エクサフロップス級計算機に向けたプログラミングモデルに関する一考察

高木 亮治* 、 堤 誠司†

A Study on Programing Models for ExaFLOPS Scale Computers

by

Ryoji Takaki* and Seiji Tsutsumi†

ABSTRACT

PetaFLOPS scale computers such as the next-generation supercomputer K, are being developed in the world. These supercomputers still don't have enough capability to conduct detailed numerical simulations for actual ows in aerospace elds. At the moment, much faster computer with ExaFLOPS capability has been studied. One of the big challenges to realize ExaFLOPS scal computers is to achieve high level power efficiency, which greatly changes existing hardware architectures. This change may dramatically degrade the performance of existing CFD programs. Therefore, a new programing model for CFD is necessary for such novel architectures. As a rst step of a discussion of the new programing model, a loop structure of CFD program is discussed in this paper, based on the architecture trend of ExaFLOPS scale computers.

1. はじめに

現在開発が進められている次世代スーパーコンピュータ「京」が今秋から本格的な稼動を開始する。「京」は理論ピーク性能で 10 ペタフロップスの演算能力を有し、様々な分野における数値シミュレーションでのブレイクスルーが期待されている。航空宇宙分野における流体解析においても、風洞模型スケールの LES 解析の実用化などが期待されているが、表 1 に示すように実機に対する LES 解析を行うには依然として計算能力が不十分であり、更なる高性能計算機の開発が必要とされている。

スーパーコンピュータのランキングである Top500[1] のデータなどから、2018 年頃には「京」の 100 倍の演算性能を有するエクサ‡フロップス級計算機が出現すると予想されており、日本においてもエクサフロップス級計算機実現に向けた検討 [2] が進められている。エクサフロップス級計算機を実現するためには様々な技術課題が存在するが、計算機システムとして見た場合、最も重要な課題は消費電力の削減と実装密度の向上である。これらの技術課題を踏まえて、京と同程度の制約 (消費電力は $20MW$ から $30MW$、設置面積は $2,000m^2$ から $3,000m^2$) の下で 2018 年頃に実現されるであろうエクサフロップス級計算機として 4 つのシステム案が検討されている。それらのシステムの中には従来のシステムバランス (演算性能、メモリ搭載量、メモリ帯域) とは大きく異なるものも存在する。表 2 に現在想定されている 4 つのシステム案の性能予測を示す。

現在検討されている技術課題や想定されるシステムの特性の中で、流体解析を行う上で最も大きな影響を与えると思われるのは、メモリ帯域と演算性能の比である B/F およびメモリ容量である。表 2 の中では「容量・帯域重視」が流体解析などメモリ帯域が必要となるアプリケーション向けのシステム案であるが、他のシステムと比べて演算性能が非常に低い。一方「メモリ容量削減」も B/F が 0.5 であり現状の「京」と同程度であるが、メモリ搭載量が非常に少なく、メモリを比較的必要としない非定常解析を前提としても 1EFLOPS に対して最低限 0.005[EByte] は必要§な事を考えると流体解析には不適切と思われる。

これまで、流体解析プログラムは高いメモリバンド幅を要求するプログラムであり、高いメモリバンド幅を有するベクトル計算機との相性が良いと言われてきた。しかしながら、科学技術計算の分野においても専用計算機的なベクトル計算機から汎用計算機的なスカラー並列計算機へと計算機アーキテクチャの移行が行われ、しかも急激に増加する演算性能に比して、メモリ性能の伸びが追いつかず、B/F は確実に減少する傾向にある。前述したエクサフロップス級計算機の想定される 4 つのシステムのうち、「汎用 (従来型)」は様々な計算に適用可能な汎用性を指向したもので、次世代スーパーコンピュータ「京」の延長線上の計算機システム (スカラー並列計算機) であり、一般的なスカラー計算機の将来像を示しているが、B/F は 0.1 へと減少し次世代スーパーコンピュータ「京」の 1/5 となっている。ちなみに B/F の長期的な減少傾向はスカラー計算機だけではなく、ベクトル計算機も例外ではなく、従来 B/F が 4(NEC の SX-8 以前) であったものが最近では 2.5(NEC の SX-9) となっている [3]。

筆者らが開発を行ってきた圧縮性流体解析プログラム UPACS[4] はベクトル計算機の時代から開発されており、ベクトル計算機のアーキテクチャを指向した高いメモリバンド幅に依存したプログラム構造を暗黙のうちに踏襲している。現在ではベクトル計算機からス

*宇宙航空研究開発機構 宇宙科学研究所/情報・計算工学センター
†宇宙航空研究開発機構 情報・計算工学センター
‡エクサはペタの 1,000 倍

§現在筆者らが実施している JAXA 統合スーパーコンピュータシステム (JSS) を用いた非定常解析では 1 プロセス (40GFLOPS) あたり 20 万点の格子を用いており、その際にメモリ使用量は 200MByte となるのでそこから外挿して予測した値

表 1: LES 解析に必要な計算規模の予測

スケール	Re 数	格子点数	時間刻み幅 [μ 秒]	ステップ数	計算時間 [時間]	計算能力 [FLOPS]
研究	10^5	500 万点	2	20 万	5	8 Tera
風試	10^6	10 億点	0.4	100 万	5	10 Peta
実機	10^7	2,000 億点	0.08	500 万	5	10 Exa

表 2: エクサフロップス級計算機のシステム性能予測 [2)]

	総演算性能 [EFLOPS]	総メモリ帯域 [EB/s]	B/F	総メモリ量 [EB]
汎用 (従来型)	0.2〜0.4	0.02〜0.04	0.1	0.02〜0.04
容量・帯域重視	0.05〜0.1	0.05〜0.1	1	0.05〜0.1
演算重視	1〜2	0.005〜0.01	0.005	0.005〜0.01
メモリ容量削減	0.5〜1	0.25〜0.5	0.5	0.0001〜0.0002

カラー計算機への移行が進み、そのためスカラー計算機向けのチューニングを実施することで、実行性能の向上を図っているが、今後は更なる B/F の減少が予想されるため、小手先のチューニングでは限界が見え始めている。エクサフロップス級計算機においてはさらなる B/F の低下が示唆されており、低い B/F においてもそれなりの実行性能を発揮する流体解析プログラムを開発する必要がある。

ここでは、これまでのプログラミングモデルを一旦リセットし、低 B/F を前提とした圧縮性流体解析プログラムの実現を目指して、どの様なプログラム構造が適切かについての検討を試みる。まず、手始めにプログラムのループ構造についての検討を行ったので、その結果について報告する。

2. ループ構造の検討

JAXA が開発している UPACS の主要部分を抜き出してカーネルプログラムを作成し、これを用いてループ構造の検討を行った。カーネルプログラムでは、一般曲線座標系で記述された支配方程式を対象として、右辺対流項の計算部分 (基本変数を用いた 2 次精度 MUSCL、van Albada のリミター、数値流速は SHUS) と左辺時間積分 (1 次精度 Euler 陽解法) を実装している。現時点では粘性項、陰解法による時間積分、境界条件は考慮していない。対象となるこれらの計算を行う際のループ構造として

- ループ A : 従来のループ
- ループ B : 局所性を意識したループ (空間スイープの 3 重ループの数をできるだけ減らした)

を実装し、計算速度の違いを調べた。

ループ A は従来のプログラムに良く見られる構造で、圧縮性流体の離散方程式をプログラムとして実装する際に、対流項の計算に用いるセル面での物理量の外挿 (MUSCL+リミター)、セル面における対流項の数値流束の計算、(セル面における粘性流束の計算)、更新ベクトル ΔQ の計算、時間積分 (左辺の計算) のように、それぞれの計算を分割して、それぞれに対して空間の多重ループで計算を実行する。そのために 1 ステップの計算を実行するのに、何度も空間スイープを行うことになる。ループ A を模式的に書くと以下のようになる。ここで、dir=1,3 のループは 3 次元のインデックス方向 (i、j、k 方向) のループである。

```
do dir=1,3

  do k=1,kmax
  do j=1,jmax
  do i=1,imax
    MUSCL による外挿
  enddo
  enddo
  enddo

  do k=1,kmax
  do j=1,jmax
  do i=1,imax
    数値流束の計算
  enddo
  enddo
  enddo

  do k=1,kmax
  do j=1,jmax
  do i=1,imax
    更新ベクトルの計算
  enddo
  enddo
  enddo

enddo

do k=1,kmax
do j=1,jmax
do i=1,imax
  時間積分
enddo
enddo
enddo
```

この様にループ A では 3(MUSCL、数値流束、ΔQ) × 3 方向 (i,j,k 方向)+1(時間積分) の計 10 回の空間スイープを実行することになる。空間を何度もスイープすることはそれだけメモリアクセスが増加し、キャッシュを有効に活用することができなくなる。そのため、アルゴリズムの観点からメモリアクセスを減らすこと

を意図して、空間のスイープを極力減らし、データの再利用性を心掛けるループ B を考える。ループ B では空間のスイープは必要最低限な 2 回 (右辺の計算と左辺の計算) とした。つまり、あるセル (i,j,k) に着目し、そのセルで必要な右辺の計算を全て行い、それが終わると次のセルに移動する。この様にして 1 つの空間スイープで右辺を全て計算する。次に左辺に関して、2 つ目の空間スイープで計算を済ませることとする。ここで注意すべき点として、一般には数値流束の計算はセル面のループで回すが、これをセルのループで回すと何も考慮しないと 1 つの面での数値流束を 2 回計算することになる。これを避けるためにはフラグを設定して計算したかどうかを判別する必要があるが、ここでは構造格子である利点を活かして、個々のセルでは各方向でインデックスが増える方向のセル面での数値流束の計算を実施することとする。そのためインデックスの始点側境界 ($i = 1$ or $j = 1$ or $k = 1$) での処理が必要となる。

```
do k=1,kmax
do j=1,jmax
do i=1,imax
  do dir=1,3
    MUSCL による外挿
    数値流束の計算
    更新ベクトルの計算
    境界での処理 (MUSCL、数値流束、更新ベクトルの
計算)
  enddo
enddo
enddo
enddo

do k=1,kmax
do j=1,jmax
do i=1,imax
  時間積分
enddo
enddo
enddo
```

ループ A、B ともに最外ループである k のループを対象に OpenMP でスレッド並列化を行った。

3. 数値実験

ループ A および B を実装したカーネルプログラムを幾つかの計算機上で実行し、それぞれの性能測定を実施した。計算対象は単純な立方体格子であり、格子ブロックサイズ (=ループ長) を変えて測定を行った。性能測定に用いた計算機環境を表 3 に示す。コンパイル時の最適化オプションは富士通コンパイラー (JSS)、インテルコンパイラー (PC-S)、GNU コンパイラー (PC-S, PC-N) でそれぞれ、-O5、-fast、-O3 を用いた。

3.1 JSS での測定

JSS(富士通 FX1、CPU は SPARC64VII) 上でループ A とループ B の比較を行った。計算は 1 プロセス、4 スレッドである。図 1 にループ A、B それぞれの計算時間と L2 キャッシュのミス率の傾向を示す。図より、ループ B は狙い通り L2 キャッシュのミス率が半減していることがわかる。しかしながら計算速度としてはループ A の方がループ B よりも速い結果となった。この原因であるが、ループ B はキャッシュのミス率は改善されたが、ループの中身が大きくなった分、レジスター溢れや、パイプライン処理の最適化など、他の性能要因の影響によって性能が悪化した可能性がある。コンパイラーの最適化能力と関連するので、引き続き詳細な検討が必要である。

両方に共通する傾向として、経過時間に細かな振動が見られるが、これはスレッド数 4 の周期となっており、格子ブロックサイズがスレッド数で割りきれる場合が局所的に経過時間が短く、余りが 3 の場合に局所的に経過時間が長くなるためである。また、格子ブロックサイズが小さい場合はデータがキャッシュに収まるため、演算ネックとなり格子ブロックサイズの増加とともに経過時間が増加している。一方、格子ブロックサイズが 60 を越える辺りからデータがキャッシュから溢れるため、メモリバンド幅ネックとなり、格子ブロックサイズが増加しても経過時間は殆ど変化しなくなると考えられる。

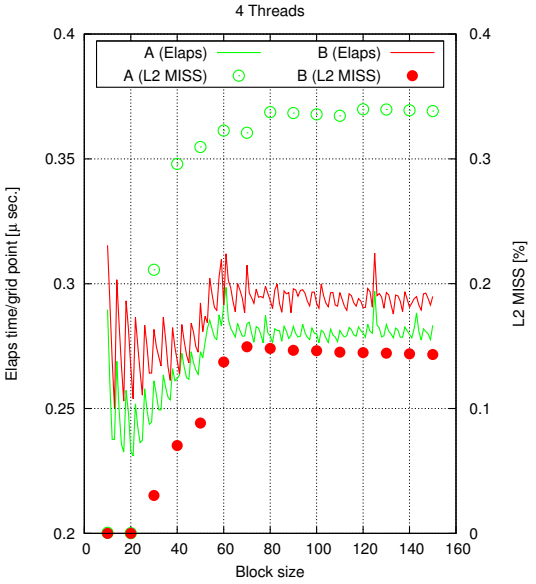


図 1: JSS におけるループ性能 (経過時間と L2 ミス率)

図 2 に仮想的にメモリバンド幅を変化させた時の影響を示す。ここでは 1 つの CPU 内に 1 プロセス × 1 スレッド (1P/CPU) で計算を行った場合と、4 プロセス × 1 スレッド (4P/CPU) で計算を行った場合を比較することで仮想的にメモリバンド幅が変化した場合の計算性能の変化を調べた。詳細に関しては A を参照のこと。1P/CPU は CPU のメモリバンド幅をほぼ占有できるが、4P/CPU は 4 プロセスでメモリバンド幅を共有するため、1P/CPU のケースに比べて 1/4 のメモリバンド幅とみなせる。図より、ループ A はメモリバンド幅が減少すると 10%程度性能が下がるが、ループ B は 2%程度しか下がらず、この範囲ではメモリバンド幅にあまり影響を受けないことがわかる。

3.2 インテル系 CPU での測定

インテル系 CPU でも PC-S(Core i7-3960X) を中心にループ A および B の比較を行った。図 3 に 1 スレッド

表 3: 計測環境

Name	CPU	# of Cores	CPU Clock [GHz]	GFLOPS	Memory bandwidth [GB/s]	Compiler
JSS	SPARC64 VII	4	2.5	40	40	Fujitsu
PC-S	Core i7-3960X Sandy Bridge	6	3.3	158.4	51.2 (DDR3-1600) or 42.7 (DDR3-1333)	Intel or GNU
PC-N	Core i7-965 Nehalem	4	3.2	51.2	31.2 (DDR3-1333)	GNU

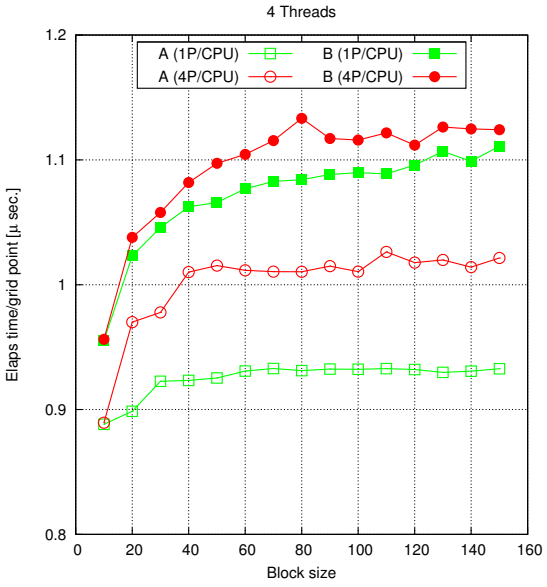
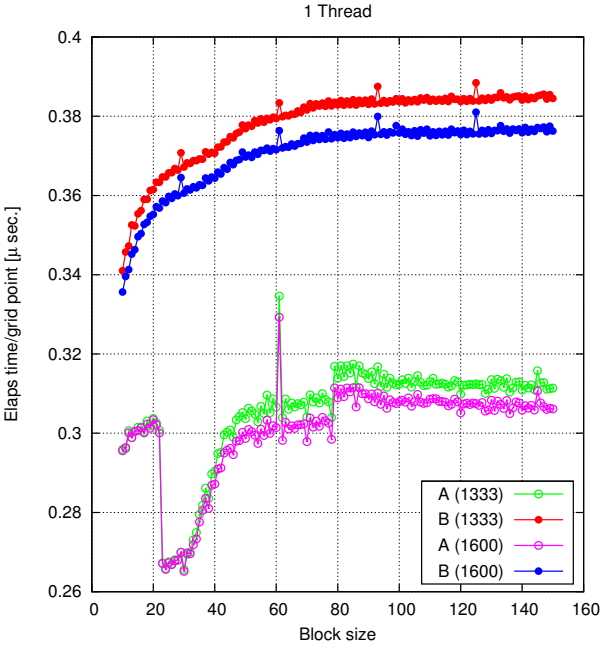


図 2: JSS におけるループ性能 (メモリバンド幅の影響)

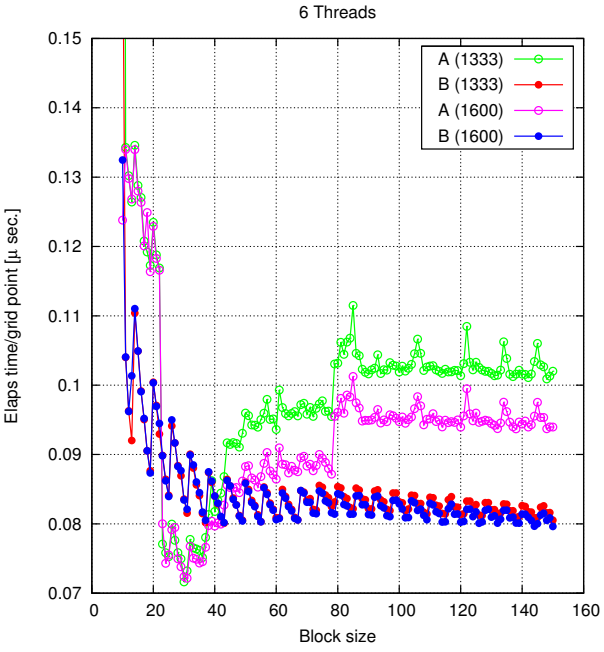
と 6 スレッドの場合 (どちらも 1 プロセス実行) の測定結果を示す。横軸は格子ブロックサイズ (=ループ長)、縦軸は 1 格子点あたりの計算時間 (経過時間) である。ここで、図中の 1333 および 1600 はメモリクロックの値を示し、大きい方 (1600) がデータ転送能力が高いため、ループ A、B ともメモリクロックが高い方が性能が高いことがわかる。

ループ A は格子ブロックサイズが 23 および 78 の前後で不連続な特性、また 1 スレッドのケースで格子ブロックサイズが 63 の時に局所的な性能悪化が見られる。キャッシュやメモリバンク競合などの原因が考えられるが詳細は不明である。一方ループ B は比較的素直な特性を示しており、一般的な傾向 (格子ブロックサイズが小さい範囲では演算器ネックのため演算量の増加に伴って計算時間が増加し、格子ブロックサイズが大きくなりキャッシュが溢れる様になるとメモリネックになり計算時間がほぼ一定となる) が見られる。スレッド並列の場合は格子ブロックサイズが小さい範囲ではスレッド並列のオーバーヘッドが顕著になり、結果的に計算時間が増加していると考えられる。スレッド並列の場合、プログラム中の各ループはスレッド数で分割されるため、ループ長がスレッド数で割った余りに応じて計算時間が変動している様子が観察できる。

ループ A とループ B との比較では、スレッド数の増加および格子ブロックサイズの増加など、メモリアクセスの負荷が大きくなると、ループ A とループ B で計



(a) 1 スレッド



(b) 6 スレッド

図 3: PC-S(Core i7-3960X) におけるループ性能

算性能の逆転現象が見られ、ループ B がループ A に比べて 18%程度良い性能を示している。この事は、それぞれのループでメモリ周波数の違いによる計算時間の差を見ても同様のことが言える。つまり、それぞれのループで 1333 と 1600 の違いを見ると、メモリ性能が低くなった場合に計算速度がどの程度悪化するかがわかる。ループ B はメモリ性能が悪化してもほとんど計算時間が悪化していないが、ループ A はメモリ性能の悪化に対して計算時間が 7%程度 (6 スレッドの場合) 増加している。この結果からもループ B はループ A に比べてメモリ性能にあまり依存しないと考えられる。

3.3 B/F による整理

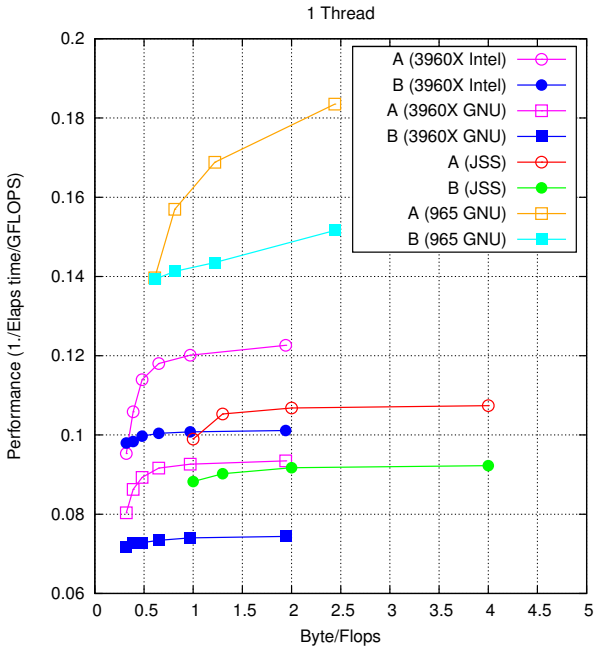
理論メモリ性能 (Byte/s) と理論演算性能 (FLOPS) の比を B/F と呼ぶが、アプリケーションの特性を議論する際に重要な指標となる。一般に圧縮性流体解析プログラムは高い B/F が必要と言われている。1CPU で実行するプロセス数を変化させることで、1 プロセス当たりのメモリバンド幅を仮想的に変化させることを考えた。この手法を用いてそれぞれのループのメモリバンド幅が減少した際の性能特性を調査した。メモリバンド幅を仮想的に変化させる手法の詳細については A を参照の事。なお、A の結果より、JSS および Core i7-965 は比較的この手法の精度が良いと考えられる。

測定結果を図 4 に示す。この図で「3960X」、「965」は表 3 の PC-S (Core i7-3960X)、PC-N (Core i7-965) を示す。また、「Intel」、「GNU」はそれぞれインテルコンパイラー、GNU コンパイラーを示す。

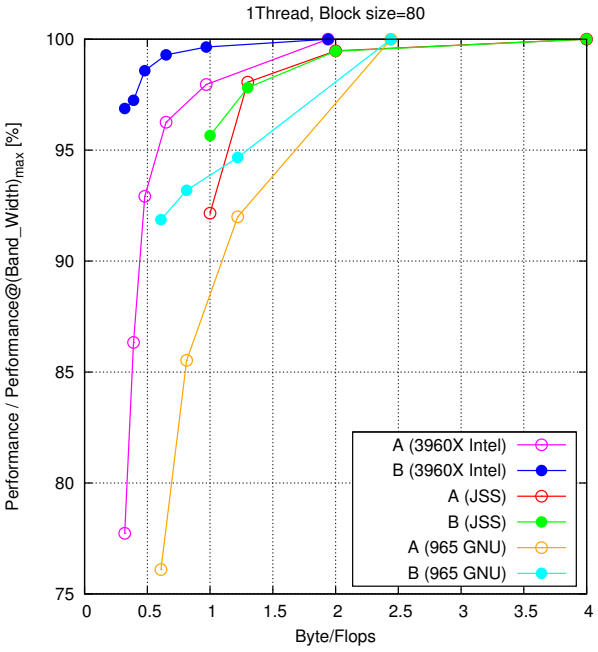
格子ブロックサイズは 80 である。横軸は B/F であるが、ここでは理論性能の値を用いた。理論性能に対して実際に出る性能 (実行性能) は計算機システムによって異なるため、異る CPU の結果を比較する際は注意が必要である。

図 4(a) の縦軸は「性能」を示す。ここで「性能」は単位理論性能あたりの計算速度 (経過時間の逆数) とした。JSS を始めとして全ての CPU でループ A の方が良い性能を示している。しかしながら、B/F が減少するとループ A は急速に性能が悪化している。一方、ループ B は B/F の減少にともなう性能悪化はそれほど酷くないことがわかる。1 ケースだけではあるが、B/F が最低の時にループ B の方が速い結果が得られている。更に B/F が減少した場合に 2 つのループの特性がどうなるかは現時点では不明ではあるが、低 B/F の領域ではループ B の方が良い性能を示す可能性がある。

図 4(b) の縦軸は「計算効率」を示す。ここで、「計算効率」は前述の「性能」を最大 B/F の時の「性能」で正規化した値である。つまりある CPU に対して B/F が最大の場合 (1CPU に対して 1 プロセスを実行し、メモリバンド幅を占有した場合) の「性能」を 100 とした時の「性能」比である。この図からも、B/F が減少した時にループ A はループ B に比べて急激に性能が悪化していることがわかる。



(a) 計算性能



(b) 計算効率

図 4: B/F の変化による計算性能の変化

4. まとめ

UPACS の主要部を切り出したカーネルプログラムを作成し、2 種類のループ構造に対して計算性能の比較を行った。従来のループ構造よりも局所性を意識し空間スイープを極力減らしたループは、狙い通りキャッシュミス率を従来のループ構造に比べてほぼ半減させることができた。計算性能を支配する他の最適化要因が複雑に関係するため、新しいループ構造が常に良いという結果は得られなかったが、低メモリバンド幅のシステムでは有利になる可能性が示された。今回の測定では低 B/F 領域を十分に設定することができなかったため、更なる低 B/F 領域での測定を実施する予定である。更には、粘性項、陰解法を含めた評価や、ループ構造だけではなく、キャッシュの有効利用など低 B/F を前提としたプログラミングモデルの検討を今後進めていく予定である。

A STREAM と B/F の制御

メモリ性能を測定するベンチマークテストである STREAM[5] を用いて、今回性能測定を行った CPU のメモリ性能を測定した。STRAM では COPY、SCALE、ADD、TRIAD の性能を測定できるが、ここでは COPY と TRIAD の性能をブロックサイズ N を変化させて測定した。ここで COPY は

```
do i=1,N
  c(i) = a(i)
enddo
```

となる。図 5 に測定結果を示す。

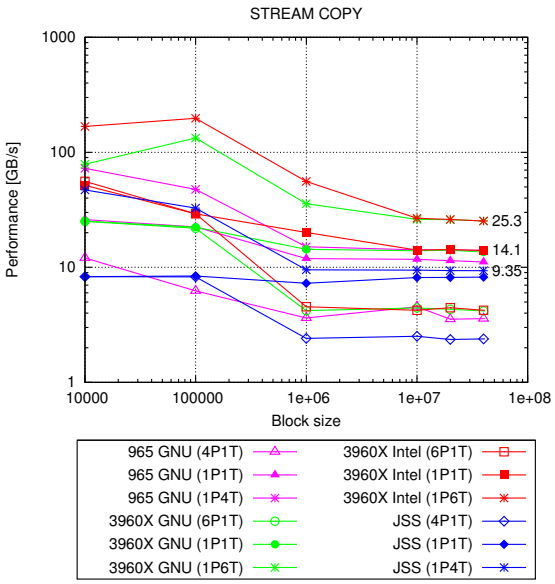


図 5: STRAM を用いたメモリ性能の測定結果

ブロックサイズが小さい場合はキャッシュを有効に活用することができ、一般的にメモリ性能が良いことがわかる。最大のブロックサイズ（4×10^7）での COPY 性能は Core i7-3960X が 25.3GB/s、Core i7-965 が 14.1GB/s、JSS が 9.35GB/s となった。ちなみに、この性能は OpenMP を用いて CPU 内の全コアを使ったスレッド並列での性能である。それぞれの理論メモリバンド幅は 51.2GB/s、31.2GB/s、40GB/s であるため、実行効率は 49.4%、45.2%、23.4%となる。

それぞれの CPU でコア数分プロセス×1 スレッド、1 プロセス×1 スレッド、1 プロセス×コア数分スレッドの比較を行った。ちなみに 1 プロセス×コア数分スレッドはマルチコア CPU を利用する際の標準的な手法であり、ハイブリッド並列の基本となる。ここで注目したいのは 1 プロセス×1 スレッドと 1 プロセス×コア数分スレッドとの比較である。特に JSS においては、1 プロセス×1 スレッド (8.22GB/s) と 1 プロセス×4 スレッド (9.35GB/s) の COPY 性能の差が小さい。これは 1 プロセス×1 スレッドの場合は 1 スレッドが CPU のメモリバンド幅をほぼ使い切ることができることを意味する。一方、4 プロセス×1 スレッドはメモリバンド幅の観点からは 1 プロセス×1 スレッドの場合と比べて 4 倍のメモリアクセスが発生するため、相対的に 4 分の 1 のメモリバンド幅と考えることができる。実際、4 プロセス×1 スレッドでは 2.39GB/s となり、1 プロセス×1 スレッドに比べて 1/3.44 倍である。この結果を踏まえて以下の様に考える。JSS の場合は、CPU あたり、4 コアを有し、1 コアの演算性能は 10[GFLOPS]、CPU 全体でのメモリバンド幅は 40[GB/s] である。ここで 1CPU に 1 プロセス×1 スレッドを実行すると、1 コアを使った演算であるため演算性能は 10[GFLOPS]、また CPU のメモリバンド幅をほぼ占有できると考えると、メモリバンド幅は 40[GB/s] となるため、B/F は $40/10 = 4$ と考えられる。次に 1CPU に対して複数のプロセス (各プロセスは 1 スレッド) を実行する。各プロセスは 1 コアで実行されるので演算性能はプロセス数がコア数を越えない範囲では常に一定で、プロセス当たりは 10[GFLOPS] となる。一方でプロセス間でメモリバンド幅を共有することになるため、プロセス当たりのメモリバンド幅は大体プロセス数分の 1 と考えられる。つまり、2 プロセスであれば 1/2、4 プロセスであれば 1/4 と考えられるので、B/F はそれぞれ $40/2/10 = 2$、$40/4/10 = 1$ と考えられる。複数プロセスの実行によるオーバーヘッドの影響など、厳密にはこの通りにはならないが、この方法で B/F が変化した時の性能特性の傾向を見ることは可能と考える。

インテル系の CPU では、1 プロセス×1 スレッドと 1 プロセス×コア数分スレッドとの性能差は、JSS に比べると大きく Core i7-3960X は 14.0GB/s 対 25.3GB/s、Core i7-965 は 11.1GB/s 対 14.1GB/s となり、特に Core i7-3960X は 1CPU に 1 プロセス×1 スレッドの場合に CPU のメモリバンド幅を占有できるとは言えない。また、コア数分プロセス×1 スレッドと、1 プロセス×1 スレッドの比は Core i7-3960X が 14.0GB/s 対 4.23GB/s で 1/3.31(6 コア)、Core i7-965 が 11.1GB/s 対 3.58GB/s で 1/3.10(4 コア) となり、Core i7-3960X は想定からの乖離が大きい。

以上の結果より、CPU 内のプロセス数を変化させて仮想的に B/F を変化させるという方法は、JSS および Core i7-965 に対しては定量的にもほぼ適用できると思われる。しかしながら、Core i7-3960X ではこの考え方は定量的には問題があるが、定性的な議論には利用できると思われる。

参考文献

1) TOP500 Supercomputing Sites, http://www.top500.org.

2) HPCI 技術のロードマップ白書, http://www.open-supercomputer.org/workshop/report/hpci-roadmap.pdf.

3) 長嶺七海、百瀬真太郎. JSS V システムの効率的利用について. 第 41 回流体力学講演会/航空宇宙数値シミュレーション技術シンポジウム 2009 論文集, pp. 153–158. JAXA-SP-09-011, 2010.

4) R. Takaki, K. Yamamoto, T. Yamane, S. Enomoto, and J. Mukai. The Development of the UPACS CFD Environment. In A. Veidenbaum, K. Joe, H. Amano, and H. Aiso, editors, *High Performance Computing, 5th International Symposium, ISHPC 2003, Tokyo-Odaiba, Japan, October 2003. Proceedings*, Vol. 2858 of *Lecture Notes in Computer Science*, pp. 307–319. Springer, 2003.

5) STREAM: Sustainable Memory Bandwidth in High Performance Computers, http://www.cs.virginia.edu/stream/.