White paper
FUJITSU Supercomputer PRIMEHPC FX1000 先進のソフトウェア

富士通株式会社

目次

HPC ミドルウェアの特長

HPC ミドルウェアの概要

富士通では、スーパーコンピュータ「京」(*1) をはじめ、エクサスケール規模のシステムの運用管理とアプリケーション利用環境を提供する HPC ミドルウェア「FUJITSU Software Technical Computing Suite」（以降、Technical Computing Suite と表記）を開発しています。FUJITSU Supercomputer PRIMEHPC FX1000（以降、PRIMEHPC FX1000 と表記）に搭載した ARM (*2) アーキテクチャの CPU は、幅広いソフトウェアに対応する汎用性を有しています。Technical Computing Suite では、これまでの HPC ミドルウェア開発の経験と技術をコミュニティに展開し、コミュニティとともに HPC 利用者の利便性を高めています。

以下に、PRIMEHPC FX1000 での HPC ミドルウェアの体系と概要を示します。

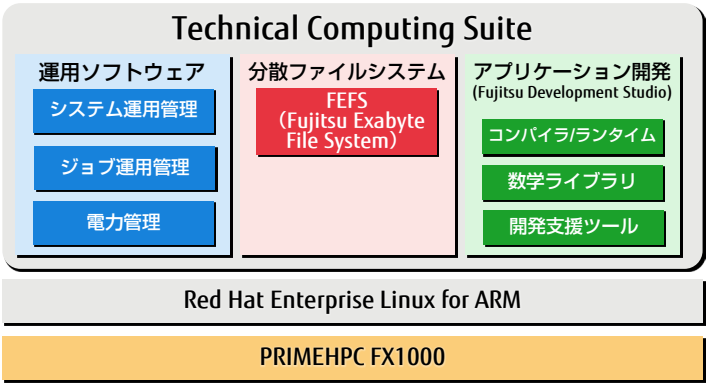

図 1　HPC ミドルウェアの体系

■ オペレーティングシステム

RHEL 系ディストリビューションを OS に採用しているため、PC クラスタと同様の使い勝手で PRIMEHPC FX1000 を利用できます。また、Python/Ruby 等のスクリプト言語を利用して運用性を向上させることもできます。さらに、PRIMEHPC FX1000 に搭載している OS 用アシスタントコアを、以下の目的で活用しています。

- **システムノイズの削減**

  システムノイズ（OS ジッタ）はジョブ実行の外乱になります。数万ノードの計算では、外乱がノード数に比例して増大し性能低下につながる場合があります。

  一般的な Linux サーバで 10^{-3} 台であるシステムノイズ（平均ノイズ率）が PRIMEHPC FX1000 では 10^{-5} 台であることに加えて、従来製品の 5 分の 1 に削減し、超並列計算プログラムの並列度アップとスループットを向上させています。

表 1 システムノイズ比較

	PRIMEHPC FX1000	PRIMEHPC FX100	PC クラスタ
平均ノイズ率	1.02×10^{-5}	5.10×10^{-5}	1.08×10^{-3}

- **非同期通信処理の実現**

  MPI ノンブロッキング通信機能を用いて、ユーザープログラム内で MPI の通信処理を実行している間（通信を開始する関数が呼び出されたのちに、通信を待ち合わせする関数が呼び出されるまでの間）に演算処理を実行し、通信と演算をオーバーラップさせて実行する方法があります。

  PRIMEHPC FX1000 では、アシスタントコアに MPI 非同期通信スレッドを割り当て、通信と演算のオーバーラップを実現しています。さらに、CPU 高負荷時にも MPI の非同期通信スレッドを高優先度で動作させる処理を Linux カーネル部に実装し、MPI 通信の応答保証機能を実現しています。

- **ファイル I/O 処理とジョブの分離強化**

  インターコネクトで相互接続された計算ノードから、InfiniBand などのデータ転送ネットワークに接続された FEFS（Fujitsu Exabyte File System）を使用するために、I/O 処理のルーティングを行っています。「京」や PRIMEHPC FX10 では IO 専用ノードで I/O 処理のルーティングを行っていますが、PRIMEHPC FX1000 ではアシスタントコアでルーティング処理を実行し IO 専用ノードを不要にしています（計算ノード兼中継ノード）。また、計算ノードの FEFS クライアント機能をアシスタントコアで処理し、ファイル I/O 処理とジョブの通信処理の分離を実現しています。

■ 運用ソフトウェア

多数の利用者が同時にプログラムを実行できるようにするため、効率よく計算機資源を割り当てるソフトウェアです。また、大規模システム全体の管理を容易にし、システムの一部で故障が発生してもシステム全体では運用を継続できるようにします。

- **システム運用管理**

  数百～数十万の計算ノードを、統一された集中管理ビューに表示します。集中管理ビューでは、システムの起動・停止を制御し、異常箇所を監視・切り離すことで、システムの自動運転を可能にしています。「京」で実現された大規模システムの管理基盤技術をさらに拡張し、PRIMEHPC FX1000 や PC クラスタのハイブリッド構成を管理できるようにしています。

- **ジョブ運用管理**

  数万ノードを使用する単一ジョブの投入だけでなく、多数の利用者のジョブ投入、豊富なジョブバリエーションを考慮し、大規模システムでの効率的なジョブスケジューリングを可能にしています。

- **電力管理**

  ハードウェアの電力性能を最大限に利用する電力管理機能を新たに提供しています。

■ 分散ファイルシステム（FEFS）

「京」のファイルシステムをベースにした、大規模システムに対応する高速なファイルシステムです。FEFS には以下の特長があり、大規模環境でも遅延なく高速なファイルアクセスが可能です。

- 数千台規模のストレージ装置（OSS/OST）(*3) から構成されるファイルシステムを、十万台規模のクライアントから安定した状態で利用できます。
- エクサバイトクラスの大容量までサポート可能なファイルシステムです。
- 一部の利用者が大量のファイル I/O を行っても、ほかの利用者のファイル I/O が停滞しません。

■ アプリケーション開発（Fujitsu Development Studio）

Fortran、C、C++ 言語で記述された科学技術計算向けプログラムの開発（コンパイル、デバッグ、チューニングなど）および実行を支援する統合的なソフトウェア群です。自動並列化、OpenMP、MPI といった並列化技術もサポートしています。

- **コンパイラ／ランタイム、数学ライブラリ**
  Arm の新しい HPC 拡張命令、Scalable Vector Extension（以降、SVE と表記）や新しい言語規格に対応しています。PRIMEHPC FX1000 のハードウェア機能を活用して、科学技術計算向けプログラムを並列処理によって高速に実行できます。
  PRIMEHPC FX10 や PRIMEHPC FX100 のアプリケーションに対して上位互換を担保するオプションをサポートすることで、ユーザー資産の継承を可能としています。

- **開発支援ツール**
  アプリケーション開発環境では、ラージページメモリ獲得・解放関数の処理コスト情報も利用できます。オペレーティングシステムがアプリケーション層から収集した情報をグラフィカルに参照することで、超並列コンピュータのアプリケーション・チューニングが効率よく行えます。

Technical Computing Suite の新たな取り組み

エクサスケールの大規模運用にも十分に応える性能改善に加えて、中小規模センターや企業ユーザーによるプライベートなスーパーコンピュータシステムの運用における使い勝手の向上にも取り組んでいます。

■ アプリケーションの高速実行の実現

開発支援ツールの利便性向上や性能改善として以下の特長があります。

- コンパイラの最適化を強化し、SIMD 効率を向上させて SVE を活用したアプリケーションの高速実行を実現しています。
- Eclipse (*4) を活用したデファクトスタンダードな開発環境でユーザービリティの向上を実現しています。

■ 並列計算機システムの効率利用

運用ソフトウェアでは、機能強化によるシステムの運用稼働率の向上を図っています。

- ジョブ投入時にユーザージョブの実行に特化した仮想化環境の選択が可能です。
- 従来比 10 倍のスケジューリング性能とジョブ情報表示レスポンスの向上を実現しています。
- ローリングアップデート機能により、システム全体の運用を止めずに部分的な保守を可能とし、保守効率化の向上を実現しています。
- 他社スケジューラーと互換性ある I/F を提供する豊富な API が使用できます。ユーザー独自のスケジューリングアルゴリズムをアドオン可能とし、共同利用センターなどから寄せられる多種多様の要件への対応力をさらに向上しています。

■ 電力管理機能による省電力運用

ジョブの消費電力を計算資源として扱ってジョブスケジューリングを行い、センターの許容電力を超過することなく計算機センターを運用できます。

エンドユーザーは、電力の計測・制御 API を使用することで、アプリケーションに最適な電力効率チューニングを行うこともできます。

*1 理化学研究所と富士通が共同で開発。「京」は理化学研究所の登録商標です。

*2 ARM、ARM ロゴは ARM Ltd.またはその関連会社の商標または登録商標です。

*3 OSS（Object Storage Server）は、ファイルデータを制御するサーバです。OST（Object Storage Target）は、OSS に接続するストレージ機器です。

*4 Eclipse は米国およびその他の国における Eclipse Foundation, Inc. の商標もしくは登録商標です。

アプリケーションの開発

アプリケーション開発環境の概要

PRIMEHPC FX1000 には、RHEL に含まれる GCC などのアプリケーション開発環境に加えて PRIMEHPC FX1000 の性能を最大限に引き出す Fujitsu Development Studio が搭載されています。

PRIMEHPC FX1000 は、メニーコアで構成されるメモリ空間を共有した計算ノードを Tofu インターコネクト D で結合した階層型の分散並列スーパーコンピュータです。

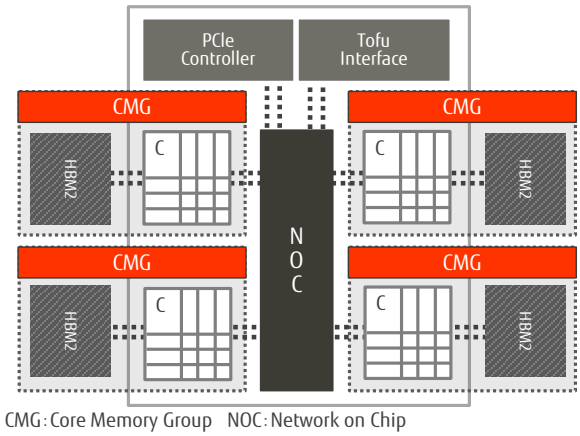

図 2　PRIMEHPC FX1000 の計算ノード

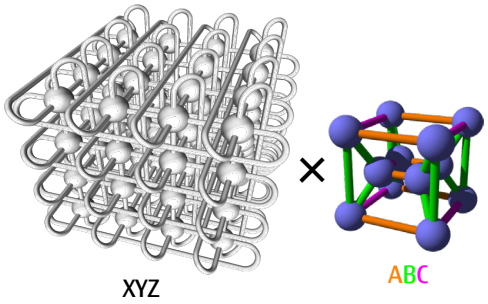

図 3　Tofu インターコネクト D

Fujitsu Development Studio は、PRIMEHPC FX1000 の性能を最大限に引き出せるように、コア、ノード内、ノード間の各階層の高速化に対応した Fortran/C/C++コンパイラ・ランタイム、数学ライブラリ、および開発支援ツールを提供します。

Fujitsu Development Studio の構成

Fujitsu Development Studio の C/C++コンパイラは、Clang/LLVM と互換性がある clang モードで動作させることができます。clang モードを使うことで、容易にオープンソースのアプリケーションを動作させることができます。

さらに、Fujitsu Development Studio は、さまざまな利用形態に対応できるように、PRIMERGY で動作するクロスコンパイラと、PRIMEHPC FX1000 で動作するネイティブコンパイラの 2 種類のコンパイラを提供します。

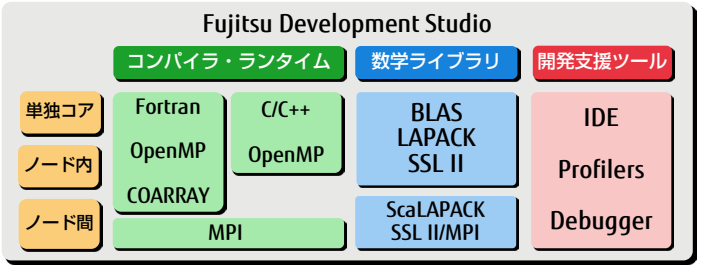

図 4　Fujitsu Development Studio の構成

最新の標準規格および業界標準仕様への対応

PRIMEHPC FX1000 の上で最先端の規格を用いたアプリケーション開発を可能にするため、またオープンソースのアプケリーションを容易に動作させるために、Fujitsu Development Studio は最新の標準規格および業界標準仕様をサポートしています。

表 2 サポート標準規格一覧

言語	サポート規格仕様
Fortran	ISO/IEC 1539-1:2018（Fortran 2018 規格）の一部 ISO/IEC 1539-1:2010（Fortran 2008 規格） ISO/IEC 1539-1:2004、JIS X 3001-1:2009（Fortran 2003 規格） ISO/IEC 1539-1:1997、JIS X 3001-1:1998（Fortran 95 規格） Fortran 90 規格および Fortran 77 規格
C	ISO/IEC 9899:2011（C11 規格） ISO/IEC 9899:1999（C99 規格） ISO/IEC 9899:1990（C89 規格） ※GNU コンパイラの拡張仕様もサポート
C++	ISO/IEC 14882:2017（C++17 規格）の一部 ISO/IEC 14882:2014（C++14 規格） ISO/IEC 14882:2011（C++11 規格） ISO/IEC 14882:2003（C++03 規格） ※GNU コンパイラの拡張仕様もサポート
OpenMP	OpenMP API Version 5.0 の一部 OpenMP API Version 4.5
MPI	Message-Passing Interface Standard Version 3.1 および 4.0（仮称）の一部

コア内の高速化機能

■ ベクトル化機能

Fujitsu Development Studio の Fortran/C/C++コンパイラは、PRIMEHPC FX1000 に搭載されている A64FX の SVE（Scalable Vector Extension）機能を活用するベクトル化機能を搭載します。

SVE の活用により、IF 文や数学関数を含むような複雑なループをベクトル化することができます。加えて、これまでベクトル化の効果を得ることが難しかった回転数の少ないループを、SVE のマスク付きの SIMD 命令を利用することでベクトル化し、高速に実行することができます。さらに、SVE の特長である異なる SIMD レジスタ幅のマシン間で可搬できる SIMD レジスタ幅可変の SIMD バイナリの作成も可能です。

図 5 に SVE 命令のデータアクセスパターンを変えた場合の性能を示します。

PRIMEHPC FX1000 のデータアクセスは、PRIMEHPC FX100 のデータアクセスに比べて 2.0～2.3 倍程度高速です。

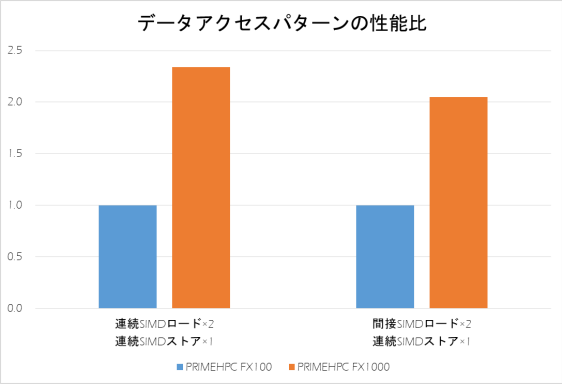

図 5　データアクセスパターンの性能比

■ データの型に応じた SIMD レジスタの活用

A64FX では、512bit 幅の SIMD レジスタを用いて、浮動小数点演算の場合、倍精度: 8 要素、単精度: 16 要素、半精度: 32 要素、整数の場合、8 バイト整数: 8 要素、4 バイト整数: 16 要素、2 バイト整数: 32 要素、1 バイト整数: 64 要素を、並列に計算できます。Fujitsu Development Studio の Fortran/C/C++コンパイラは、この機能を活用することで、浮動小数点計算を中心とする科学技術計算のアプリケーションに加えて、整数計算を含む幅広いアプリケーションのパフォーマンスを向上させます。

■ ソフトウェアパイプライニング機能

Fujitsu Development Studio の Fortran/C/C++コンパイラは、ソフトウェアパイプライニングと呼ばれる高度な命令スケジューリング機能を搭載しています。

ソフトウェアパイプライニング機能は、プログラム中のループに対して、演算器の数や各種命令のレイテンシ、レジスタ数などを考慮して、ある回転と次回転以降の命令を重ね合わせるように命令を並べ替えることで、ループ内の命令の並列実行率を高めて高速化します。

さらに、Fujitsu Development Studio の Fortran/C/C++コンパイラは、ソフトウェアパイプライン化を促進するために、以下の機能を実現しています。

- **ループ分割機能**

  ループ内の文の数が多くなると、レジスタなどが不足して、ソフトウェアパイプラインを適用できなくなります。ループ分割機能は、文数の多いループを、レジスタ数、メモリアクセス数、キャッシュ効率などを考慮してソフトウェアパイプラインが適用できる複数のループに分割し、ソフトウェアパイプライン化を促進します。

- **数学関数の非分岐型インライン展開機能**

  ループの内部から数学関数を呼び出している場合、インライン展開したとしても処理に条件判定による分岐が必要なため、ソフトウェアパイプライン化できません。Fujitsu Development Studio の Fortran/C/C++コンパイラは、三角関数、指数関数、さらに平方根と除算について、三角関数補助命令、指数関数補助命令、および逆数近似命令を活用して分岐を含まないインライン展開を実現することで、ソフトウェアパイプライン化を促進します。

  図 6 に、基本的な演算や数学関数の性能を示します。基本的な演算や数学関数のループでは、PRIMEHPC FX1000 は、PRIMEHPC FX100 に比べて 2.3～4.2 倍の速度で処理できます。

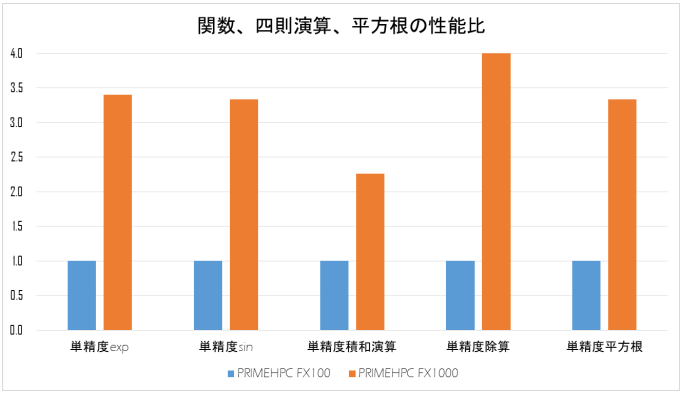

図 6　関数、四則演算、平方根の性能比

■ インテリジェントプリフェッチ機能

A64FX には、キャッシュミスによる速度低下を防ぐために、単純なメモリアクセスパターンに対してハードウェアが自動的に先行してメモリフェッチを実施するハードウェアプリフェッチ機能と、複雑なメモリアクセスパターンにも対応できるプリフェッチ命令の 2 つのプリフェッチ機能が搭載されています。

Fujitsu Development Studio の Fortran/C/C++コンパイラには、単純なメモリアクセスパターンは効率がよいハードウェアプリフェッチ機能に任せ、ハードウェアプリフェッチ機能が対応できないメモリアクセスパターンに対してプリフェッチ命令を生成するインテリジェントプリフェッチ機能が搭載されています。

さらに、インテリジェントプリフェッチ機能は、ループごとにプリフェッチする距離を調整する機能を搭載しており、より効率がよいプリフェッチを実現します。

■ セクタキャッシュ指示機能

A64FX には、キャッシュの一部を高速メモリとして扱えるセクタキャッシュ機能が搭載されています。

Fujitsu Development Studio の Fortran/C/C++コンパイラには、簡素な指示行でセクタキャッシュに搭載するデータを指示する機能が搭載されています。セクタキャッシュ指示機能を使用することで、再利用性のあるデータを高速メモリ上に保持することが可能となり、性能低下を防ぐことができます。

■ FP16（半精度浮動小数点型）のサポート

A64FX と Fujitsu Development Studio の Fortran/C/C++コンパイラは FP16 型をサポートしています。

機械学習などの AI では、FP16 型に演算精度を落とすことが可能です。FP16 の SIMD 演算を利用すれば、演算性能が従来の FP32（単精度浮動小数点型）の 2 倍に向上します。FP16 を活用することで、従来の科学技術計算分野のアプリケーションだけでなく、AI のアプリケーションでも性能の向上が期待できます。

■ コア内の高速化機能による数学ライブラリの高速化

Fujitsu Development Studio は、コア内の高速化機能を適用した高速数学ライブラリとして、線型代数分野で著名な BLAS、LAPACK、広い分野のアルゴリズムをサポートし R&D ユーザーに幅広く利用されている富士通の数学ライブラリ SSL II、さらに 4 倍精度の値を double-double 形式で扱うことで高速化する高速 4 倍精度基本演算ライブラリを提供します。

これらの数学ライブラリをアプリケーションから呼び出すことで、容易にアプリケーションの高速化を実現します。

表 3　数学ライブラリ（逐次版）（スレッドセーフ）

BLAS、LAPACK	米国で開発され Netlib で公開されているデファクトスタンダードな線形計算ライブラリ。BLAS 約 80 種、LAPACK 約 400 種ルーチン。 BLAS の一部で FP16 を独自にサポート。
SSL II	幅広い分野をカバーする、約 300 種の Fortran ルーチンのライブラリ。
C-SSL II	SSL II への C インターフェース。
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、演算を行うライブラリ。一部スレッド並列ルーチンを含む。

図 7 に BLAS の行列積ルーチンの性能を示します。

A64FX では、BLAS の行列積ルーチンは、単精度は倍精度の 2 倍の性能、FP16 ではさらにその 2 倍の性能を実現しています。

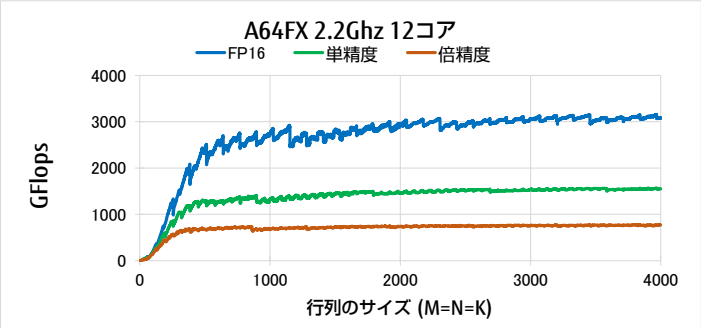

図 7　行列積の性能

ノード内並列による高速化

■ スレッド並列化機能

PRIMEHPC FX1000 のノード内のメニーコアを活用するため、Fujitsu Development Studio の Fortran/C/C++コンパイラは、ループに対する自動並列化とスレッド並列用の業界標準仕様である OpenMP をサポートしています。

A64FX は、スレッド並列のための高速なハードバリア機構を搭載しています。Fujitsu Development Studio のランタイムライブラリは、このハードバリアを活用することで、高速なスレッド並列実行を実現します。

■ ネットワーク階層に対応した MPI の高速化

A64FX は、CMG（Core Memory Group）を 4 つ搭載し、それらをリングバスによって接続する NUMA（Non-Uniform Memory Access）アーキテクチャを採用しています。PRIMEHPC FX1000 では、A64FX を Tofu インターコネクト D で結合するため、CMG を単位としたプロセス並列化の場合、Tofu インターコネクト D と A64FX 内のリングバスによる階層ネットワーク構造で結合されることになります。

Fujitsu Development Studio の MPI の Alltoall のアルゴリズムには、図 8 に示すように Tofu インターコネクト D に加えて、A64FX のリングバスにおける通信路の競合が最も少なくなるように最適化したアルゴリズムが実装されています。これにより、図 9 に示すように A64FX の性能をリングバスの性能を最大限に引き出して高速化を実現しています。

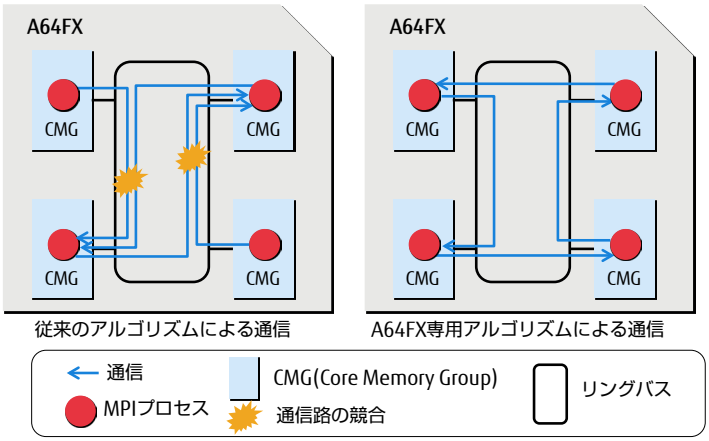

図 8　A64FX 専用集団通信アルゴリズム

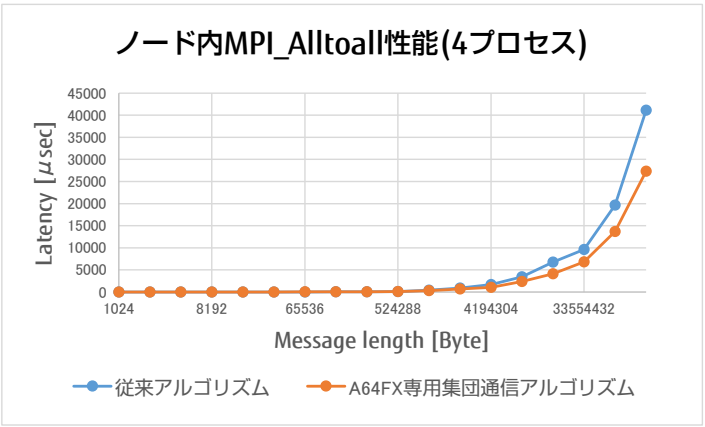

図 9　A64FX 専用 Alltoall アルゴリズムの性能

■ スレッド並列機能による数学ライブラリの高速化

Fujitsu Development Studio は、スレッド並列化した高速数学ライブラリとして BLAS、LAPACK と、SSL II を提供します。

これらの数学ライブラリをアプリケーションから呼び出すことで、スレッド並列による高速化を実現します。

表 4　数学ライブラリ（スレッド並列版）	
BLAS、LAPACK	逐次版と同一インターフェース。LAPACK をタスク並列化した PLASMA を含む。Python（NumPy、SciPy）からも利用可能。
SSL II スレッド並列機能	重要機能約 80 ルーチンのスレッド並列版。混在使用できるように逐次版 SSL II と別インターフェース。
C-SSL II スレッド並列機能	SSL II スレッド並列機能への C インターフェース。

ノード間並列による高速化

■ Tofu インターコネクト D を利用した MPI の高速化

PRIMEHPC FX10 や PRIMEHPC FX100 に搭載されている Tofu、Tofu2 では、ノードあたりのネットワークインターフェース数は 4 でしたが、PRIMEHPC FX1000 の Tofu インターコネクト D では、ネットワークインターフェース数を 6 に増やしました。このため、従来の 4 方向同時通信から、6 方向同時通信へと、通信機能が大幅に強化されています。

Fujitsu Development Studio の MPI は、ノード当たりの通信バンド幅が必要な Allgather や Bcast などの集団通信アルゴリズムを、この機能強化を活かして改良しました。

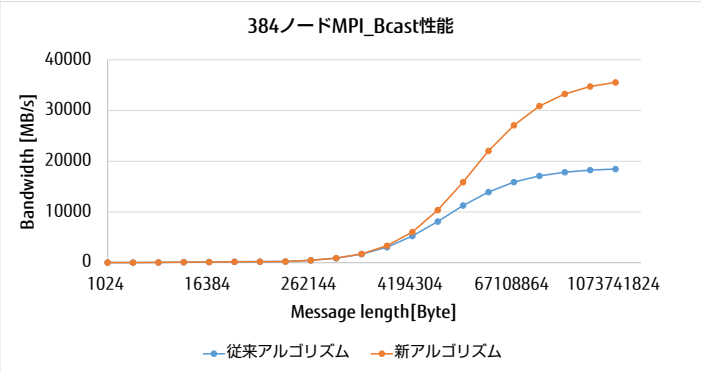

図 10　6 方向対応した専用集団通信アルゴリズムの性能(Bcast)

■ MPI 並列機能による数学ライブラリの高速化

Fujitsu Development Studio は、MPI 並列化した高速数学ライブラリとして ScaLAPACK と SSL II/MPI を提供します。

これらの数学ライブラリをアプリケーションから呼び出すことで、MPI 並列による高速化を実現します。

表 5　数学ライブラリ（MPI 並列版）	
ScaLAPACK	BLAS、LAPACK の機能を MPI で並列化したライブラリ。約 200 種ルーチン。
SSL II/MPI	3 次元 FFT 機能の MPI 並列版。

アプリケーション開発支援ツール

■ 統合開発環境

Fujitsu Development Studio は、オープンソースの統合開発環境である Eclipse、および科学技術計算分野のアプリケーション開発向けに拡張するプラグイン Parallel Tools Platform（PTP）を搭載しています。

PTP を導入することで、ジョブスケジューラーと連携して、ソースコード編集、コンパイル、ジョブ投入、ジョブの状態確認、性能情報採取・表示を Eclipse 上でシームレスに実施できます

■ プロファイラ

- 性能解析の手順

性能解析の手順は、利用者の状況に応じて二分されます。状況に応じて、以下のように「基本プロファイラ」、「詳細プロファイラ」および「CPU 性能解析レポート」を使用することで効率的な性能解析を行うことができます。

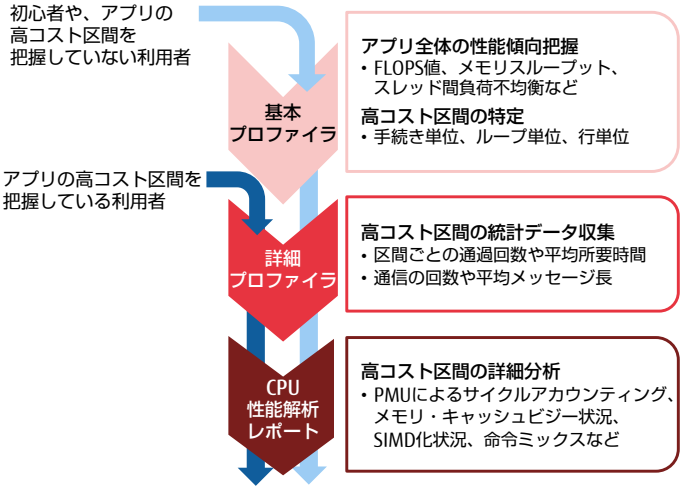

図 11　性能解析の手順

- プロファイラによる段階的な性能解析

アプリケーションの性能解析を、「基本プロファイラ」、「詳細プロファイラ」および「CPU 性能解析レポート」で支援します。

基本プロファイラは、サンプリング方式でアプリケーション全体の性能傾向・コスト分布情報を収集し、収集したコスト分布は、手続き単位・ループ単位・行単位で表示することで、アプリケーションの性能の概要を把握することを支援します。

基本プロファイラの結果からホットスポットを特定し、その部分のより詳細な性能情報を得るために、詳細プロファイラを使用します。詳細プロファイラを用いると、指定した区間の性能状況、MPI の統計的通信状況を把握できます。

さらに、CPU 性能解析レポートを用いると、CPU に内蔵された PMU（Performance Monitoring Unit）カウンターの情報をグラフや表の形式で体系立てて分かりやすく表示します。CPU 性能解析レポートによりアプリケーションのボトルネックを詳細に把握することができます。

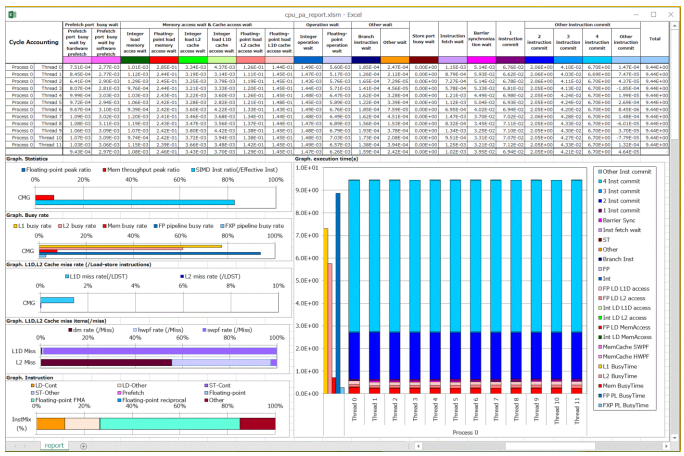

図 12　CPU 性能解析レポートの例

■ 並列実行デバッガ

Fujitsu Development Studio は、大規模並列処理でよく発生するトラブル（異常終了やデッドロックなど）の調査を、実際のアプリケーションの実行シーンで支援できる 3 つのデバッグ機能を搭載しています。なお、並列実行デバッガのデバッガエンジンには、デファクトスタンダードな GDB を採用しています。

- **異常終了調査機能**
  大規模並列実行時にシグナルが発生して異常終了したアプリケーションの実行情報（バックトレース、シグナル発生時アドレスを含む命令列、レジスタ内容、メモリマップ）を取得し、得られた情報を見やすく要約して表示します。
- **デッドロック調査機能**
  大規模並列実行時にアプリケーションが終了しない、または応答がない場合に、アプリケーションのすべてのプロセスに対して、アプリケーションの実行情報（バックトレース、フレームごとのローカル変数の値および引数変数の値、メモリマップ）を取得し、その結果を見やすく要約して表示します。
- **コマンドファイルによるデバッグ機能**
  GDB のコマンドファイルを使用したデバッグを、ジョブ環境でも実行できるようにする機能です。プロセスごとに異なるコマンドファイルを指定できるので、アプリケーションの特性に応じた柔軟なデバッグが実行できます。

ジョブ運用管理

ジョブスケジューラー開発への取り組み

富士通では、大規模システムでも利用者がストレスを感じることなく大量のジョブを操作できるバッチジョブ実行環境（ジョブスケジューラー）の開発に取り組んできました。

PRIMEHPC FX1000 のジョブスケジューラーは、約 8 万ノード規模で実績のある「京」の大規模ジョブスケジューラー、小規模センターのシステム稼働率改善や PC サーバのジョブと組み合わせて実行する運用機能を強化した PRIMEHPC FX100 のジョブスケジューラーをベースに、大規模システムで求められる以下のジョブ運用機能を開発しました。

- **多数のジョブを確実に制御する機能（大規模対応）**
  大規模システムは、多数のジョブを同時に制御する必要があります。また、多数のジョブが存在する場合にも利用者がストレスを感じることなくシステムを利用できるようにする必要もあります。このためにジョブスケジューラーの性能を向上しました。
- **共同利用センターの多種多様な要件への対応（運用性）**
  大規模システムは、共同利用センターで運用されることが多く、センターごとにジョブ実行に関する運用ポリシーがそれぞれ異なり、細かく取り決められています。PRIMEHPC FX1000 では、ジョブ実行に関するセンターごとの運用ポリシーに応じてジョブスケジュールを自由にカスタマイズできるように、以下の機能を開発しました。
  - ・ジョブ実行ユーザーの使い勝手向上：ジョブ実行環境
  - ・きめ細かなセンター運用ができるカスタマイズ機能

また、共同利用センターでの運用では、利用するユーザー数が多く、大小さまざまなノード規模を要求するジョブが投入されます。さまざまなジョブが混在する環境において、よりシステムの稼働率を向上するために以下の機能を開発しました。

- **高稼働率機能：Adaptive Elapsed-time job scheduling**
  大規模なジョブを実行するために大量にノードを確保すると一時的に稼働率が低下するなど、高優先度のジョブによって稼働率が低下する場合があります。ジョブの実行時間をジョブのスケジューリング結果に応じて変更できるようにすることで、隙間時間を利用してシステムの稼働率をより向上できるようにしました。

ここでは、PRIMEHPC FX1000 の特長である、これらの機能強化点について紹介します。

多数のジョブを確実に制御する機能（大規模対応）

多数のジョブが存在しても利用者がストレスなく利用できるようにするため、ジョブスケジューラーの性能を向上しました。

■ ジョブのスケジューリング性能の向上

スケジューリングのアルゴリズムを、多数の利用者による小規模ジョブが大量に投入されることを考慮したものに改良しています。これによって、例えば 1 万個の小規模ジョブが同時に投入されても数秒でスケジューリングが完了します。

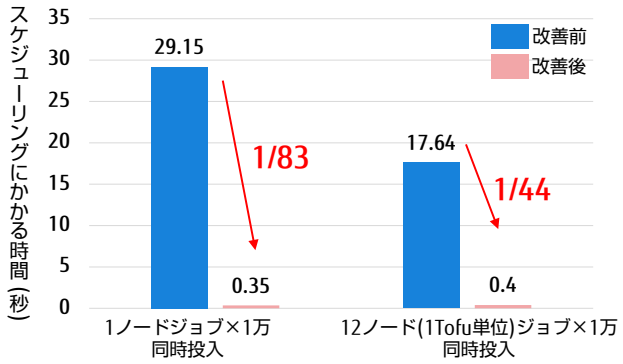

図 13　スケジューリング時間の短縮

■ ジョブ情報の参照性能の向上

ジョブ数が多くなるとジョブ情報の参照に時間がかかるという課題がありました。OSS（オープンソースソフトウェア）のデータベースを活用した情報検索処理の改良によって、例えば 100 万個のジョブが存在する場合でもすばやく情報を参照できるように高速化しました。

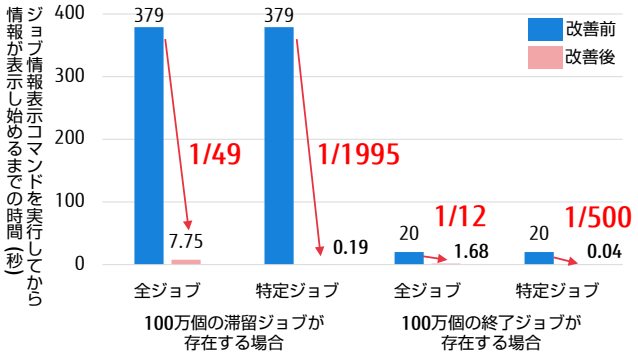

図 14　ジョブ情報参照時間の短縮

■ ジョブの操作性能の向上

ジョブに関する操作（削除、固定、およびパラメーター変更）の処理を改良し、多数のジョブを対象にした場合でもストレスのない操作を可能にしました。

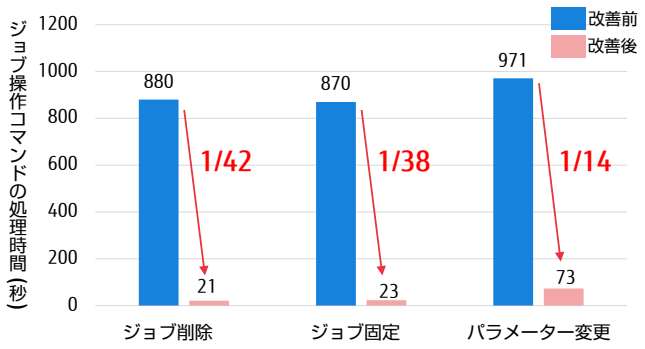

図 15　ジョブ操作時間の短縮

ジョブ実行ユーザーの使い勝手向上：ジョブ実行環境

センターごとにアプリケーションにマッチした独自のジョブ実行環境を配備して利用可能とする機能を開発しました。この機能により、ユーザーはシステムに配備されたジョブ実行環境の中から自身のジョブを実行するのに適した環境を選択して、使用することができます。また、管理者によってあらかじめシステムに配備されるジョブ実行環境に加えて、ユーザー自身が用意したジョブ実行環境を利用するための機能も備えています。

ジョブ実行環境として以下を利用することができます。

- Docker モード

  ユーザーのジョブスクリプトを含むすべてのジョブプログラムを、ジョブ投入時に指定されたコンテナイメージから起動したコンテナ上で実行します。Docker イメージとしてパッケージングされた OS をそのまま利用してジョブを実行でき、アプリケーション実行に必要なソフトウェア環境の導入が容易になります。

- KVM モード

  ユーザーのジョブプログラムを KVM 上で実行するモードにも対応していきます。KVM モードでは、仮想マシン管理のためにオープンソースソフトウェア QEMU を使用して KVM を利用します。ハードウェアの仮想化支援を利用することで高速なハードウェア仮想化環境上でのジョブ実行が可能となります。OS カーネル層のモジュール開発者などプログラム実行に特権を必要とするユーザーに対して有効です。

きめ細かなセンター運用ができるカスタマイズ機能

利用者の環境に最適な運用を実現するため、PRIMEHPC FX1000 のジョブスケジューラーでは、以下のカスタマイズ機能を提供しています。

- 任意の資源スケジューリングを実現するカスタム資源機能
- ジョブ実行フローを制御するフック機能
- 統計情報の選択、新規追加を可能にするジョブ統計情報カスタマイズ機能
- 独自のスケジューリングアルゴリズムを組み込むためのスケジューラープラグイン機能
- 独自のコマンドや運用支援ツールを作成するためのコマンド API

以下に、PRIMEHPC FX1000 のジョブスケジューラーで強化された機能について紹介します。

■ カスタム資源機能

ソフトウェアライセンスやジョブの消費電力予測値などを任意の資源として定義します。例えば、同時に利用可能なジョブ数／プロセス数に制限があるソフトウェアライセンス（フローティングライセンス）や、利用可能なノードに制限があるソフトウェアライセンス（ノードロックライセンス）をカスタム資源として定義し、ジョブ投入時に必要なソフトウェアライセンス数を要求することで、利用可能な時刻・計算ノードで実行が開始されるようジョブがスケジューリングされます。

■ フック機能

フック機能は、ジョブ実行の処理で発生する特定のイベントを契機として、管理者が用意した処理（出口処理）を実行するための仕組みです。出口処理により、ジョブの実行を制御したり、ジョブの属性を変更したりできます。例えば、ジョブ実行の時点でジョブ実行ユーザーに割り当てられた予算を確認し、不足している場合にジョブの実行を拒否するような制御が可能になります。

■ ジョブ統計情報カスタマイズ機能

PRIMEHPC FX1000 のジョブスケジューラーには、ジョブが使用したノード数・CPU 時間・メモリ量などの資源量や、ジョブの実行時間、ジョブプロセスに設定された各種制限値などの情報（ジョブ統計情報）を記録する、ジョブ統計情報機能があります。

ジョブ統計情報カスタマイズ機能は、ジョブ統計情報として記録する項目を選択したり、独自の項目を定義して記録できるようにする機能です。管理者によって定義された独自の統計情報項目には、前述のフック機能を利用して任意の値を設定することができます。また、独自の統計情報項目は、その他のジョブ統計情報の項目と同様に、エンドユーザー向け当該情報として表示するか、またはジョブ統計情報として記録するかを選択可能です。

■ スケジューラープラグイン機能

スケジューラープラグイン機能は、システム管理者が作成する独自のスケジューリングアルゴリズムをジョブスケジューラーに組み込み、PRIMEHPC FX1000 のジョブスケジューラーのスケジューリングアルゴリズムと置き換える機能です。スケジューラープラグイン機能を利用することで、ジョブの優先度の評価数式や制御方式そのものをセンターの運用に最適化されたものに変更でき、よりきめ細かな制御を実現できます。

■ コマンド API

コマンド API は、PRIMEHPC FX100 のジョブスケジューラーが提供するエンドユーザー向けコマンドと同等の機能（ジョブの操作や情報取得）を提供するインターフェース群です。管理者およびエンドユーザーは、コマンド API を利用することで既存のコマンドの機能を拡張した独自のコマンドを作成したり、運用／システム利用のためのユーティリティを作成したりすることが可能になります。

高稼働率機能：Adaptive elapsed-time job scheduling

ジョブ投入時にジョブの実行時間の最小値を指定できるようにする機能です。実行時間の最小値が指定されたジョブは、指定された最小時間までの実行が保障されます。指定された最小時間が実行開始から経過した後も、他のジョブの実行を妨げない状況であればジョブの実行時間が延長され、継続して実行されます。

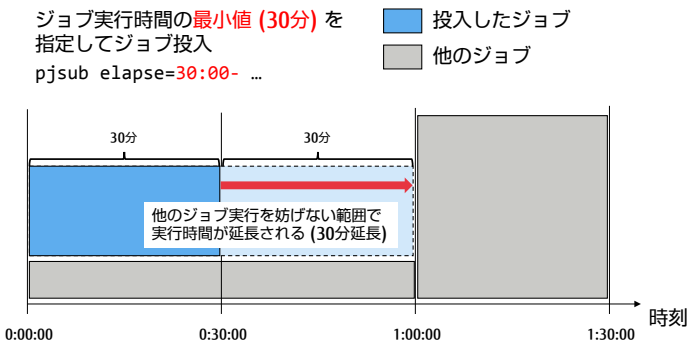

図 16　実行時間の最小値を指定したジョブの実行例

■ 稼働率改善の例 1

高稼働率機能を活用することで、一連のジョブの実行をより早く完了させることが可能になります。ここでは、総実行時間として 3 時間を必要とする計算タスクの実行例を示します。

(a) 実行時間 3 時間のジョブを 1 個投入
(b) 実行時間 1 時間のジョブを 3 個投入
(c) 実行時間の最小値として 1 時間を指定したジョブを、計算が終了するまで投入し続ける

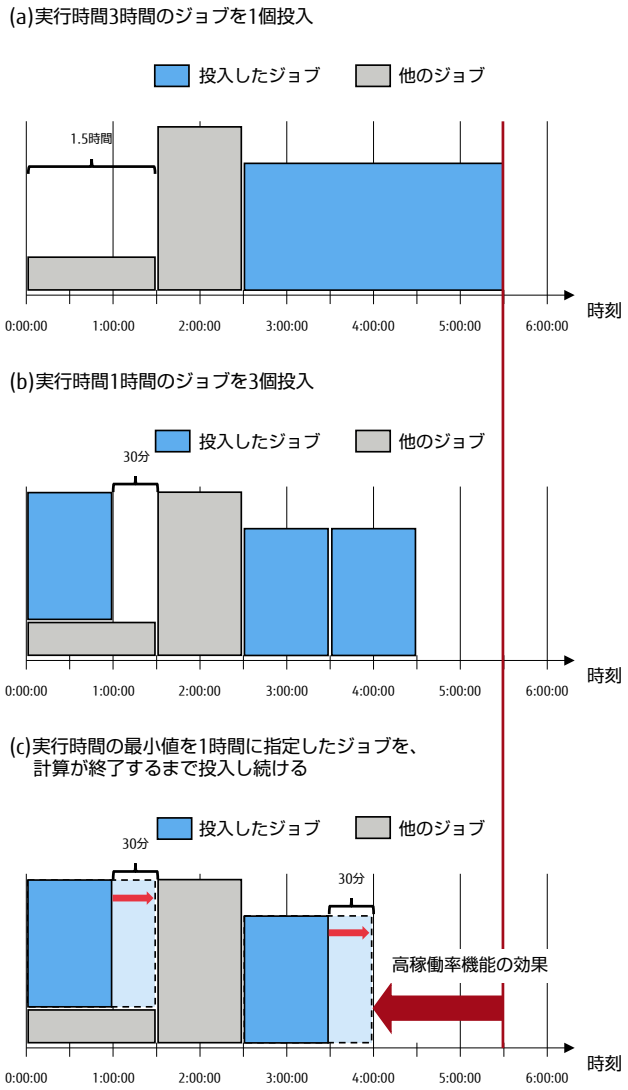

図 17　稼働率改善の例 1

図 17 の(a)では 3 時間連続して使用可能な計算ノードが必要です。そのような連続使用可能なノードを確保できずにジョブの実行開始が遅れた場合、一連のジョブの実行が終了するまで 5 時間 30 分を要します。

図 17 の(b)では 1 時間のジョブは早期に実行できますが、残り 2 個のジョブの実行開始が遅れてしまうため、一連のジョブの実行が終了するまでに 4 時間 30 分を要します。

図 17 の(c)では後続のジョブの実行が開始されるまで隙間なく継続してジョブを実行できます。このため、一連の計算タスク終了までの時間は 4 時間となり、3 方式の中で最も短時間になります。

■ 稼働率改善の例 2

ジョブ実行時間の最小値を指定したジョブが多く投入されることで、システム全体の計算ノード使用率が高まることが期待できます。

スーパーコンピュータ「京」の充填率推移を解析した結果、実行予定時間に長時間を指定されたジョブが大規模ジョブのスケジューリングに影響を与えていました。

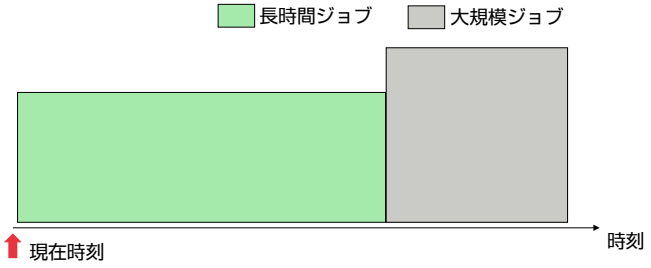

図 18　大規模ジョブの待機状態

この場合、大規模ジョブの実行開始予定時刻までの間、計算ノードが利用可能な状態のままになるため、バックフィル機能によって低優先度のジョブの実行が先に開始されます。ただし、低優先度ジョブの実行開始後、先に実行されていた長時間ジョブが予定よりも早く終了した場合、大規模ジョブは低優先度ジョブが終了するまで実行を開始できず、多くの計算ノードが利用されないまま時間の経過を待ち続けることになります。

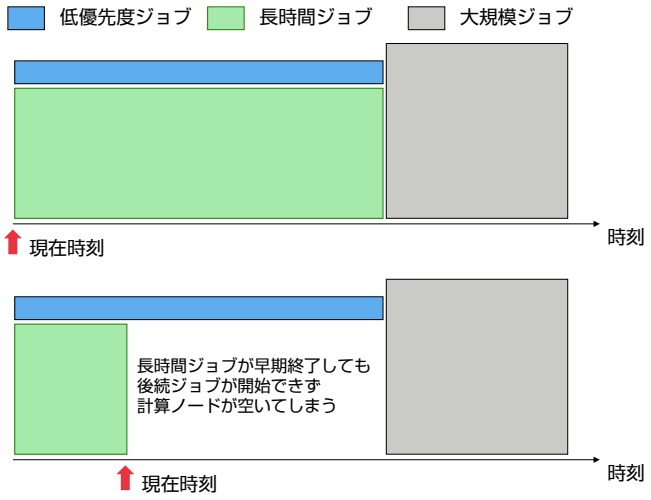

図 19　長時間ジョブの早期終了による影響

バックフィル機能で先に実行開始されるジョブに実行時間の最小値が指定されていた場合、当該ジョブの実行は、先に実行されていたジョブが終了した時点で中断されるため大規模ジョブの実行時間が早まり、多くの計算ノードが使用されない状況を避けることができます。

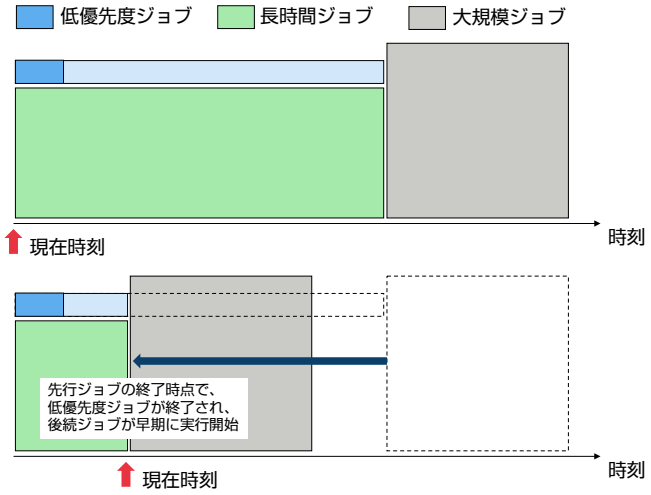

図 20　稼働率改善の例 2

システム運用管理

システム運用管理への取り組み

近年のスーパーコンピュータシステムは、システム性能の向上を目的とし、数千～10 数万の計算ノードで構成されることが一般化してきています。このような大規模なシステムを効率よく運用するためには、状態管理および運転制御を容易にするための「システム運用管理」がより重要になります。

富士通では、「京」の運用経験を生かし、PRIMEHPC FX1000 を含めた大規模システムを柔軟かつ効率的に管理できるシステム運用管理を開発しました。

- 柔軟かつ効率的なシステム運用の実現
  - ・クラスタを集中管理
  - ・導入から保守、通常業務監視まで一貫管理
- 運用を止めない高可用性システム
  - ・異常検知時のジョブの自動再投入
  - ・重要ノードの冗長化と自動切り替え
  - ・計算機センター運用を停止させない、部分的な保守の実現

これらを実現するためのシステム運用管理の機能および実現手段について、以下に紹介します。

柔軟かつ効率的なシステム運用の実現

■ クラスタを集中管理

PRIMEHPC FX1000、PC サーバ、およびディスク装置で構成されるストレージ群を「クラスタ」として扱い、クラスタ単位で運用管理します。

PRIMEHPC FX1000 と PC サーバを 1 つのクラスタ構成にすることができます。エンドユーザーがログインノードからそれぞれの計算ノードにジョブを投入できる柔軟なジョブ実行環境を実現しています。

■ 導入から保守、通常業務監視まで一貫管理

管理者は、システム管理ノードで、導入から保守までの業務を集中管理できます。

- 分散処理型インストーラーによるソフトウェアの導入

  大量のノードに、OS やさまざまなパッケージをインストールする作業は、非常に時間がかかるだけでなく、その後の管理も難しいものになります。このため PRIMEHPC FX1000 では、大規模システムに対応したインストーラーを提供しています。このインストーラーでは、パッケージやジョブ実行環境として必要な初期設定を一括して実施できます。

- システム構成情報・状態表示の統一

  システムを構成する各ノードの情報には、そのノードの実装 CPU 数や実装メモリ量などのハードウェア情報、そのノードに設定されている IP アドレスやノードの役割などのソフトウェア設定情報といった、さまざまなものがあります。また、それらのノードに電源が投入されているか、何らかのハードウェア故障またはソフトウェア異常が発生していないかといった、ノードの状態に関する情報も存在します。

システムの大規模化に伴い、利用者や管理者が把握したい情報は広範囲に及びます。システム構成情報は、利用者や利用シーンに応じて、さまざまなコマンドで扱われます。コマンドの表示形式や指定形式が統一されていないと、利用者に混乱を与えてしまいます。PRIMEHPC FX1000 向けの各種ソフトウェアでは、システム構成情報に関する表現形式が統一されています。

運用を止めない高可用性システム

■ 異常検知時のジョブの自動再投入

PRIMEHPC FX1000 ではノードの異常を 2 種類の方法で検知できます。

1 つ目がソフトウェアによるシステム監視です。ノードグループの階層構造を使用して、負荷を分散しながら効率よくノードやサービスの状態を収集し、ノード異常を検知します。2 つ目がハードウェアによる異常通報との連携です。PRIMEHPC FX1000 には、ノードやインターコネクトに関するハード異常を通報する仕組みがあり、その通報によってノードの異常を即座に検知できます。

これらの方法でノード異常を検知した場合、該当ノードを運用対象から除外します。同時に、該当ノードを含む、実行されているジョブプロセスをすべて終了し、利用可能なノードでジョブを自動的に再実行して、システムの運用を継続します。

■ 重要ノードの冗長化と自動切り替え

PRIMEHPC FX1000 の運用管理に欠かせない重要なノードとして、システム管理ノード、計算クラスタ管理ノードなどがあります。これらの重要なノードが故障によって起動できない状態になると、システムのすべて、または一部が停止し、運用を継続できなくなってしまいます。

PRIMEHPC FX1000 では、これらの重要なノードをすべて運用系、待機系の冗長構成にすることが可能です。運用系のノードで異常が検知された場合には、自動的に待機系のノードに切り替えて運用を継続できます。

■ 計算機センター運用を停止させない、部分的な保守の実現

ソフトウェアの保守としてパッチを適用する際に、設定更新のためにすべてのノードを再起動する必要があった場合、大規模なシステムであるほど運用停止の時間が長くなります。ただし、ソフトウェアの修正内容によっては、必ずしもノードの再起動が必要ないパッチや、旧バージョンとの混在が可能なパッチが存在します。旧バージョンとの混在が可能なパッチに対してクラスタ全体のジョブ運用を停止せずに、一部の計算ノードでクラスタ内のジョブ運用を継続しながら部分的な保守をすることをローリングアップデートと呼びます。

システムソフトウェアの各パッケージで、ローリングアップデートの可否や、修正適用後に必要なシステム操作（ノードの再起動、サービスの再起動、不要など）をパッケージ管理 rpm の付帯情報としてパッチごとに提供することで、管理者によるソフトウェア保守に必要な作業を見える化し、ソフトウェア保守による計算機センターの運用停止時間の最小化を実現しています。

分散ファイルシステム

スーパーコンピュータの演算能力を十分に発揮するためには、10 数万ノードからのファイル I/O 時間をいかに短く抑えるかが重要です。このためファイルシステムには高い並列 I/O 性能が求められます。また、ファイルシステムの安定性はスーパーコンピュータ自体の安定性に直接関係するため、ファイルシステムには高い信頼性も求められます。

ここでは性能と信頼性を両立させた分散ファイルシステム「FEFS」について紹介します。

Lustre ベースのクラスタ型ファイルシステム

FEFS は、HPC 向けファイルシステムのデファクトスタンダードとして実績のあるオープンソース Lustre 2.10 をベースとした、クラスタ型の分散ファイルシステムです。Lustre が持つ高い並列性やスケーラビリティなどの優れた性能・機能を継承しつつ、「京」のような超大規模システムから中小規模のセンター運用まで幅広いシステムに適用できるように富士通が拡張を行い、高性能と高い安定性、使いやすさを実現しています。

大規模システムでの豊富な実績

FEFS は、「京」の開発・運用を通して培った技術を継承しています。「京」での実績に加え、富士通の PC クラスタシステムや PRIMEHPC FX10/FX100 での実績により、システムの安定稼働に貢献しています。

■ テラバイト毎秒を超える高いスケーラビリティ

クラスタ型ファイルシステムの FEFS は、使用する OSS 数に比例して総スループットをスケールアウトできます。要求性能に応じた台数を用意することで、テラバイト毎秒を超えるスループットを実現できます。

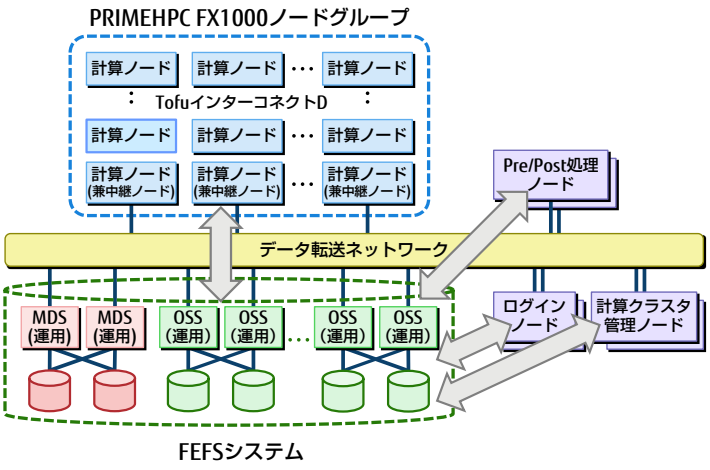

図 21　PRIMEHPC FX1000+FEFS システム構成

PRIMEHPC FX1000 は、1 ラックあたり最大 8 ノードの計算ノード（兼中継ノード）を持ち、ファイルサーバにアクセスするための InfiniBand アダプタを搭載しています。InfiniBand を搭載するノード数は性能要件に応じて柔軟に選択できるため、搭載数に比例してスループット性能をスケールアウトできます。

■ 並列アプリケーションへの外乱排除

10 数万ノード規模となる高並列な MPI アプリケーションの性能を引き出すには、システムデーモンによる外乱を排除することが重要です。システムデーモンが動作すると、並列プロセス間の同期処理が遅延し、アプリケーションの実行時間が延びるためです。FEFS は、定期的に動作するファイルシステムデーモンの処理を削減し、実行時間への影響を抑えています。

PRIMEHPC FX1000 は、48 個の計算用コアに加えて、OS 用のアシスタントコアを 2 または 4 個備えています。I/O 処理を行うシステムデーモンをアシスタントコアで動作させ、計算コアをアプリケーションに占有させることで、ジョブ実行に対する外乱を排除しています。

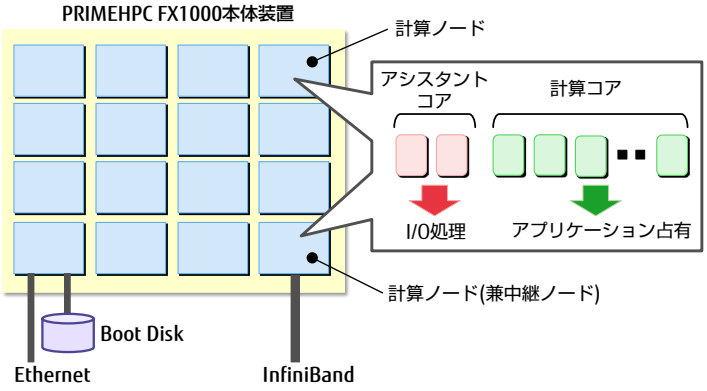

図 22　アシスタントコアを活用した I/O 処理と計算処理の分離

■ マルチ MDS によるメタデータアクセスの負荷分散

ファイルシステムを MDS と MDT の複数の組で構成することで、システム性能のボトルネックになりがちなメタデータアクセスの負荷を分散できます。マルチ MDS ペアを増設することで、ファイルシステムの管理ファイル数の増加が可能です。

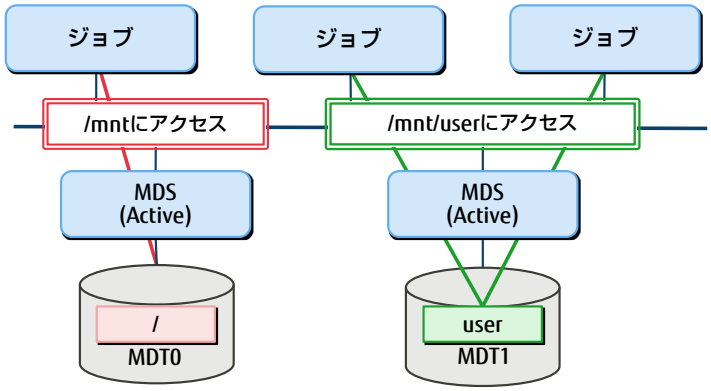

図 23 サブディレクトリ単位のメタデータ管理

ハード故障時にも運用を継続させる信頼性

大規模システムの安定稼働には、性能と並んで高い信頼性が重要です。多数のファイルサーバやストレージ装置、ネットワーク機器で構成されるクラスタ型ファイルシステムでは、一部が故障したり停止した場合にも、システムの運用を継続できることが求められます。

FEFS は、ハードウェアの二重化により耐故障性を高め、システム管理ソフトウェアと連携したノード監視・自動交替により、単点故障や保守中もファイルシステムのサービスを持続することができます。

■ 耐故障性

数百台を超えるファイルサーバ、ストレージ装置で構成される大規模なファイルシステムでは、故障を自動的に検出し、故障箇所を迂回してファイルシステムのサービスを持続させることが重要です。

FEFS は、ハードウェアを二重化し、ソフトウェア制御によってサーバおよび I/O 通信の経路を切り替えることで、単点故障時にもファイルシステムとしてのサービス持続を可能にしています。

PRIMEHPC FX1000 では、計算ノード（兼中継ノード）の InfiniBand が故障などで通信できなくなった場合に、別の計算ノード（兼中継ノード）の InfiniBand を使うように自動的に通信経路を切り替え、ファイルサーバへのアクセスを持続させることで、耐故障性を高めています。

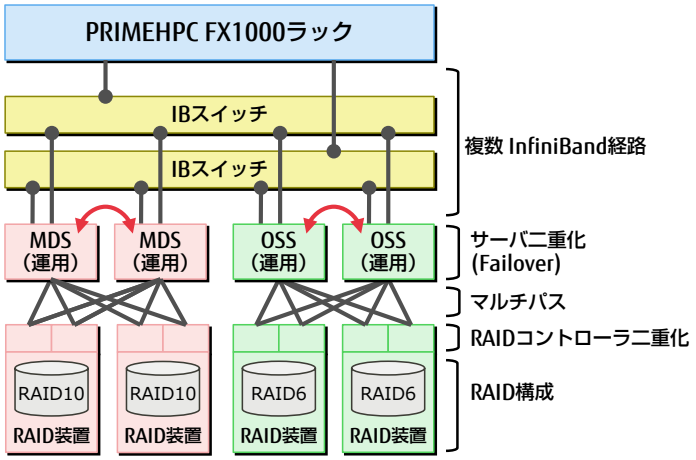

図 24　耐故障性

■ 階層的なノード監視と自動交替

大規模システムでは、故障の検出と、その故障が影響する範囲のノードへの交替通知を自動化することが必須です。

計算ノードとファイルサーバ間で監視パケットを受け渡してノード状態を監視する方式では、規模のべき乗に比例して監視パケットが大量に発生してしまい、計算ノード間の MPI 通信および計算ノードとファイルサーバとの間のデータ通信が阻害されます。

FEFS では、システム管理ソフトウェアと連携して階層的なノードの監視と交替の制御を行い、通信負荷を最小化しています。これは、PRIMEHPC FX1000 ラック内の監視、複数ラックをグループ化したノードグループ単位での監視、その上位のノードグループの監視とツリー上に監視を行うことで実現しています。

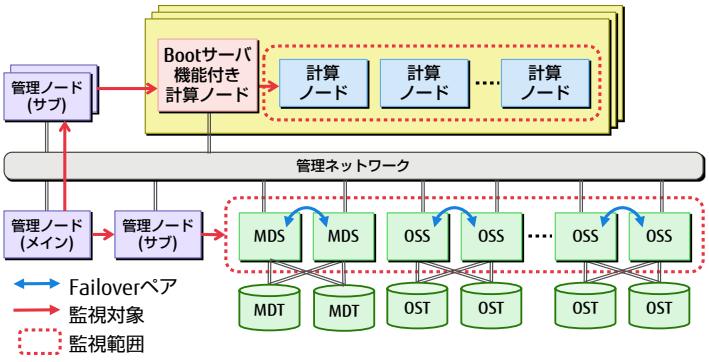

図 25　ノード監視と自動交替

運用性を高める機能

多数の利用者が利用する大規模システムでは、特定の利用者が大量にファイル I/O を行った場合でも、ほかの利用者に影響を及ぼさないことが求められます。また、計算ノードのジョブからファイルアクセスが行われている場合でも、ログインノード上の利用者のレスポンスに影響しないことが求められます。

FEFS は、利用者ごとのフェアシェア QoS 機能と、ログインノードのレスポンス保証 QoS 機能により、これらの課題を解決しています。

■ 利用者ごとのフェアシェア QoS 機能

複数の利用者が同時に使用するログインノードでは、特定の利用者が大量のファイル I/O を行うと、ほかの利用者のファイル I/O が極端に遅くなってしまうことがあります。

FEFS は、クライアント側で 1 人の利用者が同時に発行できる I/O リクエスト数に上限を設けることで、1 人の利用者によって大量の I/O リクエストが発行され I/O 帯域が占有されることを抑止できます。

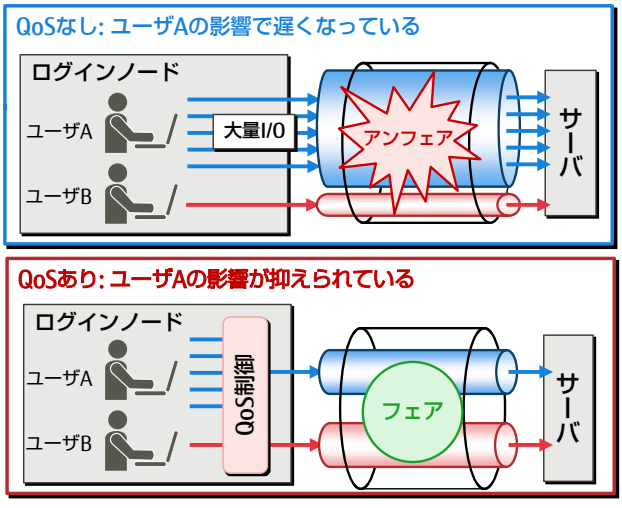

図 26　QoS による複数利用者間のアクセス均等化

■ ログインノードのレスポンス保証 QoS 機能

利用者に対するアクセスレスポンスは、使いやすさに直結するため、ジョブに対するアクセスレスポンスよりも重要です。

FEFS は、ログインノード上の利用者に対するアクセスレスポンスを保証するため、ノード群別に I/O 要求を処理するサーバスレッドを割り当てる機能を備えています。これにより、計算ノードのジョブがファイル I/O を行っていても、ログインノードでファイル操作を行う利用者のレスポンスを維持できます。

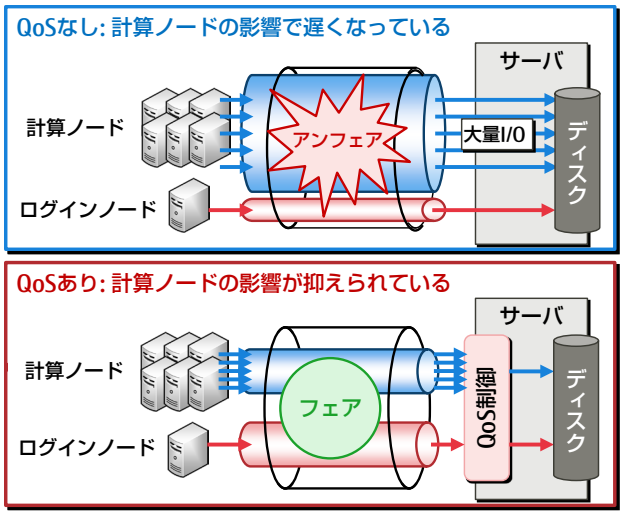

図 27　ログインノードのレスポンス保証

Lustre コミュニティへの貢献

Lustre の開発・サポートは、広範なオープンソースコミュニティによって支えられています。富士通は、コミュニティの一員として Lustre の発展に貢献しています。

自社製スーパーコンピュータの開発を通じて、現出した課題を解決しコミュニティにフィードバックすることで、Lustre の性能改善や品質安定を実現するとともに、HPC 向け分散ファイルシステムの発展を目指します。

電力管理

省電力への取り組み

計算ノード数が数万～数十万に及ぶスーパーコンピュータシステムの大規模化に伴い、その消費電力も大きなものになっています。社会全体で省エネルギーへの関心が高まる中、スーパーコンピュータシステムの消費電力が増加していることから、以下のような課題が現出しています。

- システムの拡張により、施設の電源設備からの電力供給が追いつかない
- 設置している施設の節電計画に基づき、システムの消費電力を抑制しなければならない

これらの課題に対して、PRIMEHPC FX1000 が提供するパワーノブで CPU 周波数やメモリアクセスを制御して電力消費を抑えることができます。ただし、HPC においてはジョブの実行性能も求められます。HPC における電力管理は、近年、システムの運用面での電力管理や、ジョブの実行性能を考慮した省電力化など、ジョブの実行性能と両立させるための研究が盛んに行われています。

富士通ではジョブの実行性能に影響を及ぼさないように省電力化を実現するために、図 28 に示すようにシステム管理者が運用面での省電力化を、エンドユーザーが自分のジョブの実行性能を考慮した省電力化をそれぞれ行うための以下の機能を提供しています。

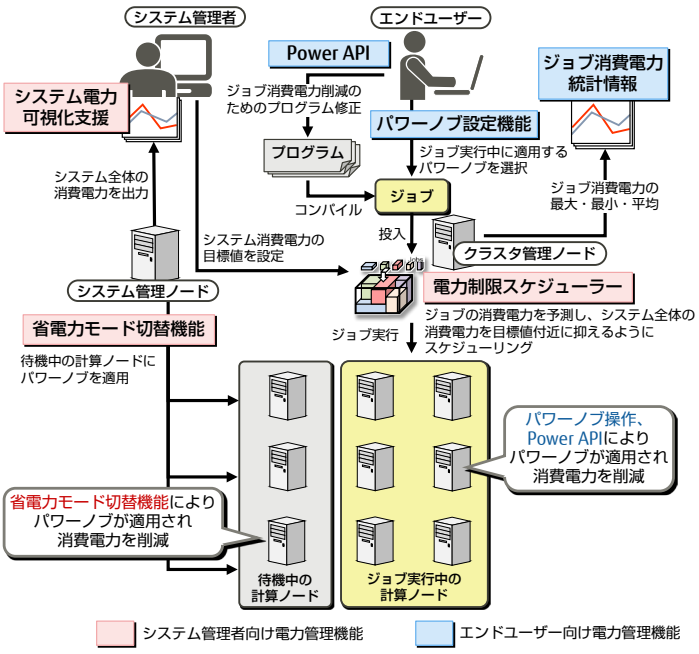

図 28　電力管理機能の構成

システム管理者向け電力管理機能

■ 省電力モード切替機能

計算ノードでは、ジョブを実行していない待機状態中にも電力が消費されます。この待機電力はジョブの実行に関係しないため、システム全体にとっては無駄な消費電力と見なされ、削減が求められます。

省電力モード切替機能は、待機状態にある計算ノードの CPU やメモリを省電力モードに自動的に移行し待機電力を最小限に抑える機能です。

計算ノードがジョブを実行していない待機状態になっている間、PRIMEHPC FX1000 が提供するパワーノブを自動的に適用して、待機中に消費される無駄な電力を削減します。図 29 はパワーノブ省電力機能によるパワーノブの適用状況と消費電力の変化を示しています。

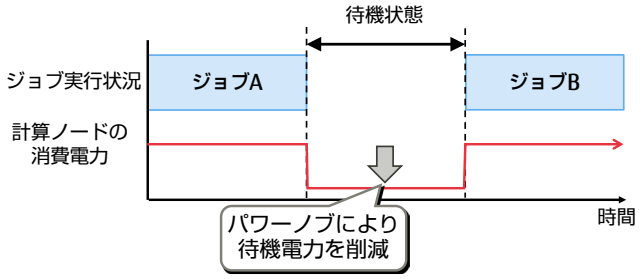

図 29　省電力モード切替機能による消費電力の推移

図 29 において、省電力モード切替機能はジョブ A の実行終了と同時にパワーノブを適用し消費電力を抑えます。そして、ジョブ B の実行開始直前に省電力モード切替機能はパワーノブの適用を解除します。これによりパワーノブの影響を受けずにジョブ B を実行ができます。

■ 電力制限スケジューラー

施設の節電計画などでは、システム全体の消費電力の削減目標が定められています。システム全体の消費電力の目標値が定められている場合、ジョブの消費電力を予測できれば、定められた目標値付近に収まるようにジョブをスケジューリングすることができます。

電力制限スケジューラーでは、これまでに実行されたジョブの電力消費状況を元に、新規に投入されるジョブの消費電力を予測し、予測したジョブ消費電力に基づいてジョブスケジューリングを行います。

電力制限スケジューラーによる電力管理状況を評価するために、実際のスーパーコンピュータ環境でのジョブ運用データを九州大学様から提供いただき、システム全体の消費電力の推移をシミュレーションしました。電力制限スケジューラーを適用した場合と適用しなかった場合に関して、図 30 は 100 万秒間（約 11 日間）でのシステム全体の消費電力の推移を示し、表 8 はシミュレーション期間中でのシステム全体の消費電力の平均、最大、最小を示しています。

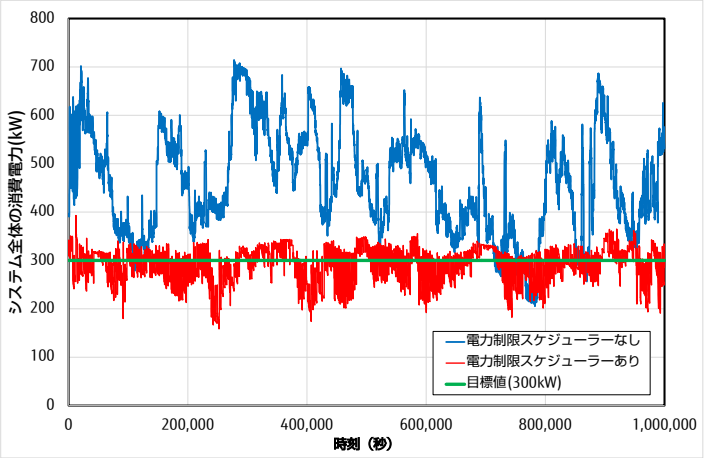

図 30　電力制限スケジューラー適用時のシミュレーション

表 8　システム全体の消費電力の比較

	電力制限スケジューラーあり	電力制限スケジューラーなし
システム全体の平均消費電力	283kW	462kW
システム全体の最大消費電力	393kW	714kW
システム全体の最小消費電力	159kW	206kW

図 30 では電力制限スケジューラーを適用しなかった場合、システム全体の消費電力は最大 714kW まで大きく変動しています。一方、電力制限スケジューラーを適用した場合、システム全体の消費電力は目標値である 300kW 付近を大きく超えずに推移しています。

また、平均値でも、電力制限スケジューラーを適用しなかった場合の 462kW に対して、適用した場合は 283kW となり、電力制限スケジューラーによって目標値を超えないようなスケジューリングが行われていることがわかります。

■ システム消費電力可視化支援機能

省電力モード切替機能、電力制限スケジューラーによる電力管理状況を監視するための機能です。システム全体の消費電力を参照することができるコマンドや API を提供しています。

エンドユーザー向け電力管理機能

PRIMEHPC FX1000 が提供するパワーノブは、CPU 周波数の制御、メモリアクセス制限、命令発行数制限など多岐にわたります。エンドユーザーは自分のジョブの処理内容に応じて、これらのパワーノブの中から適切なものを選択することができます。エンドユーザーが自分のジョブに任意のパワーノブを適用するための方法には、ジョブ投入時に指定する方法と、プログラム内で指定する方法があります。

■ パワーノブ設定機能

エンドユーザーは、適用したい任意のパワーノブをジョブ投入時に指定できます。指定されたパワーノブはジョブ実行中に適用されます。

実行例を以下に示します。

```
$ pjsub -L freq=1800 -L node=10 run
```

ここで、pjsub はジョブを投入するためのコマンドです。また、freq は CPU 周波数を変更するパワーノブを、node はジョブを実行する計算ノードの数を、run は実行するジョブ名を示します。この例では、ジョブ run を実行する計算ノード（10 ノード）に対し、CPU 周波数を 1.8GHz（1,800MHz）に設定しています。

ジョブ run が計算処理と IO 処理を一定間隔で繰り返すジョブである場合、上記のコマンドを実行したときのジョブ内の処理の推移を図 31 に、ジョブの消費電力の推移を図 32 に示します。

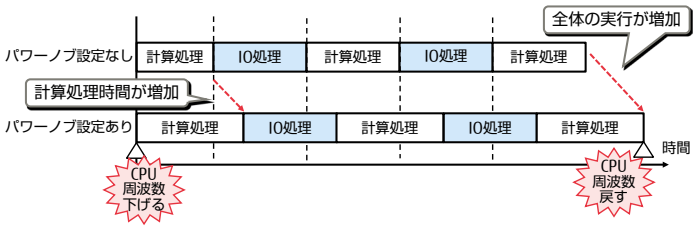

図 31　パワーノブ設定有無でのジョブ内の処理の推移

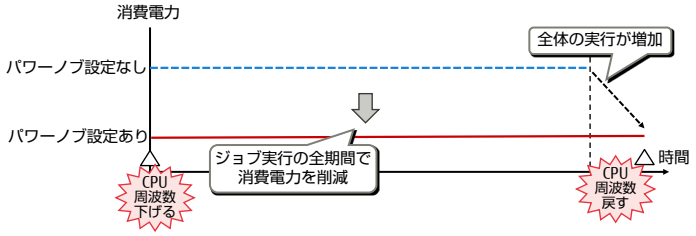

図 32　パワーノブ設定有無での消費電力の推移

パワーノブ設定機能では、ジョブの実行開始直前から実行終了直後まで、指定したパワーノブが適用され続けます。CPU の周波数が通常時 2GHz である場合、上記コマンドによりジョブ実行の全期間で CPU 周波数が 1.8GHz に低下します。図 31 において CPU をあまり使用しない IO 処理は周波数低下による影響を受けませんが、計算処理は周波数低下により処理時間が増加し、ジョブ全体の実行時間も増加しています。しかし、図 32 のように、周波数低下によりジョブの消費電力は常に抑制されます。このように、パワーノブ設定機能はジョブの処理内容によりジョブ全体の実行時間が増加する可能性がありますが、ジョブ投入コマンドのオプションでパワーノブを指定することで、エンドユーザーは容易にジョブ消費電力を抑制することができます。

■ Power API

PRIMEHPC FX1000 が提供するパワーノブのうち、任意のパワーノブを任意のタイミングで適用できるライブラリを提供します。これによりエンドユーザーは自分のジョブのプログラム内で、適用したいパワーノブを指定することができます。なお、ライブラリで提供する API 仕様は Sandia National Laboratories が提唱したライブラリインターフェース（Sandia Power API）に基づいています。

Power API が提供する API を使用することで、ジョブプログラム中の任意のタイミングで任意のパワーノブを適用することができます。

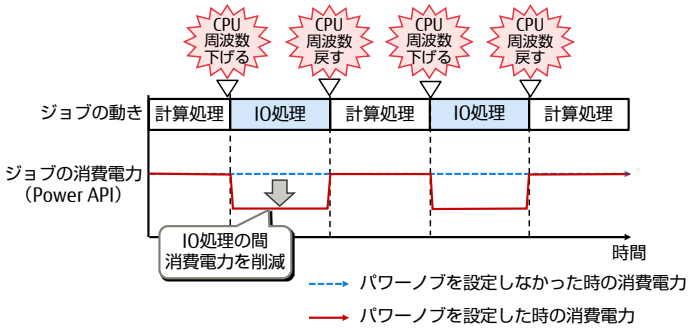

図 33　Power API によるジョブ消費電力の推移

図 33 は、計算処理と IO 処理を一定間隔で交互に行うジョブに対して、Power API により IO 処理の間のみ CPU 周波数を下げるようにプログラムを修正した場合の消費電力の推移を示しています。図 33 における CPU の通常時の周波数が 2GHz だった場合、CPU をあまり使用しない IO 処理中のみ CPU 周波数を低下させることで、ジョブ全体の実行時間を増加させずに IO 処理中の消費電力が抑制されます。エンドユーザーは Power API を用いることで、適切なタイミングで適切なパワーノブを指定するようにプログラムを修正する必要がありますが、ジョブ全体の実行時間に影響を与えずに消費電力を抑制することができます。

■ ジョブ消費電力統計情報

ジョブ消費電力の平均、最大、最小などの統計情報をエンドユーザーが参照するための機能です。これにより、エンドユーザーはパワーノブ設定機能や Power API を適用したジョブの電力消費状況を確認することができます。

参考情報
PRIMEHPC FX1000 に関する情報は、当社営業までお問い合わせいただくか、以下の Web サイトをご参照ください。

https://www.fujitsu.com/jp/products/computing/servers/supercomputer/index.html

FUJITSU Supercomputer PRIMEHPC FX1000 先進のソフトウェア
富士通株式会社
2019 年 11 月 12 日　第 1 版
2019-11-12-JP