Testing HPC C++ software with GoogleTest: adjusting the test framework for distributed-parallel tests using MPI

Melven Röhrig-Zöllner
Johannes Holke
German Aerospace Center (DLR), High-performance Computing
What is Googletest?

- Unit testing library for C++ codes ([https://github.com/google/googletest](https://github.com/google/googletest))

- Offers an easy to use framework to test your code (and more)

- Features include
  - Value-parameterized tests
  - Type-parameterized tests
  - User-defined assertions
  - Many more
What is GoogleTest? Examples

Check if our library correctly computes the norm of the vector (0 0 0)

main.cpp

```c++
#include "gtest/gtest.h"

int main(int argc, char** argv)
{
    ::testing::InitGoogleTest(&argc, argv);
    return RUN_ALL_TESTS();
}
```

test_vec.cpp

```
TEST(t8gtest_vec, normzero) {
    const t8_test_vec zero = {0, 0, 0};
    EXPECT_EQ(t8_vec_norm(zero), 0);
}
```
What is Googletest? Examples

Test passed

```
Running 1 test from 1 test suite.
Global test environment set-up.
1 test from t8_gtest_vec
t8_gtest_vec.normzero
1 test from t8_gtest_vec.normzero (0 ms)
1 test from t8_gtest_vec (0 ms total)

Global test environment tear-down
1 test from 1 test suite ran. (0 ms total)
1 test.
```

Test failed

```
Running 1 test from 1 test suite.
Global test environment set-up.
1 test from t8_gtest_vec
t8_gtest_vec.normzero
1 test from t8_gtest_vec.normzero (0 ms)
1 test from t8_gtest_vec (0 ms total)

Global test environment tear-down
1 test from 1 test suite ran. (0 ms total)
0 tests.
1 FAILED TEST
```
What is GoogleTest? Output of our phist test suite

```
[ RUN ] DSymmMatTestWithUnalignedViews_163_30.read_matrices
[   OK ] DSymmMatTestWithUnalignedViews_163_30.read_matrices (0 ms)
[   OK ] DSymmMatTestWithUnalignedViews_163_30.A1_probe_symmetry (1 ms)
[------------------] 2 tests from DSymmMatTestWithUnalignedViews_163_30 (1 ms total)
[------------------] 2 tests from ZSymmMatTestWithUnalignedViews_163_30

To avoid possible inconsistent ordering between matrices, we force sigma=1 right now, see issue #225.

[ RUN ] ZSymmMatTestWithUnalignedViews_163_30.read_matrices
[   OK ] ZSymmMatTestWithUnalignedViews_163_30.read_matrices (0 ms)
[ GHOST] PE0 INFO at tsmtssnPlain_kernel() <tsmtssnVar2Plain_var_var.cpp:53>: In UNALIGN
[ GHOST] PE1 INFO at tsmtssnPlain_kernel() <tsmtssnVar2Plain_var_var.cpp:53>: In UNALIGN
[ GHOST] PE2 INFO at tsmtssnPlain_kernel() <tsmtssnVar2Plain_var_var.cpp:53>: In UNALIGN
[   OK ] ZSymmMatTestWithUnalignedViews_163_30.A1_probe_symmetry (1 ms)
[------------------] 2 tests from ZSymmMatTestWithUnalignedViews_163_30 (1 ms total)
[------------------] Global test environment tear-down
[==========] 14642 tests from 1096 test cases ran. (4888 ms total)
[        PASSED ] 14642 tests.

YOU HAVE 3562 DISABLED TESTS
```
### What is Googletest? Macro Assertions

<table>
<thead>
<tr>
<th>EXPECT/ASSERT_*</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>EQ (A,B)</td>
<td>A == B</td>
</tr>
<tr>
<td>NEAR (A, B, eps)</td>
<td></td>
</tr>
<tr>
<td>NE (A,B)</td>
<td>A != B</td>
</tr>
<tr>
<td>LT (A,B)</td>
<td>A &lt; B</td>
</tr>
<tr>
<td>LE, GT, GE</td>
<td>&lt;=, &gt;, &gt;=</td>
</tr>
<tr>
<td>TRUE (A)</td>
<td>A == true</td>
</tr>
<tr>
<td>FALSE (A)</td>
<td>A == false</td>
</tr>
</tbody>
</table>
What is MPI

• „Message Passing Interface“ (https://www.mpi-forum.org/)

• API for distributed memory parallelism

• Manages interaction of parallel processes and exchange of „messages“

• Backbone of many HPC libraries

• Multiple different implementations (OpenMPI, mpich, mpi4py, Intel MPI, etc.)
MPI example: hello world

```c
int main (int argc, char* argv[])
{
  int mpiret;
  int mpirank;

  mpiret = MPI_Init (&argc, &argv);
  CHECK_MPI (mpiret);

  mpirank = MPI_Comm_rank (MPI_COMM_WORLD, &mpirank);
  CHECK_MPI (mpiret);

  std::cout << "Hello from " << mpirank << std::endl;

  MPI_Finalize ();

  return 0;
}
```

Each proces has unique „rank“ 0, 1, 2, … NP-1
MPI example: broadcast

Rank 0 sends 1 integer at position “somevalue“ to all processes

```c
int somevalue = 0;
if (mpirank == 0) {
    somevalue = 42;
}

std::cout << mpirank << " has value: " << somevalue << std::endl;
MPI Bcast (&somevalue, 1, MPI_INT, 0, MPI_COMM_WORLD);
std::cout << mpirank << " has value: " << somevalue << std::endl;
```

```
> mpirun -np 4 ./mpi_bcast
1 has value: 0
2 has value: 0
3 has value: 0
0 has value: 42
0 has value: 42
0 has value: 42
0 has value: 42
1 has value: 42
2 has value: 42
3 has value: 42
```
MPI outline

• Point-to-point communication
• Reduction operations
• Gather and Scatter type operations
• etc.

MPI calls can be
• Collective – All processes must call the function.
• Blocking – The program can only continue when the MPI operation is finished.

Possibility of Deadlocks

```c
if (mpirank == 0) {
    MPI_Bcast (&somevalue, 1, MPI_INT, 0, MPI_COMM_WORLD);
}
```

→ Process 0 will wait forever on the other processes!
We want to test our MPI parallelized libraries using GoogleTest!
**Problem 1 – Communication after Assert**

Compute the MPI rank and check if no error occurred.

This will only pass on process 0. All other processes will abort.

Trigger blocking MPI communication

Process 1

```plaintext
int mpiRank;
int mpiErr;
mpiErr = MPI_Comm_rank (MPI_COMM_WORLD, &mpiRank);
ASSERT_EQ (mpiErr, MPI_SUCCESS);
ASSERT_EQ (0, mpiRank);
std::cout << mpiRank << ": starting MPI communication\n";
doMPICommunication ();
std::cout << mpiRank << ": finished MPI communication\n";
```

Process 0

```plaintext
Running 1 test from 1 test suite.
Global test environment set-up.
1 test from MPI_Test
MPI_Test,BlockingCommunicationAfterAssert
MPI_Comm_rank set-up
Expected equality of these values:
0
mpiRank
Value is 0
 MPI_Test,BlockingCommunicationAfterAssert [0 ms]
1 test from MPI_Test [0 ms total]
Global test environment tear-down
1 test from 1 test suite run. (0 ms total)
0 tests:
0 tests, listed below:
MPI_Test,BlockingCommunicationAfterAssert
1 FAILED TEST
```
Solution to problem 1 – synchronized assertions

- Introduce new macros ASSERT_*_MPI
- Synchronize assertion result using MPI communication
- If one process fails the assertion, all will fail the assertion
Solution to problem 1 – implementation details

- GTest uses an “AssertionResult” class:
  - New variable “global”
  - If global → result is synchronized
- Basic idea:
  - “AssertionResult” needs boolean arithmetic (e.g. negate results, check for failure, …)

  → tri-state logic instead of boolean:

<table>
<thead>
<tr>
<th>A</th>
<th>not A</th>
<th>Test result (A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>all true</td>
<td>all false</td>
<td>success</td>
</tr>
<tr>
<td>all false</td>
<td>all true</td>
<td>failure</td>
</tr>
<tr>
<td>mixed</td>
<td>mixed</td>
<td>failure</td>
</tr>
</tbody>
</table>
Problem 2 – Processes do different things

Only happens on process 1 and not on process 0

Test seems to succeed! ("false success")
Solution to problem 2 – synchronize final test result

- Synchronize test result using MPI communication
- If one process fails the test, all will fail
- Implementation similar as for problem 1…

Process 0

```bash
> mpiexec -n 2 ./test1
... Running 1 test from 1 test case.
[ ] Global test environment set-up.
[ ] 1 test from MPI_Test
  [ ] RUN MPI_Test.IndependentProcesses
  [ ] FAILED MPI_Test.IndependentProcesses (1 ms)
  [ ] 1 test from MPI_Test (2 ms total)
[ ] Global test environment tear-down
[ ] 1 test from 1 test case ran. (2 ms total)
[ ] PASSED 0 tests.
[ ] FAILED 1 test, listed below:
[ ] FAILED 1 MPI_Test.IndependentProcesses
```

1 FAILED TEST
Problem 3 – Output

- Usually, the output of all processes is mixed up:

```bash
$ mpirun -np 2 ./test1
Running main() from gtest_main.cc
Running main() from gtest_main.cc
[==========] Running 2 tests from 1 test case.
[==========] Global test environment set-up.
[ RUN ] MPI_Test.IndependentProcesses
[==========] Running 2 tests from 1 test case.
[==========] Global test environment set-up.
[ RUN ] MPI_Test.IndependentProcesses
/home/melven/googletest_examples/test1.cpp:11: Failure
Expected: 42
To be equal to: 0
[ FAILED ] MPI_Test.IndependentProcesses (0 ms)
[ FAILED ] MPI_Test.IndependentProcesses (1 ms)
[ RUN ] MPI_Test.simple_fail
/home/melven/googletest_examples/test1.cpp:16: Failure
Expected: 2
To be equal to: 1
[ FAILED ] MPI_Test.simple_fail
/home/melven/googletest_examples/test1.cpp:16: Failure
Expected: 2
To be equal to: 1
```

```bash
[ FAILED ] MPI_Test.simple_fail (0 ms)
[ FAILED ] MPI_Test.simple_fail (1 ms)
[ FAILED ] 2 tests from MPI_Test (2 ms total)
[==========] Global test environment tear-down
[==========] Running 2 tests from 1 test case ran. (2 ms total)
[ PASHED ] 0 tests.
[ FAILED ] 2 tests, listed below:
[ FAILED ] MPI_Test.IndependentProcesses
[ FAILED ] MPI_Test.simple_fail
2 FAILED TESTS
```

```bash
[ FAILED ] MPI_Test.simple_fail (1 ms total)
[==========] Global test environment tear-down
[==========] Running 2 tests from 1 test case ran. (1 ms total)
[ PASHED ] 0 tests.
[ FAILED ] 2 tests, listed below:
[ FAILED ] MPI_Test.IndependentProcesses
[ FAILED ] MPI_Test.simple_fail
2 FAILED TESTS
```
Solution to problem 3 – only output from process 0

- Per default, only print output on process 0
- Same applies to the XML output
  → Can be overwritten by the user...

```c
if( rank == 0 ) {
    // Configures listeners for default output. Initializing it here
    // allows us to check if this is the MPI root process to prevent
    // duplicate output.
    listeners()->SetDefaultResultPrinter(new PrettyUnitTestResultPrinter);
    // Configures listeners for XML output. This makes it possible for users
    // to shut down the default XML output before invoking RUN_ALL_TESTS.
    ConfigureXmlOutput();
} else {
    listeners()->SetDefaultResultPrinter(NULL);
}
```
Further changes in the background

• Check / call MPI initialization in (MPI_Init / MPI_Finalize)

• Use dedicated MPI communicator
  → unit test communication does not interfere with user communication

• Perform file-IO on the master only (for now) + communicate result:
  • processes can run on different nodes
  • processes on the same node “see” the filesystem differently
    (proc. 0 creates folder, not immediately visible on proc. 1 – OS may only ensures this for one process!)
Not a solution

• There are multiple repos / stackoverflow-answers / etc. that only solve “problem 3” (mixed-up output)

• This is not sufficient!
  • Problem 1 → dead-locks
  • Problem 2 → false success

(we have seen this in bigger software projects in use for years!)

→ You cannot test parallel software without thinking about the constraints / implications of the parallelization!
Difficulties & constraints of parallel testing with MPI

• MPI built for high-performance…

• No detection of communication errors (dead-locks, mismatching messages / etc.) → need to run unit tests with a tool, e.g. must: https://itc.rwth-aachen.de/must/

• No death tests

• User must decide where to use local vs. global assertions! (ASSERT_EQ vs. ASSERT_EQ_MPI)
Conclusion

• Please use and cite our MPI GoogleTest version: https://github.com/DLR-SC/googletest_mpi

• It’s been in use in our institute since 2014 in several software projects:
  • Sparse linear algebra (phist, spliss)
  • Helicopter simulation (VAST)
  • CFD mesh management (FSDM, t8code)
  • Multi-linear/tensor algebra (pitts)

• What could be done to improve googletest_mpi?
  • Show detailed error message from first failing process
  • Show ranks of failing processes / how many ranks failed
  • Add “_MPI” variant for all EXPECT_/ ASSERT_ statements