Postale : Performance Optimization by Software Transformation and Algorithms & Libraries Enhancement

Marc Baboulin

Inria Saclay et Université Paris-Sud

Laboratoire de Recherche en Informatique

4 juin 2014

Contexte scientifique

Super-calculateur le plus rapide (TOP 500 benchmark)

Janv. 2009:	1.1 Pflop/s	2.5 MW
Déc. 2013:	33.8 Pflop/s	17.8 MW
	($\times$30)	($\times$7)

Objectif: 1 Eflop/s vers 2020-2022 à 20 MW

Contexte scientifique

Super-calculateur le plus rapide (TOP 500 benchmark)

Janv. 2009:	1.1 Pflop/s	2.5 MW
Déc. 2013:	33.8 Pflop/s	17.8 MW
	($\times$30)	($\times$7)

Objectif: 1 Eflop/s vers 2020-2022 à 20 MW

Principaux enjeux:

Contexte scientifique

Super-calculateur le plus rapide (TOP 500 benchmark)

Janv. 2009:	1.1 Pflop/s	2.5 MW
Déc. 2013:	33.8 Pflop/s	17.8 MW
	($\times$30)	($\times$7)

Objectif: 1 Eflop/s vers 2020-2022 à 20 MW

Principaux enjeux:

- **Energie** (fin de course aux GHz, développement des multicœurs)

Contexte scientifique

Super-calculateur le plus rapide (TOP 500 benchmark)

Janv. 2009:	1.1 Pflop/s	2.5 MW
Déc. 2013:	33.8 Pflop/s	17.8 MW
	($\times$30)	($\times$7)

Objectif: 1 Eflop/s vers 2020-2022 à 20 MW

Principaux enjeux:

- **Energie** (fin de course aux GHz, développement des multicœurs)
- **Communications** (goulot d'étranglement, flops importent moins)

Contexte scientifique

Super-calculateur le plus rapide (TOP 500 benchmark)

Janv. 2009:	1.1 Pflop/s	2.5 MW
Déc. 2013:	33.8 Pflop/s	17.8 MW
	($\times$30)	($\times$7)

Objectif: 1 Eflop/s vers 2020-2022 à 20 MW

Principaux enjeux:

- **Energie** (fin de course aux GHz, développement des multicœurs)
- **Communications** (goulot d'étranglement, flops importent moins)
- **Hétérogenéité** (accélérateurs: GPUs, Xeon Phi)

Contexte scientifique

Super-calculateur le plus rapide (TOP 500 benchmark)

Janv. 2009:	1.1 Pflop/s	2.5 MW
Déc. 2013:	33.8 Pflop/s	17.8 MW
	($\times$30)	($\times$7)

Objectif: 1 Eflop/s vers 2020-2022 à 20 MW

Principaux enjeux:

- **Energie** (fin de course aux GHz, développement des multicœurs)
- **Communications** (goulot d'étranglement, flops importent moins)
- **Hétérogenéité** (accélérateurs: GPUs, Xeon Phi)
- **Qualité numérique** (précision, reproductibilité)

Contexte scientifique

Super-calculateur le plus rapide (TOP 500 benchmark)

Janv. 2009:	1.1 Pflop/s	2.5 MW
Déc. 2013:	33.8 Pflop/s	17.8 MW
	($\times$30)	($\times$7)

Objectif: 1 Eflop/s vers 2020-2022 à 20 MW

Principaux enjeux:

- **Energie** (fin de course aux GHz, développement des multicœurs)
- **Communications** (goulot d'étranglement, flops importent moins)
- **Hétérogenéité** (accélérateurs: GPUs, Xeon Phi)
- **Qualité numérique** (précision, reproductibilité)
- Tolérance aux défaillances (Sequoia BG/Q: 1,25 /nœud/jour)

Outline

Outline

Outline

Objectifs de Postale

Postale: Equipe-projet d'Inria Saclay dans le domaine du HPC

Tirer parti des nouvelles architectures dans les applications, en allant des compilateurs jusqu'aux bibliothèques logicielles

- Développer méthodes et logiciels pour l'**optimisation et la transformation de programmes**
- **Configuration d'architectures** pour une application ou un algorithme donné
- Tirer parti de la **connaissance du programmeur/utilisateur** pour développer des codes efficaces
- Proposer des **algorithmes/bibliothèques efficaces et précis** pour les architectures hautement parallèles et/ou hétérogènes

Objectifs de Postale

Postale: Equipe-projet d'Inria Saclay dans le domaine du HPC

Tirer parti des nouvelles architectures dans les applications, en allant des compilateurs jusqu'aux bibliothèques logicielles

- Développer méthodes et logiciels pour l'**optimisation et la transformation de programmes**
- **Configuration d'architectures** pour une application ou un algorithme donné
- Tirer parti de la **connaissance du programmeur/utilisateur** pour développer des codes efficaces
- Proposer des **algorithmes/bibliothèques efficaces et précis** pour les architectures hautement parallèles et/ou hétérogènes

Maitrise de la chaîne complète, des algorithmes à l'architecture

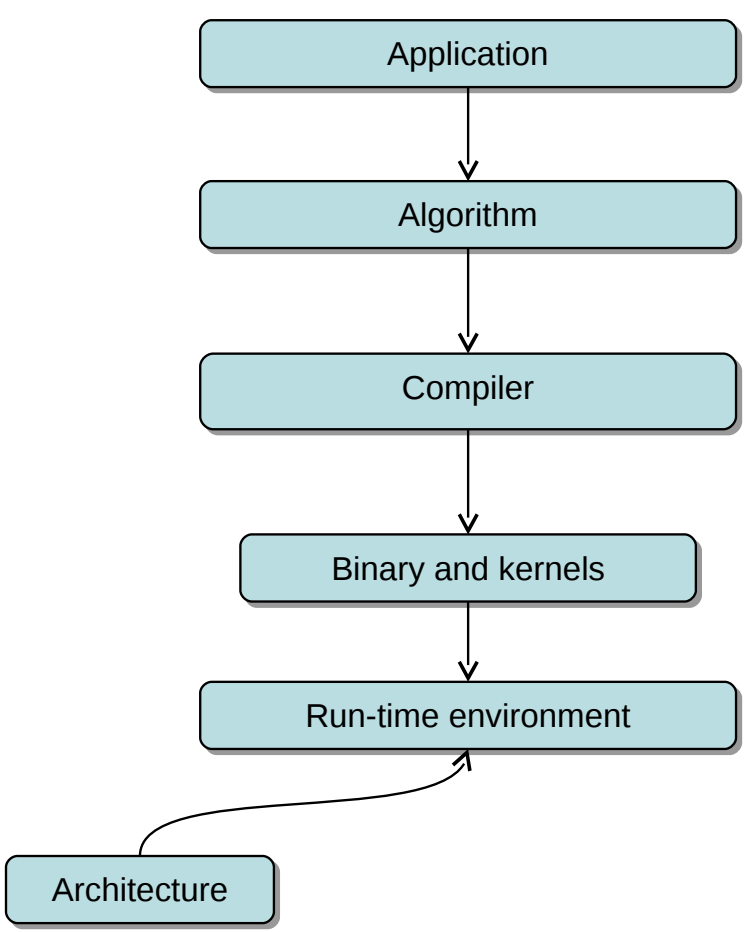
Membres permanents de l'équipe

- **Marc Baboulin**, Resp. d'équipe, Professeur Univ. Paris-Sud
- **Christine Eisenbeis**, Directrice de recherche Inria Saclay
- **Daniel Etiemble**, Professeur Univ. Paris-Sud
- **Joël Falcou**, Maître de conférences Univ. Paris-Sud
- **Grigori Fursin**, Chargé de recherche Inria Saclay
- **Lionel Lacassagne**, Maître de conférences HDR Univ. Paris-Sud
- Collaborateur extérieur: **Cédric Bastoul**, PR Univ. Strasbourg

Les membres de Postale appartiennent à l'équipe **ParSys** (Parallel Systems) du **Laboratoire de Recherche en Informatique**

Thèmes de recherche

Postale : from applications to parallel architectures



Topic 3 : High-level libraries & applications

Topic 2 : Programmer/compiler interface

Topic 1 : Architecture & program optimization

Tuning collaboratif de programmes

- **Systématiser l'expérimentation** par une approche collaborative: *Collective Mind* public repository (`http://c-mind.org/repo`) pour partager tout type d'objet de recherche incluant les méta-descriptions (données, benchmarks, outils, modèles)
- **Systématiser l'auto-tuning** en appliquant des techniques d'apprentissage et de fouille de données pour **prédire l'optimisation de programmes** à partir de données collectées
- Support d'un nouveau modèle de publication où **tous les résultats sont partagés et validés par la communauté**

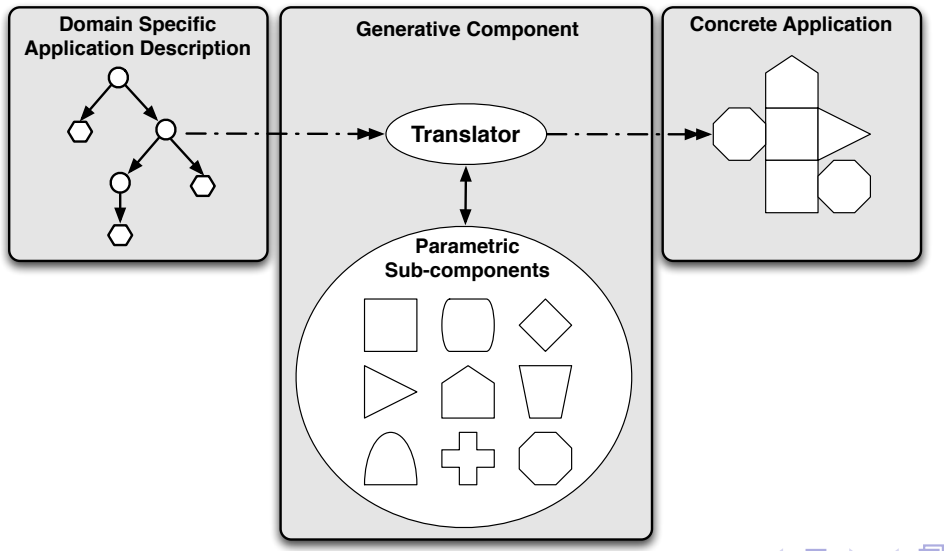
Prise en compte de contraintes algorithmiques dans les techniques de compilation

- Etendre le modèle polyédrique pour prendre en compte les transformations **communication-avoiding** (e.g., calculs aux points frontières...)
- Outils d'optimisation pour le **placement de données**: transformation de data layout, génération automatique d'instructions MPI...
- **Consommation d'énergie**: calcul réversible, monitoring et modèle de coût de l'énergie des processeurs, algorithmes en précision mixte

Interface programmeur-compilateur

Exploiter les informations spécifiques au domaine d'application en utilisant la programmation générative

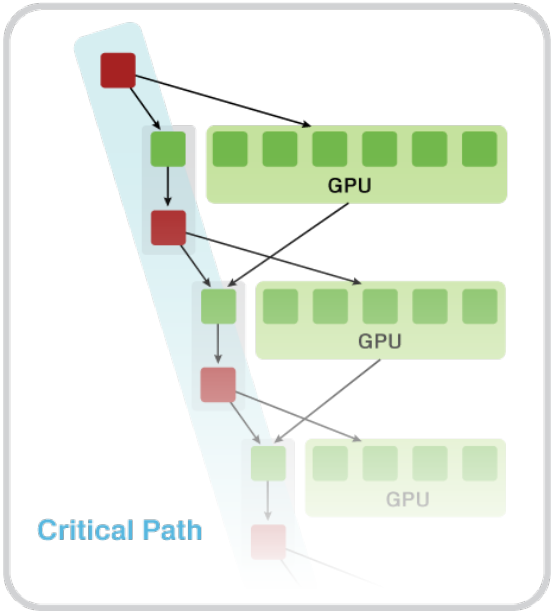
- Proposer un **langage déclaratif** lié au domaine d'application (ex. traitement d'image, algèbre linéaire, Lattice QCD)
- Extraire l'**information de haut niveau** pour le compilateur et améliorer la performance, considérer les **caractéristiques de l'architecture** comme un composant additionnel



Bibliothèques de haut niveau et applications

Accélérer les simulations numériques sans perte de précision

- **Algorithmes hybrides** pour les architectures hétérogènes



- **Minimiser les communications et les synchronisations** dans les algorithmes d'algèbre linéaire (réduire pivotage, look-ahead...)
- **Qualité numérique** dans les applications HPC

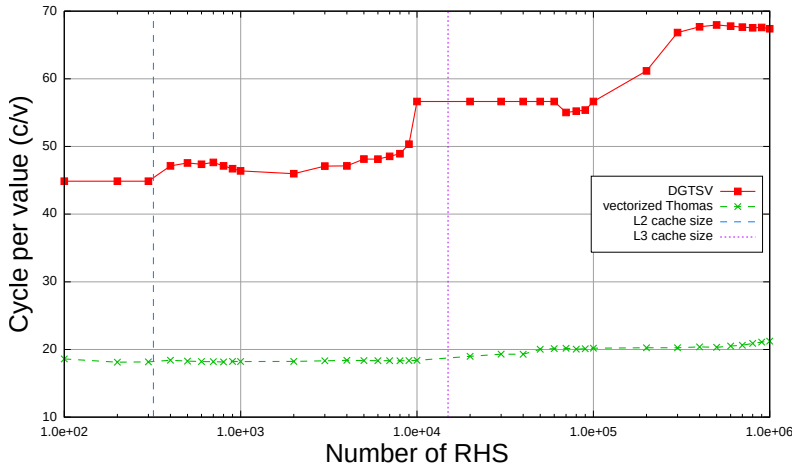
Bibliothèques de haut niveau et applications

Transformations pour le traitement d'image

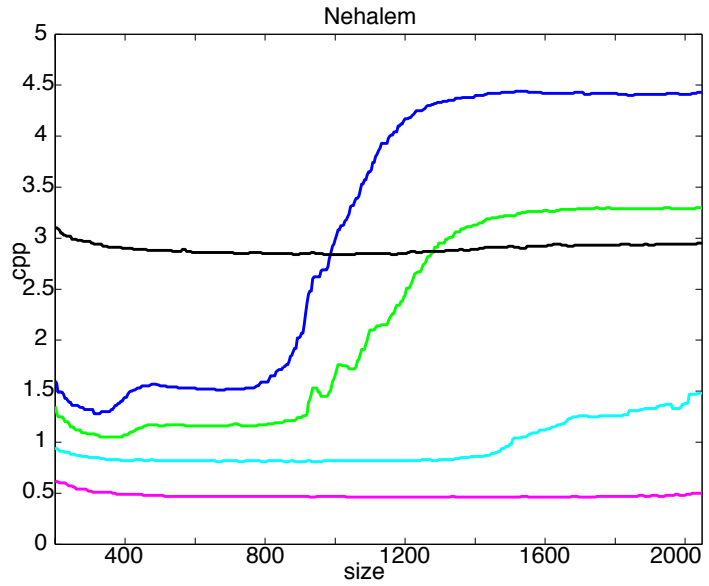
- **Re-design d'algorithmes séquentiels irréguliers** pour environnement parallèle (Connected Component Labeling, Light Speed Labeling)
- **Implémentation temps réel** d'applications de vision par ordinateur pour systèmes embarqués
- **Bibliothèque pour allocation mémoire**
- **Benchmarking et optimisation de code** (Xeon Phi, Kalray...)
- **Codage d'arithmétique** pour systèmes embarqués

Exemples d'applications

Equations de Navier-Stokes: Optimisation de solveurs tridiagonaux (Magny-cours)



Traitement d'image: algorithme de Harris (détection de points)



Equipe associée

Projet R-LAS (Randomized Linear Algebra Software)

Equipe-associée avec **Université du Tennessee** (Jack Dongarra)

Objectif:

Proposer des algorithmes plus rapides que les algorithmes déterministes (moins de calcul et/ou de communication) tout en garantissant la précision des résultats.

Exemples:

- Eviter le pivotage dans l'élimination de Gauss en utilisant des transformations aléatoires, Random Butterfly Transformations (RBT) $\rightarrow$ moins de communication
- Calcul d'estimateurs statistiques de conditionnement (SCE) $\rightarrow$ moins de calculs

Equipe associée

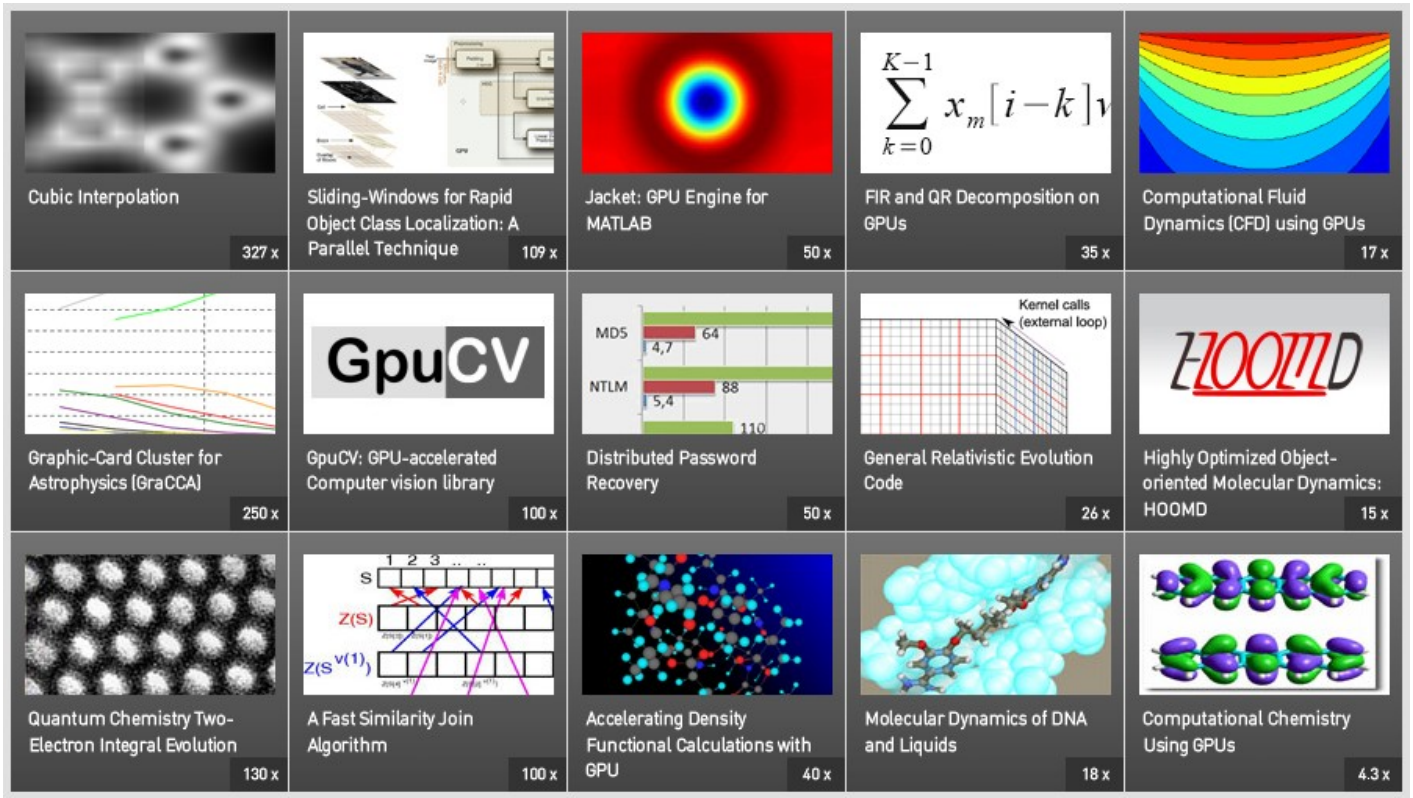
R-LAS (Randomized Linear Algebra Software)

Exemple d'approche globale proposée par Postale:

- Développement d'algorithmes numériques
  adaptation à plusieurs types de solveurs (dense/creux, itératifs/directs)
- Génération de code par DSL
  codes prototypes, analyse de performance, test sur nouvelles architectures
- Tuning des paramètres par approche collaborative
  - nombre de récursions pour RBT
  - nombre de vecteurs orthogonaux pour SCE

Outline

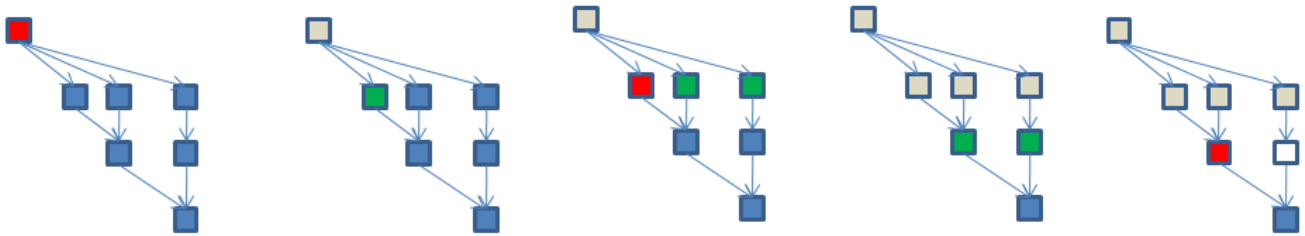
Why GPU-based computing



- Most HPC applications report high speedups with GPUs.
- Top 500, November 2013:
  53 systems with accelerators.
  #2 and #6 systems use NVIDIA GPUs (#1 and #7 systems use Intel Xeon Phi)
  Green500 list: Top 10 are all GPU-accelerated

Principles of hybrid implementation

1. Exploit strengths of each architectural component
2. Minimize communication and data transfers
3. Properly schedule the tasks execution over the CPU and the GPU
4. Exemple: **MAGMA** (linear algebra library for hybrid architectures)



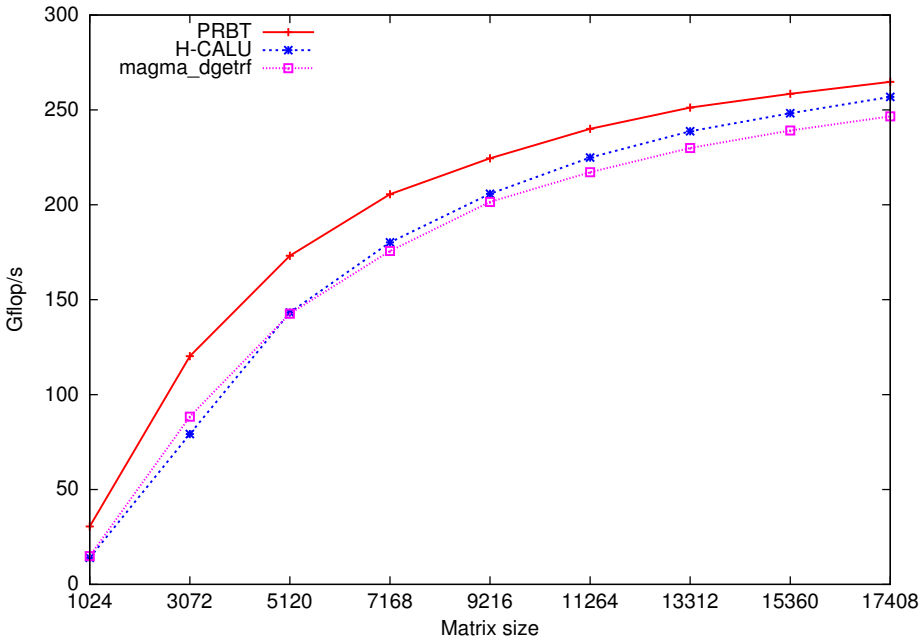
Task splitting in hybrid LU factorization (4 panels)

Solvers that reduce data-communication costs

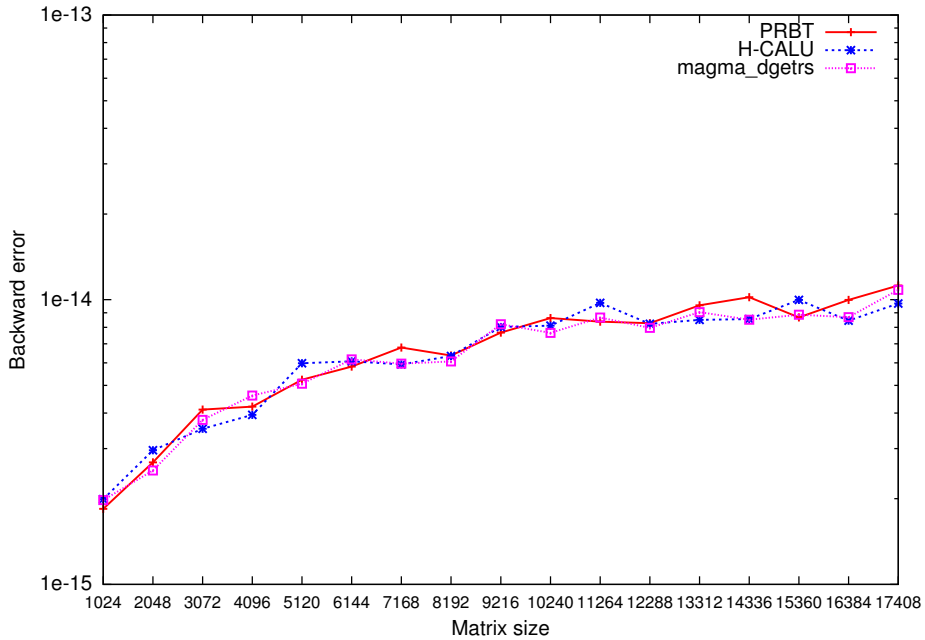
Our recent development in hybrid LU factorization:

- **Communication in pivoting can be reduced** by using tournament pivoting$\rightarrow$ **H-CALU solver**
- **We can remove completely the pivoting** by preprocessing the system by randomization ($\mathcal{O}(n^2)$ flops) $\rightarrow$ **PRBT solver**

Performance/accuracy of hybrid LU implementations



Performance results



Componentwise backward error
($\omega = \max_i \frac{|Ax-b|_i}{(|A|\cdot|x|+|b|)_i}$)

Experiments on AMD (16 threads) + NVIDIA Fermi Tesla S2050

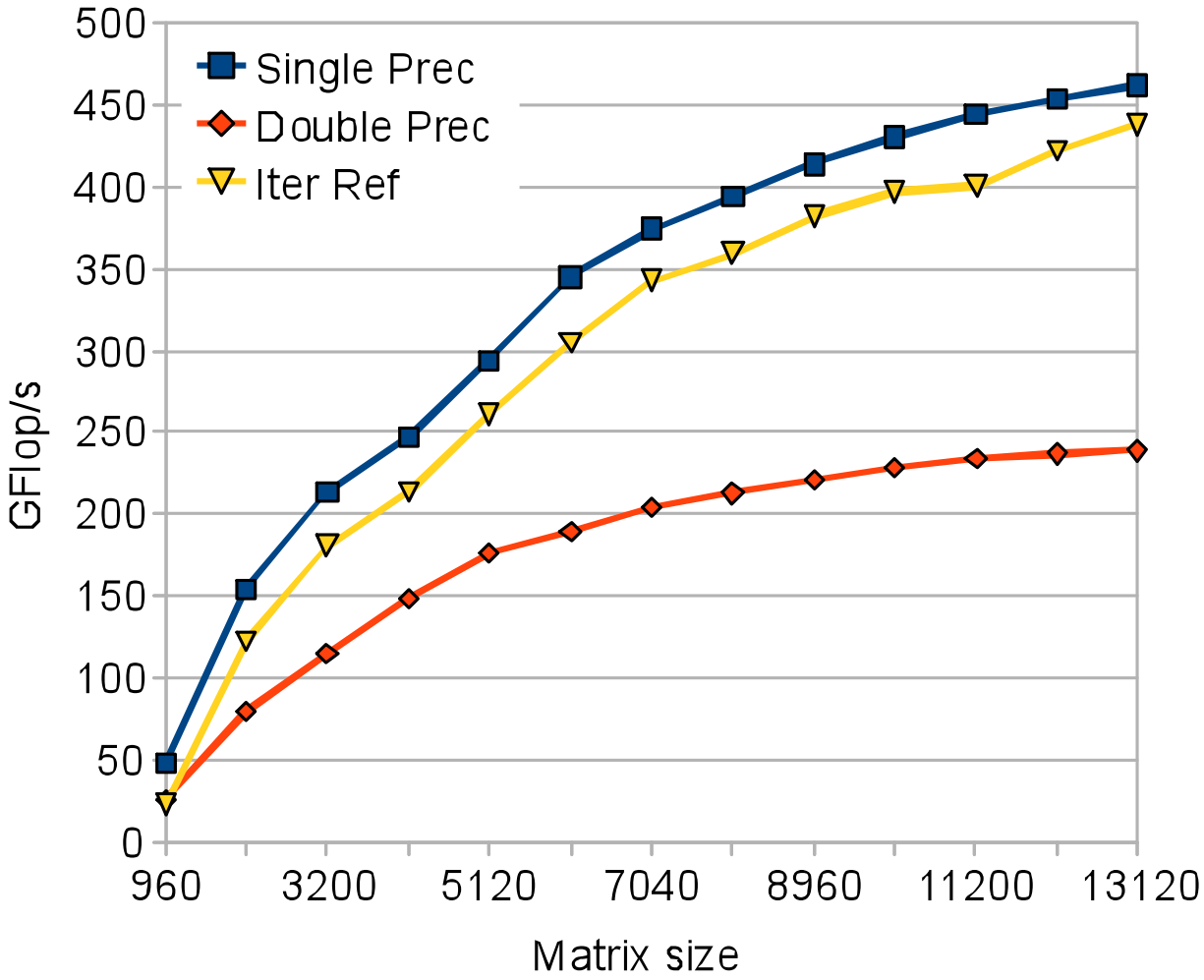
Mixed precision algorithms

- Bulk of the computation in 32-bit arithmetic
- Postprocess the 32-bit solution by refining it into a solution that is 64-bit accurate
  Can be performed on the GPU
- Problem must be ”not ill-conditioned”
- Application to **linear systems** and **linear least squares**

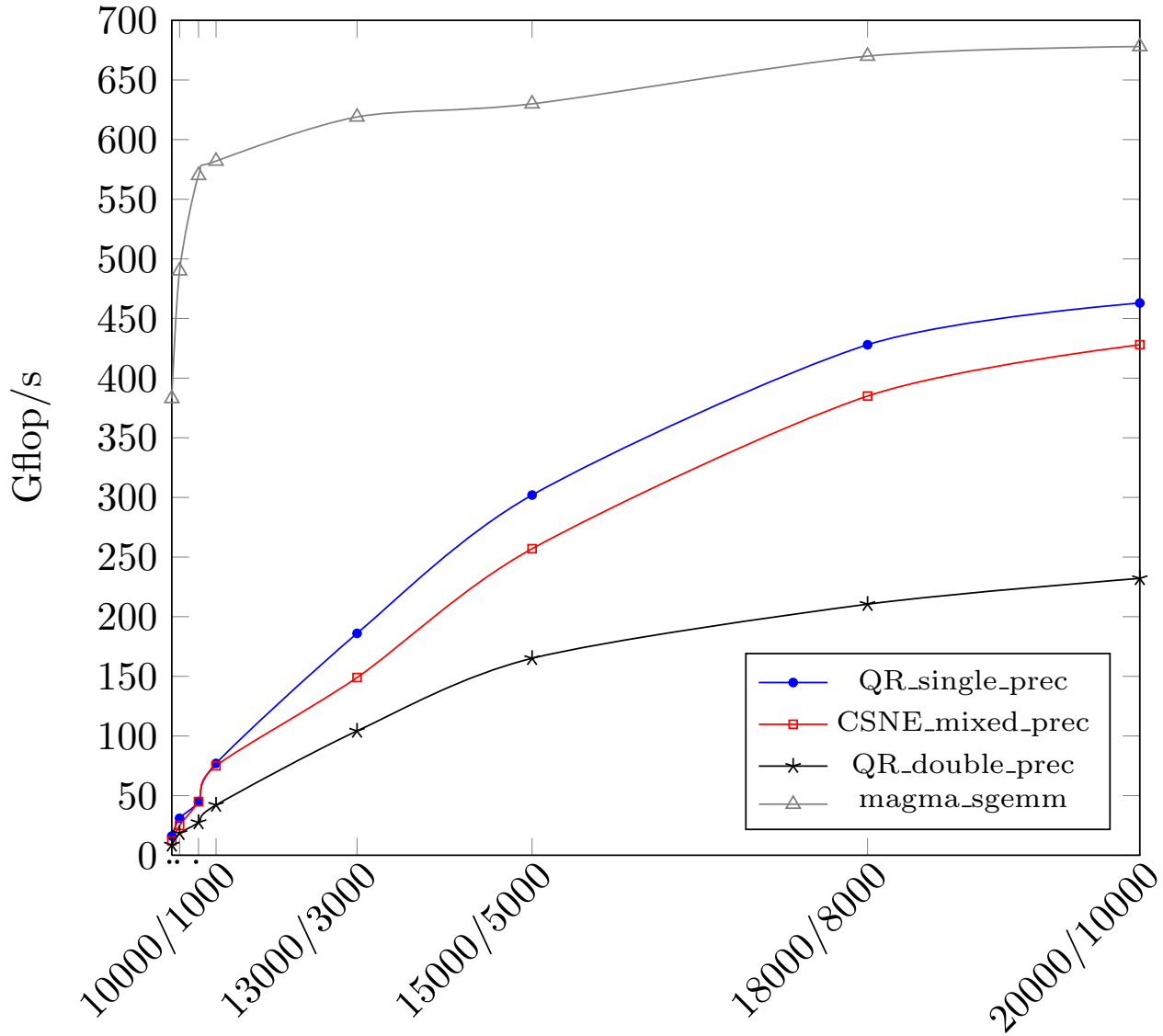
Mixed precision algorithms

Example of the LU factorization

1:	$LU \leftarrow PA$	(ε_s)	$\mathcal{O}(n^3)$
2:	solve $Ly = Pb$	(ε_s)	$\mathcal{O}(n^2)$
3:	solve $Ux_0 = y$	(ε_s)	$\mathcal{O}(n^2)$
	do $k = 1, 2, \ldots$		
4:	$r_k \leftarrow b - Ax_{k-1}$	(ε_d)	
5:	solve $Ly = Pr_k$	(ε_s)	
6:	solve $Uz_k = y$	(ε_s)	
7:	$x_k \leftarrow x_{k-1} + z_k$	(ε_d)	
	check convergence		
	done		



Mixed-precision LU factorization - NVIDIA Fermi Tesla C2050



Mixed-precision least squares solver (CSNE) - NVIDIA Tesla C2075

Software

- **Collective Mind**: computer systems optimizations using machine-learning approach (G. Fursin)
- **myNRC**: image-oriented library for allocation and manipulation of SIMD 1D, 2D and 3D structures (L. Lacassagne)
- **NT2**: C++ high level framework for scientific computing (J. Falcou)
- **(Sca)LAPACK, PLASMA, MAGMA**: linear algebra solvers and randomized algorithms (M. Baboulin)

Résumé

- Recherche centrée sur des **sujets majeurs de la communauté** HPC (énergie, communication, hétérogeneité...)
- Utiliser nos **compétences complémentaires** au service des applications
- Développer des **collaborations dans Paris-Sud et Paris-Saclay** (LIMSI, Labo Maths, LPT, Maison de la Simulation..)
- Concevoir et valoriser les **bibliothèques logicielles**
- Développement de **partenariats industriels** (EDF, ARM...)
- Implication dans la **formation à la recherche** (Master 2 Info, Master 2 HPC)