データベースシステム 128－28
（２００２．７．１８）

非同期バックアップにおけるログ格納の高速化の影響

安部洋平 †[1] 宮崎 純 †[2] 横田治夫 †[3]
†[1] 東京工業大学 大学院情報理工学研究科 計算工学専攻
†[2] 北陸先端科学技術大学院大学 情報科学研究科
†[3] 東京工業大学 学術国際情報センター
†[1]yabe@de.cs.titech.ac.jp, †[2]miyazaki@jaist.ac.jp, †[3]yokota@cs.titech.ac.jp

あらまし 大規模ストレージシステムの管理を効率的に行う一つのアプローチとして, 我々は自律ディスクを提案している. 自律ディスクはネットワークに直接接続されてストレージクラスタを構成し, 集中コントローラを持つことなくクラスタ単位でアクセスを効率よく処理する. さらに負荷分散や冗長化といった様々な管理もクラスタ内で処理することで高いアベイラビリティとスケーラビリティをもたらす. 自律ディスクではデータの信頼性を確保するためにクラスタ内でプライマリとバックアップの二箇所にデータを配置する。また, 同期更新ではなく非同期更新を行なうことによってスループットを犠牲にすることなくデータを二重化する. 非同期更新を行なうシステムの構築方法は数種類存在し, 各方法の評価が必要である. 本稿では自律ディスクでの非同期更新を実現する方法について検証し, 各方法をスループット, 価格対性能比の観点から評価する.

キーワード 自律ディスク, ネットワーク接続ディスク, 分散ディレクトリ, クラスタ, 非同期更新

Influence of improvement in the speed of log storing in asynchronous backup

Youhei Abe†[1] and Jun Miyazaki †[2] Haruo Yokota†[3]
†[1] Department of Computer Science, Tokyo Institute of Technology
†[2] School of Information Science, Japan Advanced Institute of Science and Technology
†[3] Global Scientific Information and Computing Center, Tokyo Institute of Technology
†[1]yabe@de.cs.titech.ac.jp, †[2]miyazaki@jaist.ac.jp , †[3]yokota@cs.titech.ac.jp

Abstract We propose autonomous disks to enable distributed control in the storage-centric configurations. Data is arranged to two places, a primary and backup, within a cluster. Moreover, data is doubled, without sacrificing a throughput by performing not synchronization but asynchronous backup. Some kinds of construction methods of a system using asynchronous backup exists, and evaluation method is required for them. This paper estimates systems from a viewpoint of a throughput and cost performance.

Keywords autonomous disks, network disks, distributed directory, asynchronous backup

1　はじめに

近年コンピューターを通して扱う情報量は増大し, その結果, 情報の記憶装置であるストレージの管理コストの急騰という新たな問題が生じた. この問題の解決策の一つとして, ストレージをシステムの中心に据える構成が注目されている.

ストレージ中心システムが管理コストを削減するための問題のうち, 最も重要なものはストレージ間のデータ信頼性の問題である. データの信頼性を上げる方法の一つとして, データを二重化しプライマリとバックアップの二箇所に配置する. バックアップデータの更新をプライマリと同期して行なうとバックアップの更新が完了するまでデータ挿入が完了せず, スループット低下の要因になる. よってバックアップ更新はシステムアイドル時などに, 非同期に行なう方法が考えられる.

データのバックアップは従来, サーバー側で行なわれストレージ側で自律的に行なわれることはなった. しかしシステムの規模の増大にともない, ストレージが巨大になることは避けられず, そのバックアップをサーバー側で行なうことは管理コストの増大を招く. もちろんサーバー側のソフトウェアの設定により, バックアップを自動化することは可能である. しかし, バックアップをサーバー側で行なうと, OS や管理用ソフトウェアに依存し, 新たにストレージ容量を追加した場合などに容易に対応できない.

これらの問題は, ストレージが自律的に処理を行なえないことに起因する. 受身のストレージは, サーバーからの命令を受けるまで動作を行えない. そのため, 管理用のサーバへの依存を断ち切れない.

一方で, ディスク上の高性能なディスクコントローラを用いた高機能ディスクの研究が活発に行われている [1, 2, 3]. ディスクコントローラ上の演算能力の一部をアプリケーションの実行に用いることでストレージ上のデータを最もデータに近い場所で処理することが可能となり, データマイニングなどのアプリケーションに用いた際に高い性能を発揮することが確認されている.

我々はこの演算能力をストレージの自律的な管理に用いるアプローチが有効であると考えている. ネットワーク上のストレージがそれぞれ自律分散的な管理を行うことで, サーバに依存しない管理が可能となる. 我々はこのような背景の基で自律ディスク [4] を提案している.

我々はこれまでに自律ディスクの様々な機能の提案や, その機能を検証するため自律ディスクの機能をシュミレーションにより検証してきた [5, 6, 7].

自律ディスクはストレージクラスタを構成し, その各ノードがアクティブに動作することでネットワークの通信コストとホストの負荷を下げる. データの信頼性向上のためにクラスタ内でデータを二重化し, バックアップデータの更新には非同期更新を用い処理速度の向上を計る. 以前にシュミレーター上に実装された自律ディスクでの非同期更新は主に更新に必要なログを単一のディスクに格納する方式を採用してきた [7]. しかし, 非同期更新を実現するためのシステムの構成は従来の方法以外にも方式が考えられ, 各方式の検証が必要である.

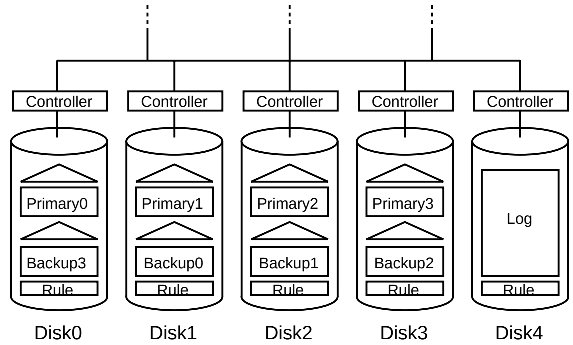

図 1: 5 台の自律ディスクからなるクラスタの構成例

本稿では非同期更新のためのシステム構成を議論し, その評価を行なう. 評価方法としてはシステムのスループット, 価格対性能比 (コストパフォーマンス) を採用した.

まず 2 で自律ディスクについて説明し, 3 で自律ディスクの持つ機能のうち, 非同期バックアップについて述べる.4 では非同期バックアップのための各種システム構成について議論する. 5 では各構成のコストパフォーマンスを比較する. 6 で本稿のまとめと今後の課題を述べる.

2　自律ディスク

図 1 の例は, 独立したログディスクを用いた 4 台のデータディスクと 1 台のログディスクからなるクラスタの構成例である.

- **ネットワーク上でストレージクラスタを構成** クライアントからのアクセスをクラスタとして処理する. そのため, クライアントから見たクラスタの受け口が多く, 高いスループットを有する.
- **高機能な分散ディレクトリを採用** 各ディスクが分散ディレクトリを持つことで, クラスタへの同時アクセスの平行処理が可能となる. また, クライアントはクラスタ内の任意のディスクにアクセスするだけで実際にデータが格納されているディスクから結果が返ってくる.

- **ストリームインターフェース** 従来のブロックアドレスベースのインターフェースとは異なるストリームインターフェースを用いる. この結果, データオブジェクトの論理的な取り扱いが可能となる.

- **ログを介した非同期プライマリバックアップ** 高いアベイラビィリティを実現するためにはクラスタ内にバックアップを保持する必要があるが, バックアップ操作は更新処理のレスポンスを悪化させる. そのため, Write-Ahead-Log (WAL) プロトコルに従い非同期バックアップを行う事でレスポンスの悪化を避ける. バックアップは論理バックアップを想定している.

3 非同期バックアップ

クラスタ内でのバックアップは非同期に行なわれる. 図 2 はこの概念を示したものである. クライアントから送られた更新コマンドがデータディスク 1 に送られると, データディスク 1 はまずログディスクにログを送り, ログ書き込み完了を確認してから実際の更新をする (Write-Ahead-Log プロトコル). ログ書き込み完了通知を受け取った後, データの書き込みを行い, クライアントに対し完了通知を行なう. この時点でクライアント側では更新処理が完了したと判断するが, まだバックアップは更新されていない. ログディスクはシステムアイドル時やログディスク内の容量が欠乏した時などにログをバックアップに送りバックアップの更新処理を行なう. プライマリデータとバックアップデータの更新は非同期に行なわれる. バックアップの更新が完了したあとはログディスク内のログは必要なくなるので消去される. よって, ログを格納するにはディスクではなく, 信頼性が確保されればメモリなどのデバイスを用いる構成も可能である.

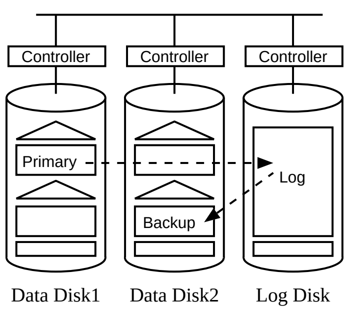

図 2: 非同期バックアップの例

4 非同期更新のためのシステム構成

4.1 分散ディスクログ方式

まず, 最初に考えられる簡単な構成方法としてデータとログを格納するディスクを分けないで構成する方法がある. この方式を分散ディスクログ方式と名付ける. この概念を図 3 に示す.

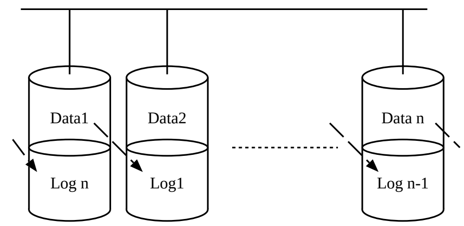

図 3: 分散ディスクログ方式

図の Data 1 の領域に対応するバックアップは単一障害に耐えるために異なるディスクの領域 (図では Log 1) に書き込まれる. この方式の利点としては, データとログを同一のディスクに格納するためディスク容量を効率良く利用できるということがある. しかし, データとログとはディスク内で異なる領域に存在するため書き込みのたびにシークが発生し, スループットが低下するという欠点がある.

4.1.1 分散ディスクログ方式性能の見積り

分散ディスクログ方式の性能を評価する. 評価指標にはシステムの平均スループット T_{avg} $(operation/sec)$ とコストパフォーマンス $CP = \frac{T_{avg}}{price\ of\ system}$ を用いる. 平均スループットは次の式で導かれる.

$$T_{avg} = \frac{t_1 \cdot T_{system} + t_2 \cdot T_{system\ backup}}{t_1 + t_2} \tag{1}$$

ここで, T_{system} はシステム内で非同期バックアップが動作していない間のシステムを外側から見たときのスループット (更新処理/sec) である. $T_{system\ backup}$ はシステム内で非同期バックアップが作動している間のシステムを外側から見たときのスループットである. よって $T_{system} > T_{system\ backup}$ である. ここでシステムに処理できる最大の更新処理を発行したとする. このとき, しばらくの間システムは更新処理に専念し, 非同期バックアップを行わない (通常処理時). なぜなら非同期バックアップを行うのはシステムアイドル時かログ格納領域が足りなくなった時だからである. システムが更新処理を受け付けてしばらく時間 ($t_{1\ sec}$) が経つとログ格納領域が足りなくなるのでシステムは非同期バックアップの動作を開始し, たまったログを処理し始める (バッ

クアップ処理時). このログを処理するまでの時間を $t_{2\ sec}$ とすると平均スループットは (1) で与えられる.

性能の見積りには以下の仮定を用いる.

- 1 回の更新処理で発生するデータ格納時のディスク I/O
  ディスクアクセスを k 回, 各アクセスごとに A_d バイトのデータだとすると合計 $k \cdot A_d$ バイトのディスクアクセスが発生
- 1 回の更新処理で発生するログ格納時のディスク I/O
  ディスクアクセスを l 回, 各アクセスごとに A_l バイトのデータだとすると合計 $l \cdot A_l$ バイトのディスクアクセスが発生
- ディスクのランダムアクセススループットを T_R (byte/sec)
- ネットワークの帯域を T_{net} (byte/sec)
- 1 回の更新処理でネットワーク上に流すデータ量 D (byte)
- 更新処理完了通知が消費するデータ量を D_{ack} (byte)
- ディスク 1 台の値段を C_{disk}

以上の仮定を元に分散ディスクログ方式の平均スループット T_{avg} とコストパフォーマンス CP を求める. T_{system} はディスク 1 台の通常処理時スループット T_{disk} にディスク最大台数 N をかけたものに等しい.

$$T_{system} = T_{disk} \cdot N \tag{2}$$

分散ディスクログ方式では 1 台のディスクにデータの更新処理とログ処理の両方が重なる. よって, 単体のディスクが 1 秒間に処理できる更新処理の回数 T_{disk} は,

$$T_{disk} = \frac{T_R}{k \cdot A_d + l \cdot A_l} \tag{3}$$

ディスクが最大に接続されているときは, ネットワークの帯域を使い切っているときである. 更新処理 1 回につきネットワークの帯域を $2 \cdot D_{byte}$ 使う. なぜなら, クライアントからの更新処理受け付けとログの書き込みにそれぞれ D_{byte} 使うからである. つまりディスク 1 台につき $2D \cdot T_{disk}$ の帯域を使うことになり, 接続できるディスクの最大台数 N は

$$N = \frac{T_{net}}{2D \cdot T_{disk}} \tag{4}$$

になる. (4) を (2) に代入すると,

$$T_{system} = \frac{T_{net}}{2 \cdot D} \tag{5}$$

となる.

非同期バックアップが動作している場合, ディスクのスループット T_R の内, β の割合を非同期バックアップに使用したとすると, 非同期バックアップ時の T_{disk} は

$$T_{disk\ backup} = (1-\beta)T_{disk}$$

となり, これに N をかけて

$$T_{system\ backup} = T_{disk\ backup} \cdot N = \frac{(1-\beta) \cdot T_{net}}{2 \cdot D}$$

となる.

T_{avg} を求めるために t_1 を求める. 1 台のディスクが 1 秒間出すログの量は $T_{disk} \cdot l \cdot A_l$ であり, 全体では N 台あるのでシステム全体が 1 秒間に出すログは

$$T_{disk} \cdot l \cdot A_l \cdot N = \frac{T_{net} \cdot l \cdot A_l}{2 \cdot D} \tag{6}$$

となる. よって T_{system} を保つ時間 t_1 は

$$t_1 = \frac{Log\ Space}{\frac{T_{net} \cdot l \cdot A_l}{2 \cdot D}}$$

である. システムが $T_{system\ backup}$ である時間を求める. 1 台のディスクで考えるとログを β の割合で読み出すので 1 秒間には $T_R \cdot \beta$ のログが減るが, $T_R \cdot (1-\beta)$ の部分では更新処理を行っているのでここの部分ではログが増える. その量は (6) を導いた議論と同じように考えると

$$(1-\beta)T_R \frac{l \cdot A_l}{k \cdot A_d + l \cdot A_l} = T_{disk}(1-\beta) \cdot l \cdot A_l$$

である. 以上より 1 台のディスクで 1 秒間にログが減る量は

$$\beta T_R - T_{disk}(1-\beta) l A_l$$

であり (この値がプラスになる β であることがログが減る条件), 全体で考えると t_2 は

$$t_2 = \frac{Log\ Space \cdot \theta}{(\beta T_R - T_{disk}(1-\beta) l A_l) \times N}$$

となる. 一時的に蓄えられたログはバックアップに更新処理を伝える前に上書きされてしまうような同一 ID の無駄な重複データを取り除く. このことで不要なアクセスを減らし性能を向上させることができる [8]. この係数を θ と置く $(0 < \theta < 1)$. T_{avg} は上記の結果を (1) に代入する事によって求まる.

コストパフォーマンスは

$$CP = \frac{T_{avg}}{C_{disk} \cdot N}$$

となる.

4.2　集中ディスクログ方式

分散ディスクログ方式ではデータとログが同じディスク内の異なる領域に格納されているために書き込みのたびシークが発生し, スループット低下の要因となっていた. これを改善するために, 集中ディスクログ方式ではデータとログを格納するディスクを分離し処理速度の向上を計った方式である. この概念を図 4 に示す.

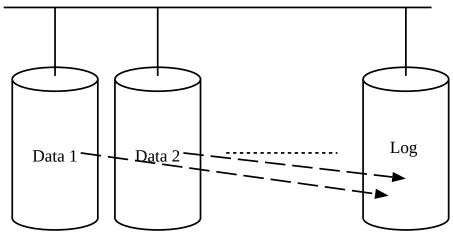

図 4: 集中ディスクログ方式

集中ディスクログ方式ではデータディスク n 台に対しログディスク 1 台が対応する. ログは集中してログディスクで格納する. ログディスクでのログの追加は分散ディスクログ方式とは異なりシーケンシャルに行える. これによりログ追加動作の高速化を計る.

4.2.1　集中ディスクログ方式性能の見積り

4.1.1 での仮定に以下の項目も付け加える.

- ディスクのシーケンシャルスループットを T_s $_{(byte/sec)}$ とする

システム内のデータディスクの最大台数を N とする. データディスクは分散ディスクログ方式とは異なりデータの更新のみに専念できるので, 1 台あたり

$$T_{disk} = \frac{T_R}{k \cdot A_d}$$

の更新処理を実行可能. システムのスループット T_{system} は分散ディスクログ方式と同じで

$$T_{system} = T_{disk} \cdot N \tag{7}$$

となる. 次にデータディスクの最大台数 N を考える. データディスク 1 台ごとに

- ログディスクで $T_{disk} \times l \cdot A_l$ の書き込みが発生
- ネットワークで $T_{disk} \times 2 \cdot D$ の帯域を消費

となる. なぜならデータディスクでは 1 秒間あたり T_{disk} の更新処理を実行できそのおのおのについてログ書き込みが発生するからである. ネットワークでも 4.1.1 での議論と同じように, それぞれの更新処理ごとに $2 \cdot D$ の帯域を消費する. ここでデータディスクの最大台数には次の 2 通りが考えられる.

- ネットワークより先にログディスクでの処理が限界
- ログディスクより先にネットワークでの処理が限界

ネットワークより先にログディスクでの処理が限界に達した場合のデータディスクの最大台数はログディスクでのシーケンシャルスループットをデータディスク 1 台が発生するログ書き込み量で割ったものに等しい.

$$N = \frac{T_s}{T_{disk} \cdot l \cdot A_l} \tag{8}$$

(8) を (7) に代入すると,

$$T_{system} = \frac{T_s}{l \cdot A_l}$$

となる. このとき, 非同期バックアップ動作時のシステムを外側から見たスループット $T_{system\ backup}$ はログディスクのスループットをバックアップ読みだし用に β の割合で使用したとすると

$$T_{system\ backup} = (1-\beta) \cdot \frac{T_s}{l \cdot A_l}$$

となる. システムが T_{system} を維持する時間 t_1 はログディスクが空の状態から領域を使い切るまでなので

$$t_1 = \frac{Log\ Space}{T_s}$$

である. $T_{system\ backup}$ である時間 t_2 を求める. ログディスクでは β の割合でログが減り, $(1-\beta)$ の割合でログが増える (通常処理) なので 1 秒間にログが減る量は $\beta T_s - (1-\beta)T_s = (2\beta-1)T_s$ である (この値がプラスであることがログが減る条件). よって t_2 は

$$t_2 = \frac{Log\ Space \cdot \theta}{(2\beta-1) \cdot T_s}$$

である. 上記の値を (1) に代入して T_{avg} が求まる.

ログディスクより先にネットワークでの処理が限界の場合には, 分散ディスクログ方式と同じように

$$N = \frac{T_{net}}{2D \cdot T_{disk}} \tag{9}$$

となり, (9) を (7) に代入すると,

$$T_{system} = \frac{T_{net}}{2 \cdot D}$$

ネットワークの帯域の内, β を非同期バックアップ用に使うとすると $T_{system\ backup}$ は

$$T_{system\ backup} = (1-\beta)\frac{T_{net}}{2\cdot D}$$

システム全体では 1 秒間に T_{system} の更新処理が発生, 更新処理 1 回ごとに lA_l のログが発生する. よって t_1 は システム全体での 1 秒間のログ量 $T_{system}lA_l = \frac{T_{net}lA_l}{2D}$ でログ容量を割った値になる

$$t_1 = \frac{Log\ Space}{\frac{T_{net}lA_l}{2D}}$$

次に t_2 を求める. ログはバックアップ処理により 1 秒間にログディスクから βT_{net} 減る. しかし, $(1-\beta)T_{net}$ の分は通常処理をしているので, ログは 1 秒間に $(1-\beta)\frac{T_{net}}{2D}lA_l$ の分増える. よって t_2 は

$$t_2 = \frac{Log\ Space\theta}{\beta T_{net} - (1-\beta)\frac{T_{net}}{2D}lA_l}$$

コストパフォーマンス CP はデータディスク N 台, ログディスク 1 台であることを考慮すると

$$CP = \frac{T_{avg}}{C_{disk}\cdot (N+1)}$$

になる.

ネットワークとログディスクのどちらかがボトルネックになるかは (9) と (8) を比較するこでわかる. ログディスクがボトルネックになる条件は, (9) > (8), つまり

$$\frac{T_{net}}{2\cdot D\cdot T_{disk}} > \frac{T_s}{T_{disk}\cdot l\cdot A_l}$$

となるので

$$\frac{T_{net}}{2\cdot D} > \frac{T_s}{l\cdot A_l}$$

のときログディスクがボトルネックになり, 不等号が逆のときにネットワークがボトルネックになる.

4.3　複数集中ディスクログ方式

集中ディスクログ方式においてログディスクがボトルネックになっているときにはログディスクの台数を増やすことによってシステムのスループット向上を計ることができる. この構成はデータディスク n 台に対しログディスク m 台の構成になる.

4.3.1　複数集中ディスクログ方式性能の見積り

ログディスクの最大台数はネットワークの帯域がボトルネックになっている時と考えられる. よって, $T_{system}, T_{system\ backup}, t_1, t_2$ は集中ディスクログ方式でのネットワークがボトルネックになっている場合と同じ議論が成立する.

次に, ディスクの最大台数を考える. データディスク N 台 (N は (8) より) とログディスク 1 台のセットが最大で N’ セットあったとすると 1 台のデータディスクは $T_{disk}\cdot 2\cdot D$ の帯域を消費し、それが 1 セットで N 台あるので, N’ は次の式で求められる.

$$T_{net} = 2\cdot D\cdot T_{disk}\cdot N\cdot N'$$

つまり,

$$N' = \frac{T_{net}}{N\cdot T_{disk}\cdot 2\cdot D}$$

となる. CP は上記より,

$$CP = \frac{T_{system}}{C_{disk}\cdot (N+1)\times N'}$$

となる.

4.4　分散メモリログ方式

集中ディスクログ方式ではログの格納にディスクを用いていたがこれをより高い性能を得られるメモリに変えることによりシステム全体としてのスループット向上が計れる. これはデータディスク n 台に対しログメモリ m 台の構成である. これを分散メモリログ方式と名付けた. この概念を図 5 に示す.

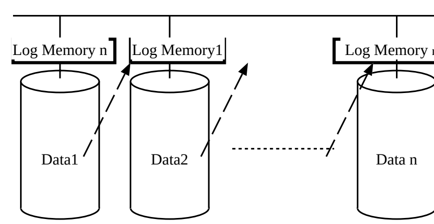

図 5: 分散メモリログ方式

利点としてはメモリを用いることによるスループットの向上, およぎログが 1 箇所に集中していないのでボトルネックが発生しずらい等があるが欠点としてメモリを多く用いるので価格コストが高くなるという点がある. 単一故障を前提とすれば, ログに利用するメモリは不揮発性ではなく揮発性メモリで良い. なぜならばログメモリに障害が発生しその内容が失われたとしてもプライマリデータはデータディスク側に残っているのでデータ損失が生じないからである [9].

4.4.1 分散メモリログ方式性能の見積り

ディスクの最大台数を求める. ネットワークがボトルネックになり, ログメモリがボトルネックになことはない. なぜならディスクよりもメモリの方がスループットが大きく, また更新処理のデータ (kA_d) よりもログデータ (lA_l) の方がデータ量が少ないからである. ネットワークがボトルネックになると考えれば分散ディスクログ方式と同じ議論であり,

$$T_{disk} = \frac{T_R}{k \cdot A_d}$$

$$N = \frac{T_{net}}{2 \cdot D \cdot T_{disk}}$$

$$T_{system} = \frac{T_{net}}{2 \cdot D}$$

$$T_{system\ backup} = (1-\beta) \cdot \frac{T_{net}}{2 \cdot D}$$

$$t_1 = \frac{Log\,Space}{\frac{T_{net} l A_l}{2D}}$$

$$t_2 = \frac{Log\,Space \cdot \theta}{\beta \cdot T_{net} - (1-\beta) \cdot \frac{T_{net}}{2D} l A_l}$$

となる. ここでの $Log\,Space$ はログメモリの容量になる.

コストパフォーマンスは

$$CP = \frac{T_{avg}}{(C_{disk} + C_{Mem}(S_{Mem})) \times N}$$

となる. ここで S_{Mem} はメモリ容量であり, C_{Mem} はメモリの価格コストを出す関数である.

5 コストパフォーマンスの比較

4 であげたコストパフォーマンスを比較する. 我々は現在, 自律ディスクの機能を PC 上にシュミレーションにより実装しその機能を検証している. シュミレーターで使用している PC のハードディスクは Seagate 社の STA320011A(20GB) である. 転送ブロックサイズ A_k を 4kb としたときの STA320011A の転送速度を iometer [10] で計測したところ $T_s = 12_{\ MB/sec}, T_R = 0.5_{\ MB/sec}$ であった. また, k と l はインデックスツリーの高さに依存する [11]. ここではツリーの高さを k=5 とした. l はログページを書き込むので 1 に, D は 1 度の更新処理で 1 ページのデータを送るとして 4*kbyte* とし, C_{disk} を 10000*yen*, メモリの値段を $1G_{\ byte}$ で $40000_{\ yen}$ とした. $Log\,Space = 1Gbyte$ とし, $\beta = 0.6$ とする. ネットワークを 100BASE を使用したときのグラフが 6 であり, 1Gbps ether を使用したときのグラフが 7 である. 100BASE, 1Gbps ether ともに分散ディスクロ

	分散ディスク	集中ディスク	分散メモリ
性能 (更新処理/sec)	1562	1562	1562
コスト (円)	74×10^4	64×10^4	67×10^4
性能/コスト	0.0021	0.0024	0.0023
台数	74	64	63

表 1: ネットワークが 100BASE の時

	分散ディスク	複数集中ディスク	分散メモリ
性能 (更新処理/sec)	15620	15620	15620
コスト (円)	740×10^4	629×10^4	629×10^4
性能/コスト	0.0021	0.0024	0.0024
台数	740	629	625

表 2: ネットワークが 1Gbps ether の時

グ方式はコストパフォーマンスが悪い. 分散ディスクログ方式ではディスクのスループットを有効に利用できていないためであると考えられる. 集中ディスクログ方式は 100BASE はネットワークがボトルネックになり, 1Gbps ether ではログディスクがボトルネックになった. 集中ディスクログ方式で, 100BASE ではログディスクの持つスループットを完全に使い切れない. 1Gbps ether では逆にログディスクがボトルネックになる. よって, 1Gbps ether では比較に複数集中ディスクログ方式を用いた.

ネットワークが 100BASE の時の, ネットワークがボトルネックとなっているときの各方式と性能, コスト, 性能コスト比, ディスク台数の関係を表 1 に示す。同様にネットワークが 1Gbps ether の表を表 2 に示す.

本稿での議論はシステム構成を考える際にネットワークを使い切った場合の最大スループットを検証した. しかし, ネットワークの帯域を使い切る構成を考えると表 2 のようにディスク台数が数百台になってしまい現実的な構成ではない. 他のアプローチとしては, データ量を基準として, あるデータ量に対しての最適なシステム構成を考える, といったアプローチも存在する.

6 おわりに

本稿では自律ディスクにおける非同期バックアップを実現するシステムを検証した. 分散ディスクログ, 集中ディスクログ方式, 複数集中ディスクログ方式, 分散メモリログ方式の各方式の構成方法を評価した. 評価はスループット, コストパフォーマン

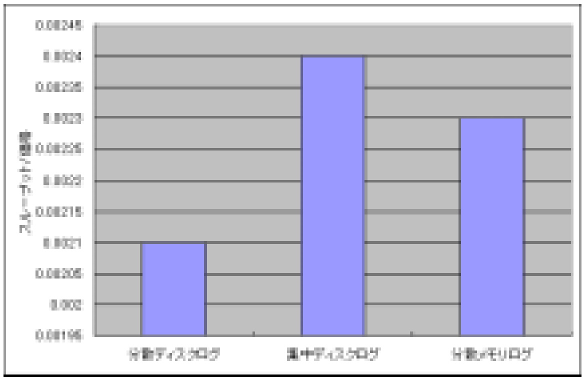

図 6: コストパフォーマンスのグラフ (100BASE)

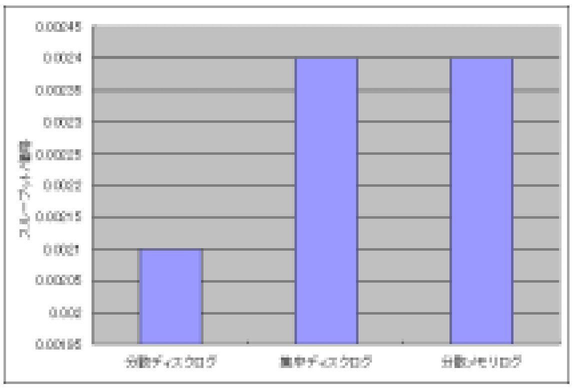

図 7: コストパフォーマンスのグラフ (1Gbps)

スの観点から比較した. ディスクのブロックサイズ (A_d, A_l) を $4_{k\ byte}$ にするとディスクのスループットが落ち, ディスクの台数が数百台のオーダーになってしまう. ブロックサイズを大きくすればディスクのスループットが増大しディスク台数が数十台のオーダーまで落ちると考えられる. 今後はブロックサイズを大きくした環境でのシステム構成で計算機実験によって評価を行い, またネットワークを基準にしたシステム構成ではなくデータ量を基準にしたシステムの構成を検証していく.

謝辞

本研究の一部は, 文部科学省科学研究費補助金基盤研究 (12680333, 13224036, 14019035) および情報ストレージ研究推進機構 (SRC) の助成により行われた.

参考文献

[1] Kimberly Keeton, David A. Patterson, and Joseph M. Hellerstein. A Case for Intelligent Disks (IDISKs). *SIGMOD Record*, 27(3):42–52, Sep. 1998.

[2] Anurag Acharya, Mustafa Uysal, and Joel Saltz. Active Disks: Programming Model, Algorithms and Evaluation. In *Proc. of the 8th ASPLOS Conf.*, Oct. 1998.

[3] Erik Riedel, Garth Gibson, and Christos Faloutsos. Active Storage for Large-Scale Data Mining and Multimedia Application. In *Proc. of the 24th VLDB Conf.*, pages 62–73, 1998.

[4] Haruo Yokota. Autonomous Disks for Advanced Database Applications. In *Proc. of International Symposium on Database Applications in Non-Traditional Environments (DANTE'99)*, pages 441–448, Nov. 1999.

[5] 阿部 亮太. ルール処理機能を有する高機能化ディスクの構成に関する研究. Master's thesis, 東京工業大学, 2001.

[6] 伊藤 大輔 , 横田 治夫. 自律ディスクにおけるディスク故障時/追加時のクラスタ再構築法. In **第 *12* 回データ工学ワークショップ論文集**, *DEWS2001 2B–4*. 電子情報通信学会データ工学研究専門委員会, 2001.

[7] 安部洋平 , 横田治夫. Java による耐故障ネットワークディスクのルール処理の実装. In **第 *11* 回データ工学ワークショップ論文集**, *DEWS2000 3B–2*. 電子情報通信学会データ工学研究専門委員会, 2000.

[8] 後藤 正徳 , 横田 治夫. 耐故障バッファリングディスクシステムの性能評価. In **電気情報通信学会論文誌** *D-I Vol. J84-D-I No.6*, pages 819–829, 2001.

[9] 宮崎 純 , 横田 治夫. 高信頼性 fat-btree 構成への neighbor-wal プロトコルの適用. In **第 *13* 回データ工学ワークショップ論文集**, *DEWS2002 C1–2*. 電子情報通信学会データ工学研究専門委員会, 2002.

[10] http://www.intel.co.jp/jp/developer/design/servers/devtools/iometer.

[11] 阿部 亮太 , 横田 治夫. 自律ディスクと従来のディスクのコスト比較. In **第 *11* 回データ工学ワークショップ論文集**, *DEWS2000 3B–1*. 電子情報通信学会データ工学研究専門委員会, 2000.