



White Paper

SGI® Altix 3000 グローバル共有メモリアーキテクチャ

日本SGI株式会社

1. はじめに.....	2
2. NUMAflex.....	3
3. Intel Itanium 2 プロセッサ.....	4
4. キャッシュの一貫性 (コヒーレンシ).....	5
5. インターコネクトネットワーク.....	6
6. 信頼性、可用性、保守性 (RAS).....	9
7. Peer I/O.....	10
8. Linux でのスケーラビリティの実現.....	10
9. ノード間アクセス (XPMEM と XPNET).....	11
10. XPMEM ユーザ API.....	13
11. 共有メモリポインタのサンプルコード.....	14
12. おわりに.....	15
13. 参照.....	15

## 概要

本ドキュメントは、グローバル共有メモリアーキテクチャと SGI Altix 3000 サーバ、及びスーパークラスタのその優れた特徴を説明しています。グローバル共有メモリアーキテクチャでのメモリの実装とサイズ、システムのスケラビリティの実現、キャッシュの構成とその一貫性維持、ノード内、及びノード間のメモリアクセス、バンド幅とレイテンシ、及び RAS 機能について述べています。また、Altix 3000 システムの通信インフラについて、ハードウェアとソフトウェアの実装に関しての説明も提供しています。本ドキュメントは、アプリケーションの開発者、システム管理者に必要な情報を提供します。

## 1. はじめに

SGI Altix 3000 は、システム全体でのキャッシュの一貫性を維持可能な共有メモリマルチプロセッサシステムです。SGI® Origin® 3000 システムで開発された SGI® NUMAflex™ システムアーキテクチャを更に進化・発展させたものを使用しており、性能的には大幅な強化・改善が為されています。Origin® システムは MIPS® プロセッサと IRIX® オペレーティングシステムを使用していますが、Altix 3000 シリーズは、NUMAflex と業界標準の Intel® Itanium® 2 プロセッサ、及び Linux® オペレーティングシステムにより構成されています。

Altix 3000 シリーズは、複雑なジョブの実行に対しての優れた処理性能と大規模なメモリ空間での完全なワークフローの実現により、これまでの Linux オペレーティングシステムによるクラスタや他社の SMP (シンメトリックマルチプロセッサ) アーキテクチャではなしえない新たな機能と問題解決までの時間の短縮を可能にしています。Altix 3000 システムはグローバル共有メモリと呼ばれるシステム全体にわたる一つの共有メモリ空間で、非常に大規模なデータセットを管理することによって、テクニカルアプリケーションを実行するために要求される時間とリソースを劇的に減少させます。

グローバル共有メモリは、すべてのノード上のプロセッサと I/O を含むすべてのシステムリソースに対して透過的な一つのメモリアドレス空間を提供します。このようなグローバル共有メモリをもつシステムは、システムのメモリにあるすべてのデータに直接アクセスすることが可能であり、I/O 処理やネットワークのボトルネックに左右されることなく非常に高速に処理が可能です。一方、このようなグローバル共有メモリを持たない複数の分散したノードから構成されるシステムでは、常にデータをノード間で移動し、そのコピーを行うことが必要になります。データはメッセージの形式でノード間で移動しますが、これらのデータ処理には非常に複雑なプログラミングを必要とし、またプロセッサがデータの到着を待つ時間 (レイテンシ) が非常に増大するために性能が低下します。グローバル共有メモリでは、NUMalink™ のような洗練されたシステムメモリインターコネクトや SGI が提

供するメッセージパッシングツールキット (MPT) や XPMEM などのような共有メモリ呼び出しを可能にするアプリケーションライブラリが必要となります。

## 2. NUMAflex

NUMAflexは、SGIが開発したNUMA(キャッシュコヒーレント非均一メモリアクセス)プロトコルを直接ハードウェアで実装しています。ハードウェアで実装することで、最大限に性能を発揮することを可能とし、またシステムのモジュール化を図ることを可能としました。NUMAflexの名前は、プロセッサ数、メモリ容量、及びI/O能力の3つの次元を独立にスケール可能という柔軟性(flexibility)に由来しています。AltixのNUMAflexの設計におけるキーとなるものは、SHUBと呼ばれるASICコントローラです。それは、FSB( Itanium2 搭載バス)、DIMMメモリ、I/Oシステム、及びシステム内の他のNUMAflexコンポーネントへのインターフェースとなります。

Altix 3000 は複数のコンポーネントモジュール(ブリック)により構成され、それらの大半は Origin 3000 と共有されます。Cブリック(計算ブリック)はプロセッサアーキテクチャによって、異なった実装がなされています。AltixのCブリックは2つのSHUBに接続した4つのプロセッサと32GBのメモリを2つのノードボードに搭載し、それらを3Uのブリックに格納しています。Cブリックの概略図を図1に示しています。

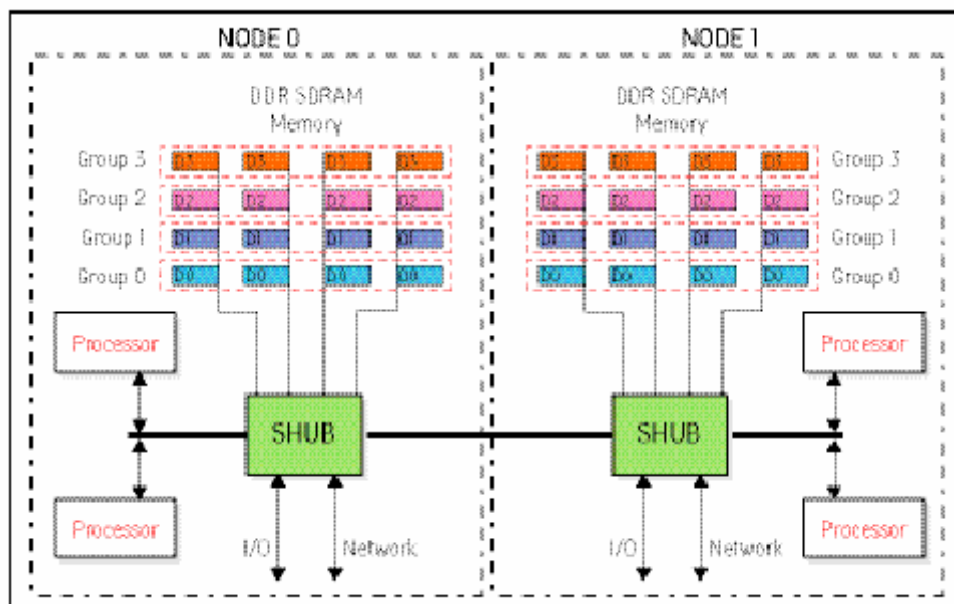


図 1. Altix 3000 C ブリック概略図

Mブリック(メモリブリック)はプロセッサ無し of Cブリックと言えます。一つのMブリ

ックは、通常は C ブリックのみが接続するインターコネクタファブリックの任意の場所に接続可能であり、プロセッサを追加することなく、シングルノードでマルチテラバイト級の共有メモリシステムを実現することが可能です。残りのコンポーネントは、R ブリック(8ポート NUMALink™3 ルータブリック)、IX ブリック(ベース IO ブリック)、PX ブリック(PC-X 拡張ブリック) 及び D ブリック 2(第2世代 JBOD ブリック)です。R ブリックは、C ブリックと M ブリック間のインターコネクタファブリックを構築するために使用されます。IX ブリックと PX ブリックは、I/O チャンネルを介して C ブリックと接続しています。SGIは、Altix 3000 に対して、ファイバーチャンネル SAN、RAID、及びオフラインストレージ製品などや様々なネットワークインターフェイスを提供しています。

### 3. Intel Itanium 2 プロセッサ

Intel Itanium 2 64 ビットプロセッサは、業界トップの単一プロセッサ性能を Altix 3000 シリーズにもたらしめています。NUMAflex アーキテクチャでは、ハードウェアによるキャッシュ coherence 制御により、512 プロセッサまでのアプリケーションのスケラビリティを実現することが可能です。Intel は Itanium 2 ファミリーにおいてソケット互換のプロセッサファミリーを提供しています。Intel はプロセッサファミリーでの新しいプロセッサを継続的に開発し、また、これらの新しいプロセッサを短期間で製品化しています。ユーザが最新のマイクロプロセッサ性能を利用することが可能であるように、Altix 3000 のプラットフォーム上でも利用可能であることが求められます。Itanium 2 プロセッサは、システムに対して毎秒 6.4GB のバンド幅をサポートするプロセッサバスを持っています。

Altix 3000 は、高性能アプリケーションの要求に対して最適化されています。Itanium 2 プロセッサは 1 本のバス上で 4 つのプロセッサまでサポートしていますが、Altix 3000 は 1 本のバス上で 2 つのプロセッサしか搭載していません。というのは、一プロセッサに対して、他のシステムの 2 倍のメモリバンド幅を提供するためです。

プロセッサが要求したデータが図 2 で示されているチップ上のキャッシュの一つに存在しない場合、プロセッサはグローバル共有メモリからキャッシュラインにデータを送るようリクエストを出します。2 つ以上のプロセッサが独立に同じデータを処理するような場合にも、Altix 3000 のハードウェアはソフトウェアの介入無しにデータの一貫性を保持することが可能です。

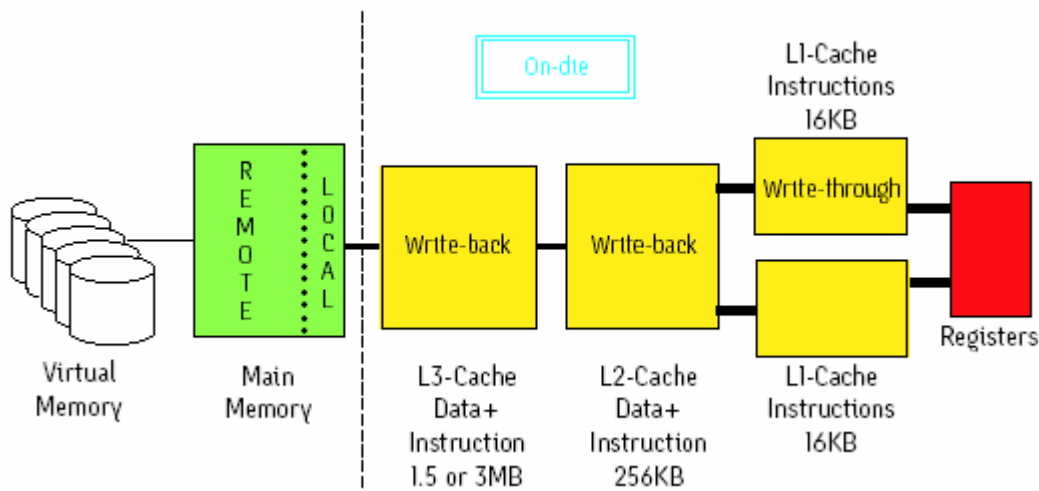


図 2 Itanium2 のメモリとキャッシュ階層

#### 4. キャッシュの一貫性 (コヒーレンシ)

Altix 3000 シリーズのキャッシュコヒーレンシプロトコルは SHUB ASIC に実装されており、それは、Itanium 2 プロセッサでのスヌーピングと NUMAflex インターコネクトファブリック間で使用されるディレクトリベースの方法に対してのインターフェースとなります。あるプロセッサがデータ取得要求を出した際に、そのデータが隣接したキャッシュに存在する場合は、データは、データが存在するプロセッサのキャッシュから要求を出したプロセッサに直接送られます。そのため、メモリにそのデータに関するリクエストを送るための遅延が生じません。

スヌーベースのシステムでは、すべての処理はキャッシュコヒーレンシを保持するためにシステム内のすべてのプロセッサに対して透過であり、そのため、大規模システムであるとシステムのオーバーヘッドが問題となります。しかしながら、ディレクトリベースのキャッシュコヒーレントシステムは、あるキャッシュラインを使用しようとしているプロセッサのみが、そのキャッシュラインの処理状態を知っていればよいことになります。これにより、キャッシュコヒーレンスを保持するためにシステム内を流れる情報量が削減でき、結果としてシステムのオーバーヘッドを減少させ、遅延を低下し、実際のデータ処理のために高いバンド幅を提供することが可能となります。

過去 10 年間では、“Beowulf” クラスタシステムが非常にポピュラーとなりました。これは、低いハードウェアコストとオープンソースソフトウェアの可用性のためです。そのようなクラスタシステムでは、キャッシュコヒーレンスはユーザの責任であり、性能面での問題を含みながらも、大抵はコモディティのインターコネクト上でのデータのコピーによってなされました。Altix 3000 では、高性能な NUMAflex のネットワークを使用した、データ

を共有するためのキャッシュコヒーレントロード・ストアが可能となりました。

## 5. インターコネクトネットワーク

Altix 3000 は NUMAflex テクノロジーの新たな機能である、NUMAlink 4 通信チャネルを使用します。NUMAlink は、従来の SGI のスケーラブルシステムで使用されていました。NUMAlink 4 は、これらのこの低レイテンシで高バンド幅な技術を受け継いでいます。NUMAlink 4 は、NUMAlink 3 との物理的な接続について互換性を保持しており、NUMAlink 3 の 2 倍のバンド幅を提供します。NUMAlink 4 は、先進の双方向信号処理技術によってこの性能を可能としています。

Altix の NUMAflex ネットワークは、ファットツリートポロジで構成されています。図 3 では、512 プロセッサ構成のこのトポロジを示しています。図 3 において丸は R ブリックを表し、線は NAMUlink のケーブルを表しています。また、中央の 128 個の小さな四角は C ブリックを表しています。

ファットツリートポロジは、システムサイズが増すと、それに伴ってバイセクションバンド幅も線形に増加するため、よいスケーラビリティを実現することが可能です。Altix 3000 は、また、増加したバイセクションバンド幅のために、デュアルプレーン、もしくは並列ネットワークを提供しています。このデュアルプレーンの構成は、Altix の C ブリック上で 2 つ NUMAlink ポートを提供することによって可能となっています。Altix でも初めは NUMAlink 3 ルータブリックが採用されていましたが、NUMAlink 4 ルータブリックを使用することにより、バイセクションバンド幅を 2 倍とすることが可能となりました。またこれにより、Itanium 2 プロセッサファミリの新たな世代の要求に応じたシステム性能を可能にしています。

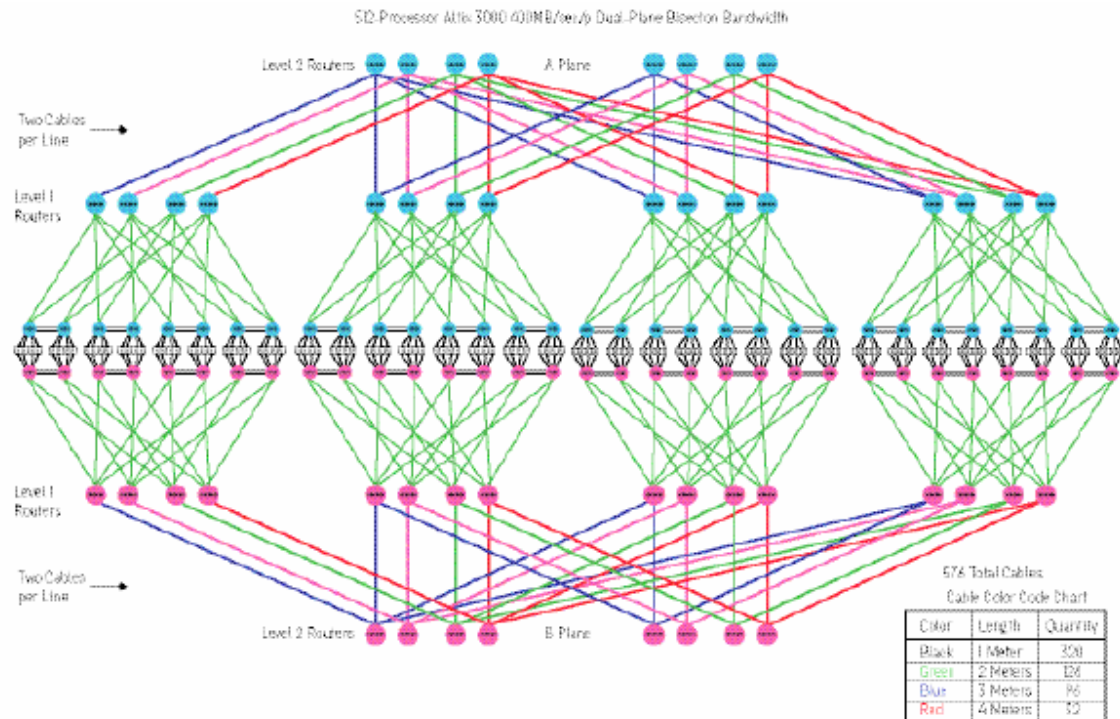


図 3 512 プロセッサデュアル“ファットツリー”インターコネクトトポロジ

Altix 3000 のメインメモリの特徴は、以下の表 1 に示されており、プロセッサに対するバンド幅が NUMalink3 と NUMalink4 について表に挙げられています。最大のローカルメモリ量が 512MB と 1GB DIMM について表に挙げられています。というのは、DIMM メモリは Mブリックに追加されるため、プロセッサと独立にスケールアップできるからです。一方で、16 プロセッサで 4TB のキャッシュコヒーレントメモリを構築することが可能です。

# of SHUBs or nodes	Maximum # of processors	NUMalink 3 Bandwidth MB/sec/p	NUMalink 4 Bandwidth MB/sec/p	Maximum Local Memory w/512MB DIMMs*	Maximum Local Memory w/1GB DIMMs*	Maximum Local Memory w/2GB DIMMs*	# of Routers	Maximum # of Router Hops
8	16	800	1600	64GB	128GB	256GB	2	3
16	32	800	1600	128GB	256GB	512GB	4	4
32	64	400	800	256GB	512GB	1TB	8	4
64	128	400	800	512GB	1TB	2TB	20	5
128	256	400	800	1TB	2TB	4TB	40	5
256	512	400	800	2TB	4TB	8TB	112	7
512	1024	400	800	4TB	8TB	16TB	288	10
1024	2048	400	800	8TB	16TB	32TB	576	10

\*M-bricks can be used to scale memory independently up to 128GB per CPU.

表 1. Altix 3000 のシステム、及びローカルメモリの特徴

物理アドレス空間は、36 ビット (SHUB あたり 64GB のアーキテクチャ上の制限) のメモリアドレスと、11 ビット (1K ノード、もしくは 2K プロセッサ) の計算ノードアドレスに分割されます。Altix の製品化の最初の段階では、512 プロセッサまでキャッシュの一貫性を保持することが可能でしたが、今後のプランとして、2048 プロセッサまでの単一の NUMAflex ネットワークを構築することが可能となります。この 2048 プロセッサシステムでは、NUMAflex ネットワーク上に 4 つの 512 プロセッサで構成される共有ドメインを作成し、そのドメイン間の通信には、共有ドメイン間でのデータのやり取りに関するコピーレント I/O セマンティクスを使用します。また、NUMALink の低レイテンシで高バンド幅な特徴も利用しています。将来的な強化では、数千のプロセッサまでキャッシュの一貫性は保持される予定です。

前世代の NUMAflex システムでは、SGI が独自に開発した DIMM メモリを使用し、データとシステム内のキャッシュの一貫性を維持するためのディレクトリ情報を格納していました。Altix 3000 では、業界標準の DIMM テクノロジーを採用しており、性能を犠牲にすることなく業界標準を採用することでの経済的な利点も提供します。

Altix 3000 のメモリサブシステムは、図 2 に示されている PC スタイルのダブルデータレート (DDR) SDRAM DIMM を使用しています。C ブリック内のそれぞれの SHUB ASIC は、4 つの DDR バスをサポートしています。それぞれの DDR バスは、4 つまでの DIMM (それぞれの DIMM は 72 ビット長であり、64 ビットはデータ、8 ビットは ECC となっています。) を含みます。4 つのメモリバスは独立であり、毎秒 12.8GB までのメモリバンド幅を提供するために同時に処理を行うことが可能です。

DIMM Type	DDR Speed	DIMM Size	Local Memory Bandwidth
PC2100	266 MHz	512MB, 1GB or 2GB	8.5GB/sec
PC2700	320 MHz	512MB, 1GB or 2GB	10.2GB/sec
PC3200	400 MHz	512MB, 1GB or 2GB	12.8GB/sec

表 2. Altix 3000 で使用されている DDR SDRAM メモリ

システムは、PC2100 (DDR-266MHz)、PC2700 (DDR-320MHz)、及び PC3200 (DDR-400MHz) の DIMM をサポートしています。これらの DIMM を使用した際の総メモリバンド幅は、それぞれ毎秒 8.5GB と毎秒 10.2GB です。256Mb DRAM 技術の利用により、DIMM 毎に 512MB の容量が与えられます。これは、プロセッサ毎に 4GB、ノード毎に 8GB のベース容量を与えます。1GB DIMM テクノロジーは C ブリック毎に 32GB、プロセッサ毎に 8GB の容量を提供します。異なる DIMM サイズは、C ブリック内で混在可

能ですが、一度に 8 つの DIMM をセットで追加しなくてはなりません。システムは、2GB DIMM がリリースされた際に利用できるように設計されています。

それぞれの SHUB ASIC はキャッシュの一貫性を維持するための最新のディレクトリ情報を自身のキャッシュ内にも保持しています。このため、キャッシュの一貫性についての情報を調べるのに必要とされるメモリバンド幅を削減できるため、メモリサブシステムをデータ処理のために効率よく使用することが可能です。

システムが起動されると、キャッシュの一貫性の維持のためのディレクトリ情報を格納するためにシステムのおよそ 3% のメモリスペースが確保されます（比較してみると、典型的なメモリ構造では、エラーの検出やコードの修正のために 12% のメモリを確保します）。

このメモリ上のディレクトリ空間は、ディレクトリキャッシュ内で使われていないディレクトリ情報を確保するために使用されます。異なった DIMM のバス上のキャッシュライン上のデータは、同時に平行に格納されます。そのため、もしメモリ参照のためのディレクトリ情報が、チップ上のディレクトリキャッシュに存在しない場合は、ディレクトリ情報はデータ（異なる DIMM のバス上に格納されている）と同時にメモリから読み込まれます。この最適化された格納方法では、システムがバス使用の衝突を最小限にすることが可能であり、データをやり取りするための最大限のバンド幅が保証されます。ローカルプロセッサバスはピークバンド幅として 6.4GB/s を保持しており、ローカルメモリサブシステムは、リモートプロセッサと I/O メモリのリクエストを処理するために利用可能なバンド幅を残しながら、ローカルプロセッサの要求を十分に満たすためのバンド幅を保持しています。

キャッシュ利用が可能なメモリのサポートに加えて、メモリサブシステムは、メモリ内のアトミックオペレーション（AMO）の機能もサポートしています。AMO は、ロックやバリアのような高速でスケラブルな通信プリミティブを提供するために使用されます。これらの AMO は、異なるプロセッサキャッシュ間でのキャッシュライン全体でのデータ転送を必要としません。もし、キャッシュライン全体でのデータのやり取りを行うとすると、データの衝突の可能性も高く、性能劣化の原因になります。

## 6. 信頼性、可用性、保守性（RAS）

Altix 3000 シリーズは、堅牢なオペレーティング環境を提供します。多くの技術を駆使することで、システムでのデータフローの保護を図っています。

メモリは、全てのシングルビットエラーを修正し、マルチビットエラーを検出します。

chipkill エラー検出を提供することも可能です。

NUMAlink チャンネルは、CRC プロテクションコードをもつチャンネルを通るメッセージの流れを保護します。もし NUMAlink チャンネルの終了を受け取る際に、CRC エラーが検出されると、メッセージは再送され、システムが信頼性のある通信を提供することを可能にします。プレーンのうちの一つに障害が発生した場合にも十分な処理を維持できるように設計されており、デュアルプレーンの NUMAlink インターコネクトファブリックでは、より高い可用性を提供します。

Itanium 2 マイクロプロセッサはキャッシュにも ECC 保護を提供します。ECC は、プロセッサバスでも使用されており、シングルビットエラーを検出、修正し、マルチビットエラーを検出します。

システムは Linux カーネルが実行されている複数のノードから構成することが可能です。そのため、もし一つのノードが致命的なエラーをこらむった際には、他のノードは、障害が発生したノードが復帰するか再起動されるまで処理を継続することが可能です。

Altix 3000 パッケージでは、N+1 ホットスワップファンと N+1 冗長電源サプライヤを C ブリック、R ブリック、及び I/O ブリックで提供します。

## 7. Peer I/O

Altix のキャッシュコヒーレントプロトコルは、次世代の I/O ブリックを直接、NUMAlink ファブリックに接続することを可能とするように設計されています。この “perr-I/O” の機能としては、NUMAflex システムが処理能力と独立に I/O をスケールすることを可能としています。また、余計なインフラストラクチャのコストを掛けることなく、ユーザが必要とするであろう正確な要求を構築することを可能としています。

## 8. Linux でのスケーラビリティの実現

標準 Linux オペレーティングシステムを使用して Altix 3000 では、64 プロセッサまでのシングルシステムイメージを実現しています。これまでのクラスタでは、それぞれのホストは実行が想定されるプロセスが必要とするであろう最大のメモリを持つ必要がありました。もしプロセスがシングルホストで提供される量よりも多くのメモリを必要とした場合は、プログラムを変更し、複数のホストにプロセスが必要とするメモリを分散する必要があります。Altix 3000 システムでは、メモリは、全プロセスが使用できる巨大な共有リソースとなり、搭載された全メモリを個々のプロセッサが全て使用することが可能です。そのため、プログラムを変更することなく、個々のプロセッサが使用できるメモリサイズは、ク

ラストシステムよりも遥かに大きなものとなります。例えば、一つの Linux カーネルが動作している 64 プロセッサシステムは、一つのプロセスが 1TB までのメモリを使用することが可能です。これは、アプリケーション開発者に非常に大規模な作業領域を提供します。そのため、開発者はノード構成の制限にとらわれることなく、作業に集中することが可能です。

その他の主な Altix 3000 システムの利点としては、同じ NUMALink インターコネクで接続された複数のホストで構成されるシステム全体に対するロード/ストアアクセスに関して、ユーザメモリ領域を共有できるということです。ホスト間でのメモリセグメントの共有を実現する SHMEM や XPMEM のようなインターフェースを使用することで、ユーザアプリケーションは、メモリに直接ロード/ストアアクセスを行う事が可能です。これらの機能は、以前はシステム V 系のシステムや同様の共有メモリ構造を持つ小規模の SMP システムでのみ可能でした。

## 9. ノード間アクセス (XPMEM と XPNET)

SGI NUMAflex アーキテクチャは、複数のノードをもつことが可能です。それらのノードは、独立なシステムであり、それぞれに OS が起動されています。Altix 3000 システムの物理メモリは、ファイヤウォールによって分割することも可能です。ファイヤウォールは、他のノード上のプロセスがノード境界を越えてメモリや CPU、及び I/O アクセスを行うことを拒否したり、もしくは許可したりするために起動したり停止したりするために利用されます。

SGI NUMAflex アーキテクチャはブロック転送エンジン(BTE)が使用出来ます。BTE は、キャッシュコヒーレント DMA エンジンとして機能します。BTE は SHUB ASIC 上に実装され、一つの物理メモリ領域から他の領域に非常に高いバンド幅でデータをコピーするために使用されます。一度初期化されると、プロセッサを介さずにノード境界を越えて BTE がデータ転送を行います。

ノード間メモリアccessは、同一ホストの Altix 3000 システム上でも、プロセスが使用しているメモリへのアクセスが可能です。ノード間メモリアccessでは、ノード内、ノード外を問わずに、メモリへのアクセスが可能です。メモリへのアクセスは、BTE を使用してのデータ転送でも、直接に物理メモリ上へのアクセスでも可能です。SGI Linux を構築するためにノード間共有メモリ(XPMEM)実装ソフトウェアやネットワーキング(XPNET)ソフトウェアのレイア(ソフトウェアスタック)を図 4 に示します。

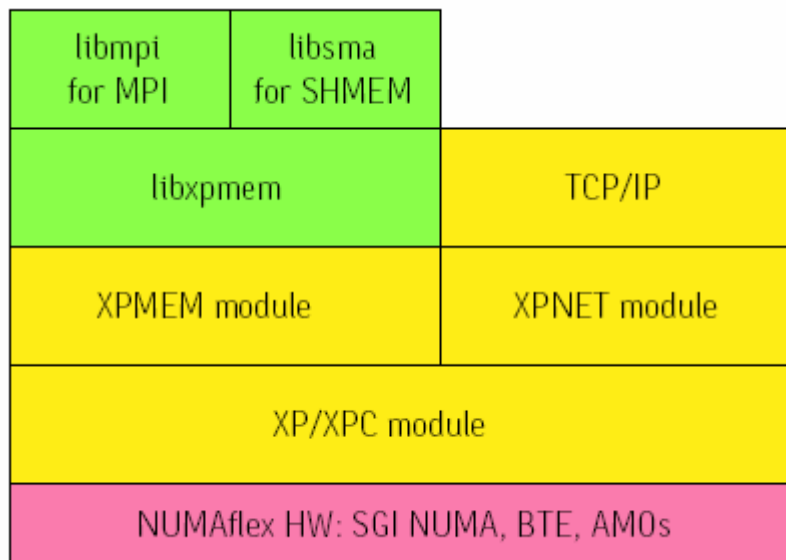


図 4. XPMEM と XPNET に関するソフトウェアスタック

XP と XPC カーネルモジュールは、NUMAlink インターコネクト上でのデータ転送のための信頼性のあるフォルトトレラントなノード間通信チャネルを提供します。XPNET は、NUMAlink インターコネクトを使用して、高速な TCP と UDP プロトコルをアプリケーションで使用することを可能とします。

ノード間メモリアクセスは、libxpmem ユーザライブラリと XPMEM カーネルモジュールを介してユーザアプリケーションで利用可能となります。XPMEM により、ソースプロセスは、ノード内、及びノード間のプロセスがソースプロセスの仮想アドレス空間にアクセスできるようにすることが可能です。ソースプロセスは、どのプロセスがそのメモリにアクセスすることができるかについての制限を定義することが可能です。他のプロセスはこのメモリ領域にアクセスを要求することが可能であり、許可されれば、アクセスすることが可能となります。一度アクセスに対する許可が出れば、リモートプロセスはソースプロセスが行うのと同様に、キャッシュコヒーレントロード/ストアによって、そのメモリ上のデータに対する処理を行うことが可能です。

XPMEM は、スワップされないように、メモリ内にあるノード間で共有された物理ページをロックします。これは、物理ページがノード境界を越えて初めて使用された場合にのみ、XPMEM カーネルモジュールによって動的になされます。これに先立って、物理ページはメモリ内でロックされるということはありません。

SGI メッセージパッシングツールキット (MPT + SHMEM) は、上記で簡単に説明された

“プロセスとプロセス間での” インターフェースを介して XPMEM を使用するために最適化されています。XPMEM の将来的な機能強化では、すでに存在する System V 共有メモリインターフェースと同様のインターフェースを含む予定です。キャッシュコヒーレンスドメイン（非キャッシュ共有メモリや高速な”メモリとメモリ間での” データコピーなど）間での XPMEM インターフェースの使用についても将来的な実装を検討しています。

IP プロトコルベースの MPICH や LAM-MPI のようなオープンソースメッセージパッシングライブラリの実装は、XPNET の上位に構築されています。

## 10. XPMEM ユーザ API

XPMEM API は、カーネルモジュールをユーザが使用できるようにしたライブラリ（`libxpmem`）により構成されています。`xpmem_make()` ライブラリ関数は、そのアドレス空間のユーザが指定したセグメントを表す一意のハンドルをもつユーザプロセスを提供します。もし、そのアドレスセグメントに外部からのアクセスが要求された場合には、他のプロセスとそのハンドルを共有することは、ユーザプロセスの責任となります。

プロセスがハンドルを持った場合、そのハンドルはそれによって示されるアドレスセグメントへアクセスすることが許可される、ということが保証されなければなりません。もし `xpmem_get()` ライブラリ関数の実行が成功すると、ID がそのプロセスに返されます。ID は、セグメントにアクセスするために使用されます。`xpmem_get()` の操作は、実際のアクセス処理の前に、その処理とは別になされます。あるセグメントにアクセスするための認証は一度だけ行われればよく、アクセス処理のたびに実行されません。

一度必要な ID が分かれば、そのプロセスは必要なメモリアクセス処理を行うための `xpmem_copy()` や `xpmem_attach()` などのような様々なライブラリ関数を使用することが可能となります。`xpmem_copy()` は、可能な場合は BTE を使用して、あるアドレスセグメントから他のセグメントへデータをコピーします。`xpmem_attach()` は、他のプロセスのアドレスセグメントから呼び出し側のプロセスのアドレス空間へデータを結びつけるために使用され、物理メモリが実際にプロセス間で共有されます。XPMEM API は、エンドユーザにメモリ共有の機能を提供する高レベルの API を構築するための利便性のある低レベルのメカニズムを提供します。最適化された MPI send/receive、MPI と SHMEM put/get、及び SHMEM リモートポインタ（以下参照）に加えて、ユーザは、明示的にシステム V 系のセグメントもしくは単一の連続した仮想メモリ領域にマップされる分散メモリ領域へのアクセスを共有することが可能となります。

## 11. 共有メモリポインタのサンプルコード

以下の簡単なコードは、リモートプロセスのメモリに存在するデータ構造にアクセスするために、どのように `shmem_ptr` を使用するかを示しています。実際には `libsma (SMEM)` 内で `XPMMEM` を使用することによってアクセスされます。このプログラムでは、もしプロセス 1 が同じ `Altix 3000` システム内の他のホスト上で動作していたとしても、MPI プロセス 0 は 100 個の整数データを MPI プロセス 1 に存在する `bigd` 配列に直接格納します。リモートに存在する `bigd` 配列の仮想アドレスが、`shmem_ptr( )` から返され、プロセスは単純にそのアドレスにデータを格納します。プロセス 1 が送信されたデータを読む前に、バリア同期がなされます。`shmem_ptr()` は、その利用の簡単さと威力が、コード開発者にとっては、その利用を検討するに値するインターフェースになっています。

```
#include <mpi.h>
#include <mpp/shmem.h>
main(int argc, char **argv)
{
    static int bigd[100];
    int *ptr;
    int i;
    MPI_Init(&argc, &argv);
    if (_my_pe() == 0) { /* MPI rank 0 */
        /* initialize PE 1's bigd array */
        ptr = shmem_ptr(bigd, 1);
        for (i=0; i<100; i++)
            *ptr++ = i+1;
    }
    shmem_barrier_all();
    if (_my_pe() == 1) { /* MPI rank 1 */
        printf("bigd on PE 1 is:\n");
        for (i=0; i<100; i++)
            printf(" %d",bigd[i]);
        printf("\n");
    }
    MPI_Finalize();
}
```

## 12. おわりに

Altix 3000 は、HPC アプリケーションにとって、強力な新しいサーバ、及びスーパークラスターのアーキテクチャといえます。高速性、拡張性、グローバル共有メモリ、ハードウェアによるキャッシュの一貫性の維持、及び一連の共有メモリ API によって、ユーザは短時間でよりよい方法によって、大規模で複雑な計算問題を解くことが可能となります。

## 13. 参照

1. インテルプレスアナウンス, 2003年1月16日:

[www.infoworld.com/articles/hn/xml/03/01/16/030116hntanium.xml?s=IDGNS](http://www.infoworld.com/articles/hn/xml/03/01/16/030116hntanium.xml?s=IDGNS)

2. MPI プログラママニュアル, MPT リリースノート, mpi man ページ, 及び intro...shmem man ページ: <http://techpubs.sgi.com>

3. [www-unix.mcs.anl.gov/mpi/mpich](http://www-unix.mcs.anl.gov/mpi/mpich)

4. [www.lam-mpi.org/](http://www.lam-mpi.org/)

(C)2002 SGI Japan, LTD. All rights reserved.


Silicon Graphics, SGI, Origin, IRIX、および SGI のロゴマークは米 Silicon Graphics, Inc./日本 SGI 株式会社の登録商標です。また NUMAflex, NUMalink, SGI Linux は米 Silicon Graphics, Inc./日本 SGI 株式会社の商標です。Linux は Linus Torvalds の登録商標です。Intel および Itanium は Intel Corporation の登録商標です。UNIX は The Open Group の登録商標です。MIPS は MIPS Technologies, Inc. の登録商標です。その他の商標については商標の所有者に所有権が属しています。

## 日本SGI株式会社

〒150-8001 東京都渋谷区恵比寿4丁目20番3号 恵比寿ガーデンプレイスタワー

TEL: 03-5488-1811 (大代表)

東京本社 TEL: 03-5488-1800 (代表) FAX: 03-5420-7030

 TEL: 0120-161-086 FAX: 0120-161-087

西日本支社 TEL: 06-6343-6700 (代表) FAX: 06-6343-6713

中部支社 TEL: 0565-35-2561 (代表) FAX: 0565-35-2189

つくば東北支所 TEL: 0298-58-1551 (代表) FAX: 0298-58-1071

アソシエイトセンター TEL: 045-882-3700 (代表) FAX: 045-882-0850