Towards an efficient implementation of CADNA in the BLAS : Example of DgemmCADNA routine

SCAN'2012

September 2012

Séthy MONTAN[1,2], Jean-Marie CHESNEAUX[2], Christophe DENIS[1], Jean-Luc LAMOTTE[2]

[1] EDF R&D/SINETICS, Clamart.
[2] UPMC - LIP6/PEQUAN, Paris.

SORBONNE UNIVERSITÉS

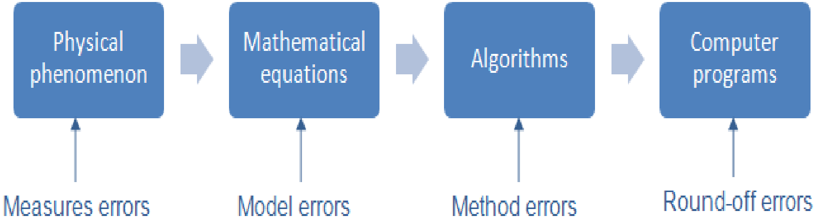
CHANGER L'ENERGIE ENSEMBLE

Summary

1. Motivations

Numerical simulation

Several approximations !



Computed results can be wrong !

- Round-off errors at each elementary arithmetic operation
- Detect and control these errors
  - Numerical validation

Numerical validation : Tools/Methods (1)

Methods for accurate computations

- Multiple precision arithmetic :
  - ex : MPFR, Gnu MP, MPFI.
- Compensated methods :
  - compensated summation algorithms, compensated dot product algorithms...

Numerical validation : Tools/Methods (2)

Methods for rounding error analysis

- Inverse analysis :
  - provides error bounds for the computed results.
- Interval arithmetic : the result of an arithmetic operation between two intervals contains all values that can be obtained by performing this operation on elements from each interval.
- Probabilistic approach :
  - uses a random rounding mode (CESTAC Method) ;
  - estimates the number of exact significant digits of any computed result.

High Performance Computing at EDF R&D

Codes	Tools	Hardware
Code_Aster	MPI/OpenMP	Ivanoe
Code_Saturne	BLAS/LAPACK	Blue Gene
TELEMAC	MUMPS/PASTIX	Clamart2
Code_...	. . .	Z600
. . .	. . .	. . .

Find an adapted tool for the industrial context (EDF)

The CADNA Library [SJDC07].

2. The CADNA Library

The CESTAC method :

The CESTAC method (Contrôle et Estimation Stochastique des Arrondis de Calculs) was proposed by M. La Porte and J. Vignes in 1974 [VLP74].

It consists in running the same code several times with different round-off error propagations. Then, different results are obtained.

➧ the part that is common to all the different results is assumed to be also in common with the mathematical result ;
➧ the part that is different in the results is affected by the round-off errors.

Discrete Stochastic Arithmetic

- N differents runs with random rounding mode ($+\infty$ ou $-\infty$ with the probability 0.5) ;
- N different results R_i :
  - choosing as the computed result the mean value $\overline{R}$ of R_i ;
  - estimating C_R the number of exact significant decimal digits of $\overline{R}$.
- $N = 3$
  - $X = (X_1, X_2, X_3)$
  - $\forall \Omega \in (+, -, \times, /)$, $X \Omega Y = (X_1 \omega Y_1, X_2 \omega Y_2, X_3 \omega Y_3)$
- If $C_R \leq 0$ or $\forall i, R_i = 0$, a result R is a computed zero (@.0).

- New order relationships.

- Discrete Stochastic Arithmetic (DSA).

The CADNA library

- The CADNA library implements Discrete Stochastic Arithmetic. It allows the estimation of round-off error propagation in any scientific program [JCL10].

- 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 is a library which can be used with Fortran or C++ programs and also with MPI parallel programs. CADNA can be downloaded from `http://www.lip6.fr/cadna`

How to use the Cadna Library

- CADNA provides two new numerical types, the stochastic types (3 floating point variables x,y,z and a hidden variable acc) :
  - type (*single_st*) in single precision
  - type (*double_st*) in double precision.
- All the operators and mathematical functions are overloaded for these types.
- To use the library :
  1. declaration of the CADNA library
  2. initialization of the CADNA library
  3. substitution of the floating point type by stochastic types
  4. change of output statements to print stochastic results with their accuracy
  5. termination of the CADNA library

High Performance Computing at EDF R&D

Codes
- Code_Aster
- Code_Saturne
- TELEMAC
- Code_...
- . . .

Tools
- MPI/OpenMP
- BLAS/LAPACK
- MUMPS/PASTIX
- . . .
- . . .

Harware
- Ivanoe
- Blue Gene
- Clamart2
- Z600
- . . .

Is it possible to study the numerical quality of every industrial code with CADNA ?

3. CADNA Implementation in scientific libraries

Different extensions for CADNA

1. MPI extension for CADNA : CADNA MPI

2. BLACS extension for CADNA : CADNA BLACS

3. Efficient implementation of CADNA in BLAS

MPI/BLACS extensions for CADNA

- Definition of stochastic types to exchange data
- Definition of reduction operators
- C/C++ (MPI2) , Fortran 90 (MPI1)

  - (--) The Sendind time of a stochastic float is 4 times more long than a normal float one.
    - Size of stochastic type = 4 *times* size of normal float
  - (++) It is possible to use CADNA with any code using MPI and BLACS.

BLAS : Basic Linear Algebra Subprograms

- Functionality
  - Level 1 : vectors operations (ex *xAXPY*) ;
  - Level 2 : matrix-vectors operations (ex *xGEMV*) ;
  - Level 3 : matrix-matrix operations (ex *xGEMM*).

- Implementations

versions	daxpy	dgemv	dgemm
Netlib	1.18482	1.15347	1.35378
Mkl 1 threads	2.02116	2.11232	7.53686
Goto 1 threads	2.86331	2.12331	7.52166
Mkl 8 threads	1.63618	2.79974	58.0523
Goto 8 threads	1.63618	4.60287	56.3343

TABLE: GFLOPS for daxpy, dgemv et dgemm : 4096*4096 matrix (4096 vector).

How to use CADNA with Blas routines ?

- The easiest solution (V1) : Remplaced *float* by *float_st* et *double* by *double_st* :

```
void cblas_dgemm(const enum CBLAS_ORDER ↩
    Order,const enum CBLAS_TRANSPOSE ↩
    TransA,const enum CBLAS_TRANSPOSE ↩
    TransB,const int M,const int N,const ↩
    int K,const double_st alpha,const ↩
    double_st *A,const int lda,const ↩
    double_st *B, const int ldb,const ↩
    double_st beta, double_st *C, const int ↩
    ldc);
```

- Linalg : A template version of BLAS ; it can be used with stochastic types

Direct Implementations of DGEMM with CADNA

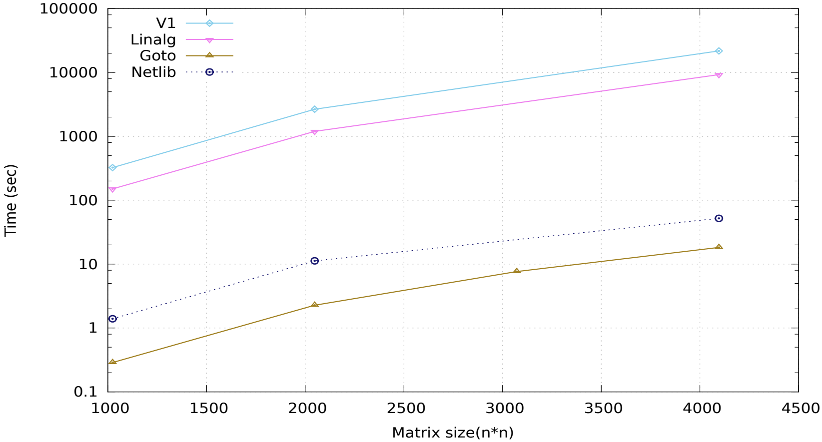


FIGURE: Versions with and without CADNA

Direct Implementations of DGEMM with CADNA(2)

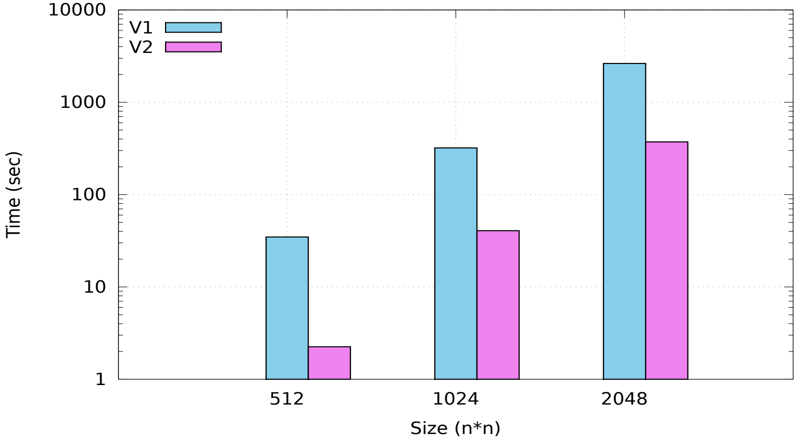


FIGURE: Overhead due to the CESTAC Method

Why these overheads ?

- An overhead greater than **1000** for a 1024*1024 matrix
  - DGEMM with 3 inner loops => cache misses
  - Use of stochastic types and the discrete stochastic arithmetic
  - Random rounding mode $V1 > 7xV2$

Efficient implementation of DgemmCADNA

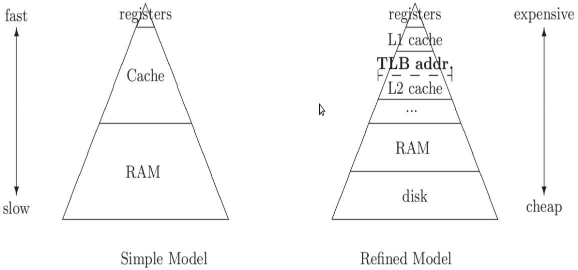
« Implementing matrix multiplication so that near-optimal performance is attained requires a thorough understanding of how the operation must be layered at the macro level in combination with careful engineering of high-performance kernels at the micro level. »

K. Goto, 2008 [GVDG08].

- Solutions to reduce the overhead :

1. Efficient use of data (memory access)
2. Minimize the CESTAC Method impact
3. Optimize the inner loop

Efficient use of data or data reuse



- Use tiled algorithms
- Optimize cache locality
- Exploit temporal and spacial locality
- Reduce cache misses
- Reduce TLB misses

An Iterative tiled algorithm

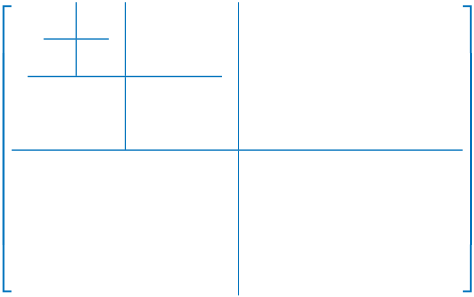
$$\begin{bmatrix} C_{11} & C_{12} & \dots & C_{1N} \\ C_{21} & C_{22} & \dots & C_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ C_{N1} & C_{N2} & \dots & C_{NN} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1N} \\ A_{21} & A_{22} & \dots & A_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ A_{N1} & A_{N2} & \dots & A_{NN} \end{bmatrix} \times \begin{bmatrix} B_{11} & B_{12} & \dots & B_{1N} \\ B_{21} & B_{22} & \dots & B_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ B_{N1} & B_{N2} & \dots & B_{NN} \end{bmatrix}$$

every C_{ij} is computed by :

$$C_{ij} = \sum_{k=1}^{N} A_{ik} B_{kj}$$

A recursive tiled algorithm DGBRn

$$\left[\begin{array}{c|c} C_{00} & C_{01} \\ \hline C_{10} & C_{11} \end{array}\right] = \left[\begin{array}{c|c} A_{00} & A_{01} \\ \hline A_{10} & A_{11} \end{array}\right] \times \left[\begin{array}{c|c} B_{00} & B_{01} \\ \hline B_{10} & B_{11} \end{array}\right] \qquad (1)$$



An iterative tiled algorithm based on the hardware (hierarchical memory) DGBIn

- 3 levels of partitioning. One level for every cache level. The matrix (submatrices) is partitioned in submatrices (blocks). At each step, 3 blocks must fit in this level of cache memory.

1. First level for Cache L3
2. Second level for Cache L2
3. Third level for Cache L1

$$A(n*n) = \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1N} \\ A_{21} & A_{22} & \dots & A_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ A_{N1} & A_{N2} & \dots & A_{NN} \end{bmatrix}$$

BDL : *Block Data Layout*

- *Column Major order*

$$M = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 4 & 7 & 2 & 5 & 8 & 3 & 6 & 9 \end{bmatrix}$$

- *Row Major order*

$$M = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{bmatrix}$$

BDL : *Block Data Layout* (2)

Consider matrix $A(n \times n)$ partitioned in $N \times N$ submatrices A_{ij} :

$$A(n*n) = \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1N} \\ A_{21} & A_{22} & \dots & A_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ A_{N1} & A_{N2} & \dots & A_{NN} \end{bmatrix} \qquad A_{ij}(p*p) = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1p} \\ a_{21} & a_{22} & \dots & a_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ a_{p1} & a_{p2} & \dots & a_{pp} \end{bmatrix}$$

Data within one such block A_{ij} are mapped onto contiguous memory :

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1p} \, a_{21} & a_{22} & \dots & a_{2p} \dots a_{p1} & a_{p2} & \dots & a_{pp} \end{bmatrix}$$

Theses blocks are arranged in *row-major order* :

$$\begin{bmatrix} A_{11} & A_{12} & \dots & A_{1N} A_{21} & A_{22} & \dots & A_{2N} \dots A_{N1} & A_{N2} & \dots & A_{NN} \end{bmatrix}$$

Reduce the impact of DSA

Unroll every Cadna arithmetic operation : **NO MORE** operator overloading

```
C[i] = A[i] + B[i] ;
```

```
C[i].x = A[i].x + B[i].x ;
if (random) rnd_switch();
C[i].y = A[i].y + B[i].y ;
if (random) rnd_switch();
C[i].z = A[i].z + B[i].z ;
rnd_switch();
```

Reduce the impact of DSA (2)

less calls to rnd_switch()

```
if(random) rnd_switch()
C[i].x = A[i].x + B[i].x ;
C[i].z = A[i].z + B[i].z ;
C[i+1].z = A[i+1].z + B[i+1].z ;
C[i+2].x = A[i+2].x + B[i+2].x ;
C[i+2].y = A[i+2].y + B[i+2].y ;
C[i+3].x = A[i+3].x + B[i+3].x ;
rnd_switch();
C[i].y = A[i].y + B[i].y ;
C[i+1].x = A[i+1].x + B[i+1].x ;
C[i+1].y = A[i+1].y + B[i+1].y ;
C[i+2].z = A[i+2].z + B[i+2].z ;
C[i+3].y = A[i+3].y + B[i+3].y ;
C[i+3].z = A[i+3].z + B[i+3].z ;
```

Optimize the kernel

Listing 1 – Inner loops

```
    for(int i = 0; i < nb_block; i++){
      for(int k = 0; k < nb_block; k++){
        for(int j = 0; j < nb_block; j++){
          Cij = Aik * Bkj /*kernel*/
```

$$C_{00} = A_{00} \times B_{00}$$
$$C_{01} = A_{00} \times B_{01}$$
$$C_{00} = A_{01} \times B_{10}$$
$$C_{01} = A_{01} \times B_{11}$$
$$C_{10} = A_{10} \times B_{00}$$
$$C_{11} = A_{10} \times B_{01}$$
$$C_{10} = A_{11} \times B_{10}$$
$$C_{11} = A_{11} \times B_{11}$$

Results

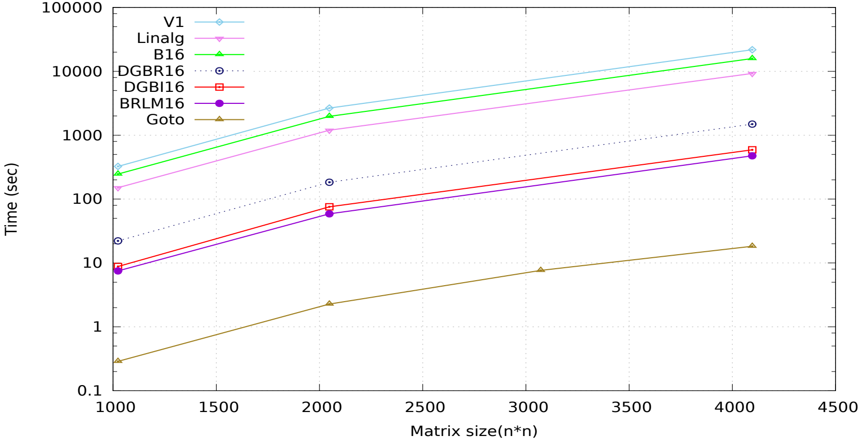


FIGURE: Different versions of DgemmCADNA

Results (2)

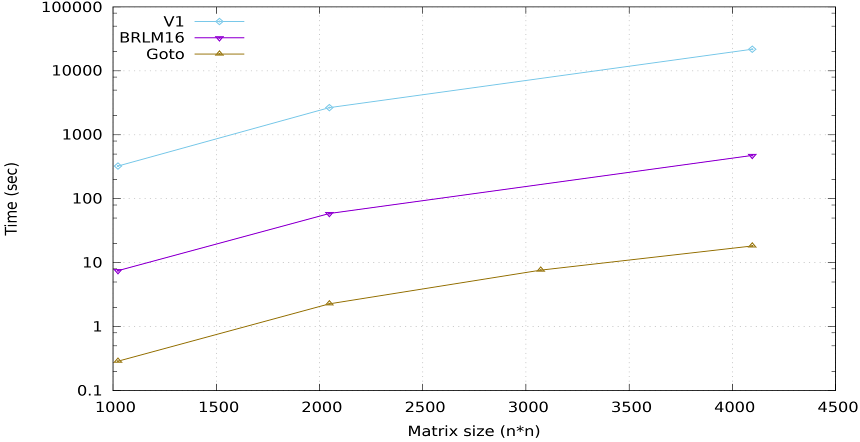


FIGURE: Comparison to GotoBLAS

4. Conclusion

Conclusion et Future work

- Conclusion
  - CADNA extensions for MPI and BLACS ;
  - DgemmCADNA subroutine :
    - 45*x* faster than the first version
    - Gotoblas 25*x* faster than DgemmCADNA
- Future work
  - Include the CADNA autovalidation ;
  - Theorical proof of the CESTAC Method modification
  - Work on the other blas routines (level 1 and level 2)
  - Experimental test phase for the implemented routines in a industrial codes (TELEMAC)

References I

J.M. Chesneaux, S. Graillat, and F. Jézéquel.
Numerical validation and assessment of numerical accuracy.

Christophe Denis and Sethy Montan.
Numerical verification of industrial numerical codes.
ESAIM : Proc., 35 :107--113, march 2012.

U. Drepper.
What every programmer should know about memory.
2007.
http://people.redhat.com/drepper/cpumemory.pdf.

K. Goto and R.A. Geijn.
Anatomy of high-performance matrix multiplication.
ACM Transactions on Mathematical Software (TOMS), 34(3) :12, 2008.

K. Goto and R. Van De Geijn.
High-performance implementation of the level-3 blas.
ACM Transactions on Mathematical Software (TOMS), 35(1) :4, 2008.

F. Jézéquel, J.M. Chesneaux, and J.L. Lamotte.
A new version of the cadna library for estimating round-off error propagation in fortran programs.
Computer Physics Communications, 181(11) :1927--1928, 2010.

J. Kurzak, W. Alvaro, and J. Dongarra.
Optimizing matrix multiplication for a short-vector simd architecture-cell processor.
Parallel Computing, 35(3) :138--150, 2009.

References II

Neungsoo Park, Bo Hong, and Viktor K. Prasanna.
Tiling, block data layout, and memory hierarchy performance.
IEEE Transactions on Parallel and Distributed Systems, 14 :640--654, 2003.

NS Scott, F. Jézéquel, C. Denis, and J.M. Chesneaux.
Numerical 'health check' for scientific codes : the cadna approach.
Computer physics communications, 176(8) :507--521, 2007.

J. Vignes and M. La Porte.
Error analysis in computing.
Information Processing, 74 :610--614, 1974.

Thanks !

sethy.montan@edf.fr