Implémentation efficace de la bibliothèque CADNA dans les bibliothèques de calcul et de communication scientifiques

26 mai 2011

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

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

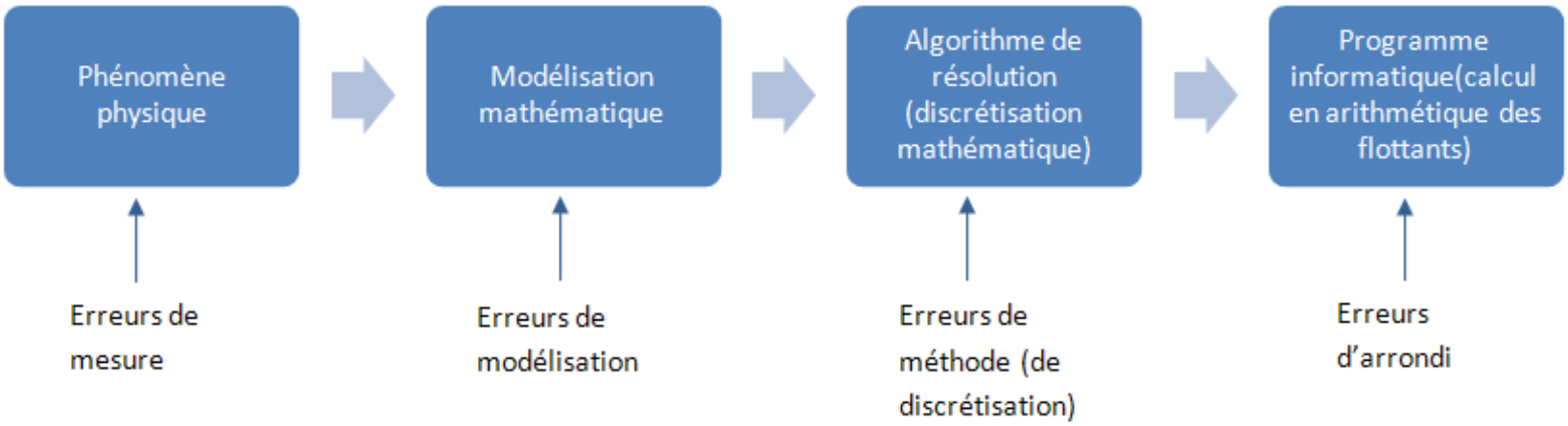
LIP6 UPMC SORBONNE UNIVERSITÉS EDF CHANGER L'ENERGIE ENSEMBLE

Sommaire

1. Validation numérique et CADNA

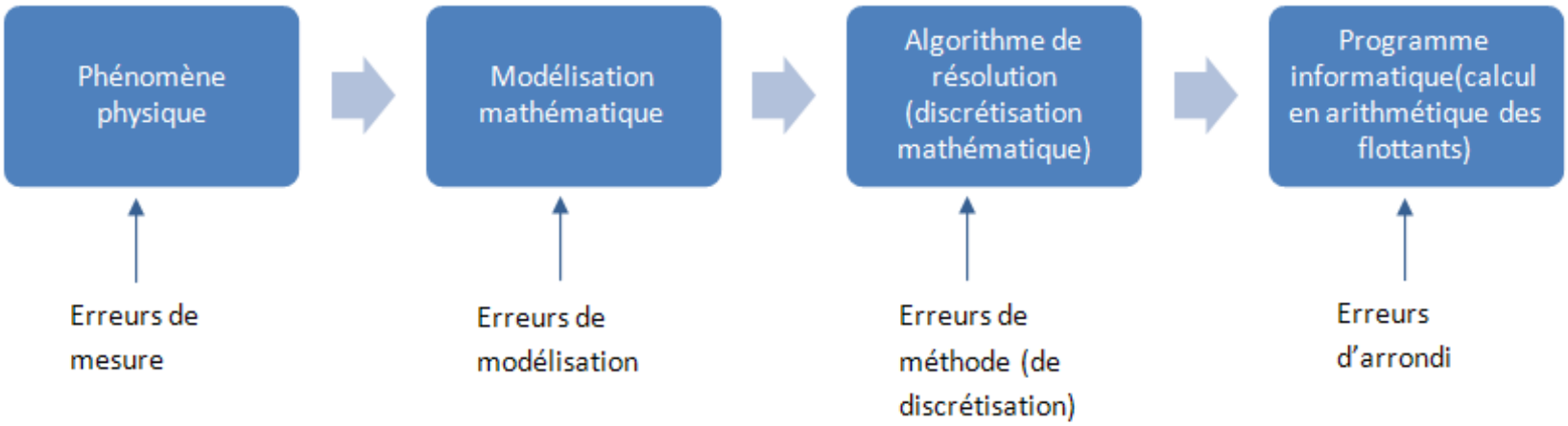
Simulation numérique

Plusieurs approximations aux différentes étapes !



Simulation numérique

Plusieurs approximations aux différentes étapes !

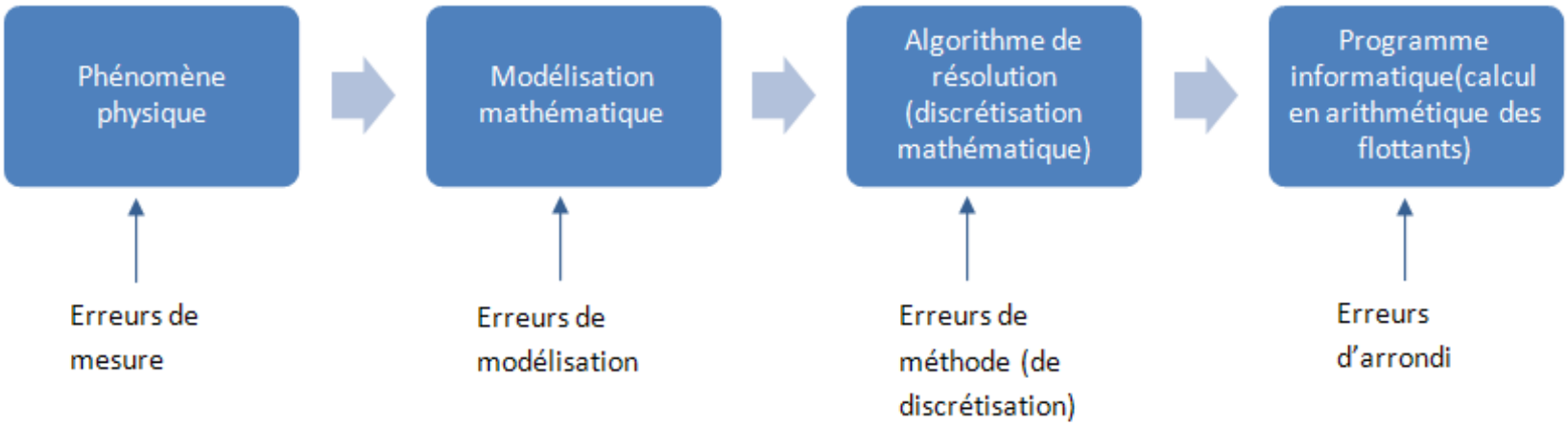


Arithmétique flottante : résultats de calcul potentiellement erronés !

- Les erreurs d'arrondi existent intrinsèquement dans un code numérique
- Il faut donc détecter et contrôler la propagation de ces erreurs

Simulation numérique

Plusieurs approximations aux différentes étapes !



Arithmétique flottante : résultats de calcul potentiellement erronés !

- Les erreurs d'arrondi existent intrinsèquement dans un code numérique
- Il faut donc détecter et contrôler la propagation de ces erreurs => VALIDATION NUMERIQUE

CADNA : un outil de validation numérique open source

CADNA implémente la méthode CESTAC

- Mesurer l'effet de la propagation des erreurs d'arrondi dans un code séquentiel
- Détecter les instabilités numériques puis afficher diagnostic à la fin de l'exécution

float_st

```
float x,y,z ;
int acc = -1 ;
```

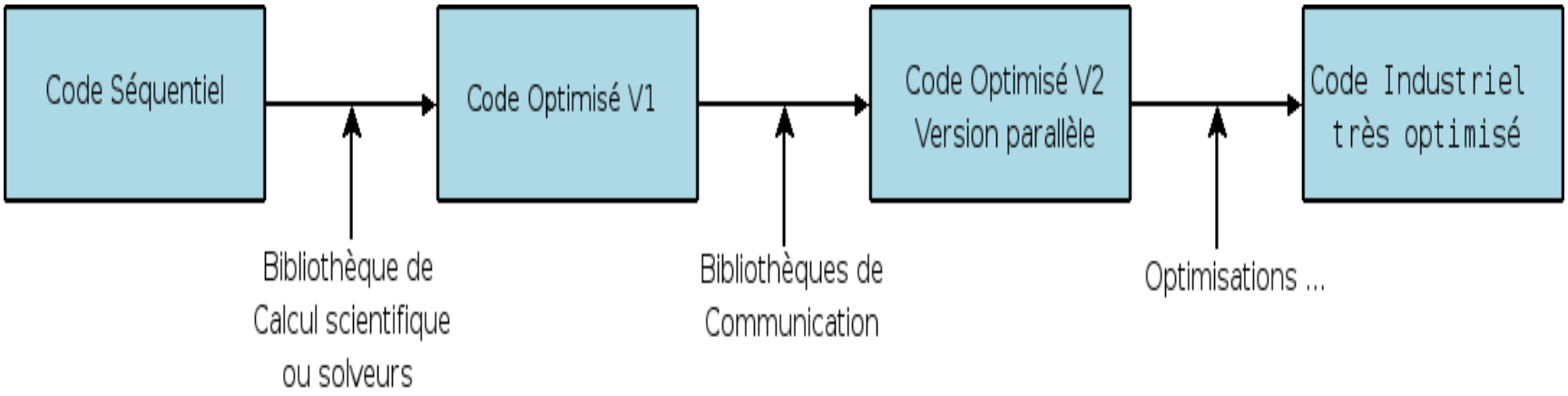
double_st

```
double x,y,z ;
int acc = -1 ;
```

- Surcharge de toutes les opérations mathématiques
- 2 fonctions obligatoires : *cadna_init() cadna_end()*

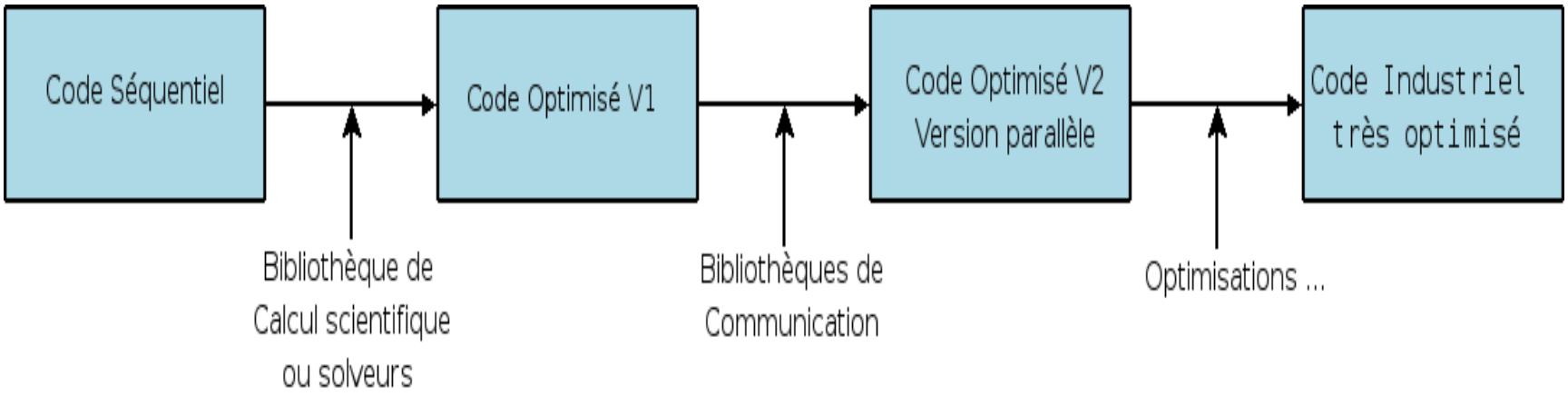
Peut-on valider un code industriel avec CADNA ?

Code industriel se construit en plusieurs étapes ...



Peut-on valider un code industriel avec CADNA ?

Code industriel se construit en plusieurs étapes ...



- On peut valider seulement la version séquentielle avec CADNA
- Et pour les autres ?

2. Les bibliothèques de communication

1. Validation numérique et CADNA

2. Les bibliothèques de communication

- Le standard MPI
- Les routines BLACS

3. Les bibliothèques de calcul

4. Conclusion

MPI : Message Passing Interface

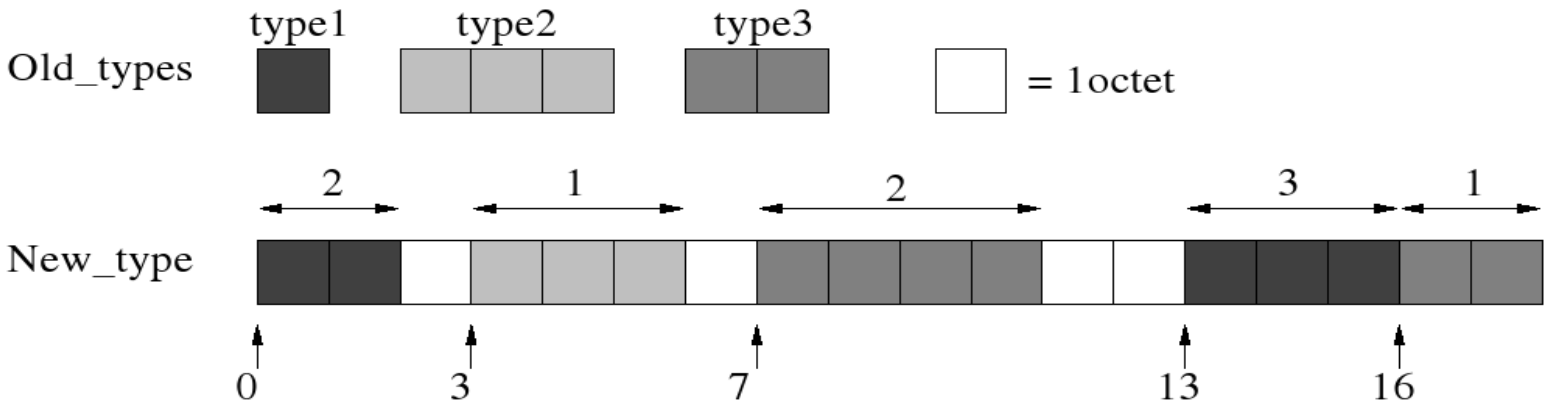
Standard d'interface de bibliothèque de communication et d'environnement parallèle

- Communication par échange de messages des processus
  - distants
  - sur des machines qui peuvent être hétérogènes
- Echange de données (collective, point-à-point)
- Pour applications écrites en C, C++ ou Fortran
- Exemple d'implémentations :
  - domaine public : LAM, OpenMPI, MPICH
  - implémentations constructeurs : IBM, HP, SUN ...

Extension de MPI pour CADNA : CADNA MPI

- Définition des types stochastiques pour les échanges de données
  - MPI_SINGLE_ST pour la simple précision
  - MPI_DOUBLE_ST pour la double précision
  - MPI_(DOUBLE / SINGLE)_ST_INT pour les réductions
- Définition des opérateurs de réductions
  - SUM, PROD, MAX, MIN
  - MAX_LOC, MIN_LOC
- Redéfinition des fonctions *cadna_init() cadna_end()*
- Développement C/C++ (MPI2) et Fortran 90 (MPI1)

CADNA MPI : Les Types stochastiques



nb_blocs=5

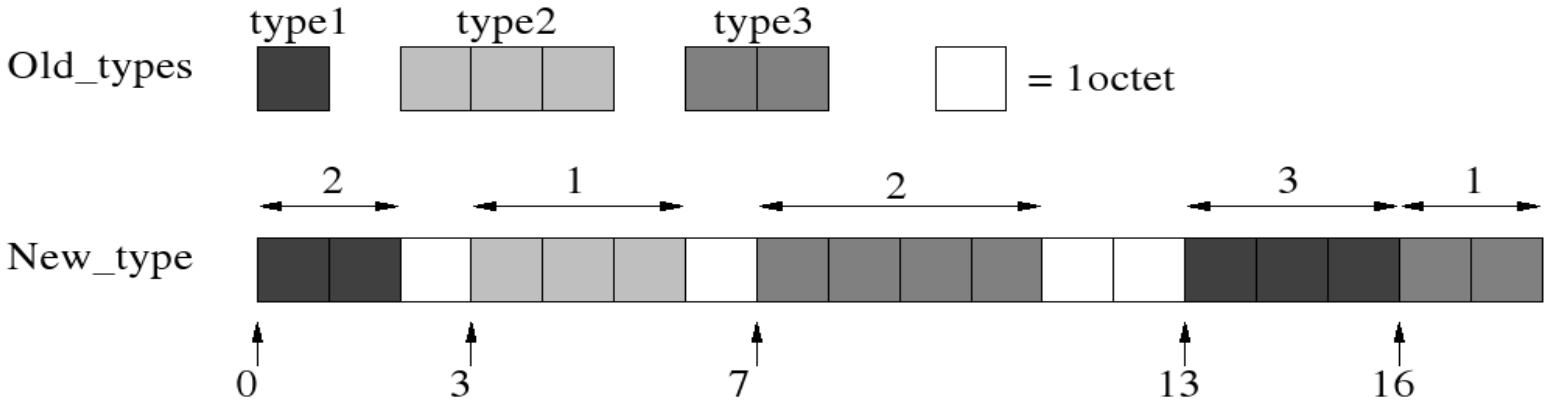
longueur_bloc=(2, 1, 2, 3, 1)

tab_types=(type1, type2, type3, type1, type3)

deplacement=(0, 3, 7, 13, 16), en nombre d'octets

```
MPI_Type_Struct(nb_element,longeur_bloc,\
        deplacement,tableau_types,new_type)
MPI_Type_Commit(new_type)
```

CADNA MPI : Les Types stochastiques



nb_blocs=5

longueur_bloc=(2, 1, 2, 3, 1)

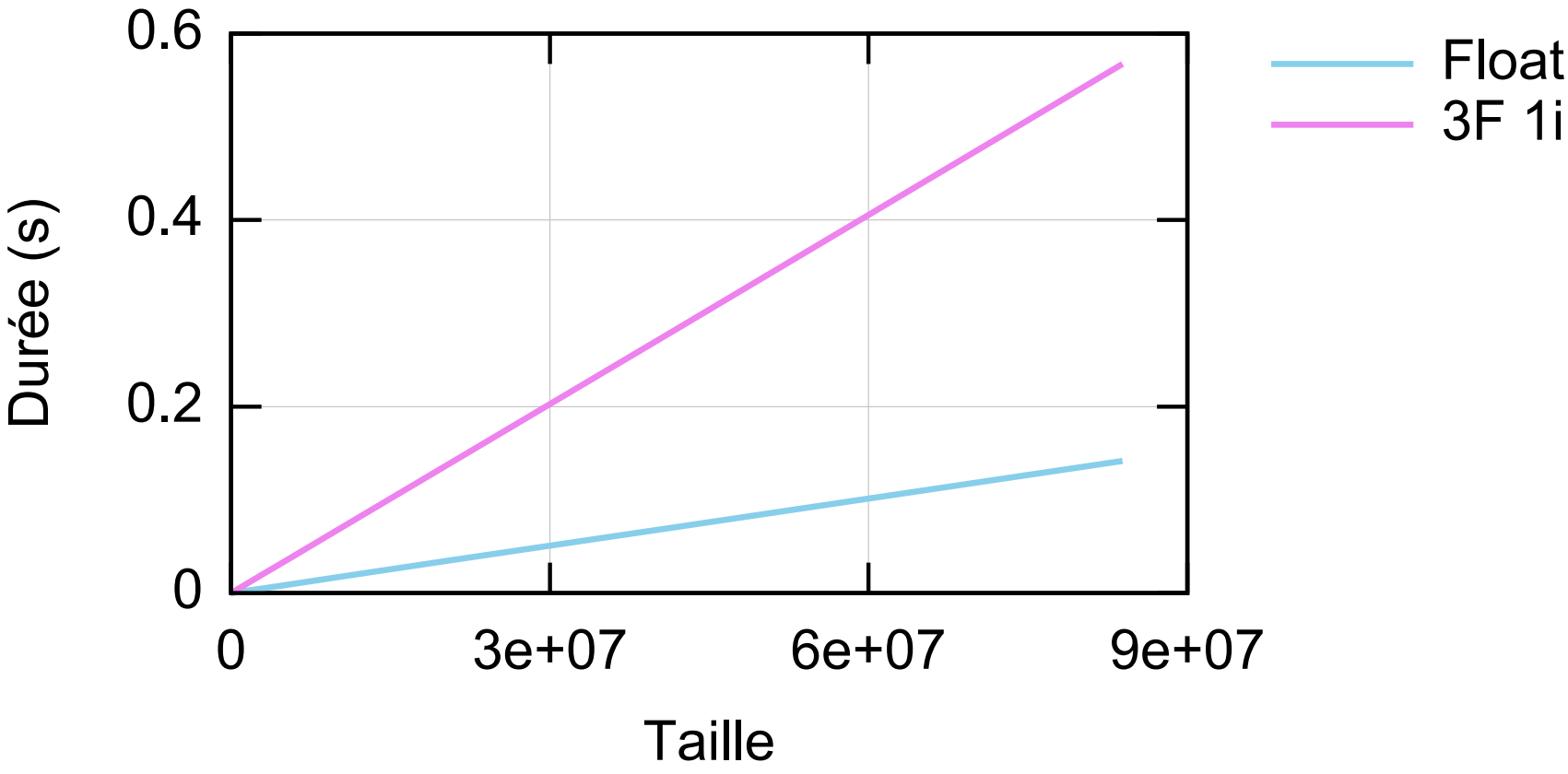
tab_types=(type1, type2, type3, type1, type3)

deplacement=(0, 3, 7, 13, 16), en nombre d'octets

```
MPI_Type_Struct(nb_element,longeur_bloc,\
        deplacement,tableau_types,new_type)
MPI_Type_Commit(new_type)
```

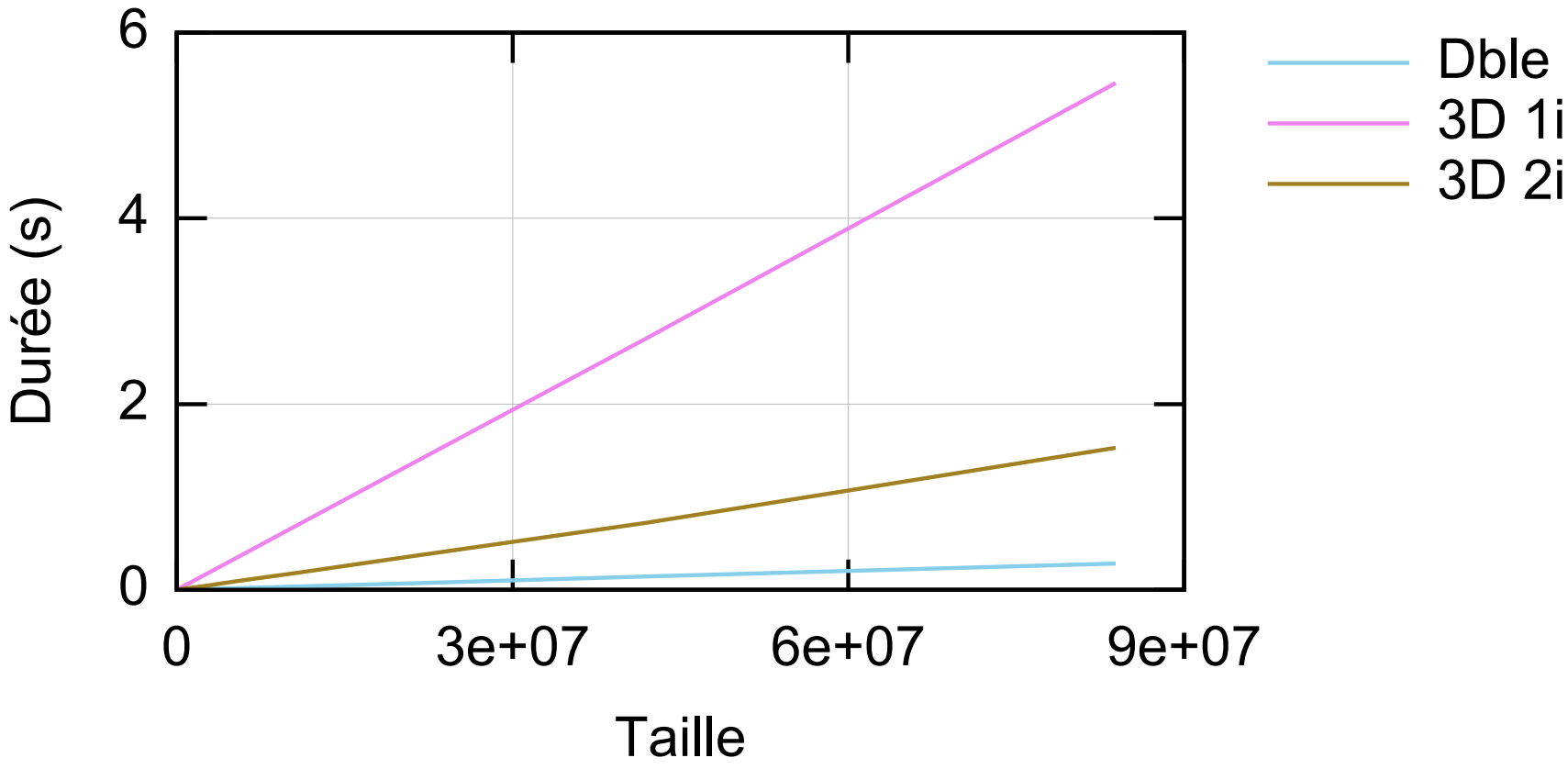
- Les nouveaux types sont compatibles avec toutes les routines de communication (collective,point à point)

CADNA MPI : Les Types stochastiques(2)



- Rapport correct (4) entre float et float_st

CADNA MPI : Les Types stochastiques(3)



- Ajout d'un padding ds le double_st

```
class double_st {
  double x, y, z;
  int accuracy ;
  int pad ;
}
```

CADNA MPI : Les Opérateurs de réductions

Opérateur de Réduction MPI

Faire des opérations (+,-,min,max) sur des données situées sur différents processeurs

- MPI_CADNA_{ SUM / PROD / MIN / MAX }_{ SP / DP }
- MPI_CADNA_{ MAX_LOC / MIN_LOC }_{ SP / DP }

```
void function(in*,inout*,len,MPI_Datatype *t){
 for(int i=0; i<len;i++)
    inout[i] = in[i] op inout[i]
}

MPI_Op_Create(Mpi_user_function *fonction,
        int commute,MPI_Op *op);
```

Exemple de programme avec CADNA MPI

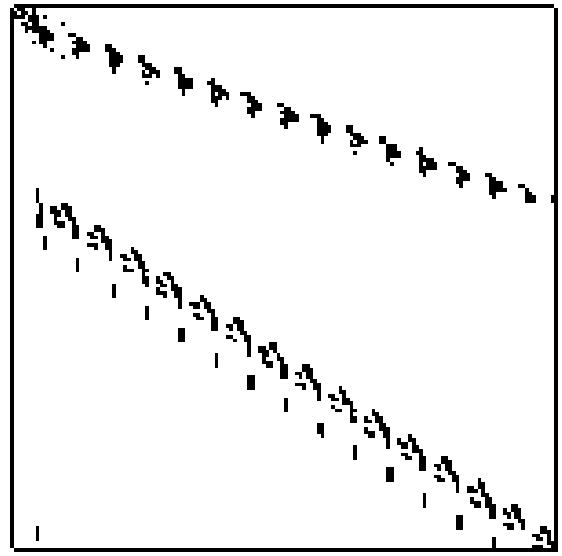
```
#include<cadna_mpi.h>
int main(int argc,char*argv[]){
//Initialisation MPI
cadna_mpi_init(rang,-1);
float_st tabloc[5];

if(rang==0)
 MPI_Send(tabloc,5,MPI_SINGLE_ST,1,tag,WORLD);

if(rang==1)
 MPI_Recv(tabloc,5,MPI_SINGLE_ST,0,tag,WORLD);

MPI_Allreduce(tabloc,tabred,5,MPI_SINGLE_ST,
      MPI_CADNA_SUM_SP,MPI_COMM_WORLD);
cadna_mpi_end(rang);
MPI_Finalise();return 0;
}
```

Résolution parallèle système linéaire avec pivot de Gauss

Matrice 2021*2021

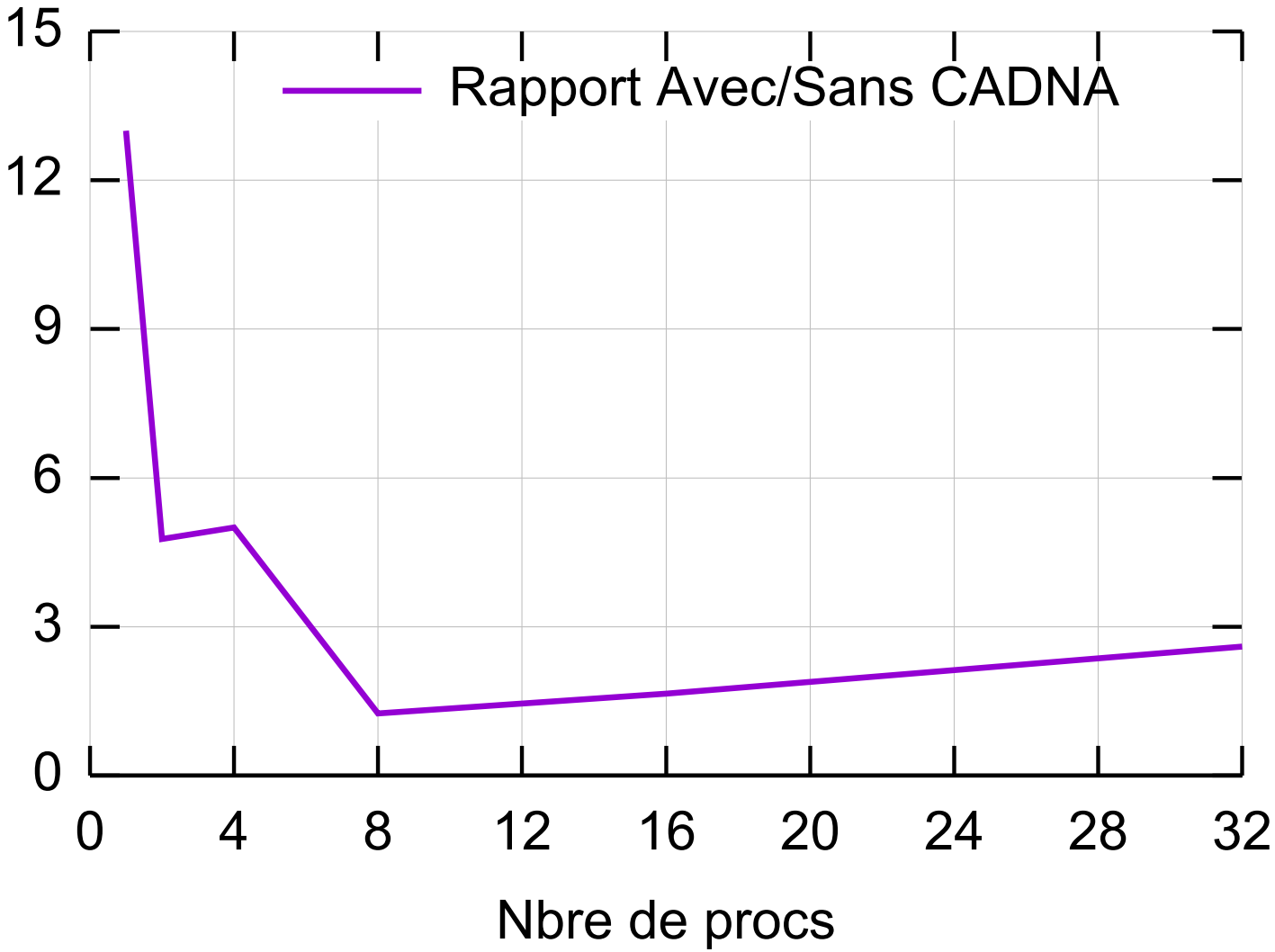
7353 valeurs

Source : Matrix Market (Chemical Engineering)



- Très peu d'instabilités détectées (3)
- Instabilités proviennent de :"-=" (cancellation)

Résolution parallèle système linéaire avec pivot de Gauss (2)



BLACS Basic Linear Algebra Communication Subprograms

Bibliothèque de communication permettant, sur une grille prédéfinie de processus et un contexte précis :

- d'échanger des blocs de matrices entre ces processus
  - rectangulaire M, N, LDA
  - trapezoidale UPLO, DIAG
- de les diffuser globalement sans utiliser les tags ou msgid (ils sont générés automatiquement)
- de calculer sur eux des réductions (min, max, somme).

Ecrit en C , interface f77 compatible f90

Extension de BLACS pour CADNA : BLACS CADNA

Routines de communication BLACS :

{I,S,D,C,Z}{GE,TR}{SD,BS,RV,BV}2D

```
void Cigesd2d(...)
STRBS2D(........)
```

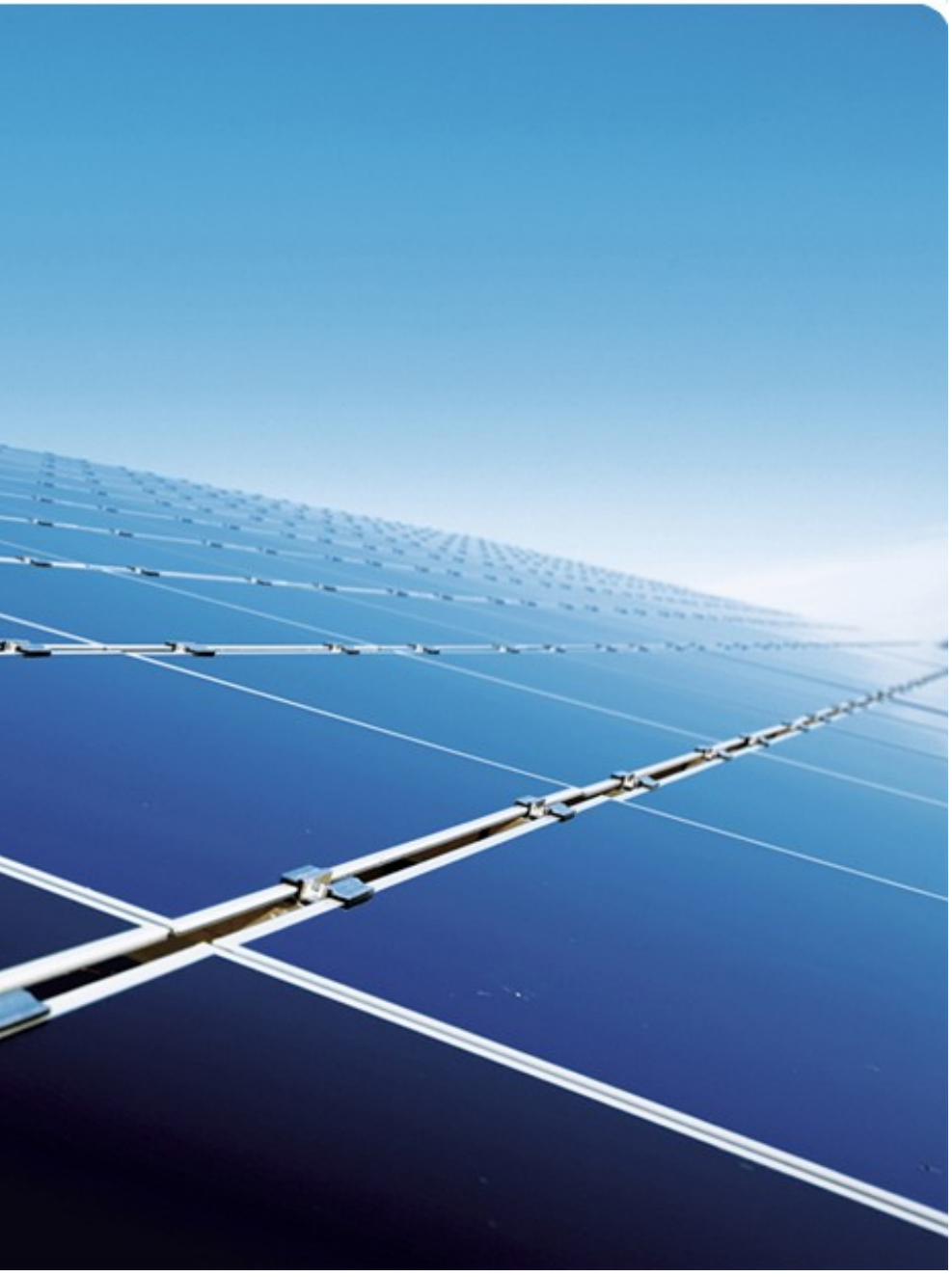
- BLACS développées en C et interface Fortran à partir du meme code source
- CADNA : deux versions Fortran et C++

soucis de compatibilité

Solution :

- Utiliser des void* pour le buffer d'envoi
- Ajouter un paramêtre MPI_Datatype, MPI_Op

Routines d'envoi de matrices stochastiques



3. Les bibliothèques de calcul

Comment peut-on valider les routines de calcul ?

En introduisant directement les types CADNA :

- Remplacement de *float* par *float_st* et *double* par *double_st*
- Surcharge de toutes les opérations arithmétiques

```
void cblas_daxpy (const int N, const ↩
    double_st alpha, const double_st *X, ↩
    const int incX, double_st *Y, const int ↩
    incY)
```

Comment peut-on valider les routines de calcul ?

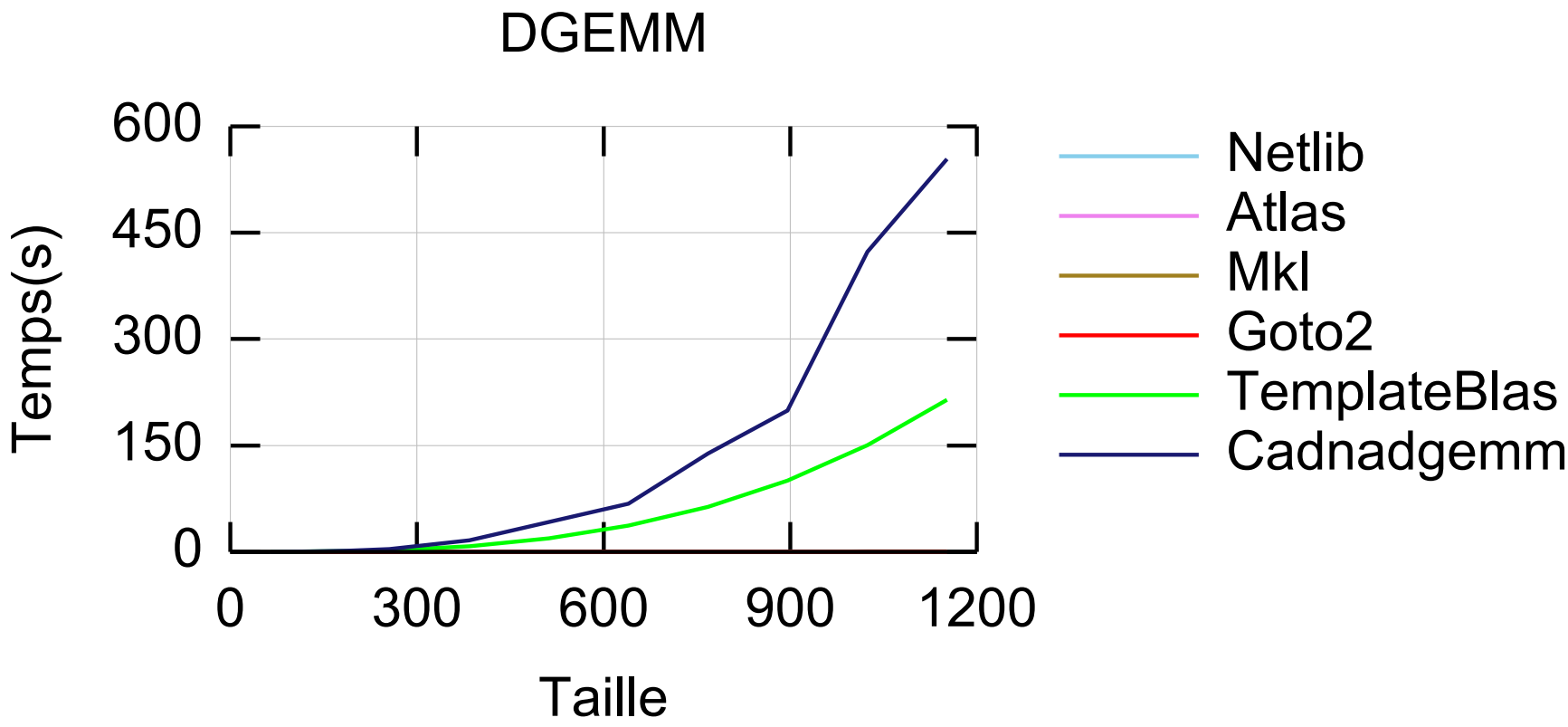
En introduisant directement les types CADNA :

- Remplacement de *float* par *float_st* et *double* par *double_st*
- Surcharge de toutes les opérations arithmétiques

```
void cblas_daxpy (const int N, const ↩
    double_st alpha, const double_st *X, ↩
    const int incX, double_st *Y, const int ↩
    incY)
```

- **Cette implémentation est-elle efficace ?**

Implementation directe de DGEMM avec CADNA



taille	CadnaDgemm	TemplateBlas	Netlib	Goto2
512	42.29	18.8	0.05	0.005
1024	423.43	150.4	0.42	0.049

Implementation directe de DGEMM avec CADNA(2)

Sûrcout de 1000 pour le produit matriciel

- DGEMM avec 3 boucles : constamment en défaut de cache
- Changement de mode d'arrondi vide le pipeline des instructions
- Surcharge des opérateurs arithmétiques +/*
- Mode autovalidation et Détection des pertes brutales de précisions

Implémentation efficace de DGEMM CADNA

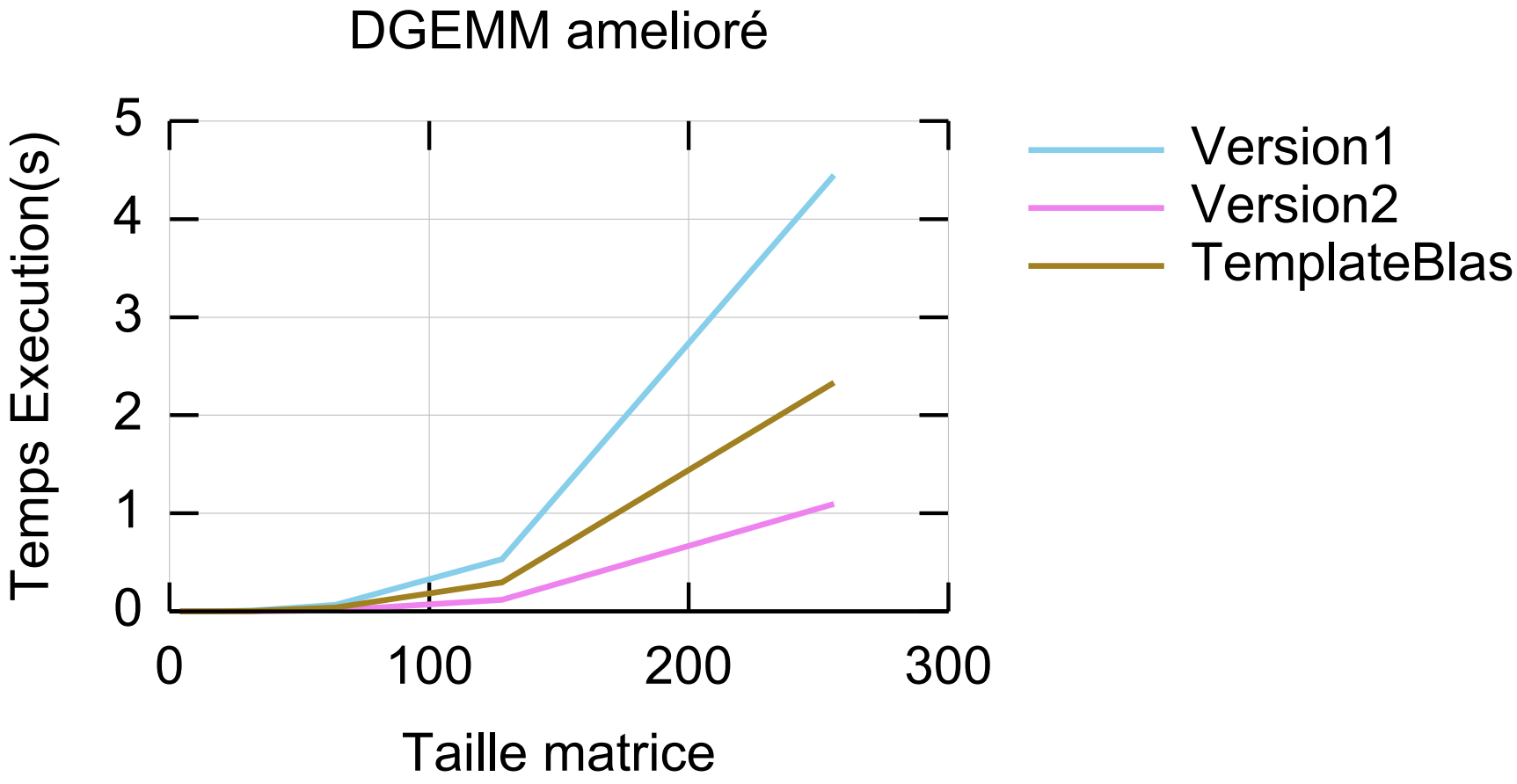
Déroulement des boucles cadna : **pas** de surcharge d'opérateur

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

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

Implémentation efficace de DGEMM CADNA (2)

Déroulement des boucles cadna : **pas** de surcharge d'opérateur



Implémentation efficace de DGEMM CADNA (3)

Appel moins récurent à 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 ;
```

Implémentation efficace de DGEMM CADNA (4)

- Réorganisation des algorithmes
  - par bande
  - par blocs
- Exploitation Cache L1
- Utilisation SSE

4. Conclusion

Conclusion et Perspectives

CADNA MPI / BLACS

- Echange de données de types stochastiques (matrices et tableaux)
- Opérateur de réduction
- Disponible en C / C++ (interface Fortran 90)

Perspectives

- ~~CADNA MPI / BLACS~~
- BLAS
- Mise en application sur TELEMAC 3D (Code EDF)

Merci :) !

sethy.montan@edf.fr