B-004

遠隔メモリ利用による Out-Of-Core OpenMP プログラムの性能評価実験

The Evaluation of Out-of-Core OpenMP programs using Remote Memories

大浦　陽†　緑川　博子†　甲斐 宗徳†

Hikari Oura　Hiroko Midorikawa　Munenori Kai

1 はじめに

1.1 背景

大規模科学技術計算など大量のメモリを用いる応用プログラムがあるが，スロット数などのハードウェア制限や電力などの問題から 1 つのコンピュータに搭載できる DRAM の容量には限界がある．大容量メモリ利用には，多くの場合，既存コードやアルゴリズムを再設計して，分散メモリである MPI 並列プログラムに変換し，クラスタ実行することが必要になる．しかし，共有メモリ（グローバルビュー）を前提として設計された逐次コードやマルチスレッドプログラムをローカルビューしか持たない MPI 並列プログラムに変換することが困難な場合や，変換は可能であってもそのコスト・時間が高い場合も存在する．そこで，高速ネットワークを用い，遠隔メモリページングを行い仮想的な大容量メモリを提供する大容量分散メモリシステム DLM(Distributed Large Memory)を開発してきた[1]．DLM ではデータを計算ノードと複数のメモリノードに分散して割り当て，ユーザプログラムを計算ノードで実行する．メモリノードにあるデータが必要な場合は，DLM 独自のページサイズ単位で転送を行う．これにより，DLM では，共有メモリ向けに設計された OpenMP や pthread プログラムなどをほとんど変更なしに実行が可能で，主メモリを超える大規模なサイズの問題を実行することが可能である．

1.2 目的

DLM において，データアクセス局所性を考慮したステンシル計算を用いて性能評価を行う．

2 性能評価実験

2.1 7 点 Stencil 計算

ステンシル計算は最も基本的な格子計算の一つである．7 点ステンシル計算は更新する 1 点のデータとそれに隣接する 6 点のデータを用いて計算を行いデータの更新していく．全ての点のデータを更新しこれを複数回繰り返す．

本実験では，データアクセス局所性の最適化として空間ブロッキングとテンポラルブロッキング[2]を用いた 7 点ステンシル計算を行い，FlashSSD を用いて 7 点ステンシル計算を実行した結果[2]との比較を行った．

本実験では，データは DRAM と DLM の 2 か所に格納する．DRAM にはテンポラルブロッキングや空間ブロッキングに使うためのデータと計算した値を格納し，DLM にステンシル計算で使用する格子のデータを格納する．

2.2 実験環境

本実験の実験環境は表 2.1 の通りである．1 ノードにつき 32GB のメモリを持つ場合と 64GB を持つ場合の 2 通りを想定して実験を行った．実行した問題サイズは 16GB, 32GB, 64GB, 128GB, 256GB の 5 通りである．実行スレッド数は 8，12，16 スレッドの 3 通りで実験を行った．繰り返しは 256 回行った．

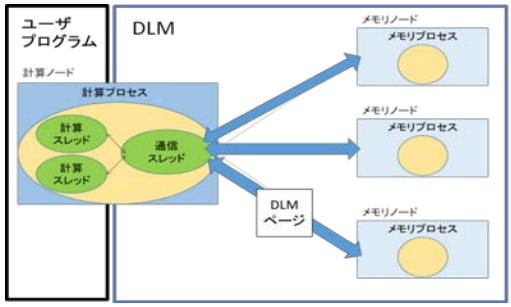

図 1.1 DLM の構成

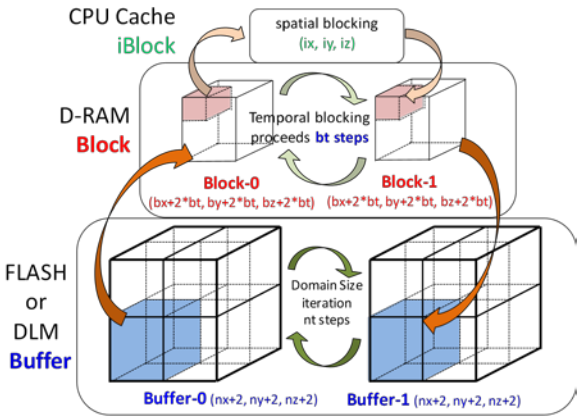

図 2.1 DLM と DRAM に格納されるデータ

CPU	Intel(R) Xeon(R) CPU E5-2687W v3 @ 3.10GHz 2CPU × 8core/node
Memory	64GB/node or 32GB/node
Cache	L2 20MB L3 200MB
Network	Infiniband FDR
OS	Linux 3.19.5
Compiler	gcc version 4.8.3
MPILib	MVAPICH2 version 2.0.1

表 2.1 実験環境

†成蹊大学理工学研究科理工学専攻 Graduate School of Science and Technology, Seikei Unicersity

2.3 実験結果

図 2.2 は計算ノードメモリ 32GiB の場合に，DLM を用いて 7 点ステンシル計算を行った時の実効 MFlops/s のグラフを示す．横軸が問題サイズ，縦軸が実効 MFlops/s を表す．問題サイズ毎に，左から順に，8 スレッド，12 スレッド，16 スレッドの実効 MFlops/s を表す．問題サイズ 16, 32, 64GiB では 12 スレッドの性能が最も高く，問題サイズ 128GB では 8 スレッドの場合が高性能であったが，大きな性能差はない．

図 2.3 は，DLM を用いて問題サイズ 16GB の 7 点ステンシル計算行ったときのスレッドごとの実行時間示す．この場合，実際には，ローカルメモリ（32GIB）に問題データがすべて収まるので，遠隔メモリアクセスは発生しない．実行時間が一番短いのは 12 スレッドで 278 秒である．DLM システム内に通信スレッドがあるため，ハードウエアコアすべてをユーザプログラムに利用するのは，一般的に効率的とは言えない．ここでは，ローカルメモリのみを利用した場合においても，計算スレッド増加による性能向上がないことから，7 点ステンシル計算がすでにメモリアクセスネック状態であり，12 スレッド以上のスレッド数増加に意味がないことがわかる．

図 2.4 は 1 ノードあたり 64GB のローカルメモリを持つと想定して DLM と SSDFlash の 7 点ステンシル計算を行った結果を比較したグラフである．問題サイズ 32GB を SSDFlash で実行したときの実効 MFlops を基準に比で表す．各問題の左が DLM，右が SSDFlash の実効 MFlops を表す．問題サイズ 32GB は DLM と SSDFlash どちらもローカルメモリだけで計算を行っている．DLM は問題サイズ 32GB から 64GB にかけての性能低下が 13%で SSDFlash の 27%よりも性能が落ちにくいが，問題サイズ 64GB から 128GB にかけての性能低下は 16%で，SSDFlash の 1%よりも性能が大きく低下した．問題サイズ 256GB での DLM の性能は問題サイズ 32GB での性能の 54%である．

2.4 考察

問題サイズ 16, 32, 64GB での実行時間が 12 スレッドで一番短かったのは，メモリ競合が発生したからだと考えられる．1 ノードにつき 32GB のメモリを持っている想定で実験を行ったので，問題サイズ 16GB はメモリノードのデータを使わず，計算ノードのデータだけで計算を行う．そのため DLM の通信性能の影響ではなく，13 スレッド以上のスレッドでは，メモリ競合が発生しスレッドの恩恵を受けられなかったと推測できる．一方．問題サイズ 128GB の実行時間は 8 スレッドが一番短い．現在の DLM の実装では，計算ノードとメモリノード間のデータの通信を行う通信スレッドの実装が，ユーザプログラムのマルチスレッドからのページ要求を逐次的に処理を行う実装であるため，より多くのスレッドからの要求が重なるとオーバーヘッドが増加する，問題サイズが大きくなると，全体の時間に占める遠隔メモリアクセス時間の割合が増え，ページ要求が多重にスレッドからくることになり，より実行時間が増えてしまったと考えられる．

図 2.4 で，ローカルメモリのみを用いる場合（32GB 問題）の実効 MFlops が DLM と SSDFlash の性能が異なる理由は，SSDFlash 向けステンシルプログラムには，より最適化が適用されていたためだと考えられる．

3 おわりに

DLM を用いることにより，ユーザプログラムのコード変更をほとんどなしに，ローカルメモリの 4 倍のサイズの問題に対し，ローカルメモリのみを利用した場合の約 64%の性能でマルチスレッドステンシル計算ができるとわかった．今後の研究では計算ノードとメモリノード間の通信方式をより効率的に行う改良を加えていく．

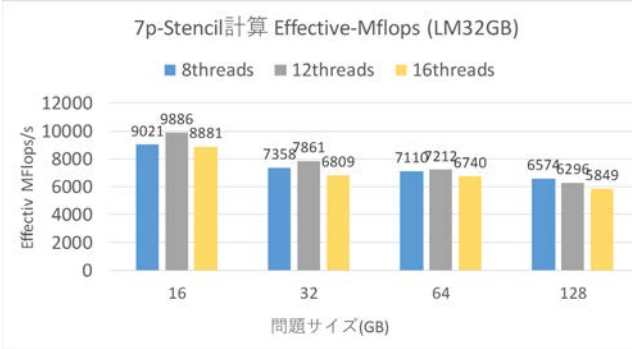

図 2.2　8,12,16 スレッドの実効 MFlops/s の比較

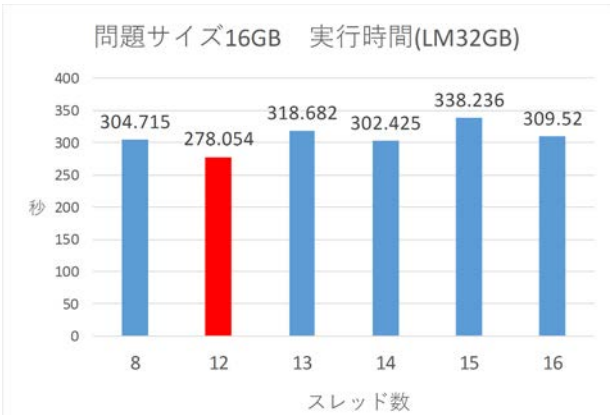

図 2.3 問題サイズ 16GB スレッド別の実行時間

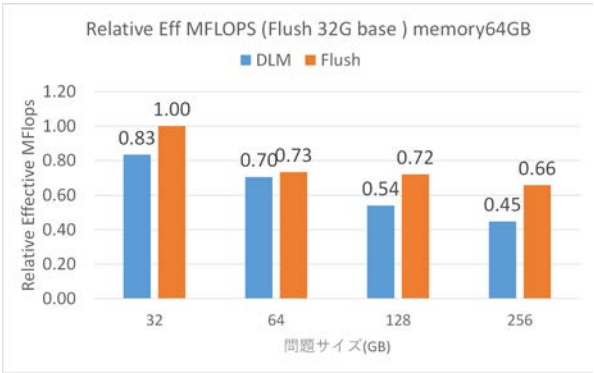

図 2.4 DLM と Flash の実効 MFlops の関係

参考文献

[1] 緑川他:"クラスタをメモリ資源として利用するための MPI による高 速大容量メモリ", 情報処理論文誌，ACS，Vol.2, No.4, pp.15-36, 2009.
[2] H.Midorikawa ,H.Tan :"Locality-Aware Stencil Computations using Flash SSDs as Main Memory Extension",CCGrid2015