Vol.2019-HPC-170 No.17
2019/7/25

NEC SX-Aurora TSUBASAにおける バンク競合の回避に関する一検討

江端 直樹[1,a)]　江川 隆輔[2,b)]　磯部 洋子[2,c)]　高木 亮治[3,d)]　滝沢 寛之[2,e)]

概要：近年，多くの科学技術計算分野のアプリケーションがメモリ性能によって律速されており，高い実効メモリ性能への需要はますます高まっている．しかし，様々な原因から実効メモリ性能は理論的なメモリ性能に大きく劣る場合がある．バンク競合はその一例で，特定のメモリバンクに複数のアクセスが競合することで長い遅延が生じ，著しい実効性能の低下を招く．そのため，高い実効性能を実現するためには，バンク競合による性能低下を回避することが必須である．本研究では，規則的なアクセスパターンを持つプログラムにおいて，バンク競合による性能低下が発生する条件を明らかにする．発生条件を知ることによって，バンク競合の影響が小さくなるようにデータ配置を調整し，性能低下を回避することが可能になる．具体的には，メモリ構成情報とカーネルループで扱う配列の先頭アドレスをパラメータとして，バンク競合による性能低下が発生する条件を定式化する．また，高い理論メモリ性能を持つ計算システムである NEC SX-Aurora TSUBASA において，この条件を検証し，高い精度でバンク競合を予測できることを確認した．

A study for avoiding bank conflicts on NEC SX-Aurora TSUBASA

1. はじめに

近年の高性能計算システムでは，演算性能が急速に向上しているのに対し，メモリバンド幅の向上は比較的遅い．その結果，今日の多くの実アプリケーションがメモリ性能によって律速される傾向にある [1]．したがって，高い実効性能を得るためには，高い実効メモリバンド幅を達成することが必要になってきている．

近年 NEC が発表した SX-Aurora TSUBASA(SX-Aurora) [2] は最新のベクトル型コンピュータで，世界トップクラスの理論メモリバンド幅を持つ．SX-Aurora は HBM(High Memory Bandwidth)2 規格のメモリモジュールを 6 個搭載しており，それぞれの HBM2 モジュールは 8 本のチャネルとチャネルあたり 32 個のバンクを備えている．このメモリ構成によって，SX-Aurora は 1.22 TB/s の理論メモリバンド幅を達成しており，メモリ律速のアプリケーションを高速に実行できると期待されている [3]．

しかし，SX-Aurora の実効メモリバンド幅は，特定のメモリアクセスパターンにおいて著しく低下する場合がある [4]．その要因の 1 つにバンク競合の影響がある．特定のチャネルやバンクに対して同時に複数のアクセスがあった場合，メモリアクセス遅延が大幅に長くなり，実効メモリバンド幅が著しく低下する．バンク競合の有無はメモリ上のデータレイアウトに依存しており，データレイアウトは問題サイズやランタイムなどの様々な動的要因によっても変化する．そのため，複雑化した現代のプロセッサでは，バンク競合による実効メモリバンド幅低下の有無を事前に予測することは困難である．

本研究では，規則的なアクセスパターンを持つプログラムにおいて，バンク競合が発生する条件を調査し，バンク競合を回避する方法を提案する．バンク競合が発生する条件を特定するために，本研究ではカーネルループでロードされる配列の（カーネルによっては，ストアされる配列の）先頭アドレスに着目し，各配列の先頭アドレスの差分をパ

1 東北大学情報科学研究科
Graduate School of Information Sciences, Tohoku University
2 東北大学
Tohoku University
3 JAXA(宇宙航空研究開発機構)
Japan Aerospace Exploration Agency
a) naoki.ebata.r7@dc.tohoku.ac.jp
b) egawa@tohoku.ac.jp
c) isobe@tohoku.ac.jp
d) ryo@isas.jaxa.jp
e) takizawa@tohoku.ac.jp

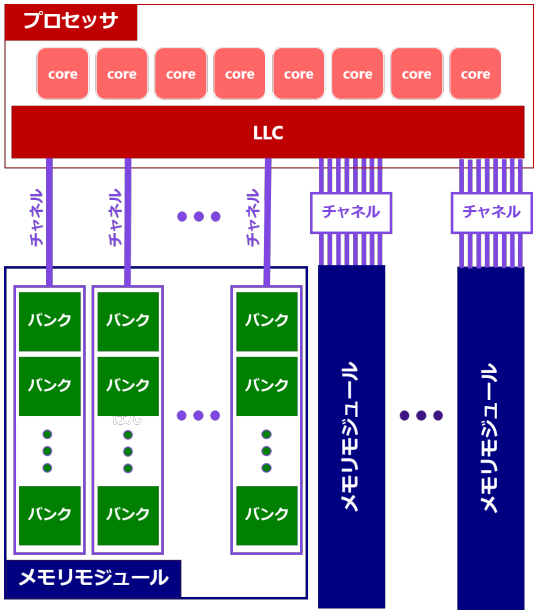

図 1　SX-Aurora のメモリ構成

ラメータとして，バンク競合が発生する条件を定式化する．さらに，配列にメモリを割り当てる段階で，バンク競合発生条件を満たさないようにデータレイアウトを調整し，バンク競合を回避する単純な方法を提案する．

2. SX-Aurora のメモリシステム

2.1 メモリ構成

SX-Aurora は世界トップクラスの 1.22 TB/s の理論メモリバンド幅を実現するために，多数のチャネルとバンクを用意している．SX-Aurora のメモリ構成を図 1 に示す．まず，SX-Aurora は HBM2 規格のメモリモジュールを 6 個持ち，それらはそれぞれ 8 本のチャネルを持つ．このように，6×8 本のチャネルを用意することで高い理論メモリバンド幅を実現している．さらに，チャネルからスムーズにデータを送出できるように，各チャネルは 32 個のメモリバンクを備えている．

また，SX-Aurora ではメモリセルと呼ばれる単位でデータを管理しており，メモリセルのサイズ 128byte の連続したデータは同一のメモリセルに対応付けられてる．

2.2 データ配置規則

SX-Aurora では，アドレスが連続したデータはなるべく別のメモリモジュール，チャネル，バンクに配置するデータ配置規則がとられている．これは，プログラムが空間的局所性を持つ場合，連続したデータは同時にアクセスされるため，なるべく別のモジュールやチャネル，バンクに配置することで，アクセスが競合せず，低遅延で読み出しや書き込みができるためである．

例えば，m 番目のメモリモジュールの c 番目のチャネルの b 番目のバンクを $B(m, c, b)$ と表すとき，メモリアドレス上で連続したデータは $B(0,0,0) \rightarrow B(1,0,0) \rightarrow B(2,0,0) \rightarrow ... \rightarrow B(5,0,0) \rightarrow$ というように，まず異なるメモリモジュールのバンクに対応づけられる．次に，SX-Aurora では $0 \leq m \leq 5$ であるから，$B(5,0,0) \rightarrow B(0,1,0) \rightarrow B(1,1,0) \rightarrow ... \rightarrow B(5,1,0) \rightarrow B(0,2,0) \rightarrow .. \rightarrow B(5,7,0) \rightarrow$ という順に，チャネル優先でメモリセルとバンクが対応する．さらに，$0 \leq c \leq 7$ であるから，$B(5,7,0) \rightarrow B(0,0,1) \rightarrow B(0,0,1) \rightarrow ... \rightarrow B(5,0,1) \rightarrow B(0,1,2) \rightarrow .. \rightarrow B(5,7,31) \rightarrow B(0,0,0) \rightarrow ...$ とバンク順に対応づける．SX-Aurora では，$0 \leq c \leq 31$(or 16) であるから，$B(5,7,31)$ の次は $B(0,0,0)$ に戻る．

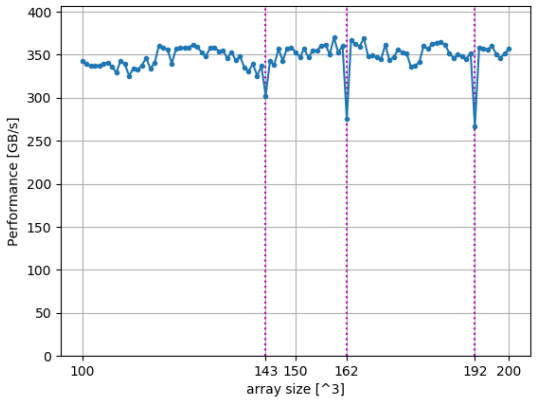

図 2　SX-Aurora における STREAM Triad の性能挙動 (シングルスレッド)

2.3 バンク競合

バンク競合は，同一のバンクに配置されている複数のデータに同時にアクセスする際に生じる．同一バンクに対する複数のアクセスは逐次的に処理されるため，一部のアクセスには待ち時間が生じる．その結果，大幅なメモリアクセス遅延となり，実効メモリバンド幅の低下が起きる．バンク競合の一例として，図 2 に，STREAM Triad [5] を SX-Aurora において配列サイズ 100^3〜180^3 でシングルスレッド実行した時の性能挙動を示す．配列サイズを 3 乗にしているのは，科学技術計算で 3 次元配列がよく利用されるためである．また，実行条件などは 4 節で述べるものと同じである．配列サイズが 143^3 と 162^3 と 192^3 で性能が鋭く低下しており，これはバンク競合の影響によるものである．このように，バンク競合によって，SX-Aurora 本来のメモリ性能は大きく損なわれる．さらに，近年のメモリ構成は複雑化しており，これらの配列サイズでバンク競合が起きることを予測するのは困難になっている．

3. バンク競合の発生条件と回避策

3.1 競合条件

本研究では，カーネルループでロードもしくはストアさ

表 1　変数の説明

N_{array}	ループ内でロード (or ストア) される配列の数
p_i	各配列の先頭アドレス ($1 \leq i \leq N_{array}$)
m_i	先頭アドレスをメモリセルに対応づけたもの
d_{ij}	先頭アドレスが割り当てられている メモリセル間のアドレス空間上の距離
X	基準値
S_{cell}	メモリセルのサイズ (Byte)
N_{module}	モジュールの数
$N_{channel}$	モジュール当たりのチャネル数
N_{bank}	チャネル当たりのバンク数

れる配列の先頭アドレスを用いて，バンク競合の発生を予測する．カーネル内部でロードされる配列の方が多い場合はロードされる配列に着目し，ストアされる配列が多い場合はストアされる配列に着目する．バンク競合の有無は，カーネル実行時にロードもしくはストアされる各配列要素のメモリ上の相対的な位置関係に依存している．本研究では，各配列要素の相対的な位置関係と各配列の先頭アドレスの相対的な位置関係が等しいものと仮定して，各配列の先頭アドレスに着目する．提案する発生条件では，各配列の先頭アドレスと対応しているバンクが近い位置にある場合に，バンク競合が発生すると判定する．

以上の仮定より，本研究では次のようにバンク競合を予測する．提案予測モデルで用いられる変数を表 1 に示す．

(1) まず，カーネルループ内でロードされる配列の数とストアされる配列の数を比較して，ロードとストアどちらを対象にするか決定する (多い方を対象にする)．

(2) 次に，(1) で定めた通りに，ロード (もしくはストア) される各配列の先頭アドレスを p_i とする．このとき，$1 \leq i \leq N_{array}$ が成立し，N_{array} はループ内でロードされる (もしくは，ストアされる) 配列の数を示している．

(3) また，それらのアドレスをメモリセルに対応付ける．この変換は単純で，

$$m_i = [p_i / S_{cell}] \tag{1}$$

で導出できる ([] はガウス記号)．

(4) 変換後，すべての (i, j) の組み合わせについて，

$$d_{ij} = |m_i - m_j| \bmod (N_{module} \times N_{channel} \times N_{bank}) \tag{2}$$

を計算する．d_{ij} はアドレス空間における m_i と m_j が格納されているバンクの相対距離に該当する．

(5) 最終的に，d_{ij} と基準値 X を比較して，

$$d_{ij} \leq X \text{ or } (N_{module} \times N_{channel} \times N_{bank} - X) \leq d_{ij} \tag{3}$$

を満たすとき，バンク競合による実効メモリバンド幅低下の危険性があると判定する．この基準値 X は，現段階では経験的に $X = 32$ としている．

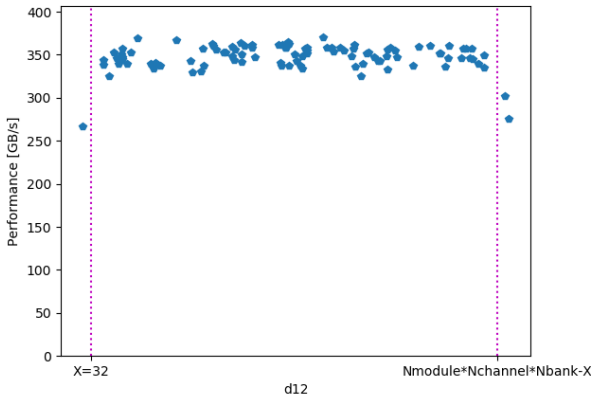

図 3　d_{12} と性能の関連性 (STREAM Triad)

配列数が少ないプログラムであれば，1 つでも式 (3) を満たす (i, j) の組み合わせがあれば，実効メモリバンド幅低下が生じる可能性が高い．一方，配列数が多いプログラムでは，式 (3) を満たす (i, j) の組み合わせの数が多いときにバンク競合の影響が顕著になる．

この条件の根拠として，図 3 に，STREAM Triad における d_{ij} と実効メモリバンド幅の関係性を示す．図 3 は図 2 のデータを d_{ij} を x 軸にしてプロットしなおしたものである．STREAM Triad は，`c(i) = a(i) + scalar * b(i)` を計算するベンチマークで，ストアよりもロードが多いためロードに着目する．ロードされる配列は `a` と `b` のみであるため，d_{12} のみが評価対象となる．図 3 から分かるように，実効メモリバンド幅が低下しているのは，d_{12} が 0 もしくは $N_{module} \times N_{channel} \times N_{bank}$ に近いときである．これが式 (3) の意味するところであり，妥当性の根拠となる．

なお，この競合条件はシングルスレッドを想定している．マルチスレッドで実行した場合，ループは分割され，ループ内のアクセスパターンは必ずしも各配列の先頭アドレスの差分とは対応しなくなる．シングルスレッド実行時とマルチスレッド実行時の比較は，4.2，4.3 節で行う．

3.2　競合回避策

3.1 節で述べたバンク競合の発生条件を用いて，配列にメモリを割り当てる段階でバンク競合を回避する単純な方法を提案する．提案手法は次のような手順をとる．

(1) 別々に宣言されている配列を 1 つ次元の大きい 1 つの配列に書き換える．(例，`a(N,N),b(N,N),c(N,N)` → `arrays(N,N,3)`) これは，配列を 1 つにまとめることで，実行環境によらず各配列の相対的なアドレス差分を固定することができるためである．

(2) 配列にメモリを割り当てた後，各配列の先頭アドレス情報を取得する．(例，`adrs0 = loc(arrays(0,0,0))`, `adrs1 = loc(arrays(0,0,1))`) このとき，取得するアドレス情報は例示したように 2 つのアドレスだけで

表 2　UPACS-Parts の各プログラム詳細

プログラム名	ロード配列数	ストア配列数	プログラム型
cflux-1d	10	5	ストリーム
vflux-1d	16	5	ストリーム
muscl-1d	5	10	ステンシル
cfacev-1d	6	16	ステンシル

よい．1 つの配列として宣言してあるため，任意の `i` において，`arrays(0,0,i)` と `arrays(0,0,i+1)` のアドレス差分は等しい．そのため，取りうる $p_i - p_j$ の値は，

$$p_i - p_j = n \times (adrs1 - adrs0) \tag{4}$$

と表現できる．ただし，$1 \leq n < N_{array}$ である．

(3) 3.1 節の競合条件と照らし合わせ，競合が起きないようならば次の処理に移る．競合する場合は，一度メモリを開放し，`p` でパディングしてメモリを再度確保し，(2) の手順に戻る (例，`arrays(N,N,3)`→`arrays(N,N+p,3)`)．ただし，この繰り返しには上限を設定する必要がある．これは，どんな `p` でパディングしても，バンク競合発生条件を回避できない場合もありうるからである．

4. 評価

4.1 評価条件

本研究では，CFD(Computational Fluid Dynamics) プログラムのカーネル群である UPACS-Parts(Unified Platform for Aerospace Computational Simulation - Parts) を SX-Aurora 上で実行して，提案手法の効果を評価する．UPACS-Parts はストリーム型とステンシル型のアクセスパターンを持つプログラムがあり，提案手法の対象とする規則的なアクセスパターンを持つプログラムだと言える．

4.1.1 UPACS-Parts

評価には UPACS-Parts と呼ばれるプログラム群を用い，配列サイズを変化させて実行した．UPACS [6] は，マルチブロック構造格子を用いた圧縮性流体の数値シミュレーションの共通基盤コードであり，CFD の基盤技術を共有化・標準化する目的で宇宙航空研究開発機構 (Japan Aerospace eXploration Agency，JAXA) によって開発されている．また，UPACS-Parts は UPACS の主要カーネルを取り出したプログラム群である UPACS-Parts の各プログラムの詳細については表 2 に示す．ロードされる配列数とストアされる配列数の比較に基づき，`cflux-1d` と `vflux-1d` ではロードされる配列に，`cfacev-1d` と `muscl-1d` ではストアされる配列に着目してバンク競合を予測する．なお，UPACS および UPACS-Parts は三次元空間を扱うプログラムのため，ループ構造は 3 以上の多重ループが基本となっている．UPACS-Parts のプログラム群にはメモリアクセスの高速化のために，ループを一重化したバージョンが含まれており，本評価ではそれらを用いた．

表 3　SX-Aurora の詳細

型番	NEC SX-Aurora TSUBASA A300-2 Type 10B
理論倍精度演算性能	2.15 TFLOPS
動作周波数	1.4 GHz
理論メモリバンド幅	1.2 TB/sec
メモリ容量	48 GB
VEOS	VEOS 2.0.3
コンパイラ	nfort 2.1.2

4.1.2 評価環境

本研究の評価は SX-Aurora TSUBASA A300-2 Type 10B を用いて行う．表 3 に SX-Aurora の詳細な情報を示す．

4.2 競合条件の予測精度

本節では，3.1 節で述べた競合条件がどの程度実際の実効メモリバンド幅と関連性を持つか評価する．評価のために，式 (3) を満たす配列の組み合わせの数を N_{hit} として定義する．たとえば，配列 `a`，`b`，`c`，`d` のうち，(`a`,`c`) と (`b`,`d`) の組み合わせで式 (3) を満たすとき，$N_{hit} = 2$ となる．カーネルループ内で扱う配列の数が少ない場合には (STREAM など)，$1 \leq N_{hit}$ のときに高確率でバンク競合による実効メモリバンド幅の低下が顕著になる．一方，配列数の数が多い場合は (UPACS-Parts など)，N_{hit} がある程度大きな値をとるときに実効メモリバンド幅が低下する．本研究は $N_{array} \leq N_{hit}$ の場合，N_{hit} がある程度大きいとみなす．

表 4 および表 5 に N_{hit} と平均実効メモリバンド幅の関係性を示す．それぞれのプログラムについて，$N_{hit} = 0$，$1 \leq N_{hit} \leq N_{array}$，$N_{array} < N_{hit}$ における平均実効メモリバンド幅を示した．なお，配列サイズが小さいときはベクトル長やキャッシュの影響などのバンク競合以外の要因によって実効メモリバンド幅が大きく変動するため，対象とする配列サイズはそれらの影響が十分に小さい 100^3〜180^3 とした．また，図 4 には N_{hit} に対する実効メモリバンド幅の分布を示す．

表 4 および表 5 から，一部のプログラムでは N_{hit} の多寡によって大きく実効メモリバンド幅が変化することがわかる．シングルスレッド実行の `vflux-1d` や 8 スレッド並列実行の `cflux-1d` や `cfacev-1d` では，式 (3) を満たす配列の組み合わせの数が少ないとき ($1 \leq N_{hit} \leq N_{array}$) と多いとき ($N_{array} < N_{hit}$) における平均実効メモリバンド幅は 100GB/s 以上異なる．また，図 4 から，シングルスレッド実行では N_{hit} が小さい場合は，バンク競合による実効バンド幅の顕著な低下がみられないことがわかる．これらの結果から，N_{hit} と実効メモリバンド幅には相関があり，本研究で提案する競合条件によって競合の有無を高精度で予測できることがわかる．

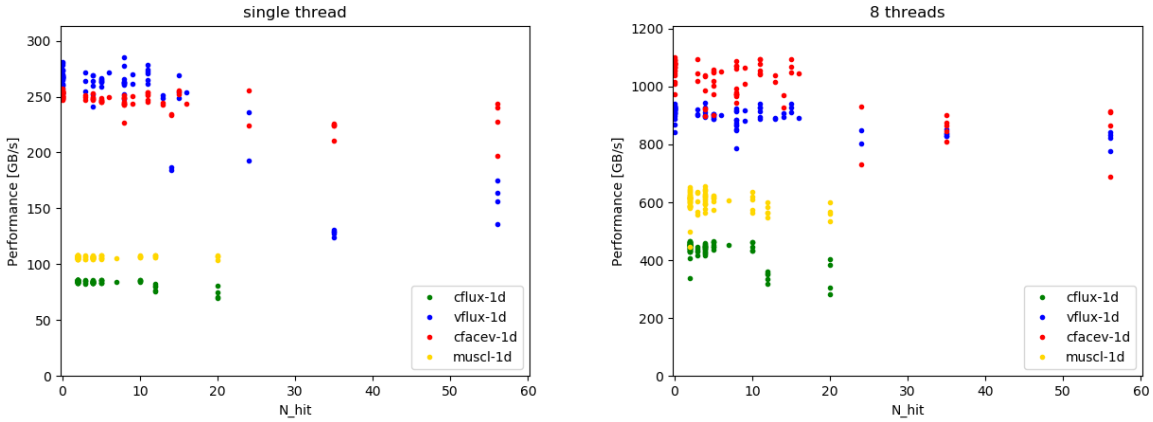

図 4　N_{hit} と実効メモリバンド幅の関係

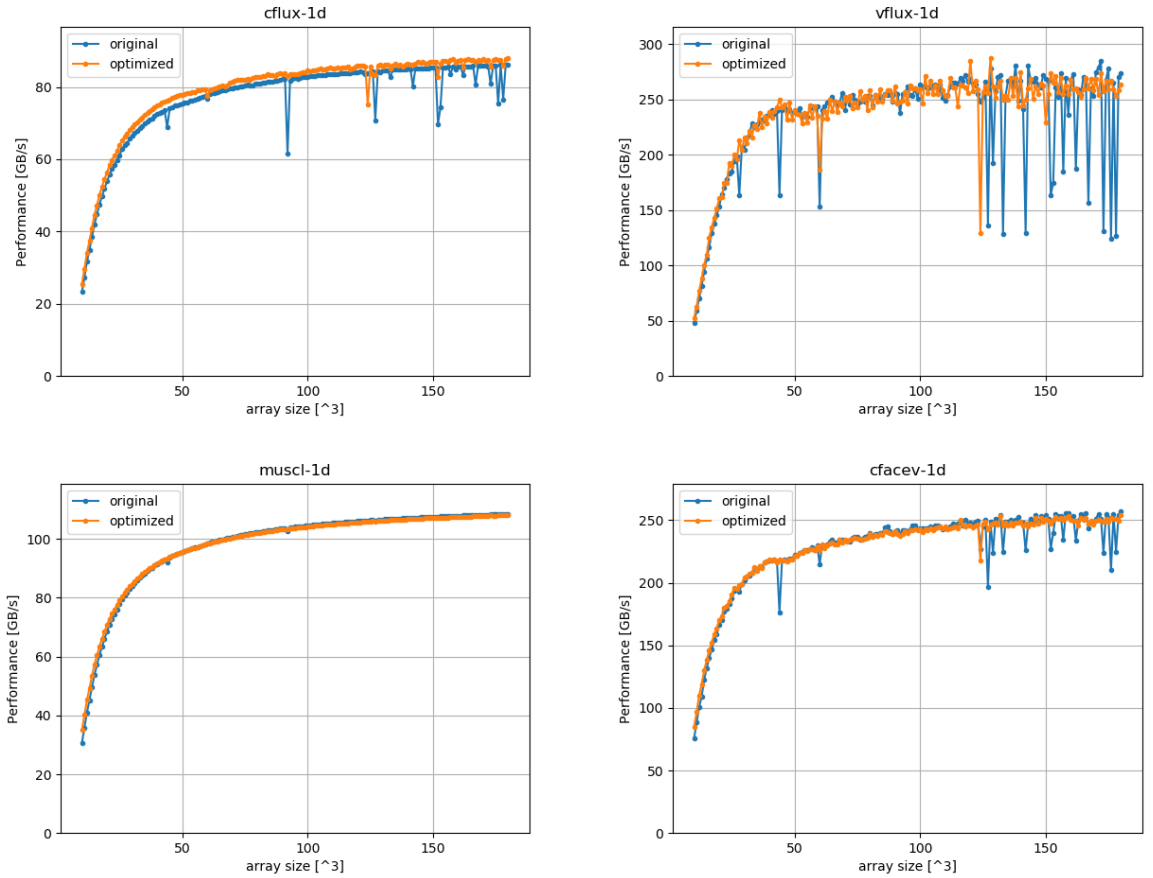

図 5　バンク競合回避策の効果 (シングルスレッド)

ただし，マルチスレッド実行においては，競合条件の予測精度はやや劣る．図 4 の 8 スレッド実行における実効メモリバンド幅分布をみてもわかるように，$0 \leq N_{hit} \leq N_{array}$ においてもいくつかの配列サイズにおいて実効バンド幅の顕著な低下を確認できる．これはマルチスレッド実行によって，複数のスレッドがそれぞれメモリにアクセスするようになり，同時にアクセスされるアドレス間の位置関係が配列の先頭アドレス間の位置関係と必ずしも等価ではなくなったからだと考えられる．マルチスレッド実行における競合予測は今後の重要な課題の 1 つである．

表 4　シングルスレッド実行時の N_{hit} と実効メモリバンド幅の関連性，() は該当する配列サイズの数を示す．

プログラム名	$N_{hit} = 0$	$1 \leq N_{hit} \leq N_{array}$	$N_{array} < N_{hit}$
cflux-1d	該当なし (0)	84 GB/s (72)	76 GB/s (9)
vflux-1d	267 GB/s (26)	259 GB/s (44)	154 GB/s (11)
cfacev-1d	251 GB/s (26)	246 GB/s (44)	226 GB/s (11)
muscl-1d	該当なし (0)	106 GB/s (72)	106 GB/s (9)

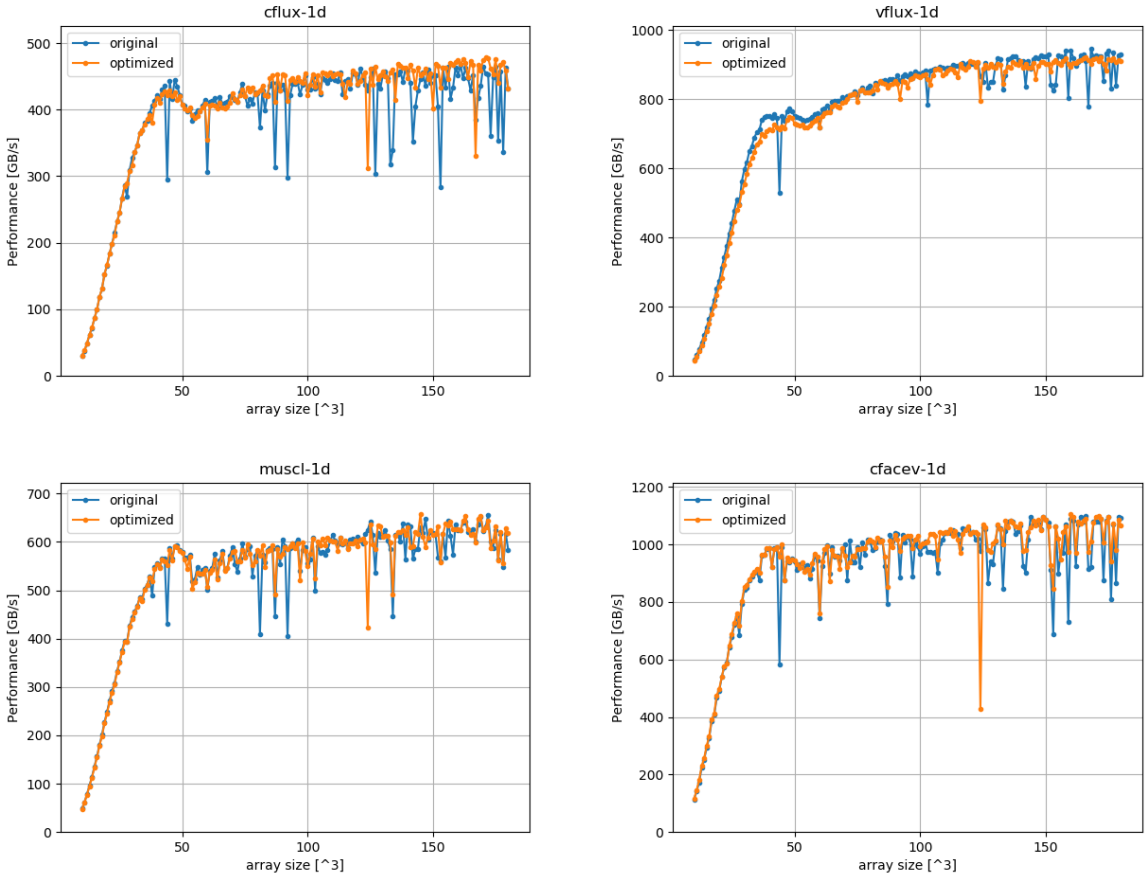

図 6　バンク競合回避策の効果（8 スレッド）

表 5　8 スレッド実行時の N_{hit} と実効メモリバンド幅の関連性，() は該当する配列サイズの数を示す．

プログラム名	$N_{hit} = 0$	$1 \leq N_{hit} \leq N_{array}$	$N_{array} < N_{hit}$
cflux-1d	該当なし (0)	444 GB/s (72)	344 GB/s (9)
vflux-1d	914 GB/s (26)	899 GB/s (44)	828 GB/s (11)
cfacev-1d	1066 GB/s (26)	1019 GB/s (44)	849 GB/s (11)
muscl-1d	該当なし (0)	606 GB/s (72)	569 GB/s (9)

4.3　競合回避策の効果

図 5 にシングルスレッド実行における競合回避策の効果を示す．凡例の original が元来のコードの実効メモリバンド幅を示し，optimized が提案手法を適用したコードの実効メモリバンド幅を示す．

`cflux-1d` と `vflucx-1d`，`cfacev-1d` においては，バンク競合による実効メモリバンド幅低下を格段に減らすことに成功している．ただし，一部の配列サイズで鋭く実効メモリバンド幅が低下しており，バンク競合の影響が顕在化してしまっている．これは，3.2 節で述べたパディングを適用してもバンク競合発生条件を回避できない例である．こういった事例では，パディングする次元を変えたり，別の方法でデータレイアウトを変えることで実効メモリバンド幅低下を回避できると考えられる．`muscl-1d` は，元来のコードでもシングルスレッドでは，バンク競合の影響が見受けられなかった．

図 6 に 8 スレッド実行における競合回避策の効果を示す．マルチスレッド実行時においても，ある程度バンク競合による実効メモリバンド幅低下を軽減できた．しかし，シングルスレッド実行時と比較すると，競合回避策の効果は小さい．これは，4.2 節で述べたように，マルチスレッド実行することによって，アクセスパターンが必ずしも各配列の先頭アドレスの差分に対応しなくなったためである．特に `cfacev-1d` と `muscl-1d` でバンク競合の影響を完全には排除できていない．これは `cfacev-1d` と `muscl-1d` がステンシル型のプログラムであり，メモリアクセスが比較的複雑になることに起因していると考えられる．以上より，マルチスレッド実行に本手法は現時点では不十分であり，さらなる条件の追加や，新しいデータレイアウトの調整方法を導入する必要があることが明らかになった．

5.　おわりに

SX-Aurora に代表される近年のプロセッサにおける実効メモリバンド幅は，バンク競合によって著しく低下する危険性がある．バンク競合が発生するか否かはデータレイアウトに依存し，特定の問題サイズにおいてのみ著しく実効メモリバンド幅が低下する．バンク競合が発生するデー

タレイアウトの条件は自明ではなく，実行前にバンク競合の有無を予測することは容易ではない．そこで，本研究では，カーネルループでロードされる配列の先頭アドレスに着目し，バンク競合による実効メモリバンド幅低下が生じる条件を定式化した．また，この条件を利用し，配列にメモリを割り当てる段階で，バンク競合を回避するデータレイアウトに調整する方法を提案した．さらに，提案手法をCFD プログラムのカーネルにおいて検証し，バンク競合回避に効果があることを明らかにした．また，本評価ではSX-Aurora を用いて行われたが，同様のバンク競合による実効バンド幅低下は Fujitsu FX100 等でも見られており，その回避策は広く他のプロセッサでも効果を期待できる．ただし，マルチスレッド実行への対応など解決すべき問題も明らかになっており，今後さらなる解析と検討が必要である．

謝辞

本研究の一部は，次世代領域研究開発事業量子アニーリングアシスト型次世代スーパーコンピューティング基盤の開発，科研費基盤研究 (B) 16H02822 の支援を受けている．

参考文献

[1] Patterson, D. A. and Hennessy, J. L.: *Computer Organization and Design: The Hardware/Software Interface*, MORGAN KAUFMANN PUBLISHERS (2011).
[2] NEC SX-Aurora TSUBASA: `https://jpn.nec.com/hpc/sxauroratsubasa/index.html`.
[3] Komatsu, Kazuhiko and Momose, Shintaro and Isobe, Yoko and Watanabe, Osamu and Musa, Akihiro and Yokokawa, Mitsuo and Aoyama, Toshikazu and Sato, Masayuki and Kobayashi, Hiroaki: Performance evaluation of a vector supercomputer SX-Aurora TSUBASA, *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis*, IEEE Press, p. 54 (2018).
[4] Naoki Ebata, Ryusuke Egawa, Yoko Isobe, Ryoji Takaki and Hiroyuki Takizawa: Memory First : A Performance Tuning Strategy Focusing on Memory Access Patterns, *ISC 2019 Research Poster*.
[5] STREAM: Sustainable Memory Bandwidth in High Performance Computers: `https://www.cs.virginia.edu/stream/`.
[6] 山本一臣, 高木亮治, 山根敬, 榎本俊治, 山崎裕之, 牧田光正, and 岩宮敏幸: CFD 共通基盤プログラム UPACS の開発 (2000).