非同期バックアップにおけるログ制御手法と性能

安部　洋平† 　横田　治夫††

† 東京工業大学大学院 情報理工学研究科計算工学専攻
†† 東京工業大学学術国際情報センター
E-mail: †yabe@de.cs.titech.ac.jp, ††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. 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にデータを配置し, ログを用いた非同期バックアップによりバックアップを更新する. 非同期バックアップの性能はホストからのアクセス頻度, ログ制御手法に影響される. 本研究ではログ制御手法を実現する構成方法について提示し, 模擬自律ディスクによって評価を行う.

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

The Effect of Log Control Method on the Performance of Asynchronous Backup

Youhei ABE† and Yokota HARUO††

† Department of Computer Science,Information Science and Engineering in
Tokyo Institute of Technology
†† Global Scientific Information and Computing Center, Tokyo Institute of Technology
E-mail: †yabe@de.cs.titech.ac.jp, ††yokota@cs.titech.ac.jp

Abstract　We proposed the autonomous disks as an approach to efficient management for large scale storage systems. The autonomous disks configure a storage cluster with no centralized controller, and store data into duplicate portion, primary and backup, to realize the high availability. To keep consistency between the primary and backup, log based asynchronous catch-up mechanism is adopted. The performance of the asynchronous catch-up is affected by log control methods and access frequency pattern from clients. In this paper, we evaluate the relationship between the performance of catch-up and environment using pseudo autonomous disks constructed from PCs and LAN.

Key words　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. この問題に対する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の問題のうち, 最も重要なものはデータ信頼性の問題である. データ信頼性を上げる方法にはいろいろあるが, その一つとしてデータ二重化がある. データ二重化では同一のデータを異なるストレージの二箇所, プライマリとバックアップに配置する方法である.

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かった. しかしシステムの規模の増大に伴い, ストレージが巨大になると, そのバックアップをサーバー側で行なうことはデータ増大以上の管理コストの増大を招いてしまう.

管理コスト増大に対応するため, 我々はネットワークストレージをストレージ側で自律的に管理するアーキテクチャとして, 自律ディスク [1] 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 PC を用いた模擬自律ディスクにより検証してきた [2]〜[5].

その中でデータ二重化を実現する方法である非同期バックアップについて, 非同期バックアップを構成するシステムの構成 [5] について検証を行った。これまでの議論ではプライマリ, バックアップ, ログデータをクラスタ内のディスク, メモリ領域

に配置するシステム構成方式について議論した.

しかし非同期バックアップの性能はシステム構成方式に加えてログ制御手法にも影響を受ける. 本研究ではログ制御手法を実現する構成方法を提示し性能を調査する.

まず 2. 章では自律ディスクと非同期バックアップの概念について説明する. 3. 章ではログ制御手法について説明し 4. 章ではログ制御手法を評価する方法について議論する. 5. 章と 6. 章では実験を行う. 7. 章では実験の考察を行い, 8. 章でまとめを行う.

2. 自律ディスクと非同期バックアップ

現在のディスク装置は高速なコントロールチップと大容量のキャッシュメモリを搭載し, 高度な演算能力を保有している. そのような能力を有効に利用する研究は主に従来アプリケーションが行っていた処理をディスクに行わせるとうアプローチであった ([6]〜[8]) . これに対し, 我々は高機能ディスクが持つ演算能力をディスク自体の管理に利用するアプローチとして自律ディスク [1] を提案している.

自律ディスクはネットワーク上でストレージクラスタを形成し各ノードが自律的, 相互に通信することにより処理を行うストレージアーキテクチャである.

自律ディスクではデータ二重化によりデータの信頼性を保つ. ストレージクラスタ内の異なるデバイスに同一のデータをプライマリ, バックアップとして保存する. データ二重化を行う方法には同期による方法と非同期による方法の二種類ある. 自律ディスクでは各ノードが自律的に動作するという特性を利用し非同期による方法, 非同期バックアップによってデータ二重化を実現する.

非同期バックアップを行う際のデータの流れを図 1 示す.

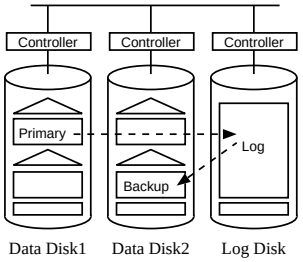

図 1: 非同期バックアップの例

自律ディスクではプライマリデータの書き込みは WAL(Write-Ahead-Log) プロトコルに従う. WAL プロトコルではプライマリデータの更新を行う前にログを書き込まなければならない. そのために次のような流れでデータ更新を行う.

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 はプライマリデータの書き込みを行う前にログディスクにログを送る. ログディスクはログの書き込みを行った後, プライマリデータを保持しているディスクに対してログ書き込み完了通知を返す. プライマリデータを保持しているディスクでは, ログ書き込み完了通知を受け取った後プライマリデータの更新を行う. その後クライアントに対し完了通知を返す. クライアントに完了通知を返すタイミングとは非同期に, ログディスクはログをバックアップに送りプライマリデータとバックアップデータの一慣性を保つ.

非同期バックアップでのログは一般のデータベースのログとは異なる. データベースのログは取得し続け, ディスク領域に保存される. 非同期バックアップでのログはキャッチアップを行った後ログ領域から削除することが可能である (一時的な保存). 単一故障を前提とすれば, ログデータはディスクだけではなく揮発メモリにも保存が可能となる [9].

2.1 非同期バックアップの性能を決めるパラメタ

非同期バックアップの性能はプライマリ, バックアップ, ログデータをクラスタ内でどのように配置をするかというシステム構成方式に加え, キャッチアップを行う際にログディスクに蓄積されたログをどのようにバックアップに送るかというログ制御手法によっても影響される.

3. ログ制御手法

ログ制御手法とはログディスクが保持しているログの制御方法のことである. ログ制御手法によってログディスクが保持しているログをいつどのようにキャッチアップするのかが決定される. Anujan Varma et al. [10] によって RAID と不揮発性キャッシュを利用した際のログ制御手法が紹介されている. しかし自律ディスクではログの格納に揮発性メモリを利用できる点やシステム構成方式によって性能に影響が出る点が異なる.

3.1 直接更新法

ログがログディスクに到着するとすぐにバックアップに更新を行う. ログはログ格納領域に一時的に保持されるがすぐにバックアップに送られる. ログの蓄積は行わない. よってこのアルゴリズムを使用した場合はログ格納容量の大きさは, 非同期バックアップの性能に関与しないと考えられる.

3.2 High/Low watermark 法

ログを蓄積していき, ログ量がある上限 (High mark) を越えたらバックアップに更新を開始する. ログ量が下限 (Low mark) に達したらキャッチアップを終了し再びログが蓄積されるのを待つ. ログ領域として確保されている容量より High mark 点は低く設定され, High mark 点をログ量が越えてもログ容量との間が緩衝になりすぐにログがあふれることはない.

ログを一時的に保持することにより後から来た ID のみが有効であることから, 同一 ID が重複した場合のログ圧縮ができ, ログ格納容量が大きい程, 性能向上が考えられる. 以降 High/Low watermark 法を H/L 法と記す.

3.3 High/Low watermark + 負荷監視法

High/Low watermark 法で High mark と Low mark の間にログ量がある場合, ログ書き込み負荷を監視し, ログ書き込み頻度が設定されている閾値より小さくなったときにキャッチアップを行う方法. ログ書き込み頻度に緩急がある場合, ログディスクの負荷が低いときを利用し有効にキャッチアップを行える. ただし, ログ書き込み頻度が閾値より小さくならない場合には High/Low watermark 法と同じになる. 以降 High/Low watermark + 負荷関し法を H/+L 法と記す.

4. 評 価 方 法

ログ制御手法を評価するためにはクライアントからのアクセ

スパターンを設定する必要がある. しかし, 一般的なアクセス環境は不定期な間隔で時間局所性があり環境によってそのアクセスパターンが変化してくるためそれをシミュレートするのは難しい. そこで本研究では, アクセスパターンを人工的に作成し, そのデータに基づくシミュレーションによって非同期バックアップの評価を行いたい.

4.1 アクセスパターン

本研究では人工的なアクセスパターンとして以下に挙げる三つのパターンを用いた.

4.1.1 アクセスパターン:一定

クライアントからのアクセスパターンが常に一定の頻度で保たれている場合. 大きなファイルの読み書きを行うようなアプリケーションはこのようなアクセスパターンに対応する.

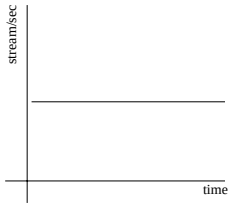

図 2: アクセスパターン:一定

4.1.2 アクセスパターン:波形

クライアントからのアクセスが整流正弦波状で起こる場合.

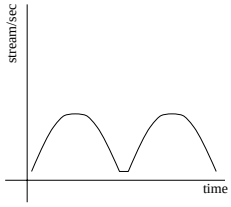

図 3: アクセスパターン:波形

4.1.3 アクセスパターン:一定 + バースト

クライアントから一定のアクセスがしばらく続いたあと, バースト的に大きい頻度でアクセスが送られる. このパターンが周期的に繰り返される. 一般の時間局所性があるアクセスパターンに対応している.

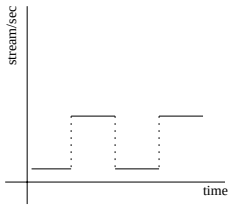

図 4: アクセスパターン:一定+バースト

また、実際に評価実験を行う際には更新を行う際の streamID を zipf 分布に従って生成することにより ID の偏りを表現する.

4.2 同期方式との比較

データ二重化を実現する方法には非同期方式だけではなく同期方式でも実現できる. 非同期バックアップと異なる点はログデータが存在せず, プライマリデータを保存しているディスクは直接にバックアップを書き込むということである. ログに書き込むステップが省略されるがバックアップよりログの方がインデックスの更新を行わず append-only なのでシーク, 同期待ちが減り書き込みが高速にできるや配置場所をメモリにした場合ディスクに配置する場合より回転待ち時間がない, といった利点がある.

表 1: 実験環境の緒元

CPU	Intel Pentium3 933MHz
Chipset	Intel i815 (ATA100)
Memory	PC133 (CL=3) 256MB
HDD	Seagate ST320011A(7200rpm, Barracuda ATA IV)
Network	GbE 1000Mbps
OS	Linux 2.2.17, glibc–2.1.3
Java	IBM JRE1.3.0
	(IBM build cx130–20010626 (JIT enabled: jitc))
Hub	1000Base Switch

5. 評価実験

アクセスパターンを設定しそれをログ制御手法に適用することにより, 各ログ制御手法の性能を調べる.

実験は PC 上での模擬自律ディスクの実装により行なう. 表 1 に実験システムの緒元を示す.

実験データは以下のデータを利用した.

- 1 ストリームサイズを 4kB
- ストリーム総数 200MB
- ストリーム ID は Zipf 分布に従う. また, streamID に偏りを持たせるために $y = \frac{c}{x^{\theta}}$, において θ=0.0(偏りなし), 1.0(偏りあり) の場合を設定.
- ディスク台数 6
- アクセスパターン 1($100_{stream/sec}$)
- アクセスパターン 2(低 $10_{stream/sec}$, 高 $100_{stream/sec}$, 周期 200_{sec})
- アクセスパターン 3(最大 $100_{stream/sec}$, 周期 200_{sec})

ログ制御手法は以下の条件を利用した.

- H/L 法の Low mark には 0MB, High mark には 30MB, ログ容量は 50MB.
- H/L+法の閾値には $50_{stream/sec}$
- 分散ディスクログ方式を用いてログ格納にはディスク領域を使用.

6. 実験結果

6.1 アクセスパターン:一定

6.1.1 ID 偏りなし

以降のグラフの横軸はストリーム数に, 縦軸はレスポンスタイムを示し, レスポンスタイムの分布を示している.

パターン 1 では低頻度の部分がないために H/L 法と H/L+法は同じ挙動を示す.

同期 (図 5) と直接更新法 (図 6) とを比べると直接更新法の方がレスポンスが大きい. これは直接更新法ではログを取得しているが同期ではログを取得していない分レスポンスが小さくなるためである. 図 7 の H/L 法ではログを蓄積している間はレス

ポンスが低く, High mark 点を越えてキャッチアップをしている間はレスポンスタイムが高くなっている. これが繰り返され, グラフの形が周期的になっている.

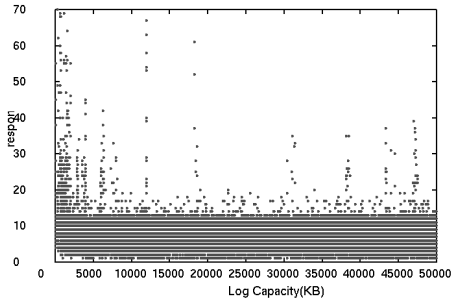

図 5: 同期 (パターン:一定 偏りなし response time プロット (平均 9.01ms))

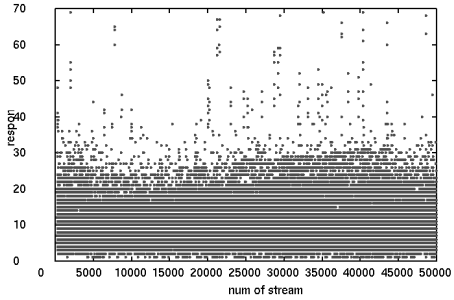

図 6: 直接更新法 (パターン:一定 偏りなし ログ格納容量 30MB response time プロット (平均 11.36ms))

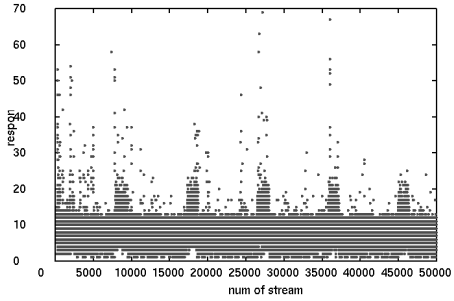

図 7: H/L 法 (パターン:一定 偏りなし ログ格納容量 30MB response time プロット (平均 8.15ms))

6.1.2 ログ容量を増やした場合

アクセスパターンが一定, 偏りなしの場合でログ格納容量と平均レスポンスタイムとの関係を示したのが図 8 である. 同期と直接更新法のレスポンスはログ容量に依存することはないので一定の値を示している. H/L 法はログ容量を増やす程, ログ圧縮の効果によりレスポンスを小さくすることができ, ログ容量を十分に大きく取れば同期よりレスポンスタイムを小さくできる. また, H/L 法でログ容量が大きくなるとログ圧縮の効果が小さくなりグラフの傾きが小さくなっている.

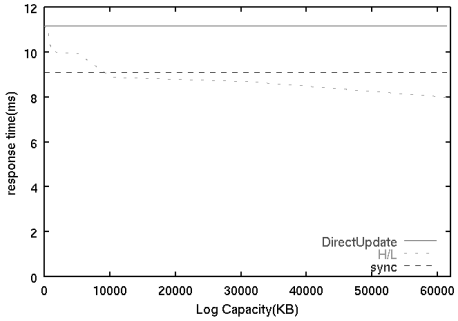

図 8: パターン:一定 平均レスポンスタイムを比較

6.1.3 ID 偏りあり

ID に偏りがある場合には H/L 法 (図 10) は偏りなしの場合と比較してよりログ圧縮が効くためにレスポンスが小さくなる. 直接更新法の場合 (図 9) には偏りありでもログ蓄積をしていないために挙動に変化はない.

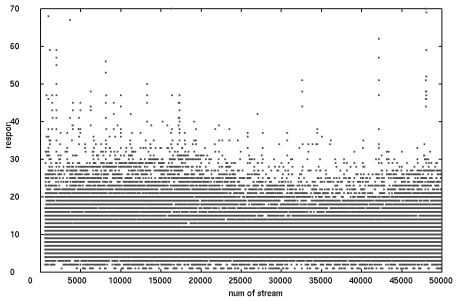

図 9: 直接更新法 (パターン:一定 偏りあり ログ格納容量 30MB response time プロット (平均 11.16ms))

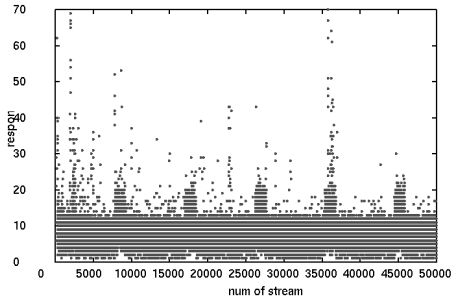

図 10: H/L 法 (パターン:一定 偏りあり ログ格納容量 30MB response time プロット (平均 8.02ms))

アクセスパターンが一定の場合は H/L 法と H/L+法は同じ挙動を示す. ログ容量が大きい程ログ圧縮によりレスポンスタイムが小さくなる.

6.2 アクセスパターン:波形

6.2.1 ID 偏りなし

アクセスパターンが波形の場合も, H/L 法 (図 11) はアクセスパターンが一定の場合と同じようにログ蓄積期間とキャッチアップが周期的にならんでいる. アクセスパターンが一定の場

合と異なるのはアクセスパターンが波形の場合, 低頻度の部分があるため H/L+法 (図 12) を用いると, 頻度が低いときにキャッチアップを行えるので H/L 法と比べてレスポンスが低くなり, H/L 法よりレスポンスが小さくなる点である (図 13).

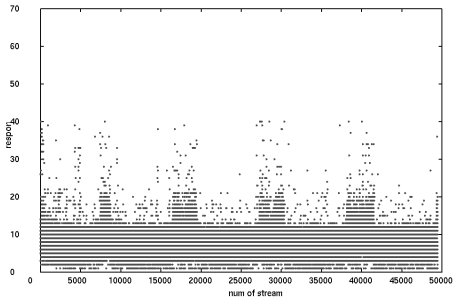

図 11: H/L 法 (パターン:波形 偏りなし ログ格納容量 30MB response time プロット (平均 7.99ms))

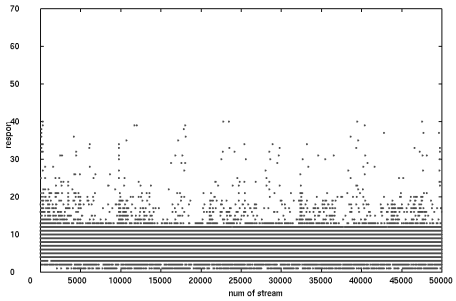

図 12: H/L+法 (パターン:波形 偏りなし ログ格納容量 30MB response time プロット (平均 7.58ms))

6.2.2 容量を増やした場合

アクセスパターンが波形において容量とレスポンスタイムを比較したのが図 13 である. アクセスパターンが一定の場合と異なるのは H/L 法と H/L+法はログ容量を大きく取れば, ログ圧縮によりレスポンスタイムを小さくできる点である. また, H/L+法は H/L 法と比較して低頻度時キャッチアップのために H/L 法よりレスポンスタイムが低くなる. ログ容量が大きくなると H/L 法と H/L+法の傾きが小さくなるのは一定の場合と同じである.

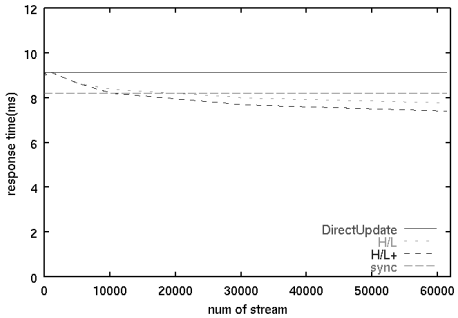

図 13: パターン:波形 平均レスポンスタイム比較

6.2.3 ID 偏りあり

ID に偏りがある場合, H/L 法 (図 14), H/L+法 (図 15) は偏りがない場合と比較しログ圧縮がより多く効くようになりレスポンスが小さくなる.

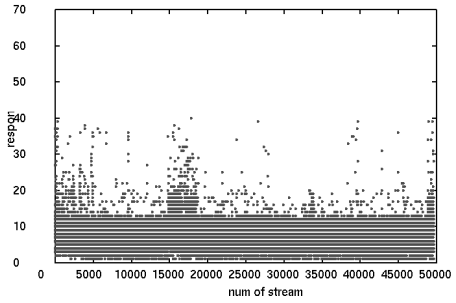

図 14: H/L 法 (パターン:波形 偏りあり ログ格納容量 30MB response time プロット (平均 7.67ms))

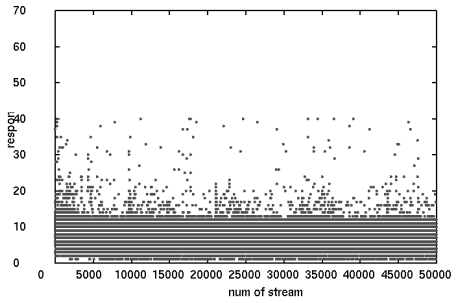

図 15: H/L 法 (パターン 3 偏りあり ログ格納容量 30MB response time(平均 7.67ms))

パターンが波形の場合には H/L+法の方が H/L 法に比べて低頻度時にキャッチアップできるのでレスポンスを小さくできる.

6.3 アクセスパターン:バースト

6.3.1 ID 偏りなし

直接更新法 (図 17) ではアクセスパターンがバーストの場合の周期的な形がレスポンスに現われている. レスポンスはアクセスパターンが一定のときと同じように他と比較してもっとも大きい. H/L 法 (図 18) はアクセスパターンが一定の場合と同じようにログ蓄積期間とキャッチアップが周期的にならんでいる. H/L+法 (図 19) では頻度が低いときにキャッチアップを行うために H/L 法と比べてグラフが低くおさまりレスポンスも小さくなっている.

6.3.2 ログ容量を増やした場合

アクセスパターンがバーストの場合の容量とレスポンスの関係を示したのが図 20 である. バーストの場合, 波形と異なりアクセス頻度の変化が大きいので H/L 法と H/L+法のレスポンスが波形の場合と比較して開きが大きい. ログ容量が大きくなると H/L 法と H/L+法の傾きが小さくなるのは一定の場合と同じである.

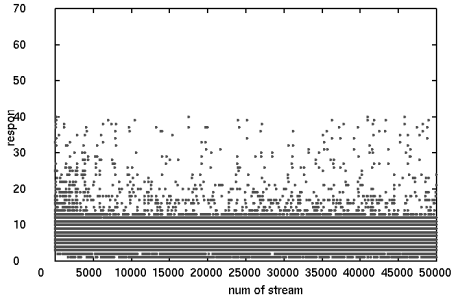

図 16: 同期 (パターン:バースト 偏りなし response time プロット (平均 8.3ms))

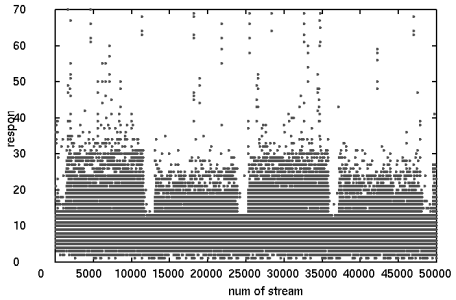

図 17: 直接更新法 (パターン:バースト 偏りなし ログ格納容量 30MB response time プロット (平均 10.90ms))

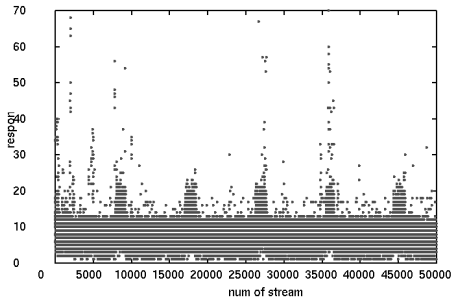

図 18: H/L 法 (パターン:バースト 偏りなし ログ格納容量 30MB:response time プロット (平均 8.05ms))

6.4 ディスクログをメモリログにした場合

同期, 非同期方式共にプライマリ, バックアップデータはディスク領域に保存するが非同期方式の場合ログデータはディスク領域だけではなくメモリ領域にも保存することが可能である [5]. ログデータをメモリ領域に保存した場合のグラフ分布が (図 21) である. ディスクログ (図 7) と比較してログをメモリに保存した分, 高速になりレスポンスが小さくなっている. これは非同期方式が同期方式と比較して持つ利点の一つである.

7. 考　　察

アクセスパターン一定の場合, ログ容量が小さい内は同期方式のほうが非同期方式よりレスポンスが小さくなった. これは同期と比較してログ取得がオーバーヘッドになるためである.

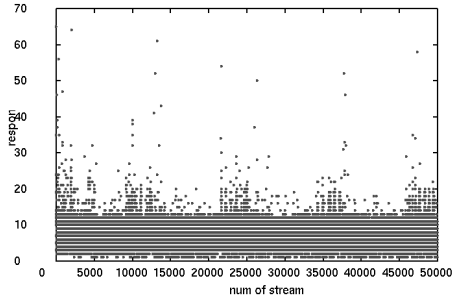

図 19: H/L+法 (パターン:バースト 偏りなし ログ格納容量 30MB:response time プロット (平均 7.47ms))

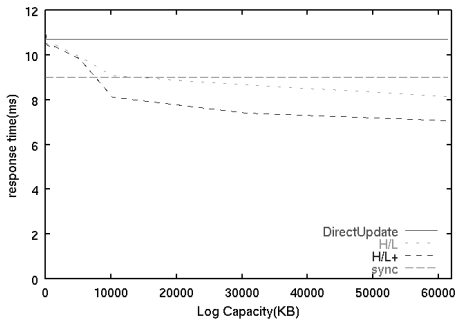

図 20: パターン:バースト 平均レスポンスタイム比較

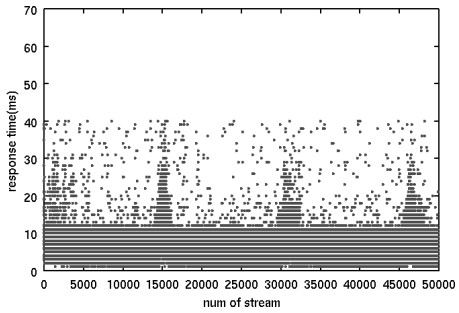

図 21: H/L 法 (パターン:一定 偏りなし ログ格納容量 response time(平均 7.23ms))

ただし, H/L 法でログ容量を大きくとりログ圧縮を十分にかければログ圧縮の効果によりレスポンスを小さくすることができる.

また, 波形とバーストのアクセスパターンでは, H/L+法が最もレスポンスが小さくなった. これはアクセスパターンに低頻度の部分があるために, ログ圧縮と低頻度時キャッチアップが有効に働くためである. ただしアクセスパターンが低頻度になるまで, ログを蓄積しておける分のログ容量がなければ性能は低下する.

非同期方式ではログ領域をディスクだけではなくメモリに取る構成も可能である. ログをメモリに保存することによりディスクに保存する場合と比較してレスポンスが小さくなることを示した. これによりメモリを十分に確保することができれば同期方式と比較してより大きな性能向上を実現することができる.

8. 終　り　に

分散ストレージシステムに置ける信頼性向上の 1 つのアプ

ローチとして, 我々が提案してる自律ディスクの非同期バックアップ機構について検証を行った.

非同期バックアップではログ制御手法により, ログからバックアップへデータを送るタイミングが決定される. 本研究ではログ制御手法の 3 方式について性能比較を行った. そのために, アクセスパターンを仮定し実験を行った.

その結果, アクセスパターンに低頻度の期間がある場合には H/L+法が低頻度時に効率的にキャッチアップできるために同期方式よりレスポンスが小さくなることを示した.

低頻度の期間がない場合, ログ容量が小さいときには同期方式の方がレスポンス低だが, 非同期方式:H/L 法でもログ容量を十分に大きくとることにより, ログ圧縮の効果のために同期よりレスポンスを小さくできることを示した. 実験の範囲内では約 10MB のログ容量で同期方式より非同期方式の方がレスポンスが下回り, ログ容量約 60MB で同期方式より 1〜1.5ms 低くなった.

今後の課題としては, 今回設定したアクセスパターン以外のパターンを適用した場合の性能調査やログ容量からログがオーバーフローした場合の対処方法などが考えられる.

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.

文　　献

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 安部洋平 , 宮崎純 , 横田治夫. 非同期バックアップにおけるログ格納の高速化の影響. In **信学技法** *Technical Report of IEICE DE2002-39*, pages 67–72. 電子情報通信学会, 2002.

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

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

[8] 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.

[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] Anujan Varma , Quinn Jacobson. Destage algorithms for disk arrays with nonvolatile caches. In *IEEE transactions on computers vol47, No2, Feburuary*, pages 228–235. IEEE, 1999.