社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

[招待講演] ビックデータ利活用のための計算基盤

松谷　宏紀†,††,†††

† 慶應義塾大学 理工学部　〒 223-8522 神奈川県横浜市港北区日吉 3-14-1
†† 科学技術新興機構さきがけ
††† 国立情報学研究所

あらまし　ビッグデータ利活用のための基盤技術として構造型ストレージ（NoSQL）や機械学習、分散処理フレームワークなどが挙げられる。我々はこれらを高速化、高効率化するために 10GbE インタフェースを有する FPGA NIC や PCI-Express over 10GbE による GPU クラスタを用いた研究をしている。本発表ではこれらの取り組みについて紹介する。

キーワード　ビッグデータ、FPGA、GPU、データベース、機械学習、分散処理

[Invited Talk] Computer Infrastructure for Utilizing Big Data

Hiroki MATSUTANI†,††,†††

† Faculty of Science and Technology, Keio University 3-14-1, Hiyoshi,Yokohama, JAPAN 223-8522
†† PRESTO, Japan Science and Technology Agency
††† National Institute of Informatics

Abstract　Structured storages (NoSQLs), machine learning, distributed processing frameworks are fundamental technologies for utilizing big data. This talk introduces our recent work that focuses on acceleration of these key technologies by introducing FPGA NIC that has 10GbE interfaces and networked GPU cluster based on PCI-Express over 10GbE.

Key words　Big data, FPGA, GPU, database, machine learning, distributed processing

1. はじめに

現在、ありとあらゆる分野においてビッグデータ処理技術と機械学習ベースの AI 技術が重要になっているが、これらを活かすには本来強力な計算リソースが必要となることが多い。そこで、本研究では、ビッグデータ処理や機械学習に関するオープンソースプロダクトのフレームワークを維持しつつ、FPGA や GPU のようなアクセラレータを活用することで、ビッグデータ処理と機械学習技術の低消費電力化および低コスト化を狙う。

まず、本研究が対象とするビッグデータ処理および機械学習技術を紹介する。ここではビッグデータ利活用のための統合システムの一例として Lambda アーキテクチャ [1] から着想を得たシステムを紹介する。図 1 上部に処理の流れを示す。図の左端から入力されたストリームデータはストリーム処理やバッチ処理によって加工もしくは集計され、データベースに蓄積されている。この流れを以下のとおり 3 つに分けて説明する。

- データの収集：ネットワークサービスや IoT（Internet of Things）デバイスによって生成されたビッグデータを集め、データベース（Realtime View）を更新する処理である。データの受信と配送にはメッセージキューイングシステム（例えば Apache Kafka）、データベースの更新処理にはストリーム処理フレームワーク（例えば Apache SparkStreaming、Apache Storm）が使用できる。外れ値検出や変化点検出のようなオンライン機械学習も行われる。
- データの集約：1 日に数回など定期的に全データを解析し、集計結果をデータベース（Batch View）に格納する。このためにバッチ処理フレームワーク（例えば Apache Hadoop、Apache Spark）および機械学習フレームワーク（例えば Apache Mahout、Spark 付属の MLlib や GraphX）などが使用できる。また、分散処理のために計算機間でデータを交信する際にはシリアライゼーション処理（例えば Apache Thrift などが利用可能）が必要になる。
- データの蓄積と検索：上述の Realtime View と Batch View を格納するデータベースがこれに相当する。Realtime View には時々刻々と更新される直近のデータ、Batch View には定期的に集計される過去の集計済み全データが格納される。ユーザからの問い合わせに対し、Realtime View と Batch View の検索結果を結合したものを返答することで全データ性および

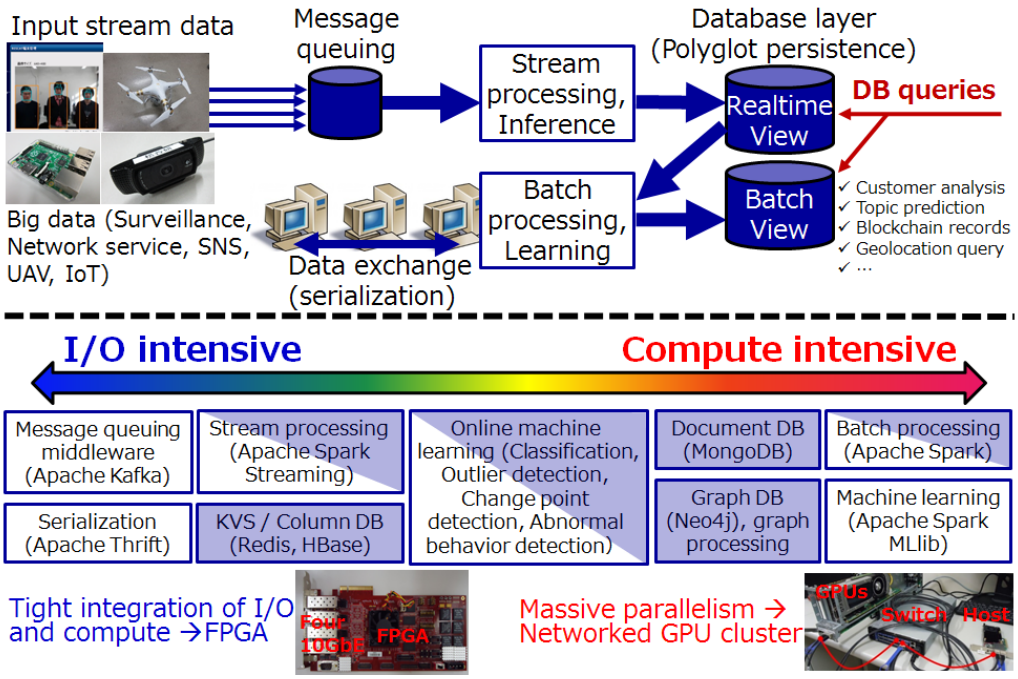

図 1　Lambda アーキテクチャにおける要素技術 [1] と最近の研究内容 [2]〜[11]

Fig. 1　Fundamental technologies in Lambda architecture [1] and our recent work [2]〜[11]

即応性の両立を図る。

2.　アクセラレーション戦略

上述の処理を計算強度という指標で分類する。ここでは、計算と I/O（ネットワークやメモリアクセス）の大雑把な比率を計算強度とする。筆者の主観で分類した結果を図 1 下部に示す (注1)。図の右端は計算負荷が高く、図の左端は I/O 負荷が高いものとしている。例えば、ストリーム処理はネットワーク処理に比して計算は軽めであるが、バッチ処理では計算強度が高くなる。

次に、アクセラレーションのためのソリューション（FPGA、GPU など）について考える。GPU は、数千にも上る計算コアを用いた高い並列処理が可能であるが、GPU 外部とのデータのやり取りがボトルネックになることがある。FPGA は、Look-Up Table（LUT）ベースのプログラマブル素子によって演算が実現されるため単位面積当たりの計算性能は GPU に劣るものの、ネットワークやメモリ、ストレージなどと直結できるため I/O と密に結合した演算が可能である。

以降の 2.1 節、2.2 節、2.3 節ではデータの蓄積と検索、データの収集、データの集約に関するアクセラレーション手法を紹介する。

2.1　データ検索のアクセラレーション

ビッグデータ向けのデータベースとしてここではポリグロット永続化を想定する。ポリグロット永続化とは「多様なデータベース」という意味である。これは「ビッグデータを扱うにはスケーラビリティの高いデータベースが必須であるが、スケーラビリティを高めると今度は汎用性が犠牲となり、特定用途特化型のデータベースとってなってしまう。このような特定用途特化型データベースを使って豊かなネットワークサービスを構築するには、複数の特定用途特化型データベースを相補的に使う必要がある」という考察を基にしている。具体的には、キーバリューストア（KVS）型、カラム指向型、ドキュメント指向型、グラフ型データベースなどを相補的に、つまり適材適所で利用することを想定している。これまでに FPGA や GPU を用いた各種データベースの高速化を研究してきた [2]〜[7]。

KVS では、データをキーとバリューの組として扱う非常にシンプルなデータベースである。計算負荷が低いため、ネットワーク処理がボトルネックになりやすい。そこで、ネットワーク処理と計算を密結合できるデバイスとして FPGA ベースのネットワークインタフェース（FPGA NIC と呼ぶ）を用いた KVS のアクセラレーションを研究してきた。文献 [5] では 10GbE インタフェースを 4 個有する NetFPGA-10G ボードを用いて KVS の一種である Redis のハードウェアキャッシュを実現した。ただし、FPGA ボードに搭載できる DRAM 容量は大きくはないため、文献 [5] では FPGA ベースのハードウェア KVS キャッシュに加え、Linux カーネル内にソフトウェアベースの KVS キャッシュを設けることを提案した。我々は前者を L1 NoSQL キャッシュ、後者を L2 NoSQL キャッシュと呼び、要求されたデータが L1 NoSQL キャッシュに存在しない場合は L2 NoSQL キャッシュをルックアップし、L2 NoSQL キャッシュにもヒットしない場合はアプリケーション層で動作する Redis に問い合わせが行くようにした。

カラム指向型では、各行データは複数個のカラムから構成され、行キーを基にソートされた状態で扱われる。このため

(注1)：この分類は人によって意見が分かれるところである。あくまでも参考程度に扱って欲しい。

HBase では startRow と stopRow を用いた範囲問い合わせも可能である。カラム指向型も KVS 同様、ネットワーク処理がボトルネックとなりやすいため、我々は HBase を対象に FPGA NIC を用いたアクセラレーション手法 [7] や Linux カーネル内キャッシュを用いたアクセラレーション手法 [6] を提案してきた。

ドキュメント指向型ではデータをドキュメントの集合、グラフ型データベースではデータをグラフとして扱う。前者では正規表現ベースの文字列探索、後者ではグラフ探索を伴う問い合わせが生じる。これらの処理は一般的に計算負荷が高いため、単一 GPU を用いたドキュメント指向型データベース MongoDB の高速化 [3]、グラフ型データベース Neo4j の高速化 [2] を提案してきた。ただし、データベースで扱うデータに比して GPU のデバイスメモリはあまりにも小さいため、GPU とデータベース間でデータ交信が頻発し、性能の新たなボトルネックとなっていた。そこで、GPU 内にキャッシュできるデータ容量をスケーラブルに増強するために、我々は PCI-Express over 10GbE 技術によってネットワーク接続された GPU クラスタを用いるアプローチを提案している [4]。文献 [4] では多数の GPU のデバイスメモリを分散共有メモリとして扱い、分散ハッシュテーブルから着想を得たハッシュ技法によってデータを各 GPU のデバイスメモリに分散させている。

2.2 データ収集のアクセラレーション

主要な要素技術はストリーム処理フレームワークとオンライン機械学習である。これらの処理は総じて計算負荷は低めであり、ネットワーク処理と計算を密結合できる FPGA NIC を用いたアプローチが有利であると考えている [8]〜[10]。

オンライン機械学習処理の代表例として異常検出が挙げられる。異常検出はさらに外れ値検出、変化点検出、異常行動検出などに分類できる。例えば、ネットワークから流れてくるサンプルデータに対し、通常とは異なる値、傾向、振る舞いなどを検出するために利用できる。我々はこのうち外れ値検出を FPGA NIC で高スループット化する研究を行ってきた [8], [9]。文献 [8] では、マハラノビス距離を用いて外れ値を検出する手法を FPGA NIC 上に実現し、10GbE ラインレートの 95.8%のスループットを実現した。文献 [9] では、k-Nearest Neighbor（kNN）や Local Outlier Factor（LOF）アルゴリズムを FPGA NIC 上に実現する方法を提案している。LOF による外れ値検出では、過去のサンプルデータと入力サンプルデータの比較が必要だが、当然、FPGA NIC の限られた DRAM に過去の全サンプルデータを保持することはできない。そこで、最近アクセスされた過去のサンプルデータクラスタのみを FPGA NIC にキャッシュしておき、一方で、FPGA NIC にキャッシュされている限られた情報だけでは外れ値かどうか判別できないような入力データについてはアプリケーション層にて完全な LOF 処理を行うアプローチを提案している。

ストリーム処理フレームワークに関しては、まず、One-at-a-time 方式と Micro-batch 方式に大別できる [1]。One-at-a-time 方式ではデータ要素 1 つ 1 つに対し決められた処理を適用するのに対し、Micro-batch 方式では短い時間間隔の間に到着したデータ要素をまとめて 1 つの Micro-batch とし、この Micro-batch 毎に決められた処理を適用する。前者の実例として Storm、後者の実例として SparkStreaming が挙げられる。通常のソフトウェア処理の場合、高機能な処理を One-at-a-time 方式で実現しようとすると計算負荷が高くなり過ぎる。一方、Micro-batch 方式では Micro-batch のサイズに応じて外れ値や変化点などのイベントを検出するまでの遅延が増大してしまう。そこで、我々は One-at-a-time 処理と Micro-batch 処理を組み合わせる 2 段階ストリーム処理を研究している。具体的には、Micro-batch 方式の SparkStreaming に対し、FPGA NIC 上に実現した One-at-a-time 処理を組み込むアプローチを提案している [10]。

上記に加え、現在では変化点検出アルゴリズムの FPGA NIC 上での実現、メッセージキューイングミドルウェアや各種シリアライゼーションプロトコルの FPGA NIC を用いた高スループット化に関する研究課題にも取り組んでいる。

2.3 データ集約のアクセラレーション

バッチ処理フレームワークとして我々は Hadoop/MapReduce に加え Spark にも注目している。Spark では RDD（Resilient Distributed Dataset）と呼ばれる分散共有メモリ上にデータを保持し、RDD から別の RDD を生成する処理（Transformation）や RDD を集約する処理（Action）が行われる。

バッチ処理は計算負荷が高くなりやすく、GPU によるアクセラレーションが向くことが多い。文献 [11] では Spark の RDD に対する Transformation 処理や Action 処理を GPU にオフロードしているが、これだけでは 2.1 節で述べたとおり、GPU と Spark の間のデータ転送が新たなボトルネックとなる可能性がある。そこで、文献 [11] では文献 [4] と同じアプローチを採用している。具体的には、PCI-Express over 10GbE 技術によってネットワーク接続された GPU クラスタを前提に、GPU のデバイスメモリに RDD をキャッシュしておく。GPU のネットワークトポロジによっては CPU から近い GPU、遠い GPU が生じてしまうが、文献 [11] では RDD の系統グラフを基に頻繁にアクセスされる RDD は近い GPU、それ以外は遠い GPU にキャッシュするアイディアを提案している。

3. アプリケーション

上述の要素技術は様々なアプリケーションのアクセラレーションに活用できる。例えば、FPGA NIC による KVS は DDoS Mitigator [12]、ビットコインの基盤にもなっている分散合意手法ブロックチェーンのキャッシング [13] にも応用できる。KVS コアについては SOTB 65nm プロセスによる実チップ化も行った。また、ネットワーク接続された GPU クラスタ技術は VR（Virtual Reality）アプリケーション [14]、FPGA ベース 10GbE スイッチはディープラーニングにおける集約演算 [15] や MapReduce の遅延タスクの代理応答 [16] にも応用できる。

謝辞 本研究の一部は、JST 戦略的創造推進事業さきがけ、総務省 SCOPE（152103004）、セコム科学技術振興財団の補助による。本研究は研究室の学生たちによって達成されたものである。ここで改めて感謝の意を示す。

文　献

[1] Nathan Marz and James Warren, Big Data: Principles and Best Practices of Scalable Real-Time Data Systems, Manning Publications, 2015.

[2] S. Morishima and H. Matsutani, “Performance Evaluations of Graph Database using CUDA and OpenMP-Compatible Libraries,” ACM SIGARCH Computer Architecture News, vol.42, no.4, pp.75–80, Sept. 2014.

[3] S. Morishima and H. Matsutani, “Performance Evaluations of Document-Oriented Databases using GPU and Cache Structure,” Proceedings of the International Symposium on Parallel and Distributed Processing with Applications (ISPA’15), pp.108–115, Aug. 2015.

[4] S. Morishima and H. Matsutani, “Distributed In-GPU Data Cache for Document-Oriented Data Store via PCIe over 10Gbit Ethernet,” Proceedings of the International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Platforms (HeteroPar’16), Aug. 2016.

[5] Y. Tokusashi and H. Matsutani, “A Multilevel NOSQL Cache Design Combining In-NIC and In-Kernel Caches,” Proceedings of the International Symposium on High Performance Interconnects (Hot Interconnects 24), pp.60–67, Aug. 2016.

[6] K. Tamura and H. Matsutani, “An In-Kernel NOSQL Cache for Range Queries Using FPGA NIC,” Proceedings of the International Conference on FPGA Reconfiguration for General-Purpose Computing (FPGA4GPC’16), pp.13–18, May 2016.

[7] A. Hamada and H. Matsutani, “Design and Implementation of Hardware Cache Mechanism and NIC for Column-Oriented Databases,” Proceedings of the International Conference on Reconfigurable Computing and FPGAs (ReConFig’16), Nov. 2016.

[8] A. Hayashi, Y. Tokusashi, and H. Matsutani, “A Line Rate Outlier Filtering FPGA NIC using 10GbE Interface,” ACM SIGARCH Computer Architecture News, vol.43, no.4, pp.22–27, Sept. 2015.

[9] A. Hayashi and H. Matsutani, “An FPGA-Based In-NIC Cache Approach for Lazy Learning Outlier Filtering,” Proceedings of the International Conference on Parallel, Distributed, and Network-Based Processing (PDP’17), March 2017.

[10] K. Nakamura, A. Hayashi, and H. Matsutani, “An FPGA-Based Low-Latency Network Processing for Spark Streaming,” Proceedings of the Workshop on Real-Time and Stream Analytics in Big Data (IEEE BigData 2016 Workshop), Dec. 2016.

[11] Y. Ohno, S. Morishima, and H. Matsutani, “Accelerating Spark RDD Operations with Local and Remote GPU Devices,” Proceedings of the International Conference on Parallel and Distributed Systems (ICPADS’16), Dec. 2016.

[12] Y. Tokusashi, R. Nakamura, H. Tazaki, and H. Matsutani, “mitiKV: An Inline Mitigator for DDoS Flooding Attacks,” Internet Conference (IC’16), Oct. 2016.

[13] 榊原優真，中村幸平，松谷宏紀，“FPGA NIC を用いたブロックチェーン分散型台帳のキャッシング，” 電子情報通信学会技術研究報告 CPSY 研究会，Jan. 2017.

[14] 森島　信，岡崎真博，松谷宏紀，“VR アプリケーションのためのリモート GPU 割り当ての検討，” 電子情報通信学会技術研究報告 CPSY 研究会，Jan. 2017.

[15] 竹本一馬，林　愛美，森島　信，松谷宏紀，“GPU の計算結果を集約する FPGA スイッチの検討，” 電子情報通信学会技術研究報告 CPSY 研究会，Jan. 2017.

[16] 三塚皐矢，林　愛美，松谷宏紀，“10GbE FPGA スイッチによる MapReduce 遅延タスクの代理応答，” 電子情報通信学会技術研究報告 CPSY 研究会，Jan. 2017.