1G-05

シナリオテンプレートを用いた自動チューニングに関する研究*

佐藤 大智[1] 平澤 将一[2] 滝沢 寛之[2,1] 小林 広明[1]
Daichi Sato Shoichi Hirasawa Hiroyuki Takizawa Hiroaki Kobayashi
[1] 東北大学大学院情報科学研究科 〒980-8579 宮城県仙台市青葉区荒巻字青葉 6-6-01†
[2] 東北大学サイバーサイエンスセンター 〒980-8578 宮城県仙台市青葉区荒巻字青葉 6-3‡
{da175@sc.cc, hirasawa@sc.cc, takizawa@, koba@}tohoku.ac.jp

概要

近年，HPC システムにおいて高性能を達成するためには，アプリケーションコードを各システム向けにチューニングする必要がある．チューニングの自動化のためには，チューニングの手順を記載したシナリオをアプリケーション開発者が独自に記述する必要がある．シナリオは個々の事例ごとに記述する必要があり，再利用性が低い．本研究では，シナリオの再利用性を高めてアプリケーション開発者の労力を軽減することを目的とし，簡単なディレクティブの記述で構築可能な典型的なチューニングシナリオのテンプレートを用いる自動チューニング機構を提案する．評価では，自動チューニングによる性能最適化の効果，およびシナリオテンプレートとインタフェースによって再利用性の高いチューニングシナリオを構築できることを示す．

1 緒言

近年，高性能計算 (High-Performance Computing, HPC) システムの構成は，従来と比較してより複雑化しており，対象システムにアプリケーションコードを適応させる (以下，チューニングと呼ぶ) 作業には膨大な労力を要する．このため，その労力を削減することが重要な課題となっている．この課題に対して，自動性能チューニングまたは自動チューニングと呼ばれる技術が近年特に注目されている．自動チューニングは対象システムにアプリケーションコードを適応させる作業を自動化できるため，プログラマの労力を大幅に軽減することができる．HPC システムアーキテクチャの多様化は加速しており，エクサスケールの時代では自動チューニングがさらに重要である．

近年，自動チューニングを行うシステムソフトウェア (以下，自動チューニング機構と呼ぶ) を構築するためのフレームワークが開発されている [1]．このフレームワークを用いて構築した自動チューニング機構を既存のコードに適用するためには，チューニングの手順をアプリケーション開発者が独自に記述する必要がある．以下，チューニング手順の記述を**チューニングシナリオ**と呼ぶ．ここで，チューニングシナリオは標準的な書き方が決まっておらず個々の事例に対して個別に記述する必要があるため，シナリオの再利用性が低いという問題がある．例えば，カーネルループに対してキャッシュブロッキングを適用する場合，ブロックサイズの探索範囲を指示し，その範囲内でブロックサイズを変化させながらコンパイル，実行，および性能比較を行う手順をアプリケーションごとに記述する必要がある．

一方，典型的なシナリオを想定した場合，チューニングシナリオの記述には共通して利用可能な部分があるため，共通して利用可能な部分をまとめることで，シナリオのひな型 (**シナリオテンプレート**) を作成可能である．さらに，個々の事例に必要な情報を記述するためのインタフェースを開発することで，再利用性の高いチューニングシナリオを実現できる可能性がある．

そこで，本研究では典型的なシナリオを想定し，チューニングシナリオのひな型となるシナリオテンプレートと，シナリオテンプレートとチューニングに必要な情報を連携させるためのインタフェースを設計する．

2 自動チューニングフレームワークとコード変換

これまで，パラメータの調整を必要とする性能最適化手法は数多く研究されているが，その調整を行う自動チューニング機構は独自に構築されることが多い．そこで近年，自動チューニング機構を構築するための環境として，自動チューニングフレームワーク OpenTuner [1] が開発されている．OpenTuner を用いることで，様々な事例に対して自動チューニング機構を構築することができる．

一方で，OpenTuner は既にパラメータを調整することで性能が向上するように書かれているコードを対象としている．そのようなコードを**自動チューニング可能なコード**と表現する．OpenTuner を用いて自動チューニングを行う場合，まずは既存のコードを自動チューニング可能なコードに修正する必要がある．自動チューニング可能なコードには，チューニング時に変更されるパラメータや複数のコード変種が含まれる場合が多く，元のコードよりも複雑で可読性が低くなる傾向にある．

先行研究 [2] では，ユーザが独自に定義できるコード変換 [3] を利用して，コードを自動チューニング可能なコードに修正することでこの問題を解決している．元のコードを自動チューニングする直前にのみ自動チューニング可能なコードに変換することで，アプリケーション開発者は変換後のコードを意識せずにアプリケーションコードの保守を続けることができる．しかしながらチューニングシナリオは依然として個々の事例ごとに記述する必要があるため，チューニングシナリオの再利用性は低い．

そこで本研究では，コード変換を用いることで直接的なコードの修正を回避しながらチューニングシナリオの再利用性を高めることを目的とする．

3 シナリオテンプレートを用いた自動チューニング

本研究ではチューニングシナリオの再利用性を高めてアプリケーション開発者の労力を軽減するために，典型的なシナリオを想定したシナリオテンプレートを設計する．さらに，個々の事例に必要な情報を記述するインタフェース (**個別情報記述インタフェース**) を設計し，アプリケーション開発者が容易に自動チューニングを適用できる機構を提案する．

本論文で提案する自動チューニング機構では，元のコードと変換ルールを入力とする．出力はチューニングされたコードと最適なパラメータである．本提案機構によって実現され

---

* An Auto-tuning System using Scenario-Template
† Graduate School of Information Sciences, Tohoku University 6-6-01 Aramaki-Aza-Aoba, Aoba, Sendai 980-8579, Japan
‡ Cyberscience Center, Tohoku University 6-3 Aramaki-Aza-Aoba, Aoba, Sendai 980-8578, Japan

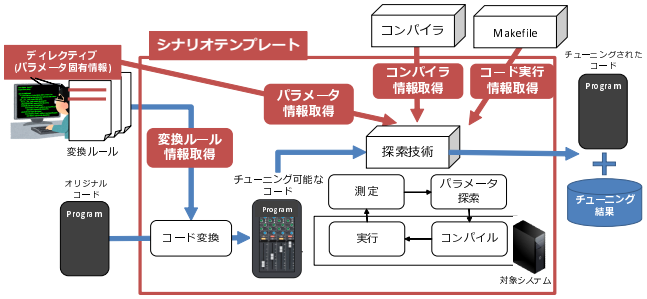

図 1　提案機構による自動チューニングの流れ

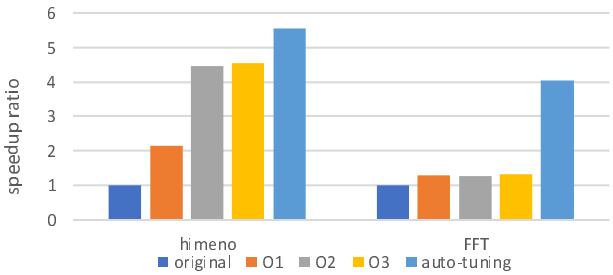

図 2　自動チューニングの結果

る自動チューニングの過程の概要を図 1 に示す．はじめに，ユーザは元のコードを自動チューニング可能なコードに変換する変換ルールを定義する．このとき，ユーザは自動チューニングに必要なパラメータ情報 (パラメータ名，探索空間，パラメータタイプ，パラメータの優先度) と変換ルール情報 (変換対象コード，変換ルールの適用順) をディレクティブとして変換ルールに挿入する．その後，ディレクティブから取得した情報をもとに，シナリオテンプレートはコード変換機構を用いて元のコードを自動チューニング可能なコードに変換する．また，ユーザはコマンドラインの引数によって，自動チューニングを行うためのコンパイルと実行の手順についても指示する．シナリオテンプレートは，これらの情報を用いてコンパイル，実行，および探索を繰り返し行うことで最適なパラメータを決定する．

4　性能評価

提案機構について，自動チューニングの効果とチューニングシナリオの再利用性を評価する．ベンチマークとして，姫野ベンチマークと並列 FFT アプリケーション [4] を用いる．姫野ベンチマークでは性能に関する 2 つのパラメータと 455 種類のコンパイラオプションついてチューニングを行う．並列 FFT では，通信隠蔽に関する 3 つのパラメータとコンパイラオプションについてチューニングを行う．2 つのアプリケーションに対する自動チューニングのパラメータは異なるため，提案機構を用いない場合，それぞれ個別にシナリオを記述する．

はじめに，典型的なシナリオを用いた自動チューニングによる性能最適化について評価する．比較対象は元のコードをコンパイラオプション O0，O1，O2，O3 で実行した場合の実行時間である．これらの実行では，ループの最適化やデータレイアウトの変更などのパラメータは元のコードと同様に設定する．図 2 に各実行時間を元のコード (O0 による実行) について正規化した速度向上率を示す．図 2 より，提案機構を用いたことで姫野ベンチマークでは 5 倍以上，並列 FFT についても約 4 倍の高速化を達成している．また提案機構は，高性能を達成することで知られている O2，O3 よりも高性能を達成したことが分かる．よって，提案する自動チューニング機構では自動チューニングによる高い性能最適化の恩恵を享受できることが分かる．

次にチューニングシナリオの再利用性について，典型的なシナリオの共通な行数とチューニングシナリオの記述に要するコード行数を用いて評価する．従来の自動チューニング機構を用いて典型的なシナリオに沿ったシナリオを記述すると，姫野ベンチマークでは 302 行，並列 FFT では，319 行のシナリオの記述が必要である．このとき，2 つのシナリオで共通化できる行数は 278 行である．このことから，典型的なシナリオに沿ってシナリオを記述することで，シナリオの大部分を再利用できることが分かる．一方，2 つのシナリオではパラメータやコード変換に関する記述が異なるため，2 つのアプリケーション間でシナリオをそのまま再利用することができない．このことから，典型的なシナリオを記述して個々の事例で異なる情報を指示することで，高い再利用性を持つシナリオテンプレートを構築できることが分かる．

提案機構では，シナリオテンプレートと個別情報記述インタフェースを用いることで，変換ルールへのディレクティブの挿入のみで自動チューニング機構を実現できる．姫野ベンチマークにおいて変換ルール内で定義されるパラメータは 2 種類のため，パラメータに関するディレクティブは 2 行となる．また，姫野ベンチマークでは 4 種類のコード変換を組み合わせてコード変換を行うため，変換に関する指示として変換ルールの順番を指示する 4 行のディレクティブを挿入する．挿入されたディレクティブをもとに，パラメータの組み合わせを持つようなコード変種を持つ自動チューニングな可能なコードに変換を行う．次に，変換されたコードをコンパイル，実行，および測定を繰り返すことによって最適なパラメータを決定する．このとき，コンパイラオプションのチューニングも同時に行う．このことから，姫野ベンチマークでは全部で 6 行のディレクティブを挿入するだけで自動チューニングを実現することができる．同様に並列 FFT では，6 行のディレクティブを追記することで自動チューニングを実現可能である．したがって，提案機構を用いることでチューニングシナリオをすべて記述する場合と比べて大幅に少ない行数で自動チューニングを実現できることが分かる．

5　結言

本研究では，チューニングシナリオの再利用性を高めてアプリケーション開発者の労力を軽減することを目的とし，簡単な記述で構築可能なシナリオテンプレートを用いる自動チューニング機構を提案した．評価より，シナリオテンプレートと個別情報記述インタフェースを開発することで再利用性の高いチューニングシナリオを構築できることを示した．

参考文献

[1] J. Ansel et al., ”OpenTuner: An Extensible Framework for Program Autotuning,” in 23rd ACM International Conference on Parallel Architectures and Compilation Techniques(PACT), pp. 303-316, 2014.

[2] H. Takizawa et al., ”Making a Legacy Code Auto-tunable without Messing It Up,” in International Conference on Supercomputing, pp. 1-2, 2016.

[3] H. Takizawa et al., ”Xevolver: An XML-based Code Translation Framework for Supporting HPC Application Migration,” in 21st IEEE International Conference on High Performance Computing (HiPC), pp. 1-11, 2014.

[4] 高橋 大介, ”並列 FFT における通信隠蔽の自動チューニング,” 計算工学講演会論文集, Vol. 21, pp. 1-4, 2016.