Chapter

1

並列化入門

この章では、並列化プログラミングをはじめて学ぶ読者のために、一般的なプログラミングの並列化についてハードウェアとソフトウェアの両面から説明していきます。次章以降の OpenCL の解説への橋渡しとなります。

1-1 なぜ並列化が重要なのか

CPU の動作周波数が毎年のように向上し、同じソフトウェアが時間の経過とともに速くなったのは、今となっては古き良き時代の話となってしまいました。 Intel CPU の周波数が 4GHz に近づいた 2004 年頃からは、消費電力の増加、発熱量の増加が、いわゆる「Power Wall」という深刻な問題として立ちふさがり、CPU の動作周波数は頭打ちとなりました（**図 1.1**[注1]）。そのような環境下において、2004 年以降、CPU の進化は、動作周波数の向上を諦め、演算コアの増加という形で現れるようになってきています。CPU の動作周波数はまったく変わらないか、むしろ若干遅くなる傾向にあるため、1 つのコアで動作する従来からの単純なソフトウェアは、CPU を最新のものに変えたとしてもほとんど速くなりません。最新プロセッサの性能を引き出すためには、対象となるソフトウェアが複数の演算器を使えるように最適化し、演算を並列処理させる必要があるのです。

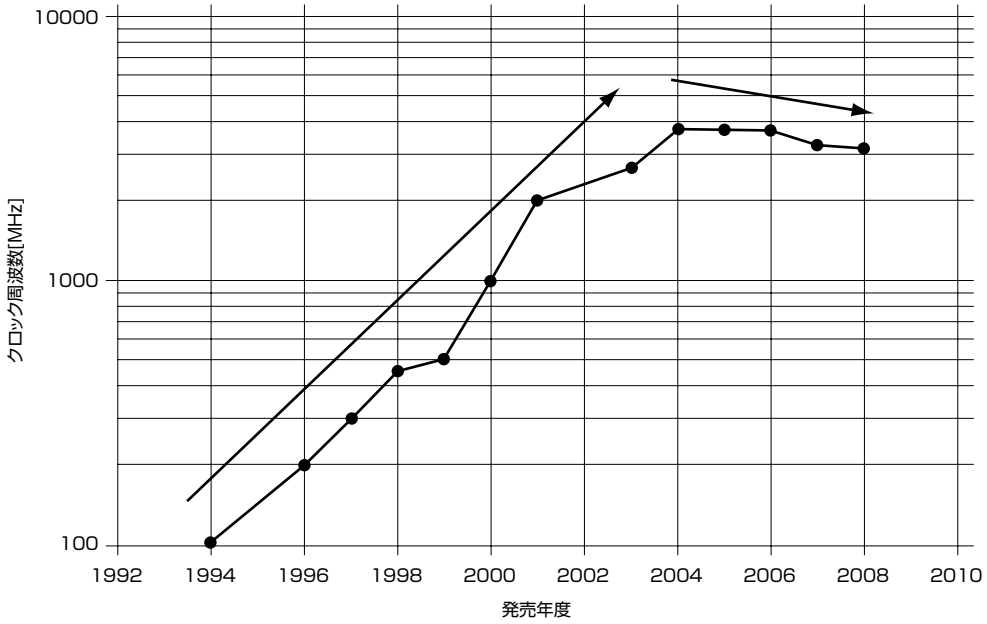

図 1.1：Intel プロセッサにおける動作周波数の変遷

注 1　Intel 公開資料（http://support.intel.co.jp/jp/support/processors/index.htm）などを元に筆者が作成したもの。

現在はコンシューマ向けのノートパソコンにもデュアルコア CPU が搭載される時代です。並列化技術は、一部の高度な演算処理のためだけではなく、さまざまな場面で利用可能になっています。

1-2 並列コンピューティング（ハードウェア）

まず、「並列コンピューティング」とはなんでしょうか？ Wikipedia によれば、「並列コンピューティング（へいれつ、parallel computing、並列計算、並列処理）とは、コンピュータにおいて複数のプロセッサで 1 つのタスクを動作させること。問題を解く過程はより小さなタスクに分割できることが多い、という事実を利用して処理効率の向上を図る手法である。」[注2] とあります。

実際に「複数の処理装置で 1 つのタスクを動作させる」ためのハードウェア・アーキテクチャはさまざまです。規模の大きいものから小さいものへと例を挙げていくと、

- インターネット接続された多種多様な計算資源を結び付け、ある 1 つの目的を持った演算を行うグリッドコンピューティング
- スーパーコンピュータのアーキテクチャとして有名な超並列計算機（Massively parallel processor、MPP）システム
- 汎用コンピュータを組み合わせて作るクラスタサーバ（Cluster server）システム
- 同一の CPU を複数個搭載し密に結び付け、1 つの計算機を構成する対称型マルチプロセッシング（Symmetric Multiprocessing、SMP）システム
- CPU チップ内部に複数個の演算コアを持つマルチコアプロセッサ（Multicore Processor）システム

などが代表的な並列処理システムといえるでしょう。

1-2-1 フリンの分類

コンピュータ・アーキテクチャを分類する指標としてよく知られているものに、フリンの分類（Flynn's Taxonomy）があります[注3]。マイケル・J・フリン（Michael J. Flynn）は、命令ストリーム（あ

注 2　Wikipedia, http://ja.wikipedia.org/wiki/ 並列コンピューティング

注 3　Flynn, M., Some Computer Organizations and Their Effectiveness, IEEE Trans. Comput., Vol. C-21, pp. 948, 1972.

る1つの処理を実現するための命令の列）の並行度とデータストリーム（処理対象となるデータの列）の並行度に基づき、すべてのコンピュータ・アーキテクチャを次の4つに分類しました。

この分類に従うと、先に挙げたクラスタサーバシステムやSMPシステムなど、並列コンピューティングを行うためのハードウェア・アーキテクチャのほとんどはMIMDのカテゴリとなってしまいます。そこで、さらなる分類のために、ハードウェアのメモリ構成に注目するのが一般的です。

1 Single Instruction, Single Data stream（SISD）

SISDシステムは、1つの命令ストリームが1つのデータストリームを処理する、命令にもデータにも並列性のない逐次的なシステムです（**図 1.2**）。シングルコア・シングルプロセッサを搭載した旧来のパソコンがこれにあたります。

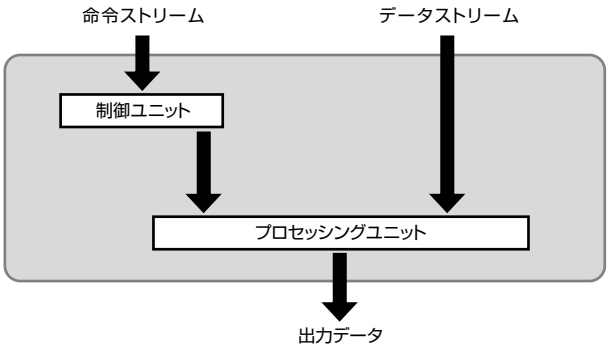

図 1.2：SISD アーキテクチャ

2 Single Instruction, Multiple Data streams（SIMD）

単一の命令が複数の演算器に通知されますが、処理されるデータはそれぞれの演算器で異なるシステムです（**図 1.3**）。ベクタ計算機と呼ばれる一部のスーパーコンピュータがその代表例です。構造はかなり異なりますが、最近の一般的なマイクロプロセッサにもSIMD演算器を備えるものは多く、例として、Intel系CPUのSSE命令、Cell Broadband EngineのSPE命令などが挙げられます。

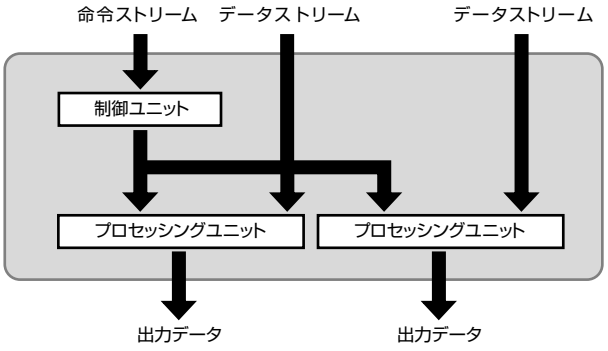

図 1.3：SIMD アーキテクチャ

3 Multiple Instruction, Single Data stream（MISD）

定義としては、複数の命令ストリームが単一のデータストリームを処理するシステムですが、この定義に合う実際のシステムはほとんど存在しません。

4 Multiple Instruction, Multiple Data streams（MIMD）

複数の演算器がそれぞれ異なるデータストリームをそれぞれ異なる命令ストリームで処理するシステムです（**図 1.4**）。

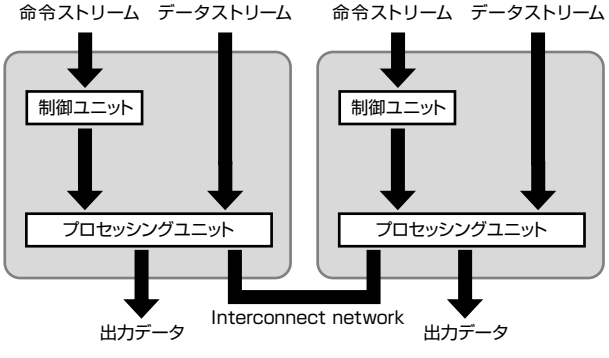

図 1.4：MIMD アーキテクチャ

メモリ構成に注目すると、並列コンピューティングシステムは、共有メモリ型と、分散メモリ型に大きく分類できます（**図 1.5**）。共有メモリ型とは、システムを構成する各 CPU がシステムの持つどの領域にもアクセス可能な構成です。分散メモリ型とは、システムを構成する CPU それぞれが固有のメモリを持ち、異なる CPU のメモリには直接アクセスできない構成です。

このようなメモリ構成の違いは、そのままデータ通信手段の違いに現れます。**図 1.5** に示したシス

テムで、各 CPU で 1 つのプロセスが走っている場合、共有メモリ型であれば、プロセス間でのデータのやり取りは単に共通のメモリ空間の Read/Write になりますが、分散メモリ型では、各 CPU は異なる OS に管理された異なるメモリ空間を持っているので、データの送受信はユーザが明示的に行う必要があります。

次節では、この 2 つの異なる並列システムについて詳しく見ていきましょう。

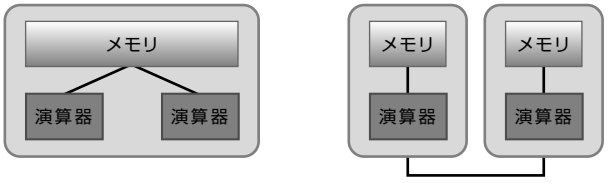

図 1.5：代表的な 2 つの並列処理システム

1-2-2 分散メモリ型

PC1 台では時間がかかりすぎて解けない問題に対して、複数の計算機をネットワークにつないで 分散処理させる方法があります。これが「クラスタサーバ」と呼ばれるシステムであり、分散メモリ型並列処理システムの代表例といえるでしょう。 大規模シミュレーションのようないわゆる HPC（High Performance Computing）と呼ばれる分野では、昔からこのような分散処理の手法がとられてきました。

分散メモリ型に分類されるもう 1 つの並列コンピューティングシステムには、超並列計算機（MPP）システムがあります。 MPP システムは、CPU、メモリ、通信ポートで構成される特殊な計算ノードを、独自の高速ネットワークで多数接続したものです。 NEC が開発した地球シミュレーターや、IBM が開発した Blue Gene は MPP システムの代表例です。

クラスタと MPP の違いは明確ではありませんが、特殊なハードウェアを使用せず、コモディティーな計算機とコモディティーなネットワークを組み合わせて構築したものが一般にクラスタと呼ばれています。構成要素が低価格なため、MPP に比べて非常にコストパフォーマンスが高いのが特徴です。実際に、従来までのスーパーコンピュータの典型であった MPP システムは、完全にクラスタサーバにとって代わられてしまっています。 2009 年 11 月現在の TOP500 Supercomputer Sites[注4] によれば、トップ 500 にランクインしたシステムのうち、MPP が 16.2% なのに対し、クラスタは 83.4% にもなり

注 4　TOP500 Supercomputer Sites, http://www.top500.org/

ます（**図 1.6**）。

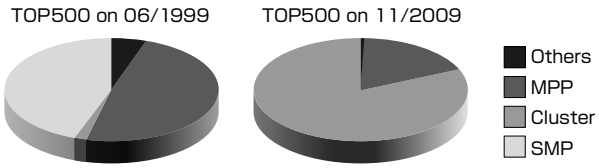

図 1.6：TOP500 Supercomputer Sites におけるアーキテクチャ別統計

クラスタの弱点は、プロセッサ間のデータのやり取りが大変遅いことです。これは、データのやり取りが外部ネットワーク経由であるためです。一般に入手可能な外部ネットワークも、Myrinet、Infiniband、10Gbit Ethernet など、日々新しい規格が登場し、従来の Gigabit Ethernet よりもはるかに高速なデータ通信を可能にしています。しかし、それでもプロセッサがローカルのメモリにアクセスするスピードに比べれば、1 桁以上遅いのが現状です。

よって、クラスタは、並列化した場合にプロセッサ間のデータのやり取りが少ないアルゴリズムに適したシステムといえます。このようなアルゴリズムを「粗粒度並列性（coarse-grained parallelism）」アルゴリズムと呼びます。例えば、あるシミュレーションにおいて、非常に多くの試行が必要であるが、1 つ 1 つの試行には依存関係がなくそれ自体で完結する場合などを考えましょう。この場合、演算中に計算ノード間のやり取りがまったく起きないので、クラスタに適したアプリケーションとなります。金融のデリバティブ商品開発のためのリスクシミュレーションは、その具体的な例です。

1-2-3 共有メモリ型

共有メモリ型システムは、すべての演算器が単一のアドレススペースを共有し、演算器間のデータ通信は共有メモリへの Read/Write で実現します。分散メモリ型と異なり、データの分配・収集の必要がないので、ソフトウェアの観点でみるととてもシンプルなシステムになります。

共有メモリの代表例は対称型マルチプロセッシング（SMP）システムです（**図 1.7** 左[注5]）。 PC 用の x86 CPU をマルチプロセッサ化する方法を定めた、Intel Multiprocessor Specification[注6] は、15 年も前の 1994 年にバージョン 1.0 がリリースされており、今では 2Way ワークステーション（搭載可能な CPU 数を「Way」という単位で表現します）は、極めて一般的なものになっています。しかし、構成上、プロセッサの数が増えるとメモリアクセス時の競合が多くなり、プロセッサと共有メモリ間の

注 5　http://www.software.intel.com/en-us/articles/optimizing-software-applications-for-numa/

注 6　Intel Multiprocessor Specification, http://www.intel.com/design/pentium/datashts/242016.htm

データ転送帯域幅が落ちるため、そこがボトルネックとなります。そのため、SMP システムではプロセッサの数に対して性能をスケールさせる、すなわち数に見合った性能向上を得ることは難しく、あまり多くのプロセッサを搭載することはできません。 2Way サーバは低価格で、広く普及していますが、32Way や 64Way のような多数の CPU を持つ SMP サーバでは特別なハードウェアが必要となり、一般的に非常に高価になります。

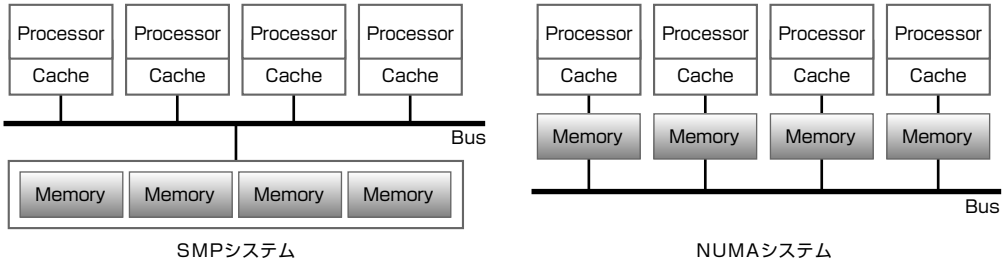

図 1.7 : SMP と NUMA

共有メモリ型のもう 1 つの代表的なシステム構成が、NUMA（Non-uniform memory access）システムです。 NUMA システムも、システムのメモリの全領域にすべてのプロセッサからアクセスできる共有メモリ型の一種です。しかし、あるプロセッサから見た場合に、自分に近いメモリ（ローカルメモリ）と遠いメモリ（リモートメモリ）の区別が存在する点が SMP と異なります（**図 1.7** 右）。 NUMA の場合、多数のプロセッサを搭載した場合にも、メモリバンド幅がボトルネックになりにくいのが特徴です。しかし、共有メモリへのアクセスコストは均一でなく、リモートメモリへのアクセスはローカルメモリへのアクセスに比べて遅くなります。アクセスコストの均一性を高めるためにそれぞれのプロセッサにキャッシュを用意し、それぞれのキャッシュのデータの（プロセッサ同士で食い違いのないように）整合性を保つために特別なハードウェアを持たせたのが、cc-NUMA（Cache coherent NUMA）です。

サーバ用 CPU である AMD Opteron や、Intel Xeon 5500 番台は、CPU チップ内部にメモリコントローラーを持っているため、複数の CPU 構成をとった場合には必然的に NUMA アーキテクチャになります。そのため、これらの CPU にはキャッシュコヒーレンシーを保つためのハードウェアが組み込まれています。また、従来までの CPU では、Front Side Bus（FSB）と呼ばれるポートに複数の CPU やチップセットがバス接続されていましたが、NUMA アーキテクチャでは CPU 間の通信速度をより高速に保つ必要があるために、FSB が廃止され、Point-to-Point 接続のチップ間通信ポート（インターコネクトポート）が設けられています。これらの高速インターコネクトポートは、Intel では Quick Path Interconnect（QPI）、AMD では HyperTransport という名称で呼ばれています。

このように、SMP、NUMA という概念を学んだ上で、現在主流の x86 系サーバ製品を見てみると、

なかなか興味深いことがわかります。**図 1.8**[注7]を見るとわかるように、Dual Core、Quad Core と呼ばれるマルチコアプロセッサの中身は複数のプロセッサ・コアが対称に並んでいるため、SMP 構成といえます。そして、これらマルチコアプロセッサ同士を複数結んだシステム全体は、NUMA 構成となっています。つまり、現在の 2Way 以上の x86 サーバ製品は、「SMP を NUMA でつないだシステム」ということができます。

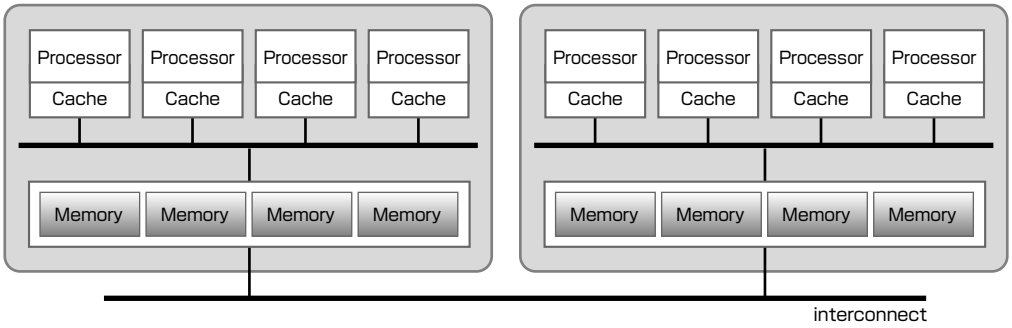

図 1.8：現在の 2Way x86 サーバの典型例

1-2-4 アクセラレータ

前節まで見てきたさまざまな並列処理システムは、いずれも汎用 CPU をなんらかの方法で複数個接続したものでした。 CPU1 個では処理が間に合わない問題に対して複数個の CPU で対応する、というのは素直なアプローチです。しかし、もう 1 つの考え方として、メインの CPU とは別にある特定の演算のみを汎用 CPU よりはるかに高速で処理できるようなハードウェアに任せるという方法があります。このような付加的なハードウェアを「アクセラレータ」と呼びます（**図 1.9**）。

アクセラレータの例としては、Cell Broadband Engine（Cell/B.E.）や GPU（Graphics Processing Unit）、DSP（Digital Signal Processor）などが挙げられます。

注 7　http://software.intel.com/en-us/articles/optimizing-software-applications-for-numa/

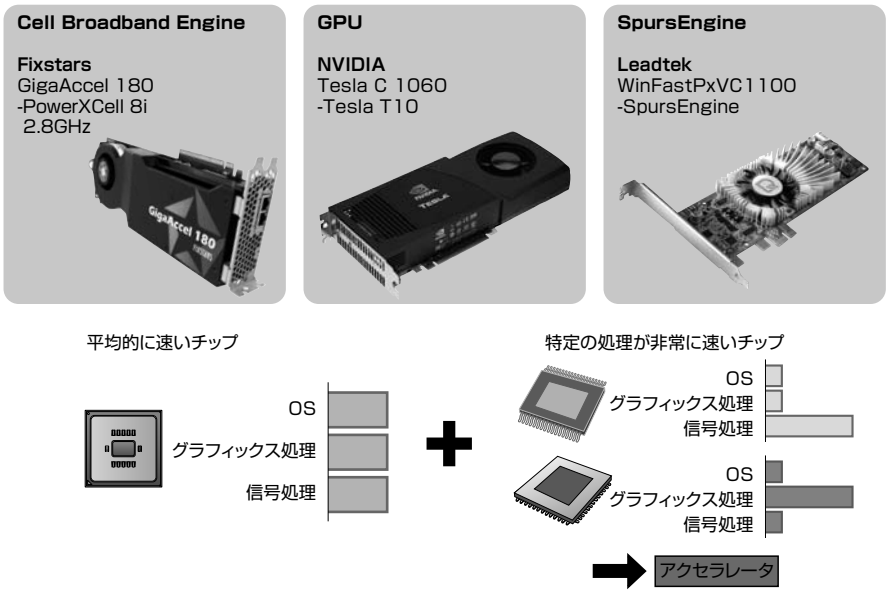

図 1.9：アクセラレータの例

これらアクセラレータの共通の特徴としては、複雑な制御系の機構を実装せず、浮動小数点演算（DSP の場合は固定小数点のチップもあります）性能に特化した単純な構成の演算コアを持っているということです。また、単純であるがために、物理的に演算コア 1 つがチップ上で占める面積が小さく、1 チップに多数の演算コアを詰め込むことができます。

例えば Cell/B.E. では、PowerPC Processor Element（PPE）と呼ばれる、頻繁なスレッド切り替えが必要な処理の動作に適した汎用のプロセッサコア 1 基と、Synergistic Processor Element（SPE）と呼ばれる、シンプルで小型な、浮動小数点演算を高速に処理することに特化したプロセッサコア 8 基の、合計 9 つの演算コアが Element Interconnect Bus(EIB)と呼ばれる高速なバスで接続されて 1 チップに実装されています（**図 1.10**）。

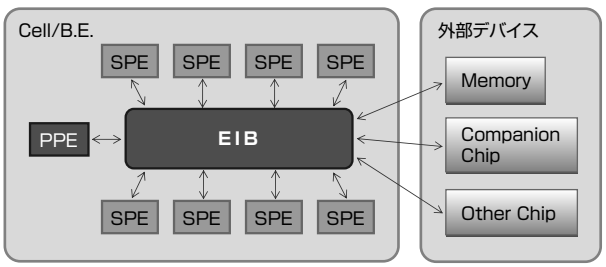

図 1.10：Cell Broadband Engine の概要

NVIDIA の GPU チップ Tesla T10 では、Scalar Processor と呼ばれる最小単位の演算コアが 8 個集まって Streaming Multiprocessor（SM）と呼ばれる単位の機構を形成し、その SM が 1 チップに 30 セットあるので、合計 240 コアが 1 チップ上に実装されています（**図 1.11**）。

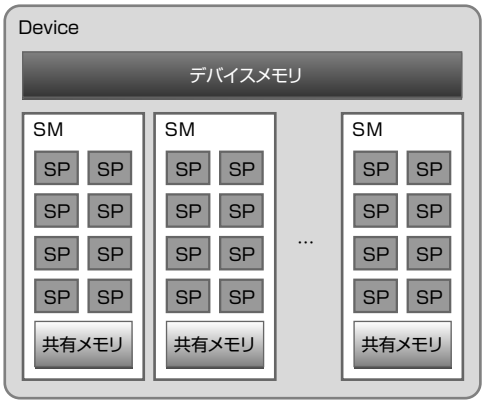

図 1.11：NVIDIA Tesla T10 概要

近年、このアクセラレータが大変注目を浴びてきています。その理由としては、汎用 CPU の浮動小数点演算能力の伸びが鈍化し、数 10GFLOPS 程度に留まっている一方で、Cell/B.E. や GPU など、100GFLOPS ～ 1TFLOPS を誇る浮動小数点演算性能に特化したチップが、PCI Express 接続の演算処理ボードとして安価に提供されるようになったためでしょう。また、ここにきて「グリーン」という観点から、工場や研究所で使われるような製品であっても低消費電力を求められるようになり、数で勝負するタイプのクラスタサーバシステムにとっては厳しい外部環境となってきているのも遠因といえそうです。

例えば電子基板や半導体の検査装置では、工場ライン全体の生産スピードを維持したまま、年々微細かつ複雑になっていく検査対象に対応するために、より迅速な画像処理が必要になっています。また、超音波診断装置や CT スキャンなどの医療画像機器分野においても、検査装置から得られる 2D 画像は年々高精細になっており、汎用 CPU ではこれらの高精細な画像に対して、実用的な時間内に処理を終えることができません。 このような要求を実現するために、クラスタサーバを用いてしまうと、どうしても大きな空間が必要になり、消費電力も大きくなってしまいます。据え置き型の機器であれば問題ないのかもしれませんが、ある程度の可搬性を保とうとすると、クラスタサーバでは難しくなります。そこで GPU や Cell/B.E. のような演算処理能力に特化したデバイスをアクセラレータとして搭載し、制御用の汎用 CPU と組み合わせて「ハイブリッドシステム」を構築すると、コンパクトで処理能力の高いシステムを実現できます（**図 1.12**）。