2018

Fortran FLUSH statement SYNC= specifier proposal

Nathan T. Weeks
Iowa State University, weeks@iastate.edu

Fortran FLUSH statement SYNC= specifier proposal

Nathan T. Weeks
weeks@iastate.edu
Iowa State University
April 14, 2017

Introduction

This paper describes a proposed change to the Fortran standard to provide a mechanism for Fortran programs to ensure that the transfer of data to an external file via preceding WRITE statement(s) has completed.

The Fortran 2008 FLUSH statement behavior is largely processor dependent, and can make no such guarantees:

1. (Fortran 2008, Note 9.58) *'Because this part of ISO/IEC 1539 does not specify the mechanism of file storage, the exact meaning of the flush operation is not precisely defined. The intention is that the flush operation should make all data written to a file available to other processes or devices, or make data recently added to a file by other processes or devices available to the program via a subsequent read operation. This is commonly called "flushing I/O buffers"'.*
2. Fortran 2008 Annex A.2 lists the processor dependent behavior: *the action caused by the flush operation, whether the processor supports the flush operation for the specified unit, and the negative value assigned to the IOSTAT= variable if the processor does not support the flush operation for the specified unit (9.9)*
3. A survey of processor-dependent behavior of the FLUSH statement in Fortran compilers supported in the Cray environment on Edison and Cori:
    a. Cray Fortran
        i. The only surveyed implementation that explicitly guarantees data transfer to an external file.
            1. *Execution of a FLUSH statement causes memory resident buffers to be flushed to the physical file.* (Cray Fortran Reference Manual, S-3901-84, p. 138)
    b. GNU Fortran
        i. Explicitly *doesn't* guarantee that data transfer to an external file.
            1. *"...flush the runtime library's I/O buffer so that the data becomes visible to other processes." This does not guarantee that the data is committed to disk.*
               https://gcc.gnu.org/onlinedocs/gfortran/FLUSH.html
    c. Intel Fortran (2017)
        i. Processor-dependent behavior is not entirely clear from the documentation. I strace'ed a simple program containing FLUSH, and didn't observe a system call resembling an fsync(), so "other processes"

may refer to other processes on the same host (i.e., flush only runtime library I/O buffers like gfortran)

1. *"Causes data written to a file to become available to other processes or causes data written to a file outside of Fortran to be accessible to a READ statement."* https://software.intel.com/en-us/node/679280

ii. Intel Fortran does support a (non-portable) COMMITQQ function (https://software.intel.com/en-us/node/692664) that appears to behave like the Cray FLUSH statement

Motivation

1. **Application checkpointing**
   Large-scale or long-running HPC applications may periodically checkpoint state to an external file. In the event of a failed image, the application can restart from the point within execution at which the last checkpoint file was created. The behavior of the FLUSH statement in the Fortran 2008 (and tentatively Fortran 2015) is largely processor-dependent, resulting in no portable mechanism for a Fortran program to ensure that data intended for a checkpoint file was transferred.
2. **Coordinate I/O to a single file**
   The Fortran 2008 restriction that effectively prohibits multiple images (or MPI ranks, etc.) from having the same file open at the same time (9.5.4.4: "...a file shall not be connected to more than one unit at the same time") has (tentatively) been lifted in Fortran 2015 and made processor-dependent (12.5.4.4: "It is processor dependent whether a file can be connected to more than one unit at the same time"). For processors supporting such connections, Fortran image control statements + the FLUSH statement are insufficient to coordinate I/O to an external file from multiple images.

   While MPI + MPI I/O may be the preferred solution for providing single-file I/O capabilities for the foreseeable future, it would be desirable to have the basic facility available to non-MPI Fortran applications (e.g., using coarrays) for processors that will support it.

Current Approaches

On POSIX systems, a current workaround for the lack of a Fortran-standard mechanism for ensuring external file I/O completion is:

1. Create an interface block to fsync() (or fdatasync()),
2. Retrieve a file descriptor using an implementation-specific procedure; e.g.
   a. gfortran: FNUM (https://gcc.gnu.org/onlinedocs/gfortran/FNUM.html)
   b. Intel Fortran: PXFFILENO (https://software.intel.com/en-us/node/679699)
3. Call fsync()/fdatasync() with the file descriptor retrieved in step #2 as the argument

This approach is demonstrated here:
https://gcc.gnu.org/onlinedocs/gfortran/Data-consistency-and-durability.html

To support multiple compilers, preprocessor directives must be utilized to select compiler-specific procedures for obtaining a file descriptor from a Fortran file unit number. Non-POSIX systems (e.g., Windows) may require different procedure calls (with additional preprocessor directives to support).

Proposal

This paper proposes the addition of a SYNC= specifier to the Fortran FLUSH statement.

- When SYNC='YES', the FLUSH statement does not return until data from previous WRITE statements has been transferred to an external file connected to the specified file unit number.
    - *COMMENT:* The data should subsequently be available to other *images* (potentially executing on different hosts) that access the same external file (e.g., residing on a shared file system) if supported by the processor.
- When SYNC='NO' or if the SYNC= specifier is omitted, then the behavior of the FLUSH statement is processor dependent.
    - *COMMENT:* e.g., a processor may flush user-space Fortran runtime library I/O buffers, leaving data potentially in the kernel-space page cache, and accessible only to other processes / images / threads on the same host.
- In all cases, if the processor does not support the specified FLUSH statement on the specified file unit number, then the IOSTAT= specifier is set as currently worded in the Fortran 2015 draft (N2123):
    - *a processor-dependent negative integer value different from IOSTAT_EOR and IOSTAT_END, if the IOSTAT= specifier appears in a FLUSH statement and the processor does not support the flush operation for the specified unit.*

Discussion

A processor for a POSIX system may implement SYNC='YES' via fdatasync(), which effectively guarantees that data have been transferred ("synchronized I/O data integrity completion"), but unlike fsync(), does not guarantee file metadata/attributes (e.g., ctime, mtime, atime) have been transferred. Fortran has no concept of such file attributes, but a processor is of course still free to implement SYNC='YES' with a system call like fsync().

Conceptually, for correct behavior, SYNC='YES' needn't actually force or hasten the transfer of data to an external file, but rather need only cause the FLUSH statement to block until the transfer is complete.

In addition to the FLUSH statement, it might be convenient to the application programmer to also support the SYNC= specifier in CLOSE and/or WRITE.