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

2008－EVA－24 (10)
2008／3／19

並列プログラムのバッチ処理環境における自動キュー選択型スケジューリングの性能評価

川並秀観[†] 義久智樹[††] 金澤正憲[†††]

近年，多くの企業や研究機関において，膨大な計算資源を提供できるスーパーコンピュータが広く利用されている．スーパーコンピュータでは，並列ジョブへ計算資源を割り当てるスケジューリングの方法によって，その性能が大きく左右される．多くのスーパーコンピュータでは，バッチキューイングによる即時的なスケジューリングが採用されており，このような環境では，小規模なジョブと大規模なジョブの間で公平かつ効率の良い計算資源の割り当てを行えることが大きな課題となる．本研究では，バッチキューイング環境において，ジョブの規模に対応した待機時間を実現するためのスケジューリング手法を提案し，その性能をシミュレーションによって評価する．

Performance Evaluation of Automatic Queue Selection Scheduling on Batch Mode Processing Environment

Hozumi Kawanami,[†] Tomoki Yoshihisa[†††] and Masanori Kanazawa[†††]

In recent years, supercomputers which can serve a large amount of resource are used by more and more organizations such as business enterprise and research institution. Performance of supercomputer is strongly affected by scheduling method which determines how to allocate resource to parallel jobs. In most supercomputer systems, real-time scheduling method over batch queuing system is adopted. Under such conditions, it is important and necessary to serve effective and fair resource allocation according to scale of each jobs. In our study, we propose and evaluate a new scheduling method where submitted jobs are automatically forwarded to a proper queue according to their execution condition.

1. はじめに

近年，豊富な計算資源による高速な処理能力を持つ計算機の需要が増加しており，三次元の物理演算やタンパク質の構造解析，暗号解析など様々な分野において大規模な数値計算，シミュレーションが実行されている．このため，多くの研究機関や大学，企業でスーパーコンピュータが導入されている[1)2)]．

スーパーコンピュータは，大規模な問題を解くことを目的としており，多くの計算資源を必要とするジョブが同時に多数投入される状況が起こり得る．バッチキューイング環境では，バッチ処理されるジョブは一般的に実行時間が非常に長く，ある程度の待機時間は問題とならない．しかし，ジョブの経過時間の分散が非常に大きい環境では，ジョブの待機時間を経過時間に応じて抑制するべきである．例えば，経過時間が120分であるジョブに対して30分の待機時間は大きな問題とならないが，5分で終わってしまうジョブに対して30分の待機時間は適当でないと考えられる．システムが非常に混雑している状況では，スケジューリングの方法次第で，小規模なジョブに対して過大な待機時間が発生してしまう．また，ジョブの並列度が増えるにつれ，実際の計算処理に消費するCPU時間は飛躍的に増加するため，並列度についても経過時間と同様の措置が必要である．表1は並列度とCPU時間の相関の一例である．

京都大学所有のスーパーコンピュータシステムHPC2500では，ログの情報から，小規模なジョブを対象としたキューにおいて過大な待機時間が発生していることが確認できる．本研究では，このような実行規模と待機時間の不均衡を解消し，ジョブの実行規模に合わせた待機時間を課すスケジューリング手法を提

† 京都大学大学院 情報学研究科
Graduate School of Informatics, Kyoto University
†† 大阪大学サイバーメディアセンター
Cyber Media Center, Osaka Univesity
††† 京都大学 学術情報メディアセンター
Academic Center for Computing and Media Studies, Kyoto University

表 1 HPC2500 における並列度と CPU 時間の相関

並列度	CPU 時間中央値 [msec]
1	1.26e+4
4	5.17e+5
8	3.62e+5
16	1.41e+7
32	2.38e+8
64	1.63e+8
128	1.86e+9
512	1.99e+10

案する．また，上述の HPC2500 を対象としたシミュレーションを行い，提案手法の性能を評価する．

以下，2 章で研究背景についての説明を行い，3 章で提案手法の紹介を行う．4 章で実験の結果を示し，5 章でまとめる．

2. 研究背景

2.1 並列コンピューティングのアーキテクチャ

並列化ジョブのスケジューリングを考える際，その手法は，キューイングシステムとプランニングシステムのいずれかに分類される[3]．本節では，それぞれの特徴について説明する．

キューイングシステムとは，名前の通り，キュー構造を利用したスケジューリングシステムである．ジョブの投入先となる複数のキューが存在し，それぞれのキューは，クラスと呼ばれる計算資源の利用に関する制約を持っている．キューイングシステムでは，キューの中に待機ジョブが発生していない場合，システムに到着するジョブの実行要求に対して，計算資源の割り当てを試みる．空き状態の計算資源が足りずジョブの実行を即座に行えない場合には，そのジョブはキューの中で待機する．待機中のジョブがある場合には，設定されたポリシーに従って待機列に並ぶ．リアルタイムの空き計算資源状況のみを考慮したスケジューリングを行うため，待機しているジョブは，実行開始時間，終了時間に関する保証を全く受けられない．スケジューリング手法によっては，キューの先頭ジョブが実行を開始できない場合，他の実行可能ジョブを優先的に実行する．このような場合，計算資源の使用率は向上するが，並列度の大きいジョブがいつまでも実行されずに待機し続ける飢餓状態が発生する可能性がある．キューイングシステムはスケジューリングにかかるコストが低く実装が容易であり，今日，実装されている多くのスケジューリングシステムはキューイングシステムに分類される．

一方のプランニングシステムでは，現在および将来について計算資源の利用を計画，予約しており，システム内の全てのジョブに対して実行開始時間を割り当てる．このため，全てのジョブに対する実行開始時間の保証を行うことが可能である．プランニングシステムにはキュー構造が存在せず，到着する全てのジョブ実行要求は即座に実行開始時間を計画される．キューイングシステムと比較してスループット，利用率が高いという利点があるが，スケジューリングのコストが高く，システムの実装がキューイングシステムと比べて複雑なことが難点である．

本研究で提案する手法は，キューイングシステムにおける運用を想定している．

2.2 バッチ処理型プログラム実行環境

本研究では，バッチキューイングシステムのように，バッチ処理方式とキュー構造による計算処理を行うプログラム実行環境をバッチ処理型プログラム実行環境と呼ぶ[4][5]．通常，並列ジョブを実行するバッチ処理型プログラム実行環境には複数のキューが存在しており，それぞれに異なるクラスが設定されている．

バッチ処理型プログラム実行環境では，全てのジョブは，いずれかのキューを通して計算資源を割り当てられる．前述のように，それぞれのキューには異なるクラスが設定されているため，投入されるジョブの規模によって実行可能なキューが制限される．これにより，ジョブの規模ごとにキューを振り分けることでジョブの規模の多様性を吸収できる．ジョブの規模，すなわち，ジョブが使用する計算資源の多少によって最適なスケジューリング手法は異なるため，一般的には，それぞれのキューに適当なポリシーを設定することで性能の最適化を行う．バッチ処理型プログラム実行環境では，ユーザの意思によって利用するキューを明示的に選択する方法が一般的である．

並列度が大きいジョブや大量の計算資源を消費するジョブは，実行可能なキューが大きく制限される．しかし，投入するキューを吟味する必要の無い小規模なジョブが全体の大部分を占める場合，多数のユーザが無作為にキューを選択することで，ジョブの実行要求が特定のキューに片寄り得る．多くの場合，システム全体の CPU 資源空間の中にパーティションと呼ばれる複数の部分集合が定義されており，それぞれのキューはこれらのパーティションに関連付けられている[6]．原則として，キューへ投入されたジョブは実行に必要な計算資源をそのキューに関連付けられたパーティションから割り当てられる必要がある．このため，他のキューで空き計算資源が存在しているにもかかわらず，選択するキューによっては待機状態となるジョブが発生し得る．

2.3 関連研究

スーパーコンピューティング環境の演算性能は，運用するスケジューリング手法によって大きく左右されるため，様々な手法が広く研究されている[7][8]．本節では，スーパーコンピュータにおけるジョブスケジューリングの既存研究を紹介する．

文献[9]では，Achim によって，スーパーコンピュータの稼動中，状況に応じて最適なスケジューリングポリシーを選択する dynP という手法が提案されている．dynP では FCFS，SJF，LJF の基本的なポリシーから負荷状況に適したポリシーを選択する．一般的に，スーパーコンピュータの負荷は時間と共に大きく性質が変化するため，単一のポリシーでは変化する負荷状況に対応できない．dynP は，負荷状況に対してシステムが自動的にポリシーを選択することで，パラメータやポリシーの同定といった煩雑なチューニング作業を省略することを目的としている．

文献[10]で，Aida らは FPFS(Fit Processors First Served)，FPMPFS(Fit Processors Most Processors First Served) と FCFS，バックフィル有り FCFS の性能をシミュレーション，実験，解析によって評価している．FPFS，FPMPFS は，キューの先頭のジョブが実行を開始できない場合でも他のジョブを積極的に実行させていく手法である．FPFS，FPMPFS，バックフィル有り FCFS は，FCFS に比べて計算資源の利用率が高く，応答時間の平均値を減少できると結論付けている．Aida らはジョブの飢餓状態を回避する手段として，各ジョブに待機時間の限界値を設定している．待機時間が限界値を超えたジョブは FCFS におけるキューの先頭ジョブと同等の優先権を得ることができるが，限界値を超えた時点から実行開始までの待機時間については保証されていない．

提案手法では，自動キュー選択に加えて，特定の実行中ジョブが使用している計算資源を引き継ぐことで，特定の待機ジョブに優先的な実行権を与えている．これにより，優先権を得た待機ジョブが他の待機ジョブの待機時間を増加させることを防いでいる．

3. 提案手法

現在の HPC2500 の環境では，それぞれのキューにおける待機ジョブの序列は FCFS に従っている．また，先頭のジョブが実行を開始できない場合，直後のジョブから順に実行の判定を行い，実行可能であると判定されたジョブを実行させていく．これを FirstFit と呼ぶ．キューの中のいずれのジョブも実行開始不可である場合には，通常の FCFS と同様に他のジョブが終了するまで待機する．HPC2500 におけるスケジューリングはプランニングシステムでは無いため，それぞれのジョブに実行開始の時間を保証していない．従って，キューの先頭以外のジョブを優先して実行させることで，一度にまとまった量の空き資源が発生する機会が減少し，並列度の大きい大規模ジョブが飢餓状態となる可能性がある．飢餓状態の発生を抑制するため，HPC2500 では，FirstFit による追い越しで実行できるノードを制限している．しかし，並列度が 1 ノードあたりの CPU 数を大きく超えるジョブは，飢餓状態に陥る可能性が高く，大規模な並列ジョブの実行は管理者による手動操作が必要となる場面が存在する．提案手法では，ジョブの実行規模に応じた待機時間を与えることで小規模なジョブの応答性を向上すると共に，飢餓状態を解消する手段として**接続予約方式**を新たに提案する．

図 1 は提案手法のモデルである．ユーザは端末上のインターフェースからジョブマネージャに対してジョブの実行要求を行う．ジョブマネージャは自動キュー選択のポリシーに従い，ジョブに対して最適なキューを選択する．本研究における提案手法では，システムの容量を超えるだけの要求が到着している場合，大規模なジョブの待機時間と引き換えに，小規模なジョブの待機時間を抑制させる．

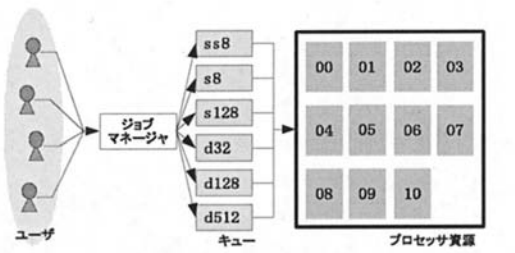

図 1　提案手法のモデル

3.1 自動キュー選択

本研究で提案する自動キュー選択では，ユーザが投入するジョブに対してジョブマネージャが最適な実行キューを選択する．この際，ジョブマネージャには少なくともジョブの並列化情報が与えらることを前提としている．基本的な方針として，投入されたジョブは実行可能キューの内，並列度の上限が最も厳しいキューへ割り当てられる．キューに設定されるクラス情報は環境によって様々であるため，ここでは今回の実験に利用した HPC2500 の環境を例として自動キュー選択の手順を説明する．HPC2500 におけるキューとクラ

ス情報の概要は表 2 の通りである．

HPC2500 に設定されたキューはスレッド並列とプロセス並列の観点から，三つの区分に分けられる．まず，ss8 と s8 は並列度が小さいジョブの処理を主眼としているため，最大 CPU 数，プロセス並列数，スレッド並列数はいずれも 8 までである．提案する自動キュー選択では，並列度が 8 までのジョブは経過時間が 2 時間未満の場合 ss8 で，それ以外は s8 で実行される．

次に，s128 はスレッド並列に特化したキューである．表 2 からも分かるように，スレッド並列数が 32 を超えるジョブは s128 のみ実行可能である．このことから，スレッド並列数が 32 を超えるジョブは全て s128 で実行される．

最後に，残りの d32，d128，d512 は，プロセス並列を重視したキューである．特に，d512 は他のキューと比較して非常に多数の CPU を利用したプロセス並列ジョブを実行できる．d32，d128，d512 それぞれにおけるプロセス並列数の上限は 32，128，512 である．ss8，s8，s128 ではプロセス並列数の上限が 8 であるため，並列度が 32 以下で，プロセス並列数が 8 より大きく 32 以下のジョブは d32 で実行される．同様に，並列度が 32 より大きく 128 以下で，プロセス並列数が 8 より大きいジョブは d128 で実行される．d32，d128 では実行できないプロセス並列数のジョブは d512 で実行される．

表 2　HPC2500 におけるキューのクラス情報

制限値	ss8	s8	s128	d32	d128	d512
CPU	8	8	128	32	128	512
プロセス	8	8	8	32	128	512
スレッド	8	8	128	32	32	32

3.2　接続予約の流れ

新たに投入されたジョブは，キューで待機しているジョブの有無にかかわらず実行開始を試みる．実行可能である場合には，他の待機ジョブを追い越して実行を開始する．不可である場合には，FCFS のルールに従うよう，キューの最後尾に並ぶ．従来の手法と同様に，提案手法でも FirstFit のメカニズムを利用する．

接続予約方式で実行の予約を行う場合，まずはじめに，CPU 数の要求条件が予約を要求するジョブと一致するジョブ，もしくは使用している CPU 数が最大のジョブを予約実行フラグの立っていない実行中ジョブから唯一つ選択する．このような特徴を持つジョブを，以降ではそれぞれ**要求一致ジョブ**，**最大空間ジョブ**と呼ぶ．また，実行の予約を行うジョブを**予約要求ジョブ**と呼ぶ．要求条件に一致するジョブが複数存在する場合，実行開始時刻が最も早いジョブを要求一致ジョブとする．使用 CPU 数が最大であるジョブが複数存在する場合には，スレッド並列数が最大であり，かつ実行開始時刻が最も早いジョブを最大空間ジョブとする．要求一致ジョブが存在する場合には，そのジョブが終了した際に生じる空き計算資源を予約要求ジョブがそのまま引き継いで実行する．このとき，他のジョブに予約されないよう，要求一致ジョブに予約実行フラグを立てる．要求一致ジョブが存在しない場合，最大空間ジョブの終了した際に生じる空き計算資源を解放予定資源に追加する．また，最大空間ジョブに予約実行フラグを立てる．接続予約の開始時には，解放予定資源は空集合である．解放予定資源のみで予約要求ジョブが実行を開始できるか調べ，実行開始可能であれば，最大空間ジョブの終了直後に予約要求ジョブを実行させる．実行開始不可であれば，十分な量の解放予定資源が得られるまで最大空間ジョブを選択し続け，解放予定資源に追加していく．実行に十分な解放予定資源が集まった時点で最大空間ジョブの選択をやめ，解放予定資源から実際に使用する計算資源を決定する．最大空間ジョブを複数回選択した場合，予約要求ジョブはこれらのジョブが全て終了した直後に実行を開始する．また，最大空間ジョブの終了によって解放される計算資源は，対応した予約要求ジョブのみ使用できる．解放予定資源の総量が予約要求ジョブの実行に必要な計算資源よりも多い場合には，適当な最大空間ジョブの終了時に，余剰分の CPU を予約から解放する．要求一致ジョブと最大空間ジョブの選択は，ともに，予約要求ジョブのキューで実行されているジョブから選択することを基本としている．しかし，このキューで実行中のジョブが使用している CPU 数の総和が予約要求ジョブの要求する CPU 数に満たない場合には，最大空間ジョブの選択を行う母集合を全ての実行中ジョブへと拡張する．以降では，この一連の実行予約を接続予約と呼ぶ．

3.3　接続予約を行う条件

本研究では，3.2 節で示した接続予約を行う条件として，三通りの場合を考える．

一つ目は，待機ジョブにそれぞれ独立した優先度を与え，その値が一定の閾値を超えた場合である．この優先度は，全てのジョブについて初期値は 0 である．新たにキューへ到着したジョブが他の待機ジョブを追い越して実行を開始した場合，および，FirstFit にキュー内での順序を追い越されたジョブが発生した場合に，追い越されたジョブの優先度へ一定の優先度定数が加

算される．これら優先度定数の加算による優先度パラメータを**固有優先度**と呼ぶ．固有優先度がキューに設定された一定の閾値を超えた場合，そのジョブは即座に接続予約を行う．また，複数の待機ジョブの固有優先度が同時に閾値を超えた場合には，キューの先頭に近いジョブから順に接続予約を行う．それぞれのジョブに加算される優先度定数は式 (1) で与えられる．式 (1) の定義より，並列度とスレッド並列数が大きいほど優先度定数は値が低くなる．すなわち，並列度とスレッド並列数が小さいジョブほど閾値を超えやすく，接続予約が行われやすい．固有優先度と閾値による接続予約を**優先度型接続予約**と呼ぶ．

$$\left(\frac{thread(j)}{thread_{\max}(q)} \cdot \frac{cpu(j)}{cpu_{\max}(q)}\right)^{-\frac{1}{2}} \qquad (1)$$
$$j \in J, \quad q \in Q.$$

J : 全てのジョブの集合
Q : システムに存在する全てのキューの集合
$thread_{\max}(q)$: キュー q の最大スレッド並列数
$thread(j)$: ジョブ j のスレッド並列数
$cpu_{\max}(q)$: キュー q の最大並列度
$cpu(j)$: ジョブ j の並列度

二つ目は，キューの先頭にあるジョブが，ジョブの終了による資源の解放が起こった際に実行を開始できない場合である．これを，**先頭型接続予約**と呼ぶ．先頭型接続予約は実行される頻度が高いと考えられるため，全てのキューで実行した場合，計算資源が接続予約に過剰に使用されることが考えられるため，d512 でのみ行うものとする．

三つ目は，新たにジョブが到着した時，そのキューで待機しているジョブが存在しないにもかかわらず，実行を開始できない場合である．これを**非待機型接続予約**と呼ぶ．今回の実験では，全てのキューがシステムの全 CPU を共有しているため，大部分の CPU が使用されている状態で並列度の大きいジョブが到着すると，そのままでは飢餓状態に陥る可能性が高い．このため，三つ目の条件による非待機型接続予約を付け加えることで，確実な実行を保証することができる．

同時に行える接続予約の数はキューごとに制限が設けられているため，接続予約を実行する条件を満たしている場合でも，既存の予約ジョブ数が制限に達していれば接続予約は行われない．このような場合，接続予約を行うことができなかったジョブは，予約を棄却され，キューの中で他の待機ジョブと同等の扱いを受ける．

表 3　優先度接続予約における各キューの閾値設定

キュー	s128	d32	d128	d512
閾値	30	20	20	20

表 4　各キューの接続予約制限数

	s128	d32	d128	d512
提案 (1)	0	0	0	0
提案 (2)	1	3	2	1

4. 評価実験

本節では，HPC2500 を基にしたモデルにおける従来手法と提案手法のシミュレーション結果を示す．

まず，提案手法のシミュレーションに用いたパラメータ設定を示す．設定を要するパラメータは，接続予約の同時実行上限数および，固有優先度の閾値である．ただし，接続予約は並列度の大きいジョブの飢餓状態を回避することが目的であるため，今回の実験では，並列度の小さいジョブのみを実行する ss8，s8 では接続予約を行わない．各キューの閾値と接続予約上限数は，それぞれ表 3，表 4 の通りである．「提案 (1)」は接続予約を行わない場合であり，「提案 (2)」は接続予約を含む全てのアプローチを組み合わせた場合である．「従来」は，現在，HPC2500 で運用されているスケジューリング手法の場合を表している．

次に，実験の結果を示す．図 2 は待機時間中央値の比較を，図 3 は待機時間最大値の比較を表している．図 2 より，提案 (1) は d32，d128 とジョブ全体における待機時間中央値が最も低いとわかる．同時に，s8，s128，d512 では従来より中央値が大きく増加しているとわかる．提案 (2) は ss8 で最も低い待機時間中央値を示しているが，他のキューでは従来と比較して大きく値が増加している．しかし，ジョブ全体での待機時間中央値は従来の場合とほぼ等しい値を示している．また，図 3 より，提案 (1) は s128，d128 において待機時間最大値が最も低く，その他のキューでは従来と同程度の値を示している．提案 (2) は ss8，s8，s128 において従来より低い最大値を示しており，この内，ss8，s8 では最も低い値を示している．

5. まとめ

本研究では，バッチ処理型プログラム実行環境において，キューのクラスとジョブ性質の不一致によって生じる小規模なジョブの応答性悪化を解消するため，自動的にキューを選択する手法を提案した．さらに，並列度が大きいジョブを確実に実行させる手段として，接続予約方式を新たに提案し，自動キュー選択と組み合わせた．京都大学学術情報メディアセンターの

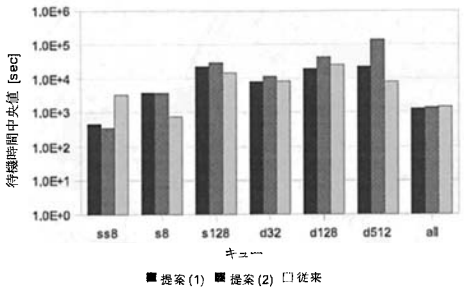

図 2　待機時間中央値の比較

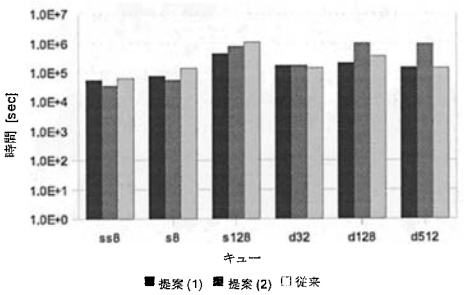

図 3　待機時間最大値の比較

HPC2500 をモデル化し，従来手法と提案手法のシミュレーションによる比較実験を行ったところ，提案手法によって，それぞれのキューでクラスに対応したジョブを実行させ，小規模なジョブを対象としたキューにおけるジョブの待機時間中央値および最大値を減少できることが分かった．また，接続予約を用いた場合，大規模なジョブを対象としたキューの待機時間中央値および最大値は大きく増加し，小規模なジョブは逆に減少することが分かった．更に，提案手法を用いた場合，従来手法よりもジョブ全体の待機時間中央値を減少することができた．

今後の展開としては，優先度型接続予約における閾値の影響や小規模ジョブ用のキューで接続予約を行った場合の影響の調査が考えられる．

謝　辞

本研究は，文部科学省特定領域研究「情報爆発のための装着型入出力デバイスを用いた情報操作方式」(19024056) の研究助成によるものである．ここに記して謝意を表す．

参 考 文 献

1) Usage of SCS MAFFIN, "http://www.affrc.go.jp/ja/info/scs/index.html".
2) 京都大学学術情報メディアセンター, "https://web.kudpc.kyoto-u.ac.jp/".
3) Matthias Hovestadt, Odej Kao, Axel Keller and Achim Streit, "Scheduling in HPC Resource Management Systems: Queuing vs. Planning," in *Proc. 9th International Workshop on Job Scheduling Strategies for Parallel Processing*, 2000, pp. 1-20.
4) Olaf Arndt, Bernd Freisleben, Thilo Kielmann and Frank Thilo, "A comparative study of online scheduling algorithms for networks of workstations," Cluster Computing, Vol. 3, No. 2, pp. 95-112, 2000.
5) Olaf Arndt, Bernd Freisleben, Thilo Kielmann and Frank Thilo, "Batch Queuing in the WINNER Resource Management System," in *Proc. International Conference on Parallel and Distributed Processing Techniques and Applications*, Las Vegas, Jun. 1999, pp. 2532-2529.
6) Dror G. Feitelson, Larry Rudolph, Uwe Schwiegelshohn and Kenneth C, "Theory and Practice in Parallel Job Scheduling," in *Proc. 3rd International Workshop on Job Scheduling Strategies for Parallel Processing*, Lecture Notes in Computer Science, vol. 1291, pp. 1-34, 1997.
7) Junwei Cao and Falk Zimmermann, "Queue Scheduling and Advance Reservations with COSY," in *Proc. 18th IEEE International Parallel & Distributed Processing Symposium*, Apr. 2004, pp. 63.
8) Warren Smith, Ian Foster and Valerie Taylor, "Scheduling with Advanced Reservations," in *Proc. 14th International Parallel and Distributed Symposium*, 2000, pp. 127-132.
9) Achim Streit, "Evaluation of Unfair Decider Mechanism for the Self-Tuning dynP Job Schedulers," in *Proc. IPDPS 13th International Heterogeneous Computing Workshop*, 2004, pp. 108.
10) Kento Aida, Hironori Kasahara and Seinosuke Narita, "Job Scheduling Scheme for Pure Space Sharing Among Rigid Jobs," in *Proc. 4th International Workshop on Job Scheduling Strategies for Parallel Processing*, Lecture Notes in Computer Science, vol. 1459, pp. 33-45, Aug. 1998.