

ジョブの時系列電力変動の推定手法の検討

宇野 篤也^{1,a)} 末安 史親³ 山本 啓二¹ 肥田 元² 池田 直樹² 辻田 祐一¹

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を考慮した運用を行なう必要性が増している. ジョブ実行時にシステム側で CPU の周波数等を制御して消費電力が閾値を超えないように調整するといった手法も提案されているが, ジョブの実行効率の点ではそのような操作をなるべく行うことなく消費電力が超過しないようにジョブをスケジューリングし実行できることが望ましい. 我々はジョブ毎の消費電力の変動を考慮したジョブスケジューリング手法を提案している. 本手法では, ジョブが投入された時点でそのジョブの消費電力を予測し, ジョブスケジューリングを行う. これまで, 投入されたジョブスクリプトから実行されるアプリケーションを推測する手法を検討してきた. 今回, ジョブ毎の時系列消費電力の推定手法について検討を行い, 「京」で実行されたジョブのデータを用いて評価を行ったので報告する.

1. はじめに

計算機システムの大規模化, 高性能化にともないシステムの消費電力は増大の傾向にある. 消費電力は現在開発が進められているエクサスケール級のスーパーコンピュータでも重要な課題であり, 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. 「京」を含めた多くのシステムでは, 予想される最大消費電力を上限としてシステムおよび施設の設計が行われており, 計算機システムの大規模化による最大消費電力の増加に対応することは難しくなっている. 通常の運用では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く, 施設側の消費電力の上限をシステムの最大電力よりも低く設定し, オーバ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ることで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]. これらの方式では, 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提案している. 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必要がある. これまでに, 投入されたジョブスクリプトから実行されるアプリケーションを推測する手法を検討してきた [5]. 今回, ジョブ毎の時系列消費電力の推定手法について検討を行い, 「京」で実行されたジョブのデータを用いて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

図 1 に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の概要を示す. 本手法では, ジョブへの制約をできるだけ少なくしてシステム電力を最大限に活用できるように, 投入されたジョブの消費電力を予測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. 具体的には, 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測し, システム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大きく分けて, 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ている [6].

2.1 ジョブのアプリケーションの推定方法

ジョブ毎の消費電力の推定は消費電力推定モジュールで行う. 過去に実行されたジョブの情報に基づいてユーザが投入したジョブの消費電力を推定する.

これまでに, ジョブ毎の消費電力の推定を行うために,

¹ 国立研究開発法人理化学研究所 計算科学研究センター

² 株式会社富士通ソーシャルサイエンスラボラトリ

³ 富士通株式会社

a) uno@riken.jp

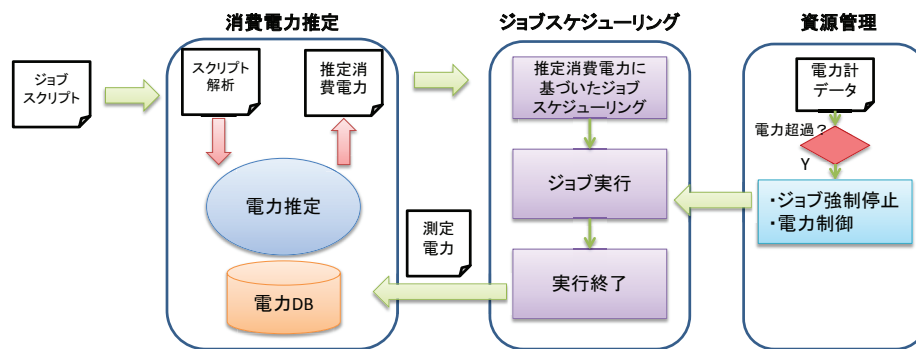


図 1 提案フレームワーク

表 1 ジョブスクリプトからのアプリケーションの推定結果

評価方法	ジョブ数	正解数	誤り数	正解率
Model:2016	87,410	72,418	14,992	83%
Model:all	214,714	195,441	19,273	91%

ユーザが投入したジョブのアプリケーションの推定方法について検討してきた。この推定手法では、ジョブの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。投入時情報とは、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ジョブが使用するノード数や指定経過時間、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。これらを利用して推定を行う。具体的には、ジョブスクリプトを文章と見なし、文章からその文章が所属するクラスを推定する文章分類問題としてジョブスクリプト进行分类する。分類には、ジョブスクリプトを入力、分類結果を出力とする機械学習モデルを使用している。この推定モデルでは、スクリプトを空白や記号で文字列を区切り分かち書きしたシンボル集合とみなし、LMの種類を doc2vec アルゴリズム [7] を用いて学習する。

2.2 ジョブのアプリケーションの推定結果

表 1 に「京」で実行されたジョブのアプリケーションの推定結果を示す。Model:2016 は、2016 年 8 月 22 日から 2016 年 12 月 31 日までに実行されたジョブスクリプトで推定モデルを作成し、2017 年 1 月 1 日から 2017 年 3 月 31 日までに実行されたジョブスクリプトを対象に評価した結果である。Model:all は、2016 年 8 月 22 日から 2017 年 3 月 31 日までに実行されたジョブスクリプトでモデルを作成し、2016 年 8 月 22 日から 2017 年 3 月 31 日までに実行されたジョブスクリプトを評価した結果である。

Model:2016 では、学習モデルにない新規のジョブが誤りとなり正解率は 83%で、Model:all では正解率は 91%であった。実際の運用では日々推定モデルを更新し新規の LM を学習することから、Model:all と同程度の約 9 割の

正解率になると予想している。

3. ジョブの消費電力の推定

ユーザがプロダクションランのようにほぼ同じ特性のジョブを繰り返し実行する場合、ノード障害等が発生しない限りその消費電力は類似の結果となることが予想される。一方、同じアプリケーションでも実行時のパラメータやノード数が異なる場合、その消費電力の振る舞いは異なることが予想される。

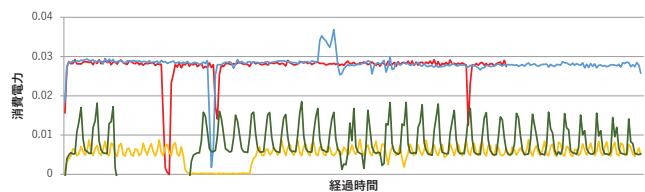


図 2 同一アプリケーションに分類されたジョブの消費電力

図 2 に同じアプリケーションに分類されたジョブの消費電力を示す。図 2 からわかるように、同じアプリケーションであってもその消費電力は大きく異なる場合があり、アプリケーションの推定だけではそのジョブの消費電力の推定は難しいことがわかる。そこで、ジョブ毎の消費電力の推定に使用するパラメータと推定消費電力の関係について調査を行った。

ジョブ毎の消費電力推定の入力パラメータとして、ユーザがジョブを投入する際に得られる情報（ジョブ投入時情報）を使用し、過去に実行されたジョブの消費電力データから投入されたジョブの消費電力を推定する。表 2 に入力パラメータに使用するジョブ投入時情報を示す。

同じ特性をもったジョブを実行した場合でも、その消費電力データは完全に一致するとは限らない。そのため、消費電力データの類似度で評価することにした。今回の評価で用いた消費電力データの比較方法について説明する。

今回使用した「京」の消費電力データは 10 分間隔で計測したものである。計測のタイミングはジョブの開始時刻からではなく、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

表 2 ジョブ投入時情報

名称	説明
UID, GID	ユーザ ID, グループ 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
ジョブクラス	ジョブスクリプトから推測したアプリケーションの種類

異なる場合があるうえ、複雑な波形の比較を行う必要がある。そこで、比較を容易にするため一定区間の平均値を計算し、区間毎に比較を行うことにした。図 3 に平均値データの作製方法を示す。一定区間における測定値の平均を求め、波形データをステップに変換する。図 4 に 30 分毎の平均値を求めた結果を示す。ここでは平均値を求めているが、目的に合わせて最大値や平均 $+3\sigma$ 等を利用することも考えられる。

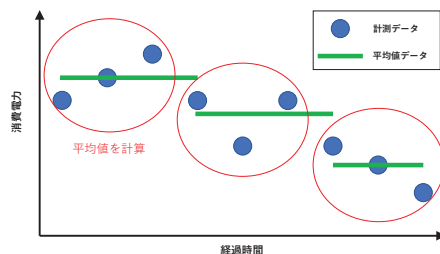


図 3 平均値データの作製方法

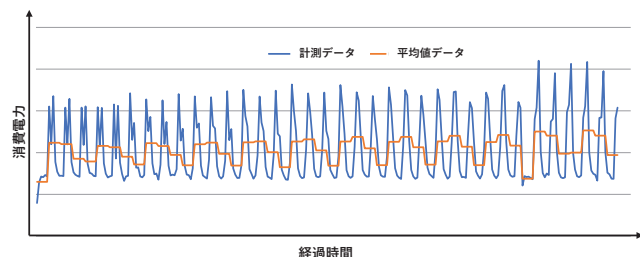


図 4 計測データと平均値データ

消費電力データの比較には、 n 次元のユークリッド距離を使用した。前述の方法で平均化したデータの各区間毎にユークリッド距離を計算する (図 5)。消費電力データ p と q を比較する場合の計算式は以下になる。計算結果が 0 に近い方が類似度が高い。

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

なお、データ量が一致しない場合は不足側のデータの当該区間の消費電力は 0 として比較する。

4. 評価

入力パラメータに基づく分類による消費電力推定と機械学習による消費電力推定について評価を行った。

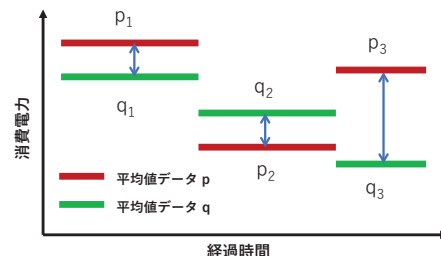


図 5 平均値データの比較方法

4.1 入力パラメータに基づく分類による消費電力推定

消費電力データの推定に用いるパラメータを変えることで、推定結果がどのように変わるか評価した。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。評価には消費電力の時系列データとして 30 分間隔の平均値を使用するため、実行時間が 1 時間以上あった 42,745 ジョブを対象した。

以下のパラメータを使用した場合について比較した。

- ジョブスクリプトから推定されたアプリケーションの種類
- グループ ID
- ユーザ ID

具体的には入力パラメータを変更してジョブ进行分类し、それらの消費電力データの標準偏差を求め比較している。また、共通のパラメータとしてノード数と指定経過時間も使用した。

前述のパラメータ毎に分類した結果を表 3 に示す。パラメータが増えるにつれて分類数は増加し、ノード数のみの場合で 2,000 ケース前後まで、ノード数と経過時間の場合で 4,000 ケース以上まで増えていることがわかる。

図 6 に各条件でのケース毎の標準偏差を示す。ケースは標準偏差の小さい順にソートしている。各グラフの前半のケース (グラフの左側) における累計割合の増加率が低いケースは該当するジョブが少なく標準偏差が小さくばらつきがないことを表している。分類条件が増えることでこの部分も増加していることがわかる。各ケースに分類されるジョブ数が 10 以上の場合に注目してみると、アプリケーション種別を利用した場合の標準偏差が他の場合と比較して、大きくなっていることがわかる。一方、グループ ID とユーザ ID についてはほぼ同じ結果となっている。「京」の場合、課題単位でグループを割り当てていて、同一グループ内では同じ課題に取り組むため、実行されるジョブの内容も類似するためであると考えている。アプリケーション

表 3 分類結果

分類条件		分類前	ノード数のみ (全て)	ノード数と経過時間 (全て/10 ジョブ以上)
アプリケーション種別	ケース数	313	1,854	4,273 / 653
	ジョブ数	42,745	42,745	42,745 / 34,149
グループ ID	ケース数	169	1,879	4,340 / 673
	ジョブ数	42,745	42,745	42,745 / 33,949
ユーザ ID	ケース数	577	2,203	4,561 / 684
	ジョブ数	42,745	42,745	42,745 / 33,557

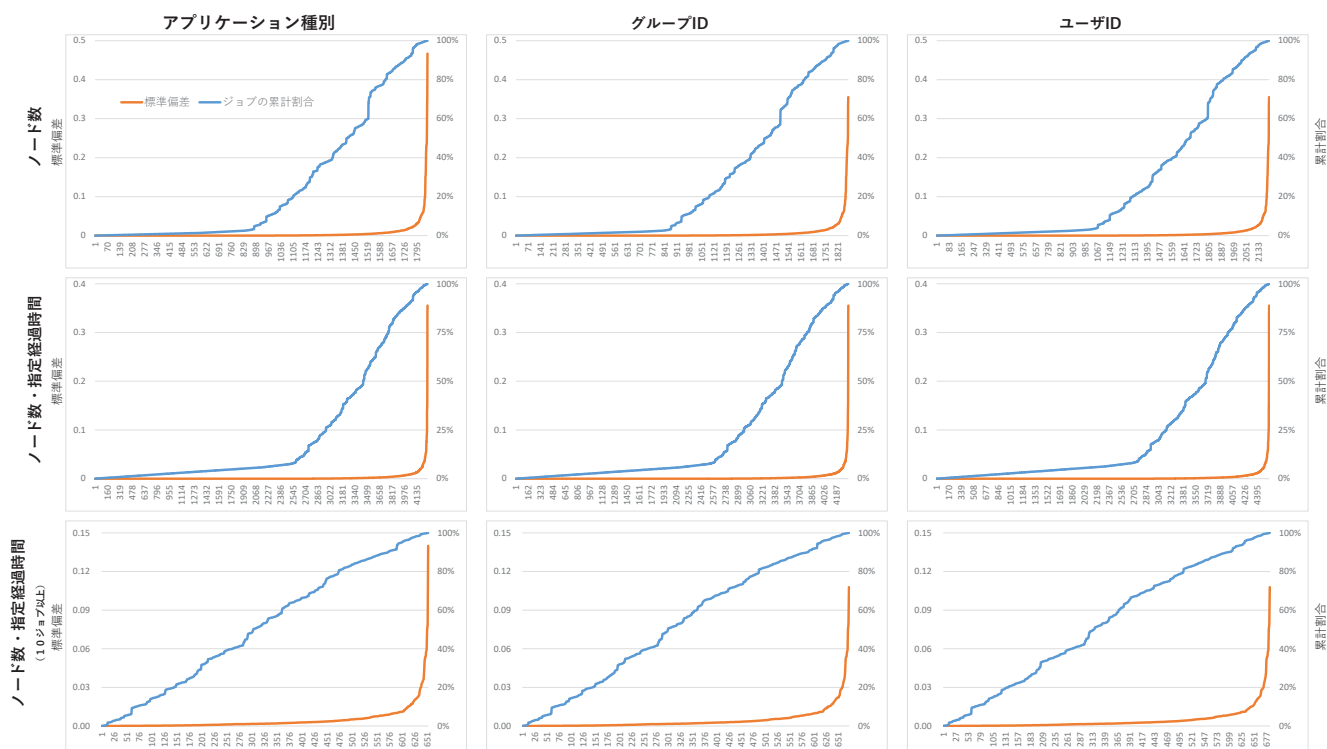


図 6 各条件におけるケース毎の標準偏差

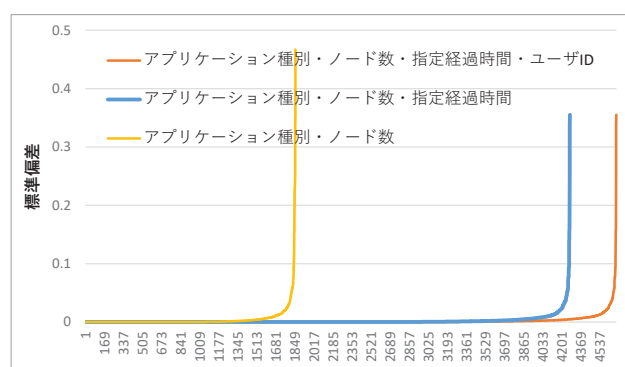


図 7 パラメータを増やした場合の標準偏差の変化

種別の場合、複数の課題で利用され課題毎に実行条件が大きく異なることから標準偏差が大きくなっていると考えられる。

図 7 にパラメータを増やした場合の標準偏差の変化を示す。パラメータの増加につれてケース数は増加し全体的に標準偏差は小さくなるのがわかる。アプリケーション種別+ノード数+指定経過時間の場合と、アプリケーション

種別+ノード数+指定経過時間+ユーザ ID の場合を比較すると、後者の方がケース数が増加し精度は向上しているが、標準偏差の最大値はほぼ同じ結果となっている。そこで、標準偏差が悪化しているケースについて調査を行った。

図 8 の分類前に、標準偏差が大きいケースの分類された各ジョブの消費電力データを示す。同一ケースでも時系列パターンが大きく異なるデータが含まれていることがわかる。そこで、これらの消費電力データの分類を試みた。同一ケースに分類された消費電力データのユークリッド距離を元に分類を行った。ここでは閾値 $\delta = 0.001$ で分類した結果、パターンに再分類された (図 8 の分類結果 (1)–(5))。

図 8 の分類結果 (2) や (5) をみると、消費電力がマイナスになっているデータが含まれていることがわかる。「京」では、全てのノードに電力計が設置されていないため、温度情報を元に消費電力を推定している [8]。この推定方法では、CPU 温度と CPU が搭載されているシステムボード (SB) の排気温度から消費電力を推定する。

$$P = a \cdot T_{cpu} \times b \cdot T_{air} + c$$

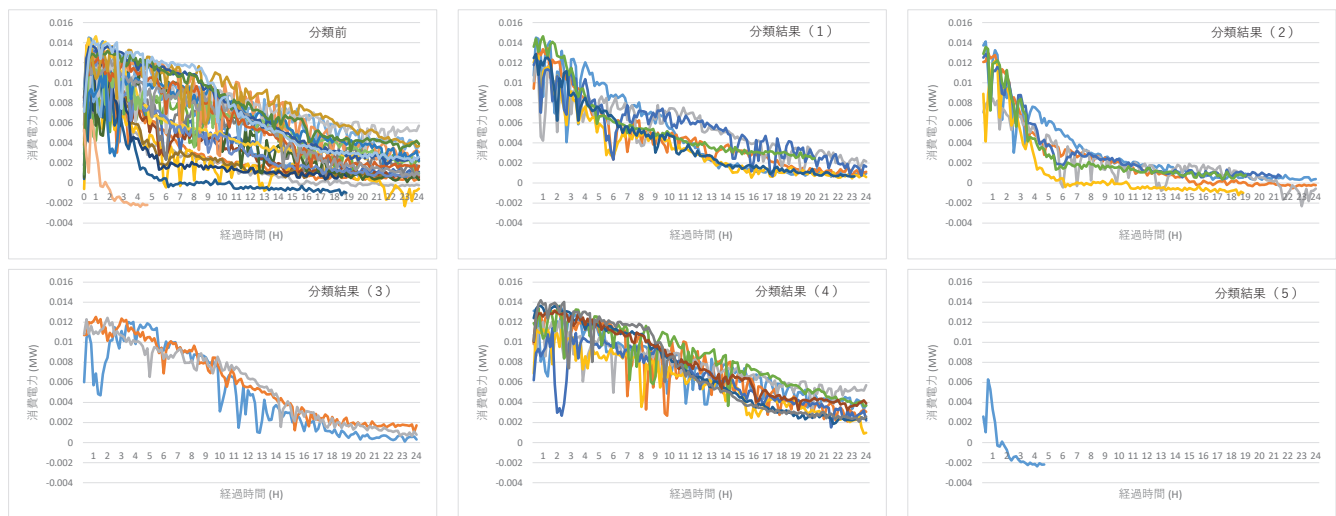


図 8 消費電力の再分類結果

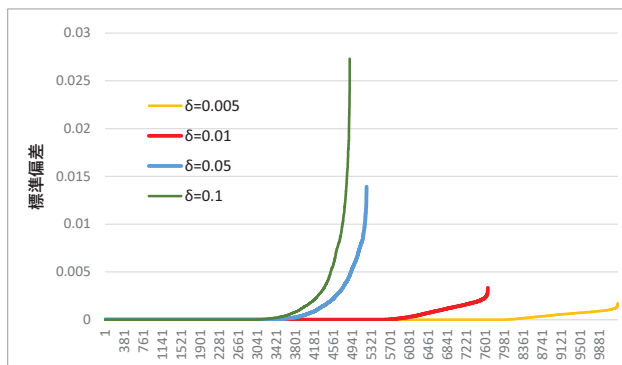


図 9 消費電力の再分類を実施した場合の標準偏差

P がシステム全体の消費電力を、 T_{cpu} は平均 CPU 温度変化を、 T_{air} は平均 SB 排気温度変化をそれぞれ表す。係数 a, b, c は、計測されたシステム全体の消費電力を元に求めた値を使用しており、計算ノード毎の係数は使用していない。今回使用した係数は以下の通りである。

$$a = 0.802393382361262$$

$$b = 0.345223838880426$$

$$c = 7.67202252302052$$

今回の評価に用いたジョブの多くは一部の計算ノードを使用しており、計算ノード毎の個体差が推定値に大きく影響しているものと推測される。精度を上げるためには、計算ノード毎に係数を調整する必要があると考える。

図 9 に閾値を変化させて消費電力の再分類を実施した場合の標準偏差を示す。再分類により標準偏差が大幅に改善できることがわかるが、これは入力パラメータによる分類結果にさらに複数のケースが含まれていることを示しており、ユーザ ID やノード数、指定経過時間等のパラメータだけを用了分類では不十分であることを同時に示している。

4.2 機械学習による消費電力推定

機械学習を用いて、予測に使用するパラメータが消費電力の予測精度にどのように影響があるか評価を行った。今回の評価では問題を簡単にするため、ジョブの消費電力は時系列データではなくジョブの実行開始から終了までの電力の 1 ノードあたりの平均値を用いた。

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、ノード数等を使用し、出力パラメータを平均消費電力とした。この評価では入力パラメータを変えることで予測値がどのように変化するか評価した。具体的には、入力パラメータの組み合わせ毎に予測値を求めその予測精度を比較した。機械学習のアルゴリズムには Random Forest[9]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。ジョブと電力のデータセットを 5 分割し、うち 1 つをテストデータ、残り 4 つを学習データとし、それをテストデータを変えて 5 回評価する交差検定を行った。評価指標には、電力の予測値と実測値との差の 2 乗の平均である平均平方誤差を用いた。

評価結果を表 4 に示す。入力パラメータとして、ユーザ ID とアプリケーション種別それぞれについて入力パラメータを増やした場合の平均平方誤差を求めた。ユーザ ID とアプリケーション種別を比較すると、全ての場合でユーザ ID の方の誤差が小さい。また、どちらの場合でも (ノード数、指定経過時間) と入力パラメータを増やすごとに予測誤差が少なくなることがわかる。これらの結果は入力パラメータに基づく分類による推定結果と一致する。また、入力パラメータとしてユーザ ID とアプリケーション種別、ノード数、指定経過時間を利用した場合が最も誤差が小さくなった。「京」の場合、前述のように課題単位では同じ種類のジョブを実行する傾向があることがわかっているが、

表 4 機械学習による評価結果

入力パラメータ	平均平方誤差
ユーザ ID	28.54
アプリケーション種別	34.56
ユーザ ID+ノード数	17.00
アプリケーション種別+ノード数	17.95
ユーザ ID+指定経過時間+ノード数	14.73
アプリケーション種別+指定経過時間+ノード数	15.96
アプリケーション種別+ユーザ ID+指定経過時間+ノード数	13.26

必ず同じ種類のジョブが実行されるとは限らない。そのため、ユーザ情報のみの場合はそれを区別することができない。ユーザ情報に加えてアプリケーション情報を用いることで区別することが可能となり、精度が向上したと考えられる。

入力パラメータに基づく分類による推定では、評価に用いた入力情報だけでは不足するという結果となったが、ジョブスクリプトの違いを入力パラメータとして機械学習を用いることで、改善することが可能になると考えている。

5. おわりに

本稿では、投入されたジョブの消費電力を予測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ために、ジョブの時系列電力変動の推定手法について検討を行った。

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予測する。「京」で実行されたジョブ情報を用い消費電力を予測したところ、アプリケーション種別やユーザ ID、ノード数、指定経過時間などのパラメータを使用することで消費電力を推定することが可能であることを確認したが、場合によってはばらつきが大きい場合があることがわかった。推定精度を向上させるためには、ジョブスクリプトの特徴を利用するなどの手法を検討する必要があると考えている。

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, Bronis 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ニ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: "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)
- [7] Quoc Le and Tomas Mikolov: "Distributed representations of sentences and documents," In Proceedings of the 31st International Conference on Machine Learning (ICML 2014), pp.1188-1196 (2014)
- [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 聡, 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] Breiman, Leo: "Random Forests," Mach. Learn., Vol.45, No.1, pp.5-32 (2001)