要 旨

コードシェーカ再訪:Haswell と Skylake アーキテクチャ上での コード配置効果を考慮した適正な性能評価

坪内 洋平

近年，アーキテクチャの技術が発達するに伴ってプログラムの動作する環境が複雑化している．動作環境には，キャッシュメモリや投機実行メカニズムなど性能向上の仕組みが備わっており，プログラムを実行すると常にこれらの影響を受ける．これは，ソフトウェアの開発においてプログラムの小規模な変更による実行速度の変化を調べたいときに問題になる．同じプログラム，同じハードウェアであっても，OS によってプログラムがロードされるアドレスが異なるとプログラムの実行速度が変化（以下，コード配置効果）するため，プログラムの実行速度の計測が困難になる．そこで，コード配置効果の影響を考慮したプログラムの実行速度を計測する方法が必要になる．

松田らの提案したコードシェーカ [1, 2] はコード配置効果の影響を考慮してプログラムの実行速度を計測する手法である．問題は松田らのコードシェーカの性能評価実験は Nehalem 世代以前のアーキテクチャ上で行っている点である．現在，最新のアーキテクチャは Skylake 世代であり，キャッシュの速度やキャッシュサイズなどが変わっている．これにより，例えばコード配置効果が小さくなってコードシェーカが不要になるかもしれない．そこで，Nehalem 世代以降のアーキテクチャ上でコードシェーカを使ってコード配置効果を考慮した実行速度の計測ができるか検証する必要がある．

本論文では，コードシェーカを再実装した．正しく実装できているか確かめるため，松田らのコードシェーカの性能評価実験で使用した Nehalem アーキテクチャと同世代の CPU

を使ってプログラムの性能比較を行った．その結果，松田らのコードシェーカと比べてコード配置効果が小さかったが，松田らの実験と同じような実験結果の分布が得られたため，正しく実装できたと考えられる．次に，Haswell と Skylake アーキテクチャ上においてコードシェーカを使ってコード配置効果を考慮してプログラムの実行速度の計測を行うことができるか検証した．その結果，Haswell アーキテクチャではコードシェーカを使わないプログラムの実行時間の分布図に対して，コードシェーカを使ったプログラムの実行時間の分布はばらつきが大きく，コード配置効果の影響を考慮した結果が得られた．また，Skylake アーキテクチャにおいても同様の結果が得られた．これらの結果から，Haswell と Skylake アーキテクチャ上でもコードシェーカが有用であることを確認した．

キーワード　アーキテクチャ，性能評価

Abstract

Codeshaker Revisit: Performance Evaluation that Takes into Account the Code Placement Effect Programs on Haswell and Skylake Architecture

Youhei Tsubouchi

In late years, environments of program executions become complicated as CPU architecture technology advances. The execution environment is equipped with mechanisms for performance enhancement including cache memory and the speculative execution mechanism. When we execute a program, it always comes under influences of them. When we measure changes of execution speed caused by a small change of a program in software development, this is a problem. Because execution speed of a program differs, even with the same program and the same execution environment, depending on the addresses to which the program is loaded by the OS (we call this "code placement effect"), measurement of execution speed of programs is difficult. Therefore, a method to measure execution speed of programs that takes into account the code placement effect is necessary.

Matsuda et al. suggested the codeshaker that measures execution speed of programs while removing influence of the code placement effect. However, Matsuda et al. experimented performance evaluation using the codeshaker on the Nehalem generation architecture. The latest architecture is the Skylake generation, whose cache size and speed are different. This may, for example, reduce significance of the codeshaker be-

cause the code placement effect may becomes small. Therefore, it is necessary to inspect whether we can measure execution speed taking into account the code placement effect by using the codeshaker on architectures newer than the Nehalem generation.

In this thesis, we replicated the codeshaker. We firsr ensured that our implementaion of the codeshaker is proper by an experiment of performance comparison of programs on a Nehalem architecture CPU, which is of the same generation as the one Matsuda et al. used. As a result, though the code placement effect was smaller than the codeshaker of Matsuda et al., we ensured that we properly implemented the codeshaker because distribution of measured performance was similar to the one of the experiment of Matsuda et al. We next inspected whether we can measure execution speed of programs taking into account the code placement effect with the codeshaker on the Haswell and Skylake architectures. As a result, we found that the distribution of execute times of programs measured using the codeshaker had a larger variance than distribution of execute times of programs measured without using the codeshaker. This means that the codeshaker takes into account the code placement effect on the Haswell architecture. A similar result was observed in the Skylake architecture as well. From these results, we conclude that the codeshaker was useful even on the Haswell and Skylake architectures.

key words architecture, performance evaluation