

# ジョブの時系列消費電力の推定

宇野 篤也<sup>1,a)</sup> 末安 史親<sup>3</sup> 山本 啓二<sup>1</sup> 肥田 元<sup>2</sup> 池田 直樹<sup>2</sup> 辻田 祐一<sup>1</sup>

**概要：**近年，計算機システムの大規模化にともない，システムの消費電力は運用上の課題の一つとなっている。システムの最大消費電力を前提に施設等が設計されている場合は消費電力はさほど大きな問題となるないが，施設等の消費電力の上限がシステムの最大消費電力よりも低く設定されている場合は，ジョブ実行時の消費電力を考慮して運用を行う必要がある。ジョブ毎の消費電力を考慮してジョブスケジューリングを行う場合，ジョブが実行される前にそのジョブの消費電力を推定し，スケジューリングを行う必要がある。これまで，投入されたジョブの要求資源やジョブスクリプトから得られる情報をもとにジョブ毎の時系列の消費電力を推定する手法を検討してきた。今回，「京」で実行されたジョブのデータを用いてジョブ毎の時系列の消費電力を推定し評価を行ったので報告する。

## 1. はじめに

近年，計算機システムの大規模化，高性能化にともない計算機システムの消費電力は増大の傾向にある。現在，理化学研究所が中心になって開発を行っているスーパーコンピュータ「富岳」でも消費電力は重要な課題の一つであり，効率のよいシステム電力の運用が必要とされている [1]。

理化学研究所が運用しているスーパーコンピュータ「京」[2] の消費電力は，無負荷時で約 10MW, Linpack など高負荷ジョブ実行時には 14MW を超える場合もある。これまでに開発された「京」を含めた多くのシステムでは，設計段階から最大消費電力を上限としてシステムおよび施設が作られている。しかし，計算機システムの大規模化にともなう最大消費電力の増加に，同じ手法で対応することは難しくなってきている。そのため，施設等の最大消費電力を計算機システムの最大消費電力よりも低く設計し，計算機システムの消費電力が施設の最大消費電力を超過しないように運用するといった対応が求められつつある。これは，通常の運用では計算機システムで消費される電力は最大消費電力よりも低い場合が多いという点を利用したものである。これらの運用を実現するために，オーバープロビジョニングや電力制約適応型システムといった方式を用いたシステム電力を制限内で最大限に活用する手法が提案されている [3][4]。これらの方では，ジョブ実行時の CPU やメモリなどの消費電力を動的に管理し，システム全体の消費

電力が許容範囲内に納まるようジョブの実行を制御する。しかし，ジョブの実行効率の観点から見た場合，CPU の周波数等の操作はなるべく行うことなく実行できることが望ましい。

ジョブの実行効率を下げることなく電力制限内でジョブを実行する方法として，我々はジョブの消費電力の変動を考慮したジョブスケジューリングを検討している。本手法では，ユーザがジョブを投入した時点でそのジョブの消費電力を推定する。これまでに，投入されたジョブスクリプトから得られる情報をもとにジョブ毎の時系列の消費電力を推定する手法を検討してきた [5][6]。今回，「京」で実行されたジョブのデータを用いてジョブ毎の時系列の消費電力を推定し評価を行ったので報告する。

## 2. 消費電力の変動を考慮したジョブスケジューリング

我々が提案する消費電力の変動を考慮したジョブスケジューリングでは，ジョブが投入された時点でジョブの消費電力を推定しジョブ毎の消費電力の変動を考慮したスケジューリングを行う。

図 1 に我々が提案するジョブスケジューリングの概要を示す。本手法では，ユーザが投入したジョブの消費電力を過去の実行結果から推定し，システムの消費電力が上限値に近くなるようにスケジューリングを行う。このフレームワークは，消費電力推定，ジョブスケジューリング，資源管理の 3 つのモジュールで構成される [7]。各モジュールの機能概要は以下の通りである。

消費電力推定モジュール ジョブの実行結果に基づいた電

<sup>1</sup> 国立研究開発法人理化学研究所 計算科学研究センター

<sup>2</sup> 株式会社富士通ソーシアルサイエンスラボラトリ

<sup>3</sup> 富士通株式会社

a) uno@riken.jp

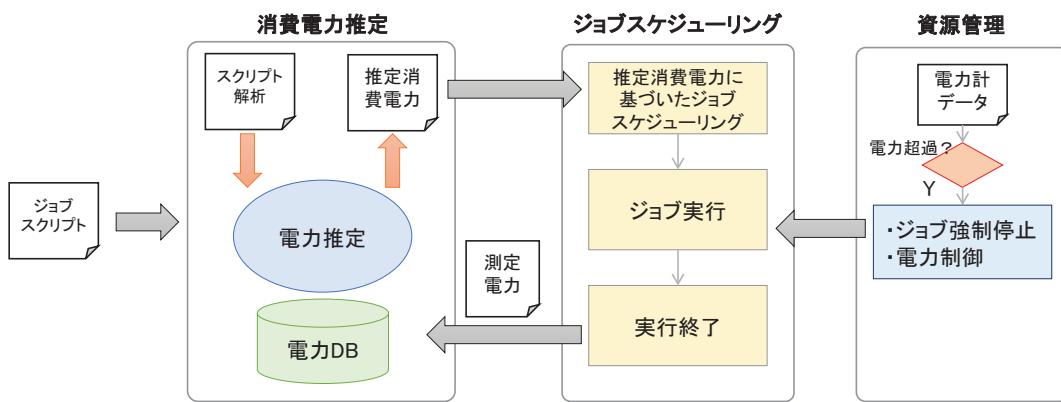


図 1 消費電力の変動を考慮したジョブスケジューリングの概要

カデータベースの構築と、ユーザが投入したジョブの消費電力推定を行う。  
ジョブスケジューリングモジュール システム電力の範囲内で効率よく計算ノードを利用できるようにスケジューリングを行う。  
資源管理モジュール システム全体の消費電力を監視し、上限値を超えないように制御を行う。

### 3. ジョブの消費電力の推定

消費電力の変動を考慮したジョブスケジューリングを行うためには、ジョブ投入時点でジョブ毎の消費電力を推定する必要がある。ジョブ投入時に得られる情報（以下、ジョブ投入時情報と呼ぶ）は、ジョブパラメータやジョブスクリプト、過去の実行実績等である。例えば「京」の場合、ジョブスクリプトにはユーザが指定したノード数やジョブの実行時間、ジョブが使用するファイル名（ステージング情報）、ロードモジュール（LM）のファイル名、ジョブ内の処理内容などが記載されている。これまでに、これらジョブ投入時情報をもとに、ジョブで実行されるアプリケーションを推定する手法を確立した<sup>[5]</sup>。「京」で実行されたジョブで評価したところ、約90%の精度でジョブ投入時情報からアプリケーションを推定することが可能であった。また、過去の実行実績には、実行されたLMの情報、消費電力の情報等といった情報が含まれる。

ジョブ毎の消費電力を推定するには、ジョブ投入時情報と過去の実行実績（消費電力）を紐付ける必要がある。「京」で実行されたジョブについて、ジョブ投入時情報とそのジョブの消費電力の関係を調査したところ、ユーザはほぼ同じ特性のジョブを繰り返し実行することが多く、その場合の消費電力は類似していることが判った<sup>[6]</sup>。しかし、ジョブ投入時の指定パラメータ等が全く同一のジョブの場合でも、その消費電力が全く同じになるわけではない。これは、実行されるノードが毎回同じとは限らないうえ、同時に他で実行されているジョブの影響を受ける場合があるからである。そのため、消費電力データのクラスタリング

を行う。

#### 3.1 消費電力データのクラスタリング

膨大なデータを分析する際、データに含まれる特徴を抽出する手法として、対象となるデータのクラスタリングを行う。特に、時系列データのクラスタリングは、様々な分野で複雑かつ膨大なデータから意味のある特徴を抽出するのに使用されている<sup>[9]</sup>。時系列データのクラスタリングでは、分類の手段として対象となるデータの類似性を利用することが多い。今回のジョブの消費電力の時系列データでは、クラスタリングに時系列データの形状および総消費電力量を利用した。形状の類似性のみ利用した場合、単純に拡大または縮小された波形の比較で類似性が高く評価されてしまうという問題が起こりうる。これを防ぐために、総消費電力量も比較に用いることにした。

今回の評価で使用した「京」の消費電力データは5分または10分間隔で計測されたものである。計測のタイミングはジョブの開始時刻ではなく、システム全体の経過時間を基準にしているため、仮に全く同じ消費電力データだとしても計測結果は異なる場合がある。また、消費電力データの比較を行う際、対象点数が多い複雑なデータの比較は計算量が増えるという問題もある。さらに、ジョブスケジューリングの観点からは細かい間隔でのデータはそれほど必要とされることもあることから、一定区間の平均値を計算しデータの簡略化を行うことにした。「京」では、30分間隔の消費電力が閾値を超えないようにジョブの実行を制御している（超過が予測された場合は、実行中のジョ

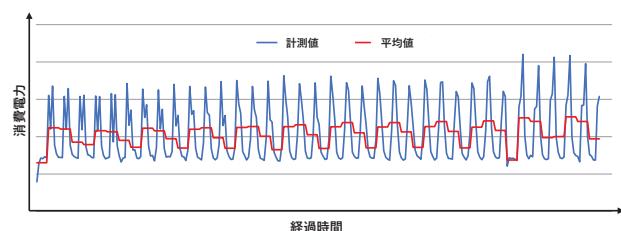


図 2 計測データと平均値データ

表 1 クラスタリングに使用したジョブ投入時情報

名称	説明
ユーザ ID	ユーザ ID
ノード数	ジョブ投入時に指定されたノード数
指定経過時間	ジョブ投入時に指定された経過時間
アプリケーション種別	ジョブスクリプトから推定したアプリケーションの種類

ブを停止して消費電力を削減する) [8]。そこで、今回の評価も 30 分間隔の平均値とした。図 2 に 30 分毎の平均値に変換した結果を示す。今回は平均値を使用したが、目的に合わせて最大値や平均 +3σ 等を利用することも考えられる。

時系列データの形状比較の手法にはいくつかあるが、今回は以下の 2 手法を使用した。

- ヨークリッド距離
- 動的時間伸縮法 (DTW:Dynamic Time Warping)

ヨークリッド距離は、対象となる 2 点の距離を求めることで類似度を比較する。消費電力の時系列データをデータ数の次元のベクトルとみなし、2 点間の距離を計算する。消費電力データ  $p$  と  $q$  を比較する場合の計算式は以下のようになる。計算結果が 0 に近い方が類似度が高い。

$$d(p, q) = \sqrt{\sum (p_i - q_i)^2}$$

動的時間伸縮法 (DTW) は、形状の類似度を比較するため、時間または大きさの異なる形状でも比較できるという特徴がある。DTW では、時系列データの各点に対して総当たりで距離を計算し、要素が各点同士の距離となる行列を作成する。そして、始点から終点まで通る経路のうち、行列の要素の合計が最小となる経路を探し、その経路の要素の合計を対象データの距離とする。結果が 0 に近い方が類似度が高い。

なお、今回の比較では、計算量を削減するため実経過時間の差が 1 時間以内の場合を比較対象とし、不足側のデータの当該区間の消費電力は 0 として処理している。

#### 4. 評価

今回提案する手法について評価を行った。今回の評価では、2016 年 8 月から 2017 年 3 月までに「京」上で実行されたジョブを使用した。評価では前述の手法で作成した 30 分間隔の平均消費電力の時系列データを使用するため、ジョブの実行時間が 1 時間以上のジョブを対象とした。また、消費電力はノード単位の平均値を使用した。

##### 4.1 消費電力データのクラスタリング

分類手法による分類結果への影響を評価した。クラスタリングを行うには分類の基準となる閾値を決める必要がある。クラスタ内での各データの距離を計算し最適な閾値を求めるといった手法も提案されているが、今回はジョブ投入時情報のみを利用した分類結果をもとに各分類手法を用

いた場合の標準偏差を求め、それを基準に閾値を求めるにした。ジョブ投入時情報のうち、アプリケーション種別、ノード数、指定経過時間、ユーザ ID を使用して分類を行った(表 1)。

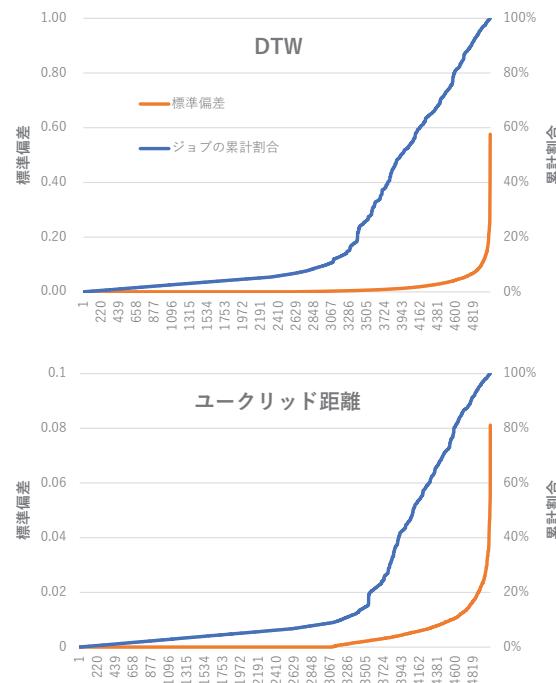


図 3 入力時情報でクラスタリングした場合の各手法の標準偏差

図 3 に、ジョブ投入時情報を用いて分類し、各クラスタに属するデータとクラスタの平均データとの距離の標準偏差を求めた結果を示す。それぞれでクラスタを標準偏差の小さい順にソートしているため、DTW とヨークリッド距離の累計割合は異なっている。DTW とヨークリッド距離の結果を比較すると、同じクラスタでも DTW の距離はばらつきが少ないことがわかる。図 4 に、ジョブ投入時情報で分類した場合の、分類されたクラスタの平均総消費電力

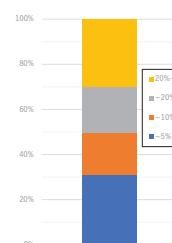


図 4 入力時情報でクラスタリングした場合の総消費電力量の誤差

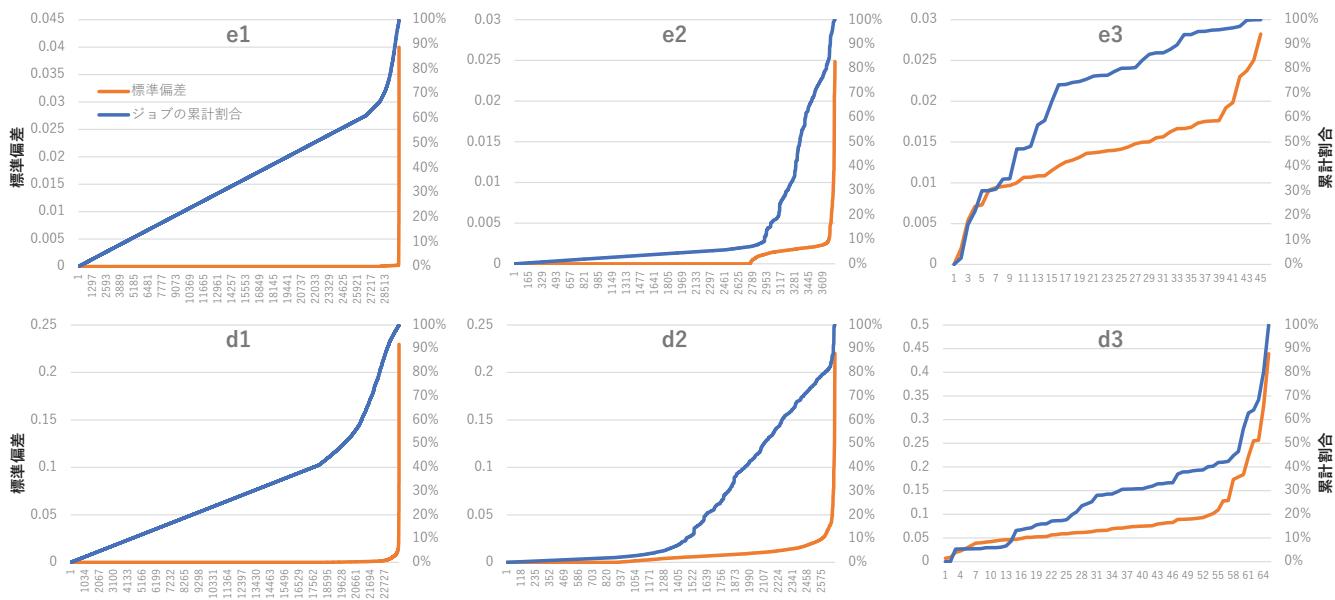


図 5 各分類条件におけるケース毎の標準偏差

量に対するそのクラスタに属するジョブの総消費電力量の誤差の割合を示す。約 50% のジョブが 10%以下の誤差、約 70% のジョブが 20%以下の誤差であった。誤差が 20%以上のジョブを調査したところ、同一クラスタに分類されたジョブ数が少なく、かつ、それらの総消費電力量の差が大きいため、平均値との差が大きくなっていた。

今回使用した閾値を表 2 に示す。これらは、図 3 の結果から全ジョブ数の 8 割が含まれる場合の標準偏差を基準に決定した。

表 2 クラスタリングに用いた閾値

分類手法	(1)	(2)	(3)
ユークリッド距離	0.001035	0.01035	0.1035
動的時間伸縮法	0.004	0.04	0.4

これらの閾値用いて消費電力の時系列データを分類した。表 3 に各パラメータで分類した場合のクラスタ数を示す。閾値を下げることで細分類化されていることがわかる。

表 3 分類結果

分類手法	(1)	(2)	(3)
ユークリッド距離	29,795	3,754	45
動的時間伸縮法	23,755	2,685	65

図 5 に各条件で分類した場合の標準偏差を、図 6 に分類した各グループ毎にノード単位での消費電力を求め各ジョブ毎の消費電力との相対誤差を計算した結果をそれぞれ示す。クラスタは標準偏差の小さい順にソートしている。図中の d,e はそれぞれ DTW, ユークリッド距離を表し、1 から 3 の数字は使用した閾値を示している。閾値を下げることで細分類化されたため、相対誤差が小さくなり、電力誤差も小さくなっている。これらの結果から、DTW もユー

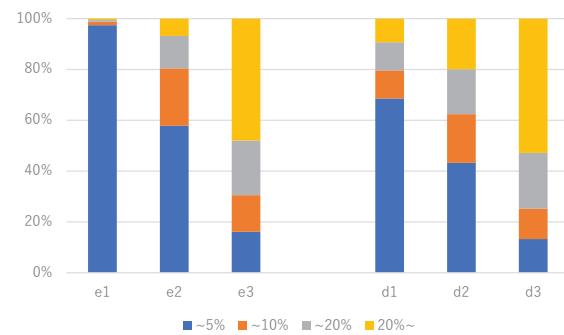


図 6 各分類条件における総消費電力量の誤差

クリッド距離も閾値を適切に設定することで同様の振る舞いを得ることができると思われるが、計算量の観点からはユークリッド距離が有利と考えられる。

#### 4.2 機械学習による消費電力推定

次に機械学習を用いて、ジョブ毎の時系列消費電力の推定を行った。予測モデルの入力パラメータとして、ユーザID、指定経過時間、ノード数、アプリケーション種別を使用し、出力パラメータを消費電力データの分類結果とした。

消費電力データの分類方法別に推定結果を評価した。機械学習のアルゴリズムには Random Forest[10] を使用した。Random Forest は複数の決定木を用いて、その結果を平均化する集団学習アルゴリズムの一種であり、様々な機械学習手法の中でも良く知られている手法である。「京」で実行されたジョブのデータをもとに前述のクラスタリング手法に基づいてデータセットを作成した。作成したデータセットからジョブ ID 順に 20,000 件のデータを抽出し、このうち 16,000 件を学習データに、4,000 件をテストデータにランダムに選択して評価した。期間中の全てのデータで

はなく、20,000 個を抽出したのは評価環境の制限によるものである。評価指標には、推定した分類結果の正解率と、ジョブ全体の消費電力の誤差の 2 つを用いた。図 7 に分類結果の正解率を、図 8 にジョブ全体の消費電力の誤差をそれぞれ示す。閾値を大きくした場合、同一クラスタに分類されるジョブ数が多くなるため正解率は高い。一方、図 6 の結果からもわかるように、閾値を大きくした場合は総消費電力の誤差は大きくなり、推定消費電力の誤差も大きくなる。推定精度を改善するには、クラスタに分類した際の誤差を小さくし、かつ、分類結果の正解率も改善する必要がある。

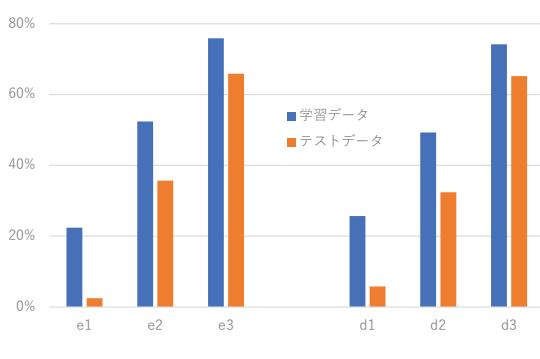


図 7 各分類条件における推定結果

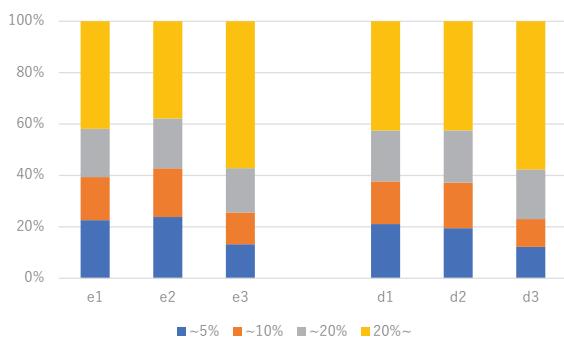


図 8 各分類条件における推定誤差

クラスタリングをジョブ投入時情報のみで行うことでき分類結果の正解率を高くすることが可能である。これは、ジョブ投入時情報で一意にクラスタを決定することができるからである。一方、分類に消費電力データは考慮されないため、同一クラスタ内での消費電力データの誤差は大きくなる(図 4)。これを改善するために、クラスタリングにジョブ投入時情報とユークリッド距離の両方を使用した。今回の評価では、ユークリッド距離の閾値は表 2 の(2) 0.01035 を使用した。図 9 に推定消費電力量の誤差を図 10 に正解率をそれぞれ示す。ジョブ投入時情報とユークリッド距離の両方を使用してクラスタリングを行うことで、ジョブ投入時情報のみを使用した場合より正解率は下がるもの、推定消費電力の誤差は改善できていることが

わかる。今回の推定では、ジョブ投入時情報のみで推定を行っているが、ジョブ実行開始直後の電力データを推定に使用することで、更に精度を上げることができると考えている。

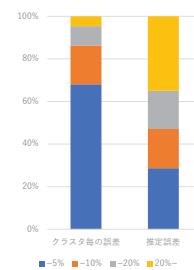


図 9 ジョブ投入時情報とユークリッド距離でクラスタリングを行った場合の推定誤差

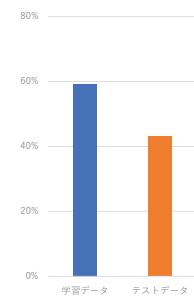


図 10 ジョブ投入時情報とユークリッド距離でクラスタリングを行った場合の正解率

#### 4.3 システム全体の消費電力推定

提案手法を用いて、システム全体の消費電力の推定を行った。クラスタリングには、ジョブ投入時情報とユークリッド距離を使用した。学習データとして 2017 年 1 月から 2 月の間に 1 時間以上実行されたジョブ 12,250 件を使用し、2017 年 3 月に 1 時間以上実行されたジョブ 7,560 件の消費電力データの推定を行った。図 11 にシステム全体の消費電力の測定値と推定値、推定値の誤差を示す。対象期間中の平均誤差は約 31% であった。推定値と測定値が大きく乖離しているジョブについて調査を行った。図 12 にジョブ毎のノード数別の 30 分あたりの推定誤差を示す。一部ノード数で誤差率が大きい場合があるが、全体的に誤差率に大きな差は見られない。一方、推定誤差量はノード数に比例して大きくなっていることがわかる。これは、今回の消費電力をノード単位で推定していることによるものと考える。

そこで、ジョブ単位の消費電力を同じ推定手法で推定した。図 13 にジョブ単位で推定したシステム全体の消費電力の測定値と推定値、推定値の誤差を、図 14 にジョブ毎のノード数別の 30 分あたりの推定誤差を示す。ノード単位で推定した場合と比べて推定値は改善され、対象期間中

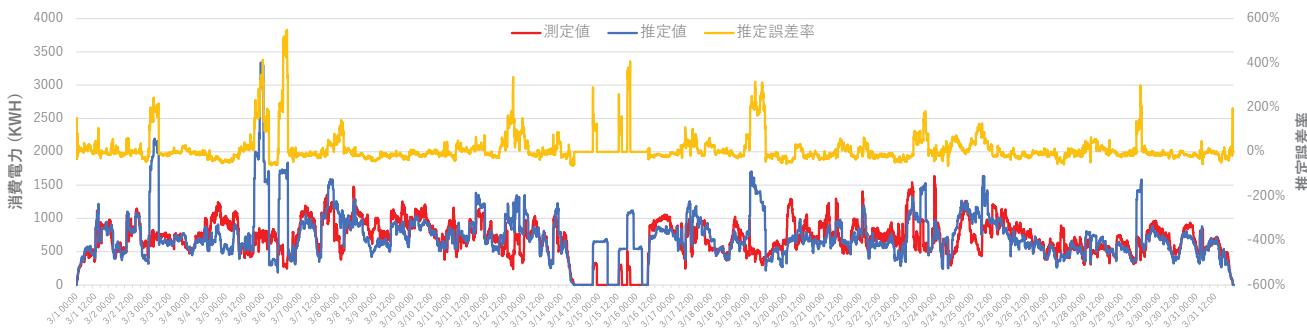


図 11 システム全体の消費電力の測定値と推定値および測定誤差（ノード単位で推定）

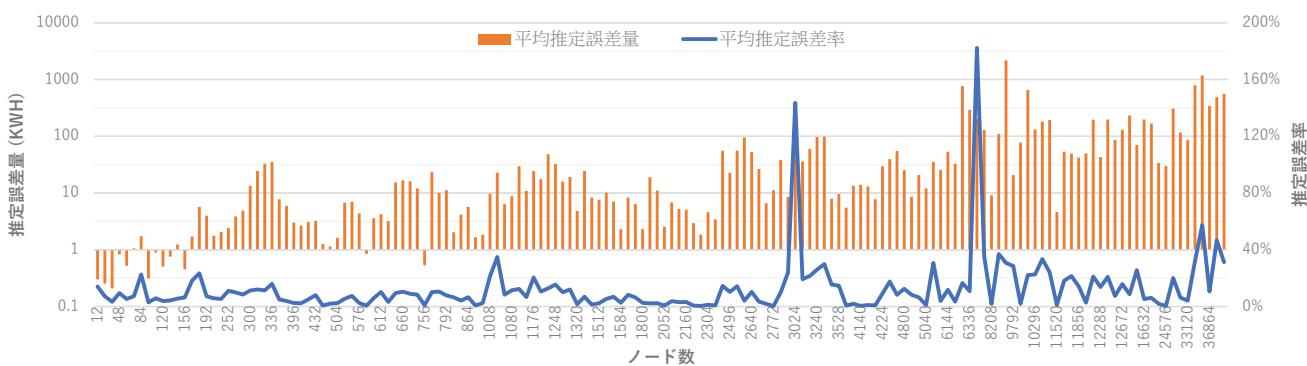


図 12 ジョブ毎のノード数別の推定誤差（ノード単位で推定）

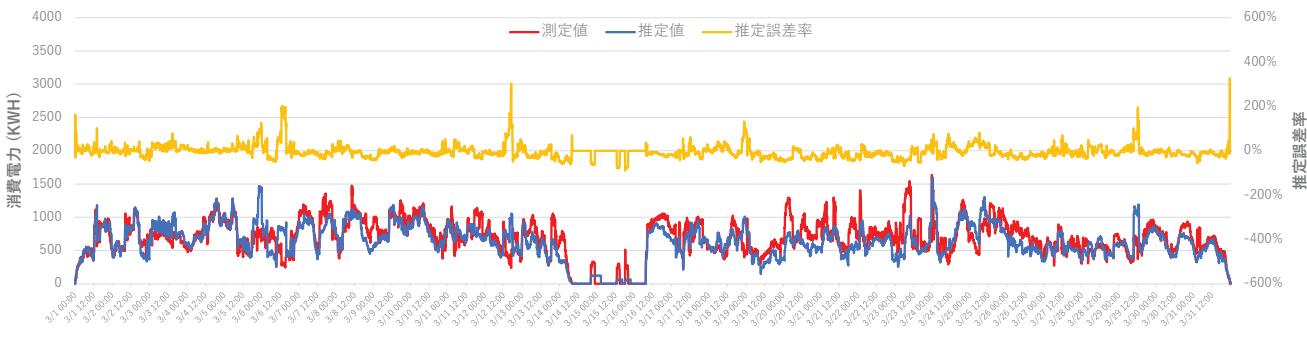


図 13 システム全体の消費電力の測定値と推定値および測定誤差（ジョブ単位で推定）

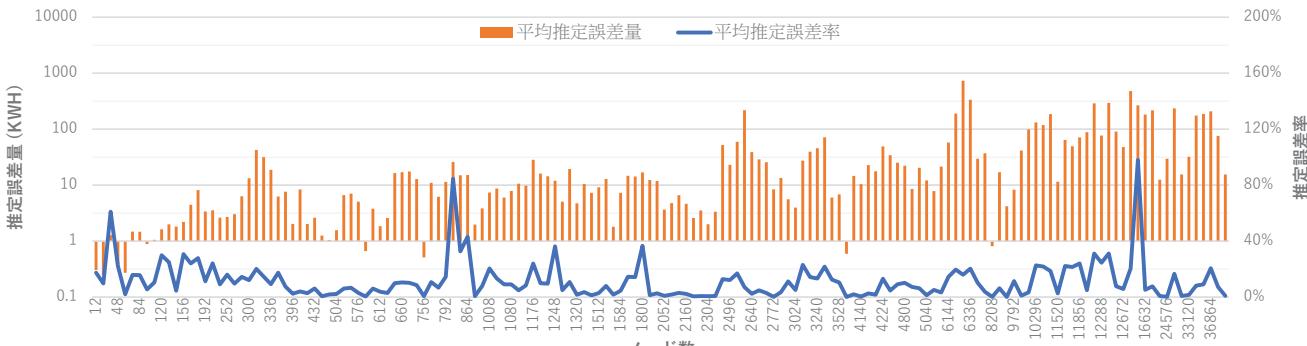


図 14 ジョブ毎のノード数別の推定誤差（ジョブ単位で推定）

の平均誤差は約 19% となった。

他に、推定誤差が大きくなる原因として、ある特性のジョブが学習データに含まれずテストデータにのみ含まれる場合がある。今回は、学習データとテストデータを分けて推定を行ったため、テストデータにのみ含まれるジョブの推定誤差は大きくなる可能性が高い。この問題について

は、ジョブ実行毎に学習を行うことで改善できるものと考えている。

## 5. おわりに

本稿では、投入されたジョブの消費電力を推定しジョブ毎の消費電力の変動を考慮したスケジューリングを行った

めに、ジョブの時系列電力変動の推定を行った。

ジョブ毎の消費電力の時系列データの推定では、ユーザがジョブを投入した際に得られる情報と過去に実行されたジョブのデータをもとに投入されたジョブの消費電力を推定する。推定にあたり、ジョブの消費電力データのクラスタリングを行うが、消費電力データとジョブ投入時情報を用いることで精度を改善できることを確認した。「京」で実行されたジョブ情報を用い消費電力を推定したところ、誤差が大きくなる場合もあるが、平均で19%程度の誤差であった。今回の評価では、ジョブ投入時情報のみを利用したが、推定精度を向上させるためには、さらなる情報を利用する必要があると考えている。例えば、ジョブ実行開始直後の電力データを利用することで、実行中のジョブの消費電力の推定精度を改善することが可能になると考えている。

今後は、推定手法を改良するとともに推定した時系列消費電力データを用いたジョブスケジューリングでの評価を行っていきたいと考えている。

## 参考文献

- [1] 秋元 秀行, 三浦 健一, 末安 史親, 平井 浩一, 住元 真司, 宇野 篤也, 山本 啓二, 塚本 俊之: 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)
- [2] 特集: スーパーコンピュータ「京」, 情報処理, Vol.53, No.8, pp.752–807 (2012)
- [3] Tapasya Patki, David K. Lowenthal, Anjana Sasidharan, Matthias Maiterth, Barry L. Rountree, Martin Schulz, Bro-nis R. de Supinski: “Practical Resource Management in Power-Constrained, High Performance Computing”, Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing, pp.121–132 (2015)
- [4] 坂本 龍一, カオ タン, 近藤 正章, 井上 弘士, 上田 将嗣, Tapasya Patki, Daniel Ellsworth, Barry Rountree, Martin Schulz: オーバープロビジョンング環境での大規模 HPC システムの電力と性能評価, 情報処理学会研究報告, Vol.2017-HPC-160, No.1 (2017)
- [5] Keiji Yamamoto, Yuichi Tsujita, and Atsuya Uno: “Classifying Jobs and Predicting Applications in HPC systems,” ISC High Performance 2018, Lecture Notes in Computer Science Vol. 10876, pp.81–99 (2018)
- [6] 宇野篤也, 末安史親, 山本啓二, 肥田元, 池田直樹, 辻田祐一: “ジョブの時系列電力変動の推定手法の検討”, 情処研報 Vol.2018-HPC-167 No.20, (2018)
- [7] 宇野篤也, 末安史親, 山本啓二, 肥田元, 池田直樹, 辻田祐一: “消費電力の変動を考慮した ジョブスケジューリングの検討”, 情処研報 Vol.2017-HPC-161 No.5, (2017)
- [8] 宇野篤也, 肥田 元, 井上 文雄, 池田 直樹, 塚本 俊之, 末安 史親, 松下 聰, 庄司 文由: 消費電力を考慮した「京」の運用方法の検討, 情報処理学会論文誌, ACS, Vol.8, No.4, pp.13–25 (2015)
- [9] Saeed Aghabozorgi, Ali Seyed Shirkhorshidi, Teh Ying Wah: “Time-series clustering - A decadal review,” Information Systems, Vol.53, pp.16–38 (2015)
- [10] Breiman, Leo: “Random Forests,” Mach. Learn., Vol.45, No.1, pp.5–32 (2001)