Low-precision climate computing:
Nobody needed those bits anyway

Milan Klöwer
Atmospheric, Oceanic and Planetary Physics
University of Oxford

supervised by

Prof. Tim Palmer, University of Oxford
Dr. Peter Düben, European Centre for Medium-Range Weather Forecasts

A thesis submitted for the degree Doctor of Philosophy

Oxford, September 2021

DON’T PANIC

Abstract

Who needs those bits anyway?

Contents

1 Introduction

2 Methods

3 Information-preserving compression

4 Beyond floats and posits

5 Running on 16-bit hardware

6 Conclusions

Appendix

A.1 Open-source software developments

SoftPosit.jl

- Authors: M Kloewer, M Giordano, C Leong
- URL: github.com/milankl/SoftPosit.jl
- License: MIT
- Version: 0.3.0

SoftPosit.jl is a software emulator for posit arithmetic. The package exports the Posit8, Posit16, Posit32 number types among other non-standard types, as well as arithmetic operations, conversions and additional functionality. The package is a wrapper for the SoftPosit C-library written by C Leong.

StochasticRounding.jl

- Authors: M Kloewer
- URL: github.com/milankl/StochasticRounding.jl
- License: MIT
- Version: 0.1.0

StochasticRounding.jl is a software emulator for stochastic rounding in the Float32, Float16 and BFloat16 number formats. Both 16bit implementations rely on conversion to and from Float32 and stochastic rounding is only applied for arithmetic operations in the conversion back to 16bit. Float32 with stochastic rounding uses Float64 internally. Xoroshio128Plus is used as a high-performance random number generator.

ShallowWaters.jl

- Authors: M Kloewer
- URL: github.com/milankl/ShallowWaters.jl
- License: MIT
- Version: 0.3.0

ShallowWaters.jl is a shallow water model with a focus on type-flexibility and 16bit number formats, which allows for integration of the shallow water equations with arbitrary number formats as long as arithmetics and conversions are implemented. ShallowWaters also allows for mixed-precision and reduced precision communication.

ShallowWaters is fully-explicit with an energy and enstrophy conserving advection scheme and a Smagorinsky-like biharmonic diffusion operator. Tracer advection is implemented with a semi-Lagrangian advection scheme. Runge-Kutta 4th-order is used for pressure, advective and Coriolis terms and the continuity equation. Semi-implicit time stepping for diffusion and bottom friction. Boundary conditions are either periodic (only in x direction) or non-periodic super-slip, free-slip, partial-slip, or no-slip. Output via NetCDF.

Sherlogs.jl

- Authors: M Kloewer
- URL: github.com/milankl/Sherlogs.jl
- License: MIT
- Version: 0.1.0

Sherlogs.jl provides a number format Sherlog64 that behaves like Float64, but inspects your code by logging all arithmetic results into a 16bit bitpattern histogram during calculation. Sherlogs can be used to identify the largest or smallest number occuring in your functions, and where algorithmic bottlenecks are that limit the ability for your functions to run in low precision. A 32bit version is provided as Sherlog32, which behaves like Float32. A 16bit version is provided as Sherlog16T, which uses T for computations as well as for logging.

Sonums.jl

- Authors: M Kloewer
- URL: github.com/milankl/Sonums.jl
- License: MIT
- Version: 0.2.0

Sonums.jl is a software emulator for Sonums - the Self-Organizing NUMbers. A number format that learns from data. Sonum8 is the 8bit version, Sonum16 for 16bit computations. The package exports number types, conversions and arithmetics. Sonums conversions are based on binary tree search, and arithmetics are based on table lookups. Training can be done via maximum entropy or minimising the rounding error.

Float8s.jl

- Authors: M Kloewer, J Sarnoff
- URL: github.com/milankl/Float8s.jl
- License: MIT
- Version: 0.1.0

Float8s.jl is a software emulator for a 8bit floating-point format, with 3 exponent and 4 significant bits. The package provides the `Float8` number type, as well as arithmetic operations, conversions and additional functionality. The software emulator is based on conversion to and from Float32, which is used for arithmetic operations.

LogFixPoint16s.jl

- Authors: M Kloewer
- URL: github.com/milankl/LogFixPoint16s.jl
- License: MIT
- Version: 0.1.0

LogFixPoint16s.jl is a software emulator for 16-bit logarithmic fixed-point numbers with 7 signed integer bits and 8 fraction bits. The package provides the `LogFixPoint16` number type, as well as arithmetic operations, conversions and additional functionality. The software emulator is based on either integer addition or look-up tables and is therefore a comparably fast emulator.

Lorenz96.jl

- Authors: M Kloewer
- URL: github.com/milankl/Lorenz96.jl
- License: MIT
- Version: 0.3.0

Lorenz96.jl is a type-flexible one-level Lorenz 1996 model, which supports any number type, as long as conversions to and from Float64 and arithmetics are defined. Different number types can be defined for prognostic variables and calculations on the right-hand side, with automatic conversion on every time step. The equations are scaled such that the dynamic range of numbers can be changed. The scaled equations are written division-free.

Lorenz63.jl

- Authors: M Kloewer
- URL: github.com/milankl/Lorenz63.jl
- License: MIT
- Version: 0.2.0

Lorenz63.jl is a type-flexible Lorenz 1963 model, which supports any number type, as long as conversions to and from Float64 and arithmetics are defined. The Lorenz equations are scaled such that the dynamic range of numbers can be changed. The scaled equations are written division-free.

Jenks.jl

- Authors: M Kloewer
- URL: github.com/milankl/Jenks.jl
- License: MIT
- Version: 0.1.0

Jenks.jl is the Jenks Natural Breaks Optimization, a data clustering method to minimise in-class variance or L1 rounding error. Jenks provides a data classification algorithm that groups one dimensional data to minimize an in-class error norm from the class mean but maximizes the same error norm between different classes.

Acknowledgements

I am very grateful for the support and very fruitful discussions with my supervisors Tim Palmer and Peter Düben, and especially for the freedom to develop my own ideas.

I would like to thank the whole Julia community for an uncountable effort to develop a very modern high performance computing language that is high-level, easy to learn and was proven to be increadibly useful for reduced precision simulations. I also would like to thank everybody who developed the matplotlib plotting library, which was used for every figure in this report.

Funding

I gratefully acknowledge funding from the European Research Council under grant number 741112 *An Information Theoretic Approach to Improving the Reliability of Weather and Climate Simulations* and from the UK National Environmental Researc Council (NERC) under grant number NE/L002612/1.

I would like to thank the whole Julia community for an uncountable effort to develop a very modern high performance computing language that is high-level, easy to learn and was proven to be increadibly useful for reduced precision simulations. I also would like to thank everybody who developed the matplotlib plotting library, which was used for every figure in this report.