計算機アーキテクチャ 130－26
（1998. 8. 7）

PCクラスタを用いた仮想ディスクシステム

張替 誠、加納 義樹、吉澤 収一、中島 達夫
北陸先端科学技術大学院大学 情報科学研究科
{harigae,yoshiki,shuichi,tatsuo}@jaist.ac.jp
http://mmmc.jaist.ac.jp:8000/

概要

従来のRAIDに代表されるディスクアレイシステムは、複数のディスクを用いる事でストレージレベルでの耐故障性を保証している。しかし、計算機の障害が発生した際、オペレーティングシステム(OS)の停止により提供されていたサーバのサービスも同時に停止してしまう。したがって、定常的にサーバのサービスを提供するためにはストレージの耐故障性と障害時にクラスタノードが高速に復帰できる高可用性が求められている。

本研究では、以上の問題を解決するために既存のOSとストレージ間にOSに依存しない仮想的なディスクを構築する。また、仮想的なディスクを複数の計算機で動作させ、キャッシュとディスク内容を重複させることでシステムの耐故障性と可用性の向上を目標とする。

A Virtual Disk System using a PC cluster

Makoto HARIGAE, Yoshiki KANO, Shuichi YOSHIZAWA, Tatsuo NAKAJIMA
{harigae,yoshiki,shuichi,tatsuo}@jaist.ac.jp
http://mmmc.jaist.ac.jp:8000/
Japan Advanced Institute of Science and Technology

Abstruct

Disk array systems like in RAID guarantees fault tolerance through the usage of multiple disks. However, when a failure occurs, the Operating System(OS) halts, then interrupting all services provided by the servers it hosts. A fault tolerant scheme becomes necessary if servers have to provide steady and highly available services and cluster nodes should recover from a crash.

We describe in this paper a highly available and fault tolerant disk system. This system is implemented as an independent virtual disk running on several host machines between the OS and the storage system. In addition, it replicates the contents of the cache memory and the storage in order to archive availability and fault tolerance.

1 はじめに

単体PCの性能向上、高速なネットワークサブシステムの確立により、PCクラスタの普及が進んでいる。PCクラスタは、ノードとなるPC自体が安価であることと、システムの拡張が容易なことから、Webサーバやファイルサーバ、データベースサーバ等のスケーラブルサーバへ応用が期待できる。

本研究では、OSに仮想ディスクデバイスドライバを追加することにより、OSや既存のアプリケーションを極力変更することなく、利用目的に応じて動的に変更可能な仮想ディスクシステムの構築を目標とする。この仮想ディスクは、クラスタ上の複数のマシンの複数のディスクとディスクキャッシュを分散共有することにより、可用性の高い耐故障ストレージとして扱うことができる。

本稿では、ノード間通信で用いるデッドロックフリーなアトミックマルチキャスト[1]を含む高速ネットワークドライバについて述べ、分散ディスクキャッシュと分散ディスク管理からなる仮想ディスクシステムの概要について述べる。

2 高速ネットワーク

2.1 Myrinet

PCクラスタに用いるMyrinet[2]はMyricom社[3]が開発、商品化した高速LANである。図1に示すようにMyrinetのホストインタフェースにはLANaiと呼ばれるRISCプロセッサが搭載され、Myrinet上の通信プロトコルを制御している。ホストインタフェース間は、カットスルールーティングを行なうクロスバ型のスイッチで接続される。

Myrinetは、(1)1.28+1.28Gbit/s双方向の非常に高いバンド幅を持ち、ハードウェアの通信レイテンシィは数マイクロ秒以下と小さく高速である、(2)ホストインタフェース上のLANaiプロセッサはプログラマブルであるため、独自の通信プロトコルの実装が容易である、(3)ネットワークトポロジの変更が容易である等の特徴を持っている。

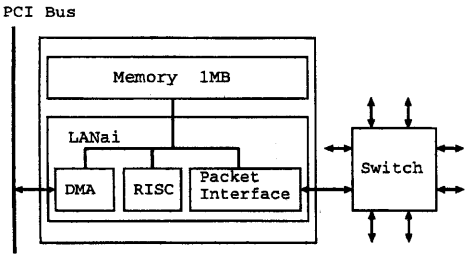

図 1: Myrinet の基本構成

2.2 従来の問題点

ネットワーク技術の発展により、計算機は高速ネットワークを通して接続され、ネットワーク上のサービスや他の計算機を含む計算機資源をいつどこからでも利用することが可能となった。高速ネットワークに対応するためにはハードウェアだけでなく、デバイスドライバアーキテクチャも高速対応であることが必要不可欠である。しかし、従来の高速ネットワーク向けのデバイスドライバアーキテクチャは並列アプリケーション用 PC クラスタなどの並列計算機には適していたが、インターネットサーバなどのアプリケーションを実行するための PC クラスタには適していなかった。問題は複数のアプリケーションが存在する時のパフォーマンスはほとんど考慮されていなかったことから生じる。

ネットワークに接続された計算機はハードウェアの構成だけでなく、デバイスドライバの構成によってもその性能は大きく変化する。これは、高速ネットワークを通して受信したパケットが、デバイスドライバを介してアプリケーションに渡されるまでの過程におけるパフォーマンスの低下が大きいためであると考えられる。このパフォーマンスの低下の多くはポーリング、通信バッファ、共有メモリなどの手法を用いてデバイスドライバを構成する時にアプリケーションの実行状態などをあまり考慮しなかったことによるものと思われる。

例えば、通信バッファを用いた手法ではアプリケーションの増加に従いパケットの通信バッファへのアクセスやメモリの消費量が増えることにより速度が低下する。また、ポーリングを用いた手法ではアプリケーションの数の増加に伴いデバイスドライバに対するポーリングする時間が増加することにより速度が低下する。その他の手法もこれらと同様な要因で速度の低下を引き起こす。また、プロトコルの実行レベルがユーザレベルであるか、カーネルレベルであるかの影響も効率に大きな影響を与える。

そこで、従来提案されている手法を統合することによりインターネットサーバのような複数のアプリケーションを同時に効率よく実行する必要があるシステムのためのデバイスドライバアーキテクチャを提案し、実装、評価を行なう。

2.3 スケーラブルサーバのための高速ネットワークドライバのアーキテクチャ

図 2の (1) に示すようにアプリケーションとカーネル内のデバイスドライバ間の通信を高速に行なうため U-Net[4] で提案されている手法を使用して API を構成する。また、パケットフィルタとして DPF (Dynamic Packet Filter)[5] を使用することで高速に複数のアプリケーションへのパケットの配達を行なうことを可能とする。

インターネットサーバでは耐故障性を向上するためアトミックマルチキャストのサポートが重要となるが、本研究では図 2の (2) に示すようにアトミックマルチキャストをカーネル内の実装とユーザ空間のライブラリの 2 つの実装を行なうことにより最適な実装法に関して検討する。

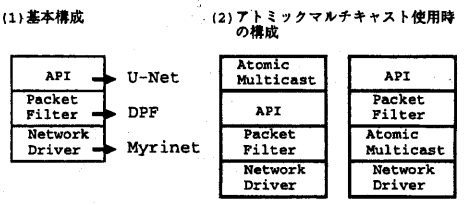

図 2: アトミックマルチキャスト使用時の構成

3 仮想ディスクシステム

3.1 全体構成

構築する仮想ディスクシステムの全体構成を図 3 に示す。我々のシステムでは、通常のディスクデバイスドライバと上位の UNIX サブシステム (ファイルシステム等) の間に仮想ディスクサーバを置くことで、利用目的に応じたディスクシステムを提供する。仮想ディスクサーバはディスクデバイスドライバとして構築することにより、既存の OS やアプリケーションを変更せずに利用することが可能である。

仮想ディスクサーバは、分散キャッシュ層と分散ディスク層から構成される。分散キャッシュ層では、大容量ディスクキャッシュの提供、高速な同期書き込み、遅延書き込み内容を補償するキャッシュのリプリケーションを行ない、高機能なディスクキャッシュを提供する。分散ディスク層では、シーク時間を低減するログディスク管理、各ノードに存在するディスクを分散共有する分散ディスク管理を行なうことにより、利用目的に応じたディスクレイアウトを提供する。

3.2 分散ディスクキャッシュ

計算機の故障時に OS 上のキャッシュ内容を保護/修復するキャッシュシステムとして Rio(RAM I/O) ファイルキャッシュ[7] がある。Rio は、クラッシュ時にメモリ内容をバッテリーバックアップで保護し、

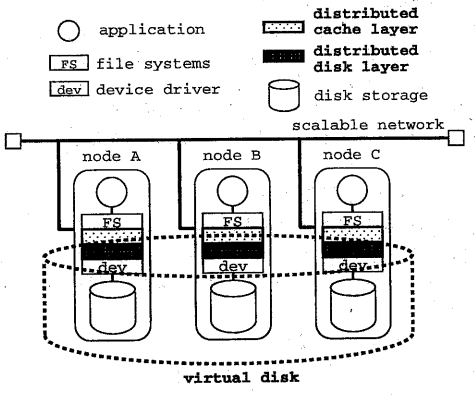

図 3: 仮想ディスクシステムの全体構成

リブート時にその内容を修復することで非同期書き込みを用いて同期書き込み程度のデータの信頼性を得ている。しかし、計算機の故障時にはアプリケーションは停止し、計算機が復旧してからキャッシュが利用可能となるためアプリケーションの持続的な運用はできない。

一方、分散ディスクキャッシュの研究には、集中サーバにおける通信のボトルネックを改善したxFS[8]がある。xFS では各クラスタノード上で大容量ディスクキャッシュを設けて、ディスクアクセスを減少させたことで高速性と可用性を示した。しかし、故障発生時からのキャッシュの復旧については言及しておらず、キャッシュ修復は全てキャッシュミスとして取り扱われるため大容量キャッシュとなった際にはキャッシュミスがシステム全体のパフォーマンスを損ねる心配がある。本研究では以上の経験を踏まえ、次の (1)〜(3) の実現が必要である。(1) 高速同期書き込み：ディスクへの書き込みを非同期で行ない同期並の信頼性を得る。(2) 高可用性：平均修理時間間隔 (MTTR) を短することで更なる可用性を高める。そして、(3) 持続的なキャッシュアクセス：障害発生時にあるノードが停止した際にもアトミックなディスクアクセスが可能。本節では以上の 3 つの特徴を持つアプリケーション側の分散キャッシュの概要について説明をする。

3.2.1 高速な同期書き込み

UFS ではディスクへ書き込みの際、高いスループットでアクセスを可能にするためにデータの書き込みは非同期で行ない、ディレクトリや i-node などのメタデータの書き込みには同期で処理を行なっている。前者は、システムクラッシュによりデータが失われる事が問題であり、後者はデータをディスクに書き込むまでアプリケーションの実行が停止する事が問題となっている。そこで、ノード間でキャッシュの複製を作り、アトミックマルチキャストを使いキャッシュデータを高速に複製し、分散キャッシュを用いることで以上の 2 つの問題を同時に解決することが可能である。

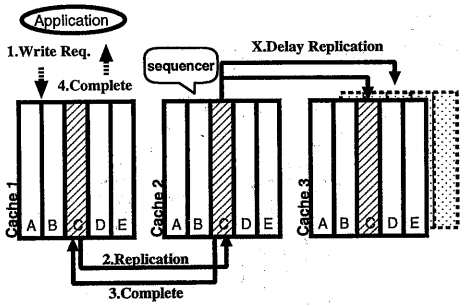

図 4: キャッシュ内容の重複化

具体的には、 以下の 4 つの手順 (図 4) を踏まえる。

(1) アプリケーションより書き込まれた更新データを一時的にキャッシュに書き込む（現時点ではこのキャッシュデータはキャッシュ上では有効でない）(*WriteReq*)。

(2) アトミックマルチキャストのシーケンサノード (図 4の Cache2 を持つノード) 上のキャッシュに書き込み処理を行なう (*Replication*)。

(3) 書き込み処理が終了するとシーケンサノードはキャッシュデータ送信ノードに *Complete* メッセージを発行し、(1) で Cache1 に書き込まれたデータを有効にする。また、アプリケーション側には書き込みが完了したことを通知する (*Complete*)。

(4) シーケンサノードは、書き込まれたデータを他のノードにアトミックマルチキャストを用いて伝搬する (*X.DelayReplicaton*)。

キャッシュからディスクへの書き込みは、後述の分散ディスク管理の章で記述されているように、遅延してキャッシュからディスクへ書き込み処理が行なわれる。

以上のように、アプリケーションはディスクへ直接書き込み処理を行なうので高速に同期書き込みが可能になる。

3.2.2 高可用性

キャッシュ故障のほとんどは電源故障等が原因となる RAM 上に存在するキャッシュデータの喪失である。しかし、キャッシュデータをクラスタノード間で重複化し、クラスタノードの平均修理時間間隔 (MTTR) を短くすることで高可用性が可能である。特に本研究では、大容量のキャッシュを想定しているために故障時には高速な修復が必要不可欠である。

そこで既に他のノードに複製していたキャッシュデータを高速ネットワークデバイス Myrinet を用いてクラスタノードが復旧した後、キャッシュデータをアトミックマルチキャストのシーケンサとなっているノードから全キャッシュデータを高速に複製し、通

常の状態に戻す。

3.2.3 持続的なキャッシュアクセス

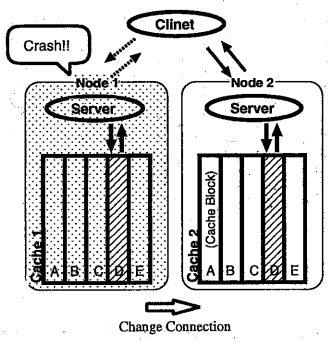

図 5: 持続的なキャッシュアクセス

アプリケーションがクラスタ上のキャッシュデータを持続的にアクセスするためには、各ノード上のキャッシュデータを重複させ、ノードの故障時には故障ノードを回避し、稼働中ノードにアクセス処理を変更する必要がある。本研究では、ノードを利用しているアプリケーションがノードの変更処理を行なう。具体的には、図 5のように *Cache*1 のデータをあらかじめ *Cache*2 で重複しておき、*Node*1 が故障した際にアプリケーションである *Client* が自ら *Server* への接続を *Node*2 へ変更する。*Client* は *Node*2 の *Server*へ接続することで、同じキャッシュデータを持つ *Cache*2 を使用することができる。したがって、障害時にも持続的なキャッシュアクセスが可能となる。

3.3 モジュール群

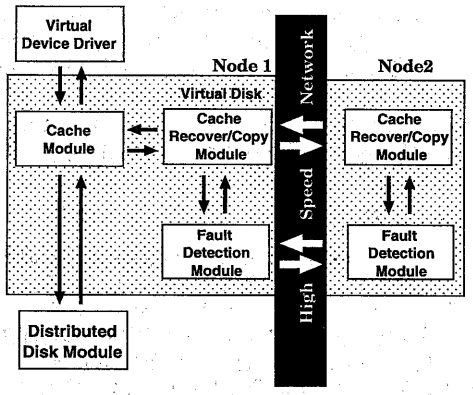

図 6: 分散キャッシュモジュール群

分散キャッシュは、故障監視モジュール、キャッシュモジュール、そして、キャッシュ修復モジュールの 3 つのコンポーネントから構成される。以下にこれらのコンポーネントについて解説する。

(1) 故障監視モジュール

クラスタノードの故障状態を監視する。具体的にはクラスタからシーケンサノードを特定し、シーケンサノードに向けてその他のクラスタノードの故障監視モジュールが定期的に Keep Alive メッセージを送信し、各クラスタノードが稼働していることを確認する。

(2) キャッシュモジュール

大容量揮発性メモリを利用することでクラスタノード間でデータの複製を作り、キャッシュからディスクへの書き込みを遅延することで高速同期書き込みを行なう。

(3) キャッシュ修復モジュール

故障ノードのキャッシュを修復する。修復の手順は、まずシーケンサノードを探し、そのノードのキャッシュデータを用いて複製を行い修復する。

3.4 分散ディスク管理

分散ディスク層では、各ノードに分散しているディスクを単一イメージの仮想ディスクとして扱うために、仮想ディスクと物理ディスクとの対応関係を管理する。図 7に Petal[6] を基に設計した分散ディスク管理のためのモジュール群を示す。

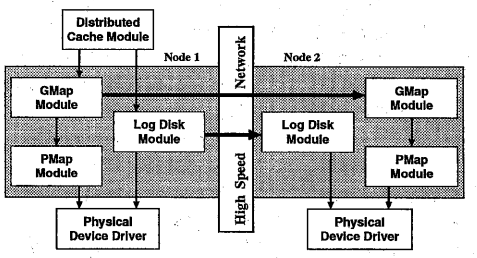

図 7: 分散ディスク層のモジュール群

3.4.1 分散仮想ディスク

各ノードに存在するディスクをクラスタの一部及び全体で共有することにより、仮想的な RAID ディスクのように扱うことができる。共有の方法としては、ストライピング、ミラーリング、分散パリティ、デクラスタリングなど、負荷分散、信頼性向上を目的としたさまざまな手法がある [10]。本システムでは、ディスクレイアウトのポリシーを柔軟に変更できるシステムを構築する。

3.4.2 論理・物理マッピング

分散仮想ディスクの論理ブロック ID を、対応する物理ディスクのブロック ID に変換するときに、以下の 2 つのテーブルを用いる。

- グローバルマップ (GMap)

仮想ディスクの論理ブロック番号から、対応するディスクを持っているノード ID とそのディスク ID を知るためのテーブル。

- 物理マップ (PMap)
  論理ブロック番号から自分のノードで持っているディスクの物理ブロック番号への変換を行なうためのテーブル。各ノードごとのディスクについての固有の情報である。

論理ブロック ID から物理ブロック ID への変換は、次のように行なわれる。

1. グローバルマップモジュールが GMap を参照して、仮想ディスクの論理ブロック ID から、対応する物理ディスクを持っているノードに対して、アクセス要求を渡す。
2. 対応するノードでは、物理マップモジュールが PMap を参照し、論理ブロック ID に対応する物理ブロック ID に対してアクセスを行なう。

冗長性を確保するために、1 つの論理ブロック ID が複数の物理ブロック ID に変換される場合、読みだしに関してはグローバルマップモジュールがアクセス可能なものから 1 つを選び、書き出しに関しては全ての物理ブロックに対して書き出しを行なう。

3.4.3 ログディスク

ログディスクの実装として、Logical Disk[9] におけるログディスクの実装 (Log-structured Logical Disk: LLD) がある。LLD では、書き込み要求のあったダーティなディスクブロックをセグメントと呼ばれるメモリ領域にたくわえる。セグメントが満たされたとき、セグメントの内容をメタデータ (各ブロックの論理ブロック番号、更新時刻など) とともにディスクの連続領域に書き出すことで、シーク時間を低減している。

LLD の問題点として、マシンに障害が起きたときにセグメント中の遅延書込内容が失われてしまうことが挙げられる。書き出し頻度を上げるためにセグメント長を短くすることで被害を小さくすることができるが、セグメント長を短くすることは性能の低下をまねく。

本研究では、分散ディスク層に仮想ログディスクモジュールを実装し、ログディスクを多重化することにより、遅延書込内容の失われない信頼性の高いログディスクの実装を検討する。

3.5 故障発生時の取り扱い

本研究で提案する分散仮想ディスクシステムは、多様な故障に対する耐故障性を支援する。耐故障性の言葉の範囲には、単一のハードウエア要素 (ディスク, 入出力チャネル、入出力制御装置) の故障からシステムソフトウエアあるいはアプリケーションソフトウエア (プログラムによるプロセッサの停止、ユーザプロセスの誤り、トランザクションアボート) までが含まれるが、本稿では、クラスタノードにおける 1 台以上のプロセッサの喪失に限定する。ここでいう喪失は、故障したクラスタノードのメモリに格納された全てのデータが無効になることを意味することにする。

以上より本稿で採用されるモデルは、FailFast、つまり、各ノードで自己検査を行うことでそのノードが正常に稼働するか、稼働しないかのいずれかであるモデルを採用した。以降の節ではこの故障モデルを用いて分散キャッシュ、並びに、分散ディスクの故障時の取り扱いについて検討する。

3.5.1 分散キャッシュの扱い

本研究では、ディスクへ高速なアクセスを行なうためにクラスタノード上でキャッシュデータの重複による同期書き込みを行い、アプリケーション側からはキャッシュをハードディスク並の安定記憶装置として見立てることを可能にした。この時、クラスタノード障害時にこの重複キャッシュデータ用いて迅速な修復が可能であり、高い可用性を得ることができる。

具体的には、前述のアトミックマルチキャストで用いられているグループ間通信をモデルに用いる。まず、故障したクラスタノードを修復する際に故障クラスタノードがアトミックマルチキャストのシーケンサノードを特定する。このシーケンサノードは必ずクラスタ内の最新キャッシュデータを保持しているので、このキャッシュデータを元にして複製を行ない、故障ノードを迅速に修復することができる。

3.5.2 分散ディスクの扱い

ディスクの多重化が行なわれている場合は、故障ディスクの隠蔽を行なう。分散キャッシュ層の故障監視モジュールを用い、故障ノードの検出を行ない、GMap から故障ノードを削除し、同一内容のデータブロックを持っているノードに対してアクセスを行なうようにする。多重化しているディスクを故障から復旧させる場合、正常に稼働しているディスクからデータの操作を行ないデータの一貫性を回復させる必要がある。このとき、ログディスクとして扱っているディスクについては、通常のブロック単位でのディスクとは異なり、メタデータをもとに必要なブロックのみ転送を行なえばよいと考えられる。

4 応用システム

ネットワークファイルサーバ (NFS) の応用を検討した。本研究で提案する手法を NFS の大容量ストレージに用いた (図 8)。仮想ディスクシステムをストレージとして用いることで従来の NFS で損なわれていた高速な同期書き込みとシステムの耐故障性の向上が可能となる。FT-NFS での NFS サーバは仮想デバイスドライバを通じて、分散仮想ディスクに

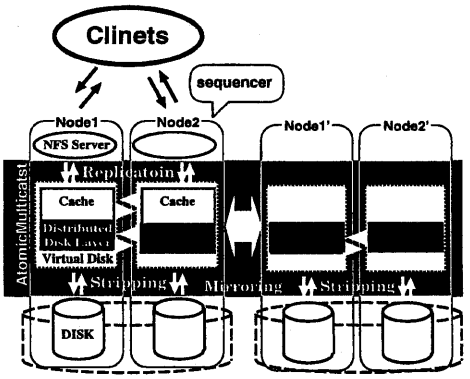

図 8: FT-NFS Server

アクセスを行う。仮想ディスク中のキャッシュは完全に同じデータを保持しているので、例え、*Node1* が故障しても *Node2* を NFS クライアントが選択することにより、持続的に NFS サーバへアクセスすることが可能である。また、*Node1* の修復時にもキャッシュデータとディスクデータの修復をしなければならないが、シーケンサノードとなっている *Node2* に問い合わせて *Node2* のキャッシュデータの複製とログディスクのジャーナリングを行なうことにより迅速に修復することが可能である。

一方、現在検討しているシステムは耐故障性の観点から次の 2 つの問題がある。(1)NFS サーバはクライアントのステート情報を保持するため、万一 NFS サーバのノードが故障した際にはステート情報を失い、クライアントサーバ間の通信が復旧できない。(2) 一意の NFS サーバに接続していたクライアント群は、NFS サーバノードの故障と同時に NFS サーバも異常終了するので持続的に NFS サーバにアクセスできない。これらの問題を解決するために、(1) についてはサーバのステートを常に他のサーバで保持する機構を設ける。故障したノードを復旧する際にはそのステートを複製し NFS サーバ状態を元に戻すことでクライアントサーバ間の通信を復旧させる。また、(2) については NFS サーバクライアント間の通信機構として、既存の物理 IP アドレス上に一意な論理 IP アドレスを導入する。したがって、一部のクラスタノードが異常終了した後もクライアントが一意な論理 IP アドレスを使用することで自動的に別のクラスタノードを選択し、持続的な運用を可能にする機構を今後検討する。

5 関連研究

高速性、耐故障性、及び、高可用性等を得る研究として Petal [6]、xFS[8] がある。Petal はクラスタノード間でディスク内容の冗長性を持たせて、耐故障性と高速性の向上を行なった。また、xFS においてはクラスタノード間で協調キャッシュを行いディスクへのアクセスを極力少なくすることでストレージシステム全体のパフォーマンスの向上を行なっている。

これらの研究と本稿の分散仮想ディスクとの最大の違いは、Petal 並びに xFS では耐故障性と高速性について議論したが、本稿の分散仮想ディスクでは上記の耐故障性と高速性を満たしながらも高速なネットワーク Myrinet と耐故障性メッセージプリミティブであるアトミックマルチキャストを用いてクラスタノード故障時に迅速に復帰することができる可用性の高いシステムである点である。

6 結論

本稿では、OS の変更を必要としないデバイスドライバレベルでの実装によりソフトウェア RAID を実現できる仮想ディスクシステムという新しい PC クラスタの概要について説明した。そして、この仮想ディスクシステムでは、既存の PC クラスタにおける問題点が改善でき、効率良く実行することが可能となる手法を提唱した。

我々の仮想ディスクシステムでは、アクセス時間と冗長度を最小限に抑えたシステム、耐故障性を十分に備えたシステムの構築を可能としている。しかし、ステートを持つサーバは仮想ディスクだけでは扱うことができずサーバソフトウエア自体の変更も必要となるが、ステートレスなら仮想ディスクを用いて PC クラスタ上でサーバソフトウエアを変更せず効率良く実行できる可能性がある。

現在、仮想ディスクシステムを PC に実装するための準備を進めている。それと同時に、仮想ディスクシステムの有効性を示すためにも、これらの耐故障性、及び、高速化手法の最適化と検証を進めて行くつもりである。

参考文献

[1] A.S. タネンバウム=著, 水野忠則+鈴木健二+宮西洋太郎+佐藤文明=訳, 分散オペレーティングシステム, プレンティスホール出版, ISBN4-931356-21-4, Page 378-387, 1996.

[2] N.J.Boden,D.Cohen,R.E.Felderman,A.E.Kulawik, C.L.Seitz,J.N.Seizovic and Wen-King Su, *Myrinet - A Gigabit-per-Second Local-Area Network*, IEEE MICRO Volume 15 Number 1 February 1995.

[3] http://www.myri.com.

[4] http://www.cs.cornell.edu/Info/Projects/U-Net.

[5] Dawson R.Engler,and M.Frans Kasshoek, *DPF:Fast,Flexible Message Demultiplexing using Dynamic Code Generation*, ACM SIGCOMM '96 CONFERENCE Volume 26 Number 4 October 1996.

[6] Edward K.Lee and Chandramohan A.Thekkath *Petal: Distributed Virtual Disks* ACM SIGPLAN(ASPLOS VII) Volume 31 Number 9,Page 84-92, September 1996

[7] Christopher Aycock, Gurushankar Rajamani, David Lowell K.Lee and Chandramohan A. Thekkath *The Rio File Cache: Surviving Operating System Crashes* ACM SIGPLAN(ASPLOS VII) Volume 31 Number 9,Page 74-83, September 1996

[8] Thomas E.Anderson,Michael D.Dahlin,Jeanna M.Neefe, David A.Patterson,Drew S.Roselli, and Randolph Y.Wang *Serverless Network File Systems* ACM Symposium on Operating Systems Principales(December 1995)

[9] Wiebren de Jonge *The Logical Disk: A New Approach to Improving File System* page 15-28, ACM SIGOPS 12,1993

[10] Gart A. Gibson *Redundant Disk Arrays*,Page 34-93 The MIT Press, 1991