Programming Improvements in RELAP5-3D

George L. Mesina

Idaho National Engineering & Environmental Laboratory
Idaho Falls, Idaho 83415

2003 RELAP5 International User's Seminar
August 27-29, 2003
West Yellowstone, MT

Outline

- ***Fortran 90***
- ***OpenMP Parallel***
- ***Bitpacking Conversion***

Reasons for Fortran 90 Conversion

- ***Code modernization (keep up with the compilers)***
  - Vendors supply only FORTRAN 90 and 95
  - FORTRAN 2000 standard is nearing completion
- ***Reduce maintenance & development cost***
  - Readability
  - Easier to modify, fewer errors
  - Maintainability (bug fixes, shelf-life of code, etc.)
- ***Better language features***
  - allocate, modules, derived types, etc.

Fortran 90 Conversion Overview

- ***Long-term project with multiple tasks***
  - Current tasks: OpenMP parallel, bitpacking
  - Upcoming Tasks: Elimination of FA-array, I/O Changes, Internal methods changes (SCNREQ, RENODE, etc.)
- ***More than 100,000 lines of codes will change***
  - Some will change multiple times during several tasks
  - Much of the work must be automated
- ***Reliability: The goal is to introduce no errors as a result of the conversion.***

Reasons for OpenMP Parallel Task

- ***RELAP5-3D uses direct calls to the KAI parallel subroutine library to implement parallelism.***
- ***This is unviable because KAI was bought out; its software support will cease.***
- ***Industry standard for parallelism is OpenMP.***
  - With OpenMP, the code will parallelize with modern Fortran compilers on most O/S.
- ***Reduced cost for maintenance and development over KAI library calls.***
  - OpenMP is <u>easier to read</u>.
  - OpenMP coding <u>simpler</u>.
  - There will be <u>fewer</u> code errors.

Description of "Parallel" Task

- ***RELAP5-3D uses calls to the KAI parallel subroutine library to implement parallel.***
- ***Convert calls of KAI subroutines to OpenMP directives.***
- ***Convert style of parallelism from "one fork" to "natural parallelism."***
- ***Parallelize 3D hydro subprograms with openMP.***
- ***Test carefully.***

Starting Status

- ***RELAP5-3D partially converted to OpenMP.***
- ***Subsequent code development impaired parallelism.***
  - Some OpenMP loops became non-parallel by introduction of non-parallel code.
  - Some OpenMP directives became incorrect.
- ***Parallel errors occurred in some problems.***
  - Deadlocks
  - Random errors
- ***Calculations differed when number of processors was increased for some problems.***

Parallel Task Plan

1. ***Stabilize RELAP5-3D for standard test problems.***
   - Eliminate code aborts and freezes.
   - Fix random errors.
   - Get serial calculations to agree exactly w/ those from one, two, and four processors.
2. ***Complete parallelization of code.***
3. ***Improve parallel speed-up.***

Parallel Task Plan

- ***Write a program to place OpenMP directives before every loop.***
- ***Hand process each subroutine to eliminate directives for non-parallelizable loops.***
- ***Use of program:***
  - ***Replace incorrect directives.***
  - ***Add OpenMP to subroutines not previously parallelized.***
- ***Carefully Test RELAP5-3D performance.***

Parallel Task Status

- ***Converter program written.***
- ***Over 80 subroutines converted or reprocessed.***
  - Only neutron kinetics subroutines remain unfinished.
- ***Output compared for 100 standard test cases.***
  - Same to last decimal place printed for:
    - *serial, 2 threads, and 4 threads.*
- ***No parallel errors remain.***

Bitpacking Background

- ***Introduced to save memory.***
  - A logical value or flag with limited settings does not need an entire 4- or 8-byte word.
  - Compress many of these into selected bits within one word.
- ***Bits are set and retrieved via bit-oriented intrinsic functions.***
  - Originally the intrinsic functions were machine dependent.
  - Fortran 90 provides an expanded library of bit-functions.
- ***Reading and coding bit operations is difficult.***
  - A constant source of errors.

Bitpacking Background

- ***Layout of bits in an 8-byte word***

63	62	61										2	1	0

  – Bit positions are numbered from the right starting at 0.

  – The value of each bit is either 0 or 1.

- ***The value of the integer represented by the bits***

  – Let b_i be the value of the bit in position i.

$$m = \sum_{i=0}^{n-1} b_i 2^i .$$

Bitpacking Operations

- ***All bitpacking operations previously done with compositions of these 6 operators.***
  - IAND, IOR, XOR, NOT, ISHFT, ISHFTC
  - First three refer to the numerical expansion, m, rather than the bits, i.
- ***The numerical expansion, m, of the bits is often a large number.***
  - To understand the operation, must determine the bits, i, it represents.
- ***Most bitpacking operations require combinations of these functions and numbers.***
  - This causes difficulty in reading and developing the code.

Fortran 90 Bitpacking Task

- ***Purpose of Bitpacking Conversion is to replace complex constructs with simpler ones.***
  - Use new bit intrinsic functions in FORTRAN 90.
  - Create new bit functions in a module.
- ***New Fortran 90 functions refer to bit locations, i, rather than the numerical equivalent, m.***
  - IBSET(A,B) sets bit B in variable A to 1.
  - IBCLR(A,B) clears bit B in variable A to 0.
  - BTEST(A, B) returns true is bit B in A is 1, false otherwise.
  - IBITS(A, B, C) extracts a byte of length C from A starting in position B. That is bits B through B+C-1.

Compare Old & New Bitpacking

Previous	*Fortran 90 & Module Fctns*
VAR = IOR(VAR, 16384)	VAR = IBSET(VAR, 14)
IAND(VAR, not(262144)) .NE. 0	BTEST(VAR, 18)
IAND(ISHFT(VAR,-3), 127)	IBITS(VAR, 3, 7)
ISHFT(IAND(JC(JX),1572864),-19)))	IBITS(JC(JX), 19, 2)
IOR(IAND(IMAP(i),NOT(ISHFT(63, 18))), ISHFT(FLOMAP(IX),18))	IBYTECOPY(FLOMAP(IX),6,0, IMAP(i),18)

- ***Fortran 90 functions are simpler and easier to understand.***
- ***IBYTECOPY is a new module bit function.***
  - There are 4 others.

Fortran 90 Bitpacking Task

- ***Method of conversion***
  - Identify and categorize bitpacking constructs.
  - Write program to automate conversion of most constructs.
  - Hand convert only those constructs with few instances or high complexity.
  - Carefully test each significant conversion (over 50).
- ***Testing***
  - Over 100 standard test cases run with & without conversion.
  - Output compared character by character.
  - Accept conversion only if NO “non-time” differences found.

Bitpacking Status

- ***New Fortran 90 module of bitpacking functions written and in use.***
- ***All programmable bitpacking finished.***
  - Over 3800 statements converted.
- ***No differences due to conversion in output.***
  - Checked to last decimal place printed.
- **Task complete, except for final report.**