Vol.2019-HPC-172 No.23
2019/12/19

「京」における電力超過防止の自動化への取組み

肥田 元 [1,a)]　宇野篤也 [2]　塚本俊之 [2]　池田直樹 [1]　末安史親 [3]　井上文雄 [2]

概要：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．2019 年 8 月に運用を終了した「京」でも消費電力は運用上の課題であった．特に電力超過問題は電力契約の見直し等につながることから重要な課題である．これに対し，消費電力の多い全ノードを利用するジョブを事前に審査し電力超過を未然に防止する取り組みや，消費電力が多いと思われる一番規模を大きいジョブを 1 本，短時間に停止する仕組み等を構築してきた．本稿では「京」での電力超過防止の自動化への取り組みとその運用実績について報告する．

キーワード：ＨＰＣ，「京」，電力超過対策，ジョブ緊急停止、事前審査制度

1. はじめに

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汎用並列スーパーコンピュータで，2012 年 9 月に共用を開始し 2019 年 8 月に運用を終了した．当時としては低消費電力の CPU の採用などにより消費電力を下げる取り組みを行っていたが，計算ノード数が 82,944 ノードと規模が大きいためシステム全体の消費電力も 10MW を超えており，運用コストに占める電力料金の割合も高かった．

2013 年には，特定の期間に実行する大規模ジョブ（36,865～82,944 ノード）で全ての計算ノードを使用するジョブの消費電力が我々の想定を超える事象が発生した．これは単なる違約金の支払いに留まらず，契約電力の見直しを迫られることとなり運用コストの増大を招く結果となった．契約電力の超過（以下，電力超過）の有無は，毎時 0 分または 30 分からの 30 分間の電力会社からの供給電力量が，契約電力量を超過しているかどうかで判定する．

電力超過を防ぐ取り組みとして，全ての計算ノードを使用するような大規模ジョブについて予め消費電力を予測する仕組み（事前審査制度[1]）と，電力超過が予想される場合に手動だが消費電力が多いと思われる一番規模を大きいジョブを 1 本，短時間に停止する仕組みを用意した．事前審査制度により大規模ジョブ実行時の電力超過を防止することができるようになったが，大規模ジョブ実行以外の常時 100 本前後が同時に実行する通常ジョブ（36,864 ノード以下）運用時にも発生する可能性があった．そこで，通常ジョブの運用の対策も含め常時消費電力量を監視し，電力超過を防止する仕組みを導入した．

本稿では，これら「京」での電力超過防止の取り組みの詳細とその運用実績について報告する．

1 株式会社富士通ソーシアルサイエンスラボラトリ
2 国立研究開発法人理化学研究所 計算科学研究センター
3 富士通株式会社
a) hida.hajime@fujitsu.com

2. 「京」の電力環境

「京」を設置していた理化学研究所 計算科学研究センター（以下，R-CCS）では，電力会社からの商用電力とガスタービン発電によるコジェネレーションシステム（以下，CGS）の自家発電電力により運用を行っている．

図 1 に「京」運用時の電源設備を示す．

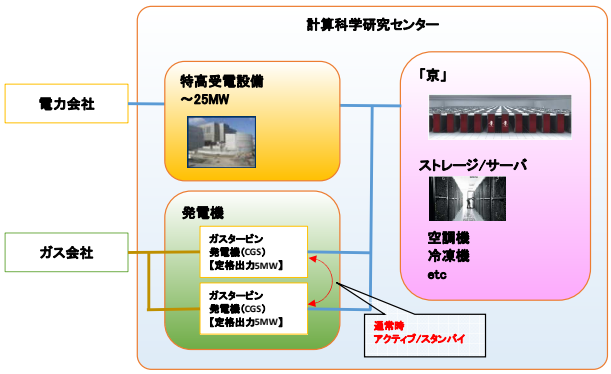

図 1　「京」運用時の電源設備

R-CCS では定格出力 5MW の CGS を 2 台備えているが，通常運用では 1 台ずつ交互に運用している[2]. CGS を同時に 2 台運用することにより，電力超過に対応することはできるが，CGS は停止状態から発電可能な状態になるまでに 1，2 時間ほど必要である．そのため，大規模ジョブ実行など事前に電力超過が見込まれる場合を除き，通常運用時の電力超過には間に合わないという問題がある．

共用開始当初の R-CCS 全体の最大消費電力見込みを表 1 に示す．

表 1　R-CCS 全体の最大消費電力見込み（共用開始時）

内訳	消費電力
「京」本体	10MW
ジョブ実行による増分	～4MW
その他施設	～3MW
合計	～17MW

電力会社からの供給電力は，R-CCS 全体の消費電力から CGS により発電した電力を差し引いた値となり，「京」の運用中に消費電力を削減するには，実行中のジョブを停止するかシステム（の一部）を停止するしかない．

そこで，運用中の消費電力を監視し，電力超過が見込まれる場合には実行中のジョブを停止し，電力超過を防ぐ仕組みを実装することにした．

3. 電力超過予測時の自動ジョブ停止手法

電力超過の予測方法と，電力超過防止のためのジョブ停止機能（以下，緊急ジョブ自動停止機能）について説明する．一連の処理フローを図 2 に示す．

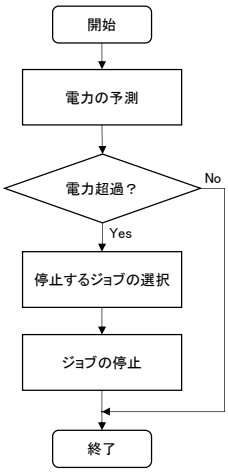

図 2　処理フロー

3.1 電力の予測

電力超過は，毎時 0 分または 30 分からの 30 分間の電力会社からの供給電力量が，契約電力量を超過しているかどうかで判定する．そこで，1 分間ごとに消費電力を監視し，それまでの累積値をもとに 30 分間の累積値を予測することにした．具体的には，リアルタイムに 1 分毎の消費電力収集し，30 分の区切り内の区切りから現在の時刻までの累積値から 1 分間の平均値を求め，30 分間の累積値に換算して予測を行う．予測時点での累積値をもとにするため，30 分間の実消費電力が同じ場合でも，各予測時点での 30 分間の消費電力の予測結果は大きく異なる場合がある．図 3 に予測時点での消費電力の予測が異なる例を示す．

図からわかるように，30 分間の前半で電力超過が予測される場合でも，その後の利用状況によっては電力超過とならない場合がある．

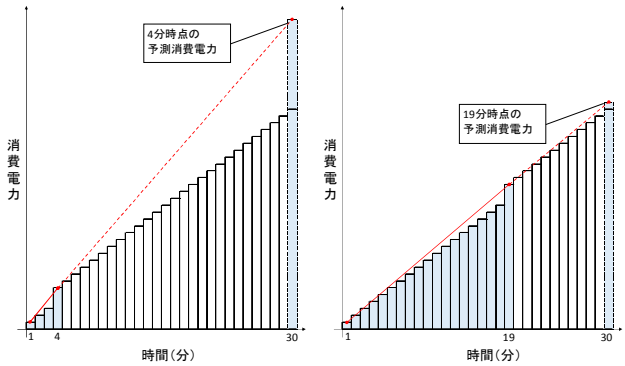

図 3　30 分後の消費電力予測の違い
（左）4 分時点での予測（右）19 分時点での予測

3.2 電力超過判定

前述の通り，各予測点での電力超過判定はその時点までの実績をもとに行うため，例えば 30 分間の前半と後半では電力超過までの猶予が大きく異なる．そこで，電力超過判定までの猶予に応じて電力超過判定に用いる閾値を可変（以下，可変閾値）とすることにした．

3.2.1 可変閾値

可変閾値は毎分毎に，過去の実績からジョブを停止することで削減できる 1 分間の最大電力（以下，ジョブ削減最大電力）に 30 分の残りの時間を乗算し契約電力を加算して設定する．図 4 に消費電力の推移と可変閾値の関係を示す．

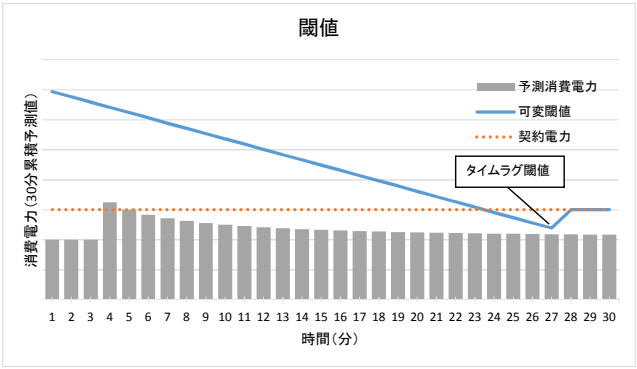

図 4　閾値

図 4 の可変閾値が 23 分過ぎから契約電力以下となっているのは，残り 3 分間ジョブの停止が間に合わなかった場合を想定し，過去最大の消費電力が 3 分間続いても契約電力を超えない値（以下，タイムラグ閾値）になるようにしているためである．

式（1），（2）に W_{lt} のタイムラグ閾値と W_{vt} の可変閾値を求める式を示す．タイムラグ閾値以降の残りタイムラグ時間の可変閾値は契約電力としている．

$$W_{lt} = \frac{W_c - W_{max} \cdot T_{lag}}{T - T_{lag}} \cdot T \qquad (1)$$

$$W_{vt} = \left(T - T_{lag} - T_{now}\right) \cdot W_{dmax} + W_{lt} \qquad (2)$$

ここでの各パラメータは以下のとおりである．W_{lt}：タイムラグ閾値，W_c：契約電力，W_{max}：1 分間の過去最大電力量，T_{lag}：タイムラグ時間（3 分），T：周期（30 分）．W_{vt}：可変閾値，T_{now}：周期 30 分内の現在時刻（分），W_{dmax}：1 分間のジョブ削減最大電力．

電力超過と判定した場合は，超過判定分の電力を削減することになる．

3.3 停止するジョブの選択

電力超過と判定された場合，超過電力に相当する実行中のジョブを停止する．そのためには，実行中のジョブ毎の消費電力を求める必要がある．この時，単純にジョブを停止するのではなく，ジョブを停止することで失われる計算資源量（計算ノード×時間，以下，ノード時間積）が最小となるようにジョブを選択する．

3.3.1 ジョブ毎の消費電力の推定

「京」のラックには CPU，メモリ等の消費電力を測定するセンサーが搭載されていないため，温度情報を元にジョブ毎の消費電力を推定している[3].

ジョブ毎の推定消費電力は 5 分間隔で収集した温度情報から計算で求めているため，1 分単位で収集している「京」本体の消費電力の実測値と比較すると誤差がある．また，ジョブが全て停止している場合（計算ノードのアイドル電力のみ）の消費電力は約 10MW であることから，「京」本体全体の消費電力からの差分がジョブ全体の消費電力となる．実測値と温度情報からの推測値には誤差があるため，温度情報から推測したジョブ毎の消費電力の割合を実測値に適用して補正することでジョブ毎の消費電力を推測している．図 5 に補正前と図 6 に補正後の結果について示す．

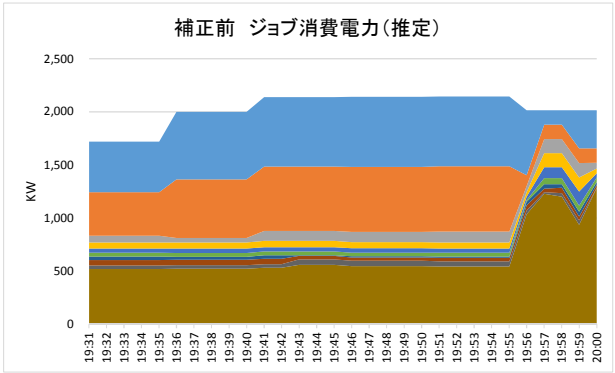

図 5　補正前

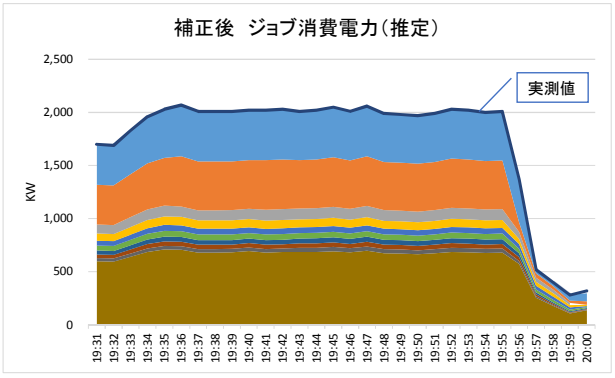

図 6　補正後

グラフの色分けは，推定消費電力の大きいジョブの上位 9 ジョブとそれ以外のジョブを合算した推定消費電力とした．実際の運用では，実行中のすべてのジョブの推定消費電力を求めている．

消費電力の大きいジョブを停止することが目的であるため，この程度の補正で良いと考えた．

3.3.2 ジョブの選択

削減すべき電力に相当するジョブを選択する場合，消費電力の多いジョブから順に削減すべき電力となるまでが対象となる．このとき 1 つのジョブで削減すべき電力を超える場合は，2 つ以上のジョブの組み合わせと比較し削減すべき電力に近い方を選択する．また，ジョブの消費電力の合計が削減すべき電力以上であることが条件となる．

ジョブを停止するとこれまで実行したノード時間積が失われることになり，消費電力のみでジョブを選択した場合，ノード時間積の損失が大きくなる可能性がある．このことから，削減すべき電力量に加えて損失するノード時間積が最小となる組み合わせでジョブを選択する必要がある．2 つの組み合わせの最適解を求めるために遺伝的アルゴリズム（以下，GA）を用いた[4].

GA は試行回数を増やせば精度が上がるが処理に時間がかかり，対象のジョブが多い場合も同様である．処理時間については，早急にジョブ停止を行う必要があるため，ジョブを選択する時間は 1 秒以内とした．GA を 400 回の進化計算ループで行った場合，ジョブ 10 本～60 本の最適解を求めるには 7 秒から 12 秒程度かかった．1 秒以内とするため，進化計算ループ毎の終了時に求めた最適解が 10 回連続で同じ解であれば，打ち切ることとした．

対象ジョブ数については，図 7 の検討当時の 2016 年 11 月の 1 ヶ月分の同時実行ジョブ数の平均値 130 本から絞り込むこととした．

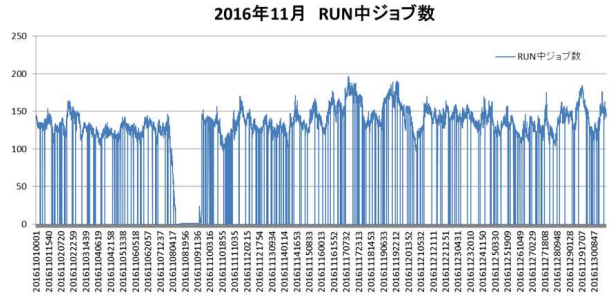

図 7　1 ヶ月の同時ジョブ実行数

電力への影響が少ないと思われる，384 ノード以下のジョブを除いた場合，平均ジョブ数は 52 本であった．影響があると思われる 1,000 ノード以上の平均ジョブ数は 14 本であった．このことからジョブは 14 本から 52 本程度が対象となると考えた．

処理時間については，対象となるジョブ数を 10 本から 100 本であらかじめ正解のあるテストデータで検証したところ，1 秒未満で処理が完了するジョブ数は 60 本以下であった．精度については，ジョブの本数が少ない方が高い結果となった．表 2 に検証結果を示す．

表 2　テストデータによる GA の精度検証

ジョブ数	テスト回数										処理時間平均
	1	2	3	4	5	6	7	8	9	10	
10	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	1秒未満
20	OK	OK	OK	+1	OK	OK	OK	OK	OK	OK	1.0秒
30	OK	OK	OK	OK	OK	+1	OK	OK	OK	OK	1秒未満
40	+9	OK	OK	OK	OK	OK	OK	+1	+2	+1	1秒未満
50	+4	+3	OK	+4	OK	+1	+1	+1	OK	+3	1秒未満
60	OK	+5	+3	+4	OK	+3	+1	OK	+1	+5	1秒未満
70	+4	+6	+6	+17	+4	+5	+6	OK	+4	+7	1.1秒
80	+15	+1	+3	+1	+2	+20	+9	+7	OK	+9	2.1秒
90	+29	OK	+8	+2	+3	+8	+18	+1	+20	+5	1.4秒
100	+13	+65	+29	+92	+76	+21	+36	+22	+45	+4	1.9秒

表 2 の OK は正解のジョブの組み合わせを選択，数値は不正解のジョブを選択したことにより正解のジョブの組み合わせに比べ超過したノード時間積を表している．

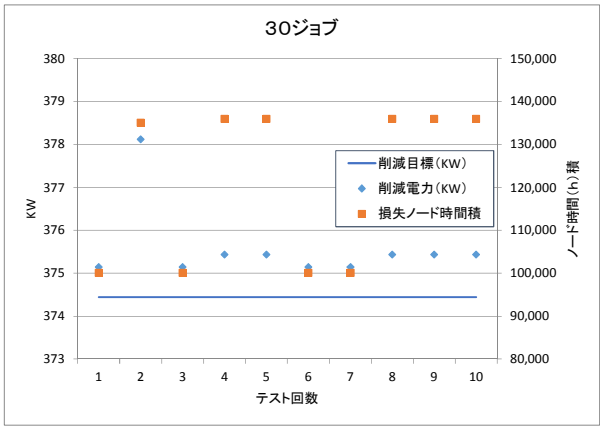

図 8　実データによる GA の精度検証（ジョブ 30 本）

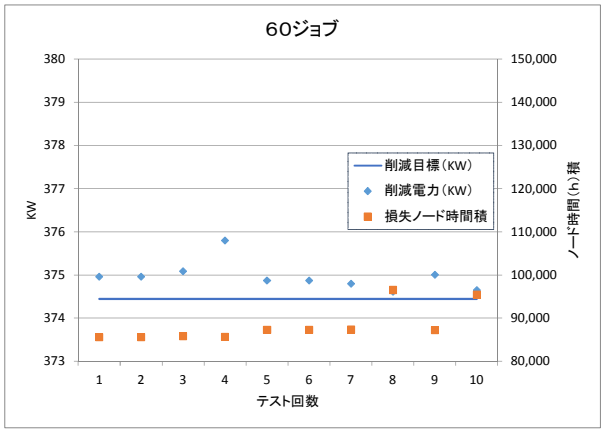

図 9　実データによる GA の精度検証（ジョブ 60 本）

処理時間平均が 1 秒未満であるジョブ 30 本と 60 本について，実データで精度を比較した図 8 と図 9 を示す．

実データでは，推定消費電力を大きい順にそれぞれ上位ジョブ 30 本と 60 本で比較した．ジョブ 30 本より 60 本の方が，削減電力およびノード時間積が良い結果となった．良い結果となった要因は，ジョブ 60 本はジョブ 30 本を包含し更に推定電力の少ないジョブ 30 本が選択肢として加わったためと思われる．

これらの結果から，1 秒以内に処理可能なジョブの推定電力の大きい上位 60 本を停止するジョブの対象とした．

3.4 ジョブの停止

ジョブの停止は，ホールドコマンド（以下，pjhold）により停止対象のジョブを一括で実行中からキュー状態に変更することで行う．この時，pjhold の実行結果をチェックし，ユーザが再実行を許可しない等の設定で停止できなかったジョブについては一括でキャンセル（以下，pjdel）する．pjdel でも停止しない場合は，ジョブの先頭ノードを強制的にダウンさせる仕組みも組み込んだが，実際の運用ではノードダウンを行う前にジョブが停止したため，動作することはなかった．

4. 緊急ジョブ自動停止機能の運用

2017 年 8 月 8 日より緊急ジョブ自動停止機能で常時電力超過を監視する運用を開始した．運用開始後，通常運用では電力超過は発生しなかったが，電源設備の故障による電力超過が 2018 年 8 月に発生した．これは，台風の影響で CGS の発電電力を「京」へ送電する電力系統が故障し，給電電力低下となったのが原因である．図 10 に CGS からの電力供給が停止し緊急ジョブ自動停止機能が動作した際の状況と消費電力の推移を示す．

CGS からの給電電力が 0:49 頃に停止し，0:50 から施設全体消費電力は，電力会社からの受電電力のみで給電され

ることとなった．0:31 から 30 分間の消費電力は故障前の消費電力が少なかったため，電力超過とはならなかった．次の 1:01 からの 30 分間では，緊急ジョブ自動停止機能により 1:03 から 1:06 の間に 123 本のジョブを停止し，1:07 に停止すべき実行中のジョブがなくなった．

表 3 にジョブの停止時刻と推定削減電力を示す．

表 3　ジョブ停止と削減電力（8/24）

ジョブ停止時刻	目標削減電力（KW）	推定削減電力（KW）	停止ジョブ（本）
2018/8/24 1:03	1,539.4	1,000.2	60
2018/8/24 1:04	1,874.4	84.0	59
2018/8/24 1:05	2,009.4	0.2	1
2018/8/24 1:06	1,644.4	0.3	3

最初の 60 本のジョブ停止で削減可能な電力の大半を削減したことがわかる．

1:24 頃に「京」本体の電源を運用管理者の判断により停止したことにより，電力超過を回避した．

今回のように，CGS からの電力供給が停止した場合には，全ジョブを停止しても電力超過を防止することはできない．しかし，超過判定までの時間を稼ぐことはでき，運用管理者等の対応時間を確保することができる．

その後，電力設備の応急修理を行い CGS からの供給電力が通常より少ない状態で運用を再開した．

運用再開後，修理が完了するまでの間に CGS からの供給電力不足により 2018 年 8 月 31 日と 9 月 4 日の計 2 回，電力超過を予測し，緊急ジョブ自動停止機能によりジョブの停止を行った．

図 11 に 8 月 31 日に電力不足により緊急ジョブ自動停止機能が動作した結果を示す．

21:19 に緊急ジョブ自動停止機能により電力超過の予測を検知し 21:20 よりジョブの停止を開始．21:29 に予測消費電力が可変閾値以下となり終了した．

表 4 に 8/31 のジョブ停止により削減された電力を示す．

表 4　ジョブ停止と削減電力（8/31）

ジョブ停止時刻	目標削減電力（KW）	推定削減電力（KW）	停止ジョブ（本）
2018/8/31 21:20	130.9	147.5	6
2018/8/31 21:21	161.1	165.1	5
2018/8/31 21:22	191.8	190.3	8
2018/8/31 21:23	222.9	222.3	6
2018/8/31 21:24	280.5	280.7	23
2018/8/31 21:25	286.1	175.1	59
2018/8/31 21:26	189.1	75.1	19
2018/8/31 21:27	176.1	81.7	6
2018/8/31 21:28	211.1	174.0	5

21:25 あたりから目標削減電力を満たすジョブがなくなり，以降，ジョブが起動する毎にジョブの停止が行われている．このことから，電力超過の要因がジョブによるものではないことがわかる．この時は，結果的には 80KW を残し電力超過とはならなかった．

図 12 に 9 月 4 日に電力不足により緊急ジョブ自動停止機能が動作した結果を示す．

19:52 に緊急ジョブ自動停止機能により電力超過の予測を検知し，19:54 からジョブの停止が始まった．19:58 に予測消費電力が可変閾値以下となり終了した．

表 5 に 9/4 のジョブ停止により削減された電力を示す．

表 5　ジョブ停止と削減電力（9/4）

ジョブ停止時刻	目標削減電力（KW）	推定削減電力（KW）	停止ジョブ（本）
2018/9/4 19:54	222.9	463.3	1
2018/9/4 19:55	150.0	472.9	3
2018/9/4 19:56	286.1	288.4	7
2018/9/4 19:57	189.1	181.1	12
2018/9/4 19:58	100.0	114.1	3

19:54 の時点では，目標削減電力に比べ推定削減電力の方が 2 倍以上となっていた．損失するノード時間積が最小となる選択によるものと思われたが，実行中のジョブを確認したところ，4 本のジョブを停止することで推定削減電力 253.6KW，ノード時間積についても GA の選択に比べ半分強の選択肢があり，GA による選択結果が最適解ではなかったことがわかった．19:55 の 3 本のジョブの停止についても同様であった．また，19:57 においても停止すべきジョブが存在するにもかかわらず，推定削減電力が目標削減電力より 8KW 足りない選択となっていた．

ジョブ停止で最適なジョブが選択されない場合もあったが，推定削減電力が目標削減電力を上回っており，160KW を残し電力超過とはならなかった．

5. おわりに

今回，「京」での電力超過防止の取り組みとして，常時システムの消費電力を監視し，超過が予測される場合には必要最低限のジョブを自動で停止し，電力超過を防ぐ仕組みを導入した．本機能の導入により，「京」の通常運用時の電力超過は防止できたものと考えている．しかしながら，CGS からの給電が停止するなど電力設備の不具合発生時には，計算ノードのアイドル電力のみで超過状態となるため，本機能では電力超過を防止することはできず，運用管理者が「京」本体の電源を停止させる必要がある．本体の停止は様々な手続が必要になるため，現段階では自動化は難しい．また，環境の制約によるものが大きいが現システムのジョブ停止に関する精度は高くなく，場合によっては電力超過を防止できなかった可能性もあったと考えている．

「京」の運用は終了したが，「京」の運用を通して得ら

れた様々な知見は現在導入中の「富岳」の運用でも活かしていきたいと考えている.

参考文献

[1] 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 Vol.2014-HPC-146,No4(2014).
[2] 黒川原佳，庄司文由：“スーパーコンピュータ「京」システム概要“：情報処理，vol53，No.8，p.759-766(2012).
[3] 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).
[4] “DEAP”. https://github.com/DEAP/deap, ver1.0,(参照 2016-11-01).

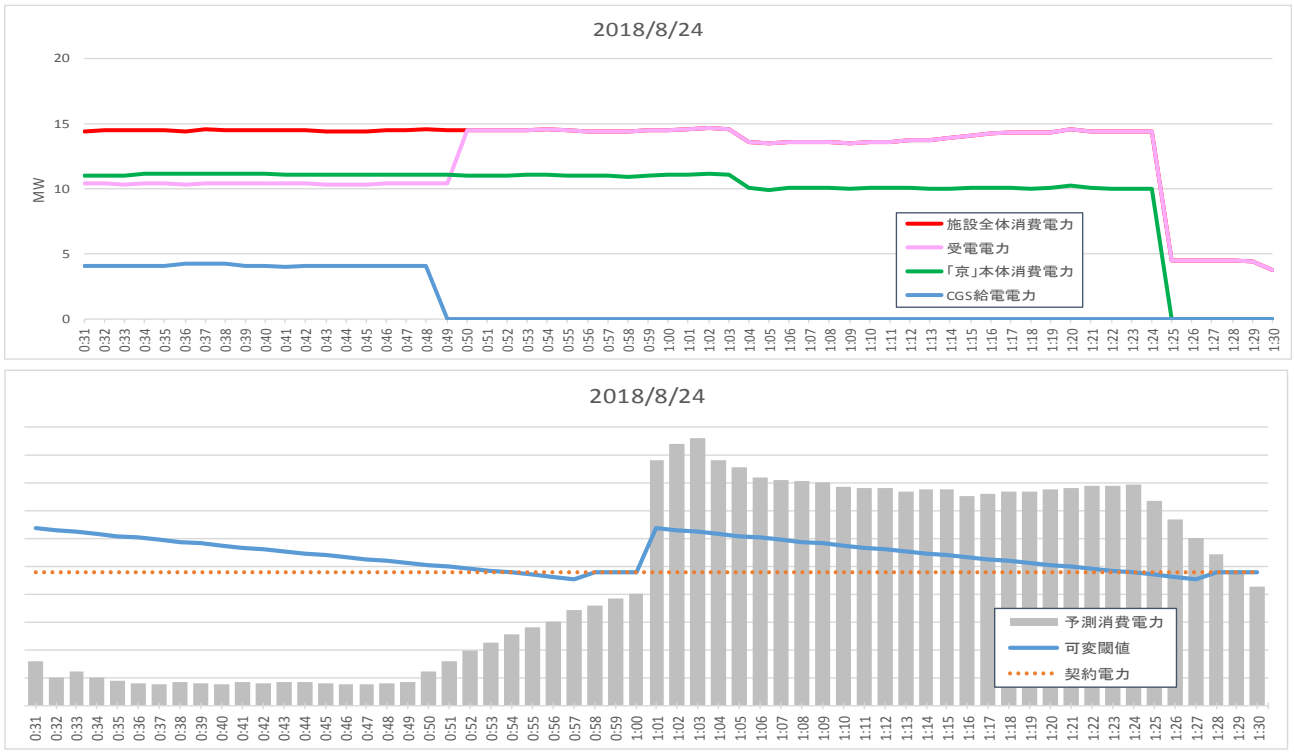

図 10　CGS からの電力供給断時の緊急ジョブ自動停止機能の動作結果（8/24）

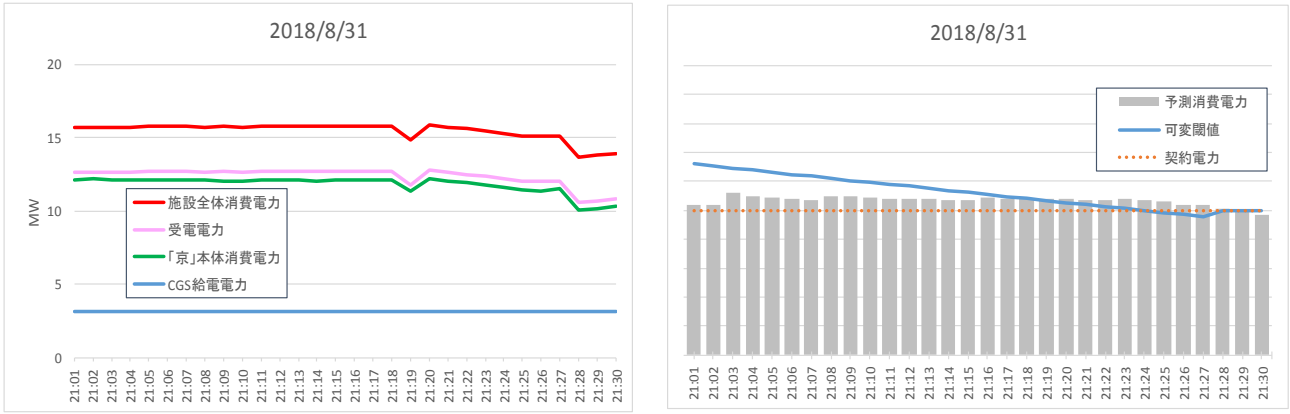

図 11　CGS からの電力供給低下時の緊急ジョブ自動停止機能の動作結果（8/31）

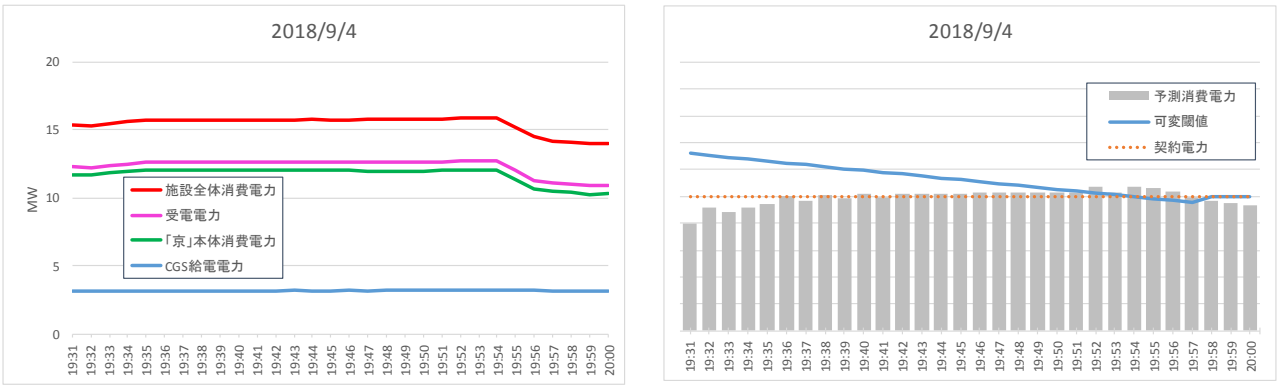

図 12　CGS からの電力供給低下時の緊急ジョブ自動停止機能の動作結果（9/4）