キャッシュ競合を制御する性能安定化機構内蔵型数値計算ライブラリについて

今村俊幸[†] 直野健[††]

数値計算ライブラリ開発においてその品質を保証する指標として「性能」というきわめて重要な項目が存在する．近年，自動チューニングという手法で高性能を確保する方式が一般化しつつあるが，当該方式によって適切コードを選択するためには，異なるパラメータでのプログラムセグメントの性能上限を正確に評価するとともに，その性能をつねに再現できることが重要となる．本論文では，データレイアウトに起因するキャッシュ不安定性に対して一考察を与えるとともに，配列の再構成と動的な再配置によって n-way set associative キャッシュの資源競合を回避する手法を提案した．本手法を用いることで，SR8000，Power4，Pentium4 のいずれのハードウェアにおいても性能劣化を改善するとともに，実測性能の標準偏差を平均値の約 2%以内に軽減することに成功した．

Development of a Numerical Library with a Performance Stabilizing Mechanism for Cache Conflicts

Toshiyuki Imamura[†] and Ken Naono[††]

In development of a numerical library, performance issue is a significant metric for guarantees of its quality. Recently a lot of systems introduce high performance libraries with the automatic tuning technology like ATLAS. To select an appropriate code segment due to the particular platform, accurate evaluation of the upper limit performance and assurance of the peak performance are essential factors. In this paper, we focus on the cache instability caused by data layouts, and new stabilizing method is proposed, on which the resource conflicts in an n-way set associative cache are avoided by using data reconstruction and dynamic rearrangement of arrays. The result of a preliminary test shows that our method stabilizes the performance on several platforms, an SR8000, a Power4 and a Pentium 4. The standard deviation of the observed performance is reduced within about 2 percent of the mean value.

1. はじめに

数値計算ライブラリの品質を保証する指標の 1 つ「速度性能」はつねに高いものが要求されているが，近年，複雑に進化し高速化する計算機アーキテクチャに応じた最適化を提案し，それを効率的に実装し「速度性能」を保証し続けていくことは次第に困難になりつつある．この難問題に対して，様々な計算機上でより高い性能を引き出す自動チューニング機能を有したライブラリの研究がさかんに行われるようになっている．自動チューニングとは従来培われてきた経験的な最適化手法を系統的に整理し，インストール時・実行時に最適なコードフラグメントを選択しプログラムを高速に実行する手法である．この手法を採用したライブラリの代表例として，行列数値計算ライブラリではテネシー大学の ATLAS[1]，カリフォルニア大学バークレー校の PHiPAC[2]，東京大学の I-Lib[3]，電気通信大学 Katagiri らの FIBER[4] プロジェクトなどがあげられる．

また，次世代の計算パラダイムとして注目される Grid 環境下において活用可能なネットワーク型ライブラリが提案されており，産業技術総合研究所の Ninf[5]，テネシー大学の NetSolve[6] や SANS プロジェクト[7] などの成果が報告されている．Grid が一般的になる従って，世界規模の仮想的な巨大計算を行う環境での数値計算ポータルの有効性が確認されつつある．ここで，数値計算ポータルの構築を考慮する場合には，高品質のサービスとして「処理速度」や「精度」に関する保証が必要不可欠であると考えられる．特に，Grid

† 電気通信大学電気通信学部情報工学科
Department of Computer Science, The University of Electro-Communications
†† 株式会社日立製作所中央研究所
Central Research Laboratory, Hitachi Ltd.

環境下ではネットワーク上の不特定かつ不均一な計算機資源を利用するため，スケジューリングや課金見積りを厳密に行う必要が発生し，安定した「速度性能」の保証が高品質なサービス実現の重要な鍵となる．

しかしながら，既存のライブラリ開発で行われてきた性能評価では主にピンポイント的なピーク性能に主眼が置かれてきたために，新しいチューニングや計算パラダイムは以下のような問題点をかかえている．

(1) チューニングを行うことによって「高速化」はなされるが，逆にピンポイント的に性能が著しく劣化する可能性がある[8),9)]．

(2) 適用が想定される問題全域にわたってその速度性能が再現されかつ維持されるかという「安定性」も含んだ詳細な評価がなされていない．

(3) チューニングの過程において各種パラメータにおけるサンプリング結果が有用な情報となるが，その情報が信頼できるものか定量的に評価したうえで用いた研究はほとんどない．

特に，(2)，(3) に示したように自動チューニングでは様々なパラメータや次元のもとで関数やループを実測し，適切なパラメータを選定するが，測定結果の信頼性がチューニングされたライブラリの品質に直結することになる．様々な要因によってライブラリの動作環境は変化しその速度性能も変動することが予想できるが，ライブラリの性能は除去可能な性能劣化要因を取り除いたうえで，客観的に保証されなくてはならない．

本論文では，安定化技術の一例として，速度安定性を阻害すると考えられているキャッシュの資源競合の問題に着目する．近代的なマイクロプロセッサに搭載されている n-way set associative キャッシュの構造とチューニングによって生じる特定のデータアクセスパターンを考慮し，キャッシュ資源競合の仕組みに一考察を与える．また，チューニングに起因する 2 種類のキャッシュ資源競合を回避するための基本的アルゴリズムを提案し，ライブラリの実行性能安定化を行い (1)，(2) の問題点に 1 つの解決手法を与える．さらに，サンプリングの変動を極力抑え，統計的にその揺れを評価することで (3) に含まれるサンプリング精度の向上に寄与したいと考える．また，本手法を組み入れた自動チューニング型数値ライブラリの構成方法についても提示する．

本論文の構成を以下に示す．2 章でチューニングとキャッシュの関係を示し，チューニングに起因するキャッシュ資源競合についてまとめる．3 章では，キャッシュ資源競合を回避するアルゴリズムとそれを用いたライブラリの構成方法を示す．4 章では実機測定を通じて提案手法の有効性を検証する．5 章で総括を行う．

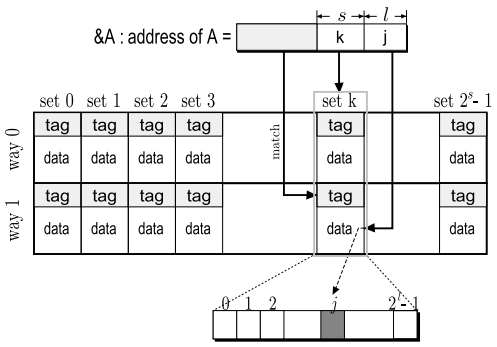

図 1 2-way set associative キャッシュの構成概念図
Fig. 1 Conceptual view of a 2-way set associative cache.

2. チューニングとキャッシュの関係

2.1 準　　備

本論文では現在ほとんどのマイクロプロセッサで採用されている n-way set associative（n 群連想記憶方式）のキャッシュを議論の対象とする．**図 1** は，2 way 構成のキャッシュの概念図を示したものである．n-way 構成の場合，キャッシュ全体は n 個のバンク（way）に分けられそれぞれのバンクではさらにラインと呼ばれる単位に分割される．ラインにはデータとタグが対応し，データ部分にはプロセッサがアクセスするデータが格納される．さらに，タグ部分にはラインの仮想記憶上でのアドレスの上位ビットを記憶する．そして，同一の列に位置する n 個のラインの集合をセットと呼ぶ．

一般に，ライン数やラインサイズは 2 のベキであり，本論文ではそれぞれ 2^s，2^l と表記することとする．仮想記憶上のアドレスとセットとの対応は，アドレスの下位 $s+l$ ビットのうちの上位 s ビット値のセットで一意に決定される．また，アドレスの下位 $s+l$ ビットを落とした値をタグ部分に格納した way が存在すれば，その way における対応ラインに指定アドレスを含むデータが格納されていることになる．一方，いずれの way にもタグ部分に対応するアドレス情報がない場合は，キャッシュミスの状態にあるという．

近代的なプロセッサでは，キャッシュは階層構造をなしており，L1（レベル 1）から L2，L3 の 3 階層のものも存在する．L1 はプロセッサに最も近い位置に設置され高速・低レインテシであるが容量は小さい．逆に，L2，L3 となるに従って大容量・低速になる．

さらに，キャッシュと主記憶の間には仮想記憶のページと物理記憶装置との対応表を格納する TLB

(Translation Lookaside Buffer) が存在する．TLB はキャッシュと同様の構造をしているが，そのエントリ数はライン数よりもずっと少ない場合が多い．

2.2 キャッシュにおける位相について

先に示したように，データに対応するセットはアドレスの特定のビット位置から一意に決定されるが，その対応は連続アドレスを round-robin 方式に割り当てるものであり，端に位置するセットでは隣接セットの取扱いが難しい．今，倍精度浮動小数点 (double) のみを扱うこととし，同一セットを使用する変数を数学的に取り扱うために，次の量を定義する．

$$[\mathrm{I}]_L := \mathrm{mod}(\mathrm{I} + 2^{L-1}, 2^L) - 2^{L-1} \quad (1)$$

$$^*\mathrm{A} := \&\mathrm{A}/2^d \quad (2)$$

ここで，&A は A のアドレスを示し，$L = s + l - d$, $d = \log_2 \mathrm{sizeof(double)}$ とする．$[\cdot]_L$ はデータが占めるキャッシュ位置の**位相**を表し，次の性質を持つ．

(1) $-2^{L-1} \le [\cdot]_L < 2^{L-1}$.

(2) $[^*\mathrm{A}]_L = [^*\mathrm{B}]_L$ のとき，A と B は同じセット位置を占める．

(3) $[^*\mathrm{A} - {}^*\mathrm{B}]_L$ は A に対する B のキャッシュ上の**変位** (ワード) を示す．

(4) $[^*\mathrm{A(I)} - {}^*\mathrm{A(J)}]_L = [\mathrm{I} - \mathrm{J}]_L$.

特に，(2) の性質にあるとき A と B は基本周期 2^L で**同位相**にあるという．また，(3) よりキャッシュ上における変数間の相対的な位置関係を認識することができる．たとえば，$|[^*\mathrm{A} - {}^*\mathrm{B}]_L| \ge 2^{l-d}$ の場合は同一セットを占めることはないが，$|[^*\mathrm{A} - {}^*\mathrm{B}]_L| < 2^{l-d}$ の場合は同一セットを占める可能性がある．

2.3 n-way set associative キャッシュでの資源競合についての考察

記憶装置の階層構造において，キャッシュ総容量は主記憶容量に比較すると少量となる．そのため，問題全体をキャッシュに収めることができず，主記憶とキャッシュ間でのデータ転送が発生し性能を劣化させる．それを改善するために，多くのコンパイラやライブラリではプリフェッチやタイリングなどの手法を用いてキャッシュラインや TLB の再利用率の改善を行う研究が多く報告されている[1),2),10)]．

キャッシュ容量に起因する性能劣化のほかに，アクセス順序に起因する深刻な問題が存在する．一般的にキャッシュセットに格納されたラインの全データが使用されてから，同一セットに対応する他のラインがアクセスされれば，キャッシュ本来の緩衝材としての役割を果たす．しかしながら，同一セットを利用するラインが way 数以上にあり，かつ，それらを順々にアクセスする場合，1 ワード利用するたびにキャッシュ–主記憶間のデータ転送が発生し著しく性能が劣化する．

```
dimension U(NU,N),V(NV,N)
do I=1,N,M
  do J=1,N
      A(J,I)=A(J,I)-V(J,I)*U(J,I) &
                   -V(J,I+1)*U(J,I+1) &
                    ...
                   -V(J,I+M-1)*U(J,I+M-1)
  enddo
enddo
```

図 2 同一配列アクセスにおけるセット競合を引き起こすプログラムの例 (外側ループのアンローリングが原因となる場合)

Fig. 2 Example code which attracts a set conflict in multiple streams of the same array (the case of an outer loop expansion).

```
do I=1,N
  do J=1,N
      A(J,I)=A(J,I)-V1(J,I)*U1(J,I) &
                   -V2(J,I)*U2(J,I) &
                   -V3(J,I)*U3(J,I) ...
  enddo
enddo
```

図 3 多変数利用によるセット競合を引き起こすプログラムの例

Fig. 3 Example code which attracts a set conflict in multiple streams of multiple variables and arrays.

本論文では，後者のキャッシュ資源競合を，同一セット利用に由来する資源競合という意味で**「セット競合」**と呼ぶ．このセット競合はチューニングによるループ最適化が原因で生じることがある．次に，チューニングが原因の 2 種類のセット競合を示す．

2.3.1 同一配列アクセスにおけるセット競合

自動チューニングを行ううえで，ループアンローリングやループ展開はきわめて有用な技法である．しかしながら，アンローリングによってループ内に異なるアドレスへアクセスする実行文が増加する．特にその展開が多次元配列の第 2 インデックス以降の場合，アドレス間隔 (以降，**整合寸法**と呼ぶ) が通常 2 以上の一定値であるため，異なるラインによる同一セット利用の可能性が周期的に生じ，セット競合が起こりうる．

図 2 は 2 重ループの外側ループを M 段展開したものである．このループにおいて，$[^*\mathrm{V(J,I+1)} - {}^*\mathrm{V(J,I)}]_L = \ldots = [\mathrm{NV}]_L$ が成り立つため，同位相つまり $[\mathrm{NV}]_L = 0$ のときはすべての実行文が同一セットを使用し，セット競合となる．

2.3.2 多変数利用によるセット競合

図 3 のようにループ中に複数の配列アクセスが存在する場合にもセット競合の可能性がある．たとえば，配列を動的に確保する際の alignment によって，way 数を超えた配列が同一のセットを使用する場合がこれに該当し，図 3 で，全配列が同位相 $[^*\mathrm{A}(1,1)]_L = [^*\mathrm{V1}(1,1)]_L = [^*\mathrm{U1}(1,1)]_L = \ldots$ でかつ，2 ならびに

4way 構成のキャッシュを使用する場合である．特に，ループ融合など最適化によってループ内の変数が増加することで，セット競合を誘発する可能性がある．

3. 性能安定化機構とその実装

文献 8)，9) では，同一配列のアクセスに起因するセット競合が原因の性能劣化が報告されているが，その性能劣化区間は 2 ベキ次元の周辺のほかに何らかの規則性があるものと予測される．

同一配列アクセスにおける性能劣化を数学的に扱うために，同一セットを使用する可能性がある整合寸法について定式化を試みる．今，配列 a の整合寸法を N_a とする．2.2 節の性質 (4) から，外側インデックスについて i 間隔のアクセス，つまり，`a(J,I)`, `a(J,I+` i `)`, `a(J,I+` $2i$ `)`, $\cdots$ の間で同一セットを利用するための必要条件は次の不等式が成立することである．

$$\exists i > 0,\ |[i * N_a]_L| < 2^{l-d} \tag{3}$$

実際プリフェッチやコンパイラの最適化などで隣接ラインも同時に利用されることを考慮し，2^{l-d} の代わりに実験から判明した整合寸法 2^L を中心とする性能劣化区間 δ を用いて，十分条件として取り扱う．

キャッシュが n-way 構成のとき，同一のセット利用は n 個まで許容することができる．したがって，k 段の外側ループアンローリングを行う場合，不等式 (3) を満足する最小の i に対して $n \geq k/i$ であれば way の構成範囲内の利用であるが，逆に $n < k/i$ であれば実装資源以上の利用となりセット競合を引き起こす．これを形式化すると以下のような判別式が導かれる．

［セット競合の判別式］

n-way set associative キャッシュを搭載したプロセッサを用いたシステム上でのループ実行を仮定する．整合寸法が N_a の 2 次元配列 a に対して，以下の不等式が成立するとき，配列 a の上位インデックスを k 段展開するアンロールは，実行時にセット競合を生じさせる．

$$0 < \exists i < k/n,\ |[i * N_a]_L| < \delta, \tag{4}$$

3.1 ヒューリスティクス法

判別式 (4) は配列 a の整合寸法 N_a にのみ依存するため，判別式を満足しない範囲にある整合寸法 $N'_a(\geq N_a)$ を発見し，その整合寸法で配列を再構成すればセット競合は起こらない．δ が決まれば上の判別式から整合寸法 N'_a は次のように求められる．

$$N'_a = N_a + \lceil(\delta - [i * N_a]_L)/i\rceil \tag{5}$$

ここで，整合寸法が増大するために配列元来の記憶域を超えることになるが，別に補助配列を確保し 2 配列をあわせて不足域を補う方式をとればよい[9)]．

本手法は，文献 9) において経験的に決められた δ で安定化できた事実を形式化したものであり，以降**ヒューリスティクス法**と呼ぶ．

3.2 位相を考慮した構成手法

今，配列の整合寸法 $\mathrm{N_a}$ が $2^u(\leq 2^L)$ の倍数のときに，次の性質が成り立つ．

$$\forall k \geq 0,$$
$$\mathrm{mod}([k\mathrm{N_a} + b]_L, 2^u) = \mathrm{mod}(b, 2^u) \tag{6}$$

この式は，整合寸法が 2^u の倍数の多次元配列の場合に，第 1 インデックスが一致するデータが格納されるキャッシュセットは，2^u を基本周期とする位相を一定とすることを意味している．

この性質を用いると，ループ内に登場する全配列の整合寸法を 2^u の倍数に再構成し，かつ，それぞれの開始データのキャッシュ上の位相を 2^u の剰余のもとですべて異なるように設定できれば，多変数利用のために生じる配列間でのセット競合を解消できる．

3.3 安定化機構を組み込んだ数値ライブラリの構成方法

先に示した，ヒューリスティクス法や位相を考慮した手法では，それぞれ 2.3.1 項，2.3.2 項に示した原因に由来するセット競合しか解消できない．本節では，先の 2 手法を組み合わせ，両問題に由来するセット競合を同時に解消する方法を提案する．

両手法を組み合わせて，セット競合を解消したキャッシュ性能安定化アルゴリズムを以下に示す．

［キャッシュ性能安定化アルゴリズム］

(0) ループ内に登場する配列を，$V_1, V_2, \ldots, V_m$，それぞれの整合寸法を $N_1, N_2, \ldots, N_m$ とする．ただし，$m \leq 2^{L-l+d}$，$p = \lceil \log_2 m \rceil$ とする．

(1) $N'_j := N_j$ とおく．

(2) 位相構成法の基本周期を $f = 2^{p+l}$ とし，各配列の整合寸法を f の倍数に調整する．

$$N'_j := \lceil N'_j/f \rceil * f \tag{7}$$

(3) 判別式 (4) を満足する場合は，次式により整合寸法を決定しステップ (2) に戻る．

$$N'_j := N'_j + \lceil(\delta - [i * N'_j]_L)/i\rceil \tag{8}$$

(4) $[\mathrm{{}^{*}V_j}]_L = j \cdot (f/2^p)$ となるように，配列個々のオフセットを決定する．

(5) $V_1, V_2, \ldots, V_m$ をそれぞれ整合寸法 $N'_1, N'_2, \ldots, N'_m$ で再構成する．これ以降，もともとの配列 V_a を意味するときは補助配列 V'_a とあわせて (V_a, V'_a) の形で表記する．補助配列が不要な場合には $V'_a = \emptyset$ とする．

(6) $(V_1, V'_1), (V_2, V'_2), \ldots,$ を用いた形に変形したコ

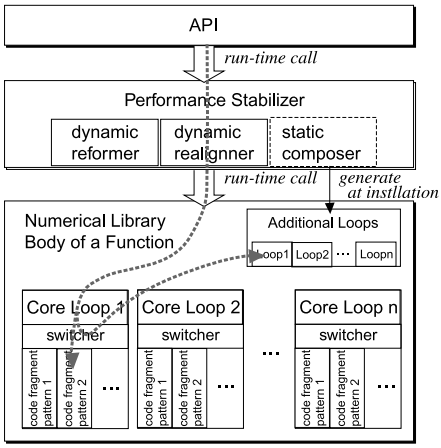

図 4 性能安定化アルゴリズムを組み込んだ自動チューニングライブラリの概要図
Fig. 4 Schematic view of the auto-tuning library introducing the performance stabilizing algorithm.

アループを実行する．

(7) 出力が必要な変数について，$(V_a, V_a') \to V_a$ の再々構成を行う．

ここに示したアルゴリズムはライブラリのインタフェース部分と本体部分との間で機能するものである．また，安定化アルゴリズムによって補助配列が導入されるため，ライブラリ本体に補助配列を処理するコードを追加しなくてはならない．ただ，補助配列のための処理は元の配列の処理とほぼ同様であるため，配列名を変えただけの複製として自動生成が可能である．

なお，安定化処理された配列に対しては従来の最適化が行えるため，これまで研究されてきた自動チューニングの枠組みでの高速化が可能である．この事実を考慮したうえで，性能安定化アルゴリズムを組み込んだ自動チューニングライブラリを容易に構成することができる．**図 4** は本アルゴリズムを性能安定化機構として数値ライブラリに組み込んだものの構成概念図である．

配列の整合寸法を動的に変更する reformer と適切な位相に配列を配置する realigner，さらに補助配列が出現したときに，補助配列を含むコードセグメントを生成する composer などの 3 要素で構成される．composer はインストール時にコード生成を行うのみだが，他の reformer，realigner は実行時にその機能を発揮する．また，補助配列用のコードセグメントが実行時に呼ばれることになるため，各コアループにおいて自動的に最適コードを選択する switcher 部分が補助配列用の追加ループを呼び出すなどの変更が必要となる．

```
do I=1,NV
  do J=1,N
      ZR(J,I)=ZR(J,I) &
        +UR(I,1)*WR(J,1)-UI(I,1)*WI(J,1) &
        +UR(I,2)*WR(J,2)-UI(I,2)*WI(J,2) &
                        ...
        +UR(I,M)*WR(J,M)-UI(I,M)*WI(J,M)
      ZI(J,I)=ZI(J,I) &
        +UR(I,1)*WI(J,1)+UI(I,1)*WR(J,1) &
        +UR(I,2)*WI(J,2)+UI(I,2)*WR(J,2) &
                        ...
        +UR(I,M)*WI(J,M)+UI(I,M)*WR(J,M)
  enddo
enddo
```

図 5 WY 表現のハウスホルダ逆変換の第 2 コアループ
Fig. 5 Second core loop of the inverse Householder transform routine implemented in a WY-representation.

なお，これら機構はコンパイラなどの構文解析機能を用いることで自動生成することができるが，本論文では安定化機構の構成法の提案にとどめた．つまり，reformer，realigner 部分は半自動生成しているが，composer によって生成されるべき追加コードや switcher の追加機能は人手によって生成している．

4. 性能測定と検証

4.1 対象問題

性能安定化機構の実システムでの検証問題として，密エルミート行列の固有ベクトル求解で必要となるハウスホルダ逆変換ルーチンを用いた．今回用いたルーチンは WY 表現[11] によりブロック化され，複素数を実部と虚部の 2 配列に分けて実装したものである．

アルゴリズム中には内積計算とベクトル和の 2 つのコアループが含まれるが，後者を FORTRAN によって記述したものが**図 5** である．WY 表現によるブロック幅のチューニングパラメータ（M）を持ち，複素数を扱うためループ内に現れる配列の数が比較的多い問題となる．したがって，同一配列のアクセスと複数配列使用に起因する 2 種類のセット競合が生じる．

4.2 実　　験

本節では，日本原子力研究所所有の日立 SR8000F1 モデルと IBM pSeries 690，デスクトップ PC（以下 Pentium4 と呼ぶ）による実験結果を示す．実験に使用した計算機に搭載されたプロセッサのスペックは**表 1** のとおりである．また同時に，使用したコンパイラと使用オプション，安定化アルゴリズムに用いたパラメータについても記した．L1 キャッシュの way 数を考慮して，チューニングパラメータを M=5 として，実験を行った．各次元で 2 回測定を行い，高い数値の方をその次元の性能として採用した．なお，pSeries690 は単一プロセッサ上に 2 コアを持つが，今回の実験では単一コアのみを使用した．

表 1 計算機のハードウェアスペックとコンパイラ情報
Table 1 Hardware and compiler specifications.

	日立 SR8000F1	IBM pSeries690	Desktop PC
プロセッサ	Power3 拡張型	Power4（2 コア）	Pentium 4
クロック	375 MHz	1.3 GHz	2.8 GHz（FSB533 MHz）
理論ピーク性能	1.5 GFLOPS	5.2 GFLOPS/コア	5.6 GFLOPS
L1 キャッシュ	128 KB，4-way	32 KB/コア，2-way	8 KB，4-way
（ラインサイズ）	128 B	128 B	64 B
L2 キャッシュ	—	512 KB × 3，8-way	512 KB，8-way
（ラインサイズ）		128 B	128 B
L3 キャッシュ	—	32 MB，8-way	—
（ラインサイズ）		512 B	
コンパイラ	日立 FORTRAN コンパイラ V01-05	IBM XL Fortran コンパイラ 8.1	Intel Fortran Compiler for Linux v7.1
（使用オプション）	-64 -Os -noparallel -pvec -model=F1	-q64 -O5 -qarch=pwr4 -qtune=pwr4	-O3 -tpp7 -axW -f-noalias
性能安定化用パラメータ (δ, L, l, p, d)	$(32, 12, 7, 3, 3)$	$(32, 11, 7, 3, 3)$	$(16, 8, 3, 3, 3)$

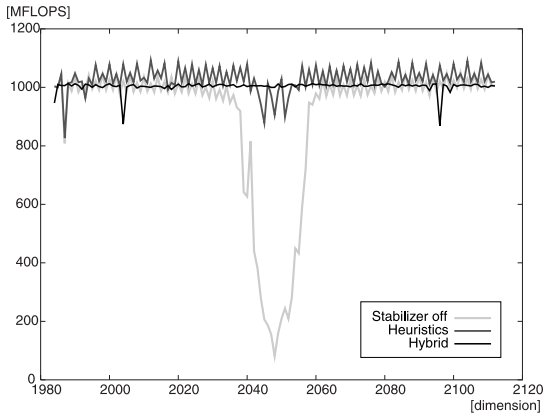

図 6 2048 ± 64 次元の性能．SR8000F1 モデル
Fig. 6 Performance observed between 2048 ± 64 dimension on an SR8001 model F1.

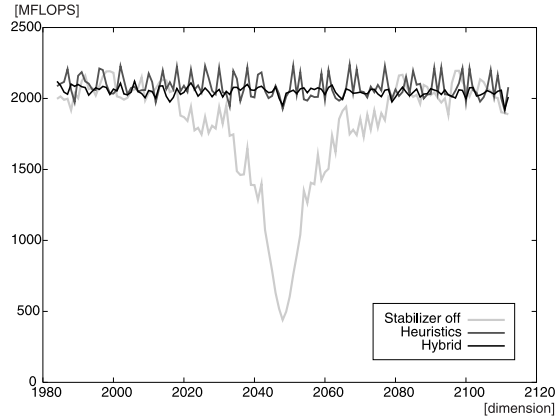

図 7 2048 ± 64 次元の性能．pSeries690
Fig. 7 Performance observed betweeen 2048 ± 64 dimension on a pSeries690.

図 6，**図 7**，**図 8** は行列の次元 N を 2048 を中心に ±64 の範囲で 1 刻みで変化させたときのハウスホルダ逆変換ルーチンの実行性能（MFLOPS）をプロットしたものである．性能安定化を行わない「安定化なし（Stabilizer off）」と，ヒューリスティクス法のみを実施した「Heuristics」，2 手法を同時に実施した複合手法「Hybrid」の 3 パターンについて測定している．

グラフから分かるように，性能安定化を行わない場合には 2048 次元での急激な劣化が観測できる．特に，SR8000F1 と pSeries690 ではその劣化の度合いが著しい．また，ヒューリスティクス法ならびに複合手法では，性能劣化が改善されていることが分かる．

次に，**表 2** は観測データを統計処理し，各実行環境における性能の区間平均，標準偏差ならびに平均値に対する割合を算出したものである．平均性能を見るとヒューリスティクス法が最も高い．また，性能が劣化する区間では「安定化なし」の結果は決して良いもの

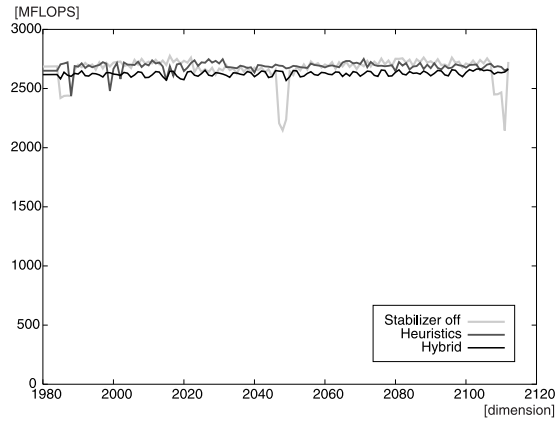

図 8 2048 ± 64 次元の性能．Pentium 4
Fig. 8 Performance observed between 2048 ± 64 dimension on a Desktop PC installed with a Pentium 4.

表 2 2048 次元を中心として ±64 近傍でのキャッシュ安定化機構の効果
Table 2 Effect of the cache stabilizing mechanism at the dimension between 2048 ± 64.

		SR8000F1	pSeries690	Pentium4
安定化なし	平均性能	906.705	1806.25	2669.8
	標準偏差	238.641	396.328	110.742
	標準偏差/平均性能	0.2631	0.2194	0.0414
Heuristics	平均性能	1028.66	2082.01	2689.18
	標準偏差	41.6849	78.7963	39.7363
	標準偏差/平均性能	0.0405	0.0378	0.0147
Hybrid	平均性能	1003.60	2048.17	2625.84
	標準偏差	17.3325	31.3857	21.1173
	標準偏差/平均性能	0.0172	0.0153	0.0080

〔上段：平均性能（MFLOPS），下段：標準偏差（MFLOPS）と標準偏差/平均性能〕

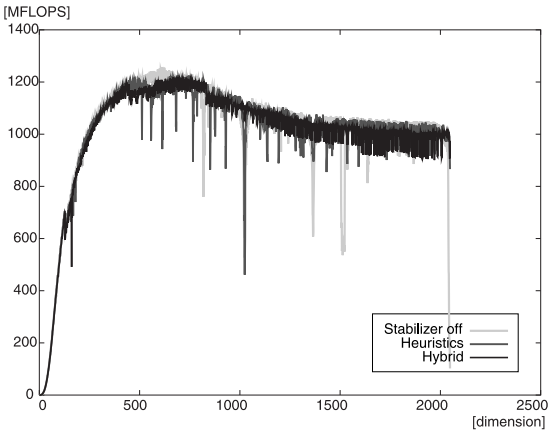

図 9 1〜2048 次元の性能．SR8000F1 モデル
Fig. 9 Performance observed from 1 to 2048 dimension on an SR8000 model F1.

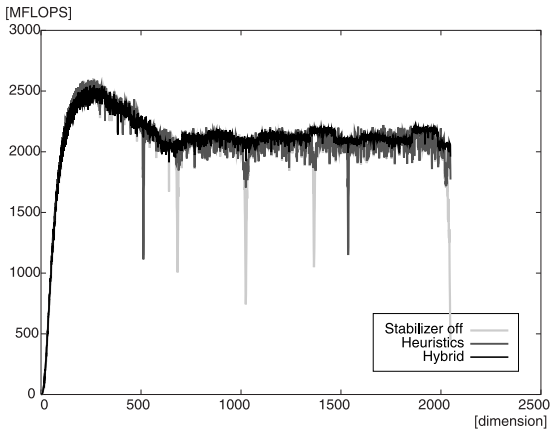

図 10 1〜2048 次元の性能．pSeries690
Fig. 10 Performance observed from 1 to 2048 dimension on a pSeries690.

ではないが，その区間から遠ざかると最も良い性能を記録するものも見られる．これは，安定化処理のオーバヘッドによるものであり，それゆえに，平均性能と比較するとわずかではあるが複合手法が最も遅くなる区間が存在する．

性能の振れ具合を見るために標準偏差に着目すると，複合手法がそれぞれの計算環境で最小値をとっていることが分かる．そしてすべての環境で平均性能に対し 2%以下であり，安定していることが分かる．ヒューリスティクス法単体でも標準偏差は小さいが，複合手法の約 2 倍に達する．

次に，さらに広い区間での性能変化を見るために 1〜2048 次元まで 1 刻みでの実行結果を**図 9**，**図 10**，**図 11** に示す．各グラフから,「安定化なし」では判別式が成立する次元で性能が著しく劣化していることが分かる☆．ヒューリスティクス法は安定化を行わない

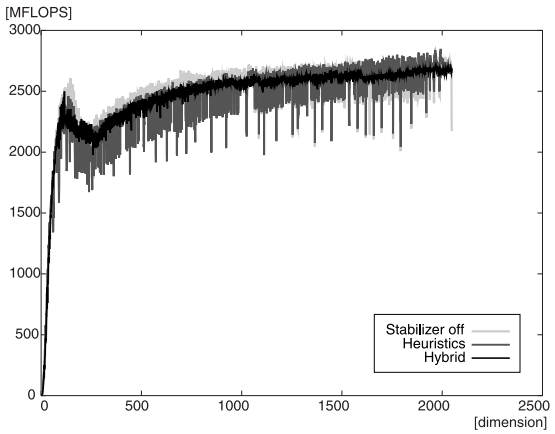

図 11 1〜2048 次元の性能．Pentium 4
Fig. 11 Performance observed from 1 to 2048 dimenstion on a Pentium 4.

ものより性能の変動は小さいが，いくつかの次元で性能劣化が生じている．複合手法は一定の揺れはあるものの，他 2 手法に見られるような極端な性能劣化は見

☆ 1 から 2048 の区間では 512, 683, 1024, 1365, 1536, 2048 次元近傍で判別式 (4) が成立する．

表 3 1 から 2048 次元までのキャッシュ安定化機構の効果
Table 3 Effect of the cache stabilizing mechanism at the dimension from 1 to 2048.

		SR8000F1	pSeries690	Pentium4
Heuristics	パラメータ (a_1, b_1)	(1143.26, 313.371)	(2134.02, 50.7317)	(2813.46, 103.668)
	(a_2, b_2)	(7286.58, 911.733)	(108437, 4707.19)	(37936.9, 1170.44)
	(c, d)	(0.00152, −165.996)	(0.00446, −104.938)	(0.0145, −103.009)
	SQRT(SSR/n)	50.90	101.1	119.3
	{SQRT(SSR)/n}/a_1	0.0445	0.0473	0.0424
Hybrid	パラメータ (a_1, b_1)	(1147.00, 374.153)	(2203.03, 55.1822)	(2770.12, 82.4459)
	(a_2, b_2)	(7972.82, 887.608)	(108428, 4825.25)	(758739, 24770.2)
	(c, d)	(0.00131, −67.1947)	(0.00461, −101.918)	(0.0160, −100.762)
	SQRT(SSR/n)	36.20	64.90	44.97
	{SQRT(SSR/n)}/a_1	0.0315	0.0294	0.0162

〔上段：fitting パラメータ，下段：二乗残差平均の平方根 SQRT（SSR/n）ならびに a_1 との比〕

られない．

また，2048 ± 64 次元区間での測定と同様の統計処理を行うために，性能特性をピーク性能に 1 次漸近収束する関数 $f(x) = ax/(x + b)$ で近似することを行う☆．しかし，グラフの形状から，全データが L1（もしくは L2）キャッシュに収まる場合とそうでない場合とで様相が異なることが分かる．そこで，次式のように 2 種類の関数を滑らかに接続し近似することとする．

$$f(x) = f_1(x) \cdot (1 + \tanh c(x + d))/2 + f_2(x) \cdot (1 - \tanh c(x + d))/2 \quad (9)$$

$f_1(x) = a_1 x/(x + b_1)$,

$f_2(x) = a_2 x/(x + b_2)$, ただし $a_1 < a_2$.

非線形の最小二乗法を用いて近似した結果を**表 3** に示す．「安定化なし」の場合はデータ変動が激しすぎるため，「ヒューリスティクス法」と「複合手法」に限って近似を試みた．近似によって求められた係数 a_1 は，十分大きな次元における漸近性能，つまりピーク性能の推定値を意味している．また SQRT（SSR/n）は二乗残差和の平均に対して平方根をとったものであり，標準偏差と同等のものと考えてよい．

2048 次元近傍と同様に，標準偏差相当の SQRT（SSR/n）はきわめて小さい値をとっておりピーク性能の推定値と比べても複合手法で 2%以下となる．ヒューリスティクス法も性能変動の平均的な振舞いは許容できるものの，「安定化なし」と同程度の不安定性を引き起こす次元が数多く見られる．

今回の実験は，各次元でわずか 2 回の試行を行ったものにすぎないのだが，複合手法において全区間での性能が滑らかな曲線で近似することができた．これは各次元での試行においてコード生成時に意図した性能が再現されていることを意味しており，複合手法が性能安定化という意味で優れていることが確認できる．

☆ 本ルーチンのコストは N に対して 3 次多項式で表現されるため，性能特性は 3 次有理式で表すべきであるが，すべての極が N が負の領域に分布するため 0 に最も近い極を主要項としてコストを近似する．

5. ま と め

本研究では，既存のライブラリ性能評価においてあまり重要視されてこなかった「安定性」に関する考察を行った．特に，チューニングによって性能が劣化するという問題とともにライブラリの高速実行を阻むキャッシュの資源競合との関係に注目した．n-way set associative キャッシュの資源競合を誘引する 2 つの問題に対して，回避アルゴリズムを提案するとともに，その仕組み「性能安定化機構」を取り入れたライブラリの構成法について述べた．また，実システムにおいて提案手法を検証するとともにその効果を確認した．提案した性能安定化機構によってライブラリの性能はきわめて安定的な挙動を示すようになった．

本研究では，n-way set associative キャッシュの資源競合にのみ注目し満足いく結果を得られたが，キャッシュの階層構造や SMP での問題の解析，さらに TLB に対してもその影響や今回手法の有用性を検証する必要がある．また，性能安定化機構のライブラリの組み込みは，半自動的に行われているが，コンパイラや指示子などにその機能を付加した開発フレームワーク構築が重要となる．さらに，性能安定化の仕組みは自動チューニング技術の基盤技術に位置付けられるものであり，両者を連携させて実行性能を高精度に保証する数値計算ライブラリを開発することが今後の課題である．

最後に，並列計算機環境をご提供いただいた日本原子力研究所計算科学技術推進センター，有用なコメントをいただいた匿名査読者各位に感謝いたします．

参 考 文 献

1) Whaley, R.C., Petitet, A. and Dongarra, J.J.: Automated Empirical Optimization of Software and the ATLAS Project, LAPACK Working Note 147 (2000).
2) Bilmes, J., Ananoic, K., Chin, C.-W. and Demmel, J.: Optimizing Matrix Multiply using PHiPAC: a Portable, High-Performance, ANSI C Coding Methodology, *Proc. International Conference on Supercomputing 97*, pp.340–347 (1997).
3) 片桐孝洋，黒田久泰，大澤 清，工藤 誠，金田康正：自動チューニング機構が並列数値計算ライブラリに及ぼす効果，情報処理学会誌：ハイパフォーマンスコンピューティングシステム，Vol.42, No.SIG12(HPS 4), pp.60–76 (2001).
4) Katagiri, T., Kise, K., Honda, H. and Yuba, T.: FIBER: A Generalized Framework for Auto-tuning Software, *Proc. 5th International Symposium, ISHPC2003*, LNCS 2858, pp.146–159 (2003).
5) Nakada, H., Sato, M. and Sekiguchi, S.: Design and Implementations of Ninf: towards a Global Computing Infrastructure, *Future Generation Computing Systems, Metacomputing Issue*, Vol.15, Issues 5-6, pp.649–658 (1999).
6) Arnold, D., Agrawal, S., Blackford, S., Dongarra, J.J., Miller, M., Seymour, K., Sagi, K., Shi, Z. and Vadhiyar, S.: Users' Guide to NetSolve V1.4.1, Technical Note of University of Tennessee, ICL-UT-02-05 (2002).
7) Dongarra, J. and Eijkhout, V.: Self-adapting Numerical Software for Next Generation Applications, *The International Journal of High Performance Computing and Applications*, Vol.17, No.2, Summer 2003, pp.125–131 (2003).
8) 直野 健，今村俊幸：自動チューニング型の固有値ソルバーについて，情報処理学会研究報告，Vol.2002, No.91, pp.49–54 (2002).
9) 今村俊幸，直野 健：性能安定化を目指した自動チューニング型固有値ソルバーについて，先進的計算基盤システムシンポジウム SACSIS2003, pp.145–152 (2003).
10) Goto, K. and Van de Geijn, R.: On Reducing TLB Misses in Matrix Multiplication, FLAME Working Note #9, Technical Report of The University of Texas at Austin, Dept. of Computer Science, TR-2002-55 (2002).
11) Golub, G. and Van Loan, C.: *Matrix Computations* third edition, The Johns Hopkins University Press (1996).

(平成 15 年 10 月 10 日受付)
(平成 15 年 11 月 28 日採録)

今村 俊幸（正会員）

1969 年生．1996 年京都大学大学院工学研究科応用システム科学専攻博士後期課程単位認定退学．同年日本原子力研究所入所．計算科学技術推進センターにて途切れのない思考を支援する並列処理基本システム STA の開発に従事．2001 年から 2002 年までシュツットガルト大学 HLRS にて招聘研究員．2003 年より電気通信大学講師．現在に至る．HPC とその周辺ソフトウェア，数値計算における並列・分散処理の研究に従事．博士（工学）．平成 11 年日本応用数理学会論文賞，同年石川賞企業部門受賞．日本応用数理学会，SIAM 各会員．

直野 健（正会員）

1968 年生．1994 年京都大学大学院理学研究科数理解析専攻修士課程修了．同年（株）日立製作所中央研究所入所．以来，並列計算機 SR2201, SR8000 向け行列計算ライブラリの研究開発に従事．現在の主たる研究テーマは，HPC の研究，並列固有値計算ライブラリ，HPC の金融工学への応用．日本応用数理学会，日本金融・証券計量・工学学会各会員．