Greg Watson

PTP User/Developer Meeting, Chicago, September 2012

IBM

Overcoming Issues Using PTP With Very Large Projects

IBM

Application Code

- Real Application Parallel Systems Version 11 (RAPS11)
- ECMWF's Integrated Forecasting System (IFS) Benchmark
  - Based on an up to date source cycle (CY36R2/Dec 2009)
  - Programming model uses MPI + OpenMP
  - Languages include F77, F90, F95, C
  - ~74MB executable
- Source files
  - 5900 Fortran files
  - 391 C files
  - 3800 header files
  - ~311K lines of code
- Model resolutions
  - T159L1
    - 49 MB dataset, vertical = 91 levels, horizontal = 125 km, time-step = 3600 seconds
  - T2047L149
    - 15 GB dataset, vertical = 149 levels, horizontal = 10 km , time-step = 450 seconds

Building The Code

- Uses makefiles, but not directly
- Build system in src/scripts directory
- Build parameters are set in various places
  - ../build/arch
  - include/config, include/project_config
- Invoke a script called ‘bld_job’ with various arguments
  - bld_xxx.job builds individual component ‘xxx’
  - bld_all.job builds the entire application by running each build step
  - bld_clean.job tries to clean the source directory (doesn’t work well)
  - mkabs_fc.job performs the final link, result in ${IFS_DIR}/bin
- No output generated to the console
  - Output from each build step logged to a file
  - If anything goes wrong, need to inspect each log file to locate error
- Due to large amount of output, recommended to run bld_all.job a number of times so that only actual errors are logged

Running The Code

- Data and scripts are provided separately to source for various problem sizes
- Can be located anywhere, but needs to know where executable is located
  - Manually update script to specify location
- Three scripts provided
  - ifs_run_tnnn_ref for a reference run
  - ifs_run_tnnn_long for a full run
  - ifs_run_tnnn_ttv for debugging with TV
- Scripts are can be run interactively, or submitted via LoadLeveler

Importing Into Eclipse

Recommendation

- Make sure heap size is set to large value (e.g. –Xmx2G)
- *Needs to be fixed for Parallel Package*

- Using synchronized Fortran project
- Top level directory contains ‘bin’, ‘data’, and ‘src’
  - Only synchronize with ‘src’
- Set initial regex filters to exclude build artifacts:
  - .*\.a, .*\.o, .*\.list, .*\.lst
- First synchronize got stuck at 12%
  - No progress after ½ hour
  - Appeared to be caused by running out of heap
  - Increased heap from 384M to 2048M
- Second synchronize took between 5-10 mins depending on network
  - Over cable modem, so not particularly fast network
- Subsequent synchronizations take about 30s
  - Not great, particularly when building, but usable

C/C++ Indexing

- C/C++ Indexer starts immediately synchronization finishes
- Sometimes the indexer appears to hang forever
- Check that it is not trying to index binary files
  - RAPS11 had executables called ‘iterator’, ‘set’, and ‘list’
  - Had been synched to local machine
- The default CDT settings recognize certain file names as headers
  - These show up in project view as C++ header files
  - There does not seem to be any way to turn this off
- Removing these files enabled the C/C++ indexer to be restarted
- Indexing RAPS11 took less than a minute

Recommendation

- Make sure binary files are not being recognized as C++ headers
- *Bug 389521 has been opened*

Fortran Indexing

Recommendation

- Only enable Fortran indexer if navigation/content assist required

- Indexing is not enabled by default
- Must be enabled through Fortran General>Analysis/Refactoring
- Indexing takes around 7-10 mins
- However, many files are dummies (3000 of 5800)
  - Many of these are over 11K lines of code
  - So the actual IFS code would take much longer to index
  - At least 2-3 time longer
- No real issues observer with indexer
  - Max heap usage was 1011M
- Most files are either .F (fixed form) or .F90 (free form – C preprocessed)
  - Refactoring not supported on either format

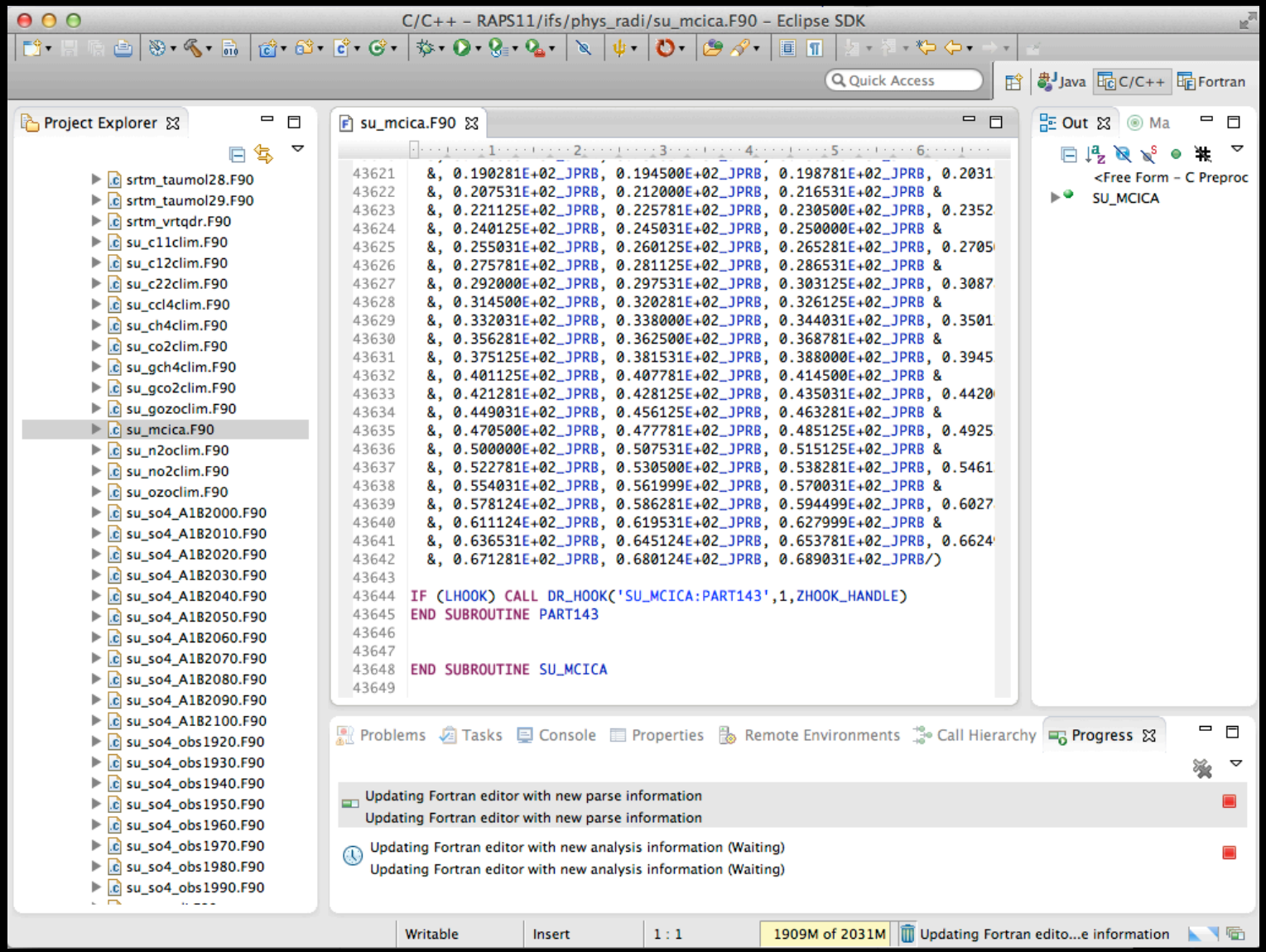
Fortran Editor

Recommendation

- Do not edit files over about 5K lines of code
- *Bugs 389565 and 386775 have been opened*

- Largest source file is 43K+ lines of code (2.8MB)
  - Takes about 60s to open
  - Then about 90s to update the editor with analysis/parse information
  - UI is locked during this time
  - Every change requires about 90s to update
  - Not really usable
  - You will need over 1G of heap to do this
- Next largest source file is about 15K lines of code (1.5MB)
  - About 15s to open
  - About 25s to update editor
  - Not really usable
- Source files around 5K lines of code
  - Take a couple of seconds to open and update
  - Probably usable
- Parse results do not seem to be released when source files are closed
  - Very easy to consume all heap, resulting in bad consequences
- Fortran 77 (.F) files display syntax error in outline view
  - Need to change source form to “Fixed Form – INCLUDE lines ignored“
  - This is a bug

Fortran Editor



C/C++ Editor

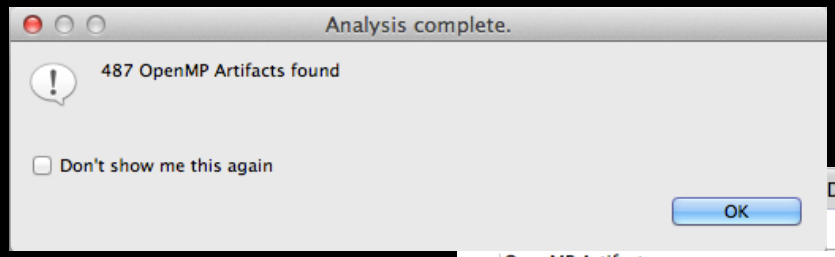
- Largest source file is 4K+ lines of code
  - Takes about a second to open
  - Update delay not noticeable
  - Codan works in background
  - No issues observed
- Just for interest, created 45K line C file
  - Still only took about a second to open
  - Update delay not noticeable
  - Automatically enables scalability mode which disables live parsing
  - Disabling scalability mode does not introduce noticeable delays

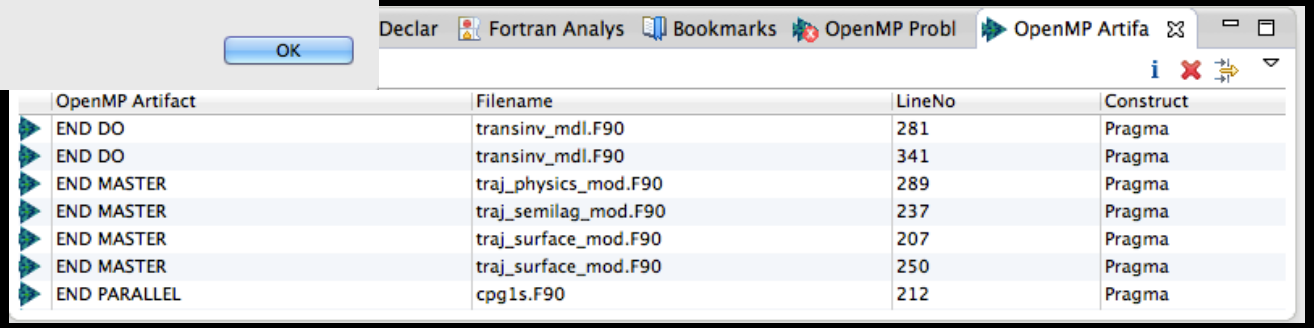
Static Analysis

Recommendation

- Usable, but would become painful for really large projects

- Running “Show OpenMP artifacts on the whole project
  - Takes about 1 minute
- Slow, but not a show stopper
  - On IFS code it would take much longer
- UI is blocked by dialog while analysis is taking place
  - Needs to be modified to work in the background
- Finds 487 artifacts, so could be potentially very useful for trying to understand code





OpenMP Artifact	Filename	LineNo	Construct
END DO	transinv_mdl.F90	281	Pragma
END DO	transinv_mdl.F90	341	Pragma
END MASTER	traj_physics_mod.F90	289	Pragma
END MASTER	traj_semilag_mod.F90	237	Pragma
END MASTER	traj_surface_mod.F90	207	Pragma
END MASTER	traj_surface_mod.F90	250	Pragma
END PARALLEL	cpg1s.F90	212	Pragma

Building

- Build scripts do not generate output to console
  - Error parser will not work
- Needed to modify build script
  - Add '| tee' to send output to console as well as log files
- Errors were being recognized but not matched to a source file
  - Could see error listed in Problems view
  - No error marker in source file
- Compile was using '-qsource' flag to generate listing file
  - This also adds 'a – ' to the start of compiler error messages
  - Not being recognized by XLF error parser
- Will be fixed in 8.0.2
  - See bug 386572
- Synchronize takes about 30 seconds prior to build
  - Annoying, but not a show stopper
  - It would be nice to make synchronization smarter so it only happens when really necessary

Running

Recommendation

- Investigate improvements to creating launch scripts for batch schedulers

- Data files and launch scripts in separate directory
- This was synchronized with a separate project
  - Large data files were excluded from the synchronize
- Took some time to get the correct parameters for job to run
  - Some LL installations use different default settings than others
  - This was done manually from the command line
  - Required frequent edit/submit/check cycles
- Painful to do this from within Eclipse
  - Need to edit script, wait for sync, open launch config, update script, Run
  - No access to commands to check status of job, e.g. 'llq –s'
- Need to investigate better workflow for creating launch scripts
- Need to investigate providing additional information about jobs