0.1. 計算理論，並列・分散計算処理

中島 康彦（奈良先端科学技術大学院大学）
山下 茂（立命館大学）

本節では，コンピュータを利用した科学技術計算あるいは計算技術の発展という観点から，計算理論，オートマトンに関連する技術史と，複数台のコンピュータを連動させて高速の計算処理を実現する並列・分散処理，ベクトル型スーパコンピュータ，グリッドコンピューティング，PC クラスタ，クラウドコンピューティングなどの並列・分散計算処理に関する技術史についてまとめる．

0.1.1 計算理論とオートマトン

情報処理装置は，**オートマトン**と呼ぶ「現在の内部状態と現在の入力から，次の内部状態と出力が決定されるような機械」によりモデル化できる．オートマトンは神経回路網のモデルととして，W. S. McCulloch と W. Pitts が 1943 年に最初に提案した．1950 年代にオートマトンに関する研究は大きく進展し，その研究成果は計算機科学の応用領域において重要な役割を果たしている．例えば，テキスト処理，コンパイラ, プログラミング言語に関して密接な関係がある形式言語の種々の言語のクラスと補助記憶の違いによるオートマトンの分類との関係についての理論整備が行われた．また，1955 年に G. H. Mealy および 1956 年に E. F. Moore がそれぞれ提案した，Mealy 型および Moore 型と呼ぶオートマトンのモデルもハードウェアの設計理論に利用されるようになった．

オートマトンの研究とは独立に，1930 年代から「計算できる」とは何かという点が研究された．A. Turing が 1936 年にチューリングマシン（以下，TM）と呼ぶ機械を提案し，機械的な手順で計算できることと TM で計算できることが本質的に等価であることを示した．

その後，計算の効率を議論する「計算量理論」と呼ぶ研究分野が進展した．この分野は，J. Hartmanis と R. Stearns が 1965 年にアメリカ数学会誌に発表した論文から始まったとされる．この分野では，問題の入力のサイズを n とした時に，問題を解く TM の計算ステップ数の最悪ケースでの上界が n の多項式関数で表現できる場合は「効率的」に解ける問題と考え，そのような問題のクラスを **P** と呼ぶ．

TM の各ステップで複数の動作が同時に可能な非決定性 TM と呼ぶ TM を考えて，上記の **P** の定義における TM を非決定性 TM で置き換えた場合の問題のクラスを **NP** と呼ぶ．**P** には含まれないため効率的に計算できないと考えられているが，**NP** に含まれるような実用的な問題が数多く知られている．**P** および **NP** に関連して，1970 年代初頭に S. Cook，L. Levin，R. Karp らによって整備された「NP 完全性」の概念によって，効率的に計算できるかどうかを数学的に議論可能となり，この分野の研究が進展した．

与えられた数が素数かどうかを確率的に判定するアルゴリズムを 1977 年に R. M. Solovay と V. Strassen が提案した．これにより，確率的に動作するアルゴリズムが研究されるようになり，J. Gill が 1997 年に，**P** の定義において，通常の TM を各ステップで複数の動作を確率的に選べるようにして，$\frac{1}{3}$ より小さい確率でしか間違えずに計算できる問題のクラスを **BPP** と定義した．**BPP** に含まれる問題は，原理的にいくらでも小さくできるエラー率を許容すれば，効率的に計算できる問題と考えることができる．逆に，たとえ現在の方式の計算機を 100 万台並べて計算したところで，**BPP** に入っていない問題は，少しのエラーを許したとしても，非常に大規模なサイズになると現実的な時間では解けないと信じられている．実際,「どんな物理的な装置を使ったとしても，その装置が使う時間，空間，エネルギーの和に関する多項式関数のステップ数を使えば，確率的な動作を許した TM がその装置の動作を小さいエラー率でシミュレートできる」という Polynomial Time Turing Hypothesis と呼ぶ仮説がある．

1994 年に P. W. Shor が，大きな数の素因数分解がその桁数の多項式関数のステップ数で量子計算機で原理的に解けることを示した．この問題は，**BPP** に入っていないと考えられているため，現在，上記の仮説を覆す可能性が最も高いのは，まだ実現していない量子計算機であると考えられている．

0.1.2 並列・分散計算処理

量子計算に限らず，アルゴリズムの改良により計算量そのものを削減できれば，計算時間を大幅に短縮できる．例えば，バブルソート（計算量 $O(N^2)$）をクイックソー

ト（計算量 $O(NlogN)$）に置き換えた場合，要素数が大きいほど高速化率が上がる．一方，並列・分散計算処理は，複数の計算資源を投入して計算時間の短縮を図る方法である．100 倍の計算資源を投入した場合の理想的な計算時間は 100 分の 1 であるものの，実際には，少なくとも計算の開始および終了時点のオーバヘッドにより効率が低下する．また，計算の途中結果を頻繁に送受する場合には，通信オーバヘッドによりさらに効率が下がる．並列・分散計算処理技術は，（1）CPU コア（1 つのプログラムカウンタが制御するハードウェア単位）内部の並列化技術，（2）ノード（1 つの主記憶空間を有し 1 つのＯＳが制御するシステム単位）内部の並列化技術，（3）複数のノードを束ねる並列化技術に大別できる．

（1）に関する要素技術は，1960 年代から 1970 年代にかけて出揃い，様々な改良が加えられて現在の製品にも広く採用されている．代表的な技術を列挙する．

スーパスカラ ハードウェアが依存関係に基づいて命令レベル並列性を抽出し，複数の演算器に対して動的に命令を投入する技術である．起源は CDC6600（1964 年）に遡る．10 個の演算器とスコアボードを用いるアウトオブオーダ実行方式により高性能を達成している．ただし各演算器はパイプライン化されておらず，演算が完了するまで次の演算を開始することはできなかった．

ベクトル演算 複数の演算器が 1 つの命令に従って一斉に演算を行う技術であり，SIMD（Single Instruction Multiple Data-stream）方式とも呼ぶ．起源は ILLIAC IV（1965 年開発開始）である．64 個のプロセッシングエレメント（PE）が一斉動作する構成であった．

演算パイプライン 演算器の途中にラッチを配置することにより，先行演算の完了を待たずに後続演算を投入可能とする技術である．ただし後続演算が先行演算の結果を入力として必要とする場合は，先行演算の終了を待たなければならない．また，除算は一般に使用頻度が低く，演算結果を得るまでのサイクル数も多いため，通常，除算はパイプライン化しない．起源は，TI ASC（1966 年），CDC7600（1969 年），CDC STAR-100（1974 年），CRAY-1（1976 年）に見ることができる．

VLIW（Very Long Instruction Word） スーパスカラがハードウェアにより命令レベル並列性を抽出するのに対して，コンパイラ等によりあらかじめ同時実行する命令をスケジュールしておく方式である．当初はホストコンピュータを高速にエミュレーションするためのマイクロプログラム制御に用いられていた．その後，簡素なハードウェア構成が省電力化に向く点が注目され，省電力指向 CPU への応用が広がった．パイオニアは，QA1（1976 年発表），ELI-512（1983 年発表），QA2（1983 年発表）である．

マルチスレッディング CPU コアにプログラムカウンタを追加して，複数プログラムの同時実行を可能とする技術である．スーパスカラなど複数の演算器を搭載する構成の場合に，キャッシュミス等に起因する演算器の空きを有効利用できる．ただし，演算器やキャッシュなど資源の多くを複数プログラムが共有するため，CPU コアを複数搭載する場合に比べると性能向上率は低い．商用機の起源は，HEP（1982 年）である．2000 年代に入ると，それまでグラフィック表示専用の I/O プロセッサであった GPU が，GPGPU（General-Purpose computing on Graphics Processing Units）として高性能計算に利用されるようになった．集積度が上がった LSI において問題となる主記憶遅延を隠蔽するために，SIMD と大規模なマルチスレッディングを組合せている．

シストリックアレイ SIMD とは対称的に，規則的なデータストリームに対して複数の演算を同時に適用する方式である．1982 年に CMU の H.T.Kung らが提唱した．現在では CGRA（Coarse Grained Reconfigurable Array/Architecture）として発展を続けている．

（2）の代表的な要素技術は，複数の CPU コアと主記憶を高速リンクにより接続するマルチプロセッサである．8 台程度までの並列化に適する SMP（Symmetric Multi Processor）と，128 程度またはそれ以上に対応可能な NUMA（Non Uniform Memory Architecture）に大別できる．前者は，複数の主記憶装置を複数の CPU コアに対して対等に接続する構成であり，対称マルチプロセッサとも呼ぶ．汎用大型計算機の性能向上を目的として導入が始まった．1980 年代後半から急速な性能向上を遂げた RISC 型マイクロプロセッサでも，1990 年代前半からマルチプロセッサ対応が進み，裾野が広がった．しかし，当時はコア単体の性能向上が著しく，コア数が２個ないし４個程度の構成では，すぐに後継 CPU の性能に追い付かれ陳腐化することが珍しくなかった．

一方，2000 年代に入るとコア単体の性能向上が鈍化し，さらに並列度を向上させるために，コンシューマ向けからハイエンド向けまで後者が幅広く使われるようになる．2010 年代には，コンシューマ向けシステムでも，1 つのソケット（外見上 1 個の CPU を搭載する物理構造）に複数の CPU コアを搭載するマルチコア構成が一般的になり，CPU コアと主記憶の間に，専用キャッシュと共有キャッシュからなる記憶階層および高度なキャッシュコヒーレントプロトコルが導入された．ハイエンド向けでは，複数ソケット構成が一般的になり，各ソケットに近接配置した主記憶を全ての CPU コアが高速リンク経由で参照する構成が一般的になった（主記憶空間数は 1 なのでノード数は 1 と数える）．いずれの構成においても，CPU コアから見ると，アドレスとタイミングによって主記憶参照の遅延時間が大きく異なる状況となり，ソフトウェアの性能チューニングは難しくなっている．

マルチプロセッサシステムにおいて単一プログラムを高速化するには，当初，fork() システムコールを用いてプロセス空間全体をコピーし，I/O 機能である pipe() を使用してプロセス間で計算結果を送受するのが一般的であった．その後登場した pthread() システムコールは，単一のプロセス空間を複数スレッドが共有する仕組みである．プロセス空間のコピーが不要になり，主記憶空間の直接参照により計算結果を授受できるようになった．マルチコアにおける共有キャッシュとの相性も良いことから，並列プログラミングに広く使われている．

（3）は，複数ノードを一般的なネットワークにより接続する構成であり分散システムと呼ぶ．ノードごとに主記憶空間およびＯＳが独立しており，複数ノードにまたがる並列化に fork() や pthread() を使用することはできず，代わりに send()/recv() などのメッセージ通信機能を使用する．ただし，最近のスーパーコンピュータのように，専用の高速リンクにより複数ノードを接続し，全体として単一プログラムの並列実行に特化した構成は，マルチプロセッサシステムと呼ぶのが一般的である．

0.1.3 ベクトル型並列スーパコンピュータ

以上の要素技術を組み合わせた商用スーパコンピュータに，VPP シリーズ（1992 年- 1999 年）がある．当時の LSI の集積度では，高性能コンピュータを構成するために複数 LSI を必要とし，結果的に主記憶との接続に必要な信号線数がボトルネックとならず，十分な主記憶スループットを生かすベクトル演算パイプラインが最適であった．また，主記憶遅延が十数サイクルであったため，スカラユニットにシンプルな VLIW が採用され，動作周波数の向上および低電力化に貢献した．さらに，数百台規模のスケーラビリティを維持するために，必然的に各ベクトルユニットが主記憶を占有する複数ノード構成となり，複数ノードを高速リンクにより接続する構成となった．しかしその後は，コストダウンの圧力から，高い主記憶スループットに依拠するベクトル演算方式は下火になり，現在では汎用プロセッサを搭載するノードを高速リンクにより多数接続する方式が主流である．

0.1.4 ＰＣクラスタ

安価な市販 PC を多数ネットワーク（Ethernet や InfiniBand）接続して高性能コンピュータを構築したものをＰＣクラスタと呼ぶ．フリーのＯＳおよび MPI（Message Passing Interface）ライブラリの搭載により，比較的容易に並列処理環境を構築できることが特長である．

0.1.5 グリッドコンピューティング

並列処理に参画するノードをより広い範囲から集める利用形態をグリッドコンピューティングと呼ぶ．2000 年代に入って研究開発が進んだ．ノード間接続には，専らインターネットが使用され，通信遅延時間は極めて大きい．また，各ノードは，普段，互いに無関係な計算を行っていることが多いため，一斉に利用するためには各ノードの空き時間のスケジュールが必要である．各ノードに専用のミドルウェアを搭載し，マスターから多数のワーカーにジョブを送信し，マスターが計算結果を収集する形態の並列処理が一般的である．

0.1.6 クラウドコンピューティング

大量の計算資源およびハードウェアの仮想化技術により，ユーザが必要とするコンピュータシステムを動的かつ安価に提供することを特長とする商用計算環境の総称である．2000 年代に入って本格的な普及が始まった．大量のユーザを抱えることにより，必要な物理計算資源のピークを平均化し，計算資源の稼働率を上げることにより利用料を下げている．