Parallelization of Discrete Stochastic Arithmetic on multicore architectures

Fabienne Jézéquel[1], Jean-Luc Lamotte[1], Olena Chubach[2]

[1] Laboratoire d'Informatique de Paris 6
Université Pierre et Marie Curie, Paris
France

[2] Odessa I. I. Mechnikov National University,
Odessa,
Ukraine

10th International Conference on Information Technology: New Generations, ITNG 2013
April 15-17, 2013
Las Vegas, Nevada, USA

Introduction

Discrete Stochastic Arithmetic (DSA)

- based on a probabilistic approach
- enables one to estimate round-off error propagation in a program ☺
- cost (memory, execution time) ☹

How to take benefit of multicore architectures to reduce the cost of DSA for the numerical validation of sequential programs?

The CESTAC method

M. La Porte, J. Vignes, 1974

The implementation of the CESTAC method in a code providing a result R consists in:

- performing N times this code with the random rounding mode to obtain N samples R_i of R,
- choosing as the computed result the mean value $\overline{R}$ of R_i, $i = 1, ..., N$,
- estimating the number of exact significant decimal digits of $\overline{R}$ with

$$C_{\overline{R}} = \log_{10}\left(\frac{\sqrt{N}\left|\overline{R}\right|}{\sigma\tau_\beta}\right)$$

where

$$\overline{R} = \frac{1}{N}\sum_{i=1}^{N} R_i \quad \text{and} \quad \sigma^2 = \frac{1}{N-1}\sum_{i=1}^{N}\left(R_i - \overline{R}\right)^2.$$

τ_β is the value of Student's distribution for $N-1$ degrees of freedom and a probability level β.

In pratice, $N = 3$ and $\beta = 95\%$.

Self-validation of the CESTAC method

The CESTAC method is based on a 1st order model.

- A multiplication of two insignificant results
- or a division by an insignificant result

may invalidate the 1st order approximation.

Therefore the CESTAC method requires a dynamical control of multiplications and divisions, during the execution of the code.

The concept of computed zero

J. Vignes, 1986

Definition

Using the CESTAC method, a result R is a computed zero, denoted by @.0, if

$$\forall i, R_i = 0 \ \text{ or } \ C_{\overline{R}} \leq 0.$$

It means that R is a computed result which, because of round-off errors, cannot be distinguished from 0.

The stochastic definitions

Definition

Let X and Y be two results computed using the CESTAC method (N-sample), X is stochastically equal to Y, noted $X\ s\!=\ Y$, if and only if

$$X - Y = @.0.$$

Definition

Let X and Y be two results computed using the CESTAC method (N-sample).

- X is stochastically strictly greater than Y, noted $X\ s\!>\ Y$, if and only if

$$\overline{X} > \overline{Y} \quad \text{and} \quad X\ s\!\neq\ Y$$

- X is stochastically greater than or equal to Y, noted $X\ s\!\geq\ Y$, if and only if

$$\overline{X} \geq \overline{Y} \quad \text{or} \quad X\ s\!=\ Y$$

Discrete Stochastic Arithmetic (DSA) is defined as the joint use of the CESTAC method, the computed zero and the stochastic relation definitions.

The CADNA library http://www.lip6.fr/cadna

The CADNA library implements Discrete Stochastic Arithmetic.

CADNA allows to estimate round-off error propagation in any scientific program written in Fortran or in C++.

More precisely, CADNA enables one to:

- estimate the numerical quality of any result
- control branching statements
- perform a dynamic numerical debugging
- take into account uncertainty on data.

CADNA provides new numerical types, the stochastic types, which consist of:

- 3 floating point variables
- an integer variable to store the accuracy.

All operators and mathematical functions are overloaded for these types.

Parallelization of Discrete Stochastic Arithmetic

3 UNIX processes are executed in parallel.
They exchange information through a communication system.

Functions and operations that require data exchange:

1st group: synchronization required
...to ensure all processes compute the same result and perform the same sequence of instructions.

- equality and order relational operations
- the absolute value function
- conversions from a stochastic type to a classical floating-point type
- functions which compute the number of exact significant digits of results

Parallelization of Discrete Stochastic Arithmetic

3 UNIX processes are executed in parallel.
They exchange information through a communication system.

Functions and operations that require data exchange:

1st group: synchronization required
...to ensure all processes compute the same result and perform the same sequence of instructions.

2nd group: a part of the computation can be performed later

- multiplications
- divisions

The control of instabilities can be postponed. It has no impact on the choice of the next instructions.

Execution of a program using multicore DSA

user program:
cadna_init(-1);
...

- Creation of a shared memory segment
- Launch of 2 other identical processes (*fork* UNIX function)

process 1:	process 2:	process 3:
...	...	...

Execution of a program using multicore DSA

user program:
cadna_init(-1);
...
$A = ...$
$B = ...$

All assignments, arithmetical operations and mathematical functions are overloaded.

process 1:	process 2:	process 3:
...	...	...
$A_1 = ...$	$A_2 = ...$	$A_3 = ...$
$B_1 = ...$	$B_2 = ...$	$B_3 = ...$

Execution of a program using multicore DSA

```
user program:
cadna_init(-1);
...
A = ...
B = ...
if (A == B)
```

Each process computes the difference between its operands.
Associativity is not necessarily satisfied in IEEE floating-point arithmetic
$\Rightarrow$ the 3 processes must have the same ordered triplet $D = (D_1, D_2, D_3)$.
The number $C_{\overline{D}}$ of exact significant digits of D is computed by all processes.

process 1:	process 2:	process 3:
...	...	...
$A_1 = ...$	$A_2 = ...$	$A_3 = ...$
$B_1 = ...$	$B_2 = ...$	$B_3 = ...$
$D_1 = A_1 - B_1$	$D_2 = A_2 - B_2$	$D_3 = A_3 - B_3$
all_to_all_exchange(D_1 , D_2, D_3)		
$D = (D_1, D_2, D_3)$	$D = (D_1, D_2, D_3)$	$D = (D_1, D_2, D_3)$
if ($D ==$ @.0)	if ($D ==$ @.0)	if ($D ==$ @.0)

Execution of a program using multicore DSA

user program:
cadna_init(-1);
...
$A = ...$
$B = ...$
if $(A == B)$
...
cadna_end();

The branch chosen is the same for the three processes.

process 1:	process 2:	process 3:
...	...	...
$A_1 = ...$	$A_2 = ...$	$A_3 = ...$
$B_1 = ...$	$B_2 = ...$	$B_3 = ...$
$D_1 = A_1 - B_1$	$D_2 = A_2 - B_2$	$D_3 = A_3 - B_3$
all_to_all_exchange(D_1 , D_2, D_3)		
$D = (D_1, D_2, D_3)$	$D = (D_1, D_2, D_3)$	$D = (D_1, D_2, D_3)$
if ($D ==$ @.0)	if ($D ==$ @.0)	if ($D ==$ @.0)
...	...	...

Several multicore versions

1. with synchronous data exchange
   any data exchange is performed synchronously

2. with a validation box
   - 1st group of functions or operations:
     synchronizations
   - 2nd group of functions or operations (multiplications, divisions):
     the control of accuracy can be postponed

   Computation box: 3 processes run 3 instances of the program and fill buffers with multiplication operands & divisors
   Validation box: 1 process checks their accuracy

3. with a validation box and an *accuracy* variable associated with any stochastic number
   Without it, the accuracy of a stochastic number may be computed several times even if this number is not modified.

Several multicore versions

1. with synchronous data exchange
   any data exchange is performed synchronously
2. with a validation box
   - 1st group of functions or operations:
     synchronizations
   - 2nd group of functions or operations (multiplications, divisions):
     the control of accuracy can be postponed
3. with a validation box and an *accuracy* variable associated with any stochastic number
   Without it, the accuracy of a stochastic number may be computed several times even if this number is not modified.

Performance test (quad-core Intel i5-2500 processor, gcc 4.6.3 compiler)
Matrix multiplication & linear system solving using Jacobi method
Versions 1 & 3 $\Rightarrow$ similar performance
cost reduced by $\approx$ 2 w.r.t. the sequential CADNA library.

Computation of integrals using the trapezoidal method

$$I_1 = \int_1^{100} f_1(x)dx \text{ with } f_1(x) = \frac{\sin(x)}{x} + \cos(x)\exp(\sin(x))$$

Execution	instability detection	execution time (s)	ratio
IEEE	-	8.80	1
sequential DSA	full	94.00	10.7
	self-validation	66.17	**7.5**
	no detection	57.57	6.5
parallel DSA (synchronous exchange)	self-validation	56.73	6.4
	no detection	30.59	3.5
parallel DSA (validation box)	self-validation	35.11	4.0
	no detection	28.06	3.2
parallel DSA (validation box & accuracy)	self-validation	32.28	**3.7**
	no detection	32.24	3.7

Computation of integrals using the trapezoidal method

$$I_2 = \int_{-1}^{2} f_2(x)dx \text{ with } f_2(x) = \frac{2x^5-10x^4+5x^3-60x^2+80x+37}{8x^4+13x^3-38x^2+43x+513}$$

f_2 is particularly unfavourable to DSA, because it contains mathematical expressions that are efficiently computed using IEEE floating-point arithmetic.

Execution	instability detection	execution time (s)	ratio
IEEE	-	0.22	1
sequential DSA	full	40.18	182.6
	self-validation	28.15	**128.0**
	no detection	20.02	91.0
parallel DSA (synchronous exchange)	self-validation	17.91	81.4
	no detection	10.96	49.8
parallel DSA (validation box)	self-validation	23.09	105.0
	no detection	8.71	39.6
parallel DSA (validation box & accuracy)	self-validation	10.85	**49.3**
	no detection	10.81	49.1

Numerical validation of the shallow-water application

Simulation of the linear flow of a nonviscous fluid in shallow-water environment with a free surface (over 8,000 lines of codes)

Numerical instabilities:

- 212 unstable multiplications
- 149,564 losses of accuracy due to cancellations

Execution	instability detection	execution time (s)	ratio
IEEE	-	7.76	1
sequential DSA	full	192.38	24.8
	self-validation	70.64	**9.1**
	no detection	70.65	9.1
parallel DSA (synchronous exchange)	self-validation	41.34	5.3
	no detection	19.42	2.5
parallel DSA (validation box)	self-validation	25.28	3.3
	no detection	16.75	2.2
parallel DSA (validation box & accuracy)	self-validation	20.17	**2.6**
	no detection	20.19	2.6

Numerical validation of the shallow-water application

Simulation of the linear flow of a nonviscous fluid in shallow-water environment with a free surface (over 8,000 lines of codes)

Execution	instability detection	execution time (s)	ratio
IEEE	-	7.76	1
sequential DSA	full	192.38	24.8
	self-validation	70.64	**9.1**
	no detection	70.65	9.1
parallel DSA (synchronous exchange)	self-validation	41.34	5.3
	no detection	19.42	2.5
parallel DSA (validation box)	self-validation	25.28	3.3
	no detection	16.75	2.2
parallel DSA (validation box & accuracy)	self-validation	20.17	**2.6**
	no detection	20.19	2.6

- moderate cost of DSA: the shallow-water application performs not only computation but also I/O tasks.
- cost reduced by 3.5 w.r.t. the sequential CADNA library with self-validation.

Conclusion

Recommended version: validation box and *accuracy* variable

cost reduced by ≈ 2 w.r.t. the sequential CADNA library

The cost on a computation kernel may be high.
It usually becomes reasonable on a real-life application.

same modifications required by the sequential CADNA library and our parallel implementation of DSA.

Recommended strategy:

1. execution with our parallel implementation of DSA to check the numerical quality of the results
2. for a more detailed analysis execution with the CADNA library

   instructions responsible for numerical instabilities:

   - identified with a debugger
   - if possible, modified to improve the numerical quality of the results.