DATE : 2016/01/08

Department of Electric and Electronic Information Engineering		ID	M123209
Name	Yuya Itagaki		

Supervisor	Shuichi Ichikawa Naoki Fujieda

Abstract

Title	Acceleration of Multiple-precision Integer Multiplication of GMP Library

In the current numerical calculation, IEEE754 floating-point number is widely used because it can be calculated by CPU hardware. Multiple-precision calculation is used for the calculation that requires more precision such as calculation of π. Multiple-precision operation, which has to be performed by software, is slower than IEEE754. GMP Library provided by GNU is popular for its performance and availability. Multiplication and division are typically slower than the addition and subtraction. Since a recent CPU consists of two or more cores (multi-core CPU), applications might be accelerated by utilizing parallelism among threads. Therefore, this study examines acceleration of multiple-precision integer multiplication of GMP with OpenMP.

This study deals with the multiple-precision calculation of variable length. Karatsuba method is often used in multiple-precision multiplication algorithm. Compared to the naive algorithm, the number of multiplications is reduced to three-quarters.

The *mpz_mul* function performs the multiplication of multiple-precision integers of variable length, which are expressed as the *mpz_t* type, in GMP. The mpz_t type has a pointer to an array of 64-bit integers, which are reallocated if necessary, and the size of the array. Here, the size of the array is simply denoted by *size*. In mpz_mul, calculation methods are different according to *size*. In this paper, two cases are considered; case 1: multiplier's *size* = 1, case 2: multiplicand's *size* = multiplier's *size*.

In the proposed approach, the parallelization techniques are applied to the both case. In case 1, the multiplicands are first divided and the divided multiplications are performed in parallel. Degree of parallelism was 2, 4, and 8 for this case. In case 2, Karatsuba method is adopted to make three multiplications be performed in parallel. Parallelism at this time was set to 3 and 9.

This study evaluated sample programs that repeat multiplications of case 1 and 2. The program for case 1 performs the multiplication of n decimal digits × 64 bit (*size* = 1). The program for case 2 multiplies n decimal digits × n decimal digits. The operations are repeated until the execution time exceeds one second. The operations per second (OPS) is calculated by dividing the number of executions by the execution time. The median value of three measurements is adopted as the result. According to the evaluation result, the parallel version of case 1 with the degree of parallelism 8 showed an acceleration of 7.727 times compared to the original library when $n = 3 \times 10^7$. The parallel version of case 2 with the degree 3 was 1.699 times faster than the original library when $n = 5 \times 10^5$.

As an application of case 1, a program that calculated π using Machin's formula was examined. In this evaluation, the speedup of the proposed method was not observed in the range of measurement, as the execution time was 1.1-2.0 times longer than with the original library. In future studies, it is necessary to examine more methods for parallelization in mpz_mul function.