オフィス・システムにおけるＯＳの性能評価

7Y-7

高崎　英治　　山上　明　　橘高　大造

（三菱電機株式会社　計算機製作所）

１．はじめに

オフィス・システムにおける性能評価は、いかに実稼働システムに近い状態で計測するかが重要である。このために、３２ビット・オフィス・コンピュータであるＭＥＬＣＯＭ８０／システム３０・４０のオペレーティング・システムＤＰＳ１０においても、オフィス・システムとしての代表的なジョブを性能評価モデルとして設定し、それを様々な角度から測定・分析する方式を確立している。
今回この性能評価モデル、測定方式及び性能改善へのアプローチの考え方を紹介する。

一般に性能向上は、以下に示すようなアプローチがとられる。

（１）　客先、ＳＥといった外部からの要求値がまずある。
（２）　要求値に対して、開発サイドの目標値を決める。
（３）　目標値に対する根拠を明確にするため現在の性能を詳細に分析する。
（４）　分析結果に基づいて見込み値を求める。
（５）　目標値と見込み値、あるいは要求値と見込み値のギャップを埋めるために種々の性能向上策を講じる。

しかしながら、現実には確かな根拠を持たずに目標値を設定し、試験段階で性能不足となるケースが多い。
そこで我々は、根拠を重視し、より精密な見込み値を求めることによって、性能向上を進めてきた。すなわち、性能測定方式（はかり）と性能評価モデル（目盛り）が整備されていればいるほど、より有効な性能向上策（道具）を作れるというのが我々の考え方である。

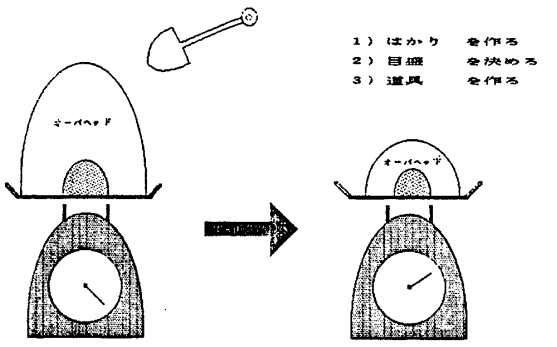

図１．１　　技術的アプローチ

２．性能評価モデル

オフコンにおける典型的なジョブには、次の２通りのジョブがあり、それぞれについて評価モデルを設定した。

（１）　インライン・モデル
（２）　バッチ・モデル

２．１　インライン・モデル

インライン・ジョブとは、ＴＳＳのように端末（ワークステーション）から起動やデータ入力を行うジョブであるが、データ・ベースを共用している点が特徴である。

典型的なジョブとしては伝票発行ジョブがあり、我々は「バチバチ・ジョブ」と命名したモデルを作った。これはＷＳとの対話により、画面で顧客番号、担当者番号等を入力していき、

（１）　５分間で何枚の伝票を発行できるか
（２）　１枚当たりのＣＰＵ時間

を測定するものである。

２．２　バッチ・モデル

一般的な事務処理システムにおけるバッチ・ジョブの実行性能を評価するためにＳＯＲＴ、仕訳帳作成など９つのジョブ・ステップからなる日次処理ジョブをモデルとして設定した。

３．　性能測定方式

以下にＤＰＳ１０で現在までに用いてきた性能測定方式を示す。

（１）　Ｈ／Ｗ
　　・Ｈ／Ｗモニター
（２）　Ｆ／Ｗ
　　・命令アドレス・カウント
　　・命令コード・カウント
　　・ＴＬＢアンヒット・カウント
（３）　Ｓ／Ｗ
　　・命令アドレス・カウント
　　・ＣＰＵタイマによるＳＶＣ性能測定
　　・Ｓ／Ｗモニタ
　　・関数コール／リターン履歴・性能測定
　　・その他

表３．１にこの中で良く使われるものの比較を示した。

３．１　命令アドレス・カウント

我々が最も良く用いているのが命令アドレス・カウント方式である。この方式は、プログラムを１ステップ実行する毎にマイクロ・プログラムで命令を実行したアドレスに対応するカウント・バッファのカウントをインクリメントすることによって、どのアドレスの命令が何回実行されたかを知るものである。

我々は、専用のツールを開発し、図３．１に示すような関数毎の実行ステップ数、コール回数を見れるようにした。また、アドレス順、ステップ数順、コール回数順に出力する機能も付加した。

表３．１　主な測定方式の比較

		長所	短所
H／W	H／Wモニター	・正確 ・経過時間、ＣＰＵ時間、ウェイト時間、総ステップ数がわかる	・台数に制限がある ・故障し易い ・セッティングが難かしい ・プロセスの時間がわからない
F／W	命令アドレス・カウント	・関数ごとのステップ数、コール回数がわかる ・Ｓ／Ｗよりも高速に測定できる	・実行時間は推定による
S／W	命令アドレス・カウント	・関数ごとのステップ数、コール回数がわかる	・測定に時間がかかる ・実行時間は推定による
	Ｓ／Ｗモニター	・経過時間、ＣＰＵ時間、ウェイト時間がわかる ・プロセスごとの時間がわかる ・フィールドで測定可能	・精度があらい

```
*****************************************
**********  DPS-10 セイノウ ソクテイ  **********
*****************************************

***** [ ] STEP スウ ジュン  *****

TOTAL       569356 STEPS
ELSE             0 STEPS
TOTAL        11096 CALLS
[  1] TIM :  912C -  9270 :  57087(STEPS) 10.027% :  153(CA
[  2] TRA : 17C30 - 188EC :  45771(STEPS)  8.039% :  195(CA
[  3] CLO :  8C38 -  912C :  40816(STEPS)  7.169% :  240(CA
[  4] WS# : AB27C - AB95C :  34809(STEPS)  6.114% :   33(CA
[  5] NAM : 725C8 - 72C78 :  20884(STEPS)  3.668% :    7(CA
[  6] IOS : EDD48 - EE3D0 :  14568(STEPS)  2.558% :   90(CA
[  7] WSB : 97B88 - 97CDC :  14496(STEPS)  2.546% :   12(CA
[  8] GET : E44F0 - E48C0 :  10220(STEPS)  1.795% :  134(CA
[  9] IRE : 71A64 - 71C00 :   9419(STEPS)  1.654% :   21(CA
[ 10] ENT :  3A78 -  3C92 :   8880(STEPS)  1.559% :  240(CA
[ 11] ISC : ED414 - EDA48 :   7524(STEPS)  1.321% :   36(CA
[ 12] IUP : 71DA0 - 72148 :   6364(STEPS)  1.117% :   10(CA
[ 13] ENT :  3D50 -  3E24 :   6237(STEPS)  1.095% :  189(CA
[ 14] WS# : AAA98 - AB27C :   6073(STEPS)  1.066% :   59(CA
[ 15] REA : 72E48 - 732C4 :   5943(STEPS)  1.043% :   33(CA
[ 16] RDW : 4E774 - 4EAF4 :   5916(STEPS)  1.039% :   44(CA
[ 17] WS# : AC29C - AC62C :   5705(STEPS)  1.002% :   59(CA
[ 18] ENQ : CA7EC - CAFC0 :   5528(STEPS)  0.971% :   25(CA
[ 19] IUC :  3C92 -  3D50 :   5420(STEPS)  0.952% :  330(CA
[ 20] RES :  7CD6 -  7DA4 :   5396(STEPS)  0.947% :   76(CA
```

図３．１　命令アドレス・カウント出力例
（実行ステップ数順）

４．　性能向上策

我々は、前章に述べた機能を単独または組み合わせて測定することによって、ＯＳやアプリケーション・プログラムのどの部分がオーバヘッドが大きいかが一目瞭然となり、的確な性能向上策を講じることができた。
以下にそのうちのＯＳに関する主なものを示す。

【アルゴリズム／方式レベルのアプローチ】
- ブロッキング
- 専用バッファの使用
- プログラムロードの高速化

【コーディング・レベルのアプローチ】
- アセンブラ化
- コンパイラ最適化
- 処理方式見直し

【命令レベルのアプローチ】
- 関数Ｆ／Ｗ化
- コール／リターン命令の新設
- 命令の選択

この結果図４．１に示すような効果が得られた。

５．　おわりに

ＤＰＳ１０は、初版以来速度性能を向上させる努力を重ね、現在では初版の２～３倍の性能を実現することができた。
これは主にＣＰＵ負荷の軽減を重点的に実施してきた結果であり、一応の成果が得られた。今後は入出力を含めたシステム全体のチューン・アップを計っていく予定である。

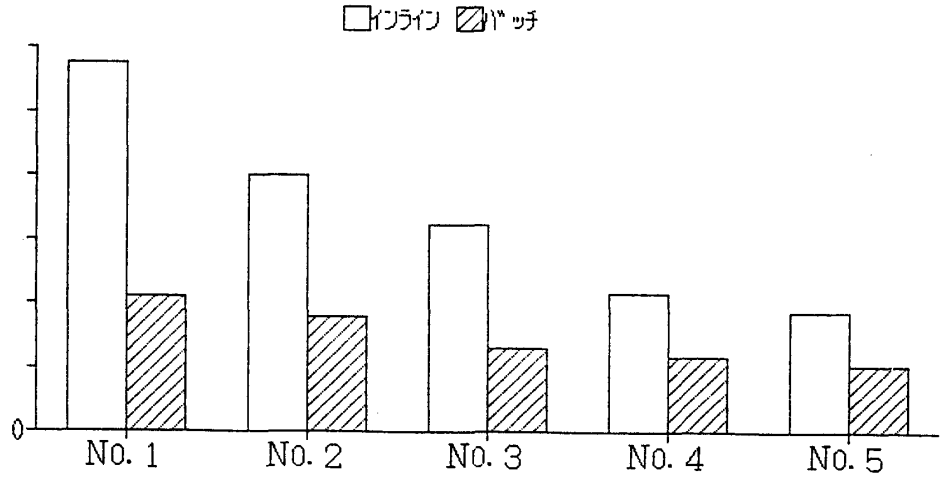

図４．１　性能改善の歴史