Portability problem with libc

Tom Favereau

March 10, 2025

Abstract

This paper addresses a portability issue in the `NaW` software, observed between Debian 10 and Ubuntu 22.04. Discrepancies arise due to floating-point precision differences, affecting results at the 17th decimal place. While such precision is unrealistic in physical simulations, it raises concerns about the robustness of the code. We explore potential causes, including compiler optimizations and library dependencies, and propose solutions.

Contents

1 Introduction

This work originates from observations made during the use of a software tool designed to model the sodium-water reaction, which we will refer to as `NaW`. During the simulations, portability issues were identified between different Linux distributions, leading to discrepancies in the results. This specific issue, initially encountered in the context of the `NaW` software, has revealed a more general phenomenon regarding the portability of scientific software across different systems.

In the development and use of scientific software, a recurring issue is portability, which refers to the ability of software to produce consistent and reproducible results across different platforms. In particular, differences between operating systems, library versions, or compilation options can have a significant impact on the accuracy and stability of results. This work examines the case of `NaW` as a starting point, but extends the analysis to discuss the broader implications for software portability in computational environments.

This document explores the possible causes of these discrepancies, discusses the proposed solutions to address the problem, and presents a general approach to improve software portability in complex computational environments.

2 Problem Overview

A software tool is facing portability issues across different Linux distributions, leading to discrepancies in results depending on the system used. For instance, variations were observed between two different versions of operating systems, which became a priority issue due to a system migration. Moreover, the results obtained on one of the platforms appeared more consistent, particularly regarding the shock wave amplitude.

These discrepancies were identified by examining sensor readings generated by the software. The calculated pressures sometimes differed by a factor of 10. A line-by-line comparison of the output files revealed that the results were identical up to a certain iteration, after which discrepancies appeared, often in the last decimal places. The divergence then grew, likely due to a butterfly effect. It was noted that the iteration at which results began to diverge varied based on test parameters, initial conditions, meshing, and compilation options. Interestingly, compilation optimization seemed to delay the onset of this divergence.

Remark. The comparison of output files was performed using the `xxdiff` tool. This tool encounters problems when processing large files. However, typically, only the end of these files is of interest. Therefore, they should be shortened using the `tail` command.

The existence of a limiting iteration leads us to consider two hypotheses:

1. A problem occurs during initialization and propagates progressively through the mesh.
2. A calculation is sensitive to the values of variables, leading to different results depending on whether Ubuntu or Debian is used.

The fact that the limiting iteration differs for sensors 1 and 2 leads us to favor hypothesis 1, where the problem propagates gradually through the mesh.

3 Testing Protocol

To eliminate potential issues related to different optimizations and the use of the MPI library, the tests will be performed with the `-O0` optimization level and

on a single processor. The exact compilation options are as follows:

```
-cpp -msse -msse2 -O0 -finit-local-zero -DPARALLEL=0
-DPETSC_LIB=0 -DCANTERA_LIB=0 -DPREC=1 -DCOMPIL=1
```

Additionally, to avoid problems related to compiler versions, the tests will be carried out with `gfortran` version 12.4.1. To prevent any potential issues caused by the default compilation options of `gfortran`, it has been manually compiled using the following command:

```
../gcc-12.4.0/configure --prefix=/home/tf279988/gcc-12.4.1
--enable-languages=c,c++,fortran --disable-multilib
--with-gmp=/home/tf279988/gcc-libs
--with-mpfr=/home/tf279988/gcc-libs
--with-mpc=/home/tf279988/gcc-libs
```

4 Problem Investigation

An initial analysis highlighted a behavioral difference between Ubuntu and Debian. This difference was localized in the `Overbee` function[1]. Specifically, the call `Overbee(0.0_DP)` returns `0.0_DP` on Debian and `-0.0_DP` on Ubuntu. Unfortunately, we were unable to reproduce this issue outside the context of `NaW`.

This function is only called when the second-order spatial scheme is enabled. We decided to perform comparisons using the first-order scheme in both time and space to simplify the analysis. Therefore, all subsequent tests will be carried out with the first-order scheme in time (Euler method) and space.

A second analysis, using these modified tests on a 6×6 mesh, was performed on the advection and `fallingDrop` test cases. The results were identical on both Ubuntu and Debian. Therefore, we decided to focus our tests on the `Vipere` test case (geometry and physical conditions for which `NaW` was developed), which involves chemical processes that we suspect may be responsible for the observed behavioral differences.

Finally, a last analysis revealed a calculation with different behavior on Ubuntu and Debian. This observation confirms hypothesis 2, which suggests that there are computational differences depending on the operating system.

The calculation in question occurs within the `TSAT_WAT` function[2]. This function is called by `RELAX_POT_CHIMIQUE`[3].

We successfully isolated the problem, and it is reproducible. To aid in understanding and reproducing this anomaly, we present below a code excerpt that allows for the reproduction of this behavior.

Remark. The code is presented here in C, as this language is more commonly used than Fortran. However, the Fortran translation produces the same results.

[1] `code/GRADIENTS/include/GRADIENTS_FUNCTION_Overbee.inc.f90`
[2] `code/models/modelNa/EOS/include/EOS_FUNCTION_TSAT_WAT.inc.f90`
[3] `code/models/modelNa/sourceM/include/SourceM_SUBROUTINE_RELAX_POT_CHIMIQUE_RSE.inc.f90`

```c
#include <stdio.h>
#include <math.h>

#define DP double

DP f(DP p);

int main() {
    DP a = 0.26181692836891457;
    DP b = 100003.33299724340;
    DP c;

     c = f(a * b);

    printf("%.15lf\n", c);
    printf("%.15lf\n", 4900.0 / (12.98 - log((a * b) / 1.0e5)));

    return 0;
}

DP f(DP p) {
    DP res;

    res = 4900.0 / (12.98 - log(p / 1.0e5));

    return res;
}

```

Figure 1: C++ code to reproduce the portability issue

Remark. To reproduce this behavioral difference, you can use either a physical machine, a virtual machine, or even a Docker container (note that this will not work in a lighter Singularity container). To compile the code, use the command: `g++ file.c` with GCC 12.4.0. We also recommend compiling with `g++ -S file.c` to examine the generated assembly code.

The analysis of this issue led us to the following conclusions:

- Inlining the function call (line 16) resolves the issue in this case. However, for values $a = 0.13893061007554738$ and $b = 107672.72827138640$, the problem reappears.
- Compiling with `-O1` resolves the issue in this case, but other values for which the problem persists have been identified, although we did not record them.
- This behavioral difference is not directly related to Ubuntu and Debian, as the problem also occurs between Debian 10 and Debian 12.
- We suspect that the `glibc` library is responsible for the issue. Indeed, it is version 2.28 on Debian 10 and 2.35 on Ubuntu 22.04. However, nothing has allowed us to confirm this. Additionally, Alpine 3.20, which uses `musl`c 1.2.4, behaves like Ubuntu, despite not using `libc`.

5 Proposed Solutions

Using Docker: The first solution we propose is to use Docker to containerize `NaW`. Indeed, using a container will allow us to maintain the software in a version that works without worrying about future updates. However, using Docker requires sudo privileges. Therefore, we recommend installing Docker and granting sudo access only for the `docker start` command and the command to run the container in interactive mode. Providing full Docker access is neither justified nor recommended, as it could lead to security issues.

Static Compilation on a Debian Machine: Since the issue is related to dynamic libraries, one solution is to compile the software statically on Debian 10. The generated binaries will then be statically linked to the correct libraries. However, this solution has the disadvantage of requiring a different machine for compilation, meaning that Debian 10 must be kept. Additionally, a static version of Open MPI will need to be installed on this Debian machine.

Separation of Logarithms: It seems that using the logarithmic property `log(a*b) = log(a) + log(b)` can correct the behavioral difference in all the cases studied so far. Therefore, it is possible that modifying this throughout the code could resolve the issue. This work has already been partially attempted,

but it seems insufficient for the `TSAT_NA` function[4]. While the solution is not fully refined, it might still be worth continuing to explore this direction.

6 Criticism

It seems, however, that a difference appearing at the 17th decimal place should not have a significant impact on the overall results of the code. Indeed, in the context of simulating physical or chemical reactions, it is unrealistic to know the initial state with a precision of 17 decimal places. This extreme precision far exceeds the reliability of experimental data or the available initial conditions. Therefore, if the software's results depend on such minute variations, it calls into question their robustness and relevance.

The dispersive nature observed in the results of the code could indicate an instability in the numerical method used for the solution. Such an instability might amplify small errors or disturbances, leading to significant deviations in the predictions. This potential issue is a priority for investigation, as it could affect the validity of simulations, particularly for complex scenarios or over long time scales.

7 Conclusion

In this work, we addressed a portability issue in the `NaW` software, highlighting the discrepancies between Debian 10 and Ubuntu 22.04 due to floating-point precision. Although these differences appear minor, they raise important concerns about the robustness and stability of numerical methods used in the software. We proposed several solutions, such as Docker containerization and static compilation, to mitigate the problem.

However, open questions remain regarding the root cause of these discrepancies. What exactly is causing the behavior observed at the 17th decimal place? Is it related solely to the precision of floating-point calculations, or does it reflect deeper issues in the software's numerical methods or compiler optimizations? Further investigation is needed to fully understand and address these challenges, particularly in the context of complex simulations and long-term predictions.

[4] ./src/code/models/modelNa/EOS/include/EOS_FUNCTION_TSAT_NA.inc.f90