サイエンティフィック・システム研究会　　2008 年 8 月 26 日
システム技術分科会 2008 年度第 1 回会合資料

マルチコア化 CPU の傾向と対策

㈱富士通研究所 IT システム研究所
久門耕一

1. 初めに

マルチコアプロセッサについては、2000 年前後の頃から、将来は、同一構造のプロセッサ(コア) を複数搭載して並列処理を行うことが主流になるだろうと言われてきた。それまで、LSI の微細化が進み、チップ上に大量のトランジスタが搭載可能になると、そのトランジスタ数の増加を用いて命令レベルの並列実行を行う複雑な設計により性能向上を図ってきた。しかし、搭載トランジスタ数が増えるにつれ設計工数は指数的に増大し続ける。このため、チップに搭載可能なトランジスタを有効活用するために同一構造のコアを複数搭載するマルチコア構造をとらざるを得なくなるという危惧からである。

既に IBM は、1999 年に発表した Power4 プロセッサでは、1 チップに 2 つのプロセッサを搭載したデュアルコア構成を取っていた。しかし、マルチコアプロセッサが広く一般に普及するようになったのは、2006 年に Intel 社と AMD 社によって、ほぼ同時に発表されたデュアルコア CPU からである。この時点から、コモディティ CPU である X86 プロセッサが率先してマルチコア化へと一気に進んできた。

マルチコアプロセッサは、従来の SMP(Symmetrical Multi Processors)システムを 1 つのチップ上で実現する事を目標として開発された。このため、SMP とマルチコアプロセッサに本質的な差はない。但し、一般に 1 チップ上に複数のコアが搭載される実装形態から、SMP システムとは異なる特徴を持つのではないかと想像される。

本稿ではマルチコアプロセッサの特徴およびプログラミング上の留意点、更に汎用 CPU と全く違うアプローチでチップ面積の増大を演算性能に繋げた GPU のアーキテクチャについて概説する。

2. 単体 CPU 高性能化からマルチコアへの歴史

2.1.高周波数=高性能の時代から、クロック至上の終焉

良く知られているように、プロセッサの性能は 1960 年代から 40 年以上に渡り、Moore の法則に従って向上してきた。基本的に、プロセッサの性能向上は、何らかの方法で、単位時間当りの処理量を増やすしかない。つまり、プロセッサの性能向上は、1 クロックあたりの処理量の増加と、クロック周波数の向上の双方で行われる。

PC 向け汎用 CPU である X86 プロセッサでは、Intel は主に性能向上をクロック高周波数化で実現してきた。そのために、高周波数化が容易なスーパーパイプライン構造を採用する Netburst アーキテクチャを開発し、高周波数化に遅れをとる AMD 社の CPU との差

異を明確化してきた。すなわち「性能は周波数で決まる」事をアピールしてきた。

Netburst アーキテクチャでは、2000 年に第一世代の Pentium4 が当初 1.5GHz 程度で(開発コード名 Willamet) 出荷されたが、2004 年に発売された第三世代の開発コード名 Presscot では、最高 3.8GHz に達し、最終目標では、10GHz を目指したと言われている。

しかし、Netburst アーキテクチャは、クロック周波数で性能向上を目指したものの、二次キャッシュのアクセスレイテンシが遅かった事と、パイプラインの細分化のため、分岐実行時でのパイプライン実行効率の大幅低下のため、現実の問題では、消費電力が増大するわりに性能向上せず、熱くて遅いと酷評されるに至った。2006 年に Netburst アーキテクチャベースのマルチコア CPU Xeon として Dempsey が出荷されたが、２コアにしたことで消費電力が極めて高くなり、その割に性能が高くなかったことから、殆んど使われること無く、新たに設計された高効率の Core2 アーキテクチャに移行した。

Core2 は、クロック当りの処理性能が高い。Core2 アーキテクチャの２コアプロセッサ、Woodcrest では、クロック周波数が 3GHz と低くなったにも拘らず、クロック当たり実行効率が Netburst アーキテクチャの 1.5 倍ほどあった。このため、Netburst アーキテクチャの 3.8GHz の CPU よりも高速で、且つ、2/3 から 1/2 程度の消費電力であった(図 1)。インテルが発表している資料では、Netburst アーキテクチャである Pentium4(Cedarmill)と、Core2 ベースの Core Duo(Yonaha)では、同等の性能の場合 3 倍程度の電力差があるとされる。

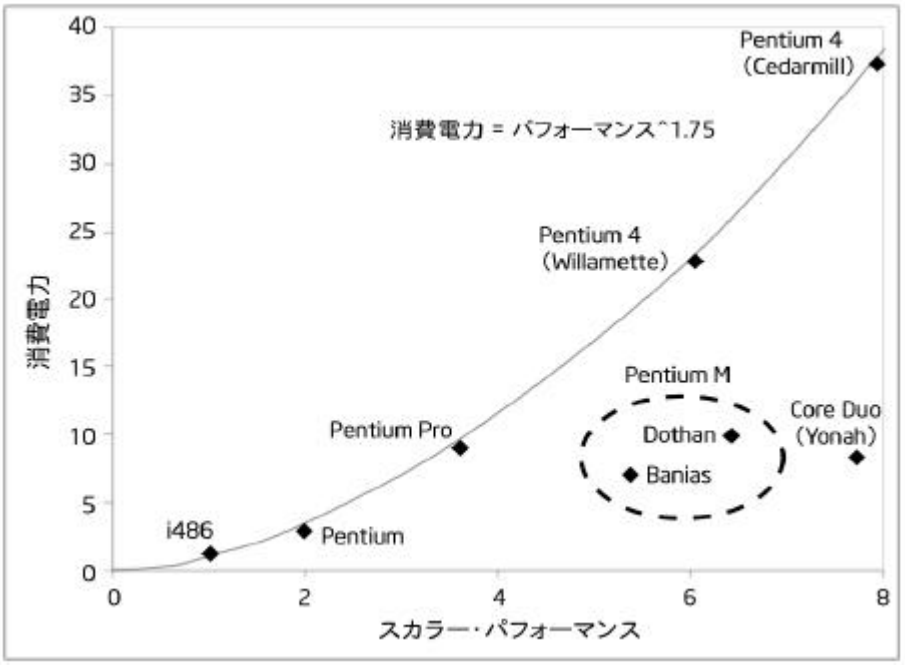

図 1.正規化された Intel 製 CPU の正規化消費電力と正規化性能[1]

X86 を開発するもう一つの会社である AMD が 2003 年に販売を開始した Opteron は、Intel の CPU に比べてコア周波数が低く、最高でも 2GHz 程度であり、また、キャッシュ容量も小さい。しかし、メモリを CPU に直結する、いわゆる「ダイレクトコネクトアーキテクチャ」を採用した事で、高メモリバンド幅と短いメモリアクセスレイテンシを特徴とし、クロック周波数が高い Netburst アーキテクチャを越える性能が得られていた。

2.2. マルチコア化へ

コモディティ商品としては、消費電力が高くなると冷却が困難になるため、実装可能な電力に上限がある。無暗とコア周波数を上げることが困難である以上、性能は 1 クロックで実行できる処理を増やすしかなくなる。そのための方法として、

1. アーキテクチャ改良により、命令の実行効率を向上させる
2. スーパスカラ構造により、命令レベルの並列度を上げる
3. SMP 構造により、複数の命令ストリームを実行する

等が考えられる。

従来の CPU チップ性能向上は、1)と 2)で行われてきた。しかし、ある程度を越えると、チップ面積増大に比べ性能向上が困難になってくる。これを表わした経験則が Intel の Fred Pollack が発見したとされる、Pollack の法則「チップの性能は、チップ面積の平方根に比例する」である。

Pollack の経験則によれば、チップ上に 2 倍のトランジスタを搭載した場合、性能向上は 1.4 倍に留まる一方、電力は 2 倍消費し、電力あたりの性能は劣化する。そのため、CPU として冷却可能な電力消費に押えるためには、コアの回路規模小さくして電力消費を押さえたまま、チップ上に N 個の CPU を搭載し、あわよくば N 倍の性能を得ようとするマルチコア CPU が優勢になってきた。さらに、複雑なチップは設計が指数的に困難になるため、単純なコアを複数並べる方式の方がコストが安くなると言う事情もある。

2006 年に、Intel 社と AMD 社は、ほぼ同時期にデュアルコアプロセッサを発表した。それぞれ Core2 アーキテクチャの Woodcrest と Rev.F Opteron と呼ばれる。

更に同年、Intel は 1 つのパッケージに 2 つのチップを搭載した 2x2 コアプロセッサ、Clovertown を出荷、その後デザインルールを 45nm にシュリンクし、キャッシュサイズを増大させた Harpertown を出荷している。基本的には、Dempsey から Clovertown までのバス構造は変らず、CPU とチップセットを接続する FSB (Front Side Bus)の周波数向上により、メモリバンド幅を増強しただけである。

他方、AMD が 2003 年に販売を開始した Opteron は、いわゆる「ダイレクトコネクトアーキテクチャ」を採用する事で、高メモリバンド幅と短いメモリアクセスレイテンシを特徴とし、クロック周波数が高い Netburst アーキテクチャを越える性能が得られていた。第二世代にあたる Opteron は Revision F と呼ばれ、２コア化している。更に、2007 年後半には、Intel とは異なり、1 チップで 4 コアを実現するマルチコアプロセッサ Opteron９０

（Barcelona）を発表している。図 2,3,4 に、IntelXeon の 4 コアプロセッサ Harpertown、AMD Opteron の 4 コアプロセッサ Barcelona、AMD Power6 の 2 コアプロセッサのシステムアーキテクチャを示す。

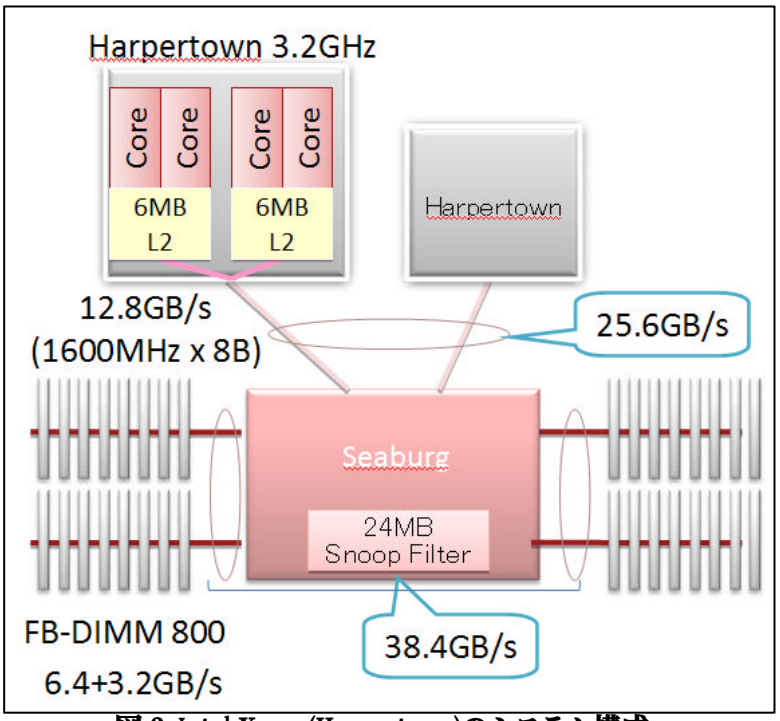

図 2. Intel Xeon (Harpertown)のシステム構成

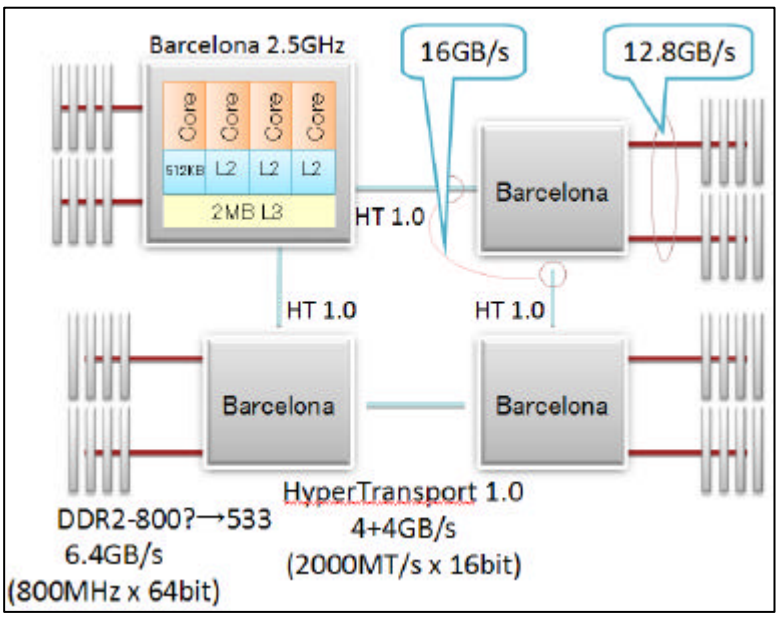

図 3. AMD Opteron (Barcelona)のシステム構成

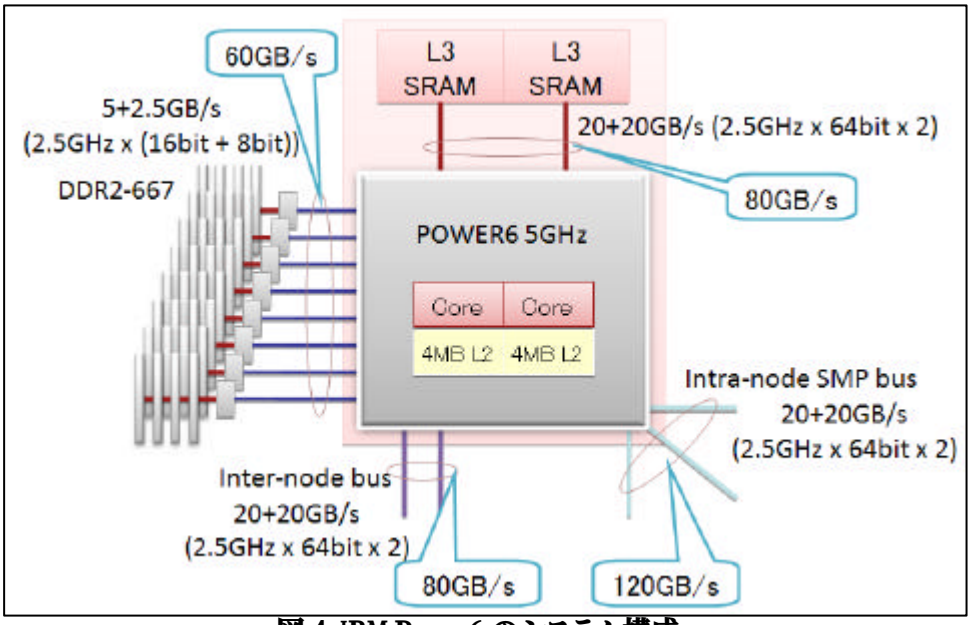

図 4. IBM Power6 のシステム構成

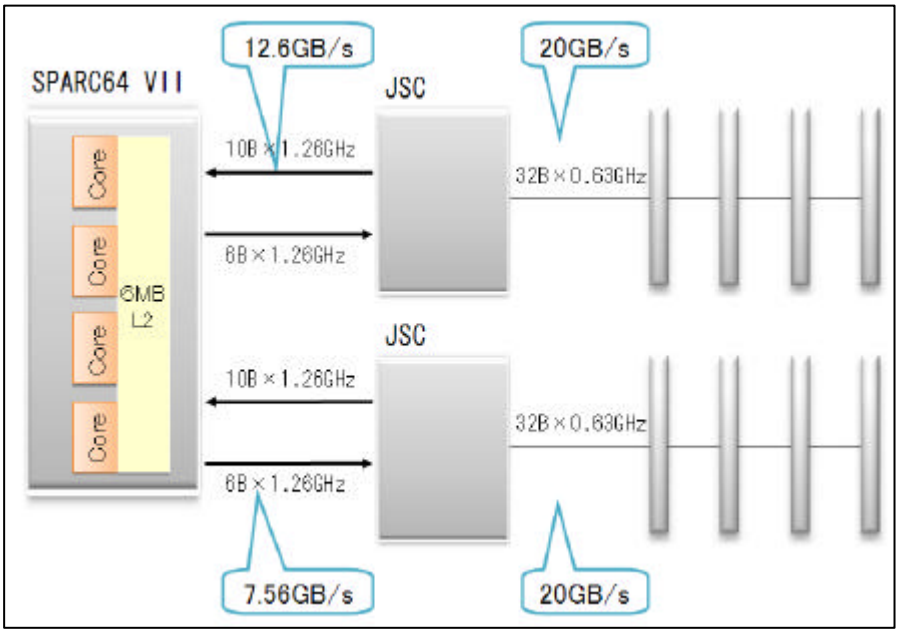

図 5. Fujitsu FX-1 のシステム構成

3. マルチコア CPU の特徴

元々、マルチコア CPU は、従来複数の CPU チップを使って作られていた SMP と同様に使用する事ができ、基本的に SMP と同様の特性を示す。

ただ、マルチ CPU で構成される SMP に比べて、マルチコア CPU の方は、チップ内の複数コアで共有するキャッシュを設けることが多く、双方の CPU で共有するデータに対するアクセスレイテンシは短く、コア間でのデータ授受のためのバンド幅が高く取れる可能

性がある。

一般に、システムの性能は、

- メモリから CPU へのデータ供給能力
- CPU でのデータ処理能力

のどちらかで律速される。

既存のシングルコア CPU とソケット互換のマルチコア CPU を作った場合、CPU 性能は N 倍になるが、メモリ系のデータ供給能力は増えないため、データ供給能力がシステム性能を律速する可能性がある。この事は、メモリバンド幅が AMD 系 CPU に比べて低い Intel 系の CPU の場合顕著となりえる。つまり、CPU とメモリとの間にチップセットが存在し、CPU からのアクセスレイテンシが長い Intel のプロセッサでは、キャッシュの有効活用はマルチコア性能を引き出すためには必須である。

3.1. マルチコア CPU のメモリバンド幅

この様に、メモリバンド幅はマルチコア CPU の性能を律速しかねないため、バンド幅を実測した結果を表 1 に示す。この表において、それぞれの CPU はコア周波数が異なるため、値の単純比較はできない。コア周波数の違いを補正するため、コアの演算能力あたりのバンド幅、Byte/Flop を算出したのが表 2 である。

表 1. マルチコア CPU のメモリバンド幅

	Barcelona(2.3G)			Harpertown(3G)			Clovertown(2.3G)		
	$size (B)	/core (GB/s)	/sys (GB/s)	$size (B)	/core (GB/s)	/sys (GB/s)	$size (B)	/core (GB/s)	/sys (GB/s)
L1D	64K	63.5	507	32K	48	384	32K	37	296
L2(独立)	512K	18	145		(none)			(none)	
共有$	2M/4	6.1	48.9	6M/2	20	160	4M/2	16	124
主記憶	—	2.3	18.2	—	0.8	6.4	—	0.8	6.3

表 2. マルチコア CPU のメモリバンド幅

	Barcelona(2.3G)		Harpertown(3G)		Clovertown(2.3G)	
	$size (B)	/core (B/F)	$size (B)	/core (B/F)	$size (B)	/core (B/F)
L1D	64K	6.9	32K	4	32K	4
L2(独立)	512K	2.0	(none)		(none)	
共有$	2M/4	0.66	6M/2	1.67	4M/2	1.74
主記憶	—	0.25	—	0.066	—	0.087

これにより、マルチコア CPU でのプログラミングの留意点が分かる。

1. L1D キャッシュのバンド幅

Harpertown では 40GB/s (4B/F)、Clovertown で 37.3GB/s (4B/F)、Barcelona では 63.5GB/s (6.9B/F)である。実アプリケーションでは、Intel の Harpertown, Clovertown

が提供する 4B/F 程度のバンド幅があれば十分であり、Barcelona が提供する L1D キャッシュのバンド幅がフルに必要とされるアプリケーションは殆んどない。いずれにせよ L1D キャッシュの有効利用は高性能を目指すためにはとても有効である。

2. 共有キャッシュへのバンド幅

Harpertown と Clovertown では 1.7B/F のバンド幅を持つが、Barceloa はチップ内キャッシュであるにも拘らず、0.66B/F と大幅に低下する。Barcelona の値は、チップ内キャッシュへのバンド幅が主記憶のバンド幅の 3 倍程度しかない事を示す。ちなみに、Barcelona の値は、2 コア CPU である RevF Opteron に比べて悪く、また同じ Barcelona 間の比較では、2 ソケットのシステムよりも 4 ソケットのシステムの方が低くなる。Barcelona においてコア数が増えた時にバンド幅が悪化するのは、Opteron の共有キャッシュの一貫性制御が Core2 と異なり L3 キャッシュが L2 キャッシュのデータを包含しない、non-inclusive キャッシュを採用しているためと見らる。Non-inclusive キャッシュのため、共有キャッシュアクセスを行うと、各コアの L2 キャッシュのデータをチェック(スヌープ)するために、マルチコア数分の snoop が出ると考えられる。しかし、AMD はキャッシュ一貫性制御に関する詳細情報を公開しておらず詳細は不明である。

3. 主記憶へのバンド幅

Barcelona では、ダイレクトコネクトアーキテクチャの採用により、Harpertown/Clovertown の約 3 倍のバンド幅が得られており、 主記憶バンド幅が律速するアプリケーションの場合、Barcelona の方が約 3 倍高速に演算出来る余地がある。Core2 の場合、共有 L2 へのバンド幅に比べ、メモリバンド幅は 1/30 程度しかなく、L2 にヒットしなかった場合には極めて遅くなる。

以上のことから、同じマルチコア CPU である Barcelona と Harpertown/Clovertown では性能の傾向がかなり異なる。ただ、どちらの場合でも、主記憶アクセスが多発するアプリケーションでは、性能は演算器性能ではなくメモリバンド幅で律速される。以上をまとめると次のようになる。

Barcelona　極めて高いバンド幅を持つ L1 キャッシュあるいは、L2 キャッシュまでを活用することが重要。つまり、ワーキングセットが 512KB 以内に収まることが望ましい。共有キャッシュである L3 キャッシュが必要とされるワーキングセットでは、余り高いバンド幅は期待できない。この事は、CPU 間でのキャッシュ間データのバンド幅が高くないことも示す。

Core2　Barcelona と同様に L2 キャッシュまでは高いバンド幅を維持し、L2 キャッシュは 6MB の共有キャッシュ（Harpertown）であるため、かなり大きなワーキングセットでも高いバンド幅を維持し続ける。一方、主記憶アクセス時のバンド幅が極端に低いことから、on cache と off cache の挙動の変化は大きく、アプリケーション性能がデータによって大きく変動する原因となる。

このように、同じマルチコア CPU であってもその設計によって全く異なる性質を示す。共有キャッシュを持つマルチコア CPU では、チップ内のコア間データ共有が高速に出来ると考えられる。次に、実際にコア間の同期オーバヘッドを計測することで、マルチコア性能を検証する。

3.2. マルチコア CPU のデータアクセスレイテンシ

一般にマルチコア CPU では、コア間でのデータ共有を目的とした共有キャッシュが用意されている事が多い。コア間のデータ共有にかかる同期オーバヘッドを、複数のスレッドで同期を取るベンチマークプログラムにより計測する。

Intel の場合、L2 キャッシュは 2 コア間で共有される共有キャッシュである。Harperdown/Clovertown には、1 パッケージに 2 チップ入っているため、チップ内アクセスレイテンシとチップ間アクセスレイテンシも計測する。

計測に際しては、同期ライブラリは自作している。マルチスレッドライブラリとして広く使われる pthread ライブラリでは、同期の待ちが必要ない場合には高速であるが、同期処理で待ちが必要になる場合には OS に制御を渡し、10us 程度と大きなオーバヘッドが発生するためである。

表 3. スレッド間同期オーバヘッド

	Barcelona (us)	Harpertown (us)	Clovertown (us)	(Phenom) (us)
チップ内同期	0.59	0.16	0.20	0.37
ソケット内同期	—	0.30	0.49	—
ソケット間同期	0.66	0.57	0.70	—
ソケット間同期 (HT 2-hop)	0.85	—	—	—

この結果を見ると、Barcelona では、チップ内、ソケット間同期のいずれも Core2 のチップ間同期と同程度に遅く、コア内だから高速であるとは言えない。Opteron では Hyper Transport 接続により 4 ソケットシステムを構成可能であるが、Hyper Transport 上で 2-hop 離れた位置の CPU では、0.2us 程度同期に時間が余分に掛る事が分かる。

Intel の Core2 では、チップ内の L2 で共有されるデータが一番速いが、チップが異なっ

ても急激に遅くなるわけではない。Core2 がキャッシュをミスヒットした時のバンド幅が急速に低下する事と比較すると、想像以上に速い。実際には、異なったソケット間は、CPU と CPU の間にチップセットが存在して接続されるため、往復で 4 回のチップクロスを伴うが、実測結果から見るかぎりチップをまたぐ事自体のオーバヘッドは小さい事が分かる。

最近の InfiniBand 接続の PC クラスタにおいて、ノード間 MPI 通信が 1-2us 程度で動作する事と比較すると、マルチコア内、あるいは SMP 内部の通信遅延は 1/2-1/5 倍程度の遅延時間となっており、1 桁も速くないことが分かる。

この事から、マルチコアでの並列実行を考える上で、多重ループの最内ループをスレッドに割当て、ループ間の実行制御を同期で行うような実行制御を考えると、最内ループの処理量がある程度大きくない場合、同期オーバヘッドが大きく、性能向上に繋がらない事が想像される。

3.3. SMT とマルチコア

マルチコアは、一つのチップ内に複数のコアを搭載する技術であるが、関連技術として SMT (Simultanious Multi Threading) がある。

表 4 に、SMT、マルチコア、SMP の比較を載せる。

表 4. SMT とマルチコアの比較

	レジスタ	演算器	パイプライン	L1 キャッシュ	共有キャッシュ	メモリ
SMT	**独立**	共有	共有	共有	共有	共有
マルチコア	**独立**	**独立**	**独立**	**独立**	共有	共有
SMP	**独立**	**独立**	**独立**	**独立**	**独立**	共有

SMT は、CPU コア内に複数のスレッド環境を用意し、メモリアクセスのように長時間パイプラインや演算器を遊ばせる操作を行ったとき、別のスレッドに切替えを行ってコアの稼働率を高め、結果的にシステムの処理性能を向上させる技術である。

Pentium4 など、SMT の一種である Hyper Threading が利用できる CPU においては、ビジネス系のアプリケーションで 5-20%のスループット向上が得られると言われる。逆に HPC 系のアプリケーションにおいては、性能低下が起きるケースが多い。

HPC アプリで SMT を使うと性能が劣化する原因は、HPC アプリでは SMT において共有している演算器、L2 キャッシュやメモリのバンド幅がアプリケーションの実行性能を律速しており、性能の上限が非 SMT 使用の場合と同程度に押えられることと、複数のスレッドが同時に動く場合、データのワーキングセットが拡大し、キャッシュがヒットしにくくなることである。

同様の性能劣化は、SMP やマルチコアでも発生するが、SMP サーバでは、外部バスまでは独立なデータパスが存在し、更にその先も共有バスのような複数チップで共有されるリソースアーキテクチャを取っておらず、外部バスもクロスバスイッチなどによりノード数

に対してスケーラブルなバンド幅を確保できるよう設計し、性能劣化を防ぐのが一般的である。

一方、マルチコア CPU の場合、L２キャッシュ以降のデータパスが共有される。このため、L2 以降の共有キャッシュのバンド幅や外部バスのバンド幅が十分に確保できない場合、それぞれのスレッドでキャッシュミスが多発すると SMT のように性能が劣化する。

4. まとめ

HPC 用途では、マルチコア採用以前にメモリバンド幅が枯渇しがちであることから、単純にマルチコア化するだけでは、性能向上が期待できず、キャッシュチューニングなどバス負荷を低減させる最適化が必須である。

チップ内の共有キャッシュに対するアクセスであっても、十分高速にアクセスできるわけではないため、極度に小さな単位での並列処理は性能の低下を招く。

HPC アプリとは異なり、いわゆるビジネスアプリケーションでは、バス負荷が性能を律速するケースが少ないため、コア数の増加に従って性能が出しやすい。しかし、コア数が更に増加する場合、メモリバンド幅がネックとなり性能が律速することが予想される。この点については、現在は共有バスアーキテクチャを採用する Intel も次期の CPU(Nehalem) では、メモリバンド幅を向上させやすいダイレクトコネクトアーキテクチャを採用すると発表しており、今後のシステムアーキテクチャは高メモリバンド幅が確保できる方向へとシフトしていくため、現在よりは性能向上を図りやすくなるが、コア数の増加に見合うだけの性能向上を得るためには、十分なキャッシュチューニングが今後とも必要となると考えられる。

Appendix. 汎用プロセッサから GPGPU 利用へ？

汎用 CPU を HPC 計算に利用するようになったのは、汎用 CPU の急激な性能向上を利用して廉価に HPC 計算を行うためであった。しかし、汎用 CPU では、コア性能の向上に見合うほどのメモリバンド幅の向上を図らなかったため、性能がメモリバンド幅ネックとなることを説明した。汎用 CPU 設計では、この問題に対応するため、キャッシュサイズの増大させることにより、バンド幅ネックを解消しようとしてきた。しかし、HPC アプリケーションのように、キャッシュヒットが難しい場合には、必ずしも期待通りの性能が得られていない。

汎用 CPU のマルチコア CPU では、バスや命令セット、メモリアーキテクチャの互換性を保ったまま多くの演算要素を投入したのに対し、GPGPU(General Purpose Graphic Processing Unit)は、全く異なった考え方で演算能力を増大させてきた。以下に述べるように、GPGPU は、キャッシュを有効活用するためのデータアクセスの局所化が困難なアプリケーションに対して、有効な計算手段となり得る。

GPGPU は並列処理に向いた画像処理を高速に実行する目的で、高いメモリバンド幅、極

めて多くのマルチスレッドCPUと演算器による高スループット演算を実現している。更に、最近のGPGPUは、ゲームやCGの処理が高度化し、物理法則に従った物体の運動シミュレーションでリアリティを増大させる事を目的に、演算高速化を図ってきた。この結果、科学計算に必要な演算能力を持つまでに至っている。

現在、最新のGPGPUは単精度ではあるが、カード一枚でほぼ1TGlopsの速度を達成し、汎用CPUのチップあたりの演算性能の10倍である。一方、倍精度演算になると性能が大幅に低下し、nVidiaのGTX-280で77.8GFlops程度となる。メモリバンド幅に関しても、nVidia社のGTX-280ではグラフィックカード上のメモリへの理論メモリバンド幅は141GB/s、memcpy相当のプログラムによる実測でも120GB/s程度が達成できている。これは理論値の85%以上である。このメモリバンド幅を、表1の汎用システムのメモリバンド幅と比較すると6倍大きく、ほぼ汎用プロセッサのL2相当のバンド幅を持っている。汎用CPUでは演算に必要なデータバンド幅をキャッシュによるバンド幅増大で賄ってきた状況とは大きく異なる。残念ながら、汎用CPUのキャッシュとは異なり、GPGPUのメモリアクセスレイテンシは遅く、実測によれば、GTX-280のアクセスレイテンシはコアクロック換算で500クロック程度となっている。

GPGPUでは、アクセスレイテンシを隠蔽する手段として、コア内に大量のスレッド実行ユニットを搭載し、メモリからデータが返ってくる500クロックの間に他のスレッドに切り替えながら実行を継続する。それらのスレッド演算に必要なリソースはレジスタと言う形で持っており、GTX-280の場合、単精度レジスタが480K（=48万）個もある。容量にすると約2MBである。 実質的にこれが汎用プロセッサのキャッシュ相当の役割を果たす。

レジスタはプログラムからロードストアを制御する必要があるため、ベクトルプロセッサと同等のアプローチである、大量のスレッドによりメモリアクセスレイテンシを隠蔽する手法は、大量に独立に実行できる演算を必要とするが、アプリケーション上可能な場合には、汎用プロセッサでは困難なアプリケーションでも、高速実行が可能になるので、今後のHPC演算のCPUとして高い利用価値があると考えられる。

参考文献

[1]「インテル R マイクロプロセッサーにおける 1 命令実行当たりのエネルギーの推移」
Technology@Intel Magagine,インテルジャパン株式会社，2006年
http://download.intel.com/jp/developer/jpdoc/energy-per-instruction-0306_j.pdf