Vol.2017-HPC-160 No.2
2017/7/26

PC クラスタにおける省電力化のための自動電源制御方式

今出広明[†1]　加賀美崇紘[†1]　三浦健一[†1]　井口裕次[†1]　坂口吉生[†1]　藤田直行[†2]

概要：近年，社会全体で節電，省電力化の意識が高まるなか，PC クラスタにおける省電力化の要請が強くなっている．省電力化対策としてバッチジョブスケジューラの中には，ジョブが実行されていない待機状態が一定時間継続した計算ノードに対し電源停止を行う自動電源制御機能を有するものも存在する．しかしこれらの機能はジョブのスケジューリング結果や電源停止，再投入処理に要する時間的，電力的なコストを考慮していないため，必要のない電源停止を行ったり，電源再投入処理の開始が遅れたりする可能性がある．このような問題に対し，我々はバッチジョブスケジューラのスケジューリング結果を元に電源停止処理，再投入処理の開始タイミングを決定する自動電源制御方式 JSCAPS (Job SCheduling Aware Power Save)を提案する．本方式では，バッチジョブスケジューラのスケジューリング結果を元に一定時間以上待機状態が続くことが予測される計算ノードに対して電源を停止する．本方式の有効性を検証するために，既存のバッチジョブスケジューラで用いられる自動電源制御方式と本方式をシミュレーション上で比較した．その結果，待機中の消費電力量の削減量は本方式が既存方式よりも大きいことが確認された．

キーワード：PC クラスタ，省電力化，自動電源制御，バッチジョブスケジューラ

1. はじめに

近年，社会全体で節電，省電力化の意識が高まるなか，PC クラスタにおける省電力化の要請が高くなっている．省電力化のための対策としては，省電力 CPU やメモリの採用により PC サーバ自身の消費電力を抑える方式や，その上でジョブを実行していない待機状態となっている計算ノードの電源を停止する方式などが挙げられる．

待機中の計算ノードの電源を停止する方式は，ジョブの実行性能に影響を与えないことから有効な省電力化方式の一つとして注目されており，バッチジョブスケジューラである Slurm[1]や LSF (IBM Spectrum LSF)[2]では一定時間待機状態が継続した計算ノードに対し電源を自動的に停止し，次のジョブの実行開始時刻になると電源を再投入する自動電源制御方式を採用している．しかしバッチジョブスケジューラのスケジューリング結果や，電源停止，再投入処理に要する時間的，電力的なコストを考慮せずに電源を停止するために，必要のない電源停止を行ったり，電源再投入処理の開始が遅れたりする可能性がある．

このような問題に対し，本論文ではバッチジョブスケジューラのスケジューリング結果を元に電源停止に必要なコストを計算，自動的に電源を停止する JSCAPS (Job SCheduling Aware Power Save) を提案し，省電力効果について評価する．

2. Slurm，LSF における自動電源制御方式

Slurm, LSF で用いられる自動電源制御方式では計算ノードの状態により，電源停止，再投入処理の開始が判断される．

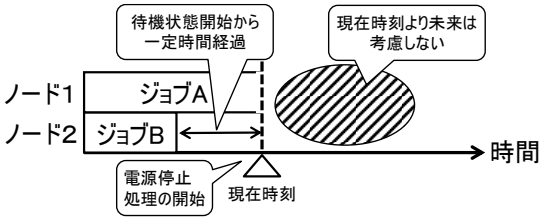

図 1　既存方式における電源停止タイミング

電源停止処理は図 1 のように，前のジョブ（ジョブ B）の実行が完了してから現在時刻までの待機状態があらかじめ決められた時間以上続いていた場合開始される．このとき現在時刻よりも未来にある次のジョブの実行開始予定時刻などは考慮されないため，待機状態が始まってから次のジョブの実行が開始されるまでの時間（待機時間）が短かった場合，電源停止処理の完了直後に電源の再投入処理を開始しなければならない．また電源停止処理を開始するまでの間に一定時間を要するため，この間は待機中の消費電力（待機電力）を余分に消費し続けてしまう．

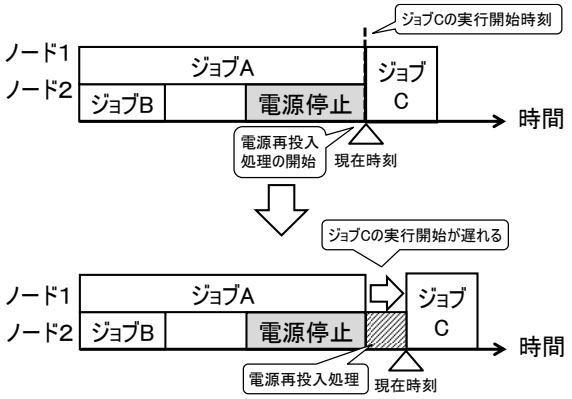

図 2　既存方式における電源再投入タイミング

†1 富士通株式会社
Fujitsu Ltd.
†2 国立研究開発法人 宇宙航空研究開発機構
Japan Aerospace Exploration Agency

電源投入処理は図 2 のように次のジョブ（ジョブ C）の実行開始予定時刻となったときに開始されることから，再投入処理が完了するまでの間はそのジョブの実行開始は遅くなり，そのジョブの実行を予定している他の計算ノード（ノード 1）は実行開始までの間待機電力を消費し続けてしまう．

既存の自動電源制御方式において効率的な省電力化を実現するための課題を以下にまとめる．

課題1 次のジョブの実行開始予定時刻を考慮しないため，待機時間が短い場合必要のない電源停止を行う可能性がある．

課題2 電源を停止するまでの間待機状態が続くため，その間の待機電力を余分に消費する．

課題3 次のジョブの実行開始予定時刻となってから電源再投入処理を開始するため，電源再投入処理の完了まで次のジョブが実行されず，他の計算ノードは待機状態が続く．

3. バッチジョブスケジューリングに基づく自動電源制御方式 JSCAPS

3.1 要件

2 章で挙げた課題を元に，効率的な省電力化を行うための自動電源制御方式の要件を以下にまとめる．

要件1 課題 1 を防ぐためには，待機時間を予測する必要がある．

要件2 課題 2 を防ぐためには，待機状態が開始されると同時に電源停止すべきか判断する必要がある．

要件3 課題 3 を防ぐためには，次のジョブの実行開始予定時刻までに電源再投入処理を完了する必要がある．

3.2 方針

JSCAPS では，各要件に対して以下の方針で自動電源制御を行う．

方針1 要件 1 に対し，バッチジョブスケジューラのスケジューリング結果から待機時間を予測し，あらかじめ決めた時間以上であれば電源を停止する．電源停止の判断基準となる時間（電源停止基準待機時間T_{thr}）は時間的，電力的なコストを考慮して決める．これにより必要な場合のみ電源停止を行うことができる．

方針2 要件 2 に対し，バッチジョブスケジューラからジョブの実行状況を定期的に参照し，ジョブの終了と同時に実行していた計算ノードに対して電源停止の判断を行う．これにより，待機状態とならずに電源停止処理を開始することができる．

方針3 要件 3 に対し，あらかじめ電源再投入処理に要する時間を見積もり，ジョブの実行予定時刻から再投入処理に要する時間を引いた時刻に再投入処理を開始する．これにより電源再投入処理の完了直後に次のジョブの実行を開始することができる．

3.3 JSCAPS による自動電源制御

JSCAPS ではジョブの実行状況をt_{itr}(s)ごとに監視し，ジョブが終了し待機状態が始まった計算ノードに対して図 3 の処理を行う．

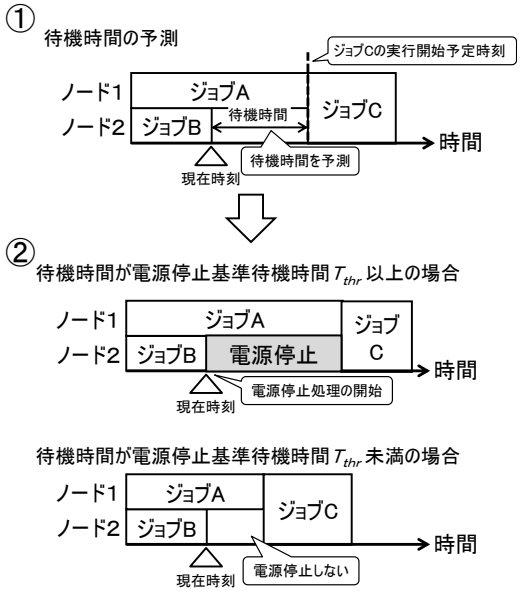

図 3　JSCAPS における電源停止タイミング

図 3 の処理の流れを以下に示す．

1. バッチジョブスケジューラのスケジューリング結果から該当ノードにおける次のジョブの実行開始予定時刻を参照し，待機時間を計算する．
2. 待機時間が電源停止基準待機時間T_{thr}以上である場合，電源停止処理を開始する．

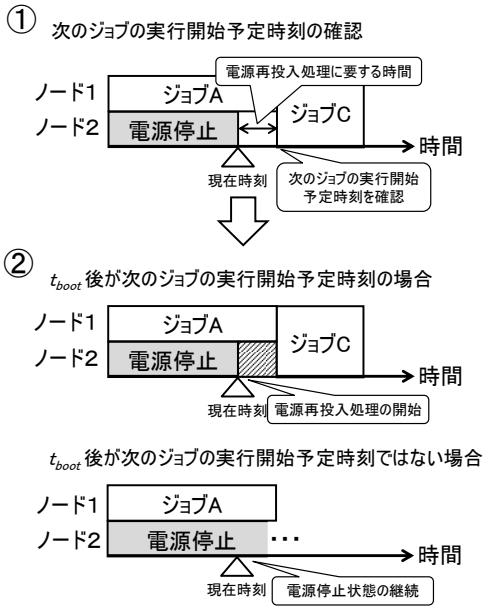

図 4　JSCAPS における電源再投入タイミング

JSCAPS により電源を停止したノードに対しては，電源再投入に要する時間をt_{boot}(s)としたとき，$T_{thr} - t_{boot}$経過後に電源を再投入すべきか判断する図 4 の処理が開始される．電源再投入時の処理の流れを以下に示す．

1. バッチジョブスケジューラのスケジューリング情報を元に，現在時刻から電源再投入処理時間t_{boot}経過後に次のジョブの実行開始予定時刻となっているか確認する．
2. 次のジョブの実行開始予定時刻となっている場合，電源再投入処理を開始する．次のジョブの実行開始予定時刻となっていない場合，ステップ３に進む．
3. 監視間隔t_{itr}経過後にステップ１に戻る．

電源停止基準待機時間について

JSCAPS では時間的，電力的なコストを消費電力量とし，予測した待機時間の間電源を停止した場合の消費電力量と，停止しなかった場合の消費電力量を計算する．電源を停止した場合の消費電力量は電源停止処理，再投入処理に必要な処理時間と消費電力を考慮して求める．電源を停止した場合の消費電力量が停止しなかった場合の消費電力量より小さくなる時間を電源停止基準待機時間T_{thr}とする．

電源停止処理に要する時間をt_{halt}(s)，電源停止状態が続く時間をt_{stop}(s)，電源再投入に要する時間をt_{boot}(s)とし，電源停止処理に要する消費電力をp_{halt}(W)，電源停止中の消費電力をp_{stop} (W)，電源再投入処理に要する消費電力をp_{boot}(W)とすると，電源停止処理を開始してから電源再投入処理を完了するまでの消費電力量E_{stop}(Ws)は，

$$E_{stop} = p_{halt} \cdot t_{halt} + p_{stop} \cdot t_{stop} + p_{boot} \cdot t_{boot}$$

となる．電源を停止しなかった場合，同じ時間での消費電力量E_{wait}(Ws)は，待機中の消費電力をp_{wait}(W)とすると，

$$E_{wait} = p_{wait} \times (t_{halt} + t_{stop} + t_{boot})$$

となる．電源停止による消費電力量を電源停止しなかった場合より小さくするためには，

$$E_{stop} < E_{wait}$$

$$t_{halt} + t_{stop} + t_{boot} > \frac{E_{stop}}{p_{wait}}$$

となることから，

$$T_{thr} > \frac{E_{stop}}{p_{wait}}$$

としてT_{thr}を求めることができる．

4. 評価

4.1 評価方針

JSCAPS はバッチジョブスケジューラのスケジューリング結果を元に電源制御を行っていることから，予測したジョブの実行開始予定時刻どおりにジョブの実行が開始された場合，省電力効果が最大となることが予測される．一方で実際には新規ジョブの投入や実行中ジョブの終了は非計画に発生するため，ジョブの実行は予定時刻どおりに行われない．このとき，JSCAPS における省電力効果が影響を受けることが予測される．したがって JSCAPS の評価検証は以下の方針で行う．

- ジョブが実行予定時刻どおりに実行された場合において，既存方式と省電力効果を比較する．
- ジョブが実行予定時刻どおりに実行されなかった場合における JSCAPS への影響を評価する．

4.2 評価項目

省電力効果の評価では，電源停止による消費電力量の削減量と，電源停止による他のジョブのスケジューリングへの影響について検証するために，以下の項目について計測する．

- ジョブ未実行中の計算ノードの消費電力量
- 平均ノード稼働率

ジョブ未実行中の消費電力量は，電源を停止しない場合待機中に消費された電力量となり，電源を停止する場合電源停止処理，再投入処理に要した消費電力量と，電源停止状態の間に消費された電力量の合計値となる．

平均ノード稼働率は以下の式により求める．

$$\text{平均ノード稼働率} = \frac{\text{計測期間中の全ジョブの実行時間}}{\text{計測期間中の経過時間}\times\text{計算ノード数}} \quad (1)$$

4.3 評価方法

JSCAPS と既存方式における省電力効果を評価するため，同一のバッチジョブスケジューラにそれぞれの方式を適用し，各計算ノードの動作状況とジョブの実行状況をシミュレーションし，ジョブ未実行中の計算ノードの消費電力量，平均ノード稼働率を計測する．

シミュレーションに用いるバッチジョブスケジューラは以下の方針でスケジューリングを行う．

- ジョブのスケジューリング優先度は FIFO とする．
- 投入されたジョブに対し，ノード ID 順に，投入時にユーザにより指定されたジョブ実行予定時間が待機時間より小さければ割り当てる．
- バックフィルスケジューリングを有効とする．

予測したジョブの実行開始予定時刻どおりにジョブが実行される場合とされない場合を評価するために，実際の PC クラスタ環境に投入されたジョブ情報をもとに，ジョブの実行予定時間を変更したジョブミックスを作成する．JAXA で運用される計算機システム"JSS2"の"SORA-PP"[3]上で JSCAPS を適用するために，作成するジョブミックスの元となるジョブ情報は"SORA-PP"で実際に投入されたジョブ情報を使用した．使用したジョブ情報を表 1 に示す．表 1 における計測期間は，消費電力量や計算ノード平均稼働率などを計測する期間である．高稼働率における計測期間はバッチジョブスケジューラのスケジューリング待ちキューに十分な数の実行待ちジョブが溜まっている期間を設定し，低稼働率における計測期間は高稼働率と同期間とな

るように設定した.

表 1 "SORA-PP"から抽出したジョブ情報

	高稼働率	低稼働率
抽出期間	2015.11.14 0:00 ～2015.11.30 23:59	2015.9.19 0:00 ～2015.09.29 23:59
ジョブ数	9,599 ジョブ	2,622 ジョブ
平均ノード稼働率	97.9%	50.4%
計測期間 ※抽出開始からの経過秒	550,000 秒 ～1,050,000 秒	172,800 秒 ～672,800 秒

表 1 で抽出したジョブ情報に対し，実行予定時間を以下の式を用いて変更し，$\alpha = 1.0, 1.25, 2.0, 5.0, 10.0$ となる 5 パターンのジョブミックスを作成した.

ジョブ実行予定時間=α×実際のジョブの実行時間 (2)

ジョブ実行時間が 100 秒となるジョブにおいて，$\alpha = 1.0$ となるジョブミックス内ではジョブ実行予定時間は 100 秒となり，$\alpha = 10.0$となるジョブミックス内ではジョブ実行予定時間は 1,000 秒となる．ジョブが実行予定時刻どおりに実行される場合の評価では$\alpha = 1.0$ のときのジョブミックスを用い，ジョブが実行予定時刻どおりに実行されない場合の評価では$\alpha = 1.0, 1.25, 2.0, 5.0, 10.0$ のときのジョブミックスを用いる.

シミュレーションに用いたパラメータを表 2 に示す.

表 2 シミュレーションパラメータ

パラメータ名	パラメータ値
電源停止処理に要する時間t_{halt}	33 秒
電源停止処理に要する消費電力p_{halt}	180W
電源再投入処理に要する時間t_{boot}	30 秒
電源再投入処理に要する消費電力p_{boot}	180W
電源停止時の消費電力p_{stop}	0W
待機中の消費電力p_{wait}	180W
状態監視間隔t_{itr}	1 秒
電源停止基準待機時間T_{thr}	335 秒
既存方式における電源停止までの待機時間	1 秒
計算ノード数	160 台

シミュレーションにおける計算ノードのパラメータは，電源停止処理と再投入処理に要する時間t_{halt}，t_{boot}と消費電力p_{halt}，p_{boot}，待機中の消費電力p_{wait}，電源停止中の消費電力p_{stop}となる．これらは"SORA-PP"における計測結果を元に値を決定した．また，計算ノード数は"SORA-PP"において表 1 のジョブ情報を採取した際に使用されていた計算ノード数と同じ値とした.

JSCAPS において設定が必要なパラメータは電源停止基準待機時間T_{thr}と計算ノード状態監視間隔t_{itr}となる．JSCAPS における電源停止基準待機時間T_{thr}は以下のように求めた.

$$T_{thr} = t_{halt} + t_{stop} + t_{boot}$$

$$T_{thr} > \frac{E_{stop}}{p_{wait}} = \frac{p_{halt} \cdot t_{halt} + p_{stop} \cdot t_{stop} + p_{boot} \cdot t_{boot}}{p_{wait}}$$

$$= \frac{180 \times 33 + 0 \times t_{stop} + 180 \times 301}{180} = 334$$

よってT_{thr}=335 秒とした．計算ノード状態監視間隔t_{itr}は最小値の 1 秒とした.

既存方式における設定パラメータである，電源停止までの待機時間は最小値の 1 秒とした．これにより既存方式で待機状態となると必ず電源停止処理が開始されるようになり，電源停止，再投入処理のために余分に消費される時間的, 電力的なコストについて JSCAPS との差が明確となる.

4.4 評価結果

4.4.1 ジョブが実行予定時刻どおりに実行される場合における省電力効果

ジョブが実行予定時刻どおりに実行された場合における省電力効果の評価として，式(2)において$\alpha = 1.0$ とした高稼働率ジョブミックス，低稼働率ジョブミックスに対して JSCAPS と既存方式を適用し，ジョブ未実行中の消費電力量と平均ノード稼働率を計測した.

高稼働率ジョブミックスにおける計測結果を表 3 に，低稼働率ジョブミックスにおける計測結果を表 4 に示す.

表 3 JSCAPS の省電力効果（高稼働率）

	JSCAPS	既存方式	省電力化なし
ジョブ未実行中消費電力量	75MWs (75%減)	208MWs (31%減)	303MWs
平均ノード稼働率	97.9%	96.4%	97.9%

※ ( )内は省電力化なしから削減された割合

表 4 JSCAPS の省電力効果（低稼働率）

	JSCAPS	既存方式	省電力化なし
ジョブ未実行中消費電力量	77MWs (89%減)	127MWs (82%減)	715MWs
平均ノード稼働率	50.3%	50.3%	50.4%

※ ( )内は省電力化なしから削減された割合

表 3 から JSCAPS と既存方式を比較すると以下のことが確認できる.

- ジョブ未実行中の消費電力量について省電力化なしからの削減量を比較すると，既存方式より JSCAPS

が大きい.

- 平均ノード稼働率について省電力化なしと比較すると，JSCAPS では低下していないのに対し，既存方式は 1%以上低下した.

また，表 4 から JSCAPS と既存方式を比較すると以下のことが確認できる.

- 表 3 と同様に，ジョブ未実行中の消費電力量について省電力化なしからの削減量は，既存方式より JSCAPS が大きい.
- 平均ノード稼働率について省電力化なしと比較すると，JSCAPS と既存方式は 0.1%しか低下しなかった

高稼働率，低稼働率に関わらずに，JSCAPS は既存方式よりも平均ノード稼働率が大きく，かつ削減した消費電力量も大きい．このことから，JSCAPS は既存方式よりも効率的な省電力化を実現できているといえる.

ジョブ未実行中の消費電力量について JSCAPS と既存方式との違いを検証するために，電源停止処理，再投入処理を行った回数と処理中の消費電力量，待機中の消費電力量を計測した．高稼働率ジョブミックスにおける計測結果を表 5 に，低稼働率ジョブミックスにおける計測結果を表 6 に示す.

表 5　電源停止処理，再投入処理状況（高稼働率）

	JSCAPS	既存方式
電源停止処理回数	1,000 回	2,540 回
電源再投入処理回数	956 回	2,538 回
電源停止処理, 再投入処理中の消費電力量	58MWs	153MWs
待機中の消費電力量	17MWs	55MWs

表 6　電源停止処理，再投入処理状況（低稼働率）

	JSCAPS	既存方式
電源停止処理回数	1,119 回	1,782 回
電源再投入処理回数	1,222 回	1,891 回
電源停止処理, 再投入処理中の消費電力量	72MWs	112MWs
待機中の消費電力量	5MWs	15MWs

表 5，表 6 から，ジョブ未実行中の消費電力量について JSCAPS が既存方式より削減できた原因は以下になると考えられる.

- 電源停止処理，再投入処理中の消費電力量について
  電源停止処理，再投入処理中の消費電力量は処理回数に依存する．既存方式は待機状態となるたびに電源を停止しているのに対し，JSCAPS では必要でないときには電源を停止しない．この結果 JSCAPS の消費電力量は既存方式よりも小さくなった.
- 待機中の消費電力量について
  JSCAPS における待機中の消費電力量は，電源を停止する必要がないと判断された場合の消費電力量と見られ．既存方式における待機中の消費電力量は，電源再投入処理が完了して次のジョブの実行が開始されるまでの間待機している他の計算ノードの消費電力量と見られる．電源再投入処理を開始するタイミングの違いが待機中の消費電力量に影響したと考えられる.

高稼働率ジョブミックスにおける平均ノード稼働率については，既存方式では電源再投入処理が完了するまで次のジョブの実行開始時刻が遅れることから平均ノード稼働率が 1%以上低下したと考えられる．低稼働率ジョブミックスでも同様の影響が出ていると考えられるが，平均ノード稼働率が低いため影響は小さいと考えられる.

4.4.2 ジョブが実行予定時刻どおりに実行されない場合における省電力効果

ジョブが実行予定時刻どおりに実行されなかった場合における JSCAPS の省電力効果を確認するために，式(2)において $\alpha = 1.0, 1.25, 2.0, 5.0, 10.0$ とした高稼働率ジョブミックス，低稼働率ジョブミックスを作成し，JSCAPS を適用した際のジョブ未実行中の消費電力量，平均ノード稼働率を計測した.

ジョブ未実行中の消費電力量について，高稼働率における α との関係を図 5 に，低稼働率における α との関係を図 6 に示す． なお， $\alpha = 1.0$のときジョブは実行予定時刻どおりに実行されていることになる.

高稼働率ジョブミックスにおいてジョブ未実行中の消費電力量は， $\alpha = 10.0$ のとき $\alpha = 1.0$ に比べて最大 116%増加しており，低稼働率ジョブミックスにおいても $\alpha = 10.0$ のとき最大 34%増加している.

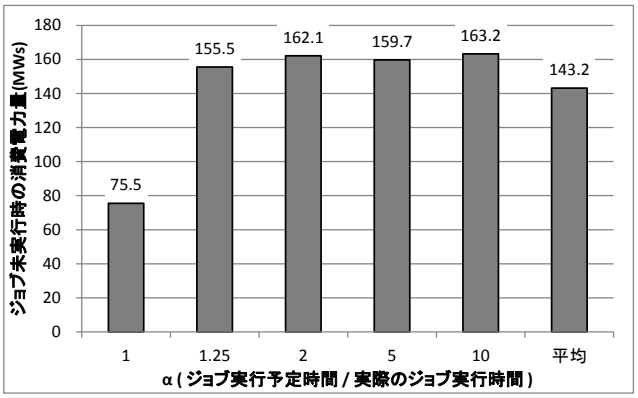

図 5　ジョブ未実行中の消費電力量の変化（高稼働率）

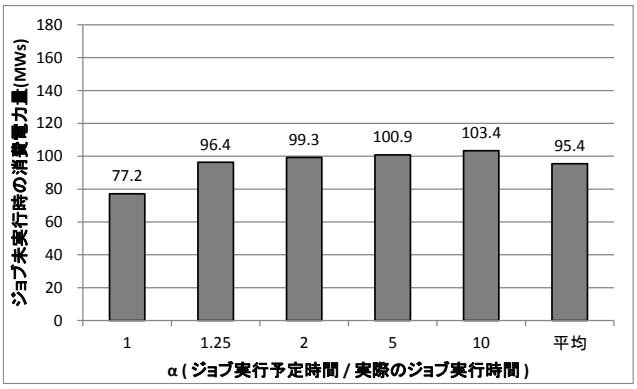

図 6　ジョブ未実行中の消費電力量の変化（低稼働率）

ジョブ未実行中の消費電力量が最大となったときの省電力効果を評価するため，$\alpha = 10.0$となる同じジョブミックスに対して既存方式を適用した場合のジョブ未実行中の消費電力量と平均ノード稼働率も測定した．高稼働率ジョブミックスにおける JSCAPS との比較結果を表 7 に，低稼働率ジョブミックスにおける比較結果を表 8 に示す．

表 7　省電力効果の比較（高稼働率）

	JSCAPS	既存方式
ジョブ未実行中の消費電力量	163MWs	201MWs
平均ノード稼働率	97.1%	96.1%

表 8　省電力効果の比較（低稼働率）

	JSCAPS	既存方式
ジョブ未実行中の消費電力量	103MWs	127MWs
平均ノード稼働率	50.3%	50.3%

表 7, 表 8 から，ジョブ未実行中の消費電力量は JSCAPS が既存方式よりも小さく，平均ノード稼働率は高いか同じであることがわかる．このことから，実行予定時刻どおりにジョブが実行されない場合でも JSCAPS の省電力効果は既存方式よりも高いといえる．

平均ノード稼働率について，高稼働率における α との関係を図 7 に，低稼働率における α との関係を図 8 に示す．計測結果から，高稼働率，低稼働率いずれも α が変化しても平均ノード稼働率は 1%未満しか変化しないことがわかった．

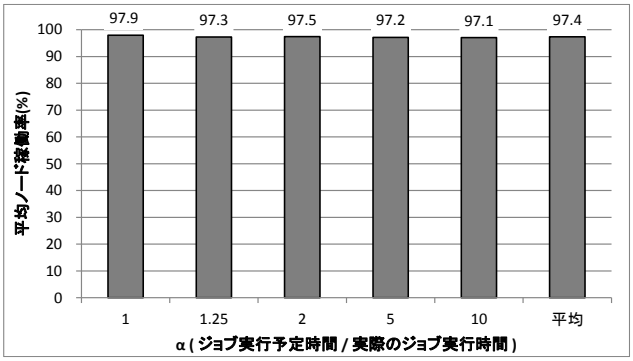

図 7　平均ノード稼働率の変化（高稼働率）

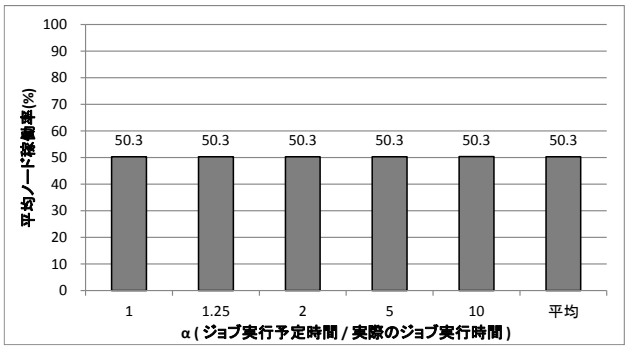

図 8　平均ノード稼働率の変化（低稼働率）

4.5 考察

ジョブが実行予定時刻どおりに実行された場合, JSCAPS は既存方式よりも効率的な省電力効果を得られることが確認された．これは JSCAPS が持つ以下の特長によるものと考えられる．

- 電源停止基準待機時間T_{thr}に基づいて電源停止に要するコストを計算することで，余分な電源停止を行わない．
- 待機状態が始まった直後に電源を停止することで，余分な待機電力を消費しない．
- 電源再投入処理が完了すると同時に次のジョブの実行が開始できるように電源再投入処理を開始することで，他の計算ノードに余分な待機をさせない．

またジョブが実行予定時刻どおりに実行されなかった場合，JSCAPS の省電力効果に影響が出るが，消費電力量の削減量を既存方式と比較すると JSCAPS の削減量が大きいことが確認された．

5. おわりに

本論文では，バッチジョブスケジューリングに基づく省電力化のための自動電源制御方式である JSCAPS について述べ，Slurm や LSF といったバッチジョブスケジューラで提供される既存の自動電源制御方式と省電力効果について比較した．比較した結果，ジョブが実行予定時間通りに実行される場合，されない場合に関わらず，JSCAPS は既存方式よりも省電力効果が高いことが確認された．

今後は"SORA-PP"において実行されたジョブ情報を元により長期的，多様なパターンでの評価を進め，実運用上での適用を目指す予定である．

謝辞 本研究で使用したジョブミックスは，JAXA スーパーコンピュータ"JSS2"で実際に投入されたジョブ情報を使用して作成した．

参考文献

[1] “Slurm Power Saving Guide”. https://slurm.schedmd.com/power_save.html.
[2] “IBM Spectrum LSF”. https://www.ibm.com/support/knowledgecenter/en/SSWRJV_10.1.0/lsf_welcome/lsf_kc_eas.html.
[3] “JAXA JSS2”. https://www.jss.jaxa.jp/.