C-001

ストレージの高性能化へ向けた汎用ハードウェアによるキャッシュメモリの設計

Design and Implementation of Cache Memory on Commodity Hardware for Storage Performance Improvement

坂下 悠貴† 工藤晋太郎† 揚妻 匡邦†
Yuki Sakashita Shintaro Kudo Masakuni Agetsuma

1. はじめに

高性能な汎用 CPU の普及に伴い、専用 LSI で実現していたストレージの高性能化を、汎用ハードウェアを用いて実現する技術が求められている。ストレージを高性能化する手段の 1 つにキャッシュメモリがある。高速な半導体メモリ上にデータをキャッシュする事で、ストレージの性能向上が可能となる。ストレージは信頼性向上のために複数のノードで構成されているため、ノード間でお互いのキャッシュへアクセス可能とする共有キャッシュメモリにより、キャッシュメモリを大容量化できる。本研究では、汎用ハードウェアを用いてストレージの共有キャッシュメモリを実現する方式を示し、性能とコストの観点で評価する。

2. 前提とするストレージアーキテクチャ

2.1 I/O 処理方式

各 CPU コアはポーリングで「I/O コマンドの到着」などのイベントをチェックする。この方式では複数の同じ処理を纏めて実行するため、処理の切り替えに伴う時間を削減し、CPU キャッシュヒット率を向上出来るため、スループット性能向上が期待出来る。

2.2 ノード間の接続方式

ノード間の通信路には、サーバ向け CPU に搭載されている汎用ハードウェアの Non Transparent Bridge（NTB）[1][2]を用いる。NTB で接続することで、互いのノードのメモリ空間に対して PCI Express 経由でアクセス可能となる。本研究では、2 ノード構成のストレージを前提とする。

2.3 他ノードのメモリアクセス方式

メモリコントローラが故障したノードに対し、コアが NTB 経由でメモリ・リードを発行すると、応答を待ち続けて別の処理を実行できず、そのコアはハングアップする。そこで、NTB 経由のメモリ・ライトだけで、他ノードのメモリ・リードとライトを実現する(図 1)。あるコア（以下 Client コア）が他ノードのメモリのデータをリードする場合、リード先のコア(以下 Server コア)のメモリにリード・リクエストを書く。Server コアはリード・リクエストに書かれたアドレスのデータをリードし、Client コアのノードのメモリに対してメモリ・ライトでデータを書き戻す。

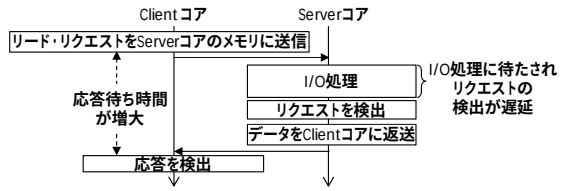

図 1 他ノードのメモリアクセス方式

3. メモリアクセス方式の課題

メモリ・リードを他ノードへリクエストするため、Server コアが I/O 処理中は応答出来ず、リクエストの検出が遅延する(図 1)。Client コアはその間に I/O 処理を実行出来ないため、システムのスループット性能が低下する。次章では、Server コアの応答時間を短縮し、スループット性能を向上する解決策を示す。

4. 解決策

本研究では、専用コア方式、論理専用コア方式、割込み方式の 3 つの解決策を示す。

4.1 専用コア方式

専用コア方式は、一部のコアをリクエストへの応答専用コア(以下、専用コア)として割り当てる(図 2)。専用コアは I/O 処理をしないため、短時間でリード・リクエストの応答ができる。ただし、専用コアで I/O 処理をしない分、システムのスループット性能が低下する。

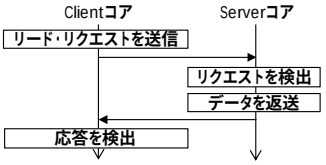

図 2 専用コア方式概要

4.2 論理専用コア方式

論理専用コア方式は、SMT(Simultaneous Multithreading)により 1 つの物理コアを 2 つの論理コアとして並列動作させ、一部の論理コアを論理専用コアとして割り当てる(図 3)。I/O 処理をしない専用コアに論理コアを割り当てる事で、専用コア方式よりもスループット性能の低下を抑止出来る。ただし、本方式を採用するためには SMT サポートの CPU の使用が前提となる。

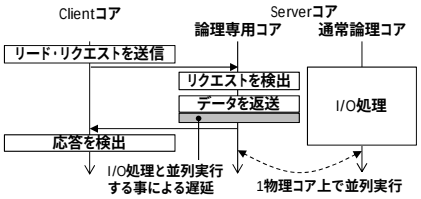

図 3 論理専用コア方式概要

4.3 割込み方式

割込み方式は、リクエスト・メッセージ送信後に Server コアへ割込みを発行し、I/O 処理に割込んで応答をリクエストする(図 4)。割込み方式では、特定のコアだけが応答するのではなく、すべてのコアがリクエストに対して応答

する。ただし、2.1 節に記したような処理の切り替えを伴うため CPU 処理時間をロスする。

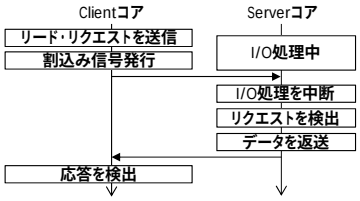

図 4 割込み方式概要

5. 評価

5.1 スループット性能のモデル式

各方式のスループット性能のモデル式を定義する。解決策の効果を評価するため、最初に解決策適用前のスループット p_0 の式を定義する。Client コアから Server コアにリード・リクエストを送信した時、Server コアは I/O 処理中のため、t を I/O の CPU 処理時間とすると、リード・リクエスト検出までの平均待ち時間 w_1 は 2/t となる。s を Server コアの処理時間、c_a をシステムのコア数とすると、p_0 は、

$$p_0 = c_a/(t + t/2 + s) \quad (0)$$

専用コア方式では、複数コアから 1 コアに対してリクエストが集中し、待ち時間 w_1 が生じる。これを待ち行列理論(M/M/1)[3]を用いて計算する。c_s を専用コア数とすると、平均到着時間 a および、平均待ち時間 w_1 は以下のように計算出来る。

$$a = t/(c_a/c_s - 1) \quad (1)$$

$$w_1 = \rho/(1-\rho)\cdot s \qquad \text{Q } \rho = s/a \quad (2)$$

以上より、専用コア方式のスループット p_1 は、

$$p_1 = (c_a - c_s)/(t + s + w_1) \text{ ただし } c_a \geq 2c_s \quad (3)$$

論理専用コア方式では、SMT 適用によるコアあたりのスループット向上率を 1.21 とする[4]。2 つの論理コアが並列動作する事で Server コアの処理が遅延する分を d とする。また、専用コア方式と同様に 1 コアにリクエストが集中して生じる待ち時間を w_2 とすると、c_a を論理コア数、c_s を論理専用コア数として扱い(1)(2)で w_2 を算出出来る。よって、論理専用コア方式のスループット p_2 は、

$$p_2 = 1.21(c_a - c_s/2)/(t + (s+d) + w_2) \quad (4)$$

ただし $c_a \geq c_s$

割込み方式では、Server コアが I/O 処理からリクエストへの応答処理に切り替わる事で CPU 処理時間をロスする。I/O 処理から応答処理への切り替え時間を i_1、応答処理から I/O 処理への切り替え時間を i_2 とすると、割込み方式のスループット p_3 は、

$$p_3 = c/(t + s + 2i_1 + i_2) \quad (5)$$

5.2 評価方法

モデル式を用いてスループットを算出する。表 1 に評価条件を示す。ストレージは規模に応じてコア数が異なるため、コア数毎にスループットを算出する。また I/O の種類によって処理時間が異なるため、各 I/O 処理時間毎にスループットを算出する。

表 1 評価条件

項目	値
システムのコア数	4, 8, 16, 32
I/O処理時間[us]	10, 20, 40, 80, 160

6. 評価結果

解決策適用前のスループット p_0 を 100%として、各方式のスループットの比較結果を図 5 に示す。システムのコア数 4 のケースでは、いずれの I/O 処理時間においても割込み方式のスループット p_3 が最大だった。これはコア数の少ない構成では I/O 処理をしない専用コアの占める割合が大きい分、スループット低下率が大きいためである。コア数が 8, 16, 32 のケースでは、I/O 処理時間 20us 以上では論理専用コア方式のスループットが最大で SMT 適用でスループットが向上した効果と考えられる。しかし I/O 処理時間が 10us では、割込み方式のスループットの方が高く、これはリクエストが集中して専用コアがボトルネックになったためである。

今回の評価では 8 コア以上では論理専用コア方式のスループット性能が最大となった。ただし本方式は SMT サポートの CPU の使用が前提で、ストレージ装置の価格が増える。よって、求めるスループット性能と許容されるコストに応じて採用する方式を選択するべきである。

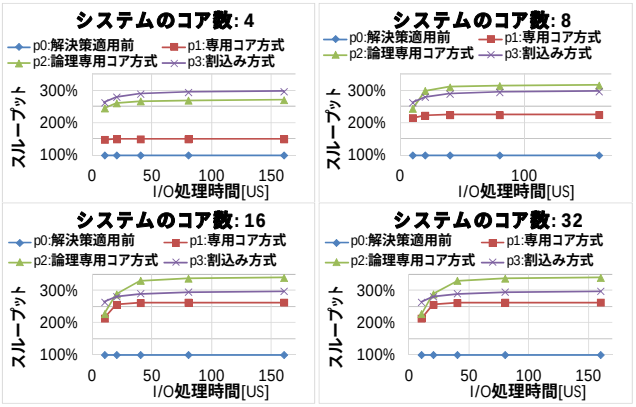

図 5 各方式のスループット比較

7. まとめ

汎用ハードウェアを用いた共有キャッシュメモリの実現方式について、モデル化してスループット性能を評価し、コア数と I/O 処理時間の条件及びコストに応じて最適な方式を選択するための方法を提案した。これにより、いずれの条件においてもスループット性能を 263%以上に向上出来る見通しを得た。

参考文献

[1] Intel, “Intel Xeon Processor E5-1600/E5-2600/E5-4600 v2 Product Families”, Intel Datasheet (2014).
[2] Intel, “Intel Xeon Processor E5-1600, E5-2600, and E5-4600 v3 Product Families”, Intel Datasheet (2015).
[3] 吉岡良雄, “待ち行列と確率分布”, ISBN-10: 4627828217, (2004).
[4] Deborah T. Marr et al Intel Corp., ”Hyper-Threading Technology Architecture and Microarchitecture”, Intel Technology Journal Q1, (2002).

†(株) 日立製作所 情報通信イノベーションセンタ, Center for Technology Innovation, Hitachi Ltd.