Vol.2017-HPC-161 No.5
2017/9/19

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

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．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 CPU の周波数等を制御してシステムの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が提案されているが，ジョブの実行効率の点から見た場合，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できることが望ましい．今回，ジョブ毎の消費電力の変動を考慮してジョブスケジューリングを行う手法について検討を行なった．本手法では，ジョブ投入時の情報からジョブの実行前にそのジョブの消費電力の時系列変動を予測する．本稿では，「京」で実行されたジョブのデータを用いた評価結果について報告する．

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．例えば理化学研究所が運用しているスーパーコンピュータ「京」の場合 [1]，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きたが，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きている．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，電力は重要な問題となっており効率のよいシステム電力の運用が必要とされている [2].

近年，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システム電力を最大限に活用する方式が提案されている．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を管理する．一方，「京」の場合，ジョブ実行時の CPU などの電力消費をコントロールする機構を持たないため，システム電力が上限値を超えないようにジョブの実行を制御する方式を採用している [3]．システムの消費電力を上限内に抑えるという点ではどちらも同じであるが，後者の方式ではシステム電力を最大限に活用することは難しい．ジョブの実行効率の観点から見た場合，前者の 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ようにジョブをスケジューリングできることが望ましい.

そこで，ジョブへの制約をできるだけ少なくし，かつ，システム電力を最大限に活用するために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の検討を行った．本手法では，ジョブ毎の時系列消費電力変動をジョブ投入時に予測し，システム電力を最大限に活用できるようなスケジューリングを実現する．

2. 関連研究

システム電力を最大限に活用する方式として，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研究が行われている．多くのシステムでは，平常運転時の消費電力はピークの半分程度と言われており，システムの消費電力を施設（電源系や空調系）の能力を超えるように設置し，ソフトウェアやハードウェアの制御により能力限界の範囲内で運用できるようにすることで効率的に施設を利用することが可能となる．しかし，ジョブの実行効率を最適化するためにはジョブの電力特性をふまえたジョブスケジューリングが必要となる．Patki[4] らはアプリケーションの実行効率とシステム電力効率を改善する手法として RMAP という資源管理手法を提案している．Power-aware backfilling を実装し，いくつかのスケジューリングポリシーについて評価を行っている．ここでは，スケジューリング時のシステム電力を最大限に活用できるようにジョブ毎の電力制御を行っている．Ellsworth[5] らは，SLURM のプラグイン機能を利用した Power-aware scheduling のためのフレームワークを提案している．RMAP や動的に電力制

[1] 国立研究開発法人理化学研究所 計算科学研究機構
[2] 株式会社富士通ソーシアルサイエンスラボラトリ
[3] 富士通株式会社
[a)] uno@riken.jp

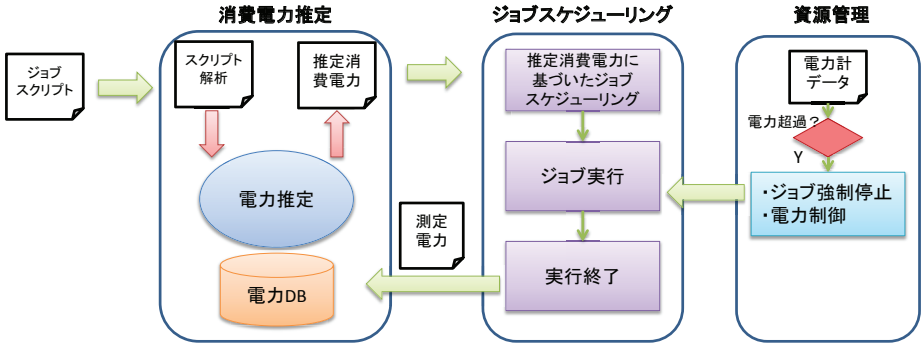

図 1 提案フレームワーク

御を行う PowSched[6] をプラグインとして実装し評価をおこなっている．坂本 [7] らはオーバープロビジョニング構成を想定した大規模な HPC システム上で電力制約を考慮したリソースマネージャを用いた電力を性能の評価を行っている．

消費電力予測の研究としては，投入されたジョブの要求リソースから消費電力を予測し，スケジューリングを行うことでシステムの電力効率を改善する研究が行われている [8] ．この研究では，電力予測はジョブ全体で行われており，時系列の電力変動は考慮されていない．

本稿で提案する手法は，システム電力を最大限に活用しつつジョブの実行効率を最適化できるように，ジョブの消費電力の時系列変動を予測しスケジューリングを実現するものである．

3. 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の概要について説明する．

本手法は，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実現する．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．**図 1** に提案するフレームワークを示す．このフレームワークは大きく分けて 3 つのモジュールで構成される．消費電力推定，ジョブスケジューリング，資源管理である．それぞれの概要について以下に説明する．

3.1 消費電力推定モジュール

消費電力推定モジュールでは，ジョブの実行結果に基づいた電力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．

これまでに，ジョブの過去の実行実績を利用したジョブの消費電力予測をおこなってきた [10]．この予測では対象となるユーザが過去に実行したジョブの情報をもとにジョブが使用するノード数をもとに電力推定を行っていたため，あまり精度は高くなく，ジョブ単位での電力情報の推測しかできなかった．今回の手法では，ジョブで実行されるアプリケーションとその消費電力の時系列データの推測を行う．

ユーザのジョブ投入時点では，以下のような情報を取得することができる．

ジョブパラメータ
ユーザ名・グループ名・ノード数・指定経過時間などのジョブ実行のために指定されるパラメータ．

ジョブスクリプト
ジョブで実行する処理内容．ジョブパラメータがジョブスクリプトに記述されている場合もある．

実行実績
過去に実行したジョブのパラメータやスクリプトおよび実行時間や電力や性能などの情報．

これらの情報をジョブ投入時情報と呼ぶこととし，ジョブ投入時情報からそのジョブで実行されるアプリケーションおよび消費電力を推定する．

3.1.1 電力データベースの構築

ジョブ実行時に得られるデータをもとに消費電力推定に用いる電力データベースを構築する．

一般的にジョブでは何らかのアプリケーションが実行されるため，ジョブの実行で得られる情報からアプリケーションの推定を行う．アプリケーションの分類を行うためにはアプリケーションの特徴を収集する必要があるが，「京」では MPI プログラムの実行時にロードモジュール（LM）のハッシュ値，ファイル内の RCS キーワード（ident 情報），シンボル情報（nm 情報）を MPI プログラム実行コマンド (mpiexec) の wrapper を利用して収集している．これらをもとに LM がどのような言語（C や Fortran）で記述されているか，どのバージョンのコンパイラが使用されたのかといった情報も取得することができる．例えば，LM のハッシュ値を利用することで同一の LM を実行しているかどうか判断することができる．LM にはコンパイル

時の一時ファイル等の情報が埋め込まれている場合があるため，同一機能の LM であってもハッシュ値が異なる場合がある．そのため，ident 情報や nm 情報も LM の特定に活用する．

以上の手法により特定した LM とジョブ実行時に計測された電力消費データをひも付け，電力データベースに登録する．

3.1.2 ジョブの消費電力推定

ジョブの消費電力推定では，ジョブ投入時情報と電力データベース登録時に推定された LM 情報を元にそのジョブで実行される LM を推定する．

ジョブ投入時情報の中で，ジョブの特徴情報を最も含んでいるのはジョブスクリプトである．「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），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．そこで，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を分類することにした．

電力データベース登録時の LM の分類に基づき，ジョブスクリプトを入力とし，分類結果を出力とする機械学習モデルを作成した．前処理として，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する．この処理では，スクリプトのコメント行を示す#から始まる行もそのままシンボル化している．このシンボル集合と LM のクラスを doc2vec アルゴリズム [11] を用いて学習させる．この機械学習により，ユーザが投入したジョブの LM を推測し，電力データベースからそのジョブの消費電力を推定する．

3.2 ジョブスケジューリングモジュール

ジョブスケジューリングモジュールでは，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．例えば，システム電力の上限を超えないようにするには，各計算ノードに一律にパワーキャップを設けることで比較的容易にシステム全体の消費電力を制御することが可能となる．しかし，電力消費の低いジョブと高いジョブが混在するような場合では，システム全体の消費電力に余裕があっても計算ノード毎のパワーキャップにより電力消費の高いジョブの実行に制限がかかることになる．これを防ぐには，システム全体で電力上限をこえないようにジョブ単位の消費電力に基づいてスケジューリングできればよい．ジョブの消費電力は時系列で大きく変化することがわかっており [9]，システム電力を効率的に利用するには時系列の電力変動の考慮が必要となる．

電力データベースに登録されている消費電力情報は時系列データだが，各データの間隔は電力情報のサンプリングレートに基づいている（「京」の場合は 5 分間隔）．サンプリング間隔を保持したままスケジューリングを行うと非常に複雑になるため，一定間隔の時系列データに変換することを考える．**図 2** にスケジューリング用時系列データの作成方法を示す．ここでは，同一 LM に分類された 2 ジョブの消費電力の時系列データから，区間毎の最大値と平均値を求めており，緑の枠内の計測値が各区間での計算対象である．

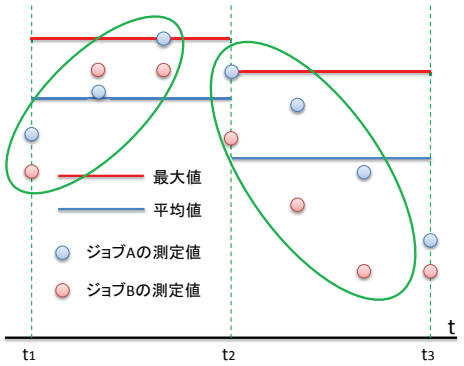

図 2 スケジューリング用時系列データの作成方法

3.3 資源管理モジュール

資源管理モジュールでは，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．制御方法としては，

- RAPL などのハードウェアが持つ機能を用いて消費電力を制御
- 超過しないように実行中のジョブを強制停止

などが考えられる．

近年のシステムでは電力制御用のハードウェアが実装されていることが多く，それらを活用した資源管理制御に関する研究が多く行われている．一方，「京」の場合，DVFS (Dynamic Voltage Frequency Scaling) のような電力制御機能はなく，30 分間の使用電力が上限を超えないように制御できればよいことから，後者の超過が見込まれる場合にジョブを停止させるという制御を行っている．具体的には，「京」の消費電力を常時監視し，電力超過の兆候や超過が発生した時点で，ジョブを停止したことにより失われるノード時間積が最小になるように，実行中のジョブを強制停止させている [3]．

4. 評価

本稿で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手法の事前評価として，「京」で実行されたジョブのデータを用い，

(1) 実行後のジョブの LM 推定
(2) 投入されたジョブの LM 推定

について評価を行った．

表 1 分類手法によるクラス数

クラス分け手法	クラス数
ロードモジュール情報取得総数	363,573
ハッシュ値の一致	41,817
シンボル集合の一致	5,923
シンボル集合の類似	592

4.1 実行後のジョブの LM 推定

前述の LM の分類手法に基づき，「京」で実行されたジョブの LM の推定とクラス分けを行った．

表 1 に 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で，MPI の実行時に LM 情報が採取できた LM をクラス分けした結果を示す．対象期間中に MPI プログラムは約 36 万回実行され，約 36 万件分の LM 情報を採取した．1 ジョブで複数の LM を実行する場合もあるため，ここではジョブ数ではなく LM 数でカウントしている．

表中の**ハッシュ値の一致**は，約 36 万のハッシュ値を完全一致でクラス分けした際のクラス数，つまりユニークなハッシュの数を示しており，約 4.2 万個の LM が約 36 万回実行されたことを意味している．このことから，ユーザは完全に同じ LM を入力パラメータを変えるなどして繰り返し実行していると推測することができる．

シンボル集合の一致は，ident 情報や nm 情報から抽出したシンボル集合を完全一致でクラス分けした際のクラス数，つまりユニークなシンボル集合の数を示している．シンボル集合は，同じソースコードであればコンパイラのバージョンの変更や，ソースコードの軽微な修正では変わらないことが多い．そのため，別々のユーザが同じソースコードから LM を作成した場合でも，ハッシュ値は異なるがシンボル集合は一致するので同じ LM に分類することができる．また，プログラムを修正せずに再コンパイルを行いハッシュ値が異なる LM が生成されたような場合でも，シンボル集合は同じため同じ LM に分類することができる．ただし，ここでは集合の一致でクラス分けを行っているため，例えば main 関数しかないようなシンボル集合の少ない LM は，main 関数の処理内容が違なる LM であっても同じクラスに分類されてしまう．このようなプログラムはテスト目的で記述されていることが多く，実行回数も少ないと思われるので，分類できなくても問題は小さいと考えている．

シンボル集合の類似は，シンボル集合を類似度でクラス分けした際のクラス数を示している．シンボル集合の完全一致による分類では，関数を一つ追加した程度の修正でも別クラスに分類されてしまう．これを防ぐため，集合の類似度でクラス分けを行う．これにより，集合全体が似ていれば同じクラスと分類することができる．ここではシンボル集合の類似度を計算するために，シンボル集合を Latent Dirichlet Allocation (LDA) アルゴリズム [12] によってベクトル化し，ベクトル同士のコサイン類似度をもとにクラス分けを行っている．

表 2 ジョブスクリプトからの LM クラスの推定結果

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%

図 3 に同じクラスとみなすコサイン類似度の閾値とクラス数の変化を示す．閾値が低くなるほど，類似度が低くても同じクラスに分類される確率が高くなるため，クラス数は減少する．ここでは最終的にコサイン類似度が 0.99 以上であれば同じクラスとみなし，結果，約 36 万の LM は 592 クラスに分類された．

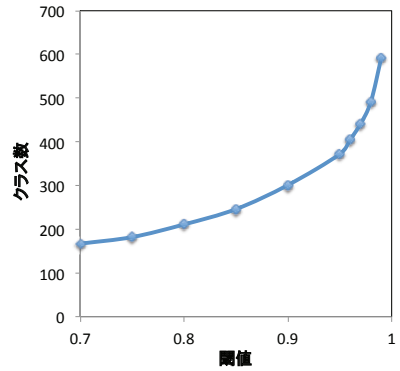

図 3 類似度の閾値とクラス数の変化

この結果に基づいて分類した LM クラス別の消費電力の時系列データを**図 4** に示す．

分類上同じ LM クラスでもノード数と実経過時間が異なる場合があるため，同一のノード数で実経過時間がほぼ同じジョブに細分類し，各 LM クラス毎に 10 件のジョブの消費電力をプロットしている．図 4 から，LM クラスの消費電力の変動は類似していることがわかる．

図 5 に，図 4 をもとに作成したスケジューリング用時系列データを示す．ここではスケジューリング間隔を 30 分として以下の値を求めている．

(1) 最大値
(2) 平均値
(3) 平均値 $+3\sigma$

どの値をスケジューリングで使用するかで電力マージンをコントロールすることができる．推定消費電力を用いている以上，確実にシステムの消費電力を上限値以下に制御することは困難であるが，資源管理モジュールで制御する電力変動を小さくすることは可能である．

4.2 投入されたジョブの LM 推定

592 クラスに分類した LM のデータを用いて，ジョブ投入時情報からそのジョブの LM クラスの推定を行った．

表 2 に投入時情報から LM クラスを推定した結果を示

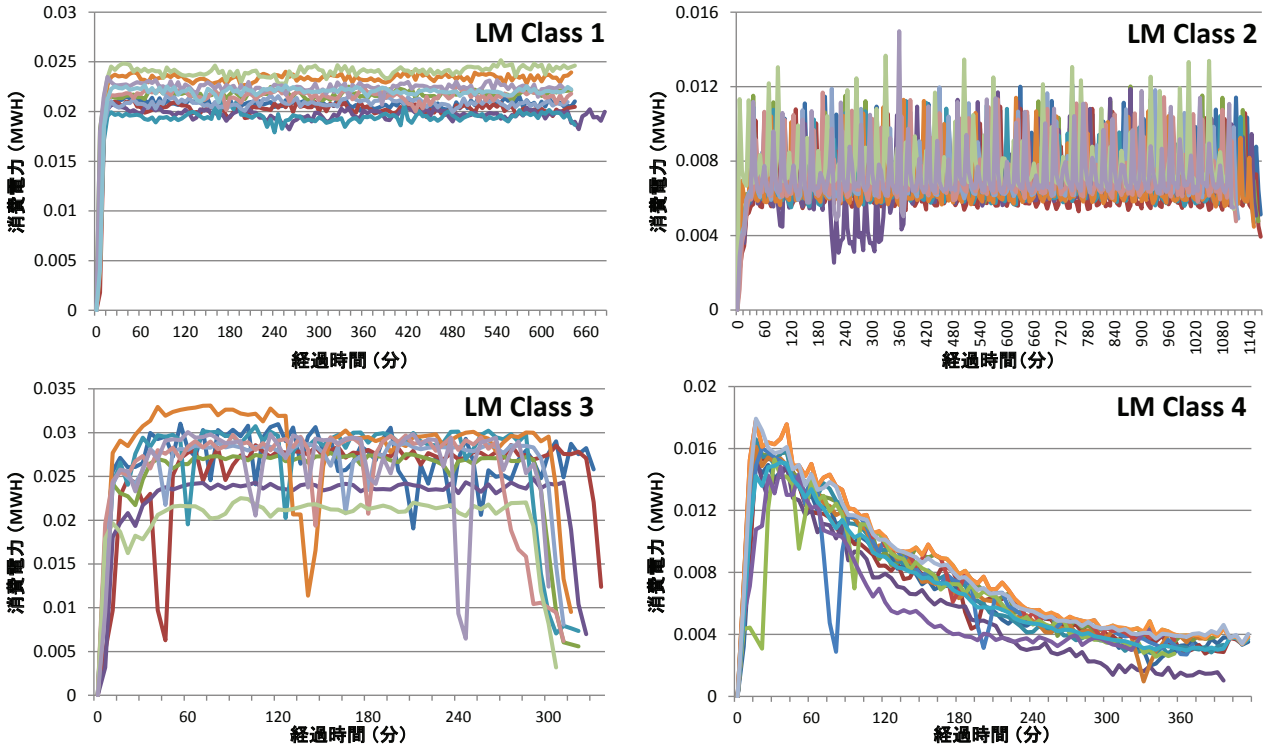

図 4 分類した LM クラス別の消費電力変動

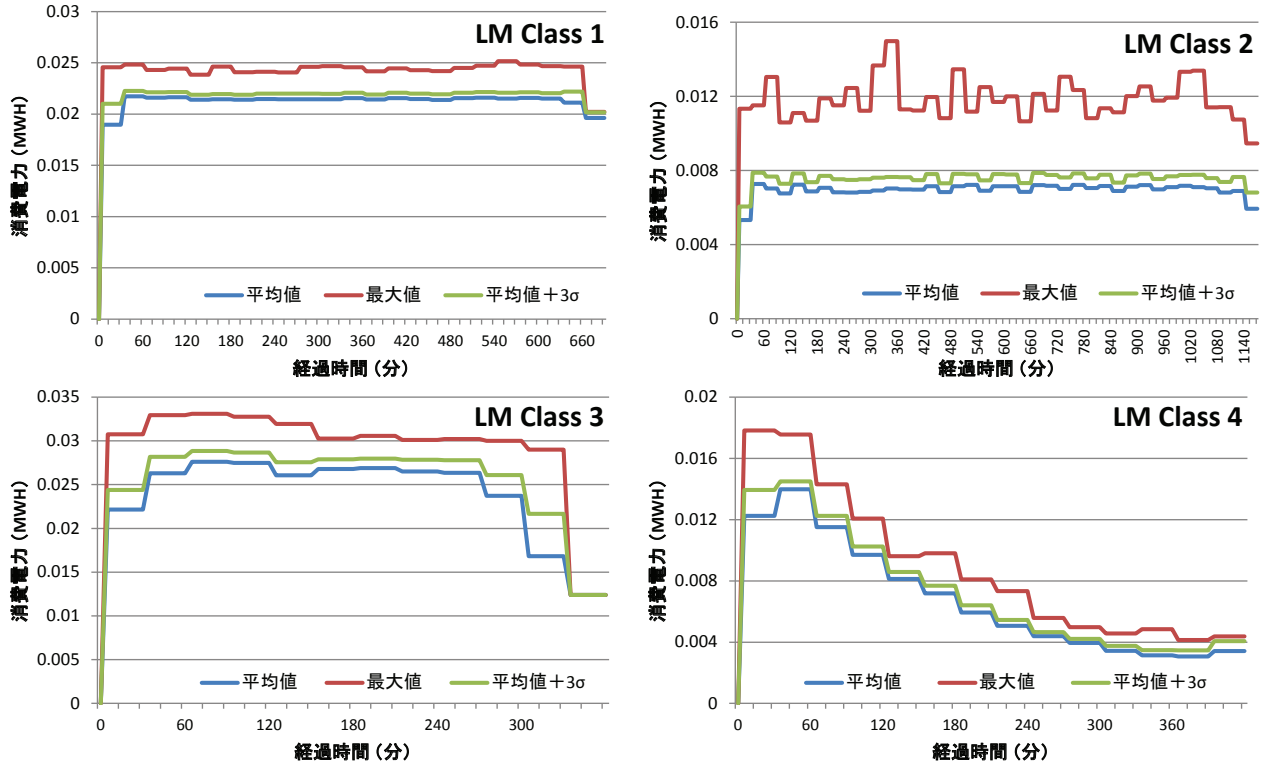

図 5 スケジューリングに用いる LM 毎の消費電力時系列データ

す．ジョブスクリプトには 2 つ以上の LM が含まれる場合もあるが，ここではジョブスクリプトに含まれる LM が 1 つの場合のみ評価対象とした．

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を使って LM 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使ってモデルを作成し，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使って評価した結果である．

Model:2016 では，2016 年に実行されず 2017 年に初めて実行されたジョブはモデルに考慮されていないため推定結果がすべて誤りとなり，正解率は 83%であった．一方，**Model:all** ではジョブの時系列は考慮されておらず，doc2vec アルゴリズムでジョブスクリプトから LM をどの程度推定できるかを評価した結果となる．その正解率は 91%であった．

実際の運用では，新規の LM に対応するため日々推定モデルを更新し続ける必要がある．これは **Model:all** とほぼ同じ条件であり，このような運用を行なった場合は正解率は約 9 割程度になると予想される．また，現在の手法ではスクリプトの前処理には改善の余地があることがわかっており（分かち書きやコメント行の考慮など），シンボルの抽出方法やモデルのパラメータ最適化などは今後の課題である．

5. おわりに

本稿では，ジョブへの制約をできるだけ少なくするとともに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実現する手法について検討を行った．

本手法では，投入されたジョブスクリプトからそのジョブで実行されるアプリケーションを推測し，ジョブの消費電力の時系列変動を予測する．「京」で実行されたジョブ情報を用いアプリケーションの推定を行ったところ，推定モデルを適宜更新することで 9 割程度の精度でアプリケーションを特定することができ，消費電力の時系列データを予測することが可能であることがわかった．今後は，推定手法の改良および実際に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] 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
[2] 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).
[3] 井上 文雄，宇野 篤也，塚本 俊之，松下 聡，末安 史親，池田 直樹，肥田 元，庄司 文由：消費電力の上限を考慮した「京」の運用，情報処理学会研究報告, Vol.2014-HPC-146, No.4 (2014).
[4] 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).
[5] Daniel Ellsworth, Tapasya Patki, Martin Schulz, Barry Rountree, Allen Malony: “A Unified Platform for Exploring Power Management Strategies,” 2016 4th International Workshop on Energy Efficient Supercomputing (E2SC’16), pp.24–30 (2016).
[6] D. A. Ellsworth, A. D. Malony, B. Rountree, and M. Schulz: “POW: System-wide Dynamic Reallocation of Limited Power in HPC,” Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing, ser. HPDC ’15. New York, NY, USA: ACM, pp.145–148, (2015).
[7] 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).
[8] Borghesi A., Bartolini A., Lombardi M., Milano M., Benini L.: “Predictive Modeling for Job Power Consumption in HPC Systems”, ISC High Performance, LNCS Vol. 9697, pp.181–199 (2016).
[9] 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).
[10] 山本 啓二，末安 史親，宇野 篤也，塚本 俊之，肥田 元，池田 直樹，庄司 文由：過去の実行実績を利用したジョブの消費電力予測，情報処理学会研究報告, Vol.2015-HPC-151, No.2 (2015).
[11] 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).
[12] Matthew Hoffman and Francis R. Bach and David M. Blei: “Online Learning for Latent Dirichlet Allocation,” Advances in Neural Information Processing Systems 23, pp.856–864 (2010).