PDS 最適化サイクル: Cell BE プログラムの最適化戦略の提案

里城 晴紀　小長谷 明彦
東京工業大学 情報理工学研究科 計算工学専攻

マルチコアプロセッサ上での並列プログラミングの最適化において頻繁に生じるメモリアクセスボトルネックを容易に検出し, 性能向上に向けた適切な最適化手法の選択を可能とする最適化戦略について報告する. 本手法では,Plan-Do-See のサイクルに基づき, (plan) メモリアクセスネック, パイプラインハザードなどの性能阻害要因の特定と, (do) ダブルバッファリングやループアンローリングなどの阻害要因を取り除くための最適化手法の選択, (see) 最適化効果の測定を繰り返すことで効率的な並列プログラミングのチューニングを可能とする. 具体例として, 粒子を並列単位とした細粒度モンテカルロシミュレーションを題材に取り上げ, そのチューニング過程において本手法の有用性を示す.

1　はじめに

近年, トランジスタ集積数の増大, クロック周波数向上の限界, 回路設計コスト低減化の観点から, 高性能プロセッサを実現するアーキテクチャとして, Cell BE のような同一のコアを多数搭載したマルチコアアーキテクチャが注目を集めている [1]. このようなマルチコアアーキテクチャでは多数の演算器と高速な通信性能を活用したオンチップ並列処理による高性能化が期待できる [2]. しかしながら, チップ内での演算性能は向上できても, メモリアクセス性能を向上させることは困難であるため, 通常のマイクロプロセッサよりもメモリアクセスボトルネックが生じやすい [3]. このため, 最適化手法を適用してもその効果が実機上での性能向上に適切に反映されず, 並列プログラミングのチューニングを困難なものとしている. PDS 最適化サイクルでは, メモリアクセスボトルネックなどの性能阻害要因を検出することにより, 適切な最適化手法の選択を可能とする [4]. 例えば, メモリアクセスボトルネックが生じている状況において, ループアンローリングを適用しても, 計算時間のチューニング効果はメモリアクセスの待ち時間に隠されてしまい, 実機上では性能時間の短縮効果は期待できない. 逆に, ループアンローリングを適用した結果, 潜在化していたメモリアクセスボトルネックが顕在化してしまい, 最適化の効果が思ったよりも得られなかったという場合も考えられる. いずれにしても, 並列プログラミングのチューニングを効率的に行うためには, 現在生じている性能阻害要因を正しく同定し, 性能阻害要因を取り除くための最適化手法を選択するための方法論の確立が鍵となる.

2　Cell BE シミュレータ

性能阻害要因を同定するために,PDS 最適化サイクルでは Cell BE シミュレータ [5] の提供するシミュレーションモードの内, fast モードと SPU Pipeline モード（pipe モード）を使用する.Fast モードでは各 SPE は 1 サイクルに 1 命令を確実に実行する. 通信遅延およびパイプライン効果は無視されているため, 実機での性能が fast モードでの予測性能よりも遅い場合は, メモリアクセスボトルネックまたはパイプラインハザードが生じていることを示唆している.

Pipe モードは SPE 個別に設定できるモードで, 命令パイプラインをシミュレートする. pipe モードにおいても通信遅延は無視されているため,fast モードと pipe モードにおける予測性能の差はパイプラインハザードによる性能遅延を, pipe モードにおける予測性能と実機との性能差はメモリアクセスボトルネックによる性能遅延を示唆している（図 1）.

一方,fast モードではパイプライン効果が反映されていないため, メモリアクセスボトルネックやパイプラインハザードが生じていない場合は, 実機での性能が fast モードでの予測性能を上回ることが予想される. 2 本の演算パイプラインが最大限に活用された場合は実機での性能は fast モードの半分となり, これがチューニングにおける目標性能となる.

3　PDS 最適化サイクル

PDS 最適化サイクルは Target,Plan,Do,See の 4 ステップを繰り返して SPE プログラムを最適化効果の高い部分から順に最適化する（図 2）. 本節では PDS 最適化サイクルの各ステップについて述べる.

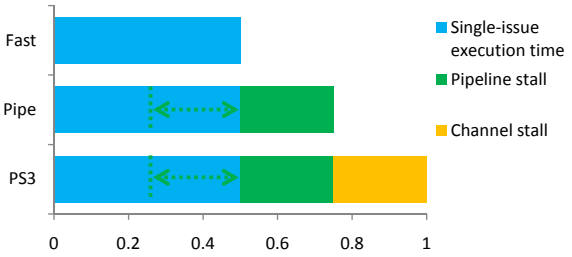

図 1: Fast モード,pipe モード,PS3 の実行時間とその差の原因. パイプラインストールの時間は 2 命令同時発行率によって変動する.

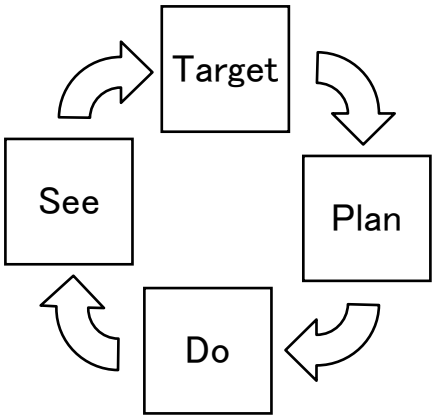

図 2: PDS 最適化サイクル. 繰り返しにより効果の高い部分から順に最適化する.

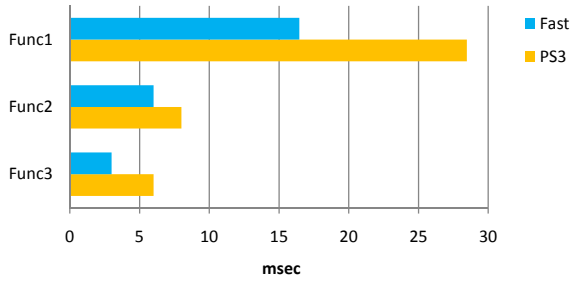

図 3: Target ステップ. 関数ごとの fast モードと PS3 上での実行時間の例. Func1 の Fast モードと PS3 との実行時間の差が最大なので,Func1 で最大最適化効果が期待できる.

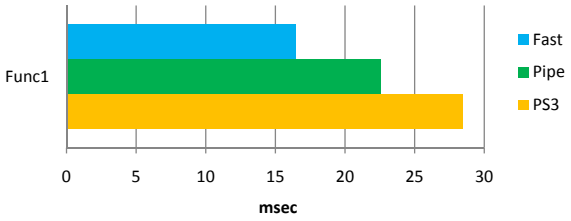

図 4: Plan ステップ.Fast モード,pipe モード,PS3 上での実行時間の例. この例ではパイプラインハザードとデータ転送コストの両方に性能阻害されている.

3.1 Target

最初に最適化効果が大きいと見られる箇所を発見する. そのために,fast モードと PS3 上でのプロファイリングを行う（図 3）. Fast モードと PS3 での実行時間の差が最も大きい部分で最大の最適化効果が期待できる. プロファイリングには SPE のデクリメンタで時間を計れる [6] ので, それを利用できる.

3.2 Plan

最適化箇所を決定したなら, 性能ボトルネックを特定し, 計画を立てる（図 4）. ここで pipe モード上でのプロファイリングを行い, Fast モード,PS3 での実行時間と比較する. 最適化計画は以下の優先順序で立てる.

1. PS3 と pipe モードでの実行時間に大きな差が見られるときはメモリアクセスの最適化
2. Fast モードと pipe モードでの実行時間に大きな差が見られるときはパイプラインハザードの除去
3. 冗長な命令の除去

冗長な命令の除去については実機とシミュレータとの比較では発見できないので注意が必要である. もし,fast モードの 2 倍の性能を引き出せたとしても, 期待より著しく低い性能となる場合には冗長な命令が多いと判断できる. 特に SIMD 化については全体性能への影響が大きいので,PDS 最適化サイクルを適用する前にあらかじめ済ませておくことが望ましい.

3.3 Do

特定した性能ボトルネックに対して適切な最適化技法を施す. ループアンローリングは命令を並べ替えることでパイプラインハザードの発生を抑える. ダブルバッファリングはデータを先読みすることでメモリアクセスの時間を隠蔽する. キャッシングはローカルストア内にデータを保持することで冗長なメモリアクセスを除去できる. これらの最適化技法については [7, 8] が詳しい.

3.4 See

最後に最適化の効果を評価するために, 再度 fast モードと PS3 でのプロファイリングを行う. PS3 での実行時間が fast モードでの実行時間よりも短くなったならば, 最適化の効果が十分にあったといえる. そうでなければ, 最適化の余地があるので PDS 最適化サイクルを繰り返す.

4 PDS 最適化サイクルの適用

本節では 2 つのプログラム, 座標変換プログラムとモンテカルロ粒子シミュレータを例題に,PDS 最適化サイクルの効果について述べる. Cell Speed Challenge 2008 規定課題の連立一次方程式求解プログラム [3] においても PDS 最適化サイクルはメモリアクセスボトルネックの発見に役立った.

4.1 座標変換プログラム

最初に簡単な例として,100 万本の 4 次元ベクトルに 4 × 4 の行列をかけるプログラムの最適化を行った. このプログラムをテストケースとしたのは,Cell BE の得意とする行列計算であり, 最適化の効果を得やすいからである. SIMD 化には Cell BE 用に用意された C++の言語拡張機能 [9] を参考にした.

図 5 に PDS 最適化サイクルの各ステップにおいて適用した最適化手法と得られた性能向上の関係を示す. 最適化によって計算時間は 26 ミリ秒から 7 ミリ秒に減少した. 第 1 サイクルではデータ転送時間を隠蔽するためにダブルバッファリングを実装した. これにより, データ転送時間はほとんど隠蔽できた. 理論上計算を終了するまでには 1 ミリ秒かかるが, 実行時間はそれよりも非常に長いため第 2 サイクルでは制御命令の除去を行った. 関数のインライン化などにより命令数は半減した. 第 3 サイクルではパイプラインハザードを除去するために, ループアンローリングを行った. パイプラインハザードはほとんど除去されて PS3 での実行時間は fast モード上の実行時間より短くなった.

4.2 モンテカルロ粒子シミュレータ

モンテカルロ粒子シミュレーションは多数の粒子の相互作用からの高次機能の創発を時空間的にシミュレートする方法として注目されている [10]. 今回はテストケースとして, 強磁性微粒子の凝集現象のモンテカルロシミュレータ [11] を Cell BE 用に改変し, 最適化を行った. 乱数生成には SIMD-oriented Fast Mersenne Twister[12] を利用している.

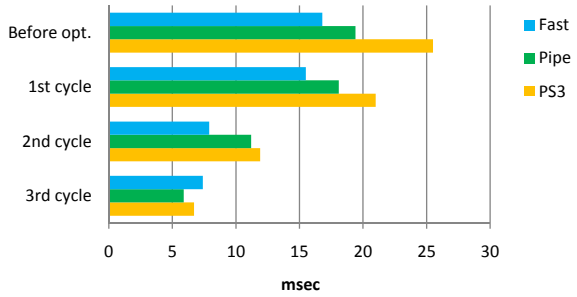

図 5: 座標変換プログラムの PDS 最適化サイクル適用後の実行時間. 各サイクルでの最適化技法は,（1st cycle）ダブルバッファリング,（2nd cycle）制御命令の除去,（3rd cycle）ループアンローリングである.

モンテカルロ粒子シミュレーションの並列化にあたっては, 領域を空間的に分割し各 SPE に割り当てる空間分割法と粒子単位に各 SPE に割り当てる細粒度分割法がある. 前者では, 凝集の進行により, 部分空間内での粒子密度のばらつきが生じると各ＳＰＥでの実行時間のばらつきが大きくなり, 結果的に並列処理の効果が失われるという問題を生じる. このため, 本テストケースでは, 粒子単位に各 SPE に割り付ける細粒度並列処理を採用し, 粒子の位置エネルギーの計算について,PDS 最適化サイクルを適用した. 乱数生成の実行時間はシミュレーション時間と比較して軽微なため PPE プログラムから SPE プログラムへの変換はしていない.

図 6 に PDS 最適化サイクルの各ステップにおいて適用した最適化手法と得られた性能向上の関係を示す. 粒子の位置エネルギーの計算を最適化することで, 最適化前は 65535 個の粒子からなる系において,1000 ステップ実行するのに 107 秒かかっていた計算が, 最適化後は 77 秒に減少した. 第 1 サイクルではデータ転送時間を隠蔽するためにダブルバッファリングを実装した. 結果, 転送時間の大部分を隠蔽することに成功した. 残りはデータの依存関係によるプリフェッチミスと考えられる. 第 2 サイクルでは, パイプラインハザードを除去するためにループアンローリングを施した. パイプラインハザードが減少し, 性能は向上した.

結果的に, モンテカルロ粒子シミュレーションにおいて並列化効率 97 パーセントを達成することができた（図 7）.

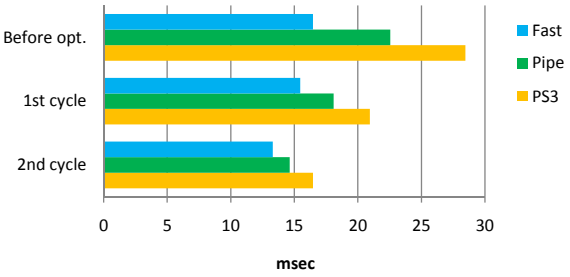

図 6: モンテカルロ粒子シミュレータの PDS 最適化サイクル適用後の実行時間. 各サイクルでの最適化技法は, (1st cycle) ダブルバッファリング, (2nd cycle) ループアンローリングである.

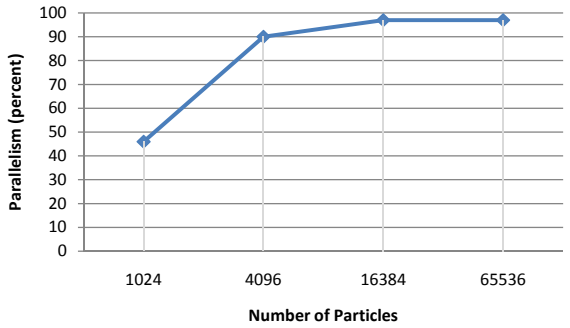

図 7: モンテカルロ粒子シミュレータにおける並列化効率.5 基の SPE による計算で最大並列化効率 97 パーセントを達成した.

5 まとめ

Cell BE シミュレータの fast モード,pipe モードと実機上での実行性能の差から性能阻害要因を同定し (Plan), 性能阻害要因を解消するための最適化戦略を適用し (Do), 得られた結果から次ぎの最適化方法を検討する (See) PDS 最適化サイクルを提案した. PDS 最適化サイクルは Cell BE のように, 演算性能が高いプロセッサにおける並列プログラムのチューニングにおいて効果を発揮する. 本手法を, モンテカルロ粒子シミュレータと連立一次方程式求解プログラムに適用し,PDS 最適化サイクルによるチューニングの効果が高いことを実証した.

参考文献

[1] J. A. Kahle, M. N. Day, H. P. Hofstee, C. R. Johns, T. R. Maeurer, D. Shippy. Introduction to the Cell multiprocessor, 2005.

[2] Thomas Chen, Ram Raghavan, Jason Dale, Eiji Iwata. Cell Broadband Engine Architecture and its first implementation, 2005.

[3] 里城 晴紀, 小長谷 明彦. オンチップ並列処理を利用した密行列 LU 分解の実装, 2008.

[4] Haruki Satoshiro, Akihiko Konagaya. PDS Optimization Cycle: New Programming Optimization Methodology for Cell BE Architecture. SACSIS 2008.

[5] IBM. IBM Full-System Simulator for the Cell Broadband Engine Processor, 2005.

[6] Sony Computer Entertainment Inc. Cell BroadBand Engine^{TM} アーキテクチャ Version 1.01, 2006.

[7] IBM. Cell Broadband Engine Programming Tutorial Version 3.0, 2007.

[8] IBM. Cell Broadband Engine Programming Handbook Version 1.1, 2006.

[9] Sony Computer Entertainment Inc. SPU C/C++ 言語拡張 Version 2.3, 2006.

[10] Ryuzo Azuma, Tetsuji Kitagawa, Hiroshi Kobayashi, Akihiko Konagaya. Particle simulation approach for subcellular dynamics and interactions of biological molecules. BMC Bioinformatics, 7(Suppl 4): S20, 2006.

[11] 佐藤 明. HOW TO 分子シミュレーション ～分子動力学法、モンテカルロ法、ブラウン動力学法、散逸粒子動力学法～. 共立出版, 第 4 章, 2004.

[12] Mutsuo Saito, Makoto Matsumoto. SIMD-oriented Fast Mersenne Twister: a 128-bit Pseudorandom Number Generator. Monte Carlo and Quasi-Monte Carlo Methods 2006, Springer, pp.607–622, 2008.