2V-4

unix*のリアルタイム性向上へのアプローチ[2]

山口 義一　斎藤 正史　伊藤 均　水野 忠則　渡辺 治

三菱電機(株)情報電子研究所

1．はじめに

unixはｴﾝｼﾞﾆｱﾘﾝｸﾞﾜｰｸｽﾃｰｼｮﾝ用のOSとして産業標準の地位を得つつある。しかし、unixは元々TSS向きに開発された為に、ﾜｰｸｽﾃｰｼｮﾝで要求されるﾏﾝﾏｼﾝｲﾝﾀﾗｸｼｮﾝに対する即応性に問題があり、ﾌﾟﾛｾｽ管理方法などの改善が要求されている[1]。更に、ﾌﾟﾛｾｽ管理方法を改善した場合にも応答性向上が阻害される原因として、入出力の処理速度が問題になってくる。ﾜｰｸｽﾃｰｼｮﾝにおいて入出力処理時間が大きい理由としては、ﾏﾙﾁﾒﾃﾞｨｱ処理に伴うﾌｧｲﾙの大容量化に対してunixのﾌｧｲﾙｼｽﾃﾑがそのような大容量ﾌｧｲﾙのｱｸｾｽに向いていないことが挙げられる。

ここでは、unixに対してﾘｱﾙﾀｲﾑ性を向上させることを目的とした、ﾌｧｲﾙｼｽﾃﾑ及びそのｱｸｾｽﾒｿｯﾄﾞの改善方法について述べる。

2．ﾌｧｲﾙｱｸｾｽへの要求

ｲﾒｰｼﾞなどのﾏﾙﾁﾒﾃﾞｨｱ情報を扱う場合に、そのﾃﾞｰﾀ量は従来のｿｰｽﾌﾟﾛｸﾞﾗﾑやﾛｰﾄﾞﾓｼﾞｭｰﾙとは比較にならないほど大きい。例えばA4ｶﾗｰ(ﾌﾟﾚｰﾝ数4枚、線密度16本/mm)のｲﾒｰｼﾞﾃﾞｰﾀでは、約8MBにもなり、ｱｸｾｽ時間が大きくなりがちである。

更にｲﾒｰｼﾞﾃﾞｰﾀは、ｳｨﾝﾄﾞｳﾒﾓﾘ(WMM)に展開してﾋﾞｯﾄﾏｯﾌﾟﾃﾞｨｽﾌﾟﾚｲに表示する方法がよくとられる[2]が、このような場合に、WMMはﾕｰｻﾞ論理空間にﾏｯﾋﾟﾝｸﾞされていないため、ﾃﾞｨｽｸ→ｶｰﾈﾙﾊﾞｯﾌｧ→ﾕｰｻﾞﾊﾞｯﾌｧ→WMMという3度のﾃﾞｰﾀ転送を行なわねばならない。このｲﾒｰｼﾞ表示方法は、ﾃﾞｰﾀ転送回数が多く、効率が悪い。

ﾕｰｻﾞは、このような入出力の間、処理の終了を待つことを強いられ、応答性を高めるには入出力処理の速度を向上させる必要がある。従って、OSは、ｲﾒｰｼﾞﾃﾞｰﾀなどのﾏﾙﾁﾒﾃﾞｨｱに適したより高速なｱｸｾｽﾒｿｯﾄﾞなどの改善が要求されている。

3．改善方法

以下の3項目をﾌｧｲﾙ入出力処理の改善の対象とした。

①ﾌﾞﾛｯｸｻｲｽﾞ可変

unixのﾌｧｲﾙｼｽﾃﾑは1KBのﾌﾞﾛｯｸから構成されているが、これを論理ﾎﾞﾘｭｰﾑ毎に、1KB・2KB・4KBのﾌﾞﾛｯｸｻｲｽﾞにすることを可能にした。これによって、2KB・4KBの論理ﾎﾞﾘｭｰﾑにおける、比較的大きなｻｲｽﾞのﾛｰﾄﾞﾓｼﾞｭｰﾙやﾃﾞｰﾀﾌｧｲﾙにｱｸｾｽするときなどは、ﾃﾞｨｽｸ入出力の起動の回数を減少させることができ、ﾃﾞｨｽｸのｼｰｸ時間・回転待ち時間によるｵｰﾊﾞｰﾍｯﾄﾞを軽減できる。

②raw ﾌｧｲﾙ化

ある論理ﾎﾞﾘｭｰﾑの連続領域にﾌｧｲﾙを構築し(rawﾌｧｲﾙと呼ぶ)、かつ、ｱｸｾｽするためのﾕｰﾃｨﾘﾃｨを作成した。ﾃﾞｨｽｸの起動1回で最大量のｱｸｾｽができるため、①以上の効果がある。

③ｲﾒｰｼﾞ入出力のﾊﾞｲﾊﾟｽ化

ﾕｰｻﾞ論理空間のﾊﾞｯﾌｧを経由せずに、ﾃﾞｨｽｸ→WMMに直接展開する機能及びその逆のWMM→ﾃﾞｨｽｸへ直接書き込む機能を追加した(図1)。これによって、ｲﾒｰｼﾞﾃﾞｰﾀを高速にWMMに展開することができる。

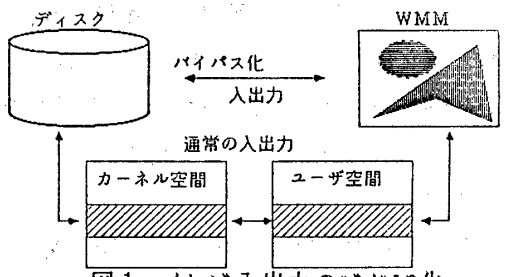

図1．ｲﾒｰｼﾞ入出力のﾊﾞｲﾊﾟｽ化

*: Unix OS was developed by Bell Laboratories and is licenced by AT&T.

An Approach for Realtime Enhancement to Unix OS [2]

Yoshikazu YAMAGUCHI, Masashi SAITO, Hitoshi ITO,
Tadanori MIZUNO, Osamu WATANABE
Mitsubishi Electric Corp.

4．測定結果及び考察

ここでは、前章①②③の改善項目に対する性能測定結果とそれらの利点を活かしたﾌｧｲﾙｼｽﾃﾑの運用方法について述べる。

①ﾌﾞﾛｯｸｻｲｽﾞ可変

論理ﾎﾞﾘｭｰﾑ単位でﾌﾞﾛｯｸｻｲｽﾞを1･2･4KBに設定できるよう改造し、ﾌｧｲﾙのread/writeの時間を測定した（表１）。ﾌﾞﾛｯｸｻｲｽﾞを大きくすることによるｱｸｾｽ時間の短縮の効果は大きい。

表１．アクセス速度の比較

ﾌﾞﾛｯｸｻｲｽﾞ	1KB	2KB	4KB
read	20.5	9.7	5.7
write	27.1	22.8	13.7

(msec/KB)

この方法は、4KBの論理ﾎﾞﾘｭｰﾑなどは、断片化が起きやすいなどの欠点があるが、OS及びﾌｧｲﾙｼｽﾃﾑを構築するためのﾕｰﾃｨﾘﾃｨの改造は少ないだけでなく、従来のｿﾌﾄｳｪｱ資産を受け継ぎやすいなどの利点を持つ。

②raw ﾌｧｲﾙ化

ﾌﾟﾛｸﾞﾗﾑﾛｰﾃﾞｨﾝｸﾞ時の速度を通常のﾌｧｲﾙの場合とrawﾌｧｲﾙ化したﾌｧｲﾙの場合で比較したところ、通常ﾌｧｲﾙでは35msec/KBに対し、rawﾌｧｲﾙでは、2msec/KBとなった。

この方法は、前項①よりも更に高速なﾌｧｲﾙｱｸｾｽが実現できるが、unixのﾌｧｲﾙ管理の持つ柔軟性を失うため、ﾌｧｲﾙｼｽﾃﾑの構築及び管理が困難になる。従って、rawﾌｧｲﾙ化は、ｼｽﾃﾑで提供されているｺﾏﾝﾄﾞのﾛｰﾄﾞﾓｼﾞｭｰﾙ（ex. vi sh）などの読み出し専用的な性質を持ったﾌｧｲﾙに適用するのが有効である。

③ｲﾒｰｼﾞ入出力のﾊﾞｲﾊﾟｽ化

ｲﾒｰｼﾞﾃﾞｰﾀﾌｧｲﾙからｳｲﾝﾄﾞｳﾒﾓﾘへの展開、及びｳｲﾝﾄﾞｳﾒﾓﾘからｲﾒｰｼﾞﾃﾞｰﾀﾌｧｲﾙへの格納動作を通常の入出力で行なった場合と、ﾊﾞｲﾊﾟｽ化入出力で行なった場合をそれぞれの1･2･4KBのﾌｧｲﾙｼｽﾃﾑについて比較した（図２）。

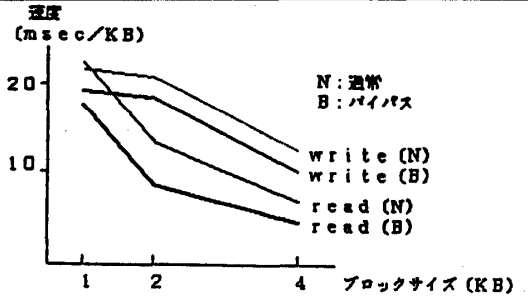

図２．バイパス化入出力の効果

この結果から見ると、単なるﾊﾞｲﾊﾟｽ化I/Oだけでは、効果が出ず、OS内のread-ahead、delay-writeとの関連から4KBのﾌﾞﾛｯｸｻｲｽﾞのﾌｧｲﾙからのreadについての効果が顕著にでることがわかる（1KBのﾌｧｲﾙｼｽﾃﾑでは２割減に対し、4KBのﾌｧｲﾙｼｽﾃﾑでは５割減）。

④ﾌｧｲﾙｼｽﾃﾑの運用

前記３種の方法を組み合わせて、ﾌｧｲﾙの用途別に使い分けることにより、ﾕｰｻﾞに対する応答性を向上させることができる。また、ｼｽﾃﾑ全体のｽﾙｰﾌﾟｯﾄの向上も期待できる。

以上のようなﾌｧｲﾙｱｸｾｽの高速化を図った場合の効率的なﾌｧｲﾙｼｽﾃﾑの構成例を図３に示す。

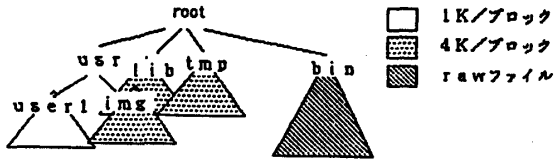

図３．ファイルシステム構成例

5．まとめ

ﾕｰｻﾞに対する応答性を向上させるために、ﾌｧｲﾙｱｸｾｽに対する改善を行ない、以下のことが実現できた。

（１）ｲﾒｰｼﾞﾃﾞｰﾀの高速ｱｸｾｽ／表示

（２）使用頻度の高いｺﾏﾝﾄﾞの高速ﾛｰﾃﾞｨﾝｸﾞ

今後は、ﾘｱﾙﾀｲﾑ性のより一層の強化・ﾌｧｲﾙｼｽﾃﾑの離散化、断片化等の問題点・使い勝手などについて考察・検討を加え、実使用を通し提案した方法の検証を行なっていく予定である。

参考文献

[1]斎藤,他:unixのﾘｱﾙﾀｲﾑ性向上のｱﾌﾟﾛｰﾁ[1],
情報処理学会第33回全国大会,1986

[2]清水,他：ﾏﾙﾁｳｲﾝﾄﾞｳ表示方式,
ﾏｲｸﾛｺﾝﾋﾟｭｰﾀ研究会 35-1,1985