2001 RELAP5 Users Seminar
Sun Valley, Idaho
September 5-8, 2001

An Executive Program For Use With RELAP5-3D©

W. L. Weaver
Bechtel BWXT Idaho, LLC
Idaho National Engineering and Environmental Laboratory
P.O. Box 1625
2525 North Fremont Ave.
Idaho Falls, ID 83415-3880

E. T. Tomlinson and D. L. Aumiller
Bechtel Bettis, Inc.
Bettis Atomic Power Laboratory
P.O. Box 79
West Mifflin, PA 15122-0079

Abstract

An executive program has been developed that coordinates the coupling of any number of other computer programs to perform integrated analyses of nuclear power reactor systems and related experimental facilities. The ability to couple programs allows the analyst to apply different analytical models to specific domains in the problem to achieve accurate results. The coupling is accomplished using the PVM message passing software and the executive program manages all phases of a coupled computation. It starts up and configures a PVM virtual machine, spawns all of the coupled processes on the PVM virtual machine, coordinates the time step size between the coupled codes, manages the production of printed and plottable output as well as restart files, and shuts the PVM virtual machine down at the end of the computation. The executive program also monitors the status of the coupled computation, repeating time steps as needed to obtain an accurate solution and terminating a coupled computation gracefully if one of the coupled processes is terminated by the computational node on which it is executing. This paper discusses the application of the executive to RELAP5-3D©.

1. Background

Several previous papers (Martin, 1995; Aumiller, 2001a; Weaver, 2001 and Aumiller, 2001b) have described the methodologies by which the RELAP5-3D© computer program (RELAP5-3D, 1999) may be coupled to another computer code either explicitly or semi-implicitly. The first two papers (Martin, 1995 and Aumiller, 2001a) describes how RELAP5-3D© was coupled explicitly to both another instance of RELAP5-3D© and to a Computational Fluid dynamics (CFD) code. The two final papers (Weaver, 2001 and Aumiller, 2001b) describes the methodology by which RELAP5-3D© can be coupled to another thermal-hydraulic code using a semi-implicit coupling methodology. Each of the coupling methodologies is discussed briefly to familiarize the reader with the concepts.

The basis of any coupling methodology is that the problem is divided into multiple domains where each domain can be simulated by a different computer program, or by a different "instance" of any of the programs. Figure 1 shows a schematic of a simulation problem in which there are two connections between the two problem domains. In Figure 1, volume 1 is adjacent to and connected to volume I, and volume 2 is adjacent to and connected to volume II. The boundary volumes in one of the domains (i.e. 1 and 2) represent normal volumes in the interior of the other computational domain (i.e. I and II). Information about these volumes must be passed between the domains at the coupling boundary to achieve an integrated solution. Coupling methodologies for passing this information between programs can be developed based on a number of different numerical methods.

A schematic of a fully explicit coupling methodology is shown in Figure 2. The dashed arrows in Figure 2 indicate data flow between the domains. In this coupling methodology, pressures in the boundary volumes are held constant during each time step and are updated at the end of the time step. The constant boundary pressures cause the stability of the coupled systems to be limited by the sonic Courant condition. As proof of this restriction, an analysis using fully explicit coupling without regard to the sonic Courant limit was performed and, as expected, numerical oscillations were observed (Aumiller, 2001a). This restriction makes fully explicit coupling impractical.

The semi-implicit coupling algorithm is shown schematically in Figure 3. Dashed arrows in Figure 3 indicate data exchanges between master and slave processes. To provide numerical stability in the semi-implicit algorithm, consistent new time velocities and pressures are required for both codes. To satisfy this requirement of consistent new-time velocities and pressures, the changes in the pressures for all volumes in the master computational domain are represented as linear functions of the mass and energy flow rates in the coupling junctions. The coefficients in these linear relations are transmitted to the slave process for the coupling volumes in the master process. Using these coefficients, the slave process can simultaneously calculate the mass and energy flow rates across the coupling location and the change in the pressure of the coupled volume in the master process. These mass and energy

flow rates are then transmitted back to the master process where they are used to compute the changes in pressure in a manner consistent with the slave process.

All of the coupling between RELAP5-3D© and the other codes has been performed using the Parallel Virtual Machine (PVM) message passing software developed at Oak Ridge National Laboratory (Geist, 1993). Data items are passed between the coupled codes in messages having unique message identifiers. In the original implementation of the coupling methodology (Weaver, 2001), RELAP5-3D© could only be coupled to one other computer code. Other restrictions inherent in the original implementation of the coupling methodology in RELAP5-3D© include the lack of coordination of the time step size to be used by the coupled codes, the inability to monitor the status of the two coupled codes, and the lack of coordination in the printed and plottable output of the two coupled codes. Each code was required to choose its own time step size which forced the user to use a fixed time step size (i.e., fixed so that they would use the same time step size for semi-implicit coupling) or fixed simulation time intervals for explicit coupling (the explicit coupling algorithm exchanges data between coupled processes at fixed time intervals). The inability to monitor the status of the code also forced the user to choose a time step size in such a way that the time step advancements would always be successful so that no time step repeats or time step size reductions would be necessary. The user also had to configure the PVM virtual machine by hand before executing the coupled calculation and the coupled codes had to be executed on the same computational node. The executive program was developed to remove these restrictions and to make the coupling methodology used by RELAP5-3D© more versatile.

2. Design of the Executive Program

The executive program was designed to remove the restrictions of the original implementation of the coupling methodology in RELAP5-3D©. It has five major responsibilities.

- It must configure the PVM virtual machine, starting the PVM daemon process on the computational nodes comprising the PVM virtual machine.
- It must start up the coupled processes on the computational nodes.
- It must tell each of the coupled processes what data to send to and what data to receive from the other processes.
- It must manage the time advancements of the coupled computation, coordinating the time step size between the several coupled codes, monitoring the status of the advancements and directing code backups and time step repeats as necessary.

- It must coordinate the production of printed and plottable output between the coupled codes so that computational results are available from all of the coupled codes at the same simulation times during the computation.

Programs that are coupled synchronously exchange data every time step while programs that are coupled asynchronously do not exchange data each time step. Programs that are semi-implicitly coupled must be coupled synchronously while programs that are explicitly coupled can be either synchronously or asynchronously coupled. A schematic of a prototypic coupled computation is show in Figure 4. In this coupled computation, RELAP5-3D© is coupled semi-implicitly to a CFD code which is used to model the coolant systems in a reactor power plant. It is also coupled synchronously to a code that performs a reactor power computation using a nodal neutron kinetics methodology. Lastly, it is coupled asynchronously to a containment analysis code. The data are exchanged among the several coupled codes as well as between the coupled codes and the executive program is also shown in Figure 4. What Figure 4 does not show is that each of the processes might be executing on a different computational node and that the communication among the processes would be carried over a network. Also not shown is that the computational nodes might be different computer architectures from different vendors (i.e., a mix of different types of UNIX workstations and PCs).

The user supplies the information needed by the executive program to initiate coupled computation such as the one shown in Figure 4 in an input file. The input file is divided into four sections (the input needed to accomplish the fourth and fifth responsibilities is contained in the same section of the input file). The input file for the executive program that was used to execute the semi-implicit verification test case is shown in Appendix A. The sections of the input file are delimited by reserved keywords. The first section of the input file is delimited by the keyword ‘virtual’. The lines following this keyword contain the names of the computational nodes to be used in the PVM virtual machine along with the location of the executable files to be used by that computational node and the location of the input files for the processes to be executed on that computational node (i.e., the working directory). Using this information, the executive program builds a PVM hostfile for the PVM virtual machine and starts the PVM daemon process on the one or more computational nodes.

The second section of the input file is delimited by the keyword ‘processes’ and specifies the processes (i.e., codes) to be executed on the several computational nodes. The names of the computational nodes contained in the first section of the input file become keywords in the second section of the input file. One or more coupled processes may be executed on each of the computational nodes. The specification of each coupled process begins with a unique name for that process as well as any command line parameters that are to be passed to that process as it begins its execution (i.e., names of input files, output file, etc.). The names of the coupled processes are used to distinguish multiple instances of the same executable file being executed in the PVM virtual machine. As mentioned earlier, each process is labelled as ‘synchronous’ or ‘asynchronous’. These labels denote whether or not the time step size for the process

is determined by the executive program. Synchronous processes, such as processes that are coupled semi-implicitly, need to use the same time step size for each advancement so their time step size is coordinated by the executive program. Asynchronous processes, such as processes coupled explicitly, need only exchange data at fixed intervals and it does not matter what size of time step they use to advance in time, only that they reach the same point in time to exchange data or write the plottable output, printable output or restart files. Changes to the timestep logic for codes asynchronously coupled through the executive are required to implement this capability.

The third section of the input file is delimited by the keyword 'messages' and specifies the data to be sent to and received from the other coupled processes. Each message uses the unique name of the sending and receiving process along with the specification of the data to be sent or received. The specification of messages occur in pairs, one message specification for the process sending the data and the other specification for the process receiving the data. The data items to be sent by the sending process are specified in terms that the sending code can understand and the same data items are specified to the receiving code in terms that the receiving code should understand. This means that the same data item may be specified by a different identifier for the sending and receiving processes. For example, the sending code may refer to the liquid density using the code variable 'rhof' while the receiving code may refer to the liquid density by the code variable 'rholiq'. The data specifications are sent to coupled codes as they appear in the third section of the input file. It is the responsibility of the individual coupled codes to understand their data specification.

The last section of the input file is delimited using the keyword 'timesteps'. This section of the input file contains data for one or more simulation intervals to be executed during the coupled computation. The data for each interval are the end time for the simulation interval, the maximum and minimum time step sizes for the simulation interval, the print, plot, restart write, and explicit coupling frequencies for that interval along with other control information for that interval.

3. Sequence of Events in a Coupled Computation

A coupled computation can be divided into two phases, that are the input and initialization phase of the computation and the transient simulation phase of the coupled computation.

3.1 Input and Initialization Phase

The executive program is executed by the user in a manner appropriate for the users operating system. The user specifies the input and the output files for the executive

program as command line parameters (default input and output files are also defined). The executive program reads the first section of its input file, constructs a PVM hostfile, and starts the PVM daemon process on the several computational nodes in the PVM virtual machine. Then the executive program spawns the several coupled processes on the one or more computational nodes. The coupled processes that are spawned read their respective input files, process the data contained in their input files and then listen to receive messages from the executive process. After the executive process has spawned all of the coupled processes, it sends messages to each of the spawned processes containing the data specifications for messages to send to and receive from the other coupled processes. Each spawned process proceeds with its own input and initialization after the coupling data specifications have been received from the executive process. The executive process listens to receive a message from each process describing its initialization status and its run status. Each coupled code sends its initialization status to the executive program at the end of its initialization process. This initialization status may be zero (initialization successful) or one (errors during input and initialization). They also send the executive program their run status. A zero (0) denotes no transient is to be executed because input or initialization errors were encountered or because this run was for input checking only. A one (1)if the file is ready for transient simulation. The coupled computation is terminated if any of the coupled processes return an initialization error or returns a zero run status. The executive program determines the global initialization and run status and broadcasts this status to all of the coupled processes.

3.2 Transient Computation Phase

Assuming that the initialization was successful for all of the coupled processes and that the run status indicated that all coupled processes are ready to perform a transient simulation, the executive program broadcasts an initial set of output control times. This message specifies the next simulation times for the production of printed output, printing of RELAP5-3D© minor edit variables, generation of plot data, writing of restart data and the next time explicit coupling data transfers are to be performed. The executive program assumes that each code will produce its own initial printed output, plot data, and restart data automatically. The executive program then coordinates the initial exchange of any explicit coupling data between asynchronously coupled processes. When more than two codes are coupled explicitly, the data exchange between the codes needs to be coordinated by the executive program. The data exchange paradigm used in all RELAP5-3D© coupled computation using PVM is that all messages received will be followed by an acknowledgement returned to the sender. The sending process waits to receive an acknowledgement before sending the next message. If all of the codes were to send all of their messages and then listen to receive all of their messages, there would be a deadlock condition because all processes would be sending and no processes would be listening for acknowledgements. The executive program broadcasts the PVM identifiers of each of the explicitly coupled processes to all of the explicitly coupled processes one at a time. The process named in the broadcast message sends its data and all of the other

explicitly coupled processes listen to receive the messages sent by the process named by the executive process. This works like the old telephone party line where each of the explicitly coupled processes must wait its turn to talk on the party line. Each explicitly coupled process receives permission to send its data in turn. This process of coordinating the exchange of explicit coupling data occurs each time the simulation time reaches the time for an explicit exchange of data.

Once any initial explicit coupling data is exchanged, time step advancements may begin. The executive program listens to receive a time step size from each of the synchronously coupled processes. Each code coupled synchronously to another code determines the time step size that it wants to use and sends it to the executive program. The executive program receives time step sizes, determines a global time step size as the minimum of the time step sizes received from the synchronously coupled processes and broadcasts the global time step size back to the synchronously coupled processes. This message also contains updated edit, print, and plot times so that output may be produced each time step rather than at predetermined intervals. Obtained output each time step is useful in debugging and this capability existed previously in RELAP5-3D©. After the synchronously coupled processes receive the global time step size from the executive program, they proceed with the time step advancement, performing any communication needed with the other coupled processes during the advancement. At the end of the advancement, just before the point of no return, each of the synchronously coupled processes sends its advancement status to the executive program. The point of no return is that point in the computations sequence after which no backup may be performed in order to fix any errors that occurred during the advancement. The executive program listens to receive the advancement status from all of the synchronously coupled processes, determines the global advancement status, and broadcasts the global advancement status to the synchronously coupled processes. Assuming that no errors have occurred during the advancement, the coupled codes and the executive program proceed to the next time step advancement. Time step advancements are performed until the end time for the simulation is reached. The executive program assumes that all of the coupled processes will terminate automatically when the end time is reached. The executive program waits for the coupled code to finish, and then shuts down the PVM virtual machine.

3.3 Error Handling

During a time advancement, any number of situations may arise that would require the solution algorithm to repeat a time step. When this occurs, an error condition is set and the time step is repeated. If any one of the synchronously coupled processes encounters an error during its advancement, it specifies the type of error in its advancement status flag. There are three categories of advancement errors; errors that cause the computation to terminate, errors that cause a backup and time step repeat with a smaller time step size, and errors that cause a backup and time step repeat with the same time step size. RELAP5-3D© has ten types of advancement

errors that the code detects and for which there are methods for fixing the errors. If the advancement status flag indicates a code termination, the executive sends a message to terminate all of the coupled processes, waits for confirmation that all of the coupled processes have terminated, and shuts down the PVM virtual machine as if the computation had finished successfully. If the advancement status indicates that any one of the coupled codes wants to perform a backup and a time step repeat, a backup message is sent to all of the coupled processes and the time step is repeated as if the time step had been successful. That is, each process sends its desired time step size to the executive and a new global time step size is chosen. If the process requesting the time step repeat needs to reduce the time step size, it sends a reduced time step size to the executive program and the global time step size will be reduced per that request. If the type of failure can be fixed using the same time step size, the global time step size will remain the same and the advancement will be repeated from the same starting point. The code requesting the time step repeat must remember the reason for the time step repeat and proceed through a different logic path during the repeated advancement to avoid the error that caused the time step repeat.

The previous discussion assumes that all of the coupled processes continue to execute and do not fail catastrophically. If any of the coupled processes fails catastrophically with a divide by zero, floating point overflow, etc., the process will be terminated by the operating system of the computational node on which the process is executing. The executive process monitors the execution status of all of the coupled processes and sends a terminate message to all executing processes if one of them fails catastrophically. It then waits for all of the processes to terminate and then shuts the PVM virtual machine down. The same process of termination occurs if a process exceeds its wait time while waiting for a message or message acknowledgement from another process. The user defines the length of time a process is to wait to receive a message or message acknowledgement from another coupled process. If the wait time is exceeded, the process exceeding the wait time sends a message to the executive program and shuts itself down gracefully. If the executive program receives a time-out message, it broadcasts a terminate message to all of the other coupled processes, waits for them to terminate, and then shuts the PVM virtual machine down as if the coupled computation had terminated normally. The wait time was implemented for the case in which a coupled code might get into a infinite loop or deadlock where it never sends an expected message but also never fails catastrophically.

In order for the user to understand the state of the coupled computation, status messages are written to the terminal from which the executive process was executed at ten second intervals. These messages are similar to the messages that RELAP5-3D© writes to the terminal screen when it executes as an uncoupled process. The messages contain the current simulation time and the current advancement count. Failure messages are also written to the terminal so that the user will understand why the coupled computation terminated. These status messages are also written to the output file of the executive program.

4. Verification of PVM Executive Program

The operation of the executive program was verified by executing the several test cases described in the previous papers under the control of the executive program. These test cases use two instances of RELAP5-3D© coupled to each other executing on the same computational node. Two test cases were executed, one using explicit coupling (Aumiller, 2001a) and the other using semi-implicit coupling (Weaver, 2001). These test cases were chosen to exercise the executive program logic for asynchronously and synchronously coupled computations. The results from the execution of the test cases as a single uncoupled process were compared to the results of the execution of the same test cases under the control of the executive program. Examination of the two versions of each test case showed that identical results were obtained on both the printed output (the printed output has five significant figures) and on the plottable output (the plottable output is written as 32 bit real numbers which equates to 7-8 significant figures). These test cases assume that no initialization or timestep size reductions occur and in fact the input for these test cases had previously been adjusted so that no timestep size reductions occurred.

Test cases were developed for each of the ten types of advancement faults in RELAP5-3D©. Two versions of each of the ten test cases were developed, that is one version of the test case as an uncoupled computation and the other as an coupled version of the test case executed under the control of the executive program. The results from the execution of the two versions of each test case were compared and identical results were obtained. These ten test cases verify that the executive program recognizes the several types of advancement faults, directs the coupled processes to perform a code backup, and coordinates the repeated attempted advancement.

Finally, several catastrophic failures were simulated by using the 'kill' operating system command to manually terminate one of the coupled processes during both the input and initialization phase of a coupled computation and during the transient phase of a coupled computation. The correct messages were written to the terminal and output file for the executive program, the other process in the coupled computation was shut down gracefully, and the PVM virtual machine was shut down as designed. The time-out mechanism was also tested by manually interrupting the execution of one of the coupled processes and observing that the correct time-out messages were sent, that the processes terminated as directed, and the PVM virtual machine was shut down.

5. Summary

An executive program has been developed to control and coordinate a computation using several computational codes coupled together using the PVM message passing

software. The design of this executive program has been described along with the sequence of events that occur during a coupled computation. The verification testing has demonstrated that the executive program performs as designed and that it is capable of initiating a coupled computation, controlling the coupled computation including recognizing and correcting faults in the coupled computation and termination of the computation when it is finished. The operation of the executive program was demonstrated using the RELAP5-3D© computer program but the executive program is general enough to be used to couple any number of simulation codes.

Acknowledgement

Work supported by the U. S. Department of Energy, under DOE Idaho Field Office Contract No. DE-AC07-99ID13727.

References

Aumiller, D. L., Tomlinson, E. T., Bauer, R. C., 2001a, "A Coupled RELAP5-3D/CFD Methodology with Proof-of-Principle Calculation," Nuclear Engineering and Design, Vol. 205, pp 83-90.

Aumiller, D. L., Tomlinson, E. T., Weaver, W. L., 2001b, "An Integrated RELAP5-3D and Multiphase CFD Code System Using a Semi-Implicit Coupling Technique," available as B-T-3367 from DOE Office of Scientific and Technical Information.

Geist, A. et. al., 1993. "PVM (Parallel Virtual Machine) User's Guide and Reference Manual," Oak Ridge National Laboratory, ORNL/TM-12187.

Martin, R. P.,1995. "RELAP5/MOD3 Code Coupling Model," Nuclear Safety, Vol. 36, No. 2, pp. 290-299.

RELAP5-3D, 1999. "RELAP5-3D Code Manuals, Volumes I, II, IV, and V," Idaho National Engineering and Environmental Laboratory, INEEL-EXT-98-00834, Revision 1.1b. (See also the RELAP5-3D home page at http://remus.inel.gov/relap5)

Weaver, W. L., Tomlinson, E. T., Aumiller, D. L., 2001, "A Generic Semi-Implicit Coupling Methodology for use in RELAP5-3D," available as B-T-3321 from DOE Office of Scientific and Technical Information.

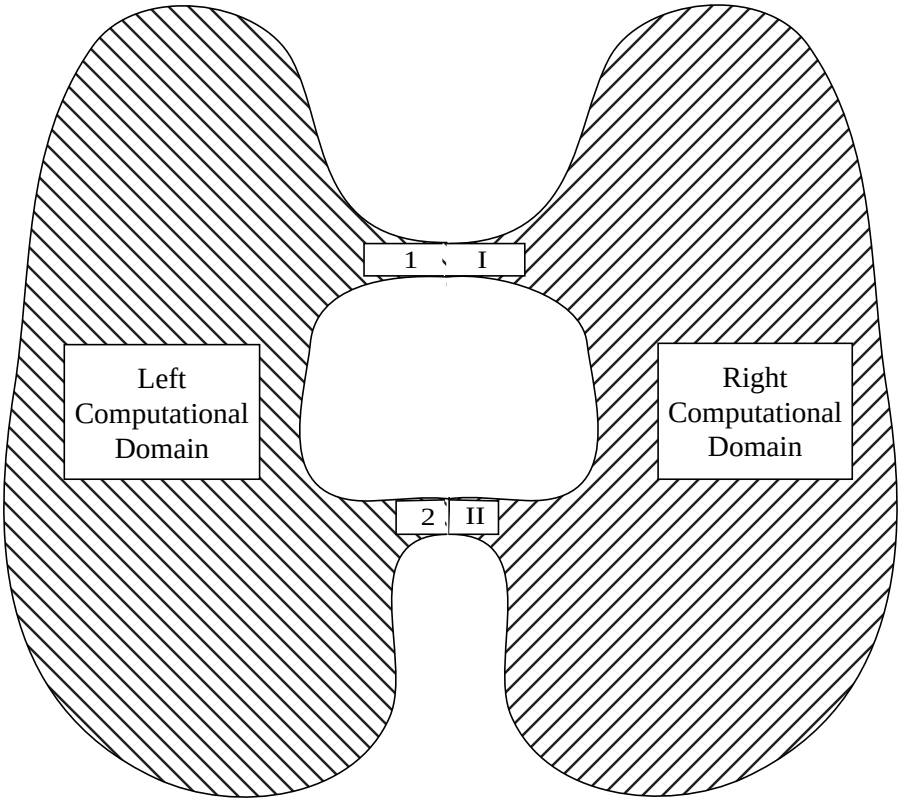


Figure 1 Schematic of Coupled Solution Domain

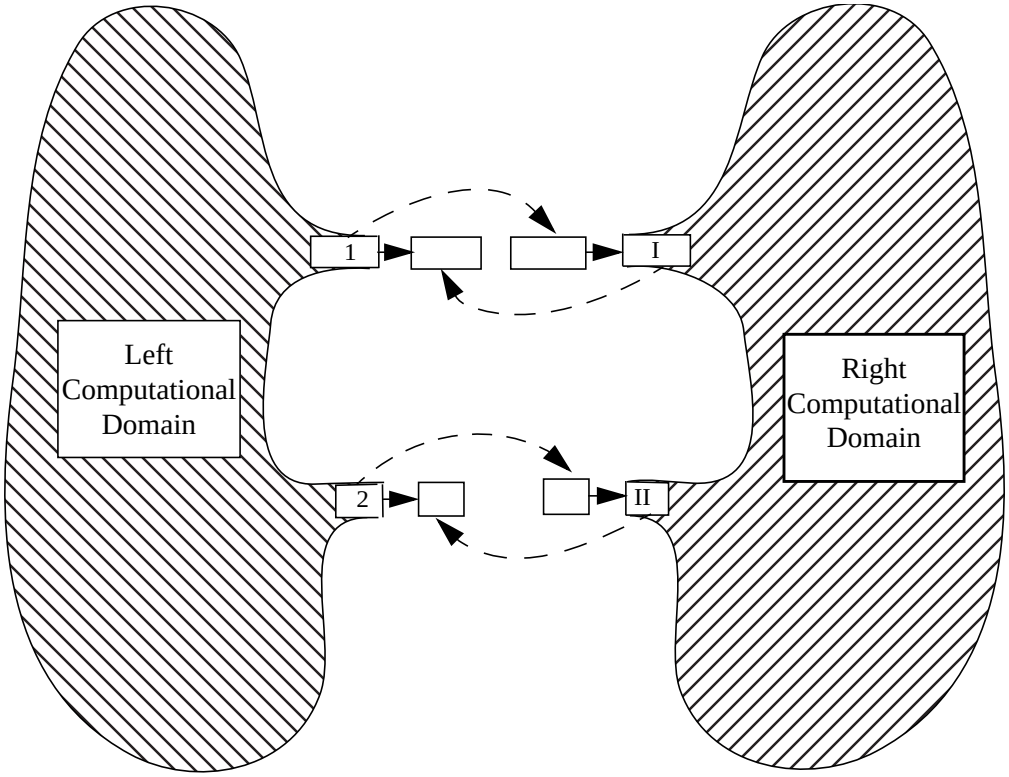


Figure 2 Schematic of Explicit Coupling Methodology

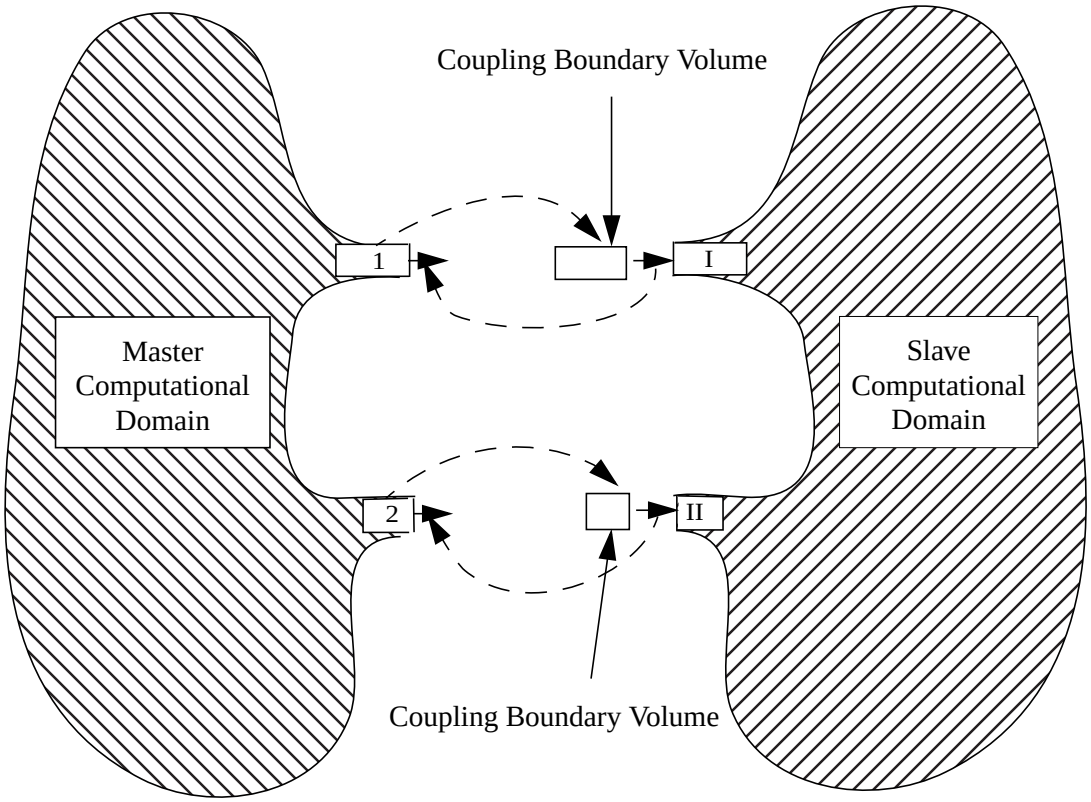


Figure 3 Schematic of Semi-Implicit Coupling Methodology

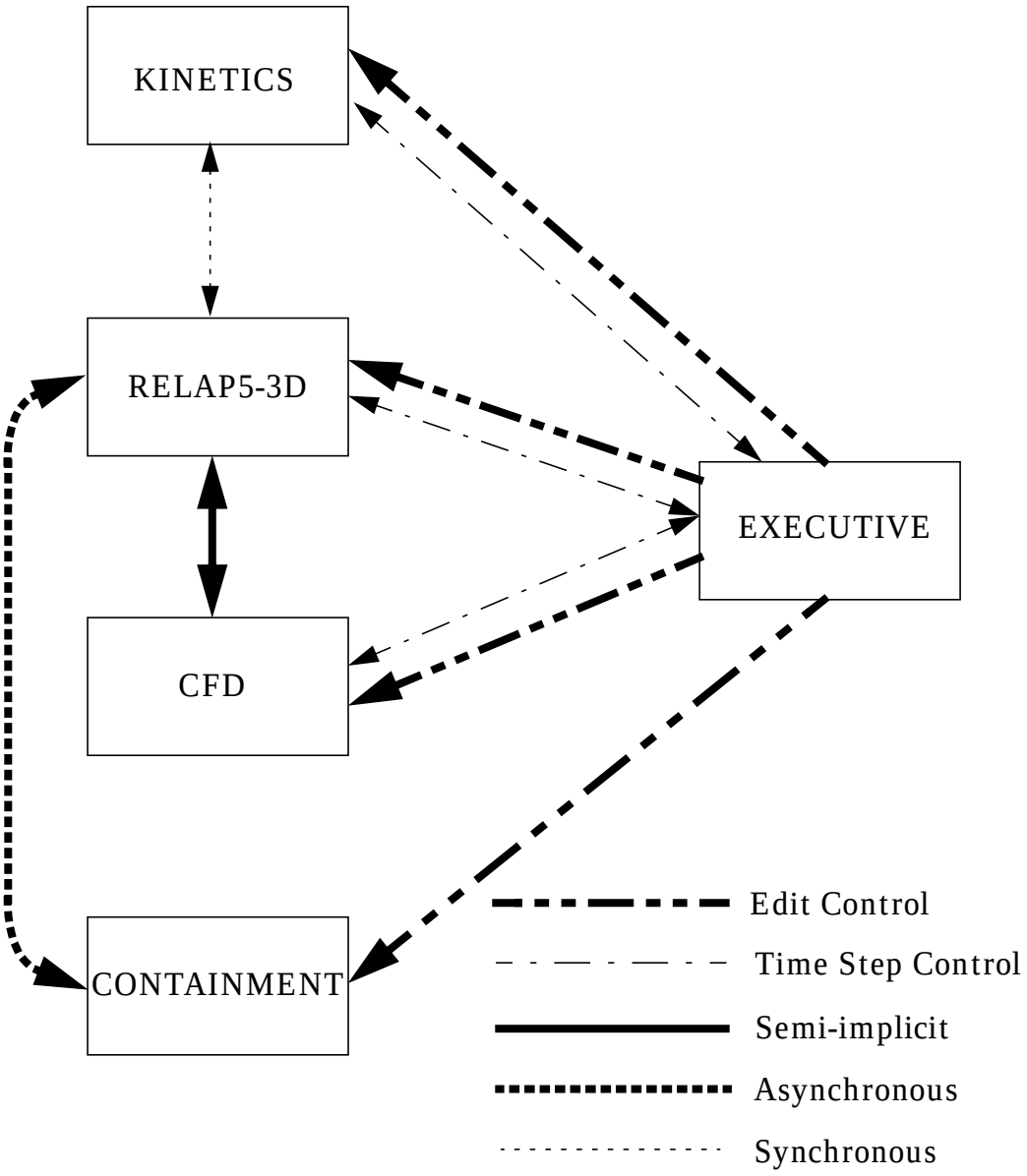


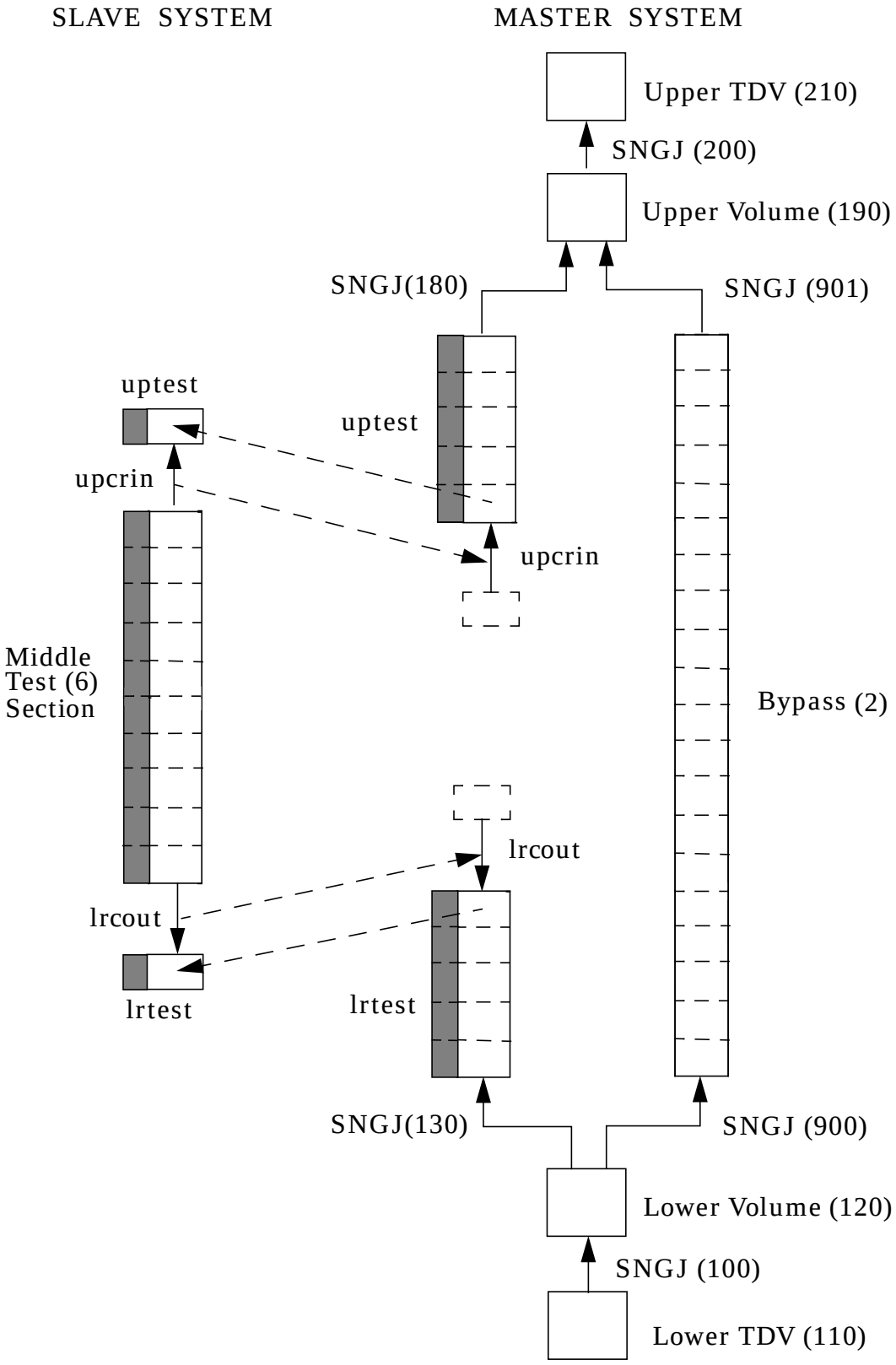
Figure 4 Schematic of a Prototypic Coupled Computation

Appendix A

Sample Input Deck for Executive Program

The input file for the semi-implicit verification test case is shown below. This test was used to verify the semi-implicit coupling methodology (See Weaver, 2001). This test case uses two instances of RELAP5-3D© named 'test' and 'loop' that are coupled semi-implicitly (semi-implicit coupling requires that the processes be coupled synchronously). The PVM virtual machine contains a single computational node (i.e., a UNIX workstation named 'ws-rjw') and both instances of RELAP5-3D© are executed on this computational node. The upper and lower coupling boundary volumes are named 'lrtest' and 'uptest' in the input decks for the two instances of RELAP5-3D© and the upper and lower coupling junctions are named 'lrcout' and 'upcrin' in the same input decks (i.e., input deck pvmtestp.i for the master process and input deck 'pvmtestc.i' for the slave process in the semi-implicit coupling). These component names are used in the specification of the data that is to be exchanged between the two processes. The simulation is executed for a total of ten simulated seconds.

```
# pvm semi-implicit test case pvmtest
virtual
        ws-rjw wd=/rjwu1/wlw/151d/run ep=/rjwu1/wlw/151d/run
processes
        ws-rjw
                loop synchronous relap5.x -i pvmtestp.i -o pvmtestp.p
                test synchronous relap5.x -i pvmtestc.i -o pvmtestc.p
messages
        semi-implicit
                loop sends test lrtest 5 uptest1
                loop receives test lcrout 0 upcrin 0
                test sends loop lrcout 0 upcrin 0
                test receives loop lrtest 1 uptest 1
timesteps
 0.5000   0.000001  0.00625  403   1  20  500  0
 1.0000   0.000001  0.00625  403  20  20  500  0
 10.000   0.000001  0.0125   403  10  50  500  0
```


SLAVE SYSTEM
MASTER SYSTEM
Upper TDV (210)
SNGJ (200)
Upper Volume (190)
SNGJ(180)
SNGJ (901)
uptest
uptest
upcrin
upcrin
Middle
Test (6)
Section
Bypass (2)
lrcout
lrcout
lrtest
lrtest
SNGJ(130)
SNGJ (900)
Lower Volume (120)
SNGJ (100)
Lower TDV (110)