Order of Evaluation Issues

George Mesina

Apr. 2013

New Fortran compilers do not force left-to-right evaluation of if-clauses. This creates evaluation issues for RELAP5-3D.

Modern multi-core computer chips can run faster than single core processors whenever their cores can be active at the same time. This is wonderful for running a calculation program, such as RELAP5-3D on one core while at the same time using other cores to check for email, clean up files, and check for software updates. These cores sometimes have multiples of the same functional units, so that a line of code with two additions can be accomplished more quickly. Even short vectors, such as the Cray vector supercomputers, have been re-introduced into the newer workstation chips.

Compiler writers have taken advantage of these new hardware features to make programs run faster. Where they've taken advantage of multiple comparison operators, it creates an issue with statement evaluation for older FORTRAN programs including RELAP5.

According to the FORTRAN 90 Handbook, page 229, a conjunction expression uses the .AND. operator and is left recursive. It states that the expression A *.AND.* B *.AND.* C must be evaluated *(*A *.AND.* B*) .AND.* C. However, it does not require A, if it is a logical comparison, to be evaluated before B.

Of course, if any clause, A, B or C, is false, the entire conjunctive expression is false. For decades many FORTRAN compilers evaluated the first clause and if it was false, determined the entire expression to be false without further evaluation. On single core processors, this reduced run time. Most FORTRAN compilers also evaluated clauses left to right. Thus, if the leftmost clause was false, the entire expression was.

RELAP5 and many other programs took advantage of this. Many protective if-statements have the form:

```
if (i .lt. UpLimit .AND. A(i) .ge. Value) then
```

When evaluated from left-to-right, the first clause prevents the array being accessed outside its bounds. However according to the ANSI Standard, either A, which is *i .gt. LowLimit*, or B which is *A(i) .ge. Value*, can be evaluated first or both can be evaluated simultaneously on modern platforms. In fact, on multi-core computers, generally all clauses are evaluated. So, when the code is compiled with array-bounds checking, it will quit on this statement whenever *A(i)* is an out of bounds reference.

To correct the problem with bounds-checks stopping the code, the if-tests are broken up. The above if-statement becomes two nested if-tests.

```
if (i < UpLimit) then
        if (A(i) >= Value) then
```

The problem with finding and fixing these is that there are so many if-statements and many of them take two or more continued lines of code. All clauses of long if-statements must be examined because with simultaneous evaluation, the if-statement's value is not determined until all clauses are evaluated.

There are over 27000 if-statements in the RELAP and ENVIRONMENTAL directories. Since most if-tests in the code do not have this form, a Linux script was written that combined continuation if-statements into a single line, and kept only those that had both an array reference and an AND-operator. This reduced the number to just over 1500 which were hand-checked. Nearly seventy if-statements fit the pattern and were broken up.

The same problem can arise with the OR-operator. An out-of-bounds array reference could access a value that makes a comparison expression true, so that an otherwise false OR-expression becomes true. We have not yet addressed this issue; however no users have reported difficulty with this yet.

The FORTRAN ambiguity on left-to-right precedence with just two operators can lead to other issues. For example consider two logical functions, $F(i)$ and $G(i)$, that are allowed to change their argument, i. The evaluation of the if-statement:

If ($F(i)$ .or. $G(i)$) then

can change depending on whether F or G is evaluated first and how the other acts with a modified value of i. Again, no issues of this sort have been reported with RELAP5-3D.

The solutions reported above for and-operators will be included in the next code product, which is scheduled for release in the summer of 2013.