社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2006－HPC－107 (15)
2006／8／1

グリッド環境に対応したジョブスケジューリング法に対するマイグレーションの評価

岩切 淳一† 山森 一人†† 吉原 郁夫†† 相川 勝†††

† 宮崎大学工学研究科
†† 宮崎大学工学部
††† 宮崎大学工学部 教育研究支援技術センター
〒 889-2192 宮崎県宮崎市学園木花台西 1-1
E-mail: †iwakiri@taurus.cs.miyazaki-u.ac.jp

あらまし グリッド環境でのスタベーションを回避するため，従来のスケジューリング法とマイグレーションを組み合わせた手法について性能評価を行い，最も効率良くスタベーションを回避できる手法について提案する．シミュレーションの結果，従来の LJF/First-Fit (Largest Job First/First-Fit) にマイグレーションを組み合わせたスケジューリングにより，オーバヘッドの増大を防ぎつつスタベーションを回避し，計算機の利用効率，Slowdown Ratio 共に向上させることができた．

Evaluation of Job Scheduling Algorithm Improving Starvation in Grid Environment

Junichi IWAKIRI†, Kunihito YAMAMORI††,
Ikuo YOSHIHARA††, and Masaru AIKAWA†††

† Graduate School of Engineering, Miyazaki University
†† Faculty of Engineering, Miyazaki University
††† Technical Center of Faculty Engneering, Miyazaki University
Miyazaki University 1-1, Gakuen Kibanadai Nishi, Miyazaki, 889-2192, Japan
E-mail: †iwakiri@taurus.cs.miyazaki-u.ac.jp

Abstract We combine migration and conventional job scheduling algorithms to avoid starvation in Grid Environment, and evaluate the performance of the combinations to propose the most efficient combination for starvation. Simulation results showd that the combination of conventional LJF/FF(Largest Job First / First-Fit) with migration could avoid the starvation with small overhead of migration. Furthermore, the LJF/FF with migration could improve both utilization of computational nodes and Slowdown Ratio.

1. はじめに

グリッド環境のような大規模並列分散環境上に投入される，MPI(Message Passing Interface) 等を利用した多数の並列ジョブを効率的に実行するためには，並列ジョブを効率的にスケジューリングする必要がある。

従来の並列ジョブのスケジューリングアルゴリズムとしては，単純な FCFS(First Come First Served)，ジョブキュー内の並列ジョブを要求する計算機の台数によってソートする LJF(Largest Job First) や SJF(Smallest Job First)，ジョブキュー内の実行可能な並列ジョブを探索する First-Fit や Best-Fit 等のアルゴリズムが提案され，それぞれ性能評価が行われている [1]～[3]．しかし，FCFS を除くこれらのスケジューリングアルゴリズムでは，一部の並列ジョブに計算機が長期間に渡って割り当てられない状態 (スタベーション) が発生する可能性がある．グリッド環境の計算機は多数のユーザに共有されているため，スタベーションが発生すると特定のユーザの並列ジョブだけが実行されず，計算機を公平に利用することができない．スタベーションを回避する手法として

表 1　スケジューリングアルゴリズムの構成

algorithm	sorting	search	migration
FCFS	no	no	no
SJF	yes	no	no
LJF	yes	no	no
FCFS/First-Fit	no	yes	no
LJF/First-Fit	yes	yes	no
FCFS/First-Fit with mig	no	yes	yes
LJF/First-Fit with mig	yes	yes	yes

は，Lifka ら [4] や Skovira ら [5] によって Backfilling 法が提案されているが，これは並列ジョブの実行時間があらかじめ分かっている場合を想定しているため，実際のグリッド環境で用いることは難しい．

本稿では，スタベーション回避の方法としてマイグレーションを用いたときの性能を詳しく議論し，各種スケジューリングアルゴリズムと組み合わせた時の有効性を示すことを目的とする．

スケジューリングアルゴリズムの性能をシミュレーションによって評価する場合，現実に近いシミュレーション環境をいかに構築できるかが極めて重要となる．従来のスケジューリングシミュレーションでは，並列ジョブの実行時間やそれらが要求する計算機の台数は一様分布に基づく単純なモデルを用いることが多かった．一方，並列計算機センターの長期間に渡る計算機の利用状況のログから，投入される並列ジョブの持つ特徴が明らかになりつつある．例えば，並列ジョブの特徴として 2 の倍数や 2 のべき乗等の特定の台数の計算機を要求する並列ジョブが多いこと，少数の台数の計算機を要求する並列ジョブが多いこと，要求する計算機の台数が多い並列ジョブは実行時間が長くなる傾向にあること等が確かめられている [1], [6]．本稿では，実際の並列ジョブの特徴をより反映したシミュレーションを行うために，この利用状況のログから作成された並列ジョブのモデルを用いる．

以下，性能評価を行うスケジューリングアルゴリズムについて 2 節で説明を行い，シミュレーションを行うジョブスケジューリングのモデルを 3 節で述べる．4 節でシミュレーションによる性能評価の結果を示し，その結果について考察する．

2. スケジューリングアルゴリズム

性能評価を行うスケジューリングアルゴリズムの基準として，広く一般的に用いられている FCFS (First Come First Served) を用いる．FCFS はジョブキュー内で到着順に並んでいる並列ジョブの中で，先頭のジョブが要求する計算機の台数が現在利用可能なアイドル状態の計算機の台数以下ならば，このジョブに計算機を割り当てる．

本稿で性能評価を行うスケジューリングアルゴリズムは，次の 3 種類の特徴の組み合わせにより構成される．

ジョブのソート:　ジョブキューに投入された並列ジョブをある規則に基づきソートし，ソート後のキューに従ってジョブを投入する．ジョブのソートについては 2.1 節で詳しく述べる．

ジョブの検索:　ソートされたジョブキュー内で，実行可能なジョブをある規則に基づき検索し，最初に条件にマッチしたジョブから順に実行する．ジョブの検索については 2.2 節で説明する．

スタベーションの回避:　ジョブキュー内を検索し条件にマッチしたジョブから，実行を行うことにより，ジョブキュー先頭の並列ジョブが長期間に渡って実行されないスタベーションに陥ることがある．先頭のジョブを実行するために，実行中のジョブをサスペンドして，割り当てる計算機を確保する。この操作をマイグレーションと呼ぶ．スタベーションについては 2.2 節で，マイグレーションについては 2.3 節で詳しく述べる．

本稿で性能評価を行う各スケジューリングアルゴリズムの構成を表 1 にまとめる．

2.1　ジョブのソート

ジョブキューに投入された並列ジョブを要求する計算機の台数に応じてソートするスケジューリング方法について述べる．要求する計算機の台数が多い順にソートを行う LJF(Largest Job First) と，逆に要求する計算機の台数が少ない順にソートを行う SJF(Smallest Job First) を考える．これらの方法でソートされたジョブキュー先頭のジョブが要求する計算機の台数が，現在利用可能なアイドル状態の計算機の台数以下ならば，そのジョブに対して計算機を割り当てる．

LJF を用いる利点として，要求する計算機の台数の多い順に並列ジョブをソートすることで，アイドル状態の計算機の台数を減少させ，計算機の利用効率を大幅に向上させることが合田 [3] により示されている．しかし，要求する計算機の台数の多い順にソートすることで同時に実行される並列ジョブ数が少なくなり，多くのジョブの待ち時間が増大する．また，ジョブキュー先頭のジョブが要求する計算機の台数がアイドル状態の計算機の台数より多くなりやすく，先頭ジョブに計算機の割り当てが行われないブロッキングが発生しやすいことが問題とされている [3]．

一方，SJF は要求する計算機の台数が少ない順に並列ジョブをソートすることで，同時に実行できるジョブの数を多くし，多くのジョブの待ち時間を短縮できる．また，要求する計算機の台数がアイドル状態の計算機の台数より多くなる回数が減少し，ブロッキングの発生を抑えることができる．計算機の利用効率に関しては，LJF ほどの大幅な向上は見られないが，要求する計算機台数が少ないジョブの数が，多数の計算機を要求するジョブの数よりも圧倒的に多い場合に限り，大幅な向上が確認されている [3]．

2.2　ジョブの検索

ブロッキングを回避する方法として，ジョブキュー内の

並列ジョブの中で要求する計算機の台数がアイドル状態の計算機より少ないジョブを検索して，計算機に割り当てる方法がある．本稿では，ジョブキューの先頭から順に実行可能な並列ジョブを検索し，一番最初に見つかったジョブから順に計算機を割り当てる First-Fit と組み合わせたスケジューリングアルゴリズムについて評価を行う．

ジョブの検索を行うことで，ジョブキュー先頭のジョブが実行待ち状態であっても，後続の実行可能な並列ジョブを先に実行することでブロッキングを回避することができる．また，同時に実行される並列ジョブ数も多くなり，待ち時間の改善につながる．しかし，後続のジョブの先行実行を行うことで，要求する計算機台数の多いジョブキュー先頭のジョブが長期間に渡って計算機に割り当てられない状態になりやすい．この状態をスタベーションと呼ぶ．

ジョブキューの検索はブロッキングを回避する有効な手段であるが，グリッド環境においてはスタベーションの発生によって，共有された資源利用の公平性が失われることが問題となる．

2.3 スタベーションの回避

2.2 節で述べたスタベーションを回避するアルゴリズムとしては Backfilling 法 [4], [5] が知られているが，これは並列ジョブの実行時間を事前に得ていなければ用いることができないため，現実的ではない．そこで，First-Fit を採用することで生じるスタベーションを回避するために，実行中の後続ジョブの途中結果をスケジューラに一時保存し，このジョブの実行を停止することでアイドル状態の計算機を確保する．この操作をマイグレーションと呼び，本稿ではこの方法によってスタベーションの回避を行う．図 1 にマイグレーションの動作例を示す．図 1 では，512 台の計算機を要求する Job1 をスタベーションから開放するため，First-Fit により先行実行中の Job2〜Job4 を一時停止させている．

マイグレーションを行う条件は，先頭のジョブが要求する計算機の台数を N_{top}，先行実行中の後続ジョブが使用している計算機の台数を N_{follow}，現在利用可能なアイドル状態の計算機の台数を N_{idle} とすると，式 (1) で定義される．

$$N_{top} \geqq N_{follow} + N_{idle} \quad (1)$$

式 (1) により，First-Fit によって検索・実行された後続ジョブの実行時間が，本来先頭ジョブが要求する計算機の台数が揃うまでに制限され，この間に実行終了していない後続ジョブはマイグレーションにより，次の計算機の割り当てが行われるまでジョブキュー内で待ち状態になる．またマイグレーションされたジョブは，次の計算機の割り当てが行われたときは途中結果からリスタートされる．このように，マイグレーションによって確保された計算機に先頭のジョブを割り当てることで，First-Fit によるスタベーションを回避することができる．

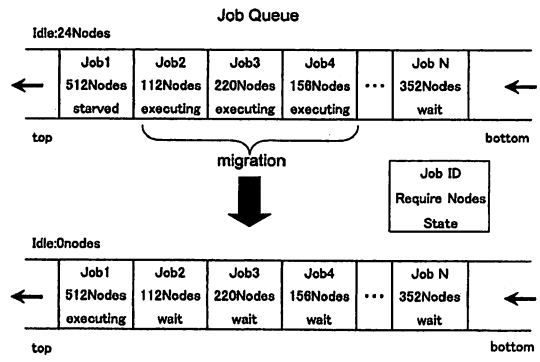

図 1 マイグレーション動作例

ただし，マイグレーションを行うために必要な時間（マイグレーションコスト）が発生するので，これを考慮したシミュレーションを行う必要がある．マイグレーションコストについては 3.4 節で詳しく述べる．

3. シミュレーションモデル

スケジューリングシミュレーションを行うために必要な，グリッド環境，スケジューラ，並列ジョブの特徴，マイグレーションコストのモデルについて述べる．なお，シミュレーション内での単位時間は分単位である．

3.1 グリッド環境

想定するグリッド環境は，計算機クラスタを持つサイト同士を広域ネットワークを介して相互接続したものである．グリッド環境は 10 サイト，1,024 台の計算機で構成され，個々の計算機 $Node_j (1 \leqq j \leqq 1,024)$ は，$Spec_j$ と $SiteID_j$ の 2 つのパラメータを持つ．$Spec_j$ は $Node_j$ の処理能力であり，$SiteID_j$ は $Node_j$ が属する計算機サイトの ID である．

3.2 スケジューラ

スケジューラはグリッド環境内の計算機を監視し，アイドル状態の計算機に対して所定のスケジューリングアルゴリズムに基づき，ジョブキューに到着している並列ジョブを割り当てる．シミュレーションでは 10 分ごとにスケジューリング処理を行う．

3.3 並列ジョブ

本稿では，MPI(Message Passing Interface) 等を用いた，要求する計算機の台数を明示して実行する並列ジョブを対象とする [7], [8]．シミュレーションでは 10,000 個の並列ジョブを投入し，性能評価を行う．各並列ジョブ $Job_i (1 \leqq i \leqq 10,000)$ は，要求する計算機の台数 $N_{node\ i}$，基本計算時間 $Calc\ time_i$，基本通信時間 $Comm\ time_i$ の 3 つのパラメータを持つ．実際の並列環境で投入される並列ジョブの特徴を表すために，並列計算機センターで長期間に渡る利用状況のログから作成されたモデルである The Feitelson 1996 Model [1] を用いて N_i と $Calc\ time_i$

を決定する．

要求する計算機台数 N_i は $\frac{1}{N^{1.5}}$ で表される Harmonic 分布に従う．Harmonic 分布を用いると要求台数の少ないジョブが相対的に多く投入されるという傾向を示すが，特に 2 のべき乗，n の 2 乗，10 の倍数という特徴的な台数を要求する並列ジョブを多く投入するように重み付けを行う．

グリッド環境では各計算機の性能は不均一であり，同じ並列ジョブでも実行する計算機の性能によって実行時間が異なる．特に MPI 等を用いた場合の実行時間は，処理能力の最も低い計算機の実行時間に依存する．そこで，Job_i が $Node_j$ に割り当てられたときの実際の計算時間 $Calc\ time_{ij}$ を，基本計算時間 $Calc\ time_i$ を基に式 (2) で定義する．

$$Calc\ time_{ij} = Calc\ time_i \times \frac{Spec_{ave}}{Spec_{min}}. \qquad (2)$$

$Spec_{ave}$ はグリッド環境に属するすべての計算機の性能の平均，$Spec_{min}$ は Job_i が実際に割り当てられる N_i 台の計算機の中での最も低い性能を示す．また，基本計算時間 $Calc\ time_i$ は，要求する計算機の台数 N_i を基にした 3 stage hyperexponential distribution [1] に従うとし，$Calc\ time_i$ の最小値は 10 分，最大値は 30 日とした．

Job_i が $Node_j$ に割り当てられたときの実際の通信時間 $Comm\ time_{ij}$ は，Job_i が割り当てられるサイト数 $N_{site\ i}$ に比例し，基本通信時間 $Comm\ time_i$ を基にして式 (3) で定義する．

$$Comm\ time_{ij} = Comm\ time_i + (N_{site\ i} - 1) \times delay. \qquad (3)$$

$delay$ はサイト数に対する重みである．また，基本通信時間 $Comm\ time_i$ は，最大値を $Calc\ time_i$ の 20% とした一様分布に従う．

3.4 マイグレーションコスト

スタベーション回避アルゴリズムによって Job_i がマイグレーションされた後に再実行される場合，進捗状況によって再実行時の基本計算時間 $Calc\ time_i$ を短く設定する必要がある．再計算の際に，マイグレーションに必要な時間 (マイグレーションコスト) を Job_i の計算時間とみなし，短縮した基本計算時間に加える．再計算時の基本計算時間 $Calc\ time'_i$ を式 (4) で定義する．

$$Calc\ time'_i = Left\ time_i + N_i \times penalty. \qquad (4)$$

$Left\ time_i$ は $Calc\ time_i$ の未実行部分の時間であり，$0 \leqq Left\ time_i \leqq Calc\ time_i$ である．マイグレーションコストは，Meehean ら [9] の実験によると要求する計算機の台数に比例して大きくなることが示されている．これを反映するために，マイグレーションコストを $N_i \times penalty$ と定めた．$penalty$ はマイグレーションコストに対する重みである．

4. 性能評価

シミュレーションに用いた各種パラメータとシミュレーションによる各ジョブスケジューリングアルゴリズムの性能評価結果について考察する．スケジューリングアルゴリズムの評価基準としては，応答時間 (Response time) やターンアラウンド時間 (Turnaround time)，応答時間と実行時間の比である SR(Slowdown Ratio)，計算機の利用効率等が一般的に用いられてる [1], [3], [10]．本稿では，これに加えてスタベーションに陥った並列ジョブの数，スタベーション回避によるオーバーヘッドついても評価を行う．

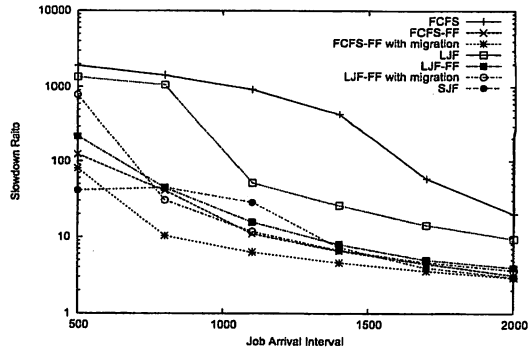

図 2 ジョブ到着間隔を変化させたときの，各スケジューリングアルゴリズムの平均 SR

性能評価では以下に述べる 4 つの観点から評価を行う．

4.1 Slowdown Ratio

Slowdown Ratio(SR) は従来から並列ジョブスケジューリングの評価基準として用いられており，式 (5) で定義される [10]．

$$SR = \frac{T_{response}}{T_{run}}. \qquad (5)$$

$T_{response}$ は並列ジョブの平均応答時間，T_{run} は並列ジョブの平均実行時間を示す．シミュレーションでは，グリッド環境へ投入される並列ジョブの平均到着間隔を 500〜2000 まで変化させたときの SR の値を調べる．

図 2 に，並列ジョブの平均到着間隔を 500〜2000 に変化させた時の SR の平均を示す．図中，FF は First-Fit を表す．図 2 より，並列ジョブの平均到着間隔が狭く，ジョブキューにジョブが溜まり易い環境では，SJF が SR を小さく抑えている一方，LJF は SR が大きくなっている．これらの特徴は合田 [3] の結果と同じである．また，図 2 より FCFS，LJF 共に First-Fit と組み合わせることにより SR をさらに小さくできることが分かる．これは，ジョブキュー先頭のジョブがアイドル計算機の不足により実行できず，後続のジョブを待たせるブロッキング状態が減少したためと考えられる．さらに，FCFS/First-Fit や LJF/First-Fit にマイグレーションを組み合わせることで，一層の SR の削減が可能であることが分かる．これはマイグレーションによってスタベーションが回避され，待ち時間を短くできたことによると考えられる．これに

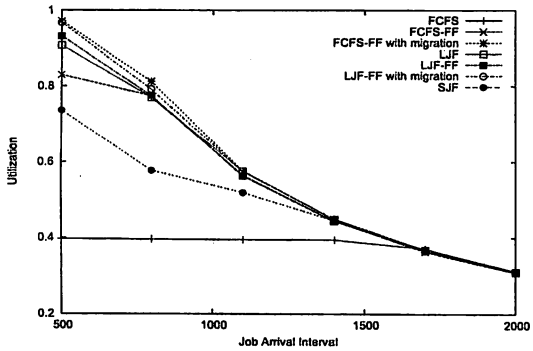

図 3　ジョブ到着間隔を変化させたときの，各スケジューリングアルゴリズムの平均計算機利用効率

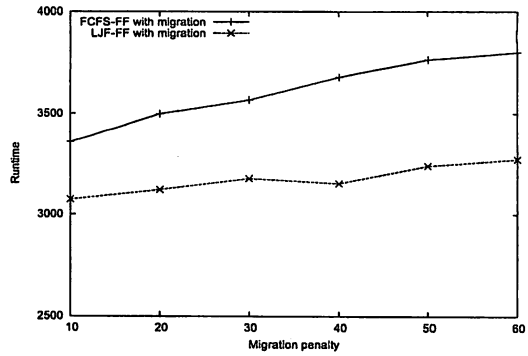

図 4　マイグレーションコストを変化させたときの，各スケジューリングアルゴリズムでのジョブの平均実行時間

ついては 4.3 節で詳しく検討する．

4.2　計算機の利用効率

計算機の利用効率は，すべての計算機に対する並列ジョブ実行中の計算機の割合として定義する．シミュレーションにより，並列ジョブの平均到着間隔を 500〜2000 まで変化させた時の各スケジューリングアルゴリズムの利用効率を調べる．図 3 に，並列ジョブの平均到着間隔を 500〜2000 に変化させた時の計算機の利用効率の平均を示す．

図 3 より，利用効率の点で LJF の方が SJF より優れていることが分かる．これは 2.1 節で述べたとおり，LJF が並列ジョブを要求する計算機の台数の多い順にソートした結果と言える．また，First-Fit を行うことでブロッキングの発生が抑えられるため，特に FCFS での利用効率が向上することが分かる．さらに，FCFS/First-Fit, LJF/First-Fit 共にマイグレーションを行うことで利用効率が向上することが分かる．これは式 (1) に示したとおり，マイグレーションにより実行を開始した先頭ジョブが使用する計算機台数 N_{top} は，常にサスペンドされた後続ジョブが使用していた計算機台数 N_{follow} 以上であることから，ジョブ実行中の計算機台数が増加するためである．

4.3　マイグレーションコストの影響

スタベーション回避のために用いるマイグレーションにかかるオーバーヘッドは，並列ジョブの平均実行時間に影響を与える．そこでマイグレーションコストの重みである penalty を 1〜60 まで変化させたときの平均実行時間により，マイグレーションの影響を調査する．

図 4 に penalty を 1〜60 まで変化させたときの，並列ジョブの平均実行時間を示す．penalty = 60 の時は，最大で並列ジョブの平均実行時間の約 20 倍のマイグレーションコストが付加される．

図 4 より，FCFS/First-Fit with migration の方が LJF/First-Fit with migration よりも penalty 増加の影響を受けやすいことが分かる．これは，LJF が並列ジョブを要求する計算機の台数の多い順にソートするため，ある時刻で実行されているジョブの数が少なくなることからマイグレーション回数も減少し，マイグレーションコストも減少するからである．実際にマイグレーションされたジョブの数を計測したところ，FCFS/First-Fit が 1300 回，LJF/First-Fit が 361 回であった．

LJF/First-Fit が penalty 増加の影響を受けにくい理由として，付加されるマイグレーションコストが相対的に小さいことが挙げられる．これは，First-Fit によって先行実行されるジョブキュー内の後続のジョブは，先頭のジョブに比べて要求する計算機の台数が少ない．マイグレーションコストは式 (4) より要求する計算機台数に比例するので，LJF/First-Fit のマイグレーションコストは小さくなる．

4.4　スタベーションジョブ数

グリッド環境において，スタベーション状態に陥った並列ジョブの数を各アルゴリズムごとに測定し，資源利用の公平性について比較・評価を行う．特に First-Fit を行うことによってスタベーションに陥った並列ジョブ数に注目する．

First-Fit はジョブキューの先頭から順に，要求計算機の台数が割り当て可能なアイドル計算機台数より少ないジョブを検索し，初めに見つかったジョブから順に計算機を割り当てる．この結果，ジョブキューの先頭に多くの計算機台数を要求する並列ジョブが存在する場合，先頭のジョブがスタベーションに陥る．スタベーションに陥った先頭ジョブのジョブキューでの待ち時間は，先行実行された後続ジョブ群の待ち時間に比べて非常に長くなることが予想される．

そこで Job_i がスタベーション状態に陥ったかどうかの判定を，式 (6) に基づき行う．ここで，$T_{wait\ i}$ は Job_i のジョブキュー内での待ち時間である．

$$T_{wait\ i} \geqq \sum_{k=i+1}^{i+50} T_{wait\ k}. \qquad (6)$$

式 (6) は，ジョブキュー先頭の並列ジョブ Job_i の待ち

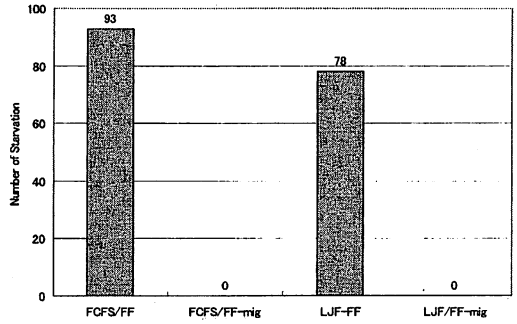

図 5　スタベーションの発生したジョブ数

時間が 50 個の後続ジョブの待ち時間の合計より大きい場合に，Job_i はスタベーション状態であるとみなす．

図 5 にマイグレーションの有無によるスタベーション発生ジョブ数の違いを示す．図 5 より，FCFS/First-Fit では 93 回，LJF/First-Fit では 78 回発生したスタベーションを，マイグレーションによりすべて解消していることが分かる．

これは，式 (1) で設定したマイグレーションを行う条件により，先行実行された後続ジョブの実行時間が，先頭ジョブが要求する計算機台数が揃うまでの時間に制限されたものによると考えられる．

5. おわりに

本稿では，グリッド環境でのスタベーションの回避を行うため，各種ジョブスケジューリングアルゴリズムに，マイグレーションを組み合わせたときの性能評価を行った．性能評価では，実際に投入される並列ジョブの特徴を反映したシミュレーションを行い，特にマイグレーションを行うことで並列ジョブに与える影響について評価した．この結果，以下のことが明らかとなった．

(1) マイグレーションを行うことにより，FCFS/First-Fit，LJF/First-Fit 共に，計算機の利用効率を向上させることができる．

(2) LJF/First-Fit にマイグレーションを組み合わせた場合，マイグレーションの実行回数が少なく，並列ジョブの実行時間にマイグレーションが与える影響が小さくなる．またマイグレーションコストが並列ジョブの要求する計算機の台数に比例するとき，小さいコストでマイグレーションを実行できる．

(3) First-Fit によって生じる並列ジョブのスタベーションは，マイグレーションにより解消できる．

一方，本稿で対象としたスタベーションは，ジョブの検索を行う First-Fit によって生じるスタベーションであり，ジョブのソートを行うことで発生するスタベーションは対象としていない．実際にジョブの整列を行うことでもスタベーションは発生するので，グリッド環境によって提供される計算機を完全に公平に利用することはできていない．従って今後の課題としては，ジョブの整列を行うことで発生するスタベーションの回避を行うアルゴリズムの提案と，その性能評価が挙げられる．

謝　辞

本研究の一部は、科学研究費助成金（若手 (B) 17700239）により行われた。関係各位に感謝する。

文　献

[1] Feitelson, D. G.: Packing Schemes for Gang Scheduling, *Job Scheduling Strategies for Parallel Processing*, Vol. 1162, pp. 89–110 (1996).

[2] Feitelson, D. G. and Jette, M. A.: Improved Utilization and Responsiveness with Gang Scheduling, *Job Scheduling Strategies for Parallel Processing*, Vol. 1291, pp. 238–261 (1997).

[3] Aida, K.: Effect of Job Size Characteristics on Job Scheduling Performance, *Job Scheduling Strategies for Parallel Processing*, Vol. 1911, pp. 1–17 (2000).

[4] Lifka, D.: The ANL/IBM SP Scheduling System, *Job Scheduling Strategies for Parallel Processing*, Vol. 949, pp. 295–303 (1995).

[5] Skovira, J., Chan, W., Zhou, H. and Lifka, D.: The EASY - LoadLeveler API Project, *Job Scheduling Strategies for Parallel Processing*, Vol. 1162, pp. 41–47 (1996).

[6] Li, H., Groep, D. and Wolters, L.: Workload Characteristics of a Multi-cluster Supercomputer, *Job Scheduling Strategies for Parallel Processing*, Vol. 3277, pp. 176–193 (2004).

[7] : http://www3.niu.edu/mpi/.

[8] N.Karonis and B.Toonen and I.Foster: MPICH-G2: A Grid-Enabled Implementation of the Message Passing Interface, *Journal of Parallel and Distributed Computing(JPDC)*, pp. 551–563 (2003).

[9] Meehean, J. and Livny, M.: A Service Migration Case Study: Migrating the Condor Schedd, *Midwest Instruction and Computing Symposium* (2005).

[10] Lo, V., Mache, J. and Windisch, K.: A Comparative Study of Real Workload Traces and Synthetic Workload Models for Parallel Job Scheduling, *Job Scheduling Strategies for Parallel Processing*, Vol. 1459, pp. 25–46 (1998).