2-14-2019

Recovery from Fail-Stop Failures in Parallel Fortran Applications

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

Glenn R. Luecke  
*Iowa State University*, grl@iastate.edu

Pieter Maris  
*Iowa State University*, pmaris@iastate.edu

James P. Vary  
*Iowa State University*, jvary@iastate.edu

Follow this and additional works at: [https://lib.dr.iastate.edu/cs_conf](https://lib.dr.iastate.edu/cs_conf)  
Part of the *Programming Languages and Compilers Commons*

Recommended Citation  
[https://lib.dr.iastate.edu/cs_conf/49](https://lib.dr.iastate.edu/cs_conf/49)

This Poster is brought to you for free and open access by the Computer Science at Iowa State University Digital Repository. It has been accepted for inclusion in Computer Science Conference Presentations, Posters and Proceedings by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.
Recovery from Fail-Stop Failures in Parallel Fortran Applications

Abstract
The Fortran 2018 standard defines syntax and semantics to allow a parallel application to recover from failed images (processes) during execution. This poster presents work to extend the GFortran compiler front end and OpenCoarrays library to support fault tolerant teams of images, enabling use of collective routines after an image failure.

Disciplines
Computer Sciences | Programming Languages and Compilers
Recovery from Fail-Stop Failures in Parallel Fortran Applications

**Summary**

The Fortran 2018 standard defines syntax and semantics to allow a parallel application to recover from failed images (processes) during execution. This poster presents work to extend the GFortran compiler front end and OpenCoarrays library to support fault tolerant teams of images, enabling use of collective routines after an image failure.

**Problem**

- The largest supercomputers today have tens of thousands of compute nodes—and a low Mean Time Between Failure (several hours)
  - Probability of failure unacceptably high for applications that use most/all of the system
  - Checkpoint/restart can be inefficient
- Fortran 2018 standardizes an API that allows an application to detect and recover from image failures, but only a partial implementation exists

**Try It**

Docker image with complete software environment (MPICH + modified GFortran & OpenCoarrays):

```bash
$ alias d='docker run -it -v $PWD:/mnt \
    -w /mnt nathanweeks/opencoarrays:cs-gso-2019'
$ d caf cs-gso-2019.F90
$ d cafrun -np 16 ./a.out

pi (est.):   3.1410373333333337
```

**Example**

Fault-Tolerant Parallel Monte Carlo Pi Calculation

```fortran
... do sample = 1, NSAMPLES
    call random_number(x); call random_number(y)
    if (hypot(x, y) <= 1) n = n + 1
  end do
  if (this_image() == 1) n_copy = n
  do
    form team (1, team, stat=status)
    ! simulate image failure
    fail = size(failed_images()) < NFAIL &
      .and. this_image() == num_images()
    change team (team, stat=status)
    if (fail) fail image
    ! result undefined if image failure during
    ! co_sum(); use copy of n on image 1
    if (this_image() == 1) n = n_copy
    call co_sum(n, result_image=1, stat=status)
  end team (stat=status)
  if (status /= STAT_FAILED_IMAGE) exit
end do
if (this_image() == 1) write(*,*) 'pi (est.):',
    (4.0d0*n/NSAMPLES)/
    (num_images()-size(failed_images()))
```

**Future Work**

- Support teams with failed images containing coarrays (for one-sided gets/puts between images)
- Prototype fault-tolerant sparse matrix eigensolver from application MFNd