Design Issues for the Parallelization of an Optimal Interpolation Algorithm

Gregor von Laszewski,[*+] Mike Seablom,[*]
Miloje Makivic,[+] Peter Lyster,[*] Sanjay Ranka[+]

[*]*Laboratory for Atmospheres*
NASA Goddard Space Flight Center
Greenbelt, MD 20771, USA

[+]*Northeast Parallel Architectures*
Center at Syracuse University
Syracuse NY 13210, USA

Contents

Design Issues for the Parallelization of an Optimal Interpolation Algorithm

Gregor von Laszewski,[*+] Mike Seablom,[*]
Miloje Makivic,[+] Peter Lyster,[*] Sanjay Ranka[+]

[*]*Laboratory for Atmospheres*
NASA Goddard Space Flight Center
Greenbelt, MD 20771, USA

[+]*Northeast Parallel Architectures*
Center at Syracuse University
Syracuse NY 13210, USA

ABSTRACT

A regionalized optimal interpolation algorithm is currently used at the NASA Goddard Data Assimilation Office (DAO) for four dimensional data assimilation. Instead of using all observations regions are defined to approximate the solution. The sequential code for the regionalized optimal interpolation is very complex and in its current form unusable for MIMD machines.

This paper describes the efforts at the DAO to parallelize the existing sequential algorithm. We outline three strategies transforming the sequential algorithm gradually to MIMD machines.

A major requirement for the new parallel algorithm is the portability to as many MIMD machines as possible. Therefore, the parallel code uses state of the art programming languages, message passing libraries, and an object oriented tool library. For concrete results we are targeting the IBM SP2, Cray T3D, Intel Paragon, and a network of DEC Alpha workstations. We show preliminary results on a DEC Alpha workstation farm.

1. Introduction

1.1. The NASA Climate Model

The NASA climate model constitutes of independent program modules (shown in Figure 1). First, data observed by satellites, weather balloons, airplains, and other sources are prepared for the model. A quality control check is performed on the data in order to eliminate data of very bad quality. After the quality control the model calculation is performed.[4] Once the model calculation is completed the data from the model is compared with the observational data in order to detect errors due to insufficiencies in the model. Different strategies can be used for this *objective analysis*, .e.g: regionalized optimal interpolation (minivolumes)[1,5] global optimal interpolation,[3] Kalman smoother,[2] and Physical-space Statistical Analysis System (PSAS).[6]

This functional decomposition of the NASA climate model makes it possible to run the different modules on different machines. Naturally one should choose the machine

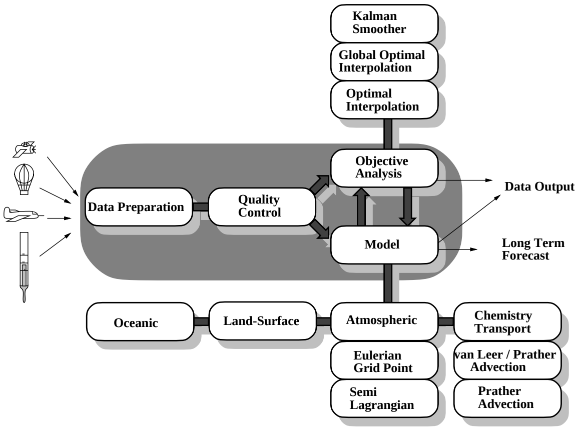

Figure 1: The climate model overview

which is best suited for the problem. In order to enable such a functional decomposition it is necessary to determine the specific input/output requirements of each module as well as the detection of global data, shared between the modules.

This paper concentrates on the efforts to parallelize the regionalized optimal interpolation algorithm currently used in the operational climate model. We show strategies for the functional and data decomposition of the regionalized optimal interpolation algorithm.

First, we introduce the regionalized optimal interpolation algorithm. Than we introduce some constraints for the parallel algorithm. In the main section we explain the parallel algorithm in detail. In the last section we show some preliminary results obtained on a network of DEC Alpha workstations.

2. The Regionalized Optimal Interpolation Algorithm

2.1. Optimal Interpolation

The analysis incorporates a variety of observations such as rawinsonde reports, surface ship observations, and satellite retrievals. At present, one analysis incorporates approximately 100,000 observations; in ten years we expect an increase in that number by two orders of magnitude. Data is interpolated from random locations to a regular grid via the optimum interpolation (OI) analysis technique.[4] OI has the advantage of using statistical estimates to determine appropriate relative weighting between *noisy* observations and a somewhat inaccurate first guess (usually a forecast model) to minimize the resulting error in the analysis.

The optimum interpolation system is computationally elaborate; one analysis consists of a global quality control[5] and a covariance matrix setup and solution. The algorithm requires significant execution time on a state of the art vector supercomputer. Because of the increasing number of observation points and resource limitations on

the currently used vector supercomputer, it will be too expensive to solve the OI's covariance matrix using the global dataset. Instead, we localize the problem by approximating the global analysis from an ensemble of regional analyses. These regional interpolations, referred to as "mini-volumes", are overlapped so that the large scale features of the analyzed quantities are not lost. Each minivolume is associated with exactly one matrix, while each analysis utilizes approximately 12,000 total minivolumes.
The current resolution of the model grid is 2×2.5 degrees. In near future the resolution is doubled so that a grid of 1×1.25 is assumed. Dense grid spacing is from advantage because the precision of the calculation and therefor the error estimation will profit. Because of the proposed dramatic increase in the amount of data ingested, the planed increase in the model grid density, and the availability of MIMD machines with very high performance, the development of a scalable massively parallel version of the analysis is essential.

2.2. The Minivolume Concept

As mentioned above, one way of conducting the interpolation is to treat the correlations between the observations as one big matrix and solve the resulting set of linear equations en gross. We refer to this strategy as *global optimal interpolation.* The disadvantage of this method is that it will take a long time to solve this equations because of it's size. Nevertheless, we expect that the precision for this method is very accurate.

Table 1: Organization and size of the minivolumes

Zone	Volumes	horiz. Grid Points	Vertical Grid Points
Polar	4	288	2
High Latitude	286	8	2
Low Latitude	240	6	2
Tropics	1152	4	2

To decrease the computational effort we try to derive a new approximation method.[5] It is based on the observation that for a correction of guessed values at a location only the observation points in a particular radius around this location have to be considered. Points far away from the location have no or nearly zero correlation. In conventional methods this is done for each grid point. In contrast the algorithm described in[5] uses a cluster of grid points rather than one grid point at a time. Hence, data selection and the solution of the resulting covariance matrix is only necessary for each cluster rather than for each grid point. The reduced observation area is referred to as *mini volume.*
Table 1 shows a typical definition of the number of grid points within each volume dependent on the latitude. At present four volume zones are defined.

2.3. The Sequential Algorithm

The optimal interpolation algorithm uses the following major steps to perform the objective analysis:

```
Read the observational data
Set up the minivolumes
FORALL minivolumes DO
    Set up the covariance matrix
    Solve the linear system constituting of the covariance matrix
    and the forcing function
END FORALL
```

2.4. Covariance Matrix

The covariance matrix is generated in the following way. First, all observational data in a fixed radius around a minivolume is chosen. From this observational data the best 75 are selected, due to the quality of the data determined by the instrument which measures it.
Than the distances between the chosen data points in the minivolume are determined. Using the minivolume one generates the covariance matrix and solves the following linear system:

$$Ax = b \tag{1}$$

$$A = \text{covariance matrix} = \sigma_i^F \sigma_j^F \rho_i j + \sigma_o^2 \delta_{ij} \tag{2}$$

$$b = \text{forcing function} = \rho(d) \sigma_i^F \sigma_g^F \tag{3}$$

Where x represents the weight of the observations.
The main source of slowdown of the current algorithm is based on redundant calculations for the setup of the correlation matrix. One has to consider that two neighboring minivolumes have large overlapping areas resulting in repeated calculation of the correlations between points. To minimize this effort we suggest to store correlations between the selected observational data points temporarily in a hash table instead of recalculating them for each minivolume.
In addition, a simple sort strategy based on the geographical location of the observation points would allow the decomposition of the data points in regions allowing multiple processors to work on different minivolumes. Because of the different size of the covariance matrix (in some regions we expect to have no data points at all) we will have different load on the different processors. This makes it necessary to introduce a dynamical load balancing strategy since we do not know in advance where the observational data will be located (see Figure 2).

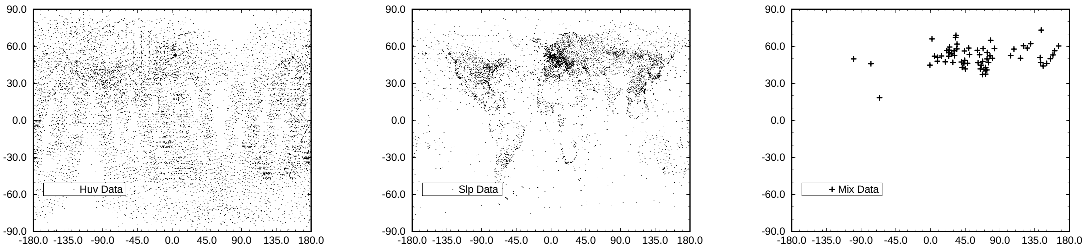

Figure 2: Observation points considered for the analysis. From left to right: wind, surface and moisture data

We clearly determine certain areas where no data values are present. This results in a different calculation load for different areas. This analysis shows that a simple distribution based on the same size of grid points will introduce load imbalance.
The current analysis of the existing sequential code shows the following needs:

- The search algorithm to set up the covariance matrix can be improved by
    1. Reducing the search on a geographical determined subset of observational points for each minivolume.
    2. Sharing the information of the correlations between neighboring minivolumes.
- Statical load balancing can be introduced by predicting the patterns of input data at a given time.
- Dynamical load balancing is necessary because of the different numbers of calculations performed in a regional area.

3. Parallelization of the Optimal Interpolation Algorithm

3.1. Software Engineering Limitations

While parallelizing the existing code we had the following project limitations in mind: First, the program should run on as many MIMD machines as possible. Second, the resulting program should be finished as soon as possible. Third, the language in which the mathematical computations are done is Fortran due to its performance.
One major problem we had is embedded in the structure of the existing code. It is designed for a specific vector supercomputers and because of its long development time and the large number of programmers involved in it's development it is rather unstructured. Since the original code was written in Fortran 77 with vector arithmetic additions it was obvious to use Fortran 90 as language for the mathematical base computations.

3.2. Software Engineering Choices

The parallelization of an existing program is often a very time intense and difficult task. In order to minimize the transformation we tried to develop a strategy to transform the optimal interpolation algorithm gradually into a parallel program.
To make use of the next generation of MIMD supercomputers we have to modify our program in such a way that it will be portable on these machines. In order to do so we decided to base our program on the availability of particular software languages and tools.
To choose a languages and software tools we have to consider their availability (or planed availability), their ease of usage, their flexibility , and their standardization.

Fortran 90 and HPF: For the main program development we chose Fortran 90 over Fortran 77 because of its better software structure capabilities. Another reason is the availability of High Performance Fortran(HPF) on many of the MIMD machines. We are of the opinion it will be possible to transfer certain parts of the Fortran 90 OI program to HPF.

C++: For some of our load balancing and data redistribution tools we chose C++, because data abstraction and the development of these routines is easier to formulate in an object oriented programming language. While following the discussion about the standardization of C++ we are of the opinion, that the incooperation of mathematical data structures in the language will enable a much easier combination of Fortran and C++. Due to the nature of the redistributed data, we do not expect a performance loss while using C++. Interfaces to Fortran and C are easy to provide.

Message Passing: To incooperate message passing in our programs we use the Message Passing Interface (MPI) library. The routines used for message passing are chosen as simple as as possible so that a replacement of the underlying message passing library is easy to do. While choosing this library we are able to use also heterogeneous computing environments.

Portability: With this choices we expect to have a highly portable program which will run on the current and next generation of supercomputers.

3.3. Parallelization Strategy

To focus on the parallelization we have to figure out first in which way data can be used efficiently on different processors. In addition we have to analyze if a functional decomposition is possible. The data objects essential for the parallelization are

- the model data specified by the grid

- the observations
- and the minivolumes.

Figure 3 shows some simple distributions of the data objects. The globe to the left is simply divided into stripes of equal latitude width. In future we will experiment with different width in the latitude bands. To use more processors for the solution of the problem the earth is divided by its longitudes.

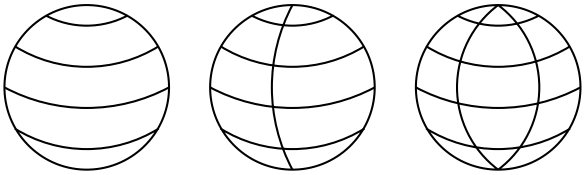

Figure 3: Different data distribution

Now that we have some idea about the distribution we take a closer look in one of the regions (Figure 4). Each Processor is responsible for the calculation of the minivolumes embedded in it's particular region. To do so each processor must have at least the model grid data and the observations embedded in the area labeled by overlap region.

The minivolumes play a separate roll in the parallel program. They can be viewed as tasks executed on the processors. Hence, the concept of data distribution for the minivolumes is not possible. The size of the overlap region is determined by the radius of the minivolumes. In our case we chose the radius to be 1600 km. Two cases occur:

1. All observation data is embedded in the assigned processor
2. Some observation data is embedded in neighboring processors.

To solve this problem, the missing observational data is fetched from the neighboring processors, before a minivolume is solved. This enables one to run the calculations on different processors completely independent from each other with no data exchange. Overlap regions are well known in finite element algorithms described in,[7] but in finite element algorithms the communication is needed.

One limitation of this parallel OI algorithm is that the regions should be not to small in order to keep the ratio between the overlap region and the assigned processor region as small as possible while providing a high degree of parallelism. Future experiments on different data will help us to determine a good parameter setting.

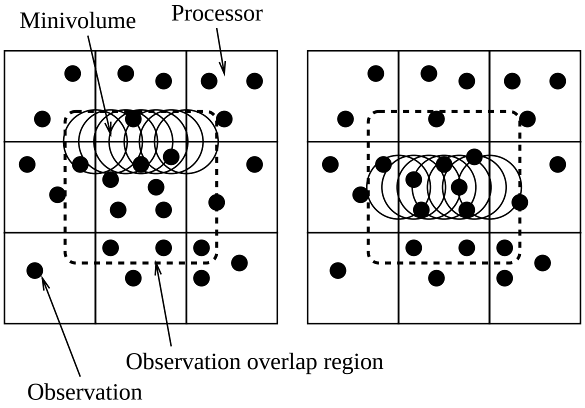

Figure 4: Overlap region

3.4. The Parallelization in Three Steps

Due to the structure of the existing program we decided to do a step by step parallelization determined by the data objects.
It is clear that the parallel algorithm in all cases for the interpolation will look alike, but the data inherited in the processors might be different. The following algorithm specifies the outline of the parallel code executed in each processor:

```
SUBROUTINE Solve
FORALL Minivolumes in this processor
    Collect the data for this minivolume
    FORALL Vertical levels
        Calculate covariance matrix
        Solve the linear equation
        Store the result
    END FORALL
END FORALL
END SUBROUTINE Solve
```

We classify the class of parallel OI (POI) with the help of a tuple space:

$$(\text{grid storage domain}, \text{observation data storage domain}) \qquad (4)$$

We are interested in the following cases:

POI(global grid, global observation) here all grid and observation data are stored in each processor.

POI(global grid, local observation) here all grid data is known to the processors but each processor stores only the observational data which is necessary for the calculation.

POI(local grid, local observation) here all processors store only the data and observation points which are necessary for the calculation.

3.4.1. Storage Pattern (Global Grid, Global Observation)

A straight forward way for a parallelization is to store in each processor all information available. This scheme is well known as data replication. It has the advantage that each processor can work independently on its assigned sub domain. In addition load balancing is easy to implement because of the data replication.
Disadvantages are clearly the high amount of memory used in each processor. Most of the data will not be used during the calculation.
This strategy is impractical due to the performance loss embedded in the data replication and due to limited memory available on the computing nodes. Only small problem instances can be solved.

3.4.2. Storage Pattern (Global Grid, Local Observation)

To reduce the memory usage in each processor we store only the observational data necessary for each sub domain assigned to the particular processor. This can be done with the help of a sorting algorithm prior to the actual optimal interpolation.[7] The following pseudo code enables this:

```
SUBROUTINE read observations
    each processor reads a number of observation data in parallel
    redistribute the observation data depended on the data distribution.
END SUBROUTINE read observations
```

There are different ways to control the distribution of the observation points as shown earlier with the block distributions. In addition we can embedded a recursive bisectioning algorithm in order to provide each processor with an equal amount of observation points. This will help to improve the load balance between the different processors.[8]
The disadvantage of this scheme is that it will have increasing storage needs while increasing the model resolution.

3.4.3. Storage Pattern (Local Grid, Local Observation)

The best algorithm in terms of the memory requirements is to to store in each processor only the data and observation points which are necessary for the calculation. The disadvantage of this algorithm is that a most of the original program has to be rewritten. In addition load balancing will become more difficult due to data redistribution.

4. **Dynamical Load Balancing**

In this part we describe the dynamical load balancing scheme used in the OI algorithm. As mentioned before the minivolume concept generates matrices of different size which are solved by Cholesky factorization. In order to use the resources of an MIMD machine efficiently we generate a dynamical load balancer organizing the task of solving those matrices in parallel.
We use a task queue for the maintenance of the tasks executed on the different machines. Each minivolume is assigned a particular task number. At the beginning each processor starts out with the same amount of minivolumes (tasks).
Now the load balancing algorithm works as follows on each processor:

```
1. Insert minivolumes assigned to this processor in the task queue

2. WHILE (minivolumes there) REPEAT

     (a) IF (minivolumes there AND a Processor is without work) THEN
             Chose some minivolumes and send them to the idle processor
         END IF
     (b) IF (a processor is finished with work) THEN
             Notify other processors that it is idle
             Receive work from another processor
         END IF
         Calculate a minivolume from the task queue

   END REPEAT

3. Terminate processor
```

5. **Results**

5.1. Experimentation Environment

The distributed computing facilities in NPAC include two clusters of high performance workstations: IBM RS/6000 cluster and DEC Alpha cluster. The IBM heterogeneous cluster consists of 8 workstations (models 550 and 340). The DEC Alpha cluster consists of 8 Alpha model 400 compute servers. These workstations represent the newest and most advanced development in the workstation technology. The cluster is supported by a high performance networking backbone consisting of the dedicated, switched FDDI segments. This solution provides full FDDI bandwidth and low latency switching to every workstation in the cluster. In addition to the Workstation clusters the NPAC facilities include an SP2 with 16 nodes.

5.2. Heterogeneous Supercomputing Environment

While combining the different architectures above and solving one concrete problem in parallel on them we will be able to conduct experiments in a heterogeneous supercomputing environment.
For this paper, we present performance numbers obtained with the Dec Alpha workstation cluster with the simple striped distribution with no load balancing on a 4 by 5 grid. We concentrate on the computational intensive part of solving the matrices.

Performance of the Striped Distribution
on Dec Alpha Workstations, Wind Analysis

Processors	Time in s	fastest Processor	Efficiency
1	1015		100 %
4	548	505	46 %
8	283	81	43 %

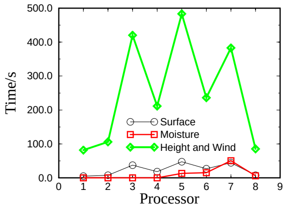

Figure 5: Load imbalance in the simple decomposition

It is very obvious that the load imbalance is responsible for the bad performance (Figure 5). Future experiments will help solving this problem. In addition we are currently conduction experiments with different data distribution schemes.

6. Conclusion

We have shown that we can parallelize an optimal interpolation algorithm. Problematic for the parallelization is the “dusty deck approach”. Besides data distribution we need also the concept of tasks to express the parallelism inherented in the algorithm easily. Load balancing will improve the performance of the parallel algorithm and is used to distribute the tasks on an MIMD machine. Due to the language choice of Fortran 90, C++ and the usage of a common message passing library our algorithm is portable on most of the MIMD supercomputers already available today. The usage of heterogeneous computing nodes is possible because of the load balancing strategy.

7. Acknowledgment

The first Author would like to thank Ricki Rood for his support and the possibility to stay for the duration of the project at NASA Goddard Space Flight Center as part of the Universities Space Research Association (USRA). He is also thankful to Geoffrey C. Fox for his helpful discussions and for enabling the project cooperation.
Additional informations about the project can be found in

http://www.npac.syr.edu/users/gregor/gregor.html

References

1. Baker, W. E., Bloom, S. C., Woollen, J. S., Nestler, M. S., and Brin, E. Experiments with a Three-Dimensional Statistical Objective Analysis Scheme Using FGGE Data. *Monthly Weather Review 115*, 1 (1987).

2. Cohn, S. E., Sivakumarun, N., and Todling, R. Experiments with a Three-Dimensional Statistical Objective Analysis Scheme Using FGGE Data. *Monthly Weather Review* (Dec. 1994), 2838–2867.

3. Daley, R. *Atmospheric Data Analysis.* Canbridge Atmospheric and Space Science Series, Cambridge Univesity Press, 1991.

4. Pfaendtner, J., Rood, R., Schubert, S., Bloom, S., Lamich, D., Seablom, M., and Sienkiewicz, M. The Goddard Global Data Assimilation System: Description and Evaluation. *Submitted to Mon. Wea. Review* (1993).

5. Seablom, M. Experiments with new quality control techniques in the NASA Optimum Interpolation Analysis System. In *Preprints, International Symposium on Assimilation of Observations in Meteorology and Oceanography* (Clermont-Ferrand, France, 1990), vol. WMO. Preprint volume, pp. 628–630.

6. Silva, A. D., Pfaendner, J., Guo, J., Sienkiewicz, M., and Cohn, S. E. Assesing the Efects of Data Selection with DAO's Physical-space Statistical Analysis System. In *International Symposium on Assimilation of Observations, Tokyo, Japan* (March 1995).

7. von Laszewski, G. Issues in Parallel Computing. Tech. Rep. SCCS 577, Northeast Parallel Architectures Center at Syracuse University, Dec. 1993.

8. von Laszewski, G. Object Oriented Recursive Bisection on the CM-5. Tech. Rep. SCCS 476, Northeast Parallel Architectures Center at Syracuse University, April 1993.