ハードウェアの評価法*

井　上　頼　昭**

1. まえがき

ここではシステムの評価法の一部として，ハードウェアの評価法について説明するのであるが，本題にはいるまえに，評価の全体的なことにちょっと触れておきたい．それから本題にはいり，主としてCPUの評価法を中心に述べ，最後にシステム的評価にも多少触れて，ハードウェアの評価法の解説としたい．

2. システムの評価とハードウェアの評価法

ほかでも色々と詳細に述べられていることとは思われるが，情報処理システムをどのような評価基準または項目で評価するかというと，最終的には使用者の立場から見た重要項目が基本と成るであろう．

つまり第一に，一定の費用の枠内で目的の仕事を十分処理できる機能と能力とを持っているかどうか．

さらに一般の場合その上に，導入後数年間の仕事の量の増大や質の変化に対して応じうる，拡張性や柔軟性を備えているかどうか．

そしてこれらは，信頼性とサービス体制をも含めての使い易さに裏付けられたものでなければならない．

さらにもう一つ，システム変更に際しての変換費用の少ないことも重要な条件である．

以上で大体いいつくしたと思われるが，実際にこれらを相当明確に評価するのは至難のわざであろう．

元来情報処理システムの評価は，精密に行なおうとするほど，その適用される仕事の形態はより詳細に決っていて，システム構成もそれに適したようにより精密に設計されていなければならない．したがってこのようにして得た評価は，一適用事例のみに対する評価に留まり，一般性のある評価とはなりえない．では一般的にというと，何となく漠然とした評価しか出てこないはずで，一種の不確定性原理のようなものである．それゆえ評価の精粗は，目的に応じて使い分けるべきである．

今まで述べてきた評価は，広くシステム構成からサポート体制までも含めて，ハードウェア，ソフトウェア両者一体となった，システムに対する評価であって，これが最終的に使用者に対して重要な意味を持つものなのである．したがって単なる一要素に過ぎないハードウェアのみに対する評価などは，使用者にとってはどうでも良いことであるが，そういってしまっては本文が始まらないので，一つの補助的目安の意味もかねて，主として基本的なソフトウェアの面から見た目で，直接的なハードウェアの評価法について，歴史的なものも振返りながら眺めて行きたいと思う．

3. CPUの評価法

ハードウェアといっても色々な要素から構成されているが，システムとしてまとまった評価は，少なくとも一般事務作業のバッチ処理用という位の適用分野が定まらないと，評価が無理なのと，個々の評価の積み重ねがある程度ないとできないので，後回しにする．

そこでまず，ハードウェアの中心をなすCPUに対する評価法を取上げよう．過去においては，CPU性能がその電子計算機全体を代表するほど重要であった．そしてその評価は，初期といえども素朴ななりに面倒なことであったが，最近のように，先行制御，並列演算方式，パイプライン方式，キャッシュメモリ方式等々，種々の高性能化手段が考案される一方，ソフトウェアも発達してくると，評価はますます厄介になってくる．それでは歴史的に順を追って説明して行こう．

3.1 加算時間による評価法

これはごく初期の単純な命令しかなく，加算命令が時間のかかる代表的命令であった頃に性能比較に用いられた．併し現在では，一般的には全く使用されない．

すなわち最近のCPUの命令構成からいって，加算命令一つでは，とてもそのCPUの命令群を代表し得ないからである．

3.2 命令ミックスによる評価法

---

* Evaluating methods for computer hardware, by Yoriaki Inoue (Fujitsu Limited, Computer Sys. Dev. Dept.)

** 富士通（株）開発部

これはある分野の平均的ジョブにおける命令の実行頻度の分布から，一命令当りの平均的実行時間を求めこれを評価値とするものである．

科学技術計算の実行に対するものには有名な GIBSON ミックス[1]があり，一般事務計算用にも幾つかのコマーシャルミックスがある．

一般に命令ミックス T_M は次の式で表わされる．

$$T_M=\sum_{n}^{i=1} f_i \cdot t_i$$

ここで t_i は命令 t の実行時間，f_i は命令 i の実行頻度，ただし $\sum_{n}^{i=1} f_i=1$.

表1，表2にその命令頻度の具体例を示す．

表1 The Gibson Mix

		%
1.	Load and Stare	31.2
2.	Fixed point Add and Subtract	6.1
3.	Compare	3.8
4.	Branch	16.6
5.	Floatihg Add and Subtract	6.9
6.	Floating Multiply	3.8
7.	Floating Divide	1.5
8.	Fixd point Multiply	0.6
9.	Fixd point Divide	0.2
10.	Shifting	4.4
11.	Logical, And, Or, etc.	1.6
12.	Instructions Not Using Registers	5.3
13.	Indexing (pseudo instruction)	18.0

表2 コマーシャル・ミックス

		%
①Decimal Add,	5 Digits	9
Compare	3 charcters	24
Move	10 chracters	25
Branch		31
Edit	11 Digitts	4
I/O		7
		%
②Compare	1 charcter	9
〃	2 charcters	5
〃	3	7
〃	6	1
〃	10	1
〃	12	3
Move	1 character	1
〃	10 Caracters	1
〃	60 〃	2
Branch	success	15
〃	not success	13
Add.	3 charcters	1
Indexing		40

このように命令頻度といっても，一般には，ある代表的な CPU で統計をとったものを基準として，他の機種にも当てはめて行くのであるから，命令セットの良く似たものでないと，良い比較はできない．

また，各々の機種でそれぞれに頻度分布をとったにしても，この方法はあくまで命令の平均実行時間を表わすものであって，同一の仕事量に対する総命令数は考慮にはいっていない．つまり複合命令や，語長やレジスタ数なども含めての命令セットの効率が良くて処理に要する命令数が減少しても，この方法では全然評価されないのが大きな欠点である．

3.3 サンプル・プログラム法

この方法は，あるプログラム全体か，あるいはその中の重要部分かを，評価対象の CPU の機械語によりコーディングし，それとハードウェア・マニュアルによる命令実行時間とから，そのプログラムを実行するのに要する時間を算出する．そしてその実行時間を評価値とする方法である．

ここで使われるサンプル・プログラムは，科学計算の例では，逆行列，ベクトル積，多元連立方程式，ヤコビアン，各種関数サブルーチンなどのようなものがある．

また事務用の場合，COBOL で良く使用される，実行ステートメントを統計的に選択して，桁数その他も統計的に定め，各々につきコーディングして実行時間を算出し，それに頻度率をかけて総和を出し，COBOL 1ステートメント当りの平均実行時間を求めると，相当実際的な基礎的評価値を得ることができる．

この方法は CPU のレジスタ数や命令セットの有効性，その他の特別な機能などもほとんどすべて織りこむことができるはずである．したがって命令ミックス法よりも一段とすぐれた方法といえる．

併しこの方法の欠点は，コーディングする人のコーディング技倆に左右される要素が大きいことである．つまり色々な機種に対して，各々の CPU の特色を良く生かし，しかも同程度に技巧を使用しなければ，ハードウェアの正しい評価ができず，コーディング技巧の巧拙が，評価を狂わせてしまう恐れが出てくることである．

また，少いサンプル・プログラムで比較する時，適当な重みづけが困難であることや，サンプル・プログラムが大きくなると作業が大変になるなどの欠点もある．

3.4 複合評価法

これは今まで述べた3.2項と3.3項との複合で割合実際的なものが考えられる．例えば，筆者等の行なった次のようなものも，この分類に属する一方法と思

う[3].

3.4.1 科学技術計算用複合評価法

これについては実行プログラムが変化に富んでいるので，モデルの設定が難かしい．そこで余り深追いせずに，命令ミックスに，サンプル・プログラムの結果からえられたステップ数による補正を加えたものを評価値とした．

E_{SH}：科学技術計算処理の時の実効的1ステップ実行時間．

T_C：コンパイル時の命令ミックス．

T_L：結合編集時の命令ミックス．

T_E：実行時の命令ミックス（≒ GIBSON ミックス）．

W_C：コンパイルの比率
W_L：結合編集の比率
W_E：実行の比率
$\}$ $(W_C+W_L+W_E=1)$.

S_C：コンパイルに対するサンプル・プログラムのステップ数による補正係数 $\left(\frac{\text{目的機種のステップ数}}{\text{標準機種のステップ数}}\right)$.

S_L：結合編集に対するサンプル・プログラムのステップ数による補正係数．

S_E：実行に対するサンプル・プログラムのステップ数による補正係数．

$$E_{SH}=W_C\cdot S_C\cdot T_C+W_L\cdot S_L\cdot T_L+W_E\cdot S_E\cdot T_E.$$

このようにして出した評価値は，大型機種の方では実測値と比較的良く合致するが（表3参照），小型機種の方では，小記憶容量のためのソフトウェアの作り方の違いによる影響が大きくて，実測値が相当悪く出ることが多い．

なお実測値との比較は一般に比率でしか行えない．

表3　科学技術計算用複合評価値比較例

	A	B	C	D
実測値比	1.0	0.78	0.82	0.53
E_{SH} 率の比率	1.0	0.72	0.80	0.53
GIBSON ミックス比率	1.0	1.04	1.07	0.69

3.4.2 事務処理計算用複合評価法

これについては仕事が比較的似たようなものが多いので，統計的にモデル・ジョブを定めた（図1参照）．

まず COBOL 1 ステートメント当りの実行時間を算出する．それには COBOL の実行ステートメントの出現頻度分布を統計的に調べ，重みづけを出す．そして各ステートメントをハンドコンパイルしてコード化し，それと各命令の実行時間とから，各ステートメントの実行時間を求め，それぞれに重みを掛け合せた後，全部を加え合せて算出する．

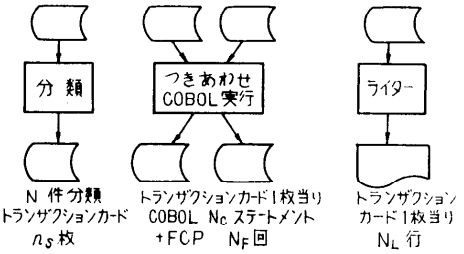

図1　事務用計算用複合評価値算出用モデル・ジョブ

$$T_C=\sum_{i=1}^{n}\cdot W_{Si}\cdot T_{Si}.$$

つぎに制御プログラムの時間の目安を出すために，制御プログラム用命令ミックスに，制御プログラムで良く使う手法を10個近くコーディングしてその各ステップ数に重みづけして加え合せ，それで先の命令ミックスを補正してそれの何倍かで FCP などの時間の目安とする．元来制御プログラムは，その機能の相違によって非常に大きさの変って来るものであるから，実際的にはその辺も考慮して倍数を定める必要があろう．

これらの単位を積上げて，モデルジョブのつき合わせ（COBOL 実行）や，リスト（ライター）の CPU 時間が出せる．なおモデル・ジョブの時間の表わし方は，トランザクション・カード1枚当りの平均処理時間で出している．

分類に関しては，方式や Way 数などは適当に決め，件数を始めとする各種パラメータは統計的に定めた．そして処理の部分は，重要部をサンプル・プログラムで求め，他の所は命令ミックスで求め両者の和で算出した．I/O 制御に関しては FCP の算出で述べたと同じ方法．そして N 件分類での CPU 時間を出して，これをカード枚数 n_S で割って，カード1枚当りの処理時間としている．

分類の CPU 時間

$$T_S=\frac{\text{N件分類 CPU 時間}}{\text{トランザクションカード枚数 } n_S}.$$

つき合わせの CPU 時間　$T_C=N_C\cdot T_{C1}+N_F\cdot T_{FCP}$.

リストの CPU 時間　$T_L=N_L(T_{LOC}+T'_{FCP}+T_W)$.

このようにして，基礎的な評価値が出たわけで，これらをもとに，後はシミュレーションをやるなり，待ち合わせ理論によるなりして，システムの評価も一応できるわけである．

実測との比較では，COBOL 1 ステートメント当

りの平均実行時間は，ソフトウェアの作りの違う小型機を除いて，非常に良く合致している．併し当然のことながら制御プログラムは大小に関係なく一率の倍数でやったので，実測とは相当違うものが出て来ている．システムを考えるのではなく，ハードウェア単体の比較評価としてはこれでも良いのである．

3.5 高性能化特殊機構付 CPU の評価法

これまで述べてきた色々な方法において，命令の実行時間は，種々な条件を十分考慮した実測に依るのが一番確実であるが，一般にはハードウェア・マニュアルに書かれた命令実行時間を使用することになる．

そこで注意しなければならないのは，マニュアルに書いてあるのは，その命令の実行だけに着目して書いてあるから，実際の総合結果とは異なってくるということである．例えば，単純なシステムで入出力チャネルが同時に働けば，記憶装置に対するアクセスにぶつかり合いが起り，チャネルのサイクルスチーム分だけ実効的速度は遅くなるはずである．

このようなことは，最近のように高性能化のための特殊機構が色々考え出されてくると，ますます複雑な条件として実効時間に大きな影響を与えてくる．

3.5.1 先行制御について

思いつくものから上げて行くと，まず先行制御の場合，記憶装置へのアクセス頻度が高まり一般にある確率でぶつかり合いが生じ実効的速度はおちる．記憶装置側はぶつかり合いをなるべく減らすよう，インターリーブやバンク分けを行なっているので，それの条件によっても異ってくる．インターリーブのウェイ数が変えられるものではウェイ数の多い方が待ちの確率は小さくなるし，データ領域と命令領域とを別にバンクにとるような配慮をソフトウェア的に行なえば，命令とデータのぶつかり合いはなくなる．

また条件分岐のような時，普通条件成立の側を先取りしてあるので，不成立だと遅くなるというようなことも出てくる．インデックス修飾なども，インデックスレジスタの書替えが何ステップ前で行われたかによって実行時間が変って来る．それから割込みがあると先取りが中断されて損するので遅くなる．

3.5.2 キャッシュ (Cache)・メモリ方式について[2)]

つぎにキャッシュ・メモリ方式の場合には，前述のような若干の補正ではすまなくなってくる．つまりキャッシュ上に必要情報がない場合，アクセスタイムが普通1桁位多くなるから，キャッシュ上にない確率によって実効速度は大変に変動する．プログラムの性質にも依るが，適当に設計されたキャッシュなら一般の科学技術計算の場合キャッシュ上にない確率は約5％程度なので，実効速度の低下は2～3割位で済んでいる．

3.5.3 ページ式メモリについて

つぎにこれは，むしろ記憶装置に属するかもしれないが，ページ式アドレス変換機構がある．これはアソシエーティブ・メモリの量やプログラムの性質にもよるが，実効時間で2～3％から5％位までの損失があるとされている．

3.5.4 ファームウェアなどについて

最後に，最近マイクロプログラミングの発達により中間言語的なものが直接実行可能なような，従来の命令セットより相当高度な機能を持つもの，つまりローカルにファームウェアと呼ばれていたようなものが，そろそろ実用に成りかけて来た．こうなって来るとこのレベルの中間のものを，ハードウェアというべきか，ソフトウェアというべきか，ちょっと判断に迷うところであるが，ともかくハードウェアの評価法は，手法分類的には大差なくとも，内容的には大分感じが変って来るものと思われる．

4. CPU 以外のハードウェアの評価法

CPU 以外は，いわゆるカタログ性能でほとんど評価が可能であるが，特に気のついた点についてざっと述べて見る．

4.1 記憶装置について

記憶装置については，CPU と密接な関係にあり，大部分は CPU の評価に含まれてしまうので容量以外はすでに大体述べてしまったことに成る．残ったものとしては，ページ式メモリで，ダイナミックなアドレス変換が容易なので，多重プログラムの時領域管理が能率的に行え，一般に 15％位の能率向上があるといわれている．またページ式メモリではヴァーチャル（仮想）メモリとして実際の記憶の大きさよりも大きなアドレスを持つプログラムを扱うことができる．ヴァーチャル・メモリ方式は，プログラムの性質にもよるが，一般には論理空間に比べて実空間が小さいほど能率が悪く，その比の数乗に逆比例するような感じである．この辺の評価は，シミュレーションか実測に頼る外なさそうである．

記憶容量は，CPU 速度と I/O 速度から CPU 占有率が定まり，これにより最大多重度が出てくるから，それだけのジョブを同時に十分収容しうる容量が

あればよい訳である．オンライン関係では，ラインのバッファが多くいるので中小型で主記憶容量リミットとなることが多かった．

4.1 チャネルについて

チャネルの能力は先ずその転送速度によって代表される．特にそのファイルの転送度は年々上昇して来ており，システム能力に大きな比重を占めるので重要である．

次にチャネルの利用率を上げ待ちを減らす方法として，フローティングチャネルとブロックマルチプレクサチャネルが考案されているが，これらはシステム構成に応じてそれなりに評価されるべきである．

4.3 外部記憶装置（ファイル）について

この能力については容量差があまりなければアクセスタイムとデータ転送速度とが決め手であることは今さらいうまでもない．

従来この分野は磁気回転体が主流を占めて来たが，近い将来には静止型のものが出現する可能性がでてきたようだ．そうなるとシステム構成なども大分変って来るかも知れない．

また現在でも自動倉庫と完全自動ヴォリュームローディングとを組合せたものが出現しても良さそうに思われる．

4.4 周辺機器について

入力については，最近キィ・ツウ・カセットとかキィ・ツウ・ディスクとかいったようなものが実用化され始めたのと，OCR とか OMR も大分使われて来たことなどから，広く総合的に評価を考える必要があろう．またこの時会話的に扱える端末も非常に有力な入力装置であることを忘れないように．

出力については用紙かけかえなどのハンドリングタイムの評価を忘れないようにすること．これからはラインプリンタを高速化するとともに，もっと扱い易いものとすべきで，この辺も完全自動化と行きたいところだ．

その他図形入出力装置や計測入出力などあるが，まだ比較的特殊用途という段階なので省略する．

4.5 回線について

回線については端末の評価と通信制御装置の評価とに分れる．端末については，現在のところ汎用の一般タイプライタ型端末，あるいは簡易グラフィックも含めたキャラクタ・ディスプレイ端末以外は比較的，専門的用途のものが多いので省略する．汎用端末の評価はいわゆるカタログ性能以外は信頼性保守性に気をつければ十分であろう．いま汎用端末に望まれているものは低価格につきるのではなかろうか．

通信制御装置に対する評価は，大雑把にはその会計の通信可能量と装置の受渡しのレベル，つまり文字かブロックか，電文かというようなレベルで，通信制御処理装置内でどれだけ伝送特有の処理をしてしまって，きれいなデータとして，CPU との受渡しが行なえるかということを表現したもの．そのような2点でとらえれば大体の評価はできたわけで，システムの大まかな評価にも大体使える．

4.6 IOP について

最近入出力はその制御やアクセスメソッドも含めて専用のプロセッサ (IOP) で処理させようという働きが起り始めているが，この場合は今までのべた CPU やファームウェアやチャネル等の評価法の内から適当なものを組み合わせて使うことになろう．

5. ハードウェア・システムに対する評価法[4),5)]

ハードウェアといってもシステムとなると相当ソフトウェア的要素がはいりこんでくるので，純然たるハードウェアに対する評価とはいえないかも知れないが，一応ハード的と思われることに限って述べて見よう．

一般にシステムの評価には，解析的なモデルによる方法[4),5)]とシミュレーション・モデルによる方法とがあるが，その他にも特定機種を基本にして評価の考え方をまとめて数式にし，ハードウェアの単体性能を代入して行なえばハードウェアのシステム評価値が出る[6)]というようなものも考えられている．何れにしろシステムとなると，処理対象と処理形態とが大まかにでも定まらないと評価は難しい．

科学用システムは，大学の計算センターのようなところや企業や研究所の計算センターのようなところがあるが各々性格を異にするようで，ちょっと共通には論じられないので省略するが，TSS やリモートバッチあるいはセルフサービス方式の比重が増大して来ているのは共通のようだ．

一般事務用バッチ処理では，CPU の項でも述べたようにある程度統計的なモデルジョブが作れるので，そのジョブが幾つ多重プログラムとして処理できるかを求め，それからスループットを算出して，あるハードウェア・システムの潜在最大能力として評価ができる．ただしこれは，その計算機の格を示すようなもの

で，ソフトウェアや運用面を加えた実際的なものとは相当くい違ったものとなろう．併しハードウェア評価としてはそれで十分と考える．

まず3.4.2項の複合評価法あたりでCPUの処理時間を算出し，次に入出力装置の処要時間を出す．そしてそれらをもとに，CPUがN_P個チャネルがN_{CH}個の二段循環待行列と考えて解析的にといて能力を求めるか，シミュレーションを行なって求めるかする．この時，能力の限界を定めるものがCPU速度か，主記憶容量か，チャネル数かということやプログラムの多重度もわかる．なおこの時1ジョブに1ラインプリンタと考えると良くラインプリンタ・リミットのようなことになり面白くないので，ライターにラインプリンタをすきなだけつけられることとし，ラインプリンタ・リミットにはならないように考えた方がよいと思う．なお主記憶の多重プログラムに対する多重度限界を出すには，1ジョブの必要主記憶容量と制御プログラムの主記憶上の占有容量とが必要で，これも大体システム規模に応じて統計的に定めればよい．

6. むすび

以上ハードウェアの評価法について，重要な点は一通り述べたつもりであるが，測定法に触れなかったのと手本に適当なデータが少く，話が具体的に欠けるきらいがあったことをお詫びしておく．なお評価の仕事に多少関係するようになってから，特に痛感するのは，今後の計算機はRAS機能と一緒に，もっと評価のための測定ということをハードウェア，ソフトウェア一体となって真剣に考えるべきであるということである．

参考文献

1) 石田晴久：ギブソン・ミックスの起源について，情報処理，Vol. 13, No. 5, '72, pp. 333-334.
2) 飯塚肇：キャッシュ・メモリシステム，情報処理，Vol. 13, No. 7, '72, pp. 467-473.
3) 井上，吉田：ハードウェアの評価に対する一つの試み，情報処理学会72年夏のシンポジウム，システムの評価報告集．
4) D. P. Gaver, Jr.: Probability Models for Multiprogramming Computer Systems, J. ACM, Vol. 14, No. 3, 1967, pp. 423-438.
5) F. Hanssman, W. Kistler, H. Schulz: Modeling for Computing Center, Planning, IBM SYS. J., No. 3, 1971, pp. 305-324.
6) K. E. Knight: Changes in Computer Performance, DATAMATION. Vol. 12, No. 9, 1966, pp. 40-54.
7) H. C. Lucas Jr.: Performance Evaluation and Monitoring, Computing Surveys, Vol. 3, No. 3, 1971.
8) R. R. Johnson: Needed: A Measure for Measure, DATAMATION, Vol. 16, No. 12, 1970, pp. 22-30.
9) G. Estrin, R. R. Muntz, R. C. Uzgalis: Modeling, Measurement and Power, SJCC, 1972.

(昭和47年8月16日受付)