ディスアグリゲーテッドコンピュータに向けたメモリセントリックアーキテクチャ

ディスアグリゲーテッドコンピュータは，一見ハードウェア検討の割合が大半を占めるようにみえます．しかし，仮にハードウェア資源が物理的にフラットに光で接続された世界が実現されたとしても，既存のソフトウェアを用いるだけでは，効率的なコンピューティングを実現することはできません．本稿では，CPUを中心として発展してきた既存のソフトウェアの問題と，ディスアグリゲーテッドコンピュータ実現に向けた新たなデータ交換モデルとして，メモリセントリックアーキテクチャについて紹介します．

石崎　晃朗（いしざき　てるあき）　山部　芳朗（やまべ　よしろう）
NTTソフトウェアイノベーションセンタ

メモリセントリックアーキテクチャとは

ムーアの法則と呼ばれる，1965年に発表された半導体集積率が18カ月で倍になるという経験則にあるように，CPU（Central Processing Unit）性能は急速に進化してきました．このCPUの進化と比較して，メモリやストレージ・ネットワークは緩やかな進化をしています．こうした背景から，現在のソフトウェアの多数は「高性能なCPUとその他の低速なデバイス」という前提に立って，可能な限りCPUで演算を行うことで他のデバイス処理時間を短縮するという方針のもとに設計されてきています．

例えば，ストレージI/O（Input/Output）については，メモリ上に蓄積されるI/O対象のデータに対してCPUでI/O単位やリクエスト処理量の調整を行うことで，効率的なI/O処理を実現しています．また，複数のアクセラレータをまたいだ処理についても，処理の仲介をCPUが行うソフトウェア処理モデルを取っています．このようなCPUが処理に介在する処理モデルを，CPUセントリックなコンピューティングモデルと呼びます．

このように，一般的なソフトウェア設計として，CPUが処理の仲介を行うようにつくられる一方で，近年はムーアの法則の限界といわれるように，CPUコアの性能の伸びが鈍化してきています．また，高速ストレージである不揮発性メモリや，FPGA（Field Programmable Gate Array）・GPU（Graphics Processing Unit）などのアクセラレータが急速に進化しており，多様なアクセラレータの性能を引き出すための新たなソフトウェア処理モデルが今後より重要となりつつあります．

そこで，私たちは多種多様なハードウェアを組み合わせたソフトウェア処理モデルを検討するにあたり，アクセラレータが処理を開始する場所としてのメインメモリに着目し，アクセラレータ連携に際してのデータ交換についてメモリを介して行う，メモリセントリックアーキテクチャ（メモリセントリック）について検討しています．

メモリセントリックで想定する基本的な動作は，データ送信元の演算器がメインメモリ（共有メモリ）へデータ配置を行うことをきっかけとして，データ受信側の演算器が自律的にデータ取得および演算を行う，メモリを中心としたデータ交換モデルです．例えば，他ノードからNIC（Network Interface Card）を介して受信したデータについてFPGAで処理することを想定した場合，従来はCPUがNICの受信制御やFPGAへの処理実行の制御を行っていたところを，本来のネットワーク受信処理と演算処理を担うNICとFPGAの処理のみで完結するため，処理を仲介するためのCPUリソース消費を抑えるとともにCPU処理にかかわる遅延時間削減が期待できます（**図1**）．

また，メモリ上に展開されたデータを基に，複数の演算器が並列処理を行うような場合に，データ送信処理は共有メモリ領域へのデータ配置1回のみ

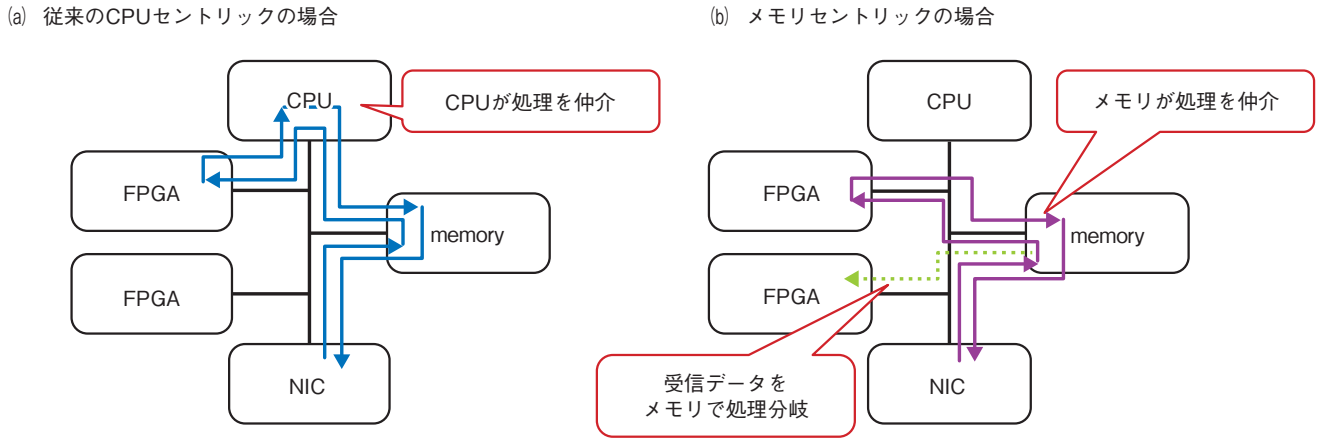

図1　CPUセントリックとメモリセントリック

で済むことから，より効率的な処理モデルとなっています（図1）．

メモリセントリックに近いコンセプトとしては，Hewlett Packard Enterprise（HPE）社のMemory Driven Computing（MDC）があります[1]．MDCはすべてのプロセッサの中心に大規模なメモリプールを配置した構成をとるといった，ハードウェアアーキテクチャを起点としたコンセプトモデルです．これに対して，メモリセントリックでは処理実行時のプロセッサ間のデータ交換にメモリを介して行うための，ソフトウェアによるデータ制御に注目したモデルです．

メモリセントリックアーキテクチャの有効性評価

メモリセントリックのデータ交換モデルのコンセプトの有効性評価にあたり，共有メモリを介した多種多様なプロセッサ間での効率的なデータ交換を行うコンセプトに近い既存ソフトウェアとして，分散データ処理のオープンソースソフトウェアであるApache Sparkの拡張機能であるSparkle[2], [3]を用いて評価を実施しました．

Apache Sparkは各Workerプロセスでデータ処理を実施するMapフェーズ，Workerプロセスで実行した処理結果について必要とするプロセスに配布するShuffleフェーズ，各Workerプロセスから集めた結果を基に演算を実行するReduceフェーズから構成されています．この中で，Shuffleフェーズは複数ノードに分散したWorkerプロセス間でTCP/IPを利用したネットワーク通信で大量のデータ交換を行う処理であり，ネットワーク通信にかかわるCPU処理の遅延時間やCPUリソース消費の観点から非常に処理コストが大きいことが知られています．この問題を解決することを目的とした，共有メモリを介したShuffle処理を可能とする拡張機能がSparkleです．SparkleはHPE社がMDCのコンセプト向けに開発したソフトウェアであり，オープンソースとして公開されていますが，評価検討を着手した際には2016年時点で開発が止まっていたため，評価にあたっていくつかのバグ修正を行っています[4]．

図2は1台のサーバ内に複数のWorkerプロセスが起動していることを前提として，従来のSparkとSparkleのデータ交換モデルを比較した図です．従来は，データ交換を行うプロセス間ごとでコネクションを作成するため，計算規模の増加に伴うWorkerプロセス増加により多数のコネクションが作成されることになります．また，コネクション単位でネットワーク処理を行うことで受信側のプロセスがデータを参照可能になります．しかし，ネットワーク処理を行うためには，転送するためのデータ操作（プロトコルスタックの処理・カーネル空間へのメモリコピー等）を送信側・受信側でともにCPU上で実施する必要があり，この

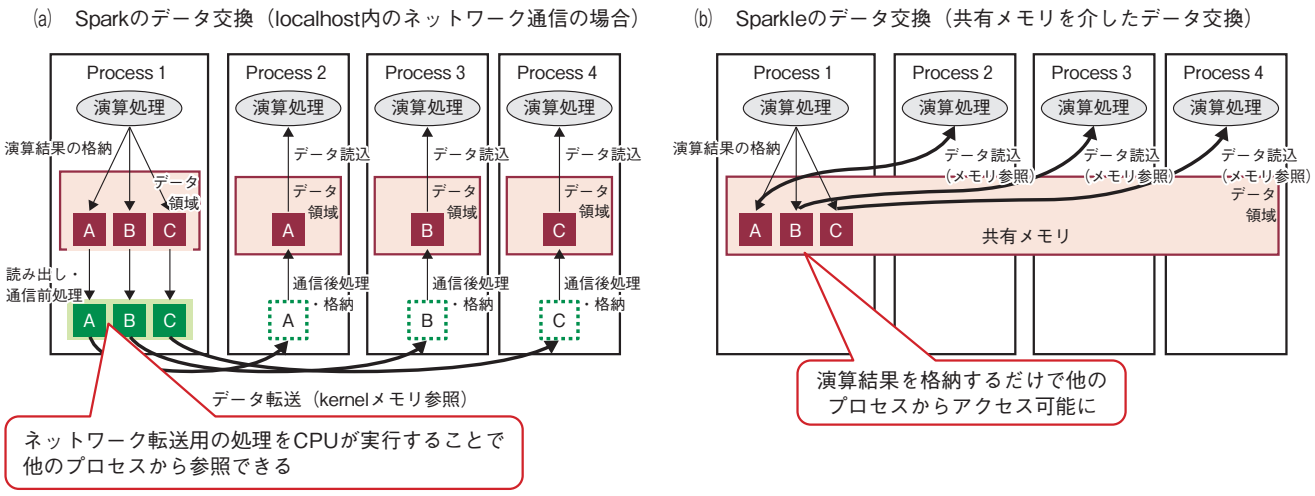

図 2　データ交換方法の差

処理が遅延時間増加要因となっています．一方，Sparkleは送信側のプロセスがデータを共有メモリ上に配置することで他のプロセスからは参照可能になるため，受信側のプロセス自身が必要とするメモリ領域を参照することでデータを取得可能となります．つまり，各プロセスは共有メモリに演算結果を配置するだけで，データ転送のみを目的としたCPUによるデータ操作を行うことなく全プロセスがそのデータを参照可能となり，効率的な送受信が可能となっています．

SparkとSparkleの性能測定結果のグラフを**図 3，4** に示します．図 3 はShuffle処理を含む基本的な 5 つの処理の性能比較結果です．処理によって処理全体に占めるShuffle処理の比が異なるため性能向上幅は異なりますが，特にReduceByでは約 6 倍の速度となっており，大幅な性能向上が確認できます．次に，図 4 はSpark Streamingを用いてストリーミング処理性能を比較した結果となります．この測定においてSparkleはSparkと比較して約 2 倍の性能向上が確認され，遅延要件の厳しい処理に対しての効果が期待されます．

このように，共有メモリを用いたデータ交換モデルの実現により，効率的なデータ交換および遅延時間の削減が可能となることが改めて確認されましたが，実現にあたってはさまざまな課題があります．その中で特に大きな課題の 1 つとして，共有メモリ管理に関する課題があります．現在，共有メモリを利用する方法としては同一ファイルを複数のプロセスでメモリマップする方法が一般的となっており，共有メモリ領域へのアクセスは一般的なC言語のポインタアクセスを用いる必要があります．これは一般的なC言語の記法で記述できる反面，複数プロセス間での共有メモリ領域のアクセス制御や空き領域管理，割り当てなどのメモリ管理はサポートされていません．そのため，アプリケーションプログラマはデータの上書きやメモリ領域の二重確保を発生させないために，各々が独自のメモリ管理機能の実装を行う必要があります．また，共有メモリ自体は，現状では単体サーバ内で利用可能であり，複数サーバをまたがった分散処理を実行するためには，複数サーバにまたがった処理を可能とするための機能の拡張が必要です．

このように，共有メモリの低遅延性を損なわず，かつプログラマがその恩恵を容易に受けられるようなソフトウェアの研究がメモリセントリックの実現に向けた重要な研究課題となっています．

今後の展開

Sparkleの評価結果からCPUに関して，共有メモリを介したデータ交換

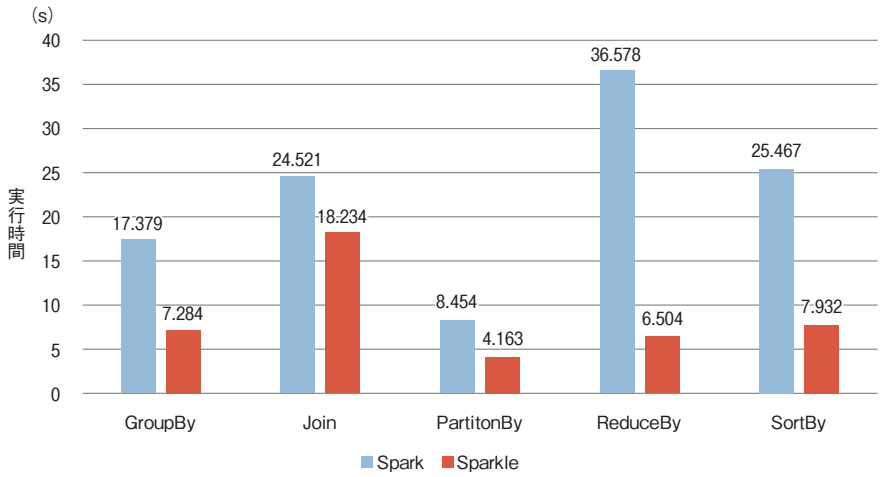

図 3　マイクロベンチマークの性能比較

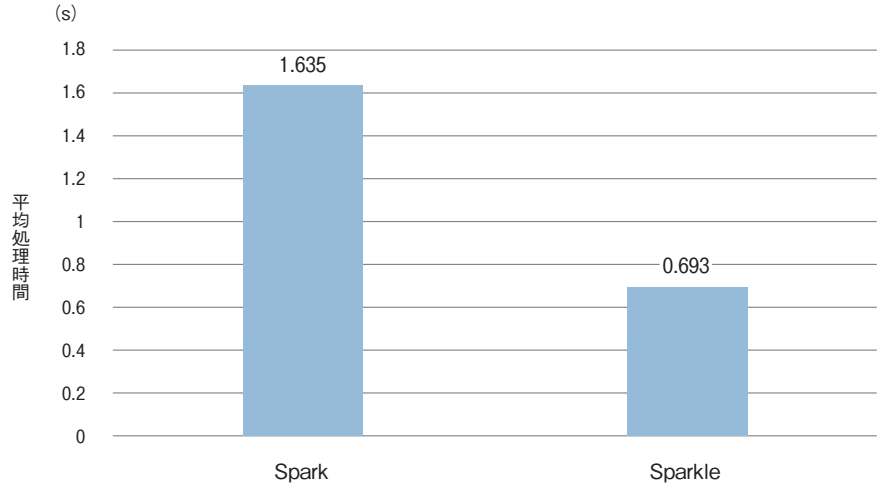

図 4　Spark Streaming の性能比較

モデルを利用する効果があることを確認しました．今後は，メモリセントリックの検討として，この共有メモリを介したデータ交換モデルについてFPGA等の他のアクセラレータへの拡張を検討し，将来的には多様なハードウェア処理をつなぐコア技術として，検討を進めていきます．また，本取り組みについては，IOWN（Innovative Optical and Wireless Network）を支えるディスアグリゲーテッドコンピュータに必要な要素技術として，計算機資源をサーバ筐体単位ではなく各々独立に利用とするためのソフトウェア技術として位置付けされており，ソフトウェア制御によりアクセラレータ間についてCPUを介さずにフラットに接続・利用することにより，遅延時間削減とCPU消費量削減による電力効率の高いコンピューティングの実現をめざしていきます．

■参考文献

(1) K. Keeton：“Memory-driven computing,” FAST 2017, Santa Clara, U. S. A., Feb. - March 2017.
(2) M. Kim, J. Li, H. Volos, M. Marwah, A. Ulanov, K. Keeton, J. Tucek, L. Cherkasova, L. Xu, and P. Fernando：“Sparkle: Optimizing Spark for Large Memory Machines and Analytics,” arXiv preprint, arXiv:1708.05746, 2017.
(3) https://github.com/HewlettPackard/sparkle
(4) https://github.com/sparkle-plugin/sparkle

（左から）石崎 晃朗/ 山部 芳朗

近年のアクセラレータや不揮発性メモリなどのハードウェアの進化に伴い，ハードウェアとソフトウェアの関係性は大きく変化しつつあります．その中で，さまざまなサービスを効率的に動作させるためのプラットフォームとして，基盤ソフトウェアの研究開発を進めていきます．

◆問い合わせ先
NTTソフトウェアイノベーションセンタ
分散処理基盤技術プロジェクト
TEL　0422-59-3488
FAX　0422-59-3739
E-mail　teruaki.ishizaki.ph＠hco.ntt.co.jp