スーパーコンピュータ「富岳」におけるファイルシステム・電力管理の機能強化

秋元　秀行　　岡本　拓也　　加賀美　崇紘　　関　堅　　酒井　憲一郎　　今出　広明
篠原　誠　　住元　真司

あらまし

理化学研究所と富士通はスーパーコンピュータ「京」（以下,「京」）の後継機として，2021年度の一般共用開始に向けてスーパーコンピュータ「富岳」(以下,「富岳」) の開発に取り組んでいる。「富岳」では「京」のソフトウェア資産を継続しつつ，演算性能･資源の利用効率･使い勝手など様々な点で改良･改善し，機能強化に取り組んでいる。このうち，ファイルシステムについては,「京」からの更なる高速化・大容量化に加え，ユーザビリティを中心に大幅に機能を拡張した。また，全ての超大規模システムにおける共通の課題として，システムの消費電力の削減，および効率的な電力利用がある。このため,「富岳」の運用ソフトの拡充の一環として，電力管理機能を新たに設計・開発を進めた。

本稿では,「富岳」において「京」から大きく機能強化したファイルシステム，および電力管理機能について述べる。

1. まえがき

理化学研究所と富士通は，スーパーコンピュータ「京」（以下，「京」）の後継機として，2021年度の一般共用開始に向けてスーパーコンピュータ「富岳」（以下，「富岳」）の開発に取り組んでいる。

スーパーコンピュータのアプリケーションの動作には，図-1に示すシステムソフトウェアが関与している。Operating System（OS）およびアプリケーション開発環境については，それぞれ別稿［1，2］に譲り，本稿ではファイルシステムと運用ソフトに注目する。ファイルシステムは，アプリケーションプログラムそのものやデータの高速・安全な保存環境を提供する。運用ソフトは，主たる機能として，システム管理機能，およびジョブ管理機能を提供する。

「富岳」では，「京」のソフトウェア資産を継続しつつ，より高い演算性能の提供と資源の利用効率化を目指した。また，スーパーコンピュータ利用の裾野を広げるために，図-1の各コンポーネントにおいて柔軟な運用や使い勝手の改良・改善に取り組んでいる。

ファイルシステムにおいては，「京」で開発したFEFS（Fujitsu Exabyte Filesystem）［3］をベースとして，階層化ストレージのユーザビリティの向上とアプリケーションのファイルIO最適化を可能にするため，ジョブ実行領域専用のファイルシステムであるLLIO（Lightweight Layered IO-Accelerator）を新たに設計・開発した。運用ソフトは，システム監視やジョブスケジューリング性能の改善に向けた取り組みに加え，ジョブスケジューラなどのAPI（Application Programming Interface）を充実し，管理者による運用カスタマイズ機能の充実を図った［4］。加えて運用ソフトとしては，全ての超大規模システムにおいて，課題となっているシステムの消費電力の削減，および効率的な電力利用を促進するための電力管理機能を新規に設計・開発し機能強化を図った。

本稿では，ユーザビリティを中心に大幅に機能を拡張したファイルシステム，および運用ソフトの機能強化として新規に設計・開発した電力管理機能について述べる。

2. ファイルシステム

「富岳」では，大容量かつ高速なストレージシステムを実現するために，「京」と同様［3］，階層化ストレージを採用している。「富岳」は高い演算性能を提供することに加え，幅広いユーザーの利用を想定したシステムであり，ストレージシステムもこの方針に従う必要がある。そのためには，階層化ストレージのユーザビリティの向上に加え，アプリケーションごとに異なるファイルアクセス特性に応じて，最適なチューニングを可能にする機能が必要である。

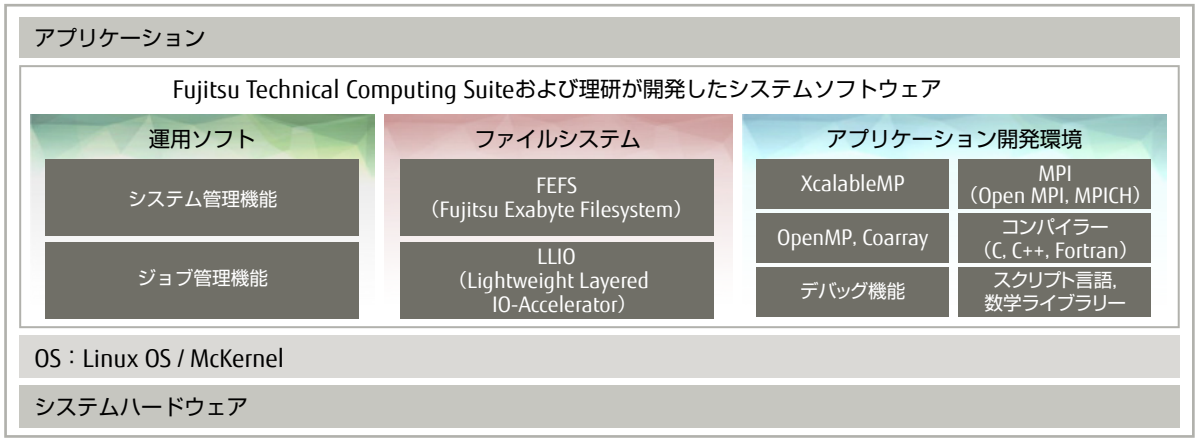

図-1 「富岳」のソフトウェアスタック（構成）

我々は，これらを実現するために，ジョブ実行専用領域のファイルシステムとして，LLIOを新たに開発した。LLIOは，用途に応じた三種類の領域をアプリケーションに提供することで，階層化ストレージのユーザビリティを向上し，かつアプリケーションのファイルIO最適化を可能にする。

本章では最初に，「富岳」のストレージシステムの概要について述べる。その後，LLIOの三種類の領域による階層化ストレージのユーザビリティ向上，一時ファイルのIO最適化について説明する。最後に，LLIOの性能測定結果について述べる。

2.1 ストレージシステムの概要

図-2に，「富岳」のストレージシステムの概要を示す。「富岳」のストレージ階層は，ジョブ実行専用の高速領域としての第一階層，ユーザーとジョブが利用する大容量の共有領域としての第二階層，商用クラウドストレージとしての第三階層で構成される [5]。本稿執筆時点（2020年7月上旬）では，第三階層のクラウドストレージの利用方法は準備中であるため，第一階層および第二階層に焦点を絞り，「富岳」のストレージシステムについて述べる。

第一階層ストレージは専用のサーバノードを持たず，NVMe SSD（Non-Volatile Memory Express Solid State Drive）を搭載した計算ノード兼ストレージIO（SIO）ノードがファイルシステムサーバとしての役割を担い，計算ノードのファイルアクセス要求はSIOノードのアシスタントコア [6] で処理される。SIOノードは，16計算ノードがグループ化されたBoB（Bunch of Blades）ごとに1ノード存在する。各ジョブの開始時には，ジョブに割り当てられたBoBのSIOノードのみを用いて一時的なLLIOのファイルシステムを生成し，各ジョブはこれを利用する。そして，ジョブ実行の完了後，このファイルシステムは解放される。

表-1に示すように，LLIOは用途に応じた三種類の領域をアプリケーションに提供する。第二階層キャッシュ領域は，階層間の名前空間の違いやデータ転送を意識することなく，第一階層ストレージの利用を可能にする。共有テンポラリ領域は，計算ノード間で共有の一時ファイルを保存するのに適した領域である。ノード内テンポラリ領域は，計算

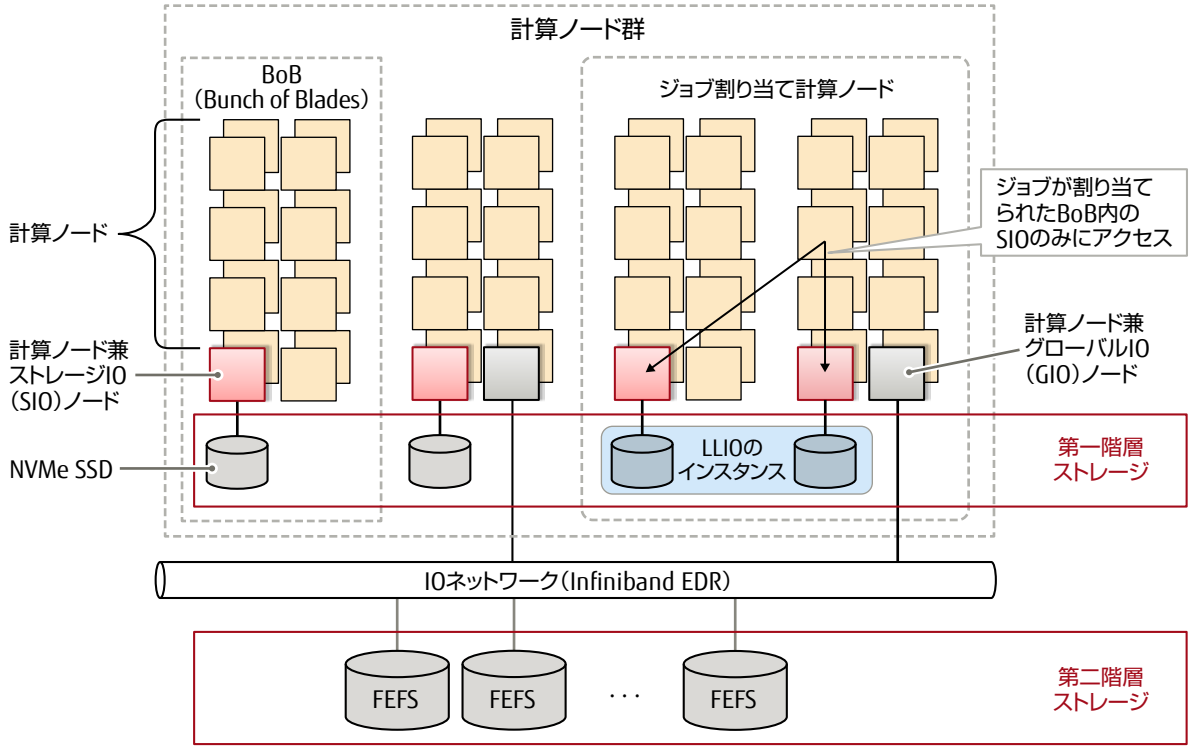

図-2 「富岳」ストレージシステムの概要

表-1　LLIOの三種類の領域

領域	名前空間	用途
第二階層キャッシュ領域	第二階層ファイルシステム透過	保存ファイル
共有テンポラリ領域	ジョブ内共有	一時ファイル
ノード内テンポラリ領域	ノード内共有	一時ファイル

ノード内のみで共有される一時ファイルの保存に特化した領域である。これらの領域についての詳細は2.2節，2.3節で述べる。

階層間は，IOネットワークを介して接続される。計算ノード群のうち，IOネットワークに接続されるのは計算ノード兼グローバルIO（GIO）ノードであり，階層間のデータ転送はGIOノードを経由して行われる。第二階層ストレージは「京」と同様，複数のFEFS [3] で構成される。

2.2 階層化ストレージのユーザビリティ向上

階層化ストレージにおいては，階層間データ転送のユーザビリティが一つの課題となる。「京」のストレージは二階層からなり，両階層は異なる名前空間を持つFEFSで構成されていた。そのため，階層間のデータ転送には，ユーザーがジョブ実行に必要なファイル，保存ファイルを明示的に指定して行うステージング方式が採用された [3]。しかし，多くのユーザーにとって，ジョブ実行に必要なファイルや保存ファイルを適切に選択することは簡単ではなく，不要なファイルが多く指定されることによって転送時間が増大する問題があった。

この問題に対して，LLIOの第二階層キャッシュ領域において，階層間データ転送をLLIOが自動で行うキャッシュ方式を採用することで，階層化ストレージのユーザビリティ向上を図った。第二階層キャッシュ領域は，第二階層のファイルシステムと同一の名前空間をアプリケーションに提供する。アプリケーションがファイルのREAD要求を発行した場合，LLIOがファイルデータを自動的に第二階層ストレージから第一階層ストレージに読み込み，キャッシングを行う。一方，アプリケーションがWRITE要求を発行した場合は，第一階層ストレージにバッファリングを行い，LLIOがアプリケーション実行とは非同期に第二階層ストレージへ書き出しを行う。このような仕組みにより，アプリケーションは階層間の名前空間の違いやデータ転送を意識することなく，高速な第一階層ストレージを利用することが可能になる。

2.3 一時ファイルのIO最適化

スーパーコンピュータのアプリケーションには，計算の途中結果を一時ファイルに書き出し，次の計算の入力ファイルとするものが多い。一時ファイルは，計算ノード間で共有されるものと，計算ノード内のみで利用されるものが存在する。これらのファイルはジョブ実行中のみ必要であるため，一時ファイルの保存に特化した領域を用いることで，ファイルIOの最適化を行うことができる。LLIOは，一時ファイルの保存領域として，共有テンポラリ領域とノード内テンポラリ領域を提供する。

共有テンポラリ領域は，計算ノード間で共有される一時ファイルを保存するのに適した領域である。共有テンポラリ領域の名前空間は，ジョブに割り当てられた計算ノード間で共有される。第二階層ストレージへのデータの書き出しを行わないため，第二階層キャッシュで発生しうる，アプリケーションのファイルIOと第二階層ストレージへの書き出しの競合による性能低下が発生せず，安定したファイルIOが可能である。

ノード内テンポラリ領域は，計算ノード内のみで利用する一時ファイルを保存するのに適した領域である。ノード内テンポラリ領域の名前空間は，各計算ノードで独立している。ノード内テンポラリ領域は，名前空間を各計算ノード間で独立させることで，専用のメタデータサーバを不要としている。そのため，大量のファイルアクセスによるメタデータサーバへの負荷集中が発生せず，計算ノード台数に比例した高い性能スループットを実現することが可能となる。

2.4 LLIOのファイルIO性能

本節では，「富岳」において1,152台の計算ノードで測定した，LLIOのファイルIO性能について述べる。

図-3に，LLIOが提供する三領域に対して，IOサイズを64 KiB，1 MiB，16 MiBと変化させて，ファ

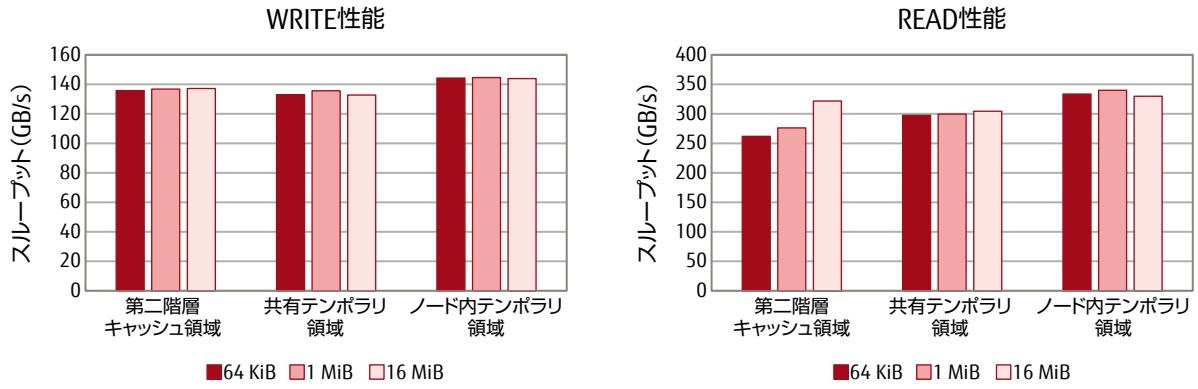

図-3　LLIOのIO性能評価

イルIO性能のベンチマークであるIOR [7] を実行した際のWRITE性能およびREAD性能を示す。WRITE性能については，各領域でIOサイズに依らず高いスループットを示している。READ性能については，第二階層キャッシュ領域は，IOサイズが大きくなるにつれて，スループットが向上している。この理由として，小サイズのIOでは，アシスタントコアにおけるソフトウェア処理のオーバーヘッドが大きく，ボトルネックとなっていることが考えられる。

今後はより大規模な環境において，多角的な指標を用いた評価を行うとともに，実アプリケーションの性能向上についても測定を行い，公開していく予定である。

3.　電力管理機能

2020年6月にISC 2020で発表されたTOP500リストにおいて「富岳」が1位を獲得し，そのプログラム実行性能の高さが証明された [8]。このベンチマークテストを実行した際の消費電力は28,335 kWであった [9]。これは，一般家庭の消費電力を400 Wとした場合，約7万世帯に相当する。「富岳」の運用においては，不必要な消費電力を削減（省電力化）すること，および限られた消費電力において最大限に演算能力を提供することが必要となる。その際に重要なポイントとなるのが運用ソフトである。

本章では，運用ソフトの機能強化の一つとして，新規に設計・開発した電力管理機能の全体像について述べる。その後，電力制御や最適化の指針につながる「富岳」の消費電力の測定機能について，結果を交えて示す。

3.1 「富岳」における電力管理機能

「富岳」においては，ハードウェア・ソフトウェアが一体となり，アプリケーション実行に電力を有効活用するための設計・開発を進めた。

ハードウェアの面では，計算ノード内のCPUやメモリーデバイスなどの動的な電力制御や電力計測機構を設計・実装した。電力制御は，動作するアプリケーション（ジョブ）の特性に応じて，デバイス状態を計算ノード内のソフトウェアから動的に変更することで，消費電力の削減や単位電力あたりの演算性能の最適化を実現する。電力計測は，制御の結果として計算ノードやデバイスの消費電力がどのように変化したかを確認・評価するために利用する。

ソフトウェアの面では，これらの機構を計算ノード内から利用するための電力制御用APIとしてPower API [10] を採用し，設計・実装を行った。

「富岳」の運用ソフトでは，Power APIを利用し，ジョブ運用に連動して電力制御・計測を行う電力管理機能を新たに設計・実装した。図-4に，電力管理の各機能の動作ノードとその構成を示す。

図-4の①は，ジョブが消費する電力の計測・集約に関する「ジョブ単位電力収集機能」である。ジョブ単位電力は，利用した計算ノード数や時間などとともに，ジョブ統計情報として記録される。各ユー

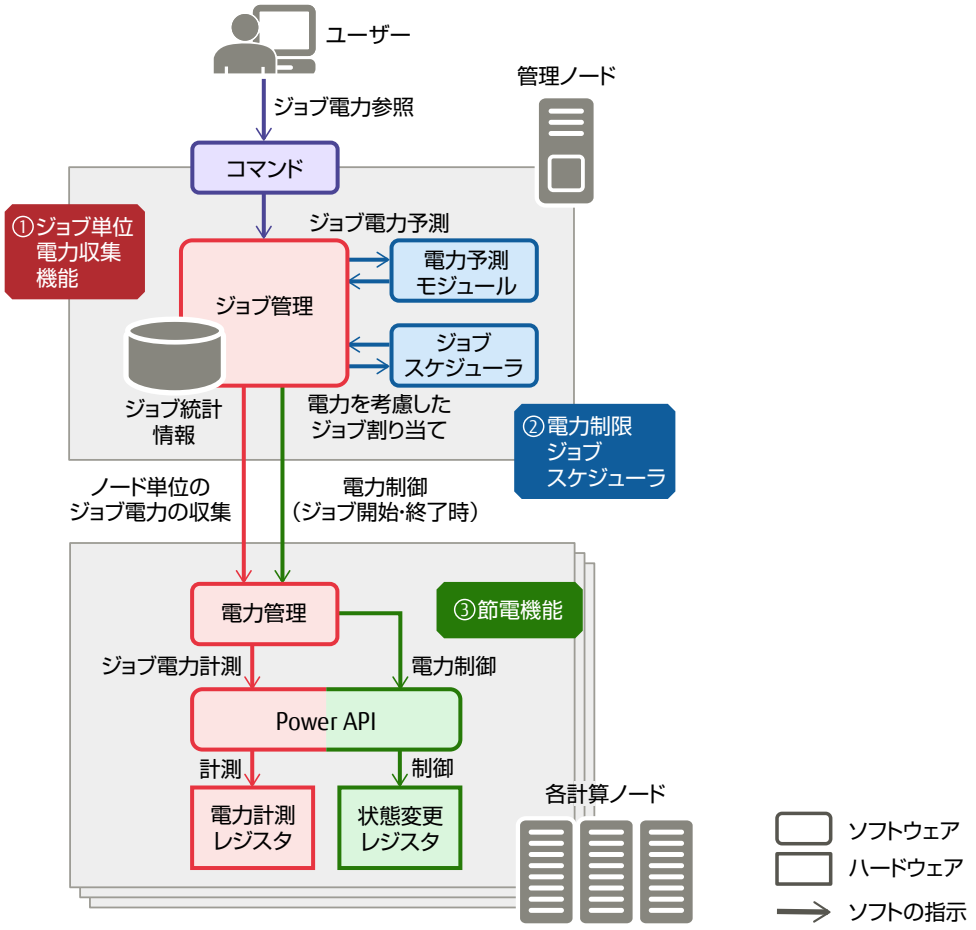

図-4　電力管理の各機能の動作ノードと構成

ザーへの課金額は，これらの情報に基づいて算出することができる。この機能については，以降の節で詳細に述べる。②は，システムの消費電力が所望の値を超えないようにジョブ実行を制御する「電力制限ジョブスケジューラ」[11] である。③は，ジョブの実行性能に影響を与えない電力制御として，計算ノードのジョブ実行有無に応じたデバイス制御による「節電機能」である。電力管理機能の詳細については，文献 [12] を参照いただきたい。

3.2 消費電力計測・評価の課題

システムの設計において，ジョブの実行する処理が同一である場合には，同じ性能や消費電力を示すことが望まれる。しかし「富岳」においては，計算ノードの個体差，および計算ノードの種別によって，消費電力にばらつきが発生してしまう。

複数の計算ノードから構成されるスーパーコンピュータでは，CPUやメモリーの半導体製造プロセスにおけるトランジスタ特性のばらつきや，動作電圧の最適化などにより，計算ノードごとに消費電力は異なる。また，一般的なスーパーコンピュータの利用方法ではシステムの計算ノード数全てを利用するジョブを実行することはまれであり，複数のジョブが同時にシステム上で実行されていることがほとんどである。このため，他のジョブが利用している計算ノードに応じて，ジョブ実行の度に異なる計算ノードが利用されることから，ジョブごとの消費電力のばらつきが発生する。

更に，「富岳」の計算ノードの一部は，計算ノード兼IOノードとして，計算処理に加えてストレージとのファイルIO処理なども行う役割を担っている。これは「富岳」固有のシステム設計に由来するばらつきであり，計算ノード兼IOノードと，通常の計算ノードとの間には，以下の違いがある。

・アシスタントコアの数

通常の計算ノードには2個のアシスタントコアが搭載されている。一方，計算ノード兼IOノードはIO処理をこなすために，通常の計算ノードより2個多い合計4個のアシスタントコアを搭載している。

・搭載されているデバイス

計算ノード兼IOノードには，ストレージそのものや外部ストレージとの接続に必要なPCI Express（PCIe）デバイスが搭載されている。

3.3 推定電力の導入

前節で述べた通り，ジョブ実行の度に異なる計算ノードが使用されるため，計算ノードの消費電力のばらつきに起因してジョブの消費電力は一意には定まらない。この課題を解決するために，「富岳」では実際の消費電力（実測電力）に加えて，消費電力のばらつき要因に影響されない電力指標として推定電力を設計・導入し，ユーザーがジョブの消費電力などの評価に利用することを促進する。

(1) 要件

推定電力は，公平な消費電力に関するジョブ統計情報の収集や，電力の消費傾向を踏まえた上でのユーザーによるジョブの消費電力最適化に利用することを目的としている。このことを踏まえ，推定電力の設計において求められる要件は以下である。

（a）アプリケーションプログラムの処理内容のみに応じて推定電力値が定まること

（a-1）計算ノードの個体差による消費電力のばらつきに影響されないこと

（a-2）計算ノード種別による消費電力のばらつきが排除されていること

（b）実測電力の増減に応じて，推定電力も増減すること

(2) 設計

要件（a-1）および要件（b）に対応するために，CPUやメモリーの各種回路のアクティビティをチェックし，その稼働率をベースに推定電力を計算する。また，要件（a-2）に対応するために，推定電力の計算に当たり，ジョブ以外の動作に利用されるアシスタントコアや，計算ノード種別によって搭載有無が異なるPCIeデバイスの消費電力を，計算ノード合計の対象から除外する。なお，推定電力は各種回路のアクティビティから計算するため，CPU内の回路ブロック単位で消費電力を計算することも可能であり，細粒度の電力情報が提供できる特長もある。

3.4 推定電力の評価

推定電力の妥当性を示すために，以下の二つについて評価した。

・推定電力のばらつき

・推定電力と実測電力の関係

評価には，HPC（High Performance Computing）システムのより現実的な性能を測定するHPC Challenge Benchmarkに含まれるベンチマークを利用した。計算ノードの消費電力は，CPUとメモリーの稼働率が大きく影響する。そのため，CPU負荷の高いルーチンとして，行列同士の積の演算処理であるDGEMMベンチマークを利用した。また，メモリー負荷の高いルーチンとして，配列のベクトル演算を行うことで実効メモリーバンド幅を測定するSTREAMベンチマークを利用した。ベクトル演算としては，メモリー負荷をより高めるために，ベクトルの各要素を定数倍するSCALA演算のMULTIPLY命令をXOR命令に変更したものを利用した。

CPUおよびメモリーの負荷率の違いによる推定電力の変化を評価するために，合計48個の計算コアを用いて，DGEMMとSTREAMを実行するコア数を変化させた。負荷パターンが異なるプログラムそれぞれを192計算ノード並列で実行し，各計算ノードの推定電力および実測電力を測定した。推定電力と実測電力の測定は，「富岳」の運用ソフトで実装されているPower API [10] を使用し，CPUの動作周波数は2.0 GHzとした。

(1) 推定電力のばらつき

表-2に，各負荷パターンに対する計算ノードごとの推定電力と実測電力の平均値および標準偏差を示す。全ての負荷パターンにおいて，推定電力は実測電力に比べてばらつきが1/10程度に抑えられている。すなわち，推定電力は計算ノードのばらつきを排除できており，要件（a）を満たしていることが分かる。

(2) 推定電力と実測電力の関係

図-5に，各負荷パターンにおける平均の推定電力

表-2　負荷パターンとノード当たりの推定電力および実測電力の関係

負荷パターン		ノード当たり推定電力（W）		ノード当たり実測電力（W）	
DGEMM実行コア数	STREAM実行コア数	平均	標準偏差	平均	標準偏差
48	0	150.3	1.3	166.6	13.6
40	8	169.2	1.2	195.3	14.2
32	16	175.5	1.3	204.3	14.1
24	24	172.2	1.1	200.5	13.4
16	32	171.5	1.1	200.4	13.5
8	40	166.7	1.1	194.3	13.2
0	48	161.3	1.1	180.8	11.7

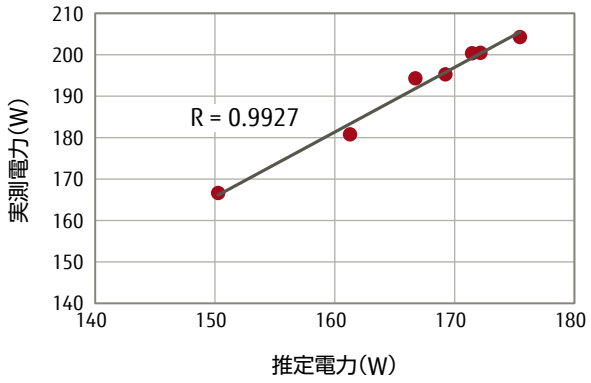

図-5　推測電力と実測電力の関係

と実測電力の関係を示す。相関係数Rは1に近い値であり，推定電力と実測電力はほぼ比例の関係にある。このことから，実測電力の増減に応じて，推定電力も増減し，要件（b）を満たしていることが分かる。

4. むすび

本稿では，スーパーコンピュータ「富岳」において，「京」からの改善・強化点であるファイルシステムおよび運用ソフトの電力管理機能について述べた。

ファイルシステムについては，階層化ストレージのユーザビリティを向上し，かつアプリケーションのファイルIO最適化を可能にするために，ジョブ実行領域専用のファイルシステムとして開発したLLIOについて述べた。また，LLIOの性能を評価した結果，高いファイルIO性能を持つことを示した。

更に，「富岳」のみならず，超大規模システムにおいて共通の課題であるシステムの省電力化，電力・ノード資源の効率化に必要な電力管理機能について述べた。また，複数計算ノードから構成されるスーパーコンピュータシステムにおける電力評価の課題と「富岳」における解決策として，推定電力の設計・導入効果について述べた。ジョブを実行するユーザーは，推定電力を利用することで，ジョブ実行された計算ノードを意識することなく単位電力当たりの性能を評価できるものと考える。運用者の観点では，利用する計算ノードに依存しない公平なジョブの消費電力に関する統計情報の収集を可能とし，運用への利用に耐えうる目処がついたと考える。

今後は，「富岳」の一般共用開始に向け，実際の利用・運用に即した環境でのソフトウェアの調整作業を継続して実施していく。その過程において，大規模な実アプリケーションによる性能・電力評価を行い，その成果を今後公開していく予定である。

参考文献・注記

[1] 張雷 他：スーパーコンピュータ「富岳」におけるOSの強化機能．富士通テクニカルレビュー，2020年No. 3.
https://www.fujitsu.com/jp/about/resources/publications/technicalreview/2020-03/article06.html
[2] 渡辺健介 他：スーパーコンピュータ「富岳」向けのアプリケーション開発環境．富士通テクニカルレビュー，2020年No. 3.
https://www.fujitsu.com/jp/about/resources/publications/technicalreview/2020-03/article07.html

[3] 酒井憲一郎 他：スーパーコンピュータ「京」の高性能・高信頼ファイルシステム．FUJITSU，Vol. 63, No. 3，p. 280-286（2012）．
http://img.jp.fujitsu.com/downloads/jp/jmag/vol63-3/paper08.pdf
[4] 宇野篤也 他：スーパーコンピュータ「富岳」の運用系ソフトウェア．富士通テクニカルレビュー，2020年No. 3.
https://www.fujitsu.com/jp/about/resources/publications/technicalreview/2020-03/article10.html
[5] 理化学研究所 計算科学研究センター：Fugaku System Configuration.
https://postk-web.r-ccs.riken.jp/spec.html
[6] 岡崎亮平 他：高性能・高密度実装・低消費電力を実現するスーパーコンピュータ「富岳」のCPU A64FX．富士通テクニカルレビュー，2020年No. 3.
https://www.fujitsu.com/jp/about/resources/publications/technicalreview/2020-03/article03.html
[7] GitHub：IOR.
https://github.com/hpc/ior
[8] 富士通：スーパーコンピュータ「富岳」TOP500、HPCG、HPL-AIにおいて世界第1位を獲得.
https://pr.fujitsu.com/jp/news/2020/06/22.html
[9] Top500.org：TOP500 LIST - JUNE 2020.
https://www.top500.org/lists/top500/list/2020/06/
[10] Sandia National Laboratories：High Performance Computing Power Application Programming Interface (API) Specification.
https://powerapi.sandia.gov/
[11] 秋元秀行 他：システム消費電力の上限を意識したポスト「京」向けジョブ運用ソフトウェアの実現に向けて．情報処理学会研究報告，Vol. 2015-HPC-152, No. 1（2015）．
[12] 富士通：White paper - FUJITSU Supercomputer PRIMEHPC FX1000 先進のソフトウェア.
https://www.fujitsu.com/downloads/JP/jsuper/primehpc-fx1000-soft-ja.pdf

著者紹介

秋元　秀行（あきもと　ひでゆき）
富士通株式会社
プラットフォームソフトウェア事業本部
スーパーコンピュータ「富岳」の運用ソフトにおける電力管理機能の開発に従事。

岡本　拓也（おかもと　たくや）
富士通株式会社
プラットフォームソフトウェア事業本部
スーパーコンピュータ「富岳」のファイルシステムの開発に従事。

加賀美　崇紘（かがみ　たかひろ）
富士通株式会社
プラットフォームソフトウェア事業本部
スーパーコンピュータ「富岳」の運用ソフトにおける電力管理機能の開発に従事。

関　堅（せき　けん）
富士通株式会社
プラットフォーム開発本部
スーパーコンピュータ「富岳」のシステムハードウェアの開発に従事。

酒井　憲一郎（さかい　けんいちろう）
富士通株式会社
プラットフォームソフトウェア事業本部
スーパーコンピュータ「富岳」のファイルシステムの開発に従事。

今出　広明（いまで　ひろあき）
富士通株式会社
プラットフォームソフトウェア事業本部
スーパーコンピュータ「富岳」の運用ソフトにおける電力管理機能の開発に従事。

篠原　誠（しのはら　まこと）
富士通株式会社
プラットフォームソフトウェア事業本部
スーパーコンピュータ「富岳」の運用ソフトの開発に従事。

住元　真司（すみもと　しんじ）
富士通株式会社
プラットフォームソフトウェア事業本部
スーパーコンピュータ「富岳」の運用ソフト・言語ソフトの開発に従事。