along σ to yield,

$$\frac{\partial}{\partial t}\ln p_* = -\int_0^1 \left(\nabla \cdot \overrightarrow{V_H} + \overrightarrow{V_H} \,.\, \nabla \ln p_*\right) \partial\sigma$$
(5)

which is approximated in semi-discretised form as

$$\frac{\partial}{\partial t}\ln p_* = -\sum_{k=1}^{K} C_k \Delta_k - \sum_{k=1}^{K} D_k \Delta_k$$
(6)

Where $C_k = \overrightarrow{V_k} . \nabla \ln p_* ;\ D_k = \nabla \cdot \overrightarrow{V_k}$

(b) **The continuity equation for moisture is**

$$\frac{\partial q}{\partial t} = - \vec{V} \cdot \nabla q - \dot{\sigma} \frac{\partial q}{\partial \sigma} + S \qquad (7)$$

(ii) Taking divergence of momentum equation,

$$\frac{\partial D_k}{\partial t} = \frac{1}{a \cos^2\phi}\left(\frac{\partial B_k}{\partial \lambda} - \cos\phi \frac{\partial A_k}{\partial \phi}\right) - \nabla^2(E_k + \phi_k + RT_{0\kappa} \ln p_*) \qquad (8)$$

(iii) Taking curl of momentum equation,

$$\frac{\partial \eta_k}{\partial t} = \frac{-1}{a \cos^2\phi}\left(\frac{\partial A_k}{\partial \lambda} + \cos\phi \frac{\partial B_k}{\partial \phi}\right) \qquad (9)$$

Where

$$A_k = \eta U + \left(\frac{RT}{a}\right)\cos\phi\left(\frac{\partial}{\partial \phi}\ln p_*\right) + \dot{\sigma}\frac{\partial V}{\partial \sigma} - \cos\phi\, F_\phi \qquad (10)$$

$$B_k = \eta V + \left(\frac{RT}{a}\right)\left(\frac{\partial}{\partial \lambda}\ln p_*\right) + \dot{\sigma}\frac{\partial U}{\partial \sigma} - \cos\phi\, F_\lambda \qquad (11)$$

It may be noted that A_k and B_k are nonlinear terms which spectral handling need special consideration because of nonlinearity.

(iv) The thermodynamic equation is

$$\frac{\partial}{\partial t}\ln \theta = \frac{H}{C_p T}$$

Where H is the heating rate per unit mass and θ is the potential temperature. This is rewritten to give the following equation for temperature:

$$\frac{\partial T}{\partial t} = - \vec{V} \cdot \nabla T + \kappa T\left(\frac{\partial}{\partial t} + \vec{V} \cdot \nabla\right)\ln p_* + \frac{H}{C_p} - \pi\, \dot{\sigma}\, \frac{\partial}{\partial \sigma}\left(\frac{T}{\pi}\right)$$
(12)

Where

$T = \pi\, \theta$; $\pi = p^\kappa$; $\kappa = \left(\frac{C_p - C_v}{C_p}\right)$; and ∇ is the horizontal gradient in the system.

Nonlinear terms create serious difficulties as their spectral form does not lend naturally to algorithm due to nonlinearity, thus nonlinear terms are evaluated in physical domain which amounts to changing the variables from spectral domain to physical domain, doing nonlinear operations in physical domain and getting them transformed back to spectral domain for numerical implementation. This technique was first used by Orszag[14] and since then it has become standard.

2.3 Rationale for Evolution of Existing Parallelization Strategy

For the complete problem, the governing equations are (6) to (9) and (12) and the domain of computation is longitude × latitude × height × time. Thus, for time marching of a single step, calculations are carried out in longitude × latitude × height (level) space. Typical values of longitude is 512, that of latitude is 256 and that of level is 16. The dependent variables are represented both in physical and spectral domains – in the spectral domain the wave number J is typically 120. A typical variable F (λ, φ) in spectral domain is represented as

$$F(\lambda, \varphi) = \sum_n \sum_l F_n^l \ P_n^l \ (\sin\phi)\, e^{il\lambda} \qquad (13)$$

where P_n^l is associated Legendre function, one of many special functions in mathematical physics [15], the number of spectral coefficients grow like J^2 .

The strategy of parallelization is dictated by nonlinear term A_k and B_k in equations (8) and (9) and the summation term in equation (6).

Nonlinearity in A_k and B_k necessitate that for every step of evolution of spectral coefficient, one has to go through the physical domain, thereby, all of the latitudes have to be spanned. This makes looping in latitude inevitable. On the other hand summation in equation (5) requires data from all the vertical layers which when computed in a sequential environment suggest that for each latitude, data from vertical layers which when computed in a sequential environment suggest that for each latitude, data from vertical layers are to be considered and are linked together as a basic unit for further processing. This suggested the following domain decomposition:

longitude × *latitude subgroup* × *vertical levels*

and this is what is used in the existing VARSHA software. There is no decomposition in longitude in VARSHA, so, for all practical purposes, the domain of computation is *latitude* × *levels* and domain decomposition takes place in latitude domain. A schematic diagram of the domain decomposition is shown in Fig. 1.

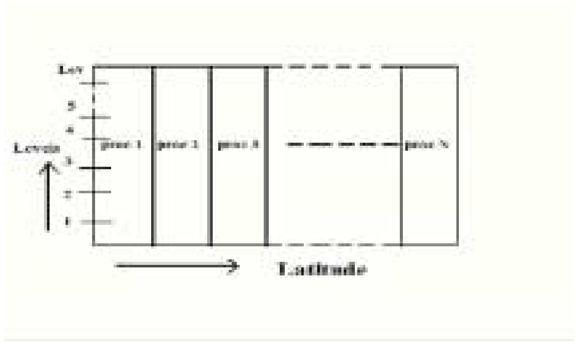


Fig 1: Domain Decomposition in Existing Strategy

For each time step size of calculation in the existing strategy the spectral coefficients of divergence, vorticity, temperature and moisture for all the vertical levels need to be communicated. This amounts to having communication size of $4J^2 levs$ for each processor.

Though, the above mentioned scheme appears natural, it permits no overlap of computation and communication. When a particular latitude group computation is over, the spectral data need to be globally communicated and unless this is done no further computation can proceed.

Volume of communication in view of global communication is best estimated as

$$\approx 4J^2 \times levs \times Nprc \times \ln_2 Nprc \qquad (14)$$

assuming optimal use of connectivity and symmetry which is in use in most of the operational HPC platforms. Here 4 stands for number of spectral variables to be communicated (in this case T, q, D and η; $\ln p_*$ is disregarded for ease of estimate as its size is relatively small). J^2 denotes the size of each spectral variable. The factor $levs$ accounts for vertical coupling-all levels are connected. The factor $Nprc \times \ln_2 Nprc$ needs special mention. It symbolizes tight coupling and global communication. If data of size $4J^2 levs$ need to be only sent to $Nprc$ processors, the volume of communication will be $4J^2 levs \times Nprc$. But the problem demands that intermediate result accruing from this message passing be again globally communicated. Thus, simple minded strategy will give $Nprc \times Nprc$ which is prohibitive when the number of processors is large. Strategy based on tree structure reduces the factor Nprc to $\ln_2 Nprc$. The fundamental essence of scalability is contained in equation (14).

The communication volume is listed below for typical value of Nprc (Number of processors in the parallel computation).

	Existing Strategy
N	Infiniband connectivity $Nprc \times \ln_2 Nprc$
2	2
4	8
8	24
16	64
64	384
128	896
256	2048
512	4608
1024	10240

Table 1: Communication Volume in units of $4J^2 \times levs$

It is clear from the above table that the strategy will be made ineffective by the rapid rise of communication demand and it is, therefore not surprising that as the number of processors grew, the large scale parallelization strategy did not succeed in the earlier experiment of VARSHA. This is the basic issue of the Nonscalibility in the present scheme.

3 Strategy for Scalability in the New Scheme

As reported in[1] the new strategy exploits the following features of the computation scheme:

(a) The vertical extent is small, therefore, per horizontal grid point communication in the vertical direction is one order of magnitude smaller.

(b) The coupling among the layers is further weak, it basically occurs only through equation(5), and terms like $\dot{\sigma} \frac{\partial V}{\partial \sigma}$ in equations (10) and (11), and

(c) Nonlinear part of communication for each latitude is independent of each other,

which results in a group of processors being associated with every level for computing its state for which it has all the necessary data except A_K and B_K as given in (8) and (9). Every processor needs a small chunk of data in form of A_K and B_K from all the remaining processors for computing the interlevel interaction term. To compute such interaction terms a separate processor which will be termed Vertical Integrator is associated. Each processor sends the necessary data to the vertical integrator and gets the processed interaction term from vertical integrator. In view of remark(c) calculation over each latitude is independent

so while vertical Integrator computes the interaction term the calculation of following latitudes can proceed in a normal way.
The domain decomposition for this strategy is given below Fig. 2.

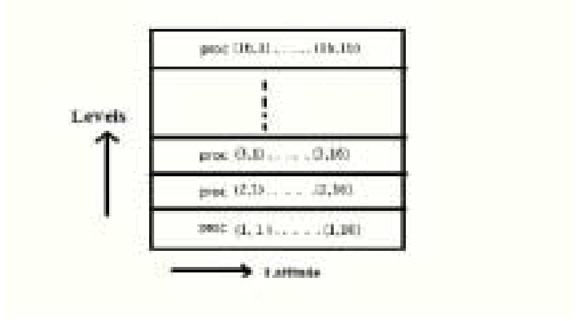


Fig 2: Domain Decomposition in Proposed Strategy (General case)

The communication overhead in the new strategy is given by

$$\textit{Communication overhead} = 4J^2 \left(\frac{Nprc}{Nlevs}\right) \times \ln_2 \left(\frac{Nprc}{Nlevs}\right)$$

It may be remarked that the above strategy critically depends on overlapping of computation and communication in the way described above. The gain in performance is dramatic as shown in the following Table 2.

	Existing strategy	Proposed strategy
Nprc	Infiniband Connectivity $Nprc \times \ln_2 Nprc$	Infiniband Connectivity $\left(\frac{Nprc}{Nlevs^2}\right) \times \ln_2 \left(\frac{Nprc}{Nlevs}\right)$
16	16	0
32	32	0.125
64	64	0.5
128	128	1.50
256	256	4
512	512	10
1240	1240	24

Table 2: Communication Overhead

The task of achieving the overlap is a major Software Engineering effort. In fact in [1] it was stated that one may arguably enquire that if the gain is so phenomenal why was it not discovered earlier. The answer is that this has to do with the complexities of the application code - a large bunch of details clouds the central idea of computation and the built-in biasing of the earlier sequential code. The present paper is centrally focused on Software Engineering aspects of achieving the stated overlap between computation and communication in the selective part of computing terms like A_K and B_K etc which will result in achieving the desired scalability. The fact that differing platform may have different ratios of computing to communication speed an additional parameter quantifying the bulk of computation time to communication time need to be incorporated in the software engineering scheme. The next section describes these developmental strategies.

4 Software Engineering

Before embarking on the software engineering of implementation details, it will be desirable to have a picture of contemporary scenario so that the present effort can be put in proper perspective.

4.1 Contemporary Scenario

In this regard one has a wealth of information in the proceedings of a workshop held at ECMWF (European Center for Medium-Range Weather Forecasts) in 2013[18-19]. ECMWF is a leading meteorological centre in the world and the topics addressed in the workshop are directly related to the present paper: we shall confine ourselves to two of the papers presented in the workshop.
The first one[18] is due to N P Wedi; M Hamrud and G Mozdzynski and the title is, "The ECMWF model: progress and challenges". The authors state: " The (energy-) cost and latency of the parallel communications associated with the grid point-to-spectral-to-grid point transforms and the communications(transpositions) within the spectral computations, have not been addressed in this article and remains a concern. Spectral-to-grid point transformations require data-rich (and energy-inefficient) global communications at every time step that may become too expensive on future parallel computers." The strategy in the present paper is directly linked to the cost and latency of the spectral to grid to spectral transform.
The second paper [19] we refer is due to G Mozdzynski *et al* and the title is 'ECMWF's IFS:"Parallelization and exascale computing challenges". We quote from this article, "The purpose of these optimizations is primarily to allow the overlap of computation and communication ". This aspect is further emphasized in the article by stating" within openMP parallel regions to overlap computation and communication and thereby improve performance and scalability".

At this point it will not be out of place to remark that during early nineties there was a spurt of papers on parallelization of spectral codes. These were based on highly simplified models of GCM and did enjoy scalability. These models did not have terms like of $\dot{\sigma}\frac{\partial V}{\partial \sigma}$ equ (11), and $\frac{\partial q}{\partial t} = S$, moisture transport related etc; and, therefore , such models were not of operational class[16-17]. Therefore, as the ECMWF workshop of 2013 shows scalable parallelization of spectral GCM still poses a serious challenge.

Such difficulties were well experienced and considered at CSIR-NAL/CSIR-4PI and strategies were contemplated to overlap between computation and communication where it hits most, i.e. between spectral to grid point to spectral region of computation where moisture transport, vorticity transport, energy transport in the vertical play a crucial role. The present paper attempts to addresses these issues through implementation of ideas of [1].

4.2 Implementation Strategies

Deadlock of communication and disruptions of computation due to floating point exceptions are the key obstacles in the development/modifications of any large parallel software. The massive size of the code and data put additional burden on the implementation of the program. Therefore the following guidelines were followed to avoid those difficulties.

i. Recasting of the original software to minimal achievable size: This part poses serious problem due to nonlinearities and mixing of various scales in computation,
ii. Evolution of a data structure which provides simplicity of sequential operation and intermix of sequential and parallel computations. During development of software, these sequential calculation provide a yardstick for proper communication and in the end these calculations are discarded from the program; doing dummy communications to ensure that no deadlock will occur and thirdly
iii. Doing partial parallel computation with selected nodes so that earlier dummy computation be replaced in the parallel implementation after correctness of intended data is ascertained in stages.

Therefore, the first step taken was to cut down the size of the program. The domain was reduced by the factor of 16 in the horizontal direction and no change was made in the vertical direction. Because of the nonlinearities in the equations and sensitivities of the data it is delicate to redefine the domain. As the horizontal size is reduced the spectral mode of representation needs to be appropriately reduced. Computational experiments were performed to arrive at a new operational code on a reduced scale; the essential parameters are decrease in longitude by a factor of 4 and similar decrease in latitude by a factor of 4.

4.2.1 Recasting the existing code : a snapshot

The ideas suggested in [1] is not implementable in the existing form of the code. In the existing code the inner loop is on vertical levels and the outer loop is on latitude. In the new strategy the outer loop has to be on levels and the inner loop is on latitudes. It can be modified with modest effort. This has to be done throughout the code but the structure enjoys a common pattern. There have been complexities due to occurrence of transform routine. Sample snapshot from subroutine 'gloopa' is given below which is illustrative:

Snapshot:

Existing code:

```
        Real, pointer :: Di_Fourier
         Real, Dimension (lonf2,lotsyn), Target :: Syn
        Di_Fourier => Syn(1,3*Levs +1)
       CALL SumE(Di , Di_Fourier,Qtt,Levs)
       Lotnfs =(6*Levs+3)*2
       Call Fftlonf(Syn(1,3*Levs +1),Dummy,Lotnfs,1)

               Modified code:
        Real,  pointer :: Di_Fourier
         Real,  Dimension  (lonf2,lotsyn,  BunchS),  Target  ::
Syn
        Di_Fourier => Syn (1, 3*Levs+1, BunchL)
    Do k= Levstart, Levend
      Di_Fourier Syn (1,3*Levs+k,BunchL)
      CALL SumE2 (Di(1,k) ,Di_Fourier,Qtt)
    Enddo
   Do k= Levstart, Levend
       Lotnfs =2
       Di_Phy =>Syn (1,3*Levs+k, BunchL)
       Call Fftlonf (Di_Phy, Dummy, Lotnfs,1)
  Enddo
```

In the existing code routine SumE computes the transform for all the levels passed as Levs in the argument. Similarly call to FFTlonf computes Inverse Fourier Transform for all levels in an optimized way. In the modified code the control is built for isolating each level. This may not be so optimized in the current form but it can be made so by combining various variables rather than levels. Introduction of additional dimension in the modified case for 'Syn' may be noted. This is introduced for making possible overlap of computation and communication.

4.3 Preparation of Code

In view of communication deadlock and floating point exception, a version of code was needed where these difficulties if not totally eliminated should be kept at minimum. To this end, a control structure was designed which has the following features:
(a) Communication pattern envisaged should remain undisturbed.
(b) Intended communication data would be computed in addition, and, comparison will be made to find inadvertent error and thus avoid floating point exception in the main computation.

To incorporate such a scenario the following data type was introduced.

```
Type prc_profile
   Integer :: nprcx
   Integer :: prc_idx
   Integer :: Lev_startx, Lev_endx
   Integer :: Latpair_startx,Latpair_endx
   Integer :: HtagSx,HtagRx
   Integer :: VtagSx,VtagRx
   Integer :: Src_idx,Dest_idx
   Integer :: Src_idVx,Dest_idVx
   Integer :: Hreq_Levelx,Hflag_Levelx
   Integer :: Vreq_Levelx,Vflag_Levelx
End Type prc_profile
Type(prc_profile), Dimension(0:1023)::Task_profile
```

The control structure is indeed simple but effective to control at run time to run full code or part of the code or sequential version in some or all of the nodes selected for parallel computation.

5 Implementation of overlap between computation and communication

This was the vital and perhaps the most delicate part of the project. To implement it first the part of the code where vertical interlinking is needed were isolated and rewritten. This has two parts which is best explained with help of equation (6).

$$\frac{\partial}{\partial t}\ln p_* = -\sum_{k=1}^{K} C_k\Delta_k - \sum_{k=1}^{K} D_k\Delta_k \qquad (6)$$

Thus to compute this term, input from all the levels are required. So after doing the processor specific part of the current latitude communication is initiated and instead of waiting for arrival of data from other processors, calculation for next latitude is initiated and this overlap is exactly needed for enhancing the parallel efficiency. The quantum of additional computation is not known apriori, which is determined by customized profiling routines which were developed as part of the programme. It is clear that to achieve this calculation is set in pipeline mode for the latitudes, the first part is the head part where there is no overlap, the second part is the main bulk where pipelining is carried out and then the last part is the winding up part. Once the last latitude is computed the process of pipelining is complete. The calculation then proceeds for completing the remaining part of the time step calculation.

Start the pipeline: Head Part

```
   .    .
 Do Ibuf=1, 2
        Do Iset=1, Nlat_set
.
         Call top_part_hori
           .
         Call Prep_Send_data_Hori_2_Vert
         .
       Enddo  !Set loop
       .
       Call sender_hori
   Enddo  !Buffer loop
```

Middle part

```
    Do Latpair=Start,End,Gap
         Do Ibuf=1,2
              .
              .
             call Receiver_horiN
             call MPI_Barrier_f()
           Do Iset=1,Nlat_set
             Call Prep_recv_data_Vert_2_Hor
             Call bot_part_hori
              .
             Call top_part_hori
             Call Prep_Send_data_Hori_2_Vert
           Enddo !Set loop
            .
            .
         call sender_hori
       Enddo ! Buffer loop
  Enddo  !Lat loop
```

Tail part

```
   Do Ibuf=1,2
```

```
        .
        .
        call Receiver_horiN
        call MPI_Barrier_f()
      Do Iset=1,Nlat_set
        .
          call Prep_recv_data_Vert_2_Hor
            .
          call bot_part_hori
     Enddo !Set loop
  Enddo !Buffer loop
```

This part of the calculation in schematic form is shown below in Fig 3.

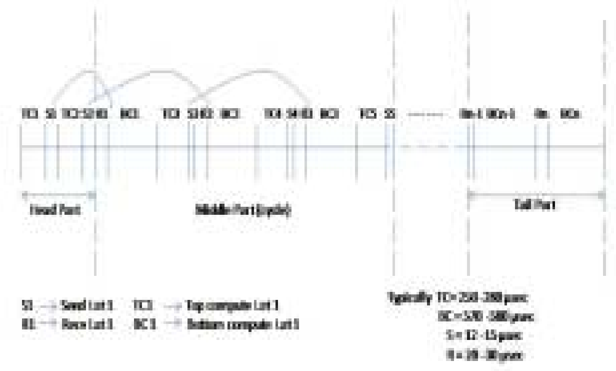
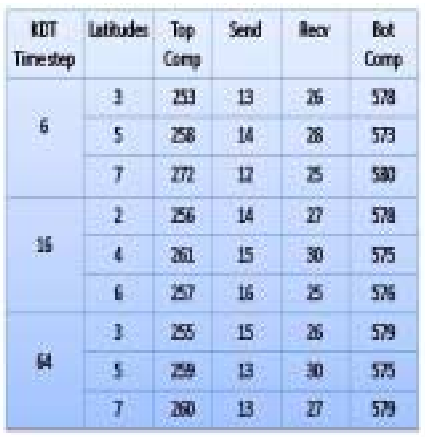

TC1 S1 TC2 S2 R1 BC1 TC3 S3 R2 BC2 TC4 S4 R3 BC3 TC5 S5 ...... Rn-1 BCn-1 Rn BCn
Head Part
Middle Part (cyclic)
Tail Part
S1 → Send Lat 1
R1 → Recv Lat 1
TC1 → Top compute Lat 1
BC 1 → Bottom compute Lat 1
Typically TC= 250-280 μsec
BC= 570-580 μsec
S= 12-15 μsec
R= 28-30 μsec

Figure 3: Temporal picture of communication and computation

Out of 256 latitudes of computation only 4 latitudes of computation has no overlap, which is in tune with the figures shown in Table 2

.

It was earlier feared that communication being a slow process, adequate computational load need to be a parameter in the final implementation. In the final analysis it turned out that the fear was not well founded, time taken for single latitude of computation and corresponding communication were comparable, whose details are given in the Table 3.



KDT Time step	Latitudes	Top Comp	Send	Recv	Bot Comp
6	3	253	13	26	578
	5	258	14	28	573
	7	272	12	25	580
16	2	256	14	27	578
	4	261	15	30	575
	6	257	16	25	576
64	3	255	15	26	579
	5	259	13	30	575
	7	260	13	27	579

Table 3 : Sample timing details in microsec of computation and communication (Illustrative time steps and latitudes have been taken)

6 Results and Discussion

For getting final results, the dimensions were restored to the original form and the key computational experiment where each vertical layer was assigned to a processor and the entire vertical interaction was assigned to a separate processor. In program, they are termed as horizontal processors and vertical integrator, schematically, the configuration and operation scheme takes the form as in Fig 3.

The crux of the matter is in attain overlap between computation and communication for the interlevel part of the dynamical code whose qualitative part is explained in the body of the paper above and quantification is shown in Table 3.

The Fig 3 along with Table 3 shows the temporal behavior of the calculation. The calculation for each latitude has three parts; in the first part calculation is made for a single level, in the second part data from all the levels are assembled and in the third part the calculation for that level is completed.

In the present implementation while communication is being made to get all the data from other levels, additional calculation for the next latitude is done, and when the required data arrives and the third remaining part is completed. This cycle gets repeated. However, to start the process overlap would not be possible and similarly to wind up the process overlap will not be possible. This totally amount to 4 latitudes for 256 latitudes of calculation. Thus the first row of Table. 2 is not equal to zero but rather a small fraction something like 4/256. The Fig. 3 and Table. 3 clearly show the overlap and the code being complex every latitude had different timing, the typical representative timing is shown in Fig 3. Thus, on the whole, the stated objective was attained. Regarding other rows in Table 2, it follows the pattern which is already implemented.

7 Conclusion

Operational softwares are large and difficult to modify particularly in terms of parallelization strategies. The present effort shows in a particular case of VARSHA software how this can be achieved by inserting additional modules such that communication pattern remains unperturbed and floating point exception is avoided by providing space for dummy computation and comparison to avoid the floating exception. Once the code gets transformed to new form, these dummy routines are removed systematically.

The customized profiler assesses the overlap and provides the estimate for quantum of computation for overlap to meet the desired objective. Finally the Fig. 3 and the Table 3 show that the stated objective was realized well and the methodology used in the work may be useful for similar efficient parallelization.

8 Acknowledgement

It is our pleasure to thank Director, NAL and Scientist in-charge CSIR-4PI for their approval, support and encouragement.

References

[1]. T.Venkatesh, U.N.Sinha, "Development of a New Scalable Parallelization Strategy for GCM Varsha and its Historical Perspective", International Journal of Computer Science and Information Technologies, Vol. 5,issue 3,pp 4485-4489,2014.

[2]. U N Sinha *et al,* NMITLI project on mesoscale modelling for monsoon related Predictions, High level document for the new model , prepared by NAL,IISC,TIFR,2003.

[3]. L F Richardson, “Weather Prediction by Numerical Process”, Cambridge University Press, London 1922; Dover, New York, 1965.

[4]. J.G. Charney, “Planetary Fluid Dynamics”, ed. P Morel Dynamic Meteorology, D Reidel publishing company Boston-USA, 1973.

[5]. Rossby, C.-G., J Marine Res.,2,38-55(1939).

[6]. N .A. Phillips, “ Principles of Large Scale Numerical Weather Prediction”, Dynamic Meteorology, ed. P Morel, D Reidel publishing company,DorDrecht-Holland,Boston-USA,1973.

[7]. A Wiin-Nielsen, “the birth of Numerical Weather Prediction”, Geophysical Institute, Tellas 1991.

[8]. N A Phillips, “Models of Weather Prediction”, Annual Review of Fluid Mechanics, Vol.2, pp 251-292, Jan 1970.

[9]. Smagorinsky J. “The beginning of Numerical Weather Prediction and Global Circulation Modeling:”early recollection Adv.Geophy,25,pp 3-37,1983.

[10]. Smagorinsky J, S. Manabe , J L Holloway, “Numerical results from a nine-level Global Circulation Model of the Atmosphere”, Mon. Wea Rev 93,pp 727-768,1965.

[11]. Shamaman, F.G., “History of Numerical Weather Prediction at the National Meteorological Centre”, Wea, Forecasting 4, pp 286-296, 989.

[12]. Sela, J G, “Spectral Modeling at the National Meteorological Centre”, Mon. Wea Rev 108, pp 1279-1292, 1980.

[13]. Akira KasaHara, “Computational Aspects of Numerical Models for Weather Prediction and Climate Simulation”, Methods in computational physics, Vol. 17, ed. Julius chang, Academic press New York, London 1977.

[14]. Orszag S.A., Transform method for calculation of vector-coupled sums: Application to the spectral form of the vorticity equation. Journal of Atmospheric Sciences, Vol 27, 890-895, 1970.

[15]. George E. Andrews, Richard Askey , Ranjan Roy, “Special Functions” , Encyclopedia of Mathematics and its Applications 71, Cambridge University Press, 2010.

[16]. D B. Haidvogel, E Curchitser, Md Iskandarani, R Hughes, "Global Modeling of the Ocean and Atmosphere using the Spectral Element Method", Atmosphere Ocean, pp 505-531, 1997.

[17]. I T Foster, B Toonen, P H Worley, "Performance of Massively Parallel Computers for Spectral Atmospheric Models", Journal of Atmospheric and Ocean Technology, Vol 13, pp 1031-1045,oct 1996.

[18]. N P Wedi, M Hamrud, G Mozdzynski, "The ECMWF model :progress and challenges", Seminar on Recent Development in Numerical Methods for Atmosphere and Ocean Modeling, Sept 2-5, 2013.

[19]. G Mozdzynski, M Hamrud, N P Wedi, "ECMWF's IFS: Parallelization and exascale computing challenges", Seminar on Recent Development in Numerical Methods for Atmosphere and Ocean Modeling, Sept 2-5, 2013.