社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2007－DPS－132 (21)
2007－GN－ 65 (21)
2007－EIP－ 37 (21)
2007／9／21

並列可変長アクセス方式における読込方式の提案

境美樹† 奥村恭弘† 伊織生美† 新井克也†

並列可変長アクセス方式の書込み方式では、1ファイルを複数の物理ファイルに分割保存し、書込み要求毎に異なる物理ファイルへの書き込みを実施することで、書込み要求を並列に処理することが可能なり、結果としてトータルスループットが向上する方式である。このように1ファイルを複数物理ファイルに分割して保存したデータをシーケンシャルに読込む際の方式として、インデクス情報を利用する方式を提案し、実験によりその有効性が確認できたので報告を行う。

A sequential read method for PVA files

MIKI SAKAI† YASUHIRO OKUMURA† KIYOSHI IORI† KATSUYA ARAI†

Abstract

Parallel Variable Access or PVA is a method to increase total write throughput by parallel processing of write requests. In PVA, a single logical file is divided into multiple physical files, and each write request is assigned to different physical file. In this paper we propose a method using index information for sequential read of the logical file from those physical files. We also report experimental results which confirm the efficiency of the proposed method.

1. はじめに

Linux上でジャーナルシステム（システムの耐障害性を実現するため処理内で実施するデータ更新処理の度取得する更新内容の差分データを保存する機構）の下回りなど、高スループットを要求される領域をターゲットとして、我々は追記型ファイルアクセスシステムである、並列可変長アクセス方式を提案した。[1]

並列可変長アクセス方式は、一般的なLinuxのファイルシステム（例えばEXT3）上で、1ファイルに対して同時複数の書込み要求が発生する場合に、通常のファイル書込みよりもトータルスループットが向上する方式である。この方式は、1ファイルを複数の物理ファイルに分割保存し、書込み要求毎に異なる物理ファイルへの書き込みを実施することで、並列に書込みを実施できるので、単位時間当たりの総書き込みデータ実効転送速度（トータル書込みスループット性能）が向上する。

しかし、本方式で書かれたデータを読み取る際には、読込み対象となるデータがどの物理ファイルに配置されているかを検索する必要があり、その分読込スピードが低下するといった問題があった。

本稿では、このように1ファイルを複数物理ファイルに分割して保存する並列可変長アクセス方式にて書き込まれたデータをシーケンシャル（書き込まれたデータ順）に読み込む際の高速化を目指す。

2章では、並列可変長アクセス方式の概要及び、問題点の説明を行い、3章では読込方式を提案する。4章・5章では各方式について評価・考察を行い、提案手法の効果を示す。

† NTT 情報流通プラットフォーム研究所, NTT Information Sharing Platform Lab.

2. 並列可変長アクセス方式とその問題点

本節では、トータル書込みスループット性能を向上させる並列可変長アクセス方式の書込み手法の概要を説明し、読込時の問題点を挙げる。

2.1 並列可変長アクセス方式の書込み手法

ジャーナルシステムなど、複数のAP（OS上で動作するプロセス又はスレッド）やシステムからの書込み要求が発生する状況を想定している、並列可変長アクセス方式は、図1の様に、同一ファイルに対して、複数のAPやシステムから書込み要求が発生する事を前提としている。

一般的に、同一ファイルへの書込み要求が同時複数発生した場合、ファイル破壊を防ぐ目的から排他制御を行い、一つの書込み要求処理を終了した後に次の書込み要

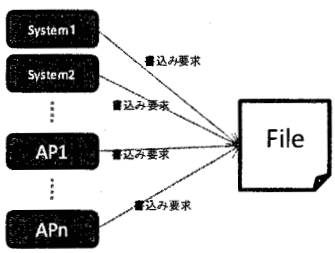

図1 複数システム／APからの書込み

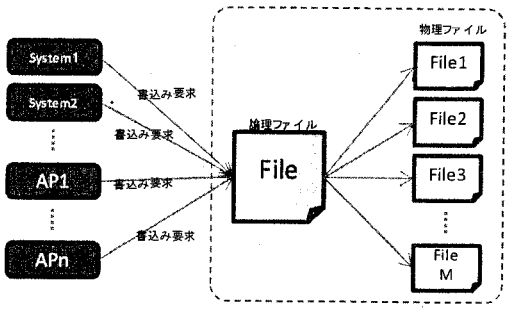

図2 並列可変長ファイル方式での書込み

求を処理するというように、複数ＡＰからの書込み要求はシーケンシャルに処理される。

これに対して、我々の提案した並列可変長アクセス方式では、図2に示した通り、1つのファイル（論理ファイルと呼ぶ）を複数の物理ファイルに分割し、書込み要求毎に書き込む先のファイルを変更することで、処理中の書込み要求の完了を待たずに次の書込み要求を並列に実施することが可能となり、結果的にトータル書込みスループット性能が向上する。

予備実験の結果では、EXT3のファイルへのシーケンシャル書込みに対して、最大4倍のスループットが出ることが確認されている。

具体的なトータル書込みスループット性能の向上のための工夫点としては、以下の2点があげられる。

A) 複数物理ファイルへデータを分散格納する。

B) 書込み先のファイルの決定はラウンドロビンでは、書込み待ちが発生する場合があるので、書込み依頼時に書込み中ではないファイルから順に行う。但し、すべてが書込み中ならば、待ちは発生する。

これらの工夫により、APから見た場合の書込み待ち時間を少なくする効果が望める。

A)の工夫によって、データを分割するためのオーバーヘッドがかからないという利点がある。B)では、書込み中ではないファイルへ順次振り分ける事で、処理待ちの時間を少なくする効果が望める。

2.2 読込時の問題点

一方、2.1の手法で書き込まれたデータを読込む場合を考える。

並列可変長アクセス方式においては、あるAPの指定したデータから順次データの読込を行う。具体的には、以下の手順で読込処理を行う。

① 【AP】読込依頼を発行（読み込み開始位置を指定）
② APから指定された位置へシーク
③ データを読込みAPへ返却
④ 【AP】次のデータの読込依頼を発行
⑤ ②へ戻る

ここで、②のAPから指定された位置へシークする際に、次の問題点がある。

- ラウンドロビンで書き込み先を決定しないので、データの通番（書込み時に書込み依頼受付順に発行されるシーケンシャルな番号で、読込時は位置指定に利用する）を指定されても、該当するデータがどの物理ファイルに配置されているかが即座には分からない。
- あるデータの次のデータを読み込む場合、同一物理ファイルに該当データが存在するとは限らない

これらの理由から、各書込みデータの物理ファイルへの配置は必ずしも書込み順にラウンドロビンでは配置されるとは限らない。例えば、図3では3つの物理ファイル内へのデータの配置状況を示しているが、通番1～3まではラウンドロビンに配置されている。しかし、システムの状況によって、通番4～5は1つ目の物理ファイルへ、通番6と9は2つ目の物理ファイルへ、通番7と8は3つ目の物理ファイルへ配置される、と言うようにランダムに近い状態で配置される現象が起きる。

通常のファイル（EXT3など）の場合は、次のデータを読込場合にはread()命令を発行するだけで良い。しかし、並列可変長アクセス方式で書かれたファイルからシーケンシャルにデータを読込際、②部分で該当データがどの物理ファイルに配置されているかを探すための処理が入るため速度低下の原因になると考えられる。

3. 読込方式の提案

本章では、並列可変長アクセス方式で書き込まれたファイルの読み込みオーバーヘッドを軽減させるための方式として、インデクス情報を用いる手法を提案する。

ここで、前述のジャーナルシステム用途などを想定していることから、主に特定の位置から順次データを読み

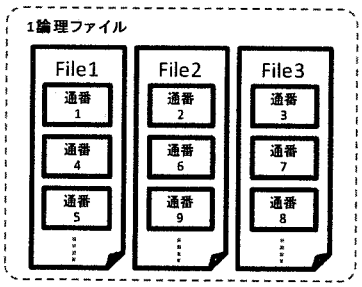

図3 書込みデータの配置例

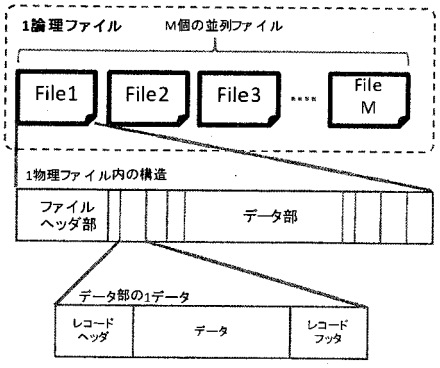

図4 並列ファイルのデータ構造

出す、シーケンシャルリードについて高速化を目指すものとする。

また、ジャーナルは、システム障害発生時にジャーナルデータを順次読込んで欠損したデータを修復していく用途に利用する。この事から、本稿では、読込を依頼するAPまたはシステムは、単一であるものとし、書込みのように同時複数の要求が発生しないものとする。

3.1 並列可変長アクセス方式のデータ構造

本節では、読込時に重要となる並列可変長アクセス方式で書き込まれたファイルのデータ構造について示す。

本方式では、データ書込み時に書込み依頼受付の順序を示す通番(1からインクリメントされる値)を付与し、読込時に書込み依頼受付順にデータの読み出しが可能な状態を作っている。

具体的なデータ構造としては、図4に示す様に、1論理ファイルを複数の物理ファイルに分割して保存している。その物理ファイルを「並列ファイル」と呼ぶ。

各々の並列ファイルは、ファイルヘッダ部及び、データ部を持っている。

ファイルヘッダ部には、ファイルの利用状況(利用サイズや書込まれているデータの最新通番など)が保持されている。

データ部は、複数のレコードから構成される。レコードは、レコードヘッダ部/データ部/レコードフッタ部から成っており、レコードヘッダ部には、そのデータの通番や書込み日時が含まれている。レコードフッタ部はレコードヘッダ部と同様の内容が含まれている。

データ書込み時には、レコードヘッダ部とレコードフッタ部をセットした情報を書き込んでいる。

3.2 読込方式

2.2で示した読込方式において、読込対象のデータがどの並列ファイルに配置されているか検索する必要があった。

ここでは、検索にかかる時間が減少するよう、あらかじめ通番とその通番のデータを保持している並列ファイルの組をインデクス情報としてメモリ上に構築しておき、そのデータを利用して読込を行う方式を提案する。

インデクス情報の構築タイミングについては、初回の読込依頼を受付けた際に、データを先読みしておき生成する方法が考えられる。また、メモリ上の領域は有限であるため、大量のデータが書き込まれている場合は、すべてのインデクス情報をメモリ上に展開できない場合がある。この場合は、APから指定された位置からインデクス情報を構築し、構築したインデクス情報のデータをすべて読み終わった時に、次の位置からインデクス情報を展開する。

この手法の場合、インデクス情報構築時には時間がかかるが構築時以外のシーケンシャルリードに対しては高速化が望める。

3.3 書込み時のインデキシング

3.2のインデキシングでは、メモリ上に構築されたインデクス情報を利用したシーケンシャルリード自体は高速化すると考えられる。しかし、メモリ上にインデクス情報を構築するために余分なファイルI/Oなどのオーバーヘッドが発生する。

そこで、本節では、書込み時に別途インデクス情報を作成し、読込時はその情報を元に読込を実施する方式をとることで、読込スループットの向上が図れるものと考える。

この手法の場合、書込み要求毎にインデクス情報をインデクスファイルへ書き出してしまうと、I/O発生によるオーバーヘッドが大きくなると予測される。そこで、インデクス情報は基本的にメモリ上に保持しておくものとする。これは、インデクス情報自体は破損しても、並列可変長アクセス方式の並列ファイル全てが残っていれば再構成出来るからである。そして、並列可変長アクセス方式の終了時もしくは、インデクス情報を保持するバッファが溢れた時に書き出しを行うようにし、インデクス情報書込みのためのI/Oオーバーヘッドを避けることにする。

4. 実験

本章では、3章で示した提案それぞれについて実施した実験結果を示す。

4.1 実験環境

実験は、以下の環境で実施した。

また、読込速度を計測する際は、カーネルキャッシュの影響を考慮し、実験ごとにOSの再起動を行った。

表1　ハードウェア構成

ハードウェア名	スペック	
IBM System X 3550	CPU	DualCoreIntel 1.6GHz * 1
	Memory	512MB * 4
IBM DS4700 Express	HDD（Share Disk）	36.4GB * 3　FC 4Gbps
	キャッシュメモリ	2GB

表2　ソフトウェア構成

ソフトウェア名	備考（Version 等）
Red Hat Enterprise Linux ES	Version4　Update4 kernel2.6.9-42

また、並列可変長アクセス方式で作成したファイルの構成は、以下のとおりである。

表3　並列可変長アクセス方式構成

項目	設定値
並列数	16
1 書込みデータサイズ	1024byte（ヘッダ／フッタも含む）
レコード数	16000

今回の実験では、16 個の物理ファイルで構成される 1 論理ファイルから、データの読込／書込み速度の測定を実施した。

4.2 読込速度

読込速度の実験では、3.2 にて提案したインデクス情報を利用した読込について検証を行った。

ここでは、インデクス情報を利用した場合とインデクス情報を利用しない場合の 2 パターンについて測定を行った。

また、本測定では、データの先読みバッファを利用し、そのバッファサイズを変動させることでの読込速度の変化も合わせて調べている。ここで、図 5 に示すように、並列ファイル毎に読込用バッファ準備し、先読みを行う。無駄読みを避けるため、先読みを実施するタイミングは読込み対象データを読込む時とする。

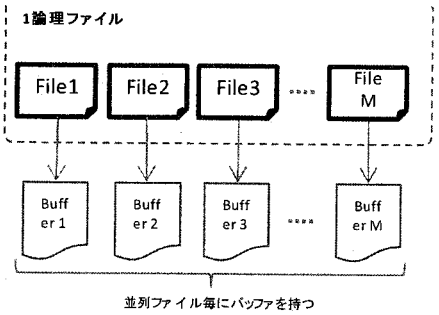

図5 バッファの配置

インデクス情報ありとなしで、各々のファイルの持つバッファ量を 1024byte～32768byte（1 レコード～32 レコード）に変動させて測定した。

ただし、これらのバッファは、並列ファイル毎に持っており 16 個の物理ファイルが存在する。そのため、バッファを 1 レコードとした場合でも、実際は 16 レコード分のバッファを持っている。

インデクス情報ありの場合は、読込対象データすべてのインデクス情報がメモリ上に存在している場合について測定を行った。

測定結果を表 4 及び図 6 に示した。バッファサイズに依らず、インデクス情報の利用により、読み込み速度が速くなることが確認できる。

4.3 書込み速度

インデクス情報をメモリ上へ構築する場合、書込み時に行うことが有効であると考える。そこで、書込み速度自体にどの程度の劣化が発生するかを調べた。

ここでは、

① インデクス情報生成がない場合（通常の並列可変長アクセス方式の書込み）
② メモリ上へのインデクス情報生成がある場合
③ ファイルへのインデクス情報書込みを毎回行う場合

とで、書込みにかかる時間を測定し、それらの比較を行った。①～③はインデクス情報生成部分のみ差異があり、他は同一シーケンスで処理を行っている。つまり、処理時間の差は、インデクス情報生成部分のオーバーヘッドとなる。

書込み時の条件は以下のとおりである。

- 書込み用 AP を 1 つ起動し、16000 回書込みを行う。
- 1 回あたりの書込みサイズは 1024byte とする。

測定結果を表 5 に示した。

表 4 並列可変長ファイル方式の読込速度

バッファサイズ(byte)	1024	4096	8192	16384	32768
インデックスなし(sec)	1.458677	0.563648	0.349364	0.245556	0.173866
インデックスあり(sec)	1.017584	0.405772	0.257395	0.179955	0.135339

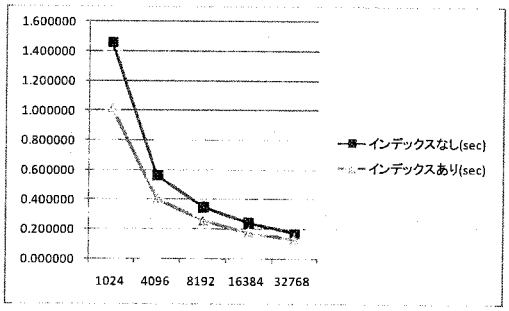

図 6 並列可変長ファイル方式の読込速度

表 5 書込み速度

	インデックスなし	インデックスあり(メモリ上)	インデックスあり (ファイル書込)
書込み時間 (sec)	6.125102	6.156044	11.658829

5. 考察

5.1 読込速度に対する考察

図 5 のグラフが示すように、インデクス情報を利用することで、利用しない場合に比べて 1.3 倍～1.4 倍の高速化が見られた。これは、インデクス情報から読込対象のデータがどの並列ファイルに配置されているかが即時に把握可能となり、検索する処理時間の短縮が図られたためである。

また、読込用バッファサイズを大きくすることで、読込速度が向上する事が確認できる。実験環境においては、バッファサイズが 32Kbyte の時、1Kbyte に比べておよそ 8 倍程度の向上がみられる。これは、バッファサイズを増やすことで、トータルの disk I/O 回数を減少させた事が速度向上のキーであると考える。1k の場合、16000 回の読込依頼に対して、disk I/O も 16000 回行われているが、4k ならば 4000 回、8k なら 2000 回と disk I/O 回数が減っている。バッファサイズを増加させた場合の速度向上が頭打ちになっているのは、メモリ上の操作が増加するためであると考える。

5.2 書込み速度に対する考察

表 5 では、書込み時のインデクス構成に要する時間を表している。

この結果から、メモリ上に展開する分には、1 書込みあたり 6μSec 程度のオーバーヘッドであることが分かり、書込み性能へのインパクトは小さいものと考える。

一方、書込み毎にインデクス情報をファイルへ書き込んでいる場合は、およそ倍の書込み時間がかかってしまう。これは、1 回の書込み要求に対して 2 回の disk I/O が発生するためである。

インデクス情報をメモリ上に展開するのは有用であるが、インデクス用領域のサイズを超える書込み要求が行われた場合は、書込み処理内で、インデクスファイルへの書込みを行う必要があり、オーバーヘッドが発生する。また、読込み要求時、該当するインデクスがメモリ上にない場合にも読み込みが発生してしまう。

つまり、常に大量の書込み要求がある様な用途の場合は適さないと思われる。逆に、一時的に大量の書込み要求が発生するが、インデクス用領域のサイズを超えない程度で一時的に要求が疎になるような環境においては、疎になっている間にインデクスファイルの書込みを行うことで、書込み処理自体へのインパクトを少なくすることが可能であると考える。

6. まとめ

本稿では、並列可変長アクセス方式を用いて作成されたファイルの読込手法の提案及び、提案手法の実機検証を行った。

その結果、インデクス情報の活用の有効性を示すことができた。また、バッファサイズを増やす事で Disk I/O 回数を減少させることが可能となり、読込速度の向上が望めることが分かった。

更に、インデクス情報を書込み時に構成することのオーバーヘッドが小さいことが確認でき、インデクス情報を利用した読込方式をとることの現実性を示すことができた。

参考文献

[1] 奥村 他, 並列可変長アクセス方式による書込みスループット向上についての検証. 情報処理学会 DPS/GN/EIP 合同研究会, 2007 年 8 月.