システムソフトウェアと
オペレーティング・システム 91－9
（2002．8．21）

高信頼化を考慮したシングルイメージ分散マイクロカーネルの実現

前田 誠司　佐藤 記代子　崎山 伸夫　矢野 浩邦　林 拓也†

本論文では，情報システムの安定運用を容易に実現するシステム仮想化手法 Single Machine Image(SMI) を提案する．SMI とは，シングルシステムイメージが提供する分散環境のシングルシステムへの仮想化と，データ複製等の高信頼化に必要なハードウェア仮想化を，一つに統合したシステム仮想化手法である．筆者らは，この SMI をマイクロカーネルに組み込んだ SMI Kernel の実装を進めており，その概要を解説する．

Design and Implementation of Single Image Distributed Micro Kernel for High Availability

Seiji Maeda, Kiyoko Sato, Nobuo Sakiyama, Hirokuni Yano and Takuya Hayashi††

In this paper, we propose the Single Machine Image as a system virtualization method for keeping information systems stable easily. SMI can provide a virtualization of single image like Single System Image and a virtualization of hardware for high availability such as data replication simultaneously. We also describe a current implementation of SMI Kernel which embeds SMI in its micro kernel.

1　はじめに

情報システムは，社内システムや企業間電子取引に加え，インターネットを介したエンドユーザへのサービス提供など，その用途は広がり続けている．しかし，その一方で，情報システムを構成する計算機のハードウエア故障，あるいは，オペレータの操作ミスに起因するシステム停止事故は後を絶たない．筆者らは，これらの現状を踏まえ，情報システムの安定運用を容易に実現するシステム仮想化手法 Single Machine Image(SMI) およびその実装方式である SMI Kernel の研究を行っている．

SMI とは，シングルシステムイメージが実現する分散環境のシングルシステムへの仮想化と，データ複製等の高信頼化に必要なハードウェア仮想化を，一つに統合したシステム仮想化手法である．SMI Kernel は，この SMI をマイクロカーネル内に実装することにより，ハードウェア故障および情報システムの運用・保守の容易化や動的拡張など，システムの柔軟性を効率よく実現することをマイクロカーネルにおいて可能とする．

本稿では，2 章で，情報システムの課題を明らかにする．3 章で，システム仮想化技術である SMI と他の仮想化技術の比較を行い，続いて，計算機リソースの仮想化および高信頼化機能について提案する．次に，4 章で，SMI をマイクロカーネルに組み込んだ，SMI Kernel のソフトウェアモジュールの構成を解説し，最後に，本研究の課題，関連研究の紹介，および，まとめについて述べる．

2　情報システムの課題

従来の情報システムでは，計算機の構成とソフトウェアの構成とが 1 対 1 で対応しているものが多い．例えば，個々の計算機毎にオペレーティングシステム (OS) をインストールし，ネットワーク設定などを個別に設定する場合がある．また，新規に計算機を追加する場合，関連する設定をシステム全体にわたって変更する必要があったり，ソフトウェア構成の動的な変更が出来ない場合な

†(株) 東芝 研究開発センター
††Corporate Research and Development Center, Toshiba Corp.

どがある．そのため，稼働中の情報システムの構成を柔軟に変更することは難しい．既存システムの処理能力増強のため，あるいは，新規システムの処理能力未達時には，新しい計算機をシステムに追加し処理能力を補うことが必要であるが，その実現は難しいと言える．

また，OS 等のシステムソフトウェアの中には，システムディスクの故障等のハードウェア故障への配慮なしに設計されたものも多く，その場合，ハードウェアによる高信頼化が必要となってしまう．あるいは，ハードウェア故障への対応をミドルウェアあるいはアプリケーションに委ねており，構成に応じて個別に対応しなければならないという問題がある．

筆者らは，システムソフトウェアにおいて，システム構成を隠蔽しシングルイメージを実現する事と同時に，ハードウェア故障を隠蔽する事によって，これら情報システムの課題を解決する．

3 Single Machine Image

Single Machine Image(SMI) とは，シングルシステムイメージ (SSI) が提供する分散環境のシングルシステムへの仮想化と，データ複製等の高信頼化に必要なハードウェア仮想化を，一つに統合したシステム仮想化手法である．

3.1 従来のシステム仮想化技術

3.1.1 シングルシステムイメージ

図 1 に SSI によるシステム仮想化のイメージ図を示す．

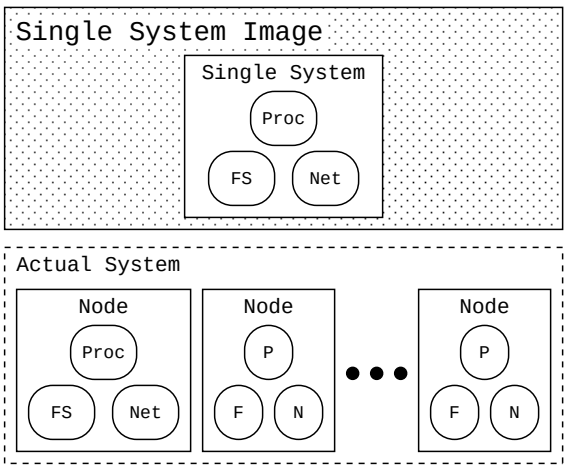

図 1: SSI によるシステム仮想化

SSI では，複数ノードに分散したプロセス・ファイル・ネットワーク等を，あたかも一つのシステム上で動作しているかのように見せる方式である．これは，プロセス・ファイル・ネットワーク等の ID 空間 (ネームスペース) を一つにするものであり，一般に，ハードウェア故障に関するサポートは無い．また，ベースとした OS の機能をシングルイメージ化するものであり，ベース OS と異なる性質の機能を提供することは難しい．

3.1.2 ハードウェア高信頼化

図 2 に RAID によるハードウェア高信頼化のイメージ図を示す．

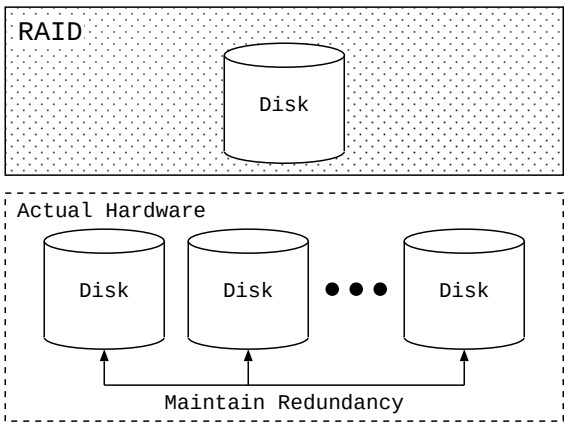

図 2: RAID によるハードウェア高信頼化

ハードウェアの高信頼化の多くは，ハードウェアの仮想化によって実現されている．具体的には，複数個のハードウェアを単一ハードウェアとして仮想化し，複数ハードウェア間で冗長性を維持することによって，高信頼化を実現する．例えば，RAID では，複数のハードディスクをあたかも 1 台のハードディスクであるかのように仮想化し，内部のハードディスク間でデータの冗長性を維持することによって，高信頼化を実現する．

この様に，個々のハードウェア，例えばハードディスクと NIC は，別々に高信頼性を維持管理する．

3.1.3 SMI によるシステム仮想化

これらの仮想化手法に対し，SMI は，分散システムのシングルシステムへの仮想化とハードウェアの仮想化を統合する．図 3 に SMI による仮想化のイメージ図を示す．

SMI では，SSI と同様に，複数ノードに分散したプロ

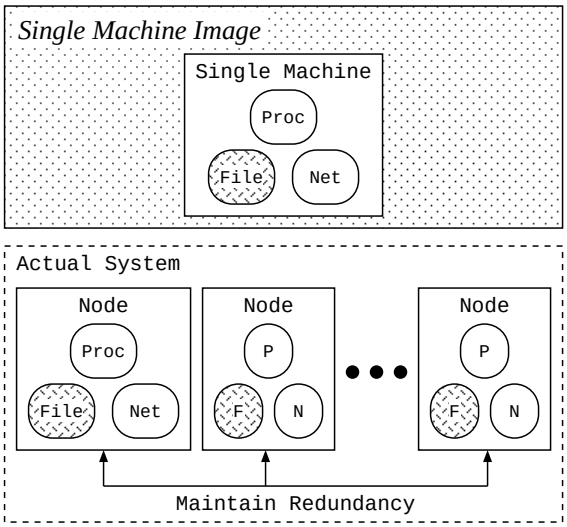

図 3: SMI によるシステム仮想化

セス・ファイル・ネットワーク等を，あたかも一つのシステム上で動作しているかのように見せる．しかし，SSI はネームスペースを一つにするだけであることに対し，SMI では，さらにノード間でデータの冗長度維持等を行うことによって，同時に高信頼化も実現する．この様にすれば，シングルイメージを形成する処理と高信頼化に必要な処理等を効率的に実装することが可能となる．

SMI 上のプロセスは，シングルイメージによって，お互いの位置を意識すること無く，他のプロセスやファイルおよびネットワークにアクセスすることができる．ファイルは，システム内の複数のノード上に複製することによって，高信頼化を実現する．

例えば，ハードディスクに故障が発生した場合，失われたファイルに対し，その複製を用いて他のハードディスク上に新しい複製を生成する．この際，実際のファイルの位置は変化するが，シングルイメージによって位置の変化を隠蔽する．この様に，シングルイメージによる位置仮想化と，高信頼化のためのハードウェア仮想化を同時に用いることによって，効率的にハードウェア故障を隠蔽することが出来る．

以降では，始めに，SMI で提供する計算機リソースおよびその仮想化方式を解説し，次に，高信頼化機能について解説する．

3.2 計算機リソース

SMI では，実ハードウェア構成的には分散している計算機をシングルイメージに見せる単位として，計算機の基本的な機能単位である計算機リソースを用いる．現在，提供する計算機リソースとして，以下の 3 種を提供する．

プロセス: 処理を実行する実体

ファイル: データを記憶するバイトデータ列

ソケット: TCP/IP によるネットワークコネクション

この様に，個々の計算機リソースは，計算機の基本的な機能を提供するものであり，これらを用いてより高度なサービスを実現することにより，様々な情報システムを構築することができる．

プロセスは，実行単位である一つあるいは複数のスレッド，および，それらが共有するメモリ空間とで構成される．

ファイルは，バイトデータ列の記録領域のみを提供し，ディレクトリ等の階層的なファイル管理機能は提供しない．

SMI は，一般的なファイルシステムの機能を，ファイル I/O 等のファイルアクセス機能と，ディレクトリ等の各ファイル間の関係を管理するファイル管理機能とに分割し，ファイルアクセス機能は SMI 側で提供し，ファイル管理機能は上位層に委ねる．これにより，SMI 側でファイルというアクセス単位を活用した最適化が実現できると同時に，上位層で特定アプリケーションに適したファイル管理機能を提供することができる．

ネットワークに関しては，TCP/IP によるネットワーク機能をソケットという形で提供する．TCP/IP は事実上のデファクトスタンダードであり，ほとんど全てのクライアントは TCP/IP による通信をサポートしている．そのため，SMI 側で直接サポートすることにより，上位層に対して標準的な通信機能を提供する．

さらに，ネットワークの高信頼化のため，クライアントから見て，すなわち，外部ネットワークに対しシングルイメージを提供する．外部ネットワークに接続された各計算機ノードは全てルータとして機能し，ルータの先に SMI が接続された仮想的なサブネットが存在するかのように動作する．受信したパケットは，SMI によって適切なノードに転送される．

この様に，プロセス・ファイル・ネットワーク等の標準的な機能を計算機リソースとして提供すると共に，その抽象度を低く抑えることによって，SMI 内での最適化，高信頼化を容易にする．

3.3 計算機リソースの仮想化

SMIでは，各計算機リソースに対してシングルイメージの形成と共に高信頼化を行うため，計算機リソースの仮想化を行う仮想リソースを用いる．

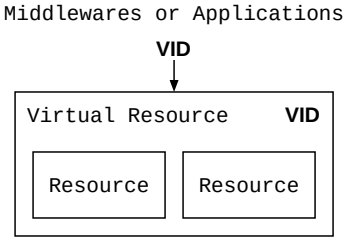

図 4: 仮想リソース

図 4 に示すように，各計算機リソースに対し，上位層からその実体を隠蔽するため，仮想リソースというデータ構造を用意する．仮想リソースには，１つまたは複数の計算機リソースの実体を関連づけ，システム内で唯一かつ固定の ID(VID) を付与する．全ての仮想リソースは，SMI 内で共有され，VID によって識別される．

上位層であるアプリケーション等は，計算機リソースをアクセスする際，計算機リソースの実体ではなく，VID を用いて仮想リソースをアクセスする．この様にすれば，上位層は，計算機リソースがどの計算機ノード上に存在するのか意識する必要は無くなる．

また，高信頼化のため，状態を複数の計算機リソース間で冗長して保持する場合，仮想リソースに対し複数の計算機リソースを対応づけることにより，計算機リソースの実際の構成を管理すると同時に，上位層から隠蔽することができる．ハードウェア故障により，計算機リソースの実体の一方が失われた場合においても，仮想リソースと計算機リソースとの関係を更新するだけで，上位層からハードウェア故障を隠蔽することができる．

3.4 高信頼化機能

3.4.1 ファイルリプリケーション

SMIでは，ハードディスク故障が発生した場合においても，記録されていたデータを保護するために，計算機ノード間でファイル単位の冗長化を行うファイルリプリケーションを実現する．

図 5 に示すように，ファイルリプリケーションでは，ファイルの冗長度を維持するため，複数の計算機ノード Node0 および Node1 の各々のハードディスク上にファイルの複製 File0 および File1 を生成する．ファイルは，任意の計算機ノード上のハードディスク上に生成することが可能であり，例えば，容量に余裕があるディスクを選択してファイルの複製を生成することができる．また，信頼性の向上や性能向上のため，ファイルの複製を多くの計算機ノード上に用意することも可能である．

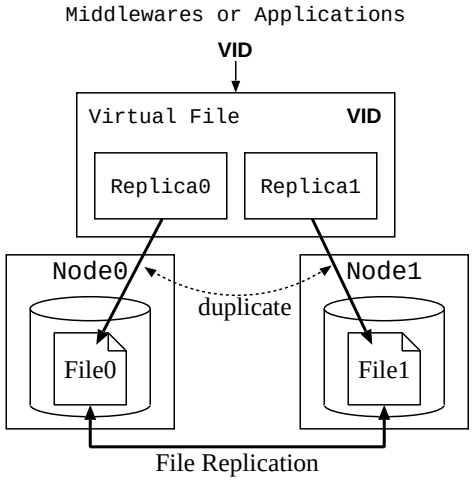

図 5: ファイルリプリケーションと仮想ファイル

また，File0，File1 と言った実際のファイルの構成を仮想化するため，仮想リソースである仮想ファイルを定義し，仮想ファイルと複製である実ファイルとの関係を構築する．この様にすれば，ファイルの複製が記録されているハードディスクの一つに故障が発生し，新たなファイルの複製を生成した場合であっても，仮想ファイル上の実ファイルとの関係を更新することによって，ファイル構成の変更を隠蔽することが可能となる．

3.4.2 プロセスペア

SMIでは，計算機ノード障害時に，実行中のプロセスを保護するために，プロセスペアを実現する．

プロセスペアは，複数の計算機ノード上で同一プログラムのプロセスを生成し，双方のプロセスへの入力を同期させることにより，複数のプロセスを全く同じ状態で保持する技術である．プロセスの実行状態とは，プロセスのメモリ空間，プロセス内の各スレッドのコンテクスト等である．

図 6 に示すように，ファイル同様，プロセスも実プロセスの構成を隠蔽するため，仮想リソースである仮想プロセスを定義し，仮想プロセスと実プロセスの関係を構築する．Process0 への入力は Process1 への入力と同期

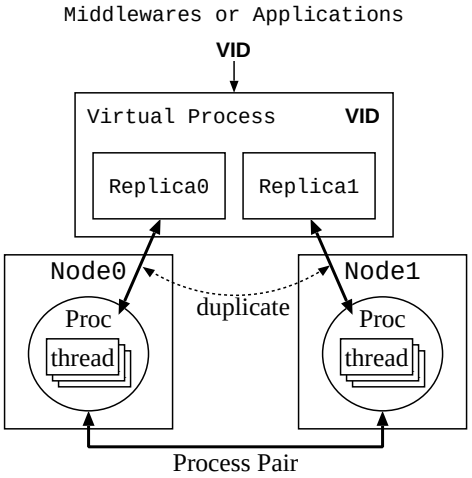

図 6: プロセスペアと仮想プロセス

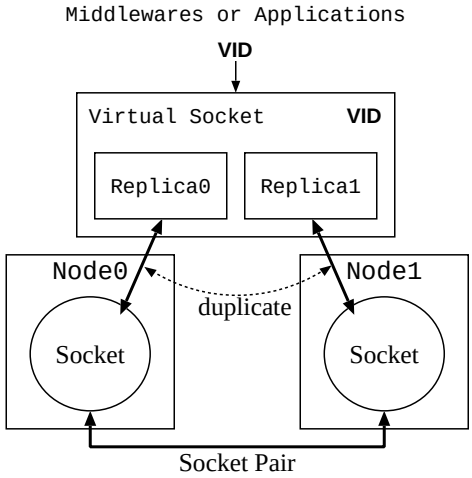

図 7: ソケットペアと仮想ソケット

され，常に同一の状態に維持する．プロセスへの入力とは，システムコールの結果であり，例えば，現在時刻の取得であれば，仮に Process0 と Process1 の実行タイミングにずれが生じたとしても，双方に同一時刻を返すことによって実現する．計算機ノード障害によって一方が失われた場合には，もう一方のプロセスを複製し，仮想プロセスの設定を更新することによって，実プロセスの消失を隠蔽することができる．

3.4.3 ソケットペア

プロセスペア同様，SMI では，計算機ノード障害時に，確立した TCP コネクションを保護するために，ソケットペアを実現する．ソケットペアは，クライアントとの通信状態を管理するソケットを複数の計算機ノード上に複製し，同期してそれらの状態を遷移させる．クライアントからのネットワークパケットやタイマ等を，双方のソケットに共に入力し処理することによって，複製された通信状態を維持する．

図 7 に示すように，ソケットも実ソケットの構成を隠蔽するため，仮想リソースである仮想ソケットを定義し，仮想ソケットと実ソケットの関係を構築する．プロセスペア同様，計算機ノード障害によって一方が失われた場合には，もう一方のソケットを複製し，仮想ソケットの設定を更新することによって，実ソケットの消失を隠蔽することができる．

4 SMI Kernel

現在，筆者らは，SMI をマイクロカーネルに組み込んだ SMI Kernel の実装を行っている．SMI Kernel は，SMI によって提供される，シングルイメージの提供とハードウェア故障に対する高信頼化をマイクロカーネル上に実現する．

以降では，まず SMI Kernel が前提とするハードウェアおよびソフトウェアの構成を紹介する．

4.1 システム構成

SMI Kernel が仮定するハードウェア構成を図 8 に示す．

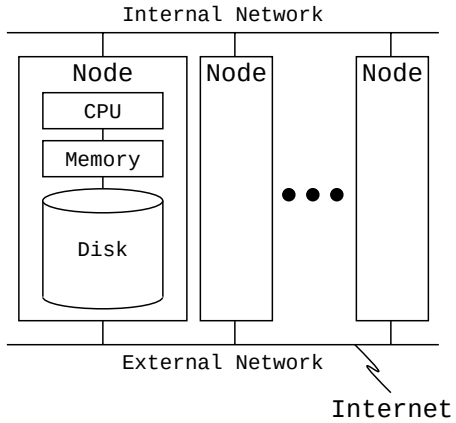

図 8: ハードウェア構成

SMI Kernel を構成するハードウェアは，以下の通りである．

計算機ノード

CPU，メモリ，ハードディスク，ネットワークI/Fで構成される計算機．一般的な計算機を用いることを前提としており，特殊なハードウェアは必要としない．

内部ネットワーク

ノード間の通信に使用するネットワーク．特定のネットワークを前提としないが，例えばEthernet等が利用できる．

外部ネットワーク

SMI Kernelと外部のシステムやクライアントとの通信に使用するネットワーク．すべてのノードが外部ネットワークに直接接続される必要はない．

図8のように，SMI Kernelを構成する各計算機ノードは，Ethernet等を用いた内部ネットワークを用いて相互に接続される．システム構成情報やファイルリプリケーションの複製データ等は，全て内部ネットワークを用いて通信する．クライアント等の外部システムとは，外部ネットワークを介して接続される．各計算機ノードは，個々にメモリやハードディスクを持ち，他の計算機ノードと独立して動作する．

次に，SMI Kernelが仮定するソフトウェア構成を図9に示す．

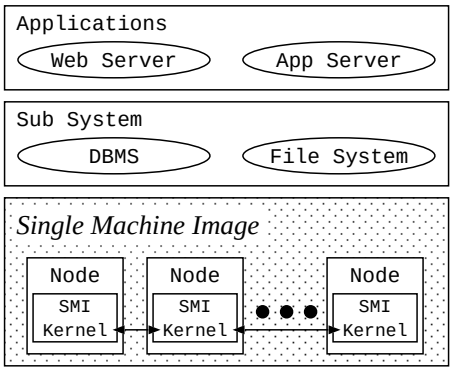

図 9: ソフトウェア構成

図9のように，SMI Kernelでは，アプリケーション・サブシステム・SMI Kernelの3層構造でシステム全体を構築する．一般的なOSが提供する機能の全てをSMI Kernelが提供するのではなく，ファイルシステムやDBMS等はサブシステムとして構築する．

SMI Kernelは，個々のノード上で動作し，各ノード上のSMI Kernelが協調動作することによって，マイクロカーネル上にSMIによるシステム仮想化を実現する．SMI Kernel上で動作するサブシステムおよびアプリケーションには，SMIによるシステム仮想化によって，高信頼化されたシングルイメージが提供される．そのため，単一計算機で構成されるシステム上で開発する場合と同じスタイルで，分散システム上でサブシステムおよびアプリケーションの開発を行うことができる．

4.2 カーネル構成

図10に，各計算機ノード上のカーネル構成を示す．

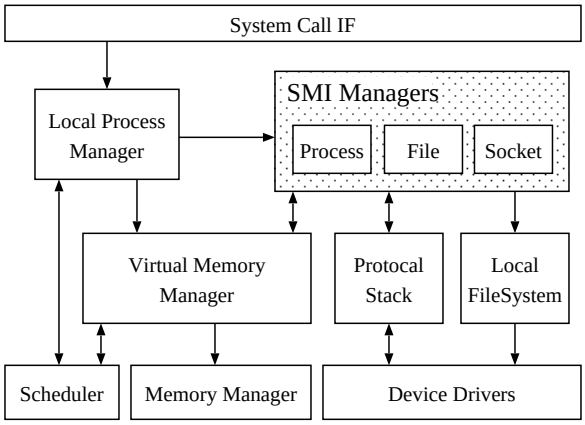

図 10: SMI Kernel

SMI Kernelは，各計算機ノード上で動作し，その計算機ノード上で動作する計算機リソースの管理を行うとともに，他の計算機ノードと連携してSMIの機能を実現する．

4.3 SMIマネージャ

システム全体の構成や状態，および，仮想リソースに関する情報は，各計算機ノード上のSMIマネージャによって維持，管理される．各SMIマネージャは相互に通信を行い，システム構成や状態に関する情報や仮想リソース情報の一貫性を維持する．

また，SMIマネージャは，その内部に，プロセスペアを実現するプロセスマネージャ，ファイルリプリケーションを実現するファイルマネージャ，クライアント等のシステム外部と通信するためのTCP/IPプロトコルスタックやソケットペアを実現するソケットマネージャを持つ．SMIが提供するシステム仮想化と高信頼化機能は，仮想リソース情報の操作，および，これらの計算機

リソース別のマネージャによって実現される．

4.4 プロセス

計算機ノードで動作しているプロセスは，システムコールインターフェースを介してSMI Kernelが提供する機能を呼び出す．プロセス間通信やファイル，ネットワークへのアクセスは，ローカルプロセスマネージャを経由して，SMI マネージャによって処理される．

プロセスは，ローカルプロセスマネージャによって管理される．プロセスを構成するスレッドおよびSMI Kernel内のカーネルスレッドは，スケジューラによってディスパッチされる．個々のプロセスは，その中に含まれる一つあるいは複数のスレッドも含め，一つの計算機ノードの中に閉じて動作する．そのため，一つのプロセスの処理を，複数の計算機ノード間で分散処理することはできない．

また，プロセスペアの一方が計算機ノード障害によって失われた際にプロセスを復元するため，ローカルプロセスマネージャは，プロセスに含まれる全スレッドのコンテクストを取り出す機能を持つ．

4.5 仮想記憶

仮想記憶マネージャによって，各プロセスのメモリ空間としての仮想記憶を提供する．仮想記憶のバッキングストアにSMIが提供する仮想ファイルを用いることによって，プロセスイメージがハードディスク故障によって失われることを防いでいる．

ローカルプロセスマネージャ同様，仮想記憶マネージャも，プロセスを復元するため，仮想記憶とプログラムファイルおよびバッキングストアの状態を取り出す機能を持つ．

4.6 デバイス

各計算機ノードに接続されたハードディスクやNIC等のデバイスは，SMI Kernel内のデバイスドライバによって制御される．ローカルファイルシステムは，ハードディスク上にファイルイメージを提供する．SMI マネージャ内のファイルマネージャは，ローカルファイルシステムが提供するファイルイメージをベースに，計算機ノード間のファイルリプリケーションを実現する．

システム内部に閉じた計算機ノード間通信は，より効率的な通信を実現するため，専用のプロトコルを用いて通信する．そのため，ソケットマネージャが持つTCP/IPプロトコルスタックとは別に，専用のプロトコルスタックを用いてノード間通信を行う．

この様に，SMI Kernelでは，個々の計算機ノード上の実行モジュールは独立しており，各計算機ノード上のSMI Kernelが協調することにより，SMIを実現している．このため，計算機ノード故障が発生した場合においても，他の計算機ノードのSMI Kernelが仮想リソースの設定を変更することによって，故障の影響範囲を小さくするとが可能となる．

5 SMI Kernelの課題

他の分散システム同様，SMI Kernelにおいても，計算機ノード間の通信量が増大すると，性能が劣化すると考えられる．また，SMI Kernelが提供するファイルリプリケーションは，データ更新時に複製との同期のため常に計算機ノード間通信を行う必要がある．そのため，計算機ノード間通信を高速に行うこと，また，通信量を減らすことが，本研究の課題と考える．現在，高速な計算機ノード間通信，および効率的なファイルリプリケーションの検討を行っている．

また，プロセス・ファイル・ソケット等の各計算機リソースの配置によっては，システム全体の性能が劣化することも考えられる．例えば，ある特定のハードディスクへのアクセスが集中すると，システム全体の性能がそのハードディスクの処理能力に制限されてしまう．そのため，プロセスの配置だけではなく，全計算機リソースの配置を考慮した負荷分散を行うことが，本研究の課題と考える．

6 関連研究

IBMの提唱するProject eLiza[1][1]は，自己構成，自己修復，自己最適化，自己防衛によって，自律的なコンピューティングを段階的に実現するプロジェクトである．自己構成とは，稼働中の動的拡張やソフトウェアの自動バージョンアップ，ワイヤレス機器からの遠隔管理等を意味している．自己修復とは，エラー頻度や再試行の回数に応じた障害発生の予測と分離，二つの論理回路での計算結果の照合と再実行等を意味している．自己最適化とは，単一OS下の負荷を優先順位に応じた計算機リ

[1]Project eLizaは，IBM社の登録商標です

ソース自動割当，複数異機種間の負荷分散等を意味している．自己防衛とは，ネットワークを経由した不正使用への自動対応，ロジカルパーティション間の独立性確保等を意味している．Project eLizaは，ハードウェアとソフトウェアを組み合わせて機能を実現している点など，SMIとは実装方式等で多くの違いはあるものの，自己構成，自己最適化という研究開発の方向性では，多くの共通点があると言える．

Mach[2]は，マイクロカーネルとしてOSの基本的な機能を実装し，OSの他の機能は，複数のサーバとして実装する．これらのサーバは，図9のサブシステムと同じ役割を果たしている．SMI Kernelは，Machからマイクロカーネルとサーバと言ったOSの構成方式の影響を受けているが，SMIを用い，マイクロカーネルレベルでシステム仮想化を行う点が大きく異なっている．

Solelc[3]は，SMI Kernelと同様に，シングルイメージを実現する分散OSである．Solelcは，複数の計算機ノード上で単一のカーネルを実現することによりシングルイメージを実現しているが，SMI Kernelでは，仮想リソースのネームスペースを一つにすることによってシングルイメージを実現している点が大きく異なっている．

7 まとめ

筆者らは，計算機システムの安定運用を容易に実現する，システム仮想化手法Single Machine Image(SMI)と，このSMIをマイクロカーネル内に実装するシングルイメージ分散マイクロカーネルSMI Kernelの研究を行っている．

本稿では，シングルシステムイメージが提供する分散環境のシングルシステムへの仮想化と，データ複製等の高信頼化に必要なハードウェア仮想化を，一つに統合したシステム仮想化手法であるSMIを提案した．SMIは，プロセス・ファイル・ネットワークといった計算機リソースを仮想化すると共に，個々の計算機リソースに適した高信頼化機能である，ファイルリプリケーション，プロセスペア，ソケットペアを提供する．

SMI Kernelは，個々の計算機ノードで独立して動作し，ローカルのプロセスやファイル等を管理すると共に，計算機ノード間で協調動作するSMI Managerによって，SMIの諸機能を実現する．

現在，SMI Kernelの実装を進めており，終り次第，性能評価を行う予定である．

参考文献

[1] Project eLiza, http://www-1.ibm.com/servers/eserver/introducing/eliza/, IBM Corp.

[2] M. Accetta, R. Baron, W. Bolosky, D. Golub, R. Rashid, A. Tevanian and M. Young, Mach: A new kernel foundation for UNIX development, In Summer Conference Proceedings, USENIX Association, 1986.

[3] 芝公仁, 大久保英嗣, “分散オペレーティングシステムSolelcの構成”, 情報処理学会研究報告, 2000-OS-84, Vol.2000, No.43, pp.237-244, 2000