FORTRAN and MPI

Message Passing Interface (MPI)

Day 4

Course plan:

- MPI - General concepts
- Communications in MPI
  – Point-to-point communications
  – Collective communications
- Advanced MPI: user-defined data types, functions
  – Linear Algebra operations
- Advanced MPI: communicators, virtual topologies
  – Parallel sort algorithms
- Parallel debugging
- Parallel performance. Summary. Tendencies

TotalView Multiprocess Debugger

TotalView is a full-featured, source-level, graphical debugger for application programs. It is a multiprocess, multithread debugger that supports multiple parallel programming paradigms including MPI, PVM and OpenMP.

Languages	Programming Paradigms
C	Multiprocess
C++	Multithread
FORTRAN 77	MPI
Fortran 90	OpenMP
Assembler	PVM
	Fork/exec
	shmem

Debugging MPI programs with TotalView
How to start and illustration of some basic features

1. Compute your code with the flag '-g'

```
isaac(hpcmn) $ mpf90 -g -o search search.f -lmpi
```

2. Start TotalView in foreground by typing the command 'totalview'

3. In the 'New Program' window

   (a) Click 'Browse' and choose the executable which you want to debug
   (b) Click on 'Parallel' in the header menu
      i. Click on 'Parallel system' and choose '

ii. Click on 'Tasks' and choose the number of processes you want to run your program on
iii. Confirm with 'OK'

Totalview shows a new window where you see source code of your main routine.

4. Browse the code and put some breakpoints by left-clicking on the line number

5. Try 'diving' into a subroutine by right-clicking on the subroutine's name

   When going down one can put more breakpoints on appropriate command lines

6. Start the execution by clicking on 'GO' in the header menu of the same window. Answer 'NO' to the question if you want to stop the process.

The execution will continue till the first break point is reached.

By clicking on 'GO' again, the execution is resumed.

7. After the program has stopped at a certain breakpoint, we are able to see the values of the variables on the different processes (again by diving).

   Totalview opens a separate window per variable. Note that if the execution has been continued to the next point the variable value could have changed its value and one has to update it by clicking on 'Update'.

8. Test the path 'Tools' $\longrightarrow$ 'Call Graph', which will plot a graph of the calls in your program (according to how the pattern of the execution of this particular run).

9. Test the path 'Tools' $\longrightarrow$ 'Message Queue Graph', which will plot a graph of the not yet completed communications.

10. Observe the possibilities to:

    (a) step through your code and execute command per command. This can be done for the individual processes separately. Check the 'Process' menu.
    (b) action points, possibility to put a barrier etc
    (c) Test the path 'Tools' $\longrightarrow$ 'Memory Debugging', which provides a list of services, such as memory leaks.