Multi-Processing of Weather Models

Bill Groscup
Control Data Corp., Minneapolis

Background

Today I am going to report on an analysis of weather models and their suitability to a multi-processor architecture. The investigation was made two years ago as part of the Flow Model Processor (FMP) project Control Data was conducting for NASA AMES Research Center. The FMP design was based on a requirement for a ten Gigaflop machine for Navier-Stokes problems; basically a numerical wind tunnel. My assignment was to determine how well current Numerical Weather Prediction (NWP) models were suited, not only for the FMP, but for multi-processing in general.

The analysis was divided into four areas of investigation:

- o Discussion with scientists
- o Review of past history
- o Analysis of available codes
- o Methods for multi-processing weather models.

I will review the findings in each of these areas.

Discussion with Scientists

The purpose of the discussions was to determine what the requirements of the meteorological community were in order for multi-processing to be employed in NWP.

A few of the salient items from these discussing were:

1) The modeler would like to avoid multi-tasking within their models if required speed can be obtained through state-of-the-art advances in processor speed, improved compiler optimization, improved mathematical formulation, etc. This same desire was prevalent 10 years ago when they faced the implementation of vector code in their models. However, since then they have adjusted to vector processing and we can expect the same to hold true for parallel processing.

2) The modelers would not like to be required to make changes to their current algorithms. New models and new algorithms may be developed in the future with multi-tasking imbedded, however there is a strong requirement to be able to run existing models in a multi-task environment without significant modification.

3) High-level languages and other tools for multi-processing are desired.

Review of Past History

Two projects were undertaken by the Navy's Fleet Numerical Oceanography Center (FNOC) that utilized multi-processing for an individual model. The first of these was a primative equation model implementation on the ILLIAC IV at NASA AMES Research Center. This project, funded by DARPA, was never successful due to both hardware and compiler problems and the project was eventually abandoned.

The second project was the conversion of a northern hemisphere Primitive Equation grid point model (Kessel & Winninghoff) into an operational four-processor model. At the time, FNOC had two CDC 6500's sharing 1M words of Extended Core Storage (ECS) (Figure 1). The model was partitioned geographically with each of three processors handling 1/3 of the horizontal grid domain (Figure 2) and the fourth processor serving as an output processor. The fields of prognostic and diagnostic variables were assembled in ECS. Model synchronization was maintained by setting flags also in ECS.

The advantages of this implementation were:

1) Very efficient use of the three processors with very little wait time.

2) The model output was available to the user almost as soon as it was produced rather than at the completion of the forecast cycle.

The disadvantages were:

1) A major restructuring of the code was required.

2) Modifications were difficult to check out and implement.

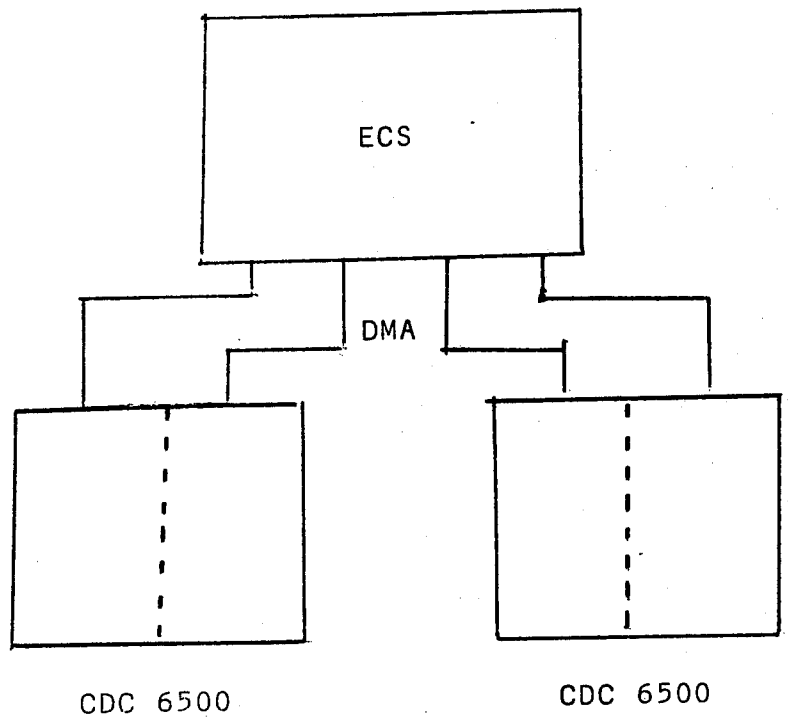
3) A current two processor version had to be maintained in order to allow for failure of one of the CDC 6500's.

While conceptually this method appears quite straight forward, it does not provide for ease of implementation and would not be a viable solution for future multi-tasked NWP models.

Analysis of Available Codes

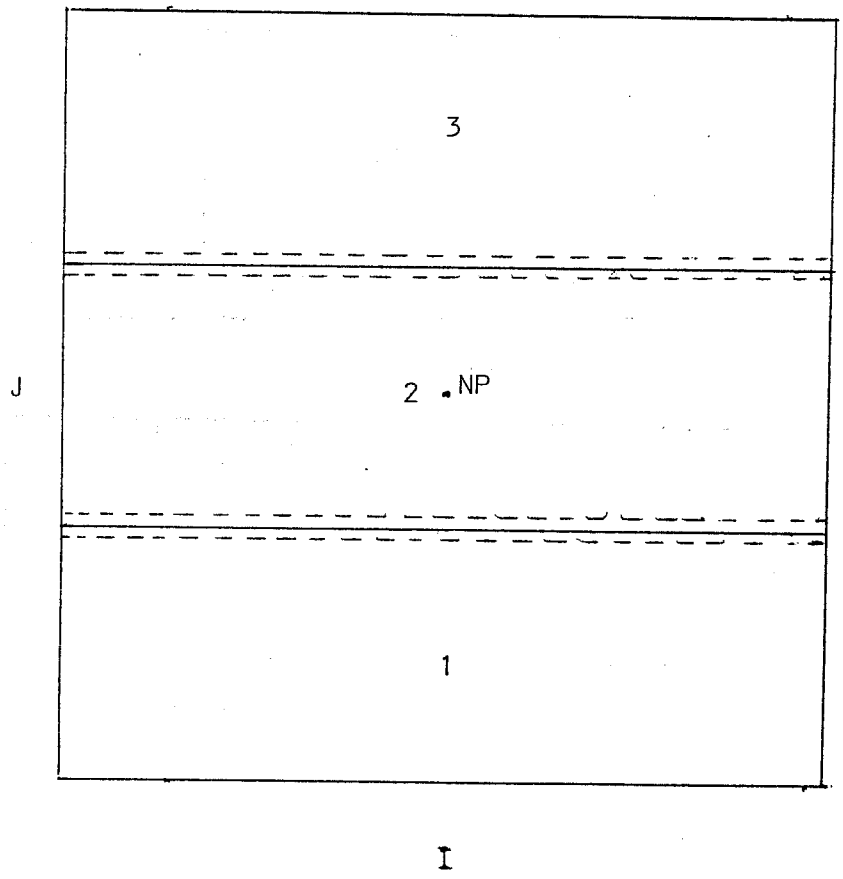
Grid point models from NOAA/GFDL, NASA/GLAS, NCAR and FNOC plus Spectral Models from NOAA/NMC and AFGWC were available for analysis.

Two investigations were undertaken, the first to determine the number of independent processes in the algorithms. If one had a dynamic multi-tasking assignment system with a very large number of processors then the larger the amount of independence the more efficient the code could become.



HARDWARE CONFIGURATION

Figure 1



FORECAST GRID

Figure 2

The second task attempted to determine the optimum design of a multi-tasked program where only a limited number of processors were available and multi-tasking assignments were static.

Without going into detail, the models examined showed that the grid point models fit best in the first category and spectral models in the second. With only a two processor Cray XMP for example, each of the two processors could be assigned a hemisphere in a spectral model and the only wait time would be due to an unequal number of points in grid point space going through physical parameterization schemes.

Methods for Multiprocessing Weather Models

Several methods of incorporating multi-tasking in a weather model were analysed. The one that is most likely to be effective in the early stages of multi-tasking is what I call explicit parallelization where one explicitly "spins-off" portions of the code and data to be processed in additional processors and then waits at synchronization points.

One would hope that a high-level language will be developed for future models. I envision this language to allow one to provide the system a "flow diagram" of their FORTRAN coded model and the system will then partition the model optimally.

In conclusion, after talking to scientists, examining past history, analyzing several weather models and looking at the tools for parallelization of weather models, I feel weather models and multiple processors are well suited for each other. After all the atmosphere is one of the best examples of parallel processes -- one reason it is so difficult to predict.