Vol.2015-HPC-151 No.2
2015/9/30

過去の実行実績を利用したジョブの消費電力予測

山本 啓二[1,a)]　末安 史親[2]　宇野 篤也[1]　塚本 俊之[1]　肥田 元[3]　池田 直樹[3]　庄司 文由[1]

概要：近年、高並列システムでは消費電力が重要な制約となっており、電力制約を考慮したジョブスケジューリングの研究が行われている。本稿では、このようなジョブスケジューリングを実現するために、バッチジョブの実行前にそのジョブの消費電力を予測する手法を提案する。提案手法は、既実行ジョブを電力情報とともに実行実績として蓄積し分析することで、これから実行されるジョブの電力を推定するものである。「京」で実行されたジョブの実行実績を用いて本手法を評価した結果を報告する。

1. はじめに

スーパーコンピュータや大規模なクラスタシステムでは，近年，消費電力が重要な制約となっている．スーパーコンピュータ「京」(以下,「京」) は，理化学研究所と富士通株式会社が共同開発した，生命科学・医療，エネルギー，防災・減災，次世代ものづくり，物質と宇宙といった様々な分野のアプリケーションを高速に処理できる汎用性の高いスーパーコンピュータで，2012 年 9 月に共用を開始して以来，概ね安定して運用している [1], [2]．低消費電力の CPU を採用するなどにより消費電力を抑えてはいるものの，規模が大きいためにシステム全体の消費電力は 10MW を越えており，運用コストに対する電力コストの割合は非常に大きい．

一般に計算機の消費電力は実行されるジョブにより変動するが，特に「京」では規模が大きいため消費電力の変動が非常に大きい．共用開始当初は，多くのジョブについてチューニングがさほど進んでいなかったことや，規模が大きくなかったことから大規模なベンチマーク等の特殊なジョブを除き，消費電力が問題になることはなかった．しかし，共用開始から 1 年が経過した頃から消費電力が大きく変動し，契約電力の上限を越える状況が時折発生するようになった．このような電力超過は運用への影響が大きいため，どのように電力の消費をコントロールするかが運用上の課題となってきた．

「京」では契約電力の超過対策として大規模ジョブ投入前の事前審査およびジョブ緊急停止の 2 つの取り組みを始めた [3]．事前審査ではユーザの申請に基づき，過去に投入した同種のジョブの電力から大規模ジョブとして投入可能な実行可能ノード数を求め，そのノード数までのジョブ実行を許可する制度である．ジョブ緊急停止では常時「京」を含む施設全体の電力を監視し，契約電力を越えそうな場合には実行中のジョブを強制的に停止し，契約電力超過を回避するという取り組みである．停止するジョブは，実行中のジョブのうちノード数の大きいものから選ばれ，順次電力が低下するまでジョブを停止する．

我々は，これらの取り組みに加えて，ジョブスケジューリングの段階で今後の電力がどのように推移するかを予測し，電力超過に備えることを考えている．これを実現するには、まず個々のジョブの電力をジョブの実行前に予測できることが必要となる．本稿では，ジョブ実行前にそのジョブの電力を予測する手法および本手法を用いてジョブの電力を予測した結果について報告する．

2. 「京」の概要

「京」は，82,944 台の計算ノードと 11PB のローカルファイルシステム，30PB のグローバルファイルシステム，フロントエンドサーバなどの周辺機器から構成される．**図 1** に「京」のシステム構成の概要を示す．

「京」の運用に必要な電力は商用電力と自家発電により供給されている．自家発電の設備として定格出力 5MW のコジェネレーションシステムを 2 台備え，通常は 1 台ずつ交互に運転している．共用開始時は「京」の無負荷時の消費電力 (「京」本体とローカルファイルシステムの消費電力) として 10MW，その他施設の電力として 3MW，ジョブ実行による変動分を 4MW と想定し，合計 17MW を電力供給の上限値とした．よって，コジェネレーションシス

[1] 理化学研究所計算科学研究機構
RIKEN Advanced Institute for Computational Science
[2] 富士通株式会社
Fujitsu Limited
[3] 株式会社富士通ソーシアルサイエンスラボラトリ
Fujitsu Social Science Laboratory Limited
[a)] keiji.yamamoto@riken.jp

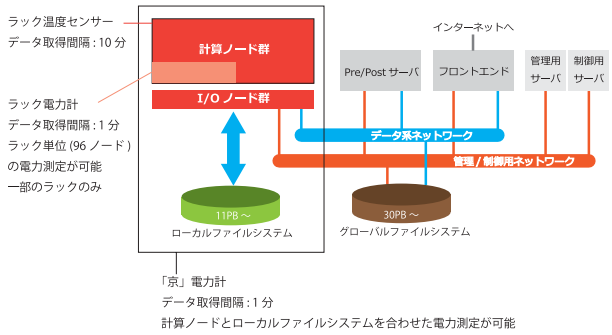

図 1 「京」のシステム構成

テムによる発電電力の 5MW を除いた 12MW を電力会社と契約した．しかし，後述の電力超過が度々発生したことにより現在は 12.75MW の契約となっている．

2.1 通常運用と大規模ジョブ実行

「京」では通常の運用時では，36,864 ノード以下の小中規模ジョブの実行が可能である．36,865 ノード以上のジョブは，原則として毎月第 2 火曜日からの 3 日間に設けた大規模ジョブ実行期間に実行する．大規模ジョブを投入するためには事前審査を経なければならない．事前審査は契約電力超過を未然に防ぐための措置で，ユーザは審査で許可したノード数までのジョブを投入することが可能である．

2.2 電力超過対策

共用開始時は電力供給の上限値を 17MW と想定し，電力会社との契約を行っていた．しかし，2013 年度の大規模ジョブ実行期間中に 3 度，契約電力の上限を越える電力超過を起こした．契約電力超過は次年度の契約電力の見直しにつながり，「京」の運用コストが増大することによる影響は非常に大きい．実際に 2013 年度の電力超過の影響により，2014 年度からは契約電力が 0.75MW 増の 12.75MW となった．

そこで電力超過を抑止する対策として，大規模ジョブ投入前の事前審査およびジョブ緊急停止の 2 つの取り組みを始めた．事前審査ではまず大規模ジョブとして実行したいジョブをユーザに少なくとも 10,000 ノード，15 分程度実行してもらい，その期間の「京」の電力変動からジョブの電力を見積もる．次に，この見積もった電力から 1 ノードあたりの電力を計算し，許容電力 (4MW) 内での実行可能ノード数を求め，そのノード数以下でのジョブ投入を許可する．

ジョブ緊急停止とは，ジョブ実行時に電力が許容範囲を越えた場合に実行中のジョブを強制的に停止する仕組みである．複数のジョブが実行中の時はノード数の大きいジョブから順次，電力が許容範囲に収まるまで停止する．これは，現在「京」では実行中の個々のジョブの電力が取得できないため，ノード数の大きいジョブを電力を消費するジョブとみなしているためである．我々は現在，温度センサーを利用してジョブの電力を推定する取り組みを実施しており，今後はジョブ電力を考慮しジョブを停止する予定である．これら 2 つの電力超過対策によって契約電力の超過は現在発生していない．

2.3 「京」での電力測定

図 1 に示すように「京」の電力を測定する設備として，「京」電力計とラック電力計の 2 種類が備わっている．「京」電力計は計算ノードとローカルファイルシステムを合わせた電力を計測することができる．ラック電力計は計算ノードが 96 台搭載されている 1 ラック単位での電力を計測することができる．ラック電力計は一部のラックにのみ備わっている．

「京」電力計はシステム全体の電力を計測するもののため，1 つのジョブが「京」を専有するような状況 (大規模ジョブ実行期間) でもないかぎり，ジョブ自体の電力を計測することはできない．また，ラック電力計もラックを専有するようにジョブを投入しないかぎり，ジョブの電力を計測することはできない．

3. 実行実績を利用したジョブ電力推定

一般的なユーザの「京」利用シナリオを考えると，まずユーザは小規模のノードでアプリケーションの開発およびデバックを行い，アプリケーション開発が進につれ中規模でのテストを行い，アプリケーション開発が終わると大規模でプロダクションランに移ると考えられる．プロダクションランでは入力データを変えたり，パラメータを変えたり，時にはアプリケーションそのものの小さな修正を行い実行されるため，実行毎の電力は似たものになると考えられる．開発時からプロダクションランまでアプリケーションが同一であればノード数と電力に相関関係があると想定できる．よって，過去にそのユーザが実行したジョブ電力の実績を元に今後実行するジョブの電力を推定することとする．

3.1 ジョブの実行実績

既に実行したジョブについて，ジョブを投入したユーザ ID, グループ ID およびノード数，指定経過時間，実行時間，実行開始/終了時間，ジョブの形状，ジョブ名，ジョブスクリプト，電力などのジョブの実行に関わるデータをジョブ実行実績として蓄積する．ジョブの電力は「京」では個々のノードに電力計が備わっていないため，簡単に求めることはできないという問題がある．我々は既に図 1 のラック温度センサーの情報を利用したジョブ毎の電力推定に取り組んでおり，温度センサーのサンプリング間隔である 10 分毎にジョブの電力を推定できることがわかっている [4]．次節で温度センサーを用いたジョブの電力推定手

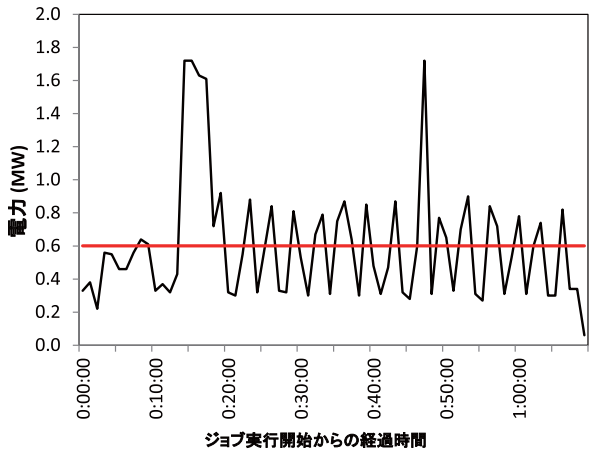

図 2 ジョブの電力変動

法について簡単に述べる．

3.1.1 温度センサーを用いたジョブ電力の推定

「京」では全ラックに対し 10 分毎にラック吸気温度，システムボードの排気温度，水冷入力温度，CPU 温度の情報を採取している．ジョブの実行時に消費される電力はすべて熱になると仮定し，温度センサーから計算した CPU の温度変化とシステムボードの排気温度変化から次の式により消費電力を求める．

$$P = a \cdot T_{cpu} + b \cdot T_{air} + c \tag{1}$$

P は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} は平均システムボード温度変化をそれぞれ表す．係数 a, b, c は標準誤差を最小化するように「京」の温度変化と電力から求める．この式を用いて温度変化からジョブの電力を求める．

3.1.2 平均ジョブ電力

実行されるジョブは時間の推移に従って様々な電力変動をとる．例えば他ノードからの I/O を待っている状況で，CPU 負荷やメモリアクセスが少ない場合は低電力となる．一方で，頻繁にメモリアクセスを繰り返す状況では高電力となる．

図 2 は 1 分単位に採取している「京」電力計の電力で，実行時間が約 70 分，ノード数が 82,944 ノード (フルノード) のジョブ実行時の電力推移を示したものである．「京」でひとつもジョブが動いていない場合の無負荷時電力である 10MW で、この電力を引いた値を図 2 では電力としている．つまり，この電力はジョブを実行したことによる「京」の電力の増加量を示している．図から，14-18 分頃および 48 分頃に約 1.7MW の電力ピークがあることがわかる．また，20-65 分頃までは 3 分毎に高電力 (0.8MW) と低電力 (0.4MW) の周期があり，定形の処理が行われていると推測できる．

現在，温度センサーの値は 10 分毎に採取しているため，そこから計算する推定電力も 10 分毎に求まる．約 70 分のジョブでは 6-7 点で推定電力を求められるが，10 分のサンプリング間隔では図 2 のような細かな電力波形を確認することはできない．本報告ではジョブの時間軸方向の電力変動は考慮せず電力の平均値をジョブの電力とみなすこととする．このジョブの場合，平均電力は 0.6MW である．

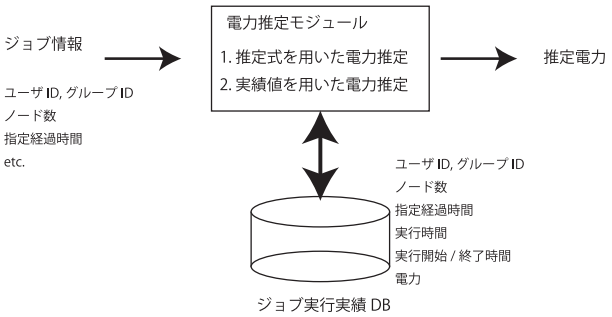

図 3 ジョブ電力推定手法の概要

3.2 電力推定モジュール

ここではユーザがこれから実行するジョブの電力を実行実績を元に推定する手法について述べる．電力を推定するためにジョブの実行前に利用できるジョブ情報は以下である．

- ユーザ ID, グループ ID
- ノード数
- 指定経過時間
- その他，ジョブスクリプト等のジョブ投入時のパラメータ

電力推定の仕組みを**図 3** に示す．電力推定モジュールはジョブ情報を受け取り，そのジョブの推定電力を実行実績を元に計算する．電力推定モジュールは推定式を用いた電力推定方法と実行実績を用いた電力推定方法の 2 種類の電力推定方法を持つ．それぞれの手法について次節以降に述べる．

3.2.1 推定式を用いた電力推定

アプリケーションを実行する場合は実行するノード数に応じて電力も相関をもって変化すると仮定し，以下の電力推定式を作り，今後実行するジョブの電力を予測する．

$$P_{job} = a \cdot N_{node} + b \tag{2}$$

P_{job} はジョブの予想電力，N_{node} は要求ノード数，a, b は係数である．係数は過去の実行実績のノード数と電力の関係から最小 2 乗法で誤差を最小化することで求める．

あるユーザのジョブ実行実績を**図 4** に示す．直線は電力推定式であり，$a = 0.000003, b = 0.0051$ が求まる．図 4 では約 1000,4000,8000 ノードのジョブの場合に推定電力と実際の電力との差が大きくなることがわかる．また約 1,000 ノードジョブでは 0.006MW から 0.029MW と約 4 倍近くの電力差があることがわかる．本手法の利点はユーザ毎に a, b の 2 パラメータのみで電力を推定できることである．

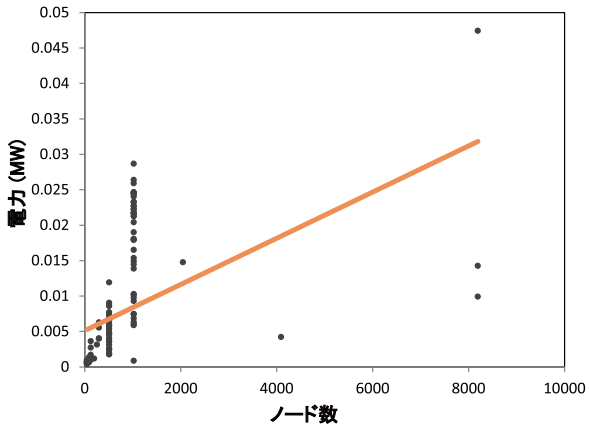

図 4　ノード数と電力の関係

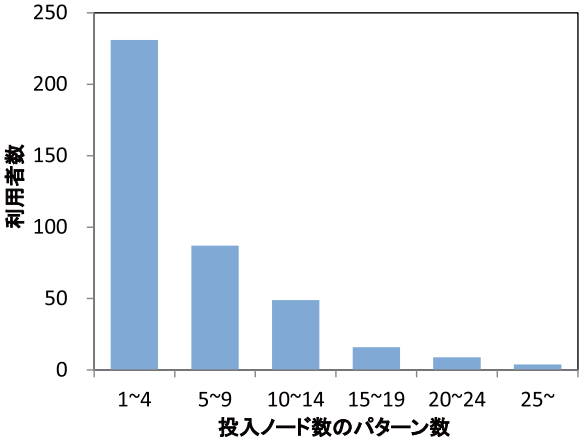

図 5　ノード数のパターン数とユーザ数

3.2.2　実行実績値を利用した電力推定

電力推定式とは違って，本手法はユーザの過去のジョブ実行実績そのものを利用する方法である．**図 5** に 2015 年 4 月から 6 月までの 3 ヶ月間に「京」で実行したジョブ約 6 万本をユーザが投入したジョブのノード数のパターン数で分類した結果を示す．この時期にジョブを実行したユーザ数は 396 であった．例えば，1 人のユーザが 1,10,100 ノードのジョブを投入する場合を 3 パターンとし，ジョブの投入数は考慮しない．約 80%の利用者は 9 パターン以下の，約 97%の利用者は 19 パターン以下のノード数を指定してジョブを実行していることがわかる．

大半のユーザは少ないパターンのノード数でジョブを投入しているため，今後投入されるジョブも過去に投入したノード数と同じ可能性が高い．よって，電力推定式を用いずに今後投入されるジョブと同じノード数での実行実績の電力そのものを電力推定に用いる．**図 6** は実行実績そのものを電力推定に用いる手法を示したものである．

図 6 の左図は，実行実績の最大電力をこれから実行するジョブの推定電力とする方法である．例えば要求ノード数が 384 ノードのジョブの電力を予測する場合は，384 ノードで過去すでに実行実績があるため，そのノード数での電力の最大値 0.05MW(左図中の 1) を予測値として採用する．要求ノード数が 480 ノードの場合は過去の実行実績がない．この場合は両側の実行実績から補間をして電力を予測する．この場合は，384 ノードでの最大電力と 576 ノードでの最大電力から補間を行い 0.055MW(左図中の 2) を求める．

図 6 の右図は実行実績の平均値を用いるものである．例えば要求ノード数が 384 ノードのジョブの電力を予測する場合は，384 ノードで過去にすでに実行実績があるため，そのノードでの電力の平均値 0.045MW(右図中の 1) を予測値として採用する．要求ノード数が 480 ノードの場合は過去の実行実績がないため，両側の実行実績の平均値から補間をして電力を予測 (右図中の 2) する．

電力超過防止の観点からは，安全なマージンが含まれると見込まれる最大値を用いた予測値を採用するのが望ましい．しかし，ジョブ一つ一つの平均値からの予測電力と最大値からの予測電力との差は大きくはないが，多数のジョブが集まるとこの差が大きくなると考えられる．実行実績の平均値を用いて電力を予測した場合は，実際の電力は予測値の上下に散らばると考えられる．一方，実行実績の最大値を用いて電力を予測した場合は，実際の電力は予測値よりも下となる可能性が高い．「京」では通常運用時に大小 100 程度のジョブが同時に実行しているが，この 100 個のジョブの最大値で電力を予測した場合の「京」全体の電力と，平均値で予測した場合の「京」全体の電力では平均値の方が個々のジョブの予測誤差が打ち消しあって「京」全体で考えると予測精度が高くなる可能性がある．大規模ジョブ実行時など 1 つのジョブしか動かないような状況では最大値を採用するのが良いと考えられ，通常運用時は平均値を採用するのが良いと考えられる．

4.　評価

本手法の評価のため，「京」で実行された 2015 年 4 月から 6 月までの 3 ヶ月分のジョブについて温度センサー情報から求められる平均電力を算出した．電力推定式 (1)，実行実績の最大値 (2)，実行実績の平均値 (3) を用いた 3 つの電力予測手法の精度を比較するため，2015 年 4 月と 5 月に実行されたジョブをジョブ実行実績として用いて 6 月のジョブの電力を予測し，既に求めた 6 月のジョブ電力の値と比較した．

図 7 に 2015 年 6 月の「京」の実際の電力と電力推定式を用いた予測電力を示す．全体的に予測電力は実際電力に追随している傾向があり，誤差は-0.2MW から+0.5MW の範囲に収まることが多い．

図 8 に図 7 と同じ期間で「京」の実際の電力と実行実績を用いた予測電力の差分を示す．最大値を用いた予測手法では誤差がマイナス方向に出ていることが多い．つまり実際の「京」全体の電力よりも大きめに電力が予測されて

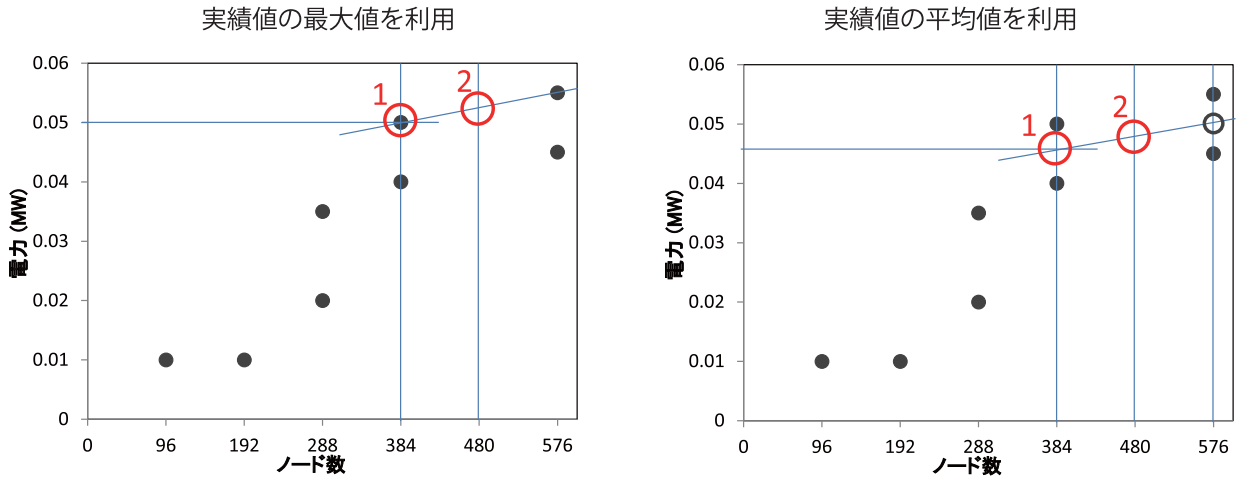

図 6 実行実績を利用した電力推定方法 (左:最大値を利用/右:平均値を利用)

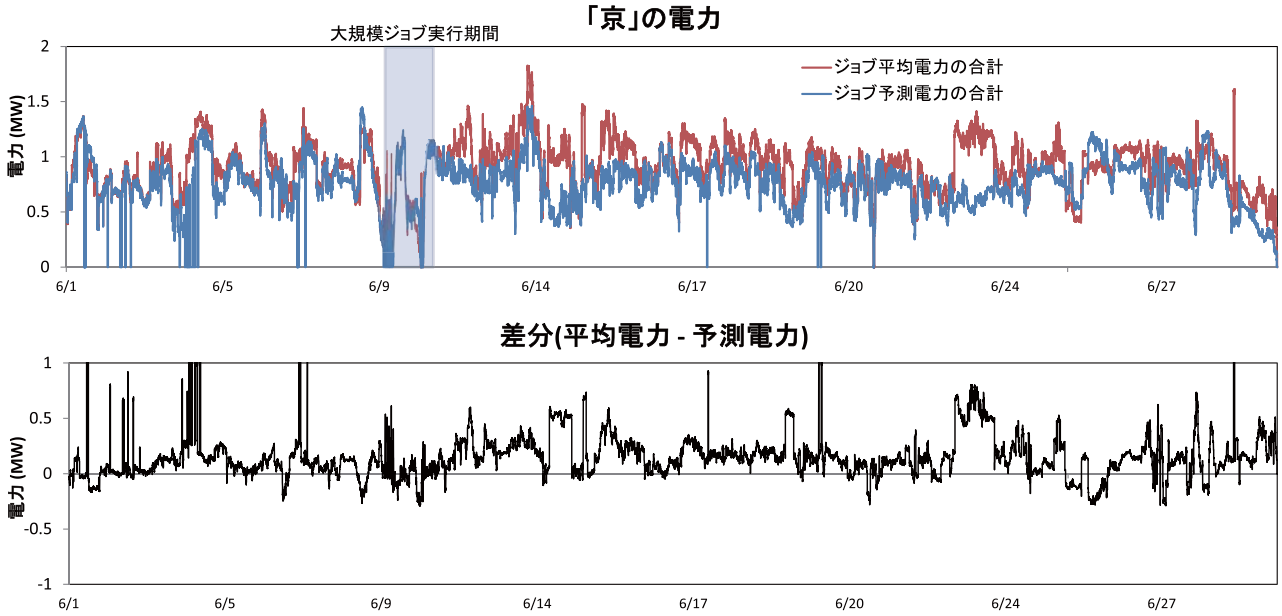

図 7 2015 年 6 月の「京」の電力予測結果/電力推定式を用いた予測

いることがわかる．一方，平均値を用いた予測手法では誤差がプラス方向に出ている傾向があることがわかる．つまり，平均値を用いる場合は，予測電力が実際の電力よりも小さいことになる．本来，平均値を用いることで「京」全体の電力で見ると誤差が打ち消しあって，プラス方向とマイナス方向に等しい誤差が発生すると予想できるが，結果は異なった．

このような結果となった理由として，すべての実行実績を用いてジョブの電力を予測していることが考えられる．つまり，ジョブの実行実績の中にはアプリケーションの不具合などで想定外に終了したジョブなどユーザの想定通りに動作しなかったジョブも含まれる．ユーザが想定通りに動作しなかったジョブを修正して再実行しても，すでにそのジョブは実行実績として残っているため予測で考慮されてしまう．

このような問題の改善策として，ジョブの終了ステータスを考慮する手法やジョブの実行時間を考慮する方法が考えられる．終了ステータスがエラーのものや，ジョブの実行時間がジョブの指定経過時間に比べて極端に短いものは実行実績として持たないようにするなど，実行実績自体をフィルタリングすると予測精度が向上すると考えられる．

4.1 誤差の分析

電力推定式を用いた予測，および実行実績を用いた予測手法についてそれぞれ誤差を分析した．ジョブ毎の予測値と実際の値との標準誤差を**表 1** に示す．この結果から，実行実績の平均値を利用してジョブの電力を予測する手法が最も良い予測精度であることがわかる．

次に，個々のジョブで誤差の大きいものに注目して分析した．図 7 図 8 共に，6 月 14 日において約 0.5MW の大き

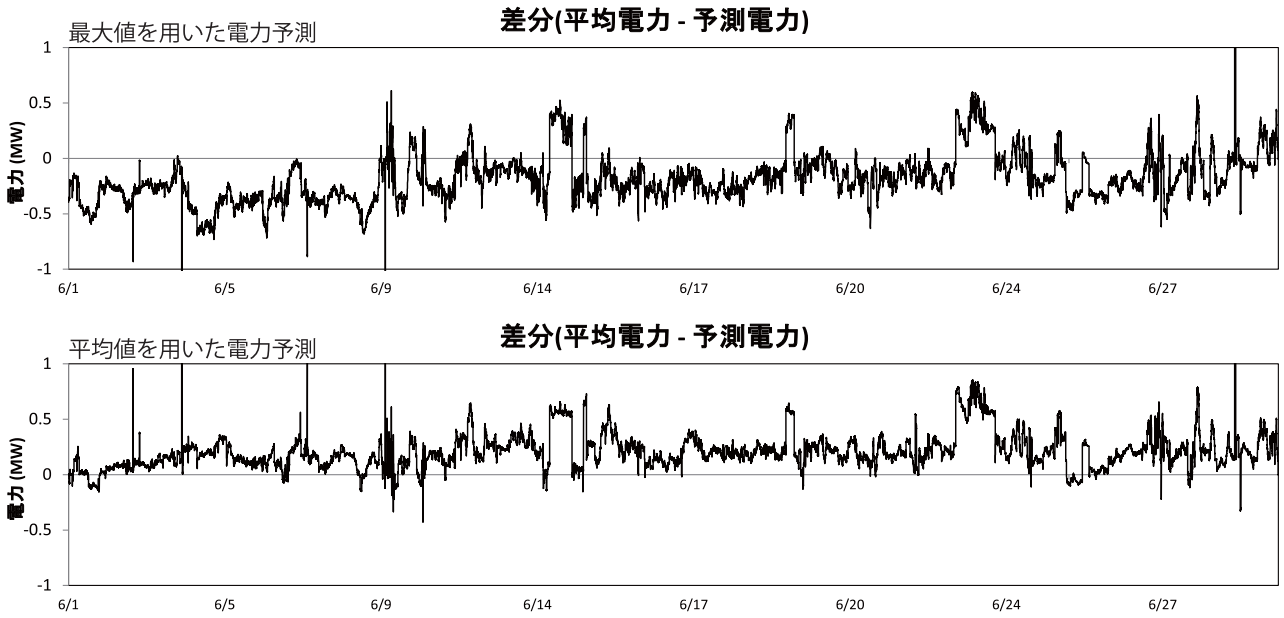

図 8 2015 年 6 月の「京」の電力予測結果/実行実績を用いた予測

表 1 各手法の標準誤差

手法	標準誤差 (MW)
(1) 電力推定式	0.0263
(2) 実行実績:最大値	0.0272
(3) 実行実績:平均値	0.0257

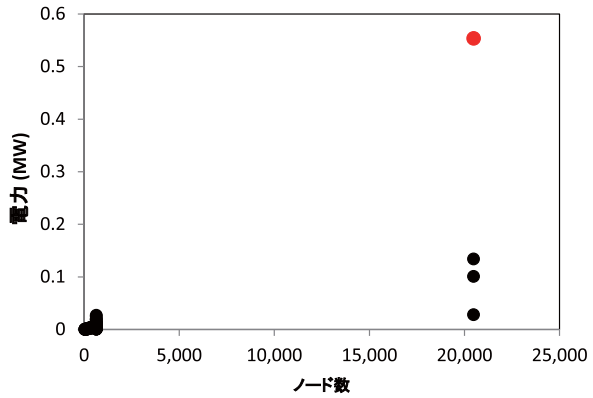

図 9 6 月 14 日時点での実行実績

な誤差が発生している．この誤差の原因を調べたところ，この期間に約 20,000 ノードのジョブが 10 時間程動いており，その予測値と実際の電力に大きな乖離があることがわかった．本ジョブを投入したユーザの 6 月 14 日時点での実行実績を**図 9** に示す．

図 9 の 0.55MW の点は，この誤差の原因となったジョブを示す．約 20,000 ノードのジョブは過去に 3 回実行しており，それぞれ 0.03MW, 0.10MW, 0.13MW の平均電力であった．この状況で 20,000 ノードジョブの電力を予測すると，平均値では 0.09MW，最大値では 0.13MW の予測値となった．しかし，実際は 0.55MW であったため，結果として約 0.5MW の誤差が生じた．また，6 月 19 日，23 日の誤差についても 14 日と同じ状況で，ノード規模が大きいため実際の電力と予測値との差が広がる結果となった．実行実績を用いる以上，図 9 の状況で 20,000 ノードジョブの電力を 0.55MW と予測するのは難しい．しかし，一度でも実行されると次回からの予測精度は上がると考えられる．

4.2 実行実績への追加タイミング

先の評価では実行実績は 4,5 月のジョブのみを用いて 6 月のジョブを予測していた．ここでは，6 月について個々のジョブ実行が終了次第，そのジョブを実行実績に追加し次回からの予測に用いる手法を評価した．**図 10** は実行実績の平均値を用いた予測手法に，実行実績にジョブを随時追加する場合と追加しない場合 (4,5 月の実績のみを利用する) の電力の実測値と予測値の誤差を示したものである．“実行実績への追加なし” のグラフは図 8 の下図と同じものである．6 月 10 日頃までは実行実績へ随時ジョブを追加する場合と追加しない場合の誤差は小さいが，6 月後半となると誤差が大きくなっていることがわかる．また実行実績へ随時ジョブを追加する場合は，実際の電力とは 0.2MW の幅で推移しており非常に精度良く予測できている．実行実績へジョブを追加しない場合にあった 6 月 19,23 日の 0.5MW 近い誤差も，実行実績を追加することによって改善している．結果からユーザは直近に実行したジョブと同傾向のジョブを実行する傾向があることがわかる．実行実績への追加する場合の標準誤差は 0.014MW であり，表 1 の結果と比べても明らかに予測精度が高いことがわかる．

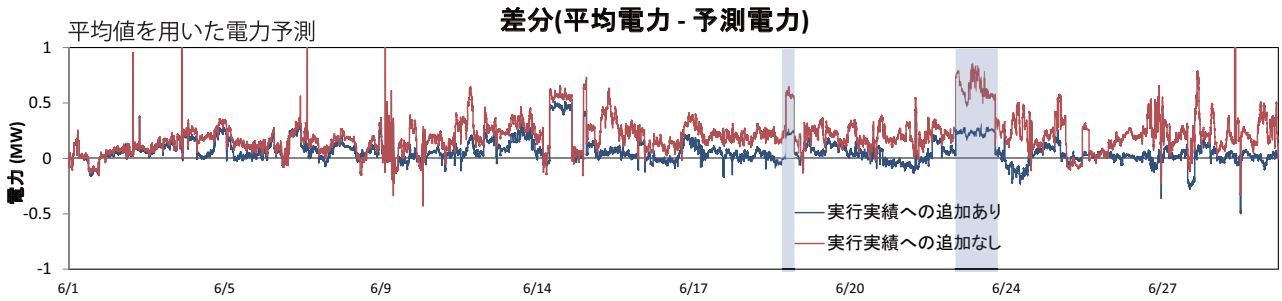

図 10 実行実績を随時更新する場合と更新しない場合での電力予測結果

5. まとめと今後の課題

本稿では，ジョブの実行前にそのジョブの電力を予測する手法について提案し評価を行った．予測の元となるデータとしてユーザが過去に実行したジョブのノード数と電力をジョブ実行実績として用いた．予測手法として，実行実績から電力推定式を作る手法 (1)，実行実績の同ノードの電力最大値を利用する方法 (2)，実行実績の同ノードの電力平均値を利用する方法 (3) の 3 種の手法を評価した．「京」にはノード単位に電力計が備わっていないため，温度センサーからジョブの電力を推定し，評価元データとした．「京」で実際に実行された 2015 年 4,5 月のジョブを実行実績とし，6 月のジョブの電力を予測し実際の電力と比較した．結果，電力平均値を利用する (3) の予測手法の精度が高いことがわかった．また，電力平均値を利用する (3) の予測手法に，ジョブの実行が終了次第そのジョブを実行実績に追加して，次回の予測に利用する手法を評価した．その結果，「京」全体では実際の電力と予測電力との誤差が 0.2MW 程度と非常に良い精度で予測できた．

今後の課題として，予測元のデータである実行実績を選別することが挙げられる．本稿では，すべてのジョブを実行実績として予測に利用していたが，中にはエラーで終了したり，ユーザの想定外の動作で終了した不完全なジョブが含まれる．これらジョブを除くことで更なる精度向上が見込まれる．また，本手法を「京」の運用に反映し「京」の 30 分後や 1 時間後などの電力変動をリアルタイムに予測し，電力超過対策に役立てる予定である．

参考文献

[1] Yamamoto, K., Uno, A., Murai, H., Tsukamoto, T., Shoji, F., Matsui, S., Sekizawa, R., Sueyasu, F., Uchiyama, H., Okamoto, M., Ohgushi, N., Takashina, K., Wakabayashi, D., Taguchi, Y. and Yokokawa, M.: The K computer Operations: Experiences and Statistics., *International Conference on Computational Science ICCS2014*, pp. 576–585 (2014).
[2] 山本啓二，宇野篤也，塚本俊之，菅田勝文，庄司文由：スーパーコンピュータ「京」の運用状況，情報処理， Vol. 55, No. 8, pp. 786–793 (2014).
[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：電力消費量の上限を考慮した「京」の運用，情報処理学会研究報告第 146 回ハイパフォーマンスコンピューティング研究会，HPC146 (2014).
[4] 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，ハイパフォーマンスコンピューティングと計算科学シンポジウム，HPCS2015 (2015).