

CAF Validation suite

Siddhartha Jana

HPC Tools Group

Department of computer Science

University of Houston,

Houston, Texas, USA

sjana2@uh.edu

February 9, 2013

Abstract

This report acts as a guide for users of the UH-CAF validation suite

1 CAF Validation suite

This test suite can be used to evaluate a CAF compiler implementation to determine the extent to which it supports the coarray-features of the Fortran 2008 standard in accordance with ISO/IEC 1539-1:2010 (E)[1].

1.1 Test Categories

1.1.1 Feature Tests

These tests can be found under `$(VALIDATION_PATH)/should_pass`. These test the correctness of the basic constructs/semantics of the Coarray FORTRAN. The tests mainly focus on the verification of the correctness of contiguous and strided remote read/write operations, the coarray syntax, the use of strided co-subscript notation, coarrays of different data types and image-query intrinsics.

1.2 Confidence Tests

These tests can be found under `$(VALIDATION_PATH)/confidence_tests`. All tests in this category verify CAF constructs whose primary purpose is to maintain consistency among images. Such constructs aid in avoiding race conditions. The tests include computation intensive loops or calls to the `sleep()` intrinsic for intentionally introducing delays in progress of certain images. This enforces an increase in the likelihood of races in case of incorrect implementations of the constructs being tested.

To determine whether, given certain values of the configuration parameters, the tests are capable of correctly evaluating the configuration parameters or not, every tests has a CROSS TEST version. Such tests include the exact same code like the original version but with certain statement deleted / replaced with semantically incorrect sections of code. Most of the time, the modifications to the original code is in the form of the absence of the CAF statements which are being tested in that program. In order to reduce redundant code, we prefer to include the CROSS test version and the original version all in one file, using conditional `'#ifdef'` definition of the macro `CROSS_`.

1.3 Fault Tests

These tests can be found under `$(VALIDATION_PATH)/fault_tests`. These test the correctness of the implementation of specifiers used for handling normal and error termination. Here's an excerpt describing these 2 types [from Page 23, section 13 of "ISO/IEC JTC1/SC22/WG5 N1824" [2]]:

It seems natural to allow all images to continue executing until they have all executed a stop or end program statement, provided none of them encounters an error condition that may be expected to terminate its execution. This is called normal termination. On the other hand, if such an error condition occurs on one image, the computation is flawed and it is desirable to stop the other images as soon as is practicable. This is called error termination.

Normal termination occurs in three steps: initiation, synchronization, and completion. An image initiates normal termination if it executes a stop or end program statement. All images synchronize execution at the second step so that no image starts the completion step until all images have finished the initiation step. The synchronization step allows its data to remain accessible to the other images until they all reach the synchronization step. Normal termination may also

be initiated during execution of a procedure defined by a C companion processor.

An image initiates error termination if it executes a statement that would cause the termination of a single-image program but is not a stop or end program statement. This causes all other images that have not already initiated error termination to initiate error termination. Within the performance limits of the processor's ability to send signals to other images, this propagation of error termination should be immediate. The exact details are intentionally left processor dependent.

1.4 Detecting / Reporting of errors

- Every test includes a module called 'crosstest'.
- The module 'crosstest' (defined in file testmofule.f90) includes the declaration of an integer scalar coarray called 'cross_err' which is modified by image with rank 1 on detecting an error. It also contains two subroutines - calc_ori and calc. The former returns the test result when the specific construct is being tested. The latter is called by the cross-test version and returns the confidence with which the original test passed.

1.5 Specific Tests

If the user wants to evaluate a CAF implementation in terms of the support to specific types of tests, the names of the tests can be specified in the file "test_file in the directory" validation\ few_tests.

The results of the tests listed in latest_results.log are all subdivided into the corresponding categories.

1.6 List of tests

The different types of tests are listed in Tables 1 2 3:

1.7 Configuration Parameters

Table 4 lists the different options that need to be set for the test-suite. These options can be initialized in the CONFIG file in the "config" directory. The make.def version of this file is auto-generated by running the script config2makedef.sh in the same directory.

1.8 APPENDIX

Tables 5 6 7 list the extent of support of coarrays in different CAF compiler implementations as detected by the UH CAF validation test suite:

References

- [1] ISO. "international standard ISO/IEC 1539-1:2010 (E) - Draft (for Ballot)", Third edition. June.
- [2] R. W. Numrich and J. Reid. Co-arrays in the next fortran standard. *SIGPLAN Fortran Forum*, 24(2):4–17, August 2005.

Table 1: Feature Test files in the UH - CAF Validation Tests suite

| File | Description |
|-------------------------|---|
| character_test.f90 | CHARACTER coarrays |
| coarray_2.4.7.6.f90 | similar translation of co-subscripts and subscripts |
| coarray_4.8.R468.f90 | reference of coarray without [] implies local object |
| coarray_5.3.6.1.f90 | attribute CODIMENSION + remote accesses at single integer/real boundary |
| dummyargs_12.3.2.2c.f90 | explicit shape, assumed size, assumed shape, allocatable dummy args |
| intrin_13.7.126.f90 | NUM_IMAGES() returns the number of images launched |
| intrin_13.7.165.f90 | THIS_IMAGE(), THIS_IMAGE(coarray), THIS_IMAGE(coarray, dim) |
| intrin_13.7.172.f90 | LCOBOUND(coarray) and LCOBOUND(coarray,dim) |
| intrin_13.7.79.f90 | IMAGE_INDEX(coarray, subs) |
| intrin_13.7.91.f90 | UCOBOUND(COARRAY[, DIM, KIND]) |
| intrin_6.7.3.2.11.f90 | ALLOCATE and DEALLOCATE act as barriers |
| item_4.8.a.f90 | Subobjects of a coarray is also a coarray |
| pointer_4.5.4.6b.f90 | association of pointer components of coarrays with local objects |
| intrin_8.5.7d.f90 | STOP and LOCK construct with STAT=STAT_LOCKED specifier |
| intrin_8.5.7e.f90 | STOP and LOCK construct with STAT=STAT_LOCKED_OTHER_IMAGE specifier |
| intrin_8.5.7f.f90 | STOP and LOCK construct with STAT=STAT_UNLOCKED specifier |
| derived_4.5.4.f90 | (non-)coarray COMPONENTS of (non-)coarray derived types |

Table 2: Confidence Test files in the UH - CAF Validation Tests suite

| File | Description |
|--------------------|---|
| atomic_8.5.2.f90 | Atomic subroutines |
| critical_8.1.5.f90 | CRITICAL - END CRITICAL sections |
| intrin_8.5.6.f90 | LOCK & UNLOCK without STAT specifier |
| sync_8.5.3.f90 | SYNC ALL without STAT specifier |
| sync_8.5.4a.f90 | SYNC IMAGES(arr) paired with SYNC IMAGES(*) |
| sync_8.5.4b.f90 | call to SYNC IMAGES(arr), should not behave like SYNC ALL |

Table 3: Fault (tolerance) Test files in the UH - CAF Validation Tests suite

| File | Description |
|-----------------|--|
| sync_8.5.7a.f90 | STOP and SYNC ALL with STAT=STAT_STOPPED_IMAGE specifier |
| sync_8.5.7b.f90 | STOP and SYNC IMAGES(arr) with STAT=STAT_STOPPED_IMAGE specifier |
| sync_8.5.7c.f90 | STOP and SYNC IMAGES(*) with STAT=STAT_STOPPED_IMAGE specifier |

Table 4: List of configuration parameters. Here *test_type*= CONF or FEATURE or FEW or FAULT

| Parameter | Description | Compiler specific(y/n) |
|--------------------------------|--|------------------------|
| BIN_PATH | The path to dump all the executables. | NO |
| NPROCS | Number of images to launch | NO |
| NITER | Number of times the test is repeated | NO |
| SLEEP | This is used to intentionally slow down certain images to cause race conditions while testing certain constructs | NO |
| TIMEOUT | This parameter is passed to the perl script call timedexec.pl which ends processes which exceed the given execution time. This is helpful for backing out while executing tests which deadlock due to incorrect implementation | NO |
| COMPILER | compiler name | YES |
| FC | command to invoke the compiler | YES |
| FFLAGS | Flags passed to the compiler. The necessary flags include the options to enable the macro pre-processor and define the macros - NPROCS, NITER and SLEEP. | YES |
| LAUNCHER | command to launch multiple images | YES |
| EXEC_OPTIONS | Flags passed to the launcher after the executable name. Not so common. | YES |
| FFLAGS_CROSS | Flags passed to the compiler while executing the cross tests. Generally the value include all the options listed for FFLAGS, plus the flag to define the CROSS_macro in the tests. | YES |
| <i>test_type</i> _COMPILE_PATH | Path to dump the messages generated by the compiler. | NO |
| <i>test_type</i> _EXEC_PATH | Path to dump the output of the executables / the messages generated by the compiler or runtime. | NO |
| <i>test_type</i> _LOG_PATH | Path to dump the output of the test results. | NO |

Table 5: Results of Feature tests

| SPEC_IDX | DESCRIPTION | OpenUH | Intel | G95 |
|---------------------|--|----------------|----------------|------------|
| character_test | CHARACTER coarrays | YES | YES | YES |
| coarray_2.4.7.6 | similar translation of co-subscripts and subscripts | YES | YES | YES |
| coarray_4.8.R468 | reference of coarray without [] implies local object | YES | YES | YES |
| coarray_5.3.6.1 | attribute CODIMENSION + remote accesses at single integer/real boundary | YES | Exec times out | Exec fails |
| derived_4.5.4 | (non-)coarray COMPONENTS of (non-)coarray derived types | YES | YES | YES |
| dummyargs_12.3.2.2c | explicit shape, assumed size, assumed shape, allocatable dummy arguments | YES | YES | comp fails |
| intrin_13.7.126 | NUM_IMAGES() returns the number of images launched | YES | YES | YES |
| intrin_13.7.165 | THIS_IMAGE(), THIS_IMAGE(coarray), THIS_IMAGE(coarray, dim) | YES | YES | YES |
| intrin_13.7.172 | LCOBOUND(coarray) and LCOBOUND(coarray,dim) | YES | YES | YES |
| intrin_13.7.79 | IMAGE_INDEX(coarray, subs) | YES | YES | YES |
| intrin_13.7.91 | UCOBOUND(COARRAY[, DIM, KIND]) | YES | YES | YES |
| intrin_6.7.3.2.11 | ALLOCATE and DEALLOCATE act as barriers | YES | YES | YES |
| intrin_8.5.7d | STOP and LOCK construct with STAT=STAT_LOCKED specifier | YES | YES | comp fails |
| intrin_8.5.7e | STOP and LOCK construct with STAT=STAT_LOCKED_OTHER_IMAGE specifier | Exec times out | Exec times out | comp fails |
| intrin_8.5.7f | STOP and LOCK construct with STAT=STAT_UNLOCKED specifier | YES | Exec times out | comp fails |
| item_4.8.a | subobjects if a coarray is also a coarray | YES | YES | YES |
| pointer_4.5.4.6b | association of pointer components of coarrays with local objects | YES | Exec times out | YES |

Table 6: Results of Confidence tests

| SPEC.IDX | DESCRIPTION | OpenUH | Intel | G95 |
|----------------|---|---------------------------|---------------------------|------------|
| atomic_8.5.2 | Atomic subroutines | Passes with 0% confidence | Fails comp | comp fails |
| critical_8.1.5 | CRITICAL - END CRITICAL sections | YES | Passes with 0% confidence | YES |
| intrin_8.5.6 | LOCK & UNLOCK without STAT specifier | YES | Fails comp | comp fails |
| sync_8.5.3 | SYNC ALL without STAT specifier | YES | Passes with 0% confidence | YES |
| sync_8.5.4a | SYNC IMAGES(arr) paired with SYNC IMAGES(*) | YES | Exec fails | YES |
| sync_8.5.4b | call to SYNC IMAGES(arr), should not behave like SYNC ALL | YES | Exec fails | YES |

Table 7: Results of Fault tests

| SPEC.IDX | DESCRIPTION | OpenUH | Intel | G95 |
|-------------|--|--------|------------|------------|
| sync_8.5.7a | STOP and SYNC ALL with STAT=STAT_STOPPED_IMAGE specifier | YES | Exec fails | Exec fails |
| sync_8.5.7b | STOP and SYNC IMAGES(arr) with STAT=STAT_STOPPED_IMAGE specifier | YES | YES | Exec fails |
| sync_8.5.7c | STOP and SYNC IMAGES(*) with STAT=STAT_STOPPED_IMAGE specifier | YES | Exec fails | Exec fails |