Porting and Benchmarking A Cloud Microphysics Parameterisation Scheme (CloudSC) To The A64FX Processor

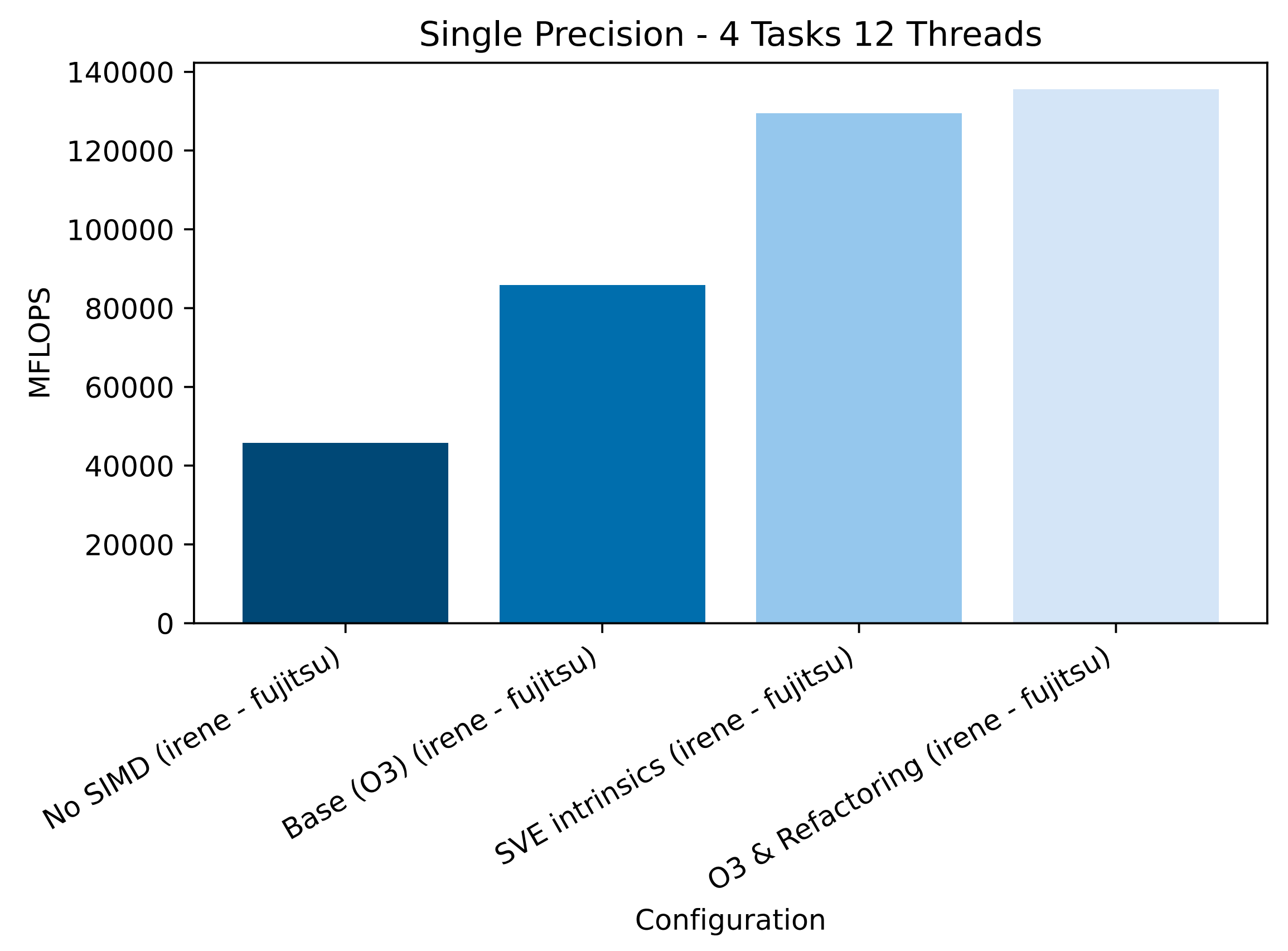
Andrew Beggs[1], Olivier Marsden[1], Ioan Hadade[1]

(1) European Centre for Medium-Range Weather Forecasts (ECMWF); Andrew.Beggs@ecmwf.int, Olivier.Marsden@ecmwf.int, Ioan.Hadade@ecmwf.int

1. Introduction

Old is new again and vectorisation has become a key pathway to increase the performance of code. To feed this increase in compute performance, **High Bandwidth Memory** has also become increasingly popular. This poster looks at the effects and strategies to utilise both Arm's **Scalable Vector Extension** (**SVE**) and **HBM** (available on **Fujitsu's A64FX processor**) on **CloudSC** - a physics component of the Integrated Forecasting System (IFS) known for being **computationally demanding**. These results are to be used for readying applications for the EPI's Rhea processor, while retaining maximum performance and minimum reliance on compiler auto-vectorisation.

2. Effects of vectorisation on performance

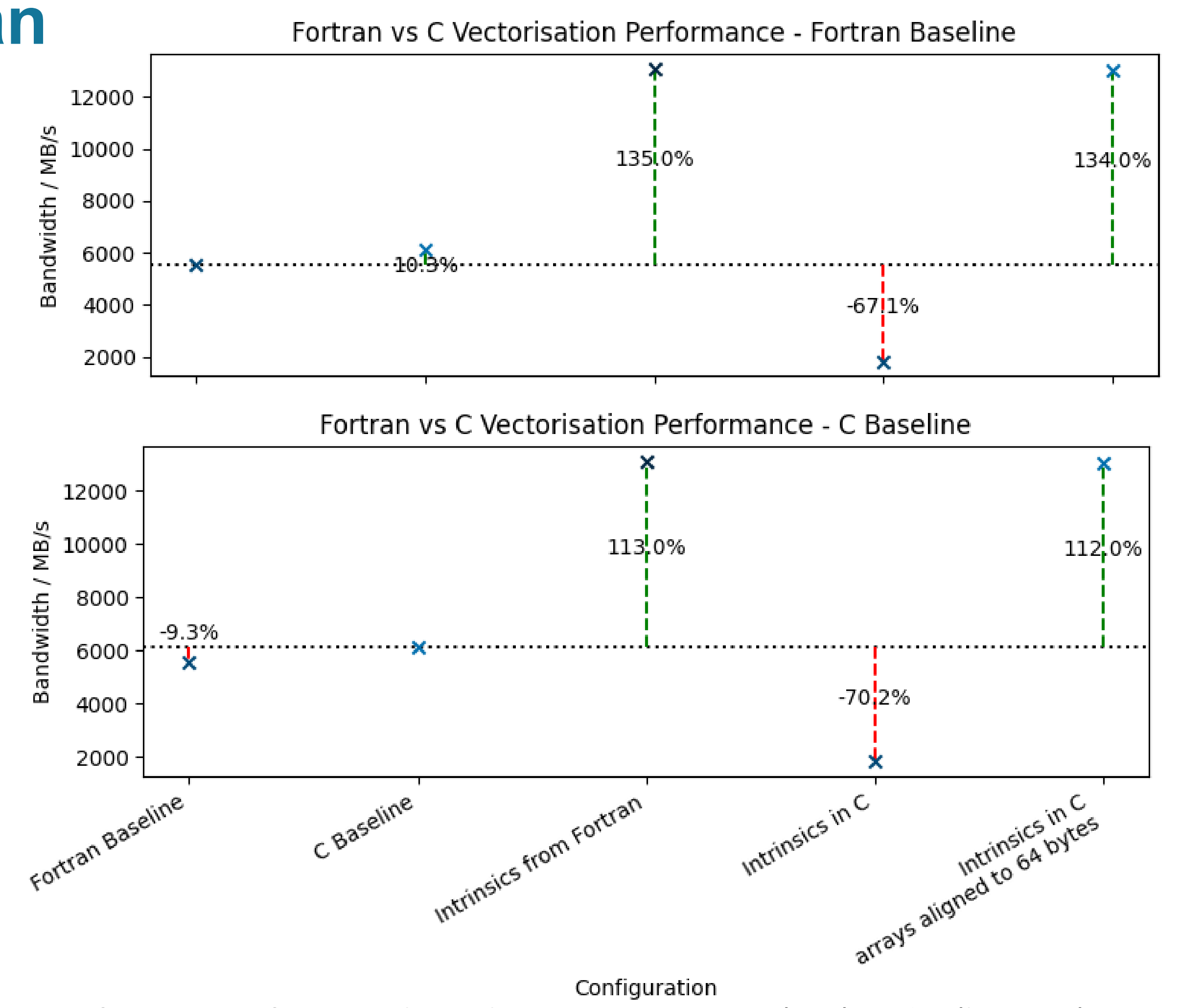


Performance of CloudSC without vectorisation, with auto-vectorisation, a handwritten SVE kernel, & auto-vectorised refactored kernel

- **Auto-vectorisation** is able to **increase performance** by 87.6%
  - A good result for only changing flags
- Refactoring the hottest loop yields an increase of 57.9% over base performance
  - The change was minimal, <10 lines
  - This is consistent with other refactorings
- A hand written **SVE intrinsics** kernel written in C and called from Fortran **achieves 95.6% of performance of the auto-vectorised** refactored loop
  - Minimal optimisations used
  - Potential for greater performance than the compiler
- This is only **5.87% of maximum theoretical FLOPS**
  - Gains will have diminishing returns going from hottest to coldest
  - How much can be gained from vectorisation alone?

3. Using C SVE intrinsics in Fortran

- The IFS, and by extension **CloudSC, is written** almost entirely **in Fortran**
  - Not feasible to translate the whole codebase
  - **SVE intrinsics are only available in C**
- The **solution** is to identify hot sections of the codebase and hand **write SVE** kernels for these **in C**, which can be **called from Fortran**
- Conditional compilation allows for one generic codebase for seamless deployment
- In these simple examples it was possible to beat the auto-vectorised code
- Care must be taken though as Fortran aligned to 64 bytes, while C aligned to 16 by default
  - Only aligning to 64 bytes would give a performance increase over the baseline, while aligning to 16 bytes caused a significant slowdown



Performance of various kernels against a Fortran (top) and C (bottom) auto-vectorisation baseline

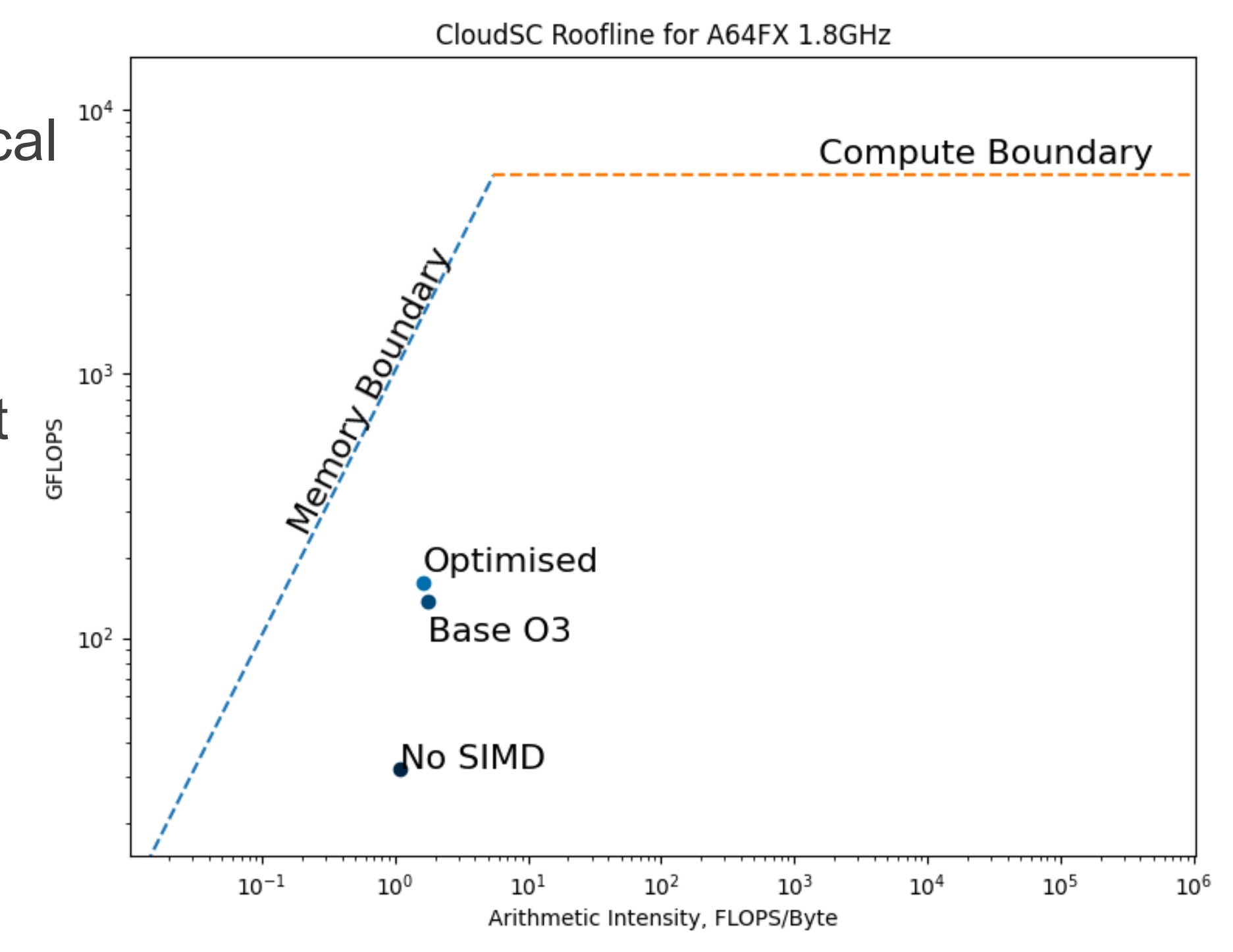
4. Impact of memory bandwidth on CloudSC

CloudSC is known to be compute intensive and often **not bound by memory bandwidth**. The following procedure provides empirical **proof** of this

- Performance is modelled with a naïve equation: $T_1 = T_0\alpha(f_0/f_1) + T_0\beta(BW_0/BW_1)$
  - T_0 & T_1: Are some performance metric from two separate runs
  - α: Compute dependency (**α = 1, purely compute bound**)
  - f_0 & f_1: The CPU frequencies used on the two runs
  - β: Memory bandwidth dependency (**β = 1, purely memory bound**)
  - BW_0 & BW_1: The memory bandwidths used on the two runs
- Runs were taken **varying only the CPU frequency**
- α and β can be easily calculated with a linear regression, these were **α = 0.651 & β = 0.343 for CloudSC**
- Not memory bound at practical problem sizes

5. Work for the future

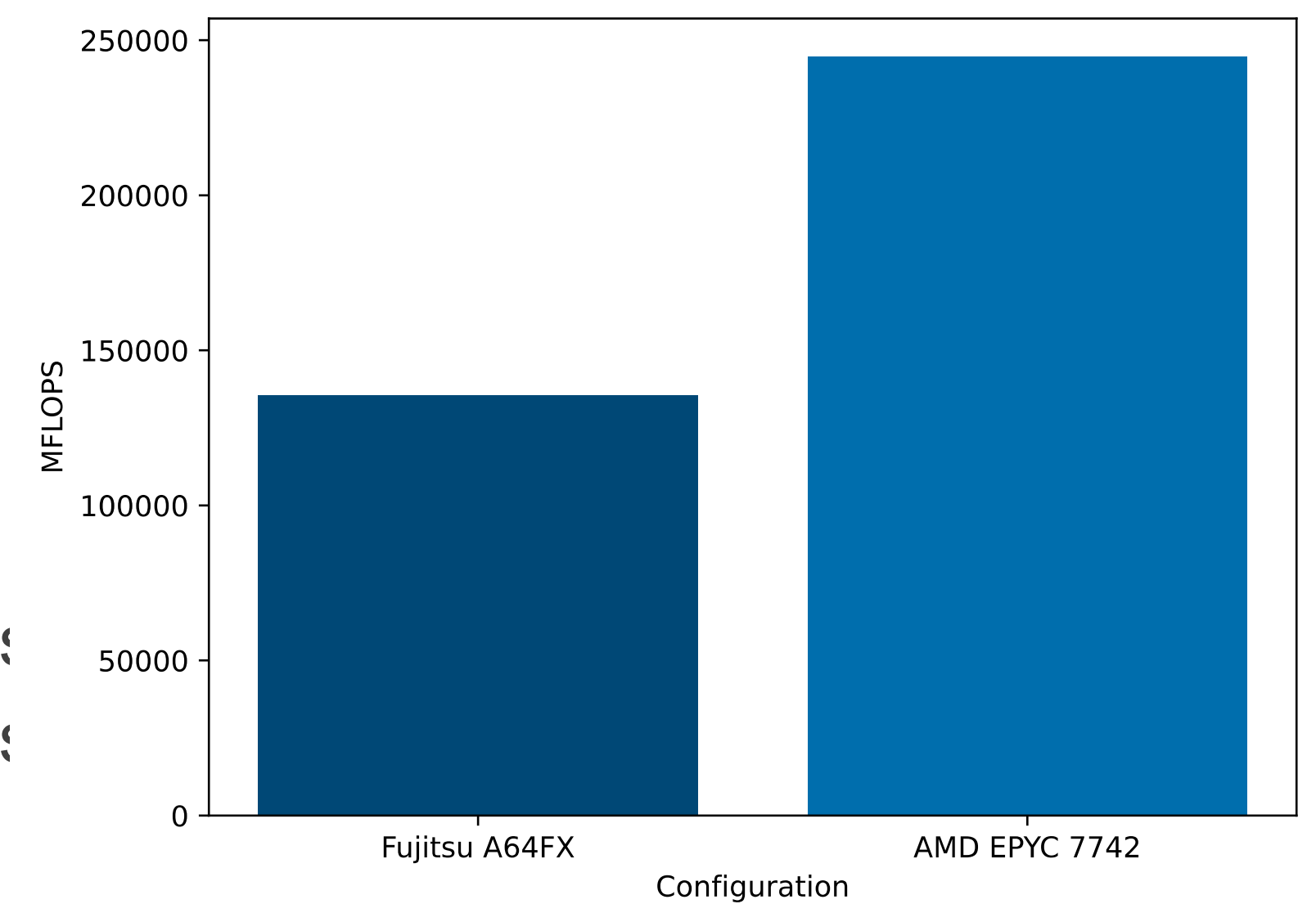
- Better overall performance
  - Currently CloudSC is only achieving ~6% of maximum theoretical FLOPS
- Use Loki to **automate** the process of **writing SVE kernels**
  - A **source to source** translation tool
  - Kernel performance can be automatically benchmarked against auto-vectorised code
  - Highlights edge cases for human intervention and tool improvement
- Investigate more components of the Integrated Forecasting System
  - **CloudSC is only one of many components**
  - Each have their own performance characteristics and bounds
- Profile performance on the Rhea processor
  - Does the code work as expected on the new platform (portability and performance)?
  - We don't know, the processor hasn't been made yet!



Roofline graph for the A64FX processor of CloudSC with various optimisations applied. Theoretical bounds used

Summary

- CloudSC is unaffected by High Bandwidth Memory as it is heavily compute bound
  - This can be determined with a very simple test and no advanced knowledge of the platform
  - Effort should be put into optimising with vectorisation
- There are a lot of gains to be made with vectorisation
  - Even with only a few lines changed
  - Compiler auto-vectorisation can only vectorise what it's given – not refactor entire algorithms
- Code bases written in "legacy languages" can still avail of cutting-edge language specific features without performance penalty
  - Care must be taken for hidden differences between language specifications
- Source to source translation could be an ideal way to quickly increase performance while compilers achieve maturity



CloudSC on a Fujitsu A64FX vs ECMWF's current operational AMD EPYC 7742

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 101033975. The JU receives support from the European Union's Horizon 2020 research and innovation programme and France, Germany, Italy, Greece, United Kingdom, Czech Republic, Croatia