ソフトウェア制御型 SSD による
大容量インメモリデータベースの高速化

桑村 慎哉 1　五木田 駿 1　柴田 清己1　風間 哲 1　吉田 英司 1　小川 淳二 1

1. はじめに

近年、大量データを高速に処理可能なインメモリデータベース(インメモリ DB)が注目されている。これはすべてのデータをメインメモリに格納することで高速処理を実現しているため、データ処理量に応じた DRAM 容量を必要とする。1 台のサーバに搭載可能な DRAM 容量には限度があるため、大規模なデータを処理する用途では、DRAM に比べて低コストかつ大容量な SSD を用いてメモリ空間を拡張する研究が進められている[1]。しかし、従来の SSD は FTL(Flash Translation Layer)により内部の動作が隠蔽されているため、データベースが NAND フラッシュメモリの性能を最大限に引き出せないという課題があった。

本論文では、サーバのソフトウェアから NAND フラッシュメモリを直接制御可能なソフトウェア制御型 SSD を用い、制御ソフトウェアをインメモリ DB に最適化することで高速化する技術を提案する。

2. 開発したソフトウェア制御型 SSD

ソフトウェア制御型 SSD とは、FTL を SSD 内部からソフトウェア層に移動し、ソフトウェアが個々の NAND フラッシュメモリへのデータ配置や読出し方法を制御できる SSD である[2,3,4]。我々が今回開発した SSD は、NAND フラッシュメモリチップごとにコマンドキューを搭載し、完了通知をホストのメインメモリへの DMA 書き込みとソフトウェアのポーリングで行うことにより、搭載された 256 個のチップの同時並列動作を可能とする[2]。サーバとは PCI Express Gen3 x8 で接続しており、ハードウェアの実測スループットは Read 5.5GB/s・Write 4.3 GB/s に達する。

3. SSD を利用したメインメモリ拡張技術

インメモリ DB のメモリ拡張技術として hybrid-memory[1]を利用した。これはメモリ空間を、ページングしない速いメモリ領域と、SSD にページングする遅いメモリ領域に分割する。DB がこの 2 つの領域を使い分けることにより、性能を落とさずメモリ容量を拡張できる。なお、SSD にページングするメモリ領域の確保は、専用のメモリ割り当て関数により実現している。

3.1 SSD へのアクセスの特徴

Hybrid-memory では、SSD へのアクセスはページングのみで発生する。ページインで発生する I/O 要求は同期読み出しである。すなわち、ページフォールトを発生させたスレッドの処理は I/O 要求が完了するまで中断される。そのため、I/O 要求の遅延時間がそのままスレッドの実行時間に加算される。I/O 要求のサイズはメモリページサイズに依存し、x86 系の Linux では通常 4KB のため、4KB の同期読み出しの遅延時間が重要となる。なお、シーケンシャルにアクセスする場合でも、ページフォールトは 1 ページごとに発生するため、4KB 以外のサイズの I/O 要求は発生しない。

一方、ページアウトは、ページインと異なり同期は不要である。複数の連続したページをまとめて一度にページアウトすることができるため、シーケンシャル書き込みのスループットが重要となる。

3.2 開発した制御ソフトウェア

3.2.1 ランダム読み出し

4KB のデータを NAND フラッシュメモリから読み出す時間は、ソフトウェアやコントローラによるオーバーヘッドがなければ一定値である。I/O 処理専用のスレッドを用意する実装ではコンテキストス

1 (株)富士通研究所
Fujitsu Laboratories Ltd.

イッチによる遅延が発生する可能性があるため、今回はページフォールトを起こしたスレッドがコンテキストを切り替えることなく、I/O 要求の発行から完了待ちのポーリングを行う実装とした。これにより、ソフトウェアによる遅延の増加を最小限に抑えた。

3.2.2 アクセスパターンに応じた先読み機能

開発した SSD の NAND フラッシュメモリのページサイズは 16KB であり、4KB より大きい。そこで、シーケンシャル読み出しでは、NAND フラッシュメモリから 16KB 単位で読み出してキャッシュとしてメインメモリに格納し、後続の I/O 要求に対してキャッシュから転送することで遅延を短縮した。さらに、アクセスパターンを解析することで、連続する I/O 要求を予測可能とした。そして、予測に基づき、事前に複数の NAND フラッシュメモリから並列に読み出してメインメモリに格納する先読み機能を実装した。これにより、シーケンシャル読み出しなどの特徴的なアクセスを高速化する。

4. 性能評価

4.1 4KB 読出し遅延評価

基礎的な性能として 4KB ランダム読み出し遅延時間を測定した。図 1 は遅延時間の累積分布を示しており、例えば、従来型 SSD は 80%の読み出し要求が 150μs 以内に完了することを示している。比較のため、従来型の NVMe-SSD の測定結果も示す。従来型 SSD に対し、提案手法は遅延が短くばらつきもほぼないことが分かる。これは、遅延を最小化したソフトウェアから NAND フラッシュメモリを直接制御することで、FTL やコンテキストスイッチなどのオーバーヘッドを削減できたためである。

4.2 インメモリ DB マイクロベンチマーク評価

インメモリ DB のワークロードを模したマイクロベンチマークにより評価を行った。ワークロードはランダムアクセスとシーケンシャルスキャンを想定した。前者はインデックスつきテーブルへのクエリ、後者はインデックスなしテーブルへのクエリに相当する。ページングしないメモリ領域と SSD にページングするメモリ領域の比率を 1:4 に設定し、各操作にかかる実行時間を計測した。ランダムアクセス結果は、従来型 SSD に対し 1.4 倍高速化した。これは、主に4.1節で示した遅延時間の差によるものである。一方、スキャン結果は従来型 SSD に対し約 3 倍高速化した。これは、先読み機能により、NAND フラッシュメモリの読み出し遅延を削減できたことが主要因である。

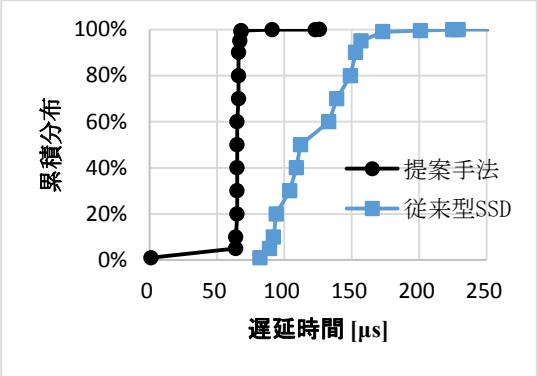

図 1 4KB ランダム読み出し遅延

5. まとめ

ソフトウェア制御型 SSD と、インメモリ DB のメモリ拡張技術に最適化した制御ソフトウェアを開発した。インメモリ DB のマイクロベンチマークで評価した結果、従来型の NVMe-SSD と比較して約 3 倍の高速化を実現した。

参 考 文 献

[1] B. Höppner, et al., “An Approach for Hybrid-Memory Scaling Columnar In-Memory Databases,” in Proc. Int. Workshop Accelerating Data Mana. Syst. Using Modern Processor Storage Archit., pp. 64–73, Sept. 2014.

[2] 風間 哲他, “高並列アクセスを可能にする高性能ソフトウェア定義型 SSD の開発,” CPSY 研究会, Oct. 2015.

[3] J. Ouyang, S. Lin, S. Jiang, Z. Hou, Y. Wang, and Y. Wang, “SDF: Software-defined flash for web-scale internet storage systems,” In Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS ’14), pp. 471–484, Salt Lake City, USA, March 2014.

[4] P. Wang, G. Sun, S. Jiang, J. Ouyang, S. Lin, C. Zhang, and J. Cong, “An efficient design and implementation of LSM-tree based key-value store on open-channel SSD,” In Proceedings of the Ninth European Conference on Computer Systems (EuroSys '14), New York, USA, April 2014.