28. プログラムの性質の実験的分析

電気通信大学 計算機科学科

深津　貞雄・南澤　英雄・亀田　壽夫

はじめに

仮想記憶の概念が商用の中型・大型計算機に導入されて以来、プログラムの性質に対する研究が一層さかんになってきた。効率的な計算機システムを設計するために、プログラムの性質を研究し、把握しておくことは重要なことである。我々は、プログラムの性質として、以下に述べるように instruction mix、CPU使用特性、lifetime 関数（主記憶使用特性を示す）の三性質をとりあげ、新たに作成したプログラムのアドレス・トレーサを用いて、これらを追求することにした。

従来、計算機の中央処理装置の性能の比較を行なうために、Gibson mix などに代表される mix 値がよく用いられてきた。これは計算機の機械語命令をいくつかのグループに分類し、各グループに属する命令の実行時間にある重みをかけて、平均をとった平均命令実行時間である。その重みとして、プログラムが平均的にどの機械語命令をよく使うかという性質の測定結果が使用されている。しかし、一般に、各命令使用頻度は各プログラムの種類や性質、計算機の利用形態により異なると考えられる。その異なり方の程度があまり大きいと、このような mix 値には信頼がおけないということになる。我々は、この点を多少とも明らかにするために、例えば、Gibson mix が与える計算機の命令実行時間につける重みのパーセンテージが実際のプログラムとどの位のバラツキがあるか等を調べることにした。

次に、プログラムの性質の1つとして、中央処理装置を連続して使用する時間あるいは、その間に実行する命令数で表現されるプログラムの CPU使用特性がある。多重プログラミングシステムにおける性能を向上させるための方法に、動的資源管理として、ダイナミック・ディスパッチング方式が実現されている。これには、CPU使用時間にもとづいてスケジューリングを行なう HasP 法などがある。〔1〕 HasP 法では、I/O 時間間隔について、その平均が短いジョブほど、高い優先順位が与えられる。これらのことを考えると、CPU使用特性は、このようなスケジューリングの設計に大きな影響を与えるはずである。すなわち、このようなスケジューリング方式は、各プログラム毎の CPU使用特性が、各プログラム内では比較的安定で、かつ予測可能であるが、プログラム間では大きなバラツキがある場合に、十分効果があると考えられる。このため、CPU使用特性の実験が望まれる。箱崎氏らは、この CPU使用特性を I/O 要求特性であると考え、I/O 要求特性がワイブル分布で近似される可能性が高いと報告している。〔2〕我々は、この I/O 要求特性のよりダイナミックな性質を調べるため、実験を行なうことにした。

更に、仮想記憶システムでは、プログラムの命令やデータのアドレス参照に関

するふるまいが、システムの性能に大きな影響を及ぼす。従って、仮想記憶システムの性能を評価する場合は、プログラムのふるまいをどのようにモデル化するかが、その評価に大きくかかわってくる。プログラムのふるまいを示す重要な関係として、lifetime 関数がある。lifetime 関数 L(m) は、プログラムに割り当てる平均メモリ量 m の関数として、page fault 間の平均実行時間を与える。lifetime 関数 L(m) は m が小さいとき下に凸、m が大きいとき上に凸であると報告されている。特に、Belady [3] らは m が小さいとき、lifetime 関数が $L(m) = c \cdot m^k$ (c, k は定数, $k \simeq 2$) であると報告している (これに対して Saltzer[4] は Simple Linear Model を提案し、lifetime 関数を a を定数として、$L(m) = a \cdot m$ と表わしている)。これにもとづいて、Belady らは Biased Replacement 方式を提案している。また、Ghanem [5], Spirn [6], Fin [7] らは、この lifetime 関数を多重プログラミングシステムにおける主記憶割り当ての最適化問題の解析に用いている。このように、Belady らの lifetime 関数が広く利用されている現在、その正しさの程度を調べ、パラメータの性質を理解することは重要であると思われる。また、アドレス参照は命令とデータの二種類があり、各々異なった性質を有すると考えられる。そこで、まず、我々はトレーサにより得た参照系列を3つの場合 (命令とオペランド、命令、オペランド) に分け、それぞれについて、lifetime 関数のパラメータである c と k とを求め、比較、検証を行なうことにした。さらに、一般に、プログラムが1つのフェイズからだけでなく、複数の性質の異なるフェイズから成り立っていることも報告されている。益田氏 [8] はメモリ管理方式の性能が locality の変化に対し、大きく変化することを報告している。このように、メモリ管理方式の性能を評価するにあたって、フェイズの変化を解析することが重要である。我々は、プログラムが複数のフェイズから成り立っていることを検証し、フェイズの時間間隔と各フェイズ内での lifetime 関数を求める。また、lifetime 関数のメモリ管理方式による違いも報告されているが、これもあわせて検討することにした。

トレーサについて

プログラム実行時の諸データをとるには、ソフトウェアを用いたトレーサによる方法と、ハードウェアを用いたモニタリングによる方法の2つに大別できるが、ソフトウェアによる方法が一般的である。ここでは、HITAC 8350 上にソフトウェアを用いたトレーサを作成し、プログラムの実行を interpretive にトレースしながら、諸データを磁気テープに出力する方法を用いた。このトレーサはアセンブラ言語で書かれたプログラムで、コアメモリ上で、サンプル・プログラム (実行形式) とリンクして、トレースを行なうものである。トレースに用する時間はトレースしないで実行するときのおよそ ~~60~70~~ 60~100 倍の時間を要する。トレーサから得られるデータは、命令コード、プログラム・カウンタの値、データ参照アドレス、実行時間などである。得られたデータの処理は FORTRAN プログラムによって行なっている。

命令の使用頻度の実験

計算機の性能を評価するために、従来、Gibson mixがよく用いられてきた。Gibson mix値はいくつかの科学技術計算用のジョブのプログラムをトレースして各種命令の頻度の統計を採ったものだと言われている。我々は、まず、プログラムの性質を調べるため、命令の使用頻度をトレーサを用いて求めた。ここではサンプル・プログラムとしてFORTRANのプログラムで、モンテ・カルロ法によるπの計算（プログラム1）と2分法による方程式の解法（プログラム2）の2つを示した。これらから得られたデータと、一般によく用いられているGibson mix値とを表1に記した。我々の実験によっても、ジョブが異なると得られるGibson mix値が異なることが確認された。

今後はもっとより多くのプログラムについて調査し、バラツキの度合いを追求する予定である。

表1　HITAC 8350によるFORTRANプログラムの命令比率　（単位%）

	命令	一般に使用されているGibson mix	FORTRANプログラム1	FORTRANプログラム2
1	add/sub/load/store	33.0	48.4	40.2
2	multiply	0.6	2.4	1.2
3	divide	0.2	2.1	0.0
4	branch	6.5	13.4	26.2
5	compare	4.0	2.6	10.2
6	transfer	17.5	20.4	8.5
7	shift	4.6	2.5	2.3
8	and/or	1.7	3.1	6.6
9	index	19.0	0.0	0.0
10	short floating add/sub	5.8	1.0	0.0
11	long floating add/sub	1.5	1.7	3.4
12	short floating multiply	3.2	1.1	0.3
13	long floating multiply	0.8	0.2	1.1
14	short floating divide	1.3	1.1	0.0
15	long floating divide	0.3	0.0	0.0

I/O要求特性の分布

プログラムのI/O要求特性がどのような分布を示すかについて、実際のプログラムについて検討を行なった。サンプル・プログラムとして、FORTRANで書かれているソーティング（プログラム1）、モンテ・カルロ法によるπの計算（プログラム2）、2分法による方程式の解法（プログラム3）を示した。いずれもFORTRANで書かれている実験データから得られたI/O要求間隔の累積分布を片対数グラフで示した。（図1）　図1をみると、I/O要求間隔は短いI/O要求間隔

のものと、長いI/O要求間隔のものの2つに大別されている。これらのプログラムについて、長いI/O要求間隔は異なるが、プログラム内においては比較的安定していることが観察される。

また、図2に経過時間に対するI/O命令の分布を示した。この図では、長いI/O要求間隔と短いI/O要求間隔のI/O命令が2度連続して実行され、しかも、長いI/O要求間隔で周期的に発生していることが示されている。このことから、これらのプログラムは、長いI/O要求間隔を周期とするループの構造をもつと考えられる。現時点では作られているサンプル数が少ないので分布への あてはめは難しいが、これらのプログラムは単一の分布形に当てはめるよりも、複数の分布形に当てはめた方が適切ではないかと考えられる。

今後は十分データをとって、分布への当てはめ、及び、I/O要求間隔の予測可能性について追求する予定である。

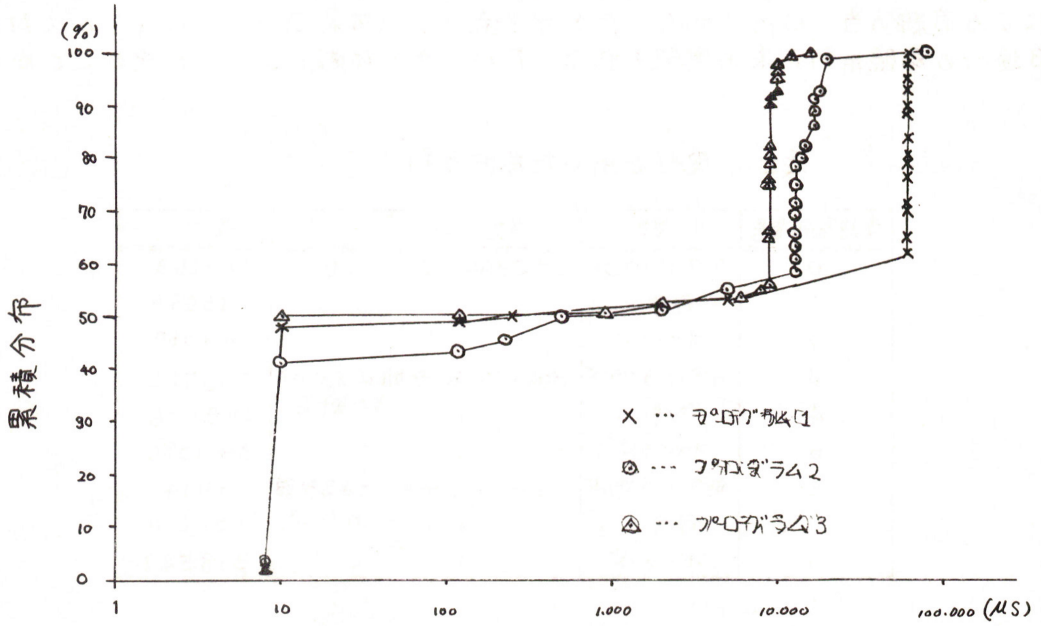


図1 プログラムのI/O要求間隔（累積分布）

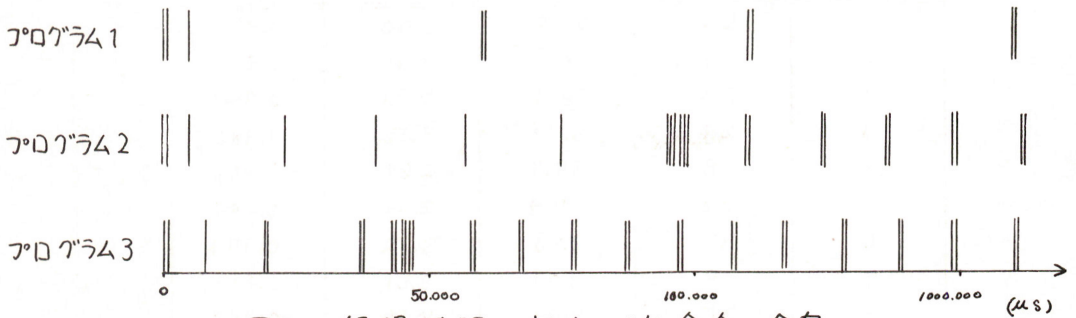


図2 経過時間に対するI/O命令の分布

lifetime関数

実験で使用した参照系列を表[2]に示す。参照系列はすべて、FORTRANプログラムからなる。前述のように、プログラムの命令参照とデータ参照の性質は異なると考えられるので、それぞれの性質を分析することも考えて、1つのプログラムに対し、3つの場合を調べた。すなわち、1)命令とオペランドの参照系列、2)命令の参照系列、3)オペランドの参照系列の3通りに分けて実験を行なった。lifetime関数はLRU方式のもとで求め、その計算には、参照系列の先頭から5000個の参照を除き、それ以降の参照を用いた。これによって、プログラムの実行初期段階の定常でない部分を除いている。なお、LRU-stackの更新手続きは、参照系列の先頭から行なっている。得られたlifetime関数を図3、図4、図5に示す。これらのグラフでは、両軸とも対数をとっているため、$L(m)=C\cdot m^{K}$ はグラフ上で直線になる。図3、図4、図5のグラフに示すように、mの小さいところ($m<10$)では、lifetime関数は、直線で近似できることがわかる。最小2乗法による直線の当てはめを行なった結果を表3に示す。同一のプログラムに対し、3通りの参照系列を求め実験を行なったが、その比較によって、次のことがわか

表2　実験で用いた参照系列

参照系列番号	分類	説明	長さ
0	命令とオペランド	FORTRAN, ソーティング	282663
1	命令	〃	83596
2	オペランド	〃	199067
3	命令とオペランド	FORTRAN, モンテカルロ法によるπの計算	536522
4	命令	〃	193136
5	オペランド	〃	343386
6	命令とオペランド	FORTRAN, 2分法による方程式の解法	419147
7	命令	〃	100600
8	オペランド	〃	318547

表3　実験結果

参照系列番号	総参照ページ数	C	K	相関係数
0	24	2.98	3.80	0.973
1	7	10.9	3.94	0.902
2	23	16.1	2.91	0.942
3	24	8.32	3.38	0.988
4	8	39.9	2.84	0.854
5	22	27.4	2.69	0.959
6	37	2.88	3.12	0.974
7	9	8.29	3.01	0.817
8	35	8.73	2.56	0.942

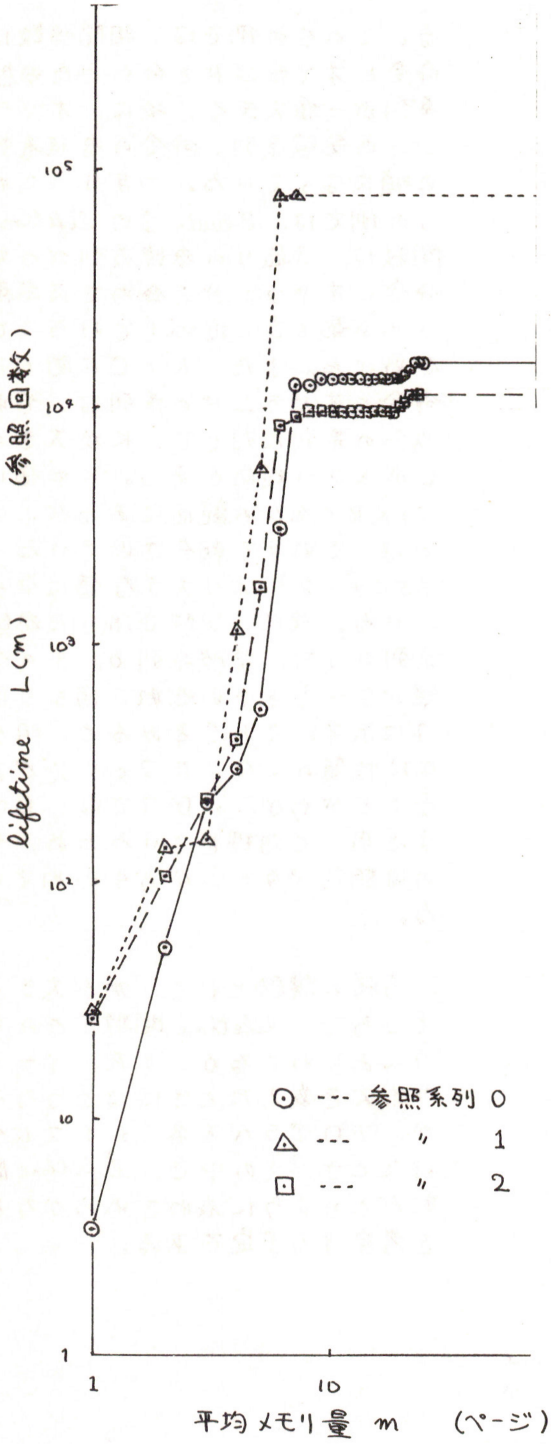


図3 lifetime 曲線

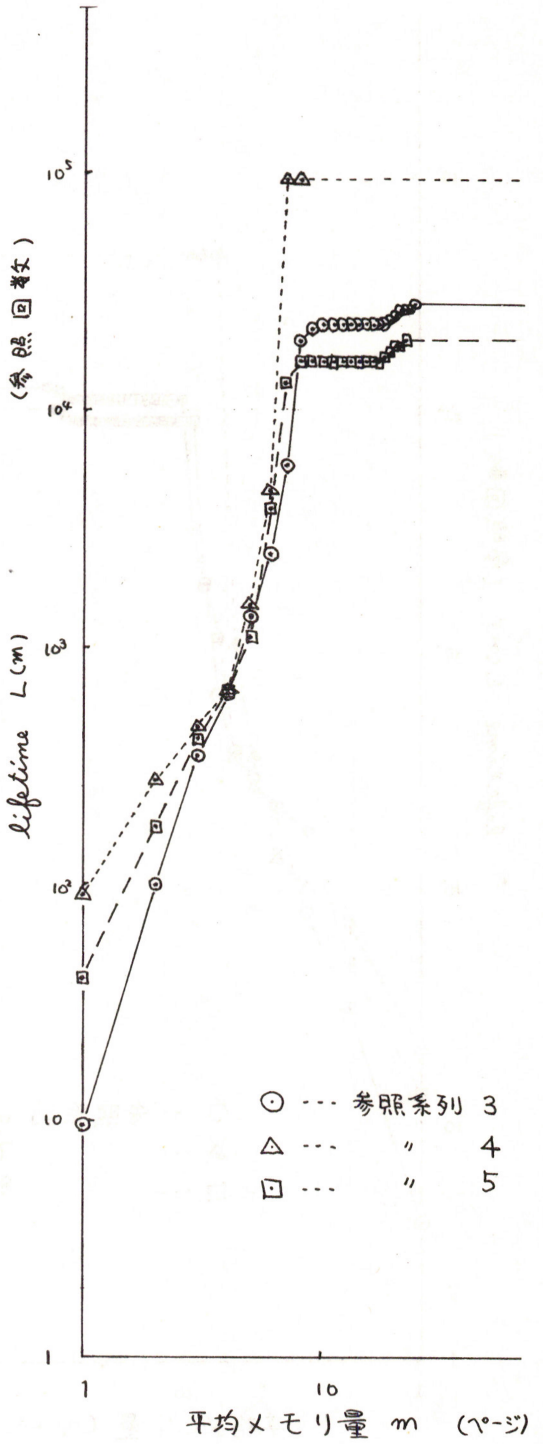


図4 lifetime 曲線

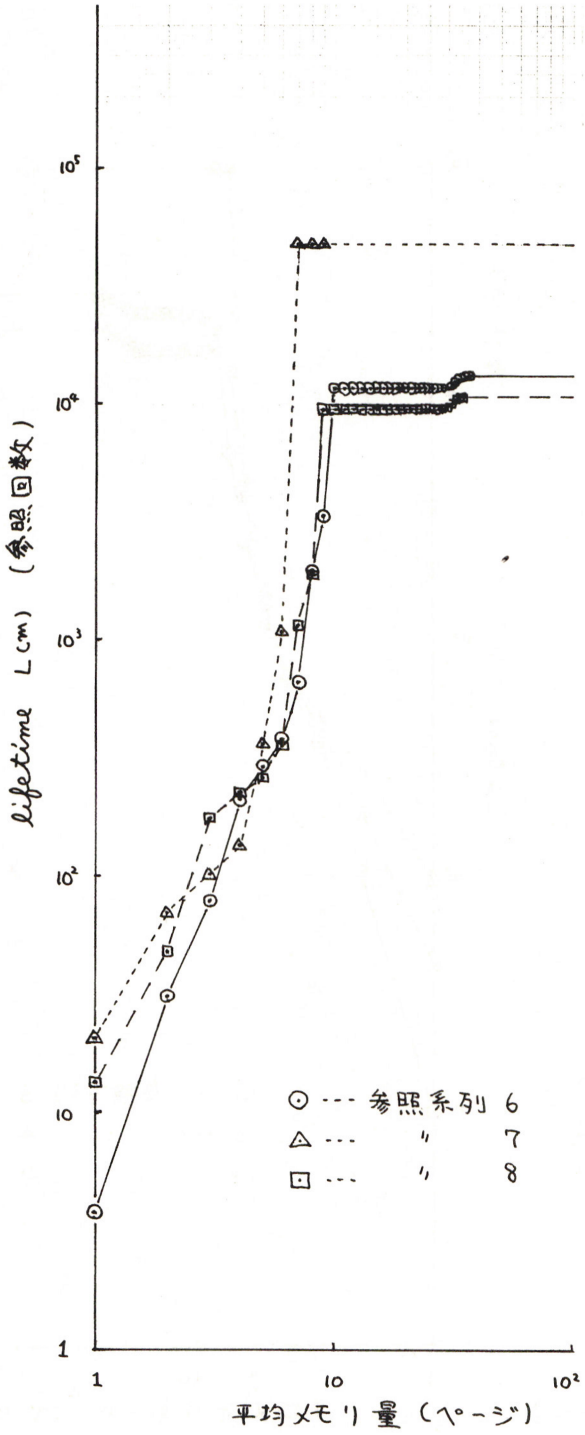
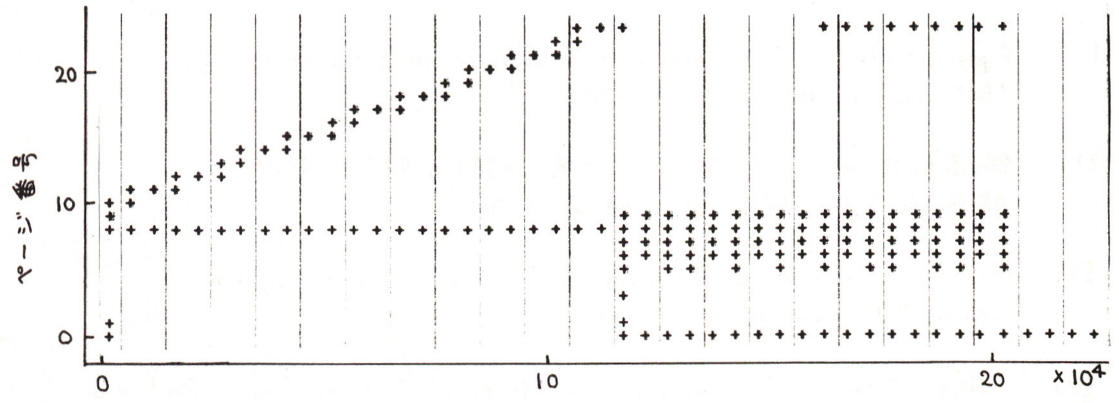


図5　lifetime 曲線

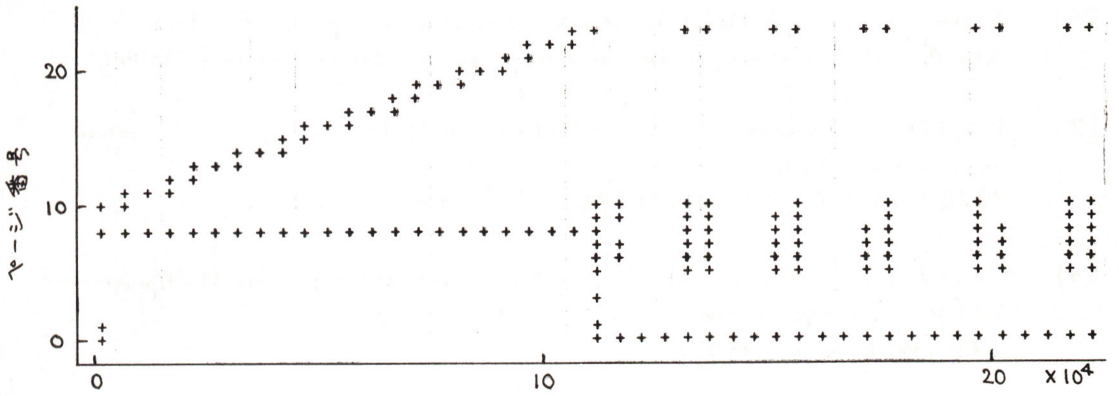
る。これらの例では，相関係数は命令とオペランドを合わせた参照系列が一番大きく，次に，オペランドの参照系列，命令の参照系列の順になっている。つまり，これらの例では，Belady らの lifetime 関数は，3通りの参照系列のうち，命令とオペランドを合わせた参照系列が最も良く近似していることがわかる。また，KとCに関して，命令とオペランドの系列は，それ以外の系列に対して，Kが大きく，Cが小さい傾向がある。Kの値は $2.5 < K < 4.0$ の範囲にあるが，これは，これまで報告されている $1.5 < K < 2.5$ より大きな値になっている。次に，実験で用いた参照系列のうち，参照系列0，3の参照パターンをそれぞれ，図6，図7に示す。これらをみると，明らかに性質のことなったフェイズがあることがわかる。図7では，DO文を用いて処理しているため，その周期的パターンがあらわれている。

今後の課題として，mが大きいところで，lifetime 関数がどのようにあらわせるか，また，ページサイズを変えたときにはどうなるか，プログラムを各フェイズに分けたとき，その中で，lifetime 関数がどのように表わされるかなどを考察する予定である。



処理時間　(参照回数)

図6　参照パターン（参照系列0）



処理時間　(参照回数)

図7　参照パターン　(参照系列3)

おわりに

ソフトウェアによるトレーサを作成し、プログラムの諸性質、主に3つについて実験を行なった。ここに示した数少ないサンプルだけみても、I/O要求間隔やlifetime関数は実際のプログラムの性質を表現するのに有効な手段であることが推察されると思われる。今後はより多くのデータをとって、解析を進める必要がある。

参考文献

[1] Ryder, K. D. : "A heuristic approach to task dispatching"
IBM Syst. J. Vol. 9, No. 3, (1970)

[2] 箱崎勝也 他 : "プログラムのIO要求特性の測定と解析"
信学論(D), Vol. J61-D, No. 3, (1978)

[3] Belady, L. A. : Dynamic Space-Sharing in Computer Systems"
Comm. ACM, Vol. 12, No. 5, (1969)

[4] Saltzer, J. H. : "A Simple Linear Model of Demand Paging Performance."
Comm. ACM, Vol. 17, No. 4, (1974)

[5] Ghanem, M. Z. : " Study of Memory Partitioning for Multiprogramming Systems with Virtual Memory"
IBM J. Res. & Develop. Vol 19, (1975)

[6] Spirn, J. R. : "A Model for Dynamic Allocation in a Paging Machine"
8th Annual Princeton Conference on Information Sciences and Systems, (1974)

[7] Fin, T. H, and Kameda, H. : " An Extension of a Model for Dynamic Allocation in a paging Machine"
情報処理学会 システム性能評価 21-1, (1977)

[8] Masuda, T : "Analysis of Memory Management Strategies for Multiprogrammed Virtual Storage Systems"
JIP, Vol. 1, No. 1, (1978)

本 PDF ファイルは 1979 年発行の「第 20 回プログラミング・シンポジウム報告集」をスキャンし、項目ごとに整理して、情報処理学会電子図書館「情報学広場」に掲載するものです。

この出版物は情報処理学会への著作権譲渡がなされていませんが、情報処理学会公式 Web サイトの https://www.ipsj.or.jp/topics/Past_reports.html に下記「過去のプログラミング・シンポジウム報告集の利用許諾について」を掲載して、権利者の捜索をおこないました。そのうえで同意をいただいたもの、お申し出のなかったものを掲載しています。

過去のプログラミング・シンポジウム報告集の利用許諾について

情報処理学会発行の出版物著作権は平成 12 年から情報処理学会著作権規程に従い、学会に帰属することになっています。

プログラミング・シンポジウムの報告集は、情報処理学会と設立の事情が異なるため、この改訂がシンポジウム内部で徹底しておらず、情報処理学会の他の出版物が情報学広場 (=情報処理学会電子図書館) で公開されているにも拘らず、古い報告集には公開されていないものが少からずありました。

プログラミング・シンポジウムは昭和 59 年に情報処理学会の一部門になりましたが、それ以前の報告集も含め、この度学会の他の出版物と同様の扱いにしたいと考えます。過去のすべての報告集の論文について、著作権者（論文を執筆された故人の相続人）を探し出して利用許諾に関する同意を頂くことは困難ですので、一定期間の権利者捜索の努力をしたうえで、著作権者が見つからない場合も論文を情報学広場に掲載させていただきたいと思います。その後、著作権者が発見され、情報学広場への掲載の継続に同意が得られなかった場合には、当該論文については、掲載を停止致します。

この措置にご意見のある方は、プログラミング・シンポジウムの辻尚史運営委員長 (tsuji@math.s.chiba-u.ac.jp) までお申し出ください。

加えて、著作権者について情報をお持ちの方は事務局まで情報をお寄せくださいますようお願い申し上げます。

期間：2020 年 12 月 18 日 〜 2021 年 3 月 19 日
掲載日：2020 年 12 月 18 日

プログラミング・シンポジウム委員会

情報処理学会著作権規程
https://www.ipsj.or.jp/copyright/ronbun/copyright.html