仮想メモリ・システムのワーキングセット最適化に関する考察と実験†

西垣 通†† 池田智明††

仮想メモリ方式の多重プログラミング・システムにおいて，ジョブの working-set を最適化する方式を提案した．最適性の評価基準としては STP (Space Time Product) を用いた．STP の最小化は，理論的には従来より論じられてきたが，実システムではジョブ特性が未知のため達成することができなかった．

本方式 OWE (Optimum Working-set Estimator) の特徴は，多重プログラミング環境における各ジョブの特性を実行中に学習し，STP を最小化するよう working-set の window size を定める点にある．

OWE を実現して実験を行い，各種のジョブについてその制御効果を確認した．さらに制御オーバヘッドについても考察を加えた．

1. まえがき

仮想メモリ方式による多重プログラミング・システムにおいて，P. J. Denning[3),4)] の提案した working-set 法は優れたメモリ制御方式と言われている．working-set 法では，ジョブが過去 window size τ 時間の間に参照したすべてのページにメモリを割り当てる．すなわちその特長は，局所参照集合を推定し，これをメモリ内に保つ点にある．さらに，(1)制御上の異常現象が生じないので制御が容易である[10)]，(2) Local LRU (Least Recently Used) 法など他の方式と比べて優れた性能が期待[5),15)]できる，などの長所が報告されている．

実際に working-set 法を適用する際の最大の問題点のひとつは，制御パラメータである window size τ をいかに定めるかである．一般にこれを増すと占有するメモリ量，減ずると page fault 率がそれぞれ増加することから，適正値が存在すると考えられる．従来その適正設定に関して，ジョブのアドレス軌跡の解析にもとづいた実験的な議論[11),18)]や，ページ参照間隔に関する単純なモデルにもとづく理論的な議論[3),16)]などが報告されている．特に G. Henderson[13)] らは，与えられたページ参照列に対し STP (Space Time Product) を最小化するよう window size τ を与えるアルゴリズムを述べたが，これは理論的に興味深い．STP はジョブのメモリ割当て量と処理時間との積であり，working-set 法に限らず一般にメモリ制御方式の評価基準として用いられる[1),2),12),17),20)]．STP 最小化は必ずしもシステムの処理能力最大化の必要十分条件ではないが，P. J. Denning[6),7)] らは page fault 処理にともなう STP を最小化する「Knee ルール」が処理能力を向上させるというシミュレーション結果を報告した．また，いくつかの仮定のもとにその最適性を理論的に示した報告[9),14)]もある．

これらの諸研究は，すべてプログラムの動作特性が与えられたと仮定している．通常のオペレーティング・システムの動作環境ではこれが未知のため，上記議論は適用できない．したがって working-set 法が用いられる従来のオペレーティング・システム[8),19)]においては，window size τ は経験的に定められ，その値の根拠も明確ではない．

本論文では，多重プログラミング環境におけるジョブの各々に対し，そのプログラム動作特性をあらかじめ知ることなく適正な window size τ を与える実用的な方式を提案，実験する．まず第2章で STP を定義し，その削減と処理能力との関係を論ずる．第3章で，ジョブ実行中にそのプログラム動作特性を学習し，STP を最小化する widow size τ を与える「ワーキングセット最適化方式 OWE (Optimum Working-set Estimator)」を提示する．さらに第4章で，OWE を実現して実験を行い，各種の定常的なジョブに対してその特性の学習精度，STP の削減効果を検証する．また制御オーバヘッドについても考察を加える．

2. Space Time Product と処理能力

長さ V のページ参照列が与えられたとする．window size τ を定めたとき，k 番目の参照における working-

† A Study and Experiment on the Optimum Working Set in a Virtual Memory System by TOHRU NISHIGAKI and CHIAKI IKEDA (Systems Development Laboratory, Hitachi Ltd.).

†† (株)日立製作所システム開発研究所

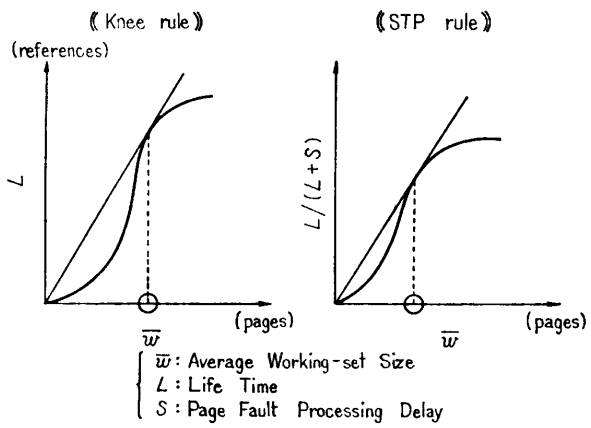

図 1 STP ルールおよび Knee ルールによる最適なメモリ割当て量
Fig. 1 Optimum working-set sizes by STP rule and Knee rule.

set size (メモリ割当て量) を w_k とすると，STP (Space Time Product) は次式で定義される．

$$\mathrm{STP} \triangleq \sum_{k=1}^{V} w_k + \sum_{k=1}^{V} e_k w_k S, \tag{2.1}$$

ただし，

$$e_k = \begin{cases} 1: k \text{ 番目の参照で page fault 発生} \\ 0: \text{上記以外} \end{cases} \tag{2.2}$$

Sは page fault 処理に要する時間をメモリ参照回数に換算した値であり，τ によらず一定とする．すなわち式 (2.1) において，時間の単位は単位参照あたりのメモリ・アクセス時間とする．w_k, e_k の平均値を，各々平均 working-set size $\bar{w}(=\sum_{k=1}^{V} w_k/V)$, page fault 率 $f(=\sum_{k=1}^{V} e_k/V)$ とよぶ．ここで，A. J. Smith[20]により実験的に確かめられている w_k と e_k の無相関性を仮定し，$\bar{w}$ と f を用いて式 (2.1) を次のように書き直す．

$$\mathrm{STP} = V(\bar{w} + Sf\bar{w}). \tag{2.3}$$

評価基準である単位参照あたりの STP は次式で与えられる．

$$X(\tau) = \bar{w}(\tau) + Sf(\tau)\bar{w}(\tau) \tag{2.4}$$

メモリ割当て量の評価基準として $X(\tau)$[1),2),12),17),20)]は代表的なものであり，これを最小化する方法を本論文では STP ルールとよぶ．

類似の評価基準として，life time L と $\bar{w}$ との比がよく知られている．L は $1/f$ に等しく，$\bar{w}$ と同様に τ の関数なので，L を $\bar{w}$ の関数として表わすことができる．単調増加関数 $L(\bar{w})$ は，$\bar{w}$ を増すにつれて下に凸から上に凸に変化する場合が多[7]く，$L/\bar{w}$ を最大化する点は曲線 $L(\bar{w})$ の「Knee」に対応する (**図 1** 参照)．したがってこの方法は Knee ルールとよばれる[6),7)]．なお，Knee ルールは式 (2.4) の第2項の最小化に等しいことから，これを式 (2.4) の第1項を無視した STP ルールと考えることもできる．

ここで $\bar{w}$ の関数 $L/(L+S)$ について考える．いま，L を $1/f$ でおきかえ式 (2.4) を用いると，

$$L/(L+S) = \bar{w}/X, \tag{2.5}$$

であるから，STP ルールは $\{L/(L+S)\}/\bar{w}$ の最大化に等しい．すなわち図1に示したように，STP ルールは曲線 $L/(L+S)$ の「Knee」に対応する $\bar{w}$ を与える方法に他ならない (なお，$L/(L+S)$ の Knee に対応する $\bar{w}$ は，L のそれより通常小さい．詳細は付録を参照されたい)．

以上より，STP ルールは投資 (=メモリ量) に対する効果 $(=L/(L+S))$ の比の最大化を意味することがわかる．ここで，$L/(L+S)$ と処理能力との関係について考える．これまでひとつのジョブに着目し，その仮想時間 (CPU 実行時間) の軸上で議論を進めてきたが，以下本章では多重プログラミング環境における複数のジョブを考え，実時間の軸上で議論を行う．いま，ジョブのプログラム動作特性のみに着目しているので，処理能力の尺度として CPU 利用率を採用する．ジョブ i による CPU 利用率を $u_{\mathrm{CPU}}^{(i)}$，その lift time を L_i とすると，

$$u_{\mathrm{CPU}}^{(i)} \leq L_i/(L_i+S) \tag{2.6}$$

である．等号はファイル入出力がなく，CPU でもページング・デバイスでも割当て待ちがないときに成立する．結局 CPU 利用率 u_{CPU} について，

$$u_{\mathrm{CPU}} \leq \min\{1, \sum_i L_i/(L_i+S)\} \tag{2.7}$$

が成り立つ．したがって，Xの削減は CPU 利用率向上のための必要条件であることがわかる．

なお，u_{CPU} にはいまひとつ上限値が存在する．ページング・デバイスの利用率を u_{PG} とし，システムの page fault 率 (単位時間あたりに各ジョブから発生する page fault 回数の合計) を η とする．また CPU の平均サービス時間を $1/\mu$，サービス完了率を λ とすると次式が成り立つ．

$$u_{\mathrm{CPU}} = \lambda/\mu \tag{2.8}$$

$$u_{\mathrm{PG}} = S\eta \tag{2.9}$$

page fault のみならずファイル入出力も CPU のサービス完了の原因となる．page fault により CPU サービスが完了する確率 η/λ は，

$$\eta/\lambda = 1/(\bar{L}\mu) \tag{2.10}$$

で与えられる[6]．ここで $\bar{L}$ は system life time であり，全ジョブに関する life time の平均値である．式 (2.8)～(2.10) より，

$$u_{CPU}=u_{PG}(\bar{L}/S) \tag{2.11}$$

が得られ，結局次式が成立する．

$$u_{CPU}\leq\min\{1, \bar{L}/S\}. \tag{2.12}$$

$\bar{L}$ が S よりやや大となるようにメモリ割当てを行う $\bar{L}=S$ ルール[6),7)]の根拠は式 (2.12) にある．$\bar{L}=S$ ルールは，STP ルールや Knee ルールと異なり，各ジョブについてそのメモリ割当て量を定めるための基準を与えるものではない．単にジョブの平均的な挙動に関するひとつの性能上の指針を与えるに過ぎない．$\bar{L}$ は L_i の重みつき平均として得られるが，この重みはジョブのファイル入出力発行特性や CPU のスケジュールなどに依存し，一様ではない．したがって一般に，$\bar{L}/S$ と $\sum_i L_i/(L_i+S)$ との大小関係は不明である．以上より，まず原則として STP ルールによりτを定め，しかる後に得られた $\bar{L}$ が S より小ならばτを少しずつ修正するという方法が，処理能力向上のための実用的な方法と考えられる．

3. ワーキングセット最適化方式 OWE

多重プログラミング・システムにおいて STP ルールを実現するワーキングセット最適化方式OWE(Optimum Working-set Estimator) につき述べる．もちろん式 (2.4) の X は厳密には事後的に定まる量であり，OWE はプログラム動作特性が定常的であるとの仮定のもとに近似的にその最小化をはかるものである．各時点で，OWE は各ジョブの w と f とを推定し，Xの推定値を最小化するτを与える．以下，その推定法を述べる．

仮想時間の軸上で，プログラムの実行開始時点から Δ ごとにサンプルした時点を $t_n(n=1,2,\cdots)$ とする．window size τ のもとで，t_n における working-set, working-set size をそれぞれ $W(t_n,\tau), w(t_n,\tau)$ とする．t_n において，最後の参照時刻が $(t_n-\tau-1)$ 以前のページはリプレースされる．ページ・リプレースメントはサンプル点のみで行われるので，これは次の近似を意味する．

$$W(t_n+\nu,\tau)\cong W(t_n+\nu,\tau+\nu). \quad (0\leq\nu<\Delta)\ (n=1,2,\cdots) \tag{3.1}$$

まず t_n における $w(\tau)$ の推定量 $\hat{w}(t_n,\tau)$ を次式で定義する．

$$\begin{cases}\hat{w}(t_n,\tau)\triangleq(1-\alpha)w(t_n,\tau)+\alpha\hat{w}(t_{n-1},\tau)\\ \hat{w}(t_0,\tau)=w(t_1,\tau).\end{cases} \quad (0<\alpha<1)\ (n=1,2,\cdots) \tag{3.2}$$

式 (3.2) は次のように書ける．

$$\hat{w}(t_n,\tau)=(1-\alpha)\sum_{j=0}^{n-1}w(t_{n-j},\tau)\alpha^j+\alpha^n w(t_1,\tau). \tag{3.3}$$

すなわち $\hat{w}(t_n,\tau)$ は過去 Δ ごとに測定した working-set size の重みつき平均であり，$w(\tau)$ の不偏推定量である．算術平均をとらなかった理由は，実環境で生ずるジョブの特性変化に追従しやすくするためである．

次に t_n における $f(\tau)$ の推定量 $\hat{f}(t_n,\tau)$ を次式で定義する．

$$\hat{f}(t_n,\tau)\triangleq\{\hat{w}(t_n,\tau+\Delta)-\hat{w}(t_{n-1},\tau)\}/\Delta. \quad (n=1,2,\cdots) \tag{3.4}$$

いま，window size τ のもとで t_{n-1}～t_n の間に生ずるpage fault 回数を $g(t_n,\tau)$ とすると，式 (3.1) のもとでこれは，

$$g(t_n,\tau)=w(t_n,\tau+\Delta)-w(t_{n-1},\tau), \tag{3.5}$$

で与えられる．式 (3.3)～(3.5) より次式が成り立つ．

$$\begin{aligned}\Delta\hat{f}(t_n,\tau)&=(1-\alpha)\sum_{j=0}^{n-1}w(t_{n-j},\tau+\Delta)\alpha^j+\alpha^n w(t_1,\tau+\Delta)\\&\quad-(1-\alpha)\sum_{j=0}^{n-2}w(t_{n-j-1},\tau)\alpha^j-\alpha^{n-1}w(t_1,\tau)\\&=(1-\alpha)\sum_{j=0}^{n-2}\{w(t_{n-j},\tau+\Delta)-w(t_{n-j-1},\tau)\}\alpha^j\\&\quad+(1-\alpha)w(t_1,\tau+\Delta)\alpha^{n-1}+\alpha^n w(t_1,\tau+\Delta)\\&\quad-\alpha^{n-1}w(t_1,\tau)\\&=(1-\alpha)\sum_{j=0}^{n-2}g(t_{n-j},\tau)\alpha^j\\&\quad+\alpha^{n-1}\{w(t_1,\tau+\Delta)-w(t_1,\tau)\}.\end{aligned} \tag{3.6}$$

すなわち，$\hat{f}(t_n,\tau)$ は window size τ のもとで過去 Δ ごとに測定した単位時間あたりの平均 page fault 発生回数の重みつき平均である．なお，上記議論と関連して一般に，

$$f(\tau)=\frac{dw(\tau)}{d\tau} \tag{3.7}$$

が成立することを付記[4)]する．

t_n における $X(\tau)$ の推定量 $\hat{X}(t_n,\tau)$ は次式で定義される．

$$\hat{X}(t_n,\tau)\triangleq\hat{w}(t_n,\tau)+S\hat{f}(t_n,\tau)\hat{w}(t_n,\tau). \quad (n=1,2,\cdots) \tag{3.8}$$

式 (3.2) (3.4) (3.8) より $\hat{X}(t_n,\tau)$ を求めるためには，$t_n(n=1,2\cdots)$ においてτの関数 $w(t_n,\tau)$ が測定できればよい．このため，各ページに非参照カウンタを設ける．なお，$w(t_n,\tau)$ のτについての最小測定単位は Δ

とし，測定点を，

$$\tau_m = m\Delta \quad (m=1,2,\cdots) \tag{3.9}$$

と書く．いま，t_n において，各ページが t_{n-1}〜t_n の間に参照されたか否かを調べ，参照されたページについてはその非参照カウンタをゼロクリアし，参照されなかったページについてはこれに1を加える（参照，非参照の区別は参照ビットを備えた計算機では容易である）．このとき，$w(t_n,\tau_m)$ はその非参照カウンタが m 以下である相異なるページ数として求められる．

以上より，$\min_m \hat{X}(t_n,\tau_m)$ に対応する τ を t_n における window size とすることができる．ただしここで，求められた window size は限られた範囲における最適値であることに注意されたい．t_{n-1} において設定した window size を τ_M とすると，t_{n-1} の時点で非参照カウンタが $(M+1)$ 以上のページはリプレースされる．したがって t_n で測定可能なのは $w(t_n,\tau_m)$ $(m=1,2,\cdots,M+1)$ であり，$\hat{X}(t_n,\tau_m)$ は $0\leq\tau\leq\tau_M$ の範囲でしか求められない．最適値がこれ以外の範囲にある場合の対策として，次の2条件が成立するときwindow size を τ_{M+1} に増加し，t_n でページ・リプレースメントは行わない．

$$\min_{1\leq m\leq M} \hat{X}(t_n,\tau_m) = \hat{X}(t_n,\tau_M). \tag{3.10}$$

$$\hat{X}(t_n,\tau_{M-1}) \gg \hat{X}(t_n,\tau_M). \tag{3.11}$$

多くの場合，X の τ による変化は比較的単純[2),11),12)]なので，このような処理で最適値に到達する可能性は大と考えられる．

なお，多重プログラミング環境における各ジョブに対して，仮想時間で Δ ごとに $w(t_n,\tau_m)$ を測定することは，次のようにして近似的に実現できる．OWE は実時間で周期的に起動されて割り込み，各ジョブのCPU 実行時間累計を用いて，前回測定以来の CPU 実行時間を算定する．これが Δ 以上であるようなジョブについて，その working-set* 内の各ページを調べ，$w(t_n,\tau_m)$ を測定する．このような処理によれば，Δ のサンプリングにおける最大誤差は OWE の起動周期に等しいことに注意されたい．

以上が STP ルールを実現するための OWE の概要である．ただし学習を行うためにはある程度の時間が必要なので，短小なジョブに対してはOWEの適用は難しい．すなわちOWEの主対象は，処理能力に多大な影響を与える長大なジョブである．

なお，式 (2.4) の第1項を無視し，$X(\tau)$ の推定量を $S\hat{f}(t_n,\tau)\hat{m}(t_n,\tau)$ とすれば，Knee ルールが実現できる．また式 (2.8) (2.10) より，

$$\bar{L} = u_{\mathrm{CPU}}/\eta \tag{3.12}$$

が成り立つので，CPU 利用率 u_{CPU} とシステムの page fault 率 η とを定期的に測定し，これから system life time $\bar{L}$ を求め，$\bar{L}<S$ のときは状態に応じて τ を修正することができる．処理能力向上のために式 (2.4) の第1項をいかに扱えばよいか，$\bar{L}$ と S の差をいかに制御に反映すればよいか，等については，なお検討の余地が残されている．

* 本論文ではジョブ間で共用するページは考察の対象外とする

4. 実験と評価

4.1 負荷および測定条件

ワーキングセット最適化方式OWEを実現し，大型計算機 HITAC M-180 を用いて実験を行った．OWE の最終目標は処理能力の向上にあるが，処理能力はメモリ制御方式のみならず負荷や種々の周囲条件によっても甚だしく影響を受ける．したがって本実験では，OWE によるプログラム動作特性の学習精度，STP 削減効果を検証することを直接の目的とした．実際のジョブは複雑な特性をもち，また特性自体も時間的に変動するため，この目的には不適当である．したがってここでは，理論的に特性が求められる人工的なジョブを用いた．

本実験で用いたジョブは SLRUM (Simple Least Recently Used stack Model)[21)] にしたがってページを参照する．すなわち N ページからなるプログラムにおいて，スタック距離が j の位置にあるページは $q(j)$ $(j=1,2,\cdots,N)$ の確率で参照される．$q(j)$ は時間によらず，その和 $\sum_{j=1}^{N} q(j)$ は1に等しい．$q(j)$ にさまざまの値を設定することにより，各種の特性をもつ定常的なジョブが作成できる．R. Turner[22)] らは $q(j)$ が与えられたとき $w(\tau)$ を求める次の方法をみちびいた．いま，τ の間に参照される相異なるページ数すなわち working-set size が $h(h=1,2,\cdots,N)$ である確率を $P(h,\tau)$ とすると

$$P(h,\tau) = P(h-1,\tau-1)\sum_{j=h}^{N} q(j) + P(h,\tau-1)\sum_{j=1}^{h} q(j), \tag{4.1}$$

が成り立つ．また，

$$P(h,1) = \begin{cases} 1 : h=1 \\ 0 : h=2,3,\cdots,N \end{cases} \tag{4.2}$$

であるから，式 (4.1) (4.2) より $P(h,\tau)(h=1,2,\cdots,N)$ を求めることができる．$w(\tau)$ は次式で与えられる．

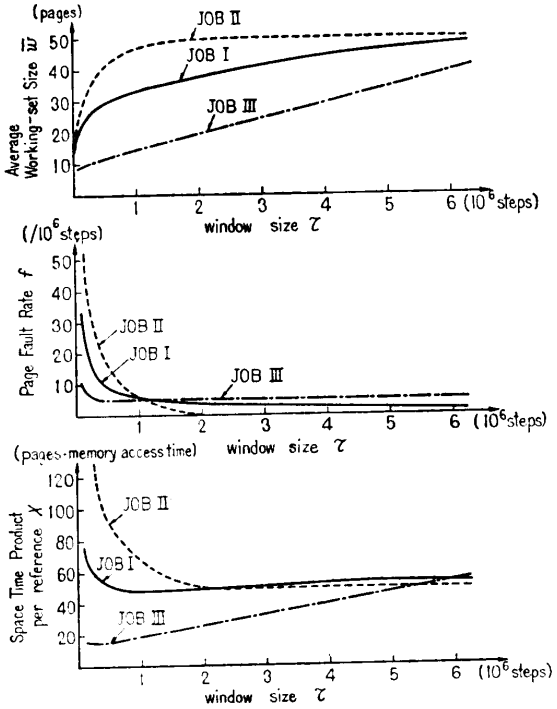

図 2 実験で用いたジョブの特性
Fig. 2 Characteristics of the jobs investigated in the experiment.

$$\varpi(\tau)=\sum_{h=1}^{N} hP(h,\tau). \tag{4.3}$$

さらに式 (3.7) を用いると $f(\tau)$ が求まるので，結局式 (2.4) を算定することができる．

3種類のスタック距離分布を設定し，これにしたがってページを参照する JOB I, II, III を作成した．上記のようにしてこれらの $\varpi(\tau)$, $f(\tau)$, $X(\tau)$ を求めた結果を**図 2** に示す．JOB I は $X(\tau)$ が極小値をもつ場合，JOB II, III はそれぞれ $X(\tau)$ が単調減少，単調増大の場合であり，プログラムの大きさ N はいずれも 50 ページである．

JOB I～III を多重プログラミング環境で実行し，OWE のもとでの working-set size を観測した．OWE の起動周期は実時間で約 100,000 ステップ，window size の調節間隔 Δ は仮想時間で 200,000 ステップとした．また，式 (3.2) において学習の重み係数 α は 0.5，式 (2.4) (3.8) の page fault 処理時間 S は 80,000 ステップとした．

working-set size の測定は OWE による正の効果の検証であるが，その負の効果として OWE 実行にともなう CPU オーバヘッドを測定した．いま，ジョブの特性は図 2 に示したように既知である．したがってジョブ実行を通じ，$X(\tau)$ を最小化する最適な window size を学習によらず固定値として与えることが可能である．このような理想的な場合について実測を行い，OWE のもとで学習により最適値を求めた場合のオペレーティング・システムのオーバヘッド増分を測定した．

4.2 測定結果と検討

OWE により window size τ を与えられる JOB I～III の各々について，実測値と理論的最適値との比較を**表 1** に示す．表 1 は，8 分間の測定を通じ，仮想時間 200,000 ステップごとに測定した window size τ の平均値，working-set size の平均値 ϖ，これに対応する単位参照あたりの Space Time Product X と，それぞれの理論的最適値との比較である．window size の実測値と理論的最適値とではやや差異があるが，一般に最適点付近では $X(\tau)$ の変動幅は小さい[2),11),12)]ことから，一応許容できる範囲にあると考えられる．実際に問題となるのは，制御パラメータ τ よりむしろ working-set size である．ϖ, X についての実測値と理論的最適値との差異は，それぞれ 2～5%，1～10% であり，各種のジョブに対して OWE の制御効果を確認した．

なお，これらの結果は，第 3 章で述べた方法によりプログラムの動作特性をマクロなレベルで測定できることを示している．従来，プログラム動作は専らアドレス軌跡を用いてミクロなレベルで解析されてきたが，この方法は処理するデータ量が多く，実際には数秒間程度の特性しか得られない．また多重プログラミング環境での特性解析にも不適当である．本論文で述べた方法は，多重プログラミング環境における数十秒～数十分間のプログラム動作の平均的特性の解析に適用可能である．

表 1 には，OWE の CPU オーバヘッドの測定結果

表 1 OWE の制御効果および制御オーバヘッドに関する実験結果（上段：実測値，下段：最適値）
Table 1 Experimental results of the OWE's control effects and overhead. (upper row: observed value, lower row: optimum value)

JOB	JOB I	IOB II	JOB III
Average Window Size (10^6 steps)	{0.82 1.20	{1.76 2.60	{0.56 0.30
Average Working-set Size (pages)	{32.5 34.2	{49.0 50.0	{11.1 10.7
Space Time Product per reference (pages·memory access time)	{48.5 48.1	{55.1 50.0	{15.8 15.2
CPU Overhead by OWE (%)		{1.16 —	{0.35 —

をも示した．OWE の処理においてはページごとに行う処理の占める部分が大なので，CPU オーバヘッドは一般にジョブの working-set size の増加関数となる．したがって working-set size 最大の JOB II と最小の JOB III について，それぞれ測定を行った．JOB II のみを5本，5多重で8分間実行した場合について，OWE のもとでのオペレーティング・システムのオーバヘッドと，あらかじめ最適な window size を与えた理想的な場合のそれとを比較した．両者の差が OWE 実行にともなう CPU オーバヘッドであり，これは 1.16% であった．JOB III について同様に比較すると 0.35% であった．JOB I については，この間の値をとると考えられる．

前述のように，$X(\tau)$ は最小値付近では τ の変化に対して余り敏感ではないので，ジョブが定常的であればひんぱんに τ を調節する必要はない．すなわち，制御系としてより安定させ，CPU オーバヘッドを削減することが可能である．$\hat{X}(t_n, \tau_m)$ の最小値を求めて τ を変える際，適当に閾値を設定することにより，τ の過度の変動を防ぐことができる．その設定値の検討も含め，CPU オーバヘッドの削減は今後の課題である．

5. むすび

仮想メモリ・システムにおいて，ジョブに割り当てられるメモリ量の評価基準として用いられる STP (Space Time Product) と処理能力との関係を論じた．多重プログラミング環境で working-set 法のもとで処理されるジョブについて，その STP を最小化するよう window size を与える「ワーキングセット最適化方式 OWE (Optimum Working-set Estimator)」を提案した．OWE は，処理能力に与える影響の大きい長大なジョブを主対象として，ジョブ実行中にそのプログラム動作特性を学習し，これにもとづいて window size を定めるものである．

OWE を実現し，大型計算機 HITAC M-180 を用いて実験を行った．各種のプログラム動作特性をもつ定常的なジョブを作成し，理論的に求めた STP の最小値と実験で観察された STP とを比較し，学習の効果を確認した．

今後の課題として，複雑な特性をもつ実際のジョブによる実験，制御オーバヘッドの削減などがあげられる．また，Knee ルールや $\bar{L}=S$ ルールなども含めた各種制御方式の実測評価も残された問題である．

終りに，本研究についてご指導いただいた東京大学穂坂衛教授，同大須賀節雄助教授に深く感謝いたします．また，本研究の機会を与えて下さった当社システム開発研究所三浦武雄所長，ならびに当社ソフトウェア工場服部陽一部長，同野口健一郎主任技師，同原田晃技師，同緒方慎八技師，の諸氏に心から感謝の意を表します．さらに，本研究を通じて終始有益な助言を与えて下さった当社システム開発研究所大町一彦主任研究員に深謝いたします．

参考文献

1) Belady, L. A. and Kuehner, C. J.: Dynamic Space-Sharing in Computer Systems, Commun. ACM, Vol.12, No. 5, pp. 282-288 (1969).
2) Chu, W. W. and Opderbeck, H.: The page fault frequency replacement algorithm, Proc. FJCC, pp. 597-609 (1972).
3) Denning, P. J.: The Working Set Model for Program Behavior, Commun. ACM, Vol. 11, No. 5, pp. 323-333 (1968).
4) Denning, P. J. and Schwartz, S. C.: Properties of the Working-Set Model, Commun. ACM, Vol. 15, No. 3, pp. 191-198 (1972).
5) Denning, P. J. and Graham, G. S.: Multiprogrammed Memory Management, Proc. IEEE, Vol. 63, No. 6, pp. 924-939 (1975).
6) Denning, P. J., Kahn, K. C., Leroudier, J., Potier, D. and Suri, R.: Optimal Multiprogramming, Acta Inf., Vol. 7, pp. 197-216 (1976).
7) Denning, P. J. and Kahn, K. C.: An $L=S$ criterion for optimal multiprogramming, Proc. ACM SIGMETRICS Int. Symp. Computer Performance Modeling, Measurement and Evaluation, pp. 219-229 (1976).
8) Doherty, W. J.: Scheduling TSS/360 for responsiveness, Proc. FJCC, pp. 97-111 (1970).
9) Fin, T.-H. and Kameda, H.: An Extension of a Model for Memory Partitioning in Multiprogrammed Virtual Memory Computer Systems, JIP, Vol. 2, No. 2, pp. 89-96 (1979).
10) Franklin, M. A., Graham, G. S. and Gupta, R. K.: Anomalies with Variable Partition Paging Algorithms, Commun. ACM, Vol. 21, No. 3, pp. 232-236 (1978).
11) Graham, G. S. and Denning, P. J.: On the relative controllability of memory policies, Proc. Int. Symp. Computer Performance Modeling, Measurement and Evaluation 1977, North Holland, Amsterdam, pp. 411-428 (1977).
12) Gupta, R. K. and Franklin, M. A.: Working Set and Page Fault Frequency Paging Algorithms: A Performance Comparison, IEEE Trans. Comput., Vol. C-27, No. 8, pp. 706-

712 (1978).
13) Henderson, G. and Rodriguez-Rosell, J.: The optimal choice of window sizes for working set dispatching, Proc. ACM SIGMETRICS Symp. Measurement and Evaluation, pp. 10-33 (1974).
14) Krzesinski, A. and Teunissen, P.: A Multiclass Network Model of a Demand Paging Computer System, Acta Inf., Vol. 9, pp. 331-343 (1978).
15) Masuda, T.: Analysis of Memory Management Strategies for Multiprogrammed Virtual Storage Systems, JIP, Vol. 1, No. 1, pp. 14-24 (1978).
16) Prieve, B. G.: Using Page Residency To Select the Working Set Parameter, Commun. ACM, Vol. 16, No. 10, pp. 619-620 (1973).
17) Prieve, B. G.: VMIN - An Optimal Variable-Space Page Replacement Algorithm, Commun. ACM, Vol. 19, No. 5, pp. 295-297 (1976).
18) Rodriguez-Rosell, J.: Experimental data on how program behavior affects the choice of scheduler parameters, Proc. 3rd ACM Symp. Operating System Principles, pp. 156-163 (1971).
19) Rodriguez-Rosell, J. and Dupuy, J.-P.: The Design, Implementation, and Evaluation of a Working Set Dispatcher, Commun. ACM, Vol. 16, No. 4, pp. 247-253 (1973).
20) Smith, A. J.: A Modified Working Set Paging Algorithm, IEEE Trans. Comput. Vol. C-25, No. 9, pp. 907-914 (1976).
21) Spirn, J. R. and Denning, P. J.: Experiments with program locality, Proc. FJCC, pp. 611-621 (1972).
22) Turner, R. and Strecker, B.: Use of the LRU Stack Depth Distribution for Simulation of Paging Behavior, Commun. ACM, Vol. 20, No. 11, pp. 795-798 (1977).

付　録

「STP ルールと Knee ルールによる最適値」

$X(\tau)$, $f(\tau)\varpi(\tau)$ がともに1個の極小値をもつと仮定し，これを与える τ をそれぞれ τ_S, τ_K とする．以下，各関数は連続で微分可能とする．式 (2.4) を微分して，

$$\frac{dX}{d\tau}=\frac{d\varpi}{d\tau}+S\frac{d}{d\tau}(f\varpi) \tag{1}$$

を得る．与件より，

$$\left.\frac{dX}{d\tau}\right|_{\tau=\tau_S}=0 \tag{2}$$

$$\left.\frac{d}{d\tau}(f\varpi)\right|_{\tau=\tau_K}=0 \tag{3}$$

である．ϖ は τ の広義の単調増加関数であるから，次式が成り立つ．

$$\left.\frac{dX}{d\tau}\right|_{\tau=\tau_K}=\left.\frac{d\varpi}{d\tau}\right|_{\tau=\tau_K}\geq 0 \tag{4}$$

τ_S は $X(\tau)$ の極小値に対応するから，式(2)(4)より，

$$\tau_S\leq\tau_K \tag{5}$$

を得る．

さらに ϖ の関数 $L/(L+S)$ の性質を調べるため，これを ϖ で微分する．

$$\frac{d}{d\varpi}\left(\frac{L}{L+S}\right)=\frac{S}{(L+S)^2}\left(\frac{dL}{d\varpi}\right). \tag{6}$$

$$\frac{d^2}{d\varpi^2}\left(\frac{L}{L+S}\right)=-\frac{2S}{(L+S)^3}\left(\frac{dL}{d\varpi}\right)^2+\frac{S}{(L+S)^2}\left(\frac{d^2L}{d\varpi^2}\right). \tag{7}$$

L は ϖ の広義の単調増加関数であるから，$L/(L+S)$ も ϖ の広義の単調増加関数であり，また L が上に凸のとき $L/(L+S)$ も常に上に凸であることが式(6)(7)よりわかる．L が ϖ の増加にしたがって下に凸から上に凸に変化し Knee をもつ場合，もし $L/(L+S)$ が同様に Knee をもつならば，式(5)よりその点は L の Knee 点以下である．(なお，この議論は $X(\tau)$ や $f(\tau)\varpi(\tau)$ が複数の極小値をもつならば成立しない．)

(昭和54年10月25日受付)
(昭和55年5月15日採録)