9

第9章

他のインテルツールとの共存

インテルでは、FORTRAN アプリケーションのパフォーマンス向上に役立つさまざまなツールとライブラリを提供しています。

9-1 パフォーマンスサイクル

アプリケーションの最適化を計画し、最終的な目標を達成するにはいくつかの手順による作業を行います。パフォーマンスの問題を識別し、解決するにはパフォーマンスサイクルと呼ばれる次に示す手順を考えてください。

- パフォーマンスデータの収集
- データの解析と問題の認識
- 問題の解決
- 拡張の実装
- 結果のテスト

次の図は、手順のフェーズと関連性、および各フェーズで推奨するツールを示しています。

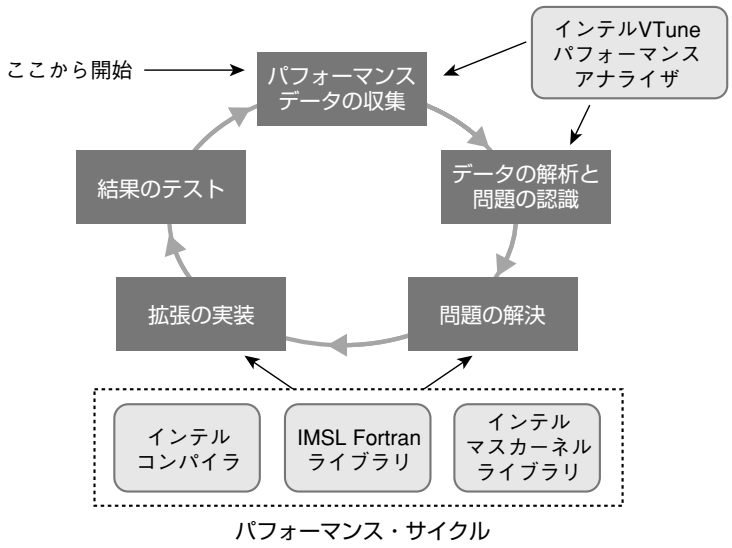

図 9.1 ●手順のフェーズと関連性、および各フェーズで推奨するツール

パフォーマンスサイクルにおける手法の要約は次のとおりです。

- 変更はできるだけ局所的に、測定はできるだけ多く行う。
- これ以上最適化を行っても効果がないと判断した場合、他にパフォーマンス問題がなければ、最適化を終了する。

パフォーマンスデータの収集

ツールを使用して、パフォーマンスのボトルネックが発生している場所を測定します。適切な解析ツールを使用すると、各段階で実装された変更や向上を判断するための客観的なデータやベースとなる基準を識別することができます。インテル VTune パフォーマンスアナライザは、パフォーマンスデータを収集し、コード全体の実行時間を測定したときに、コードの実行が遅くなっている領域、ほとんど実行されない領域、または頻繁に実行されている領域（hotspot）を素早く識別できるツールです。

データの解析と問題の認識

収集したデータがアプリケーションのパフォーマンスに関する目標を満たしているかどうかを判断します。目標値を満たしていない場合、一度に調べるパフォーマンス問題を 1 つに限定します。調査する範囲を限定することは効率的に最適化を行うには重要です。

ほとんどの場合、最初に hotspot を解決することで最良の結果が得られます。hotspot は、たいてい過度のアクティビティや遅延が原因であるため、hotspot を解決することで、他のパフォーマンス問題も明らかになります。

VTune アナライザ、または他のパフォーマンスツールを使用して、集中するべきポイントを見つけます。

問題の作成

解析フェーズと同じように、作業の範囲を限定します。注目している問題領域の解決策を作成することに専念し、問題を解決するためのツールと手法を判別して使用します。例えば、コンパイラの最適化、インテルパフォーマンスライブラリのルーチン、または他の最適化（メモリアクセスパターンの改良、除算または他の浮動小数点演算の精度の考慮、組み込み関数やアセンブリコードによるコーディング、その他の手法）を使用します。

拡張の実装

前のフェーズと同じように、実装の範囲を限定します。変更は個別に行います。一度に多く

の問題を処理しようとすると、目的が定まらず、拡張の有効性を効率的にテストできなくなります。

最も簡単な拡張は、一般的なコンパイラの最適化を有効にすることです。ライブラリを使用することで拡張が可能なアプリケーションの場合は、インテルパフォーマンスライブラリのルーチンを実装することを考慮してください。実装する場合、呼び出しインターフェースのコーディングが必要になります。

結果のテスト

解析と実装の範囲を限定した場合、このフェーズでパフォーマンスにかなりの違いがあることがわかるでしょう。具体的にパフォーマンス達成の目標を立てておくと、どの時点で目標に達したかがわかります。個別の実装が実際にパフォーマンスの向上に役立ったかどうかを判断できるように、処理時間、1 秒あたりのフレーム数のように、具体的な値を設定しテストを行います。

まだ最適化の余地があると判断した場合、または他のパフォーマンス問題がある場合は、最初のフェーズ「パフォーマンスデータの収集」に戻ってパフォーマンスサイクルを繰り返します。

9-2 マルチスレッドソフトウェア開発サイクル

ここまでに説明したパフォーマンスサイクルはシングルスレッドもしくはマルチスレッドの特定のスレッド内の性能上の問題を検出し、解決策を実装する作業に役立ちますが、アプリケーションをマルチスレッド化する場合、もうひとつ有効な作業手順があります。

解析と設計

アプリケーションをスレッド化する場合、スレッド化する範囲を見つけることが重要です。アプリケーション全体の実行時間に対し、スレッド化の対象とするコードがどれくらいの比率を占めているかによって、スレッド化した場合のパフォーマンスの上限が決まります。例えば、全体の実行時間に対し 50% しか占めないコードをスレッド化しても、スレッドを何個増やしても性能の向上は最大 2 倍しか望めません。スレッド化のための解析では、VTune パフォーマンスアナライザのコールグラフ機能が有効です。コールグラフ機能を利用すると、関数やサブ