Managing Performance vs. Accuracy Trade-offs with an Improved Bit-Level Precision Tuning

Dorra Ben Khalifa[1] and Matthieu Martel[1,2]

[1] University of Perpignan, LAMPS laboratory, 52 Av. P. Alduy, Perpignan, France
[2] Numalis, Cap Omega, Rond-point Benjamin Franklin, Montpellier, France
{dorra.ben-khalifa, matthieu.martel}@univ-perp.fr

keywords: Static analysis, code optimization, numerical accuracy, computer arithmetic, integer linear problem.

Although users of High Performance Computing (HPC) are most interested in raw performance, both storage costs and power consumption have become critical concerns. This is due to several technological issues such as the power limitations of processors and the massive cost of communications which arise while executing applications on such architectures [**?**]. In recent years, the use of reduced precision to improve the performance metrics is emerging as a new trend to save the resources on the available processors. In practical terms, as almost the numerical computations are performed using floating-point operations to represent real numbers [**?**], the precision of the related data types should be adapted in order to guarantee a desired overall rounding error and to strengthen the performance of programs. For instance, using single precision formats (`FP32`) is often at least twice as fast as the double precision (`FP64`) ones. Consequently, the natural question that arises is how to obtain the best precision/performance trade-off by allocating some program variables in low precision (e.g. `FP16` and `FP32`) and by using high precision (e.g. `FP64` and `FP128`) selectively. This process is also called by mixed-precision tuning.

Past research main goal was to improve performance by reducing precision with respect to an accuracy constraint done by static analysis [**?**, **?**] or by dynamic analysis [**?**, **?**, **?**]. The major limitation of theses techniques is that they follow a try and fail methodology: they change the data types of some variables of the program and evaluate the accuracy of the result and depending on what is obtained they change more or less data types and repeat the process.

In this article, we present a novel static approach based on a semantical modeling of the propagation of the numerical errors throughout the code. This technique is embodied into an automated tool called `POP`. The main insight of `POP`, in contrast to its former introduction in [**?**, **?**, **?**], is to solve an Integer Linear Problem (ILP) generated from the program source code. This is done by reasoning on the most significant bit and the number of significant bits of the values which are integer quantities. The optimal solution computed by a classical linear programming solver gives the optimized data types that satisfy the user accuracy requirement in a polynomial time. The originality of `POP`, in comparison with the existing tools, is that our approach finds the minimal number of bits needed for each variable, known as bit-level precision tuning to get a certain accuracy on the results. Consequently, our tuning is not dependant of any particular computer arithmetic. The main contribution of this article is to point out the results of precision tuning with `POP` in terms of bit-level optimiza-

tion and analysis time on new benchmarks[1] coming from different application domains such as embedded systems, Internet of Things (IoT), etc. Also, we provide a detailed comparison of `POP` and the state of the art.

[1] https://fpbench.org/