LITERATURE REVIEW: Accurately computing large floating-point numbers using parallel computing

Tanvir Kaykobad
School of Computer Science
Carleton University
Ottawa, Canada K1S 5B6
TanvirKaykobad@cmail.carleton.ca

October 15, 2017

1 Introduction

There are SIMD algorithms for accurately summing up a very large number of small and large floating point numbers in parallel. It is known that for accurately summing up a very large number of small and large floating point numbers one has to use snowball effect to obtain larger numbers from small ones. I would like to pursue this idea and extend it in parallel computing. I would like to introduce a huffman-tree like structure to pair floating point numbers of similar magnitude so they can be summed up more accurately, ensuring minimum rounding error in the worst case while compromising as little as possible in non-parallel computation time.

2 Literature Review

Demmel and Nguyen [2] used Rumps algorithm for floating point summation that is reproducible independent of the order of summation that may be different with the dynamic scheduling of parallel computing resources and floating point nonassociativity. Their absolute error bound is 2-28 times macheps, and requires constant amount of extra memory usage.Demmel and Hida [1] analysed several algorithms and showed that if a wider accumulator of F bits is used to sum n floating point numbers each of at most f bits, and if sum is carried out in descending order of exponents then an error of at most 1.5 times the least significant bit can occur provided that number of summands does not exceed $2^{(F-f)}$. Rump, Ogita and Oishi [6] showed that their algorithm results in a value nearest to the true sum. In 2013 the authors [3] have shown how to use tree reduced parallelism to compute sum by using parallel associative reduction, iterative refinement and conservative early detection to obtain an algorithm of order log n. Neal [5] presented two algorithms in one of which a small superaccumulator with 67 64-bit chunks each with 32-bit overlap with the next chunk was used to allow carry propagation to be done infrequently. Kai and wang [4] have shown that computing ensuring minimum error when n numbers can be both positive and negative is NP-hard. However their algorithm can sum with no more than $2\lceil \log{(n-1)} + 1 \rceil * \epsilon$, where ϵ is the worst case minimum error over all possible orders.

References

[1] J. Demmel and Y. Hida. Accurate and efficient floating point summation. *SIAM Journal on Scienti*

c Computing, 25(4):1214–1248, 2004.

[2] J. Demmel and H. D. Nguyen. Parallel reproducible summation. *IEEE TC*, 64(7):2060–2070, July 2015.

[3] E. Kadric, P. Gurniak, and A. DeHon. Accurate parallel floating-point accumulation. *21st IEEE Symp. on Computer Arithmetic (ARITH)*, 31(1):153–162, April 2013.

[4] M.-Y. Kao and J. Wang. Linear-time approximation algorithms for computing numerical summation with provably small errors. *SISC*, 29(5):1568–1576, 2000.

[5] R. M. Neal. Fast exact summation using small and large superaccumulators. *arXiv ePrint*, abs/1505.05571:153–162, 2015.

[6] S. M. Rump, T. Ogita, and S. Oishi. Accurate floating-point summation part i: Faithful rounding. *SIAM Journal on Scienti*

c Computing, 31(1):189–224, 2008.