RAID内蔵型NAS向けメモリ断片化防止機能の開発

江端 淳[†] 藤原真二[†]

RAID内蔵型NASのメモリ断片化を防止し，信頼性を向上させるメモリ断片化防止機能を開発した．RAID内蔵型NASは，RAIDシステムに内蔵するNASブレードにシステム内部のネットワーク経由でディスク装置のボリュームを割り当てることにより構成される．そして，2枚のNASブレードでNASクラスタを構成し，NASブレード間のフェールオーバ機能を備えることにより高可用・高信頼なシステムを実現している．しかし，長期運用によりNASブレードのメモリが断片化し，フェールオーバ発生時にファイルシステム引継ぎのためのマウント処理がメモリオブジェクトを確保できず，NASのサービスが停止する可能性があった．この問題を解決するため，メモリ断片化の要因となるメモリオブジェクトを一カ所に集約し大きな連続領域を確保するメモリ分割確保方式を開発した．本方式の有効性を検証するため，NASブレードのオペレーティングシステムに本方式を実装し評価した．評価の結果，本方式がメモリ断片化によるNASサービスの停止を防止することを確認した．

Development of a Reliable Memory System to Prevent Memory Fragmentation in RAID-installed NAS Blades

Atsushi Ebata[†] and Shinji Fujiwara[†]

As a step towards more reliable RAID-installed NAS systems, we have developed a new memory system that includes a reliable mechanism for preventing memory fragmentation. A RAID-installed NAS system consists of clusters of NAS blades, each of which is equipped with a failover function for high reliability and high availability, system-internal network connections, and storage units. The cluster consists of two blades. The software architecture of the blades means that memory fragmentation within a blade can reach the stage where blocks required for mounting the file system cannot be secured by the kernel and the NAS service goes down. If failover is required after that, the NAS cluster goes down. To solve this problem, we developed the divided memory allocation method, in which the memory objects that contribute to fragmentation are allocated to a fixed region within physical memory, ensuring that large contiguous address spaces are available elsewhere within the memory. To validate the divided memory allocation method, we have implemented the method in an actual operating system for NAS blades. The results of evaluation show that the method stops memory fragmentation from reaching the point where NAS service goes down, thus making NAS-system operation more reliable.

1. はじめに

情報システムの利用拡大により，システムが保持するデータ容量が急増している．その膨大なデータを管理するコストを削減するため，ストレージ統合が求められ，SAN（Storage Area Network）接続による管理の統合が進められてきた．その一方で，SANよりも管理が容易であるLAN（Local Area Network）に接続することが可能なNAS（Network Attached Storage）の利用も急増している．我々は，これらSANとNASを統合する必要があると考え，「RAID（Redundant Arrays of Independent Disks）内蔵型NAS」を提案している[1)]．RAID内蔵型NASはNAS機能を実現するNASブレードをRAIDシステムに内蔵し，二重化された専用バスで内部接続することで高速かつ高信頼なSAN/NAS統合ストレージの実現を目指す．

RAID内蔵型NASでは，高性能であるとともに，障害発生時のサービス停止時間を短くする高可用性，障害発生率を低くする高信頼性を実現することが求められている．高可用性は，2枚のNASブレードでNASクラスタを構成し，フェールオーバ機能を備えることによって実現している[3),4)]．一方，高信頼性は，Linuxをベースとしたオペレーティングシステムにファイルシステムやメモリシステムの高信頼化機能を追加することにより実現している[5)]．しかし，RAID内蔵型

† 株式会社日立製作所中央研究所
Central Research Laboratory, Hitachi Ltd.

NAS のさらなる高信頼化を性能・機能と両立しながら実現するにあたっては，いくつかの課題が残っている．

この課題の中に，長期運用時のメモリ断片化の防止がある．RAID 内蔵型 NAS は高いファイルアクセス性能を実現するため再利用性の高いファイルのメタ情報（inode）を NAS ブレードのメモリ上に常駐させる機能[1] を備えている（以下，inode 常駐機能）．またデータのバックアップなどのストレージの管理コストを低減させるため高速・多世代スナップショット機能[2] も備えている．これらの機能は大量のメモリを必要とするため，長期運用を続けると NAS ブレードのメモリが断片化する．ファイルシステムのマウント時には最大 64 kB の連続メモリ領域を確保する．このためメモリ断片化が進行すると，フェールオーバ時に障害が発生した NAS ブレードのファイルシステムを引き継ぐためのマウント処理が完了しなくなる．その結果，フェールオーバに失敗し，NAS サービスが停止する．

本稿では，RAID 内蔵型 NAS において NAS ブレードのメモリ断片化により NAS サービスが停止することを防止するメモリ断片化防止機能について検討した．また，メモリ断片化防止機能の有効性を検証するため，この機能を NAS ブレードのオペレーティングシステムに実装し，大規模なファイルシステム構成を模擬する環境を構築して評価した結果について述べる．

2. RAID 内蔵型 NAS とメモリシステム

2.1 RAID 内蔵型 NAS の構成

RAID システムに内蔵される RAID 内蔵型 NAS のシステム構成を **図 1** に示す．RAID システムは，LAN 経由でファイルアクセスを提供する NAS ブレード，SAN 経由でブロックアクセスを提供する SAN ブレード，データを格納するディスク装置，およびディスク装置と各ブレードを接続する高速内部ネットワークによって構成される．RAID システム内で，NAS ブレードにディスク装置のボリュームを割り当てることにより RAID 内蔵型 NAS を構成する．RAID 内蔵型 NAS は，1 枚の NAS ブレードと最大 128 個のボリュームによって NAS ノードを構成し，2 ノード一組で NAS クラスタを構成する．

NAS ブレードの構成を **図 2** に示す．ハードウェアは，ファイルアクセス処理を行う NAS プロセッサ，4 GB のメインメモリ，およびディスク I/O 処理を行う RAID コントローラによって構成される．ソフトウェアは，Linux ベースのオペレーティングシステム NASOS カーネルと各種サービスを提供するユーザアプリケーションから構成される．ユーザアプリケーションには，NAS ノードの障害を監視し NAS ノード間のフェールオーバ機能を提供するフェールオーバアプリケーションがある．NASOS カーネルは NFS ベースのファイル共有サービスを提供する NFS サーバデーモン，SGI 社のファイルシステム XFS をベースとした高信頼ファイルシステム，ボリュームの仮想化と高速・多世代スナップショット機能を提供する LVM（Logical Volume Manager）ドライバ，RAID コントローラを制御し NAS ブレードとボリューム間のブロックアクセスを提供する RAID コントローラドライバ，およびユーザアプリケーションや NASOS カーネルにメモリ利用環境を提供するメモリシステムから構成される．

以上の構成によって RAID 内蔵型 NAS は高信頼・高性能なシステムを実現する．次節では，本研究に特に関連するメモリシステムについて説明する．

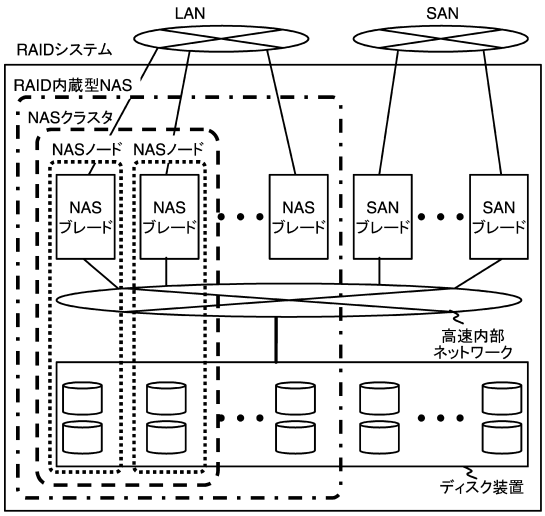

図 1 RAID 内蔵型 NAS のシステム構成
Fig. 1 System structure of RAID-installed NAS blades.

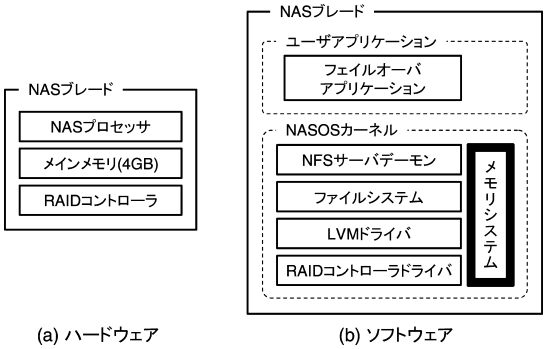

図 2 NAS ブレードの構成
Fig. 2 NAS blade structure.

表 1 メモリオブジェクトの主要用途と特性
Table 1 Characteristics of memory objects.

用途	物理メモリ上の確保単位	確保量	確保期間
スナップショット	4 kB	～0.4 GB（0.8 GB）	ほぼ恒久的
inode キャッシュ	4 kB	～1 GB	長（数時間～数日間）
データキャッシュ	4 kB	～2 GB	短（数分間～数時間）
マウント	4～64 kB	～20 MB（40 MB）	ほぼ恒久的

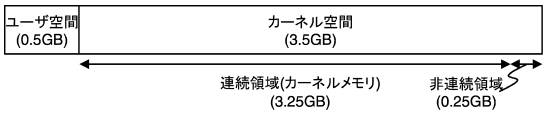

図 3 RAID 内蔵型 NAS のメモリ空間の構成
Fig. 3 Memory space of NAS blade.

2.2 メモリシステム

メモリシステムはユーザアプリケーションや NASOS カーネル（メモリシステム以外）に対してメモリの利用環境を提供する．メモリシステムが管理するメモリ資源を以下に示す．

- メモリ空間：ユーザアプリケーションや NASOS カーネル（メモリシステム以外）に対して提供される仮想的なメモリ領域．仮想アドレスが割り当てられており，物理メモリ上にマッピングされている場合，ディスク上にスワップアウトされている場合，マッピングされていない場合がある．
- ページ：物理メモリの最小単位．サイズは 4 kB.
- メモリブロック：物理アドレスが連続した 2 の N 乗個のページからなる物理メモリの空きを管理するための単位．Linux では管理容易化のため，このメモリブロックの先頭アドレスはサイズの整数倍になるように調整されている．
- メモリオブジェクト：メモリ空間上でユーザアプリケーションや NASOS カーネル（メモリシステム以外）に割り付けられるメモリ領域の単位．この実体はメモリブロック上にあり，1 つのメモリブロック内に複数ある場合，1 つのメモリブロックとそのまま一致している場合，複数のメモリブロックにまたがっている場合がある．

メモリ空間の構成を **図 3** に示す．メモリ空間はユーザアプリケーション向けのユーザ空間と NASOS カーネル向けのカーネル空間から構成される．RAID 内蔵型 NAS では NASOS カーネルの処理を優先させるため，カーネル空間を通常の Linux の 1.0 GB から 3.5 GB へ拡張し，ユーザ空間を 3 GB から 0.5 GB に縮小している．カーネル空間は連続領域（3.25 GB）と非連続領域（0.25 GB）によって構成される．連続領域は物理的に連続した領域にマッピングされている．メモリオブジェクトを確保するオーバヘッドが少ないため，NASOS カーネルは通常この領域からメモリオブジェクトを確保する．以下ではこのカーネル空間の連続領域のことをカーネルメモリと呼ぶ．非連続領域は，メモリオブジェクトが確保された時点でその領域が物理メモリにマッピングされる．ページよりもサイズの大きいメモリオブジェクトは，アドレスが飛び飛びの複数のページからなる．この領域ではメモリオブジェクトの確保が間接的に行われるためオーバヘッドはあるが，逆に連続ページの空きがなくても，メモリオブジェクトを確保できる利点を持つ．

以上，RAID 内蔵型 NAS のメモリ空間を説明したが，カーネルメモリは，サイズが 3.25 GB と大きくメモリシステムに与える影響も他の領域よりも大きいため，以降ではこの領域に焦点を当てて説明する．

カーネルメモリ上のメモリオブジェクトの主要な用途と特性を **表 1** に示す．ここで，inode キャッシュはファイルシステムがファイルの inode を格納するメモリオブジェクトの集合，データキャッシュはファイルシステムや RAID コントローラドライバがそれぞれファイルのデータとブロックアクセスのデータを格納するメモリオブジェクトの集合である．LVM が使用するスナップショット用のメモリオブジェクトは物理メモリ上の確保単位が 4 kB で，通常運用時は NAS ノードに最大 0.4 GB 確保され，フェールオーバ発生時はフェールオーバ先の NAS ノードに最大 0.8 GB 確保される．このメモリオブジェクトはスナップショットのサービス開始時に確保され，サービス終了まで確保され続けるため，確保期間は非常に長くほぼ恒久的である．以下ではこのスナップショット用のメモリオブジェクトをスナップショットメタデータと呼ぶ．inode キャッシュ用のメモリオブジェクトは物理メモリ上の確保単位が 4 kB で，最大 1 GB 確保される．このメモリオブジェクトは，ファイルアクセス時からメモリが枯渇するまで確保され続けるため，確保期間が数時間～数日間と長い．データキャッシュ用のメモリオブジェクトは物理メモリ上の確保単位が 4 kB で，最大 2 GB 確保される．このメモリオブジェクトは確保される量は非常に多いが，専用のカーネルデーモンによっ

て定期的に解放されているため，確保期間は数分間〜数時間と短い．ファイルシステムのマウント用のメモリオブジェクトは物理メモリ上の確保単位が 4〜64 kB で，通常運用時は NAS ノードに最大 20 MB 確保され，フェールオーバ時はフェールオーバ先 NAS ノードに最大 40 MB 確保される．このメモリオブジェクトは，ファイルシステムのマウント時に確保されアンマウントまで確保され続けるため，確保期間は非常に長くほぼ恒久的である．

このように，カーネルメモリのメモリオブジェクトは，用途によって，確保するサイズ，量，および期間が様々である．これらのうち，確保量が大きく確保期間が長いスナップショットメタデータ，inode キャッシュ用メモリオブジェクト，および確保サイズが大きいマウント用メモリオブジェクトは，RAID 内蔵型 NAS のメモリシステムに非常に大きい影響を与える．

3. メモリ断片化とシステムへの影響

3.1 メモリ断片化が起こる仕組み

メモリ断片化とは，カーネルメモリのメモリオブジェクトの確保・解放を繰り返すうちに空きメモリブロックが断片化し，空きメモリの総量が十分あるにもかかわらず大きなサイズのメモリオブジェクトが確保できなくなる現象である．RAID 内蔵型 NAS のようなファイルサーバは，確保期間の異なるファイルの inode とデータを同時にキャッシュするためにメモリ断片化が起こりやすい．

メモリ断片化が起こる仕組みを **図 4** に示す．カーネルメモリが未使用の状態 (1) から大量のファイルアクセスがあると，カーネルメモリ上にファイルの inode とデータが混在した状態でキャッシュされる (2)．ここで，inode キャッシュは inode 常駐機能によってカーネルメモリ上に常駐するため解放されにくく，データキャッシュは解放されやすい．したがってメモリ回収するとファイルのデータから解放されていき，inode キャッシュはメモリ上に飛び飛びに残り，カーネルメモリが断片化する (3)．ただし，この時点では inode 常駐を一時的に無効にすることによって，メモリ断片化を解消することが可能である．次にスナップショットを取得するとスナップショットメタデータはアドレスが飛び飛びで確保される (4)．その後，inode 常駐を無効化して inode を解放してもメモリ断片化を解消できなくなり，メモリ断片化が恒久化する (5)．

RAID 内蔵型 NAS では，以上のようにして，ファイルアクセスによって NAS ノードのメモリが断片化し，スナップショットによってメモリの断片化が恒久

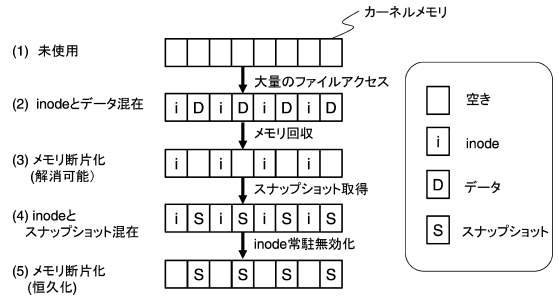

図 4 メモリ断片化の仕組み
Fig. 4 Memory fragmentation mechanism.

化する．この恒久的なメモリ断片化はスナップショット以外でも，OS 稼動中に大量にメモリオブジェクトを確保し，長期にわたって使用し続けるカーネルプロセスがある場合に，起こりうる．

3.2 メモリ断片化が与える影響

メモリ断片化が，RAID 内蔵型 NAS の高信頼化機能であるフェールオーバに与える影響ついて説明する．

RAID 内蔵型 NAS で NAS ノードに障害が発生し，その NAS ノードのサービスがフェールオーバする際，フェールオーバ先の NAS ノードはサービス引き継ぎのため各種リソースを再構築する．この再構築処理で，最も大きなメモリブロックを使用するのはマウント処理であり，最大 64 kB のメモリブロックを確保する．フェールオーバ先の NAS ノードで前節のメモリ断片化が発生し 64 kB 以上のメモリブロックが不足すると，マウントごとの 64 kB のメモリブロック確保のためのメモリ回収オーバヘッドが増加していく．そのため，マウント数に従ってマウント時間が増加していき，最後にはメモリ回収しても 64 kB のメモリブロックが確保できなくなり，マウントが完了しなくなる．そして，マウントが完了しなかったファイルシステムとその後にマウントを控えていたファイルシステムの NAS サービスが停止する．また，フェールオーバ先 NAS ノードがメモリ回収により非常に高負荷な状態になり，停止しなかった NAS サービスのサービスレベルが恒久的に低下する．

上記の現象を実証するため，PC により大規模なファイルシステム構成を長期にわたって運用する状況を模擬し，メモリ断片化発生時のマウント（全 128 ファイルシステム）の累積時間を調べた（**図 5**）．この実験では，マウント数が 71 を超えると 64 kB のメモリブロックが枯渇し，マウントの累積時間が指数的に増加し，94 マウント目が完了しなかった．マウントが完了しなかったファイルシステムとその後にマウントを控えていたファイルシステムのサービスが停止した．

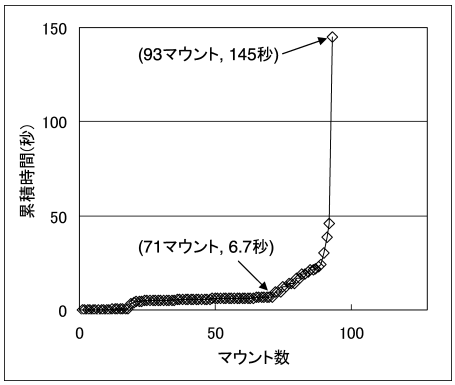

図 5 模擬環境におけるマウント処理の累積時間遷移
Fig. 5 Aggregate time of mount procedures under experimental environment.

表 2 メモリ断片化防止機能の要件
Table 2 Requirements for new reliable mechanism to prevent memory fragmentation.

#	要件	目的
1	メモリ断片化に起因するサービス停止防止	信頼性向上
2	処理オーバヘッド増加防止	性能維持
3	メモリ利用効率維持	性能維持

また，フェールオーバ先 NAS ノードでは，フェールオーバ前は 0%に近かった CPU 使用率が 90%程度まで上昇し高負荷状態が続いた．

このように，NAS ノードのファイルシステム構成が大規模化すると，NAS ノードのメモリ断片化は NAS のサービス停止という問題に発展する．

4. メモリ断片化防止機能

4.1 基本方式の検討

まずメモリ断片化防止機能の要件をあげ，次に各案を検討し，その中から要件に合う案を選択した．RAID 内蔵型 NAS を高信頼化すると同時に性能を高いレベルで実現する必要があるため，メモリ断片化防止機能は **表 2** の要件を満たす必要がある．要件 1 は本研究の目的であるメモリ断片化防止に一番重要な要件である．しかしこの要件を満たすために，メモリ確保処理自体のオーバヘッド増加や，メモリの利用効率低下に起因するメモリヒット率低下を招き，RAID 内蔵型 NAS の性能を低下させる可能性がある．このため，要件 2 と要件 3 を満たす必要がある．

基本方式案として以下の 5 つの方式案を検討した．

確保単位拡張方式：メモリブロックの先頭アドレスがサイズの整数倍であること（2 章参照）を利用し，メモリ断片化の原因となるメモリオブジェクト（スナップショット用，inode キャッシュ用）の物理メモリ上での確保単位を，従来の 4 kB から 64 kB（マウント処理が確保するメモリブロックの最大サイズ）に拡張し，マウント処理のメモリ確保失敗を防止する方式である．本方式はメモリオブジェクトの雛形の微修正のみですむため，処理オーバヘッドの増加が少ない利点を持つ．しかし，拡張した確保単位内で一部のメモリオブジェクトが使用されていると，64 kB すべての領域が解放できなくなるため，メモリの利用効率が最大で従来の 1/16 に低下する欠点を持つ．

メモリ分割確保方式：メモリオブジェクトを確保サイズ，確保期間によって複数のグループに分け，カーネルメモリを分割して各グループに対応した領域を設けることによって，カーネルメモリ全体の断片化を防止する方式である．本方式は上記と同じくメモリオブジェクトの雛形の微修正のみですむため，処理オーバヘッドの増加が少ない利点を持つ．しかし，カーネルメモリを分割するためメモリの利用効率が低下する欠点を持つ．

ガベージコレクション方式：カーネルメモリに分散したメモリ断片化の原因となるメモリオブジェクトをメモリコピーによって一カ所に集め，メモリ断片化を防止する方式である．本方式は，メモリの利用効率を従来の Linux よりも高くできる利点を持つ．しかし，メモリオブジェクトを集めるためのメモリコピーにより処理オーバヘッドが増加する欠点を持つ．

カーネル空間非連続領域利用方式：メモリ断片化の影響を受ける 64 kB 以上のメモリブロックをカーネル空間の非連続領域（図 3 参照）から確保することによって，メモリ断片化発生時のマウント処理のメモリオブジェクトの確保失敗を防止する方式である．本方式は，メモリが断片化しても 64 kB 以上のメモリオブジェクトの確保がつねに成功しフェールオーバが失敗しない利点を持つ．しかし，メモリ確保の処理オーバヘッドが高く，また，非連続領域のサイズを必要以上に大きく設計すると，メモリ利用効率が低下する欠点を持つ．

緊急退避方式：メモリ断片化により 64 kB 以上のメモリオブジェクトが確保できなくなったときのための緊急退避領域をカーネルメモリ上に設け，マウント処理のメモリ確保失敗を防止する方式である．本方式は，通常パスに処理を加えないため，処理オーバヘッドの増加がない利点を持つ．しかし，緊急退避領域が限られているためメモリ断片化による NAS サービスの停止を完全に防止でき

表 3 基本方式案の比較
Table 3 Comparison of basic methods.

方式案	サービス停止防止	オーバヘッド増加防止	メモリ利用効率維持
確保単位拡張	○	○	×
メモリ分割確保	○	○	△
ガベージコレクション	○	×	○
非連続領域利用	○	△	△
緊急退避	△	○	△

ない，緊急退避領域は普段使われないためメモリの利用効率が落ちる，などの欠点を持つ．

以上，これまでメモリ断片化防止機能を実現する基本方式案について述べた．これらを本章の初めに述べた要件で比較すると **表 3** のようになる．この中で，メモリ分割確保方式は，サービス停止防止と処理オーバヘッド増加防止の要件を満たしている．メモリ利用効率が低下するがカーネルメモリの分割数とサイズを調整することによって実用域では問題ない程度に抑えられるため，5 つの基本方式案の中でメモリ断片化防止機能の要件を一番満たしている．よって，この方式をメモリ断片化防止機能の基本方式として採用した．次節で実装方式について述べる．

4.2 実装方式の検討

前節で選択したメモリ分割確保方式を実装した．ベースとなる Linux のメモリゾーンについて説明後，メモリ分割確保方式の実装方式について説明する．

4.2.1 メモリゾーン

メモリゾーンとは，物理メモリの中でもシステムバスのアドレスの制約からプロセッサが直接アクセスできない領域である高位のメモリ（HIGHMEM）を管理するために，Linux-2.4 から導入された仕組みである．HIGHMEM のマッピング方法に関しては参考文献 6) に詳しく述べられている．ここでは，一般的な Linux がメモリゾーンを使用して行っているメモリの管理方法に焦点を当てて説明する．一般的な Linux を 4 GB の物理メモリで稼動させた場合，メモリ空間は 1.0 GB のカーネル空間と 3.0 GB のユーザ空間によって構成される．カーネル空間は 0.875 GB の連続領域（カーネルメモリ）と 0.125 GB の非連続領域から構成される．これらの空間をマッピングする物理メモリはカーネルメモリをマッピングするノーマルゾーンと，カーネル空間の非連続領域とユーザ空間をマッピングする HIGHMEM ゾーンから構成される（**図 6**）．

Linux では複数のアプリケーションが大量のメモリブロックを使用することを想定しているため，HIGHMEM ゾーンの枯渇時にユーザ空間の一部をノーマルゾーンから確保する．メモリシステムは以下 2 種類のリストによって物理メモリを管理する．

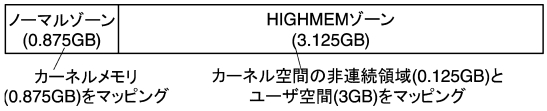

図 6 一般的な Linux の物理メモリのメモリゾーン構成
Fig. 6 Memory zone structure of Linux.

- フリーリスト：メモリゾーンの空きページの管理用リスト．各メモリゾーンごとに設けられ，メモリブロックの各サイズに対応するエントリを持つ．メモリ確保時にこのリストに指定サイズのメモリブロックがない場合，メモリシステムは指定サイズより大きいメモリブロックを切り崩し，サイズを降格させる．メモリ解放時，メモリシステムは解放するメモリブロックと物理的に隣接した空きメモリブロックの有無を調べ，ある場合は 2 つのメモリブロックを結合し，サイズを昇格させる．
- ゾーンリスト：メモリブロックの確保のメモリゾーンの巡回順序を示したリスト．最大 16 個までメモリゾーンの巡回順序を作成できる．Linux において以下の 2 つの巡回順序が記述されている．
  ノーマル指定時：ノーマル⇒終わり
  HIGHMEM 指定時：
  HIGHMEM ⇒ノーマル⇒終わり

メモリシステムによる物理メモリの管理例を **図 7** に示す．物理メモリが未使用の状態 (1) から，カーネル，ユーザアプリケーションがメモリオブジェクトを使用すると，メモリシステムはカーネルのメモリオブジェクト K をノーマルゾーンから，ユーザアプリケーションのメモリオブジェクト U を HIGHMEM ゾーンから確保する (2)．次にユーザアプリケーションが高頻度でオブジェクトを使用した場合，HIGHMEM ゾーンが枯渇する (3)．この状態からユーザアプリケーションが高頻度でオブジェクトを使用し続けると，メモリシステムは HIGHMEM ゾーンからメモリオブジェクトを確保するのを諦め，上記ゾーンリストで HIGHMEM ゾーンの巡回先として指定されているノーマルゾーンからメモリオブジェクトを確保する (4)．

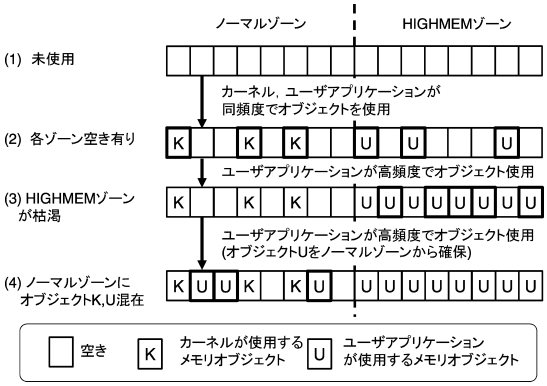

図 7 メモリゾーンを使った物理メモリの管理例
Fig. 7 Example of memory management based on memory zone.

図 8 本方式の物理メモリのメモリゾーン構成
Fig. 8 Memory zone structure of proposal method.

以上のように，メモリシステムはメモリゾーンという仕組みによって高位のメモリを管理している．

4.2.2 メモリ分割確保方式の実装方式

カーネルメモリをマッピングした 3.25 GB の物理メモリを，メモリ断片化の原因になるメモリオブジェクト（スナップショット用，inode キャッシュ用）を確保する断片化ゾーンと，メモリ断片化の原因にならないメモリオブジェクト（データキャッシュ用，マウント用 etc.）を確保する非断片化ゾーンの 2 つに分割した（**図 8**）．メモリの分割数を 2 にとどめたのは，分割にともなうメモリ利用効率の低下を最小限に抑えるためである．断片化ゾーンは，inode を従来と同じ 1 GB キャッシュしたうえ，スナップショットメタデータ 0.8 GB すべてを確保できるように 2 GB に設計した．

上記の構成で断片化ゾーンと非断片化ゾーンを完全に分離して使用すると，スナップショット未取得時に，断片化ゾーンでスナップショットメタデータ 0.8 GB の分だけ空きができ，全体のメモリ利用効率が低下する．このメモリ利用効率の低下は，スナップショットを利用せずに高いファイルアクセス性能を重視する運用で影響する．従来方式では最大 2 GB 程度のデータをキャッシュするのに対して，メモリを分割したこの構成では最大でも 1.25 GB のデータしかキャッシュできない．そのため従来方式と比べ，ディスクへの I/O が増加し，性能が低下する．このメモリ利用効率の低下によるファイルアクセス性能の低下を防ぐために，非断片化ゾーンの枯渇時に，断片化ゾーンに巡回してメモリブロックを確保できるようにした．以下にそれを記述したゾーンリストを示す．

断片化指定時：断片化⇒終わり

非断片化指定時：非断片化⇒断片化⇒終わり

以上により実用域のメモリ利用効率を維持しメモリ断片化を防止するメモリ分割確保方式が実現された．

表 4 実験環境
Table 4 Experimental environment.

PC	CPU：Intel Xeon 1.8 GHz メモリ：4 GB HDD：28 GB（SEAGATEST336706LC）
模擬 環境 1	メモリ断片化と恒久化： メモリ断片化パターン注入モジュール
模擬 環境 2	メモリ断片化： ランダムライトスクリプト メモリ断片化恒久化： スナップショットメタデータ擬似確保モジュール フェールオーバするファイルシステム（128）： ループバックデバイス（loop0〜loop127）

5. 評　　価

5.1 評 価 環 境

メモリ断片化の問題が顕在化するのは 3 章で述べたとおり NAS ノードが大規模なファイルシステム構成で長期にわたって運用を続け，その状況下でスナップショットのように大量のメモリオブジェクトをほぼ恒久的に使用するサービスを開始した場合である．これを模擬する環境を PC 上に作り，本方式を評価した．

実験環境を **表 4** に示す．PC の CPU は Intel Xeon 1.8 GHz，メモリは 4 GB，ディスクは SEAGATE ST336706LC（28 GB）である．模擬環境には，実環境とは異なるがメモリシステムのフリーリストを参照することによってメモリ断片化を観測することができる模擬環境 1 と，実環境に近い形でメモリを断片化させフェールオーバ実験を行う模擬環境 2 を用意した．

模擬環境 1 では，恒久的なメモリ断片化を起こすために，カーネルのメモリ確保関数を直接呼び出すことによって，従来方式で管理しているカーネルメモリを 0.4 GB のメモリオブジェクトで一様に断片化させるメモリ断片化パターン注入モジュールを作成した．具体的には，スナップショットメタデータ 1 ページ（4 kB）とデータキャッシュ用のメモリオブジェクト 7 ページを 1 ページずつ確保するパターンを，メモリシステムがメモリ回収を始める直前まで繰り返す．次に，デー

タキャッシュ用のメモリオブジェクトをすべて解放する．これにより，従来方式で管理しているカーネルメモリをほぼ一様に断片化させるうえに，フリーリストに連結されている空きメモリブロックを観測することによってメモリ断片化の程度を評価できる．

模擬環境 2 では，実環境に近い形でカーネルメモリを断片化させるためにランダムライトスクリプトとスナップショットメタデータ擬似確保モジュールを作成した．またフェールオーバ時にマウントする 128 個のファイルシステムには 128 個のループバックデバイス（loop0 ... loop127）を使用した．ランダムライトスクリプトは，ファイルシステム上に配置した 90 万個のファイルに対して，16 バイトのデータをライトしていき，その後，90 万回にわたってファイルをランダムに選択し 16 バイトのデータをライトする．これによって NAS ノードが長期にわたって運用を続けることを模擬する．なお，ライトするデータのサイズは 1 B から 4 kB の間であれば，実験には影響しない．スナップショットメタデータ擬似確保モジュールは，NASOS カーネルのメモリ確保関数によってスナップショットメタデータ 1 ページを連続に合計 0.4 GB 確保するモジュールである．メモリシステムは最も古くアクセスしたメモリオブジェクトから解放していくため，ランダムライトスクリプトが実環境に近い形でカーネルメモリを断片化させ，スナップショットメタデータ擬似確保モジュールがメモリ断片化を恒久化させる．なお，以下では，ランダムライトスクリプトとスナップショットメタデータ擬似確保モジュールを合わせてメモリ断片化プログラムと呼ぶ．

5.2 評価結果

5.2.1 メモリ断片化観測実験（模擬環境 1 使用）

フェールオーバ先 NAS ノードを模擬する PC に，メモリ断片化パターンを注入した際のメモリの状態を調査した．従来方式と本方式のメモリの状況を **図 9** に示す．従来方式では，起動直後には 64 kB 以上の空きメモリブロックの合計が 3,097.8 MB あり，空きメモリ全体（3,098 MB）のほぼ 100%を占めているが，メモリ断片化パターン注入後は 64 kB 以上の空きメモリブロックの合計が 8.3 MB と激減しその比率も空きメモリ全体（2,699.7 MB）の 0.3%に急低下している．この 64 kB 以上のメモリブロックの急激な減少は，メモリ断片化が進行したことを示す．これに対して本方式は，メモリ断片化パターン注入後にも 64 kB 以上のメモリブロックの合計が 1,127.0 MB（≒ 非断片化ゾーンのサイズ）あり，その比率も空きメモリ全体（2,701.6 MB）の 42%にとどまり，メモリ断片化を防止している．

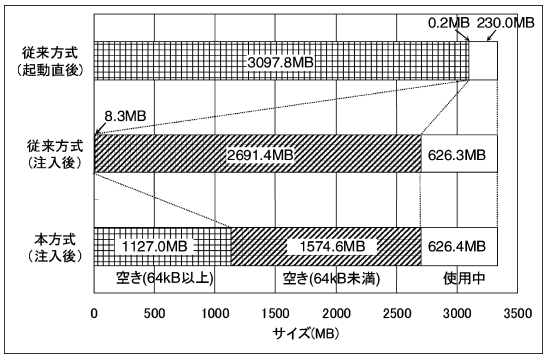

図 9 メモリ断片化パターン注入後のメモリの状態
Fig. 9 Memory state after injection of memory fragmentation pattern.

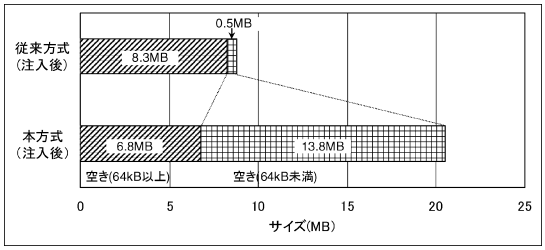

図 10 メモリ断片化プログラム実行後のメモリの状態
Fig. 10 Memory state after execution of memory fragmentation program.

以上のように本方式がメモリ断片化を防止していることが分かる．

5.2.2 フェールオーバ実験（模擬環境 2 使用）

(1) **フェールオーバ先ノードの空きメモリブロック**

フェールオーバ先 NAS ノードを模擬する PC 上で，ランダムライトスクリプト，スナップショットメタデータ擬似確保モジュールの順で実行し，空きメモリの状況を調査した．本方式と従来方式の空きメモリの状況を **図 10** に示す．従来方式の起動直後の空きメモリの状態は 図 9 と同じであるため，図示していない．断片化プログラム実行後，従来方式では，64 kB 以上の空きメモリブロックが 8.3 MB に，64 kB 未満のメモリブロックが 0.5 MB に激減している．本方式も，64 kB 以上の空きメモリブロックの合計が 6.8 MB に，64 kB 未満の空きメモリが 13.8 MB に激減している．本方式がメモリゾーンを分割しているために空きメモリブロックの合計が従来方式の空きメモリブロックの合計よりも 11.8 MB 多いが，カーネルメモリ全体（3.25 GB）の使用量の 0.4%なので大差ない．両方式の空きメモリの激減は，ランダムライトスクリプトによるファイルのライトによって，inode キャッシュやデータキャッシュが NAS ノードのカーネルメモリ上

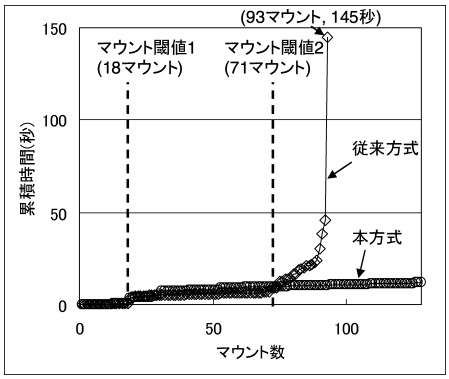

図 11 マウント処理の累積時間
Fig. 11 Aggregate time of mount procedures.

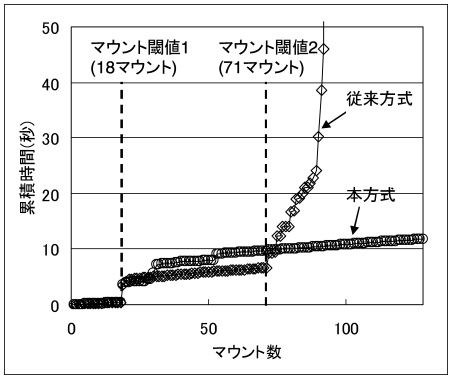

図 12 マウント処理の累積時間（Y 軸方向拡大）
Fig. 12 Aggregate time of mount procedures (expanding Y axis).

にキャッシュされているためである．

このように実環境に近い形でメモリを断片化させた場合はメモリシステムのフリーリストを参照しても，カーネルメモリの恒久的な断片化を観測できない．そのため，空きメモリ以外を観測する必要がある．

(2) **フェールオーバ先ノードのマウント時間**

メモリ断片化が NAS クラスタに与える影響を見るために，フェールオーバの処理の中で最もメモリ断片化の影響を受けるマウント処理を評価した．フェールオーバ先 NAS ノードで模擬環境 2（前節参照）のメモリ断片化プログラムを実行後，障害 NAS ノードから 128 個のファイルシステムがフェールオーバする際のマウント処理の累積時間の遷移を **図 11** に，そのY 軸方向を拡大したものを **図 12** に示す．

従来方式は，2 つのポイントでマウント処理の累積時間の増加率が大きく変わり，94 マウント目が完了しなかった．1 つ目のポイントは 18 マウント目である．これを越えた 19 マウント目でマウントの累積時間が階段状に増加した．ただし，その後，マウント処理の累積時間は 0.06 秒/マウントの傾きでほぼ一定に増加していった．2 つ目のポイントは 71 マウント目である．このポイント以降は，マウントの累積時間が指数的に増加し，94 マウント目が完了しなかった．そしてマウントが完了しなかったファイルシステムとその後にマウントを控えていたファイルシステムのサービスが停止した．また，フェールオーバ先 NAS ノードでは，フェールオーバ前はほぼ 0%であった CPU 使用率が，約 90%まで上昇し，その状態が恒久的に続いた．以下では従来方式でマウント処理の累積時間が階段上に増加した 1 つ目のポイントをマウント閾値 1 と，マウント処理の累積時間が指数的に増加し始めた 2 つ目のポイントをマウント閾値 2 と呼ぶ．従来方式に対して，本方式は，マウント閾値 1 を超えた 19 マウント目と，そのほか 31 マウント目，53 マウント目でマウント処理の累積時間が階段状に増加したものの，それ以降は 0.04 秒/マウントの傾きを保ち 128 マウントすべてを正常に完了した．なお，マウント閾値や従来方式でマウントが完了しなかったマウント数は実験によって多少のばらつきがある．

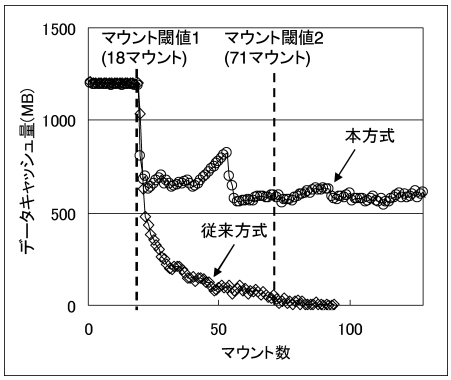

図 13 データキャッシュ量の推移
Fig. 13 Amount of data cache.

以上のように，従来方式ではマウント処理の累積時間が増加し，マウントが完了しなくなる．次にメモリ回収処理によって解放される代表的なメモリオブジェクトであるデータキャッシュの合計量の推移と，64 kB 未満のメモリブロックから 64 kB 以上のメモリブロックへの昇格量の推移を見ることによって，マウント処理の累積時間の増加とマウントが完了しなくなる現象が恒久的なメモリ断片化によることを説明する．

(a) **データキャッシュ量の解析**

データキャッシュ量の推移を **図 13** に示す．

従来方式，本方式ともに，マウント閾値 1 を超えたマウント数でデータキャッシュ量が急速に減少し始める．これはメモリ全体が枯渇しているためにマウント処理

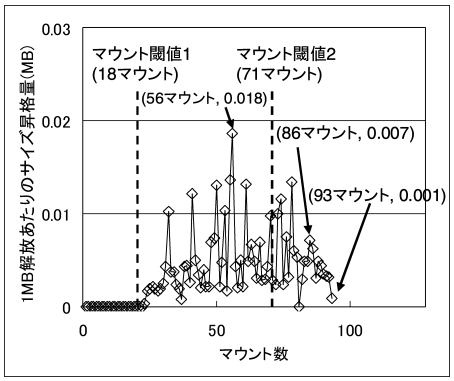

図 14 1 MB 解放あたりのサイズ昇格量（従来方式）
Fig. 14 Amount of size growing when freeing 1 MB memory object (before improvement).

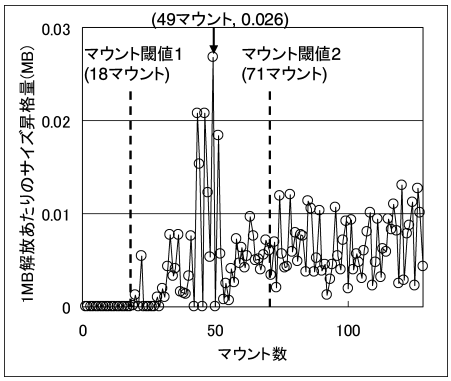

図 15 1 MB 解放あたりのサイズ昇格量（本方式）
Fig. 15 Amount of size growing when freeing 1 MB memory object (after improvement).

による 64 kB のメモリブロックの確保が失敗し，メモリ回収処理がデータキャッシュを大量に解放するためである．このデータキャッシュの大量解放によりディスクへの I/O が大量に発生するため，マウント閾値 1 を超えたところでマウント処理の累積時間が階段状に増加する．

従来方式では，その後もデータキャッシュは解放され続け，マウント閾値 2 で 36 MB になる．データキャッシュがほとんど解放されつくしているため，これ以降はカーネルメモリ上の解放可能なメモリオブジェクトがほとんどなくなる．その結果のメモリ回収オーバヘッドが増加していき，マウント累積時間が指数的に増加する．そして最後にはカーネルメモリ上の解放可能なメモリオブジェクトがなくるために 64 kB のメモリブロックが確保できなくなりマウントが完了しなくなる．

これに対して本方式は 23 マウントを越えたあたりでデータキャッシュ量の減少が止まる．これは，カーネルメモリ上で恒久的なメモリ断片化が起こっていないために，データキャッシュの解放によって，64 kB 以上のメモリブロックへの昇格（4 章参照）が起こるためである．このように，本方式は NAS ノードの負荷増加を防止し，マウント処理累積時間の指数的な増加を防止している．

(b) **64 kB へのサイズ昇格量の解析**

次に，上でも触れた 64 kB 未満から 64 kB 以上のメモリブロックへのサイズ昇格の推移を **図 14**（従来方式）と **図 15**（本方式）に示す．この 64 kB 以上のメモリブロックへの昇格量は，64 kB 未満のメモリオブジェクトを 1 MB 解放したあたりの値である．

従来方式ではマウント閾値 1 を超えたマウント数から 64 kB 以上のメモリブロックへの昇格が起こり，56 マウントでピークを迎える．昇格量が高かったマウントの次は，昇格量が少ない．これは前回のマウントで起こったメモリ回収が 64 kB 未満のメモリブロックを，余分に 64 kB 以上のメモリブロックへ昇格させたためである．このように昇格量の振幅が大きい区間では 64 kB 以上のメモリブロックに昇格可能なメモリオブジェクトがカーネルメモリ上に十分に残っていることを示す．従来方式ではマウント閾値 1 からマウント閾値 2 までの区間は昇格量の振幅が大きいため，64 kB 以上のメモリブロックに昇格可能な 64 kB 未満のメモリオブジェクトがカーネルメモリ上にあると考えられる．しかし，86 マウントを最後のピークに昇格量は減少していき，93 マウントでは昇格量がほぼ 0 に近づく．そのため次の 94 マウント目では 64 kB 以上のメモリブロックを確保できなくなり，マウントが完了しなくなった．

これに対して，本方式はマウント閾値 2 以降も昇格量の振幅が大きく，64 kB 以上のメモリブロックへ昇格可能なメモリオブジェクトがカーネルメモリ上につねに残っている．

以上のように，本方式によって NAS ノードのメモリ断片化と，それによって起こるフェールオーバ時の NAS サービスの停止を回避できることを確認した．

6. 関 連 研 究

コンピュータシステムのメモリ断片化を防止する技術はすでに提案されている[7),8)]．これらの技術は汎用的であるが，高性能・高機能な NAS におけるメモリ断片化の防止には対応できない．

Rice 大学の Navarro らは，FreeBSD ベースのシステムにおいて，物理メモリの断片化の度合いを定期的に監視し，連続ページが生成されるように選択的に

ページを回収する機能を提案している[7]．この機能は，回収対象のページが恒久的に使用されるメモリオブジェクトを格納している場合はそのページを回収できないため，恒久的なメモリ断片化は解消できない．

Microsoft の Larson らは，メモリブロックの解放時に物理的に隣接した空きメモリブロックがあった場合に，両メモリブロックのサイズや先頭アドレスに関係なく結合するメモリアロケータ LKmalloc を提案している[8]．この LKmalloc は空きメモリブロックの先頭アドレスに制約を設けている Linux のメモリシステムよりも連続ページを確保しやすい．しかし，LKmalloc は Linux のメモリシステムと同様に物理メモリ全体にメモリオブジェクトが分散する可能性があり，恒久的なメモリ断片化は防止できない．

本稿では，高性能・高信頼な NAS におけるメモリ断片化の防止を実現するメモリ断片化防止機能を開発した．開発した機能は，上記の提案機能と比べて，メモリ断片化の原因となる恒久的に使用され続けるメモリオブジェクトを一カ所に集約し大きな連続領域を確保することができる．

7. おわりに

RAID 内蔵型 NAS のメモリ断片化を防止し，信頼性を向上させるメモリ断片化防止機能を開発した．メモリ断片化防止機能の開発では，信頼性向上と性能維持を重視し，NAS ブレードの物理メモリを 2 つの領域に分割しメモリ断片化の原因となるメモリオブジェクトを 1 つの領域に集約するメモリ分割確保方式を採用した．この方式を Linux のメモリ管理の仕組みであるメモリゾーンを使い RAID 内蔵型 NAS のオペレーティングシステムに実装した．本方式の有効性検証のため，大規模なファイルシステム構成で長期運用している NAS クラスタで，片方の NAS ブレードに障害が発生し，そのサービスが他方の NAS ブレードにフェールオーバした状況を PC 上で模擬し，本方式を評価した．評価の結果，従来方式ではフェールオーバ先 NAS ノードでファイルシステム引継ぎのためのマウント処理がメモリ確保に失敗し NAS サービスが停止する状況を，本方式で解決できることを確認した．

謝辞 本稿の執筆にあたり，匿名査読者諸氏から数多くの有益なコメントをいただいた．ここに，感謝の意を表する．

参考文献

1) 薗田浩二ほか：RAID システム内蔵型 NAS（1）—アーキテクチャ概要，情報処理学会第 66 回全国大会，5D-2 (2004).
2) 中野隆裕ほか：RAID システム内蔵型 NAS（2）—多世代スナップショット機能，情報処理学会第 66 回全国大会，5D-3 (2004).
3) 宮田賢一ほか：RAID システム内蔵型 NAS（3）—障害処理機能，情報処理学会第 66 回全国大会，5D-4 (2004).
4) 坂口明彦ほか：RAID システム内蔵型 NAS（4）—高信頼内部通信機能，情報処理学会第 66 回全国大会，5D-5 (2004).
5) 中村隆喜：RAID システム内蔵型 NAS（2）—高信頼ファイルシステム，FIT2004, B-009 (2004).
6) Bovet, D.P., et al.：詳解 Linux カーネル第 2 版 (2003).
7) Navarro, J., et al.: Practical, transparent operating system support for superpages, *Proc. 5th Symposium on Operating Systems Design and Implementation*, pp.89–104 (2002).
8) Larson, P., et al.: Memory Allocation for Long-Running Server Applications, *Proc. 1st International Symposium on Memory Management*, pp.176–185 (1998).

(平成 17 年 1 月 13 日受付)
(平成 17 年 4 月 26 日採録)

江端 淳

1993 年筑波大学基礎工学部卒業．1995 年同大学大学院修士課程修了．同年（株）日立製作所入社．入所以来中央研究所に勤務．大型計算機，ネットワークストレージの研究開発に従事．

藤原 真二（正会員）

1988 年京都大学工学部卒業．1990 年同大学大学院修士課程修了．同年（株）日立製作所入社．中央研究所に勤務．1998〜1999 年スタンフォード大学 Visiting Scholar．1999〜2001 年日立アメリカ．現在は，中央研究所でデータベース，ネットワークストレージの研究開発に従事．電子情報通信学会，IEEE，ACM 各会員．