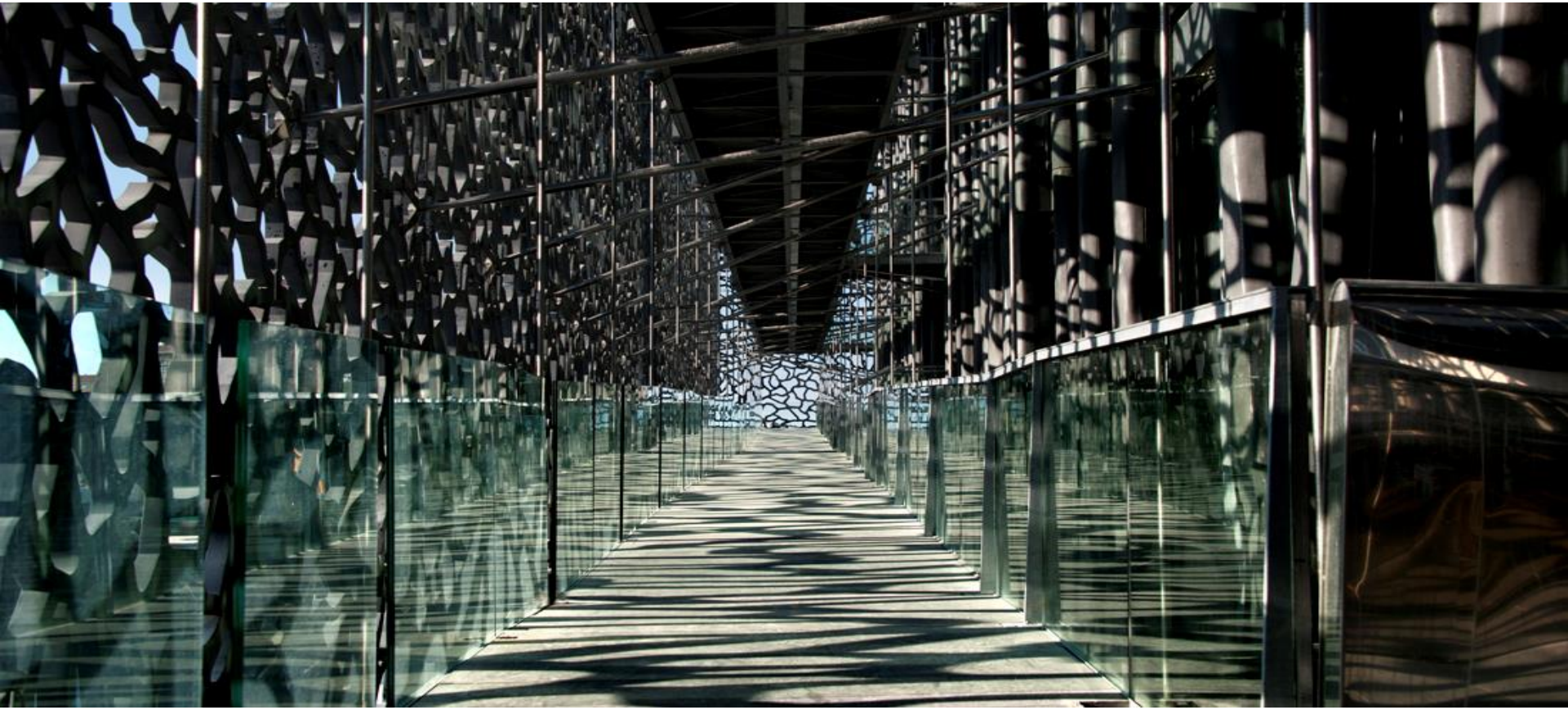
Nouveautés sur les noyaux d'algèbre linéaire de LinBox



[static.mucem.org : Lisa Ricciotti]

Brice Boyer, Jean-Guillaume Dumas, Pascal Giorgi, Thomas Izard, Clément Pernet, Dave Saunders, Ziad Sultan

JNCF Luminy Novembre 2014

LinBox

Algorithmes (det., rang, pol. car., formes normales …)

FFLAS-FFPACK

Alg. Lin. Dense exacte

GIVARO

Corps finis

BLAS

Alg. Lin. Dense numérique

GMP

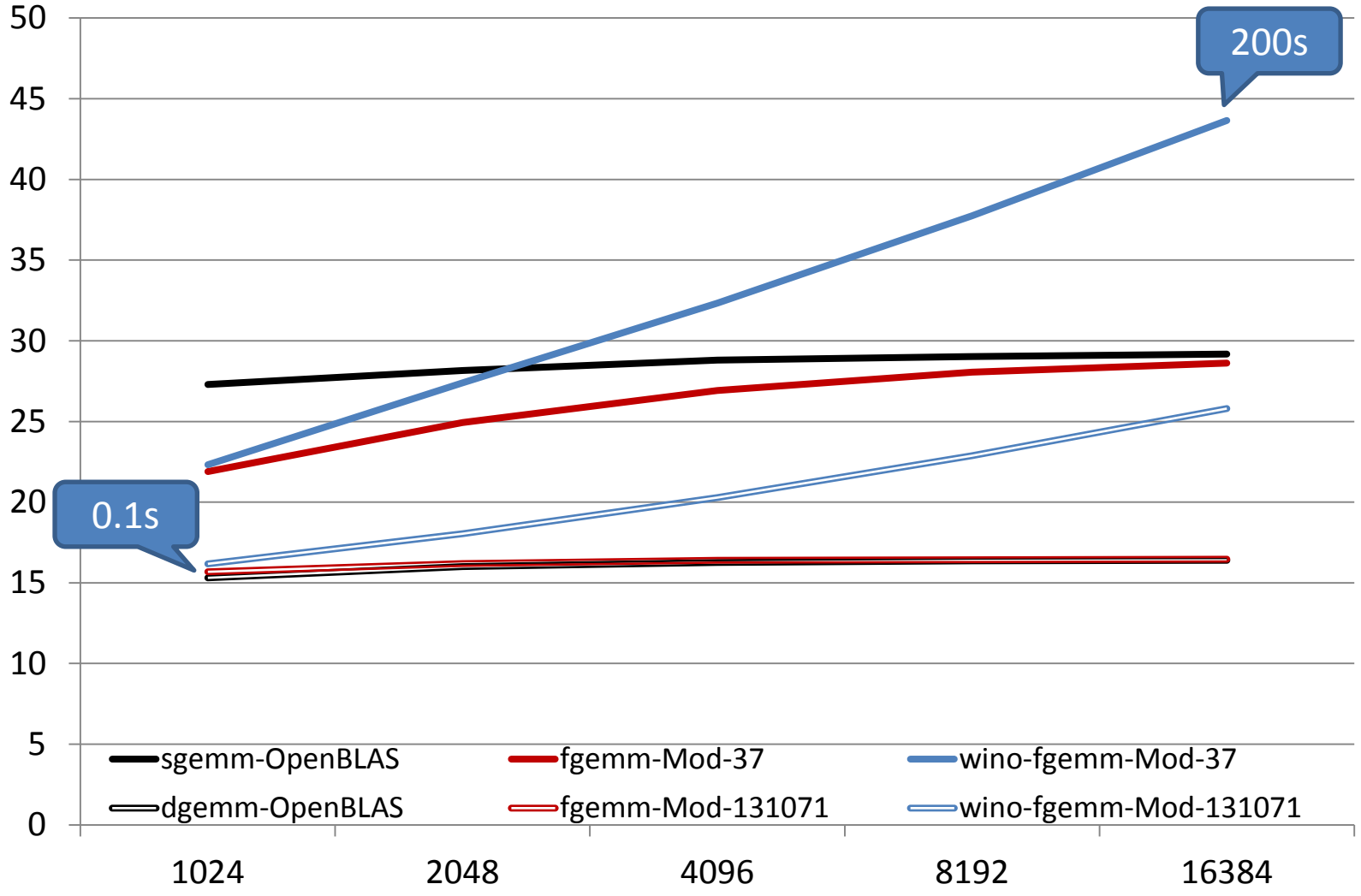
Entiers/Rat. multiprécision

Paradigme FFLAS des 10 dernières années

- Arithmétique flottante en avance
  - SSE, SSE3, SSE4, AVX, AVX2
- Division entière lente
- Produit scalaire modulo p
  - $[a_1, \ldots, a_n] \cdot [b_1, \ldots, b_n]^T$ modulo p
  - $np^2 < 2^{\text{mantisse}}$ …

⇨ Arithmétique flottante pour le calcul exact

Vitesse de la multiplication de matrices exacte (Gffops normalisés=$2n^3/time/10^9$) sur 2.8Ghz



ChangeLog fflas-ffpack-2.0

- SIMD au dessus des BLAS numériques denses
- Précision arbitraire FFLAS_mp
- SparseFFLAS au dessus de SparseBLAS
- Wrapper pour Matlab (en cours ...)
- Parallélisme : pFGEMM, pFTRSM, pPLUQ

3000x3000 mod 7
g++-4.9.1 sur Xeon 2.2Ghz

Temps en secondes	finit == réduction modulaire		
	défaut	SSE4.1	AVX
Modular<float>	0,101	0,009	**0,007**
Modular<double>	0,102	0,019	**0,012**
ModularBalanced<float>	0,110	0,011	**0,006**
ModularBalanced<double>	0,115	0,029	**0,012**

[van der Hoeven, Lecerf, Quintin 2014]

⇨ Réduction du surcout en petites tailles

⇨ Réduction du surcout restes chinois ...

ChangeLog fflas-ffpack-2.0

- SIMD au dessus des BLAS numériques denses
- Précision arbitraire FFLAS_mp
- SparseFFLAS au dessus de SparseBLAS
- Wrapper pour Matlab (en cours ...)
- Parallélisme : pFGEMM, pFTRSM, pPLUQ

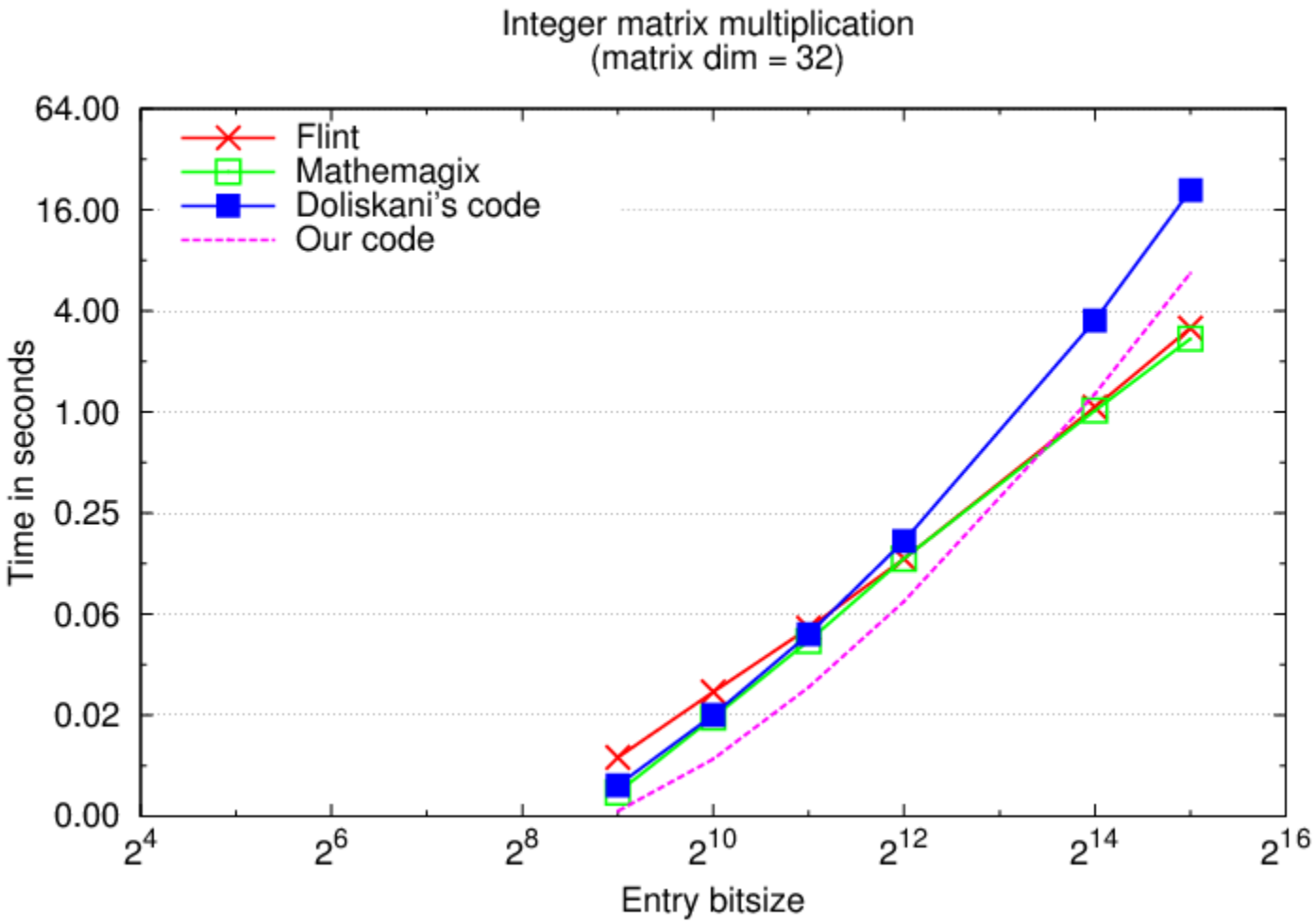
Précision arbitraire FFLAS_mp *[Giorgi]*

- p en grande précision
- Par restes chinois modulo $[m_1,\ldots,m_k]$
  - FGEMM_mp : $O(kn^w + n^2 I(k))$
  - En pratique sur moyenne taille de p $\approx O(kn^w + n^2k^2)$

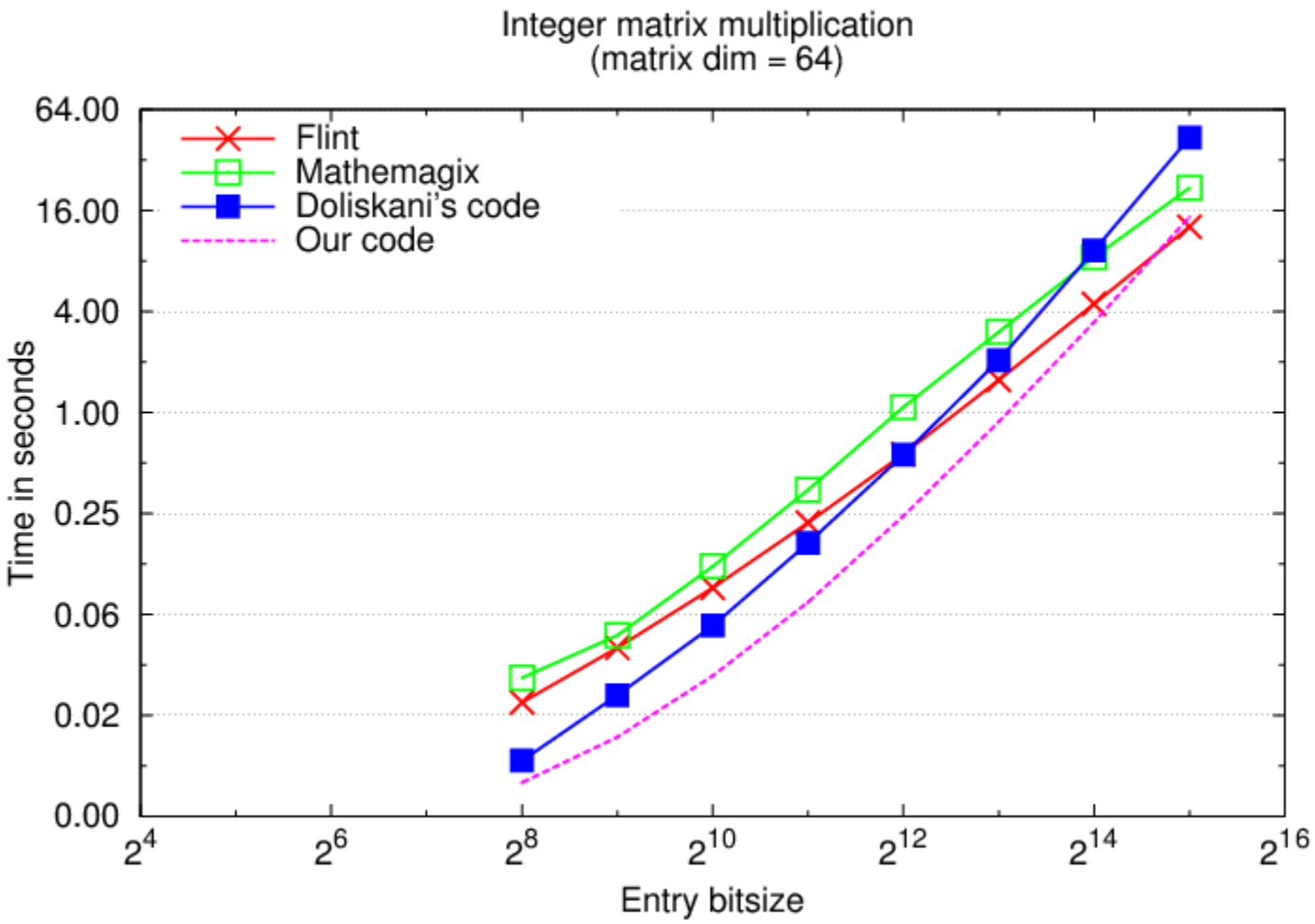
- Marix-mul : (mod p) mod m_i, simultanément
  1. $C = A \times B$ sur les résidus (i.e. sur $\mathbb{Z}$)
  2. Remontée sur $\mathbb{Z}_{/p\mathbb{Z}}$ par multiplication de matrices
  3. Réduction modulo les m_i par finit

  ⇨ $O(kn^w + n^2k^{w-1})$

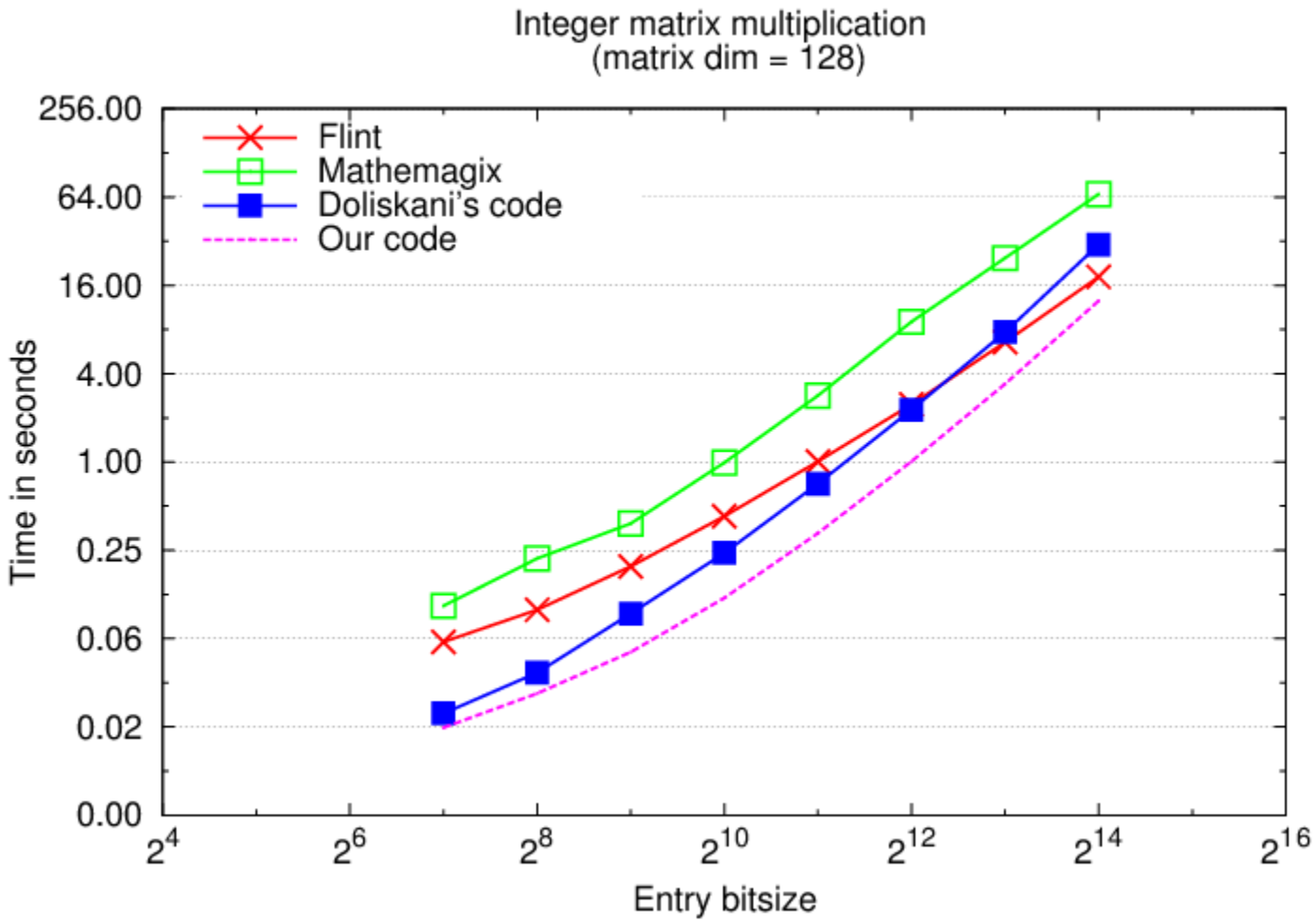
Multiplication de matrices entières



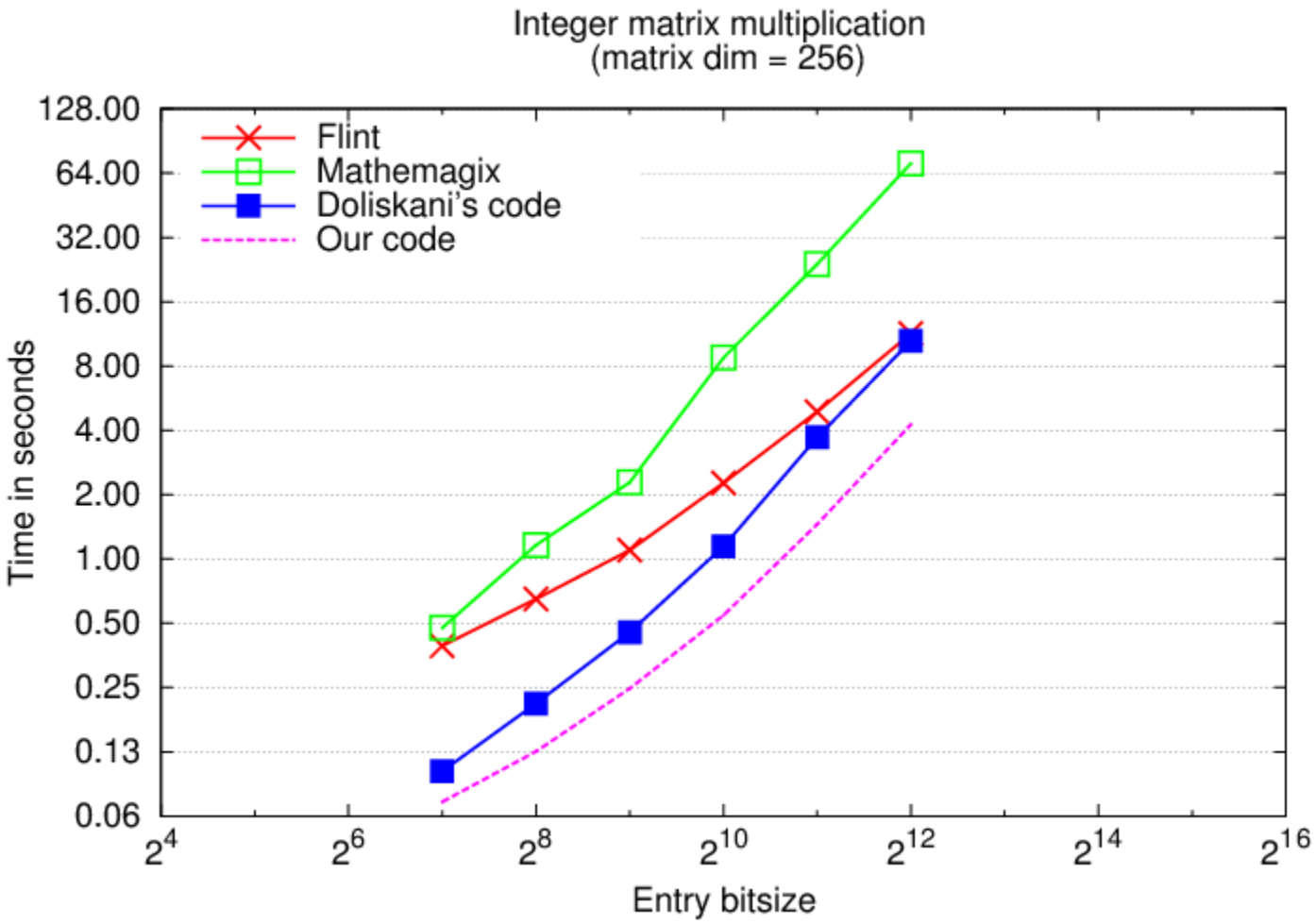
Multiplication de matrices entières



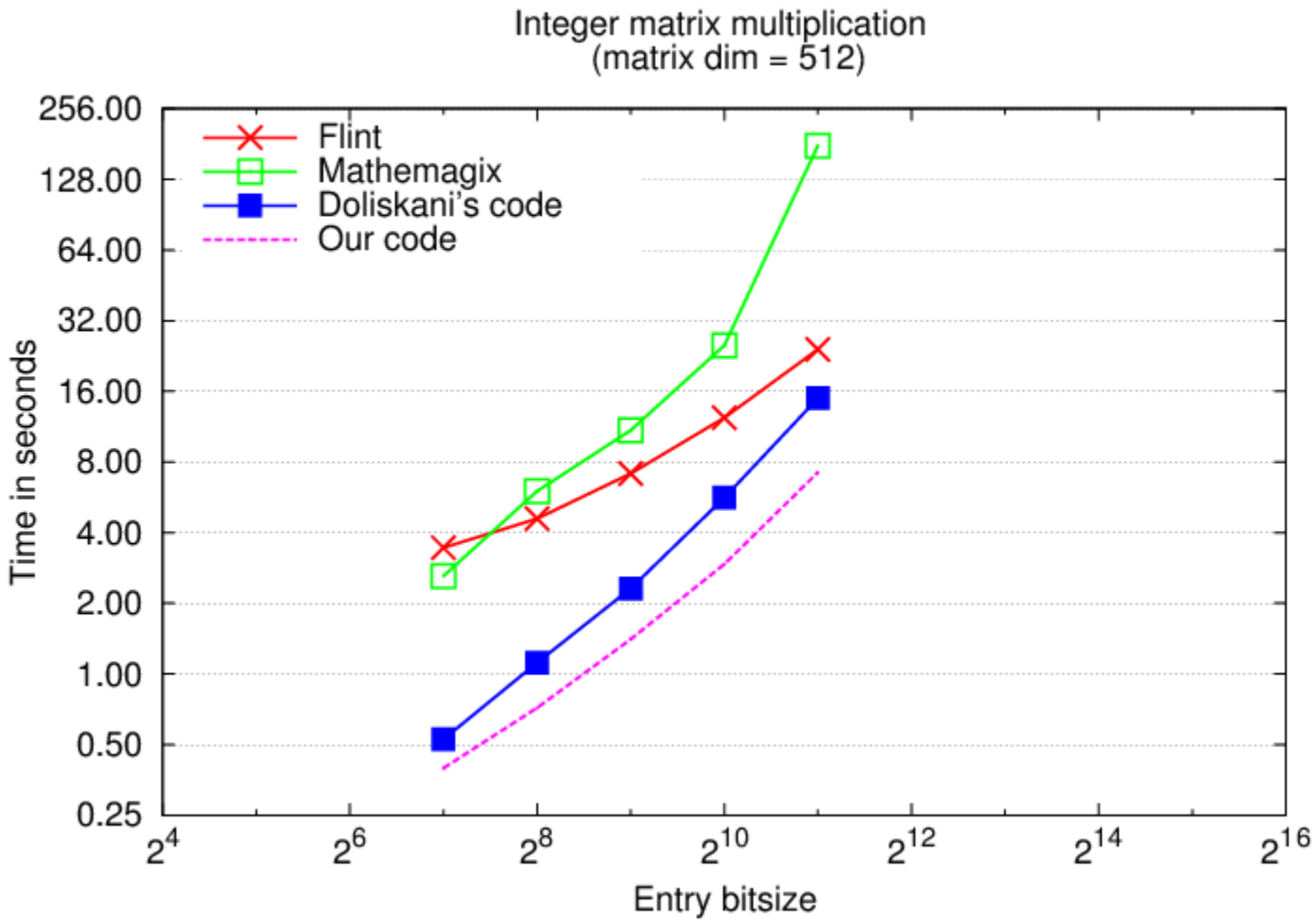
Multiplication de matrices entières



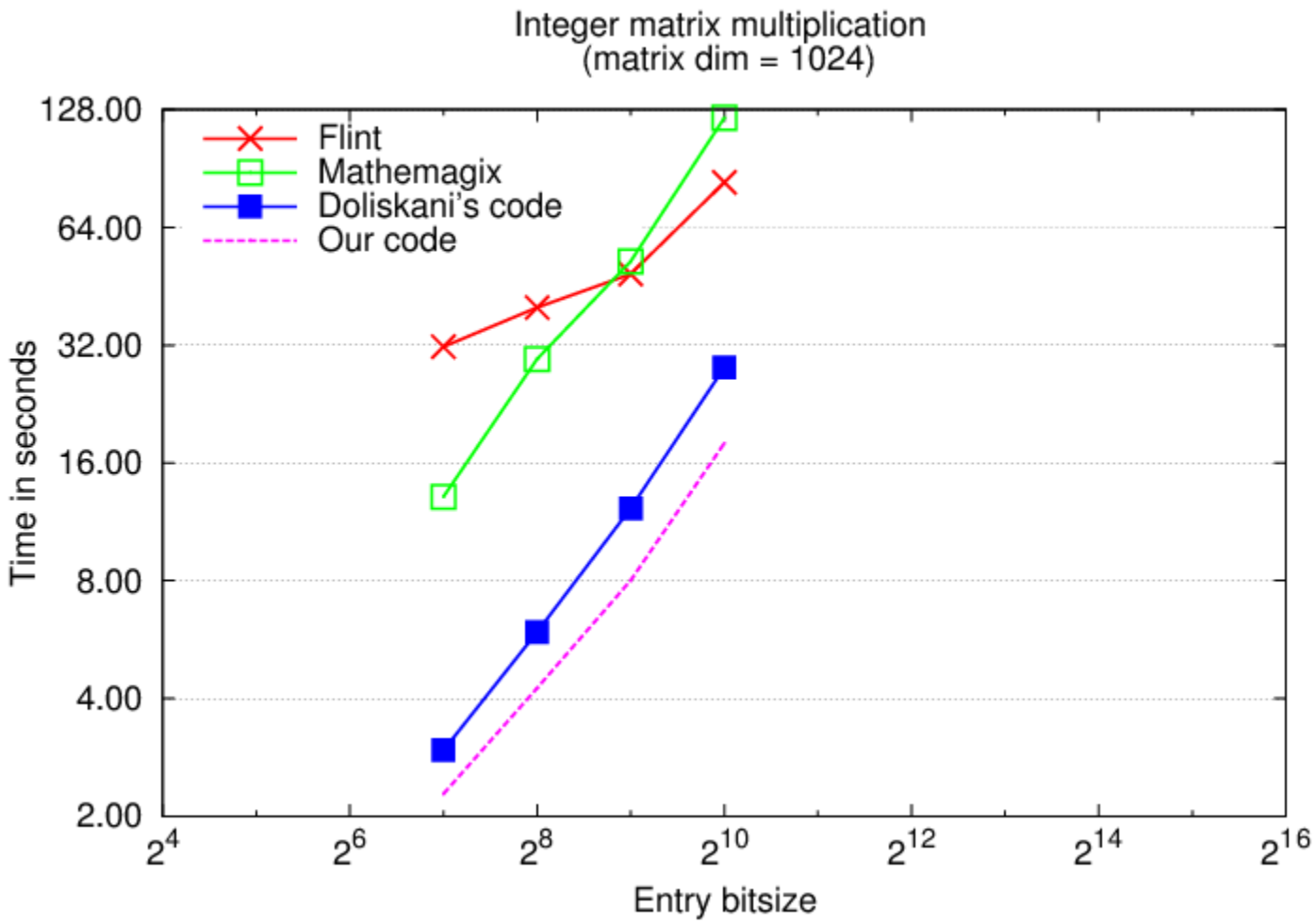
Multiplication de matrices entières



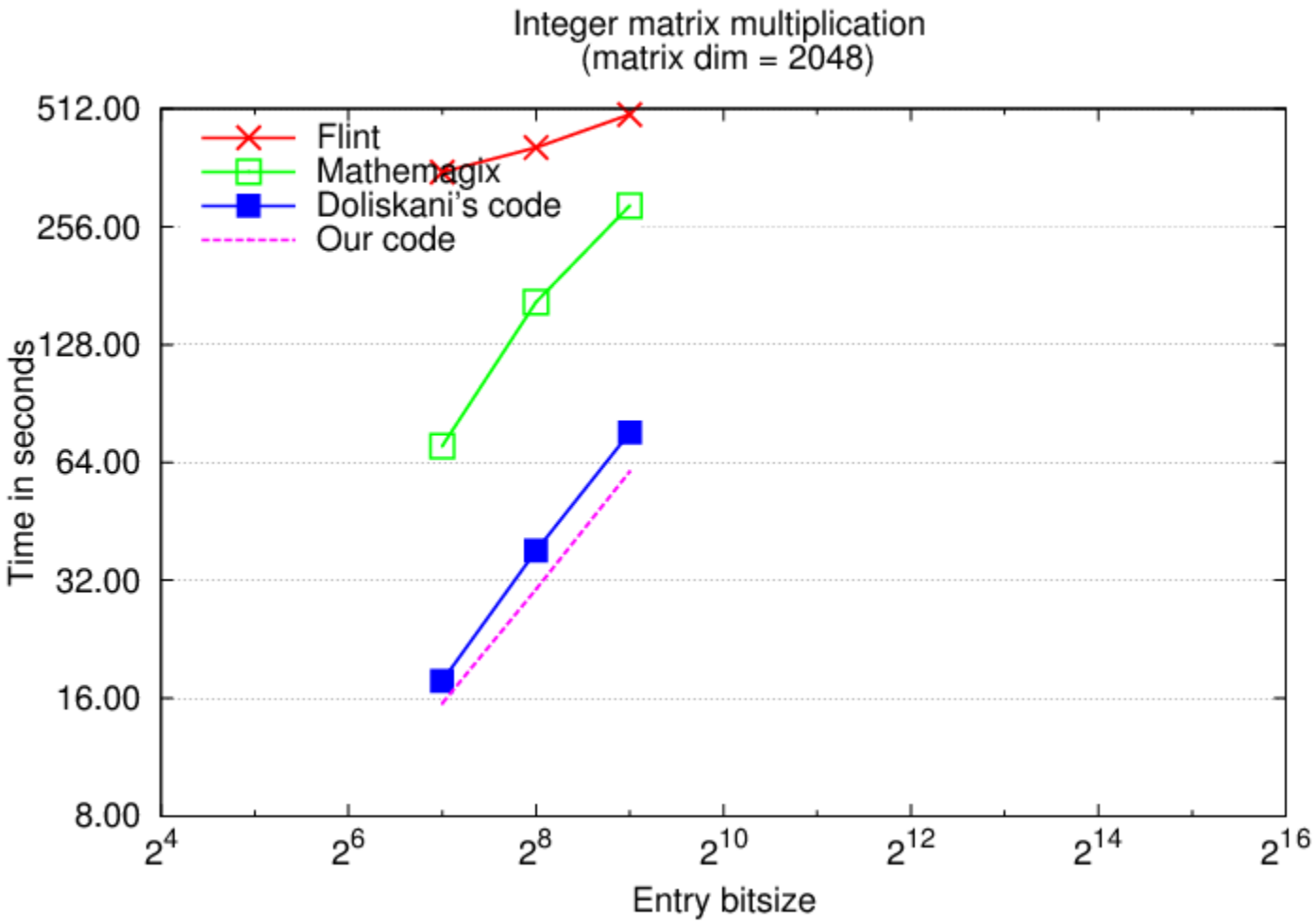
Multiplication de matrices entières



Multiplication de matrices entières



Multiplication de matrices entières



ChangeLog fflas-ffpack-2.0

- SIMD au dessus des BLAS numériques denses
- Précision arbitraire FFLAS_mp
- SparseFFLAS au dessus de SparseBLAS
- Wrapper pour Matlab (en cours ...)
- Parallélisme : pFGEMM, pFTRSM, pPLUQ

FGEMV sur matrice creuse

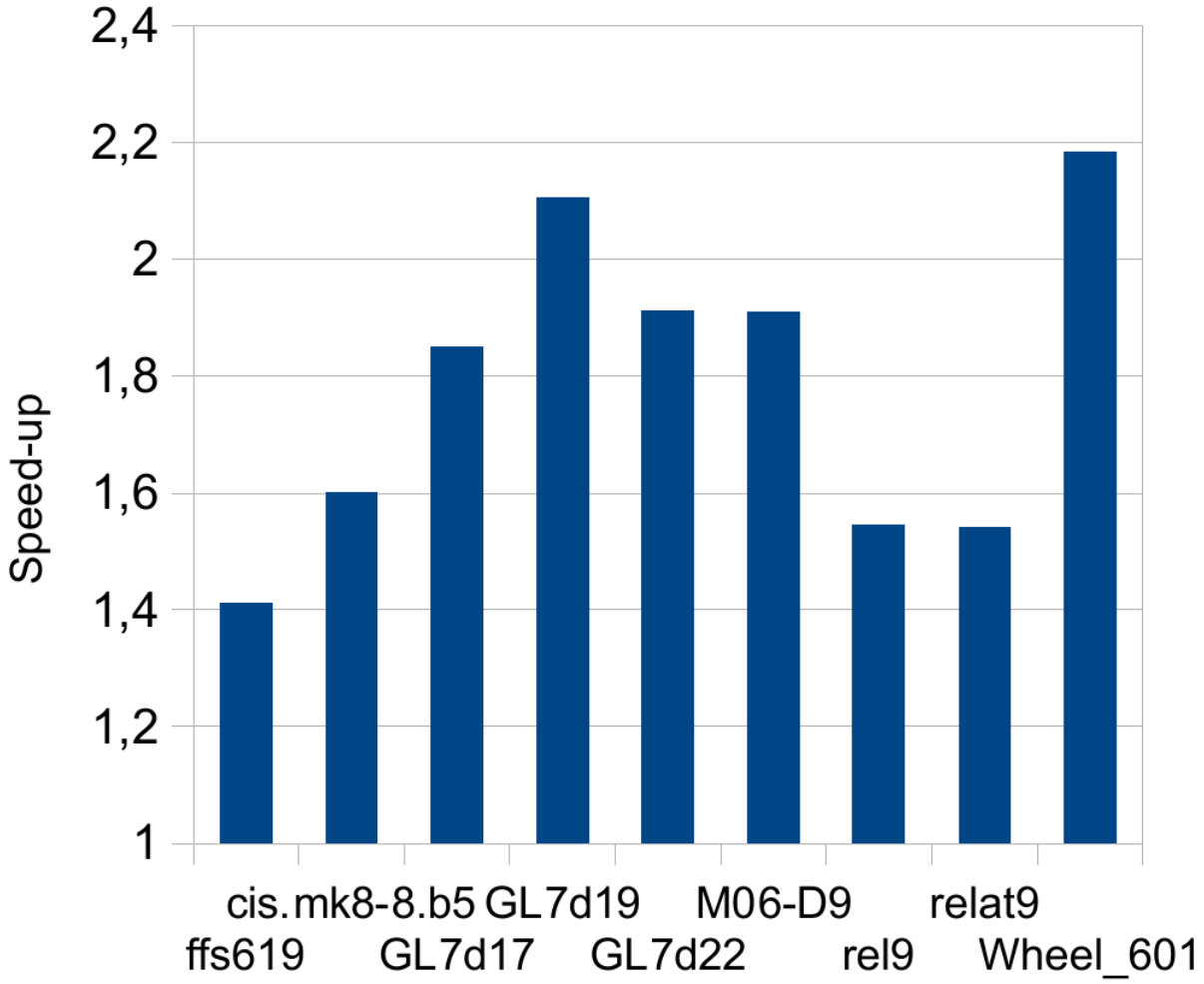
- Paradigme FFLAS aussi sur matrices creuses
  - $[0,\ldots,0,a_1,0,\ldots,a_k,0,\ldots,0] \cdot [b_1,\ldots,b_n]^T$ modulo p
  - $kp^2 < 2^{\text{mantisse}}$

⇨ Arithmétique flottante pour le calcul exact creux
  - Utilisation des SparseBLAS

[Boyer,Vialla]

sp_FGEMV

[Giorgi,Vialla ICMS'14]



ChangeLog fflas-ffpack-2.0

- SIMD au dessus des BLAS numériques denses
- Précision arbitraire FFLAS_mp
- SparseFFLAS au dessus de SparseBLAS
- Wrapper pour Matlab (en cours ...)
- Parallélisme : pFGEMM, pFTRSM, pPLUQ

Rump et IntLab

[Rump 2010]

- Test d'inversibilité en Matlab/Intlab

```
1. R = inv(A);                          % approximate inverse
2. setround(-1);                        % rounding downwards mode
3. C1 = R*A-eye(n);                     % lower bound for RA-I
4. setround(1) ;                        % rounding upwards mode
5. C2 = R*A-eye(n);                     % upper bound for RA-I
6. C = max(abs(C1),abs(C2));            % upper bound for |RA-I|
7. c = norm( C , inf );                 % upper bound for ||RA-I||∞
```

⇨ Si $||RA-I||_\infty<1$ l'inversibilité de toutes les matrices dans un voisinage de A est prouvée (Ax=0⇨(RA-I)x=x)

⇨ Complexité : 3 MM(n) ≈ 9 PLUQ(n)

⇨ Sinon ???

Certificat d'inversibilité exact

- Transformer la matrice flottante en rationnels
  - [fA,eA] = log2(A); % $A = 2^{53} fA \cdot 2^{eA-53}$
  - ⇨ $(2^{53} fA)$ et eA sont entiers
- Choisir un nombre premier p aléatoirement
  - Prendre A mod p
    - Si A inversible : det(A mod p) ≠ 0 avec une grande probabilité, complexité : 1 PLUQ(n) = $O(n^w)$
    - Si A non-inversible : det(A mod p) = 0
    - Preuve de singularité : remonter sur $\mathbb{Z}$, $n^{3+\circ(1)}$

Certificat d'inversibilité

Matrice	Matlab approx.	IntLab intervalles	LinBox Mod p
100×100	0.00	0.02	0.01
200×200	0.01	0.10	0.03
300×300	0.04	0.31	0.03
400×400	0.09	0.50	0.06
500×500	0.17	0.77	0.10
1000×1000	0.91	4.49	0.32
1500×1500	1.84	14.05	1.65
2000×2000	3.97	32.64	1.99
2500×2500	8.46	62.74	3.57

- Matrices aléatoires inversibles, sur un Pentium P6100, 2GHz
- Matlab 7.6.0 (R2008a) et LinBox-1.1.6 avec les mêmes BLAS sous-jacents
- LinBox ne garantit pas l'inversibilité autour de A

Interface Matlab pour FFLAS-2.1

- DSL (domain specific language)
- Générateur automatique MEX
  - `Routine = FGEMM`
  - `vars = F:Field, alpha:Element<Element>, beta:Element<Element>, A:Matrix<Element>, B:Matrix<Element>, C:Matrix<Element>, m:rows(op(A)), m:rows(C),n:cols(op(B)), n:cols(C), k:cols(op(A)), k:rows(op(B))`
  - `cxx = (F, NoTrans, NoTrans, m, n, k, alpha, A, ld(A), B, ld(B), beta, C, ld(C)) -> Void`
  - `matlab_cxx = (F, NoTrans, NoTrans, n, m, k, alpha, B, ld(B), A, ld(A), beta, C, ld(C))-> Void`
  - `matlab_user = (F, alpha, A, B, beta, C) -> C`

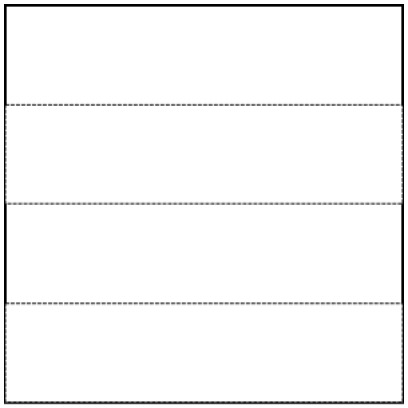
ChangeLog fflas-ffpack-2.0

- SIMD au dessus des BLAS numériques denses
- Précision arbitraire FFLAS_mp
- SparseFFLAS au dessus de SparseBLAS
- Wrapper pour Matlab (en cours ...)
- Parallélisme : pFGEMM, pFTRSM, pPLUQ

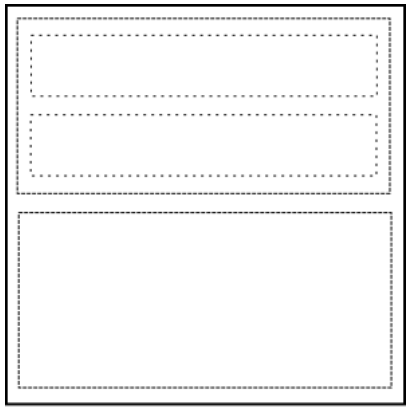
Routines parallèles SMP *[Sultan]*

- Mappage des données sur bancs NUMA
  - Découpage itératif pour un meilleur placement
  - Découpage récursif pour une meilleure adaptation
- Data-flow et vol de travail
  - Petits blocs pour mieux ordonnancer
- Parallélisme classique > Parallélisme Strassen
- Réductions modulaires retardées
  - Gros blocs pour en tirer parti

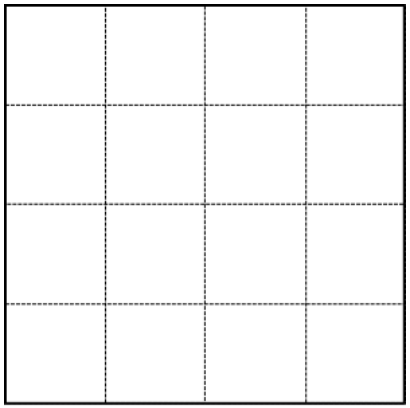
Découpes ...



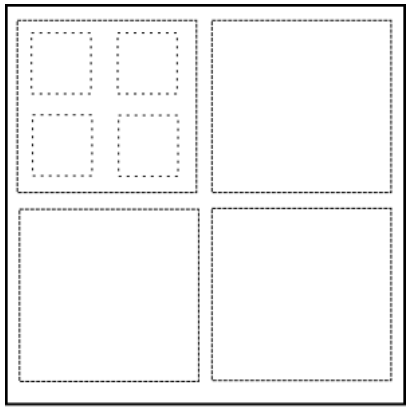
Slab iterative



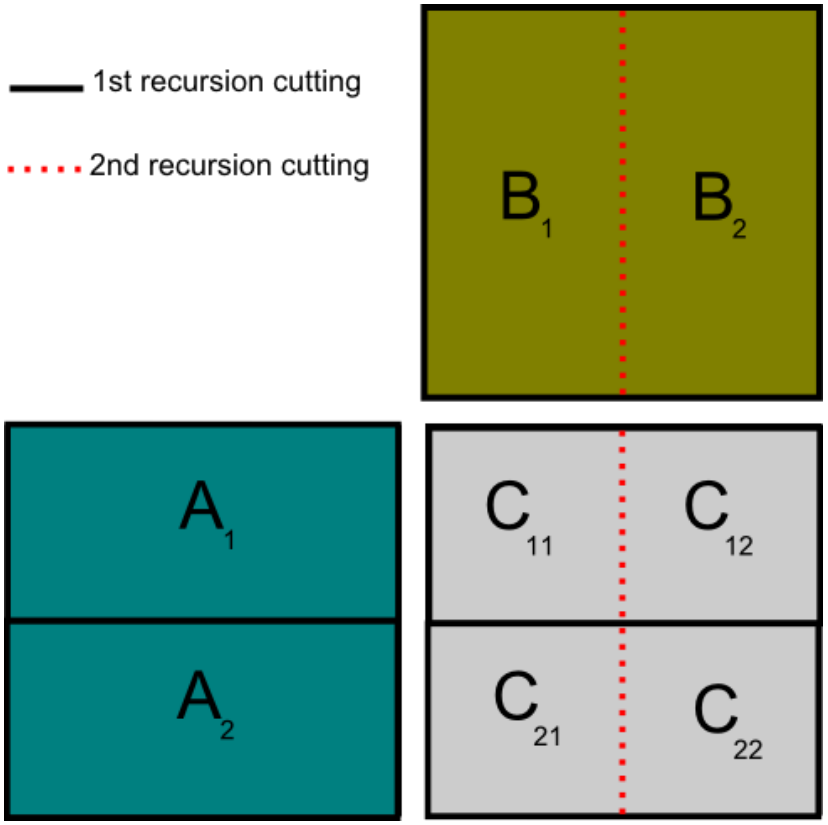
Slab recursive



Tile iterative



Tile recursive

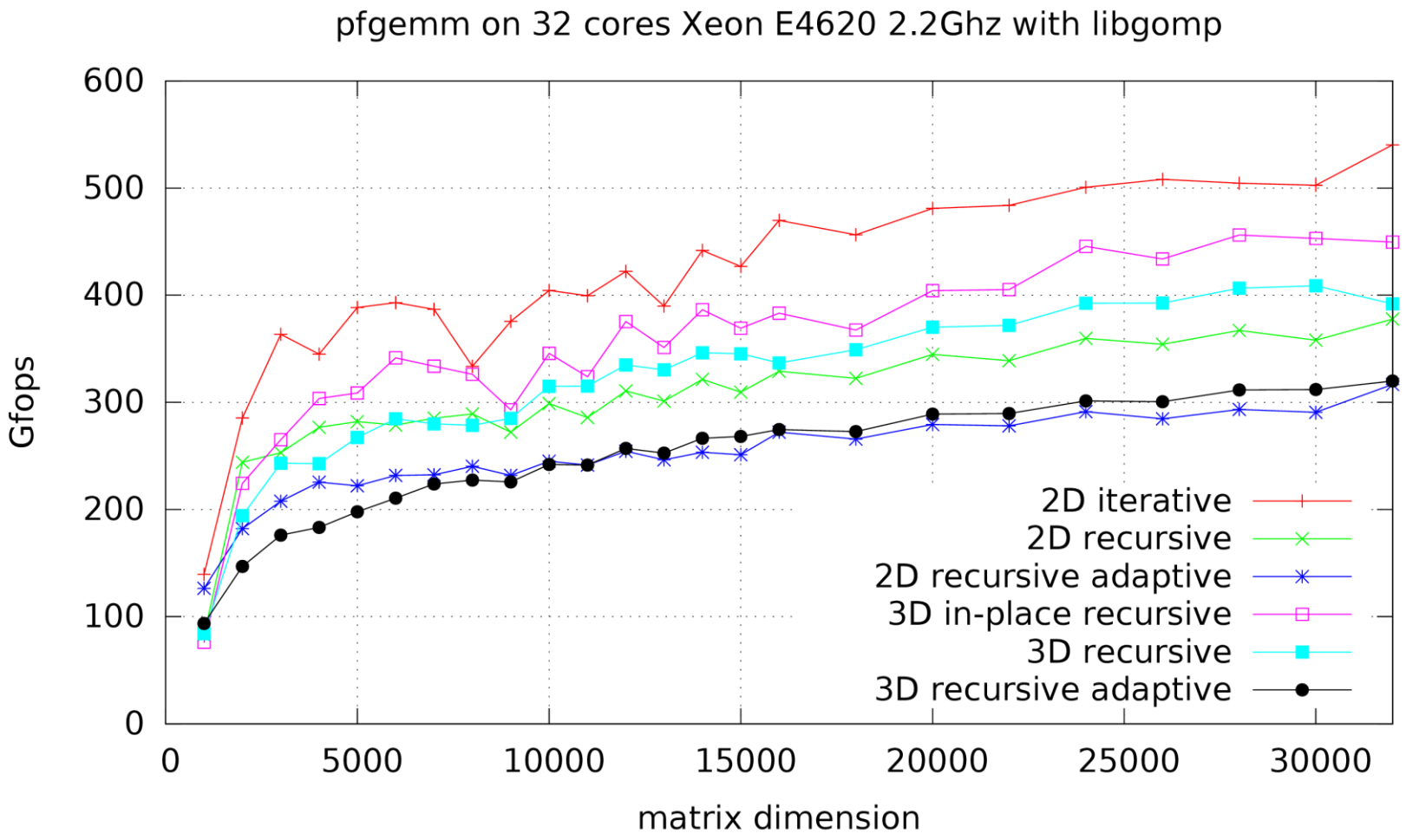


Variantes pFGEMM, OpenMP vs Kaapi

32 cœurs Gffops normalisés $2n^3/time/10^9$

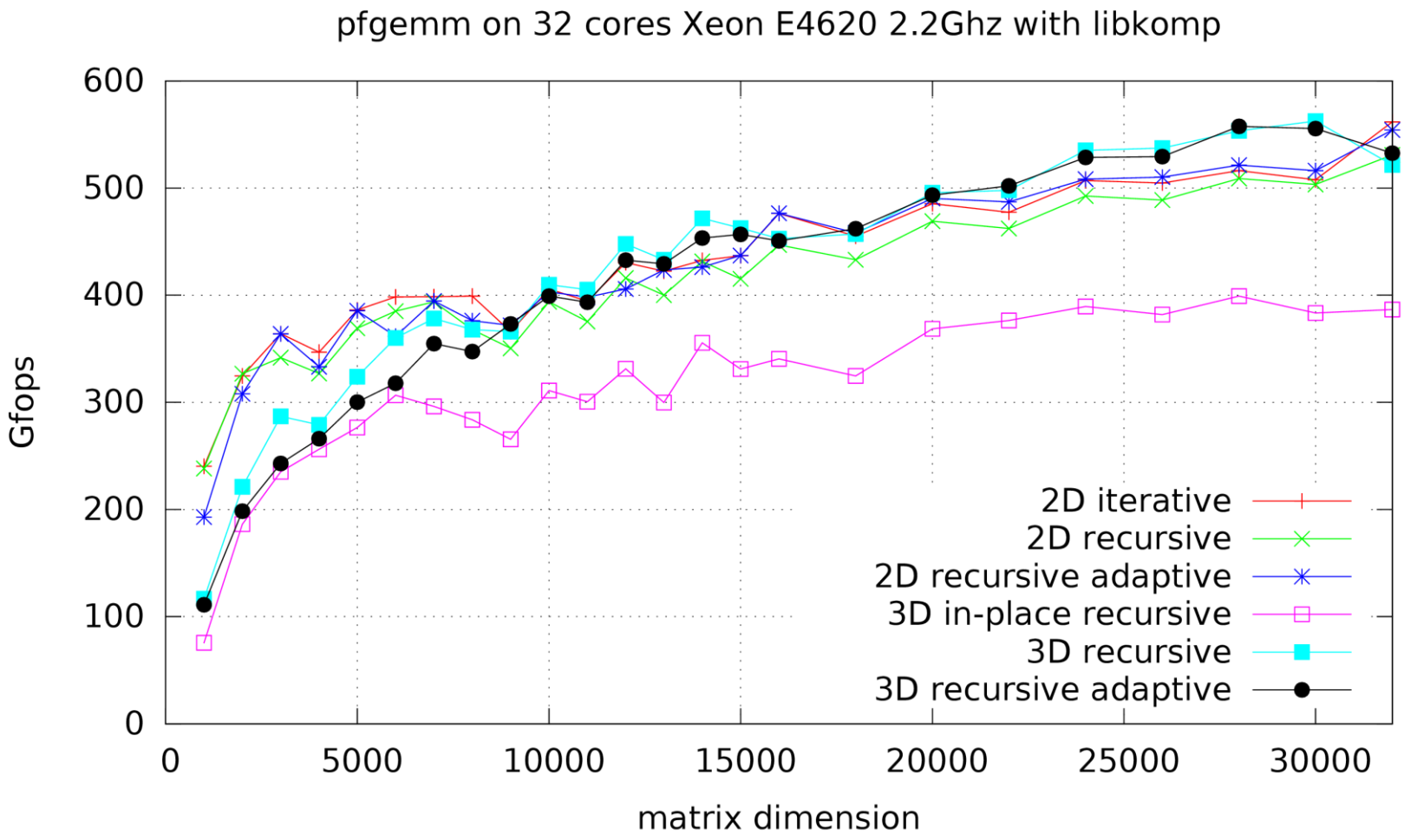
Variantes pFGEMM, OpenMP vs Kaapi

32 cœurs Gffops normalisés $2n^3/time/10^9$



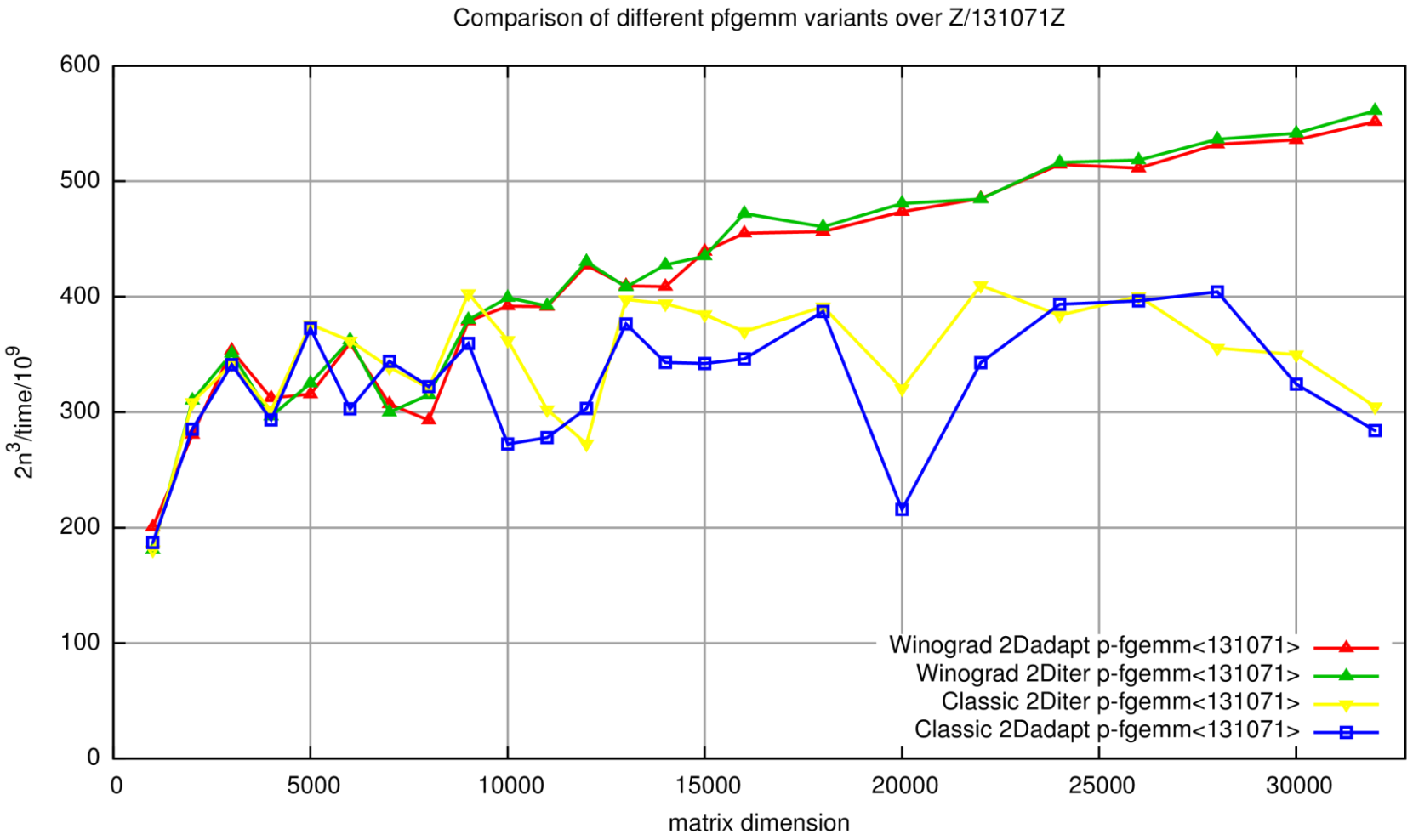
Variantes pFGEMM, OpenMP vs Kaapi

32 cœurs Gffops normalisés $2n^3/time/10^9$



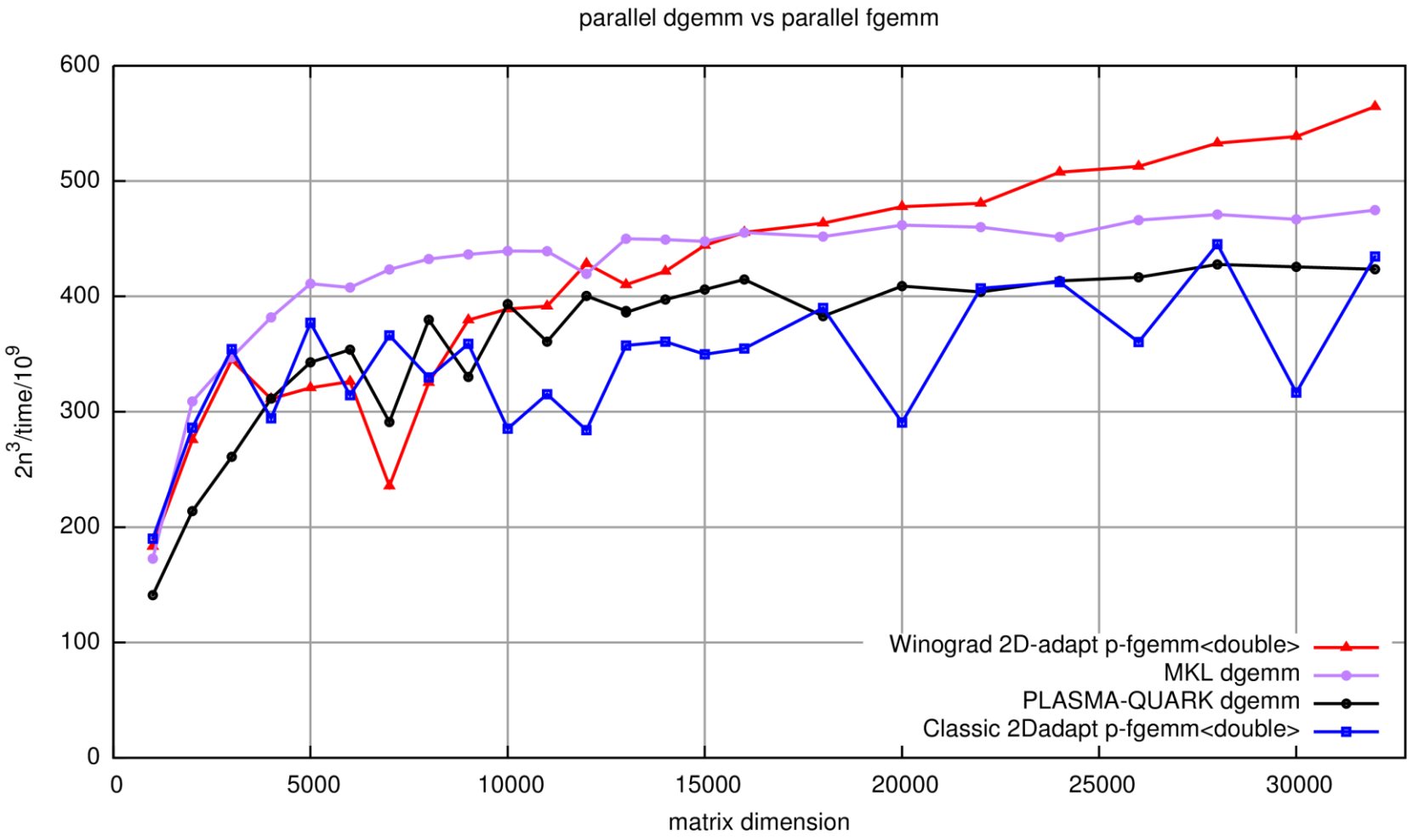
Variantes pFGEMM, impact Strassen

32 cœurs Gffops normalisés $2n^3/time/10^9$

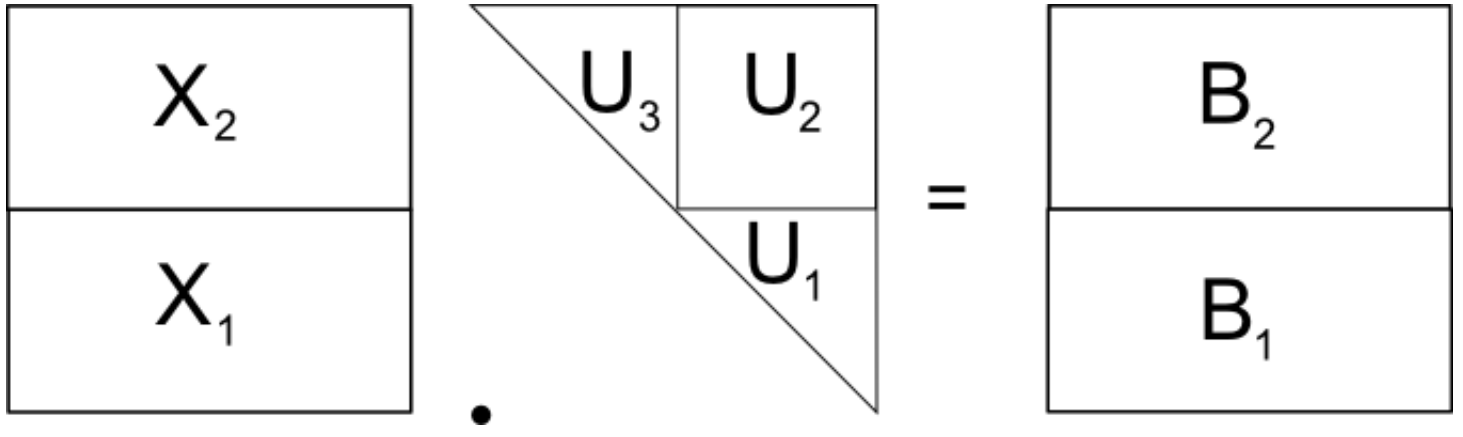


pFGEMM vs MKL vs PLASMA-QUARK

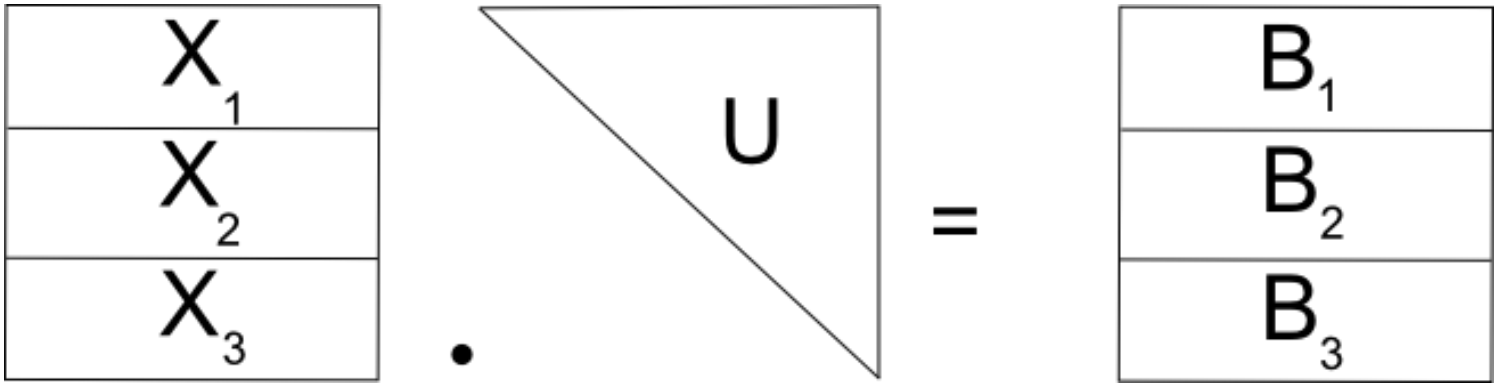
32 cœurs Gffops normalisés $2n^3/time/10^9$



pFTRSM : hybride

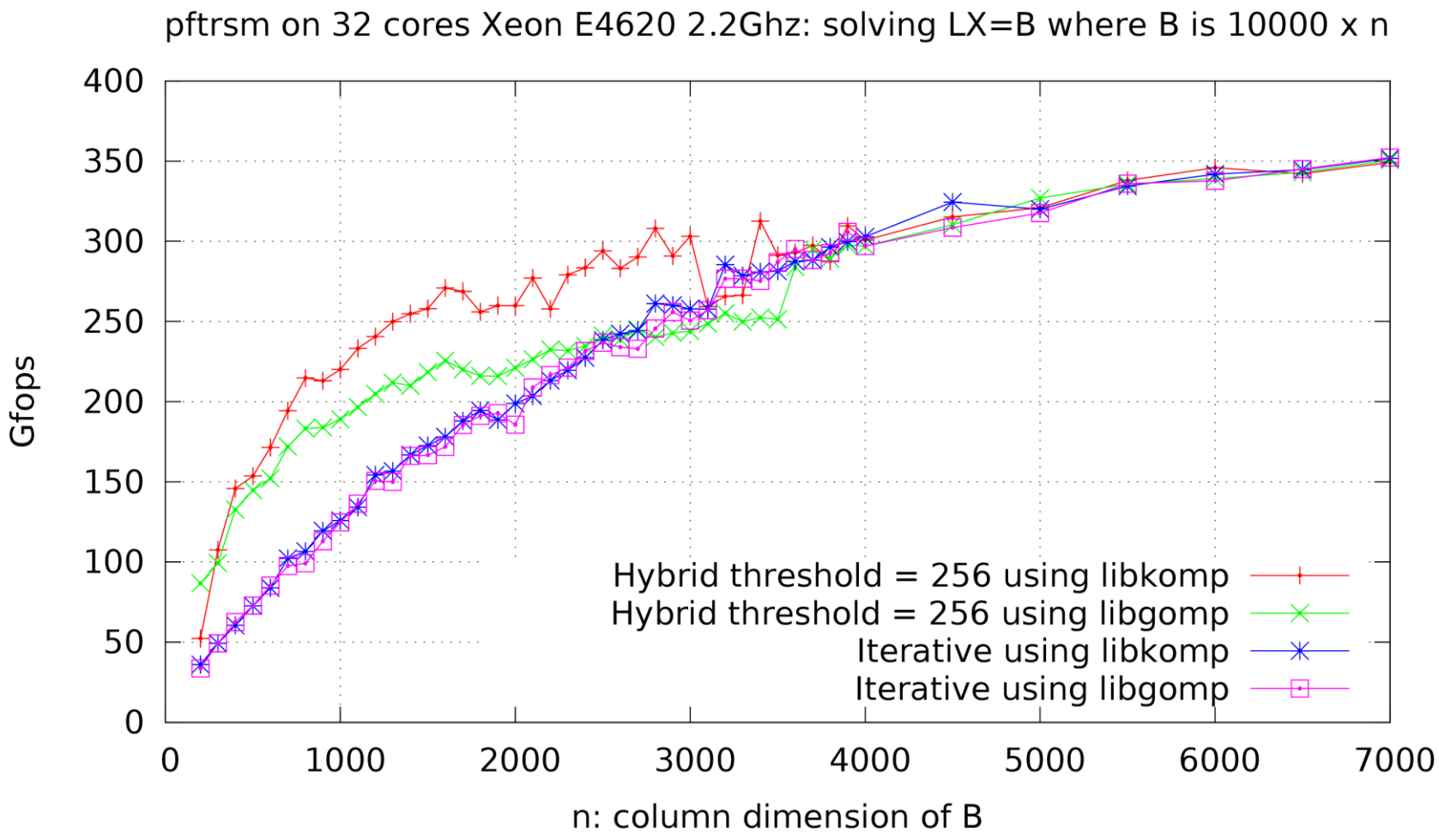


puis



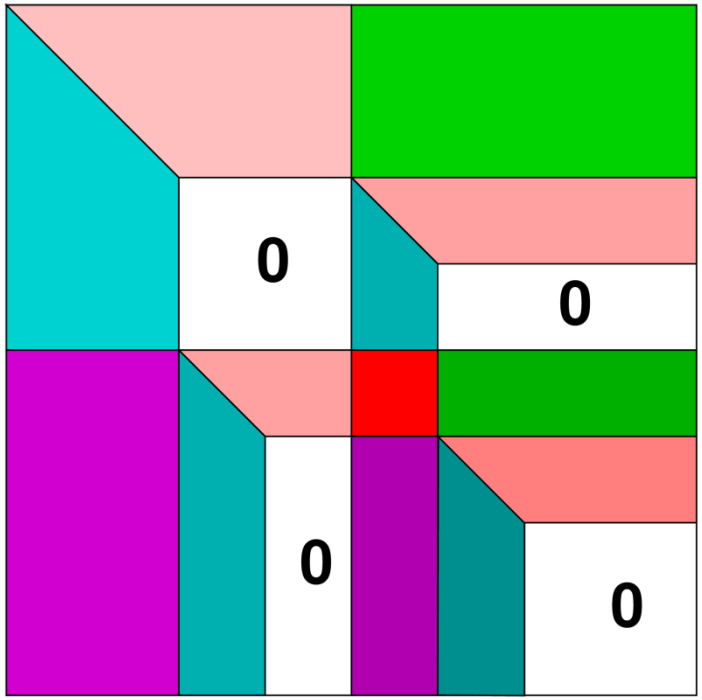
Impact de la taille du second membre

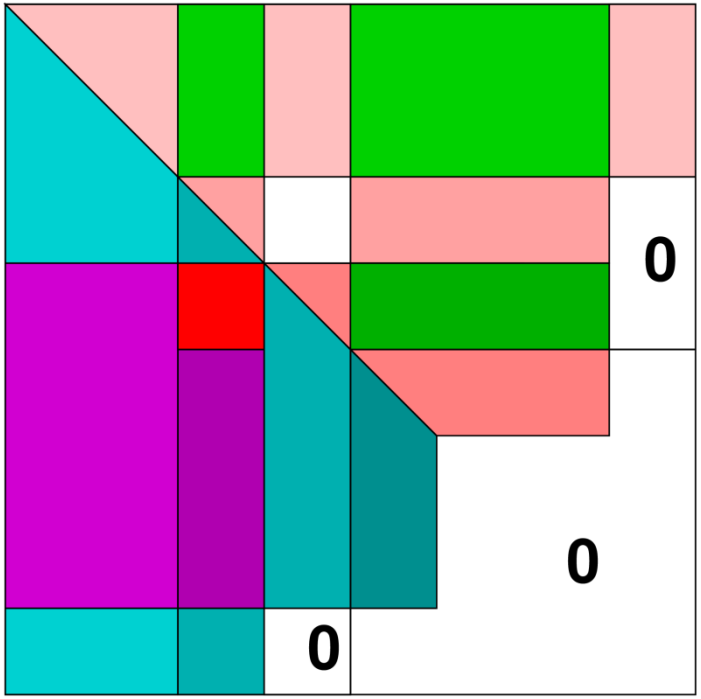
32 cœurs Gffops normalisés $n^3/time/10^9$



pPLUQ [D,Pernet,Sultan EuroPar'14]

quad-recursif mais préserve le profil de rang





⇨ Nécessite du couplage parallèle

- pFGEMM, pFTRSM, DAG de dépendances …

pPLUQ avec pivotage et déficiences potentielles de rang
vs
MKL & PLASMA-Quark sans pivotage
32 cœurs Gflops normalisés $2n^3/3/time/10^9$