# テスト・プログラムによるシステム評価*

亀田壽夫**　恒川純吉***

## 1. はじめに

電子計算機システムの性能を知るために先ず行なわれるのは，適当なテスト・プログラムを走らせて，計算所要時間などのように簡単に得られるデータを検討することである．この方法は，システムの内部構造の詳細に精通することを要求せず，測定結果もすぐに利用できる手軽な方法である[23]．ここでは，そのような方法（benchmark test などと呼ばれるものを含む）について述べる．内部構造に立ち入った測定法（モニタリングなど）については本特集の別の解説があるので参照されたい．また，システムの使い易さ，明解さ，単純性，一般性などの質的側面も重要であり見落せないものであるが，ここでは，量的な測定を中心にして述べる[15]．

上のようなテストが特に問題になるのは，計算機を新たに導入する際の機種選択の場合であると思われるが，一般に，機種選択に用いたテストの結果を積極的に発表するようなことは行なわれないようである．従って，そのようなケースを沢山集めて眺めわたすことはできなかった．既に導入された計算機について，テスト・プログラムをかけて調べながら，機器構成などの最適化をはかることについてはいろいろ発表されている．ここでは，このような論文・解説等，公表された材料に基づいて検討することにする．

## 2. 総　説

システム測定法は，そのシステムをどう扱うかによって二つに分けられる．一つは，内部構造を見ず，システムを black box として扱う方向である．つまり，刺激（stimulus）としてテスト・プログラムをかけて，それに対する計算機システム応答（response）としての throughput, turn around time, 計算時間, accounting data[38] などを分析することによって，システムの性能を調べようとするものである．これは stimulus approach などと呼ばれる[17]．

もう一つは，システムの内部状態の記録を得て分析しようとするもので，analytic approach などと呼ばれる[17]．hardware monitoring や software monitoring などはこの方に含まれる*．これを interior measurement process と呼び，前者を，ユーザ・プログラムによるモニタリングなども含めて，exterior measurement process と呼ぶこともある[5]．以下では，この stimulus approach の方を扱うわけである．

この種のテストに関して，benchmark という言語がよく用いられる．bench mark の原義は，水準点のことであるが，計算機用語としては，計算機システムの処理速度を調べるための，標準的なテスト・プログラム（単数）をあらわすことが多い．計算機を比較する際に，いくつかの benchmarks（すなわち benchmark programs）を走らせて計算時間の測定が行なわれる．CPU の処理速度をあらわすものとしては，別に kernel programs というものが使われるが，benchmarks の方は入出力機構やシステム・プログラムも含んだシステム全体の性能を調べようとするものである．（kernel programs については本特集号の他の解説を参照されたい．）benchmark program と同じような状況で用いられるものに synthetic program がある[8]．Buchholz[3] の考えた synthetic program は，パラメータの値を変えることによって，入出力や計算の様相が変えられるようになっているテスト・プログラムである．benchmark と synthetic program の大きな違いの一つは，前者が，それ自体意味あるプログラムで結果も意味を持つのに対し，後者は，一般に，それ自体プログラムとして意味がなく，従って結果も意味を持たないという点にある[8]．synthetic のもとの意味は，「総合的」，「合成の」，「代用の」などであるが，synthetic program もパラメータによっていろ

---

* Performance Evaluation of Computer Systems by Running Test Programs, by Hisao Kameda (Univ. of Electro-Communications) and Junkichi Tsunekawa (The Institute of the Union of Japanese Scientists and Engineers, Inc.)

** 電気通信大学
*** 日本科学技術研修所

* 数理的取扱いを analytic と呼ぶこともある．

いろなものに作り変えられ(「合成の」),様々な目的にあわせられるものであり(「総合的」),プログラム自体としての意味はもたなくてもよく,実際のプログラムの代用として用いられる(「代用の」)ものと考えてよいであろう.計算結果が意味がないということからもわかるように,synthetic program は,CPU の性能というよりも,入出力を含んだシステム全体の性能や釣合を調べるのを目的とする.もちろん,benchmark にも I/O 機能の測定ということは含まれている[2].

以上のようなテスト・プログラムを走らせて,accounting data などの測定量を得るわけであるが,一般に量は何らかの比較のしるしであり,比較に用いられて有効性を発揮すると思われる.この場合の比較としては次のようなものが考えられる.

(1) 相異なった機種間の比較:機種選択の場合や,計算機の性能比較表を作って発表する時などに行なわれる.

(2) 同一の機種についての,機器構成 (configuration) の違いの間の比較:例えば,既に導入された計算機について,機器構成が最適化されるまで継続して行なわれる.

(3) 部分的に改良されたソフトウェアの間の比較:システム・プログラムが改良され最適化されるまで行なわれる.

(4) シミュレーション等により予測した性能値と,実際の性能との比較:シミュレーション等の有効性を検査するために必要である.

(5) 漠然と予期していた性能と,現実の性能との間の比較:これは,どの程度まで深く予想していたかによって違いがある.何か起るかを前もって予想していない場合には,いくつかテストをしてみて,意外なことがあるときに,システムの問題点が明らかにされたということになる.もっとくわしい予想をしている場合,例えば,特定の性能を満たすために適当な機器構成を考えたという場合などは,実際に測定して予期した性能が得られなければ,何か考え違いがあったはずだということになる.これは,プログラムの論理ミス (logical bug) と同じように,一種の bug と考えられ,'performance bug' などと呼ぶことがある[6].機器構成だけでなく,ハードウェア設計,オペレーティング・システムの構造などについても同様である.

上のいずれの場合でも予想以上に時間がかかってしまったことが,測定により明らかになったとき,何か問題があることがわかって,その追求が行なわれるということになる.

(6) 使用者プログラムのアルゴリズムや使用する言語の間の比較:数値計算法の開発などにおいてよく行なわれる[11),24)],更に,記号処理言語を選ぶ際の測定などもこの中に含まれるであろう.

(7) カタログ性能と実際の性能の比較:これは,カタログの読み誤りから生ずる誤解を防ぐためにも重要である.

テスト・プログラムとしては,テストの目的により,ほかにもいろいろなものがある.コンパイラのテストとしての文法テスト・プログラムや,エラー・メッセージ・テスト・プログラムなどは,正しさや質的性能を明らかにしようとするものである.

ところで,benchmark の効用の一つは,得られた結果の正しさも調べられることにある.結果の正しさの検査としては,関数の精度検定などもよく行なわれる.また,テスト・プログラムを工夫することにより,副作用を調べることによって,システム・プログラム,特にコンパイラの動作などを探り出すこともできる[32].

## 3. 具体的説明

### 3.1 benchmark

上で述べたように,benchmark とは,計算機システムの処理速度を比較するための標準的なテスト・プログラムのことである.このような benchmark として適当なものは,計算機の標準的な使い方をよく表わしたもので,かつ,問題の性質が一般の人にもすぐわかるようなものであろう.つまり,一般にいくつかの benchmark programs からなるセットでテストされることが多く,この benchmark でどれだけになったといういい方になるので,問題の性質が簡単に伝わる方が都合がよいのである.

発表されている例をいくつかあげる.もちろん,これらが代表的なものだというわけではない.

| | |
|---|---|
| (1) Dopping[7] | 重み |
| 1. Card to MT (Magnetic Tape) 変換 | 20% |
| 2. Sorting | 40% |
| 3. MT からの printout | 40% |
| 4. 入出力 | |
| 5. CPU と主記憶装置の性能テスト | |

これについては,5つの機種でテストした結果もあわせて発表されている.

(2) Williams 他[34] 重み

1. Marketing information retrieval and mailing list preparation 35%
2. Billing and sales analysis 30%
3. Scientific information retrieval 30%
4. Statistics 5%

（3） Gosden 他 (Auerbach, Inc.) [2),9),12)]

1. Generalized file processing problem
2. Random access file processing problem
3. Sorting
4. Matrix inversion
5. Generalized mathematical problem

1. の説明および結果の議論については，一般雑誌に紹介されている[9),12)]．

5. は，例えば，$w=\sqrt{\left[\sum_{i=1}^{5}\left(\left(\sum_{j=1}^{5}A_jX_i^j\right)\Big/Y_i\right)\right]}$．

（$X_i, Y_i$：入力）を計算する問題．

（4） PL/I の比較[26)]．

PL/I と COBOL, FORTRAN, JOVIAL などの言語との比較のために7つの問題を用意し，同じプログラマーに同じ問題を2通りの言語で書かせたものについて測定したものである．これも benchmark と呼ばれているのでここに記しておく．

1. GROSS PAYROLL (通常の給料計算)
2. ALOREP 2 (航空積荷問題)
3. MMI (Man/Machine Interaction)
4. TSME (Throughput Simulation in a Multi-programming Environment)
5. VIG (Vehicle Impact and Guidance)
6. SPP-A (Simulation Post-Processor-A)
7. SPP-B (Simulation Post-Processor-B)

1, 2 は COBOL との比較，3, 4, 5 は FORTRAN との比較，6, 7 は JOVIAL との比較に用いられた．

以上の他にも，TSS とバッチ処理システムとの比較に用いられたプログラム[27)]なども含めていろいろな例が考えられるが[11)]，ここにはその一部を示した．

種々の機械に，異なったいくつかの benchmark program をかけてみると，処理速度の順位は benchmark program 毎に変ることが多い．例えば，Joslin らのテストによると，4つのシステム (A, B, C, D) に4つの benchmark (W, X, Y, Z) を走らせた結果，実行時間は次のようになったという[14)]．

| 問題＼システム | A | B | C | D |
|---|---|---|---|---|
| W | 1.00 | 1.10 | 2.10 | 2.57 |
| X | 1.36 | 1.00 | 2.09 | 2.00 |
| Y | 1.00 | 1.32 | 2.72 | 1.35 |
| Z | 1.12 | 1.00 | 2.12 | 4.05 |

（数値は 実行時間の相対値）

これを見るとわかるように，(A, B), (C, D) の2グループ間の順位は変らないが，グループ内では，問題毎に順位が変っている．このような例は，ほかにもよく見られるものであり，このことは，benchmark の標準化・固定化を急ぐことが好ましくないこと（つまり，その問題のためだけに最適化が行なわれるのは望ましくないこと）を示している．また，5つぐらいの benchmark だけでは，十分なテストができないことも了解されるであろう[24)]．benchmark program の選定には注意が必要である．その結果に重みをかけて加えあわせ，処理速度として単一の数値を得て比較することは，更に慎重になされなければならない．

benchmark の選定や，重みの決定のためには，プログラム言語の各命令文の使用頻度の統計をとり，各使用者のプログラムの傾向を調べておくことが参考になるであろう[18)]．

特に留意すべきは，入出力機能のテストである．入出力性能は，機器構成に大きく依存するが，使用者プログラムのプログラム技法によっても大いに変る．少なくともプログラム技法の違うものをいろいろテストする必要があるが，このためには，異なったテスト・プログラムを数個用意するだけでは不十分である．大きな自由度，例えばパラメータで変更できる等のことが可能でなければならない．これを満足するのが，synthetic program である．

### 3.2 synthetic programs

Buchholz による synthetic programs は次のようなものである[3)]．master file に対し detail input を与え，match する master record に対し適当な処理 (compute kernel) を行なうことを繰り返し，結果として updated master file と detail output を得るという標準的な問題である．master record の数と detail record の数をカードからパラメータの形で与えることができ，また，compute kernel の計算量や所要記憶容量を recompile によって変えることで，計算や入出力などの重みの違いを表現できるというものである．

このような，テスト・プログラム作成の自動化の方向に沿ったものであるが，パラメータ化に止まらず，更に，入出力文の分布パターンも変えてテストすることの自動化も考えられる．つまり，入出力文の分布パ

ターンを適当な記号言語で表わし，その記号言語に対応するプログラムをプログラム・ジェネレータで生成し，できたプログラムを計算機にかけて測定しようとするものである．このようなものとして CONTEST がある[16]．CONTEST では，生成されたプログラムに，パラメータを通じて重みを変えることも行なわれる．このような柔軟さをねらった道具は，いろいろなプログラム技法に対するシステムの反応を調べ，システムの癖をさぐるためにも用いることもできよう．

自動化としては，コンパイラの文法テスト・プログラムの自動生成を行なう試みもある[10]．これは，与えられた言語の文法に合った任意のプログラムを生成して，そのプログラムをコンパイラにかけてみて正しさの検討を行なうというものである．

以上の方法は，自動化ということと，それに伴うものとしての生成されたプログラムの内容的無意味さとにおいて共通点がある．

### 3.3 その他のテスト・プログラム

プログラム自体には意味がなく，自動生成でないテスト・プログラムとしては，一般的な文法テスト・プログラムやエラー・メッセージ・テスト・プログラムがある[21]．更に，精度検定プログラムや，副作用を利用したコンパイラの分析プログラムなども興味あるテスト・プログラムであるといえよう[32]．コンパイラのオブジェクト・コード最適化の程度を調べるプログラム (FOPTEST) もある[31]．以上は，コンパイラの性能を調べるものである．

テスト・プログラムの効用の一つ，性能の意外性の発見あるいは顕在化として興味あるものに，エンプティ・ジョブ (STOP END ジョブ) によるテストがある[29]．これは，適当なジョブ制御文のほかに，例えば FORTRAN の場合，STOP 文と END 行だけしか含まないもので，実行可能なプログラムのうちで，最小限のステートメント数を持つものである．このような，ほとんど何も仕事をしないはずのプログラムを処理するのにも，最近の OS は数十秒の時間を要することをこのテストが明らかにしたのである．

テストしたい目的にあわせて，ほかにもいろいろなテスト・プログラムが考えられるであろう．

### 3.4 ジョブの流れ (job stream)

前節までに述べたのは，大体において，単独のプログラムについての説明であるが，現在の計算機では，多重プログラミングが行なわれるものが多く，また，入出力もディスクなどのバッファ記憶を経由するものが普通になっていると思われるので，単独なジョブだけでなく，いろいろなジョブの集まり (job-mix) を流してみて，全体的な性能を表わす量，例えば，単位時間当りの仕事の処理量である throughput や，計算結果がどの位早く手元にもどるかを示す turn around time 等を測定することが重要である．この場合に問題になるのは，どのようなプログラムを混ぜ合わせるかである．現実のジョブの流れを表わすには，実際に流れているジョブの流れのもとで測定すればよいのであろうが[37]，機器構成などの条件をいろいろ変えてテストしてみる場合には，再現性が問題となる．実際には，現実のジョブを適当に選んで，数十個のジョブからなるジョブの流れを作って，計算機に処理させて測定しながら，諸条件(機器構成，多重プログラミングの多重度，OS の構成など) のうちどれがよりよいかを調べることは，各所で行なわれている[13,19),22),28),30)]．

上述の synthetic program を用いて，ジョブの流れ (job stream) の組み合わせ方も，パラメータで自由に変えられるようにする試みもある[35]．つまり，適当なパラメータ値を与えて，任意の job-mix と同じ特性を持つ job stream を真似させようとするものである．

job stream を合成するためには計算機の使われ方を示すジョブ統計が参考になるであろう．幾つかの計算機センターではどのような種類のジョブがどのように使用されたかを示すジョブ統計をとっている[33]．

### 3.5 総合的なテストの一例

これまで述べたようないろいろな側面を，できるだけ多く，簡単にテストする目的でわれわれが考えたものについて，参考のために，紹介することにする．よいテストをするための一つのたたき台として用いていただければ幸いである．

(A) FORTRAN 処理システムの比較テスト[21]

1. STOP END job (5個)
2. CONTEST プログラム (入出力テスト)
3. エラー・メッセージ・テスト・プログラム
4. 文法テスト・プログラム
5. 各 FORTRAN 文の実行測定テスト
6. 最適化テスト (FOPTEST)
7. 計算処理速度テスト

   最短距離，行列の積，数値積分，多重回帰分析，行列の積については，5通りの解法を，次元を4通りにかえてテストした[36]．

更に，7の各プログラムについては，各ステートメントの出現頻度も調べた．

結果については，文献21を参照されたい．多くの機種について測定することにした（実際は，6メーカーの10機種）ので，システム全体の測定条件をそろえるのがむずかしく，job stream のテストはできなかった．

(B)　機種選択のためのテスト

1. STOP END job　　5個
2. 数ステートメント・ジョブ　　5個
3. 数十ステートメント・ジョブ　　20個
   （そのうち10個は文法的誤りをもつ．）
4. 倍長精度検定プログラム
5. エラー・メッセージ・テスト・プログラム
6. CPU 性能テスト
   行列の固有値，行列の積，積分，輸送問題

持ち運び易さのため，カード枚数に制限が生じ，このため，十分なテストでないことは覚悟して行なった．注目すべきことは，誤りを含んだ数十ステートメントのジョブの処理速度は，STOP END job の処理速度より速かったことである．当然のことであるが，正しいプログラムでも，自分のところでパンチしなおしたものには，いろいろトラブルが生じたのに対し，カードごと借りてきたものには，コピーをしたものでも，全く問題はなかった．

## 4. 検　討

性能をあらわす量として，overhead のような内部的な状態を示す量がよくあげられるが，最終的に重要なのは，Throughput, Turn around time, Availability 等の外部的な量であることは明らかである[1),4)]．従って，モニタリング等による内部状態の測定も重要であるが，テスト・プログラムによるシステム評価は，常に基本となるべきものとして必須のものである．

モニタリングなどの内部測定 (analytic measurement) とテスト・プログラムによる外部測定 (stimulus measurement) の違いは次のようにまとめられる[17)]．

| | 内部測定 (analytic) | テスト・プログラム (stimulus) |
|---|---|---|
| 開発コスト | 内部構造を調べなければならず，高い． | 標準的プログラムを用意するだけでよいので，低い． |
| 操作コスト | 測定のために多少の overhead が生じることがある． | テストのために専用の時間を設けなければならない． |
| 測定能力 | システムの動作についてくわしくデータがとれる． | Throughput, TAT (accounting analysis に基づく) など． |
| 結　果 | 事後に十分な検討が必要． | 結果は直ちに利用できる． |

実際にいくつかテストしてみて感じたことは，計時ルーチンに十分な精度が与えられていないシステムが多いことである．テストは，必ずしも機種間の比較だけでなく，使用者が自分のプログラム技法向上のために用いることがあるので，十分な精度が望ましい．経過時間 (elapsed time) や CPU time を別々に，少くとも msec 単位以下ではかれるようなものが欲しい．もちろん，時間の正確さもチェックしておかなければならない．また，ハードウェアの時計が変ったのに，計時ルーチンがもとのままになっているため，時計の目盛りが数倍も違っているというような誤りも稀にはあるので，注意を払う必要がある．

## 5. おわりに

ここでは，benchmark, synthetic program を中心に解説した．これらは，特に機種選択の際に問題とされる．しかし，これらのテストは，機種選択の場合だけでなく，システムが生きて更新を続ける限り，常にかわらず重要な反省材料を提供するものである．また，機種選択には，計量できないソフトウェアの性質などが重要なことはいうまでもない．

現時点では，テスト・プログラムとして決定版を作ることのできる段階でもないし，そもそも決定版などがありうるかどうかには疑問である．ただ注意すべきことは，見落しにより，測定すべき性能のうちどれかの一側面をテストするプログラムが欠けてしまうことであり，これを防ぐためには，見落しを防ぐための，包括的なリストの例をいろいろと検討しておくことが重要であると思われる．現実にテストするときには，手近にある問題を利用することが多いと思われるが，その場合にも，以上のような配慮が必要であると考えられる．

## 参考文献


1) Arbuckle, R. A.: "Computer analysis and throughput evaluation," Computers and Automation **15**, 1 (Jan. 1966), 12-15, 19.
2) Auerbach Info. Inc.: "Standard EDP Report," 1971.
3) Buchholz, W.: "A synthetic job for measuring system performance," IBM Systems J. **8**, 4 (1969), 309-318.

4) Calingaert, P.: "System performance evaluation: Survey and appraisal," Comm. ACM **10**, 1 (Jan. 1967), 12-18.
5) Campbell, D. J. & W. J. Heffner: "Measurement and analysis of large operating systems during system development," Proc. AFIPS 1968 FJCC, Pt. 1, 903-914.
6) Cantrell, H. N. & A. C. Ellison: "Multiprogramming system performance measurement and analysis," Proc. AFIPS 1968 SJCC, 213-221.
7) Dopping, O.: "Test problems used for evaluation of computers," BIT **2**, 4 (1962), 197-202.
8) Drummond, Jr., M. E.: "A perspective on system performance evaluation," IBM Systems J. **8**, 4 (1969), 252-263.
9) Gosden, J. A. & R. L. Sisson: "Standardized comparisons of computer performance," Proc. IFIP 62, 57-61.
10) Hanford, K. V.: "Automatic generation of test cases," IBM Systems J. **9**, 4 (1970), 242-257.
11) 橋本: "各種プログラム言語の比較," IBM Review, No. 36 (1971), 10-17.
12) Hillegass, J. R.: "Standardized benchmark problems measure computer performance," Computers and Automation **15**, 1 (Jan. 1966), 16-19.
13) 池田: "大型電子計算機システムの効率測定と循環待ち行列の理論による解析," 情報処理 **12**, 9 (1971), 568-576.
14) Joslin, E. O. & J. J. Aiken: "The validity of basing computer selections on benchmark results," Computers and Automation **15**, 1 (Jan. 1966), 22-23.
15) 亀田: "プログラムの評価に関する一考察," 第11回プログラミングシンポジウム報告集, (1970年1月) C 1-10.
16) —: "CONTEST 方式を中心としたオペレーティングシステムの性能測定."*
17) Karush, A. D.: "Two approaches for measuring the performances of time-sharing systems," Software Age (March, 1970) 10-13, (April, 1970) 26-27, 40, (May, 1970) 13-14.
18) Knuth, D. E.: "An empirical study of FORTRAN programs," SOFTWARE **1**, 2 (1971), 105-133.
19) 近藤: "名大センターにおけるジョブ処理効率調査."*
20) Lucas, Jr., H. C.: "Performance evaluation and monitoring," Computing Surveys **3**, 3 (1971), 79-91.
21) Moriguti, S., et al.: "A comparative evaluation of FORTRAN processing systems," Proc. 1st UJCC, (1972), 153-160.
22) Murphy, J. O. & R. M. Wade: "The IBM 360/195 in a world of mixed job streams," Datamation **16**, 4 (1970), 72-79.
23) 小田: "マルチプログラミング OS の評価."*
24) Opler, A.: "Measurement of software characteristics," Datamation **10**, 7 (July 1964), 27-30.
25) Pitts, G. N., et al.: "Minimizing computer cost for the solution of certain scientific problems," Proc. AFIPS 1970 FJCC, 515-518.
26) Rubey, R. J., et al.: "PL/I comparative evaluation," American Data Processing, Inc. (1969).
27) Schatzoff, M., et al.: "An experimental comparison of time sharing and batch processing," Comm. ACM **10**, 5 (May 1967), 261-265.
28) 高橋: "東北大学大型計算機センターにおける OS の効率測定," 第3回研究セミナー報告, 京大大型計算機センター (1971年12月), 64-70.
29) 田中・高橋: "エンプティジョブの処理件数," 数理科学 (1971年11月), 50-54.
30) 栃内, 他: "処理件数による OS の能力測定について," 第3回研究セミナー報告, 京大大型計算機センター (1971年12月), 48-63.
31) 恒川・杉本: "FORTRAN 処理系のオプティミゼイションの評価."*
32) 牛島: "スパイプログラム."*
33) Walter, E. S. & V. L. Wallace: "Further analysis of a computing center environment," Comm. ACM **10**, 5 (May 1967), 266-272.
34) Williams, Q. N., et al.: A methodology for computer selection studies," Computers and Automation **12**, 5 (May 1963), 18-23.
35) Wood, D. C. & E. H. Forman: "Throughput measurement using a synthetic job stream," Proc. AFIPS 1971 FJCC, 51-56.
36) 矢島・恒川: "行列の積に関するプログラム技法の比較," TOSBAC Report, No. 5 (1970年9月), 42-51.
37) 山縣: "大型計算機センターにおける OS 動作状況の推移について," 第3回研究セミナー報告, 京大大型機センター (1971年12月), 23-45.
38) 米田: "Accounting Information について."*

(昭和47年8月24日受付)

* システムの評価に関するシンポジウム報告集, 情報処理学会 (1972), (印刷中)