Multiplication entière parallèle (proposition de stage L3)

Lieu. INRIA/LORIA, Nancy, `www.loria.fr`.

Encadrants. Emmanuel Thomé et Paul Zimmermann, directeurs de recherche, équipe Caramba, `Emmanuel.Thome@inria.fr, Paul.Zimmermann@inria.fr`.

Contexte. Savoir multiplier rapidement de grands entiers est crucial en algorithmique et calcul formel. De nombreux algorithmes « rapides » s'y ramènent. Par exemple, si on dénote par $M(n)$ le coût de la multiplication de deux entiers de n bits, on sait effectuer une division ou une racine carrée en $O(M(n))$ également, et calculer un pgcd en $O(M(n)\log n)$. En pratique, la bibliothèque GMP [3] implante les algorithmes les plus efficaces que l'on connaisse pour les tailles effectivement atteignables. Voici les temps de multiplication de deux nombres de n mots de 64 bits sur un processeur Intel Xeon Gold 6130 à 2.1Ghz, avec GMP 6.1.2 :

n	10^6	10^7	10^8	10^9
temps (secondes)	0.34	4.49	50.4	576

De manière assez surprenante, il n'existe pas de version parallèle de la bibliothèque GMP. Ainsi la seule solution pour paralléliser un code faisant appel à GMP est d'effectuer en parallèle plusieurs opérations. Mais cela n'est pas toujours possible : par exemple, dans la *batch cofactorization* de NFS [1], on a à un moment de l'algorithme une seule énorme multiplication à effectuer.

Objectif du stage. L'objectif du stage est d'implanter une multiplication d'entiers parallèle qui soit plus efficace que celle (séquentielle) de GMP, avec un *speedup* le plus proche possible de t où t est le nombre de threads utilisés. On commencera dans un premier temps par faire un panorama de l'état de l'art des algorithmes possibles et de leurs implantations, notamment dans GMP, la bibliothèque Flint (`flintlib.org`), la couche NTT de GMP-ECM [4], l'implantation de [2], ... Le code pourra utiliser les routines de base de GMP, et devra utiliser la même API que GMP, afin qu'on puisse l'utiliser facilement dans les applications dépendant de GMP. Tous les moyens sont autorisés pour arriver à cet objectif !

Références

[1] Bernstein, D. J. How to find small factors of integers. `cr.yp.to/factorization.html`, 2002.

[2] Gaudry, P., Kruppa, A., and Zimmermann, P. A GMP-based implementation of Schönhage-Strassen's large integer multiplication algorithm. In *Proceedings of the 2007 International Symposium on Symbolic and Algebraic Computation, ISSAC'2007* (Waterloo, Ontario, Canada, 2007), C. W. Brown, Ed., pp. 167–174.

[3] Granlund, T., and the GMP development team. *GNU MP : The GNU Multiple Precision Arithmetic Library*, 6.1.2 ed., 2016. `http://gmplib.org/`.

[4] Zimmermann, P., and Dodson, B. 20 years of ECM. In *Proceedings of the 7th Algorithmic Number Theory Symposium (ANTS VII)* (Berlin Heidelberg, 2006), F. Hess, S. Pauli, and M. Pohst, Eds., vol. 4076 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 525–542.