Vol.2015-HPC-149 No.1
2015/6/26

ジョブスケジューリングにおける スケジューリング空間の評価

関澤 龍一[1,a)]　宇野 篤也[2]　山本 啓二[2]　若林 大輔[3]　庄司 文由[2]

概要：
近年，HPC システムでは高並列化やノード間ネットワークの複雑化により，ジョブスケジューリングのコストが大きくなってきている．「京」のように常時数百本のジョブがキューイングされているようなシステムでは，システムを効率的に運用するためにはジョブスケジューリングの効率化が必須である．スケジューリングのコストはスケジューリングの対象空間のサイズに大きく依存する．本研究では，スケジューリング対象空間を制限することで，ノード利用率の低下を抑えつつ，スケジューリングコストを減らすことができないか検討を行った．検討の結果，バックフィルを使用した場合，スケジューリング対象空間をジョブの最大指定経過時間より少し長く設定することで，ノード利用率の低下を抑えつつスケジューリングコストを減らすことができることがわかった．

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をはじめとする HPC システムでは，一般的に大小様々なジョブが同時に多数投入される．これらジョブの時間規模やノード規模を考慮して計算資源を割り当てるのがジョブスケジューラの役割である．どのジョブにどのノードを割り当てるかは，スケジューリングアルゴリズムによって決まる．そのため，計算資源を効率的に活用するためには，使用するスケジューリングアルゴリズムが重要となるが，それと同時にスケジューリングに要するコストも無視できないパラメータとなる．なぜなら，いくら効率の良いアルゴリズムを採用したとしても，スケジューリングに時間がかかりすぎると実際の運用では役に立たないからである．

HPC システムは用途の拡大や計算速度性能の向上に伴い，高並列化やノード間ネットワークが複雑化する傾向にある．例として，TOP500 [1] の上位にランキングした HPC システムのコア数を挙げる．2009 年 11 月の TOP500 では，1 位から 10 位までにランクインした HPC システムの平均コア数は約 135,000 であったが，5 年後の 2014 年 11 月の TOP500 では約 825,000 であった．5 年間で 6 倍以上にコア数が伸びており，HPC システムが高並列化の傾向にあることが分かる．2014 年 11 月の TOP500 ランキングで 4 位のスーパーコンピュータ「京」（以下「京」）の場合，コア数は 705,024，ノード数は 82,944 で，ノード間ネットワークは 6 次元ネットワークである Tofu インターコネクトを採用している [2]．加えて「京」で実行されるジョブは，2012 年 10 月から 2013 年 9 月の 1 年間では多い月で約 60,000 件と非常に多い [2] [3]．HPC システムの複雑化に伴い，ジョブスケジューリングも複雑化し，そのコストも大きくなっている．ここでいうジョブスケジューリングにおけるコストとは，ジョブスケジューリング 1 回に要する時間である．スケジューリングコストは，スケジューリングアルゴリズムやスケジューリング対象空間，スケジューリング対象のジョブなどで決まる．特に「京」のようにジョブやノード数が多く，さらにノード間ネットワークも複雑なシステムでは，スケジューリングコストを低くすることは非常に重要である．

本稿では，スケジューリングコストを低減させる手段としてスケジューリング対象空間に着目し，ノード利用率をなるべく落とさずに，スケジューリングコストを下げることができないか検討を行った．その検討内容および結果について報告する．

[1] 富士通株式会社
Fujitsu Limited
[2] 理化学研究所計算科学研究機構
RIKEN Advanced Institute for Computational Science
[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)] r_sekizawa@jp.fujitsu.com

© 2015 Information Processing Society of Japan

2. ジョブスケジューリングコスト低減方法の検討

2.1 スケジューリングコスト

ジョブスケジューリングのアルゴリズムに関係なく，スケジューリングコストを低減させる方法として，スケジューリング探索数を少なくする方法がある．通常，ジョブスケジューラはジョブが投入された際，そのジョブが要求するノード数や指定経過時間に基づいて，そのジョブに割り当てるノードを探す．そのため，この時の探索数を少なくすることができれば，スケジューリングコストを下げることができる．しかし，無暗に探索数を減らすとノード利用率の低下を招く場合がある．例えば，我々が提案している区間ジョブスケジューリング法の評価では，比較した中でもっともコストを低くしたパターンが，もっとも悪い利用率となった [4]．このように，スケジューリングコストの低減と同時にノード利用率が低下しては意味がない．ノード利用率の低下をできるだけ抑えた上で，スケジューリングコストを低減させる必要がある．

2.2 スケジューリング対象空間

ノード利用率の低下をできるだけ抑えた上で，スケジューリングコストを低減させる方法として，スケジューリング対象空間を調整することを検討した．ここでいうスケジューリング対象空間は，時間軸方向における探索数と言うこともできる．言うまでもなく，ノード利用率とスケジューリング対象空間の関係を評価するためには，スケジューリングアルゴリズムの動作を考慮に入れる必要がある．

以下では，スケジューリング対象空間の長さを T，ジョブで指定可能な経過時間の最大値を t_0 と表すこととする．

一般的に，遠い未来までスケジュールを組めば組むほどスケジューリングコストは高くなる．極端な例を挙げると，3 日間先までスケジューリングを行う場合と 30 日間先までスケジューリングを行う場合ではスケジューリングコストは 10 倍違うことになる．ジョブに割り当てたノードを固定しない場合，スケジューリングはノードの使用状況が変化する毎に繰り返される．そのため，仮にひと月近く先までスケジューリングしても，実際にそのジョブが実行されるときには状況はまったく異なることになる．つまり，遠い未来にスケジュールしても全く意味が無いものとなる．

このように T を長く設定しても，探索数が増えるだけで意味をもたない．逆に T を無暗に短くしても，探索数が減りノード利用率が悪化する場合がある．そこで，適切な探索数となる T を設定することで，ノード利用率の低下を抑えつつスケジューリングコストを下げることを考えた．

2.3 設定範囲の検討

スケジューリング対象空間 T の設定範囲について検討した．今回は，「京」や他のサイトで広く使われているバックフィルを使用した場合のスケジューリング対象空間の設定範囲について評価することにした．前提として，実行中のジョブは使用ノードが確定し，実行待ちジョブは全て仮割り当てとする．スケジューリングアルゴリズムには，バックフィルの他 FCFS（First-Come and First-Served）を使用した．バックフィルは空きノードがある場合にジョブの実行順序を入れ替えてスケジューリングを行うアルゴリズムで，FCFS はジョブの投入順に優先順位を決めるアルゴリズムである．

バックフィルでスケジューリングする場合，実行待ちジョブのスケジューリング状況で全体のスケジューリング状況が大きく変わることがある．例えば，2 時間先に実行待ちジョブがスケジューリングされると，それ以降はノードが空いていても指定経過時間が 2 時間未満のジョブしかそれよりも前にスケジューリングできなくなる．これを考慮すると，スケジューリング対象空間が $T = t_0$ の前後の場合でスケジューリング状況が大きく変わる可能性が考えられる．**図 1** に，T が $T < t_0$ の場合と $T \geq t_0$ の場合に，ノード数と指定時間が異なるジョブ 5 つをスケジューリングした例を示す．矩形がジョブを，矩形内の数字はジョブの投入順をそれぞれ表している．

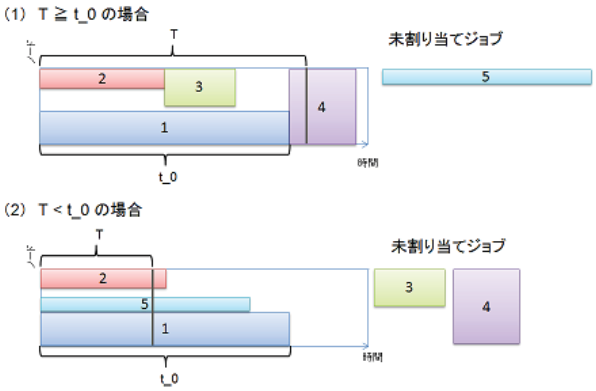

図 1 T の大きさを変えた場合のスケジューリング例

(1) の場合，ジョブ 1 からジョブ 4 までが順にスケジューリングされ，ジョブ 5 が未スケジューリングの状況となる．一方，(2) の場合はジョブ 3 と 4 はスケジューリングされず，ジョブ 5 がバックフィルによりスケジューリングされている．これは T の後方にはスケジューリング探索ポイントが存在しないためである．結果，ジョブ 5 がジョブ 3 と 4 を追い越してスケジューリングされる．このように，$T = t_0$ の前後でスケジューリング状況が大きく変わる場合があることが分かる．

そこで，スケジューリング対象空間を変化させた場合に，スケジューリング状況がどのように変わるか調査した．

© 2015 Information Processing Society of Japan

3. 評価

ジョブスケジューラシミュレータを使い，スケジューリング対象空間を変化させた場合の，スケジューリング状況の変化を評価した．評価には「京」の統計情報から生成したジョブミックスを使用した．

3.1 評価環境

3.1.1 ジョブミックス

ジョブミックスは様々なジョブの集合体で，個々のジョブ毎に，投入時刻 (投入間隔)，ノード数，指定経過時間，実実行時間の 4 つのパラメータをもつ．ジョブミックス毎のばらつきを考慮し，ランダムに作成した複数のジョブミックスを使って統計的に評価した．今回は，「京」で 2013 年 12 月から 2014 年 5 月までに投入されたジョブ情報をもとに生成したジョブミックスを使用した．

3.1.2 ジョブスケジューラシミュレータ

評価に用いるジョブスケジューラシミュレータについて説明する．シミュレータはジョブミックスを読み込み，ジョブの投入時刻に従ってジョブの投入，ノードの確保，ジョブの実行と終了，ノードの開放といった一連のジョブスケジューラの動作をシミュレートする．本シミュレータでは，ジョブスケジューリングは，新規にジョブが投入されたときと実行中のジョブが終了したときに行われる．また，シミュレータはジョブ毎にジョブが実行待ち状態から実行へ遷移した時刻およびジョブが確保したノード情報を出力する．ジョブ毎の出力結果から，ノード利用率とジョブ投入から実行開始までの待ち時間をそれぞれ計算する．今回の評価では，スケジューリングアルゴリズムには FCFS とバックフィルを使用し，連続した 1 次元のノードをジョブに割り当てることとした．

3.1.3 シミュレーションパラメータ

投入するジョブの最長指定経過時間 t_0 は，「京」と同じ 24 時間とした．これは，評価に使用するジョブミックスが「京」の統計情報をもとに作成されているためである．

今回の評価では，スケジューリング対象空間を変化させた時のノード利用率とジョブの実行待ち時間を比較した．このスケジューリング対象空間の大きさは，スケジューリングコストに相当する．また今回の評価では，スケジュール空間を 1 時間間隔で区切ったパターン（ **図 2** ）を使用した．例えば T =24 時間の場合，0 から 24 時間まで 1 時間毎に 25 箇所の区間が設けられるためスケジューリングコストは 25 となる．

スケジューリング対象空間 T を，20 時間から 50 時間まで 1 時間ずつ順に増やしていき，それぞれの値におけるノード利用率とジョブの実行待ち時間を評価した．シミュレーション時間短縮のため，システムの最大ノード数を「京」のノード数である 82,944 ノードから 768 ノードに縮小し，

図 2 評価に使用するスケジュール区間

ジョブミックスのノードも同様の比率で縮小し，小数点以下の端数については切り上げた．シミュレーション期間は 14 日間とし，ジョブミックスの密度は 85%,90%,95%の 3 種類を用意した．ジョブミックスの密度とは，システムの全計算資源に対する投入ジョブのノード時間積の総和の割合を表している．ジョブミックスは密度毎に 20 セット用意し，スケジューリング対象空間毎にそれぞれ 20 回のシミュレーションを行い，平均値を評価した．

3.2 シミュレーション結果

ノード利用率とジョブ実行待ち時間の観点をもとに評価結果について述べる．結論から述べると，$T = t_0$(24 時間) 以降でもノード利用率とジョブ実行待ち時間に差がみられた．

3.2.1 ノード利用率

図 3 に 1 日目から 14 日目までのノード利用率を示す．

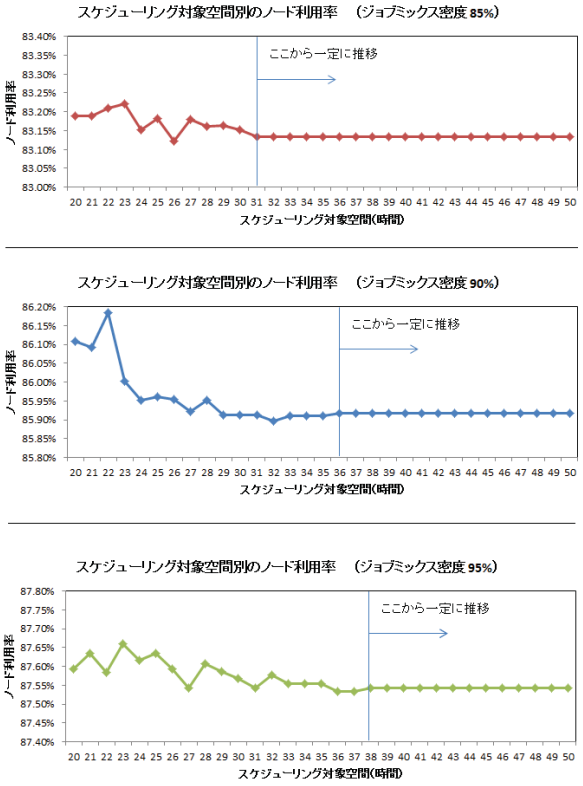

図 3 スケジューリング対象空間別のノード利用率（ジョブミックス密度 85%,90%,95%）

まず密度 85%については，T = 23 時間でのノード利用

© 2015 Information Processing Society of Japan

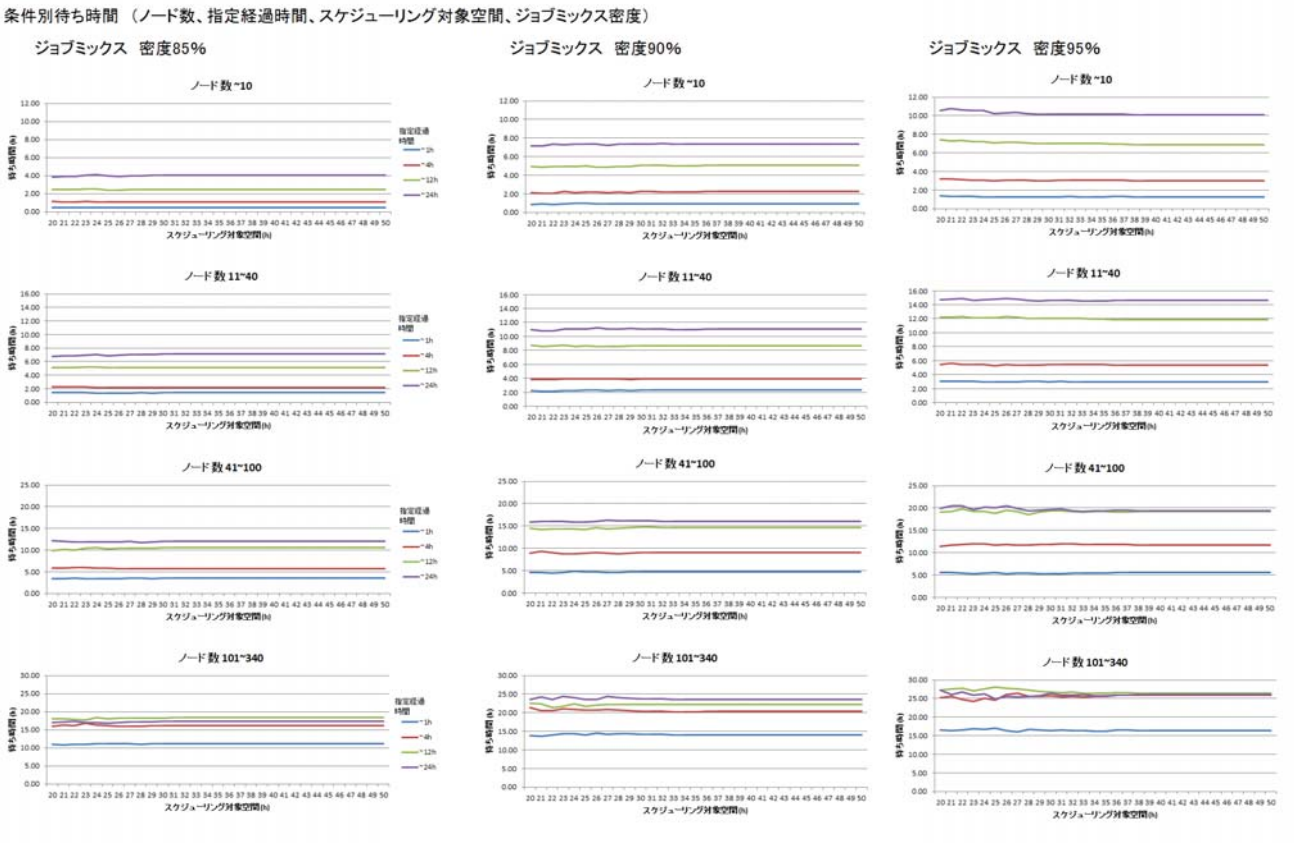

図 4　条件別ジョブ実行待ち時間

率がもっとも高く 83.22%であった．一方もっともノード利用率が低かったのは 26 時間で 83.12%であった．$T < t_0$ の方がノード利用率は高く，31 時間以降は 83.13%と全て同じ値となった．

次に密度 90%では，$T = 22$ 時間がもっともノード利用率が高く 86.18%で，32 時間がもっともノード利用率が低く 85.90%であった．22 時間をピークに 36 時間までは徐々にノード利用率が減少しており変動幅は約 0.28%であった．37 時間から 50 時間時間までは，36 時間のノード利用率と完全一致しており 85.92%であった．

そして密度 95%の場合は，$T = 23$ 時間がもっともノード利用率が高く 87.66%で，36,37 時間がもっともノード利用率が低く 87.53%であった．変動幅は約 0.13%となった．24 時間から低下傾向に推移しており，27 時間のときにノード利用率が大きく低下した．38 時間以降は値が 87.54%と一定した推移となっている．

3.2.2　ジョブ実行待ち時間

図 4 に，スケジューリング対象空間に沿って横に並べたノード規模と指定経過時間別のジョブ実行待ち時間を示す．この図は，各ジョブミックスに 1 日目から 14 日目までの間に実行を開始したジョブのジョブ実行待ち時間の平均値である．

密度 85%に着目すると，ノード数が 100 以下までの範囲では，ジョブ実行待ち時間の増減はあまり無い．若干だが長時間ジョブでは，スケジューリング対象空間の増加に比例してジョブ実行待ち時間が増加している．ノード数が 101 以上 340 以下においては，$T = 20$ 時間から 24 時間まではジョブ実行待ち時間に差異が発生している．25 時間以降はほぼ横ばいの数位となっており，ノード利用率と同じく 31 時間以降は全て同じ数値となっている．

次に密度 90%は，ノード数が 10 以下の範囲において，微少であるが全時間でジョブ実行待ち時間が伸びている．ノード利用率が高かった $T = 22$ 時間では，ノード数 101 以上 340 以下のジョブ実行待ち時間が別空間よりも短い数値となっている．

最後に密度 95%について述べる．ノード数が 10 以下，11 以上 40 以下の範囲においては，スケジューリング対象空間の増加と逆行する形でジョブ実行待ち時間が短縮している．一方ノード時間積が大きいジョブについては，$T = 38$ 時間以降の一定値になるまでジョブ実行待ち時間に差異が発生している．ノード利用率もジョブ実行待ち時間と同様に一定になるまでは前後している．

総じて，スケジューリング対象空間の増加におけるジョブ実行待ち時間の変化は，あまり大きく無い．ノード数が 101 以上 340 以下の範囲でも 2 時間程度の変動幅となっており，ノード数が 100 以下の範囲においては最大でも 1 時間程度の差となっている．

まとめると，ノード利用率とジョブ実行待ち時間の評価

© 2015 Information Processing Society of Japan

結果から以下の傾向が見られた．

- $T \geq t_0$ であってもノード利用率とジョブ実行待ち時間は異なる値となった．
- T がある程度の長さに達して以降は，ノード利用率とジョブ実行待ち時間が完全一致した．
- 0.3%に満たない僅差ではあるが，$T < t_0$ の方が $T \geq t_0$ よりも高いノード利用率となった．

ノード利用率とジョブ実行待ち時間の差は，スケジューリング時の実行順序のずれにより生じたものと考えられる．そこで，実行順序がずれる前後でのスケジューリングの変遷を分析しこれら 3 点について考察を行うことにした．

4. 考察

4.1 $T \geq t_0$ でノード利用率が異なる原因

シミュレータ内部のスケジューリング状況を解析し，$T \geq t_0$ でノード利用率が異なる原因を調査した．ノード利用率が異なる場合について調査した結果，ノード数が大きいジョブと実行中ジョブの割り当て箇所によっては，同じジョブミックスでもスケジューリング状況に大きな差が生じる場合があることが判明した．

図 5 は，スケジューリング対象空間が T =25,26 時間でノード利用率に差が生じる場合のスケジューリング状況を示したものである．矩形はジョブを表し，内部の数字はジョブの投入順を示す．

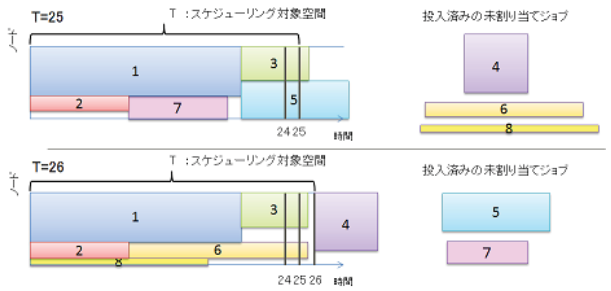

図 5 ノード利用率に差が生じる場合のスケジューリング状況

ジョブ 1,2,3 までの割り当ては T =25,26 時間の両方で同じ結果となる．割り当てに差が生じ始めるのはジョブ 4 からである．

$T = 25$ 時間では，ジョブ 4 を割り当てることができる箇所はない．ジョブ 5 はジョブ 3 と同じ開始時間に割り当てることができるが，次のジョブ 6 を割り当てることができない．以降はバックフィルとしてジョブ 7 が割り当てられ，ジョブ 8 は未割り当てのジョブとして残る．

$T = 26$ 時間だと，ジョブ 4 を割り当てることができる．しかしジョブ 4 が蓋となりジョブ 5 が時間軸をオーバーしてしまうので，ジョブ 3 と同時刻での割り当てができなくなってしまう．さらにジョブ 5 が未割り当てとなった分，ジョブ 6 がちょうど埋まる．代わりにジョブ 7 を割り当てる空間がなくなり，ジョブ 8 を実行できる空間が余る．そしてジョブ 8 の実行が開始される．

ジョブ 4 の割り当て有無によりバックフィル可能な空間に時間方向の差ができ，$T = 26$ 時間のみでジョブ 5 への割り当てが阻害された結果，ジョブ 8 の実行タイミングにずれが起きている．この実行順序のずれがノード利用率の差につながった．

このように，スケジューリング探索空間の大きさでスケジューリング状況に差が生まれ，ノード利用率に違いが生じる場合があること分かった．

4.2 T がある程度の長さに達して以降，結果が完全一致する原因

図 5 の例では，スケジューリング状況の差はスケジューリング対象空間の最後尾に大きなジョブがスケジューリングされ，且つ，前方に実行中のジョブが存在することにより発生していた．T を変化させる以上，この差は必ず発生する．また一方で，T がある程度の長さに達して以降は，ノード利用率とジョブ実行待ち時間が完全一致するとの結果が得られている．以上から，この差が発生してもジョブの割り当てが阻害されない最小の T があると思われる．

ここで，スケジューリング対象空間が 48 時間以上 ($T \geq 48$) の場合を考えてみる．48 時間以上の空間があれば，経過時間が 24 時間のジョブの実行開始直後であっても 24 時間以上のバックフィルを行える空間を確保できる．先ほどと同様に，スケジューリング対象空間の最後尾に大きなジョブがスケジューリングされた状態をモデル化し，T =48,49 時間での差を比較する．

図 6 に比較した 2 パターンのモデルを示す．

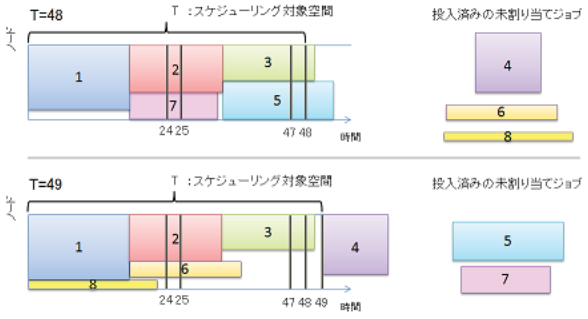

図 6 T=48,49 を例としたスケジューリング結果の差異

ジョブ 1,2,3 までの割り当ては両方で同じ結果となる．割り当てに差が生じ始めるのはジョブ 4 からである．

$T = 48$ 時間では，ジョブ 4 を割り当てることができる箇所はない．ジョブ 5,7 は割り当てることができるが，ジョブ 6,8 を割り当てることができない．

$T = 49$ 時間では，ジョブ 4 を割り当てることができる．しかしジョブ 4 が蓋となった結果，ジョブ 6,8 は割り当てることができるが，ジョブ 5,7 を割り当てることができな

© 2015 Information Processing Society of Japan

い．そしてジョブ 8 の実行が開始される．

T =25,26 時間の例と同じく，一方でのみ実行されているジョブが存在しておりノード利用率に差異が発生している．つまり，例え $t_0 \times 2$ よりも十分に大きなスケジューリング対象空間を設けた場合であっても，投入されているジョブの条件や順序によっては，ノード利用率に差が発生し得ることになる．

しかし，今回実施したシミュレーションでは T が 48 時間以下であってもノード利用率とジョブ実行待ち時間が完全に一致する結果が得られている．T の値が大きくなるとスケジューリング対象空間の最後尾と実行中のジョブの間も大きくなり，ノード利用率が変わる状況になるには，多くのジョブが必要となる．言い換えると，ジョブ数が少ない場合にはノード利用率の変化が起こりにくいといえる．以上から，ジョブが多ければ多いほど，ノード利用率に差が生じる可能性が高くなることになる．これは，ジョブミックスの密度が増えるほど完全一致し始める T が後退していた現象に整合する．

4.3 $T < t_0$ の方が $T \geq t_0$ よりも高いノード利用率となった原因

図 1 におけるモデルで示した通り，この T の差はジョブの実行順序に影響を与える．$T < t_0$ ではノード数の小さいジョブが早く実行され，$T \geq t_0$ ではノード数の大きいジョブが早く実行される．従って投入されるジョブの傾向の差が，最終的なノード利用率の差として現れたと考えられる．投入するジョブの傾向を変えて評価しなおすことで，T と t_0 の差におけるノード利用率との関係性が見えるかもしれないが，これは今後の課題である．

5. 関連研究

今回の研究で，スケジューリング対象空間はバックフィルに大きく影響を受けることが分かった．バックフィルは以前からジョブスケジューリングにおける有用なアルゴリズムとして知られており，バックフィルを交えたスケジューリングの評価も過去に多数行われている．Rodero らは複数システムを跨いだスケジューリングにおいて有効なバックフィル手法 (Grid Backfilling) の評価を行った [6]．Keleher らは 4 種類のバックフィル手法をそれぞれ適用した場合における，ジョブスケジューリング性能の比較を行っている [7]．バックフィルさせるジョブの優先順を変えて評価した例もある [8]．

これらの研究では，それぞれが自ら提案したバックフィル手法自体を評価していた．一方，本研究ではバックフィルを使用した場合における，スケジューリング対象空間の長さとノード利用率の関係について評価しており，過去の研究と比較して独自性をもっている．

6. まとめ

本稿では，スケジューリングコストを低減する手段としてスケジューリング対象空間に着目し，ノード利用率の低下を抑えつつスケジューリングコストを低減する手法について検討を行った．今回の評価では，バックフィルを使用した場合のスケジューリング対象空間の長さとノード利用率の関係について，シミュレーションで検証を行った．スケジューリング対象空間 T をジョブの最大指定経過時間 t_0 よりも長く指定した場合，ノード利用率に大きな差は発生しないことが分かった．ノード利用率に差が生じる原因を調査したところ，スケジューリング対象空間の最後尾へのジョブの割り当て状況と実行中ジョブの位置関係から，バックフィル時に時間方向での制限が発生することが原因であることが分かった．これは，T の長さに関係なく発生する可能性のある現象で，ジョブミックスの密度によりその発生確率は変化する．ジョブミックスの密度が高い場合に発生確率は高くなるが，T を長く設定することでその発生確率を低くすることができる．ジョブミックスの密度が 85%,90%,95%の場合では，ノード利用率の差は 0.3%程度であった．

以上の結果からバックフィルを用いた場合は，スケジューリング対象空間 T をジョブの最大指定経過時間 t_0 より少し大きく設定する場合に，ノード利用率の低下を抑えつつスケジューリングコストを低減できることが分かった．スケジューリング対象空間はスケジューリングアルゴリズムに大きく影響を受ける．今後は，他のアルゴリズムについても評価を行っていきたいと考えている．

参考文献

[1] TOP500: TOP500 Supercomputer Sites, Top500.org (online), available from (http://www.top500.org) (accessd 2015-5-20).

[2] 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., Yokokawa,M.: The K computer Operations: Experiences and Statistics., International Conference on Computational Science ICCS2014, pp. 576-585 (2014).

[3] 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).

[4] 山本啓二，宇野篤也，関澤龍一，若林大輔，庄司文由：区間スケジューリングを用いたジョブスケジューリングの性能評価，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).

[5] 山本啓二，宇野篤也，関澤龍一，若林大輔，庄司文由：区間ジョブスケジューリング法へのファイルステージング導入に伴う性能評価，情報処理学会研究報告，第 148 回ハイパフォーマンスコンピューティング研究会，HPC148 (2014).

[6] Rodero, I., Guim, F., Corbalan, J., Goyeneche, A. (2008). The grid backfilling: a multi-site scheduling architecture

© 2015 Information Processing Society of Japan

with data mining prediction techniques. In Grid Middleware and Services (pp. 137-152). Springer US.
[7] Keleher, P. J., Zotkin, D., Perkovic, D. (2000). Attacking the bottlenecks of backfilling schedulers. Cluster Computing, 3(4), 245-254.
[8] Baraglia, R., Capannini, G., Pasquali, M., Puppin, D., Ricci, L., Techiouba, A. D. (2008). Backfilling strategies for scheduling streams of jobs on computational farms. In Making Grids Work (pp. 103-115). Springer US.

© 2015 Information Processing Society of Japan