オペレーティング・システム　50－3
（１９９１．３．１５）

ＤＩＲＯＳにおける性能見積り手法

谷口　秀夫　　箱守　聰
ＮＴＴデータ通信（株）　開発本部

分散処理システムにおいて、システムの処理効率を上げるためには、応用プログラムの処理だけではなくオペレーティングシステムの処理も分散できることが大切である。ここでは、分散型リアルタイムオペレーティングシステム（ＤＩＲＯＳ：ＤＩstributed Real-time Operating System）について、応用プログラムが発行するシステムコールの各プロセッサ負荷を見積る手法を述べている。ＤＩＲＯＳでは、オペレーティングシステムの処理を部品化している。これにより、種々の分散形態におけるプロセッサ負荷見積りを容易にしている。

A Design of Performance on DIROS

Hideo TANIGUCHI,　Satoshi HAKOMORI
Development Headquarters
NTT DATA COMMUNICATIONS SYSTEMS CORPORATION
Kowa Kawasaki Nishi Bldg. 66-2 Horikawa-cho, Saiwai-ku,
Kawasaki-shi, Kanagawa 210, Japan
JUNET:tani@rd.nttdata.jp

On distributed systems, it is important to distribute not only application's processings but also operating system's processings in order to get a high performance. This paper discribes a design of each processor's weight for system-calls by application's call on DIROS. As operating system's processings are modules on DIROS, it is easy to design each processor's weight at any case.

1．まえがき

複数のプロセッサを結んだ分散システムにおいて、システムの処理効率を上げることは重要な課題である。このためには、応用プログラム（以降、ＡＰと略す）の処理をどのように分散させるかが鍵になる。また、ＡＰの処理を分散させることと関連して、オペレーティングシステム（以降、ＯＳと略す）の処理をどのように分散させるかも非常に大切である。

ＡＰ処理によるプロセッサ負荷の分散を、その時々のシステムの状態に合わせて動的に制御する試みがなされている[1]。しかし、制御のための処理負荷が大きく、充分とはいえない。ＯＳ処理の分散については、分散を可能にするＯＳ構成についての研究がいくつか発表されている[2][3]。しかし、ＯＳ処理によるプロセッサ負荷を分散させる手法については触れていない。

筆者らは、分散型リアルタイムオペレーティングシステム（ＤＩＲＯＳ：ＤIstributed Real-time Operating System）について、ＯＳ処理のプロセッサ負荷分散の手法を検討した。ＤＩＲＯＳは、リクエスト制御方式[4]に基づいて構築されている。そのため、ＡＰ処理だけではなくＯＳ処理も分散できるプログラム構造を持つ。また、ＯＳの処理プログラムをモジュール化しているため、各プロセッサの負荷を見積ることが容易である。

本稿では、ＡＰ処理を各プロセッサに分散させた時、ＡＰプロセスが発行するＤＩＲＯＳのシステムコールについて、各プロセッサの負荷を見積る手法について述べる。

2．性能見積りの必要性と問題点

2．1　必要性

端末制御装置は、ホスト計算機と端末の間に位置し、処理の中継などを行なう。この装置上でのトランザクション処理には、以下の特徴がある。

［特徴１］処理は、大きく次の三つに分類できる。一つは、端末との通信であり、データの送受信や制御を行なう処理である。二つめは、ホスト計算機との通信である。もう一つは、トランザクションの処理内容を記録するための磁気ディスクへのアクセスなどである。さらに、これらの処理を行なうためのメッセージ通信やセマフォによる排他制御などがある。以上に述べたように、大半の処理は、ＯＳがＡＰに提供しているシステムコールを利用した処理である。つまり、トランザクション処理によるプロセッサ負荷の大半は、ＯＳの処理である。

［特徴２］処理の内容や処理の量は、システム構築段階で明確である。そのため、システムの能力を最大限に引出す必要がある。つまり、各プロセッサの能力を限界近くで利用する。

［特徴３］トランザクション処理の効率を最大限に保つため、サービス実行中にＡＰプロセス生成などのサービス以外の処理は行なわないことが好ましい。つまり、各ＡＰプロセスが走行するプロセッサは、システム立ち上げ時に決定され、生成される。それ以降の変動はない。

上記の特徴により、各ＡＰプロセスが走行するプロセッサを決定する時の「ＡＰ処理のプロセッサ負荷バランス」と「ＯＳ処理のプロセッサ負荷バランス」を設計する必要がある。具体的には、種々のＯＳ処理の分散形態について、システムコールの命令実行数を各プロセッサ毎に明確にすることが求められる。

2．2　問題点

命令実行数を測定する方法には、プロセッサの命令トレース割込み機能を利用する方法や命

令信号のモニタによる方法などがある。いづれの方法も、測定が大変であるため、測定回数を少なくすることが大切である。

多くの場合、各システムコール単位にその処理の内容に応じて命令実行数を測定している。つまり、スタンドアロン環境における測定回数は、システムコール数と処理内容数で決定される。これに対し、分散環境においても上記の方法で測定すると、実測する項目数が非常に多くなる。具体的には、プロセッサ間にまたがる処理について、各プロセッサ上の命令実行数を測定する必要がある。この測定は、ＯＳ処理の分散形態やプロセッサ間を結合している通信路の種別の各々について必要である。

したがって、各処理について、効率的なプロセッサの命令実行数の測定が必要になる。

３．ＤＩＲＯＳ

３．１　ハードウェア構成

ＤＩＲＯＳが走行するハードウェア環境の例を図１に示す。図１に示すように、各プロセッサ間は、ハードウェアバスとＳＣＳＩ（Small Computer System Interface）およびＬＡＮ（Local Area Network）の３種の通信路で結ばれている。

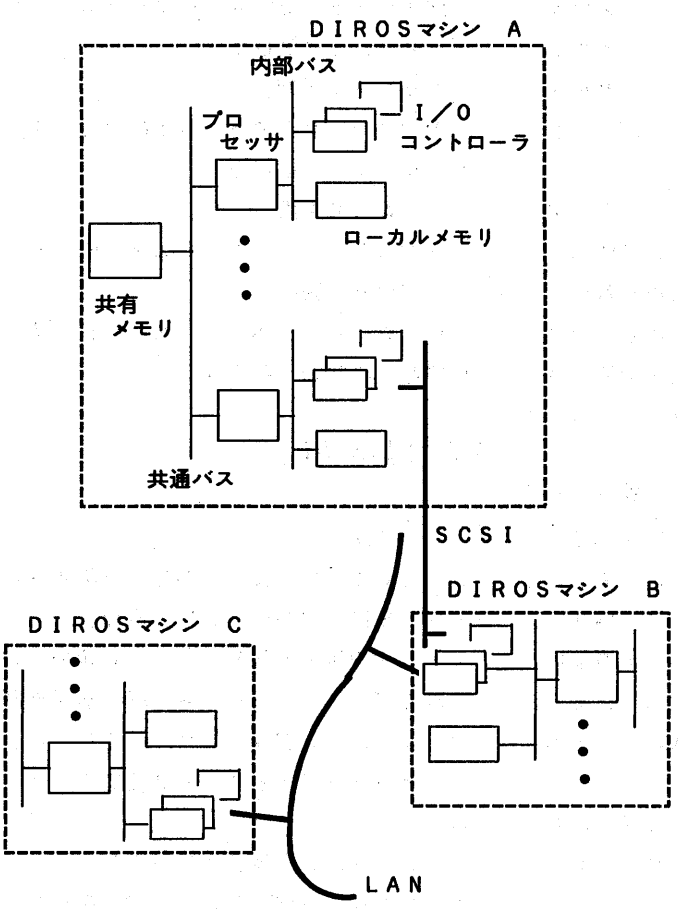


図１　ハードウェア環境の例

３．２　基本構造

ＤＩＲＯＳは、その構造に大きな２つの特徴を持つ。一つは、リクエスト制御方式に基づく処理プログラムの構造である。もう一つは、非完了型でシステムコールをＡＰに提供していることである。

リクエスト制御方式の概要を以下に述べる。

（１）システムコールに対応する機能を実現している処理を一つの単位として、ＯＳのプログラムを分割する。この処理の単位を処理モジュールと呼ぶ。

（２）処理モジュールは、分散環境で一意の処理モジュール識別子により識別する。処理モジュール識別子は、その処理モジュールが走行する計算機とプロセッサの情報および処理内容を識別する通番の情報を持つ。

（３）処理モジュール間での

情報のやりとりは、リクエストブロックと呼ぶ制御ブロックで行なう。リクエストブロックは、情報をやりとりする各処理モジュールの処理モジュール識別子と情報本体を持つ。

（４）処理モジュール間の通信は、プロセッサ内に閉じるだけではなく、プロセッサ間にまたがる。そのため、複数種の通信路を介して通信することになる。これらの通信方式の違いを吸収し、統一した処理モジュール間の通信インタフェースを提供している。

（５）処理モジュール間は、四つのインタフェースを持つ。それは、依頼と結果取得と取消および結果応答である。また、これらのインタフェースは、六つのルーチン（以降、ユニットと呼ぶ）で制御する。それは、アクションユニット，レスポンスユニット，リザルトユニット，クリーンユニット，キャンセルユニットおよびキャンセルアクションユニットである。

非完了型のシステムコールは、一つのＡＰプロセスが処理を並列的に行なうことを可能にしている。これにより、サービスを実現するのに必要なプロセス数を減らすことができる。つまり、サービス提供におけるＯＳ処理を軽減できる。非完了型のシステムコール提供の形態は、次のようになっている。

（Ａ）依頼システムコールにより、処理をＯＳに依頼する。このシステムコールは、依頼する処理の内容に応じて、様々なものがある。

（Ｂ）結果取得システムコールにより、処理の結果を取得する。このシステムコール処理は、すべての依頼システムコールに共通である。しかし、その返却値は、当該の依頼システムコールに依存する部分もある。

（Ｃ）取消システムコールにより、依頼した処理を取消すことができる。このシステムコール処理は、すべての依頼システムコールに共通である。取消を要求した時点での依頼処理の実行度合によって、取消処理の内容が異なる。

３．３　機能

ＤＩＲＯＳは、多くの機能を持つ。プロセッサ間にまたがった処理を可能にしているシステムコール機能について、リモートへの処理機能を以下に述べる。

（１）ファイルアクセス

別プロセッサにあるファイルへアクセスする。

（２）デバイスアクセス

別プロセッサに接続された入出力装置や通信回線を利用して、データを送受信する。

（３）メッセージ通信

別プロセッサ上のメールボックスに対し、メッセージを送受信する。

（４）セマフォ制御

別プロセッサ上のセマフォに対し、Ｐ／Ｖオペレーションを行なう。

（５）プロセス生成

別プロセッサ上にプロセスを生成する。

（６）プロセス制御

別プロセッサ上のプロセスに対し、停止や再開などの制御を行なう。

４．性能見積り

４．１　部品化

２．２節で述べたような問題を解決するため、ＤＩＲＯＳは、次の対処を行なった。

（１）ＯＳの処理を部品化した。具体的には、処理モジュールと、それらの間の通信を制御するリクエスト制御の部分を部品とした。

（２）処理を実行する時の各部品間の関係を明らかにした。

（３）各部品の命令実行数を測定した。

部品化された処理モジュール間の関係を図２に示す。通信制御部は、通信方式の違いを吸収している。この部分は、プロセッサ間を結ぶ通

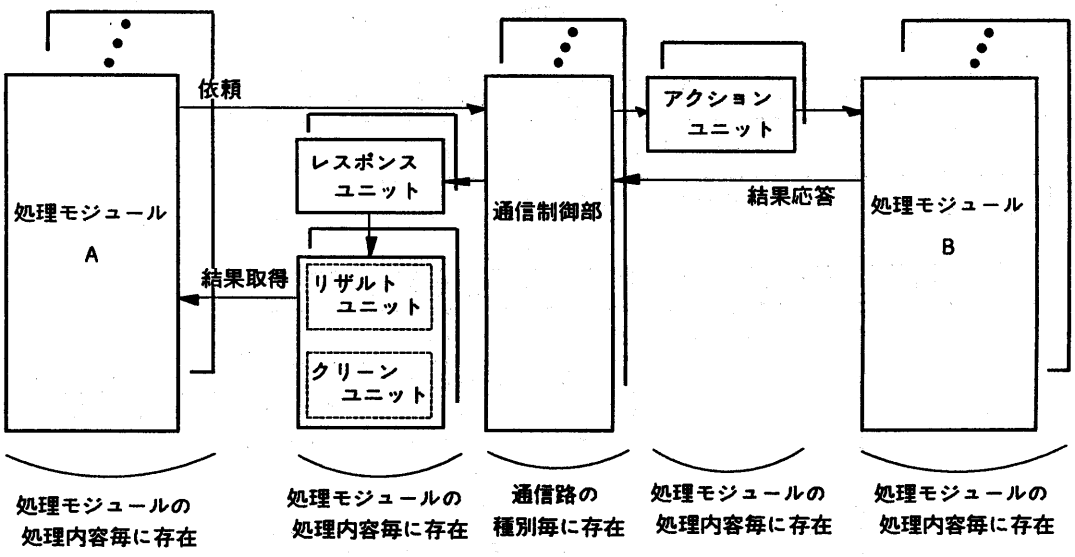


図２　部品化された処理モジュール

信路の種別に対応して、部品化した。各処理モジュールやユニットは、処理内容に対応して部品化した。

処理を実行する時の総命令実行数は、図２に示した各部品の命令実行数を処理の内容に合わせて加算することにより計算できる。

４．２　計算法

図２に示した部品化を、依頼システムコールと結果取得システムコールの処理に適用した場合について、各部品と処理流れの関係の例を図３に示す。図３において、Ｌ1，Ｓ，Ｔs1およびＡなどは、各部品の命令実行数を示す。

図３から、各処理の命令実行数を計算できる。一つのプロセッサに閉じた処理の場合、各処理の命令実行数は次の部分に分類できる。

［部分１］依頼システムコールに基づく処理部分の命令実行数

［部分２］結果取得システムコールに基づく処理部分の命令実行数

［部分３］非同期に実行される処理部分の命令実行数

この場合、各部分の命令実行数は、すべて同じプロセッサ上のものである。

次に、複数のプロセッサにまたがる処理の場合について述べる。例えば、図３において、システムコールを発行するＡＰプロセスが走行しているプロセッサと、実処理を行なうプロセッサが、異なる場合である。この場合、各処理の命令実行数は、上記の３分類に加え、さらにその各部分について、以下の分類ができる。

［分類Ａ］ＡＰプロセスが走行しているプロセッサ上の命令実行数

［分類Ｂ］実処理を行なうプロセッサ上の命令実行数

これらの部分と分類について、各命令実行数の計算式を表１に示す。

４．３　特徴

４．１節に述べた対処により、ＤＩＲＯＳにおける性能見積りは、以下の特徴を持つ。

（１）測定項目数の抑制

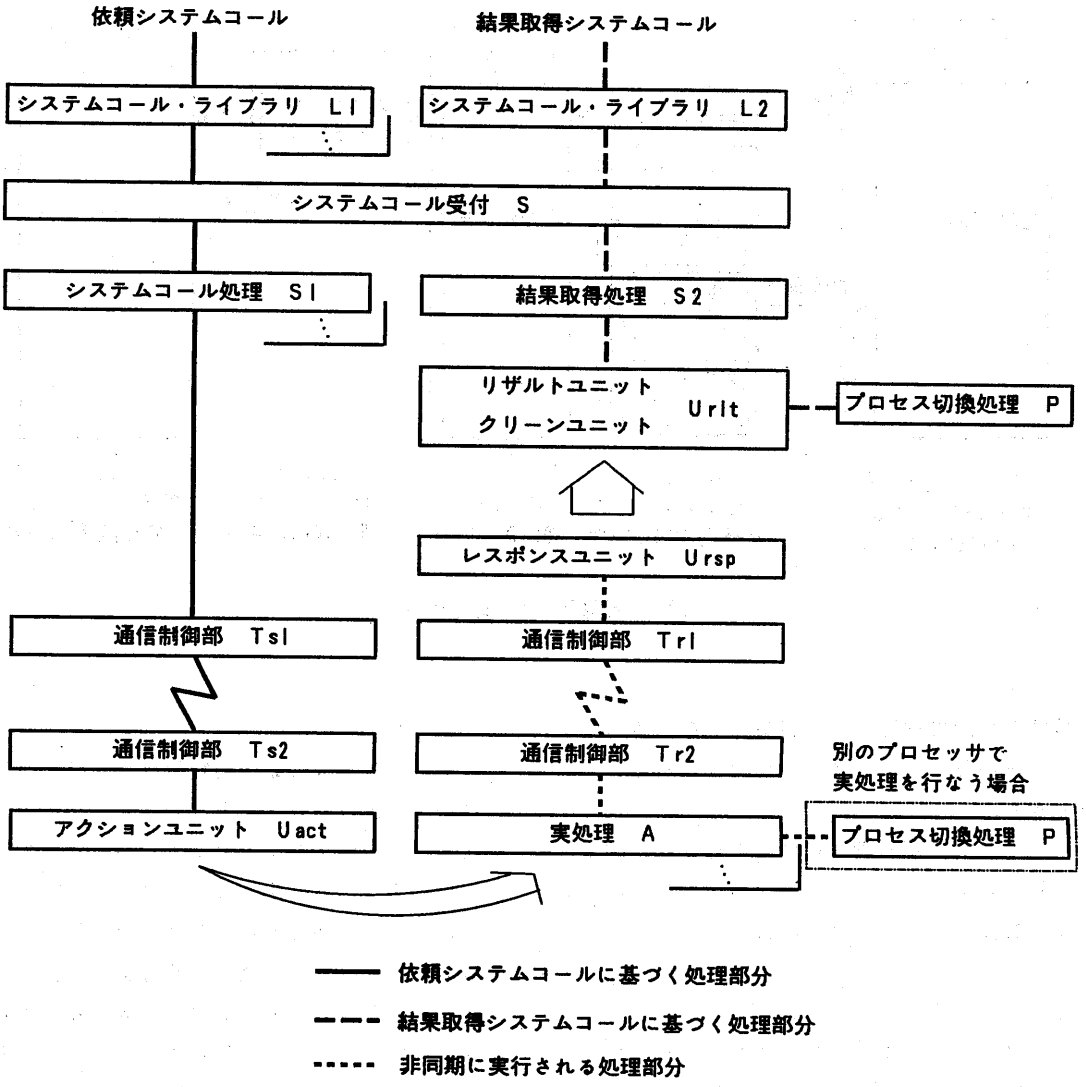


図3　システムコールの処理における部品化の例

図3において、各部品の命令実行数は、次の性質を持つ。システムコールの処理内容に依存する部品は、L1とS1とAである。また、各プロセッサ間を結んでいる通信路の違いに依存する部品は、Ts1とTs2とTr1とTr2の四つである。したがって、実測を必要とする項目の数は、システムコールの処理内容の数とプロセッサ間を結合する通信路の種類の数の和にあたる数でよい。

（2）機能の追加や削除への対処

処理が部品化されているため、OSの機能を実現している処理の追加や削除を行なっても、命令実行数の見積りは容易である。例えば、処理を追加する場合、当該の部品について命令実行数の測定を行なうだけでよい。

（3）複数プロセッサにまたがる処理への対処

4．2節で述べたように、二つのプロセッサにまたがる処理について、各プロセッサの

表１　各処理部分の命令実行数の計算式

場合	分類 / 部分	ＡＰプロセスが走行しているプロセッサ上	実処理を行なうプロセッサ上
一つのプロセッサに閉じた処理	依頼システムコールに基づく処理	Ｌ1＋Ｓ＋Ｓ1＋Ｔs1＋Ｔs2＋Ｕact	－
	結果取得システムコールに基づく処理	Ｌ2＋Ｓ＋Ｓ2＋Ｕrlt＋Ｐ	－
	非同期に実行される処理	Ａ＋Ｔr2＋Ｔr1＋Ｕrsp	－
複数のプロセッサにまたがる処理	依頼システムコールに基づく処理	Ｌ1＋Ｓ＋Ｓ1＋Ｔs1	Ｔs2＋Ｕact
	結果取得システムコールに基づく処理	Ｌ2＋Ｓ＋Ｓ2＋Ｕrlt＋Ｐ	－
	非同期に実行される処理	Ｔr1＋Ｕrsp	Ｐ＋Ａ＋Ｔr2

－：ありえない

命令実行数を見積ることができる。また、図３の実処理において、さらに他プロセッサへの処理が発生し、３プロセッサ以上のまたがるような処理においても各プロセッサの命令実行数を見積ることができる。具体的には、実処理の命令実行数の見積りにおいて、そのまま計算法を拡張すればよい。

（４）自動設計化

４．２節で述べたように、各部品の命令実行数とシステムコールの処理内容による処理流れから、計算式により、各システムコールの命令実行数や非同期に実行される命令実行数および各プロセッサの命令実行数を明らかにできる。そのため、サービスに合わせたシステム構成の設計に向けて、ＡＰ処理やＯＳ処理を各プロセッサへ分散する形態を設計するシステムの構築が可能である。

４．４　見積り例

先に示した命令実行数の計算法を用いて、各プロセッサの負荷を見積り、ＡＰ処理の分散形態を設計する例について述べる。例として、以下の場合を想定する。

［想定］ＡＰの処理は、ファイルに格納されたデータを加工し結果を出力するものである。データは、二つのプロセッサの各々に接続された磁気ディスクに分散して格納されている。

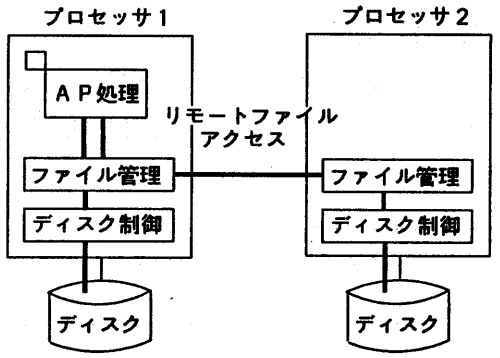
この［想定］における次の二つの処理方式について、各プロセッサの命令実行数を見積る。

［方式１］ＡＰプロセスは一つのプロセッサ上にあり、他プロセッサにあるファイルからのデータ読み出しはリモートファイルアクセス機能を利用する。

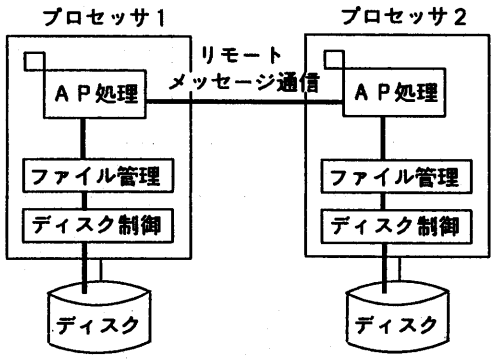
［方式２］ＡＰプロセスは各プロセッサ上にあり、ＡＰプロセス間での結果の授受はリモートメッセージ通信機能を利用する。

各方式の様子を図４に示す。

データ長と命令実行数の関係を図５に示す。図５から、次のことがわかる。



（Ａ）［方式１］



（Ｂ）［方式２］

図４　各処理方式の概要

（１）プロセッサ間の負荷バランスは、［方式２］の方がよい。
（２）結果を送信するメッセージ長が、データ長に関係なく一定であり、かつ短い場合、［方式２］が有効である。

５．むすび

ＤＩＲＯＳでは、ＯＳ処理を部品化し、各部品の命令実行数を明らかにしている。各システムコールの各プロセッサ上での命令実行数は、各部品の命令実行数を処理内容に合わせて組み合わせることにより見積ることができる。このようになっているため、命令実行数を明らかにするための測定項目数が少なく、種々の分散形態に対応できる。

残された課題として、処理内容の変動がある。

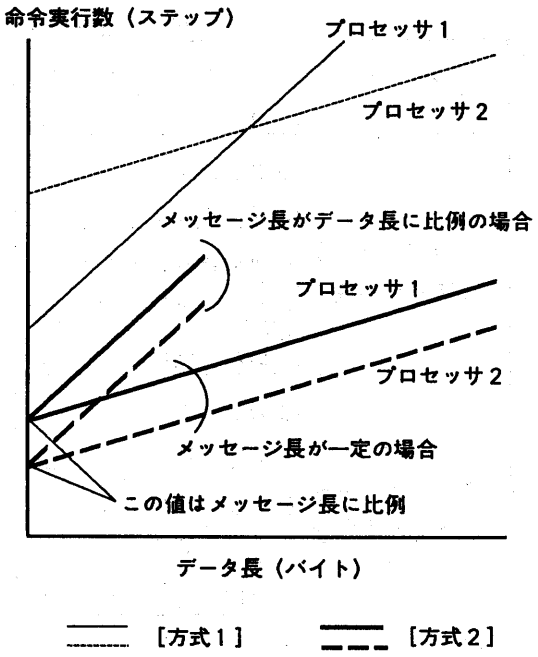


図５　データ長と命令実行数の関係

例えば、メッセージ送信の処理は、既にメッセージ受信のシステムコールが発行されているか否かで処理内容が異なってしまう。処理内容の変動の仕方は明確であるから、各変動の仕方を確率的に扱う方法が考えられる。

〈参考文献〉

[1]M.M.Theimer, et.al., ``Preemptable Remote Execution Facilities for the V-System", Proceedings of the 10th Symposium on Operating System Principles, ACM SIGOPS, 1985.
[2]高野, 田胡, 益田, ``ﾌﾟﾛｾｽ・ﾈｯﾄﾜｰｸによる分散型ｵﾍﾟﾚｰﾃｨﾝｸﾞ・ｼｽﾃﾑ", 情処学論, Vol.29, No.4, pp.359-367(1988).
[3]谷口, ``OS機能の分散を可能にするOS構成法", 信学論, Vol.J72-D-I, No.3, pp.168-174(1989).
[4]谷口, 他, ``分散型ﾘｱﾙﾀｲﾑｵﾍﾟﾚｰﾃｨﾝｸﾞｼｽﾃﾑ:DIROS", 情処研報, 89-OS-45-9(1989).