Titre	**Implémentation d'un algorithme de partitionnement de maillages sous contraintes mémoires**
Contact	C. CHEVALIER–cedric.chevalier@cea.fr
Contexte	Dans le cadre de simulations numériques faisant intervenir des éléments ou volumes finis, les données sont portées par un maillage qui discrétise l'espace de calcul en éléments simples, les mailles. Pour une exécution sur un supercalculateur disposant par définition de plusieurs unités de calcul, ou UC, il est alors nécessaire de distribuer ce maillage et les données associées entre les différentes UC. Les outils classiques [1,2] partitionnent les données de la simulation entre UC, c'est-à-dire attribuent chaque maille à exactement une et un seule UC, dans le but d'équilibrer les temps de calcul sur chaque UC. Or, l'espace mémoire de chaque UC ayant tendance à diminuer sur les calculateurs modernes, il devient nécessaire de prendre en compte la consommation mémoire de manière explicite lors du partitionnement. Dans ce but, des travaux récents menés lors d'une thèse au CEA/DAM ont permis de valider des heuristiques de partitionnement sous contraintes mémoire. [1] C. Chevalier and F. Pellegrini, PT-SCOTCH: a tool for efficient parallel graph ordering. Parallel Computing, 34(6-8), pp 318-331, 2008. [2] E.G. Boman, U.V. Catalyurek, C. Chevalier, and K.D. Devine, The Zoltan and Isorropia Parallel Toolkits for Combinatorial Scientific Computing: Partitioning, Ordering, and Coloring, Scientific Programming vol. 20, no. 2, 2012,
Objectifs	Partant d'un algorithme existant de partitionnement, l'objectif du stage sera d'en fournir une implémentation C++ en s'appuyant sur le partitionneur Scotch [2] comme brique élémentaire. Plus précisément, l'algorithme proposé suit une structure multi-niveaux : 1. Un graphe bi-parti G1 représentant le maillage initial est contracté en un graphe G2 de plus petite taille. 2. Le problème de partitionnement es alors résolu pour G2 3. La solution obtenue pour G2 est raffinée en une solution pour G1. Chacune des étapes fait intervenir des connaissances en théorie des graphes, optimisation et recherche locale. Selon le déroulement du stage, l'implémentation pourra être purement séquentiel, concurrente (plusieurs threads) ou hybride concurrente et distribuée (threads+MPI). Si ce stage requiert des compétences en algorithmique et en optimisation, il nécessite également de bonnes connaissances en développement de code C/C++.
Domaines de spécialité requis	Informatique
Langages/logiciels	C/C++
Mots clés	Partitionnement, théorie des graphes, implémentation, parallélisme
Formation recherchée	BAC+5
Durée du stage	5 à 6 mois
Stage pouvant se poursuivre en thèse	Oui