エクサスケールに向けた高性能アプリケーション開発

High-performance application development towards exa-scale computation

○ 小野 謙二, 理研 AICS, 神戸市中央区港島南町 7-1-26, E-mail: keno@riken.jp
三上 和徳, 理研 AICS, 神戸市中央区港島南町 7-1-26, E-mail: kazunori.mikami@riken.jp
Kenji Ono, AICS, RIKEN, 7-1-26, Minatojima-minami-machi, Chuo-ku, Kobe, 650-0047, Japan
Kazunori Mikami, AICS, RIKEN

Toward next generation extreme-scale computing, it is obviously essential to establish an extreme-scale computing environment in addition to develop a new computer system. Further, it is of great importance to develop high performance applications and its utilization, which contribute to society through scientific discovery. In this paper, we discuss about how we can develop high performance codes efficiently with high productivity and continuity beyond generations.

１．はじめに

Extreme-scale computing では，まず，それを実現する計算機システムと環境の開発が必要となる．同時に，その計算機を活用して科学的な発見や社会的な貢献を果たすアプリケーションの開発と活用促進が，科学技術の社会還元として重要になる．社会還元の一つとして産業利用が取り上げられるが，そこでは大規模な計算機資源を利用した高い信頼性のあるシミュレーション，複雑現象の解析，迅速で短時間のシミュレーション，多数のケーススタディによる最適化などが期待されている．また，既に Extreme-scale computing に向けて，アーキテクチャとアプリケーションをどのように開発していくかについて，計算機科学の研究者と実際のアプリケーション側の研究者が集まり，議論を始めている[1, 2].

本稿では，大規模並列環境における高性能なアプリケーションを開発・維持していくための検討について報告する．

２．エクサスケールにおけるプログラミングの課題とチャレンジ

エクサスケールの計算機を構成するデバイスには，CPU コア，キャッシュ，インターコネクト，メモリシステム，ディスク，アクセラレータなどがある. これらのデバイスを組み合わせる上で，最も強い制約条件の一つは消費電力であると認識されている．この制約条件のもと，高い演算性能をもつ計算機システムを構築するためには，低消費電力のデバイスを多数結合する，非通信時のネットワーク消費電力を抑える，消費電力に対する演算性能比の高い加速機構を併用するなどのアイデアなどがあり，場合によっては CPU と加速器の混合のようなヘテロなアーキテクチャが出現する可能性もある．また，一定の電力バジェット内で演算器，メモリ，ネットワークにどのように電力を割りつけるか，またメモリモジュールの構成ポリシ（容量優先か速度優先か）など，そのバランスによってはアプリケーションの実行特性も大きく変わってくる.

高性能アプリケーションを記述するためには，アーキテクチャの性能を引き出すようなコーディングが必要であるが，複雑化するアーキテクチャがアプリ開発者へ要求するスキルが高くなっていく問題も生じる.

一方，ハードウェアとソフトウェアのライフサイクルを考えてみると，最先端のスパコンの寿命は長くて 5 年程度であるのに比べて, アプリケーションは長いものでは 20 年を超えて使われ続けていく．ものづくりの分野では，シミュレーションとともに実験による比較・確認が行われ，シミュレーションの妥当性をチェックしながら設計検討を行ってきたためである．すなわち，長期にわたる両者の相関データベースの蓄積が，ひとつの重要なノウハウになっている．したがって，ある時点からアプリケーションの特性が変わるとそれまで蓄積してきたデータベースが使えなくなってしまうという問題点があるため，アプリケーションの長期的な継続性の保証が重要である．このため，既存のアプリケーションを新しいアーキテクチャの計算機に移植する作業が必須となる. この場合ターゲットとなるアーキテクチャは標準規格をサポートするだけでなく，移植が容易であることと，チューニングと呼ばれる高性能化作業を効率よく実施できることがアプリケーションの開発と利用の上で重要である。これは，新しくコードを開発する場合においても同様に必要であり，従来型のデバッグ作業中心のツール環境と対比して、より汎用的で移植性が高く，より高性能指向であるという意味で高生産性開発環境と称される．コード開発の効率化という点では，高生産性の開発環境への期待は大きい.

上述したように，高性能化のため複雑化する計算機アーキテクチャに対して，高性能を達成するプログラム開発には，アーキテクチャの理解，並列アルゴリズムとチューニング技術の習得，物理シミュレーションアルゴリズムなど，それぞれ専門性が高く，かつ多岐にわたるスキルが必要となる．このような状況におけるプログラム開発の高生産性が大きなチャレンジのひとつと位置づけられる.

３．関連研究

生産性を高めるプログラム開発の方法として，これまでライブラリやフレームワークなどが利用されてきた．ライブラリとは，ある分野の有用かつ汎用的なサブルーチン群で，アプリケーションの構築法には立ち入らないように抽象化されたプログラム部品と位置づけられる．例えば，行列計算ライブラリや FFT ライブラリなどである[3-7]．これに対してフレームワークは，特定種のアプリケーションに再利用できるコンポーネント群であり，アプリケーションアーキテクチャを形成する．例えば，流体計算フレームワークなどがあり，抽象化レベルや目的に応じて様々なものが提案されている[8-16].

これらのライブラリ/フレームワークなどのミドルウェアは，計算科学ソフトウェア構築に必要な共通的機能を提供するものであり，複数の異なる計算機アーキテクチャに対してチューニングされたサブルーチンを提供することもできる．この点で，自動チューニング機構[17]の実装提供が可能になる点は大きなメリットとなる.

その他の高生産性に関する研究としては，並列ライブラリ[18]や OpenMP[19], UPC[20], Co-array Fortran[21], XcalableMP[22]などの並列処理言語系や，ある特定の領域のシミュレータを記述するための言語 DSL(Domain Specific Language)などがある．ソース変換を行うような Rose compiler[23]や，プログラムの最適化を目的として設計された LLVM[24]などが，DSL を記述するために利用されることもある．

４．複雑なアーキテクチャに対する効率の良いアプリケーション開発

並列アプリケーションは，ハードウェアとアプリケーションの間で，図 1 に示すように様々なレベルのソフトウェアスタックから構成される．高性能かつライフサイクルの長いアプリケーションを構築するためには，うまく抽象化されたソフトウェア部品であるミドルウェア群を効果的に使いながらアプリケーションを記述していくことが望ましい．逆に，数値計算ミドルウェアの構築は，できるだけ多くのアプリケーションの主要演算を抽象化し、その機能と高性能実装を提供することにある．このアプローチにより， プログラミング生産性を格段に高め、高性能化を実現するとともに，開発コスト・メンテナンスコストの削減が期待できる．

多くの研究者や技術者がエクサスケール計算機を使い，幅広い成果を創出するため，複雑な問題を少ないコーディング労力でプロトタイピングするフレームワークを提供するためには，エクサ向きのアルゴリズムの共通的な演算パターンや通信パターンの分析が手がかりとなる．アプリケーション主演算は、たとえば以下の尺度で分類できるだろう[26, 白書].

- 計算機ノード内におけるメモリアクセスパターン
- 計算機ノード間の通信パターン
- 通信回数（通信レイテンシ）と通信量（通信バンド幅）
- I/O 性能（データアクセス頻度とデータ量）
- その他アプリケーションから抽出できる尺度

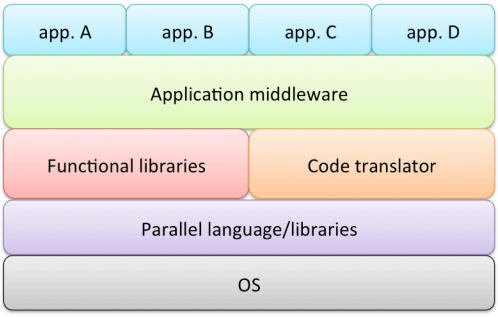

Fig. 1　Software stack.

ミドルウェア開発の課題としては，再利用性を考慮したミドルウェア機能の抽象化レベルと実装がある．この点は，利用する側のアプリケーションユーザとミドルウェアを開発する側の計算幾科学の研究者とのコラボレーション（Co-design）を進めていく必要がある．

５．おわりに

エクサスケール計算機とアプリケーションの開発に向けて，ソフトウェアライフサイクルの視点から効率的な開発方針を検討した．様々なアーキテクチャへの対応や性能可搬性，最適チューニングなどの性能面とプログラムの開発効率，継続性の点からはミドルウェアを利用したコード開発が期待される．その一方で，ミドルウェアの開発整備には，エクサで利用するアルゴリズムをうまく抽象化した設計が要求され，計算機科学の研究者とアプリ開発を行う計算科学の研究者のコラボレーションが重要になってくる．

参考文献

(1) http://www.exascale.org/bdec/
(2) http://hpci-aplfs.aics.riken.jp/
(3) http://www.netlib.org/lapack/
(4) http://www.netlib.org/scalapack/
(5) http://www.fftw.org/
(6) Balay, S., Gropp, W., McInnes, L.C., and Smith, B., "PETSc 2.0 Users Manual, Rv. 2.0.16," Technical Report ANL-95/11 (1997)
(7) http://trilinos.sandia.gov/index.html
(8) Baden, S.B., Colella, P., Shalit, D., Van Straalen, B., "Abstract KeLP", 10th SIAM Conference on Parallel Processing for Scientific Computing, Portsmouth, Virginia, March (2001)
(9) Wijesinghe, H.S., Hornung, R.D., Garcia, A.L., and Hadjiconstantinou, N.G., "Three-dimensional Hybrid Continuum-Atomistic Simulations for Multiscale Hydrodynamics," Journal of Fluids Engineering, Vol 126, pp. 768-777 (2004)
(10) Hornung, R.D., and Kohn, S.R., "Managing Application Complexity in the SAMRAI Object-Oriented Framework," in Concurrency and Computation: Practice and Experience (Special Issue), 14, pp. 347-368 (2002)
(11) Henshaw, W.D., “Overture: An Object-Oriented Framework for Overlapping Grid Applications,” AIAA conference on Applied Aerodynamics (2002), also UCRL-JC-147889
(12) Goodale, T., Allen, G., Lanfermann, G., Masso, J., Radke, T., Seidel, E., and Shalf, J., “The Cactus Framework and Toolkit: Design and Applications,” Vector and Parallel Processing - VECPAR '2002, 5th International Conference, Springer (2003)
(13) 太田，白山，”オブジェクト指向フレームワークによる流体計算統合環境”，日本計算工学会論文集，No. 19990001 (1999)
(14) Ono, K., Tamaki, T., and Yoshikawa, H,, ”Development of a framework for parallel simulators with various physics and its performance,” Lecture Notes in Computational Science and Engineering, Springer, Vol. 67, pp.9-18, (2009).
(15) http://www.openfoam.com
(16) https://simtk.org/home/openmm
(17) Naono, K., Teranishi, K., Cavazos, J., and Suda. R. eds., “Software Automatic Tuning,” Springer, 2010
(18) http://www.open-mpi.org
(19) http://openmp.org/wp/
(20) http://upc.gwu.edu
(21) http://www.co-array.org
(22) http://www.xcalablemp.org
(23) http://rosecompiler.org/ROSE_HTML_Reference/main.html
(24) http://llvm.org
(25) Maruyama, N., Nomura, T., Sato, K., and Matsuoka, S., "Physis: An Implicitly Parallel Programming Model for Stencil Computations on Large-Scale GPU-Accelerated Supercomputers," Proceedings of the 2011 ACM/IEEE Conference on Supercomputing (SC'11), pp. 1-12, (2012)
(26) http://www.open-supercomputer.org/workshop/sdhpc/