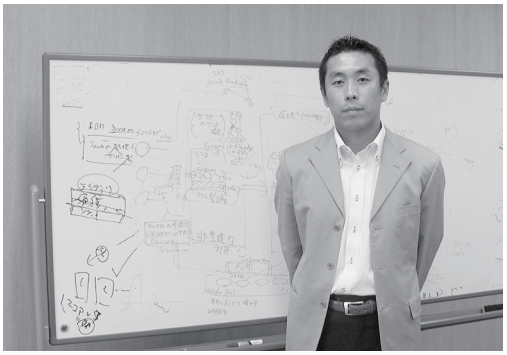
In Laboratory Now

研究室訪問 2

リアルタイムでの大規模データ処理

鈴村 豊太郎 研究室～計算工学専攻



鈴村 豊太郎 客員准教授

近年、ネットワーク技術の進歩やさまざまな通信機器の普及により、私たちは大量の情報をリアルタイムで受け取れるようになった。

時々刻々と送られてくる大量の情報をリアルタイムで扱うために、データストリーム処理という新たなデータ処理が注目されている。データストリーム処理は、送られてくるデータを逐次的に処理することで、リアルタイムでのデータ処理を可能とする。

鈴村研究室では、データストリーム処理のソフトウェアを設計することを想定し、その上で浮かび上がってくるさまざまな課題を研究している。

情報爆発とデータストリーム処理

情報爆発

近年、ネットワーク技術の進歩と、それに伴うさまざまな通信デバイスの発達で、多様な情報をリアルタイムに受け取れるようになった。

例えば現在では、携帯電話には、GPS、電子決済、Web ブラウザなどの機能が搭載されている。携帯電話会社は、これらの利用データを分析することで、顧客の行動傾向や嗜好を常に把握することができる。また、金融業界では、インターネットを通じて株の取引が高速で行われるようになり、今ではマイクロ秒レベルで争われる取引もある。

リアルタイムに情報がやり取りされるということは、同時にその情報量の爆発的な増加を意味する。実際、携帯電話会社は、顧客の携帯電話から保存しきれないほど膨大なデータを受け取っている。また、金融業界では、取引の記録であるマーケットデータの量が５年間で６倍になっている。

このように、リアルタイムに利用できるデータが増大している一方で、それを活用するためのコンピュータの処理は追いついていないのが現状である。そこで、従来とは違った考え方で行うデータ処理方法が注目されている。それが、データストリーム処理である。

データストリーム処理とは

データストリーム処理とは、簡単にいえば、送られてくるデータをストレージに蓄積せずに、メモリ上でリアルタイムに処理するデータ処理方法である。このことを説明するために、まずはコンピュータの構成について説明しよう。

コンピュータは、大まかにいえば次の三つのパーツから構成される。計算を行う中枢であるプロセッサ、実行中のプログラムやその処理対象データなどを一時的に記憶する主記憶装置（メモリ）、そしてさまざまなデータを大量に格納する外部記憶装置である。外部記憶装置はストレージともいわれ、通常はハードディスクが使われる。

このようにコンピュータが二種類の記憶装置を使い分けているのは、次のような特性の違いによる。

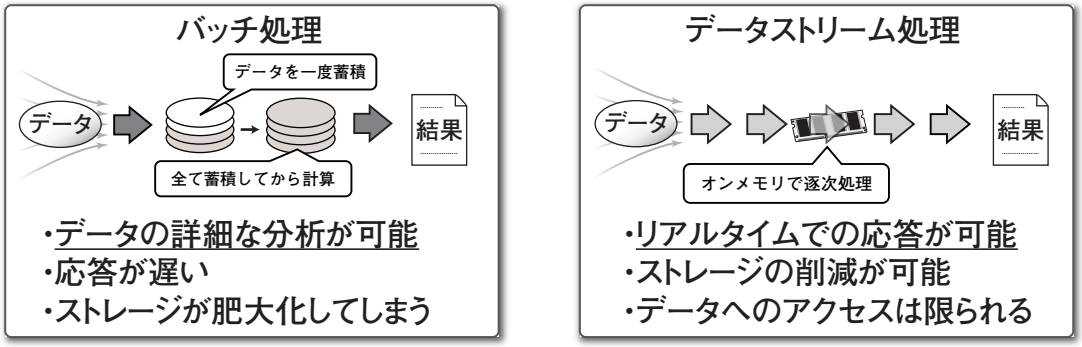


図１　バッチ処理とデータストリーム処理

まず、メモリの容量はハードディスクの容量より遥かに小さい。それに加えて、メモリには常に電力を供給し続けていないと記憶を保持できないという特徴がある。このため大量のデータを保持することはハードディスクにしかできない。しかし一方で、機械的な装置であるハードディスクのデータの読み書きの速さは、電子的な装置であるメモリのそれよりも遥かに劣る。このため実行中のプログラムの一時的なデータなどは、メモリに保持され、書き換えられていく。

従来のデータ処理方法では、対象データはハードディスクなどのストレージに一旦全て蓄積されたのち、処理される。この方式はバッチ処理と呼ばれる。しかし、この方式ではデータが到着してからそれを処理するまでに大きな時間差が生じるため、高いリアルタイム性が要求されるような処理には対応できない（**図１**）。

そこで、データストリーム処理が必要となる。データストリーム処理では、到着したデータをストレージに蓄積せずに、メモリ上で次々と処理していく。このためリアルタイムでの処理が実現され、加えてストレージ容量の削減も期待できる。

もちろん、全ての処理がデータストリーム処理で行えるわけではない。データストリーム処理は、容量の小さいメモリ上で次々とデータを処理していくため、全体のデータを一度に見渡せないという特徴がある。逆にバッチ処理はデータ全体が手元にある状態で処理を行うので、詳細な分析が可能となる。このように、リアルタイム性が必要な場合はデータストリーム処理、詳細な分析が必要な場合はバッチ処理、といった使い分けがなされる。

データを蓄積せずにメモリ上でのみ処理することは、画像表示や動画再生など限られた目的で以前から行われてきた。これに対して、データストリーム処理は、音声や画像も含めた多様なデータを対象としており、必要であればそれらを複合的に用いた処理も行う。この特徴がよく現れたデータストリーム処理の実用例を一つ紹介しよう。

データストリーム処理による自動株売買

データストリーム処理はさまざまなことに利用可能であり、既に実用化されている例も多い。ここでは、金融業界における例を取り上げて、データストリーム処理がどのようなところで使われるのかを紹介する。

金融業界には、株の売り買いをコンピュータの判断により自動的に行うアルゴリズムトレーディングという分野がある。この分野では、取引が非常に高速で行われるため、ネットワーク経由による遅延やデータ処理にかかる時間が損得に大きく関わってくる。そこで、データストリーム処理を導入することが考えられる。

ここでは米国に工場を持つあるメーカーを想定する。米国ではハリケーンの影響が大きく、時には工場の一時操業停止などの形でメーカーの株価に影響を与える。そこで**図２**のようなハリケーンの影響を計算に入れた自動株売買が考えられる。

一番上はニューヨーク証券取引所（NYSE）からのデータである。毎秒 160 万～ 500 万回行われている取引のデータが全て送られてくる。その下は、そのメーカーの財務情報である。これは年数回程度しか更新されない、静的なデータである。

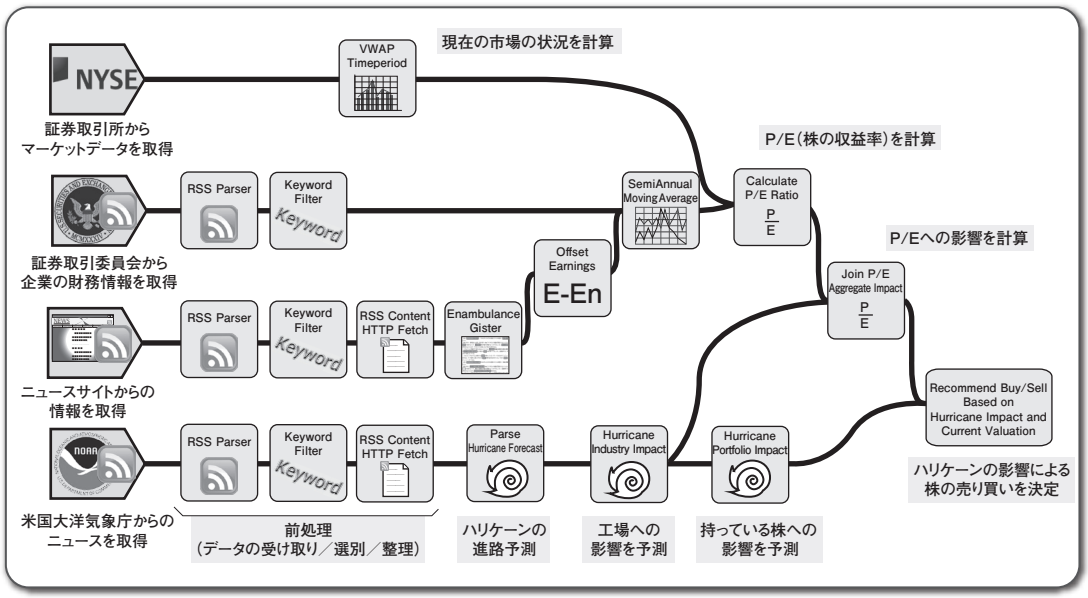


図２　ハリケーンの影響を考慮した自動株売買のデータ処理の流れ

一番下が米国の気象庁からのデータであり、これを元にハリケーンの進路を予測する。そこからメーカーの工場への影響を予測し、最終的に株への影響を算出する。残りはニュースサイトからの情報で、現地の被害状況など、より具体的な情報を取得する。これらの情報を複合的に分析することで、コンピュータが株の売り買いを判断する。

これらのデータ処理は全て、蓄積されずにメモリ上で行われるため、データ処理にかかる時間は非常に短いものとなる。株の自動売買において、この処理時間の短さは重要なファクターとなっている。

ミドルウェアの必要性

データストリーム処理の実用例を紹介したところで、実際にこれらのソフトウェアをどのようにして作るか、ということに話を移そう。

JavaやC言語などの一般的な汎用プログラミング言語でプログラムを書く場合、ストレージを使わずに、メモリ上でのみデータを扱うこと自体は、特別なことではない。したがって、データストリーム処理のソフトウェアを通常のソフトウェアと同じように作ることも、不可能ではない。

しかし、実際に、**図２**のようなデータ処理を行うソフトウェアを作るには、通信による各種データの受け取り、複雑な処理の流れの構築、複数の計算機への分散処理など、数多くの煩雑な手続きを記述しなければならない。このため、上で紹介したような多様なデータを対象としたオンメモリデータ処理は、一般的には行われてこなかった。

この問題を解決するためには、ミドルウェアというものが必要になる。ミドルウェアとは、コンピュータの基本的な制御を行うオペレーティングシステム（OS）と、ユーザーに直接的に操作を提供するアプリケーションソフトウェアとの中間に入るソフトウェアである（**図３**）。これにより、データストリーム処理を行うソフトウェアの開発が容易になる。

鈴村研究室では、鈴村先生がIBM社の研究所に所属している関係から、同社のデータストリーム処理のミドルウェアである、System Sを使っている。

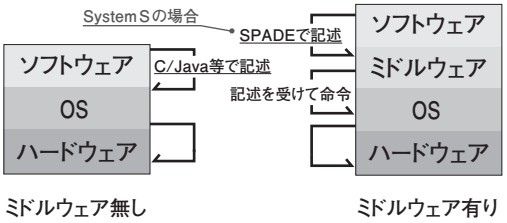


図３　ミドルウェアの概念図

System Sは、データストリーム処理に特化したSPADEというプログラミング言語を備えている。SPADEは、データの受け取りや、各処理へのデータの流しこみなど、データ処理の流れを構築するための基本的な命令を備えており、必要に応じて新たに命令を定義することができる。SPADEにより記述されたデータストリーム処理が、System S上で実行される。

また、データストリーム処理も含め大規模なデータ処理では、複数の計算機を用意して、処理を分散させるのが一般的である。System Sは、用意した計算機の台数などの情報を記述しておくことで、自動的に分散処理を行う機能も備える。

これらのプログラミング言語や機能により、System Sは任意のデータストリーム処理ソフトウェアの構築を可能にしている。別の言い方をすれば、多様なデータを処理できるというデータストリーム処理の特徴は、ミドルウェアの汎用性、拡張性に依存していると言える。実装の観点からいえば、データストリーム処理にとって、ミドルウェアが決定的に重要なのである。

鈴村研究室では、このミドルウェアを用いてデータストリーム処理を行うことを考え、その上で浮かび上がってくる課題を研究している。次からは、鈴村研究室で行われたそれらの研究を紹介しよう。

GPGPUによる大量センサーの同時処理

データストリーム処理では、莫大な数のデータソースを扱い、なおかつそれらをリアルタイムで処理することが要求される。このような状況では、処理を高速化することに加えて、一計算機あたりどれだけ多くのデータソースを処理できるかということも、現実的には重要な課題となる。

これらの課題を踏まえて、具体的な状況として、大量のセンサーを用いて異常を検知する状況を想定する。工場の生産ラインの監視や、ネットワーク通信量の監視などによる異常検知は、リアルタイムでの応答が要求されるため、データストリーム処理により扱われている。

異常を検知するための方法としては、特異スペクトル変換というアルゴリズムを用いる。特異スペクトル変換は、多様なデータの異常を検知することができるという長所がある一方で、計算量が非常に大きいという短所がある。

これを代表的なプロセッサであるCPUで処理する場合を考える。CPUは、非常に簡単にいえば、一度に一つのデータを処理するプロセッサである。処理速度が非常に高速なため、複数の処理を同時に行いたい場合はそれらを交互に処理し、擬似的に並列実行している。CPUでセンサーデータを処理するとき、対象が一つのセンサーであれば、非常に短い時間で処理することができる。しかし、多数のセンサーを対象とする場合、計算量はセンサー数に比例して増加するため、処理時間もどんどん大きくなってしまう。結果として、あまり多くのセンサーデータをリアルタイムに処理することはできない。

そこで鈴村研究室では、GPUを使うことで、リアルタイム性を満たしながらより多くのセンサーを処理できるようにする研究が行われた。GPUとは、Graphics Processing Unitの略で、画像処理に特化したプロセッサである。3D描画への要求などを背景として演算性能が著しく向上したことで、画像処理以外の一般の計算にも用いられるようになった。これはGPGPU（General-Purpose computation on GPU）と呼ばれ、近年非常に注目されている技術である。

ここでは、なぜGPUを使うと多数のセンサーを処理できるのかを、実際に研究で使われたNVIDIA社のGPUを例に説明しよう。

NVIDIAのGPUは、**図4**のように、多数のStreaming Multiprocessor（SM）という処理ユニットから構成される。SMは、GPUで一般的に使われている、SIMD（Single Instruction Multiple Data）という構造をもっている。SIMDとは、そのユニットの中に複数の演算器を持つことで、一回の命令で複数のデータに対して同じ処理を実行するものである。NVIDIAのGPUはSIMD構造のSMを多数持つことで、膨大な数の並列処理を可能としている。

さらにNVIDIAのGPUには、それぞれのSMを最大限に活用するために、大量の命令を待機させておく仕組みがある。一般に、プロセッサが命

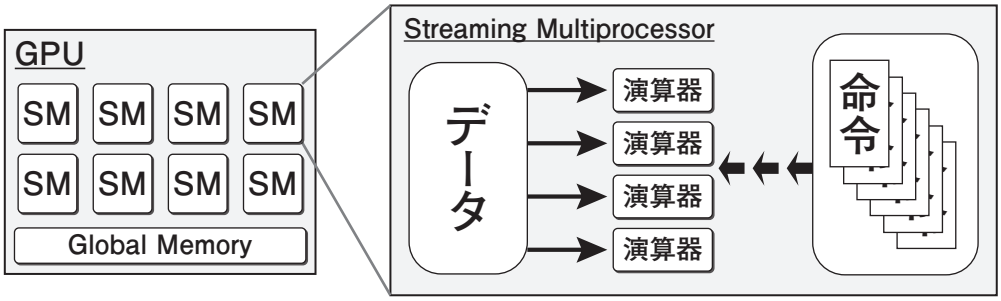


図４　NVIDIA の GPU の概要

令を発行してからメモリの処理対象データを読み出すまでの時間は、プロセッサの演算速度から見ると非常に長い。プロセッサはこの間に数百から数千回の演算を行うことができる。そこでNVIDIA の GPU では、処理したい命令を大量に待機させておき、処理対象データがメモリから読み出せた命令から次々と実行していくという仕組みが取られている。これにより、メモリの読み出しにかかる時間を隠蔽することができるのだ。

以上のような構造により、GPU は膨大な数の並列処理を可能としている。GPU の並列性に着目して、大量のセンサーデータを処理させたのが今回の研究である。

ただし、実際に GPU を有効に使うためには、並列性やメモリアクセスの時間を考慮したプログラミングを行う必要がある。GPU が大量の並列処理を可能とする構造を持っていても、プログラム側で十分な数の並列処理を適切に割り当てることができなければ、GPU の真価は発揮されない。

鈴村研究室で今回行われた研究では、NVIDIA の GPU に対して最適化されたプログラミングを追求した結果、センサー 256 個に対して、CPU の 17 分の 1 以下の処理時間で異常検知を行うことができた。これにより、膨大な数のデータソースに対してデータストリーム処理を行う手段としての、GPU の可能性が示されたと言える。

バッチ処理とデータストリーム処理の動的負荷分散

鈴村研究室が行った研究をもう一つ紹介する。

最初に述べたように、データ処理にはデータストリーム処理とバッチ処理の二つがある。それぞれリアルタイムでの処理と詳細な分析を得意とする。実際のデータの多くは、この二つの処理を組み合わせることで、より活かされる。

例えば株取引の例では、データストリーム処理で株の自動売買を行い、バッチ処理で株価の長期的なトレンドを分析するという使い分けが考えられる。異常検知の例では、データストリーム処理で突発的な異常を検知し、バッチ処理で長期的な異常を検知する、という使い分けが考えられる。

データストリーム処理ではリアルタイム性が重視されるため、データ処理にかかる時間（レイテンシ）を小さく抑えなければならない。一方で、データストリーム処理で扱うデータは、突発的にデータ量が増大するようなものが多い。このような時でもやはりレイテンシを低く抑えなければならない。これに対する単純な解決策として、計算機を余分に用意して待機させておくことが考えられるが、金銭面などから現実的とはいえない。そこで、通常時はバッチ処理に割り当てられている計算機を、データ量の増大によりレイテンシの増加が見込まれる場合はデータストリーム処理に割り当てる、動的負荷分散というものが考えられる。

今回の研究では動的負荷分散を、８台の計算機を使って行った。実行した処理は、前述した特異スペクトル変換である。データストリーム処理でその時々のデータを処理し、バッチ処理で長期のデータを繰り返し処理した。

動的負荷分散の仕組みは、データストリーム処理のレイテンシを監視し、レイテンシの増大が起こった時点でデータストリーム処理に割り当てる計算機の数を増やすというものだ。動的負荷分散をしない状態での計測では、４台をデータストリーム処理に割り当て、残り４台をバッチ処理に

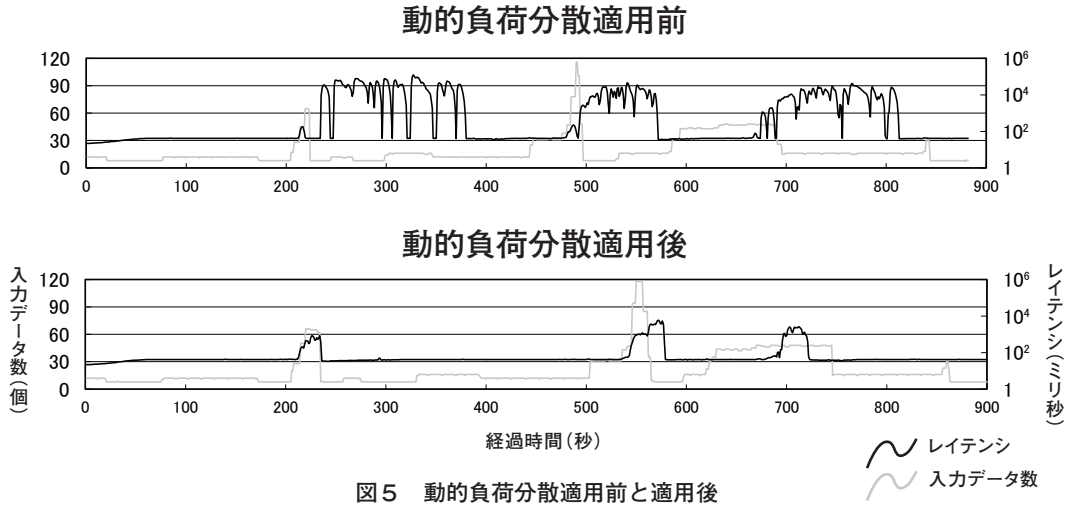


図５　動的負荷分散適用前と適用後

割り当てた。**図５**はその結果である。

グラフを見ると、レイテンシの増加が大きく抑えられていることがわかる。また、グラフには無いがバッチ処理の実行回数も３万回から４万回に増やすことができた。バッチ処理とデータストリーム処理が混在した環境では、動的負荷分散が非常に有効であることが確認された。

今回の動的負荷分散は、レイテンシについての閾値を手動で設定し、それに従って計算機の割り当て数を変更するという仕組みだった。今後の見通しとしては、この閾値を自動的に設定する機構の追加などが考えられる。

鈴村研究室のこれから

鈴村研究室では、この他にもクラウド環境をデータストリーム処理に導入する研究や、大規模なネットワークのグラフ（**図６**）をデータストリーム処理する研究などが行われている。

クラウドとは、近年脚光を浴びている、ネットワーク越しにコンピュータの処理能力やストレージを利用することができるサービスである。実際には、どこかの施設に集められた多数の計算機の一部を利用しているが、ユーザーはどこの計算機を利用しているかを具体的に意識することはない。あくまで抽象化されたサービスとして利用するのが特徴だ。これは、申請すれば即時に利用できるようになるので、突発的にデータ量が増大した場合の処理に対応することができるのではないかと考えられている。

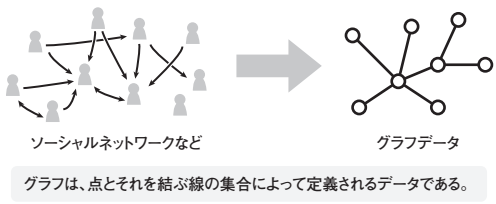


図６ グラフとは

これらの研究の先にはデータストリーム処理の次世代のミドルウェアが見える。現在、鈴村研究室では、その研究成果を実装した新たなデータストリーム処理のミドルウェアを開発中である。

この度、鈴村先生にデータストリーム処理についてお話を伺いました。それらを記事にしていく過程で、計算機についての各方面の知識がどのように研究に活かされるのかを垣間見ることができました。これは、これからコンピュータ・サイエンスの世界に入る自分にとって、非常に大きな糧になったと思っています。

末筆になりますが、度重なる取材に快く応じて下さり、迅速に対応してくださった鈴村先生にこの場を借りて御礼申し上げます。　（竹野　創平）