大規模超並列スーパーコンピューターシステム PRIMEHPC FX10 の特長

1. システムの構成概要

2012 年 4 月から稼動予定の大規模超並列スーパーコンピューターシステムとして、富士通 PRIMEHPC FX10(4,800 計算ノード構成)を納入する予定です。納入する PRIMEHPC FX10 は、50 筐体構成で総理論演算性能 1.1PFLOPS、総主記憶容量は合計 150TByte を有し、インターコネクトとして、富士通独自開発の 6 次元メッシュ/トーラスインターコネクト(Tofu インターコネクト*1)を採用した構成です。

またストレージ環境は、ローカルファイルシステムと共有ファイルシステムで構成します。ローカルファイルシステムは RAID-5 構成で、利用可能容量は 1.1PByte です。共有ファイルシステムは RAID-6 構成で、利用可能容量は 2.1PByte です。

東京大学情報基盤センターで稼働予定の大規模超並列スーパーコンピューターシステムは、国内最大規模構成で、様々なテクニカル分野での利用が可能です。システム構成の概要を「図 1-1 システムのハードウェア構成」および「図 1-2 システム全体構成」に示します。

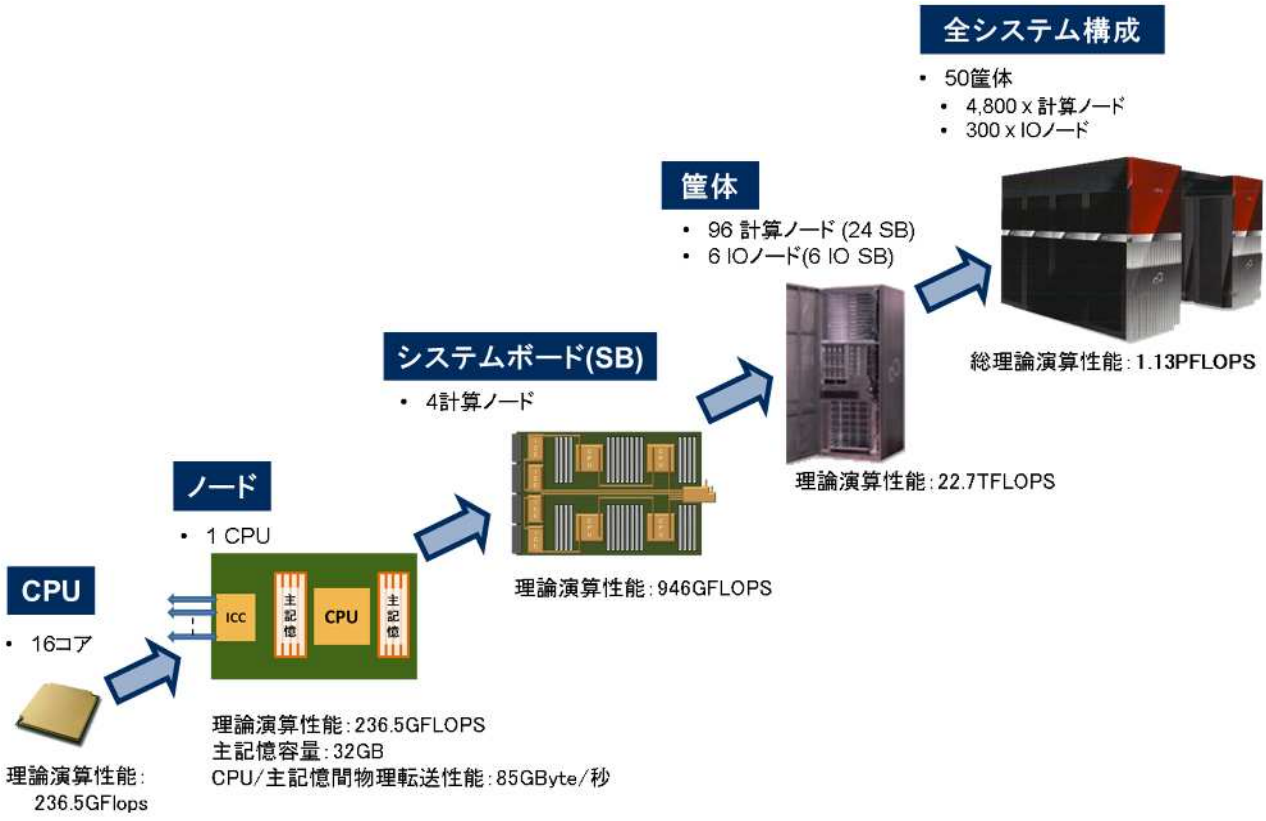

図 1-1 システムのハードウェア構成

*1 Tofu (Torus fusion) は、富士通の高速インターコネクトの呼称です。

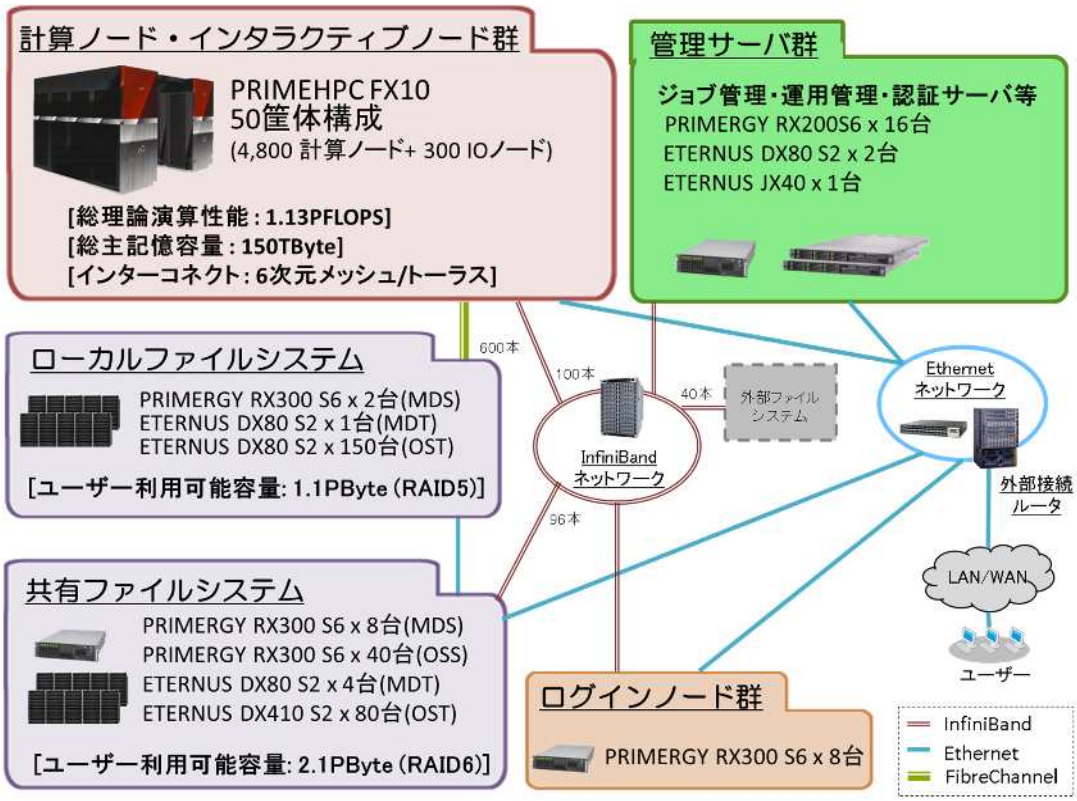

図 1-2 システム全体構成

2. PRIMEHPC FX10 の特長

PRIMEHPC FX10 は、高いアプリケーション実行性能と 20PFLOPS までスケール可能なハイスケーラビリティを実現するスーパーコンピューターシステムで主な特長は以下のとおりです。

高性能と省電力の両立

- 高性能と省電力の両立 SPARC64™IXfx プロセッサ(マルチコア化の追求，HPC 向け機能強化)
  - 環境負荷を抑える高効率冷却(システム直接水冷方式の採用，筐体外への排気熱の拡散防止)により世界トップクラスの高い消費電力性能比を実現

高並列アプリの高い実行性能の実現と開発負荷の低減

- 1CPU/ノード構成による，高いメモリバンド幅/通信バンド幅
- 10万ノード規模までスケールする富士通独自開発の6次元メッシュ/トーラスインターコネクト(Tofu インターコネクト)
- 高効率なハイブリッド並列を容易に実現する VISIMPACT

大規模システムでの高い信頼性と運用性の実現

- メインフレーム技術を継承した，部品レベルでの高い信頼性
- 高い耐故障性と運用性を兼ね備えた Tofu インターコネクト
- 大規模システムの効率的な運用を実現する HPC ミドルウェア

3. ハードウェアの特長

3.1 高性能と省電力を両立した自社開発プロセッサ SPARC64™IXfx

3.1.1 自社開発プロセッサ SPARC64™IXfx の概要

SPARC64™ IXfx は 16 個のコア、コア間で共有される 12MB のレベル 2 キャッシュ、およびメモリコントローラーなどから構成されています。半導体には最先端 40nm テクノロジーを採用しています。また、処理性能と消費電力とのバランスを重視し、高クロック化ではなくコア数を増やすことで、消費電力の増加を最小限に抑えつつ性能向上を実現しました。各コアは IU（Instruction control Unit）、EU（Execution Unit）、SU（Storage Unit）の 3 つのユニットにわかれます。IU は命令のフェッチ、発行および完了を制御します。EU は 2 つの整数演算ユニット、ロード・ストア命令用の 2 つのアドレス計算ユニット、および 4 つの浮動小数点積和演算ユニット(FMA: Floating-point Multiply and Add)から構成され、整数演算、および浮動小数点演算命令を実行します。各コアで 1 サイクルあたり 8 個、チップ全体で 128 個の浮動小数点演算が実行可能です。SU はロード・ストア命令を実行します。コアごとに 32KB のレベル 1 命令キャッシュとデータキャッシュをそれぞれ内蔵しています。

3.1.2 メモリコントローラー内蔵

SPARC64™ IXfx は、メモリコントローラーをプロセッサに内蔵しました。そのためチップセットを経由せずメモリアクセスが可能となり、低レイテンシー化と高スループット化を実現しました。

3.1.3 高信頼性

命令実行中にエラーが発生した場合は、その命令をハードウェアで自動的に再実行して処理を継続する命令リトライ機構を備えています。またプロセッサ内の大部分の回路をエラー訂正コードで保護します。プログラム実行に関連する部分については、すべてエラー検出コードで保護することで、データ保全性を確保しています。

3.2 科学技術計算命令拡張 HPC-ACE

HPC-ACE（High Performance Computing - Arithmetic Computational Extensions）は SPARC-V9 命令セットアーキテクチャーに対する HPC 向けの拡張命令セットです。

3.2.1 レジスタ数の拡張

SPARC-V9 における浮動小数点演算レジスタの数は 32 本です。この本数は他社汎用プロセッサと同程度ですが HPC 用アプリケーションの性能を最大限に引き出すためには必ずしも十分ではありません。その対策として HPC-ACE では浮動小数点レジスタ本数を SPARC-V9 の 8 倍の 256 本に強化しました。ソフトウェアパイプライニングなどの最適化により、アプリケーションが持つ命令レベルの並列性を最大限に引き出します。HPC-ACE では SXAR（Set eXtended Arithmetic Register）と呼ばれる前置命令を新たに定義することで、256 本という多数のレジスタを指定可能としました。

3.2.2 SIMD 演算

SIMD（Single Instruction Multiple Data）は、1 つの命令で複数のデータに対する演算を実行させる技術です。HPC-ACE は SIMD 技術を採用し、1 つの命令で 2 つの浮動小数点積和演算を実現しました。また複素数の乗算高速化のための SIMD 演算もサポートしています。

3.2.3 セクタキャッシュ

HPC-ACE では、従来のキャッシュとローカルメモリの長所を兼ね備えた、ソフトウェア制御可能なキャッシュ(セクタキャッシュ機能)を開発しました(「図 3-1 セクタキャッシュ機能の概要」を参照)。従来のキャッシュにおけるハードウェアによる制御では、再利用頻度の低いデータが、再利用頻度の高いデータをキャッシュメモリから追い出してしまい、性能向上の妨げになる場合がありました。セクタキャッシュ機能は、キャッシュ上のデータをグループ分けし、再利用頻度の高いデータを別の領域(セクタ)に割り当てることにより、再利用頻度の高いデータをキャッシュメモリに保持します。セクタキャッシュ機能は、キャッシュの使いやすさを踏襲しつつ、必要に応じてソフトウェアによる制御を行うことで性能向上が可能です。

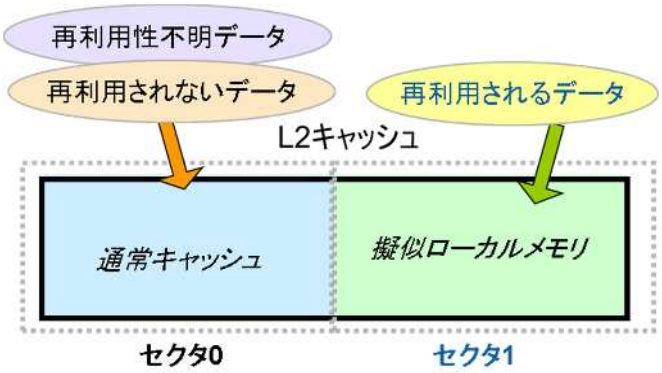

図 3-1 セクタキャッシュ機能の概要

3.2.4 三角関数 sin, cos の高速化

三角関数の sin, cos 関数を高速化するため、専用命令を追加しました。従来は、多数の命令を多数組み合わせて処理を行っていましたが、専用命令化により命令数を削減し高速化しました。

3.2.5 条件付き実行

if 文を含むループの実行を効率化するために、条件付き実行の命令を追加しました。具体的には、まず比較命令で条件判定結果を浮動小数点レジスタに書き込みます。次に比較結果を用いて、選択的に浮動小数点レジスタ間のデータ転送、または浮動小数点レジスタからメモリストアを行います。これにより if 文を含むループについても、ソフトウェアパイプライニングによる最適化を行います。

3.2.6 除算、平方根近似

逆数近似計算を行う命令を追加しました。これにより、従来パイプライン処理ができなかった除算、平方根のパイプライン処理を可能にして、処理スループットを向上させています。

3.3 VISIMPACT・ハードウェアバリア

3.3.1 ハイブリッド並列

VISIMPACT（Virtual Single Processor by Integrated Multicore Architecture）は、自動並列化をサポートするハードウェア機構と、高機能な自動並列化コンパイラにより、1 プロセスを複数コアで効率的に並列処理します。容易なプロセス／スレッドのハイブリッド並列化により、超高並列においてボトルネックとなるプロセス間通信時間の削減を実現します。VISIMPACT を実現するハードウェア機構として、SPARC64™IXfx は共有レベル 2 キャッシュとハードウェアバリアを備えています。

3.3.2 共有レベル 2 キャッシュ

SPARC64™IXfx は、12MB の共有レベル 2 キャッシュを持っています。プロセッサ内の全 16 コアでレベル 2 キャッシュを共有することでコア間でのデータ共有を容易にし、さらにキャッシュ上で隣接するデータを別々のコアで処理した場合の性能低下を防いでいます。

3.3.3 ハードウェアバリア機構

SPARC64™IXfx はハードウェアバリア機構を持っています。1 つのプロセスを複数のコアで並列実行させた場合、コア間で待ち合わせを行うケースが生じます。通常のプロセッサではこの待ち合わせ処理(同期処理)をソフトウェアで実現していますが、SPARC64™IXfx は専用のハードウェアによりこの処理を 10 倍以上高速化しています。ハードウェアバリア機構によって同期処理のオーバーヘッドを大幅に削減し、粒度の小さい演算ループに対しても、複数コアによる並列化処理を効率的に行えます。

3.4 高性能・高信頼 Tofu インターコネクト

3.4.1 インターコネクト・コントローラー ICC

PRIMEHPC FX10 のプロセッサ SPARC64™IXfx には専用のインターコネクト・コントローラーICC が 1 対 1 で接続されます。ICC は PCI Express ルート・コンプレックスと Tofu インターコネクトを統合した LSI です。Tofu インターコネクトは ICC 間のパケット転送を行う Tofu ネットワーク・ルータ(TNR)と、プロセッサからのパケット送受信を行う Tofu ネットワーク・インターフェース(TNI)、集団通信を処理する Tofu バリア・インターフェース(TBI)で構成されます。TNI は ICC に 4 つ実装され、TNR は 10 ポートの Tofu リンクを備えます。ICC は Tofu リンクにより、最大 10 個の ICC と相互接続します。

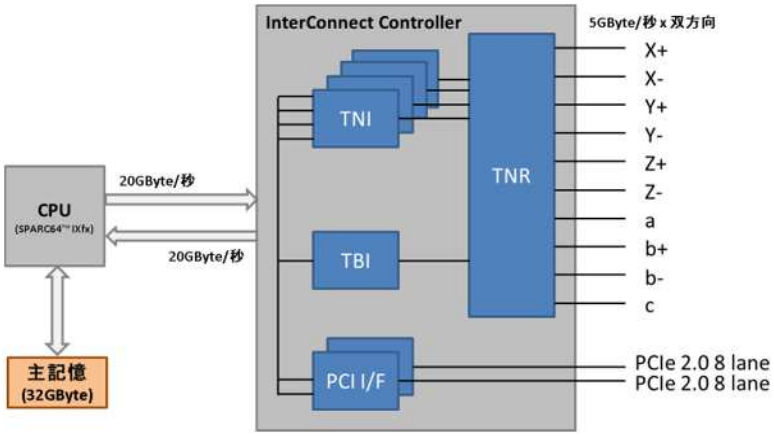

図 3-2 Tofu インターコネクト(ICC)

3.4.2 RDMA 通信

TNI は RDMA 通信機能を備えます。RDMA 通信は、宛先ノードのソフトウェアを介在せずに、データの読み出しや書き込みを行う通信です。TNI は RDMA 通信コマンドの連続実行が可能です。1 コマンドあたり最大 16MB のデータを転送します。データは最大 2KB のパケットに分割して転送します。TNI は独自の 2 レベルアドレス変換機構により、仮想アドレス・物理アドレス変換とメモリ保護を行います。アドレス変換機構はハードウェアにより主記憶上のアドレス変換テーブルを検索します。また、キャッシュ機能により変換オーバーヘッドを最小化します。低遅延パケット転送 TNR は Virtual Cut-Through 方式により、パケットを受信し切る前に次 TNR への転送を開始し低遅延を実現します。

3.4.3 4 方向同時通信

ICC は 4 つの TNI により 4 方向送信と 4 方向受信を同時に行います。非同期通信は複数並行転送、同期通信はデータ分割多重転送、集団通信は多次元パイプライン転送により、転送時間を短縮します。

3.4.4 仮想チャネル

Tofu インターコネクトはルーティングのデッドロック回避に 2 チャネル、要求/応答の用途別に 2 チャネルで合計 4 つの仮想チャネルを備えます。用途別仮想チャネルは要求パケットの輻輳による応答パケットの遅延を緩和します。Tofu インターコネクトの各受信ポートは 32KB の仮想チャネルバッファを備えます。

3.4.5 リンクレベル再送信

Tofu リンクは 1 ホップごとにエラーを修復する、リンクレベル再送信機能を備えます。TCP/IP や InfiniBand の送受信ノード間再送処理に比べ、Tofu インターコネクトのリンクレベル再送信は高速伝送路のビットエラーによる性能劣化を大幅に低減します。リンクレベル再送信のために、Tofu リンクの各送信ポートは 8KB の再送信バッファを備えます。

3.4.6 高信頼設計

ICC の SRAM および全データパス信号はエラー訂正コードで保護され、二次宇宙線などに起因するソフトエラーに耐性があります。デバッグ系を除く全制御信号はパリティで保護されます。ICC は機能ごとにモジュール設計されており、モジュール間を異常が伝播しない設計になっています。

3.5 スケーラブル・高可用性 6 次元メッシュ / トーラス

3.5.1 6 次元メッシュ / トーラスの構成

6 次元メッシュ / トーラス・ネットワークの X 軸・Y 軸は筐体間を、Z 軸・B 軸はシステムボード間を、A 軸・C 軸はシステムボード上を接続します。各次元の軸を X, Y, Z, A, B, C と呼びます。Z 軸は座標 0 に I/O ノード、座標 1 以上に計算ノードが配置されます。B 軸は、3 つのシステムボードをリング接続して冗長性を確保します。A 軸・C 軸はシステムボード上の 4 ノードを接続します。

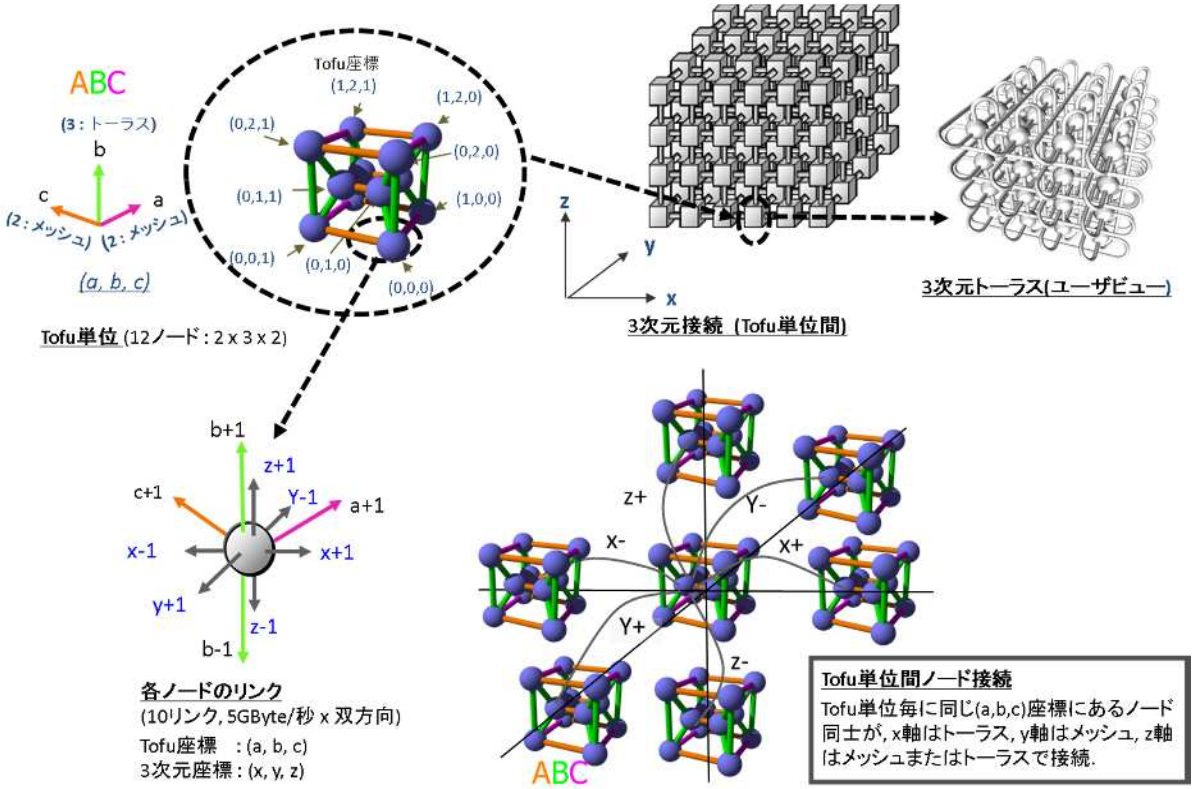

図 3-3 インターコネクトのトポロジーイメージ

3.5.2 高スケーラビリティ

6 次元メッシュ／トーラス・ネットワークは直接網と呼ばれるネットワークに属し、ケーブルを接続するだけでノード数の拡張が可能です。スイッチのポート数に制約されないため、最大システムで 10 万ノードを超える高いスケーラビリティを実現します。また外部スイッチが不要であるため、規模によらずノードあたりのコストは一定です。Tofu インターコネクトに必要な筐体間ケーブル数は、システム規模にかかわらずノードあたり 2 本です。

3.5.3 拡張次元オーダー・ルーティング

Tofu インターコネクトの拡張次元オーダー・ルーティングは、パケットを ABC 軸、XYZ 軸、ABC 軸の順に、ABC 軸を 2 回ルーティングします。通信ライブラリは送信コマンドごとに ABC 軸ルーティング経路を指定し、データ分割多重転送の経路分散や、故障回避経路の選択を行います。システムはジョブ開始時に故障ノードの位置情報を通信ライブラリに通知し、通信ライブラリによる故障回避を可能にします。

3.5.4 3 次元トーラス・ランクマッピング

Tofu インターコネクトは隣接通信を用いた通信パターンの最適化を容易にするため、ユーザーが指定する大きさの 1 次元／2 次元／3 次元トーラス空間をユーザービューとして提供します。ユーザー指定トーラス空間上の位置はランク番号で識別されます。3 次元トーラスが指定された場合、システムは XYZ の 1 軸と ABC の 1 軸の組合せによる 3 つの空間を形成します。そして、各空間で一筆書きの隣接関係を保証するようにランク番号を与えます。「図 3-4 トーラス・ランクマッピングの例」に 6×9×6 サイズ 3 次元トーラス指定時の、ランク番号割当ての一例を示します。

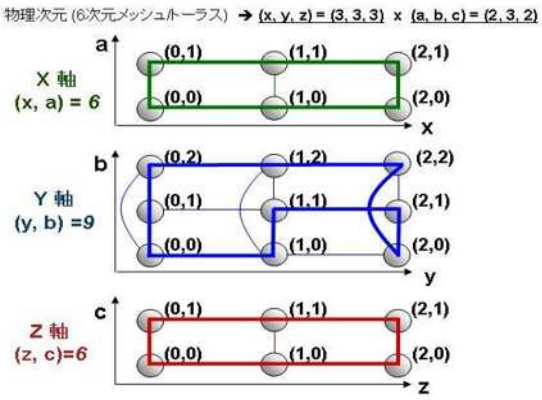

図 3-4 トーラス・ランクマッピングの例

3.5.5 高可用性

3 次元トーラス・ランクマッピングにおいて B 軸を含む空間は、B 軸が 3 ノードのリングであるため、1 ノードを避けて一筆書きを行うことが可能です。このため故障システムボードの保守交換時も周辺システムボードの運用継続が可能であり、システムの可用性を高めます。

4. ソフトウェアの特長

4.1 Technical Computing Suite の概要

富士通は、1977 年に国内初のスーパーコンピューターを開発し、その後、分散メモリ形ベクトルスーパーコンピュータ「VPP シリーズ」で並列処理システムを提供するなど、並列プログラム開発環境・並列処理ライブラリ・ジョブ運用管理機能など、高速な科学シミュレーションを効率的に実行し、システム運用を効率化するためのシステムソフトウェアを一貫して開発・提供してきました。

PRIMEHPC FX10 では、従来のスーパーコンピューターシステムの経験・技術を継承し、高速・大規模な科学シミュレーションを実現するために数万ノードレベルの高いシステムスケーラビリティ性を持つシステムソフトウェア「Technical Computing Suite」を提供します。

以下システムソフトウェアの体系と概要を以下に示します。

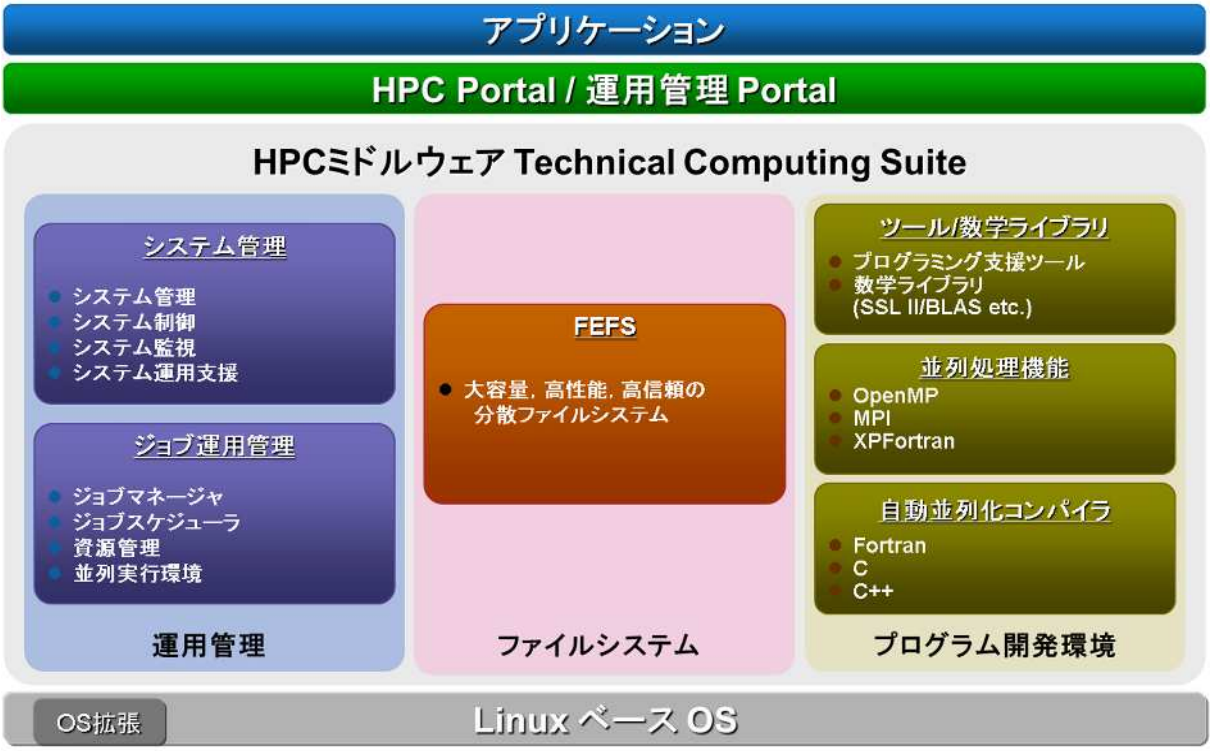

図 4-1 Technical Computing Suite のソフトウェアの体系

4.2 業界標準 OS の採用

XTC OS は、Linux カーネルバージョン 2.6.25.8 をベースとしており、多数のアプリケーションを容易に移植することが可能です。PRIMEHPC FX10 のハードウェア性能を最大限に引き出すよう拡張を行っています。主な特徴は以下の通りです。

表 4-1 XTC OS の主な特長

項 目		仕 様
アーキテクチャ対応	SPARC64™IXfx 対応	レジスタ拡張、SIMD 拡張、ハードウェアバリア対応
性能向上	TLB ミスヒット削減/メモリ効率改善	ラージページサイズ対応
信頼性向上	縮退運用	CPU 内部エラー、メモリエラー時の故障箇所切り離し
	保守ツール	異常時の情報採取機能(メモリダンプ、システム情報等)

4.3 大規模システムの効率的な運用を実現するシステム / ジョブ運用管理機能

PRIMEHPC FX10 では、システム管理機能およびジョブ運用管理機能によってシステム管理者が数百～十万台規模の計算ノードを一元的に管理します。これにより、大規模システムを効率的に利用でき利用者のニーズに柔軟に対応します。

4.3.1 システム管理

システム管理は、システムの運用管理に必要な機能を網羅的に備えており、運用管理の統合化(複数のコンポーネントを一元化)、自動化・省力化(自動監視・運用制御などの自動運転機能)を実現し、24時間人手を介さず無人運転を実現します。

システムの分割運用や、システムの電源制御、フェイルオーバー等の運用制御、管理者向けのシステム監視機能を備えています。

4.3.2 ジョブ運用管理

ジョブ運用管理は、多数のユーザのジョブ投入、豊富なジョブバリエーションを考慮し、大規模システムの効率的なジョブ・スケジューリングを可能にします。

またシステム規模の拡大に伴い、ジョブスケジューラが扱うジョブ数も飛躍的に増大します。PRIMEHPC FX10 は数万の計算ノードによる超大規模システムを実現することが可能なため、100 万を超えるジョブをシステムとして扱うことを想定しています。

4.3.2.1 大規模システムへの対応

1. ジョブ操作等の操作レスポンスを改善

   ジョブの受付けから終了までのライフサイクルに対する処理内容を再整理し、各処理をマルチプロセス、マルチスレッド化し、ジョブ投入性能において従来製品の 10 分の 1 のレスポンスを実現しました。そのため多数の利用者が大量にジョブを投入してもレスポンスが確保された状態で利用が行えます。

2. ジョブ・スケジューリング性能の向上

   PRIMEHPC FX10 向けジョブスケジューラは、特に処理コストの高い、膨大な計算資源の中からジョブ実行に最適な計算資源を選択する処理を並列処理化することにより、処理コスト低減し、スケジューリング性能を向上しています。

4.3.2.2 ジョブの種類

ジョブは、ジョブモデルとジョブタイプがあります。ジョブモデルとは、ジョブ投入コマンドによって投入されたジョブに対して設定された実行形態です(「表 4-2 ジョブモデル一覧(抜粋)」参照)。

ジョブタイプは対話型処理またはバッチ(一括)処理の実行形態です(「表 4-3 ジョブタイプ一覧を参照」)。

表 4-2 ジョブモデル一覧(抜粋)

ジョブモデル	説明
通常ジョブ	一般的なジョブ
ステップジョブ	投入された複数のジョブを 1 つのまとまりとして扱い、その中で実行の順序、依存関係を持つジョブ
ワークフロージョブ	条件分岐や繰り返しなどの実行順序記述によって、複数のジョブの組み合わせにより処理するジョブ

表 4-3 ジョブタイプ一覧

ジョブタイプ	説明
インタラクティブジョブ	利用者による端末からのデータ入力により、対話的に実行されるジョブの形態。デバッグ等での利用を想定
バッチジョブ	対話的に実行されないジョブの形態 ノードダウン等の異常発生時は、ジョブ投入オプション指定時は、自動でジョブの再投入が行われる

ジョブの並列度とは、計算ノードの割り当て方によりジョブを区別したものであり、以下「表 4-4 ジョブ並列度一覧」の 2 形態があります。マルチノードジョブの場合、1 つのジョブで全ての計算ノード(4,800 ノード)を使用するプログラムを実行することが可能です。

表 4-4 ジョブ並列度一覧

ジョブの並列度	説明
シングルノードジョブ	実行のために単一ノードを必要とするジョブ ・1 プロセスによる逐次処理を行うジョブ ・1 プロセス複数スレッド(スレッド並列)による処理を行うジョブ
マルチノードジョブ	実行のために複数ノードを必要とするジョブ ・プロセス並列プログラム(MPI ジョブ) ・プロセス並列とスレッド並列によるハイブリッド並列プログラム

4.3.3 I/O 競合への対応(ファイルステージング機能)

大量のジョブが同時実行される環境ではジョブ間の I/O 競合による実行時間のブレが拡大されることが想定されます。この問題に対して、PRIMEHPC FX10 のファイルステージング機能は、ファイル転送とジョブ実行を分離し、また計算ノードと独立した I/O ノードによるファイル転送を実現することで対応しています。

ファイルステージング機能は、共有ファイルシステムとローカルファイルシステムの間で、バッチジョブ実行に必要なデータを効率的に転送する機能で、バッチジョブ実行とは独立した非同期ステージングで行われます。

ジョブ実行前に入力ファイルをローカルファイルシステムに自動転送(ステージイン)し、ジョブ終了後に出力ファイルをローカルファイルシステムから共有ファイルシステムに自動転送(ステージアウト)します。ジョブ実行と独立した非同期ステージング(「

図 4-2 同期ステージングと非同期ステージング」を参照)を行うことで、先行するジョブが予定より早く終了した時に、次のジョブに必要なファイルのステージインが完了しておらず、CPU は空いているのにジョブが開始できないという事態を避けることができ、計算ノードの稼働率向上が可能です。

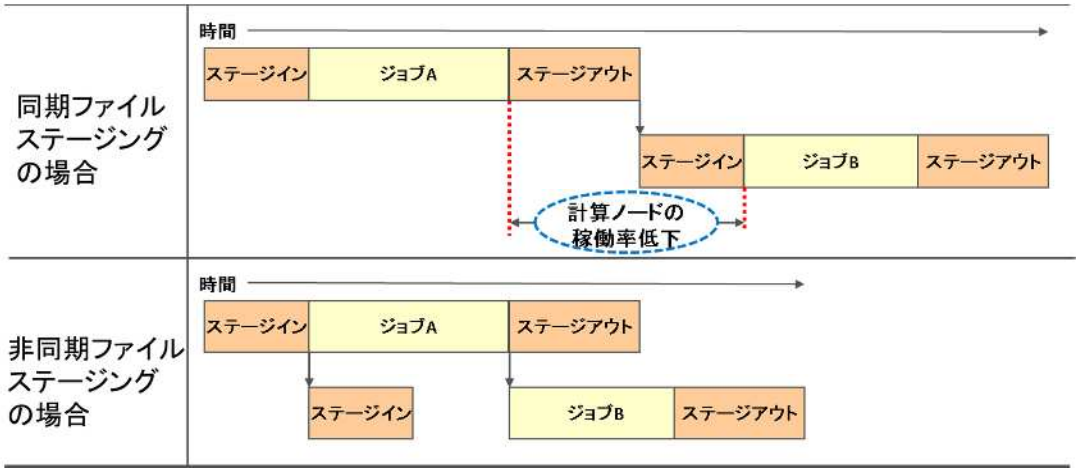

図 4-2 同期ステージングと非同期ステージング

4.4 大容量、高性能、高信頼分散ファイルシステム FEFS

4.4.1 FEFS の概要

FEFS(Fujitsu Exabyte File System) は Lustre ファイルシステムをベースに開発したファイルシステムで、数万規模のクライアントによるファイル利用を想定した大規模分散ファイルシステムです。Lustre の優れた技術を受け継ぐと共に、Lustre との互換性を維持しつつ、大規模システム向けに、最大ファイルサイズ、最大ファイル数等の拡張を大規模システム向けに実施しています。

以下「表 4-5 FEFS の主な機能一覧」に FEFS の主な機能を記載します。

表 4-5 FEFS の主な機能一覧

機能	概要
ストライピング機能	データを複数ディスクに分散させる機能
フェイルオーバー機能	障害発生時に自動的に処理を継続する機能
ジャーナリング機能	ファイル操作の整合性を保持する機能
QoS 機能	ユーザ間のフェアシェアおよびノード間の優先度制御を行う機能
quota 機能	ディレクトリ単位での quota 機能や quota 管理コマンド
ACL(Access Control List)機能	ユーザレベルでファイルアクセスを制御する機能
POSIX API	POSIX のファイル API をサポートする機能

納入する大規模超並列スーパーコンピューターシステム(「図 4-3 ファイルシステム構成」参照)では、ローカルファイルシステムおよび共有ファイルシステムの 2 種類の FEFS ファイルシステムを利用可能です。

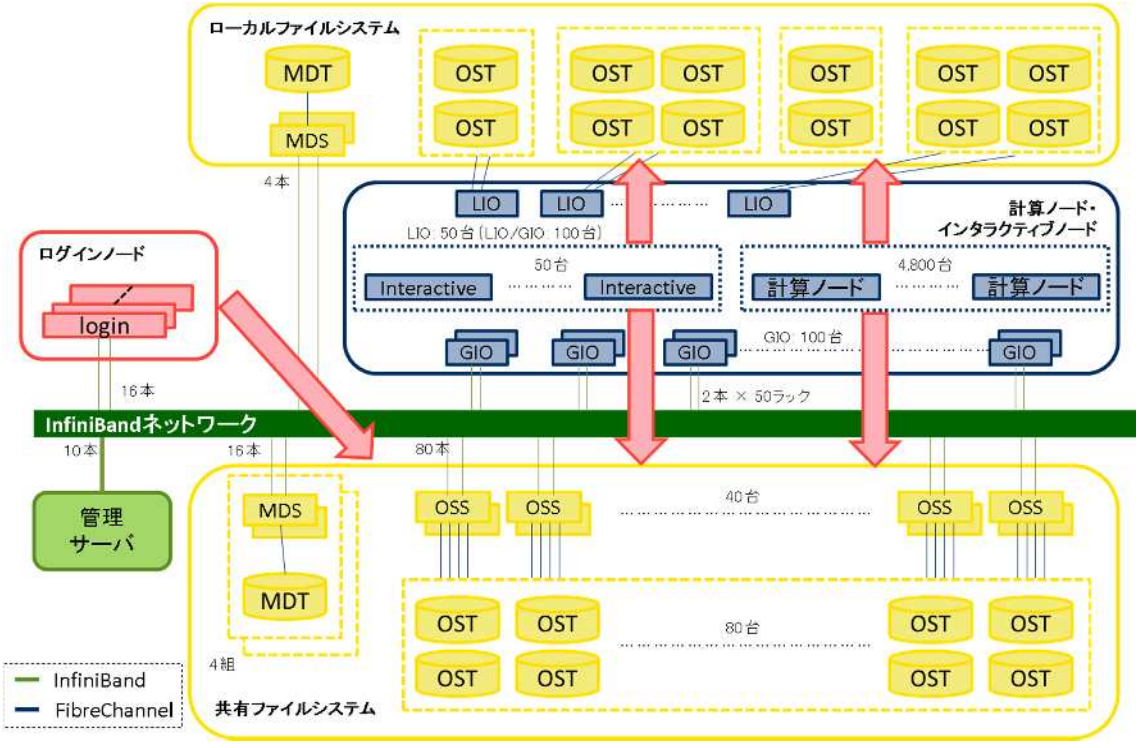

図 4-3 ファイルシステム構成

表 4-6 ファイルシステムの利用用途

種別	容量	用途
ローカルファイルシステム	1.1 PByte	ファイルステージング用領域として使用
共有ファイルシステム	2.1 PByte	ホームおよびグループ用領域

4.4.2 FEFS の機能

1. ストライピング機能

FEFS では、1 つのファイルを先頭から一定データサイズ(=ストライプ)に分割し、書き込み先 OSS をラウンドロビンで切り替え、複数ディスクに分散格納することにより、ファイルアクセス負荷を分散させる機能を提供します。分散格納されたファイルデータに並列アクセスすることにより、利用可能な帯域幅を増やすことができ、高速な I/O を実現します。また、1 個の物理的な OST の容量を上回るサイズを持ったファイルを格納することが可能になります。

2. フェイルオーバー機能

MDS は active/standby の冗長構成となっており、各クライアントは active の MDS に対して RPC リクエストを行います。active な MDS に何らかの障害が発生した場合、standby の MDS が MDT をマウントすることで処理を継続します。

OSS は active/active の冗長構成となっており、各クライアントは IO を行う OST をマウントしている OSS に対して RPC リクエストを行います。OSS に何らかの障害が発生した場合は、もう一方の OSS が OST をマウントすることで処理を継続します。

3. ジャーナリング機能

ファイル操作の整合性維持のためのジャーナリング機能を提供します。ジャーナリング機能は、メタデータを対象とし、一貫性のある処理として扱うためにファイル操作をトランザクション単位で扱います。これにより、ファイル操作の開始時にトランザクションのジャーナルを記録し、終了時にそのトランザクションをジャーナルから削除することで、ファイル操作によるデータの変更がディスクに書き込まれることを保証します。データの変更中にサーバダウンなどの障害が発生した場合、処理中の情報はジャーナルに記録されているため、ジャーナルの情報をもとに整合性がとれなくなったファイルを修復または破棄します。これによりファイルシステムの整合性を保ちます。

4. QoS 機能

FEFS の QoS(Quality of Service)機能は、(1)ユーザー間のフェアシェア制御と、(2)FEFS クライアントノード間の優先度制御の 2 種類があります。

ユーザー間のフェアシェア制御では、ユーザー単位にファイルアクセス量を制御します。1 ユーザーによる大量ファイルアクセスが原因で、その他のユーザーのファイルアクセスが遅くなるといった現象を回避できます。(「図 4-4 ユーザー間フェアシェア」を参照)

FEFS クライアントノード間の優先度制御では、クライアントノード単位にファイルアクセス量を制御します。計算ノードによる大量ファイルアクセスが原因で、ログインノードのコマンドレスポンス(ファイルアクセス)が遅くなるといった現象を回避できます。

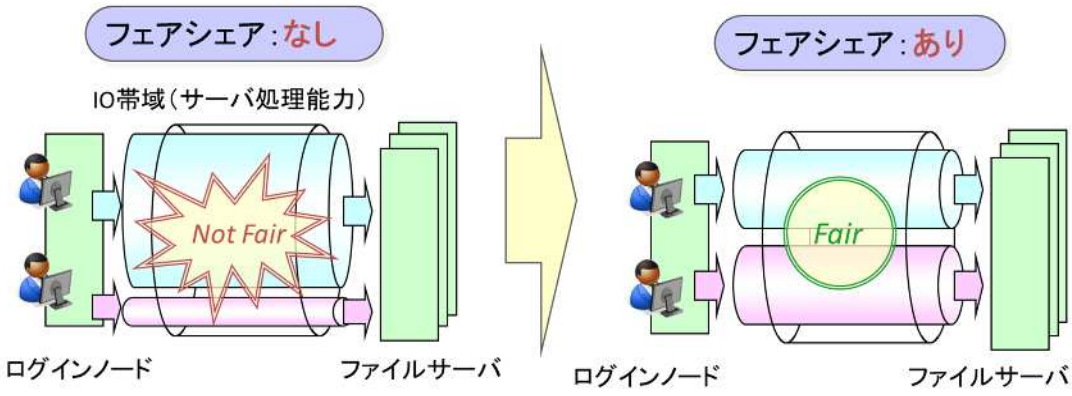

図 4-4 ユーザー間フェアシェア

4.5 ハードウェアの性能を最大限引き出す多様な言語処理系

4.5.1 言語処理系の概要

ハードウェア提供メーカーである富士通が開発した Fortran77/90 コンパイラ、C コンパイラ、C++ コンパイラに加え、デバッガ、プロファイラ、数学ライブラリを含む統合言語パッケージです(「表 4-7 言語処理系のコンポーネント一覧」を参照)。

各コンパイラは、SPARC64™IXfx の HPC 向け拡張機構 HPC-ACE を効果的に利用する最適化機能を有し、以下の特長を持ちます。

- 拡張レジスタ(浮動小数点 256 個、整数 188 個)の利用
  - 命令レベル並列度の向上
  - レジスタ不足に伴うレジスタ内容の退避・復元回数の削減
- SIMD 命令の活用
  - 自動ベクトル化を応用した SIMD 命令の自動生成
  - IF 文を含むループの SIMD 化(マスク付き SIMD 化)
- VISIMPACT(Virtual Single Processor by Integrated Multi-core Parallel Architecture)
  - マルチコア CPU でのコア間スレッド並列を実現し、高効率なハイブリッドモデル(MPI+自動並列)の実現を支援
  - コア間のハードバリア機能により、ソフトウェアバリアと比較してコア間の同期が最大 10 倍の高速化を実現
  - 共有 L2 キャッシュメモリにより、False sharing(*2)による性能劣化の影響を軽減し、自動ベクトル化技術を応用した、高効率な自動並列化
- セクタキャッシュ(ソフトウェア制御キャッシュ)の利用
  - セクタキャッシュ(*3)をユーザーが意識して利用するためのディレクティブ

表 4-7 言語処理系のコンポーネント一覧

コンポーネント	機能
Fortran77/90 コンパイラ	Fortran77/90 コンパイラ
C コンパイラ	C コンパイラ
C++コンパイラ	C++コンパイラ
MPI	MPI ライブラリ
XPFortran	並列化 Fortran 言語
BLAS/LAPACK/ScaLAPACK	BLAS/LAPACK/ScaLAPACK ライブラリ
SSL II 系数学ライブラリ	数値計算ライブラリ(SSL II 、SSL II スレッド並列機能、C-SSL II、C-SSL II スレッド並列機能、SSL II /MPI)
高速 4 倍精度基本演算ライブラリ	高速 4 倍精度基本演算ライブラリ
プログラミング支援ツール	アプリケーションプログラムの GUI 開発環境
デバッガ	GUI デバッギングツール
プロファイラ	プロファイリングツール

*2 個々にキャッシュを持つ複数のコアから構成されるシステムにおいて、コアごとに固有の非共有データが同一キャッシュライン上にある場合に、該当データに対する書き込みによってキャッシュラインの無効化が発生し、キャッシュ間データ転送が発生する現象。

*3 再利用性のあるデータを選択的にキャッシュに残す仕組みで、主記憶装置へのアクセスの回数削減によるプログラムの高速化が可能。

4.5.2 Fortran77/90 コンパイラ

高度な最適化機能を有する Fortran77/90 コンパイラです。翻訳オプションを指定するだけでコンパイラが自動的に並列処理を行うプログラムを作成する「自動並列化」や、指示行により並列処理を行う「OpenMP」に対応し、VISIMPACT との組み合わせで、高効率なハイブリッドモデル(マルチスレッド並列+MPI)を実現します。なお、「自動並列化」による並列処理機能は計算ノード内で有効です。

- プログラム診断(引数妥当性、未定義データ参照、配列オーバー)
- 精度敏感性診断
- エラーモニタリング
- トレースバックマップ
- 2GByte 以上のファイルの入出力
- 自動並列化
- OpenMP 並列
- 日本語メッセージ
- 4 倍精度浮動小数点演算サポート
- 高精度時間計測関数の提供
- 規格外の拡張仕様に関する警告メッセージ出力
- Red Hat Enterprise Linux 上で動作するクロスコンパイラの提供

4.5.3 C コンパイラ

日本語メッセージを出力する国際規格準拠の高性能 C コンパイラです。C89 規格、C99 規格をサポートしており、従来のプログラム資産を変更することなくコンパイルするだけで高速実行が可能となります。なお、「自動並列化」による並列処理機能は計算ノード内で有効です。

- GNU コンパイラ拡張仕様をサポート(一部)
- Fortran と同様の最適化技術を採用し、各種アプリケーションプログラムの高速実行を実現
- 自動並列化
- OpenMP 並列
- 日本語メッセージ
- 4 倍精度浮動小数点演算サポート
- 高精度時間計測関数の提供
- 規格外の拡張仕様に関する警告メッセージ出力
- Red Hat Enterprise Linux 上で動作するクロスコンパイラの提供

4.5.4 C++コンパイラ

C++言語で記述されたプログラムの開発を支援する高性能 C++コンパイラです。主要な機能は C コンパイラと同等の機能を提供します。

4.5.5 コンパイラの言語規格

以下「表 4-8 コンパイラの標準規格対応」にコンパイラの対応規格を記載します。

表 4-8 コンパイラの標準規格対応

コンパイラ種類	規格
Fortran77/90	JIS X 3001-1982 ISO/IEC 1539-1980 JIS X 3001-1:1994 ISO/IEC 1539:1991 JIS X 3001-1:1998 ISO/IEC 1539-1997 JIS X 3001-1:2009(Fortran 2003 規格)の一部 OpenMP API V3.0 仕様
C	ISO/IEC 9899:1990 ISO/IEC 9899:1999 OpenMP API V3.0 仕様
C++	ISO/IEC 14882:2003(export キーワード除く) OpenMP API V3.0 仕様

4.5.6 MPI

MPI は Open MPI をベースに開発したメッセージパッシングインタフェースライブラリです。

1 対 1 通信では、Tofu インターコネクトの性能を最大限に引き出すため、複数の転送方式を適切に切り替えて実行します。集団通信のうち、使用頻度の高い関数については、インターコネクトのトポロジー構成を意識した輻輳を抑える専用アルゴリズムを採用し、ハードウェア機能を使用した高速なバリア・リダクション演算が可能です。主な機能は以下の通りです。

- Tofu インターコネクト対応
  - 仮想 13 次元トーラスへのプロセス配置
  - 効率的な集団通信
  - ランクと座標を求める拡張 API
- MPI 2.1 に準拠
- 動的プロセス生成
  - MPI_Comm_spawn 関数
  - MPI_Comm_spawn_multiple 関数
- MPI-IO(並列入出力機能)
- 拡張されたコンパイラのサポート
  - Technical Computing suite に含まれる Fortran77/90、C、C++
  - GNU GCC に含まれる Fortran77/90、C、C++
  - g95
- MPMD 実行をサポート
- ノード間メッセージ通信方法として Eager 方式と Rendezvous 方式をサポート
- MPI 関数のフック機能
- デバッグ機能
  - デッドロック検出
  - 領域破壊検出
- 通信統計情報
- 拡張 RDMA インターフェース

4.5.7 XPFortran

XPFortran は、富士通が開発したデータ並列型言語で、翻訳指示行を Fortran ソースプログラムに挿入することによりプロセス並列化を可能とする並列化 Fortran 言語です。XPFortran には、並列処理を行うために以下のような機能を保有しています。

- グローバル空間とローカル空間
  - 物理的には各ノードのメモリを利用しながら、論理的には一つのメモリ空間に見えるグローバル空間を提供
  - 各プロセスにはローカル空間も存在し、ローカル空間に割り当てられているデータは高速にアクセス可能
- 同期・排他制御
  - 文の並びや DO ループの繰り返しなどの分割処理終了時に自動でバリア同期可能
  - 共通の資源を複数プロセスからアクセスする場合の排他制御文
- プロセス間データ転送
  - グローバル空間上の変数にアクセスした際に自動でプロセス間データ転送を実施
- マルチスレッド実行との併用
  - 自動並列/OpenMP によるマルチスレッド並列化と XPFortran によるマルチプロセス並列化によるハイブリッド実行可能

4.5.8 数学ライブラリ

数学ライブラリとして、BLAS/LAPACK/ScaLAPACK、SSL II(Scientific Subroutine Library II)系数学ライブラリおよび高速 4 倍精度基本演算ライブラリを提供します。これらのライブラリは、SPARC64™IXfx の性能を引き出すためのチューニングを行っています。

表 4-9 BLAS/LAPACK/ScaLAPAK の機能概要

ライブラリ名	バージョン	説明
BLAS	ー	・ 全ルーチンについて SPARC64™IXfx の性能を引き出すチューニング ・ Level3 の全ルーチン、Level2 の重要ルーチンについてスレッド並列ルーチンを提供
LAPACK	3.2.2	・ 重要ルーチンについてスレッド並列ルーチンを提供
ScaLAPACK	1.8	・ ScaLAPACK1.8 の追加機能を提供

表 4-10 SSL II 系数学ライブラリの機能概要

ライブラリ名	説明
SSL II	・ 以下の機能を持つ Fortran 用汎用サブルーチンライブラリ - 線形計算ライブラリ　- 固有値固有ベクトルライブラリ - 非線形計算ライブラリ　- 極値問題ライブラリ - 補間・近似ライブラリ　- 変換ライブラリ - 数値微積分ライブラリ　- 微分方程式ライブラリ - 特殊関数ライブラリ　- 擬似乱数ライブラリ ・ スレッドセーフ(複数スレッドから並列呼び出し可能)
SSL II 拡張機能	・ SSL II を拡張した Fortran 用サブルーチンライブラリ - 線形計算ライブラリ　- 固有値固有ベクトルライブラリ - 変換ライブラリ ・ スレッドセーフ(複数スレッドから並列呼び出し可能)
SSL II 拡張機能 II	・ SSL II を更に拡張した Fortran 用サブルーチンライブラリ - 線形計算ライブラリ　- 固有値固有ベクトルライブラリ - 変換ライブラリ　- 擬似乱数ライブラリ ・ スレッドセーフ(複数スレッドから並列呼び出し可能)
SSL II スレッド並列機能	・ OpenMP Fortran のサブルーチン ・ 並列効果の見込める重要機能に対して SMP 向け並列処理に適合したインターフェースで並列数値計算アルゴリズムを提供
C-SSL II	・ Fortran 版 SSL II の逐次機能のサブセットを C 言語インターフェースで利用可能 ・ スレッドセーフ(複数スレッドから並列呼び出し可能)
C-SSL II スレッド並列機能	・ Fortran 版 SSL II のスレッド並列機能のサブセットを C 言語インターフェースで利用可能
SSL II/MPI	・ MPI で並列化された 3 次元フーリエ変換ルーチン

表 4-11 高速 4 倍精度基本演算ライブラリ

ライブラリ名	説明
高速 4 倍精度 基本演算ライブラリ	・ 4 倍精度の値を double-double 形式で表現し高速に演算を行うライブラリ

4.5.9 プログラム支援ツール

プログラミング支援ツールは、アプリケーションプログラム開発の各種作業フェーズを支援する GUI 開発環境です。独自のファイルエクスプローラやエディタをはじめ、デバッガ、プロファイラなど高機能の開発ツールを実装しています。プログラミング支援ツールは主要な特徴を「表 4-12 プログラミング支援ツールの機能概要」に示します。

表 4-12 プログラミング支援ツールの機能概要

機能	内容
マネージャ機能	各機能の起動、各種メッセージの表示、サーバへのコマンドの投入機能などを行うメイン画面を提供します
プログラム作成支援機能	ファイルの作成/操作を行うファイルエクスプローラ、ファイル内容表示/編集のエディタを提供します
アプリケーションビルド支援機能	Makefile ファイルの作成/実行を行うビルダを提供します
アプリケーション実行支援機能	実行スクリプトの作成/実行を行うエグゼキュータを提供します
デバッグ機能	Fortran77/90、C、C++コンパイラで作成された逐次アプリケーション、並列アプリケーション(スレッド並列、MPI)に対して使用可能な GUI デバッギングツールです ・ アプリケーションの実行制御(実行中断、再開、ステップ・ネクスト実行) ・ アプリケーションのブレークポイントの設定、解除 ・ アプリケーションのウォッチポイントの設定、解除 ・ アプリケーションのバリアポイントの設定、解除(スレッド並列) ・ 変数値の表示、変更、変数値の自動表示設定などの変数操作 ・ スタックフレーム(呼び出し経緯)の表示とフレームの変更
プロファイリング機能	アプリケーションの実行性能情報を収集/解析するプロファイラを提供します ・基本プロファイラ サンプリングによりプログラム全体のチューニング情報（コスト）を収集します - 経過時間、ユーザ CPU 時間、システム CPU 時間の内訳など - サンプリングに基づくコスト、同期待ちコスト、MPI ライブラリ通信コスト - アプリケーション実行時のプロセッサ動作状況 - 手続きの呼出し経路とコスト - ソースコードの各行にコスト情報を付加して出力 ・詳細プロファイラ カウンタにより、プログラムの測定区間のチューニング情報を収集します - 測定区間の呼び出し回数、経過時間、CPU 時間の内訳など - 測定区間の MPI ライブラリの実行情報 - 測定区間のハードウェアモニタ情報
トレーサ機能	MPI 関数トレース情報を収集/解析する機能を提供します

図 4-5 プログラム支援ツールの利用イメージ

5. システムの利用イメージ

システムの利用の流れを以下に記載します。

1. ログイン・ファイル転送
   ログインノード群に SSH プロトコルでログインし、必要なファイルは scp や sftp を用いてファイル転送します。セキュリティ強化のため、SSH ログインや scp/sftp は、公開鍵認証方式を用います。
2. ファイル編集・操作
   ログインノードで、プログラムに必要なソースコードや Makefile 等を作成・編集します。
3. プログラムの作成(コンパイル・リンク)
   Fortran/C/C++コンパイラや MPI 等の並列ライブラリを使用しプログラムを作成します。導入システムは、ログインノードと計算ノードでアーキテクチャが異なるため 2 種類のコンパイラがあります。
   【クロスコンパイラ利用時】
   ログインノードで、クロスコンパイラを用いてコンパイル・リンクをします。
   【オウンコンパイラ(ネイティブコンパイラ)利用時】
   ログインノードからインタラクティブジョブを起動し、インタラクティブノード上でオウンコンパイラを用いてコンパイル・リンクします。計算ノードとインタラクティブノードは、同一環境のためバイナリ互換性があります。
4. 実行
   ログインノードからバッチジョブ投入コマンド(pjsub)にてジョブを投入します。またインタラクティブジョブ実行は、投入コマンド(pjsub –interact)にてジョブ実行します。
5. デバッグ・チューニング
   統合支援ツールを使用して、デバッガや、プロファイラを起動します。プロファイラは、性能情報を取得します。性能情報をもとにチューニングをします。

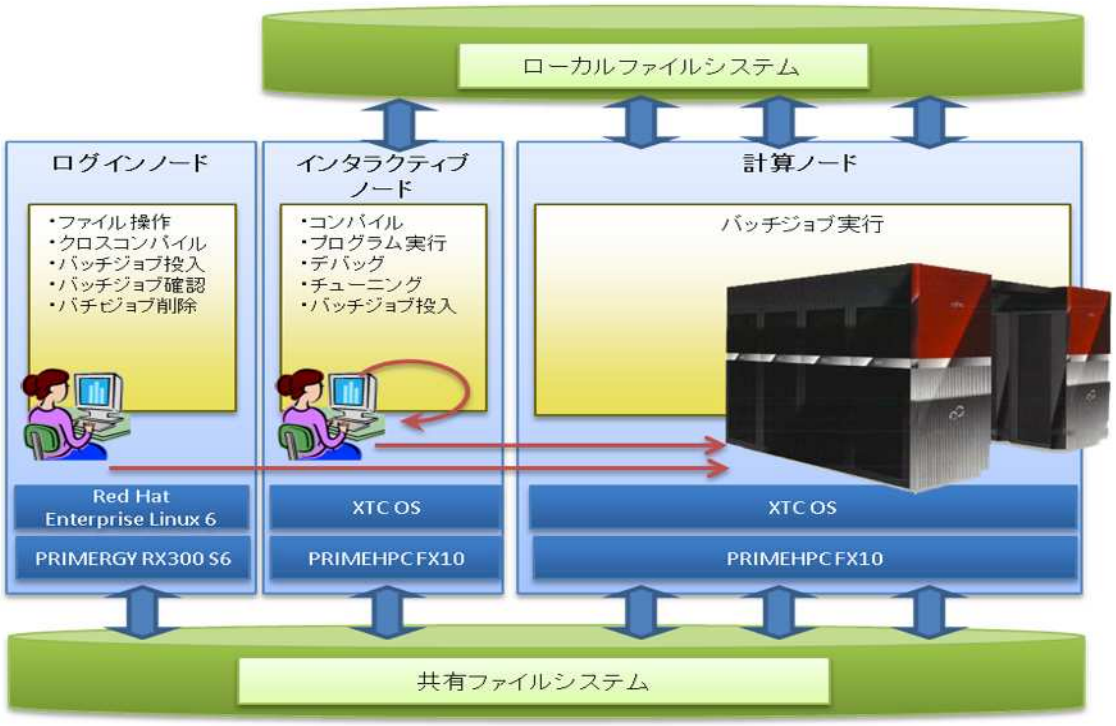

図 5-1 システムの利用イメージ