Numerical Weather Prediction on the Supercomputer Toolkit

Pinhas Alpert[1], Alexander Goikhman[2,⋆], Jacob Katzenelson[2], and Marina Tsidulko[1]

[1] Dept. of Geophysics and Planetary Sciences, Tel-Aviv University, Tel Aviv 69978, Israel
[2] Dept. of Electrical Engineering, Technion – Israel Inst. of Technology, Haifa 32000, Israel

Abstract. The Supercomputer Toolkit constructs parallel computation networks by connecting processor modules. These connections are set by the user prior to a run and are static during the run. The Technion's Toolkit prototype was used to run a simplified version of the PSU/NCAR MM5 mesoscale model [9]. Each processor is assigned columns of the grid points of a square in the (x,y) space. When $n \times n$ columns are assigned to each processor its computation time is proportional to n^2 and its communication time to n. Since the Toolkit's network computes in parallel and communicates in parallel, then, for a given n, *the total time is independent of the size of the two dimensional array or the area over which the weather prediction takes place.* A mesoscale forecast over the eastern Mediterranean was run and measured; it suggests that were the Toolkit constructed from ALPHA processors, 10 processors would do a 36 h prediction in only about 13 minutes. A 36 hours prediction with full physics for the whole earth will require 2 hours for 80 ALPHA processors.

1 Introduction

This is a joint work of two groups: a group that developed the prototype of the Supercomputer Toolkit and a group that studied numerical weather prediction. We started from the question "How many processors are required to predict the weather over the east Mediterranean?" We ended with an estimate for the number of processors to predict the weather over the whole earth. Our cooperation began once we realized that there is a match between the computational structure of a domain decomposition numerical weather prediction scheme and the architecture of a Supercomputer Toolkit computational network. It is proper, therefore, that we start the introduction with this match. Moreover, our main results, in particular, time and performance estimations, follow readily from this match and from the performance of the Toolkit processor.

The Supercomputer Toolkit [2] constructs special purpose computers by interconnecting standard modules. The main module is a processor (including

⋆ *Current address:* IBM Research Lab, Haifa, Israel

H. Zima et al. (Eds.): ISHPC 2002, LNCS 2327, pp. 335–345, 2002.

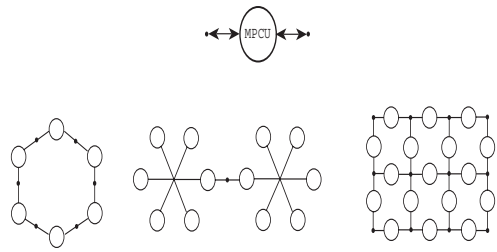

Fig. 1. The abstraction of the Toolkit processor and its interconnection graphs. Each Toolkit processor is a memory-CPU (MCPU) unit with two bidirectional I/O ports. Any graph can be formed as long as the cardinality of each node is less than 8. A ring, a communication cluster and the covering of 2D space are illustrated.

CPU and memory) that has two I/O ports. Processors modules are interconnected to networks by physical connection of ports. Figure 1 depicts a graphical abstraction of such a processor and some networks generated by interconnection of ports. Processors whose ports are connected together are called *neighbors*; neighbors can exchange information between them through the connected ports.

Processors of such a computational network can compute in parallel. Moreover, a processor can read/write through its left- and right-port simultaneously. Thus, considering the ring structure of Fig. 1, a processor can send information to its left neighbor while at the same time read information sent from its right neighbor; each of its neighbors can do the dual action with its own neighbors. In other words, the network's processors can communicate in parallel as well as compute in parallel.

Consider the numerical weather prediction problem. As is well known, weather prediction is a $3D$ partial differential equation problem. The height z corresponds to the atmospheric height and is often taken as 15-20km. The sizes of the other two dimensions, call them x and y, depend on the area to be covered and it can range from 50-5000 km, for mesoscale modeling. A grid is defined over the $3D$ space. Typically, the height is sampled (unevenly) 30-50 times; the x and y dimensions are sampled evenly each 10 to 60 km, depending on the area and the detail to be covered.

The partial derivatives with respect to x, y and z are replaced by differences resulting in ordinary differential equations in time with the state variables being the values of the pressure, velocity, etc., at the grid points. These equations are integrated numerically in what can be considered as a mixed method: The grid points along the z direction at each given x_i, y_j form a *column* that is considered a 'cell'. A time increment Δt is chosen to satisfy the CFL criterion[1].

[1] Courant-Friedrichs-Levy stability criterion; see, for example, [10] p.442, with respect to the speed of sound (3 sec per km) and with respect to the x,y distance between grid points.

Each cell calculates the value of the state at each of its vertical sample points at time $t+\Delta t$ using an *implicit* integration formula taking into account the states of the neighboring cells at time t as constants. Finally, the result obtained at $t + \Delta t$ is declared as the state for $t + \Delta t$ and the process repeats for the next time increment.

Since the speed of sound, which together with the grid step determines the integration time step, does not change significantly with the weather, the CFL criterion implies a constant bound on the time step size and (when the grid is fixed in space) the above step can be performed by any cell independently of time or place[2].

Thus, the $3D$ weather prediction problem has the following computational structure: it is a *two* dimensional array of cells in which each cell communicates with its neighbors only and calculates the next state from its previous one plus the states of all its neighbors[3].

If we assign each cell a Toolkit processor and connect the processors to a grid that covers the $2D$ region (see the right most part of Fig. 1), then the computation can be performed by each processor doing the computation of a cell and, once values are computed, the processor exchanges the results with its neighbors (only!) and starts working on the computation for the next time step.

It is easy to see that if we partition the $2D$ space to regions, say squares, each containing $n \times n$ cells, the property that a cell communicates with its neighbors only is preserved or, inherited by the region. I.e., the processor with the cells of a region has to communicate only with its neighbors for the state of the cells that are on the region's boundary. Even on this level of abstraction one can deduce two major properties of the above Toolkit's computation network, i.e., a network for which the $2D$ space is partitioned to squares and each processor contains $n \times n$ cells: (a) Since the computation is done in parallel and the computing time of each processor is a function of the number of cells 'residing' in it, *the total network computation-time does not increase with the size of the network.* (b) Since the communication is done in parallel and the communication-time of each processor is proportional to n, *the total network communication-time does not increase with the size of the network.*

Section 2 describes the Toolkit in somewhat more detail; it explains how the implementation supports (a) and (b) above. Section 3 describes the MM5 model that we ported to the Toolkit and whose performance we measured. The actual porting of the MM5 program to the Toolkit appears in section 4. The experimental results relevant to prediction over the East Mediterranean and over the complete earth is described in section 5. The last section compares the Toolkit with numerical weather prediction programs running on other computing systems and summarizes our conclusions.

---

[2] Actually, to reduce the truncation error the Δt derived from the CFL criterion and the x, y grid step is divided by some integer n, typically 4, and $\frac{\Delta t}{n}$ is used as the integration time step. In our program results were accumulated each Δt only.

[3] One might call MM5 a $2\frac{1}{2}D$ problem.

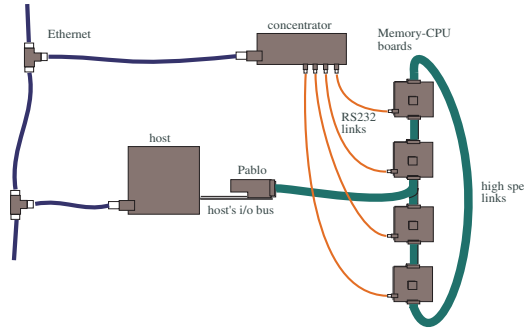

Fig. 2. A general description of the Technion's Toolkit system and its modules. The host is a conventional UNIX workstation. Four memory-CPU modules are shown; they are connected in a ring via wide-band links. The Pablo is an standard extension card connecting one of the memory-CPU modules (called the master) with a standard connector of the host. Each memory-CPU module has a relatively slow connection with the host through a RS232 link that connects to a concentrator and the Ethernet.

2 The Supercomputer Toolkit

Introduction: The Supercomputer Toolkit is a family of hardware modules (processors, memory, interconnect, and input-output devices) and a collection of software modules (compilers, simulators, scientific libraries, and high-level front ends) from which high-performance special-purpose computers can be easily configured and programmed. Although there are many examples of special-purpose computers, see [5], the Toolkit approach is distinguished by constructing these machines from standard, reusable parts. These parts are combined by the user; especially, the connections are set by the user prior to a run and remain static throughout the run. Thus, the Toolkit is a general method for building special-purpose computers for heavy scientific/engineering computing. The following is a brief description of the Toolkit system. It presents the minimum information for an application programmer. The Technion's Toolkit description appears in [2].

Description: Figure 2 is a general description of the Technion's Toolkit system. The system consists of a host, which is a conventional workstation, Toolkit processor boards (memory-CPU boards; four in the figure) and communication components. Fast communication links connect the Toolkit boards. These links are user connected ribbon cables. The fast links are used for communication at computing time. The user uses the host for program development, for loading the network of Toolkit boards, for starting the system and for collecting the results. The memory-CPU board contains an Intel i860 processor (60 advertised Mflops double precision arithmetics), memory, a special communication controller and two ports. The Toolkit processor (board) is considered as a device with two ports. Processors can be connected by connecting their ports together. Figure 1 illustrates such connections. Thus, arbitrary graphs can be formed where processors are the *branches* and the *nodes* are the actual connections.

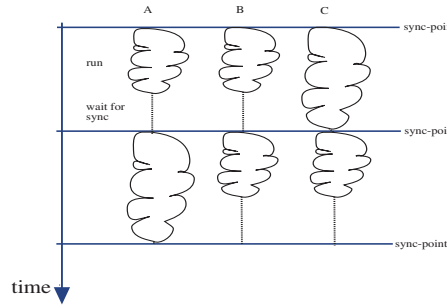

Fig. 3. Programming model – the user's image of the Toolkit processes is illustrated with three processors A, B and C. The synchronization implies a programming model that features *asynchronous* programs, between *synchronization barriers* in which the communication among processors takes place, i.e. *sync-point* in the graph.

Communication: The connections, or the fast links of Figure 2, use a communication method modified from the MIT Toolkit [1][2]. For communication, processors synchronize their instructions (using a synchronization line, see [1][2]) and deliver/accept blocks whose sizes are known *a priori*. This yields a fast transfer rate between neighboring processors – 0.5 giga bit per second per port. The maximum information rate that this method can deliver is half the memory speed per port as there are two ports that operate simultaneously sharing the same bus and the same memory. The synchronization is needed even when running the same programs on all processors since the i860 arithmetic unit timing is data dependent.

The above synchronization implies a simple yet powerful programming model that features *asynchronous* programs, written in, say, C or FORTRAN, between *synchronization barriers* in which the communication among processors takes place. This model of computation is illustrated in Figure 3. According to this model, each processor program looks like a 'normal' program in a high level language that has calls to synchronization routines embedded within it. The calls in different processors have to correspond. For example, to implement the first synchronization barrier (sync-point) each processor program has to have a procedure-like call at an appropriate place. If, for example, processor A has to send information to its neighbor, processor B, at a certain point, then processor B must have a call accepting the information at the same point. This scheme should be repeated for all sync-points and all processor programs.

Referring to standard computer architecture terminology the Toolkit network forms a distributed memory computer; each processor accesses its own memory only; the Toolkit is certainly a MIMD (multiple instruction multiple data) machine; it is not inherently a VLIW (very large instruction word) machine even if it can run as such. Its processor is conventional while its communication method is unique and attains extreme speed for communication between neighbors due to synchronization and the use of properly terminated ribbon cables.

3 A Brief Description of the MM5 Model

The MM5 mesoscale model was developed at the National Center for Atmospheric Research (NCAR) and Penn State University (PSU). The model is based on the integration of primitive hydrodynamic equations [7] and was originally developed for CRAY computers as a FORTRAN program. The MM5 is the fifth-generation model; it is the latest in a series of mesoscale models developed in NCAR starting from the early 70's [7]. A complete model system consists of the preprocessing programs as well as the calculation routines. The MM5 model is the first non-hydrostatic model in the NCAR series, but has also the hydrostatic option [9]. Second-order centered differences are used for spatial finite differencing for most equations. A fourth order scheme is applied for the diffusion terms. A second-order leapfrog time-step scheme is used for temporal finite differencing. The fast terms in the momentum equation that are responsible for sound waves have to be calculated on a shorter time step. The horizontal grid is an Arakawa–Lamb B grid, where the velocity variables are staggered with respect to the scalar variables [8]. A large number of physical parameterization schemes is used in the MM5 model. However, for simplicity, in the runs presented here, only the dry model without the boundary layer physics is employed.

4 Porting MM5 to the Toolkit

General: The code of the MM5 model was modified in order to adapt it to the Toolkit. This modification was done in a semi-automatic way: macros defined the partition of the grid of the domain into symbolic sub-domains. Loops along the domain were identified by hand and their ranges changed according to the sub-domains. The resulting program and a sub-domain index were submitted to a macro processor that yielded the program to be run on the corresponding Toolkit processor. During a time step of integration each processor calculates over its own sub-domain, taking into account at most two auxiliary rows (the number of rows depends on the variable; only one of them requires two auxiliary rows; all the rest require one row only) of cells attached to each sub-domain along its boundary whose values are taken from its neighboring sub-domains. Although for most finite difference algorithms of the model only one neighboring row of cells (in each direction) is required, here, two neighboring rows were necessary because of the fourth-order diffusion scheme.

After the state at time $t + \Delta t$ is calculated, the local communication phase takes place; i.e., the communication among the processors is performed. Processors that were assigned neighboring sub-domains are directly connected to each other (consider again Fig. 1 for covering a 2D space by a network of Toolkit processors). Since the MM5 system with full physics is a very large system, in order to try it out, we removed some options such as the handling of humidity in the air (including convection and micro-physics) and the planetary boundary layer. We had some code modified because the MM5 was written to run on the CRAY with a large amount of memory, and with a dialect of Fortran that we did not have.

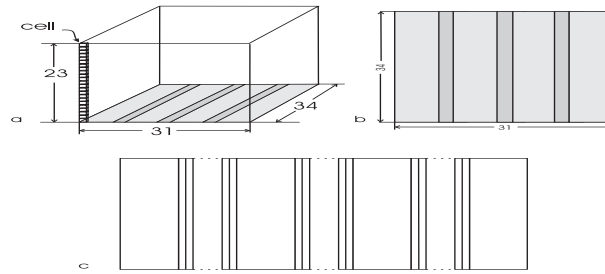

Fig. 4. Sub-domains and data communication in the Toolkit simulation of the Antalya cyclogenesis, (a) The model structure where the cell represents a column of grid-points in meteorological nomenclature. (b) The horizontal model domain and its four sub-domains; The dark shading area consists of four rows of cells. The left two rows belongs to the left domain but their state is used by the right domain; the right two rows belong to the right domain and their state is used by the left domain. The total number of cells is 31x34. (c) Parallelization: two rows of cells, shadow cells, are added to each side of sub-domain to hold the state of the two external rows of the adjacent sub-domain.

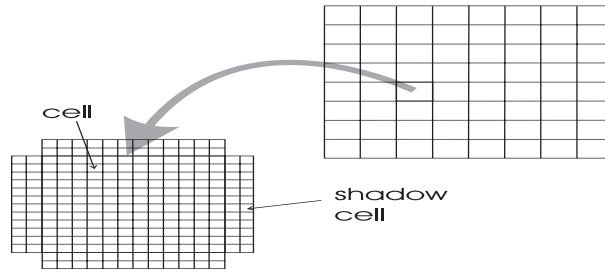

Fig. 5. The covering of a square area by square sub-domains and the cells and shadow cells of a sub-domain.

The case of the Antalya shallow lee cyclogenesis described by [6], was chosen for the Toolkit run. Figure 4 illustrates the overlapping of the sub-domains and the data communication in this particular run. Figure 5 illustrates the covering of a $2D$ space and the shadow cells.

Simulation: The simplified MM5 program was flow analyzed by hand. We found a fair amount of dead code, and a generous amount of arrays used as storage for intermediate results. The dead code was eliminated but for lack of time we did nothing about the intermediate arrays. This extra storage was a problem: the example for which we prepared the initial data had a grid of $31 \times 34 \times 23$ (the last number is the number of vertical levels) partitioned between four processors ($8 \times 34 \times 23$, $8 \times 34 \times 23$, $8 \times 34 \times 23$, $7 \times 34 \times 23$, these numbers are without the overlapping), see Fig. 5.When the code was ready we found that it required about 25% more memory than we had on a Toolkit processor. Since re-writing the code to reduce its volume significantly is a considerable amount of work, we decided to take advantage of the computational structure of the MM5 and

get an estimate of the computation time and communication time by running the code on simulators. Thus, instead of re-writing the code, it was decided to run the resulting program on our simulators[4]. The run of the i860 instruction level simulator resulted in the time measurement for running the problem on the current Toolkit. Running the process level simulator (Fortran source) gave a good indication of the amount of time that is required to run the problem on an ALPHA 200 (166 MHz) based Toolkit.

5 MM5 over the East Mediterranean

To get insight into the ability of the Toolkit to run the MM5 we return to the following question: How many processors are required to predict the weather over the east Mediterranean?

The model domain for the 'East Mediterranean' is taken to be an area of 3480km by 2700km, i.e. 59×46 grid-points with 60 km grid interval. The vertical domain is about 16 km, i.e. top-pressure of 100 hPa, with highly-nonuniform spacing (50 meter near ground and 1 to 2 km at the top; total of 23 levels). The forecasting time was taken to be 36 hours.

Using the instruction level simulator, we found that each Toolkit processor (33MHz clock) did one problem-cycle (i.e., calculating the state at $t + \Delta t$ from the state at t), with 272 (i.e., 34×8) columns, in 14.3 second; running one column for one cycle required 52.6 mili-second. The amount of communication time was about 2% of the total time and in the sequel it is ignored. According to the CFL criterion, the number of problem-cycles for 36 hours is:

$$\frac{T}{\Delta t} \approx \frac{T * c}{\Delta x} = \frac{36 * 3600}{60 * 3} = 720$$

where T and Δx are the time interval and the horizontal grid distance, respectively and c is the speed of sound. The time to predict the weather for 36 hours is: $720 * 14.3 = 2h51$ minutes. If one can use processors with 272 columns per processor to cover the $59 \times 46 = 2714$ grid points needed for this problem, one needs approximately 10 Toolkit processors.

The process-level simulator was then run with the same structure as described before. Since this was an ALPHA 200 processor running native code generated

[4] We use two simulators, [3]: The first simulator simulates Toolkit system processes by running corresponding host processes. For example, to simulate a Fortran program running on, say, three Toolkit processors, the program is translated using the host's Fortran translator and run, by the simulator, as three host processes that communicate via send/receive, etc. This simulator enables finding bugs in the programs and, most importantly, in the communication between the processes. The second simulator is an i860 instruction-level simulator extended to include the communication system as well as to run a Toolkit system, i.e., several processors and their communication. The simulator includes the timing of instructions and the timing of the communication. It is the kind of simulator used to debug kernel code and we judge the accuracy of its timing to be pretty good.

from Fortran, the measured performance can be considered as the amount of time the problem would have taken if the Toolkit were constructed from ALPHA processors. The time for one cycle, with 272 columns, was found to be 1.06 seconds. That means that with an ALPHA 200 based system, 10 processors using 272 column each will do the prediction in $720 * 1.06 = 763$ seconds or 12 minutes and 43 seconds. We believe that the above are conservative estimates and that the program could be speeded-up considerably. We attribute the fifteen fold speed up of the ALPHA to a faster processor (166 MHz clock versus 33 MHz clock) and to a better compiler.

6 Summary and Conclusions

The Toolkit's computation network most significant qualitative property relevant to this application is that the Toolkit's computation network computes in parallel and communicates in parallel; with equal number of cells per processor computation time, communication time and the *total computation time are independent of the size of the two dimensional array or the area over which the weather prediction takes place.*

The dependency of the number of computers on the distance of the XY grid is another qualitative property. Assume that the domain area is given, and the grid distance is divided by two. Clearly, the number of columns is increased four fold. From the CFL condition follows that the integration step, Δt, has to be halved. Thus, if the number of cells per processor is kept and the number of processors is increased fourfold, the total time is doubled. If it is assumed that the time per step is linear with the number of cells per processor (which is not quite so, see below), to achieve the same run time the number of processors has to be increased by eight. This is a property of the problem expressed in terms of processor numbers.

Some detailed qualitative properties are the comparison of the Toolkit system with other parallel processing systems suitable for running the MM5 [12] [14] [13] and the role of the microprocessor cache in such computation [4]. This discussion in not included here for lack of space.

Conservative estimates performed for a typical mesoscale 36 hour forecast over the eastern Mediterranean suggest that were the Toolkit constructed out of ALPHA processors, 10 processors would do the prediction in only about 13 minutes.

This estimate can be extended to a global General Circulation Model (GCM) run covering the whole earth that is about 40 times larger then our eastern Mediterranean model region. Hence, a Toolkit system with 80 ALPHA processors will do a global mesoscale (i.e. 60 km grid interval and 23 vertical levels) 36 hours prediction within about 1 hour only. Another factor of 2 in the computing time is required in order to account for the full physics that was not incorporated in our MM5 Toolkit experiments. The factor of 2 was determined by independent simulations with and without the full physics on other computers (CRAY and SP2) using the same structure as on the Toolkit. In other words, 80 ALPHA

processors will do a prediction over the whole globe in about 2 hours with a resolution comparable to the finest grid achievable today with GCMs in the largest operational weather prediction centers. The Toolkit application performed here uses the MM5 explicit schemes in the horizontal plane.

In summary, the unique property of the Toolkit that computes and communicates in parallel with communication time being negligible, indicates thst the long sought goal of numerical weather prediction with high resolution depends only on the number of available processors with a proper communication scheme and that the number of processors is not that large.

Acknowledgments

The authors thank Professors H. Abelson and G. J. Sussman, MIT, for supportive listening, encouraging and criticism. Professor A. Chorin, UC Berkeley, for the inspiration to tackle partial differential equation. Dr. Gerald N. Shapiro, Fire*star Consulting Co., for a hand with the hardware work and many helpful discussions. Dr Reuven Granot, Israel Department of Defense, for bringing the weather prediction problems to our (JK) attention.

This work was supported by Maf'at, Israel Ministry of Defense, and by the Israel Ministry of Science and Technology. Seed money for this work was given by Intel Corp. We thank them all.

We thank the National Center for Atmospheric Research (NCAR) the MESO-USER staff, and particularly Drs. Sue Chen, Wei Wang and Bill Kuo for the help in adopting the PSU/NCAR meso-scale model and its new versions at Tel-Aviv University. Thanks are due also to the Israel Met. Service for the initial and boundary conditions. The study was partly supported by the US-Israel Binational Science Foundation grant No. 97-00448.

References

1. Abelson, H., et al., "The Supercomputer Toolkit: A general framework for special-purpose computing," International Journal of High Speed Electronics, **3**, Nos. 3 and 4, 337–361, 1992.
2. J. Katzenelson, et al., "The Supercomputer Toolkit and Its Applications," Int. Journal of High Speed Electronics and Systems, Vol. 9, No. 3, 807–846, 1999.
3. Goikhman, A., "Software for the Supercomputer Toolkit," M.Sc. thesis, Dept. of Electrical Eng., Technion – Israel Inst. of Technology, Haifa, Israel, 1997.
4. Goikhman, A. and J Katzenelson, "Initial Experimentation with the Supercomputer Toolkit," EE memo 996, Dept. of Electrical Engineering, Technion – Israel Institute of Technology, Haifa, Israel, 1995.
5. Adler, B.J., "*Special Purpose Computers*," Academic Press, Inc. 1988.
6. Alpert, P., "Implicit filtering in conjunction with explicit filtering". *J. Comp.Phys.*, **44**, 212-219, 1981.
7. Anthes, R. A., and T. T. Warner, "Development of hydrodynamic models suitable for air pollution and other meso-meteorological studies," *Mon. Wea. Rev.*, **106**, 1045-1078, 1978.

8. Arakawa, A., and V. R. Lamb, "Computational design of the basic dynamical process of the UCLA general circulation model," *Methods in Computational Physics*, **17**, 173-265, 1977.
9. Grell, G. A., et al., "A description of the Fifth-generation Penn State/NCAR Mesoscale Model (MM5)," Nationl Center for Atmospheric Research Tech. Note 398+STR, Dept. of Meteorology, The Pennsylvania State Univ., 1994.
10. Holton, J. R., "An Introduction to Dynamic Meteorology," *Academic Press*, NY, pp. 511, 1992.
11. Pielke, R. A., "Mesoscale Meteorological Modeling," *Academic Press*, NY, 1984.
12. Russell, R. M., "The CRAY-1 Computer System," Comm. of the ACM, vol 21, no 1, January 1978, pp 63-72.
13. Sterling, T.L, et al., "How to build a Beowulf," MIT Press, 1999.
14. IBM staff, Special issue on SP2, IBM Systems Journal, Vol 34, No2, 1995.