ホワイトペーパー

インテル® Xeon® プロセッサー

サーバー・パフォーマンス

技術分析

インテルのフロントサイド・バス・アーキテクチャーに関する比較分析

エンドユーザーに優れたパフォーマンスを提供するインテル® サーバー・プラットフォームの設計について

インテル® Xeon® プロセッサーおよび Itanium® 2 ベースのプラットフォームは、パラレルなバス・アーキテクチャーを使用して、プロセッサー間やチップセットとプロセッサー間を接続しています。このバスは、通常、フロントサイド・バスまたは FSB と呼ばれています。このところ、FSB アーキテクチャーは時代遅れのアーキテクチャーである、あるいはシステムのボトルネックであると批評されることが増えてきています。実際には、FSB は進化し続けている非常に能力の高いバス・アーキテクチャーです。本資料では、FSB を含むシステム・アーキテクチャーが、ユーザーのニーズを満たすべく、どのように進化し、また今後どのように進化していくかを解説します。本資料で提供されている情報は、主にインテル® Xeon® プロセッサー・ベースのプラットフォーム上で FSB を使用した場合について (HyperTransport* バスを使用する AMD のプラットフォームと比較して) 解説していますが、本資料中の解説の多くは、Itanium® ベースのプラットフォームにも当てはまります。

はじめに

インテル® サーバー・プラットフォーム・アーキテクチャーは、現代のコンピューター産業にとって欠かせない存在です。これまでに、非常に多くのインテル サーバーが導入され、現在も使用されています。これらのシステムでは、プロセッサーとシステムのその他の部分 (I/O、メモリーなど) を接続するバスを中央に持つ、プラットフォーム・アーキテクチャーを使用しています。このバスは、「フロントサイド・バス (FSB)」と一般に呼ばれています。FSB と共に、プラットフォーム・アーキテクチャー全体が堅牢な設計であるだけでなく、拡張や変更が容易な設計になっています。本資料は、サーバーユーザーの増加する要望に応えるべく、インテルがバスとプラットフォームをこれまでどのように進化させてきたかについて解説します。また、競合する システム・ソリューション (AMD) との比較についても解説します。本資料に含まれる内容は以下のとおりです。

- システムバスの比較: FSB と HyperTransport*
  - 技術的な違い
  - 帯域幅の違い
  - バス・トランザクションの違い
- FSB の進化
  - 周波数の増加によるバス帯域幅の増加
  - マルチ FSB によるプラットフォーム帯域幅の増加
- FSB トラフィック削減プラットフォーム機能
  - オンダイ・キャッシュ・サイズの増加
  - スヌープフィルター
  - インテル® I/O アクセラレーション・テクノロジー

定義

バスの定義

最初に、基本的なバスの定義から説明します。バスとは、コンピューターのある部分から別の部分にデータを転送する信号線の集合です。コンピューター内をデータが移動するための高速道路であると考えると良いでしょう。バスの「種類」を定義する主な属性は 5 つあります。

バス方向:

単方向: データは 一方向にのみ流れます (一方通行の道を想像してください)。単方向バスは通常、各方向に 1 つ、ペアで実装されます。

双方向: データは双方向に流れますが、通常は同時には流れません (進行方向が時間によって異なる一方通行の道を想像してください)。

データ形式:

エンコードパケット: アドレスとデータの両方をエンコードして一緒に送ります。受信側のエージェントでデコードする必要があります。

個別 (非エンコード): アドレスは、データとは別にバス上で送られます。データやアドレスをエンコードする必要はありません。

バス帯域幅:

バスを構成している信号線の数。信号線の数が多ければ多いほど、各クロックでより多くのデータを転送することができます。

FSB 周波数 / 転送レート:

新しいパケット情報が転送される頻度。多くの場合、バス周波数は実際のデータ転送レートよりも低くなります。より正確な値を表すため、メガヘルツ (MHz) ではなく、1 秒当たりに転送される数を 100 万単位で表した値 (MT/s) が使用されます。

バス・エージェントの数:

バスに接続されているコンポーネントの数。2 エージェントはポイント・ツー・ポイント・バス、3 つ以上のエージェントはマルチエージェント・バスです。

バス属性	インテル® Xeon® プロセッサー	インテル® Itanium® 2 プロセッサー
バス方向:	双方向	双方向
データ形式:	個別 (アドレスとデータで別)	個別 (アドレスとデータで別)
バス帯域幅:	アドレス: 36-40 ビット データ: 64 ビット (ECC を含めると 72)	アドレス: 50 ビット データ: 128 ビット (ECC を含めると 144)
バス周波数 (転送レート):	アドレス: 200 – 667MT/s データ: 400 – 1333MT/s (実装に依存)	アドレス: 400 – 800MT/s データ: 400-800MT/s(実装に依存)
バス・エージェントの最大数:	2、3 、または 5 (実装に依存)	3 または 5 (実装に依存)

表 1: インテル® プロセッサーの FSB 実装の詳細

FSB の説明

前述の定義に関して、インテル® プロセッサーにおける FSB 実装の詳細を表 1 に示します。この表からわかるように、FSB の一部の属性は特定の実装に依存しており、時が経つにつれて変わっていきます。このような変化に対応できることは、FSB アーキテクチャーの利点の 1 つです。

HyperTransport* の説明

比較のため、HyperTransport* バスが AMD Opteron* プロセッサー上でどのように実装されているかを次に示します。

バス属性	HyperTransport*
バス方向:	単方向
データ形式:	エンコードパケット
バス帯域幅:	16 ビット (各方向)
バス周波数:	1600 – 2000MT/s (実装に依存、周波数が徐々に高くなると予測)
バス・エージェントの最大数:	2

表 2: HyperTransport* バス実装の詳細

FSB の進化: プラットフォームのニーズに対応

FSB がどのように進化してきたか、最初に FSB の周波数から解説します。

FSB 周波数 / 転送レート

プラットフォームで要求される帯域幅の増加に伴い、FSB 周波数 (転送レート) も高くなってきました。次の図から、FSB の周波数 (転送レート)、つまり帯域幅は過去 4 年間で 3 倍以上にもなっていることがわかります。一方、CPU コアの周波数は 2 倍にも達していません ('02 年前半は 2.2GHz で、'06 年前半は 3.8GHz)。

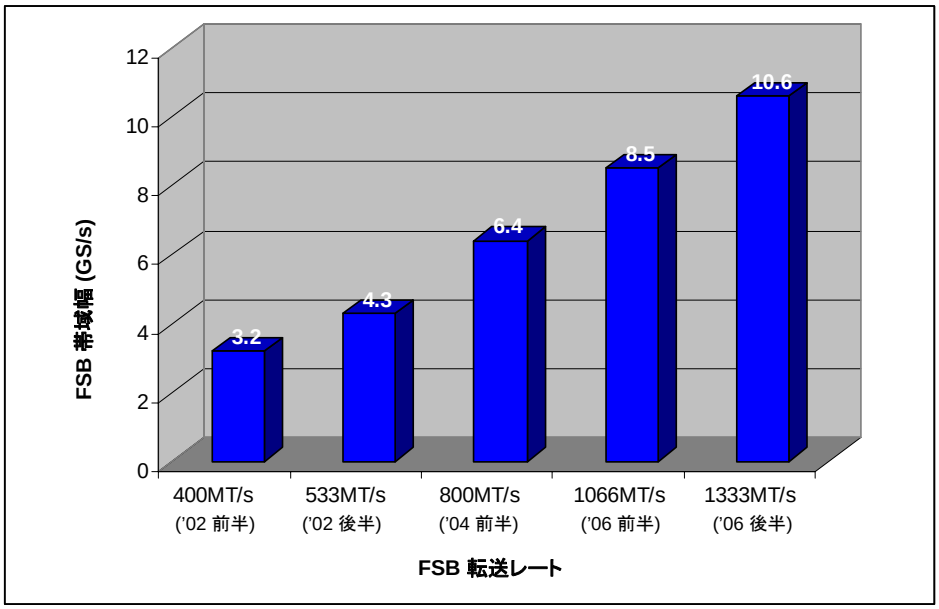

図 1: FSB 帯域幅の増加

プロセッサーへのシステム帯域幅

最初は、システムには 1 つの FSB しかありませんでしたが、プラットフォームの性能を全体的に向上させるため、FSB の周波数を高くするだけではなく、複数の FSB が利用できるようにプラットフォーム・アーキテクチャーも進化してきました。2006 年 7 月頃から、インテルのプラットフォームは、プロセッサーごとに 1 つの FSB が装備されはじめる予定です。これにより、プロセッサーからシステム全体への帯域幅もさらに増加します。図 2 および図 3 は、2002 年以降、プロセッサーごと、およびプラットフォーム全体で利用できる FSB の帯域幅が大幅に増加していることを示しています。2007 年には、MP プラットフォーム (4 プロセッサー) は、プラットフォーム・レベルと各 CPU レベルの両方で FSB 帯域幅が 10 倍に増加する予定です。DP プラットフォーム (2 プロセッサー) は、これよりも短期間で、ほぼ 7 倍に増加することがわかります。

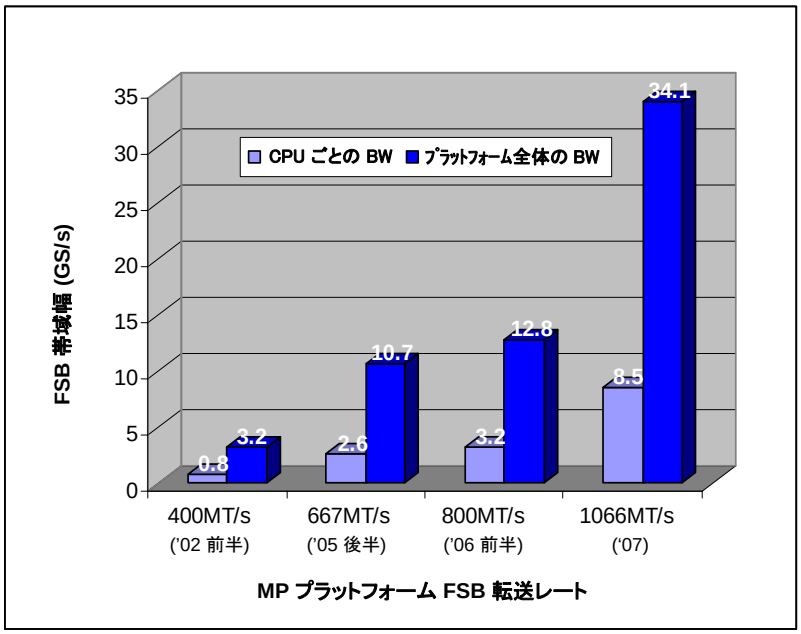

図 2: MP プラットフォーム帯域幅の増加

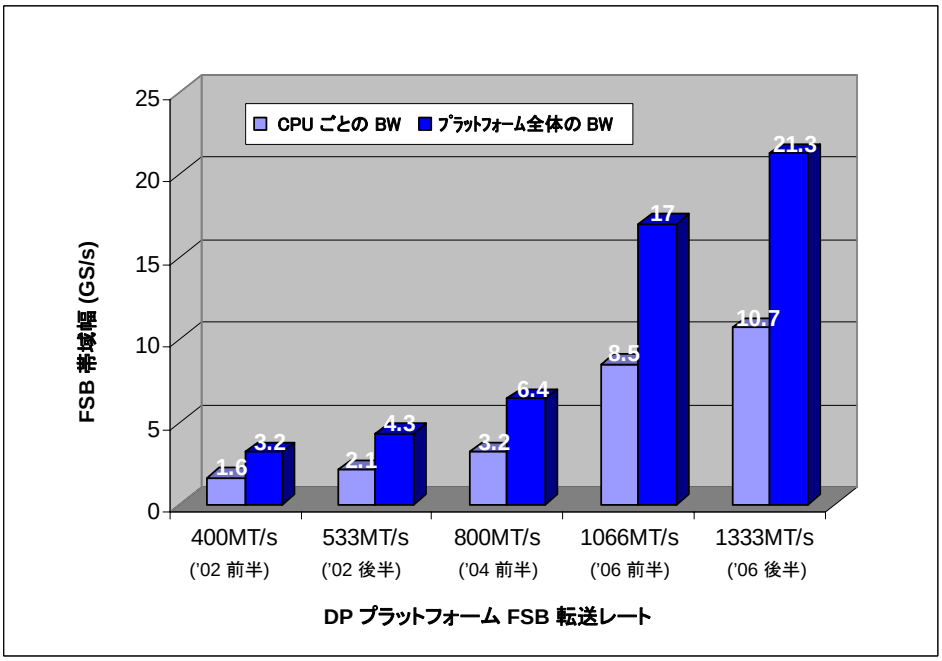

図 3: DP プラットフォーム帯域幅の増加

インテル® プラットフォーム全体の進化

プラットフォーム全体への高い評価に応え続けるため、FSB の進化に加えて、インテル® プラットフォームの他の部分も進化し続けてきました。さまざまなプラットフォーム・コンポーネントに対する強化により、FSB の有効性が高まり、可能な限り速く、データを必要な場所に効率的に転送できるようになりました。

キャッシュサイズ

インテル® プロセッサーには、同等クラスの AMD プロセッサーと比較して通常 2 倍の容量のオンダイ L2 (レベル 2) キャッシュと L3 (レベル 3) キャッシュが搭載されています。キャッシュのサイズが大きくなると、アプリケーションで必要なデータがより多くプロセッサーに置かれ、メモリーアクセスがさらに少なくなります。この結果、平均レイテンシーが削減され、システムバスに必要な帯域幅も削減されます。実装の異なる L1 命令キャッシュを直接比較することは困難なため、L1 キャッシュのサイズは比較していません。具体的には、AMD プロセッサーにはデコードされていないアセンブリー・レベルの命令を格納する従来の L1 命令キャッシュがあり、Intel NetBurst® マイクロアーキテクチャー・ベースのインテル® プロセッサー (2002 ～ 2005 年頃) には、マイクロオペレーション・レベルにデコードされた命令トレースを含むキャッシュが実装されています。

図 4 と図 5 はそれぞれ、デュアル・プロセッサー・システムおよびマルチプロセッサー・システム用に設計されたインテル® プロセッサーと AMD プロセッサーのキャッシュサイズを示しています。これらの図からインテルの DP (デュアル・プロセッサー) の L2 + L3 キャッシュサイズは、これまで常に AMD プロセッサーと同じかまたはそれ以上であることがわかります。また、本資料で述べているように、AMD プロセッサーは L3 キャッシュをこれまで実装したことがありません。

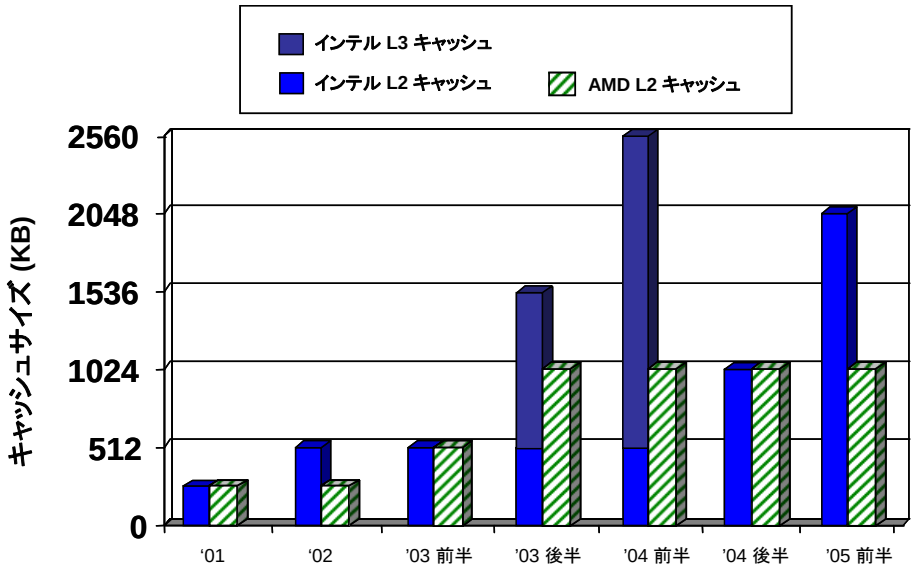

図 4: デュアル・プロセッサー CPU のキャッシュサイズ (インテル vs. AMD)

マルチプロセッサー対応 CPU では、オンダイキャッシュの量の違いはさらに大きく (インテル® Xeon® プロセッサー MP など、4 プロセッサー以上への対応) なります。オンダイキャッシュが大きくなると、より多くのデータをプロセッサーの近くに格納できるため、必要な FSB またはシステム・バス・トランザクションがより少なくなります。(注: AMD は MP 対応プロセッサーを 2003 年以降に出荷)

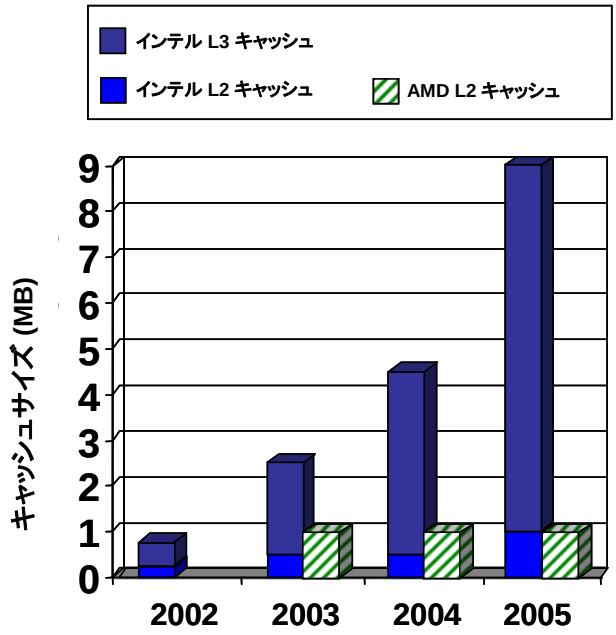

図 5: MP プロセッサー CPU のキャッシュサイズ (インテル vs. AMD)

キャッシュ・コヒーレンシー・トラフィック

FSB 周波数が高くなると、バス当たりのエージェントの数は減ります。各プロセッサーにそれぞれ 1 つの FSB が装備されていると、プラットフォーム全体の帯域幅が増加した場合、性能が低下する可能性があります。これは、各プロセッサーがそれぞれのバス上にあるためにキャッシュ・コヒーレンシー・トラフィックが増加して起こります。データの読み出し要求がチップセット経由で I/O デバイスから行われた場合、データはプロセッサーの 1 つのキャッシュに存在するか、メインメモリーに存在します。従来のマルチエージェント共有 FSB では、1 つのスヌープ要求がバスに行われます。すべてのプロセッサーは同時にスヌープ要求を受け取り、要求されたデータの最も新しいコピーがどのプロセッサーにあるかをチェックします (オンチップキャッシュに存在)。このため、読み出し要求ごとに 1 つのスヌープ要求が行われます (図 6)。

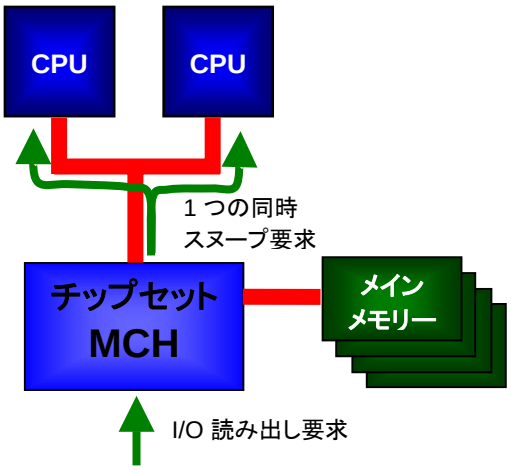

図 6: 2 つのプロセッサーで FSB を共有

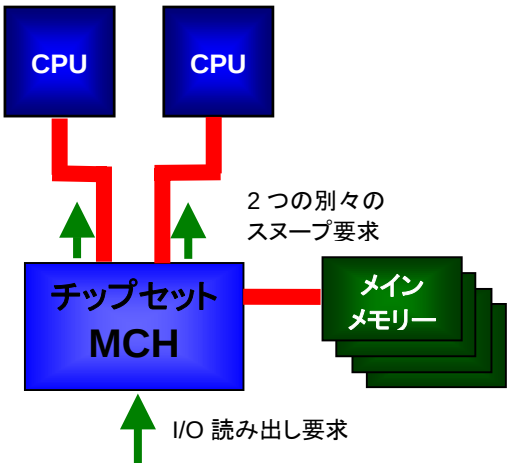

図 7: 2 つのプロセッサーで別の FSB

複数の FSB を持つプラットフォーム (図 7) では、1 つのスヌープ要求をシステムのすべてのプロセッサーに送ることはできません。代わりに、個別のスヌープ要求を FSB のすべてのプロセッサーに送る必要があります。この場合、2 つのスヌープ要求が行われます (2 プロセッサー・システムの各 FSB で 1 つ、4 プロセッサー・システムの場合は 4 つ)。

つまり、プロセッサーの 1 つが最も新しいデータを戻して応答すれば良い (データがメインメモリーにのみ存在し、プロセッサーのオンチップキャッシュに存在しない場合は、応答する必要がない) 場合でも、1 つの読み出し要求は 2 つのスヌープ要求になります。この「不必要な」スヌープ要求のために、FSB アドレス/結果バスで利用可能な帯域幅が削減されます。この不必要なスヌープ要求の数はバス・トランザクションの種類と各プロセッサーのキャッシュの状態によって異なります。例えば、すべてのプロセッサーで現在共有されているキャッシュラインへの書き込みトランザクションは、すべてのプロセッサー・バス上のバス・トランザクションのままになります。

この問題に対処するため、インテルの最も新しいデュアル・プロセッサー対応チップセットである Greencreek† では、「スヌープフィルター」と呼ばれる機能を装備しました。

スヌープフィルターは、システム上の各プロセッサーのオンチップキャッシュに現在存在するすべてのキャッシュラインの状態 (Modified、Exclusive、Shared または Invalid のいずれか) のルックアップテーブルです。スヌープフィルターは、次のように動作します:
I/O からプロセッサー、またはあるプロセッサーから別のプロセッサーへのデータの読み出し要求が作成されると、スヌープフィルターは要求されているアドレスを確認して、スヌープフィルターのタグ配列と比較します (タグ配列には各プロセッサーのキャッシュに現在存在しているメモリーアドレスが含まれています)。

† 開発コード名

読み出し要求が 1 つのプロセッサーのアドレスとのみ一致 (キャッシュヒット) した場合、スヌープフィルターはデータが存在するその 1 つのプロセッサーにのみ要求を転送します。スヌープフィルターは、要求が他のプロセッサーのキャッシュで「ミス」しているため (キャッシュミス)、要求を他のプロセッサーに転送しません。指定されたプロセッサーのキャッシュをミスする要求を「フィルター」することで、FSB アドレス/リクエスト帯域幅が不必要なスヌープ要求によって減少することはなくなります (図 8)。

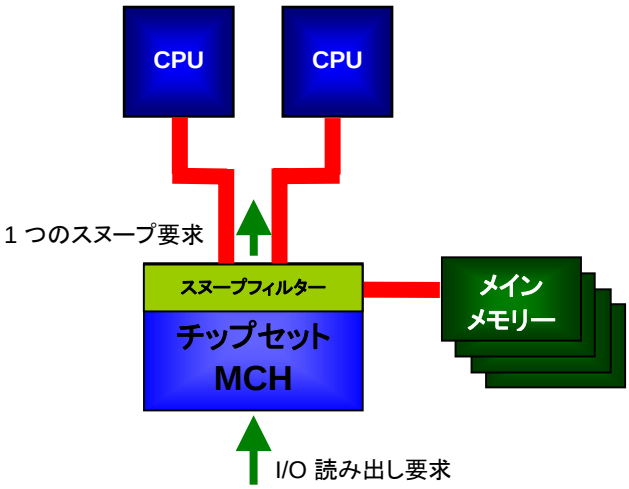

図 8: スヌープフィルターの動作 (単一キャッシュヒットの場合)

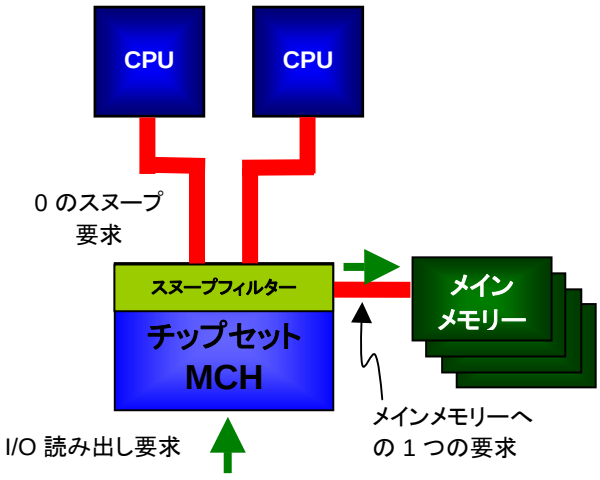

図 9: スヌープフィルターの動作 (キャッシュミスの場合)

要求されているデータがどのプロセッサーのキャッシュにも存在しない場合 (キャッシュミス)、スヌープ要求は作成されず、代わりに要求は直接メインメモリーに送られます (図 9)。これらは読み出し要求の例であり、実際の動作はバス要求の種類によって異なります。

スヌープフィルターを使用すると、各トランザクションで遅延が発生することはないのでしょうか? 遅延は非常に限定的な条件でのみ発生します。インテル® チップセットに実装されたスヌープフィルターは、システムがバス要求を生成する能力と同じ割合でルックアップ要求を受け取る能力があります (詳細は、この後の「スヌープフィルター詳細」セクションを参照)。このため、能力的な遅延はありませんが、実際のスヌープ応答は、通常のスヌープ応答よりも 2 バスクロック後に FSB に返されるため、最初のスヌープ・フィルター・ルックアップで遅延が発生します。以降のスヌープ・フィルター・ルックアップは通常、パイプライン化されるため、遅延が発生することはありません。

性能に関してはどうでしょうか? スヌープフィルターの目的は、FSB アドレス/リクエストバスの不必要なトラフィックを削減することです。アドレス/リクエスト・トラフィックが減ると、次の 2 つの場合に性能が向上します。1) 負荷が非常に高い場合。2) 大量の I/O トラフィックがある場合。最初に、負荷が非常に高い場合について説明します。

負荷が非常に高い場合: アドレス/リクエストバスのトラフィックが多い場合、アドレス/リクエストバスはシステム性能のボトルネックになります。この場合、スヌープフィルターによって不必要なスヌープトラフィックを減らすと、システム全体の性能が向上します (実際のシステムトラフィックが増えてスヌープトラフィックが減るため)。高いアドレス/リクエストバス要求は、負荷が非常に高いシステムや多くのプロセッサー (または多くのプロセッシング・コア) を搭載しているシステムで発生します。プロセッシング・コアが多くなると、コア間で生成されるキャッシュ・コヒーレンシー・トラフィックも増加します。

大量の I/O トラフィック: スヌープフィルターが性能向上に役立つその他の構成としては、大量の I/O トラフィックがある場合です。これは、I/O によって使用されるデータは通常、プロセッサーのキャッシュには存在しないのですが、I/O によって要求されるデータは最も新しいコピーであるので、それを保証するために、キャッシュは必ずスヌープされます。この結果、プロセッサーへの I/O スヌープ要求のほとんどは結果として不必要で、FSB に不必要なスヌープ要求が作成されます。スヌープフィルターがあると、I/O 要求はメインメモリーに直接送られ、不必要な FSB スヌープは生成されません。スヌープ・フィルター・ルックアップはプロセッサーのキャッシュをスヌープする時間よりも速いため、通常の「クリーンな」スヌープ・フィルター・ルックアップでは、I/O レイテンシーも削減されます。スヌープフィルターがないと、不必要なスヌープが生成されることになります。

スヌープフィルターは負荷が非常に高い場合に最適に動作するため、その能力は大規模なサーバーやハイエンド・ワークステーション・アプリケーションのワークロードに最も適しています。

スヌープフィルター詳細

前述したように、システム全体の性能にスヌープフィルターが与える影響は、バス・トラフィックの量と種類によって異なります。スヌープフィルターは、システムのすべての FSB が開始できる要求とほぼ同じ割合でルックアップを行うことができるので、負荷が非常に高い場合に最適に動作します。各プロセッサーは、2 バスクロック (BCLK) ごとにバスに新しい要求を出すことができます。スヌープフィルターを含むデュアル・プロセッサー・チップセットでは、BCLK につき 1 つの割合でルックアップを行うことができます (交互のクロックで各バスから 1 つの要求)。1066MT/s FSB では BCLK 周波数は 266MHz です。データバスは「クアッドパンプ」で、実際の BCLK 周波数の 4 倍で動作します。アドレス/リクエストバスは「ダブルパンプ」で、実際の BCLK 周波数の 2 倍で動作します。

アドレス/リクエストバスの帯域幅だけがシステム性能を制限する要因ではないため、これらのトラフィックが少ない場合、スヌープフィルターを使用しても性能はほとんど向上しません。バス・トラフィックが非常に少ない場合、スヌープフィルター要求はパイプライン化されず、前述した最初の要求だけでなく、より多くの要求に 2 サイクルのレイテンシーが追加されます。したがって、負荷が非常に少ない条件では、スヌープフィルターはシステムの性能を多少低下させる可能性があります。

インテル® I/O アクセラレーション・テクノロジー

インテル® I/O アクセラレーション・テクノロジー（以降インテル® I/OAT）は、プロセッサーからチップセットにネットワーク・トラフィックの負荷を移動する新しいテクノロジーです。インテル® I/OAT は CPU の使用率を最大 20-30% 削減できるので、FSB トラフィックも同じかまたはそれ以上削減されます。この CPU 使用率と FSB トラフィックの削減は、チップセットに内蔵されたインテリジェント DMA エンジンによって実現されます。DMA エンジンは、オペレーティング・システムが保持するメモリー空間から、データを要求しているアプリケーションが保持するメモリー空間へ、メインメモリーにある必要なネットワークデータをコピーします。DMA エンジンがない場合、CPU がデータをコピーし、 FSB を通過してメモリーのある場所から別の場所にデータを移動する必要があります (図 10)。DMA エンジンがあると、データは FSB を通過しないため、貴重な FSB サイクルを節約することができます (図 11)。DMA エンジンは、インテルの Blackford† および Greencreek† チップセットに内蔵されています。また、今後のほとんどのインテル DP/MP サーバー用チップセットにも搭載される予定です。

次の 2 つの例は、FSB トラフィックにおける効果を示しています。図 10 は、インテル® I/OAT がない場合のネットワークからアプリケーションへ送られる典型的なネットワーク・パケットの主なステップを示しています。図 11 は、インテル® I/OAT がある場合のネットワーク・パケットの主なステップを示しています。あまり重要でないステップについては省略しました。

† 開発コード名

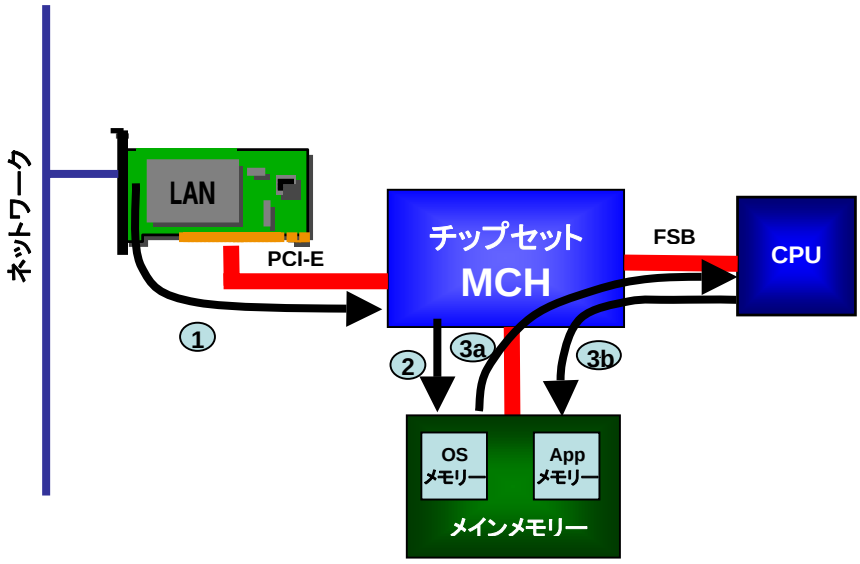

図 10: 典型的なネットワーク・トランザクション (I/OAT なし)

典型的なネットワーク・パケットのステップ (インテル® I/OAT なし):

ステップ 1: パケットがネットワークから MCH (メモリー・コントローラー・ハブ) に送られる

ステップ 2: パケットペイロードが OS が所持するメモリーに書き込まれる

ステップ 3: データペイロードが (プロセッサーによって) OS が所持するメインメモリー 位置からアプリケーションが所持するメインメモリー位置 (要求するアプリケーションによってデータが実際に使用される位置) に移動される。この移動を行うには、ペイロードデータは以下の条件を満たす必要がある。

ステップ 3a: OS メモリーからプロセッサーにコピーされる

ステップ 3b: プロセッサーからアプリケーション・メモリーにコピーされる

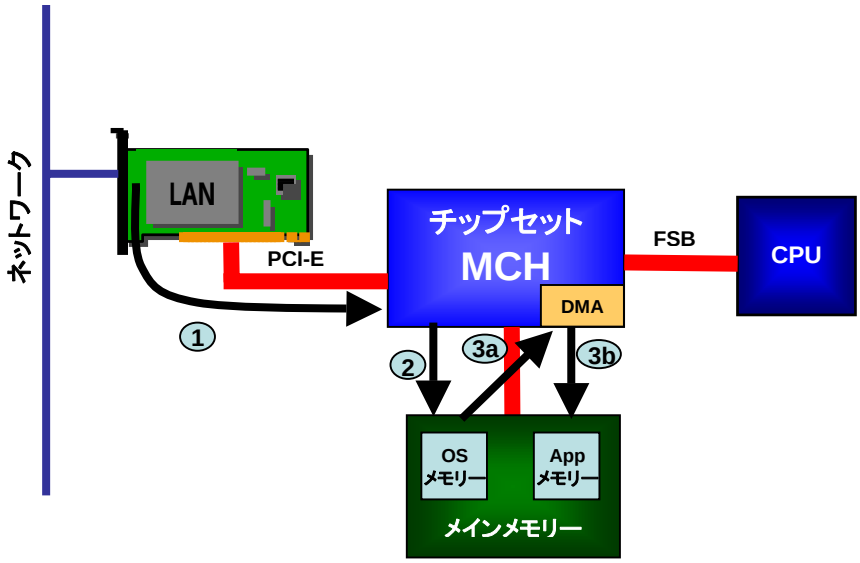

図 11: 典型的なネットワーク・トランザクション (I/OAT あり)

典型的なネットワーク・パケットのステップ (インテル® I/OAT あり):

ステップ 1: パケットがネットワークから MCH (メモリー・コントローラー・ハブ) に送られる

ステップ 2: パケットペイロードが OS が所持するメモリーに書き込まれる

ステップ 3: データペイロードが (チップセットの DMA エンジンによって) OS が所持するメインメモリー位置からアプリケーションが所持するメインメモリー位置 (要求するアプリケーションによってデータが実際に使用される位置) に移動される。この移動を行うため、ペイロードデータは FSB を使用しないでメモリーのある位置から別の位置に単純に移動される。

ステップ 3a: OS メモリーから MCH にコピーされる

ステップ 3b: MCH からアプリケーション・メモリーにコピーされる

インテル® Core™ マイクロアーキテクチャー

現世代のインテルのデスクトップ・プロセッサーおよびサーバー・プロセッサーは、主に Intel NetBurst® マイクロアーキテクチャーをベースにしています。Intel NetBurst® マイクロアーキテクチャーは、非常に高速で実行するように設計されています。この高速で実行する「エンジン」をビジーにするため、Intel NetBurst® マイクロアーキテクチャーは多くのデータを投機的に要求しますが、分岐予測ミスを含むさまざまな原因によって、一部のデータは実際には使用されません。マイクロアーキテクチャーが要求したデータのすべてを使用するとは限らないということは、プロセッサーがその要求に対して効率的に動作した場合よりもさらに高い FSB 帯域幅が必要になることを意味します。

対照的に、現世代のインテルのモバイル・プロセッサー (インテル® Pentium® M プロセッサーおよびインテル® Core™ Duo プロセッサー) は、異なるマイクロアーキテクチャーを使用しています。このアーキテクチャーは、より低いクロック周波数で動作しますが、要求したデータの使用においてはるかに効率的で、より多くのデータを使用するため、必要な FSB 帯域幅はより低くなります。

インテルは 2006 年中に、すべての x86 プロダクトライン (デスクトップ、サーバー、およびモバイル) を、Intel NetBurst® マイクロアーキテクチャーとインテルのモバイル・マイクロアーキテクチャーの利点を兼ね備えたインテル® Core™ マイクロアーキテクチャーに移行する予定です。インテル® Core™ マイクロアーキテクチャー・ベースのプロセッサーは、優れた電力効率と高性能を両立するように設計されています。この 2 つを実現する機能の 1 つは、指定されたワークロードで使用されるデータ要求の割合を高め、FSB トランザクションを少なくするための変換を行う、より効率的でよりインテリジェントなプリフェッチャーです。FSB 帯域幅を低くするということは、プロセッサーを常に効率的に使用する必要があることを意味します。

FSB と HyperTransport*

インテルの FSB と AMD の HyperTransport* バスを比較する場合、各バスの帯域幅を比較するのは一般的な方法の 1 つでしょう (図 12 を参照)。

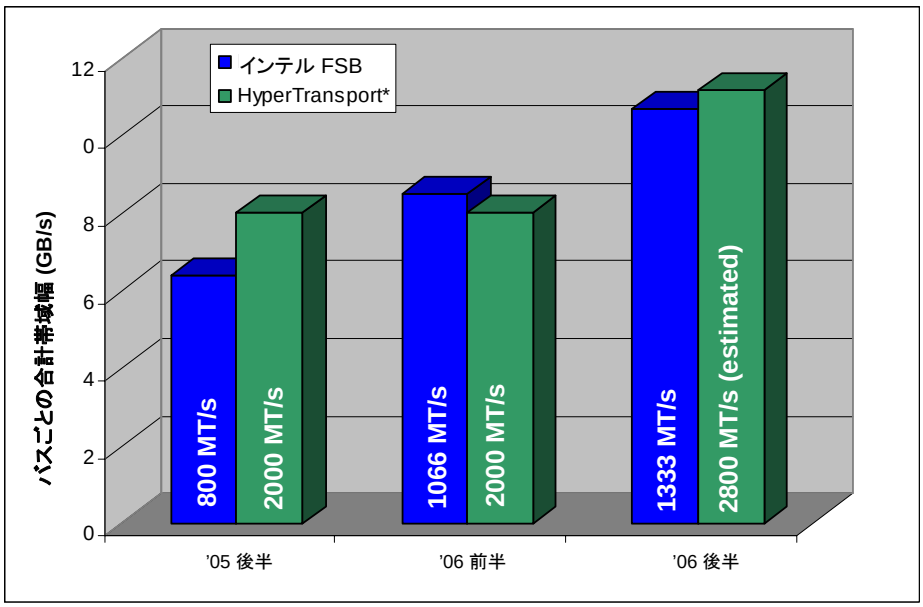

図 12: (システムバスごとの) バス帯域幅 (FSB vs. HyperTransport*)

注: 上の図は、2006 年後半に HyperTransport* バスの周波数が 1.4GHz (2.8GT/s) になると予測しています。本資料の執筆時点では、具体的な数値は AMD から公表されていません。

しかし、バス・アーキテクチャーと全体的なプラットフォーム・アーキテクチャーは異なるため、バス帯域幅の値を比較してもどちらのバスが優れているかという比較にはなりません。これには、次の 2 つの理由があります。

1) 個別アドレスバスと共有アドレスバス

2) プラットフォーム・アーキテクチャーの違いにより、バス・トラフィックの組み合わせと量が異なる

それぞれの理由について以下に説明します。

個別アドレスバスと共有アドレスバス

インテルの FSB ベース・プラットフォームは、アドレスとデータでバスが個別になっていますが、図 10 で示されているインテルの FSB の帯域幅は、利用可能なデータバスの帯域幅のみを示しています。FSB には追加のアドレスピンがあるため、アドレスとコントロール情報を渡すために追加のバス帯域幅を利用できます。しかし、HyperTransport* バスベースのプラットフォームでは、1 つのバスがデータだけでなく、すべてのアドレス、コントロール、およびエラーチェック情報を渡すためにも利用されます。このため、HyperTransport* ベースのプラットフォームでデータ転送に利用できる合計帯域幅は、図 12 の値よりも実際には小さくなります。

どの程度小さくなるのでしょうか? アプリケーションに依存しますが、一般的に、アドレス/コントロール・バスにはデータに必要な帯域幅の約半分の帯域幅が必要です。この仮定は、インテル FSB ベースのプラットフォームでは、アドレス/コントロール帯域幅がデータ帯域幅のほぼ半分であるという事実に基づいたものです (下記の計算を参照)。このインテル® プラットフォームは、アドレスバスの帯域幅で制限されることもあり、またデータバスの帯域幅で制限されることもあるように、非常にバランスがとれています。

インテル FSB の計算: アドレスとデータ帯域幅の比率

- バス転送レート
  - インテルのアドレスバス転送レート = インテルのデータバスの転送レートの ½
- 転送当たりのバイト数
  - アドレスバス: ~9 バイト**
  - データバス: 8 バイト

データ・バス・クロック当たりのアドレスバイト数 = ½ * 9 バイト = 4.5 バイト/データ・バス・クロック

データ・バス・クロック当たりのデータバイト数 = 1 * 8 バイト = 8 バイト/データ・バス・クロック

- データバス帯域幅へのアドレスは ~ 4.5 から 8 または ~1/2

** 上記の計算に含まれているアドレス/コントロール・バス信号:

A[39:3], BPRI#, BR[3:0], DEFER#, RS[2:0]#, RSP# , TRDY#, ADS#, AP[1:0]#, BNR#, DBSY#, DP[3:0]#, DRDY#, BPM[5:0]#, HIT#, HITM#, A20M#, LINT0/INTR, LINT1/NMI, SMI#, SLP#

アドレス/コントロール信号に必要な帯域幅が少なめになるように、リセット、ストロボ、およびエラー信号のような、ほとんど使用されていない一部のコントロール信号は計算に含まれていません。

HyperTransport* バスのデータサイクルのアドレス/コントロール比率がインテル FSB (アドレス帯域幅がデータ帯域幅の半分) に似ていると仮定すると、HyperTransport* バスで利用可能なデータバス帯域幅は、バスの合計帯域幅の ~2/3 になります (アドレス/コントロールで 1/3、データで 2/3)。HyperTransport* バスで実際に利用可能なデータ帯域幅はワークロードによって異なりますが、ほぼこの比率になります。

HyperTransport* バスの計算: データが利用可能な帯域幅

- バス転送レート (アドレスおよびデータ)
  - 2000MT/s (1000MHz HyperTransport* バス)
- 転送当たりのバイト数 (アドレスおよびデータ)
  - アドレスおよびデータ: 2 バイト
- 合計のバス帯域幅 (アドレスおよびデータ)
  - 一方向: 2000MT/s * 2 バイト = 4GB/s
  - 両方向: 2 * 4GB/s = 8GB/s
- データの帯域幅 (HyperTransport* バス)
  - 一方向: 2/3 * 4GB/s = **2.67 GB/s**
  - 両方向: 2/3 * 8GB/s = **5.33 GB/s**

上記の仮定を使用すると、**データ**転送に利用可能な帯域幅は (2000MT/s で動作している) HyperTransport* バスでは ~5.33GB/s、1066MT/s で動作している FSB では 8.4GB/s になります (図 13)。また、HyperTransport* バスでは 一方向で最大帯域幅が上記の値の半分になるのに対して、FSB ではどちらの方向でも帯域幅を 100% 利用できる点にも注意してください。

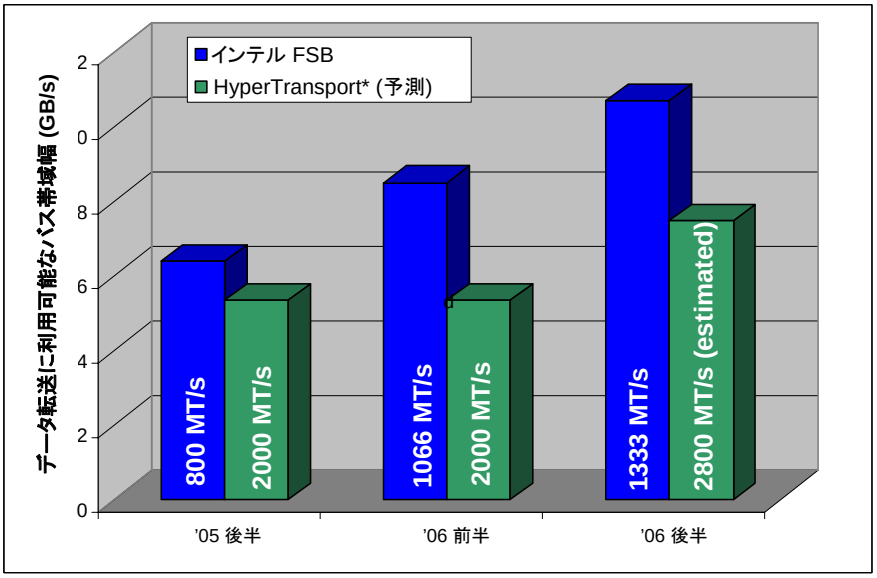

図 13: (バスごとの) データ転送に利用可能なバス帯域幅 (FSB vs. HyperTransport*)

プラットフォーム・アーキテクチャーの違い

インテル FSB ベース・システムと、AMD HyperTransport* ベースのシステムにおけるトラフィックの種類と量は本質的に異なります。これらの違いは、主にシステム内のメモリー (および I/O) の位置の違いによるものです。インテル FSB ベースのシステムでは、システムのすべてのプロセッサーから等しい距離で単一のメモリー配列が配置されています。この種のシステム・アーキテクチャーは、一般的に SMP (Symmetric Multi-Processing - 対称マルチプロセッシング) と呼ばれています (図 14 を参照)。

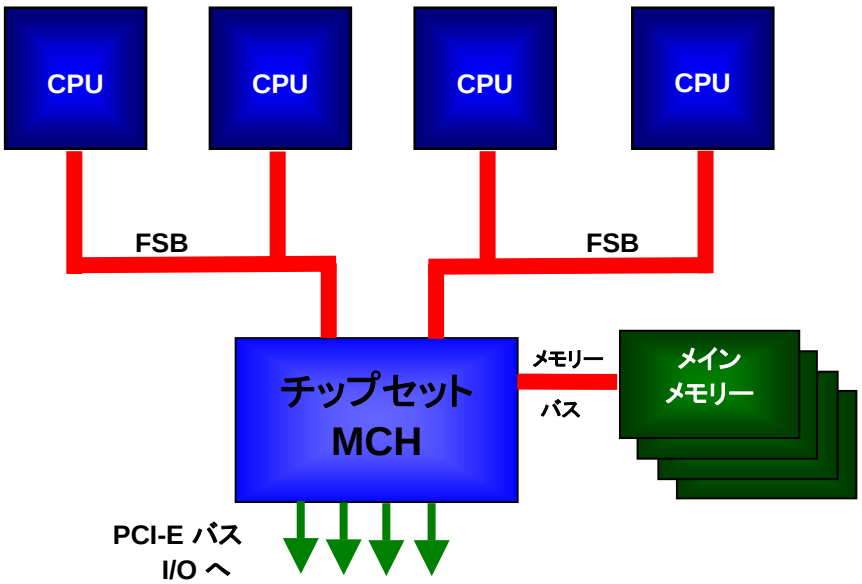

図 14: インテル FSB SMP システム・アーキテクチャー

AMD HyperTransport* ベースのシステムでは、システムの各プロセッサーにローカルメモリーが配置されています。この種のシステム・アーキテクチャーは、NUMA (非均一メモリーアクセス) と呼ばれています (図 15)。

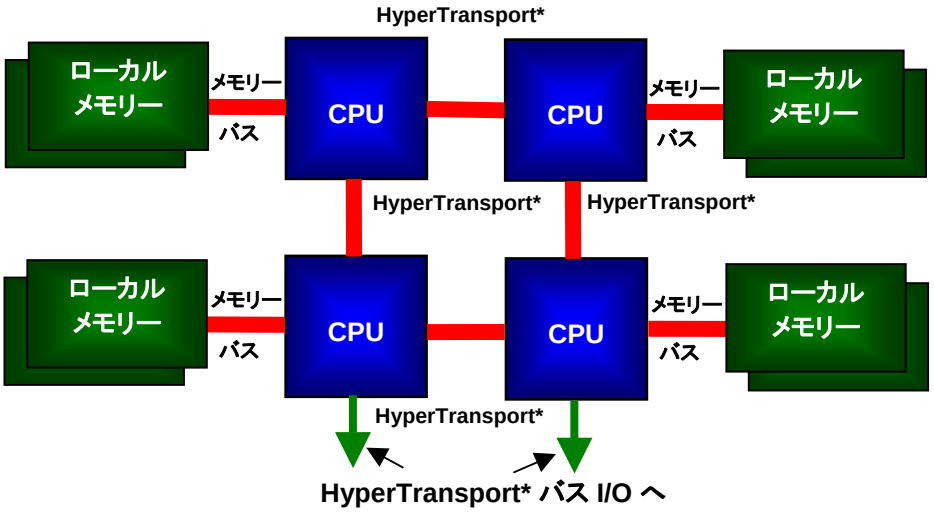

図 15: AMD HyperTransport* NUMA システム・アーキテクチャー

メモリー・レイテンシーの現実

NUMA システムでは、メモリーは複数の場所にあり、それぞれの場所にアクセスする時間が異なるため (非均一)、OS がアプリケーションのメモリーアクセスを、そのアプリケーションが実行されているプロセッサーのローカルメモリーに制限する場合には適しています。ほとんどのアクセスをローカルメモリーに対して行うことができれば、メモリーアクセスの平均レイテンシーを低く保つことができます。しかし、NUMA システムで平均メモリー・レイテンシーを低く保つには、次の 3 つのポイントに注意する必要があります。

1. アクセスはすべてローカルメモリーに対して行われると考えることは非実用的である (特にメモリーを多く必要とするアプリケーション)。
2. メモリー要求がローカルメモリーに対して行われた場合でも、システムの他のキャッシュがすべてスヌープされるまで (他のキャッシュがより新しいデータのコピーを持っているかどうか確認するまで)、アプリケーションはデータを使用できない。このため、ローカル・メモリー・レイテンシーよりも、(4 プロセッサー・システムでは 2 ホップ離れる) 最も遠いプロセッサーへのスヌープ・レイテンシーが重要である。
3. データがローカルメモリーにない場合、データを (1 または 2 ホップして) 非ローカルメモリーからフェッチする必要があるため、平均メモリー・レイテンシーが高くなる。

AMD では、ローカルメモリーへのレイテンシーが低いだけではなく、スヌープ・レイテンシーと遠いメモリーへのレイテンシーも十分低いため、キャッシュ・スヌープ・ペナルティーと遠いメモリーへのアクセスのペナルティーを考慮する必要はないと説明しています。しかし、平均メモリー・レイテンシーには、オンダイキャッシュへのレイテンシーも含める必要があります。一般的に、プロセッサー・キャッシュのレイテンシーは、近い将来にプロセッサーによって要求されるコードやデータにアクセスする場合に最も低くなります。キャッシュのヒットレート (例えば、要求されたデータが既にキャッシュにある場合のアクセス数とメモリーアクセス総数の比率) が高くなると、メモリーへの平均レイテンシーは低くなります。前述したように、インテルのより大きなキャッシュは、キャッシュのヒットレートを非常に高くし、平均メモリー・レイテンシーを最小限に保つことができることを意味します。

メモリー・レイテンシーは、次に取得する命令またはデータを予測するプロセッサーの能力によっても影響される点にも注意が必要です。プロセッサーの分岐予測とプリフェッチ機能は、キャッシュ要求が発生する前にデータをキャッシュにフェッチできるため、メインメモリーのレイテンシーを削減または除去できます。インテル® プロセッサーには、強化された分岐予測とプリフェッチ機能が備わっているため、キャッシュ・ヒット・レートを高く保つことができます。

バス・トラフィックの違い

SMP と NUMA アーキテクチャーの違いにより、2 つのシステム・アーキテクチャーのバス・トラフィックは、次の表に示すような違いがあります。

トラフィックの種類	インテル FSB	AMD HyperTransport*
メモリー	すべて	非ローカルのみ
I/O	すべて	すべて
キャッシュ・コヒーレンシー (スヌープ)	すべて	すべて

表 3: 典型的なシステムバスのトラフィック (FSB vs. HyperTransport*)

これまでに説明したように、AMD HyperTransport* ベースのシステムではメモリーがローカルにあるため、生成される全体的なバス・トラフィックは少なくなります (例えば、CPU からローカルメモリーに存在するデータへのアクセスには HyperTransport* バスは使用されません)。「非ローカル」メモリー (例えば、異なる CPU に存在するメモリー) にあるデータへのアクセスにのみ HyperTransport* バスが使用されます。しかし、インテルの FSB ベースのシステムには、表 4 に記述されているような、「システムバス」のトラフィックを削減する機能があります。

トラフィックの種類	インテル FSB トラフィック削減機能
メモリー	より大きなオンダイキャッシュ (通常、AMD のキャッシュサイズの 2 倍)
I/O	インテル® I/O アクセラレーション・テクノロジー (Q2 '06)
キャッシュ・コヒーレンシー (スヌープ)	スヌープフィルター (Q2 '06)

表 4: FSB トラフィック削減機能

最終的な結論は、こういった異なるプラットフォーム・アーキテクチャーを評価する場合、単なるバス帯域幅の値のみで比較するべきではありません。どのプラットフォームがベストなのかを判断する場合、プラットフォームの他の部分の特徴についても必ず考慮してください。

まとめ

これらの 2 つのプラットフォームは異なったシステム設計アプローチを取っています。アーキテクチャーが異なるため、どちらが優れているか個々の項目だけを比較して判断することはできません。プラットフォームのすべての機能に目を通して、それらが全体的なプラットフォームのパフォーマンス向上にどう役

立っているかどうかを判断する必要があります。IA ベースのサーバー・プラットフォームと FSB システム・アーキテクチャーは、ユーザーのニーズを満たすために、今後も多くの新しい機能を追加し (表 5)、高性能かつ堅牢なサーバーシステムを提供していきます。

新しいプラットフォームの機能	プラットフォームへの影響
FSB 周波数の増加	FSB 帯域幅の増加
プラットフォームごとにより多くの FSB	合計の FSB 帯域幅の増加 FSB トラフィック増加の可能性 (スヌープ)
より大きなオンダイキャッシュ	FSB トラフィックの削減
スヌープフィルター	スヌープ・トラフィック増加の最小化
インテル® I/O アクセラレーション・テクノロジー	FSB トラフィックの削減
次世代のマイクロアーキテクチャー	FSB トラフィックの削減

表 5: 最新の FSB の改良とプラットフォームへの影響

「良い」システム・アーキテクチャーを決めるための最高の基準は、個々の機能ではなく、アプリケーションのパフォーマンスです。インテルのサーバー・プラットフォームは、より優れたパフォーマンスを提供するように設計されています。

出典:

1. プロセッサーのデータシートおよび関連資料: www.intel.com または www.amd.com
2. インテル: 出荷前の製品を含む

著者紹介

Scott Huck　インテルのエンタープライズ事業本部でコンペティティブ・アーキテクトを担当

インテル株式会社
〒300-2635 茨城県つくば市東光台5-6
http://www.intel.co.jp/

2006 年 3 月 第 1.1 版