TSUBAME 共同利用 平成 28 年度 学術利用 成果報告書

利用課題名 高性能計算向け分散メモリ・ストレージ統合システムの研究

英文：The Unified System of Distributed Memory and Storages for High Performance Computing

利用課題責任者
緑川 博子
所属
成蹊大学 理工学部
http://www.ci.seikei.ac.jp/midori/

邦文抄録（300 字程度）

筆者らは，高性能 Flash SSD を用い，主メモリサイズを超える大規模ステンシル計算を高速に処理するため，空間・時間ブロッキングと，マルチコアによる高並列・非同期入出力を併用したアルゴリズムを開発した．さらに，各ノードに SSD を備えたクラスタ向けのアルゴリズムを開発し，クラスタノード搭載メモリの総和を超える規模のステンシル計算問題を高速処理することを可能とした．最新 NVMe（PCIe 接続）Flash を備えたクラスタ(4-8 ノード)での性能と比較する目的で，低速かつ小容量 SATA 型 SSD ノードしかもたない TSUBAME2.5 クラスタにおいて，利用可能な 32 ノードを用い，多数ノードにおけるスケーラビリティなどを調査した．

英文抄録（100 words 程度）

This paper evaluates our novel algorithm proposed for out-of-core stencil computation using distributed memories and SSDs over a cluster. Not only the multi-tiling algorithm in temporal and spatial spaces but also a highly parallel asynchronous input/output are introduced in the algorithm to increase data access locality and to bridge DRAM and Flash SSD latency divide. The result indicates the effectiveness of our algorithm even when using traditional SATA-based SSDs distributed over multiple nodes in TSUBAME2.5.

Keywords: Out-of-core ; stencil computing; temporal blocking; asynchronous I/O; flash

背景と目的

多くの科学技術計算や工学的シミュレーションにおいては，より現実に近い大規模なサイズの問題を解くこと，より正確な結果を得るために，高解像度，細粒度でのシミュレーション計算を行うことが求められており，常に大容量のメモリを必要としている．

一方，スーパーコンピュータの各ノードに搭載できる DRAM の容量は，システム全体の消費電力や実装可能なメモリスロット数などの面で制限されている．また，一つのスーパーコンピュータを共用する様々なユーザの用途に応じた計算・メモリ利用の要求レベルは異なっていることが多く，要求の最大サイズに合わせて，システムの各ノードの搭載メモリ容量をすべて増やすには限界がある．

本利用課題では，各ノードの物理メモリサイズ（DRAM）の合計をこえるような大きなサイズの問題に対して，従来ファイルの置き場として用いられてきた SSD や HDD などのストレージも活用し，統合的なメモリシステムをソフトウエアで構築する．これにより，多様なユーザの要求レベルに対する柔軟性を確保しつつ，高性能かつ大規模なメモリ利用を可能にする．すなわち，現在，高性能計算において性能上大きなボトルネックになっているメモリアクセスについて，1 ノードの物理メモリサイズを超えるような大容量なデータに対しても高性能なアクセスを可能にするような記憶システムの構築を目的とする．

各ノードにある DRAM メモリ，SSD などを用い，複数ノードに分散した記憶資源を統合的に扱うためのソフトウエアシステムを設計，構築するための実装実験，性能評価を行う．特に記憶階層について着目し，各ノードの計算処理において，メモリアクセス局所性を生かすようなアルゴリズムを導入し，大規模サイズの問題に対して，多数ノードを用いて高速処理できるようにすることを目指す．応用分野は，当面，ステンシル計算，行列計算などの典型的な科学技術計算などとする．

概要

これまでに，主メモリサイズを超える問題サイズであるステンシル計算（Out-of-Core Stencil Computations）を，テンポラルブロッキング最適化

によりデータアクセス局所性を高め，PCIe バス接続型の高速 Flash SSD を主メモリの拡張として用いることにより，３つの手法（swap, mmap, aio）で評価を行ってきた[1]-[6]．この結果，非同期入出力を用いた手法（aio 法）は，他の手法に比べ高性能で，DRAM だけを用いる通常実行の性能（MFlops）の 80%-87%程度の性能が得られることがわかった[6]．さらに，各ノードに SSD を持つクラスタ向けに拡張したアルゴリズムも構築した[7]．現在，改良ステンシルアルゴリズムも構築，評価中で，筆者の所属組織におけるサーバにおいて最新フラッシュデバイスを備えたサーバを用いた性能評価実験を行っている．これらの性能と比較する意味で，低性能・小容量 SSD しか持たない TSUBAME2.5 において，主メモリサイズより大きな容量の SSD を持つ利用可能な 41 ノードのうちの 32 ノードを用い，多ノード実行時の性能評価実験を行った．この結果，限られた性能の SSD であっても，容量が大きければ， out-of-core ステンシル計算処理において，十分，実用可能な性能が得られることを確認した．

本年度の TSUBAME 利用の主目的は，このクラスタ向けアルゴリズムに関し，ノード数に対するスケーラビリティなどを調査することである．

結果および考察

実験には，SSD が搭載されているクラスタシステムの TSUBAME2.5 のキュー S96 のジョブとして実行した．コンパイラは gcc 4.3.4(オプション-O3)でビルドし，MPI は MVAPICH2-2.0.1 を用いた．TSUBAME で利用できる SSD は，ext3fs で各ノードのローカルストレージとしてマウントされており，ブロックデバイスを直接参照することは出来ない．そこで，各ノードの Sub-Buffer 初期化時に SSD のマウントポイント上にファイルを生成し，それを aio で参照するようにした．TSUBAME2.5 のローカル SSD は，SATA 接続 SSD を RAID0 構成にしたもので容量が小さい（実際にユーザが使える容量は約 153GiB）．したがって，SSD/DRAM 比も小さくなり大容量のメモリ拡張実験としてではなく，多数ノードの実験環境として利用した．

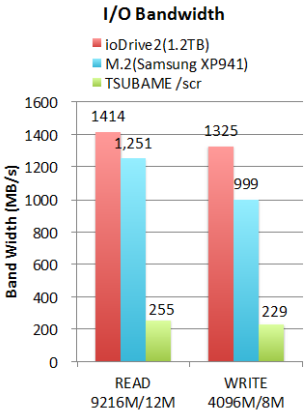

図 1　I/O バンド幅

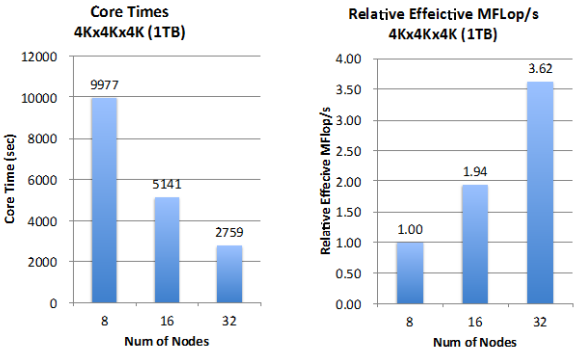

図 2　1TiB 問題コア実行時間　図 3　1TiB 問題相対性能

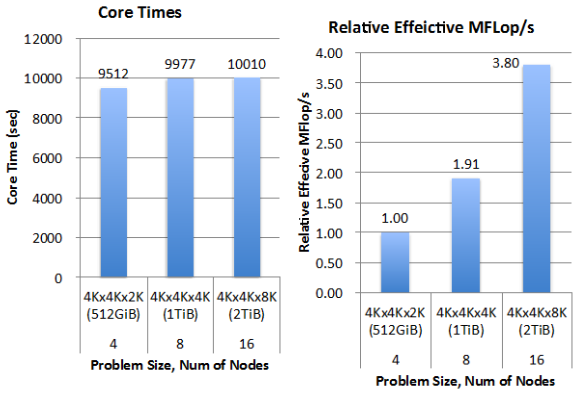

図 4 コア実行時間　　図 5　相対性能

予備実験として，簡単なベンチマークを行い SSD の性能を調査した．前述した筆者所属組織におけるサーバに搭載されているフラッシュ(ioDrive2)と TSUBAME に搭載されている SSD の I/O バンド幅を図 1 に示す．これは，64GiB の問題サイズのステンシル計算において実際に SSD に対してリードライトするデータサイズを用いて，各 SSD の性能を調査したものである．アクセスパターンはシーケンシャル，16 スレッド(TSUBAME は 12 スレッド)でそれぞれ 9216MiB のデータを 12MiB の単位で読み込み，4096MiB のデータを 8MiB 単位で書き込んだ時のバン

ド幅である．4.4〜5.8 倍の性能差がある．

多数ノードクラスタでの性能実験

近傍 7 点ステンシル計算について，格子サイズ 4094x4096x4096（1TiB），256 タイムステップの問題を，空間ブロック 4094x1024x512 格子，時間ブロック 32 ステップ，内部ブロックループ 2048x2x30 のパラメータで，ノード数を 8(2，4 分割)，16(4，4 分割)，32(4，8 分割)と変更して実行した．処理に要した時間を図 2 に，得られた相対性能を図 3 に示す．8 ノードを 1 とした場合の相対並列効率は 16 ノードで 97%，32 ノードで 91%であった．この 1TiB の問題サイズは，SSD の容量の制約(約 153GiB)から 7 ノード未満では実行できない．このため，1 ノード実行時との比較はできない．

1 ノードの担当する問題規模を一定にした場合についても計測した．1 ノードあたりの問題格子を 4Kx2Kx1K（128GiB)とし，ノード数に合わせ問題サイズを大きくした．ノード数 4，8，16 の順に，問題格子は 4Kx4Kx2K（512GiB)，4Kx4Kx4K（1TiB)，4Kx4Kx8K（2TiB)となる．ブロッキングのパラメータは，問題サイズ 1TiB 固定と同じ値を指定した．処理に要した時間を図 4 に，得られた相対性能比を図 5 に示す．ノード数が増すにつれ，実行時間は長くなる傾向が見られるが，8 ノード，16 ノードどちらとも，4 ノードと比較して全体の約 5%程度の増加が見られた．ノード数が増えるほど，アルゴリズム中の全ノードでバリア同期を取る部分の影響が現れていると考えられる．しかし，少数ノード時の実験と同様，ノード内のステンシル計算がコア実行時間の大部分を占めており，ノード間のバッファ交換に要する時間が相対的に短い．各ノードの担当する問題サイズを大きくすることで，実行時間に対するバッファ交換の影響を低減できると考えられる．

これらの結果は，多数のノードを利用することにより，シングルノードでは実行できない規模のステンシル計算を，大きな性能低下を伴わずに実施できること示している．

問題格子の形状と分割方法

前節での実験を実施するにあたり，各ノードが担当する分割された問題のサイズは，クラスタのリソースの範囲内であることを考慮しておかなければならない．すなわち，問題格子のサイズと形状，そしてノードに搭載された SSD のサイズにより決まる問題分割方法について検討した．

8 ノードにおいて，問題の形状が異なるがデータサイズは同じ 1TiB のステンシル計算を 2 パターンで行い比較した．但し，各ノードが担当する問題サイズは，SSD の容量の制約があるので，ブロッキングパラメータ(bt)を調整することで対応した．これら問題格子，ブロッキングパラメータとコア実行時間を表 1 に示す．

表 1　問題規模が同じステンシル計算の各種パラメータと実行時間

shape	cube	cuboid
node	8	8
Problem Size (GiB)	1024	1024
Domain-Buffer	4Kx4Kx4K	2Kx2Kx16K
calc step (nt)	256	256
temporal block (bt)	32	128
time loop iteration	8	2
split	(2, 4)	(1, 8)
Sub-Buffer	4Kx2Kx1K	2Kx2Kx2K
Sub-Buffer Size(GiB)	140	136
Block	4Kx1Kx512	2Kx1Kx1K
Block Size(GiB)	38	50
core time (sec)	9977	5902
Effective-MFlop/s	17624	29779

形状が立方体の問題を 4Kx4Kx4K とし，8 ノードで(2，4)分割すると，1 ノード担当分である Sub-Buffer は，4Kx1Kx2K となる．一方，直方体の問題を 2Kx2Kx16K として，8 ノードで(1，8)分割すると，Sub-Buffer は，2Kx2Kx2K となる．Sub-Buffer には，時間ブロック bt ステップ分のノード間「のりしろ」が分割方向の両端に追加されるが，当然 Sub-Buffer のサイズは，ノードに搭載された SSD のサイズ以内である必要がある．このため，直方体の問題では z 軸方向のみの「のりしろ」だけなので bt を 128 ステップとしたが，立方体の問題では y，z 軸方向の「のりしろ」が追加されるため，bt を 32 ステップへ縮小した．

これら問題のコア実行時間を比較すると，同じ問題サイズを扱っているにもかかわらず，立方体では，直方体の 1.7 倍の時間を要した．この差は，時間ブロックの縮小したことで，時間ループの繰り返しが 2 回から 4 倍の 8 回に増加したことに依るもので，MPI 通信によるバッファ効果に依るものではない．

立方体，直方体の問題実行時の時間内訳をそれぞれ図 6，図 7 に示す．両者を比較すると，立方体では，ステンシル更新での Sub-Buffer，Block 間での I/O の時間（図中ラベル calc_R，calc_W）が約 3 倍増加しており，時間ループの繰り返し回数が増えることで SSD-DRAM 間の I/O 回数も増え，その分，実行時の性能低下を引き起こしていることがわかる．また，各ノードでステンシル更新の演算時間（図中ラベル calc）が約 20%増加している．これもテンポラルブロッキングの冗長計算が，時間ループの繰り返し回数分積算されたものである．これら結果から，問題実行時の時間ブロックパラメータ bt は，慎重に決定する必要があり，如何に bt を大きくし，ステンシル更新演算処理におけるテンポラルブロッキングによるデータ参照局所性を高めるか，が重要である．

立方体問題，直方体問題でのコア実行時間における各処理に要した時間内訳を，それぞれ図 8，図 9 に示す．直方体では，Sub-Buffer，Block 間での I/O（図中ラベル cacl_R，calc_W）がコア実行時間の 20%であるが，立方体では 38%と，コア実行時間に対し大きな割合を占めている．直方体では時間ブロッキングが大きい分，ステンシル更新の演算を高速化できていることがわかる．ノード間の各処理に要する時間のばらつきがノード間の同期時間（図中ラベル barrier）となる．ノード間のバッファ交換は，どちらの形状でもコア実行時間の 4%と，ステンシル更新処理に比べ，非常に小さい割合であり，分割方法によるバッファ交換回数の違いは性能に大きな影響を及ぼさない．

ステンシル更新の処理は独立性が高く，ノードが増加した場合でも，並列効率はあまり低下しないことを示唆している．

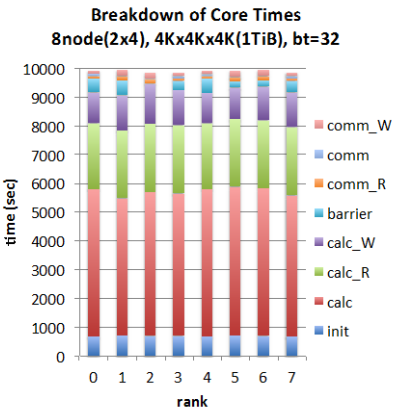

図 6　立方体問題の時間内訳

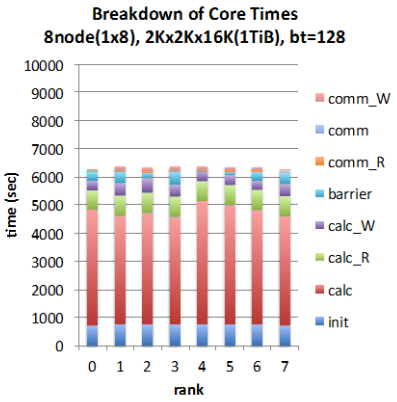

図 7　直方体問題の時間内訳

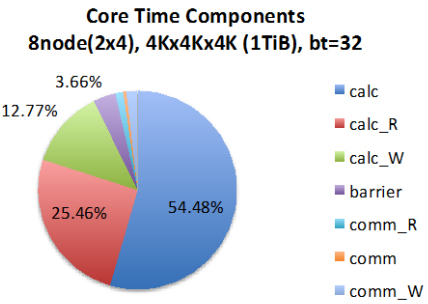

図 8　立方体問題コア実行時間成分

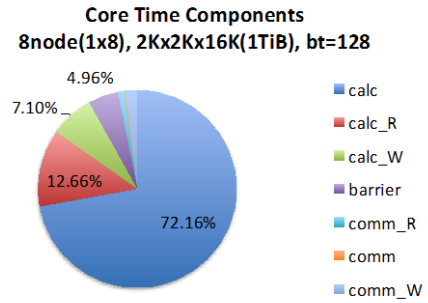

図 9　直方体問題コア実行時間成分

まとめ、今後の課題

メモリ階層構造を考慮したテンポラルブロッキングによるステンシル計算を，マルチノード向けに拡張したアルゴリズムを設計実装し，初期評価を行った．その結果，ノード間同期によるオーバーヘッドによる性能低下があるものの,実験で用いた 32 程度のノード数であれば，提案するアルゴリズムが実用になることがわかった．すなわち，ノードに搭載される SSD 容量を主メモリに対して数倍～十数倍以上にすることで，主メモリ総量を超えた大規模なステンシル計算を，Flash 付クラスタにおいて，実行可能となる．

SSD-DRAM 間のテンポラルブロッキング適用において，時間ブロックサイズ（bt）は， DRAM-Cache 間におけるテンポラルブロッキング適用時と同様に最適値があると考えられる．しかし，今回の問題パラメタの範囲では，bt ステップを比較的大きくすることがより高性能につながっている．さらに大きな時間ステップ数，データサイズの場合には，bt の値に適切な範囲があると考えられる．シングルノードにおける各種ブロッキング自動パラメータ選択アルゴリズムは開発している[6][8]が，マルチノードシステムにおいては，新たに MPI 通信が必要になるため，MPI 通信バッファとの兼ね合いも考慮する必要がある．今後は，与えられた問題と，それを実施するクラスタの環境なども加味した最適なブロッキングパラメータを決めるチューニング手法を確立したい．

コア時間の内訳を見ると，ステンシルの更新演算が最も時間を要していることがわかった．この部分に対しては，評価に用いたテンポラルブロッキングステンシルアルゴリズムが冗長計算を伴うものであるため，これを新たな冗長計算なしステンシルアルゴリズムに変更することや，GPGPU によるアクセラレーションを実施することも効果があると考えている．しかし，一方で，高並列非同期入出力（aio）を用いた提案アルゴリズムにより入出力時間の影響が全体に占める割合が減ったことにより，たとえ，性能が多少低い FlashSSD であっても,容量が主メモリに対し十分大きければ，out-of-core のステンシル計算には利用できることもわかった．

次期 TSUBAME など，今後増えると予想される各ノードに大容量の PCI バス接続型 SSD が搭載したクラスタでは，データ参照局所性を高めたアルゴリズムにより，SSD をメモリ拡張として用いることが実用上，十分効果のあることが示された．

参考文献

[1] Hiroko Midorikawa, Hideyuki Tan and Toshio Endo:"An Evaluation of the Potential of Flash SSD as Large and Slow Memory for Stencil Computations", Proc of the 2014 International Conference on High Performance Computing and Simulation (IEEE HPCS2014), pp.268-277, July 2014

[2] 緑川博子，丹英之：″大規模ステンシル計算のための Flash SSD 向けテンポラルブロキングの性能評価″，情報処理学会，ハイパフォーマンス研究会 Vol.2014-HPC-145, No.22, pp.1-9,（2014, 7/29）

[3] Hiroko Midorikawa, "Using a Flash as Large and Slow Memory for Stencil Computations". Flash Memory Summit 2014, August 2014

[4] 丹英之,緑川博子:″ブロックデバイス非同期 I/O によるフラッシュストレージを用いたステンシル計算の性能評価″,電子情報通信学会，コンピュータシステム研究会 CPSY2014-52, pp.31-36,（2014,10/10）

[5] Hiroko.Midorikawa, "Using a Flash SSDs as Main Memory Extension with a Locality-aware Algorithm". 2015 Non-Volitile Memories Workshop, in UCSD, March, 2015

[6] Hiroko Midorikawa, Hideyuki Tan, “Locality-Aware Stencil Computations using Flash SSDs as Main Memory Extension”, Proc. of CCGric2015, May , 2015

[7] 丹英之,緑川博子："SSD 搭載クラスタを用いた大規模ステンシル計算のための out-of-core アルゴリズム"，ハイパフォーマンス研究会 Vol.2015-HPC-149, No.4, pp.1-7, (2015, 6/25）

[8] Hiroko Midorikawa："Blk-Tune: Blocking Parameter Auto-Tuning to Minimize Input-Output Traffic for Flash-based Out-of-Core Stencil Computations"，will appear in IPDPS2016 Workshop，iWAPT