Parallel Applications of the USNRC Consolidated Code

Jun Gan, Thomas Downar
1290 Nuclear Engineering Building
Purdue University
West Lafayette, IN, 47907-1290
ganj@ecn.purdue.edu, downar@ecn.purdue.edu

John Mahaffy
Applied Research Laboratory
The Pennsylvania State University
P.O. Box 30
State College, PA 16804
jhm@psu.edu

Jennifer Uhle
U.S. Nuclear Regulatory Commission
Washington, DC 20555
jxu1@nrc.gov

ABSTRACT

The United States Nuclear Regulatory Commission has developed the thermal-hydraulic analysis code TRAC-M to consolidate the capabilities of its suite of reactor safety analysis codes. One of the requirements for the new consolidated code is that it supports parallel computations to extend code functionality and to improve execution speed. A flexible request driven Exterior Communication Interface (ECI) was developed at Penn State University for use with the consolidated code and has enabled distributed parallel computing. This paper reports the application of TRAC-M and the ECI at Purdue University to a series of practical nuclear reactor problems. The performance of the consolidated code is studied on a shared memory machine, DEC Alpha 8400, in which a Large Break Loss of Coolant Accident (LBLOCA) analysis is applied for the safety analysis of the new generation reactor, AP600. The problem demonstrates the importance of balancing the computational for practical applications. Other computational platforms are also examined, to include the implementation of Linux and Windows OS on multiprocessor PCs. In general, the parallel performance on UNIX and Linux platforms is found to be the most stable and efficient.

Keywords: reactor simulation, thermal-hydraulics, parallel computing, message passing

1. Introduction

The Transient Reactor Analysis Code, TRAC-M[1,2], has been developed by the United States Nuclear Regulatory Commission (NRC) for the safety analysis of commercial Light Water Reactors. The goal is to consolidate the capabilities of its four different thermal-hydraulic system analysis codes (TRAC-P, TRAC-B, RAMONA, and RELAP5). A primary motivation for this redevelopment effort is to update the physical models for the next generation of nuclear reactors, and to improve computational efficiency.

TRAC-M solves the two fluid mass, energy, and momentum equations using the well established Stability Enhanced Two Step (SETS) method developed by Mahaffy [1]. At each time step a solution is initiated with the stabilizer motion equation which is linear in the unknown stabilizer velocities. After solving the stabilizer motion equations, the version of SETS implemented in TRAC-M proceeds to solve the "basic" equations for motion, mass and energy. Apart from the use of stabilizer velocities in the momentum transport term of the motion equation, these equations are equivalent to the standard semi-implicit method. After solving for the velocity, temperature and pressure field, the final step in the SETS method is to solve the stabilizer mass and energy equations. At this point, the new time velocities have been determined and are treated as constants in the solution of the equations. The density and enthalpy are then determined by solving the mass and energy equation. When the fluid flow equations are solved in parallel using more than one task, it is necessary to maintain implicit coupling between tasks to avoid numerical instabilities. A detailed description of how this done, as well as a detailed description of the implementation of the SETS method in TRAC-M is provided by Mahaffy in references [9-11].

TRAC-M is designed to provide simulations of transients involving all aspects of a nuclear reactor's primary and secondary coolant systems. However, it is generally not practical to include all of physical models into one code. Therefore it was decided to employ code modularity in which separate code modules are used to provide the solution of different physical models. TRAC-M is typically used to model the nuclear reactor primary and secondary fluid systems. An example of a reactor safety problem requiring a different code module would be the analysis of the reactor containment. The ECI would be used to couple TRAC-M to a detailed containment model constructed by using a containment analysis code such as CONTAIN.

The distributed computing ability in TRAC-M is initiated by an extension of the Exterior Communication Interface (ECI) concept of component modularity and inter-component communications. "Exterior" components are specified for the connected components that are exterior to TRAC-M and modeled by another program, or another copy of code itself. For example, the primary system and secondary system of a reactor could be modeled using separate copies of the TRAC-M code which can be executed in parallel. The ECI is employed to execute data transfers in each copy of the code. Currently, message passing services are provided by the Parallel Virtual Machine (PVM) protocol, however, the message passing calls are isolated and can be easily replaced by alternate libraries such as MPI.

This paper is focused on the parallel applications of the consolidated code in the safety analysis of commercial nuclear reactors. A study is performed on various platforms that are representative of the platforms found in the reactor analysis user community. The remainder of the paper is organized as follows. In the next section the general aspects of parallelism and an overview of the ECI is provided. In Section 3 the parallel applications are implemented for the safety analysis of the Large Break Loss of Coolant Accident (LBLOCA) for the Westinghouse advanced reactor, AP600. Load balance studies are performed using various domain decomposition strategies. In Section 4 the performance of the interprocess message passing library (PVM) is tested for various platforms, and results are compared for the parallel application of a Pressurized Thermal Shock (PTS) test problem. Finally, a summary and conclusions, as well as the direction of future research are provided in section 5.

2. Parallelism Using the Exterior Communication Interface

Component-based parallelism is implemented in TRAC-M by the ECI which also extends the capability of the consolidated code by facilitating the coupling to other programs for special physical models. This is depicted in Figure 1 which illustrates the basic structure of the parallel model which enables both domain decomposition of the physical model and functional decomposition based on the equations being solved. For example, the physical model can be decomposed into various parts of the reactor plant such as the steam generators and reactor primary system and can be executed as separate TRAC-M

processes as depicted in the figure. The reactor model can also be functionally decomposed as shown in the figure where the reactor core power distribution is determined by solving the neutron spatial kinetics equations with a separate module. The exchange of information required by the coupling of the physical models or field equation solutions is managed by the ECI. Under most circumstances one copy of the TRAC-M code acts as a central process to coordinate the calculation. However, it is also possible to use other programs as the central process.

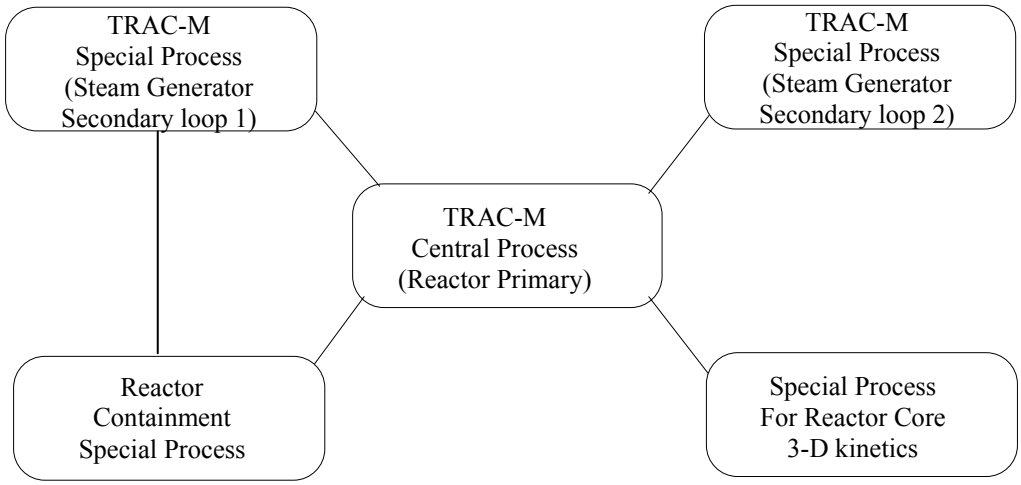


Fig. 1 Processes in a Reactor Transient Simulation

Communication is not restricted between the central process and the satellite processes. Fig. 1 illustrates a direct communication path between a satellite consolidated code process and the special containment process, providing information on fluid flow from the loop to the containment. This data is exchanged directly between these processes without passing through the central process.

The ECI performs the actual transfers through the use of transfer tables and buffered interprocess communication. It is based on a dynamic configuration of interprocess data transfers. The content and timing of data transfers are established during the initialization of the system calculation based upon exchange of requests between processes. Processes receive and store these requests and form a complete list of information that it must transmit and receive at each synchronization point in the calculation.

These lists are stored in the form of transfer tables illustrated in Figure 2. Each element of the list contains a “from” and a “to” pointer variable. For transmissions the table’s “from” pointer is associated with the location in memory of a variable requested by another process, and the “to” pointer is associated with a location in an interprocess communication buffer that will be transmitted to the requesting process by a message passing service which is currently provided by PVM.

Figure 2 also illustrates the key features of the data transfer between two consolidated code processes. The first stage is internal to one process. When a scheduled transfer is triggered at a synchronization point, the export transfer buffer is filled by the operation of pointer equivalencing. The data of the export buffer are then transferred to the import buffer in the required process using the interprocess message passing library. The final stage of table driven data transfer is a mirror image of the first stage. Here the “from” pointers are associated with elements of the import transfer buffer, and the “to” pointer associated with the final storage locations for the data. In addition to the above data communication, the ECI automatically drives an exchange of basic status information at each synchronization point. If a fatal error has occurred it instructs the other processes to terminate execution.

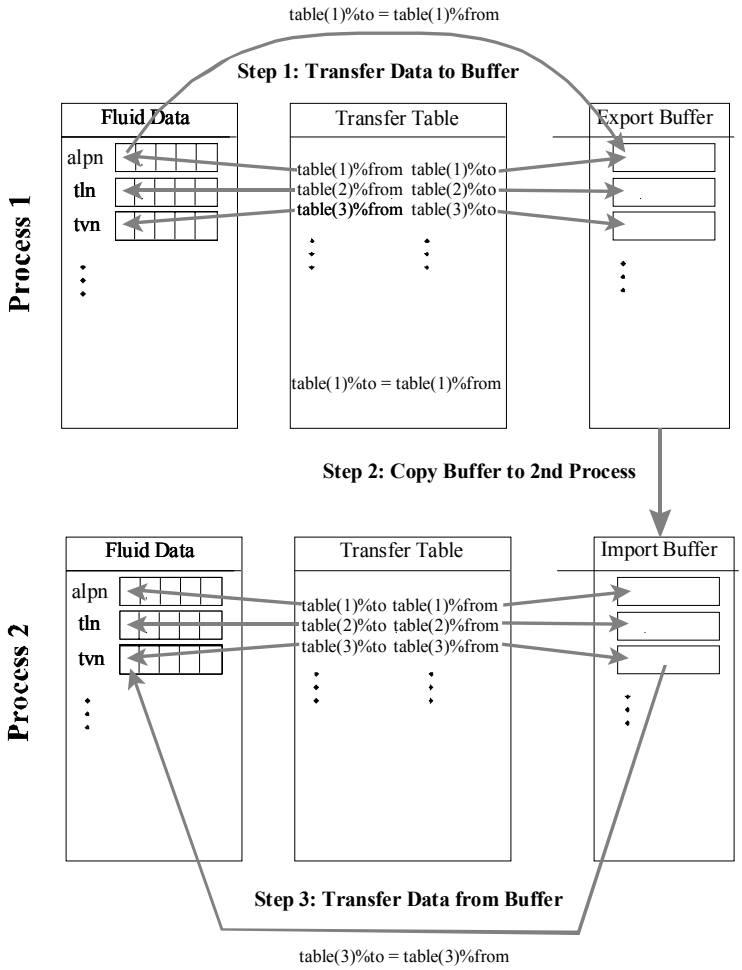


Fig. 2 Table Driven Data Transfer

3. Parallel Applications – AP600 LBLOCA Model

This application involves the parallelization of a large break loss of feed-water transient for an advanced Light Water Reactor, AP600. This is one of the important reactor safety transients and typically represents a significant computational burden in the overall assessment of the reactor plant safety. The system model is illustrated in Figure 3 and the break is modeled to occur at one of the cold legs for feedwater which cools the reactor. There are two 3-D vessels in the AP600 reactor model, one is the core itself and the other is the downcomer, which is illustrated in the Figure 4.

To demonstrate the effect of load distribution on parallel performance, three domain decompositions of the system of AP600 were tested:

- Model A – two tasks model: reactor vessel / 1D loops (which include feed-water and steam lines, steam generators, turbine, etc.)
- Model B – three tasks model: separate downcomer vessel from core vessel into a new task, i.e. core vessel / downcomer vessel / 1D loops

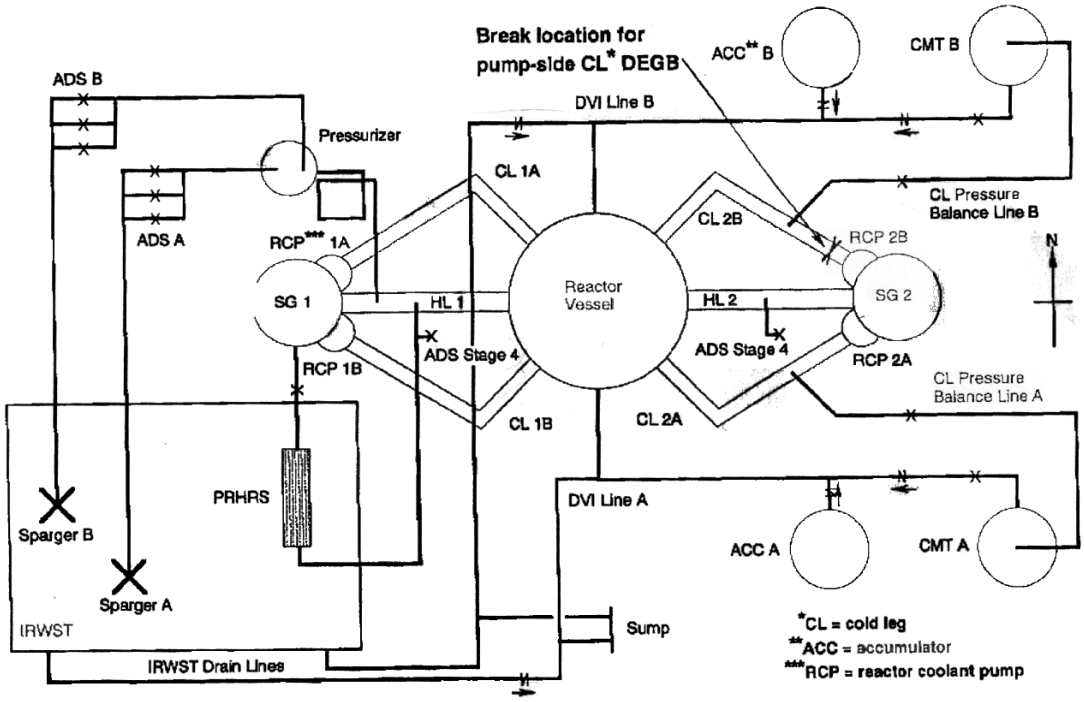


Fig. 3 AP600 System Model for LBLOCA Analysis

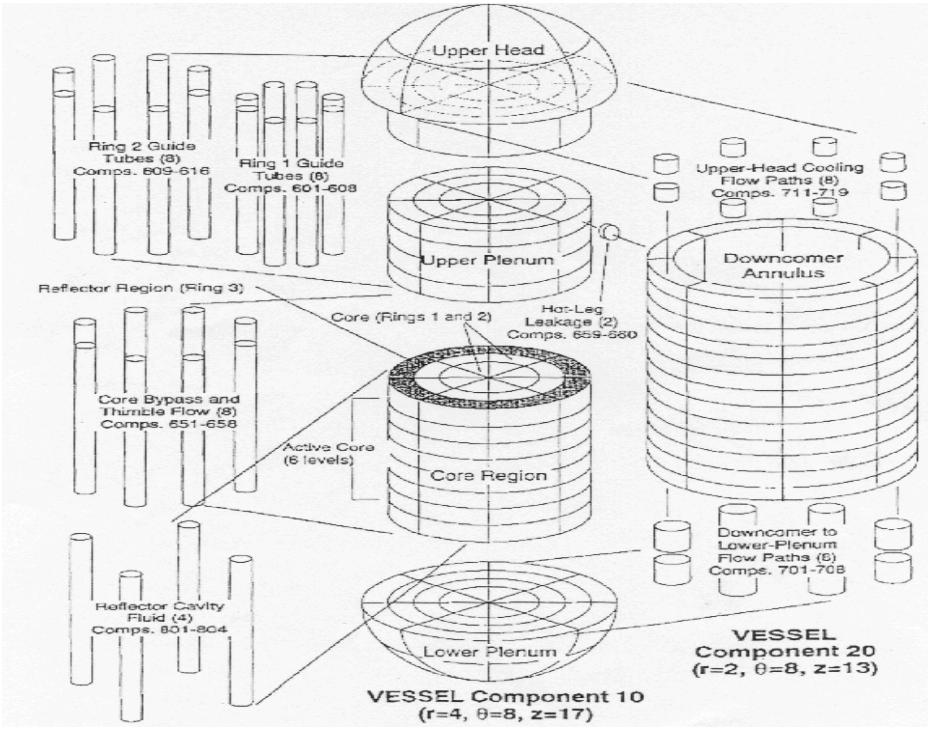


Fig. 4 AP600 Reactor Vessel Model for LBLOCA Analysis

- Model C – two tasks model: include downcomer vessel into 1D loops task, i.e. core vessel / downcomer vessel + 1D loops

The following table describes the load distributions for each of these domain decompositions, where total thermal-hydraulics component cell number is shown in the column labeled TH cells and the ratio of that of 3-D to 1-D is listed in the bracket. The total number of heat structure mesh is listed in the column labeled HS mesh.

Table 1. Load Distributions of Three Domain Decompositions in AP600 LBLOCA Model

Model	Task	TH Cells	HS Mesh	Total
A	1 (Core + Downcomer)	960 (752:208)	353	1313
	2 (1D loops)	388	330	718
B	1 (Core)	728 (544:184)	283	1011
	2 (Downcomer)	232 (208:24)	70	302
	3 (1D loops)	388	330	718
C	1 (Core)	728 (544:184)	283	1011
	2 (Downcomer +1D loops)	620 (208:402)	400	1020

As indicated in the profile in Table 1 the most balanced load is Model C which would therefore be expected to provide the best efficiency and this is confirmed by the results summarized in Table 2. The TRAC-M code was executed on an 8-CPU DEC Alpha 8400 which had a serial runtime of 145.498 seconds for 200 time steps of this problem. The parallel application results are summarized in the Table 2, where T_{calt} is the calculation time, T_{comm_t} is the communication time, and T_{idle_t} is the idle time on each process. The calculation time illustrates the actual load distribution of three domain decompositions, which is consistent with the cell profile in the Table 1. The amount of data transfer increases as the number of tasks increases and therefore the overall communication time of the 3-task model (Model B) is the largest. In the case of idle time, task 1 has the largest load in each model and therefore the other tasks will remain idle until task 1 reaches the synchronization points and therefore have the largest T_{idle_t}. As expected Model C has the best overall efficiency of 73.7% because it has the best load distribution.

Table 2. Parallel Performance Summary for Various Domain Decompositions

Model	Task	T_{calt}	T_{comm_t}	T_{idle_t}	Tot_t	Speedup	Efficiency
A	1	106.787	3.915	1.113	111.815	1.304	65.2%
	2	42.829	4.085	64.418	111.332		
B	1	89.418	8.243	1.761	99.422	1.468	48.9%
	2	25.753	9.484	63.711	98.948		
	3	43.372	9.195	46.438	99.005		
C	1	88.911	5.047	4.859	98.817	1.475	73.7%
	2	67.492	6.085	24.896	98.473		

4. Performance Comparison on Various Platforms

The use of personal computers (PC) for scientific computing has become increasingly popular because of their very favorable performance to cost ratio. In the work here, the performance of multiprocessor PCs was compared to the DEC Alpha for TRAC-M parallel applications. Specifically, the following platforms were analyzed with the Operating Systems indicated: (1) Xeon PII (dual CPU: 400MHz; Windows 2000); (2) Xeon PIII (4-CPU: 550MHz; Linux, RedHat); (3) Xeon PIII (dual CPU: 800MHz; Linux, RedHat). A similar level of compiler optimization was used on all platforms.

First the performance of the PVM message passing library was investigated on each platform. The bandwidth and latency are measured using the "Ping-Pong" message passing method. Two PVM protocols were implemented in the consolidated code: *psend* and *precv* for bulk data transfer and *pack/send* and *recv/unpack* for status checking. The results are compared in Figures 5 and 6 for each protocol and for the four platforms used in this work. For the applications tested here, the largest message size at each synchronization point is normally less than ten thousand bytes. Therefore, the communication time will be latency dominated for the problems here. As indicated in the Figures, the performance of PVM on the two Linux systems is comparable to that on DEC Alpha, whereas the performance of PVM on Windows 2000 is very poor. This is consistent with the experience of other researchers, for example Hollingsworth et. al. [8] observed that the longer code paths and poor memory hierarchy utilization result in very poor performance of data transfer on NT compared to Linux. Based on these results, Linux appears to be the preferred choice for parallel processing on PCs. This was confirmed with the parallel execution of TRAC-M for practical applications.

The application used here is a reactor pressure vessel model used to analyze Pressurized Thermal Shock (PTS). The model is illustrated in Figure 7 and consists of a 3-D reactor vessel, an upper and lower plenum, and four flow loops. In the application here a two-task model is used: the 3-D reactor vessel is assigned to one process (Task 1) and the remainder of the model which is primarily 1-D flow loops is assigned to the other process (Task 2). The two-task parallel model was executed on each of the four platforms, and the results are summarized in Table 3. One of the first observations from the results is the significant difference in the execution times across platforms for the 3-D sections (Task 1) versus the 1-D sections (Task 2). This is caused primarily by the difference in the instruction sets generated by each compiler, even though a similar level of optimization was used. The implication of this is that balancing the load through domain decomposition can not be based solely on the numbers of cells or mesh per task, but also on the relative performance of the compiler with the types of operations associated with each cell or mesh. The results for the WIN2000 platform are especially disappointing and suggest that WIN2000 is currently not a competitive option for multiprocessing on a PC.

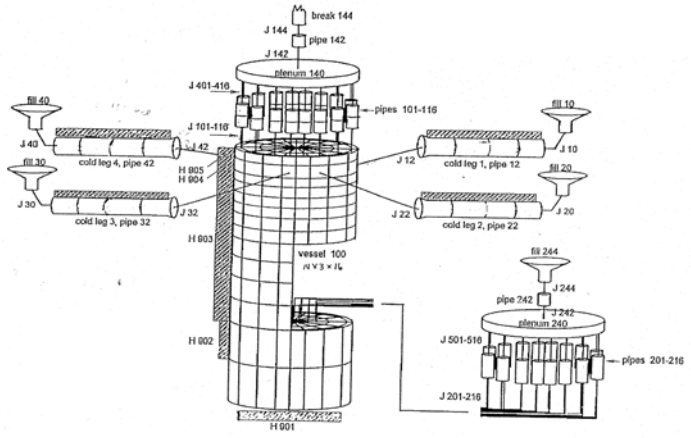
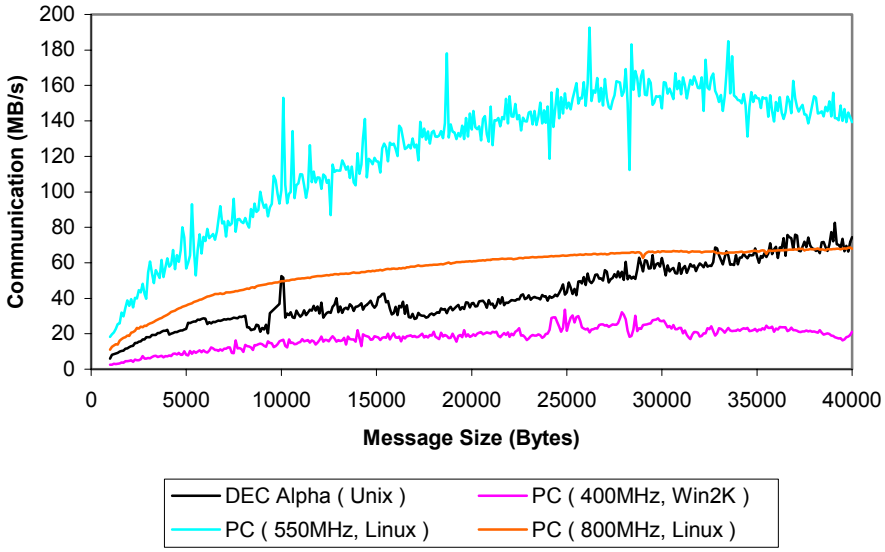


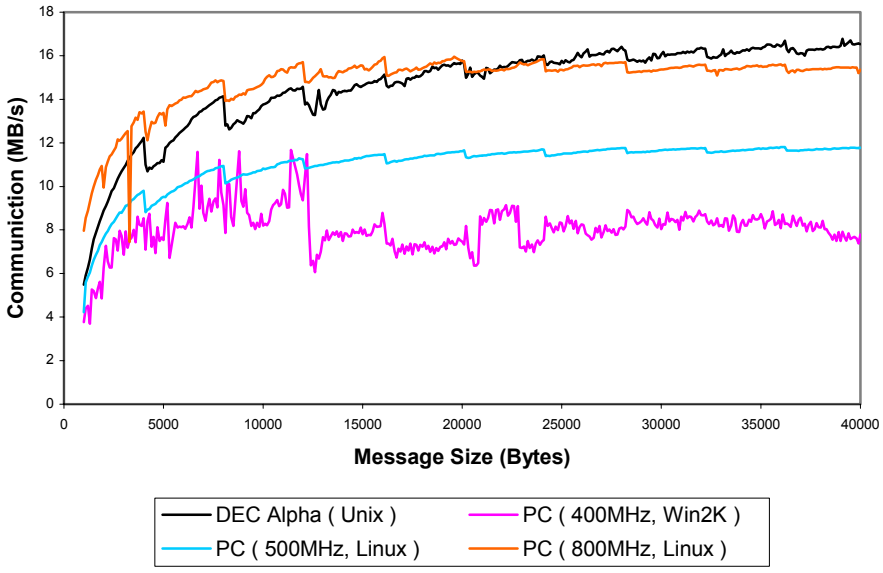
Fig. 7 Model of PTS Problem

Fig. 5 Bandwidth on Various Platforms using *psend* and *precv* PVM Protocols



Latency:
DEC Alpha: 32.157 µs;
PC (400Mhz): 343.750 µs
PC (550Mhz): 51.000 µs;
PC (800Mhz): 74.875 µs ;

Fig. 6 Bandwidth on Various Platforms using *pack/send* and *recv/unpack* PVM Protocols



Latency:
DEC Alpha: 33.672 µs;
PC (400Mhz): 304.688 µs;
PC (550Mhz): 26.667 µs;
PC (800Mhz): 69.000 µs

Table 3. Execution Results for PTS Application

50 time steps

Platform	Serial	Task	T_{calt}	T_{comm_t}	T_{idle_t}	Tot_t (CPU)	Speedup	Efficiency
Triton Dual-CPU Linux	80.300	1	46.830	0.270	-	47.100	1.705	85.2%
		2	38.940	0.330	7.83	39.270		
Lambda Four-CPU Linux	116.79	1	67.350	0.440	-	67.790	1.723	86.1%
		2	55.850	0.530	11.41	56.380		
Res30 DEC Alpha UNIX	81.732	1	18.727	1.720	47.886	20.447	1.206	60.3%
		2	66.035	1.735	-	67.770		
Xeon Dual-CPU WIN2000	76.594	1	30.047	17.719	12.968	47.766	1.261	63.1%
		2	48.656	12.078	20.547	60.734		

5. Conclusions

The flexible component-based parallelism imbedded in the USNRC code TRAC-M by the Exterior Component Interface (ECI) considerably enhances the code's functionality and offers the potential for enhanced computational efficiency through multiprocessing. Parallel applications of the code were reported here for several multiprocessor platforms and operating systems. Reasonable parallel efficiencies were achievable by balancing the computational load which must be based not just on the balanced assignment of cells and mesh to each task, but also by examining the relative performance of each compiler on the types of operations associated with each cell and mesh. The most promising multiprocessing results were achieved on UNIX and Linux platforms, and because of the favorable price to performance ratio, a PC with a Linux OS appears to be the most attractive option for the applications studied here.

Although the applications examined here were practical examples of reactor safety analysis, the size of the problems targeted for future work will be considerably larger. Specifically, problems are currently being analyzed such as a PWR Main Steam Line Break that are much more CPU intensive and require the solution of both the thermal-hydraulics as well as the neutronics equations. Current efforts are underway to parallelize the 3-D kinetics module (based on the PARCS spatial kinetics code), which is coupled to TRAC-M. Because the solution of the reactor core spatial kinetics problem is tightly coupled and exhibits primarily fine grain parallelism, threaded parallel models such as OpenMP are being developed for PARCS. The coupled neutronics and thermal-hydraulics equations will then be solved with TRAC-M/PARCS using the ECI, which will also be further developed in the future to accommodate alternate parallel models and to facilitate user access to alternate parallel platforms.

Acknowledgements

This work was performed under the auspices of the US Nuclear Regulatory Commission.

References

1. Schnurr, N. M. et al., TRAC-PF1/MOD2 Theory manual, Los Alamos National Laboratory Report LA-12031-M, US Nuclear regulatory Commission Report NUREG/CR-5673, 1992.

2. Spore, J. W. et. al., TRAC-M Programmer's Guide, Nuclear Systems Design and Analysis Group TSA-10, Los Alamos National Laboratory, Los Alamos, New Mexico, May 1998.

3. Bertsekas, D. P. and Tsitsiklis, J. N., Parallel and Distributed Computation Numerical Methods, Prentice Hall, NJ, 1989.

4. Kumar, V., Grama, A., Gupta, A., and Karypis, G., Introduction to Parallel Computing Design and Analysis of Algorithms, The Benjamin/Cummings Publishing company, Inc., RedWood City, CA, 1994.

5. Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R., and Sunderam, V., PVM: Parallel Virtual Machine, 1994, Cambridge, MA: The MIT Press.

6. Geist, A., Kohl, J., and Papadopoulos, P., PVM and MPI: A Comparison of Features, Calculateurs Paralleles Vol. 8 No. 2, P137-150, June 1996.

7. Gropp, W., Lusk, E., and Skjellum, A., Using MPI: Portable Parallel Programming with the Message Passing Interface, 1995 Cambridge, MA: MIT Press.

8. Hollingsworth, J. K., Guven, E., and Akinlar, C., Benchmarking a Network of PCs Running Parallel Applications, 1998 IEEE International Performance, Computing and Communications Conference. Proceedings (Cat. No.98CH36191). IEEE. 1998, pp.1-7. Tempe/Phoenix, AR, USA, February 16-18, 1998.

9. Mahaffy, J. H., Uhle, J., Dearing, J., Downar, T., Johns, R., and Murray, C., Architecture of the USNRC Consolidated Code, Proceeding of ICONE 8, 8th International Conference on Nuclear Engineering, Baltimore, MD, USA, April 2-6, 2000.

10. Mahaffy, J. H., Downar, T., and Wang, Q., Solution of Flow Equations in the USNRC Consolidated Code, International Meeting on "Best-Estimate" Methods in Nuclear Installation Safety Analysis (BE-2000) Washington, DC, November, USA, 2000.

11. Mahaffy, J. H. and Uhle, J., Distributed/Parallel Reactor Simulations Using the USNRC Consolidated Code, International Meeting on "Best-Estimate" Methods in Nuclear Installation Safety Analysis (BE-2000) Washington, DC, November, USA, 2000.