ハイパフォーマンス
コンピューティング 77−11
（1999. 8. 2）

冗長分散格納によるPCクラスタ上の動的負荷均衡化

小西 弘一　相場 雄一　河合 英紀†　大町 健一‡
赤峯 享†　菊地賢太郎　福島俊一†[1]

NEC C&Cメディア研究所　†NEC ヒューマンメディア研究所
‡NEC 情報システムズ

無共有 (Shared nothing) 型のサーバクラスタにおいて動的IO負荷分散とノード障害対応を可能にするためにデータを分散冗長格納する一方式を提案する。従来の分散冗長格納との違いはストライピングと複製を異なる大きさのブロックを単位として行う点にある。筆者らはこれにより正常運用時の性能に影響するストライプ単位と一部ノードが停止している縮退運転時の負荷分散に影響する複製単位を別々に最適化することを狙っている。今回は全文検索システムに適用して正常運用時の性能を評価した結果を報告し、ディスクキャッシュの取り合いを緩和するためには原本優先のスケジュールが必要なことを示す。

Dynamic Load Balancing based on Distributed Redundant Storing on a PC Cluster

Koichi Konishi　Yuichi Aiba　Hideki Kawai†　Ken'ichi Ohmachi‡
Susumu Akamine†　Kentaro Kikuchi　Toshikazu Fukushima‡

NEC C&C Media Labs.　†NEC Human Media Labs.
‡NEC Informatec Systems

This paper presents a novel scheme of distributed redundant data storing which allows dynamic I/O load balancing and fault-tolerance in shared-nothing PC clusters. The scheme differs from conventional ones in that it stripes across nodes a file in one block size while it stripes the file's replica in a different size. The authors expect the scheme allows independent tuning of the two sizes, the former for performance in normal operation and the latter for load distribution in reduced operation. Results of preliminary evaluation of the scheme applied to a parallel full-text search system shows that the scheduler must have some preference for originals over copies in order to alleviate contention for disk cache.

[1]Email: {konishi,aiba,kikuchi}@ccm.cl.nec.co.jp, {kawai,akaimine,fuku}@hml.cl.nec.co.jp, ohmachi@ats.nis.nec.co.jp

1 はじめに

クラスタ技術はサーバシステムの高信頼化技術として以前から使われており、近年はIntel系PCサーバ製品[4]にも適用されている。System Area Network(SAN)と総称される高速ネットワークが共用品化[2]し標準化[3]された結果、PCサーバクラスタは数十台規模の高性能並列サーバに向かっている。しかし、現在のPCサーバクラスタのスケーラビリティには疑問がある。

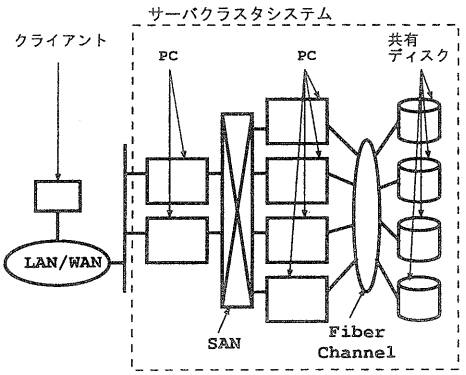

図 1: 共有ディスク型サーバクラスタ

現在のPCサーバクラスタ製品は原則的にFiber ChannelとRAID装置を用いる共有ディスク型(図1)である。これについて筆者らは以下のように考える。この構成ではクラスタ規模が大きくなるにつれFiber ChannelとRAIDが性能のボトルネックとなりその解消のためのコストがかさむ。高いスケーラビリティを達成するには無共有(Shared nothing)型構成の(図2)クラスタがよい。

しかし単純な無共有型では、共有ディスク型ならば容易な負荷均等化やノード障害対応が難しい。高負荷ノードや停止ノードが持つローカルディスクに他のノードがアクセスするのは困難で、そこにしかないデータを必要とする処理は基本的に他のノードに割り当てられない。

負荷均等化や耐障害性はデータを分散冗長格納すれば実現できる。この方式は例えばビデオサーバ向けの並列ファイルシステムTiger Shark[7]が用いている。しかしTiger Sharkの方式は個々の要求が巨大なビデオデータ全体を先頭から最後までアクセスする場合を想定している。インデクス検索など、個々のアクセス対象が比較的小さくアクセス順序が不定の場合の有効性は明らかでない。

本稿ではインデクス検索等に適した分散冗長格納形式として二粒度冗長分散格納を提案し、これを用いて全文検索サーバを実装して予備的な性能評価を行った結果を報告する。次の節では冗長分散格納に関する従来手法について述べる。第3節では二粒度冗長分散格納を説明する。第4節では本方式を用いた全文検索サーバの構成を示す。第5節では評価基準と評価手法を述べる。第6節で結果を示す。最後に本稿のまとめを示す。

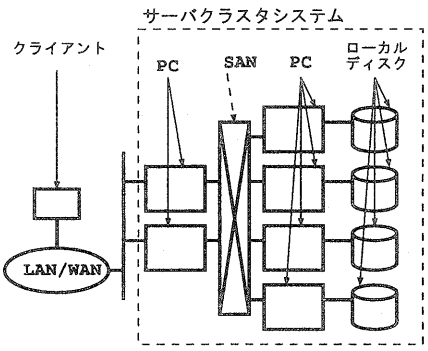

図 2: 無共有型サーバクラスタ

2 冗長分散格納

ファイルの分散配置や冗長配置は従来から分散ファイルシステム、分散HTTPサーバ、RAID装置[1]、ビデオサーバシステム等で行われている。

例えばTiger Sharkは高いビデオデータ供給スループットと耐故障性を得るためにビデオデータなどのマルチメディアストリームファイルをストライプと呼ばれる断片に分けて各断片を複数ノードのローカルディスク群に分散格納する。ファイルの複製も原本と同じ順序で、ただし原本とは異なるランダムに選んだノードを起点として格納する。二つのストライピング方針がある。一つはラウンドロビンで、全ファイルに同じ順序を用いる。もう一つは均等ランダム(図4)で、k個のディスクがある時k個の連続するブロックごとに異なるランダムに選んだ順序で格納する。

均等ランダムでは一つのファイルを構成するブロックの数nが$n \gg k$を満たすならば、同じディスクにあるブロックの複製が全ディスクに分散配置される。このためあるノードが停止したとき、

ブロック ID	1	2	3	4	5	6	7	8
原本	3	2	0	1	3	2	0	1
複製 1	0	1	3	2	0	1	3	2
複製 2	2	0	1	3	2	0	1	3

図 3: ラウンドロビン: 表中の数字は配置先ノード ID

ブロック ID	1	2	3	4	5	6	7	8
原本	3	2	0	1	0	3	1	2
複製 1	0	1	3	2	2	0	3	1
複製 2	2	0	1	3	3	1	2	0

図 4: k=4 の場合の均等ランダム

そのノードの分の負荷を残りの全ディスクで均等に負担することができる。これに対しミラリングではあるディスク上のブロックの複製はすべて他の一つのディスクにある。したがって、あるディスクが故障するとその分の負荷はすべて一つのディスクが担わなければならない。

均等ランダムストライピングはビデオストリームファイルのような巨大なファイルを先頭から最後まで順に読み出す処理に対しては有効である。しかし、巨大なファイル中の様々な位置に不特定の順序で小さな単位のアクセスを繰り返す処理には適用しにくい。後者の処理では一つのアクセスが複数のノードに渡らないようストライプを大きくとる方が効率がよい。しかしそうすると一ファイルを構成するブロックの数が少なくなり、$n \gg k$ を満たさなくなる。すると各ブロックの複製の分散配置に、また従って停止ノードの負荷の負担にも偏りが生じてしまう。

TigerShark では動的な負荷分散の必要がないので複製は耐故障性の実現にしか使われていない。しかし複製は動的な負荷分散にも利用できる。つまり原本へのアクセス負荷が高い場合に複製を使えばよい。書き込みアクセスに関し原本と複製を使い分けるには整合性維持のための操作が必要であるが、読み出しに関してはその必要がない。したがって検索処理のように読み出しアクセスがほとんどである場合は複製を利用する動的な負荷分散を行うことに利点があると考えられる。

次の節ではアクセス単位が小さくアクセス順序が不特定の場合にも有効な分散冗長格納方式として二粒度分散冗長格納を提案する。

3 二粒度分散冗長格納

n 個のノードからなる無共有型クラスタにおいて、各ノードが一つずつローカルディスクを持つとする。全部で n 個のローカルディスクがある。

大断片 ID	0		1		2		3	
小断片 ID	1	2	3	4	5	6	7	8
原本	0	0	1	1	2	2	3	3
複製 1	3	1	0	2	1	3	2	1
複製 2	1	2	2	3	0	1	1	2

図 5: 二粒度分散冗長格納:$N = 4, M = 2$ の場合

クラスタ全体で扱おうとするデータをまず N 個の大断片に分割し、さらに各大断片を M 個の小断片に分割する (図 5)。そして同一大断片に由来する M 個の小断片 (以下この小断片のグループを大断片と呼ぶ) が同じディスクに載るよう小断片を配置する。次に各小断片の複製を作り原本とは異なるディスクに配置する。このとき同一大断片中の小断片の複製がすべて異なるディスクに載るようにする。この冗長性は動的負荷分散およびフェイルオーバによる高可用性の実現に利用する。

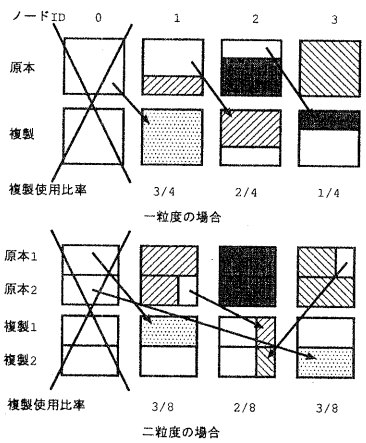

図 6: 縮退運転時の複製利用比率

従来の冗長分散格納との違いはストライピングの単位とミラリングの単位を分離した点にある。本方式ではストライピングの単位は大断片、ミラ

リングの単位は小断片である。ストライピング単位は並列IOの性能に主に影響するのに対し、ミラリングの単位は動的負荷分散操作の自由度に影響する。従来の冗長分散格納ではこの二つの単位を一つのものとして扱っていたため、並列IO性能を最適にする値と負荷分散効率を最適にする値が異なる場合に問題が生じた。二粒度分散冗長格納の狙いはこれらを別々に最適化することにある。

また筆者らは以下のことを期待している。二粒度分散冗長格納は一粒度に比べ縮退運転時の各ノードでの複製利用比率が低い。このためディスクキャッシュの入れ換えが小さく、縮退運転への移行時の性能劣化を押えることができる。図6に動的負荷分散実施時に4ノードが3ノードに縮退した場合について予想した原本と複製の利用比率を示す。一粒度では複製利用比率が3/4に達するのに対し二粒度では3/8以下に押えられる。

本方式の適用先としてはファイルシステムにおけるブロック管理および特定のアプリケーションにおけるファイル管理の両方が考えられる。次節以降では後者の例として並列全文検索サーバへの適用とその性能評価結果について述べる。

4 並列全文検索サーバ

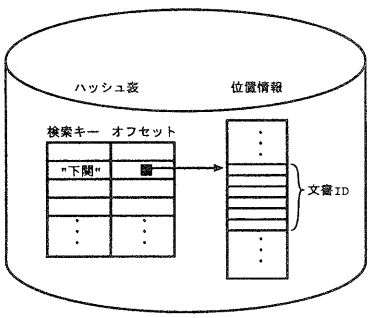

図 7: インデクスファイルの構造

全文検索サーバは大量の文書群の中からクライアントが指定した文字列を含むすべての文書を検索してその文書IDの一覧をクライアントに返す。大規模な全文検索サーバの例にはWeb検索サービスがある。最大級のものはワークステーションクラスタ[8]や並列マシン上[9]に実装されている。

インデクスを用いた全文検索[6]ではあらかじめ検索対象文書に現れる全キーについてそれを含む文書のIDのリスト(位置情報)を作成する。検索サーバのディスクにはこの位置情報と、各キーに対応する位置情報がディスク上のどこに格納されているかを示すインデクスを格納する。

今回用いた全文検索ライブラリ[5]はキーとして検索対象文書に現れるn文字組(漢字か仮名かなどの字種ごとに異なるnを用いる)を用い、位置情報のインデクスとしてハッシュ表を持つ(図7)。これらを格納するファイル群をまとめて以下ではインデクスファイルと呼ぶ。

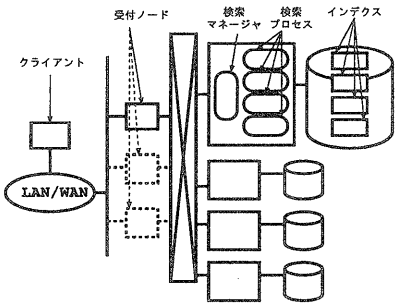

図 8: 並列全文検索サーバ

評価に用いた並列全文検索サーバの構成を図8に示す。サーバは受付プロセス、検索マネージャ、検索プロセスの三種類のプロセスで構成される。受付がいるノードを受付ノード、検索マネージャ他がいるノードを検索ノードと呼ぶ。各検索ノードのローカルディスクにはいくつかのインデクスファイルを格納する。複数の受付ノードを設けることができるが、今回は一つとした。

二粒度冗長分散配置は検索対象文書群を分割してその結果ごとに独立にインデクスを作る形で適用した。つまりノード数N, 細分割数Mの時文書単位に$N \times M$個の小断片に分割し、各小断片ごとに独立にインデクスを作成する。ある検索要求(クエリ)に対する検索結果は全小断片を用いて個別に検索した$N \times M$個の結果を合わせたものである。評価に用いたインデクス配置は後述する。

受付プロセスはクライアントからあるクエリを受け付けると、各々が異なる小断片に対応する$N \times M$個の部分クエリを生成し、その一つ一つを対応する小断片のインデクスを持つ検索ノードの検索マネージャに投げる。各部分クエリに対応するインデクスの原本または複製の中のどれを用いて処理するかの決定(スケジュール)も行う。そして全部分クエリに対する検索結果を受け取ってクライ

アントに返す。

検索ノード上では各検索プロセスが特定の一つのインデクスによる検索を担当する。検索マネージャは検索プロセスに部分クエリを転送する。インデクスごとに検索プロセスを設けたのは一検索プロセス中で複数のインデクスを扱う困難を避けるためである。検索マネージャで部分クエリを中継するのは、受付と検索ノード間の通信に用いた高速通信ライブラリでは同一ノード上の複数の検索プロセスと受付の間で直接通信が行えないためである。

部分クエリのスケジュールは受付がクエリの受付時と検索ノードからの結果を受信した時に行う。各部分クエリごとに、それを処理できるノードは二つある (冗長度 (複製の個数)1 の時)。受付は両ノードの未処理クエリ (そのノードに対し処理を要求したがまだ終了していないもの) 数を比べ、少ない方のノードに部分クエリを投げる。ただし複製側の未処理クエリ数には**原本優先度 p** を掛けてから比較する。p が大きいほど原本に投げるクエリの割合が高くなる。また、未処理クエリ数がある閾値 (最大未処理クエリ数) より大きい、忙しいノードにはクエリを投げない。このため投げる先がない部分クエリが発生することがあり、これは未決キューに溜める。ある検索ノードから結果を受信すると、未処理クエリ数が一つ減るので、そのノードにスケジュールできるクエリが未決キューにあれば、原本と比べて原本の方が短くない限り、そのノードに投げる。

5 評価方法

検索サーバのスループットを従来方式 (一粒度) と二粒度の冗長分散格納について測定し比較した。また両方の場合の負荷分布を測定した。

5.1 評価環境

表 1: 各ノードの仕様

	台数	プロセッサ	メモリ量
A 群	8	400MHz Pentium II	512MB
B 群	16	200MHz Pentium Pro	128MB
C 群	8	166MHz Pentium	64MB

評価に使ったクラスタは Myrinet[2] で 32 台の PC を接続したものである。各ノードの仕様を表 1に示す。すべて Linux マシンでカーネルは 2.0 である。Myrinet 上の通信には FM2.02 の上に載せた MPI を用いた。この MPI 通信の性能は B 群のマシン間で片道遅延 20μ sec、最大バンド幅約 50MB/sec である。さらに 100BaseTX が全ノードと FreeBSD マシンを結んでいる。

表 2: スループット (クエリ / 秒)

スケジュール方針			ノード数	
	M	p	1	8
静的	1		37.7	28.6
動的	1	1.0	—	24.7
動的	1	1.2	—	27.6
静的	2		17.5	15.7
動的	2	1.0	—	11.9
動的	2	1.2	—	14.3

5.2 測定方法

検索対象文書にはロボットがインターネットから集めた Web ページ、クエリにはある公開 Web 検索サイトが受け付けたクエリログから取り出したものを用いた。

10000 クエリを FreeBSD マシン上のクライアントから連続投入し、投入開始から全クエリに対する応答を受けとるまでの時間をクライアント上で測定した。同時に各検索プロセスが 10000 個のクエリ処理に費やした時間を測定した。測定前には全ノードをリブートしてディスクキャッシュを空にした。文書数はノード当たり 15 万件、インデクスにして約 300MB とし、ノード数は 1, 8, 細分割数 M は 1, 2 、原本優先度 p は 1.0, 1.2 、スケジュール方針は静的および冗長度 (複製の数) 1 での動的、最大未処理クエリ数は 50 とした。

6 評価結果

表 2に各評価でのスループットを、図 9、図 10、図 11、図 12に各評価での負荷分布を示す。

原本優先度 p を 1.0 に取ると、動的負荷分散によって負荷分布は均等になったがシステムのスループットはかえって下がった。これは一つのノード上で原本と複製それぞれへのアクセスがキャッシュ領域を取り合ったからだと考えられる。p=1.2 とすることでこの競合は緩和されたが、静的負荷分散のスループットを越えることはできなかった。さらに大きな p や、未処理キュー長以外の指標によるスケジュール方針による評価が必要である。

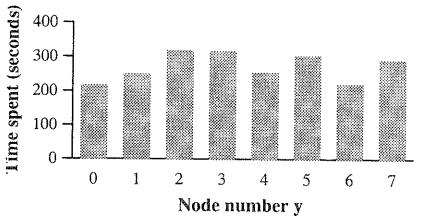

図 9: 静的負荷分散、 M=1

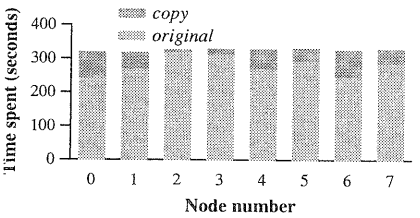

図 10: 動的負荷分散、 M=1, p=1.2

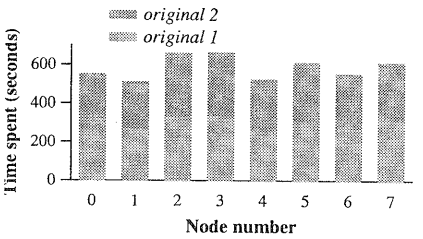

図 11: 静的負荷分散、 M=2

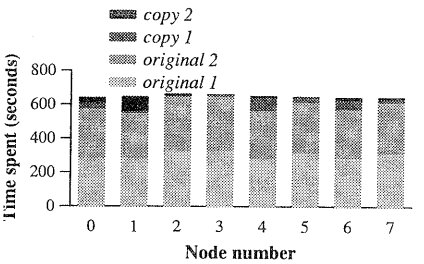

図 12: 動的負荷分散、 M=2, p=1.2

静的負荷分散においてM=2とするとM=1に比べてスループットは半減した。この理由は、一ノード上の二つの小断片原本に対し個別にアクセスを行ったため、読み出し量としてはM=1の時と変わらないもののアクセス回数、したがってシーク時間が2倍になってしまったためと考えられる。この問題の回避策としては、原本は大断片のまま格納して一つのインデクスとしてアクセスするようにし、複製のみを小断片に分割格納する方法が考えられる。

7 おわりに

無共有クラスタにおける並列I/Oの動的負荷均等化のための二粒度分散冗長格納方式を提案し、並列全文検索サーバに適用して予備的な性能評価を行った結果を述べた。本方式は、ストライピング幅とミラリングの単位を別々に決めることにより、正常運転時の高性能と縮退運転時の負荷均等化を独立に最適化することを狙いとする。評価の結果、原本を優先しないとスループットが大きく下がること、原本優先度が1.2程度ではもっとも高い負荷に負荷分布が揃ってしまうことが判明した。今後はスケジュール方針の改良を試み、また縮退運転時と受付を複数にした場合についても評価を行う予定である。

参考文献

[1] D. A. Patterson, G. Gibson, R. H. Katz, *A Case for Redundant Arrays of Inexpensive Disks(RAID)*, Report No. UCB/CSD 87/391, University of California, Berkeley, CA 1987.

[2] *Myricom Home Page*, `http://www.myri.com`

[3] VI Architecture, `http://www.viarch.org`

[4] CLUSTERPRO, `http://www.ace.comp.nec.co.jp/product/3rd/cluster/page/index.htm`

[5] 福島、赤峯、会森、田村、柴田、中塚、大規模テキスト並列検索エンジン *RetrievalExpress (1)* 並列検索方式、情処第55回全国大会, 4N-07, 1997年

[6] 岩波ソフトウェア講座、「自然言語処理」 pp.431-pp.434 11.4 全文検索 (b) インデックスを用いた全文検索

[7] R. L. Haskin, *Tiger Shark — A scalable file system for multimedia*, IBM J. Res Develop. Vol.42 No.2 March 1998

[8] Inktomi, `http://www.inktomi.com`

[9] *Web*検索サービス: *InfoNavigator*, FUJITSU, Vol 49, No.5, pp. 353-357, 1998