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

December 7, 2017

1 Introduction

There are SIMD algorithms for accurately summing up very large number of small and large floating point numbers in parallel. It is known that for accurately summing up such a set of numbers one has to use snowball effect to obtain larger numbers from smaller ones. I would like to pursue this idea and extend it in parallel computing by introducing 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 [?] 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 [?] 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 [?] showed that their algorithm results in a value nearest to the true sum. In 2013 the authors [?] 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 [?] 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 [?] 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. Goodrich and el [?] presents an algorithm named MapReduce that according to their experimental evaluation achieves up to 80X performance speedup as compared to the state-of-the-art sequential algorithm.

The algorithm yields lineary scalability with both the input dataset and number of cores in the cluster. H. Leuprecht and W. Oberaigner [**?**] proposed a parallel algorithm, a pipeline version of Pichat and Bohlender algorithm, where the sum is associated with a tree. They also discuss the properties a multiprocessor architecture should have for efficient implementation of the algorithm. Malcolm [**?**] divided each of the n t-digit numbers, forming qn t-digit floating point numbers is then added to one of several auxiliary t-digit accumulators. Finally, the accumulators are added together to get the computed sum.