Fortran Compiler News

October 01, 2014

Hi all,

Quite a bit going on again this month.

1) PGI 14.9 and Intel 15.0.0 are now on yellowstone. Both have some relevant bug fixes, but particularly Intel, which has not been updated for some time.
1) CLM has had some issues with PGI 14, which seem to be a consequence of having multiple types in the same scope, where those types have methods with the same name (in this case, multiple unrelated classes that have an "UpdateAccVars" method). PGI says that the release this month should fix the issue; in the meantime I believe that Stefan has committed a workaround.

2) Little-known fact: PGI has a 264 character line limit which is enforced *after* preprocessing, so it can be confounded by __FILE__ macros that expand to a very long string.
3) I discovered a PGI compile error that occurs when using move_alloc on components of derived types.

4) Other PGI issues are still being looked at by their compiler team, which is working with CISL and myself. E.g. figuring out pointer-related bugs, compiling pFUnit.
5) We continue to have issues with CAM-FV runs on Intel 14 and Intel 15, though Jim has a workaround that turns down optimization on a couple of MPI-related modules. We still don't know exactly what code is triggering the error.

6) I've temporarily dropped work on getting OpenMP to work with NAG 6.0, because I hit a wall (and have other things to do). But in the process, I created patched OpenMPI and valgrind builds that may actually be more useful. I've made some progress toward getting basic CLM and CAM cases to run cleanly under valgrind. This is good because none of our supported compilers have very good uninitialized data checks. OpenMPI also has its own MPI memchecker tool which works with valgrind, but I haven't used it yet.

Scoreboard:
Closed:
- GNU: 3
- IBM: 1
- Intel: 10 (+4)
- NAG: 6
- PGI: 30 (+2)

Open:
- GNU: 6
- IBM: 3
- Intel: 5 (-3)
- NAG: 2
- PGI: 8 (2 of these are likely fixed in 14.9, but I haven't tested this, 1 more is likely fixed a while ago)

Total:
- GNU: 9
- IBM: 4
- Intel: 15 (+1)
- NAG: 8
- PGI: 38 (+2)

- Sean

September 02, 2014

Hi all,

It's that time of month again. Lots and lots of news this time.

1) I've updated the bug list with information about the actual versions we test with (also, what we would like to be supporting based on the conversation we had at the CSEG meeting mid-month).
2) NAG has released version 6.0. This *may* allow us to use it with OpenMP since some significant bugs have been fixed, and OpenMP 3.1 is now fully supported. I'm still intermittently working on that on goldbach; the hitch is clmtype/clmtypeInitMod, which is causing some kind of run-time error.
3) While GCC has always had trouble with arrays of characters, NAG also seems to have a problem with functions that return allocatable character arrays.
4) PGI has an internal error if you use "move_alloc" to move an array into a derived-type component.

5) I've confirmed that most of the open tickets I have with PGI have been fixed as of version 14.7. At least two out of the remaining issues (including the move_alloc one) will be fixed in imminent 14.9 release. PGI is catching up to Intel and IBM so far as known bugs are concerned, although it will take a bit of time before newer versions of PGI are installed on systems that we support. So, kudos to the various people in CISL and to the PGI/NVIDIA folks that we're talking to for pushing that forward.
6) PGI also has started testing using cheap CESM cases (I believe low-res CAM-SE F compsets), and they are working on getting pFUnit to compile, too.
7) On the other hand, PGI 13/14 has at least one regression that causes some kind of machine-specific reproducibility error when auto-vectorization is enabled. I've heard something about this on Bluewaters, and I had an issue that may or may not be related on Intel systems with my gravity wave code.
8) GCC has been on version "4.x" for nearly a decade, so it has changed its version numbering scheme for future updates. To summarize, the next major stable version of GCC will be 5.1.0, and the minor updates for that will be 5.2.0, 5.3.0, and so on. The major version after that will be 6.1.0, with minor updates 6.2.0, 6.3.0... You get the idea.
9) In the meantime, gfortran 4.9.1 claims full support for OpenMP 4.0.
Scoreboard:
Closed:
GNU: 3
IBM: 1
Intel: 6
NAG: 6 (+2)
PGI: 28 (+3)
Open:
GNU: 6
IBM: 3 (-1)
Intel: 8
NAG: 2 (-1)
PGI: 8 (+1)
Total:
GNU: 9
IBM: 4 (-1)
Intel: 14
NAG: 8 (+1)
PGI: 36 (+4)
(The "-1" IBM bug is one where the test case turned out to be invalid, i.e. the bug was in my code and not the compiler.)
- Sean

August 01, 2014

Hi all,

A couple of new bugs, but some good news from PGI this time.

1. There is a very strange bug on XLF where optimization of division in an argument can apparently mess up the compiler's ability to distinguish constructors.
2. I've started experimenting with the F2008 "contiguous" attribute, purely for performance, and I've logged a couple of GNU and PGI bugs for that feature.
3. NAG has an internal compiler error if you do something like this with a generic method:
   foo = [(a%b(i), i = 1, n)]
   (This is "implied do" syntax, a limited form of list comprehension.)
4. PGI has notified me that several more of the bugs I have reported are fixed in PGI 14.7. I believe that CISL is in the process of getting an PGI 14.7 update on yellowstone, so we can see how that version does soon.

Scoreboard:
Closed:
GNU: 3 (New count)
IBM: 1
Intel: 6
NAG: 4
PGI: 25 (+4)
Open:
GNU: 6 (New count. Probably very incomplete; array components have issues.)
IBM: 4 (+1)
Intel: 8
NAG: 3 (+1)
PGI: 7 (-3)
Total:
GNU: 9 (New count.)
IBM: 5 (+1)
Intel: 14
NAG: 7 (+1)
PGI: 32 (+1)
-Sean

July 01, 2014

Hi everyone,

Another slow month (which is usually good, since it means not too many bugs). A couple of news items about bleeding edge versions:

1. GCC has added IEEE support for Fortran to their trunk. It will be a while before version 4.10 comes out, but when it does, it should have the IEEE intrinsic modules. gfortran is last of our supported compilers to add this feature.
2. PGI has responded to many of the issues with object-oriented features that I've been tracking. We can hope that version 14.6 will be easier to work with.

Scoreboard:

Closed:

IBM: 1

Intel: 6

NAG: 4

PGI: 21 (+9)

Open:

IBM: 3

Intel: 8

NAG: 2

PGI: 10 (-9)

Total:

IBM: 4

Intel: 14

NAG: 6

PGI: 31

blocked URL
-Sean

June 06, 2014

Hi all,

It has been a relatively quiet month for compiler bugs. Here's the list:

1. PGI has reported a very large number of Fortran bugs fixed with 14.4 and 14.6. I haven't updated the wiki yet because I don't have access to these versions, but probably a lot of those bugs are fixed this year.

2. PGI 13.9 seems to occasionally have errors for assignments involving array pointers. I only have one example of this, and it has proven difficult to create a small test case (e.g. the bug goes away with added print statements). This may correspond to one of a few pointer optimization bugs that PGI reports as fixed in recent versions.

3. Intel 13 has bugs in a few more F2003 edge cases, but these are mostly fixed in Intel 14. There's an internal compiler error in some cases if you mix and match scalar and array arguments to "min" or "max", and assign the result to an allocatable array.

Scoreboard:

Closed:

IBM: 1
Intel: 6 (+3)
NAG: 4
PGI: 12

Open:

IBM: 3
Intel: 8
NAG: 2
PGI: 19 (+1)

Total:

IBM: 4
Intel: 14 (+3)
NAG: 6
PGI: 31 (+1)

-Sean

April 30, 2014

Greetings, all!

I'm thinking of doing this around the first of each month. Or at least just the "scoreboard" part, since I like fuzzy statistics.

The biggest issues found since the last update:

1. I sent an email earlier about an Intel bug that gets wrong answers from reduction intrinsics (specifically "sum" and "product"), at level -O2 and higher. The best advice I can give right now is to avoid any complicated indexing in arguments passed to sum or product. This might be one of the errors that's well suited for the validation (verification?) tool to discover, since it can compare answers from different compilers/options.

2. I (re-)discovered a serious bug in PGI having to do with polymorphic objects that have more than one method attached to them. This was addressed in PGI 13.7, but until we drop support for earlier versions, it's a pretty serious constraint on uses of polymorphism.

3. Several compilers (Intel, GNU, PGI) encounter segfaults when they allocate components of objects that have been marked with an OpenMP "private" clause. Strictly speaking, allocatable components were not supported by the OpenMP standard until OpenMP 4.0.

   To avoid problems, you can give each thread one object out of a shared array, or you can try to push the object down into a routine that is called after the OpenMP parallel region has already started. Either case ensures that a valid object is created for each thread.

4. GCC 4.9 was released with "preliminary" support for finalization, as well as for types that contain character variables with an allocatable length. These are two F2003 features that our other compilers already support, meaning that in the future we may be able to use them.

   However, gfortran still has problems with **arrays** of character variables that have allocatable length.

5. The scoreboard of tracked bugs. The numbers in parentheses are the differences from the last time I emailed these.

Closed:

IBM: 1

Intel: 3

NAG: 4

PGI: 12 (+2)

Open:

IBM: 3

Intel: 8 (+2)

NAG: 2

PGI: 18 (+2)

Total:

IBM: 4

Intel: 11 (+2)

NAG: 6

PGI: 30 (+4)

There are no new compiler versions in this list, so the only changes are discoveries of pre-existing bugs.

-Sean

April 10, 2014 (Intel bug)

Hi all,

By reducing a test case from Tom Clune, I created a reproducer for an Intel regression, which can be seen in the case attached (or see the link below).

https://wiki.ucar.edu/display/ccsm/Fortran+Compiler+Bug+List#FortranCompilerBugList-Intelinlineproductonsection

The attached case is very short and simple Fortran 90 code, which correctly prints "1" if compiled with -O1, and incorrectly prints "0" if compiled with -02. This is a fairly dramatic example, but it's possible to get much smaller errors out of this bug (e.g. nudging the answer by a few percent).

With this test case, I found that this bug is in every Intel 14 and 13 version I could find dating back to 13.0.1, but not Intel 12.1 (on pleiades).

This bug probably doesn't happen very much; you have to use sum or product, with an array section in just the right way, with certain loop bounds, with assignment into an array.

But it's very sneaky. You could potentially have code that works perfectly fine, increase some hard-coded array bound, and then it would be broken. I don't really have a great recommendation for this, except that if code breaks in a weird way with Intel, you can always try -O1 or -O0 on it and see if it gets better.

-Sean

April 10, 2014

Hi all,

It's been a busy week and a half for me; I have recently filed around 10-15 bug reports on issues in Intel and PGI. The wiki page is updated as usual, but I want to give a summary of the big things to avoid.

1. Fortran has "structure constructors", which are simple constructors that are predefined for you. Don't use them except for the simplest types, with the simplest of arguments, because both Intel and PGI have trouble with them.

2. I've heard 5 separate users in the past week (plus me), complain about late versions of Intel 13 and Intel 14 doing things that look like either stack corruption, or optimization gone awry. Unfortunately, except for one very convoluted test case, these have only shown up in runs of big code bases (e.g. CESM system tests). It's not obvious (to me) whether there's a serious regression or two going around, or if it's different people coincidentally encountering different issues at about the same time.

   I don't think that *all* of these cases are really related. But if anyone has such an issue and wants to try reproducing on a higher/lower Intel version (or different machine) than they originally used, that might be informative.

3. PGI still racks up almost as many bug reports as *every other compiler combined*.

   Two rules of thumb. First, compile with PGI frequently, especially object-oriented code, because you often have to design around bugs.

   Second, if you encounter a internal error on PGI, and you don't know the cause, see if you get a different error message on Intel or NAG. Sometimes PGI just segfaults and core dumps on invalid code, whereas another compiler gives you an actual error message. If you fix that other compiler's error, PGI might work again too.

4. Here's a little summary of compiler bug counts that I've tracked so far. These are actual bugs that I've encountered or been asked about over the past, I don't know, something like 6 to 9 months.

Closed bugs:

Intel: 3

NAG: 4 (plus part of 1 open bug)

PGI: 10 (plus parts of 2 open bugs)

IBM: 1

Open bugs:

Intel: 6 (plus suspected bugs that have not been isolated)

NAG: 2

PGI: 14 (plus two "families" of bugs that may have multiple causes and change symptoms from version to version)

IBM: 2 (plus one that's probably invalid)

Totals:

Intel: 9

NAG: 6

PGI: 26

IBM: 4

I didn't count the GNU bugs, because a) we don't seem to spend as much effort and testing time on the gfortran port, so I just don't have much of a record, and b) a lot of its "bugs" are really unimplemented or partially implemented features, which are hard to count. (Do you count all the problems with allocatable strings as one bug, several bugs, or a couple of major unimplemented features?)

IBM probably would have a higher bug count, except that I never really get questions about XLF, so I don't have much to go on there.

-Sean

March 11, 2014

Hi all,

A reminder about the location of the wiki page tracking compiler bugs:

https://wiki.ucar.edu/display/ccsm/Fortran+Compiler+Bug+List

Alice has helpfully added this link to the CSEG development tools page.

I'll point out two types of bugs that can be difficult to recognize:

1. Intel sometimes encounters an internal compiler error because of entities that are *re*-exported from a module. If there's an internal compiler error for no clear reason, it may be that one of the dependencies of the module (or dependencies of dependencies) is to blame.
2. PGI continues to have issues with use statements that are at the procedure level (as opposed to the module level). There was some improvement in this situation between PGI 11 and PGI 12. Unfortunately, there are still rare cases where moving a use statement is necessary to work around a bug.

Lastly, I want to point out that gfortran 4.9 is slated for April, and will close part of the feature gap between it and other compilers. You can make allocatable-length characters a component of derived types (though not an array of them), and it will have partial support for finalization. In both cases, every other compiler we support already has already had these features for years, so hopefully next year we can widen our accepted subset of Fortran 2003.

On the down side, GNU is breaking both module and binary compatibility again, so any and all Fortran 90 libraries will need to be rebuilt with the new version.

-Sean