A Parallel Virtual Machine Interface for CATHENA

Darryl Dormuth

Safety Thermalhydraulics Branch
Atomic Energy of Canada Limited, Whiteshell Laboratories
Pinawa, Manitoba R0E 1L0

Abstract

In pursuit of best-estimate analysis for CANDU® safety and design issues, several computer codes that model such processes as systems thermalhydraulics, fuel behaviour, reactor kinetics, and reactor controllers will be required to interact with each other to facilitate the modelling of integrated effects among reactor systems. It would be cumbersome, in terms of usage, software management, and quality assurance practices, to incorporate all potential numerical models into a single computer code. Instead, a "coupled-code" methodology can be employed that relies on message-passing via computer network protocols to exchange the data among the computer codes. This approach has the advantage of maintaining the computer codes as separate entities which makes their usage, software management and quality assurance easier. In this paper, the coupled-code methodology is applied to the AECL thermalhydraulics code CATHENA (CANadian THErmalhydraulic Network Algorithm) so that it can be used in conjunction with other safety and design codes for better-estimate analyses of reactor behaviour.

Presented at the 19th Annual Canadian Nuclear Society Conference
1998 October 18-21
Toronto, Ontario

1. INTRODUCTION

When design engineers need to examine a condition or event that could affect the safety of a reactor, they customarily utilize computer codes to aid in their examination. Often two or more computer codes are needed to analyze such events because the models for the affected physical systems are not contained in a single code. If significant feedback occurs between physical systems that are not modelled by a single code then information (usually in the form of time-dependent boundary conditions) must be exchanged between the codes to capture the feedback behaviour [1,2]. Typically this information is exchanged manually between codes either in an iterative procedure until a converged solution is obtained or in a "start-stop" procedure over small time intervals [3,4]. This form of information exchange has limitations, most notably the time required to do the analysis and the difficulty in capturing rapid feedback among physical systems such as void-reactivity feedback encountered during some postulated events. Automating this information exchange would reduce the analysis time and provide a more efficient way of capturing feedback effects among reactor systems that are modelled by different codes.

One way to automate information exchange between separate codes is to combine them into a single executable [5]. This has the advantage of fast information transfer as all data remains in memory but can require considerable computer resources to load and execute if the original codes are large. Also, the combined-code executable becomes a new code itself and may be subjected to software quality assurance practices and procedures which can add overhead to its development and maintenance. An alternative to this approach is to keep the codes separate and couple them with a computer network through which pertinent data can pass. This approach has the advantage of maintaining the computer codes as separate entities which makes their usage, software management and quality assurance easier but it does require that an interface be built in each code so that the necessary data can be exchanged in a consistent and timely manner [6].

In this paper, the design criteria and implementation of a network interface are described for the system thermalhydraulic code CATHENA which gives it the ability to communicate with a variety of other safety analysis codes. This network interface makes use of a software package called PVM (Parallel Virtual Machine) which is a library of subroutines that performs all the necessary functions for a group of processes, potentially located on different computers, to work as a collective. It is a shareware package maintained and supported by Oak Ridge National Laboratories and is available for UNIX and Microsoft Windows operating systems.

Implementation of the CATHENA network interface is demonstrated with a simulation of a power reduction in a CANDU 6 reactor. For this example, CATHENA is coupled to two other codes - a reactor controller code for Point Lepreau and a point neutron kinetics code. This example was chosen because the controller and point kinetics models reside in the current reference version of CATHENA permitting a direct means of verifying the network interface. Also, once the interface is successfully implemented with these models, the groundwork is laid for connections to other controller models such as those for Gentilly-2 or Wolsong and to other reactor kinetics models such as CERBERUS.

2. DESIGN REQUIREMENTS

To make the network interface as accessible as possible to CATHENA users and to minimize its development and maintenance, several design criteria were considered.

Robustness

The intent of the network interface is to be able to couple CATHENA to many types of analysis codes. In terms of usability and code maintenance it is desirable to have one interface that can handle all the desired connections, such as to

a) multi-dimensional reactor kinetics codes,
b) reactor controller codes,
c) finite-element codes for structural behaviour of pipe components,
d) fuel behaviour codes, or
e) aerosol codes for aerosol transport in the primary heat transport system.

One Reference Version

Each reference version of CATHENA must contain all the source for the network interface which means the interface has to be coded so that it works in the same way on all supported CATHENA computer platforms. Also, CATHENA must operate unimpaired if network connections are not specified or network software is absent from the computer being used. It should be noted that if a network connection is requested and network software is absent then an appropriate error message should be provided to the user.

Maintain CATHENA Solution Method

As stated above, the network interface must be robust enough to handle connections to many types of safety analysis codes and be available on supported computer platforms. This requires that a protocol be established between CATHENA and any code connecting to it so that data can be transferred in a correct and consistent manner. As CATHENA is a finite-difference code (in time and space), it steps through the time domain of the problem by executing a main loop of coding for varying sizes of time intervals (commonly referred to as time-steps). CATHENA also has the ability of repeating a time-step if conditions in the thermalhydraulic system dictate that a smaller time interval was needed to capture the dynamics.

Codes that connect to CATHENA must be able to follow CATHENA's time-step solution method. If data are transferred at every CATHENA time-step then any connecting code must allow for varying sizes of time intervals and be able to consider the possibility that a time-step will be repeated. It is also important that data transfers occur at a location in the code that is executed at every time-step. Often there are sections of coding that are not always executed, depending on system conditions, and data transfers in the sections are to be avoided. Based on these requirements, the network interface must provide enough information to a connecting code for it to handle CATHENA's solution method and the data transfers must be at a location within CATHENA where all the pertinent thermalhydraulic data is accessible at every time-step.

Minimize Additional Code Maintenance
CATHENA is written in FORTRAN-77 and to minimize code maintenance effort the network interface should also be written using standard FORTRAN-77 coding practices. The routines used to access the network must be portable to all supported computer platforms which means that if third-party software is used it too must be available on these supported systems.

3. IMPLEMENTATION

The approach taken in implementing the design requirements of the CATHENA interface was to minimize the amount of new internal coding and to maintain the basic processing structure. Although the potential for parallelism exists using the network interface, only sequential operations were considered for the first version. This means that for each data transfer to an external code CATHENA will send its data and wait, the external code will receive the data and perform its calculations for the given time interval, CATHENA will receive the results, and then CATHENA will continue its processing until the next time for a data exchange. By implementing a sequential processing approach initially, new internal coding is minimized and CATHENA's basic processing structure is maintained.

One of the first challenges in building a coupled-code system is determining how start the simulation. Each code must start execution, establish a connection to the other codes in the system, and discover what data is to be exchanged. It was decided for the CATHENA network interface that the user would start CATHENA and then CATHENA would start the external codes. Using this start-up procedure, CATHENA becomes responsible for starting up the other codes and synchronizing communication instead of the user. Information needed to initiate the execution of each remote code would be contained in the CATHENA input file. To keep the interface as robust as possible, the input file would contain three pieces of information to start a remote process:

1) The executable file name and location of the remote code. The location would include the network node and the directory on that node where the executable resides.
2) The directory on the node where working files will be stored.
3) The name of the input file for the remote code. This input file will be contained in the working directory.

The input for the network interface is provided in a similar fashion as for the point kinetics and output models. In the System Control Model section of the input file, the user creates a 'REMOTE PROCESS' model that contains all the appropriate information regarding the start-up and data transmission.

When the remote code is started by CATHENA it will read the two character strings containing the working directory and input file name, call an operating system routine to point to the working directory, and then open the input file and read its contents. Any remote code that is to be coupled to CATHENA through this interface must adhere to this protocol. Each remote code should have some mechanism for determining whether it was

started by a user to be run as an independent process or whether is was started by CATHENA as part of a coupled-code simulation. The PVM software package (discussed below) provides an easy way of testing if a process was started by a user or another process.

Parallel Virtual Machine (PVM) is a software package that enables a collection of heterogeneous computers to be used as a single computational resource [7]. The name Parallel Virtual Machine refers to the virtual parallel computer that is created when a group of computers are networked together with the PVM software. PVM was developed at the Oak Ridge National Laboratories and the University of Tennessee in the late 1980's and early 1990's and is made available as shareware to the scientific community. It employs a message-passing form of distributed processing in which data is exchanged in packets sent across the network and it is available for UNIX and MS-WINDOWS operating systems that use Transfer Control Protocol/Internet Protocol (TCP/IP).

PVM consists of essentially two parts: a background process that runs on each computer in the virtual machine (commonly known as a daemon process) and a library of callable C or FORTRAN routines. Each computer code, such as CATHENA, uses routines from this library to perform such tasks as starting a process on a remote computer node, transmitting and receiving data from other processes, checking the status of the virtual machine, and halting remote processes. A code will communicate with another code by first sending a message to its local PVM Daemon process (PVMD), this PVMD will then send the message to the PVMD on the appropriate remote computer, and that PVMD will pass the message onto the second code. All inter-computer communication is done through the PVMDs using TCP/IP socket protocols.

Once the remote process is started, the next steps are to establish what data are to be transferred, the order in which they are to be sent and received, and the frequency of transmissions. For the first version of the interface it was decided that this information would be contained in the input files of each code. This means that before commencing a simulation the user must verify that the lists between the two codes match so that data are sent and received in the appropriate order and at the appropriate frequency. The use of the PVM message passing routines provides a flexible method of transmitting the data. Each message would contain a set of data that would be structured according to what the user requests in the input file. The user would also specify in the input file the order in which the messages would be transmitted.

The start-up and message-passing form of data transmission described above establishes the basic protocol for the CATHENA network interface. Any code that is to be coupled to CATHENA through this version of the interface must follow the same protocol and it must also include the applicable PVM routines. The PVM routines are also available for some UNIX scripting languages such as PERL so that driver programs could be written around some codes to give them access to the CATHENA interface. It should also be noted that codes can be networked on the same computer through this interface, although PVM and the appropriate networking software must be available.

4. EXAMPLE

To illustrate the capabilities of the CATHENA network interface, a power reduction transient in a CANDU 6 reactor was chosen as an example case. In this transient, the power was reduced from 100% to 77% full power at a rate of 0.5% per second and requires thermalhydraulic, reactor kinetics, and reactor controller models to simulate the event. CATHENA contains a reactor controller model for Point Lepreau (called LEPCON - LEPreau CONtoller) developed by New Brunswick Power [8] and a point neutron kinetics model so this event can be simulated by CATHENA itself. By removing the kinetics and controller models from CATHENA and making them into separate stand-alone executables, the event can also be simulated using these two codes coupled to CATHENA through the PVM network interface and the results can be directly compared to those produced by the single CATHENA simulation.

Figure 1 shows how the parallel virtual machine is set up for this example. The removed kinetics model is named POKIN (POint KINetics) and the controller model, LEPCON-S (LEPreau CONtroller Stand-alone). As mentioned in Section 3, the coupled calculations are done in a sequential manner and in Figure 2 the processing order for this simulation is presented. The number of data items in each message are shown over the arrows. No reduction in simulation time was expected from the coupled-code simulation versus the single CATHENA run because no parallelism in the computations was exploited. In fact a slight increase in simulation time was seen because of the overhead in network communications.

Plots of the power transients from the two cases are shown in Figure 3 and indicate a very good comparison between the two simulations. The slight deviations between the two transients are attributable to differences in numerical round-off. This example not only demonstrates that the PVM network interface works for multiple connections it also illustrates a way of verifying that it is functioning correctly.

5. SUMMARY

In this paper, the design and implementation of the PVM network interface for CATHENA were presented. It was shown that using the PVM software package, a network interface can be built to meet the requirements outlined in Section 2. A basic protocol was established that permits the coupling of CATHENA to many types of safety analysis codes. To demonstrate this, an example was provided in Section 4 in which a power reduction in a typical CANDU 6 reactor was modelled with reactor controller and reactor kinetics codes connected to CATHENA through the PVM interface. Future investigations will include coupling CATHENA to a multi-dimensional reactor kinetics code and other reactor controller models.

6. ACKNOWLEDMENTS

The author would like to express his thanks to Dr. Bruce Hanna and Mr. Marc Lavack for their comments and to New Brunswick Power for their permission to reference the Point Lepreau controller model. This work was partially funded by Hydro Québec, New Brunswick Power and AECL, through the CANDU® Owners Group under WPIR 12, and their support is appreciated.

7. REFERENCES

[1] DeVaal, J., Gauld, J., Klein, M., McDonald, B., Dormuth, D., Whitehouse, D., Lavack, M., Chiang, H-W., and Wilken, B., Parallel Computing Applied to the Modelling of the Complex Interacting Phenomena in Severe Fuel Damage Experiments, presented at the ICHMT International Seminar of Heat Transfer in Severe Reactor Accidents in Cesme, Turkey, 21-26 May 1995.

[2] Tetner, A. et. al., Advances in Parallel Computing for Reactor Analysis and Safety, Communications of the ACM, Vol. 37(4), 1994.

[3] Dormuth, D. and McDonald, B., A Parametric Study of Thermalhydraulic Factors Affecting Pressure Tube Rupture, CANDU Owners Group Report COG-91-307 (Protected), 1993.

[4] Liu, W., McDonald, B., Richards, D., Gold, M., Rouben, B., Barkman, J., Sills, H., Carlucci, L. and Luxat, J., Thermal-Hydraulic Interfacing Code Modules for CANDU Reactors, presented at the OECD/CSNI Workshop on Transient Thermal-Hydraulic and Neutronic Code Requirements, Annapolis MD, November 5-8, 1996.

[5] Carlucci, L., Gauld, J., Richards, D. and Arimescu, V., Coupling Subroutine Version of ELOCA for High-Temperature Fuel Behaviour to CATHENA System Thermalhydraulics Code, presented at the 20th Annual CNS Simulation Symposium, Montreal PQ, 1996.

[6] McDonald, B. Wallace, D. Hanna, B. and Dormuth, D., INTARES - A Concept Code for Integrated Thermalhydraulics and Aerosol Safety Analysis, presented at the CSNI Workshop on Aerosol Behaviour and Thermal-Hydraulics in Fontenay-aux-Roses, France, 26-28 November 1991.

[7] Geist, A., Beguelin, J., Dongarra, J., Jiang, W., Manchek, R., and Sunderam, V., PVM: Parallel Virtual Machine. A User's Guide and Tutorial for Networked Parallel Computing, MIT Press, Cambridge MA, 1994.

[8] Girard, Rene, LEPCON Abstract, Users Manual, and Program Description, New Brunswick Power Report IR-03500-51 Vol. 1,2,3, 1995.