B-041

多コアファイルサーバのメモリ回収遅延を削減する分割リクレイム方式

Split-Reclaim Method for Reducing Page De-allocation Delay on Multi-core File Server

深谷 崇元† 松沢 敬一† 揚妻 匡邦† 中村 隆喜†
Takayuki Fukatani Keiichi Matsuzawa Masakuni Agetsuma Takaki Nakamura

1. はじめに

近年のプロセッサ微細化技術の発展により，汎用サーバでも数十のCPUコア(コア)が使用可能となった ([1]). 多数のコアを用いた多コアサーバにて高い性能を実現するには，処理内容に応じたカーネルの処理並列化と資源管理が重要となる. 例えば [2]は, Linuxカーネルで 48 コアまで性能をスケールさせるためには，アプリケーション指向のカーネル改変が必要であることを示した. 本論文は，多コアをファイルサーバに適用するために必要となるLinuxカーネルの改変方式を対象とする.

多コアファイルサーバでは，メモリ不足時のクライアント応答時間の悪化が問題となる. これはコア数増加によりNFS(Network File System)デーモンのメモリ消費速度があがり，メモリ回収処理が長期化したことに起因する.

我々は，本問題を解決すべくキャッシュ種別ごとにメモリ回収処理を分ける，分割リクレイム方式を提案する. 本方式では，メモリ回収処理を低速なメタデータキャッシュ(SLAB)の回収処理と，高速なページキャッシュの回収処理とに分ける. ページ不足時には，高速なページキャッシュの回収処理を用いることで，メモリ回収処理遅延を削減する.

以降では，従来のメモリ回収方式の問題点を説明し，分割リクレイム方式の実装と評価を述べる.

2. 従来方式と問題点

2.1 Linux のメモリ管理と従来のメモリ回収処理

はじめに本研究の前提となる Linux カーネルのメモリ管理とメモリ回収処理について述べる.

Linuxカーネルの主なメモリ管理は，4KBのページ単位で管理を行うページキャッシュと，ページを更に分割して管理するSLABの管理となる([3]).

ページキャッシュはファイルデータなど比較的大きなデータのために使用され，SLAB はファイルメタデータ(inode)やパス情報(dentry)など 4KB 以下のデータを扱うために使用される. Linux カーネルは，ページキャッシュと SLAB を，それぞれ独立した LRU リストにて管理し，メモリ不足時にそれぞれ使用頻度の低いものから回収する.

メモリ回収処理は，メモリ回収デーモン(kswapd)によるバックグラウンド処理と，バックグラウンド処理が間に合わない場合のための，NFS デーモンなど一般プロセスによるダイレクトリクレイムの 2 種類がある(図 1).

バックグラウンド処理は，空きページ数がバックグラウンド処理開始しきい値(K_{start})以下となった際に，kswapd がページキャッシュと SLAB の回収処理を行う. 一方，ダイレクトリクレイムは，空きページがダイレクトリクレイム開始しきい値（D_{start}）以下となった際に，一般プロセスがページ獲得時，ページキャッシュと SLAB を回収する. ダイレクトリクレイム発生時には，一般プロセスが本来の処理を中断してメモリ回収を行うため，メモリ回収による処理時間の遅延（以下、メモリ回収処理遅延）が発生する.

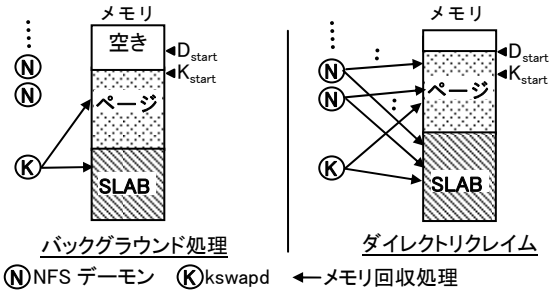

図 1 従来方式のメモリ回収処理

2.2 多コアファイルサーバでの従来方式の問題点

多コア環境ではプロセスの並列実行数が増えメモリ消費速度があがる. その結果，バックグラウンド処理のメモリ回収が間に合わなくなり，ダイレクトリクレイムが発生しやすくなる. 従来のダイレクトリクレイムでは，ページ単位のメモリ獲得要求に対して，ページキャッシュの回収処理だけではなく，SLAB 回収処理も合わせて発生する. SLAB 回収ではページ中の全ての SLAB を回収するまで空きページを増やすことができないため，メモリ回収処理に時間がかかる. 特に，ファイルサーバは，メモリ上の SLAB の容量や使用頻度が相対的に大きく，この傾向が強い. 加えて，あるプロセスのダイレクトリクレイム中にも，他のプロセスがダイレクトリクレイムを開始するため，メモリ回収処理遅延が更に悪化する.

その結果，メモリ不足時には，メモリ回収処理遅延が原因でクライアントへの応答時間が劣化し，問題となる.

3. 分割リクレイム方式

我々は，キャッシュ種別ごとのメモリ回収速度の差異に着眼した. 従来のダイレクトリクレイムは，低速な SLAB 回収を行うためメモリ回収処理遅延が問題となる. しかし，メモリ上にページキャッシュが残っている状態であれば，ダイレクトリクレイム時には高速なページキャッシュ回収のみ行えばよい. そこで，ダイレクトリクレイムを，ページキャッシュの回収処理と SLAB のメモリ回収処理とに実行契機を分ける分割リクレイム方式を提案する.

本方式では，SLAB 回収のために，新規に SLAB 専用の回収処理（SLAB リクレイム）を導入する. SLAB リクレイムでは，SLAB 容量がしきい値を超えた場合，一般プロセスが

†(株)日立製作所 横浜研究所
Yokohama Research Laboratory, Hitachi, Ltd.

SLAB 回収を行う. 一方, ページキャッシュについては, 従来のダイレクトリクレイム同様, 空きページ数がしきい値以下となった場合に, 一般プロセスが回収する.

分割リクレイム方式の動作を, 図 2 を用いて説明する. 提案方式では, メモリ回収のしきい値として, 従来の K_{start} の他に, ページリクレイム開始しきい値（P_{start}）, SLAB リクレイム開始しきい値（S_{start}）を持つ. P_{start} は従来方式の D_{start} に相当する. S_{start} のデフォルト値はメモリ容量の 5 割とし, アクセスパターンに応じ変更可能である.

バックグラウンド処理は, 従来と同じく kswapd によるページキャッシュと SLAB の回収処理を行う（図 2 左）. ページリクレイムは空きページ数が P_{start} 以下となったら, ページを獲得する一般プロセスが, ページキャッシュを回収する（図 2 中）. ページリクレイムは回収対象がページキャッシュに限定されるため, 従来のダイレクトリクレイムに比べ高速にメモリを回収できる. SLAB リクレイムは, SLAB 容量が S_{start} になったら, SLAB を獲得する一般プロセスが SLAB を回収する.

以上の説明で示したとおり, 分割リクレイム方式を用いた場合, メモリ不足時に高速なページリクレイムを使用するため, メモリ回収速度が向上する. NFS デーモンのメモリ回収処理の並列実行数も削減され, 従来のメモリ回収処理遅延を削減することができる.

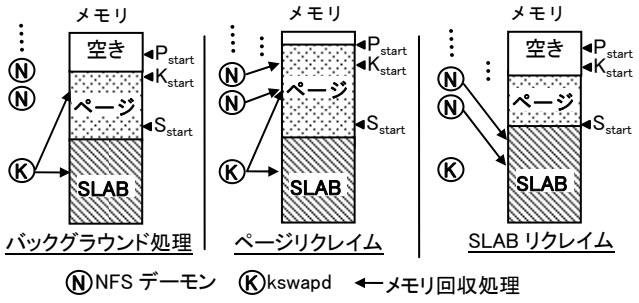

図 2　分割リクレイム方式

4. 評価

分割リクレイム方式の効果を確認するため, Linux 2.6.39rc4 に分割リクレイム方式を実装し, 性能評価を行った. 評価用サーバには Xeon7550 を 4 プロセッサ(合計 32 コア), 64GB メモリを搭載し, 28 台の SSD に Ext3 ファイルシステムを構築した. クライアントには, 8 コアサーバを二台使用し, それぞれ 300 の負荷生成プロセスを動作させた.

最初に, 標準的なファイルアクセスパターン([4])にて, 1 秒あたり 6 万回の負荷をかけた際の, リクレイム処理の発生回数, 平均処理時間, 最悪処理時間を評価した(表 1).

表 1　標準的アクセスパターンでのリクレイム処理

項目	従来	提案方式	
		ページ	SLAB
発生回数	52 回	45 回	0 回
平均処理時間	7.5 秒	9.1ミリ秒	-
最悪処理時間	8.5 秒	26ミリ秒	-

提案方式は, リクレイム処理の発生回数を 13%削減し, 平均処理時間, 最悪処理時間ともに 99%以上削減している. また, 本アクセスパターンでは, SLAB の割合が上限値に達していないため, SLAB リクレイムによる遅延は発生しない. 以上の結果より, 通常のファイルアクセスパターンにおいては, 提案方式によりメモリ回収処理遅延を大幅に削減できると言える.

次に, SLAB リクレイムのオーバヘッドを評価するため, 意図的にメモリ上の SLAB 量を増やし, 多量の SLAB リクレイムを発生させる最悪パターンによる評価を行った. 7800 万ファイルに対するファイル属性参照の負荷をかけた際の評価結果を表 2 に示す.

表 2　最悪パターンでの評価

項目		従来	提案方式	
			ページ	SLAB
リクレイム処理	発生回数	150 万回	610 回	7000 万回
	平均時間	5.1 秒	13ミリ秒	2.1μ秒
	最悪時間	25 秒	100ミリ秒	3.2 秒
全ファイルの処理時間		1100 秒	1500 秒(+35%)	

提案方式で, SLAB リクレイムが増えるため, リクレイム回数が 47 倍となったものの, 平均処理時間は 99%, 最悪処理時間は 87%削減できている. そのため, 目的としたメモリ回収処理時間の削減は実現できていると言える. 一方, 従来方式に比べ 35%処理時間が長くなっており, 提案方式のオーバヘッドとなっている.

5. おわりに

多コアファイルサーバのメモリ回収処理遅延削減のため, 分割リクレイム方式を提案した. 同方式では, ダイレクトリクレイムを, SLAB リクレイムとページリクレイムに分けることで, メモリ不足時のページ回収処理遅延を削減する. 提案方式の実機評価により, 標準的なアクセスパターンでメモリ回収処理遅延を 99%以上削減できることを確認した. 一方, 最悪パターンでは, 提案方式によりメモリ回収遅延は削減できているものの, SLAB リクレイム処理のオーバヘッドで全体処理時間が 35%長くなっている. 最悪パターンにおける, 提案方式のオーバヘッドを削減することが今後の課題である.

参考文献

[1]Andi Kleen, Linux multi-core scalability, *Proceedings of Linux Kongress 2009*, October 2009.
[2]Silas Boyd-Wickizer and etc., An Analysis of Linux Scalability to Many Cores, *Proceedings of the 9th USENIX OSDI 2010*, September 2010
[3]D. P. Bovet and M.Cesati, Understanding the LINUX KERNEL, 3rd Edition, O'REILLY
[4]SPEC.org, "SFS3.0"