CONF-971005--16

AN OVERVIEW OF THE ACTIVITIES OF THE OECD/NEA TASK FORCE ON ADAPTING COMPUTER CODES IN NUCLEAR APPLICATIONS TO PARALLEL ARCHITECTURES*

by
The OECD/NEA Task Force
B. L. Kirk, ORNL, RSICC, P.O. Box 2008,
Oak Ridge, TN 37831-6362
phone 423-574-6176
fax 423-574-6182
email blk@ornl.gov
and
Enrico Sartori, OCDE/OECD NEA Data Bank
Le Seine Saint-Germain, 12 blvd. des Iles,
F-92130 Issy-les-Moulineaux, France
phone 8011 33 1 4524-1072
fax 8011 33 1 4524-1110
email SARTORI@NEA.FR
and
Luis Garcia de Viedma, Consejo de Seguridad Nuclear
Justo Dorado, 11
28040 Madrid SPAIN

session:
invited

*Research sponsored by the Office of Fusion Energy, U. S. Department of Energy, under contract DE-AC05-96OR22464 with Lockheed Martin Energy Research.

AN OVERVIEW OF THE ACTIVITIES OF THE OECD/NEA TASK FORCE ON ADAPTING COMPUTER CODES IN NUCLEAR APPLICATIONS TO PARALLEL ARCHITECTURES

by
The OECD/NEA Task Force
B. L. Kirk, ORNL, RSICC, P.O. Box 2008,
Oak Ridge, TN 37831-6362
and
Enrico Sartori, OCDE/OECD NEA Data Bank
Le Seine Saint-Germain, 12 blvd. des Iles,
F-92130 Issy-les-Moulineaux, France
and
Luis Garcia de Viedma, Consejo de Seguridad Nuclear
Justo Dorado, 11
28040 Madrid SPAIN

Abstract: Subsequent to the introduction of High Performance Computing in the developed countries, the Organization for Economic Cooperation and Development /Nuclear Energy Agency (OECD/NEA) created the Task Force on Adapting Computer Codes in Nuclear Applications to Parallel Architectures (under the guidance of the Nuclear Science Committee's Working Party on Advanced Computing) to study the growth area in supercomputing and its applicability to the nuclear community's computer codes. The result has been four years of investigation for the Task Force in different subject fields—deterministic and Monte Carlo radiation transport, computational mechanics and fluid dynamics, nuclear safety, atmospheric models and waste management.

I. INTRODUCTION

The First International Conference on Supercomputing in Nuclear Applications (SNA 90) was held in Japan in March 1990. The conference was the first of its kind in the area of high performance computing (HPC) addressing problems in the nuclear industry.1 As defined in a United States report,2 "High performance computing refers to the full range of supercomputing activities including existing supercomputer systems, special purpose and experimental systems, and the new generation of large scale parallel architectures." SNA 90 was co-sponsored by several professional societies from many countries, including the Paris-based OECD/NEA. It was clear from the conference that there was a wide array of activities and interest in parallel computing already in place and that these activities had a potential for growth. The second conference was held in conjunction with the ANS Topical in Karlsruhe (M&C + SNA '93).3 The OECD/NEA has been a strong supporter of advanced technologies and has set up working parties over the years to follow up on specific scientific areas. The OECD/NEA-sponsored conferences (example, the Advanced Monte Carlo Computer Programs for Radiation Transport4) have been documented through published reports.

The Task Force was thus created by the OECD/NEA in the early 1990s as a forum for international exchange on the subject of adaptation of existing software to vector or parallel computers. Emphasis was placed on addressing issues of portability and scalability. Several of its members have been directly involved in the modification of software to multiprocessor systems. The following pages will discuss the application codes for which parallel architectures have been used—the list is not exhaustive. The discussion presented here will feature radiation transport (both stochastic and deterministic), collision probability, and thermal hydraulics, computational mechanics and fluid dynamics codes as presented in Ref. 5.

II. RADIATION TRANSPORT—MONTE CARLO METHOD

The Monte Carlo method has been used for over 40 years to solve radiation transport problems in high energy physics, nuclear reactor analysis, radiation shielding, medical imaging, oil well-logging, etc. Individual particle histories are simulated using random numbers, highly accurate representations of particle interaction probabilities, and exact models of 3-D problem geometry. Monte Carlo methods are sometimes the only viable methods for analyzing complex, demanding particle transport problems.

The principal limitation of the Monte Carlo method is computer power. To achieve results with acceptably low statistical uncertainty, it is often necessary to simulate millions of particle histories, consuming many hours of supercomputer time. In the past few years, Monte Carlo methods have been successfully adapted to many vector and parallel computers, enabling the solution of very large and demanding radiation transport problems in acceptable computing time.

The most significant recent computational advance is the migration of Monte Carlo calculations from supercomputers or parallel computers to clusters of inexpensive workstations. Monte Carlo methods are ideal for single workstations or loosely-coupled clusters. The availability and power of distributed computing makes Monte Carlo more accessible, improving the quality of nuclear engineering designs.

Vectorized Monte Carlo codes generally use an "event-based" algorithm. The particle attributes are stored in one large stack. For each type of event there is a queue containing pointers into the stack to those neutrons for which an event of that type is pending. During the calculation the pointers are moved from queue to queue based on the sequence of events. At any point during the tracking process, the longest queue is selected and its pointers are used to gather the particle attributes into a vector. The corresponding event is then analyzed, the altered attributes scattered back into the stack, and the pointers moved to the appropriate queue for the next event.

Parallel Monte Carlo codes generally use a "master-slave" approach, with one master process and one or more slave processes. Parallelization is achieved by providing each slave process a fraction of the particle stack to work on. Each slave tracks its share of the particles independently of the other slaves. Synchronization is only necessary between neutron generations (for eigenvalue problems) or when reaction rate tallies need to be passed to the master process for combination with the tallies from the other slaves.

In the past few years, a number of industry-standard Monte Carlo codes have been adapted successfully to parallel computers or workstation clusters, including VIM, MCNP, KENO, RACER, MONK, MCBEND, TRIPOLI, MVP, MORSE. These are production-level codes used extensively for the engineering analysis of reactor physics, shielding, and criticality safety. These codes include both multigroup and continuous-energy treatments of collision physics and cross-section data.

Researchers throughout the world have adapted production-level Monte Carlo codes to a wide variety of available supercomputers, parallel computers, and workstation clusters. To date, there have been successful efforts at parallel Monte Carlo in the United Kingdom, France, Spain, Germany, Italy, Japan, U. S., and others.

In the U. S., parallel versions of VIM, MCNP, KENO, RACER, and other Monte Carlo codes have been developed for parallel computers and loosely-coupled workstation clusters. When a "master/slave" approach is used on a workstation cluster, one major difference concerns the manner in which work is partitioned among processors. There could be wide disparities in the processing speed of each node for several reasons. First, the individual workstations used in a parallel calculation may have different processor types and speeds. Second, the interactive and system loads on each machine could vary in an unpredictable manner, even during a calculation. Third, it is even possible for a processor to get so busy that it becomes effectively unavailable.

Parallel calculations on workstation clusters using VIM, MCNP, and KENO have achieved reductions in computing time in direct proportion to the number of distributed workstations utilized. Even on a moderately busy network, speedups have been nearly linear with the number of workstations. Using the "virtual supercomputer" provided by a workstation network, a typical Monte Carlo calculation which previously required several days of computing time on a single workstation can now be completed overnight using 4-8 machines.

Within the U.K., work on the parallelization of Monte Carlo methods for particle transport has concentrated on the major production codes MONK and MCBEND. Early work was centered around the Meiko MIMD (Multiple Instruction, Multiple Data) distributed memory system utilizing the innovative transputer chips, with the message passing being performed by the Meiko proprietary CSTOOLS (which possesses many similar features to PVM). The eigenvalue MONK calculations performed by the individual processors were coordinated at the superhistory level (a ten-generation set of neutron histories) in order to minimize message passing overheads, and indeed for the size of system being employed (sixteen processors) the speed-up was near-linear. For the fixed source MCBEND calculations the only co-ordination was performed at the end of the calculation for the purpose of combining results. The versions of MONK and MCBEND have been used extensively for applications analysis within AEA Technology and provided the anticipated benefits of quick Monte Carlo criticality scoping calculations and reduced production shielding and well logging calculation times.

More recently in the U.K. attention has switched to the PVM system, with the latest version of MCBEND now being run on a cluster of UNIX workstations for the rapid analysis of complex problems. This implementation retains the familiar master/slave approach of the earlier work with the activity of the individual processors controlled either by a fixed sample size or by processor time as in the serial version of MCBEND. Input/output is restricted to the master processor and efforts to retain source code compatibility with standard single processor versions of the code have proven successful. Recent trials have demonstrated near-linear performance for a thirty processor IBM RISC system. The virtual supercomputer that emerges from this work is proving to be a major step forward in the optimization of computer resources for Monte Carlo calculations.

In Japan, very significant effort has been applied to adapting MCNP, KENO, MORSE, VIM, and other codes to vector and parallel supercomputers and to developing new vector/parallel Monte Carlo codes such as MVP and GMVP. A particularly noteworthy effort has also been the design of special machine hardware to enable more efficient vectorization of Monte Carlo, the JAERI "Monte Carlo machine" called MONTE-4. More recently, Monte Carlo codes have been successfully adapted to very cost-effective clusters of workstations.

The Monte Carlo machine MONTE-4 was built at Japan Atomic Energy Research Institute. Based on an existing supercomputer, it was equipped with special hardware called Monte Carlo pipelines and enhanced load/store pipelines to raise the vectorization ratio and boost vector performance. In addition MONTE-4 has parallel processing capability with four processors. As a result, speedup ratios have been multiplied by factors of 1.4–2.2 due to the special pipelines on single processors, and by factors of 2.7–3.4 by parallel processing with four processors. Overall, speedups greater than 10 times have been achieved for production-level Monte Carlo codes such as KENO-IV and MCNP.

The parallel and vector Monte Carlo Codes MVP (continuous energy model) and GMVP (multigroup model) were developed in Japan to solve neutron, photon, or coupled particle transport problems. Those codes were originally developed for vector supercomputers with a single processor, and hence adopted the event-based scheme. The codes realized speedups by a factor of ten to a few tens compared with the scalar versions or other conventional scalar codes.

At present, MVP And GMVP can execute in a variety of parallel computer environments, including vector parallel supercomputers with shared memory (NEC SX3, MONTE4, HITACHI S3800) or with distributed memory (FACOM VPP500/42), MIMD scalar massively parallel computers (FACOM AP1000 with 512 processors), and workstation clusters by using the PVM package.

Significant work has been applied by the Japanese to the development of the direct simulation Monte Carlo method (DSMC). The DSMC method is a numerical technique for simulating the real gas flow using simulated particles. These simulated particles represent particles sampled from the statistical distribution, and velocity components and co-ordinates of each simulated particle are renewed through representative collisions at every time step. This DSMC algorithm provides an efficient way to integrate the Boltzmann equation from the rarefied to the near-continuum region, and thus can be applied to flows where the Navier-Stokes equation is invalid.

The DSMC method is applied to the analysis of the free expansion flow where the Knudsen number (the ratio of the mean free path to the characteristic dimension) varies in a wide range. The detailed fields of the free expansion flow are required in the Atomic Laser Isotope Separation Process. Subcollision cells, which are generated automatically within fixed collision cells by calculating the local mean free path of the flow as the calculation proceeds, are devised in order to study the free expansion flow. The DSMC method is also applied to the calculation of the continuum cavity flow, and the results by the DSMC method are in good agreement with those by the Navier-Stokes equation. Also, the critical Rayleigh number in the Rayleigh-Benard convection is predicted by the DSMC method and fluctuations in a macroscopic flow associated with the transition are studied from a microscopic or atomic level.

The parallel DSMC code has been developed, because a large number of simulated particles are required to simulate the low Knudsen number flows in the near-continuum and continuum range. The DSMC code is implemented on a highly parallel computer (Fujitsu's AP1000) based on the region decomposition method, and a speedup of 42 times has been obtained using 64 processors. Recently, the serial and parallel DSMC codes are developed incorporating the adaptive scheme for the collision cells by using a message passing system PVM3 on the IBM SP2, and a good scalability is obtained by achieving a speedup of 7.9 times going from 2 to 16 processors.

III. Radiation Transport—Deterministic Method

Deterministic methods have been used to solve the linear Boltzmann transport equation since the early 1960s, with applications in reactor analysis and shielding, criticality calculations, nuclear well logging, and dose calculations, among others. Unlike Monte Carlo, deterministic transport requires discretizations in phase space (spatial, energy, and angle), which introduces considerable complications for both methods developers and code users. However, deterministic transport methods usually offer significant speed advantages over Monte Carlo methods in one and two spatial dimensions, and in three dimensional calculations where the answer is required everywhere (e.g., a dose profile), or time-dependent calculations where information is required about the time derivative (e.g., well logging).

Like Monte Carlo methods, deterministic transport calculations also require considerable computing power for problems in two and three spatial dimensions. (Calculations in one spatial dimension are typically solved quickly enough on today's workstations that no effort is warranted in speeding them up, and thus will not be considered further in this context.) Unlike Monte Carlo methods, deterministic transport methods require large amounts of memory to store the flux information. Thus, deterministic transport codes were initially limited to one spatial dimension by computer memory sizes. With the increase in computer sizes, two dimensional production codes became widely available in the late 1970s and early 1980s. As the memory on machines increased even further, three dimensional production codes were released beginning in the mid 1980s. Although most available production codes are still time independent, three dimensional time-dependent transport codes are in development.

Parallel deterministic transport codes have been implemented with varying degrees of success on many different types of architectures, including traditional Parallel Vector Processors (PVP's) such as the Cray YMP/C90/T90/J90, Massively Parallel Processors (MPP's) such as the SIMD (Single Instruction, Multiple Data) Thinking Machine's CM-200, MPP MIMD (Multiple Instruction, Multiple Data) machines like the Cray T3D, Intel Paragon, or IBM SP2, and networks of workstations. Just recently, and too soon for any results to be available on parallel transport code performance, SMP's (Symmetric Memory Processors) have become a factor in the marketplace. Although still not clearly defined, SMP's may be thought of as a number (2-64) of high-end commercial microprocessors grouped in a single box and sharing a common memory, perhaps with non-uniform access.

Parallel methods for deterministic transport codes involve domain decomposition over one or more of the three variables in phase space - energy, angle or space. Domain decomposition in energy results in the coarsest-grained algorithm, yet has been the least utilized to date, since it inherently requires an extrinsic decomposition in terms of the standard outer/inner iteration algorithm used in all production-level deterministic transport codes today. For problems involving only downscatter, energy domain decomposition appears to offer no advantages, while for problems with upscatter and/or fission the speedups gained in solving more than one energy group simultaneously must be weighed against the increased number of iterations required for convergence, which can be extremely problem-dependent.

Angular domain decomposition in Cartesian geometry has been explored on a wide variety of parallel architectures using the Arbitrarily High Order Transport code AHOT. Given the intermediate granularity of angular domain decompositions, they are perfectly suited for supercomputer machines of the Cray PVP class which support at most a few tens of ultra fast processors. Recently Azmy presented two angular domain decompositions for multitasking the production code TORT on Cray computers running the UNICOS operating system. One of these, the Direction Parallel (DP) scheme, is based on concurrently sweeping each row of the mesh in all discrete ordinates within an octant of angular space. The production nature of TORT and its large-applications portfolio demands that its parallel version execute on production machines running in typical timesharing mode. Measured performance on an 8-CPU Cray Y/MP produced Wall-Clock speedup approaching a factor of five on a relatively small, 100,000 cell test problem (TORT's largest application to date utilizes 3.6 million cells). Analysis of the parallel performance of TORT demonstrated good scalability properties with mesh size and angular quadrature order, but exhibited sensitivity to the machine's computational load during the execution period.

Using a Jacobi-type iteration on the source iteration equation, Morel et al have implemented several limited distribution 3D transport codes (NIKE/DANTE and THOR) for solving the even-parity S_n, SP_n, or P_n equations on unstructured tetrahedral meshes, and the first-order S_n equation on a rectangular mesh. Their 3D spatial decomposition scheme is intrinsic in the sense that it performs a Jacobi iteration on the angular interface flux between all cells, not just the cells along a PE boundary, so that the same iteration path is followed regardless of the number of processors used. However, it does require a third iteration level, in addition to the two required for the standard inner/outer iterations. One "sweep" of the source iteration equation (performed in a single, direct inversion using the standard sequential algorithm) requires 3N-2 Jacobi iterations to return the same solution, if the mesh size is NxNxN. Thus, Jacobi-type iteration methods require a significant number of additional iterations per source iteration for large problems. However, for problems with optically thick meshes and/or time dependence, subsequent solutions of the source iteration equation can use the previous solution as a starting guess for the Jacobi iteration cycle, reducing the required number of Jacobi iterations/source iteration to a more manageable 10-15. The Jacobi-type iteration scheme should be highly parallel, especially since their algorithms use higher-order spatial differencing methods, which usually improve the computation/communication ratio.

However, their codes (as currently implemented) typically only exhibit speedup ratios as the number of processors is doubled of 1.6 to 1.7 on the CM-200 and CM-5 due to the relatively large latencies on these machines.

The standard direct, non-iterative solution of the source iteration equation may be parallelized by recognizing that, for a given S_n direction, as the mesh sweep proceeds through the 3D spatial mesh it allows a 2D plane's worth of work to be performed simultaneously, i.e., in parallel. However, as the "diagonal plane" sweep initiates and terminates there is insufficient work to fully utilize all the PE's, and for a large NxNxN mesh it can be shown that the parallel computational efficiency is at most 33%. In reference, Koch et al proposed several methods for increasing this efficiency. The most efficient one on the CM-200 architecture, which is based on pipelining two angular octants at a time, resulted in a more tolerable parallel computational efficiency of 50%. This method was implemented in the THREEDANT code and has been in limited production use at LANL since 1994. The THREEDANT/CM code generally exhibits speedup ratios on the order of 1.6-1.7 as the number of processors is doubled for large (million plus) meshes.

The diagonal plane sweeping algorithm has also been implemented in the latest production version of THREEDANT, named DANTSYS/MPI, on a 512 PE Cray T3D at LANL. The parallel computational efficiency of the diagonal plane sweep has been improved to levels of 85-95% by pipelining angles as well as octants and using a sequential sweep within the cells assigned to a PE's spatial subdomain, thus performing the diagonal sweep over just the PE space. This increase in parallel efficiency, along with the low latency hardware of the T3D, resulted in speedup ratios of 1.8 to 2.1 as the number of processors is doubled for a 12-group, 2 million mesh criticality safety problem. For 512 PE's, the speedup ratio from 256 PE's was only 1.6 due to the increasing effects of latency within the smaller spatial subdomains. For 512 T3D PE's, the performance (wall clock time) is equivalent to approximately 50 YMP PE's, or 6 Gigaflops. Using a simple model which accurately predicts DANTSYS/MPI performance on the T3D, and extrapolating this model to the new Intel MPP at Sandia National Laboratory, DANTSYS/MPI should be able to solve a 256x256x256 problem (17 million meshes) on this machine at a performance level equivalent to approximately 725 YMP PE's, or 87 Gigaflops. DANTSYS/MPI is fully accelerated using DSA on both the inner and outer iterations, and is capable of performing both time independent (fixed source or eigenvalue) and time-dependent calculations. The DSA diffusion solver uses a parallel conjugate gradient solver which is currently preconditioned using Red-Black line sweeps in the Y and Z dimensions, and line inversions in the X direction.

Dorr et al. have explored simultaneous domain decomposition in all three variables (energy, angle, and space) using their research code Ardra. Starting with the source iteration form of the multigroup transport equation, they reduce it to a formal matrix-vector multiplication, which is then decomposed over a 3D PE space. By reverting to assembly language coding for the transport sweeps, they have achieved performance rates approaching 10 Gigaflops for their code on a 128 PE Cray T3D. The spatial decomposition uses the diagonal plane sweep approach described above for the spatial domain, so their algorithm is intrinsic in all three variables. However, by starting with the source iteration form of the multigroup equation, there is a potential for a convergence penalty (in comparison to standard inner/outer iteration techniques) when solving problems with only downscattering. Also, the additional decomposition in angle, appears to be of little, or negative, benefit to parallel performance in their code.

In the past several years, different studies have been performed at Penn State University on different domain decomposition algorithms (individual and hybrid) and different iterative techniques. Angular domain and spatial domain decomposition formulations and their hybrid combinations with different iterative schemes have been developed and incorporated into the TWOTRAN-II code. These algorithms have been tested on different parallel computers; references discuss these algorithms and their performance on the IBM 3090/400-600 and CRAY C90/J90 shared-memory computers. Based on these studies, and the memory limitations for solving large 3-D transport problems, Penn State has developed a new 3-D (Cartesian) parallel S_n code called PENTRAN (Parallel Environment Neutral-particle TRANsport). The code allows for complete phase space decomposition in distributed-memory and/or distributed computing environments using the MPI (message passing interface) library. Parallel auto-scheduling of a transport problem in PENTRAN leads to a three-dimensional Cartesian, virtual PE topology, with angular, group, and spatial decomposition axes.

PENTRAN has been implemented and tested on a distributed-memory IBM Scalable POWER-Parallel System 2 (SP2) at the Cornell Theory Center (CTC). Although tested on the SP2, the code is intended for parallel execution using MPI on any MIMD system. The code has been successfully benchmarked with existing production codes including TWOTRAN-II, TWODANT, THREEDANT, DORT, and TORT using multigroup, anisotropic, forward and adjoint, fixed-source and eigenvalue test problems. High parallel performance has been achieved on the CTC IBM SP2 computer. Testing has demonstrated that the code has a parallel fraction in a range of 95% to 99% (depending on the problem and the domain decomposition used). Based on Amdahl's law, for a 99% parallel fraction, a maximum speedup of 100 is achievable, assuming no loss to communications.

IV. NODAL TRANSPORT METHOD

A somewhat different class of "nodal" transport methods which has been extensively developed and used in recent years is the method based on the variational nodal approach developed by E.E. Lewis. This method has been implemented in the VARIANT code developed jointly by researchers at Argonne National Laboratory and Northwestern University. Initial efforts to parallelize this code are described below.

VARIANT solves the multigroup steady-state neutron diffusion and transport equations in two- and three-dimensional Cartesian and hexagonal geometries using variational nodal methods.

The VARIANT code is divided into four major structural parts that must be executed sequentially: input handling, calculation of response matrix elements (coupling coefficients) relating incoming and outgoing partial current moments, iterative calculation of the fluxes, and output of results. Within a single outer iteration, odd-even sweeps of planes are performed along the Z-direction combined with an n-color iteration scheme for each (odd or even) X-Y plane.

The objective of the parallelization effort was to reduce the overall computing time by distributing the work of the two computationally intensive (sequential) tasks, the coupling coefficient calculation and the iterative solver, equally among a group of processors. Further it was required that the parallelization strategy not alter the convergence properties of the iterative solution on a single processor. Another objective of the initial parallelization work, was to preserve the original form (including data structure) of the code. Dependent on the given problem, either the coupling coefficient calculation or the iterative solver requires most of the computing time. To maximize efficiency, the number of processors applied in the two parallel tasks was allowed to differ.

The initial parallelization approach involved a decomposition only along the axial dimension. Due to the two-cyclic iteration scheme, an odd-even pair of planes is assigned to one processor. With this decomposition approach, the parallel VARIANT code may utilize up to a few tens of processors.

A single program multiple data (SPMD) implementation for the parallel VARIANT code is obtained by introducing two control matrices which control the parallel execution of the coefficient calculation and the iterative solver. Each entry of the control-matrix denotes the processor which is responsible for the task associated with the entry. During an initialization step, the mapping between processors and tasks is performed, taking into account load balance and communication patterns which minimize message passing overhead. Identical control matrices are generated on each processor. The coefficient calculations are then performed in parallel as follows. Each processor executes the entire loop over all unique node types. However, only the node types indicated by the control matrix as being its responsibility are actually calculated on this processor. For all other node types, the complex coefficient calculations are skipped. A data exchange step follows the computation of the coefficients in which each processor obtains the non-locally computed coefficient data required for its part of the iterative solver. The parallel execution of the iterative solver is orchestrated in a similar fashion.

The parallel implementation of VARIANT is based on the Message Passing Interface (MPI) standard. To date, the parallel VARIANT code has been successfully installed and used on several distributed memory architectures, including the 128 processing node IBM SPx (where x is either 1 or 2) supercomputer and heterogeneous networks of Sun and IBM/RS6000 workstations.

V. COLLISION PROBABILITY METHOD

A typical 2D assembly calculation using collision probabilities (**CP**) consumes about 90% of the computing time in the **CP** evaluations. Consequently RZ or 3D calculations become prohibitively long. Fortunately, this method is very suitable for parallelization. Massively parallel computers, especially MPMD (massively parallel, multiple data) machines, bring a new breath to this method.

The CP method for 2D geometries consists, for each energy group, of a double integration over the neutron trajectories. For each trajectory one must calculate the contribution to the P_{ij} matrix of every traversed region I in the basic domain with every traversed region j within the necessary optical distance.

The quadrature formula consisting of the number of trajectories and their geometrical length, depends on the geometrical complexity and the cross sections of the problem treated. A complex geometry with many regions needs thousands or tens of thousands of trajectories. The transparency of materials to neutrons of the fast energy groups requires very long trajectories (in length, and consequently in number of intersections), of the order of several hundred intersections. The interface current approximation can significantly reduce the length of the trajectories, but a finer quadrature formula is necessary in order to achieve sufficient accuracy for the escape and transmission probabilities.

The easier approach consists in parallelizing of the energy group treatment. In the classical and sequential approach, the task is split into two steps: trajectory tracking and CP evaluation. Tracking is done only once, and the CP calculation is repeated for each energy group, with the same trajectories, and changed cross-sections. The optical length of the trajectories depends on the cross-sections. Thus one has to track very long trajectories, in order to obtain the necessary optical length for the fast groups. Because of the amount of data, the trajectories should be stored on disk and reread into memory for each group.

The parallel algorithm distributes several energy group calculations to each processor. Trajectory tracking is repeated by each processor for each group, and is thus not excessively penalizing, compared with other possible parallelization strategies. If this is not done one must either execute tracking on one processor, while all others are idle, or parallelize the tracking, which is difficult. Another advantage of this method is a dramatic decrease of communications. The same geometrical data are broadcast to all processors, and the corresponding total cross-sections are sent to every processor; as a result there is neither communication of tracking data between processors nor with the disk. One more advantage is that there is no overtracking because only the necessary optical length of the trajectories is calculated for each group.

The P_{ij} matrix calculation time is different for each group: the high energy groups are more transparent, i.e., the neutron trajectories are longer, and so the computation time is longer. In order to equilibrate the work of the processors and to achieve the best occupation ratio, the energy groups are distributed over different processors in decreasing order of their transparency. The longer calculations are executed first, so that the occupation times of all processors are balanced.

The feasibility of the proposed algorithm was tested on the TDT mock up, which calculates the CP in non-regular 2D XY geometries. A standard 99-group cross-section library was used.

As a first step, three versions of the proposed algorithm were implemented, on a 32 processor CM5 computer, using a hostless programming model and a specific (CMMD) message passing library. TDT is written in standard FORTRAN 77, therefore some FORTRAN 90 (and CM FORTRAN) adaptations, like dynamic allocation, were necessary. Calls to standard FORTRAN 90 intrinsic functions were introduced. Programming was simplified by introducing array instructions.

This work demonstrated the feasibility of the algorithm, either for vacuum or reflective boundary conditions, as well as for mirror reflection or by using the interface currents formalism.

The next TDT version, similar to what was implemented in APOLLO-II, was then parallelized using PVM and FORTRAN 77, and the best algorithm from the CM5. This version was tested on a CRAY 90 coupled to a 128 processor Cray T3D computer, on a 16 processor IBM SP1, a 32 processor IBM SP2 and on a homogeneous and heterogeneous networks of HP, Sun and IBM workstations. The same source code was used for all computers.

This work allowed for the assessment of performance on three different parallel architectures. All tests were very satisfactory. The IBM SPx results were better than expected; even though every processor works in a time shared mode, the proposed algorithm distributes the group calculations dynamically. The principal drawback of the CRAY 90 and T3D setup is the assignment of the partition (number of processors to be used) to the time of the execution of the job. So, for example, at a speed-up of 10 in the CP's, 50% of the APOLLO-II calculation time will be done in sequential mode, while all T3D processors will be idle. But this coupled architecture has a significant advantage for a code like APOLLO-II. Once the CP calculations are parallelized, the next more important time consumer is the flux module. So the work can be shared favorably between the numerous scalar processors of the Cray T3D for the CP numerical integrations and the vector CRAY 90 processor for the matrix inversions of the flux calculations.

The technique just described is efficient if the number of energy groups is much larger than the number of processors. For generalizing the treatment to the inverse case, when the number of processors is close to or much larger then the number of groups, the splitting of the CP calculations of each group over several processors should be made possible. In order to balance the workload, each processor working for the same group should track nearly an equal number of trajectories, with nearly an equal total number of intersections, and calculate their contribution to the CP of the group. At the end of the calculation, each processor sends its CP matrix to the first processor of the group, and a message to the host that it is ready for the next work. The first processor of the group receives and sums all the contributions to the final matrix, applies the reciprocity relations, normalizes and sends the resulting matrix to the host. The CP matrix is symmetric, so only one half (n*(n+1)/2 instead of n*n elements) of each matrix is sent. This adds to improved communication performance.

At present the improved algorithm is being implemented in the MARSYAS method of APOLLO-II, which calculates the CP in regular 2D XY geometries, such as those of PWR assemblies. Routines for within node calculations are those of the MARSYAS

code, already used in the standard sequential APOLLO-II code. At the same time an effort is being devoted to convert these routines to FORTRAN 90, to implement memory allocation and data structuring.

This version of APOLLO-II is now operational and in the phase of testing on different computers. The spectacular benefits should be achieved with the new, 256 processor CRAY T3E, installed at the CEA.

Using the algorithm described above, the CP calculation is an embarrassingly parallel problem. Paralellization of the other tasks, like flux iterations, is much more delicate, and the speed-ups will be less spectacular—it is difficult to synchronize the processors in order to obtain a good load balance. If the processors should often wait for the messages and for relatively long times, the CPU time will probably not suffer too much, but the WC time will become unacceptable. Nevertheless some other authors published a very interesting contributions on the flux iterations parallelization.

In the assembly code GTRAN2, trajectory tracking and CP numerical integration are uncoupled in two independent steps. Tracking is parallelized by assigning one angle per processor, and the numerical iterations by assigning one group per processor. The load balancing is difficult to achieve since the numbers of angles and energy groups must be a multiple of the number of processors. Even in this situation, the CP numerical integration times are not the same if the reflective (mirror) boundary conditions are used.

The successor of GTRAN2, MAGGENTA code was parallelized. Each processor is assigned to a unique geometrical region and is coupled through neutron currents at the interfaces. This allowed the parallel analysis involving 16 fuel assemblies, described in an exact 2D geometry, with 996 regions each and 12 energy groups in less than 10 minutes of wall clock time. Computational times for more assemblies or more refined energy meshes, will be more than proportionally longer, because of slower convergence, but still not prohibitive.

The CP calculations and the multigroup flux computation were parallelized in INTPN and DRAGON codes, using the same algorithm. The PVM library, in hostless mode is used. INTPN solves the transport equation in slab assemblies, so the interest of this work is essentially as a feasibility investigation. Parallelization of the flux iterations is achieved by distributing either different energy groups or different regions on sets of processors. Distributing different groups on different regions is a natural approach. In a sequential calculation, in each outer iteration, up and down scattering transfers are taken into account, in parallel the groups are decoupled, so the convergence is slower. This is acceptable, because the speed-up is still important.

VI. NUCLEAR SAFETY

H. Makowitz made a preliminary study to demonstrate the feasibility of concurrent multiprocessing in RELAP5. A further work addressed the problem of determining the overhead associated to concurrent multiprocessing. A hybrid calculational scheme was used, where an explicit coupling between physical processes was attempted for a number of time steps and an implicit iterative step for a few iterations followed. This calculational scheme is probably unsuitable for rapid phase change problems, but is adequate for typical slow transients and small brake calculations. It is important to mention that faster-than-real-time simulations could be obtained. By means of a new algorithm in the implicit iterative step and using multitasking, speedup factors ranging from 1.5 to 2.5 on a CRAY X-MP/4 were obtained. A case study of the multitasking overhead showed that this was, at best, a linear function of the number of processors and that it might saturate the speedup achieved for a number of 8 CPUs.

In the CERBERUS code, Makowitz studied the effect of proper load balancing on larger memory banked machines and the use of lower overhead constructs such as CRAY micro- and autotasking. The first version of the code implemented parallelism only in the heat transfer model. To analyze overhead and load balancing issues, a number of computational problems with an increasing emphasis on heat transfer calculations were developed to be used with CERBERUS. The results obtained on a CRAY X-MP/24 and CRAY Y-MP/832 demonstrated the effect of load balancing. No significant gain was obtained due to vectorization.

CERBERUS Version 01 incorporates a parallel heat transfer and hydraulic module, while CERBERUS Version 02 builds on Version 01 plus the matrix solver TR(k)BREMSSTRAHLUNG(n). The TR(k)BREMSSTRAHLUNG(n) algorithm is a direct procedure for solving the kinds of linear systems of equations that arise in one-dimensional network problems, such a flow through the primary side of a nuclear power plant. The TR are direct methods tailored to efficiently solve large sparse linear systems which are nearly tridiagonal by taking advantage of the structure of the matrix. The TR name arises from the splitting of the coefficient matrix, C, into the sum of a tridiagonal matrix, T, and the remainder, R. These methods, which were designed to be used in vector processors, can also be parallelized to some extent. By utilizing the TR(k)BREMSSTRAHLUNG(n) vector/parallel solver a significant overall vector gain on one CPU is obtained for large problems. However, for small and medium-size problems, where

the matrix solution does not dominate the code workload, CERBERUS will behave essentially as a scalar code. The speedups obtained on a Cray C90 with 16 CPUs are 3.3 for a small problem and 7 for a large one.

The PaRaELAP code is a parallel implementation of RELAP5 incorporated in the REPLICA system, a Real-time RELAP5 Emergency Preparedness and Licensing Interactive Computational Aid. PaRaELAP was demonstrated to sustain real-time performance on a Titan 750 dual CPU (shared memory, vector/parallel) machine during a small break loss-of-coolant accident simulation on a representative three loop Westinghouse engineering plant model.

At present, a parallel version of RELAP5 runs on the DEC Alpha and SUN workstations. A speedup of 1.44 using two processors has been obtained, nevertheless a revision of the thermal hydraulic data base to use unit strides for accessing the arrays to get a better cache performance is being carried out.

At Spain's Consejo de Seguridad Nuclear (CSN), some preliminary analysis on RELAP5 have been made. A shared memory programming model has been adopted. The two most time consuming subroutines in the Hydro module have been analyzed and parallelised on a Convex C3440. The 100-s TYPPWR SBLOCA problem reproduces RELAP5/MOD3.1 results within round-off differences. A speedup of 1.19 (1.27 theoretical) has been obtained. The total fraction of the parallel portion is estimated to be 83%. According to the preliminary results obtained, a total speedup of 2.6 with 4 processors could be reached.

The ATHLET (Analysis of THermal-hydraulics of Leaks and Transients) code has been developed at Germany's GRS with the aim to cover the whole spectrum of accident scenarios for LWRs. Its field of application comprises the analysis of anticipated and abnormal plant transients, small, intermediate and large breaks.

ATHLET has a modular structure: Thermo-fluid-dynamics (TFD), Heat Transfer and Conduction (HECU), Neutron Kinetics (NEUKIN), General Control Simulation (GCSM), and Numerical Integration (FEBE). In TFD, a modular network approach is used. By concatenation of basic elements, called objects, a given configuration can be simulated. Each fluid dynamic object supports a subset of the entire ordinary differential equation (ODE) system, which is integrated in time simultaneously by the ODE-solver FEBE.

ATHLET has been parallelised on a CONVEX C3 with 6 CPUs. Taking advantage of the object orientation in the TFD module, DO LOOP parallelisation using appropriate compiler directives has been implemented. In a typical post-test calculation of cold leg break in a scaled test facility, the speedup obtained was about 3.8.

ATHLET has been recently coupled with codes FLUBOX and QUABOX/CUBOX. FLUBOX is a 2D module for the simulation of two-phase flow in the downcomer of a reactor pressure vessel with a two-fluid model. FLUBOX has been partially parallelised, and a speedup of 7.4 in a iPSC/2 with 16 processors has been achieved.

The 3D reactor core code QUABOX/CUBOX solves the diffusion equations by a coarse mesh method. The thermo-hydraulic feedback effects are described by a parallel channel model solving equations of heat conduction in the fuel rod and fluid-dynamic equations in one-dimensional channels. The 3D reactor core model is used to solve static and time dependent problems.
The calculation of nuclear data, which is the most time consuming part of the code (70%), has been parallelised. In an Intel iPSC/2 with 16 processors a speedup of 2.7 was obtained.

At the French CEA, the thermal-hydraulics safety code CATHARE 2 is undergoing changes. In the 3D module, vectorization was implemented. Two numerical methods have been tested: a direct method (LU factorization) and an iterative one based on conjugate gradient. The iterative solver yielded to significant CPU and memory savings on large meshes. As a result of these optimizations, a vector speedup factor of 3.5 could be achieved on a Cray C90.

The present version of CATHARE is 97% parallel. Some tests have been performed on an SGI with 8 CPUs with resulting speedups of about 4. As result of the optimizations performed in CATHARE 2, the time needed for a reactor calculation on a Cray computer can be reduced by a factor of eight.

A version of the CATHARE code using PVM and a data parallelism approach based on a domain decomposition method has been recently developed for MPP computers. In this version the system (circuit) is considered to be split in sub-domains and each sub-domain is assigned to a processor. For a calculation with 5 processors (one for the vessel, one for the broken loop and three for the intact loops) speedups ranging from 2.6 to 3.7 for several computers (SPP1000, Power Challenge, DEC Alpha and CRAY C98) were obtained.

VII. COMPUTATONAL MECHANICS AND FLUID DYNAMICS

Finite element codes are easily converted for use on parallel computers, because the method itself has a high degree of parallelism. Ways of parallelizing a finite element code include domain decomposition, master-slave approach and element-by-element.

Domain decomposition is a coarse grain strategy. The problem is divided into several subdomains and each subdomain is assigned to a processor. The calculation time for each subdomain should be uniform across all processors. The global solution is obtained from the contributions of each individual processor. On a distributed memory system, this is done through message passing. On a shared memory system, the variables reside in globally-shared memory.

In the master-slave approach, the serial code represents the master program. The time consuming calculations including the solution of the global system of equations are assigned to the slave processors. The master-slave scenario is conducive to problems with a fine grain structure, allowing for proper load balancing between the slaves.

In the element-by-element method, the whole problem is formulated on the element level. This is obvious because an integral part of the calculations is done on the element level. The global solution still depends on each element's contribution.

At the Institute of Safety Research and Reactor Technology at KFA Julich, the finite element code SMART is implemented on an Intel Paragon, an IBM SP2 and a cluster of DEC Alpha workstations using coarse grain domain decomposition. SMART allows the calculation of heat transfer problems. A conjugate gradient method is used for solving the linear systems of equations. System efficiency is achieved at about 70% for a small number of processors.

VIII. AMOSPHERIC MODELS

Atmospheric modeling is an integral component of research into the propagation of airborne pollutants. The development of atmospheric models is closely linked to the advances in parallel computing. As long ago as 1922, L.F. Richardson, the first researcher to produce forecasts based on numerical methods, predicted that the construction of large-scale atmospheric models would require the use of thousands of computers operating in "parallel".

Meteorology was one of the first disciplines to advocate the use of parallel computers and at present is one of the areas where researchers are making the most efforts to use massively parallel computers as a platform for executing numerical models. Because of their close similarity to atmospheric models, models depicting the propagation of all kinds of different air-borne pollutants are also starting to be run on massively parallel computers. Researchers in almost every country in the world are attempting to parallelize existing programs. This trend is far more marked in the United States, although countries such as France, Germany and the UK are already following a similar course. At present, several centres in the United States are working on porting a number of models to parallel machines. Some of these projects are discussed below.

The Forecast Systems Laboratory (FSL), run by the National Oceanic and Atmospheric Administration (NOAA), has already started work on a number of projects on the porting of atmospheric models to multiprocessor machines with shared or distributed memory. One of the first models to be parallelized was the Quasi-Nonhydrostatic Model (QNH model). This model is used for regional forecasting and is in two parts - one dynamic, the other physical. The first part covers phenomena such as advection, diffusion and topography. The physical part consists of four packages: moist physics, radiation, surface forcing and turbulence. The first part, which is also known as the Well-Posed Topographical (WPT) model, was initially ported to the KSR1 machine. On 32 processors of the KSR1, a speedup of 21 is obtained. The complete code for the model was subsequently ported to a number of parallel machines like the Intel Paragon and IBM SP2. The IBM SP2 results are twice as fast as that of the Intel Paragon.

Another model known as SEQN (Scalable Explicit Quasi-Nonhydrostatic) has been parallelized on both shared memory and message passing systems - the KSR1 and Intel Paragon. In both cases, in order to make maximum use of the machines, operations must be optimised at both the local level (execution at the level of individual processors, which will clearly be governed by the characteristics of processors) and at the level of the overall architecture (communications between processors).

Argonne National Laboratory (ANL) has ported the Pennsylvania State University's NCAR Mesoscale Model. This model utilises

fine-grain decomposition which is mapped and dynamically managed by ANL's Program Composition Notation (PCN) language. The code has been executed on the Intel Touchstone Delta machine in several configurations. A test run with 600 processors yielded execution times 400 times faster than those obtained with the serial version.

Another model which has been ported to parallel platforms is global atmospheric circulation code PCCM2. This model contains a parallel spectrum transformation, a set of semi-Lagrangian transport equations and load-balancing algorithms. The resultant code was tested on the IBM SP2 and the Intel Paragon. Other studies have been carried out on the Intel Touchstone DELTA to determine the reasons for the imbalance in computational loads and to remedy this imbalance.

A new version of the UCLA model has been parallelized by a team from the Climate System Modeling Group at the Lawrence Livermore National Laboratory and the Atmospheric Sciences Department at the University of California . MIMD distributed memory parallel architectures were used for this model. The program was executed on the following computers: Cray C90, IBM SP2, Cray T3D, Meiko CS2.

The SKYHI code, developed by the NOAA Geophysical Fluid Dynamics Laboratory (GFDL) is used to study the dynamics, climatic evolution and chemistry of the atmosphere. Parallel versions have been developed for the Cray C90 and the TMC CM-5 (Thinking Machines Corporation).

A number of European organisations are also parallelizing the codes used in meteorological forecasting. IFS (Integrated Forecasting System) is a model developed jointly by the European Centre for Medium-range Weather Forecasts (ECMWF) and the French National Meteorological Service (the French version is known as ARPEGE). The parallel version of this code has been designed to be portable to several parallel platforms such as: Cray C90, Cray T3D, Fujitsu VPP500, IBM SP1, IBM SP2, Intel iPSC/860, Intel Paragon, Meiko CS-2, TMC CM-5. Efficiencies of the order of 90 per cent have been obtained, for example, on the Cray T3D.

A team from the Chemical Engineering Department of the University of Kentucky has parallelized a simple model of air pollution based on K-theory. One of the inputs to the model consists of a wind speed field supplied by other atmospheric models. This parallelization has proved effective on the IBM 3090-600J computer (with a limited number of processors) and has reduced run times by a factor of five. Another team from the Department of Chemical Engineering at the California Institute of Technology has ported the code from the CIT model (air quality model) to the Intel Touchstone DELTA platform (256 processors, MIMD architecture) and has obtained execution times 28 times faster than those obtained with serial processing.

IX. WASTE MANAGEMENT

As far as probabilistic assessment (PA) of nuclear waste repositories is concerned, progress has been made at the U.S. Nuclear Regulatory Commission (NRC) with parallel computations using PVM on a network of Sun workstations for Monte Carlo analysis. It works very well and it will be used for all future performance assessments. An estimation as to the amount of computational power used in the Iterative performance assessment for Yucca Mountain Repository, and to compare it with the power available on modern supercomputers, has been made.

In order to estimate the potential for speedup and the need for abstraction, we compare the computational effort from a recent PA with speeds available on current and projected computers. Actual speed comparisons must take into account factors such as the degree of vectorisation (for vector computers, e.g., Cray), scalability on parallel processors, and limitations of storage devices, so we caution that these are only order of magnitude estimates. The recent NRC PA for the Yucca Mountain Repository required on the order of 100 CPU hours (single processor) on a Cray-XMP-24 computer, even with models that were highly simplified. Using the speed from the LINPACK benchmark, we estimate that the NRC PA equalled roughly 10^{13} floating point operations (FLOPS). Since so much of the computational was related to repetitive Monte Carlo calculations, we would expect the run times to scale linearly to the number of processors on a parallel computer. Recent calculations on massively parallel computers routinely have sustained speed in excess of 5×10^9 FLOPS per second, which would be equivalent to 2000 seconds for the NRC PA problem. High-end speeds for presently available machines exceed 10^{11} FLOPS per second, or 100 seconds for the NRC problem. Performance in the next few years is expected to exceed 10^{12} FLOPS per second, or 10 seconds for the NRC problem.

X. CONCLUSION

The investigation performed by the Task Force has shown that high speed computing is a necessity for the magnitude of problems in the nuclear field. As computers become increasingly powerful, the benefits to computations especially in real time will be greatly appreciated. The Task Force will continue to monitor, organize portability and performance exercises, and support high performance computing.

XI. ACKNOWLEDGMENT

The authors would like to acknowledge the support of Dr. Iran Thomas of the U.S. Department of Energy's Office of Energy Research, of M. Akimoto, J. Altes, Y. Azmy, R. Baker, F. Brown, F. Diez, L. Garcia de Viedma, J. P. Gregoire, A. Haghighat, K. Higuchi, A. Kavenoky, H. Khalil, V. Mastrangelo, C. de Oliveira, J. Pena, R. Roy, Z. Stankovski, V. Teschendorff, J. L. Vaudescal, and the members of the Task Force who contributed to the final report.[4]

XII. REFERENCES

1. *The First International Conference on Supercomputing in Nuclear Applications (SNA 90)*, edited by Japan Atomic Energy Research Institute, Nuclear Energy Data Center, Ibaraki-ken, Japan, 1990.

2. *High Performance Computing and Communications: Toward a National Information Infrastructure (FY 1994 Blue Book)*, published by the National Coordination Office for Computing, Information and Communication, U.S.

3. *Proc. of the Joint Internatl. Conf. on Mathematical Methods and Supercomputing in Nuclear Applications*, KFK, 1993, ISBN 3-923704-11-9.

4. *Advanced Monte Carlo Computer Programs for Radiation Transport, Saclay (France), 27-29 April 1993*, Nuclear Energy Agency Organization for Economic Co-Operation and Development, Paris, 1995.

5. *High Performance Computing in Nuclear Applications: Adaptation of Computer Codes to Parallel Architectures—A State-of-the-Art Report by the Task Force of the NEA Nuclear Science Committee*, OECD/NEA, Paris, to be published.