システムソフトウェアと
オペレーティング・システム 92−14
（２００３．２．２８）

シングルイメージカーネルのモバイルインフラへの応用

前田 誠司　佐藤 記代子　矢野 浩邦†

分散システムは，システム構成が複雑であるため，その能力を引き出すことができるアプリケーションを開発することは難しい．筆者らは，複雑なシステム構成を隠蔽し，単一なシステムイメージを提供するシステム仮想化手法 Single Machine Image およびその実装である SMI Kernel の研究を行っている．ユビキタス・コンピューティングの世界の到来や IPv6 の普及より，多くの機器がインターネットに直接接続され，計算能力を持ったモバイルインフラが構築されると考えられる．しかし，単に多くの機器をネットワークで接続しただけでは，ユーザがモバイルインフラの処理能力を活用する事は難しい．

本稿では，SMI Kernel を応用し，モバイル環境のインフラとして，シングルイメージを提供する方式を提案する．移動中のユーザに対し個別のシングルイメージを提供すれば，単一機器の使用感で近隣にある複数の計算機を容易に活用できるようになると考える．

Applying Single Image Kernel to Mobile Infrastructure

Seiji Maeda, Kiyoko Sato and Hirokuni Yano††

Because of the complexity of the system structure, it is difficult to develop applications which can exploit the capability of distributed systems. We are studying Single Machine Image(SMI) as a method of a system virtualization scheme which hides the complexity of the system structure and provides Single System Image. We are also implementing SMI Kernel which delivers SMI in the kernel layer.

Following emergence of ubiquitous computing and IPv6, many devices will be attached to the Internet directly and the mobile infrastructure which has computing capability will be formed. However, just connecting many devices hardly make mobile users utilize the capability of the infrastructure.

In this paper, we propose the method of applying SMI Kernel to the mobile infrastructure and the scheme of providing single images. Single images which are provided mobile users respectively can make the users utilize ubiquitous computers like a single computer.

1　はじめに

分散システムは，複数の計算機の能力を活用できれば，単体の計算機より高い処理能力を得ることができる．しかし，システム構成が複雑であるため，その能力を引き出すことができるアプリケーションを開発することは難しい．筆者らは，分散システムの複雑なシステム構成を隠蔽し，アプリケーションに対し，単一なシステムイメージを提供するシステム仮想化手法 Single Machine Image(SMI) およびその実装である SMI Kernel の研究を行っている [1].

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とを，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 をカーネル内に実装することにより，システム仮想化をカーネルレベルにおいて可能とする．

一方，計算機が身の回りに遍在化するユビキタス・コンピューティングの世界が現実のものとなりつつある．

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

さらに，IPv6の普及より，より多くの機器がインターネットに直接接続する事が可能になると考えられる．しかし，単に多くの機器をネットワークで接続しただけでは，それらの処理能力を有効に活用する事は難しい．そこで，シングルイメージカーネルであるSMI Kernelをモバイル環境のインフラとして応用する方式を提案する．

本稿では，2章で，システム仮想化技術であるSMIおよびSMI Kernelを解説する．3章で，ユビキタス・コンピューティング世界におけるモバイルインフラとその課題を明らかにする．4章では，SMI Kernelをモバイルインフラに応用する方式を提案する．続いて5章では，予備評価の結果とその考察を行い，最後に，まとめを行う．

2 Single Machine Image

Single Machine Image(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とを，一つに統合したシステム仮想化手法である．

2.1 SMIによるシステム仮想化

シングルシステムイメージによるシステム仮想化手法や，RAID等のハードウェア仮想化手法に対し，SMIでは，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一つに統合する．図1にSMIによる仮想化のイメージ図を示す．

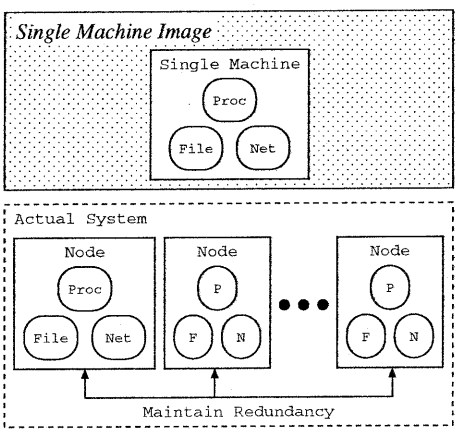

図 1: SMIによるシステム仮想化

SMIでは，シングルシステムイメージと同様に，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．SMIでは，さらに複数の計算機ノード間で、データ等の冗長度維持を行うことによって，同時に高信頼化も実現する．

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，ハードディスクの一つに故障が発生した場合，その際に失われたファイルに対し，他のハードディスク上の複製を用いて新しい複製を生成する．この際，ファイルが存在する計算機ノードは変化するが，シングルイメージを用いて，位置の変化を隠蔽する．

以降では，SMIで提供する計算機リソースおよびその仮想化方式に関して解説する．

2.1.1 計算機リソース

SMIでは，ハードウェア構成的に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，以下の3種の計算機リソースを用いる．

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，これらを用いてソフトウェアを開発することにより，様々な情報システムを構築することができる．

2.1.2 計算機リソースの仮想化

SMIでは，計算機リソースが存在する計算機ノードを隠蔽するため，各計算機リソースに対し，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の情報は，SMI内で共有され，VIDによって識別される．

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

想リソースにアクセスする．この様にすれば，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．

2.1.3 計算機リソースの高信頼化

SMIでは，計算機リソースを高信頼化するために，計算機ノード間で計算機リソースの複製を行う．プロセスは，計算機ノード間でプロセスペアを構成する．ファイルは，ファイル単位で複製を持つファイルリプリケーション [3] を用いて高信頼化する．ソケットに関しては，ソケットの状態を複製し，それらの状態遷移を同期させることによって，コネクションの状態が失われることを防ぐ．

2.2 SMI Kernel

現在，筆者らは，SMIを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とを、カーネルレベルで実現する．

図2に，SMI Kernelを用いたソフトウェア構成を示す．

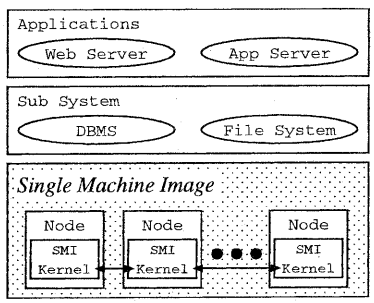

図 2: SMI Kernelを用いたソフトウェア構成

SMI Kernelは，Mach[4] などに代表されるマイクロカーネルと比較すると多機能であるが，マイクロカーネル同様，オペレーティングシステムとしての機能の一部を、サブシステムというユーザプロセスの形で実現する．サブシステムとして提供する機能は，主にデータの名前空間の管理であり，例えばファイルシステムの名前空間を提供する．

図3に，各計算機ノード上のカーネル構成を示す．

SMI Kernelは，各計算機ノード上で動作し，計算機ノード内にある計算機リソースの管理を行うとともに，

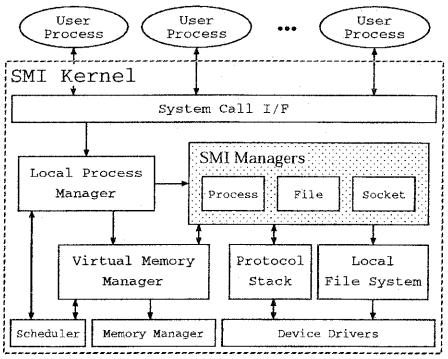

図 3: SMI Kernel

他の計算機ノードと連携してSMIの機能を実現する．

SMIの諸機能は，SMIマネージャによって実現される．プロセス，ファイル，ソケットの各計算機リソース毎に，シングルイメージの提供と高信頼化に必要な処理を行う．

SMIマネージャ以外は，通常のカーネルと同様の構成となっている．ユーザプロセスは，システムコールインターフェースを介してカーネルの機能を呼び出す．ローカルプロセスマネージャは，プロセスに含まれるスレッドの管理や，仮想記憶との対応を管理する．仮想記憶マネージャは，プロセスへの仮想記憶の提供や，ファイルキャッシュ機能を提供する．ローカルファイルシステムは，各計算機ノードに接続されたハードディスク上に，ログ構造ファイルシステムを構築する．各計算機ノード間の通信は，プロトコルスタックを介して行われる．

以降では，このSMI Kernelをモバイルインフラへ応用する手法を提案する．

3 ユビキタス・コンピューティングにおけるモバイルインフラ

図4に，ユビキタス・コンピューティングにおけるモバイルインフラの例を示す．

ユビキタス・コンピューティングの世界では，インフラとして，また，ユーザ所有の機器として，至る所に計算機が存在すると考えられる．これらの計算機は，IPv6を用いてインターネットに接続され，巨大な分散システムを形成する事になる．また，ユーザは，PDAやノートPCなどのモバイル機器を持ち，移動中においても，無

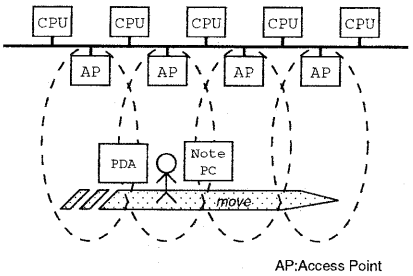

図 4: ユビキタス・コンピューティングにおけるモバイルインフラ

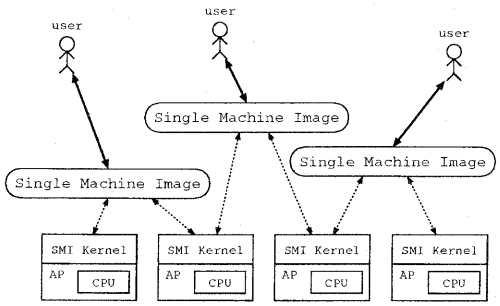

図 5: SMI Kernel によるモバイルインフラのイメージ

線ネットワークのアクセスポイントを介して，インターネットに接続すると考えられる．

この際，無線ネットワークの多くはアクセスポイントのカバー範囲が数 10〜数 100m と小さく，図 4 に示すように，ユーザは，複数のアクセスポイントのカバー範囲を、ローミング技術を用いて移動する事になる．

しかし，移動中のユーザが持つモバイル機器を含め，多くの計算機がネットワークに常時接続できるインフラが整備される反面，各ユーザが，分散している計算機を活用するための仕組みは整備されていないと言える．そのため，ユビキタス・コンピューティングの世界を活かすためには，近隣に遍在する多くの計算機を活用するための仕組みが必要であると考える．

4 SMI Kernel のモバイルインフラへの応用

筆者らは，SMI Kernel を用いて，近隣に遍在する多くの計算機リソースを活用するためのインフラを実現する．SMI Kernel をモバイルインフラへ応用するため，SMI が提供するシングルイメージを、各ユーザ別に用意し，さらに，そのイメージをユーザの移動に追従させる改良を行う．

以降では，SMI Kernel の作り出す，モバイルインフラのイメージと、ユーザ移動追従プロセスマイグレーションについて説明する．

4.1 ユーザ別 Single Machine Image

図 5 に，SMI Kernel が作り出すモバイルインフラのイメージを示す．

図 5 下部に示すように，各アクセスポイント近隣の計算機リソース (CPU) 上で，SMI Kernel を動作させる．

各 SMI Kernel 上の SMI マネージャは，CPU 間で連携し，各ユーザに対し SMI を提供する．ユーザ別にイメージを提供するので，各ユーザは，自分のニーズにあったシステムをイメージ上に構築することが出来る．

各ユーザ用の SMI は，ユーザ近隣の SMI Kernel を用いて提供する．この様にすれば，各ユーザは，自分の計算機環境を使い続けながら，遍在する計算機リソースを活用することができる．

4.2 ユーザの移動への追従

図 6 に示す様に，モバイル機器を持ったユーザは，徒歩や自動車などで移動する事が考えられる．この際，移動前に SMI を提供していた計算機を使い続けると，ユーザ近隣の計算機リソースを使えないばかりか，ネットワーク遅延やトラフィックの増大が起ってしまう．そこで，ユーザの移動に合わせて，ユーザが使用する SMI Kernel を変更し，常にユーザ近隣の SMI Kernel を用いて SMI を提供する．

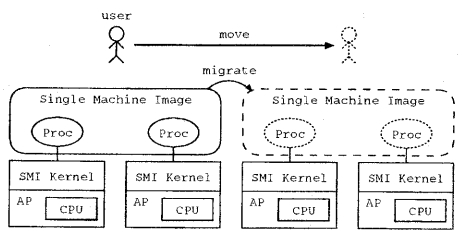

図 6: ユーザの移動への追従

通常，各ユーザ用の SMI には，幾つかのプロセスが含まれている．そのため，SMI を移動するには，各 SMI Kernel 上で動作しているプロセスを，他の SMI Kernel に移動する必要がある．SMI Kernel では，システムの動的な負荷分散を実現するため，プロセスマイグレーション [2] を実装中である．この技術を改良し，ユーザの移動に追従するプロセスマイグレーションを実現する．

ユーザの移動に追従する為には，ユーザの現在位置を特定する必要がある．しかし，ユーザは，インターネットへの接続を維持するためにアクセスポイントのローミングを継続して行うので，どこのアクセスポイントに接続しているのか調べれば，容易に位置を特定することができる．

4.3 プロセスマイグレーションの動作

SMI Kernel では，プロセスマイグレーションを GFVM[2] という仮想記憶を用いて実現する．GFVM では，仮想記憶に用いるバッキングストアを各プロセス別にファイルとして用意し，そのファイルをファイルリプリケーション [3] を用いて，複数の計算機ノード上に複製する．

図 7 に，プロセスマイグレーションの動作概念図を示す．

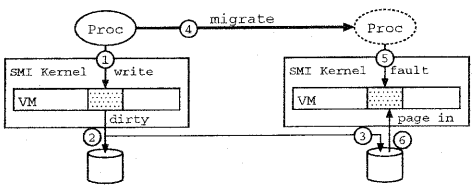

図 7: プロセスマイグレーション

プロセスによってデータの更新が行われると，仮想記憶によって該当ページのダーティフラグが設定される (図 7-1)．プロセスマイグレーション時には，まず，転送元仮想記憶上のダーティページを全てフラッシュする (図 7-2)．バッキングストアファイルは，通常，プロセスが動作している計算機ノードと，他の計算機ノードとの間で複製される (図 7-3)．続いて，CPU のレジスタ値を転送先計算機ノードに転送する (図 7-4)．転送先計算機ノード上で，プロセスの実行を再開すると，ページフォルトが発生し (図 7-5)，仮想記憶によって該当ページがバッキングストアファイルよりページインされる (図 7-5)．

ユーザ移動追従プロセスマイグレーションでは，ユーザの移動履歴を用いてユーザの移動位置を予測し，バッキングストアファイルを複製する計算機ノードを決定する．例えば，アクセスポイントのローミング記録から，ユーザの移動速度と方向を計算し，それを元に今後移動するであろうアクセスポイントの位置を予測する．ユーザの移動を正しく予測することができれば，プロセスマイグレーション後のページフォルト処理において，該当ページを保存するバッキングストアファイルがその計算機ノード上に存在するので，ページインを高速に行うことができる．

さらに，ユーザの移動を正確に予測する事に加え，ユーザの移動速度に対して十分高速にプロセスマイグレーションを行う必要がある．ユーザの移動に伴ってプロセスマイグレーションを行った際，プロセスの実行時間と比較してプロセスマイグレーションのオーバヘッドが大きいと，ユーザの使用感が著しく低下すると予想される．

5 予備評価

筆者らは，現在，SMI Kernel 上に，ユーザ移動追従プロセスマイグレーションの実装を行っている．そのプロセスマイグレーションのオーバヘッドを予測するため，予備評価を行った．

5.1 プロセスマイグレーションの予備評価

プロセスマイグレーションのオーバヘッドの大部分を占めるのは，プロセスのメモリイメージの転送である場合が多い．GFVM を用いたプロセスマイグレーションでは，仮想記憶中のダーティページのフラッシュがこれに相当する．そのため，予備評価として，ダーティページのフラッシュ時の動作に関連する部分の性能評価を行った．

図 8 に，ダーティページのフラッシュ時の動作概念図を示す．

仮想記憶中のダーティページは，まず SMI マネージャに送られる (図 8-1)．ダーティページは，ファイルリプリケーションを用いて，Node-A 上のローカルファイルシステムを用いて保存される (図 8-2) と同時に，ネットワークを介して Node-B に転送され (図 8-3)，Node-A と同様に Node-B にて保存される (図 8-4)．

これらの動作のうち，ローカルファイルシステムの I/O 性能と，計算機ノード間のネットワーク性能が，処理時間の大半を占めると考えられる．そのため，これら

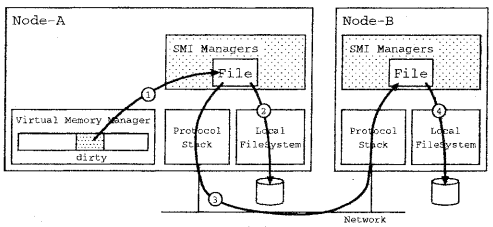

図 8: ダーティページのフラッシュ

の性能評価を行った．

表 1 に，性能評価に用いた計算機の構成を示す．

デバイス	種類
CPU	Pentium III/800EB
チップセット	Intel 810E
メモリ	SD-RAM 64MB
HDD	20GB UltraATA/66

表 1: 性能評価に用いた計算機の構成

表 2 に，ローカルファイルシステムの I/O 性能を示す．

アクセスパターン	性能 [MB/s]
random write	11.0
random read	1.73

表 2: ローカルファイルシステムの I/O 性能

性能評価では，仮想記憶のページサイズを 4KB，プロセスのメモリイメージの大きさを 64MB と仮定し，64MB のファイルを 4KB 単位でアクセスする際の性能を測定した．仮想記憶中のダーティページは，不連続に存在する可能性が高いので，ダーティページのフラッシュ時にはランダムな書き込みが発生すると考えられる．また，ページイン時にも同様にランダムな読み込みが発生すると考えられる．

表 3 に，ネットワーク性能を加味した検討結果を示す．

ここでは，ネットワークとして広域ネットワークを仮定し，10〜100[Mbps] 程度の帯域を想定する．仮に，あるユーザが使用できるネットワーク帯域が 50[Mbps]，ダーティページが 64MB 中 30%であったとすると，約 3.07[s] 必要であり，I/O 性能と合わせると，全てのダーティページをフラッシュするためには，4.82[s] 程度かかる計算となる．

また，転送先計算機において，ダーティページのフラッシュによって書き込まれたページが，ファイルキャッシュ上に残っていない場合，さらにページインの処理時間が加わることになる．

動作	処理時間 [s]
ネットワーク転送	3.07
ディスク I/O	1.75
フラッシュ処理時間	4.82

表 3: ダーティページに関する処理時間の予測

表 4 に，ページインの処理時間の予測値を示す．

ページインの割合 [%]	処理時間 [s]
5	1.85
10	3.70
20	7.40

表 4: ページインに関する処理時間の予測

この様に，ページインが発生するメモリ量が，仮に，プロセスの全メモリイメージの 5%であったとしても，処理時間が 1.85[s] かかる計算となる．

ファイルキャッシュ内には，最近使用されたファイルの内容が保持されるため，ダーティページの総量がファイルキャッシュのサイズより多い場合に，ページが取り除かれると考えられる．通常，ファイルキャッシュはダーティページ量に対して十分大きいので，ページインが発生するメモリ量は，皆無であると考えられる．しかし，他のプロセスによる影響も無視できないため，より実践的な性能評価が必要であると言える．

5.2 プロセスマイグレーションのオーバヘッドの割合

仮に，アクセスポイントが 50[m] 間隔で設置されているとすると，ユーザが徒歩で 4[km/h] の速度で移動した場合，45[s] でアクセスポイント間を移動することになる．このため，プロセスマイグレーションのオーバヘッドの割合は，ページインが全く必要がなかった場合においても，全体の約 10.7%になる．この割合では，ユーザの使用感に影響を与える可能性が高いと言える．

オーバヘッドの軽減には，以下の点が考えられる．

プロセスサイズ

64MB のプロセスサイズは，Web ブラウザ等よりも大きいサイズであり，実際にはより小さいサイズのプロセスの割合が大きいと考えられる．

ダーティページの割合の削減

プロセスマイグレーション中以外でも，更新頻度の

低いページを積極的にフラッシュすることによって，プロセスマイグレーション時のダーティページを減らすことが可能である．

I/O 性能の改善

SMI Kernel のローカルファイルシステムは，同じハードウェア構成の Linux 等と比較して 50～70%程度の性能に留まっている．チューニング等による性能向上が必要であると言える．

ハードウェア性能の改善

Pentium4/2.4GHz と高速な HDD を用いたシステムにおける性能評価では，約 20[MB/s] の書き込み性能が測定された．また，Gigabit イーサネットの普及等，ハードウェア性能が向上すれば，性能の改善が期待できる．

6 まとめ

筆者らは，分散システムの複雑なシステム構成を隠蔽し，アプリケーション開発を容易にするためのシステム仮想化手法 Single Machine Image(SMI) と，SMI をカーネル内に実装するシングルイメージカーネル SMI Kernel の研究を行っている．

計算機が身の回りに遍在化するユビキタス・コンピューティングや IPv6 の普及により，より多くの機器がインターネットに直接接続されると考えられるが，それらの処理能力を有効に活用する事は難しい．そこで，SMI Kernel をモバイル環境のインフラとして応用する方式を提案した．

モバイルインフラとしての SMI Kernel では，各ユーザ別にシングルイメージを提供する．また，プロセスマイグレーションを活用し，シングルイメージをユーザの移動に追従させる．

プロセスマイグレーションのオーバヘッドを予測するための予備評価を行なった結果，現状ではオーバヘッドが無視できない割合であることが分かった．今後は，SMI Kernel の改良を続けると共に，より実践的な性能評価を行う予定である．

謝辞 本研究の一部は，通信・放送機構の委託研究「ヒューマンセントリック ユビキタスネットワーク基盤システムに関する研究開発」の下に行った．

参考文献

[1] 前田他, “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”, SWoPP 2002, 2002-OS-91, Vol.2002, No.9, pp.63-70, 2002

[2] 佐藤他, “Global File Virtual Memory を用いたプロセスマイグレーション機能の設計”, FIT2002, 2002

[3] 矢野他, “シングルイメージクラスタシステムにおける高信頼ストレージ機能の設計”, FIT2002, 2002

[4] 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.