階層化ＯＳの性能評価手法に関する一考察

5Ｇ−5

宮内一暁、郷原純一

ＮＴＴ情報通信網研究所

1．はじめに

ＯＳをハードウェアに依存する下位層部分と非依存な上位層部分とに分離させた階層化ＯＳでは、ハードウェア非依存な部分を他のシステムに移植する事により、ソフトウェア資産を有効利用する事が出来る。

ＣＴＲＯＮは、この構成法を採用しており、ハードウェアに非依存な部分を拡張ＯＳ、ハードウェア依存の部分を基本ＯＳと呼んでいる。

移植後のＯＳの性能は移植の可否を判断する重要なデータであるため、実際に移植が行なわれる前に予め移植後のＯＳの性能を見積る必要がある。

本稿では、その見積り手法を提案し、適用方法について考察する。

2．前提条件

ここで言う移植が行われる前とは拡張ＯＳが新しいシステムでコンパイルされる前であるとする。つまり、拡張ＯＳのソースプログラムをアセンブラに展開して解析をすることが出来ない状態で見積りを行うとする。

また、ユーザの立場で行える見積り手法であるとする。すなわち拡張ＯＳのインプリメントの詳細を知る必要がなく、且つ簡易な手法である事が要請される。

移植後に出来上がるシステムの全体的特長を見定める事が目的なので、見積り精度の目標としては１割未満の誤差に納めるとする。

3．見積り式の提案

拡張ＯＳのシステムコールの処理時間（拡張ＯＳ処理時間）を拡張ＯＳ自身の走行する拡張ＯＳ内処理時間と内部発行される基本ＯＳシステムコールの処理に費やされる基本ＯＳ処理時間とに分割した上で考察を行う。（下式）

拡張ＯＳ処理時間 ＝ 拡張ＯＳ内処理時間＋基本ＯＳ処理時間
（Tti）　　（Tei）　　（Tbi）

この２式の各項を図１に表した。

ｉ＝１の時が、移植元での処理時間であり、ｉ＝２の時が、移植先での処理時間である。

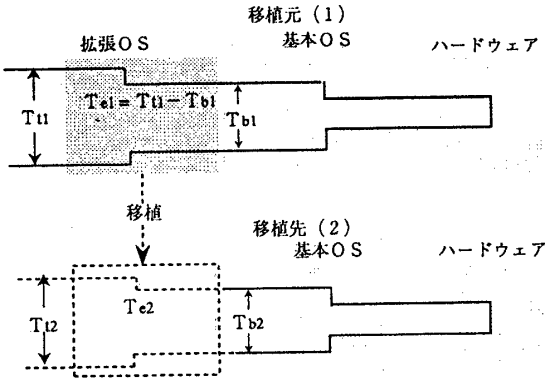

図１．移植によるシステムコールの処理時間の変化

図２が、提案する見積り式である。

前提条件よりデータ測定が可能な環境は以下の二つである。

（１）移植元のシステム
（基本ＯＳ、拡張ＯＳ共に測定可能）

（２）移植先の基本ＯＳのみが動作するシステム

基本ＯＳ使用時に呼び出し条件をトラップする事により、基本ＯＳ処理時間（Tb2）は移植先のシステム（２）で測定可能である。

移植元の拡張ＯＳ内処理時間を移植元のシステム（１）で実測可能である値の差分Tt1−Tb1によって得る事が出来るので拡張ＯＳ内処理時間（Te2）は、この時間を補正して見積る。

ここに、例えば、ルーチンタイプのインプリメント方式を取ったファイルアクセスシステムコールのように拡張ＯＳ内処理時間がＣＰＵ走行時間であれば、Te1とTe2との差はＣＰＵ処理能力の差であると考えられる。拡張ＯＳが高級言語で制作されている場合は、コンパイラの性能もＣＰＵ走行時間に影響を与える。従って、この時は補正関数としてＣＰＵとコンパイラの性能値の比が適用出来る。

また、サーバクライアントタイプのインプリメントを取っているファイルアクセスシステムコールのように拡張ＯＳ内処理時間にＣＰＵがwaitする時間がある時は、その時間を考慮した補正関数を導出する必要がある。

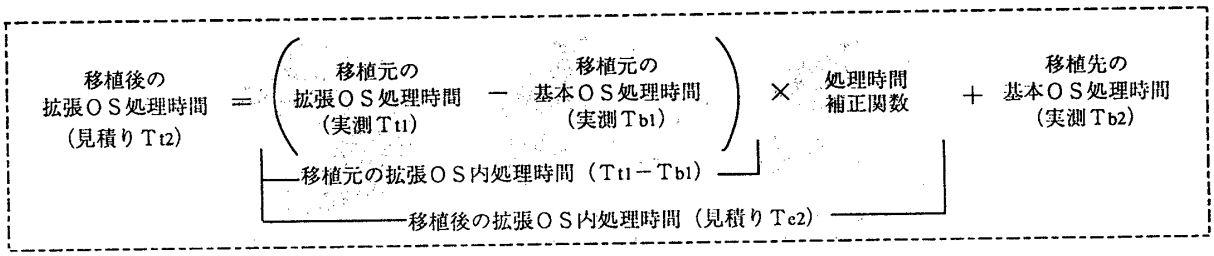

図２．提案する処理時間の見積り式

A study on performance evaluation method for two layered OS
Kazuaki MIYAUCHI,Junichi GOHARA
NTT Network Information Systems Laboratories

4．考察

4.1. 補正関数の考察

表１は、拡張ＯＳ処理とDhrystone1.1との同一環境での命令の内分けであるが、両者の命令配分の傾向は近いと言える。

これから、拡張ＯＳ内処理時間内がＣＰＵ走行時間である場合の補正関数としてDhrystoneが採用出来ると考え、Dhrystone値の適用性を分析した。

表１．インストラクションデータの内分け

		拡張ＯＳ処理	Dhrystone1.1
固定小数点命令	ロード命令	45.5%	46.3%
	ストア命令	10.5%	14.4%
	演算命令	13.0%	16.7%
浮動小数点命令		0.0%	0.0%
論理演算命令		11.4%	7.3%
１０進演算命令		0.0%	0.0%
分岐命令		19.6%	15.3%

Dhrystoneを用いると補正関数は、Dhrystone値の逆数の比で表される。

システムコール処理時間の比からなる補正関数を導き、Dhrystoneと比較した。（２種類のＣＩＳＣハードウェア上での同一ファイル管理についての例を表２に示す。）

ファイル管理内処理時間の比とDhrystone値の逆数の比は近い値となった。

表２．ファイル管理内処理時間とDhrystone値

	ハードウェア A	ハードウェア B	比
読み込み（非同期）	1.5 ミリ秒	1.3 ミリ秒	0.86
アクセス完了待ち	0.91 ミリ秒	0.73 ミリ秒	0.80
1／Dhrystone値	1／1143	1／1409	0.81

特にＲＩＳＣアーキテクチャに対して、［１］等でＯＳの処理に対してのＣＰＵの能力値としてDhrystone等ベンチマークの結果が適当でないという指摘がなされているが、［２］のカーネルに対するＣＩＳＣハードウェア上の実験結果でもDhrystoneは妥当な値を得ている。

従って、ＣＩＳＣアーキテクチャであり、且つ、ファイルアクセスのような単純な処理の場合、Dhrystone値から補正関数を導く事が可能であると言える。

4.2. 基本ＯＳ処理時間の考察

基本ＯＳがカーネルと入出力制御からなるＣＴＲＯＮの例で検討する。

（１）実Ｉ／Ｏ時間を含むシステムコールの場合

一般に、実Ｉ／Ｏ時間はＣＰＵ走行時間と比較して桁が違って長い。表３の（a）（b）は、比較的実Ｉ／Ｏ時間の短い例であるが、それでも実Ｉ／Ｏ時間を含む入出力制御処理時間が見積り時間中の９割以上を占め、カーネル処理時間は４％程度でしかない。

そこで、入出力制御処理時間のみを基本ＯＳ処理時間として扱い、カーネル処理時間は拡張ＯＳ内処理時間としても、誤差は１割未満であると判断出来る。

表３．ファイル管理システムコールの走行時間の内分け

		(a) 読み込み（同期） 4 Kバイト	(b) 書き込み（同期） 4 Kバイト
ファイル管理本体		0.36 ミリ秒（3.2%）	0.37 ミリ秒（3.3%）
カーネル	システムコールA	0.16 ミリ秒（1.4%）	—
	システムコールB	0.15 ミリ秒（1.3%）	0.15 ミリ秒（1.3%）
	システムコールC	0.16 ミリ秒（1.4%）	0.16 ミリ秒（1.4%）
	システムコールD	—	0.08 ミリ秒（0.7%）
	システムコールE	—	0.07 ミリ秒（0.6%）
入出力制御	ＣＰＵ部	2.33 ミリ秒（20.9%）	2.34 ミリ秒（20.9%）
	実Ｉ／Ｏ部	約8ミリ秒（約72%）	約8ミリ秒（約72%）

（２）実Ｉ／Ｏ時間を含まないシステムコールの場合

ファイル管理本体部の全処理時間中に占める割合は、２～３割であると予想され、実Ｉ／Ｏを含まなくても基本ＯＳ処理時間が多くを占めると考えられる。

図３の（c）（d）（e）が、この場合に該当する。

カーネル処理時間は、全走行時間中で、入出力制御システムコールを含む場合も２割程度はある。

従って、入出力制御システムコール処理時間は勿論の事、主要なカーネルのシステムコール処理時間も基本ＯＳ処理時間として扱う必要がある。

5．おわりに

今後は、見積り式中の補正関数について更に考察する。

［参考文献］

［１］
Thomas E.Anderson,Henry M.Levy,Brian N.Bershad,
and Edward D.Lazowska:
"The Interraction of Architecture and Operating System Design",
Proceedings of ASPLOS-IV,pp108-120,1991.

［２］
H.Kurosawa,O.Watanabe,Y.Kobayashi.
"An Evaluation Method of Kernel Products Based on CTRON",
TRON Project 1990,Springer-Verlag,pp191-200,1990.

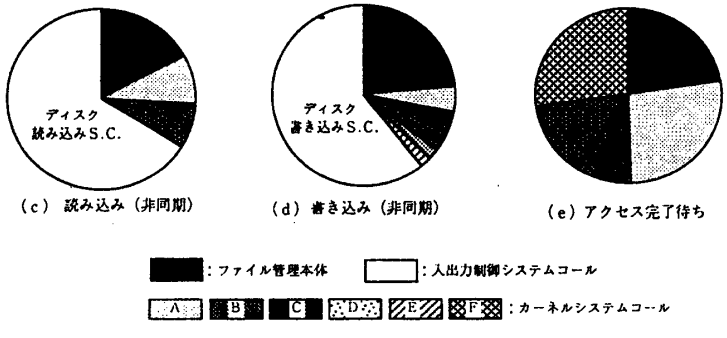

図３．ファイル管理システムコールの走行時間の内分け