THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

TECHNICAL REPORT OF IEICE.

高信頼Fat-Btree構成へのneighbor-WALプロトコルの適用

論文番号: C1–2

宮崎　　純†　　横田　治夫††

† 北陸先端科学技術大学院大学 情報科学研究科
〒 923–1292 石川県能美郡辰口町旭台 1–1
†† 東京工業大学 学術国際情報センター
〒 152–8552 東京都目黒区大岡山 2–12–1

E-mail: †miyazaki@jaist.ac.jp, ††yokota@cs.titech.ac.jp

あらまし　我々が提案している並列ディレクトリ構造Fat-Btreeを高信頼化するためには，データをスタガード配置し，グローバル論理ログと各プロセッシングエレメントでの物理論理ログを組み合わせたn-GLL+PL+stg方式が有効であることを示してきた．本稿では，高信頼性を保ちつつneighbor-WALプロトコルを適用することによりシステムのトランザクション処理性能を向上させる方法について考察する．

キーワード　B-tree, 並列ディレクトリ, Fat-Btree, 高信頼システム, ログ，WALプロトコル

On Applying Neighbor-WAL Protocol to Dependable Fat-Btrees

Paper#: C1–2

Jun MIYAZAKI† and Haruo YOKOTA††

† School of Information Science, Japan Advanced Institute of Science and Technology,
1–1 Asahidai, Tatsunokuchi, Ishikawa 923–1292, Japan
†† Global Scientific Information and Computing Center, Tokyo Institute of Technology,
2–12–1 Ookayama, Meguro, Tokyo 152–8550, Japan

E-mail: †miyazaki@jaist.ac.jp, ††yokota@cs.titech.ac.jp

Abstract　We have proposed n-GLL+PL+stg method in order to make the parallel directory structure, Fat-Btree, reliable. The n-GLL+PL+stg combines a staggered data allocation, global logging, and physiological logging. In this paper, we discuss a case in which neighbor-WAL protocol is applied to the Fat-Btree so that the performance of the transaction processing can be improved without reducing its high reliability.

Key words　B-tree, parallel directory, Fat-Btree, dependable system, log, WAL protocol

1. まえがき

データベース用無共有並列計算機では，検索および更新処理は参照されるデータが配置されている各PE(Processing Element) 上で並列に実行されることが望ましい．各 PE 間で負荷を均等にすることは処理性能向上につながるが，負荷を均等化するためのデータ分配方式が問題となる [1], [3].

従来のデータ分配方式にはハッシュ分配方式や値域分配方式 [2] などがあるが，ハッシュ分配方式では領域指定された問い合わせや，連続したアクセスの I/O 回数を削減するクラスタ化に対応できないという欠点がある．一方，値域分配方式では分割の基準値が静的に決定されるため，データの更新によってデータ配置に偏りが生じた時に均一化するコストが非常に大きくなる欠点がある．

そこで我々は高性能な並列ディレクトリ構造として Fat-Btree を提案している [8], [10]〜[12]．並列環境では単一 PE に比べてシステムの信頼性が低下するため，様々な障害から一貫した状態を保護するとともに，アベイラビリティを高めることも不可欠である．我々は Fat-Btree の信頼性向上を達成するために，いくつかの構成方法を提案している．しかしながら，信頼性を向上させるために性能を犠牲にすることはできない．本論文では，Fat-Btree の高信頼構成に neighbor Write-Ahead Log プロトコル (nWAL) を適用することにより，信頼性を保ちつつ更新処理性能を改善する方法ついて議論する．

2. Fat-Btree

Fat-Btree [8] は，B-tree 全体をページ単位で各 PE に分配する並列 B-tree の一種である．Fat-Btree は，B-tree の葉ページ (データページ) を各 PE に均等に分配する．ディレクトリ部分である B-tree の葉ページ以外は，各 PE に配置されている葉ページへのアクセスパスを含むインデクスページのみを再帰的に配置する．これにより，各 PE のディスクに格納されるのは B-tree のルートから均等に分配された葉ページまでの部分木となる (図 1)．これらのデータを保持する PE を**データ PE** と呼ぶ．

Fat-Btree の名前は，並列計算機向け相互結合網の一つである，Fat-Tree [6] と B-tree とを融合したことに因んでいる．Fat-Tree はルートに近付くほどパスが増加する木構造であり，トラフィックが集中するルート付近のバンド幅を大きくすることが目的となっている．

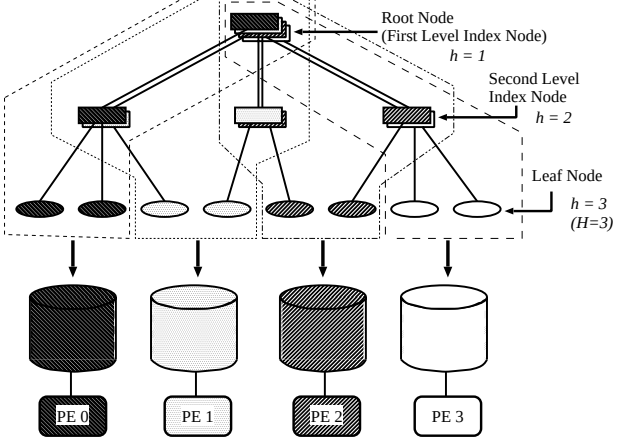

図 1　Fat-Btree

Fat-Btree では，多数の葉ページへのパスを含む上位のインデクスページは，コピーされ多くの PE に配置される．特にルートページは，全ての葉ページへのパスを含むので，全ての PE にコピーされる．一方，葉ページへのパスを少数しか持たない下位のインデクスページは，他の PE にコピーを持つ確率が低く，コピーが存在してもその数は少ない．この性質は，アクセス頻度が高い上位のインデクスページは多くの PE で処理可能とし，更新頻度の高い下位のページはコピー間の同期を減らせるため，検索および更新の両方に貢献する．

検索時，ある PE において木をルートから順に辿っていくが，もしこの PE に目的とするインデクスページが存在しなければ，そのインデクスページを持つ PE に処理が転送される．これを繰り返すことにより，最終的には目的とする葉ページに到達することができる．

一方，更新時，もしコピーを持つインデクスページを更新するならば，それらの一貫した状態を保つためにコピーページも同期して更新する必要がある．この排他制御は, INC-OPT と呼ぶ並行制御方式により効率よく行なわれる [7].

3. Fat-Btree の高信頼構成

高信頼 Fat-Btree の構成方法を議論するに際し，本論文では以下の仮定と用語の定義を行なう．

- PE はプロセッサ，メモリ，ネットワークインタフェース，および Fat-Btree のソフトウェアから構成される．
- 基本構成では，各 PE ごとにローカルディスクが 1 台ずつ接続される．
- ネットワークインタフェースおよびネットワークは故障せず，システムは分断されない．

障であり，故障時はフェイルストップする．

無故障時のシステムの動作，または障害が生じてもシステムの処理能力が無故障時と同等である状態を**通常時の動作**と言う．システムの構成要素が故障し，縮退運転などにより通常時の動作よりもシステムの処理能力を縮小してサービスを提供している状態を**障害時の動作**と言う．構成要素に故障が発生し，通常時から障害時の動作に移行することを**フェイルオーバ**と言う．

無故障のシステムがいかなる単一故障に対しても，リカバリ操作により元の無故障の状態に回復可能であるとき，そのシステムは**完全回復可能**と言う．

通常時の動作でサービスされる全てのデータに対して，検索と更新処理が可能であるとき，そのシステムは**完全サービス可能**と言う．一部のサービスのみ提供可能であるとき，そのシステムは**縮退サービス可能**と言う．また，一つ以上のデータが永久に失われたとき，**データ喪失**と言う．

Fat-Btree のリカバリを考えるとき，取り扱わなければならない障害の種類は以下の 3 種類である．

トランザクション障害　トランザクションのアボートにより生じる．

システム障害　揮発性記憶装置 (メモリ) やプロセッサの故障などにより生じ，揮発性データが消失する．ソフトウェアの異常終了に伴う揮発性データの消失もこの故障に含まれる．

媒体障害　不揮発性記憶装置 (ハードディスク) の故障により，不揮発性のデータが消失することで生じる．

いずれの障害も，ログを取ることにより基本的にリカバリ可能である．トランザクション障害はログを用いて UNDO することにより完全にリカバリできる．システム障害も，故障した構成要素の交換後，コミット済みのトランザクションを REDO，未コミットのトランザクションを UNDO することにより完全にリカバリ可能である．媒体障害は定期的に取られたアーカイブログを使用することによりリカバリ可能であるが，完全にデータが修復されずデータ喪失となる可能性がある．

3.1　ロ グ 方 式

ログ方式には物理ログ，論理ログ，物理論理ログの 3 種類ある [4]．物理ログは記憶ブロック (通常ページ) 上の特定の位置にあるデータの変化を直接記録する方式である．物理ログは記憶ブロック内に存在する変数やデータ構造の論理的な意味を考慮せず，いため，物理ログはあまり利用されない．一方，論理ログは行なわれた操作の内容を記録する方式である．論理ログでは，どの記憶ブロックのどのデータが変更されたかという物理情報は考慮しない．物理ログと論理ログの折衷案が物理論理ログであり，記憶ブロックに対して物理的，記憶ブロック内のデータに対して論理的にログを取る方式である．各ログ方式のオーバヘッドは，物理ログ，物理論理ログ，論理ログの順で大きい．

3.2　信頼性のある Fat-Btree の構成方法

3.2.1　各 PE で物理論理ログを取る方式

図 2 のように各 PE で物理論理ログを取る方式について考える．この方式を PL(physiological log) 方式と呼ぶ．PL 方式はトランザクション障害とシステム障害に対して完全回復可能である．システム障害時の動作では，縮退サービス可能であるが障害から回復すれば通常時の動作となる．しかし，媒体故障が起きればデータ喪失状態となる．

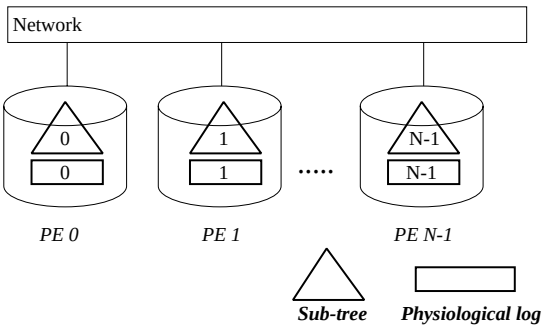

図 2　PL　方　式

3.2.2　PL とミラーリングの組み合わせ

単一媒体故障に対しても完全回復可能とするためには，各 PE のディスクを 2 台ずつ用意し，ミラーリングした上で PL 方式を適用すれば良い (図 3)．この方式を PL+mirror と呼ぶ．PL+mirror は，いずれの単一故障に対して完全回復可能である．トランザクションおよびシステム障害時の性質は PL と同一である．なお，媒体障害時は完全サービス可能である．この障害時の動作から通常時の動作への移行は，RAID レベル 1 のリカバリと同様である．

3.2.3　PL とスタガード配置の組み合わせ

PL+mirror はディスクの 2 重化によるハードウェアコストが非常に大きい．また，システム障害による障害時の動作では縮退サービス可能であるが，完全サービス可能ではない．障害時の動作でも完全サービス可能とするためには，データを 2 重化し元のデータと複製したデータを異なるディスクに置けばよい．i 番 PE

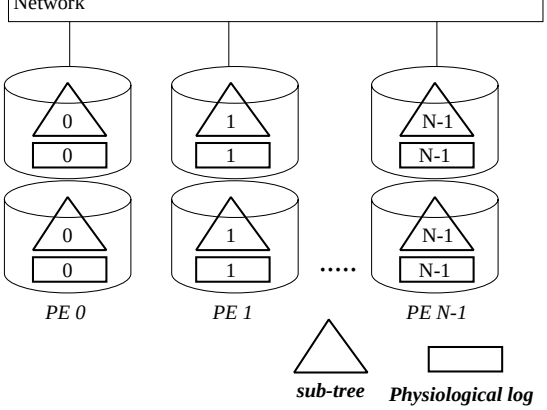

図 3　PL+mirror 方式

のディスクのデータの複製を $\sigma(i) = (i+s) \bmod N$(ただしシフト定数 s は $0 < s < N$ の整数) なる $\sigma(i)$ 番 PE のディスクに規則的に配置する方式は，スタガード配置 [4] と呼ばれる．

図 4 のような PL とスタガード配置の組み合わせをを PL+stg と呼ぶ．PL+stg はプライマリバックアップアプローチの一種である．Fat-Btree に関して，i 番 PE に存在するプライマリ側の部分木を P_i，$\sigma(i)$ 番 PE に存在する対応するバックアップ側の部分木を $B_i^{\sigma(i)}$ とすれば，もし i 番 PE のディスクが故障して P_i が失われても，P_i のバックアップ $B_i^{\sigma(i)}$ をプライマリに組み入れてシステムを再構成することにより，障害時の動作でも完全サービス可能となる．

PL+stg はいかなる単一障害に対して完全回復可能である．トランザクションおよびシステム障害時の動作は PL と同等である．なお，媒体障害時の動作でも完全サービス可能である．しかし，物理論理ログを用いているため，プライマリとバックアップがコミット時に同期しなければならないという問題点がある．

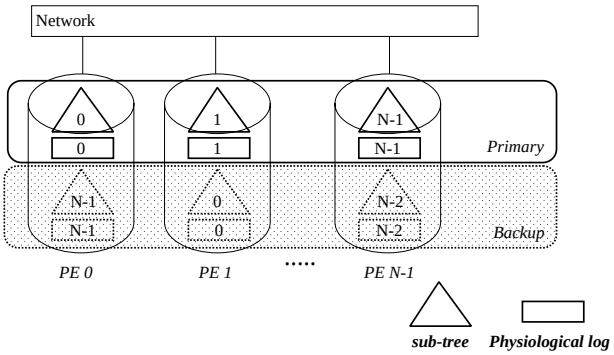

図 4　$s = 1$ の PL+stg 方式

3.2.4　グローバル論理ログとスタガード配置の組み合わせ

PL+stg ではプライマリとバックアップがコミット時に同期しなければならず，そのオーバヘッドは大きい．そこで，プライマリとバックアップを非同期更新とする方法を考える．非同期更新には操作ログ，すなわち論理ログ方式を用いてプライマリ側で実行された操作をバックアップ側に伝播すればよい．

論理ログは全ての PE での操作情報を保存するため，データ PE とは独立にグローバルな論理ログのための PE を用意する．この PE を**グローバルログ PE** と呼ぶ．このグローバルログ PE により，(1)Fat-Btree の I/O とログ I/O の分離，(2) リカバリ時に必要なログの所在が明確，(3) ログ量減少よるログオーバヘッドの軽減，という利点がある．その反面リカバリ処理が論理的になるため，リカバリのオーバヘッドが大きいという欠点がある．

図 5 のようなグローバル論理ログとスタガード配置の組み合わせを n-GLL+stg と呼ぶ．n はグローバル論理ログをミラーリングするためのグローバルログ PE の台数を示す．n-GLL+stg では，あるデータ PE がクラッシュした場合の対処は PL+stg と同様である．n-GLL+stg が，いかなる障害時の動作に対しても完全サービス可能となるためには $n \geqq 2$ でなければならない．なぜなら，$n = 1$ の時グローバル論理ログ PE に故障が発生すれば更新が行なえなくなるからである．n-GLL+stg は，各 PE でディスクを 2 重化する必要がないので，ハードディスク上のコントローラ上でデータ処理可能な自律ディスクの信頼性向上方法として提案されている [9].

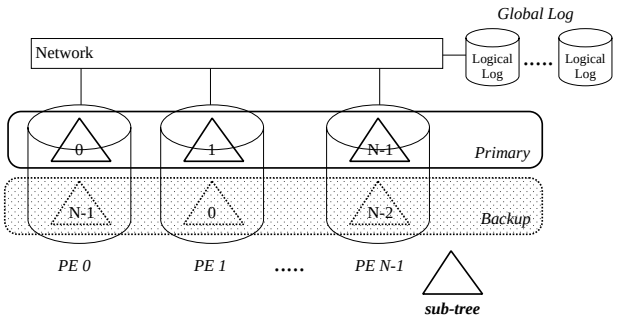

図 5　$s = 1$ の n-GLL+stg 方式

3.2.5　n-GLL+stg と PL の組み合わせ

論理ログを用いる場合，一貫しない状態でシステムがクラッシュした時単純にはリカバリできない．つまり n-GLL+stg($n \geqq 2$) では，システム障害時でさえその PE の部分木をリカバリできずにバックアップにフェイルオーバするため，アベイラビリティが低い可能性がある．そこで，n-GLL+stg に加えて各 PE で物理論理ログを追加した，n-GLL+PL+stg 方式を考える (図 6)．この方式では，データ PE にシステム

障害が生じても，その PE は物理論理ログにより高速かつ単独にリカバリ可能である．

n-GLL+PL+stg は，$n = 1$ の場合でも障害時の動作で完全サービス可能である．$n = 1$ の時グローバルログ PE に故障が起きても，PL と等価になるからである．さらに 1-GLL+PL+stg は完全回復可能である．なぜなら，グローバルログ PE がクラッシュした時には，プライマリとバックアップは整合が取れていないが，プライマリとバックアップの葉ページのデータの差分を取り，バックアップ側にこの更新差分を適用することにより，バックアップがリカバリされるからである．データ PE がクラッシュしたときは n-GLL+stg と同様のリカバリ方式が適用される．

n-GLL+PL+stg は，全ての単一故障に対して完全回復可能であり，障害時の動作は完全サービス可能である．しかし本方式は信頼性が高いものの，論理ログと物理論理ログの 2 つの処理が性能を落とす原因となる．

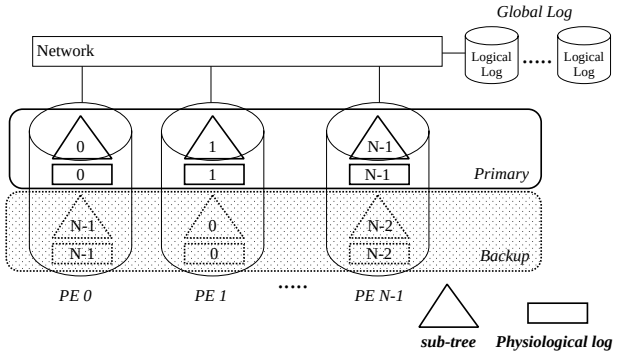

図 6　$s = 1$ の n-GLL+PL+stg 方式

3.3　各方式の信頼性

表 1 は，これまでの研究成果により各方式の性質をまとめたものである．表中のハードウェアコストの PE の内訳は，CPU，メモリおよびネットワークインターフェースである．

表 1 からデータを固く保護できるのは，PL+mirror である．しかし PL+mirror のハードウェアコストは高い．ハードウェアコストが低く，データを固く保護できるのは，PL+stg や 2-GLL+PL+stg である．しかし，PL+stg は更新時にプライマリとバックアップの同期が必要である．

一方，定常アベイラビリティに関して，低コストで高アベイラビリティを達成できるのは，2-GLL+stg と 2-GLL+PL+stg である．しかし，2-GLL+PL+stg は 2-GLL+stg よりもログオーバヘッドが大きいのが問題である．そこで，高信頼かつ高性能な Fat-Btree のために，ログオーバヘッドを削減する方法について 3. 節で議論する．

4.　高信頼 Fat-Btree の性能の見積もり

本節では，ディスクへログ書込みをする場合の通常時の動作の性能を見積もる．検索処理はログを生成しないので更新処理性能のみを考え，性能評価の尺度はシステム内で処理可能な最大更新トランザクション数 T_{system} である．故障時はデータを保護することが最重要課題であるので，本稿では故障時の性能については考えない．

近年の計算機システムでは，ネットワークのオーバヘッドはディスク I/O に比較して十分に小さく，さらに各 PE での Fat-Btree の処理とメッセージ交換はオーバラップさせることができるので，以降の性能の見積もりではネットワークのオーバヘッドは無視する．1 つの更新トランザクションのうち，更新処理のオーバヘッドを U，U に伴う物理論理ログのオーバヘッドを L_p とし，グローバル論理ログの場合は L_l とする．ここで，Fat-Btree，およびログの処理は I/O が支配的であるため，これらのオーバヘッドは I/O 処理時間と見なすことができる．

PL 方式において，単一 PE で単位時間当りに処理可能な最大負荷 W_{max} を，最大負荷時に処理可能な最大更新トランザクション数 T_1 を用いて

$$W_{max} = (U + L_p)T_1 \tag{1}$$

と表すことができる．この T_1 を用いれば，N 台のデータ PE を用いればシステム全体での最大更新トランザクション数 T_{system} は，

$$T_{system} = NT_1 \tag{2}$$

となる．PL+mirror の場合も同様である．

PL+stg の場合，1 更新トランザクション当り，各 PE でプライマリとバックアップでそれぞれ 1 つずつの更新トランザクションが必要となるので，

$$T_{system} = \frac{1}{2}NT_1 \tag{3}$$

である．

n-GLL+stg に関して，データ PE がボトルネックとなる時，つまり，データ PE で処理可能な最大トランザクション数の総和がグローバルログ PE で処理可能な最大トランザクション数を下回る時を考える．各データ PE ではログを取る必要がないので，式 (1) を変形することにより，1PE 当り

$$W_{max} = (U + L_p)T_1 = (1 + \frac{L_p}{U})UT_1 \tag{4}$$

表 1　高信頼 Fat-Btree の構成方式の比較

方式	ハードウェアコスト	媒体障害時のサービス	回復可能性	平均データ喪失時間	アベイラビリティ	ログオーバヘッド
PL	N×(PE + Disk)	縮退	データ喪失	Bad	N/A	Low+
PL+mirror	N×(PE + 2×Disk)	完全	完全回復	Best	High	Low+
PL+stg	N×(PE + Disk)	完全	完全回復	Better	High	Medium
2-GLL+stg	(N+2)×(PE + Disk)	完全	完全回復	Good	Low	Low
1-GLL+PL+stg	(N+1)×(PE + Disk)	完全	完全回復	Better−	High	High
2-GLL+PL+stg	(N+2)×(PE + Disk)	完全	完全回復	Better	High	High

の負荷を処理可能である．プライマリからバックアップへの更新は非同期に行なわれるので，システム全体では瞬間的に最大 $T'_{system} = (1 + \frac{L_p}{U})NT_1$ の更新トランザクションを処理可能である．しかし 1PE 当たり 1 更新トランザクションでプライマリとバックアップの両方の処理が必要なことから，定常時には最大更新トランザクション数はその半分の

$$T_{system} = \frac{1}{2}(1 + \frac{L_p}{U})NT_1 \tag{5}$$

となる．

一方，グローバルログ PE がボトルネックとなる場合，すなわち，データ PE で処理可能な最大トランザクション数の総和が，グローバルログ PE で処理可能な最大トランザクション数を上回る場合，グローバルログ PE で処理できる最大更新トランザクション数は，式 (1) を利用して

$$\begin{aligned} T_{system} &= \frac{W_{max}}{L_l} \\ &= \frac{U + L_p}{L_l}T_1 \end{aligned} \tag{6}$$

となる．なお，グローバルログ PE がボトルネックとならない条件は，式 (5) と式 (6) より

$$\frac{1}{2}(1 + \frac{L_p}{U})NT_1 < \frac{U + L_p}{L_l}T_1 \tag{7}$$

の時であり，式 (7) は

$$(2U - NL_l)(U + L_p) > 0 \tag{8}$$

に変形でき，$U > 0$ であるから，

$$U > \frac{1}{2}NL_l \tag{9}$$

の時である．

n-GLL+P+stg の場合，データ PE がボトルネックとなる時，各データ PE ではプライマリとバックアップはシステム全体で瞬間最大 $T'_{system} = NT_1$ の更新トランザクションが可能であるが，定常時にはその半分の

$$T_{system} = \frac{1}{2}NT_1 \tag{10}$$

となる．グローバルログ PE がボトルネックとなる時，最大更新トランザクション数は n-GLL+stg と同じく，

$$T_{system} = \frac{U + L_p}{L_l}T_1 \tag{11}$$

である．グローバルログ PE がボトルネックとならない条件は，式 (10) と式 (11) より

$$\frac{1}{2}NT_1 < \frac{U + L_p}{L_l}T_1 \tag{12}$$

の時，すなわち，

$$U > \frac{1}{2}NL_l - L_p \tag{13}$$

の時である．

5. neighbor-WAL プロトコルの高信頼 Fat-Btree への適用

5.1 neighbor-WAL プロトコル

neighbor-WAL プロトコル (nWAL) は，並列データベース HypRa [5] でログオーバヘッドを軽減するために提案された．nWAL プロトコルは，通常のログ方式のようにログデータを自分の PE の永続ストレージ (ディスク) に書くのではなく，図 7 のように自分の PE の主記憶と他の複数の PE の主記憶にログを書くことにより，ある PE にシステム障害が生じても，他の PE 上のログからリカバリ可能とするものである．本論文では，単一故障を仮定しているので自分の PE と他の 1 つの PE の主記憶にログを書くだけで十分である．

現在の並列システムでは，ディスク I/O よりもメッセージ交換の方が 2 桁以上高速であるので，nWAL の導入は性能面で大きな利益をもたらす．勿論主記憶容量は有限であるので，主記憶からログがオーバフローする場合，ログはディスクに書かれるが，

- コミット済みトランザクションの UNDO ログはディスクに書く前に消去可能
- 複数のログをまとめて逐次にディスクに書込むことが可能

であるので，ディスク I/O のオーバヘッドは適用のログ書込みよりも小さくなる．特に，n-GLL+PL+stg の物理論理ログに nWAL を適用する場合，論理ログにコミットを書込んだ時点で物理論理ログを消去可能となるため，各データ PE でのディスク I/O はさらに抑えられる．この性質により，n-GLL+PL+stg のログオーバヘッドを軽減し，高性能化することができる．

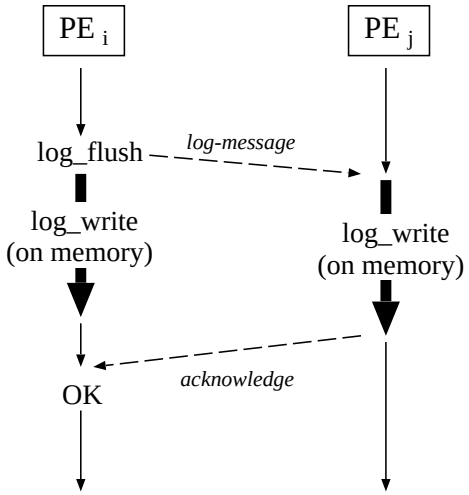

図 7　nWAL プロトコル

5.2　nWAL 適用時の処理性能

4. 節では，高信頼 Fat-Btree にディスクベースのログを適用した場合の更新トランザクションの処理性能の見積もりを行なった．本節では，nWAL を適用した場合の処理性能の見積もりを行ない，どの程度性能が改善可能であるかを明らかにする．

PL および PL+mirror の場合，1PE 当りの最大更新トランザクション数は，nWAL 使用時のログオーバヘッドを L_n とすれば，

$$\begin{aligned} T_1 &= \frac{W_{max}}{U+L_n} \\ &= \frac{U+L_p}{U+L_n}T_1 \\ &= (1+\frac{L_p-L_n}{U+L_n})T_1 \end{aligned} \tag{14}$$

となる．システム全体では式 (14) より

$$T_{system} = (1+\frac{L_p-L_n}{U+L_n})NT_1 \tag{15}$$

の更新トランザクションが処理可能である．式 (2) と式 (15) を比較することにより，nWAL の適用により $(L_p-L_n)/(U+L_n)$ の分だけスループットが向上する．

PL+stg の場合，PL の半分の処理能力となるから，

$$T_{system} = \frac{1}{2}(1+\frac{L_p-L_n}{U+L_n})NT_1 \tag{16}$$

である．式 (3) と式 (16) を比較すれば，nWAL の適用により $(L_p-L_n)/(U+L_n)$ だけスループットが向上する．

n-GLL+stg の場合，論理ログにも nWAL を適用すると仮定し，グローバルログ PE の主記憶に論理ログを格納するオーバヘッドも L_n であるとすると，グローバルログ PE がボトルネックとなる時，最大更新トランザクション数は，

$$\begin{aligned} T_{system} &= \frac{W_{max}}{L_n} \\ &= \frac{U+L_p}{L_n}T_1 \end{aligned} \tag{17}$$

である．これは式 (6) と比較して L_l/L_n 倍多くトランザクションを処理できることを示している．一方，グローバルログ PE がボトルネックとならない時は，nWAL を用いない場合と同様，定常時

$$T_{system} = \frac{1}{2}(1+\frac{L_p}{U})NT_1 \tag{18}$$

の更新トランザクションを処理可能である．この時は nWAL によるスループットの改善はない．なお，グローバルログ PE がボトルネックとならない条件は，式 (17) と式 (18) より

$$U > \frac{1}{2}NL_n \tag{19}$$

の時である．

n-GLL+PL+stg の場合，グローバルログ PE がボトルネックとなる時，最大更新トランザクション数は n-GLL+stg と同じく，

$$T_{system} = \frac{U+L_p}{L_n}T_1 \tag{20}$$

である．一方，グローバルログ PE がボトルネックとならない場合は，各データ PE では定常時，最大

$$T_{system} = \frac{1}{2}(1+\frac{L_p-L_n}{U+L_n})NT_1 \tag{21}$$

の更新トランザクションが処理可能である．これを式 (10) と比較すれば，nWAL の導入により $(L_p-L_n)/(U+L_n)$ 分だけ更新スループットが改善されることが分かる．式 (18) と式 (21) を比較すれば，もし，更新処理と物理論理ログに対する nWAL のオーバヘッドが十分に小さければ，当然の結果ではあるが n-GLL+stg との性能差はほとんどない．n-GLL+PL+stg に nWAL を適用する場合，論理ログにコミットを書込んだ時点で物理論理ログを消去可能であり，L_n の

定は現実的である．

なお，グローバルログ PE がボトルネックとならないための条件は，式 (20) と式 (21) より

$$U > (\frac{1}{2}N - 1)L_n \tag{22}$$

の時である．

6. おわりに

本稿では，Fat-Btree の高信頼構成に HypRa で使用された neighbor-WAL プロトコルを適用することにより，高信頼 Fat-Btree の更新トランザクション処理性能を改善する方法について議論した．

ハードウェアコストを考えないで良いならば，PL+mirror は最も高信頼かつ高性能である．しかし，ハードウェアコストと高信頼性の両立を考えなければならないならば，PL+stg，n-GLL+stg または n-GLL+PL+stg が良い．PL+stg はコミット時にプライマリとバックアップの同期が必要であるが，n-GLL+stg と n-GLL+PL+stg では非同期にプライマリとバックアップを更新できる点で優れている．n-GLL+PL+stg はデータを固く保護できる点で n-GLL+stg よりも優れているが，2 種類のログによるオーバヘッドが大きいという問題があった．しかしながら，nWAL プロトコルを導入することにより，n-GLL+PL+stg は n-GLL+stg とほとんど性能に差がなくなることを示した．つまり nWAL を導入することにより，n-GLL+PL+stg はハードウェアコストが低く，高信頼かつ高性能な Fat-Btree の構成方法であると言える．

今後，nWAL の高信頼 Fat-Btree 構成への導入の効果をシミュレーションもしくは実装することにより具体的に評価していきたい．

文　献

[1] G. Copeland, W. Alexander, E. Boughter, and T Keller. Data Placement in Bubba. In *Proc. of ACM SIGMOD Conf. '88*, pp. 99–108, 1988.

[2] David DeWitt and Jim Gray. Parallel Database Systems: The Future of High Performance Database Systems. *Communications of the ACM*, Vol. 35, No. 6, pp. 85–98, June 1992.

[3] S. Ghandeharizadeh and D. J. DeWitt. Hybrid-Range Partitioning Strategy: A New Declustering Strategy for Multiprocessor Database Machines. In *Proc. of VLDB Conf. '90*, 1990.

[4] Jim Gray and Andreas Reuter. *Transaction Processing: Concepts and Techniques*. Morgan Kaufmann, San Francisco, 1993.

[5] Svein-Olaf Hvasshovd. *Recovery in Parallel Database Systems*. Database Systems. Vieweg, 1996.

[6] C. E. Leiserson. Fat-Trees: Universal Networks for Hardware-Efficient Supercomputing. *IEEE Transactions on Computer*, Vol. 34, No. 10, pp. 892–901, 1985.

[7] Jun Miyazaki and Haruo Yokota. INC-OPT: A High Performance Concurrency Control for Parallel B-tree Structures. Technical Report IS-RR-2001-002, JAIST, January 2001.

[8] H. Yokota, Y. Kanemasa, and J. Miyazaki. Fat-Btrees: An Update-Conscious Parallel Directory Structure. In *Proc. of IEEE ICDE Conf. '99*, pp. 448–457, 1999.

[9] Haruo Yokota. Autonomous Disks for Advanced Database Applications. In *Proc. of DANTE '99*, 1999.

[10] 風戸広史, 横田治夫. 並列ディレクトリ構造 Fat-Btree におけるレンジ問い合わせの取り 扱い. 第 12 回データ工学ワークショップ論文集, 2001.

[11] 宮崎純, 横田治夫. 並列ディレクトリ構造 Fat-Btree の並行性制御とその評価. 情処研報 DBS-124-69, 2001.

[12] 渡邊明嗣, 横田治夫. 探索コスト評価による分散ディスク偏り制御. 情処研報 DBS-124-68, 2001.