Architectural Issues and Developments in RELAP5-3D

Dr. George Mesina

RELAP5 International Users Seminar

Idaho Falls, ID

September 12-13, 2013

Outline

- Recent Issues and Solutions
- Architectural Development
- Announcements on Compilers and O/S
- New Documents

Issues going from 4.0.3 to Version 4.1.3

- Two major issues were encountered and solved
  - Order of evaluation in if-tests
    - UP 13016
  - Issues associated with allocating and deallocating memory
    - Many UP related to this

Order of Evaluation

- The order of evaluation is left to right in the C language and numerous other programming languages.
- ANSI FORTRAN does not enforce this in any standard.
  - Historically, it has been left to right on most computer platforms.
  - With multi-core processors, it is seldom the case anymore.
- This affects many kinds of statements. Examples (.OP. means logical operator:
  1. IF (G(i) expression) .OP. (F(i) expression)) THEN
  2. IF ((protection clause) .AND. (protected clause)) THEN
- In #1, the F(i) or (Gi) are functions that change “i”, then whichever goes first can affect what the second one takes as input.

Order of Evaluation cont…

- #2. IF ((Protector Clause) .AND. (Protected Clause)) THEN
- In left-to-right evaluation, the evaluation stops whenever the first clause (the protector) is false. The second is never evaluated.
- Examples of this concept:

Protector Clause	Protected Clause
X >= 0	sqrt(x)
X /= 0	1/x
i > 0	array(i)
ALLOCATED(v)	DEALLOCATE(v)
.NOT.ASSOCIATED(v)	ALLOCATE(v(NVAR))
PRESENT(callArgument)	callArgument = 0

- The impact of evaluating the right-hand (protected) clause before the left-hand clause varies from negligible to core-dump

Order of Evaluation

- The solution is to break the if-test

```
If ((protection clause) .AND. (protected clause) ) then
```

- *Becomes*

```
If (protection clause) then
    If (protected clause) then
```

- This forces the evaluation to occur in the proper order.

Order of Evaluation

- More than 293,000 lines of code
- More than 34,000 if-statements
- More than 1200 if-statements fit the patters:
  - 2 or more clauses
  - 1 or more AND-operator(s) and
  - Either an array-reference or a function call
- 3 developers searched the1200 statements
  - In RELAP and ENVRL directories
  - Did not examine fluids directories
- More than 60 if-statements required splitting

Issues with Allocating and Deallocating Memory

- Errors with allocating and deallocating memory can cause
  - Out of bounds array access
  - Memory leaks
  - Hanging of the machine (in a non-parallel process!)
    - This has only occurred in restarts with multiple input cases.
- Out of bounds array access either fetches wrong values or overwrites values in other memory locations
  - The latter can destroy data or (machine) coding
  - It seldom evidences itself immediately
  - Therefore, it can be difficult to track down

Issues with Allocating and Deallocating Memory

- Memory leaks cause problems when memory is repeatedly created and destroyed incorrectly
  - It can occur if a pointer is eliminated without first deallocating it
  - E.G. a sub-derived type array gets destroyed by deallocating the derived type that contains it w/o destroying it first
  - The memory is “lost” to your process.
- RELAP5-3D input decks with multiple cases can cause a build-up of memory leaks
- It is an error to allocate an array that is already allocated and to deallocated one that is not allocated.

Issues with Allocating and Deallocating Memory

- It is an error to access an array that is not yet allocated.
    - IF (.NOT.ALLOCATED(a)) ALLOCATE(a(na))
    - ALLOCATE(a(1)%b(nb))
  - With multi-core computers this can produce errors if 1st core has not completed memory set up for “a” when 2nd core attempts to allocate “b”
  - A safer method:
    - IF (.NOT.ALLOCATED(a)) THEN
      - ALLOCATE(a(na), STAT=istat)
      - IF (istat == 0) ALLOCATE(a(1)%b(nb))
    - ENDIF
  - NOTE: do not need to check allocation of “b” because if “a” is not allocated, the a(1)%b is not allocated either.

Issues with Allocating and Deallocating Memory

- Initially nearly a dozen restart input decks with a significant number of input cases hung the machine
  - Linux with ifort 10.1
- INL protected nearly every allocate and deallocate statement with if-allocated-tests
- Number of failures in secondary input cases have been reduced to 3 input models.
  - Linux with ifort 11.1
- Still working to solve these final issues.

Development: Isolation

- The purpose of isolation of *data and coding* is to prevent inadvertent memory access errors
  - Reduce chance of introducing bugs into code.
- Ideally, modules are intended to supply data and coding that acts only on that data
- Modules should use the "private" attribute on memory and subprograms not intended for use outside the module.
- Ideally, modules should USE only level 0 modules
  - Level 0 modules have universal scalars.
    - E.G. intrmod, consmod, ctrlmod
  - Prevents circular references: A uses B uses C uses A
    - Simplifies installation process

Development: Isolation

- Plan to gradually remove some module references from some modules
  - Simplify by removing one module reference at a time
- For modules that need few (say up to 3 variables) from another module
  - No need to USE the other module
  - The variables can be passed into the subprogram that uses them through call parameters.
- For modules that have a subprogram that needs many variables from other modules (and many from the module containing it)
  - Consideration will be given to promoting that subroutine out of the module to independent status.
- In non-module subprograms, employ:
  - use module, only
- Existing subprograms and modules are exempted – for now

Development: Isolation

- New module verifymod.F90 models this development.
- It references two level 0 modules:
  - use intrtype
  - use ufilsmod, only: verifl
- None of its six subprograms have any use statements.
  - Two require data from outside which are accessed through the individual call sequences
- Two subroutines were spun out
  - Verfsum required data from a dozen other modules – too many
  - Verfbackup required half a dozen such references

Announcements

- In keeping up with advancements in the computing industry, decisions have been made and implemented.
  - Compilers and levels
  - Computer platforms
  - Installation procedures
- Due to limited resources, INL limits its official support of compilers, operating systems and installation procedures.
  - This limits what the RELAP5 team can support

Announcements

- Official Compiler: Intel Fortran level 11.1
  - Both Windows 7 and Linux
- Unsupported compilers
  - RELAP5-3D does install with ifort 10.1 and ifort 12.1
    - Performance is not as reliable with these two as with 11.1
  - The code will install with other compilers, but INL does not support them

Announcements

- INL IT supports Windows 7 and SUSE Linux platforms
  - Windows XP is no longer supported
  - Windows 8 is not (yet) supported
  - No other Linux is not supported (in particular: Cygwin and Redhat)
- INL RELAP5-3D Team supports installation on
  - Windows 7 with Visual Studio 2008
    - Have purchased and installed VS 2012, but not yet working with it
  - Linux via Linux C-shell scripts and Makefiles
- It is possible to install RELAP5-3D on Macintosh systems, but INL department does not support this.

New Documents for RELAP5-3D and Auxiliaries

- PROGRAMMING
  - G. L. Mesina, “Guidelines for developing RELAP5-3D coding, INL/EXT-13-29228, Rev 1, June 2013.
- INSTALLING
  - J. H. Forsmann, G. L. Mesina, “RELAP5-3D Windows 7 Build,” INL/MIS-12-27541 Rev. 1, October 2012.
  - J. H. Forsmann, “RGUI Configuration Guide ,” GDE 648, INL/MIS-13-30082, Sep 2013.
- RUNNING
  - J. H. Forsmann, J. E. Fisher, G. L. Mesina, “PYGMALION User’s Manual,” GDE-621, INL/MIS-13-28216, INL/MIS-13-30083, March 2013.
  - J. H. Forsmann, “RGUI Help Manual: RELAP5-3D Graphical User Interface,” GDE 649, INL/MIS-13-30083, Sep 2013.

SUMMARY

- Computer advancements affect RELAP5-3D performance, accordingly changes are being made.
- Reported issues relating to multi-processors have been addressed
  - Order of evaluation in if-tests
  - Issues associated with allocating and deallocating memory
- New RELAP5-3D development will employ isolation of data and code
- RELAP5-3D support announcements:
  - SUSE Linux and Windows 7 only
  - MS Visual Studio 2008
  - Intel Fortran/C 11.1
- Many new documents have been prepared and are available