Application of Redundant Positional Notations for Increasing Arithmetic Algorithm Scalability

Anatoly V. Panyukov

Chair of Economical and Mathematical Methods and Statistics,
National Research University SUSU
Chelyabinsk, Russian Federation
a_panyukov@mail.ru

XV GAMM-IMACS International Symposium on Scientific Computing, Computer Arithmetics and Verified Numerics
2012, September 23 –29, Novosibirsk, Russia

Application of Redundant Positional Notations for Increasing Arithmetic Algorithm Salabilities

Content

- Guaranteed and Perfect Accuracy Computing
- Big Numbers Addition Algorithm
- Way of Increasing Scalability

Guaranteed Accuracy Computing GNU Multiple Precision Arithmetic Library: http://gmplib.org/, 2010

Type Data mpf_t

- mpf_t (multiple-precision floating point) It Is Included gnu/mp
- gnu/mp Is of Linux Distribution Kit
- gnu/mp Is Optimized for Different Architectures
- Mantissa Length mpf_t Is Foreground Data

Interval Computing: Data Types mpfi_t

mpfi_t (multiple-precision floating point interval library)

$\Uparrow$

mpfr_t (multiple-precision floating point with correct rounding)

$\Uparrow$

gnu/mp mpf_t

- mpfi_t { mpfr_t, mpfr_t }
- Guaranteed Results

Perfect Accuracy Computing GNU Multiple Precision Arithmetic Library: http://gmplib.org/, 2010.

Data types mpq_t

```
typedef struct
{
int _mp_alloc;
int _mp_size;
mp_limb_t *_m_d;
} __mpz_struct;

typedef struct
{
__mpz_struct _mp_num;
__mpz_struct _mp_den;
} __mpq_struct;
 typedef __mpq_struct mpq_t[1];
```

Message Passing Interface Standard http://www.MPI-forum.org/docs/MPI-11-html/MPI-report.html

Contributions

1. A. V. Panyukov, et all. Class Library "Exact Computational" / Software Certificate of Registry №2009612777 2009, May, 29 // Russian Agency Official Bulletin №3. – 2009. – C. 251. (in Russian)
2. V.A. Golodov. Distributed symbolic rational calculation on x86 and x64 CPUs, Proceedings of International conference "Parallel computing tecnologies", Novosibirsk, 2012, March 26 – March, 30), Chelyabinsk, South Ural State University Press, 774 p. (in Russian)
3. A. V. Panyukov, V. V. Gorbik Exact and Guaranteed Accuracy Solutions of Linear Programming Problems by Distributed Computer Systems with MPI // Tambov University REPORTS: A Theoretical and Applied Scientific Journal. Series: Natural and Technical Sciences. – Volume 15, Issue 4, 2010. – P. 1392-1404.
4. A. V. Panyukov, V. V. Gorbik Using massively parallel computations for absolutely precise solution of the linear programming problems, Automation and Remote Control, 2012, Vol. 73, No 2, pp. 276-290.

Big Numbers Addition Algorithm

Algorithm A

This algorithm calculates a sum of two given numbers with radix $R = 2^r$: $(a_{n-1}, \dots, a_0)_R$ and $(b_{m-1}, \dots, b_0)_R$, $m \leqslant n$. The sum will be stored in number $(c_n, \dots, c_0)_R$. Here c_n is a carry bit which is equals 0 or 1.

A1.Setup. Assign m threads (for addition implementing m threads is enough as the digits $(a_{n-1}, \dots, a_0)_r$ don't take part in the main process). For each thread $i = 1, 2, \dots, m$ input corresponding digit a_i into the temporary variable $t[i]$.

A2. Addition. At each thread $i = 1, 2, \dots, m$ let be

$$f[i] = \left\lfloor \frac{t[i] + b_i}{R} \right\rfloor, \qquad c[i] = (t[i] + b_i)\,\%R.$$

A3. Carry propagation. At each thread $i = 1, 2, \dots, m$ so that $f[i] = 1$ do

$$i = i + 1; \quad f[i] = \left\lfloor \frac{c[i] + 1}{R} \right\rfloor; \quad c[i] = (t[i] + 1)\,\%R$$

till $f[i] = 1$.

Estimation of the parallelism efficiency

- At step A1 digits values of number a are read into the variable $t[i]$ for each of the m threads. Theoretically this temporary variable can be avoided and the addition can be done indirectly but in practice the most of CPUs doesn't support such operations as memory-memory but register-memory type.
- Step A2 is performed entirely parallel so its implementation time will be minimal if the quantity of threads is enough. Let A be time of elementary addition performing.
- Step A3 implementation time is depended from input summands and is varied at the range $[0, nA]$

Probability Estimation of the parallelism efficiency

- Probability to get value $2^r - 1$ after addition of digits

$$q = \frac{1}{2^r}.$$

- Probability of carry appearance is equal

$$p = \frac{1}{2^r} \cdot \frac{1}{2^r}(0 + 1 + 2 + \cdots + (2^r - 1)) = \frac{1}{2} - \frac{1}{2 \cdot 2^r} = \frac{1-q}{2}.$$

- $S = \max\limits_{i=1,\ldots,m} s_i$ is maximum carry length over all of the threads
- $P\{S \le 1\} \cong (p(1-q)+(1-p))^m = (1-pq)^m = 1-\frac{mq}{2}+o(q);$

$$P\{S \ge 2\} \cong 1 - (1 - pq)^m = \frac{mq}{2} + o(q).$$

Way of Increasing Scalability

Usage of 2^r bit registers and $R = 2^{r-1}$ radix

- i-th order digits of the summands may have representations

$$a_i = \left(0\, a_i^{r-2} \ldots a_i^1\, a_i^0\right)_2,\ b_i = \left(0\, b_i^{r-2} \ldots,\, b_i^1\, b_i^0\right)_2.$$

- i-th order addition result

$$s_i = a_i + b_i = \left(s_i^{r-1}\, s_i^{r-2}\, s_i^{r-3} \ldots s_i^1\, s_i^0\right)_2,$$

  where $s_i^{r-1} \in \{0,\, 1\}$.
- Carry Propagation

$$\tilde{s}_i = \left(0\,0\, s_i^{r-3} \ldots s_i^1\, s_i^0\right)_2 + \left(s_{i-1}^{r-1}\, s_{i-1}^{r-2}\right)_2 = \left(0\, \tilde{s}_i^{r-2} \ldots \tilde{s}_i^1\, \tilde{s}_i^0\right)_2.$$

Conclusion

Merits

- Application of Redundant Positional Notations Let Us Increase Scalability of Addition and Multiplication Algorithms.
- Application of Sign Redundant Positional Notations Let Us Increase Scalability of Addition Substraction and Multiplication Algorithms.

Demerits

- Plurality representation of numbers, and computing of binary relations requires uniqueness.
- Complexity of plurality excluding is the same as for Algorithm A.

Thank you for attention!