Guest-editorial

Parallel Software Design for Weather Simulation Codes

In 1922, British physicist Lewis Richardson published the book "Weather Prediction by Numerical Process" which described how to construct numerical models to predict the weather. Unfortunately, the lack of electronic computers made Richardson's techniques impractical, because they required many calculations at discrete grid points mapped over the earth's surface. In 1922, Richardson's approach required 3 weeks of pen-and-pencil calculating for a 24-hour forecast. To overcome this obstacle, he proposed to assign one person to each grid point in his global numerical model, and to have them calculate the weather at each grid point in parallel with all the others. A spherical room would be employed with a platform directly in the center of the room: from this platform a single "conductor" would insure the calculations proceeding in parallel were synchronized by communicating with the persons distributed throughout the room.

Following WWII, the introduction of electronic computers made Richardson's techniques practical, and numerical weather prediction (NWP) was born. Today, NWP is crucial to forecasting both global weather patterns as well as local weather events such as thunderstorms. These techniques have also been extended to forecast climate patterns extending over decades or centuries. One thing that hasn't changed since Richardson's pioneering efforts has been the need for speed: a forecast made after the weather actually happens is usually not very useful.

In addition, faster computers with larger memories allow weather modelers to use finer grids, a technique which improves forecast accuracy. Fine grids are particularly important in resolving and simulating severe weather events, such as hurricanes and the terrible storm of November 4, 1991 over the Grand Banks, referred to as the "Perfect Storm". Without fine grids, these intense, localized phenomena get lost in the large-scale global simulation.

Faster machines also allow modelers to run an ensemble of simulations, each with slightly perturbed initial conditions, to determine the predictability of the weather: if small variations in the input conditions yield large variations in the predicted weather, then the forecaster knows that the weather is inherently less predictable.

This need for speed has led weather modelers to embrace the fastest, most advanced computers, even if they are hard to program. Starting with vector machines in the mid-1970s and extending to the parallel vector processors built by Cray Research in the 1980s, weather modelers have embraced parallelism. They learned how to write vector codes in the 1970s, then in the 1980s extended their vector codes to allow multiprocessing (usually on an auto-tasked outer loop combined with a vector inner loop), and in the 1990s extended or revised their models to run on massively parallel cache-based machines. Through this process they have learned many valuable lessons in how to structure codes to allow large groups of programmers developing the model to work together independently, and about how to structure codes to run efficiently both vector and microprocessor-based parallel machines.

This special issue of the Scientific Programming includes six papers by leaders in the field of parallel numerical weather prediction. The first paper by Michalakes is on the parallel MM5 model used for mesoscale (localized, not global) weather forecasting. MM5 has been constructed to use the same source code for serial, vector, and parallel execution. Instead, Michalakes has developed a source translation tool that recognizes the MM5 code idiom, and makes appropriate translations to exploit domain decomposition for each type of parallel execution, without the use of directives. He also shows MM5 performance achieved across a variety of parallel machines, including vector machines such as the Fujitsu VPP5000 and Cray T90, as well as microprocessor-based MPPs like the SGI O2000 and the IBM SP.

The second paper by Schattler, Doms, and Steppeler describes the recently developed LM (nonhydrostatic,

Scientific Programming 8 (2000) 1–3
ISSN 1058-9244 / $8.00

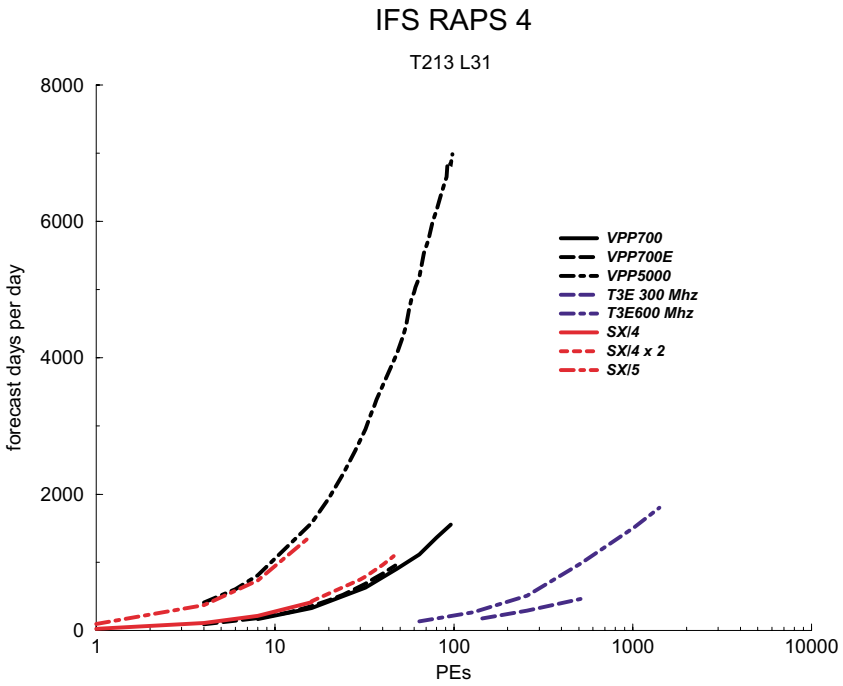

Fig. 1. IFS RAPS 4: T213L31. Courtesy of ECMWF and David Dent.

mesoscale) and GME (global spectral) weather models used by the German Weather Service. These codes emphasize modularity, parallelism, and portability. They are parallelized using domain decomposition, and good speedups were observed for up to a few hundred processors for reasonable grid sizes on the Cray T3E.

The third paper by Desgagne, Thomas, and Valin describes the performance of MC2 (a mesoscale model using semi-implicit techniques) and ECMWF's IFS (one of the oldest and best-known global spectral weather models) on the Fujitsu VPP700 and NEC SX-4M. Good scalability is observed for both models on both machines, though memory contention within a single SX-4 multiprocessor machine limits the good scalability within an SX-4 box to about 4 processors. The authors developed a mathematical model based upon the premise that lack of memory bandwidth limited scalability, and show that their performance contention model correlates well with observed performance.

The remaining three papers describe parallel global spectral models: the NASA parallel AGCM (Schaffer and Suarez), the GFDL SKYHI model (Hemler), and the Navy Global Atmospheric Prediction System (Rosmond). The latter two codes exploit parallelism in one-dimension, while the NASA model (and the IFS model mentioned in the Desgagne paper) exploit parallelism in both the latitude and longitude dimensions. Reasonably good scalability is achieved up to about 64 processors for both the Navy and SKYHI models, while the NASA AGCM showed reasonable scalability up to 256 processors. The single-processor performance on the T3E was considered low relative to the performance that could be achieved on vector machines.

In Fig. 1, the most recent performance results are given for the IFS code (a global spectral weather model described in the paper by Degagne et al.) on a variety of parallel machines. This figure makes manifest the complete dominance of the Japanese parallel vector supercomputers for weather prediction: only 12 Fujitsu VPP5000 processors are needed to exceed the performance of the IFS code on 1024 processors of a Cray T3E.

This implies a per processor difference in performance of nearly 100-to-1; Desgagne et al. report performance discrepancies of nearly 40-to-1 between on the MC2 code between the SGI Origin2000 and the NEC SX-5. And yet these Japanese machines (the Fujitsu VPP 5000 and the NEC SX-5) are full-fledged parallel processing machines that can scale to as many as 512 processors. What this means in practice is that in the field of weather prediction, the lucky owners of a Japanese supercomputer can easily out-compute most microprocessor-based massively parallel processors by a factor of 50 or more. Those who argue the Amer-

ican ban on the purchase of parallel Japanese vector supercomputers should consider these facts.

As a group, these papers provide a good survey of the current state-of-the-art not only in parallel numerical weather prediction, but on the state-of-the-art in the entire field of large (>100,000 lines of source code) parallel applications. I am not aware of any other simulation application in which parallelism is so widely used, or used to such good effect. What is also interesting is that none of the codes described in this issue used a new parallel programming language or smart compiler technology to achieve parallel execution. In each case, good old-fashioned programmer ingenuity developed through hard lessons learned, from multiple implementations of the same code across multiple parallel platforms, carried the day.

Matthew O'Keefe
Department of Electrical and Computer Engineering
Pretty Cool Software Laboratory
University of Minnesota
Minneapolis, MN 55455
USA