

Portabilité
vs Performance
vs Productivité
vs Précision



*Contribution à la Prospective IN2P3, GT09.
David Chamont (FLUO), pour le groupe Reprises.*



Problématique

- Utilisateurs et développeurs du calcul scientifique font face à une **diversité et une hétérogénéité matérielle croissante**. Pour tirer le meilleur d'un matériel spécifique, il faut le plus souvent écrire du code spécifique.
- Le défi présent consiste à développer du logiciel **portable** sans trop dégrader la **performance** des applications et la **productivité** des développeurs. Ceci avec la seule **précision utile**, et pas plus.
- Quel niveau de dégradation est acceptable ? Quelle est la précision utile ? Comment mesurer la productivité ? Quelles technologies choisir dans la **jungle logicielle** ?



La jungle logicielle

- Directives : OpenMP, OpenACC.
- Bibliothèques de vectorisation : intrinsics, Vc, VTD, Vector libm, UME, bSIMD, inastemp, xsimd, MIPP, ...
- Bibliothèques de portabilité C/C++ : Kokkos, Raja, ...
- Génération de code : TensorFlow, Loopy, xtensor.
- Accélérateurs : **OpenCL** sur GPU et FPGA, Vulkan.
- Runtimes & calcul distribué : StarPU, Spark, HPX...
- Extensions et ponts entre langages : Pythran, **SyCL**, ...
- Bibliothèques numériques : Eigen, Armadillo , MKL, Lapack, Blas, NAG, Random123,...



L'outre-Atlantique s'organise

- 2019/11, SC Denver, International Workshop on Performance, Portability and Productivity in HPC (P3HPC)
- 2019/04, Denver, DOE PPP Meeting
- 2018/11, SC Dallas, International Workshop on Performance, Portability and Productivity in HPC (P3HPC)

Dernière minute : prochain forum ORAP le 29 novembre sur "Concilier pérennité et performance" !



La problématique se formalise

$$\Phi(a, p, H) = \frac{|H|}{\sum_{i \in H} \frac{\min(F_i, B_i \times I_i(a, p))}{P_i(a, p)}}$$

*S.J.Pennycook, J.D.Sewall, and V.Lee,
"A metric for performance portability", 2016*



Mauvaises nouvelles

- Un **profilage global** est indispensable : du temps CPU, des déplacements de données, et de la précision numérique.
- GPU et FPGA ne sont efficaces que sur des cas appropriés.
- L'auto-vectorisation ne pollue pas le code... mais demande un code **implicitement adapté**.
- Les directives sont d'un accès facile... mais on des **possibilités limitées**.
- Génération de code et DSLs... **complexifient la construction** de l'exécutable.
- Il faut renoncer au confort de la **double précision** et à l'illusion de la **reproductibilité bit à bit**.



Bonnes nouvelles

- Il y a des stratégies toujours payantes quelle que soit la technologie adoptée :
 - organiser ses données en **structures de tableaux**,
 - privilégier les algorithmes **éhontément* parallèles**,
 - adopter un style de **programmation fonctionnel et asynchrone**.
- Même lorsque le gain de performance final est décevant, un essai d'optimisation et de parallélisation d'un code :
 - **améliore sa qualité et ses performances**, même en séquentiel,
 - facilite la transition vers d'autres technologies, parce les problèmes essentiels sous-jacents sont déjà traités.
- **Il ne faut pas avoir peur de se tromper de technologie.**

** copyright Vincent Lafage*



Question 1

What can be the way forward in using accelerators (GPU, FPGA) and HPC in our field ?

1. Si l'application existe, la **profiler globalement**.
2. S'assurer que la **technologie** est **adaptée** au problème traité.
3. Rendre la **précision configurable**, et s'efforcer de la réduire.
4. Structurer les **données en tableaux**.
5. Chercher l'**algorithme** le plus **parallélisable**.
6. Exprimer les noyaux de calcul dans un **style fonctionnel**.
7. Commencer par la technologie **la moins invasive**.



Question 2

Is dedicated re-coding necessary, or are abstraction layer libraries like alpaka, kokkos, SyCL the way forward ?

Ça dépend du type d'application et du dosage recherché entre portabilité, performance, productivité, précision, pérennité...

- acquisition en ligne : la performance d'abord
- reconstruction distribuée : la portabilité d'abord
- analyse de physique : la productivité d'abord



Recommandations

- **Recruter des profils "calcul"**, pour le développement comme pour l'administration des serveurs.
- **Développer la recherche appliquée**, en cofinçant des thèses avec des laboratoires d'informatique.
- **Poursuivre une veille active** des nouvelles technologies, et se doter du matériel nécessaire.
- **Former massivement** la communauté : programmation fonctionnelle, outils de profilage, techniques d'optimisation... et valoriser ceux qui le font.
- Un seul laboratoire ne peut pas réunir toutes les compétences nécessaires => construire et consolider une **Force de frappe IN2P3**.