D-007

ストレージ階層仮想化機能における高速な移動先階層の判定方法

Fast Determination of Allocation Tiers in Virtual Storage Tiering Functions

坏 弘明† 今崎 美保† 須藤 梓† 大平 良徳† 江口賢哲†
Hiroaki Akutsu Miho Imazaki Azusa Sudo Yoshinori Ohira Yoshiaki Eguchi

1. まえがき

年々増加するデジタルデータ量に対し、IT への投資額は増加していない傾向にある。このためペタバイト(PB)クラスの大容量ストレージを運用する企業は、IT システムの性能要件とコスト要件の両立を実現するために、アクセス頻度に応じて適切なストレージ階層（高速なメディア(記憶媒体)や安価で低速なメディア）にデータを格納し、データ保持コストを削減している。しかし、年々増え続けた大量のデータをアクセス頻度に応じたストレージ階層に格納することが課題となっている。その解決手段として、ストレージ領域のアクセス頻度の偏りに着目した、ストレージでのデータアクセス頻度の監視と適切なデータ格納先階層判定および格納を行う、ストレージ階層仮想化機能が注目されている[1]。

大容量ストレージでは、高速 I／O 処理を実現するため、プロセッサやローカルメモリで構成される複数のノードが、共有メモリにアクセスして処理を実行する、並列アーキテクチャが採用されている場合が多い。そこで、本研究は、並列アーキテクチャ上でストレージ階層仮想化機能を実現する際に、高速にストレージ領域毎の移動先階層を判定するための一手法について述べ、その性能スケーラビリティを評価する。

2．背景と目的

ストレージ全体のＩ／Ｏスループット性能を向上するページの配置方法として、各階層の発揮できるＩ／Ｏスループット性能から、各階層間の閾値を算出し、ストレージ領域（以降ページと呼ぶ）毎のアクセス頻度（以降ではＩ／Ｏ数と呼ぶ）と閾値から適切な階層の配置を判定する方法がある[2]。具体的には、Ｉ／Ｏ数の高い順番にページを並べ、高頻度アクセスページから上位の階層に配置した場合に、階層の能力を超える飽和点であるＩ／Ｏ数を閾値として算出する（以降これを閾値算出処理と呼ぶ）。

日々変化するＩ／Ｏ数に追随して移動先階層を決定し、Ｉ／Ｏスループット性能を向上させるには、閾値算出処理を迅速に完了させることが不可欠である。しかし、並列アーキテクチャにおいて、一般に並列化できない処理部分が発生することにより、単純にスケーラブルに処理性能を向上させることは難しい。

そこで、本研究は、並列アーキテクチャを採用する大規模ストレージにおいて、高速かつスケーラブルな閾値算出処理方式を確立することを目的とする。

ノード数の増加に対してリニアに閾値算出処理の性能が向上し(O(n))、且つ容量の増大化に対して閾値算出処理時間がリニアにスケールすること(O(n))を目標とする。

3. 単純なソートによる実現方式とその課題

閾値算出処理の単純な実現方法として、ソートにより閾値を算出する方式を考える。並列アーキテクチャ上で高速なソートを実現する方式として、並列マージソート方式[3]が考えられる。

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

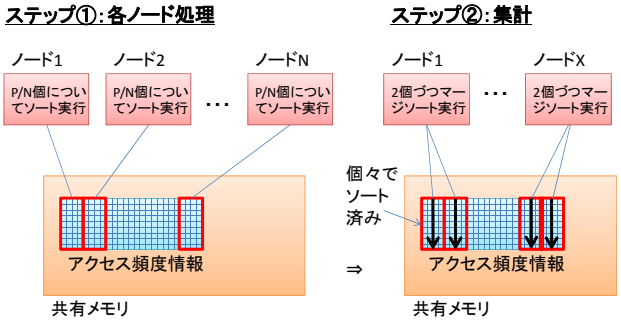

図 1 並列マージソート方式

図 1 に示すように、全ページを各ノードに対応するように分割し、各ノードでクイックソートにより並べ替えを実行し（ステップ①）、さらに特定のノードがソート済みのデータを元に、繰り返し 2 個のデータについてマージソートを実施していく（ステップ②）。最終的に作成されるソート済みのデータを元に、閾値を算出する。本方式は、O(n log n)の計算量であるため、閾値算出処理の性能がリニアにスケールしないという課題がある。

4．並列分布加算方式

閾値を決める際には、Ｉ／Ｏ数が上位のページから順にＩ／Ｏ数の累計値をもとめることができればよい。よって並列マージソート方式のような比較によるソートを実施しなくても、Ｉ／Ｏ数区間毎のページ数の分布を算出できればよい。

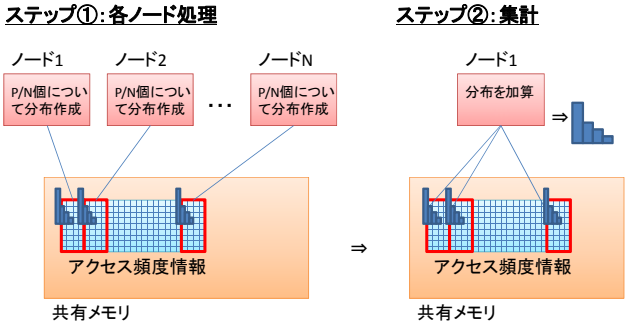

図 2 並列分布加算方式

そこで、非比較ソートであるバケットソート[4]のように、アクセス頻度の取りうる値を複数の区間に分割し、各区間について、属するページ数を数え上げたデータ（以降このデータを「分布」と呼ぶ）を各ノードで作成する。図 2 に示すように、ページを各ノードに対応するように分割し、各ノードで、集計した複数の分布の各区間のページ量をノード間で加算することにより、最終的にシステム全体の分布を作成する。そのシステム全体の分布から、ソート済みのデータを用いなくても、各階層の閾値を決定することができる。この方法を並列分布加算方式と呼ぶことにする。

5． 方式の評価

各方式の閾値算出処理の性能を評価する。共有メモリへのリードアクセスはレイテンシが長大化する。そこで、共有メモリリード回数で閾値算出処理の性能を簡易的に評価する。

5.1. 並列マージソート方式

システムページ数を P、ノード数を N とすると、各ノードでのページ単位のソートによる共有メモリへのリードアクセス回数は、クイックソート[5]を前提とすると、1 回の比較に 2 回のリードアクセスを実施すると仮定すると、

$$M_1 = 2 \cdot \left(1.386 \cdot \frac{P}{N} \cdot \log_2 \frac{P}{N}\right)$$

となる。

また、ノード間でのマージソートによる各ノードでのメモリへのリードアクセス最大回数は、

$$M_2 = P \cdot \left(1 + \frac{1}{2} + \frac{1}{4} + \cdots + \frac{1}{\frac{N}{2}}\right)$$

$$= P \cdot \left(2 - \frac{2}{N}\right) \qquad (N \text{ は } 2 \text{ の冪数})$$

となる。

以上を合計すると、

$$M_1 + M_2 = P \cdot \left(\frac{2.772}{N} \cdot \log_2 \frac{P}{N} + \left(2 - \frac{2}{N}\right)\right)$$

が閾値算出処理全体の共有メモリへのリードアクセス回数となる。

5.2. 並列分布加算方式

分布区間数を S とすると、各ノードでの分布の作成による共有メモリへのリードアクセス回数は、

$$B_1 = \frac{P}{N}$$

となる。

また、ノードが複数ある場合のノード間での分布のマージによる共有メモリへのリードアクセス回数は、加算元と先の分布で 2 回のリードアクセスを実施すると仮定すると、

$$B_2 = 2 \cdot S \cdot N \qquad (N \geq 2)$$

となる。

以上を合計すると、

$$B_1 + B_2 = \frac{P}{N} + 2 \cdot S \cdot N$$

が閾値算出処理全体の共有メモリへのリードアクセス回数となる。

5.3. スケーラビリティの評価

上述した 2 個の方式を、以下の条件で比較評価する。

表 1 評価の条件

#	項目	変数名	設定値
1	ノード数	N	可変にして評価
2	システムページ数	P	可変にして評価
3	分布区間数	S	455

分布区間の取り方によっては、区間のＩ／Ｏ数と実際のＩ／Ｏ数の誤差によるＩ／Ｏ最大スループット性能低下率(ε)の増大化が問題となるが、分布の区間として等比区間を取ることで、少ない区間数(S)で各階層のＩ／Ｏ数の総和を一定の誤差範囲に落ち着かせることができると推定している。Ｉ／Ｏ数の最大値をc_{max}とすると、εと S との関係は、

$$\log_{\varepsilon+1} c_{max} \geq S$$

と推定される。本評価ではＩ／Ｏ数を 1 ワードの非負整数で表現するとして、ε を 5%に収める場合を仮定し、S=455 で評価した。

図 3 は P を 1PB 相当のページ数(26M 個)と仮定した場合のノード数(N)に対する閾値算出処理性能のスケーラビリティを示している。

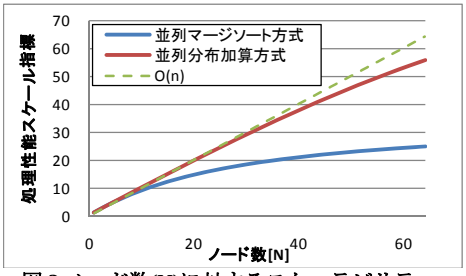

図 3 ノード数(N)に対するスケーラビリティ

縦軸の「処理性能スケール指標」は、各方式における、N=1 の時の閾値算出処理性能（$M_1 + M_2$または$B_1 + B_2$で算出される処理時間の逆数）を 1 としたときの、相対値を表している。並列マージソート方式がノード数の増加に対して頭打ちになるのに対して、並列分布加算方式はほぼ O(n)である。ただし、ノード間の分布の加算処理(B_2)がネックとなり、完全には O(n)とならない。

図 4 は N を 8 と仮定したときのシステムページ数(P)に対する処理時間のスケーラビリティを示している。縦軸の「処理時間スケール指標」は、各方式における、100TB 相当のページ数(2.5M)の場合を仮定した閾値算出処理時間（$M_1 + M_2$または$B_1 + B_2$で算出される処理時間）を 1 としたときの相対値を表している。

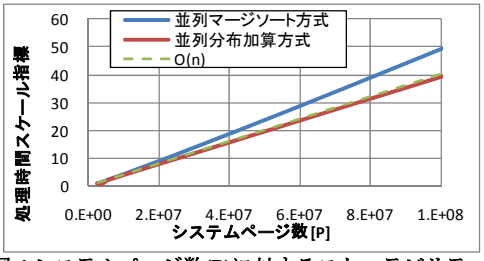

図 4 システムページ数(P)に対するスケーラビリティ

並列マージソート方式が O(n)から若干量外れていくのに対して、並列分布加算方式は P の増加に対してほぼ O(n)に収まる。

6. まとめ

並列アーキテクチャ上に実装されるストレージ階層仮想化機能において、高速に階層を判定するための一手法として、並列分布加算方式について述べ、その性能およびスケーラビリティを評価した。結果、並列分布加算方式の計算量は、目標としているノード数の増加とシステム記憶容量の増加に対して、それぞれほぼ O(n)に近づけることができ、現在の閾値算出処理にかかる時間をほぼ維持することが可能と考えられる。

しかし、ノード数の増加に対するスケーラビリティは完全には O(n)とならないため、さらなる分析と改善が今後の課題である。

---

[1] 須藤 梓,圷 弘明, 大平 良徳,江口 賢哲,山本政行," ストレージ階層仮想化機能の実現方式検討", 情報処理学会全国大会講演論文集 (2011).

[2] 大平 良徳,圷 弘明,須藤 梓,今崎 美保,江口 賢哲, “ストレージ階層仮想化機能におけるデータ配置先決定法の研究”,FIT2011 (to appear).

[3] Timothy J. Rolfe Eastern Washington University, Cheney, Washington, “A specimen of parallel programming: parallel merge sort implementation” (2010).

[4] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein , Introduction to Algorithms, second edition (2001).

[5] Jon L. Bentley and M. Douglas McIlroy , “Engineering a Sort Function”, Software - Practice and Experience, Vol. 23(11), 1249-1265 (1993).