新スーパーコンピュータのハードウェアについて

―― キャッシュメモリ ――

金澤　正憲*

1 はじめに

この3月1日からサービス開始予定のスーパーコンピュータは、現在のVPP800のようなベクトル並列機ではなくて、多数のCPUが主記憶を共有した計算機（SMPという）を並列にしたシステム（SMPクラスタという）に置き換わることを、前々回の広報でお知らせしました。新しいスーパーコンピュータをHPCと呼ぶことにします。

プログラムを速く実行させるには、同時に多くのCPUを使って並列処理を行なわせることがポイントです。即ち、スレッド並列、プロセス並列といった並列化技法を習得することが重要です。

しかし、逐次実行（1CPUでの実行）で速くしておいて、次に、それを並列化してさらに高速化するというのが普通の考え方でしょう。並列化しておいてから、それぞれのCPUで実行されるものを高速化するのは少し考えづらいと思います。つまり、プログラムを1CPUで実行するという環境でいかに速くするかという方法を知ってもらうことが意外と重要です。

四則演算の実行において、いかに絶え間なくデータがCPUに供給されるかということが問題となります。CPUでの1回の演算実行に比較して、主記憶からデータを取り出す時間は何十倍もかかります。そこで、よく参照されるデータを高速動作のCache（キャッシュ）というメモリにおいておきます。

Cacheは、IBM System 360 Model85 Ⅱに初めて登場したもので、L.A. Beladyが1966年にIBM System Journalで紹介しています。従って、結構古い技術なのです。当時の目的は、プログラムのlocality[注1)]という性質を利用すれば、プログラミング時に特に意識しなくても、よく使われそうなデータが平均的にCacheにのっていて、結果的に、実行速度が数倍程度速くなるということで、それ以後の科学技術用高速コンピュータには実装されていきました。

注1）次の2つの性質
（La）プログラムは同じ箇所を繰返し参照することが多い。
（Lb）プログラムは参照したすぐ近くを直ちに参照することが多い。

ところが、最近、Cacheの動作の仕組みを知って、CPUから参照されるデータをできるだけCacheにあるようにプログラムを工夫し、演算時間を速くしようということになってきました。そこで、Cacheの動作の仕組みを説明します。

2 Cacheの構成

主記憶のどこのデータを取ってくるかということは、プログラミング言語、例えばFortranでは変数名、配列名、配列要素で指定します。コンパイラで翻訳された結果は、バイト単位の2進数で場所を指定し、これをアドレスといいます。24ビットで表現すれば16MBまで、31ビットで表現すれば2GBまでとなります。最近では64ビット表現が標準となりつつあります。

*かなざわ　まさのり（京都大学　学術情報メディアセンター）

主記憶の一部を Cache に持ってくるには、適当な大きさの単位で扱います。単精度ならば 4 バイト、倍精度ならば 8 バイトで数を表現しています。さらに、例えば、１次元配列 A において、A(1)の次に A(2)を参照ことが多い（Lb に相当）ため、ある程度の大きさの塊（これをラインといいます）で扱うようにします。32 バイトや 64 バイトといった 2 の何乗かの値にします。

ここでは、HPC の採用している 64 バイト単位で説明します。また、アドレスは、簡単のため、31 ビットと仮定します。

HPC の Cache は図 1 に示すように主記憶に対応付けします。アドレスの上 25 ビットはライン番号を、下 6 ビットはライン内の場所を示します。さらに、対応付けを 2 次元にするために、25 ビットを 15 ビットと 10 ビットに分けます。10 ビットで横方向に 1024 セットを配置します。Cache の容量は 128KB ですから、128K÷1024÷64＝２となり、2 段重ねとなります。

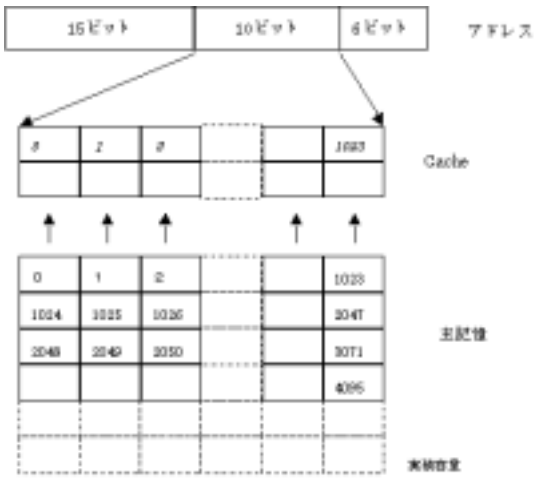

図 1 Cache の構成

Cache の構成は、64 バイトのラインを横に 1024 セット並べ、それが２ウェイある構成です。この構成が 2 つあり、片方は主記憶のアドレスの上位 15 ビットが格納されています。これを Diretory と呼びます。もう一方はそのデータそのもの、主記憶のコピーが入っています。

アドレスが与えられると、真中の 10 ビットからセット番号が確定します。その結果、2 つの Directory が候補となります。セット番号が 1024 の倍数であれば、Cache の一番左、0 の列に割り当てられます。２ウェイですからどちらか一方となります。

次に、Directory の中身とアドレスの上位 15 ビットと比較します。同じものがあれば、ヒットしたと言い、それに対応するラインに参照すべきデータがあります。なければ、ミスしたと言い、主記憶から対応するラインをとってきて、２つのうちから古い方のデータを追い出して入れ替えます。

3　プログラムの実行時間

プログラムの実行にかかる時間は、データが Cache にあるかどうかで大きく変わってきます。

ヒット率を h とすると、平均実行時間 T は、ｈが十分大きい（0.8 以上の）とき

　T　＝　h・Tc　＋（１－h）Tm

但し、Tc：Cache 上にある場合の演算時間

　　Tm：主記憶へのアクセス時間

と表せます。一般に、Tm は Tc の 10 数倍以上にもなります。20 倍とすると、ヒット率 95%で 1.95 倍、90%で 2.9 倍、80%で 4.8 倍というように、実行時間が大きく変わります。

4　データの参照方法とヒット率

Fortran で 2 次元以上の配列を扱うことはよくあります。例えば、

　Real*4　A(1024,1024)

と宣言した配列のデータの並びは、A(1,1)、A(2,1)、A(3,1)、、、A(1024,1)、A(1,2)、A(2,2)、、、A(2,1024)、A(3,1)、、、、となります。そして、各要素は 4 バイトの大きさとなります。

```
s＝ 0.0
Do  J = 1, 1024
  Do  I = 1, 1024
    s＝s＋A(I,J)
  Enddo
 Enddo
```

図２．　プログラム A(総和)

総和を求める計算を図２のようにすると、配列をアドレス順に参照することになるので、Lb の性質を活

かすことになり、一度4バイトを参照すると残りの60バイトは同じラインにあるため、Aのヒット率は60÷64＝93.75%となります。

```
Do  I = 1, 1024
  Do  J = 1, 1024
    s＝s＋A(I,J)
  Enddo
Enddo
```

図３．　プログラムB（総和）

Doループの順序を逆に図3のようにとすると、配列の参照は、4バイト×1024要素＝4096バイトごとの等間隔となります。その結果、A(1,1)とA(1,17)はセット番号０の位置に割り当てられ、A(1,33)を参照するときA(1,1)をCacheから追い出すことになります。また、Cacheに持ってきたラインの他のデータは追い出されるまでに参照されることはありません。従って、Aに関するヒット率は０％になります。

もし、1024を1025とすればどうなるでしょうか。４×1025＝4100 ごとの等間隔の参照となります。その結果、A(1,17)は、(17－1)×1025×４＝65600となり、64×1024と64異なるため、セット番号１の位置に割り当てられます。A(1,1024)でほぼ1ウェイ（64KB）分占有します。A(2,１)はA(1,1)と同じラインですので、ヒットすることになります。この結果ヒット率は前々回（93.75%）のときと殆ど同じになります。

2次元配列を扱うときは、参照の順序を考えるか、配列の大きさを２の何乗にぴたりと合わせないようにすれば、ヒット率がよくなることがあります。

次に、平均Aveと分散Devを求めてみます。

```
S = 0.0  ;  R = 0.0
Do  I = 1, 50000
  S = S + A(I)
Enddo
  Ave = S / 50000.0
Do  I = 1, 50000
  R = R + ( A(I) － Ave) **2
Enddo
  Dev = R / 50000.0
```

図４　プログラムC(平均と分散)

図4では、すべての変数は倍精度（8バイト）とします。A(I)を参照するとき1ラインに８要素入りますので、SとRを累積していくとき、Aに関するヒット率は7/8で87.5%となります。

そこで平均と分散を一度に(1 回のループで)計算できないかアルゴリズムを考え直すと、図５のようになります。

```
S = 0.0
R = 0.0
Do  I = 1, 50000
  S = S + A(I)
  R = R + A(I)*A(I)
Enddo
Ave = S / 50000.0
Dev = R / 50000.0 － Ave * Ave
```

図５．　プログラムD(平均と分散)

この場合、Rを累積するときA(I)は必ずヒットしますので、Aに関するヒット率は15/16で93.75%となります。

この他にも、いろいろな工夫をすれば、ヒット率が向上することが知られています。講習会や利用の手引き・広報で利用者の方々にお知らせします。

HPCでは、Cacheと主記憶との格差が余りにも大きすぎます。容量では128KBと512GB、速度比較では100倍程度になります。そこで、Cacheを2つのレベル設けています。今まで説明してきたのは、１次Cacheで、容量２MBの２次Cacheがあります。

2次Cacheの構成は、64バイト×8192セット×4ウェイとなっています。２次Cacheの動作は複雑で、ヒット率を推定することはかなり難しいと思われます。

1次Cacheは、命令用とデータ用の2つ（128KB×２）用意されていますが、2次Cacheは命令とデータの共用で、2MB 用意されています。ヒット率は、データの参照だけでなく、命令の参照も同時に考えなければならないので、かなり複雑になります。そこで、プログラムの動作を把握するために、ハードウェアモニタ情報（PA 情報）を収集するツールが用意されています。総合情報として、2次Cacheのミス率（１－ヒット率）、命令実行数、浮動小数点

演算命令効率などが測定できます。1次Cacheのミス率も測定することができます。

2次Cacheは2MBと容量が大きいので、小さなプログラムのときには、プログラムが殆ど2次Cache に乗ってしまうことがあります。その場合、乗り切らない場合と比較して、急激に実行時間が短縮されるといったことがあります。例えば、行列の積を求めるプログラムを fortran で作成し HPC より少しクロックの遅いマシンで通常のモードで（特に最適化機能をつかわないで）実行させてみました。400 行 400 列の行列の場合はすべての行列が 2 次Cacheにのることになり、750Mflopsとなりました。800行800列の場合は1つの行列すら乗り切らないので、230Mflopsとなりました。

さらに、今回は並列処理について説明をしていませんが、並列処理を行なうことで1CPU の扱うデータが分割され、小さくなり、2MBの2次Cacheにのってしまうということがあります。並列度と演算速度との関係をグラフで表すと、CPUが2、4、8、、、台と増加すると、演算速度が2、4、8、、、倍となるのが理想的な並列処理の効果と考えられますが、Cache の効果で、2 倍以上、4 倍以上、8 倍以上となることがあります。このような現象をスーパーリニアと呼んでいます。

５．仮想記憶と TLB

最近は、コンピュータがシリーズ化され、実装される主記憶容量が多様になってきました。その結果、実際に実行させるコンピュータ毎に、その主記憶の利用可能な領域を意識してプログラムすると大変な労力がいるので、仮想記憶方式がスーパーコンピュータと呼ばれるマシンでも採用されています。仮想記憶方式は、プログラムの大きさが実装された主記憶より膨大になるにつれて、自動的に、必要な部分を主記憶に、当面使われそうにない部分をディスクに置いておくということを考えた方式です。

仮想記憶装置では、主記憶もプログラムもページという固定した大きさで分割し、ページ単位で割り当てていく方式です。プログラム全体で割り当てと返却を繰り返すと、小さな空きが点々とできるというfragmentationが生じますが、仮想記憶ではこのようなfragmentationは回避できます。HPC では通常のプログラム部分はページの大きさは、8KBとなっています。

スーパーコンピュータでは仮想記憶方式を採用していますが、主記憶が十分用意されていますので、実質的には、ページイン/アウトというページを主記憶とディスクの間で入れ替えるという動作は通常生じません。ところが、論理アドレスから実アドレスへ変換という動作は必要です。

この変換動作のために、Cache の時と同様に変換用のハードウェアが用意されており、TLB（Translation-Lookaside Buffer）と呼ばれます。この TLB に見つからないときは、主記憶にあるページテーブルを参照しなければならず、Cache のミス以上のロスが発生します。8KB 単位の割り当てでは、TLB を余程大きくしないとミスが多発します。

大きなデータを扱う場合には、もっと大きな単位でということで、4MB の largepage を利用することができます。

翻訳時にパラメータで指定するだけですので、是非とも利用されるようお勧めします。

Fortran プログラムをコンパイルするときには、
自動最適化レベルを最高に　**−Kfast_GP2=3**
largepage に　**−Klargepage=2**
指定するのが標準と考えてください。また、64 ビットアドレスモードも標準と考えると、次のような形になります。ソースプログラムは prof.f にあると仮定します。

```
% frt pro.f −Kfast_GP2=3 −Klargepage=2 −KV9
```

６．終わりに

今回は、1CPU で基本となる最適化について、コンピュータの仕組みから物語的に説明しました。

補足　精度について

VPP800 では、単精度でも倍精度でも演算速度が変わらないことが多いので、つい倍精度を使いがちでした。HPC では、倍精度にするということは Cache の大きさが半分になるということです。ヒット率が半分になるかもしれません。十分に配慮が必要です。
単精度は、10 進 7 桁
倍精度は、10 進 15 桁
です。