1. 並列コンピュータとは？

1.1 並列コンピュータとは何か？

並列コンピュータとは、複数のコンピュータあるいはその主要部が、データを交換しながら協調して動作する構成方式である。現在、サーバ、デスクトップ、ノート PC などの本格的なコンピュータはもちろんのことスマートフォンやゲームマシンに内蔵されているコンピュータまで、全て並列コンピュータである。クーラーや電気釜などの家電に組み込まれている簡単なコントローラを除いて、世の中のほとんど全てが並列コンピュータになってしまった。このテキストは、並列コンピュータがどのような原理で動作し、どのようにデータを交換し、どのように協調して動作し、どのようにプログラムするのか。を解説する。

並列コンピュータの構成単位は、単独で動作可能なコンピュータであるもの、中央処理装置（CPU：Central Processing Unit）だけのもの、CPU にキャッシュと呼ばれる高速メモリが付いたものなど様々であるが、とにかく独立に演算を実行する能力を持っている。ここでは、この構成単位をプロセッサあるいはコアと呼ぶ。コアと呼ぶのは、半導体チップ上に複数プロセッサが搭載されている場合に限られるのだが、最近は、このような状況がほとんどなので、プロセッサ＝コアと考えて良い。

並列コンピュータはプロセッサの集合体であるため、本テキストを理解するには、元となるコンピュータアーキテクチャの知識、C 言語の知識を必要とする。しかし、並列コンピュータは今日余りにも広く普及しているため、なるべく多くの読者を対象としたい。そこで、必要な際には復習を交え、なるべく初学者でも理解できる解説を試みる。

1.2 なぜ複数のプロセッサを持つのか？

（1）みんなで働けば早くなる

複数のプロセッサを持つ第一の理由は単純で、「みんなで協力して仕事をすれば速くできるから」である。同じ能力を持つ p 台のプロセッサに対して、同時に処理可能な等量の仕事をやらせれば、実行時間を 1/p にすることができる。しかし、そんな理想的なことは滅多に起きない。これは並列に実行するためには一定の損失を伴うからだ。

① 多くのプログラムは完全に並列に実行することはできず、必ず逐次的に実行しなければならない部分（あるいは並列動けるプロセッサ数が減る部分）を含んでいる。（並列性の制限）

② 全てのプロセッサに対して同じ量の仕事を分散できないかもしれない。この時は一番重い仕事を受け持ったプロセッサの処理時間が全体の処理時間になってしまう。（負荷分散の偏り）

③ 協調して仕事を行うためには、自分が処理したデータを他人に送り、他人から結果を受け取る必要がある。データの授受のためには足並みを揃える必要があり、これを同期と呼ぶ。データの転送時間と同期時間が計算に必要な時間に加えて必要になる（データ転送と同期）。

並列コンピュータに取ってどれも本質的な問題だが、①の問題が最も本質的である。

今、一つのプログラムに対応するジョブを複数のプロセッサで実行する場合を考える。並列に実行するためには、コンパイラあるいはプログラマの指定により、ジョブを複数の実行単位に分割する必要がある。この実行単位をタスク、プロセス、スレッドなどと呼ぶ。それぞれの言葉は色々な用途で使われてややこしいのだが、ここではこのような並列性をタスクレベル並列性と呼ぶ。一方、大規模な配列を取り扱うプログラムでは、ほとんど同じ処理を大量のデータに対して実行する場合がある。このような並列性をデータレベル並列性と呼ぶ。

並列コンピュータは、上記の並列性を様々な形で利用して、高速な実行を目指すが、当然ながらそれには限界が存在する。これは、有名なアムダールの法則（Amdahl's Law）で説明される。今、単純化して、あるプログラムの実行時間のうち x の割合は完全に並列に実行でき、残りは逐次的しか実行できないとする。この場合 p プロセッサを用いた並列処理による性能向上は以下の式で表すことができる。

$$1/((1-x)+x/p)$$

ここで、x/p は p 台で並列実行した部分を指し、（1 − x）は逐次的に実行されるため、性能向上できない部分である。

例題１：　完全な並列処理可能な部分が 95%あり、残りの 5%は逐次処理する必要があるプログラムについてプロセッサ数が 10、100、10000 として、それぞれ性能向上を計算せよ。
答：x=0.95 としてアムダールの法則の式に当てはめる。1-x＝0.05 は常に変わらず、0.95 の部分が、p が増えるとともに小さくなっていく。10 台ならば、6. 9 倍、100 台ならば、16.8 倍、10000 台でも 19.9 倍となる。当たり前だがいくらプロセッサ数を増やしても、性能向上は 20 倍を超えることができない。

処理の総量が決まっている場合に、プロセッサ数を変化させて性能を測ることを、ストロングスケーリングと呼ぶ。ストロングスケーリングを行う場合、逐次的な処理が少しでも存在すると、プロセッサ数を増やしてもある程度以上の効果は得られない。そうは言っても、問題を解くためには最初にデータを配ったり、答えを収集する時など、どうしても並列にできない部分があり、完全に並列処理ができる問題は現実的ではない。この場合、性能が向上可能な限度はその性能向上手法が適用できる割合に制限される、というのがアムダールの法則の教えるところである。この様子を図 1 に示す。

では、スーパーコンピュータなど 100 万を超えるプロセッサを持つ並列コンピュータを作るのはなぜなのだろう？当たり前のことだが 100 万を超えるプロセッサを持つスーパー

コンピュータが解くのは、その規模にふさわしい巨大なスケールの問題に限られる。プロセッサ単体の解く量が決まっている場合に、プロセッサ数を変化させて性能を測ることをウィークスケーリングと呼ぶ。例題 1 を少し変えてみよう。

例題 2：プロセッサ数に比例して問題のサイズを大きくすることで、逐次処理する割合が、10 台の時を基準として 1/p になるとする。例題 1 をこの条件で解いてみよ。

答：1- x は 10 台では 5%、100 台で 0.5%、10000 台では 0.005%になる。したがってそれぞれの性能向上は、6.9 倍、66.9 倍、6666.9 倍になり、それぞれ台数相応の性能向上が得られる。

対象の問題のサイズが決まっている場合は、ストロングスケーリングを使わなければならないが、スーパーコンピュータのスピードを競う Top500 ではウィークスケーリングが使われる。

ここまでの話は、単一のジョブを並列処理する場合で、Web サーチやオンラインショッピングなどでは、互いにほとんど関係のない複数の独立したジョブを実行する。ここではこのような並列性をジョブレベルの並列性（要求レベルの並列性と呼ぶ場合もある）と呼び、それまでの並列性と区別する。このような並列性を取り扱うデータセンターやサーバでは、利用可能な並列性は湯水のようにあるため、単一のジョブを並列処理する場合の苦労はない。その代わりにこれらのシステムでは信頼性など別の要因が重要な課題となる。

（2）エネルギー効率の改善

エネルギーは、実行時間と消費電力の積になる。p プロセッサの並列処理で実行時間が 1/p になっても、p 台分の消費電力は p 倍になるので、ちっとも得をしない。先に紹介した並列処理の損失によって、実際は、実行時間は 1 /p にはならないので、かえって損をするはずだ。しかし、実際に並列処理を行うことで、単一の強力なプロセッサを使うよりもエネルギー効率は改善し、これが 2003 年以降、並列コンピュータが普及した主な理由になっている。これはなぜだろう？

消費電力のうち、プロセッサが動作することによって消費する電力は以下の式で表すことができる。

動的電力＝α ×動作周波数 f×（電源電圧 Vdd）^{2}

一方、動作周波数 f を上げるためには、電源電圧 Vdd を上げる必要があり、逆に f が低くても良ければ Vdd を下げることができる。以下の例題を考えよう。

例題 3：電源電圧が 1.5V で動作するプロセッサが動作周波数 2GHz で動作する。このプロ

セッサを 4 台使うことで性能は 4 倍になる。同じ性能を得るための動作周波数は 500MHz であるが、これを実現するための電源電圧は 0. 8V で良い。動的電力はどのようになるか？
２GH z 動作時の動的電力＝$\alpha \times 2000 \times 1.5 \times 1.5 = 4500\alpha$
500MH z で 4 台動作時の動的電力＝$\alpha \times 500 \times 0.8 \times 0.8 \times 4 = 1280\alpha$

このように電源電圧は 2 乗で効くため、電圧を下げることは動的電力の削減に大きな効果がある。もちろん、この話は理想通りには行かない。まず電力には動的電力の他に、動作しなくても常に流れる静的電力（漏れ電力、リーク電力）がある。この電力も電圧に比例して小さくなるが、プロセッサ数に比例して増えるため、どちらかというと不利に働く。また、動作周波数と、これを実現するための電源電圧は、常に比例関係にあるわけではない。電源電圧は一定以下にすると動作周波数は急激に低下してついには動作しなくなる。なので、プロセッサ数を多くすることで性能がカバーできるからといって極端に電圧を下げることはできない。

しかし、それでもプロセッサを複数にすることで、動作周波数を低くしても一定の性能が得られ、このことで消費エネルギーを削減できるという利点は大きい。

（3）資源の共用

コンピュータの構成要素は、CPU,メモリ、入出力装置の三つである。このうち、メモリや入出力装置は、一つのプロセッサで使うより、複数で使った方が、利用効率が高い。これは並列コンピュータのメリットの一つだが、同じような効果は、ネットワークを経由してプリンタを共有したりすることでも得られる。資源の共用を目的とするシステムはむしろ分散システムと呼ばれる。最近の並列コンピュータでは効率の良い共有を通り越して、共有資源が混雑しすぎて性能が下がってしまうことが多い。ジョブレベルの並列性を利用するデーターセンターで利用されるクラスタは、個々のジョブは互いにほとんど関係ないが、それぞれが利用し、蓄積するデータベースあるいはストレージを共有する必要がある。さらにネットワーク、電源、空調を共有する。多数のコンピュータから構成するのは、それぞれのジョブを高速に処理するためだが、資源を共有するために一か所に集めているとも言える。

（4）複数プロセッサがあればどれかが生き残る

多数のプロセッサを使った大規模システムではプロセッサのいくつかが動作しなくなっても、全体の動作に大きな影響を与えない利点がある。データセンターで利用されるクラスタでは、この利点を積極的に利用している。もちろん、信頼性向上のためには故障したプロセッサを発見してこれを交換する技術が必要となる。

1.3　並列コンピュータの分類

最近の並列コンピュータは、様々な要素が組み合わさっているため、クリアに分類するこ

とはできない。それでも一定の枠組みで分類することは、あるコンピュータの性質を理解するために有効な手段である。

（1）　Flynn の分類

1979 年、Stanford 大学の M.Flynn は、論文中でコンピュータを分類する方法を示した。この方法では命令流（Instruction Stream）の数とデータ流（Data Stream）の数に着目し、以下の 4 つの方式に分類する

○SISD （Single Instructionstream Single Datastream）命令流もデータ流も一つ。基本的なコンピュータの方式で、ユニプロセッサと呼ぶ場合がある。

○SIMD (Single Instructionstream Multiple Datastreams：シムディーと読む）一つの命令流で多数のデータを処理する方式。図 2 に示すようにコントローラは命令を命令メモリから取ってきて、多数のプロセッサがそれぞれのデータに対して同じ処理を行うように指示する。データレベル並列性を簡単に利用することができるため、GPU（Graphics Processing Unit）などの特定用途目的計算加速装置：アクセラレータで用いられる。通常の Intel のコンピュータにもこれと同じ考え方の命令（マルチメディア命令）が装備されている。本書ではこの方式を他のアクセラレータと共に 5 章で扱う。

○MISD(Multiple Instructionstreams Single Datastream）命令流が複数なのにデータ流が複数というのは考えにくいため、通常はこの形のコンピュータは存在しないと考えられている。アナログコンピュータや後に紹介するパイプライン処理がこれに当たるという考え方もある。

○MIMD(Multiple Instructions Multiple Datastreams：ミムディーと読む）複数命令流、図 3 に示すように、複数データ流。それぞれのプロセッサが独立に命令を取ってきて、独立にデータを処理する。協調作業を行うためには、同期を行ってデータを交換する必要がある。現在ノート PC、スマートフォン、サーバ、データセンターで動作している並列コンピュータのうちほとんどはここに分類される。

（2）　共有メモリに注目した分類

Flynn の分類はその簡単さにより現在でも使われているが、MIMD の枠が広すぎて、並列コンピュータの分類としては機能しない。そこで、1970 年代の OS 研究者を中心に共有メモリに注目した分類が提唱された。共有メモリとは、複数のプロセッサが読み書きできる共通のメモリのことである。この方式は専門家以外にはあまり使われていないが、MIMD 型計算機を分類する方法として有効である。

○UMA（Uniform Memory Access model）どのプロセッサからでも同じ時間で読み書きできる共有メモリを持つ方式。集中メモリ型とも呼ばれる。典型的な UMA は図 4 のように共有バスやスイッチで複数のプロセッサが単一のメモリを共有する方式で、集中メモリ型とも呼ばれる。メモリへのアクセスが集中するため、プロセッサ数を増やすことができないが、実現のためのコストが小さいため、ノート PC、デスクトップ PC、スマートフォンなどに

使われている多くの小規模並列コンピュータがこの分類に入る。本書では 2 章でこの方式を扱う

○NUMA（Non-Uniform Memory Access model）共通のアドレス空間を持つが、アクセスするアドレスによって、アクセス速度が異なる方式。図 5 のように、それぞれの個別のメモリを持つコンピュータが、他のコンピュータのメモリをネットワーク経由で読み書きする場合がこれに当たる。分散メモリ方式と呼ばれることもある。UMA に比べて多くのプロセッサを接続して規模を大きくすることができるため、サーバーやスーパーコンピュータの一部がこの方式を用いる。本書では 3 章でこの方式を扱う。

○NORMA（No-Remote Memory Access model：NORA と呼ぶ場合もある）共有メモリを持たず、プロセッサ間はメッセージのやり取りで協調する方式。コンピュータ同士を何らかのネットワークで結べば良いので、最も簡単に大規模並列コンピュータを作ることができる。データセンタで用いられるクラスタと、スーパーコンピュータの一部がこれに分類される。本書では 4 章でこの方式を扱う。

（3）シストリックアレイ、データフロー、ベクトル型、CGRA

最近の特殊目的用のアクセラレータの中には、今まで紹介したプログラム格納型の拡張とは違った考え方で動作するものもある。深層学習に特化した Google 社の TPU（Tensor Processing Unit）は、シストリックアレイ方式を用いていることで注目された。他にも専用目的のアクセラレータには様々な並列コンピュータ構成方式がある。本書では GPU とまとめて 5 章でこれを紹介する。

図 6 に、並列コンピュータの分類図を示す。繰り返すが、最近のコンピュータが複数の方式が組み合わされていることに注意しよう。

1.4　並列コンピュータの歴史

（1）ユニプロセッサの黄金期

初期のコンピュータは性能が不足していた。このため弱体なコンピュータを複数接続して並列コンピュータを作る試みは 1970 年代から研究レベルでは活発に行われていた。特に米国 CMU で開発された C.mmp、CM*などが UMA,NUMA の元祖として知られている。80 年代にはスーパーコンピュータとして NORMA 方式のハイパーキューブマシンが注目され、90 年代にはクラスタ方式、UMA 方式の一部が商業的に成功を収めた。スーパーコンピュータの世界では当初はベクトルコンピュータと言われる方式が優勢だったが、80 年代後半からプロセッサの数を増やす方向にシフトした。日本でも第 5 世代コンピュータプロジェクト、超並列計算機プロジェクト、RWC など様々な並列コンピュータのプロジェクトが実施され、筆者も含めて多くの研究者が様々なプロトタイプを作った。しかし、この時代に並列コンピュータはコンピュータの主流になることができなかった。並列コンピュータがコンピュータの主役に躍り出たのは 2003 年の「マルチコア革命」以降である。なぜなのだろうか？

理由は単純で、2003 年までは、コンピュータの性能を向上させるもっと良い方法が他にあったからである。コンピュータのビジネスモデルは、今も昔も、機械語にコンパイルされたコードを販売する方式である。したがって、コンピュータの性能向上は、現在使っているコードが「そのまま」速くなることが望ましい。そのまま速くする高速化技術を開発する基盤としては複雑な命令セットよりも、高速化しやすい単純な命令セットの方が優れている。まず 1980 年代のはじめに、単一命令長でレジスタ間の演算のみを許す RISC(Reduced Instruction Set Computer)が登場し、その土台を作った。以降、この土台の上に様々な高速化技術が花開き、当時の半導体技術の向上と相俟って、ユニプロセッサはその黄金時代を築いていく。

ユニプロセッサを、「そのまま」速くする手段としてまず使われたのはパイプライン化であった。この方式はプロセッサが命令を処理するそれぞれの段階（ステージ）を、レジスタで区切ってやり、その間で流れ作業を行わせる方法である。この方法ではレジスタで区切れば区切るほどステージの遅延は小さくなり、動作周波数を上げることができる。このためパイプラインのステージ数は 10－15 まで増え、動作周波数は 3GHz まで向上した。

しかしパイプライン化は、パイプラインで処理する命令に依存性があったり、分岐命令が飛ぶかどうかの判定に時間が掛かったり、メモリの応答が遅れると、動作が止まってしまう（ストール）。そこで、ストールした命令を追い越して後から来た命令を実行してしまう動的スケジュールの技術が発達し、アウトオブオーダープロセッサが登場した。また、一度に複数の命令を取ってきて一度に実行させるスーパースカラ処理、分岐を正確に予測する分岐予測技術、分岐するかどうか決まる前にどんどん命令を発行し、予測が失敗したらやり直す投機的実行技術、複数ある計算資源が空いているときに、別のプログラム（スレッド）を実行する細粒度マルチスレッド技術などが、次々に登場した。メモリシステムも、良く使うデータを格納する小容量のメモリであるキャッシュの技術の革新と、主記憶を構成する DRAM（Dynamic RAM）のデータ転送方式の改善などによって、プロセッサの性能向上をサポートした。この技術革新によって、単一のプロセッサの性能は 1980 年の中頃から 2003 年まで、年間約 50%、18 か月で倍になった。これを Moore の法則と呼ぶ。本来この法則は Intel の Gordon Moore が半導体に格納できるトランジスタの数（密度）に対して提唱したものであったのが、コンピュータの性能向上を示すものとして使われた。

並列コンピュータにより単一のプログラムの性能を向上させるためには、コンパイラかプログラマが並列に処理できるタスクの形に分割してやらなければならない。これは「コードをそのまま速くする」という要求に反している。少し待ってやれば性能が数倍になる世の中でだれがわざわざこのような苦労をするだろうか？これが、並列コンピュータがこの時代にメジャーな成功を収めることができなかった理由である。

（2）マルチコア革命

風向きが変わったのが 2003 年である。当時、これまでのユニプロセッサの性能向上を支えてきた技術はついに行き詰りを見せた。パイプラインの段数はこれ以上増やしても性能

向上に寄与しなくなり、複数命令を同時に取ってきて実行する技術も 4 命令程度で限界に達した。様々な高速化技術を取り入れたことで、本来単純な RISC を土台としたユニプロセッサの構造は手が付けられないほど複雑化した。また、性能向上を動作周波数の向上に頼ったために消費電力が増大し、放熱の限界に達した。メモリ技術はついにプロセッサについていけなくなり、Memory Wall（メモリの壁）を生じて、プロセッサの周波数を上げても、全体の性能が上がらなくなった。

80 年代から 90 年代に掛けては、半導体技術の向上がこれらの問題の解決の後押しをしてくれた。半導体技術はその微細加工の最小幅で示され、これをプロセスサイズ（テクノロジーノード）と呼ぶ。80 年代のはじめに 0.35μm であったプロセスサイズは、2000 年には 90nm まで小さくなり、これにより半導体チップに搭載できるトランジスタ数は増大し、トランジスタの動作速度は上がり、電源電圧は下がり、これにより電力も下がった。ところが 2000 年代以降、依然として搭載可能なトランジスタ数は増えるものの、動作速度の向上と電力の削減は、あまり期待できなくなった。このことにより、2003 年の時点でユニプロセッサはにっちもさっちも行かない状態に追い込まれた。

ここに至って Intel は、今までの路線を改め、ユニプロセッサの性能向上を追求することを止めて、マルチコア化を進めることを宣言した。これがマルチコア革命である。元々マルチコア化を進めていた IBM、Sun などはすぐにこれにならったため、一気に並列コンピュータの時代がやってきた。図 7 は、単一のプロセッサの性能向上を示した模式図である。2003 年以降、ユニプロセッサの性能向上は穏やかなものとなり、代わってコア数が増えていくようになった。

では、なぜここで、Intel は並列コンピュータへの道を選んだのだろうか？Intel は 90 年代、より多くの命令を同時に取ってきて実行するための新しい命令セット IA64 を提唱して、これを装備した Itanium を市場に投入した。Itanium は「そのまま速くする」という要求を満足しなかったためあまり使われなかったのだが、ユニプロセッサが行き詰った 2003 年にこれと類似の道を選ぶことができたのでないか？

並列コンピュータは確かに今までのコードを「そのまま速くすること」はできない。しかし、構成要素のプロセッサでは今までのコードがそのまま走るし、ジョブレベルの並列性にならば何もしなくても対処できる。現在の PC、スマートフォーン、タブレットのユーザーは、ストリーム画像を見ながら、音楽を聴きつつ、文章編集を行い、この間ウィルス駆除プログラムを走らせおく位のことは普通に行う。以前に比べて多数のジョブが走っているわけで、このような環境ではマルチコアは単一のジョブが高速化できなくても体感的に速く感じられる。また、先に紹介した並列化による消費電力の削減効果により、放熱の問題を解決できる。このように、並列コンピュータという選択は、決して革命的なものではなく、「マルチコア革命」とはむしろ穏健な転換であったと言える。

（3）そしてドメイン特化型計算機へ

マルチコア革命以降、コア数はどんどん増加して行ったが、これも 8 程度で限界に達し

て、増加の速度が鈍ってきた。現在、通常の用途に対してならば、マルチコア方式のコンピュータの性能は既に十分なのではないだろうか。ではこれで良いのか、というと決してそんなことはなく、AI 技術、ビッグデータ処理、IoT、自動運転など、今までとは違った性能要求は時代と共にどんどん増えている。これに対応するには、特定の用途にのみ高い性能を低い消費電力で実現するアクセラレータあるいはドメイン特化型計算機が有利である。これらの多くは大規模な並列コンピュータである。本書ではこれを 5 章で取り扱うことにする。