PROGRAMMATION ET EXPLOITATION DES PLATEFORMES HPC : DÉFIS ET CHALLENGES

François Bodin,
Professeur, Université de Rennes 1, IRISA

Jean-François Méhaut,
Professeur, Université de Grenoble Alpes, LIG

La majorité des applications HPC repose sur des technologies et pratiques logicielles inventées il y a plusieurs décennies. Les considérations énergétiques, la fin de la loi de Moore et la production d'énormes volumes de données ont initié une mutation profonde du paysage du HPC où les équilibres calcul et stockage sont remis en cause. Face à cette rupture technologique c'est l'ensemble de la pile logicielle (systèmes, supports d'exécution, compilateurs, bibliothèques) et les codes applicatifs qui doivent évoluer.

DE L'IMPACT DE L'ÉNERGIE SUR LE LOGICIEL

La consommation énergétique est devenue le principal obstacle à la progression des performances. L'augmentation du nombre de cœurs s'est substituée pour fournir une puissance de calcul toujours accrue. Les technologies d'accélération (GPU, Xeon Phi, MPPA) ont aussi permis d'augmenter l'efficacité énergétique. Cette massification du parallélisme, les nouvelles technologies de stockage (mémoires non volatiles, 3D stacking) et d'interconnexions (photonique) vont conduire à des architectures de machines, énergiquement plus efficaces, mais de structures nouvelles et variées qui devront se refléter dans l'ensemble des couches logicielles. De plus, l'augmentation exponentielle du nombre de cœurs des supercalculateurs impose d'utiliser des schémas de tolérance aux fautes.

Ces points nécessitent d'introduire des API capables de mesurer et de gérer la consommation d'énergie au niveau des applications. Les modifications de la hiérarchie de stockage et les problèmes de fautes matérielles requièrent de nouvelles approches dans la gestion des entrées / sorties et de la mise en œuvre de la résilience. Les nouvelles API devront permettre d'établir un dialogue de gestion des ressources matérielles performants et énergétiquement efficace. Cette profonde évolution devrait conduire à une attribution des ressources utilisateurs, actuellement mesurée en heures.cpu, à une allocation en watt/heure.

DONNÉES EXTRÊMES ET HPC

La précision des simulations, les approches multi-physiques ainsi que les techniques d'assimilation de données, produisent de grands volumes de données. L'exploitation et la gestion des données sont l'autre pan de l'évolution des logiciels qui va bouleverser les pratiques. Une majorité des applications HPC sont fortement limitées par le traitement des données (incluant la visualisation). Les pratiques traditionnelles de stockage sur disque de l'ensemble des données

produites pour les analyser ne seront possibles que marginalement. L'analyse devra se faire « in-situ », en mémoire du calculateur. Naturellement, cette problématique intérroge sur l'introduction de technologies de type « Big-Data ».

Du point de vue des couches basses logicielles, la cohabitation de techniques issues de la fouille de données impose, à terme, de faire cohabiter des technologies HPC traditionnelles avec celles du monde de l'analyse des données dans des Workflows complexes. Les techniques de virtualisation, telle que les Containers, font leur entrée dans les systèmes d'exploitation des supercalculateurs.

HÉTÉROGÉNÉITÉ, PERFORMANCE ET SCALABILITÉ

Les approches par décomposition de domaine et échanges de messages peinent à prendre en compte les structures de machines fondées sur des processeurs massivement parallèles. L'hétérogénéité des ressources de calcul ajoute un degré de complexité. L'ensemble de la pile logicielle, en particulier les systèmes d'exploitation, les supports d'exécution et les compilateurs, doit permettre une gestion efficace, en supportant des paradigmes de programmation parallèle multi-niveau, scalable, permettant de mélanger des approches par mémoire distribuée et par mémoire partagée. Un point important est la régulation dynamique des charges de calcul qui prennent en compte les considérations énergétiques, tout en respectant les affinités entre les tâches et les accès aux structures de données.

Il s'agit de mélanger des approches par échange de message avec des approches basées sur des graphes de tâches capables de s'exécuter indifféremment sur les cœurs de CPU ou les éventuels accélérateurs (StarPU, OmpSS). Les techniques d'optimisation automatique (auto-tuning) devront se généraliser tant l'espace des possibilités à explorer est grand et multicritères (énergie vs performance, mémorisation vs re-calcul).

COMPLEXITÉ DE LA PROGRAMMATION

Pour être efficace, la programmation parallèle doit prendre en compte l'hétérogénéité des unités de calcul, la hiérarchie mémoire (mémoire cache et système de stockage) ainsi la hiérarchie du parallélisme matériel (SIMD, SMT, multi-cœurs, clusters). La programmation hybride mixant des API telles que OpenMP, OmpSS pour le parallélisme intra-nœud et MPI ou PGAS entre nœuds ont fortement gagné en popularité ces dernières années. Cependant, l'obtention de code efficace reste un véritable challenge. La complexité croissante des machines nécessite la mise en œuvre de techniques d'optimisation de code élaborées et coûteuses en main-d'œuvre.

Les approches à base de langages et API spécifiques à un domaine (DSL) représentent une piste importante pour la séparation des aspects informatiques des aspects applicatifs. La définition de solution spécifique à un domaine est une recherche de compromis difficile. Les DSL à base de langages dynamiques présentent un intérêt pour construire des environnements de programmation de très haut niveau (typiquement sur la base de méthodes de méta-programmation). Des progrès importants restent à faire pour aider à la mise au point des programmes massivement parallèles.

Les nombreux changements technologiques vont profondément altérer la façon de concevoir les applications. L'ensemble des couches logicielles seront revues pour prendre en compte les problématiques énergétiques, de gestion/exploitation de données et de tolérance aux fautes. Les pratiques de développement feront appel aux techniques de génie logiciel pour obtenir une architecture de code assez robuste pour s'adapter à plusieurs générations de machines. Les approches spécifiques à chaque domaine (DSL) permettront d'offrir des abstractions aux utilisateurs. Il faudra trouver des modèles économiques permettant de servir une communauté donnée tout en assurant la pérennité des solutions. Les défis ne sont pas minces !