

解説

数値計算におけるベンチマーク†



島崎 眞 昭†

1. はじめに

計算機アーキテクチャの多様化やコンパイラの最適化技術の進歩にともない、異なる計算機について、マシン・サイクルやメモリのアクセス・タイム/サイクル・タイムといったハードウェアのカタログ的データを比較しても、その結果から実際のプログラムを走らせたときの計算速度の比較結果を予測することが一般には困難になっている。したがって、計算速度といった性能を定量的に比較するため、計算機作業負荷を代表するような基準的プログラムによって計算速度を実測、比較するベンチマークテストがよく用いられる。数値計算を主体とする計算機の応用分野では、解析解の得られない問題を数値計算によって解こうとする場合が一般的であり、計算量は多く、計算機の計算速度の限界に挑むような問題も多い。したがって計算機の種々の性能項目の中で、特に計算速度に対する関心が強く、計算速度の比較を中心とした各種のベンチマークが使用されてきた。対象となる計算機のアーキテクチャはベンチマークの設計に大きく影響するので、本稿では、数値計算におけるベンチマークとして、ベクトル/並列計算機などのためのベンチマークとミニコンおよびワークステーションなどで用いられる汎用計算のベンチマークとにわけて解説する。

2. 数値計算におけるベンチマークの必要性和問題点

ベンチマークテストを行う目的として、(1)命令セット・アーキテクチャなどの設計の良否の判断、コンパイラの最適化技術の評価など計算機システムの個別の設計項目の研究・評価、(2)計算機利用者の立場での計算機システムの性能比較、(3)数値計算法の評価などが考えられる。(1)は計算機システムの設計者、

研究者により行われる。命令実行時間の重み付き平均値から、単位時間当たり実行される命令の平均個数、MIPS 値 (Millions Instructions Per Second) を算出ないし測定することなどが行われる。加重平均算出法として、Gibson MIX¹⁾ がよく知られているが、それ以外の種々の方法も使われている。命令パイプラインやメモリ・キャッシュの影響なども考慮する必要がある。IBM 370 シリーズを中心とするメインフレームの計算機については、近年そのアーキテクチャが相当安定しており、事務計算または科学技術計算というように、対象分野を限定すれば、一般のプログラムにおける命令使用頻度もある程度安定性がある。したがって MIPS 値による性能比較と、実プログラム走行による性能比較との相関が高く、計算機システムの総合評価を必要とする(2)の目的にもほぼ役立った。しかし、アーキテクチャが異なる場合、MIPS 値による比較は必ずしも有効ではない。さらに、ベクトル/並列計算機や RISC CPU を用いたワークステーションの出現により、これらに対しては、異なる性能評価が必要になっている。MIPS 値は計算機ハードウェアの基本的な性能指標の一つであるが、その問題点や測定用プログラムについては本稿では触れないこととする。ただし汎用計算に関するものとして4.でワークステーションなどで用いられる Whetstone や Dhrystone といったベンチマークについて述べる。(3)は構造解析、流体力学、原子力などの分野で、手法の性能比較を行う目的で標準的な例題により行われるが、本稿では触れないこととする。

ベンチマークテストについては、(1)ベンチマークプログラムの設計、(2)ベンチマークテストの実施、(3)結果の評価の段階がある。ベンチマークテストが計算機システムの性能評価、比較法として有効性もち、公平性を保つためには、各段階について、十分検討・準備する必要がある。ベンチマークテストは計算機システムに対する全作業負荷でテストするかわりに、小規模のテストプログラムの走行結果から性能を

† Benchmarks for Numerical Computation by Masaaki SHIMASAKI (Research and Development Division, Computer Center, Kyushu University).

†† 九州大学大型計算機センター研究開発部

判断するので、ベンチマークテストプログラムは対象とする計算機システムにおける作業負荷をよく代表するものでなければ意味がない。このため、対象となる計算機システムの作業負荷を調査・分析することが重要である。この場合、現状の作業負荷の調査のみならず、その計算機システムのライフサイクルを考慮したジョブの需要予測も折り込む必要があり、必ずしも容易なことではない。単一ないし少数の固定のプログラムによる処理が、その計算機システムの作業負荷の大半を占める場合、ベンチマークプログラムの設計は比較的容易と考えられる。一方対象計算機システムの利用分野が多岐にわたり、かつ計算手法やプログラムの研究開発が中心的な利用であって、プログラムのライフサイクルが短い場合、作業負荷を代表させるベンチマークプログラムの設計は困難となる。

テストの中心項目として、(1)CPU 時間を中心とする計算速度、(2)スループット、(3)レスポンス・タイム、(4)I/O 処理効率などが考えられ、どの項目を重視するかは、対象となる計算機システムの利用形態、目的に依存する。計算速度を計測する場合、単一CPU システムならば、CPU 時間から計算速度が算出できるが、複数 CPU からなる並列処理システムの場合、占有利用の状況でウォールクロックにより計算速度を算出することが行われる。レスポンス・タイムの計測が必要な場合、ソフトウェアによる端末シミュレータの利用が行われる。周辺機器を含むテストの場合、テスト実施環境の設定管理が必要となる。ベンチマークの結果をまとめるときには、ハードウェア、オペレーティング・システム、コンパイラ（名称と版番号）、使用オプションなどの環境の詳細の明示が重要である。

文献 2)は米国の政府機関が調達を行う際のベンチマーク実施に関するガイドラインをまとめている。

ベンチマーク実施の際の問題点の一つに、プログラムのチューニングの問題がある。ベンチマークテストにおいて、プログラムのチューニングを許容するか否か、許容する場合にプログラムの変更の許容範囲をどうするかが問題となる。計算機システムに関する学術的興味からベンチマークを行う場合は別であるが、計算機システムの更新に関係している場合、ソフトウェア資産の継承の問題に対する考慮が、ベンチマークプログラムのチューニングの問題に関係し得る。システムの移行に当たり、ソフトウェア資産の量、必要な移行作業量、移行期間、移行のために費やせる経費、マ

ンパワーなどが問題となる。(1)ソフトウェアの変更は事実上不可能に近い、(2)若干の変更には対応できる、(3)新しいアーキテクチャが実現された場合、最大限に活用するため、アルゴリズムの根本的な変更にとまなうプログラム開発にも対応できるといった立場があり得る。ベンチマークプログラムの変更についても、(1)基本的にプログラムの変更は許容しない、(2)若干のプログラムの変更（何%か）は許容する、(3)最大限の性能が発揮されるように変更制限を付けないといった立場があり得る。

ベンチマーク・テストの評価についても考慮を必要とする。複数のプログラムによる計測結果を一つの指標にまとめることを行う場合、特に注意が必要である。

n 個のベンチマーク・プログラム $P_1, \dots, P_n$ を用いるとする。プログラム P_i の計算速度は V_i で、プログラム P_i に代表される作業負荷の計算量は対象とする計算機作業負荷全体のうち F_i の割合を占めるとする。計算速度 $V_1, V_2, \dots, V_n$ は一般に異なる。特にベクトル/並列計算機では、そのばらつきは大きい。このような状況のもとで、総合評価をどのように行うかは問題である。

プログラム P_i の作業負荷の割合が F_i であれば、全作業負荷を処理するとき、計算機システムの実効的な計算速度 V は

$$V = 1 / \left(\sum_{i=1}^n F_i / V_i \right) \quad (1)$$

となる³⁾。ただし

$$\sum_{i=1}^n F_i = 1 \quad (2)$$

式(1)は計算速度の異なる演算器を組み合わせたときの実効速度を計算するアムダールの式に類似の形となっている。 $V_1, \dots, V_n$ のうち、たとえば V_j が他の V_i に比較して極端に値が小さい場合、 F_j も他の F_i に比較して相当小さくないかぎり、 V に対する V_j の影響が大変大きくなる。これは不適当であるような印象を与えやすいが、 F_i がプログラム P_i の全作業負荷に対する割合を正確に表していて、かつ個々のプログラムの速度でなく、計算機システムとしてのスループットを重視する場合には、 V は合理的な値を与える。 F_i が不明である場合、 $F_1 = F_2 = \dots = F_n = 1/n$ として、単純な調和平均をとる場合がある。この場合に、ほとんどの計算機で計算速度が大きくなりにくいプログラム V_j があると、 V はほとんど V_j により決

まって、計算機システムの他の種々の特徴の比較が困難になるという問題点がある。調和平均については、 $F_1=F_2=\dots=F_n=1/n$ という仮定のもとに意味があることに留意すべきである。

単一の指標を計算する目的で、 V_i の算術平均値を求める場合がある。計算機システムのスループットの観点からは、速度の算術平均値はほとんど意味がない。各 P_i がそれぞれ異なる応用プログラムである場合、速度の算術平均値は、個別の応用プログラムでの速度の期待値としての意味をもち得る。

3. ベクトル／並列計算機におけるベンチマーク

3.1 ベクトル／並列計算機におけるベンチマークの分類

ベクトル計算機や並列計算機では、計算機方式上の工夫により高速化を図っているため、その方式に適さない演算式およびプログラムでは、その高速性が発揮できない。すなわち、プログラムによる計算速度の違いが大きく、2.で述べた問題点に対する考慮が重要となる。したがって単一レベルでのベンチマークでは十分ではないので、(1)核ループレベルのプログラムによるベンチマーク、(2)ライブラリレベルのプログラムによるベンチマーク、(3)応用プログラムによるベンチマークが使用されている。ベクトル計算機ではベクトル処理能力とともにスカラ処理能力の評価も重要であるため、両方の処理モードでベンチマークを行う必要がある。並列処理方式の計算機については、単一CPUの場合と、複数CPUを使用したときの評価を、ともに行う必要がある。また最適化のためのコンパイラ・オプション（ベクトル計算機では、ベクトル化指示のオプション、並列計算機では、マルチタスキング指示のオプション）使用に関するルールを定めておく必要がある。

3.2 核ループレベルのプログラムによるベンチマーク

核ループレベルのプログラムは1ないし数行程度の文からなるループプログラムを用い、ハードウェアの基本性能やコンパイラの基本的なベクトル化、並列化機能を調査することを目的とする。対象となる文や演算は代入文、四則演算、総和演算、内積演算、一次巡回演算、多項式演算、初等関数計算、IF文などである。ベクトル計算機ではベクトル・レジスタと主記憶との間の転送もパイプライン的に行われるため、メモリ・

バンク衝突の性能への影響を調べる必要がある。アクセスする主記憶のアドレスに関し、連続アクセス、等アドレス間隔アクセス、間接指標アクセスの場合について調べる。ベクトル計算機では、半性能長（最大性能の半分の性能が発揮されるベクトル長）は機種により、また演算の種類により異なるため、ループの繰返し数は種類だけでなく、複数個について調査することが望ましい。

核ループレベルのプログラムの場合、ループ本体の実行のCPU時間が計時精度に比較して通常短く、計測対象ループをループで多数回実行して、CPU時間を測定し、ループ繰返し数で割って、対象ループの計算時間を求めることが一般的に行われる。このときコンパイラの最適化機能により、意図と異なる結果となることがあり注意が必要である。計測対象の内側ループが外側ループに依存しないことを最適化コンパイラが検出すると、内側ループ自体をループ不変式として、外側ループ外に移動し、一回しか実行しないことがあり得る。また並列化コンパイラでは、外側ループの繰返し実行を分割し複数のCPUに割り付け実行することがある。いずれの場合も計測精度向上のためのループ実行という目的からはずれてしまう。このような最適化を阻止する一つの方法として、図-1のようなダミーの手続き呼出しを挿入し、その分を補正する方法があり、用いられている。

核ループレベルのプログラムによるベンチマークは種々作成され、用いられている。よく使用されるものとしては、DFVLRカーネル⁴⁾やリバモア・ループ^{5),6)}がある。DFVLRカーネルはオランダで作成されたが、表-1にカーネルの内容を示す。リバモア・ループは、米国のリバモア研究所の応用プログラムか

```
CALL SECOND(T1)
DO 4 J=1,M
  CALL DUMMY
  DO 2 I=N1,N2,N3
    S: 計測対象文
2    CONTINUE
4    CONTINUE
  CALL SECOND(T2)
  DO 6 J=1,M
    CALL DUMMY
6    CONTINUE
  CALL SECOND(T3)
```

被計測ループの平均実行時間: $((T2-T1)-(T3-T2))/M$
ただし SECOND(T)は実行開始からのCPU時間を与えるとする。

図-1 ループの実行時間測定におけるコンパイラ最適化の悪影響を除去する方法

表-1 DFVLR カーネルの概要
(ベクトル長 $N=100$, $N=1000$)

ループ番号	ループの概要
1	$x+y$
2	$x*y$
3	x/y
4	$(x-y)*(x+y)$
5	内積
6	$0.5*(x/z+x*z)$
7	9 次の多項式
8	一階差分
9	二階差分
10	$y+\text{constant}*x$

表-2 リバモア 14 ループの概要

ループ番号	ループの概要
1	流体
2	MLR 内積
3	内積
4	帯型連立一次方程式
5	三項行列消去 (下三角)
6	三項行列消去 (上三角)
7	状態方程式
8	PDE 積分
9	予測子積分
10	予測子差分
11	総和 (一次和)
12	一階差分
13	二次元粒子推進
14	一次元粒子推進

ら抽出されたループプログラムで、基本演算のほか、三項方程式など若干高レベルのものも含まれている。14 個のループを含むリバモア 14 ループと、IF 文などを追加した 24 ループとがある。リバモア 14 ループの概要を表-2 に示す。これらのループについては、普通 MFLOPS 値の算術平均値と調和平均値を求める。リバモア 14 ループのうち、ループ(5)、(6)、(11)は一般にベクトル化されなかったが、HITAC S-820 では最適化技法が開発され⁷⁾、ベクトル化可能となり、算術平均値と調和平均値の水準が相当向上した。

3.3 ライブラリ・レベルのプログラムによるベンチマーク

科学技術計算において、基本的で使用頻度の高いサブルーチン・ライブラリ・レベルのプログラムが使用される。典型的なものとして、行列積演算、連立一次方程式の LU 分解、対称行列に対するコレスキー分解、固有値解析、高速フーリエ変換があげられる。このレベルのプログラムになると実用的なプログラムに近く、基本的なハードウェア性能、コンパイラの自動ベクトル化機能だけでなく、アルゴリズム、プログラムの組み方まで吟味する必要がある。そして、ベンチマークにおいて、プログラムのチューニングに対する考え方、規則を明確にしておく必要がある。上記の問題に対しては、一般に各計算機の製造会社から提供されるサブルーチンライブラリがあるから、それによる結果と比較するべきである。計算機製造会社から提供されるサブルーチンはその計算機の特徴を活用し、最大の性能が発揮できるようにプログラムが作成されているのが普通であり、ベンチマークによる結果との性能の差が大きければ原因を見極める必要がある。計算機製造会社のライブラリのソースプログラムは非公開の場合もあるが、少なくとも使用アルゴリズム、記述言語 (アセンブラか FORTRAN かなど) を確認する必要がある。このレベルのベンチマークの例として、LINPACK, NAS カーネルの例を説明する。

LINPACK は米国 Argonne 国立研究所で Dongarra ら⁸⁾により開発されたプログラムで、LU 分解に基づく連立一次方程式の解法プログラムである。Dongarra らの努力により、パーソナルコンピュータやワークステーションからスーパーコンピュータまで数多くの計算機に対して結果が公表されている⁹⁾。 n 元連立一次方程式の求解に必要な CPU 時間を計測し、実行浮動小数点演算個数を $(2/3)N^3 + 2N^2$ として、MFLOPS 値を計算する。LINPACK のオリジナル版は $n=100$ の場合を対象としている。係数行列を格納する配列宣言子の第一寸法が 200 の場合と 201 の場合について実測する。LINPACK のプログラムは BLAS (Basic Linear Algebra Subprograms¹⁰⁾) を用いて作成されている。BLAS は 1970 年代後半に提唱されたもので、線形演算に基本的なベクトル演算を定義し、行列積や連立一次方程式の解法プログラムなどの線形計算プログラムをモジュラ化し、信頼性とポータビリティを向上させることを目的としたものである。当時 CRAY-1 や CDC CYBER 205 の自動

ベクトル化機能は今日と比較し未熟で、BLAS で定義されたルーチンをアセンブラで作成し、全体の効率向上を図ることが行われた (coded BLAS). 文献 9) には coded BLAS によるベンチマークの結果も収録されている。欧米では BLAS を使用したソフトウェアの蓄積も行われ、LINPACK ベンチマークは、BLAS による線形計算ソフトウェアに対するよいベンチマークと考えられた。現代の自動ベクトル化コンパイラは、行列とベクトルとが関連する多重 DO ループに対し、ベクトル・レジスタを活用して、ロード/ストア命令を削減する最適化を行う。BLAS はベクトル演算が中心で、行列とベクトルの演算では、BLAS のサブルーチンが DO ループで繰り返し呼び出される形となり、先に述べた最適化の妨げとなる。すなわち行列とベクトルとの演算は BLAS を使用せず、多重 DO ループの形にしたほうが性能が高い。行列とベクトルの演算を二重 DO ループで構成し、外側ループにループアンローリングを適用すると、現行の自動ベクトル化コンパイラに対し効率的なプログラムが構成できる¹¹⁾。このサブルーチンを用い $n=300$ の場合のベンチマークの結果が文献 9) に収録されている。またチューニングに制限を付けないプログラムで、 $n=1000$ の場合の結果も文献 9) に収録されている。これらの結果はスーパーコンピュータに関しては、ベクトル演算の BLAS による結果と大きく異なっている。BLAS (レベル 1) の LINPACK によるベンチマークは、スーパーコンピュータの我が国における利用を前提とするベンチマークとしては不適当であると考えられる。その理由は、1) ベクトル演算中心の BLAS (レベル 1) は、自動ベクトル化コンパイラの最適化に妨げとなることと、2) 我が国では BLAS が線形計算ソフトウェアの分野で普及しておらず、LINPACK が代表的な作業負荷と考えられないからである。欧米では行列とベクトル、行列と行列とを基本単位としてルーチンを構成し粒度を大きくした Level 2 BLAS, Level 3 BLAS¹²⁾ の提案とそれに基づくソフトウェアの開発が進行している。文献 9) には上述のように種々の場合の結果が収録されているが、Level 1 BLAS による LINPACK の結果の表のみが、詳細な説明なしに引用されたり、表がいわば一人歩きしているような場合がある。その結果、誤解を招く場合も生じており、ベンチマーク結果の扱いに注意すべきことを示す事例といえる。

NAS カーネル^{13), 14)} は NASA Ames 研究所の典型

表-3 NAS カーネルベンチマークの内容

	ルーチン名	内 容
1.	MXM	外積法による行列積演算の実行
2.	CFFT 2 D	2 次元複素高速フーリエ変換の実行
3.	CHOLSKY	ベクトル版コレスキ分解の実行
4.	BTRIX	ベクトル版のブロック三重対角係数行列一次方程式の解法の実行
5.	GMTRY	渦解法のための行列の生成とガウス消去法の実行
6.	EMIT	ある種の境界条件に従う新しい渦の生成
7.	VPENTA	ベクトル処理を促進するような形で 3 個の 5 重対角行列の並列逆転

的なスーパーコンピュータの応用プログラムから核部分として抽出した 7 個のプログラムから構成されるベンチマークプログラムである。7 個のプログラムの概要は、表-3 に示すとおりである。プログラムのチューニングは 4 段階とし、チューニングのために変更したプログラムの行数が 1) 0 行、2) 20 行以下、3) 50 行以下、4) 制限なしの場合について、結果が示されている。

このプログラムでは多次元配列が使用されているが、その配列宣言子の第一添字の上限が 64 の整数倍になっているものが多い。このような場合、ベクトル計算機によっては強度のメモリバンク衝突を引き起こすことがある。たとえば、FACOM VP-200 の場合、DIMENSION A (128, 100) のように配列宣言された A(I, J) について、A(I, 1), A(I, 3), ... とアクセスするとメモリバンク衝突を引き起こすが、このメモリバンク衝突は配列宣言を DIMENSION A (129, 100) のように変更できればほぼ解消できる。この例は、ベクトル計算機でのベンチマークにおけるプログラムチューニングの影響が大きく、その取扱いが問題となることを示す具体例となっている。

3.4 応用プログラムによるベンチマーク

原子力、構造解析、流体力学など各種の分野のベンチマークがある。手法別のものとして、モンテカルロ法のプログラムなどがある。日米のスーパーコンピュータに適用し、著名となったベンチマークに流体力学分野の Mendez のベンチマーク¹⁵⁾がある。

4. ワークステーションなどで用いられる汎用計算のベンチマーク

4.1 Whetstone ベンチマーク

Whetstone ベンチマークは合成型 (synthetic) ベンチマークと呼ばれるもので、英国の H. J. Curnow と

表-4 Whetstone ベンチマーク・プログラムの概要

モジュール番号	内 容	重み (反復実行回数)
1	単純変数の加減算, 定数乗算	0
2	配列変数に対する加減算, 定数乗算	$12 \times i$
3	配列要素を実パラメータとする 手続き呼出し	$14 \times i$
4	条件分枝	$345 \times i$
5	省略	0
6	整数加減算, 乗算 単純変数, 配列要素への代入文	$210 \times i$
7	三角関数 ($\sin, \cos, \tan^{-1}$)	$32 \times i$
8	手続き呼出し	$899 \times i$
9	配列参照	$616 \times i$
10	整数加減算	0
11	標準関数 ($\sqrt{}, \exp, \ln$) 計算	$93 \times i$

$i=10$ のとき全重みは 10^6 Whetstone instructions に相当する。

B. A. Wichmann¹⁶⁾ により開発された。このベンチマーク・プログラムは高水準言語で記述された単一のプログラムであるが、内容的には表-4 に示す 11 個 (実質的には 8 個) のプログラム部分から合成されている。各プログラム部分はモジュールと呼ばれる。ベンチマーク・プログラムが科学技術計算作業負荷の代表となるように、科学技術計算プログラムに関する統計調査データに基づいて、各モジュールの重み (反復実行回数) を定めている。使用された統計データは、1970 年に NPL と Oxford 大学で収集された 949 個のプログラムに関するものである。これらのプログラムは Whetstone Algol 処理系¹⁷⁾ を使って解析された。Whetstone Algol 処理系は、トランスレータがソースプログラムを中間コードに翻訳し、それをインタプリタが解釈実行する。Whetstone Algol 処理系の命令実行頻度調査が行われ、その頻度分布に近くなるように、ベンチマークプログラムの各モジュールの重みが決定された。重みは繰り返し回数であるので、非負整数の条件下で曲線あてはめによる重み決定を行った結果、10 個のモジュールのうち、モジュール 1 と 10 の重みが 0 となった。その 2 個のモジュールのソースは、実行されないけれども、対応する目的コードを調べると興味深いであろうとの理由で、ベンチマークプログラム中に残されている。計算速度は Whetstone Algol 処理系の中間コードの命令 (仮想

マシン命令) を単位時間に何個実行することに相当する速度かという表現で、たとえば 100 kiro Whetstone instructions per second のように示す。このプログラムは最初 Algol 60 で示されたが、当初より他の言語への書き直しが容易なように設計されており、実際には FORTRAN または Pascal の版が使用されている。プログラムの特徴として次の点があげられる。

1) 種々の計算速度の計算機で使えるように、時間計測の精度に応じて、全反復回数がパラメータで変えられるように設計されている。2) 同一計算を単に反復するプログラムの場合、最適化コンパイラによっては、一回しか実行しない可能性がある。これを避けるため、各モジュールの計算結果が反復回数に依存し、指定回数実行しなければならないようなプログラムとなっている。3) 数値的に発散しない漸化式を用いて、反復回数が大きくなっても、オーバフローやアンダフローが発生しないように設計されている。

Whetstone ベンチマークは、ミニコンなどに対して、浮動小数点演算を中心とする科学技術計算能力を測るためのベンチマークとして広く使用されてきた。長年の使用経験から幾つかの問題が指摘され、それに対する開発者の見解が文献 18) にまとめられている。問題点の幾つかは次のとおりである。

プログラムのサイズが小さく、すべてキャッシュに収容される可能性があって、結果の性能値が大きく変わる可能性がある。これについてはプログラムの実行環境を制御するとともに、その環境の詳細を結果とともに明示すれば、無用の混乱は避けられるであろう。2 次元配列が使用されていないという指摘があるが、1 次元配列に限定した理由としてつぎの点があげられている。すなわち、1) ベンチマークの設計の基になったプログラムの調査では、配列の平均次元数は 1.2 であったことと、2) 2 次元配列のアクセスにおいては、strength reduction などの最適化が効果的で、処理系の最適化能力の影響が強くなりすぎる可能性があったことである。命令の使用頻度が現実の応用プログラムのそれに近い疑問であるとの指摘もある。設計の基礎となった調査は 1970 年に行われており、時代の経過により事情が変化している可能性があるが、正確には不明である。計算結果をチェックするプログラムがなく、プログラム書き換えの際エラーが混入しても見逃される可能性があるとの指摘があり、文献 18) には計算結果をチェックするプログラムが与えられた。

我が国でも Whetstone Algol 処理系は HITAC

5020 での稼働実績をもつが、現在ではなじみが薄く、Whetstone instructions per second という単位は現実感に乏しい。しかし特定の機械命令によらないこと、また計算機システムの性能比較においては絶対値そのものより、相対比較が重要なので、単位のことはいずれ問題とはされていないようである。

4.2 Dhrystone ベンチマーク

Whetstone ベンチマークが、浮動小数点演算を中心とした科学技術計算分野を対象としているのに対し、Dhrystone ベンチマーク^{19), 20)}は、システム・プログラムの分野を想定し、文字列処理、整数演算を重視して設計された合成型ベンチマークである。W(h)-etstone との対比で、D(h)rystone と名付けられた。CPU/コンパイラの性能評価を目的とし、ポインタ型やレコード構造などの豊富なデータ型が、ベンチマークプログラムに使われている。ベンチマーク設計の基礎データとして、表-5 に示す高水準プログラム言語の使用実態調査報告^{21)~36)}が使用された。調査データを考慮して、ベンチマークプログラムの文の種類、オペランドのタイプ、オペランドのローカリティの分布を決定している。文の分布についてみると、代入文が 53%、制御文が 32%、手続き呼出しおよび関数呼出しが 15% (そのうち関数呼出しは $X := F(\dots)$ の形で 5%) である。プログラムは最初 Ada 版¹⁹⁾が公表されたが、現在は C 言語の版²⁰⁾も広く使用されている。オプションによりレジスタ変数指定が可能で、使用され

表-5 Dhrystone ベンチマーク設計に使用されたプログラムの統計調査データ

Knuth's FORTRAN collection ²¹⁾
XPL Data ²²⁾
Zelkowitz's PL/I Data ²³⁾
Elshoff's PL/I Data ^{24), 25)}
Tanenbaum's SAL Programs ²⁶⁾
Amsterdam ALGOL 68 Programs ²⁷⁾
Pascal Compiler Statistics ²⁸⁾
Berkeley "Project Architectural Measurement"
University Wisconsin Pascal Data ²⁹⁾
Manchester Pascal Data ³⁰⁾
Mesa Statistics ³¹⁾
C Statistics of Bell Labs ³²⁾
Pascal Statistics at the University of Colorado ³³⁾
De Prycker's Pascal and ALGOL Statistics ³⁴⁾
Ada Application Study ³⁵⁾
iMAX 432 Data ³⁶⁾

たオプションと結果と一緒に印刷される。小規模のプログラムであり、キャッシュの影響が出る可能性があり、ベンチマークの実行者が実行環境、結果の評価を行う際にこの点を考慮することになっている。Dhrystone の速度の絶対値の物理的意味は明確でないが、計算機の性能の相対比較の目的でこのベンチマークは広く使用されている。

5. おわりに

ベンチマークによる比較は定量比較であるので、公平性、明確性の点で優れている。しかしベンチマークの設計者がよく強調するように、ベンチマークはある前提のもとに設計されており、そこからくる限界がある。ベンチマーク実施者の報告書には、それらの留意事項が記述されていても、数値結果のみが部分的に引用されたり、いわば一人歩きし誤った印象を与える危険もある。しかし十分な配慮を行ったうえでのベンチマークの実施とその結果の利用は必要かつ有効で、今後とも計算機システムの性能評価法として、広く使用されるであろう。

なお本稿で扱った、Livermore Loop, NAS Kernel, LINPACK, Mendez コード, Whetstone, Dhrystone のベンチマーク・プログラムは、Oak Ridge 国立研究所の netlib³⁷⁾、または NIST の nistlib のデータベースに収容されていて、電子メールによる情報サービスが行われている。

謝辞 貴重なコメントをいただいた査読者に謝意を表します。

参考文献

- 1) Siewiorek, D. P., Bell, C. G. and Newell, A.: Computer Structures: Principles and Examples, pp. 39-61, McGraw-Hill, New York (1981).
- 2) Guidelines for Benchmarking ADP Systems in the Competitive Procurement Environment, FIPS Pub. 42-1, NBS (1977).
- 3) Worlton, J.: Understanding Supercomputer Benchmarks, Datamation, Vol. 30, No. 14, pp. 121-130 (1984).
- 4) van Kats, J. M., van der Steen, A. J. and Llurba, R.: Experiences with the First Generation of Japanese Supercomputers: Fujitsu VP-200, Hitachi S 810/20 and NEC SX-2, Proceedings ICS '87, Vol. 1, pp. 159-167 (1987).
- 5) McMahon, F.: The Livermore Fortran Kernels: A Computer Test of the Numerical

* 1989 年 Dongarra 博士の転勤にともない、netlib の計算機は Argonne 国立研究所から、Oak Ridge 国立研究所に移った。

- Performance Range, Lawrence Livermore National Laboratory Report UCRL-53745 (1986).
- 6) 唐木幸比古: ベンチマーク・テストによるスーパーコンピュータの性能比較, スーパーコンピュータ製品・技術・応用, pp. 67-70 (1989).
 - 7) Tanaka, Y., Iwasawa, K., Gotoo, S. and Umetani, Y.: Compiling Techniques for First-order Linear Recurrences on a Vector Computer, Proceedings SUPERCOMPUTING 88, pp. 174-181 (1988).
 - 8) Dongarra, J. J., Bunch, J. R., Moler, C. B. and Stewart, G. W.: LINPACK Users' Guide, SIAM Pub. (1979).
 - 9) Dongarra, J. J.: Performance of Various Computers Using Standard Linear Equations Software in a Fortran Environment, Computer Architecture News, Vol. 16, pp. 47-69 (1988).
 - 10) Lawson, C., Hanson, R., Kincaid, D. and Krogh, K.: Basic Linear Algebra Subprograms for Fortran Usage, ACM Trans. Math. Software, Vol. 5, No. 3, pp. 308-371 (1979).
 - 11) Dongarra, J. J. and Eisenstat, S. C.: Squeezing the Most out of an Algorithm in Cray Fortran, ACM Trans. Math. Software, Vol. 10, No. 3, pp. 221-230 (1984).
 - 12) Dongarra, J. J., DuCroz, J., Duff, I. and Hammarling, S.: A Proposal for a Set of Level 3 Basic Linear Algebra Subprograms, Argonne National Laboratory Report, ANL-MCS-TM-88 (Apr. 1987).
 - 13) Bailey, D. H. and Bartpon, J. T.: The NAS Kernel Benchmark Program, NASA Technical Memorandum, 86711 (1985).
 - 14) Bailey, D. H.: NAS Kernel Benchmark Results, Proceedings of the First Int. Conf. on Supercomputing Systems: SCS 85, pp. 341-345 (1985).
 - 15) Lubeck, O., Moore, J. and Mendez, R.: A Benchmark Comparison of Three Supercomputers: Fujitsu VP-200, Hitachi S-810/20 and CRAY X-MP/2, Proceedings of the First Int. Conf. on Supercomputing Systems: SCS 85, pp. 320-329 (1985).
 - 16) Curnow, H. J. and Wichmann, B. A.: A Synthetic Benchmark, Computer Journal, Vol. 19, No. 1, pp. 43-49 (1976).
 - 17) Randell, B. and Russell, L. J.: ALGOL 60 Implementation, Academic Press, London (1964).
 - 18) Wichmann, B. A.: Validation Code for the Whetstone Benchmark, NPL Report DITC 107/88 (1988).
 - 19) Weicker, R. P.: DHRYSTONE: A Synthetic Systems Programming Benchmark, Comm. ACM, Vol. 27, No. 10, pp. 1013-1030 (1984).
 - 20) Weicker, R. P.: Dhrystone Benchmark: Rationale for Version 2 and Measurement Rules, SIGPLAN Notices, Vol. 23, No. 8, pp. 49-62 (1988).
 - 21) Knuth, D. E.: An Empirical Study of FORTRAN Programs, Softw. Pract. Exper. Vol. 1, pp. 105-133 (1971).
 - 22) Alexander, W. G. and Wortman, D. B.: Static and Dynamic Characteristics of XPL Programs, Computer Vol. 8, No. 11, pp. 41-46 (1975).
 - 23) Zelkowitz, M. V.: Automatic Program Analysis and Evaluation, In 2nd International Conference on Software Engineering, pp. 158-163 (1976).
 - 24) Elshoff, J. L.: An Analysis of Some Commercial PL/1 Programs, IEEE Trans. Softw. Eng., Vol. SE-2, No. 2, pp. 113-120 (1976).
 - 25) Elshoff, J. L.: The Influence of Structured Programming on PL/1 Program Profiles, IEEE Trans. Softw. Eng., Vol. SE-3, No. 5, pp. 364-368 (1977).
 - 26) Tanenbaum, A. S.: Implications of Structured Programming for Machine Architecture, Comm. ACM, Vol. 21, No. 3, pp. 237-246 (1978).
 - 27) Grune, D.: Some Statistics on ALGOL 68 Programs, SIGPLAN Notices, Vol. 14, No. 7, pp. 38-46 (1979).
 - 28) Shimasaki, M., Fukaya, S., Ikeda, K. and Kiyono, T.: An Analysis of Pascal Programs in Compiler Writing, Softw. Pract. Exper., Vol. 10, No. 2, pp. 149-157 (1980).
 - 29) Cook, R. P. and Lee, I.: A Contextual Analysis of Pascal Programs, Softw. Pract. Exper., Vol. 12, No. 2, pp. 195-203 (1982).
 - 30) Brookes, G. R., Wilson, I. R. and Addyman, A. M.: A Static Analysis of Pascal Program Structures, Softw. Pract. Exper., Vol. 12, No. 11, pp. 959-963 (1982).
 - 31) McDaniel, G.: An Analysis of a Mesa Instruction Set Using Dynamic Instruction Frequencies, In Symposium on Architectural Support for Programming Languages and Operating Systems, SIGPLAN Notices, Vol. 17, No. 4, pp. 167-176 (1982).
 - 32) Ditzel, D. R. and McLellan, H. R.: Register Allocation for Free: The C Machine Stack Cache, In Symposium on Architectural Support for Programming Languages and Operating Systems, SIGPLAN Notices, Vol. 17, No. 4, pp. 48-56 (1982).
 - 33) Carter, L. R.: An Analysis of Pascal Programs, UMI Research Press, Ann Arbor, Mich. (1982).
 - 34) De Prycker, M.: On the Development of a Measurement System for High Level Language Program Statistics, IEEE Trans. Comput., Vol. C-31, No. 9, pp. 883-891 (1982).
 - 35) Dobbs, P.: Ada Experience on the Ada Capability Study, Ada Letters, Vol. 2, No. 6, pp. 59-62 (1983).
 - 36) Zeigler, S. F. and Weicker, R. P.: Ada Language Statistics for the iMAX 432 Operating System, Ada Letters, Vol. 2, No. 6, pp. 63-67 (1983).
 - 37) Dongarra, J. J. and Grosse, E.: Distribution of Mathematical Software via Electronic Mail, Comm. ACM, Vol. 30, pp. 403-407 (1987).

(平成元年 10 月 12 日受付)