Timing Study for RELAP5-3D Version 4 Series, 2018

Dr. George L Mesina, INL

2018 RELAP5 International Users Seminar
May 3-4, 2018
Idaho Falls

INL/EXT-18-44268

www.inl.gov

Idaho National Laboratory

Outline

- Background
- Plan
- Results
- Future Work

Background

- Reports from two sources indicated RELAP5-3D had slowed down considerably from older versions to present.
    - This prompted a timing study
- Some known causes of code slowdown
    - Replacing common blocks with allocatable arrays
        - Done so RELAP5-3D can configure itself to the exact size needed to run the input model
    - Use of array sections
    - Use of pointers
    - Use of derived types
    - Addition of new features increasing computations
    - Correction of errors requiring extra computation
    - Disruption of vector loops
    - Addition of subroutine calls or I/O within loops

Background

- Speed / Accuracy / Robustness
  - You can have 2 of the 3, but not all 3
- Version timing comparison is difficult
  - Cannot just extract timing from legacy output files
  - Timing depends upon
    - Compiler (manufacturer, version, and options)
    - Operating system software
    - Computer platforms
    - Computer activity at time of run
    - other factors

Plan

- Install, run, collect timings from RELAP5-3D
  - Run multiple times and calculate average run times
  - Establish Suite of Longer Running problems for comparison
- Automate for multiple versions
  - Requires adapting install scripts for current O/S
  - Requires same input problems and compiler options
- Automate alteration of compiler options for all versions
- Use Verification files to find decimal place change
- Profile code versions
  - How do time consuming subroutines change with compiler options

Uniform Testing

- Run everything on same computer, same O/S, similar load
  - Preferably not a server, but single-user platform
- Modify compiler options. Options must be:
  - Compatible with all versions tested
  - Available on most recent compiler level
  - Only relap/ and envrl/ directories necessary
- Same input models. Ensure same behavior
  - Use same set of input files
  - Some card-one options change meaning
- *Applying newer compilers, options, O/S libraries & software may cause older versions to run slower or faster than when originally run*

Adjust older versions to build/run/time

- Installation scripts need adjustment for
  - O/S changes to location of libraries
  - Eliminated or added application software
  - Compiler command flags that changed meaning or got eliminated
  - Add capability to run timing software to runx
- Backfit older RELAP5-3D distributions may not have had some directories and files, or they were renamed
    - PVMexec renamed to R5exec
    - New directories
      - Modules
      - run/verify
      - run/Timer
- Add scripts and replace input files as needed

Adjust older versions to build/run/time

- Older versions (<4.2.0) need additional adjustment
  - Replace license directory and coding that invokes it
  - Eliminate recently-disallowed Cray pre-compiler directives
  - Command flags in dtsgxxx
- Some non-released internal versions were excluded from the study
  - Too much work: E.G. 2.4.2 and 4.0.2
  - Beta test versions: 3.0.1 and 3.5.0
  - Fortran 90 conversion processing active: 2.4.3 through 2.9.9

Average Runtimes

- Random fluctuations in computer loads cause timing variations
- To overcome short term load differences, averages of five runs per input file are used
  - This does not remove long term differences such as the difference between day time and nighttime activity
  - All the runs were made in the daytime
- Runs were made with the following
  - Standard installation test suite
  - Verification suite
  - Long Run suite

Long Run – Suite of Longer Running Problems

- Artery Flex – Simulation of an artery using flexible wall
- Hex2d1 – Steady state run of IAEA hexagonal test case with user type feedback
- pois_cyl_he – Develop Poisueille Flow using Helium in a three-dimensional component representation of a cylindrical pipe
- Sschf1 – Modified version of Bennett's Heated Tube problem
- Todcnd – Pipe with a hot wall
- Typ12002 – Typical PWR run to 1200 second using semi-implicit
- Typ1200n2 – Typical PWR run to 1200 second using nearly-implicit

Timing Script and Programs

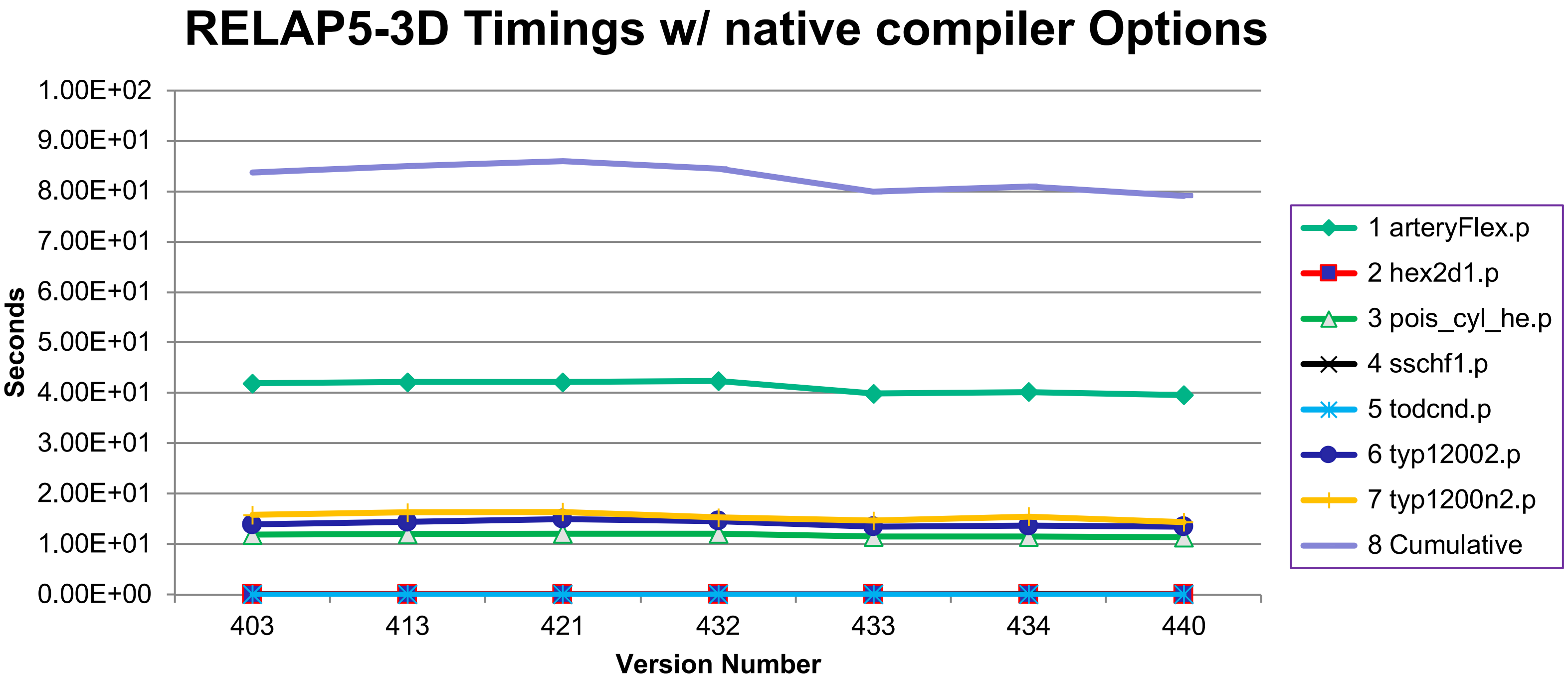
- New higher level scripts
  - rinstall - Install and time a code version
  - minstall - Install and time many code versions
  - TimeLongRun - Insert and run LongRun in a code version
  - FixCoptions - Change the compiler options, re-install, and rerun
  - Extract timing information from a collection of RELAP5-3D output files and place in a timing file
- Fortran90 programs
  - Calculate average time from 5 timing files creating an average time output file
  - Correctly compare average time files for two different versions even if some of the output files do not exist in one version

Detailed Output for Comparing Two Versions

```
Compare r3d413t to r3d421t
    * Left (1st) and right (2nd) files: Time_r3d413t.Tavg, Time_r3d421t.Tavg
    * Faster means right (2nd) time < left (1st) time
        Input model,  left time, right time,    diff.,  percent,   rate
       arteryFlex.p:  3.9904E+01 3.9373E+01     0.531      1.330  Faster
           hex2d1.p:  1.2098E-01 1.2181E-01    -0.001     -0.686 slower
      pois_cyl_he.p:  1.1434E+01 1.1330E+01     0.104      0.912  Faster
           sschf1.p:  5.2541E-02 4.9974E-02     0.003      4.886  Faster
           todcnd.p:  4.1683E-02 7.2336E-02    -0.031    -73.538 slower
         typ12002.p:  1.3304E+01 1.3777E+01    -0.473     -3.556 slower
        typ1200n2.p:  1.5592E+01 1.5100E+01     0.492      3.157  Faster
Number faster / slower / same:   4   3   0
Average of percentage change:  -9.642E+00
Suite Runtime: 1st, 2nd: 8.0449E+01, 7.9824E+01
Change in test suite runtime:    -0.625
Percentage change in test suite runtime:    -0.777
On average, Time_r3d421t.Tavg is FASTER than Time_r3d413t.Tavg
```

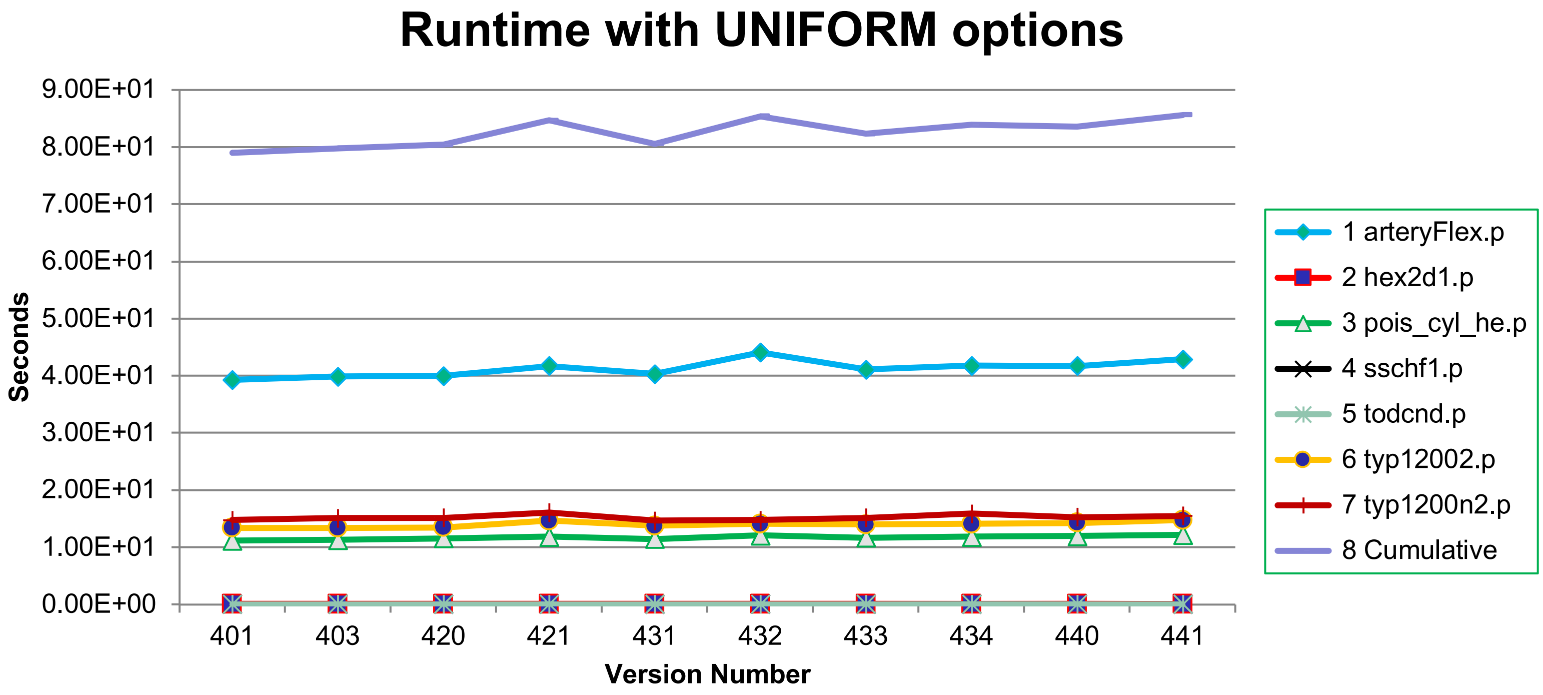
Graphical Comparison of Many Versions

- LongRun suite problem timings for recent Released Versions and the three most recent internal developmental versions.
- Top graph shows cumulative time
- With **native** compiler **options** runtimes show **code SPEEDUP**!



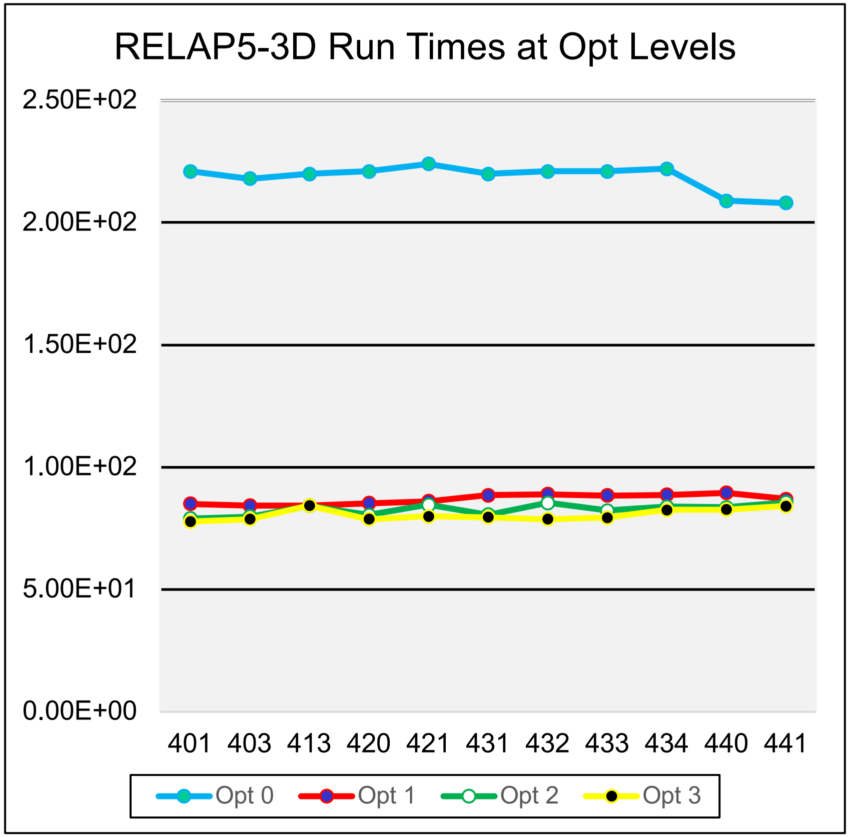
Graphical Comparison of Many Versions

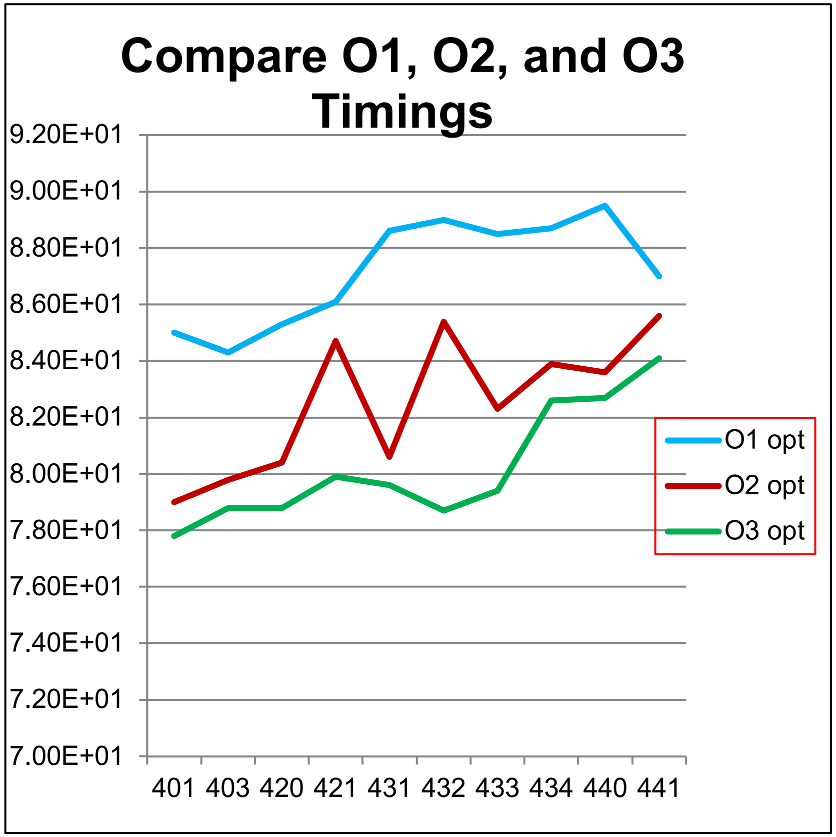
- LongRun suite problem timings with all versions having the same compiler options.
- **Uniform compiler options** showed code **SLOWDOWN**
  - Expected result



Cumulative Time Comparison O0 to O3

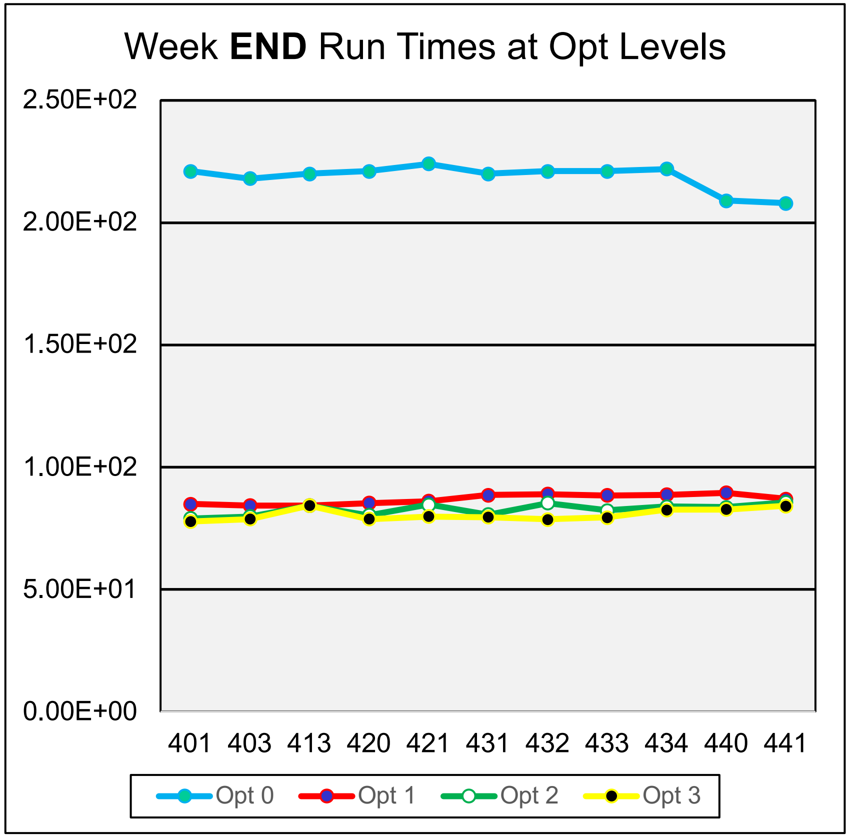
- Opt level 0 much slower than others
- Level 3 gives best speed. Expected Result

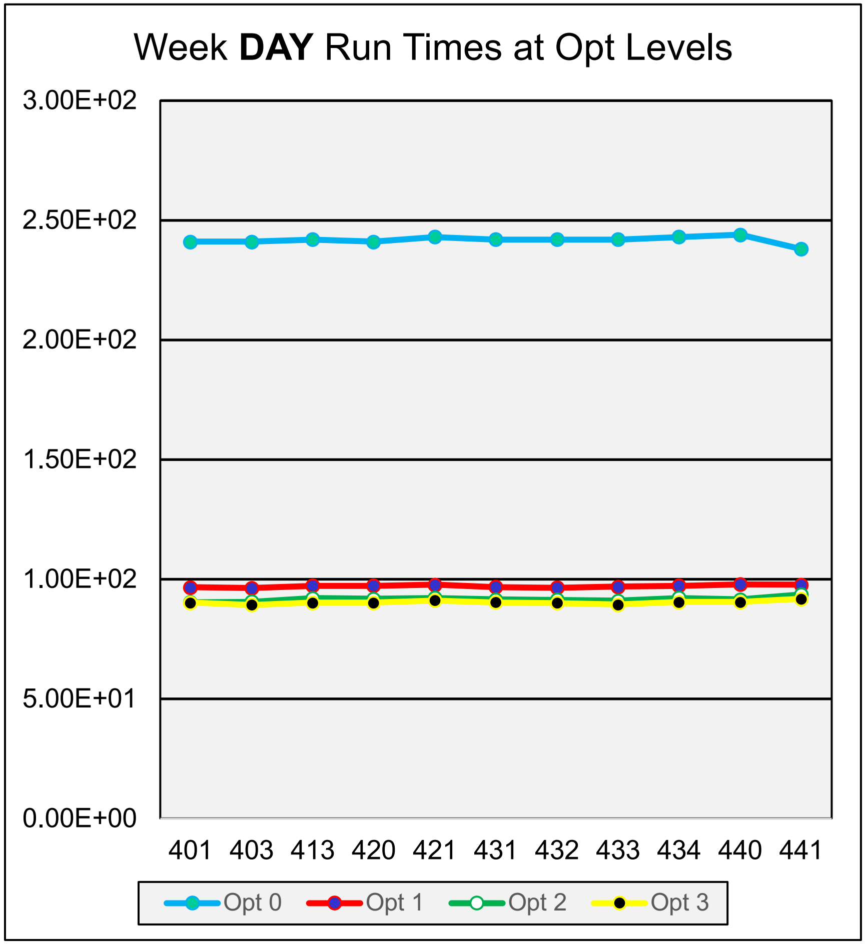




Cumulative Time: Day of Run Comparison

- Weekday runs slower. Runs while working are even slower despite extra processors





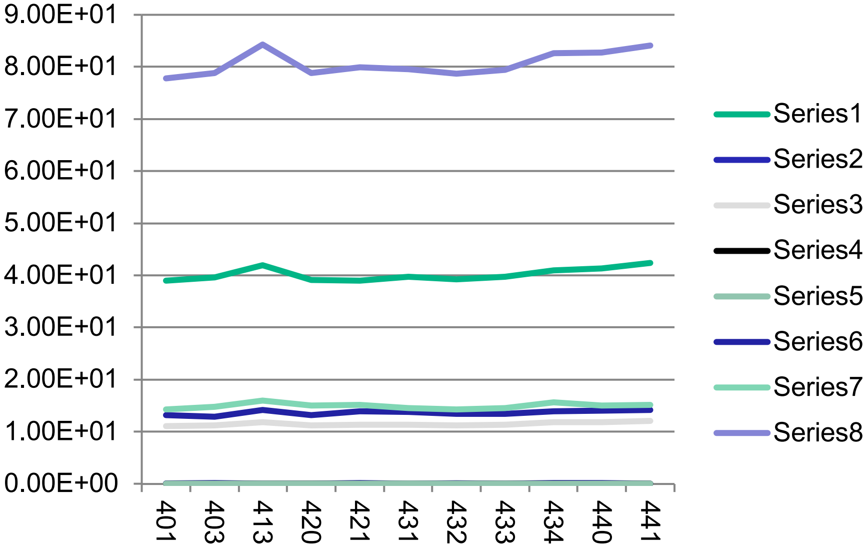
Tabular Comparison of Opt Levels 2 and 3

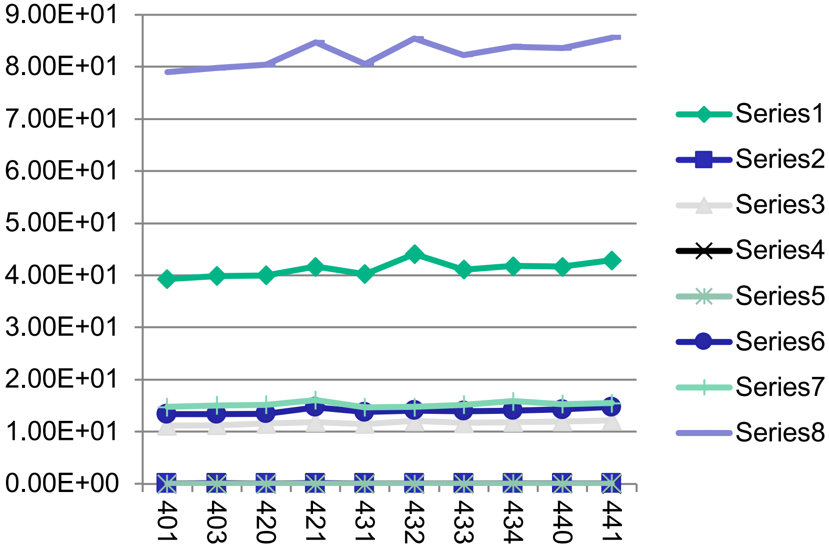
- On the LongRun suite, average timings with all versions are made using optimization levels 2 and 3.

arteryFlex	3.16%
hex2d1	-2.11%
pois_cyl_he	2.29%
sschf1	1.90%
todcnd	-8.59%
typ12002	2.87%
typ1200n2	2.64%
cumulative	2.87%

- The average performance across all code versions for individual problems are shown in the table
- Most problems run faster with opt level 3
  - Problem todcnd1 runs in a fraction of a second and was included because it tends to run slower for better compiler levels and newer versions

Average Runtime 2.9% Faster w/ O3 than w/ O2



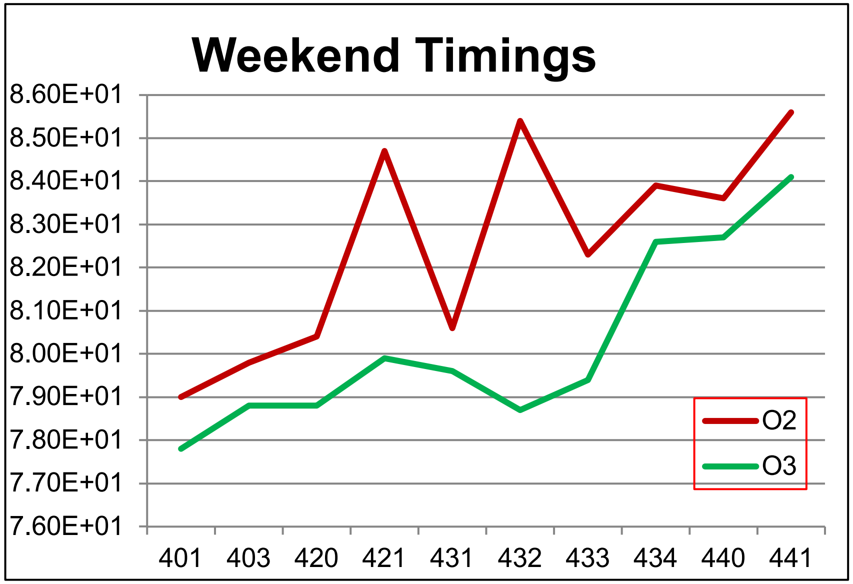


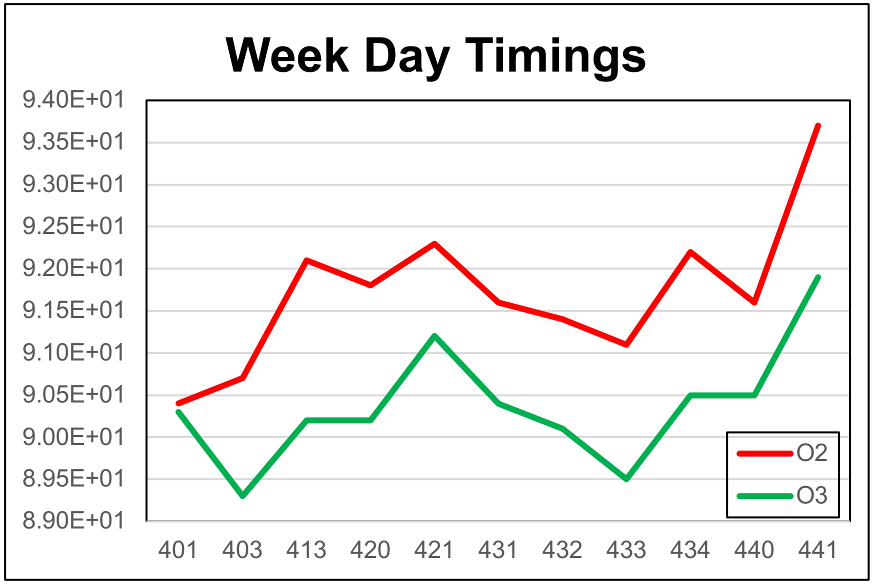
- LongRun test suite timings with –O3 on left and –O2 on right
- Runtime performance varies with individual input models
  - Code is 1.9% to 3.2% faster for most tests

Intel Statement about O3 Optimization Level

- Performs O2 optimizations
- Enables more aggressive loop transformations such as Fusion, Block-Unroll-and-Jam, and collapsing IF statements
- May set other options, depending on the compiler, operating system, and architecture. Options set may change from release to release
- Combined with options -ax or -x (Linux) or with options /Qax or /Qx (Windows), the compiler performs more aggressive data dependency analysis, which may cause longer compilation times
- May not produce higher performance unless loop and memory access transformations take place. The optimizations may slow down code in some cases compared to O2 optimizations.
- Option **O3 is recommended for applications that have loops that heavily use floating-point calculations** and process large data sets.
- Many routines in the shared libraries are more highly optimized for Intel® microprocessors than for non-Intel microprocessors

Cumulative Time Comparison O2 vs. O3





- O3 smaller variance and increase from 4.0.3 to 4.4.1
  - O3 Standard Deviation = 2.24, 0.68, Increase = 4.6%, 1.8%
  - O2 Standard Deviation = 2.26, 0.85, Increase = 4.9%, 3.7%
  - First value is weekend, second is weekday
- ***Is it time to consider making optimization level 3 the default?***

Conclusions

- Study performed using collection of 8 standard input cases
  - Apples to apples comparison requires modification of older versions to install, have same meaning of input, same compile options, etc.
  - All installation changes now automated: 401 through 441
- Detailed comparisons between two versions can be produced
- 2018 Code (441) modestly slower than older ones such as 401 or 403.
  - Ratio of average runtimes depends on run conditions and compiler options ranging from 1.8% to 4.9%
- Optimization level makes noteworthy difference in runtime
- It may be worthwhile to switch to optimization level 3
  - Comparison of all “release” test cases using O2 and O3 would be necessary