

拡張入出力制御方式XVIO(フェーズI)の開発

新井利明 木下俊之 吉澤康文 久保隆重

(日立製作所システム開発研究所)

1. はじめに

半導体技術の進歩により高速で大容量の主記憶が利用可能となっている。一方、主記憶と2次記憶とのアクセスギャップは依然大きく、会話処理において性能のネックになることが多い。拡張入出力制御機構(XVIO: eXtended Virtual Input/Output)は、利用可能な大容量主記憶を用いて物理的入出力を削減し、アクセスギャップの問題を解決する機能である。

XVIOは、高速な主記憶装置、大容量の2次記憶装置など性質の異なる複数の記憶媒体から成る記憶階層を、1つの仮想ファイルとみなして入出力を行う。仮想ファイル中のデータに対してアクセス要求があった場合に、そのデータが主記憶上に配置されていれば入出力は主記憶間のデータ転送に置換えることができる。このような主記憶のヒット率向上のため、XVIOでは仮想ファイル中のデータをそのアクセス状態に応じて記憶階層上を移動させ、システム全体の入出力時間の削減を図る。XVIOの目的は以下の通りである。

- (1) 利用可能となった大容量の主記憶装置を用いて入出力を仮想化し、物理的入出力回数の削減を図りアクセスギャップを解消する。
- (2) 多様化すると予想される記憶階層を統括的に一元管理し、新規記憶階層に対し柔軟に対処する。

我々は、XVIOのフェーズIとして、対象データセットをプログラムライブラリに限定した高速プログラムローディング機構を開発した。ここでは、このXVIOフェーズI(以下XVIOという)について報告する。

2. XVIOの概要

2.1 XVIOの目的

XVIOはプログラムの高速ローディング機構である。従来、プログラムは2次記憶上の恒久データセット上に置かれていたため、その実行には物理的入出力動作が必要であった。そのため、オンラインやTSSではコマンド処理時間に比べて処理プログラムのローディング時間が多くなり、応答上のネックとなることが多かった。この欠点を補う手段として、プログラムの仮想記憶常駐/主記憶常駐方式が一般的に採用されている。これらの手法は物理的入出力回数を削減するという点では効果的であるが、以下のような欠点も存在する。

- (1) 仮想空間上にローディングされた1つのプログラムをを複数のユーザで共用するため、対象となるプログラムはリエントラント属性を持つものに限られる。
- (2) プログラムは共通領域に格納しなければならないため、対象となるプログラムが多くなると、ユーザのジョブ固有領域を圧迫する。
- (3) プログラムが主記憶上に存在しない場合には、プログラムがページをまたぐ度にページイン動作が必要になり、反って性能が劣化する。

(4) 常駐すべきプログラムの組み合わせはIPL時に決定されるため、システムの負荷変動に追従できない。このように、あらかじめプログラムをローディングしておき、その後の制御はページング処理にたよる方式は、制約も多く、性能向上のためのきめ細かな制御も不可能である。XVIOは、プログラムローディングのための入出力を仮想化してローディング時間の短縮をはかるものである。XVIOは、主記憶装置と2次記憶装置で構築され階層構造を持つ仮想ライブラリ中にプログラムを保持し、仮想ライブラリアクセス法を用いてプログラムをユーザの仮想空間にロードする。そのため、上記の欠点は以下のように改善される。

- (1) プログラムライブラリからのローディングを高速化しているため、どのような属性のプログラムに対しても高速ローディングが可能である。
- (2) 仮想ライブラリはXVIO空間上に常駐するため、ユーザの仮想空間を圧迫することがない。
- (3) プログラム単位に記憶階層上の位置を設定するため、2次記憶装置からの読み込みが必要な場合にも入出力の発行を最小限に抑えることができる。
- (4) 各プログラムのアクセス状況を監視して主記憶上に配置すべきプログラムを決定するため、負荷の変動に追従したプログラムの高速化が可能である。

XVIOは、プログラムローディングのための入出力要求の90%以上を仮想化して、オンラインやTSSなどの性能向上を図ることを目的とする。

2.2 XVIOの機能概要

XVIOの概念を図1に示す。本機構は以下の機能で達成される。

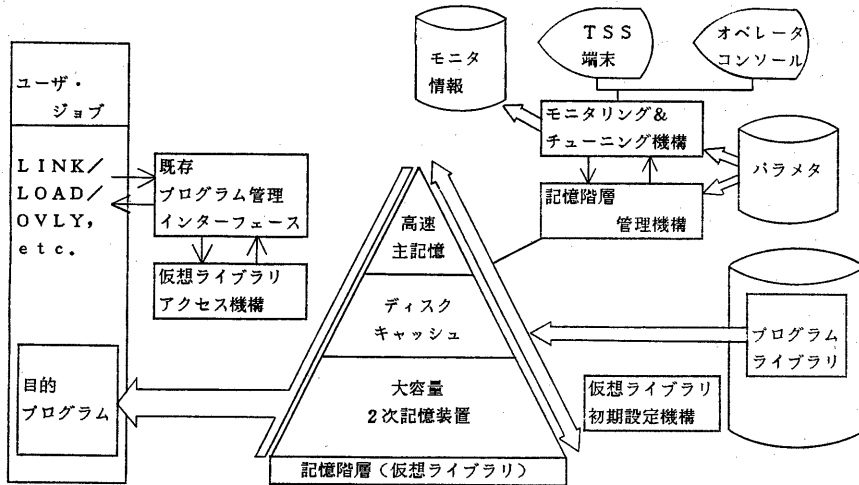


図1. XVIOフェーズIの概要

(1) 仮想ライブラリ初期設定機能

恒久データセット上のプログラムライブラリを、階層構造を持つ仮想ライブラリに取り込むものである。XVIOの初期設定時、プログラムの入れ替え時に起動される。

(2) 仮想ライブラリのアクセス機能

仮想ライブラリ中に配置されたプログラムをユーザ空間にローディングする機能である。要求されたプログラムがXVIO機構の対象であるか否かを判定するサーチ機能と、記憶階層中のプログラムを転送するロード機能からなる。

(3) 仮想ライブラリの記憶階層管理機能

仮想ライブラリ中のプログラムの記憶階層上での位置を決定し、配置する。システム全体の入出力時間が最小となるよう、アクセス頻度の高いプログラムをより高速な記憶媒体上に配置する。

(4) モニタリング&チューニング機能

XVIOの稼動状況を監視して出力するモニタリング機能と、より効果的に実行をさせるためのチューニング機能とからなる。また、XVIO機能を停止させることなく仮想ライブラリの入れ替えを行う。

3. 仮想ライブラリ

3.1 仮想ライブラリの論理構造

仮想ライブラリは、階層構造を持つ仮想的なファイルであり、XVIO機構によるローディング対象となるプログラムが格納されている。仮想ライブラリはXVIO起動時に初期設定され、プログラムのローディングに必要な情報を保持する。仮想ライブラリの論理的構造を図2に示す。

- (1) ディレクトリ部 …… プログラム単位に作成され、プログラムの記憶階層上の位置、データ部

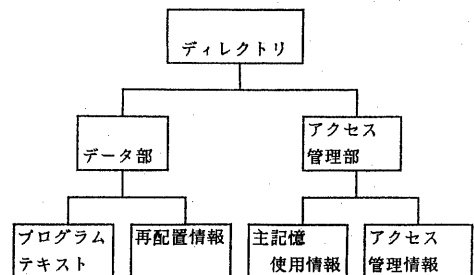


図2. 仮想ファイルの論理構造

／アクセス管理部へのポインタを保持する。アクセス時間短縮のためハッシングされている。

- (2) データ部 …………… ローディングされるプログラムの内容を保持する。ローディングの際のCPUオーバヘッド削減のため、テキスト部と再配置情報部を形式変換してある。
- (3) アクセス管理部 …… プログラムのアクセス状況などの情報を保持する。同一の記憶階層に属するプログラム毎にチェイニングされている。

プログラムは、プログラム名とデータセット名で識別される。同一のプログラムに対して複数の名称(別名)が割り当てられている場合には、その各々に対してディレクトリ部が作成され、同一のデータ部、アクセス管理部をポイントすることにより、一つの実体を共用する。オーバレイ構造を持つプログラムに対しては、オーバレイセグメント単位にディレクトリが作成される。

3.2 仮想ライブラリの物理構造

仮想ライブラリはXVIO空間に展開される。仮想ライブラリのうちアクセス頻度の高いディレクトリ部やアクセス管理部は主記憶に常駐し、容量の大きなデータ部は仮想記憶に配置して必要に応じて主記憶に取り込む。

図3に仮想ライブラリの物理構造を示す。

XVCB(XVio Control Block)はXVIOの最も基礎となる制御テーブルであり、XVIO機構を使用する全てのジョブから参照可能なように共通領域に置かれる。XVCBはディレクトリやアクセス管理部へのポインタを保持する。ディレクトリはプログラム名でハッシングされ、主記憶上に常駐される。アクセス管理部は同一の属性を持つプログラム同志でチェイニングされそのアンカーをXVCBが保持している。データ部は大容量であるためVIO用ページデータセット上に配置する。そして、そのプログラムのアクセス頻度に応じて、記憶階層管理機構が主記憶中に取り込み、入出力時間の短縮を図る。

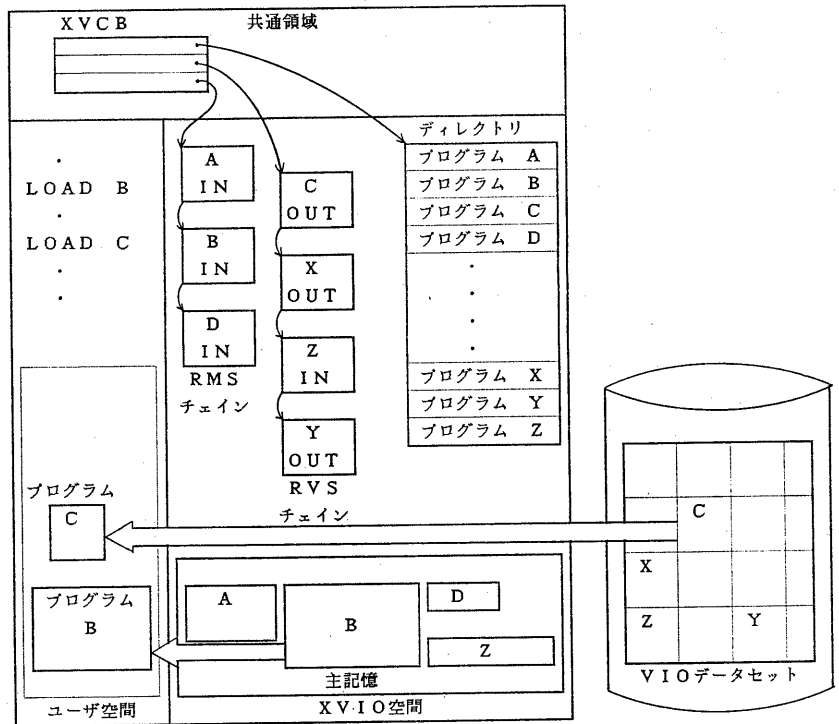


図3. 仮想ファイルの物理構造

3.3 仮想ライブラリアクセス法

仮想ライブラリアクセス法は仮想ライブラリローディング機構であり、従来のプログラムローディング機構と同一のインターフェースを持つ。以下の機能からなる。

(1) サーチ機能

サーチ機能はプログラムがXVIOの対象となっているか否かの判定を行う。サーチ機能は従来のプログラムローディング処理に先立って無条件に実行されるため、ユーザはプログラムを修正することなくXVIO機構を使用することができる。プログラムの識別は、プログラム名、データセット名、およびセグメントI.D.で行う。まず、プログラム名をハッシングし対応するディレクトリが存在するか否かを判定する。ディレクトリが発見出来ず、XVIOの対象でないと判定されプログラムに対しては、従来のプログラムローディング処理を続行する。

(2) ローディング機能

ローディング機能は、仮想ライブラリ中のプログラムを要求を出した空間に取り込み実行可能な形式とする。プログラムの記憶階層上での位置などのローディングに必要な情報は、サーチ機能によって発見されたディレクトリ中に記述されており、これに従って個々の記憶階層専用のデータ転送ルーチンが起動される。

(a) VIOデータセットアクセス機能 …… VIOデータセット上に格納されているデータをユーザ空間に取り込む。

(b) 空間間コピー機能 …… XVIO空間上の主記憶上に存在するデータを他空間に転送する。

上記の機能によってユーザ空間に取り込まれたプログラムを、アドレス定数の再配置を行い実行可能な形式とする。

(3) モニタ機能

プログラムのアクセス頻度や主記憶のヒット回数などの情報を収集する。XVIOの対象となっているプログラムだけでなく、非対象のプログラムに対しても監視を行い、チューニングの手助けとする。

図4に仮想ライブラリアクセス法の制御の流れを示す。

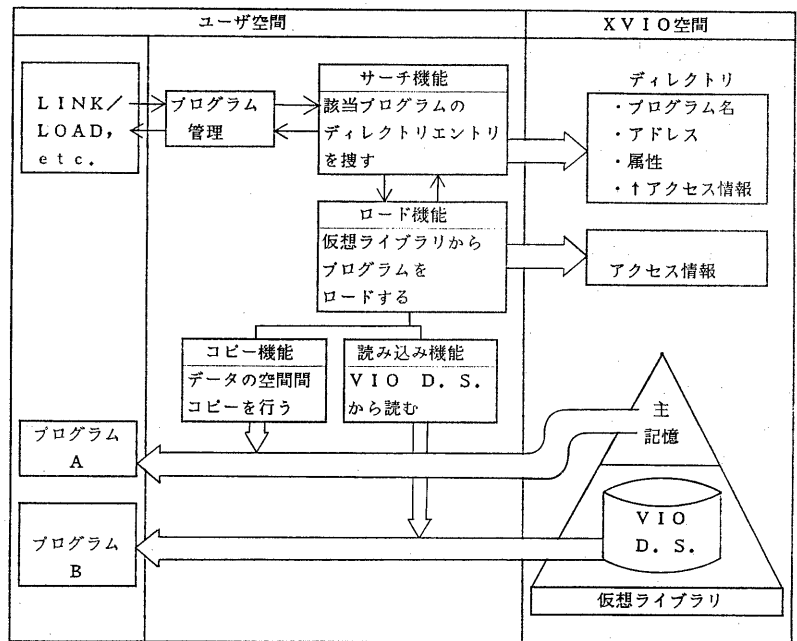


図4 仮想ライブラリアクセス法の構成

3.4 仮想ライブラリの初期設定

XVIOはジョブの1つとして、仮想空間を占有して動作する。XVIOの対象となるプログラムは、XVIOの初期設定時に仮想ライブラリに取り込まれる。仮想ライブラリの初期設定後は、プログラムのローディングはすべて仮想ライブラリから行われる。

XVIOの対象となるプログラムの指定は、XVIO起動時のJCL(DD文)、およびパラメタで行う。パラ

メタには、専用のデータセットが用意されている。XVIOでは対象となるプログラムに二つの属性を設けている。一つは、プログラムを主記憶上に常に配置しておくものであり、これをRMS(Resident in Main Storage)属性という。一方のRVS(Resident in Virtual Storage)は、仮想記憶上(VIOページデータセット上)に置かれ、必要に応じて主記憶上に取り込まれる。これらの属性は、XVIO初期設定時のパラメタ、およびXVIO稼働時のコンソールコマンド、TSSコマンドを用いて設定/再設定される。図5に仮想ライブラリの初期設定手順を示す。

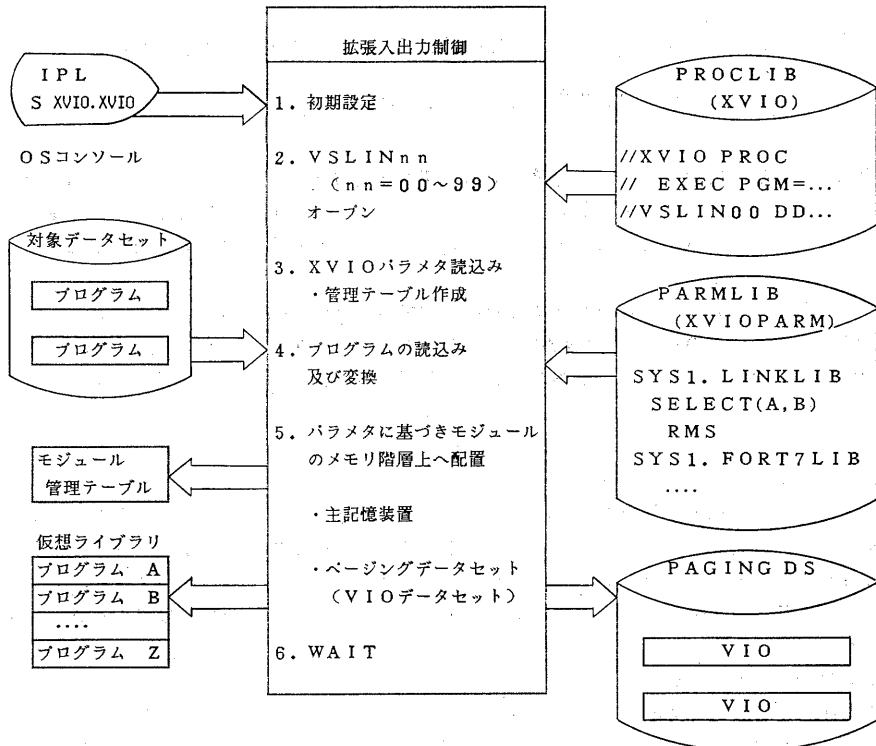


図5 仮想ライブラリの初期設定手順

4. 記憶階層管理(SHM:Storage Hierarchy Management)

XVIOは、プログラムのロード要求に対し、データの転送待ち時間を最小化することを目的としている。そのため、ロード要求回数の多いプログラムほど、高速な記憶媒体である主記憶上に置く必要がある。記憶階層管理機構は、モニタ機能が収集した情報を基に、仮想ライブラリ中のプログラムの再配置を行う。配置すべき位置の決定には、最近のアクセス頻度とプログラムの大きさを考慮する。手順は以下の通りである。

(1) SHM周期の決定

現在までの時刻をあるインターバルで区切り、各プログラムに対して最後にロードされたインターバルを記録する。

(2) RVSプログラムの優先順位の決定

RVS属性を持つプログラムに対して、高速な記憶媒体に配置するための評価基準を作る。(図6)

- (a) 最近のインターバルで参照されたものほど優先度が高い。
- (b) インターバルが同一であれば、最後のインターバル内で生じたロード回数(r)とプログラムサイズ(s)

s)を用いて、 r/s の大きいものほど優先度が高い。

(3) 上記の結果から優先度の高いものから主記憶上に配置する。

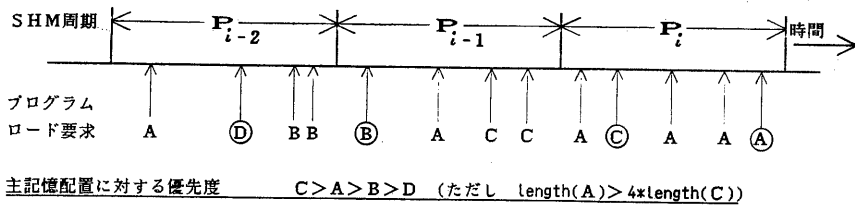


図6 RVS属性を持つプログラムの主記憶配置に対する優先度

ここでの問題は主記憶の配分である。主記憶は最も高速な記憶媒体であるため、XVIOの効率向上のためには、より多くのプログラムを主記憶上に配置した方がよい。しかし、一方、主記憶はジョブの実行には不可欠なものであり、不足した場合には不必要なページングやスワッピングが多発し反って入出力が増加しシステムの性能低下を招く。

そこで、XVIOでは、XVIO空間が使用可能な主記憶量の範囲をあらかじめパラメタとして指定することとした。記憶階層管理は、この指定された範囲内で優先度の高いものから主記憶に配置する。将来はRCM(Resource Centralized Manager)がシステム全体の観点からXVIOが使用すべき主記憶量を決定するよう改善していく方針である。

5. モニタリング&チューニング機能

XVIOは、稼動状況の把握や効果確認のためのモニタリング機能と、モニタした結果をシステムに反映させるためのチューニング機能を有する。モニタリング機能はXVIOの稼動状況を収集し表示する。オペレータコンソールからのMODIFYコマンドやTSS端末からのXVIOコマンドのサブコマンドにより、各プログラムのローディング頻度、主記憶ヒット率などをディスプレイする。また、長期的な監視のため、ログファイルに情報を出力する。

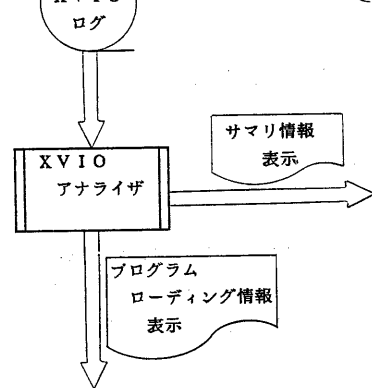
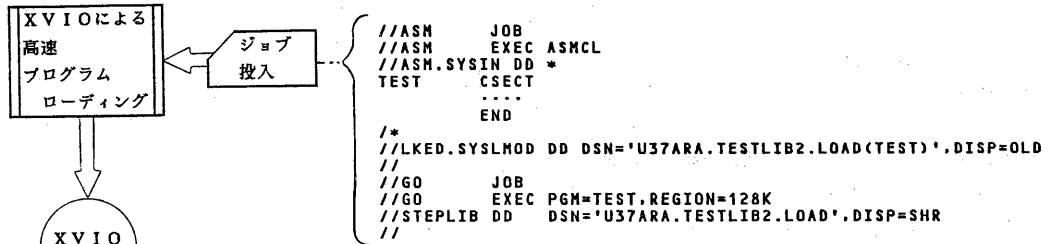
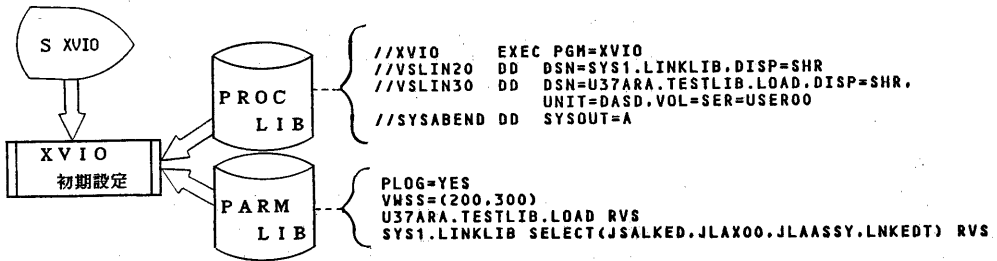
チューニング機能は、XVIOをより効率良く動作させるための機能であり、モニタリング機能と同様にオペレータコンソールやTSS端末から実行される。XVIOの対象となるプログラムの変更、プログラム属性の変更などを行う。また、XVIOの機能を停止せずに対象となるプログラムの入れ替えを行う機能、XVIOを一時的に停止させる機能など、使い易さを考慮した機能も備えている。表1に各コマンドの一覧を示す。

表1 XVIOコマンド一覧

NO	コマンド名	機能
1	REMOVE	プログラムをXVIOの対象外とする
2	CHANGE	プログラムの記憶階層属性を変更する
3	DEFINE	プログラムをXVIOの対象とする
4	REFRESH	XVIOの対象から外し、再度データセットから読み込む
5	HALT	一時的にXVIOサービスを停止する
6	RELEASE	HALT状態を解除し、XVIOサービスを再開する
7	DISPLAY	XVIOの稼動状態を表示する
8	SET	XVIOサービスに割り付ける主記憶の上下限を設定する

6. XVIOの使用例

XVIOは、すべてのジョブ、すべてのプログラムから使用することができる。XVIOが起動され、仮想ライブラリが初期設定された後は、すべてのユーザがXVIO機構を用いてプログラムのローディングを行うことができる。この機構を使用するためのユーザプログラムの修正は一切不用である。



NO.	-MODULE-	ATR	CUMM-LOAD	CUMM-HIT
1	JLAXOO	RVS	7	6
2	LNKEDT	RVS	4	3
3	LNKEDT	RVS	4	3
4	LNKEDT	RVS	4	3
5	LNKEDT	RVS	4	3
6	LNKEDT	RVS	4	3
7	LNKEDT	RVS	4	3
8	LNKEDT	RVS	3	2
9	JSALKED	RVS	4	3
10	JLAASSY	RVS	7	6
11	JSALKED	RVS	4	3
12	JSALKED	RVS	4	3
13	JSALKED	RVS	4	3
14	JSALKED	RVS	3	2
15	JSALKED	RVS	4	3
16	JSALKED	RVS	4	3

<<< PROGRAM LOADING REQUEST LOGS >>>

	-MODULE-	ATR	VSN	JOB-NAME	STAT	SIZE	OVL	LNK	CONCAT	DATA SET
10:07:40.64	JLAXOO	RVS	20	ASM	IN	2920		YES		SYS1.LINKLIB
10:07:40.66	JLAXO1			ASM				YES		
10:07:40.71	JLAXO2			ASM				YES		
10:07:40.82	JLAY10			ASM				YES		
10:07:41.08	JLAXO3			ASM				YES		
10:07:41.11	JLAXO4			ASM				YES		
10:07:41.34	JLAX11			ASM				YES		
10:07:41.61	JLAX21			ASM				YES		
10:07:41.72	JLAX31			ASM				YES		
10:07:41.91	JLAXO5			ASM				YES		
10:07:42.01	JLAXO6			ASM				YES		
10:07:42.27	JLAX41			ASM				YES		
10:07:42.53	JLAX51			ASM				YES		
10:07:42.91	JLAX61			ASM				YES		
10:07:43.00	JLAX62			ASM				YES		
10:07:44.62	JSALKED	RVS	20	ASM	IN	29280		YES		SYS1.LINKLIB
10:07:44.65	JSALKED	RVS	20	ASM	IN	29280	2	YES		SYS1.LINKLIB
10:07:45.15	JSALKED	RVS	20	ASM	IN	29280	3	YES		SYS1.LINKLIB
10:07:45.29	JSALKED	RVS	20	ASM	IN	29280	5	YES		SYS1.LINKLIB
10:07:45.32	JSALKED	RVS	20	ASM	IN	29280	4	YES		SYS1.LINKLIB
10:07:45.51	JSALKED	RVS	20	ASM	IN	29280	6	YES		SYS1.LINKLIB
10:07:45.61	JSALKED	RVS	20	ASM	IN	29280	7	YES		SYS1.LINKLIB
10:07:47.44	TEST			ASM						U37ARA.TESTLIB2.LOAD

図7 XВИОの使用例

図7にXVIOの使用例を示す。この例では、ユーザのライブラリのすべてと、システムのライブラリであるリンクライブラリ中のアセンブラの一部とリンケージエディタが、XVIOのRVS属性として定義されている。

ここに、アセンブル、リンク、ゴージャブを投入すると、アセンブラの指定されたプログラムと、オーバーレイ構造を持つリンケージエディタのすべてのセグメントが仮想ライブラリからローディングされる。XVIOの稼働情報はロギングファイルに出力され、アナライザによって解析される。

7. 効果の予測

XVIOのプログラムローディング高速化機能は、すべての分野で利用可能である。特に、入出力操作が性能上の大きな要因となっているTSS、オンラインなどのシステムでは効果的であると思われる。

XVIOの効果を予測するため、2つのシステムで入出力発行状況の測定を行った。1つはTSS+バッチの大規模システムであり、他方は大規模オンラインシステムである。各データセットへの入出力操作の回数を表2表3に示す。どちらの場合にも、プログラムローディングは全入出力の20%以上を占めている。一方、プログラムライブラリは比較的容量が小さく、入出力が集中しやすいため、頻りにアクセスされるプログラムの総量は8MB未満である。以上のことから、XVIO機構により20~25%の入出力削減が予想され、特に高トラフィック時の応答時間の減少が期待できる。

8. おわりに

本報告では、利用可能となった大容量の主記憶を利用して入出力の削減を図る拡張入出力制御機構(XVIO)のフェーズIとして、対象をプログラムライブラリに限定してプログラムローディングの高速化機構を開発した。

本機能の使用により、ユーザ・プログラムの修正なしに物理的入出力の20~25%削減が可能であり、TSSおよびオンライン処理の応答時間の短縮が期待できる。

表2 TSS&バッチシステムにおける入出力の発行の割合(%)

データセットの種類	アクセス頻度
プログラム・ライブラリ(システム提供)	20.4
プログラム・ライブラリ(センタ提供)	1.3
ユーザ・データセット	37.3
ワーク・データセット	14.4
その他	26.6

(ただし、ページングを除く)

表3 オンライン・システムにおける入出力の発行の割合(%)

データセットの種類	アクセス頻度
データベース	53.6
プログラム・ライブラリ	26.0
ジャーナル	4.2
その他	16.2

(ただし、ページングを除く)