ソフトウェアの型によるコンピュータ・パフォーマンス・データの図形的並びに統計的表現

磯　田　和　男*

（昭和50年9月8日受理）

Graphic and Statistical Presentation of Computer Performance Data Classified by Software Type

KAZUO ISODA

Abstract:—This paper purports to show that pattern of performance data varies in different software types.

We intended not to use special devices such as hardware monitor or software monitor, for these devices did not pay off and might disturb normal job processing on our small scale computer. Therefore, we could help using accounting routine output containing insufficient information.

CPU time, elapsed time, number of disk (C0) accessing, number of disk (C1) accessing and number of printed lines were available data items. Job step types were classified into FORTRAN compiler, ALGOL compiler, PL/I compiler, assembler, linkage editor and other execution step. Systematic sampling was applied to a set of accounting data from job streams in single job processing mode, and then sample means of data items of each job step type and job were computed. In addition, mean of cpu time/mean of (elapsed time−cpu time) was computed as a factor of cpu utilization of each job step type. This factor sharply reflects cpu utiliztion status.

For each job step, ratios of each mean value to corresponding one in job data and the factor mentioned above were selected as performance indicators to draw Kiviat-graph-like figures. Our figures have distinct meaning from Kiviat's one, but are useful for presenting pattern differences.

Finally, for the same purpose, multiple regression analysis was applied, where elapsed time was selected as a dependent variable.

1. まえがき

コンピュータシステム性能評価は現在コンピュータ・サイエンスの重要な一分野としての地位を確立している。一般論に関しては，すでに多くの論文，報告，図書等が刊行されている。（たとえば〔1〕，〔3〕，〔4〕，〔9〕.）最近特に話題となっているのは，キビアット・グラフ〔7〕，〔8〕，統計的手法の利用〔2〕である。本稿では，特にソフトウェアの質的相違による性能の差をコンピュータ・リソースの使用状態によって表現しようと試みた。このためキビアット・グラフに類似した表示と，回帰分析とを用いた。ジョブ全体についての回帰分析の応用に対しては〔5〕，〔6〕ですでに報告されている。

また従来用いられたCPU利用率の代りに，より感度のよい量として，（CPU時間の平均値）／（非CPU時間の平均値）を新しく導入した。ここで平均値とはジョブステップごとに測定された量の標本平均を意味する。

*茨城大学工学部情報工学科（日立市中成沢町）

2. データ

よい評価を得るためには，多重処理の効果，CPU・チャネルのオーバーラップについて，ハードウェアモニタやソフトウェアモニタを用いてデータを収集するのが本筋である。しかしながらハードウェアモニタは現在の所コスト的に利用不可能であり，ソフトウェアモニタは，メーカ側がOSの細部に関しては非公開の立場をとっているため作製が難しい上に，設置されているＨＩＴＡＣ８３５０がスモールスケールなので，ソフトウェアモニタの組みこみにより測定対象の正常なふるまいが攪乱される。

結果として，ジョブの開始・終了，ジョブステップ終了に際してSYSLOGファイルに書きこまれるロギング情報を利用することとした。この情報はかなり項目が制限されていて，チャネル・タイムやオーバーラップ関係のデータはふくまれていない欠点はあるけれども，平常のジョブ処理の状態をそのまま反映していること，ハードウェアモニタやソフトウェアモニタでは測定できないワークロードに関するデータもとれること，データ収集の容易なこと等の利点がある。さらに過去の全情報を磁気テープファイルに保存しておくことも容易にできるので種々の処理法をくりかえして行なうことも可能である。なおこのファイルの利用法として課金法の根拠づけをあげることができる。課金法もシステム性能評価の応用として研究対象になっていることを附言しておく。

さて今回の報告では，単一ジョブ処理の状態で，コア帯在時間，CPU時間，ディスク（２スピンドル）のアクセス回数，リスト量をデータ項目とし，単一処理状態の期間から系統的サンプリングにより標本を得た。但し入力リーダ及び出力ライタは多重的に働いているが，現在使っているOSではロギング情報が出力されていないので，今回はその影響を考慮に入れられなかった。

3. グラフ表示

分類対象となるソフトウェアに対し次のように番号をつける。

1 ……FORTRAN コンパイラ
2 ……ALGOL コンパイラ
3 ……PL/I コンパイラ
4 ……アセンブラ
5 ……リンケージ・エディタ
6 ……その他の実行プログラム
7 ……ジョブ全体

次に標本値から型ごとに以下に示す統計量を計算する。但し「バー」は標本平均を，Sは標準偏差を，添字ｉは型を示すものとする。計算結果は表１に示してある。

CPU時間 …… $\bar{t}_{cpu\cdot i}$, $S_{cpu\cdot i}$
コア帯在時間 …… $\bar{t}_{E\cdot i}$, $S_{E\cdot i}$
非CPU時間 …… $\bar{t}_{Ncpu\cdot i}=\overline{(t_{E\cdot i}-t_{cpu\cdot i})}$
リスト量（行数） …… $\bar{n}_{L\cdot i}$, $S_{L\cdot i}$
ディスクＣ アクセス回数 …… $\bar{n}_{C0\cdot i}$ $S_{C0\cdot i}$
ディスクC_1アクセス回数 …… $\bar{n}_{C0\cdot i}$ $S_{C1\cdot i}$

グラフ表示を行なうため，各型ｉに対し円を作り半径を１とする。円周を６等分し，各分点を通る半径を引く。便宜上時件風に１時方向，３時方向，５時方向，７時方向，９時方向，１１時方向とよぶことにする。ｉ＝７に対してはグラフは不要である。続いて以下に示す量を計算し，半径上に打点を行ない，隣接点と結んで多角形を作る。

1 時方向の半径…… $\bar{n}_{C0\cdot i}/\bar{n}_{C0\,i\cdot 7}$
3 〃 …… $\bar{n}_{C1\cdot i}/\bar{n}_{C1\,i\cdot 7}$
5 〃 …… $\bar{n}_{L\cdot i}/\bar{n}_{L\cdot 7}$
7 〃 …… $\bar{t}_{E\cdot i}/\bar{n}_{E\cdot 7}$
9 〃 …… $\bar{t}_{cpu\cdot i}/\bar{t}_{cpu\cdot i}$
11 〃 …… $\bar{t}_{cpu\cdot i}/\bar{t}_{Ncpu\cdot i}$

キビアットグラフのもとの形では，CPUアイドル，CPU時間，スーパーバイザ状態時間，プロブレム状態時間，CPU－チャネル・オーバーラップ，チャネル・アイドル中のCPU時間，CPU－チャネル・オーバーラップ（前と同じもの），チャネル時間の各量を８方向の半径上にパーセントに換算し，多角形を作る。現在のロギング情報にふくまれていない量が多いので，原形と著者の作ったものでは多少性格を異にしているが，次節に示すように，著者のものでも十分目的を達することができた。国産のコンピュータでかなり大きなシステムとして知られているものでもチャネル関係のデータが十分ロギングされていないものがあるが，もともとOSの設計時に十分顧慮されていないと，後からこの情報はとりにくい。システム性能評価の基礎データがとりやすいということは，以後の開発にも好影響をおよぼすはずである。

4. グラフによる評価

結果として図1が得られる。キビアットグラフの特長の一つとして直観的にシステム性能が把握できるという点があげられているが，この図からもこのことが確かめられる。

グラフの右半分がI／O関係となっている。ディスクC_0，C_1の使用目的について説明しておくことにする。

C_0……システム・レジデンス・ファイル
　　プログラム・レジデンス・ファイル
　　マクロ・ライブラリ・ファイル
　　オブジェクト・モジュール・ライブラリ・ファイル
　　エラー・ロギング・ファイル
　　ロールアウト・ファイル
　　システム入力ファイル
　　ロギング・ファイル
　　ワーク・ファイル1，2
C_1……システム出力ファイル
　　ワーク・ファイル3
　　システム・カタログ・ファイル

予算の関係で十分な数のスピンドル（ディスクモジュール）が装着できないことが根本的なネックになっているし，C_0，C_1への割り当ても適切でないかもしれない。しかしFORTRANとPL／Iとのコンパイラ，リンケージ・エディタではI／Oネックになっていることは一目瞭然であろう。特にPL／Iはかなりひどい状態に見える。I／O関係の改善だけではおそらく駄目に思われる。これに対してALGOLコンパイラはかなりよく出来ているようである。このコンパイラとアセンブラが似たパタンを示しているのも面白い。もちろんこのことはハードウェアのモデル（HITAC8350）とその構成，使用しているOS（EDOS）に依存しているわけであるが，この条件下での各ソフトウェアの性能比較にもなっている。他の機種や異なるOSで同じような実験を行えば性能評価としても興味がある。

新しく導入した量（$\bar{t}_{cpu\cdot i}$／$\bar{t}_{Ncpu\cdot i}$）を仮にCPU総利用因数と名づけることにする。この効果はまえがきにも述べたようにCPUの利用状態を敏感に反映しているのでCPUバウンドか否かが直観的に把握できる。利用状態が悪いということはすぐにシステムに悪い評価を下すということにはつながらないが，リソースを遊ばせておくことはもったいないわけである。この点は多重処理とか，CPU－CPU，CPU－チャネル，チャネル－チャネルのオーバーラップとかで解決はできる。しかしその基礎として単一処理での状態をよく解析しておくことは必要である。

図1ではCPUバウンドか否かが非常にはっきりしている。実行プログラム（型6）でも大へんよくCPUが働いているように見えるが，これは実行プログラムの特性がかたよっているからで，ソート・マージ等のI／Oバウンドのプログラムが行なわれていないからにすぎない。

ともあれ，ユーザ側としては機種の決定，構成の改善，主として使用する言語等に対してよい資料であり，メーカ側にとっても同様であるだけではなく，ある意味で今後の開発の方向への指針となることが期待できる。

5. 回帰分析による検討

上述したことがらを定量的にしらべるため回帰分析を行なうことにする。この方法は最近ようやく重視されはじめた。（一般論については〔2〕参照）。以下の方法については〔6〕でジョブ全体についてだけ行なわれた。

まず従属変数として$x_5 = t_{E\cdot i}$をとり，独立変数を次のように定める。

$$x_1 = t_{epu\cdot i}$$
$$x_2 = n_{C0\cdot i}$$
$$x_3 = n_{C1\cdot i}$$
$$x_4 = n_{L\cdot i}$$

x_5との偏相関係数が有意であるものを表2に示す。グラフ表示でかなり大きな値を示すものとほぼ一致している。PL／Iの場合，ディスクC_1のアクセスが甚しく多いにも変わらず表に出て来ないのはプログラムの大きさ，構造とあまり関係なく行われるC_1のアクセス回数が大きくかつ変動が少ないことを意味すると思われる。リンケージ・エディタの場合でも状況が似ている。図ではC_0アクセスが大きく影響するように見えるが，偏相関係数は有意でない。C_1アクセスの方が95％有意である。このことはディスク上のファイル割り当てが関係している。C_0のアクセス回数が大きいのは，このボリュームにオブジェクト・モジュール・ライブラリ・ファイルがあるためで，しかもリンケージのためのファイル・サーチの量はプログラムの大きさにかかわらずほぼ一定である。一方同じボリューム上にユーザ・プログラムから生成され

たオブジェクト・モジュールがのっているが，ライブラリ・サーチ量にくらべるとかなり小さい。C_1はリンケージのワーク・ファイルをふくんでいるが，このアクセス量はプログラムの大きさ，構造にかなり依存している。

回帰分析の実施に当り，まず全独立変数を考慮に入れて行ない，偏相関係数の小さいものを順次消除して行き以下に示すような最終結果を得た。ここでは途中結果および分散分析表を省略したが，係数は99％有意であった。なお式中変数名から添字 i を省略してある。

FORTRAN コンパイラ

$$t_E = 14.9 + 0.494\, t_{cpu} + 0.077\, n_{C1} + 0.027\, n_L \quad (R^2 = 77\,\%)$$

ALGOL コンパイラ

$$t_E = 11.1 + 1.09\, t_{cpu} \quad (R^2 = 75\,\%)$$

PL／I コンパイラ

$$t_E = 61.4 + 2.79\, t_{cpu} \quad (R^2 = 73\,\%)$$

アセンブラ

$$t_E = 8.03 + 1.03\, t_{cpu} + 0.045\, n_L \quad (R^2 = 98\,\%)$$

リンケージ・エディタ

$$t_E = 20.0 + 0.0404\, n_{C1} \quad (R^2 = 69\,\%)$$

その他の実行プログラム

$$t_E = 4.28 + 1.00\, t_{cpu} + 0.085\, n_L \quad (R^2 = 98\,\%)$$

6. 結　　論

もとのキビアット・グラフと意味は異なるけれども，ソフトウェアの型による性能の差を得ることができた。また回帰分析により実際スループットに影響ある項目の効果を定量的にあらわすことも成功している。今後の問題としてI／O関係の時間測定を小型コンピュータの制約のもとでどのように行なうかということと多重状態をどう処理するかとがあげられる。また適当なFOM(Figure of merit)の設定も重要であろう。

表　1　標本平均と標準偏差
(Sample Means and Standard Deviations)

FORTRAN

		m	s
t_{CPU}	(秒)	5.375	4.275
t_{NCPU}	(〃)	36.98	19.52
t_E	(〃)	42.35	22.45
n_{C0}	(回)	42.19	31.30
n_{C1}	(〃)	281.7	195.1
n_L	(行)	115.4	137.2

ALGOL

		m	s
t_{CPU}	(秒)	8.500	4.407
t_{NCPU}	(〃)	11.86	2.748
t_E	(〃)	20.36	5.198
n_{C0}	(回)	23.28	17.35
n_{C1}	(〃)	38.36	26.05
n_L	(行)	112.2	37.97

PL/I

		m	s
t_{CPU}	(秒)	28.04	16.30
t_{NCPU}	(〃)	111.5	43.70
t_E	(〃)	139.5	55.91
n_{C0}	(回)	21.64	13.88
n_{C1}	(〃)	661.2	270.1
n_L	(行)	252.2	137.4

ASSEMBLER

		m	s
t_{CPU}	(秒)	20.05	29.97
t_{NCPU}	(〃)	15.77	9.839
t_E	(〃)	35.82	33.86
n_{C0}	(回)	100.4	206.6
n_{C1}	(〃)	150.6	171.9
n_L	(行)	157.2	178.2

LINKAGE EDITOR

		m	s
t_{CPU}	(秒)	3.911	1.119
t_{NCPU}	(〃)	21.82	7.323
t_E	(〃)	25.73	7.923
n_{C0}	(回)	282.4	73.92
n_{C1}	(〃)	116.9	124.4
n_L	(行)	10.17	9.376

その他の実行プログラム
(Other steps)

		m	s
t_{CPU}	(秒)	31.28	82.39
t_{NCPU}	(〃)	7.183	15.69
t_E	(〃)	38.46	85.80
n_{C0}	(回)	9.935	67.31
n_{C1}	(〃)	23.67	136.0
n_L	(行)	342.8	1013.

表 1 (続)

JOB		m	s
t_{CPU}	(秒)	36.27	75.71
t_{NCPU}	(〃)	54.81	44.38
t_E	(〃)	91.08	94.30
n_{C0}	(回)	257.5	177.6
n_{C1}	(〃)	249.7	303.8
n_L	(行)	412.4	945.2

表 2 偏相関係数 (Partial Correlation Coefficients)

FORTRAN	$r_{51.234} = 0.7059^{**}$	t_{CPU}
	$r_{53.124} = 0.8117^{**}$	n_{C1}
	$r_{54.123} = 0.3766^{*}$	n_L
ALGOL	$r_{51.234} = 0.7043^{**}$	t_{CPU}
PL/I	$r_{51.234} = 0.4001^{**}$	t_{CPU}
ASSEMBLER	$r_{51.234} = 0.4083^{*}$	t_{CPU}
	$r_{54.123} = 0.5372^{*}$	n_L
LINKAGE EDITOR	$r_{53.124} = 0.3855^{*}$	n_{C1}
その他の実行プログラム (Other steps)	$r_{51.234} = 0.9862^{**}$	t_{CPU}
	$r_{54.123} = 0.5332^{**}$	n_L

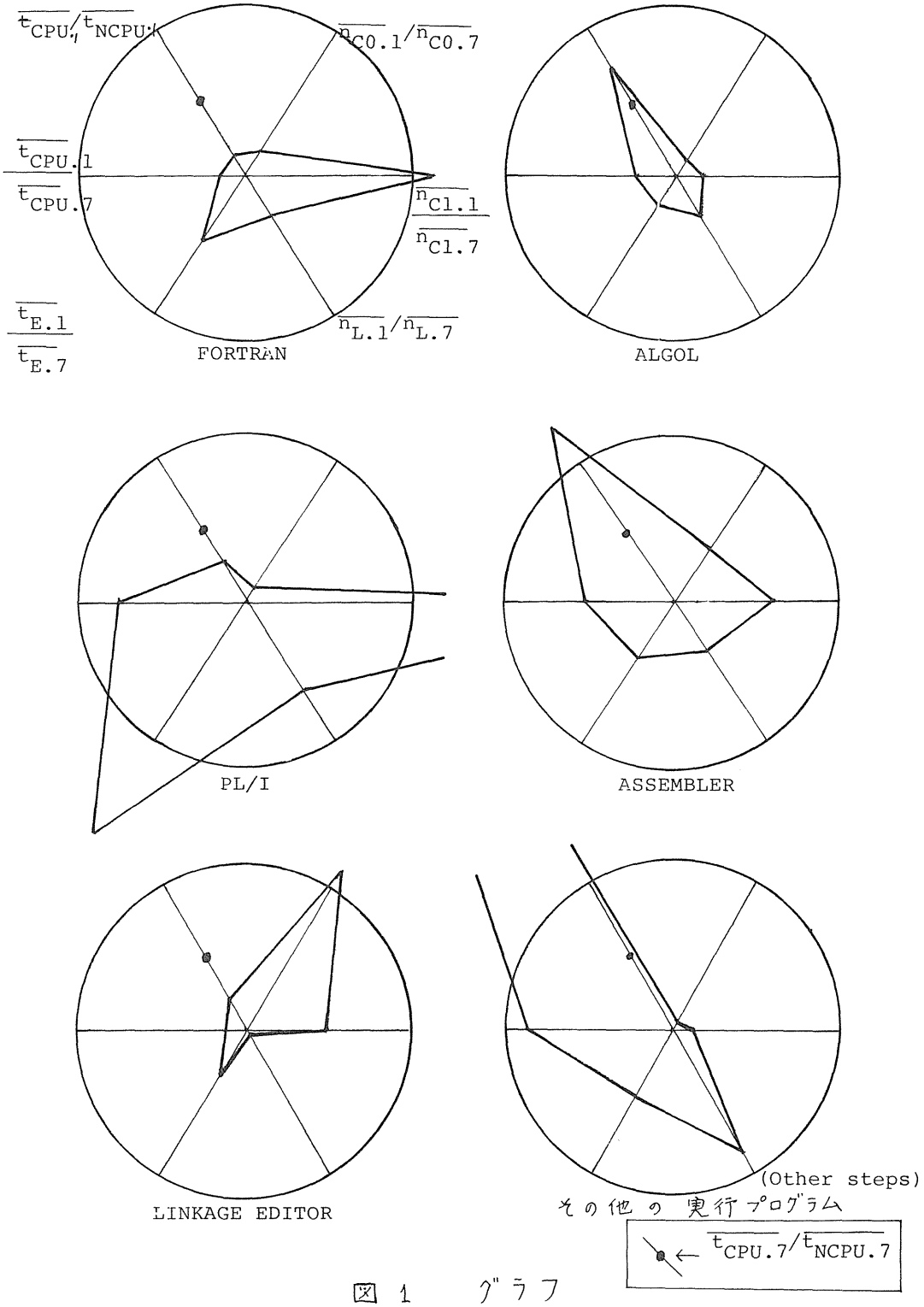

図 1　グラフ
(Graphs)

参 考 文 献

1) Drumond, M. E.：Evaluation and Measurement Techniques for Digital Computer Systems, Prentice Hall (1973)

2) Freiberger, W. (ed.)：Statistical Computer Performance Evaluation, Academic Press (1972)

3) 萩原宏：計算機システムの評価について，情報処理 13, 11 (1972), 740－5.

4) Hellerman, H. H. and Conway, T. F.：Computer System Performance, Mcgraw－Hill (1975)

5) 石黒美佐子，斎藤直之，磯田和男：実測およびアカウンティングから得られたデータの統計解析によるシステム性能と計算センタ効率の解析，第13回情報処理学会大会(1972年12月)

6) 磯田和男：ユーザの立場からみた評価と統計的方法，システム評価シンポジゥム報告集(1972年6月)，30－32

7) Kolence, K.W.：The Software Empiricist, Performance Evaluation Review 2, 2 (June 1973),31–36

8) Kolence, K.W. and Kiviat, Ph.J.：Software Unit Profiles and Kiviat Figures, ibid. 2, 3 (Sept 1973).

9) 日本情報処理開発センタ：システム評価方式の調査，同センタ資料 47－R.004 (1973年3月)