2007－EVA－20 (3)
2007／3／16

バッチ処理型プログラム実行環境における キュー選択方式に関するシミュレーション評価

川並秀観[†] 義久智樹[††] 金澤正憲[††]

近年，シミュレーションや数値計算など様々な開発分野で重要な役割を担うようになり，数値計算用の大型計算機が普及している．多くの大型計算機では，バッチ処理によるジョブの調整を行っている．バッチ処理型プログラム実行環境では，ユーザは実行プログラムとなるジョブを大型計算機に投入し，大型計算機がジョブをスケジューリングして実行することで，計算資源を有効に利用する．バッチ処理型プログラムでは，一般に，利用できる計算資源の異なる幾つかのキューが設定されており，ユーザはどのキューにジョブを投入するかを明示的に指定する必要がある．このような環境では，複数のユーザが同時間帯に同じキューへジョブを投入してしまう可能性がある．このため，各キュー間で資源の利用率に片寄りが生じ，各ジョブの処理完了時間が増加することが考えられる．本研究では，システム側でのジョブ投入の自動最適化によってこのような偏りを解消することで，大型計算機のシステムスループットの増加と利便性向上の実現を目的としている．本研究では，いくつかの自動キュー選択手法を提案し，モンテカルロシミュレーションによってこれらの性能を比較した．

An Evaluation for Queue Selection on Batch Mode Processing Environment

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

In recent years, super computer system proliferate since simulation and numerical computing using them are necessary for various research fields. In many cases, these systems adopt batch mode processing. In batch mode processing, users submit jobs, i.e. executable programs, to super computers and they schedule jobs so that computational powers are effectively consumed. Generally, users have to choose the queue to submit their job explicitly since there are several queues which have different computing resource settings. Therefore, service time for executing jobs can become longer. The goal of our study is to achieve improvement of throughput and to realize good usability of super computer. In this paper, we propose several methods for selecting queues, and estimate their performance with result of Monte Carlo simulation.

1. はじめに

近年，様々な分野において，大規模な数値計算やシミュレーション等に高速な処理能力を持つ計算機が必要とされている．このため，多くの研究機関や大学，企業がそれぞれスーパーコンピュータと呼ばれる大型計算機システムを導入するようになった[1)2)3)4)5)]．現在，このような大型計算機システムにはHPC(High Performance Computing)と呼ばれる概念が広く取り入れられており，複数の大型計算機を相互結合網によって結合したシステムが主流となっている[8)9)]．

スーパーコンピュータのように大規模な計算環境では，膨大な計算資源を必要とするジョブが同時に複数投入される状況が考えられる．この際に，投入された全てのジョブをリアルタイム処理のような対話的手法によって同時に処理しようとすると，物理メモリの合計使用量が利用可能量を超え，スワッピングによる極端な処理性能の低下が発生する．従って，こうした問題を解消し，限られた計算資源を効率的に利用するための手法が必要となる．その例として，NQS (Network Queuing System) 等の Batch Queuing System (BQS)と呼ばれる手法が広く知られている[7)10)]．BQSはリソースやジョブの管理を行うシステムであり，バッ

† 京都大学大学院 情報学研究科
Graduate School of Informatics, Kyoto University
†† 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

チ処理方式を用いてプロセッサにデータ処理を実行させる．BQSのようなバッチ処理型プログラム実行環境では，限られた計算資源をユーザが公平に利用できるよう，バッチ処理方式でデータを処理している．すなわち，ユーザは実行プログラムとなるジョブを大型計算機に投入し，大型計算機がジョブをスケジューリングして実行することで，計算資源を有効に利用する．2章で詳述するが，通常，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，ユーザはジョブを投入するキューをこの中から選択しなければならない．このため，複数のユーザが偶発的に同じキューを選択すると，他のキューにおける計算資源の利用効率が低下し，全体のスループットが下がると考えられる．

本研究では，リソースを効率的に利用する幾つかのジョブスケジューリング手法を提案し，モンテカルロシミュレーションによる性能評価を行った．ユーザによるキューの選択をシステム側で自動最適化することによって，システム利用の簡便性とリソースの有効利用によるスループット向上を実現することが提案手法の目的である．

以下では，2章でバッチ処理型プログラム実行環境について簡単に説明し，3章で提案手法を紹介する．4章で実験の設定を解説し，5章で実験結果を示した後，6章で考察を行い，7章でまとめとする．

2. バッチ処理型プログラム実行環境

バッチ処理型プログラム実行環境では，一定量または一定期間のデータをまとめて処理させるため，単独のジョブが常に計算資源を占有することが無い．このため，多くのユーザが計算資源を共有でき，効率的なジョブ処理が可能となっている．また，ユーザが実行する処理をシェルスクリプトにまとめて記述し，これを用いてジョブを投入することで，人間による入力無しに所望の作業を自動で行えるという利点がある．

通常，バッチ処理型プログラム実行環境には，最大プロセス数や最大スレッド数，最大CPU時間やメモリ量といった計算資源の利用制限がそれぞれ異なるキューが何種類か用意されている．環境によっては，相互連結されている複数のノードにまたがって計算資源を利用できる．

ユーザはジョブを投入する際にどのキューを使用するかを何らかの手段で決定する必要があり，キューを決定する手段として，ジョブ投入を行うコマンドの実行時にオプションでキューを指定する手法が利用される．多くの場合，このオプションはシェルスクリプトに記述できる．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明示的に投入するキューを選択することが原則である．高度に並列化されたプログラムや長期間にわたって実行されることが自明であるプログラムなどは，ジョブを投入するキューをそれぞれに設定された条件から判断して選ばなければならない．しかし，そのように大規模なジョブ，または特定の条件を必要とするジョブの数が，実行される全ジョブ数と比較して大幅に少ない場合，多数のユーザが無作為にキューを選択することで，ジョブ数が特定のキューに偏ることが起こりえる．このとき，計算資源を有効に利用できないノードが生じ，システムのスループットが低下してしまう．このような問題を解消する手段として，他のノードからアイドル状態のリソースを借りる手法や，アイドル状態のノードへジョブを移動させる手法が提案されている．しかし，これらの手法では高度なリソース管理やジョブ管理が必要であり，実装が非常に複雑であるという問題がある．

上述のように，現在のバッチ処理型プログラム実行環境では，人為的な計算負荷の偏りが生じることが考えられる．ジョブを投入するキューをコンピュータによって自動的に指定することで，リアルタイムの負荷状況を考慮した最適なキューイングを行うことができ，これにより，全体のスループットを向上できると考える．

2.1 関連研究

NQS[10] (Network Queuing System) はバッチジョブのスケジューリングを行うツールとしてNASAで開発された．初期バージョンのCosmicでは複数キューの実装をサポートしており，それぞれのキューでは計算資源が制限されている．キューに投入されたジョブはFIFO(First In, First Out)の原則によって処理が行われる．商用として開発された後期バージョンでは，負荷分散および高度なリソース管理機能が追加された．シンプルで実装が容易であるため，多くのスーパーコンピュータに実装されている．NQSは標準的なバッチ処理型プログラム実行環境を提供する標準プロトコルであるため応用が利きやすく，提案手法を実装して組み合わせることができると考える．

また，ジョブの振り分けを行うシステムとして，Condor[10]がある．アイドル状態のワークステーションやサーバから計算資源を収集し，長期間稼動するバッチジョブへ割り当てることを目的としてWisconsin大学が開発した．Condorの制御を行うデーモンは，プールされているホストの稼動情報をCPUの負荷や入出力デバイスの状態で判断する．アイドル状態であると判断されたホストがあれば，他のコンピュータで待機

状態にあるバッチジョブを送信して処理を行わせる．チェックポイント機能を実装しているため，アイドル状態のホストがビジー状態に遷移した際，他のアイドル状態であるホストへバッチジョブを転送してチェックポイントから処理を継続して行うことができる．本研究で提案する手法は，Condorにおいても実装可能と考える．

3. 提案手法

この章では，本研究で提案するジョブの実行順序の決定手法を紹介する．以下で述べる手法の基本的な発想は，ジョブが特定のキューに集中して投入されることで性能の低下が起こらないよう，ジョブを分散させることである．

3.1 ランダム法

ランダム法では，キューへのジョブ投入を管理するジョブマネージャに到着したジョブを，ランダムに選択したキューへ投入する．本論文では，主に他の手法との性能比較を目的としてこの手法を扱う．

3.2 MQL(Minimum Queue Length) 法

MQL法では，ジョブの到着時に最もジョブ数の少ないキューをジョブマネージャが判断し，到着したジョブをそのキューへ投入する．ジョブ数が最小のキューが複数存在する場合には，それらの内からランダムにキューを選択する．アイドル状態のキューが存在する場合に，そのキューを必ず利用できることから計算資源の有効利用が期待でき，この結果，性能が向上すると考えられる．

ここで，ジョブマネージャは，システムに存在する全てのキューにある実行中，待機中のジョブ数をリアルタイムに把握していることを仮定している．

3.3 MAQL (Minimum Average Queue Length) 法

ジョブの到着時点で，それまでの平均キュー長が最小であるキューに到着したジョブを投入する．平均キュー長の等しいキューが複数存在する場合，キューの識別番号が最も小さいキューを選択する．

MQL法ではジョブ到着時点の負荷状態のみを考慮してキュー選択を行っているのに対し，MAQL法では，過去のジョブ到着傾向を考慮したキュー選択が行えると考える．これにより，MQL法よりも性能のよいキュー選択が期待できる．

3.4 オフセット法

オフセット法では，あらかじめ定められた一定の時間間隔 (オフセット) を利用してキュー選択の制御を行う．ジョブマネージャは，最近のオフセット中にキューを離脱した (処理が完了した) ジョブの数をキューごとに把握していると仮定する．以降では，オフセット中に処理が完了したジョブをオフセットジョブと定義し，その数をオフセットジョブ数と呼ぶものとする．

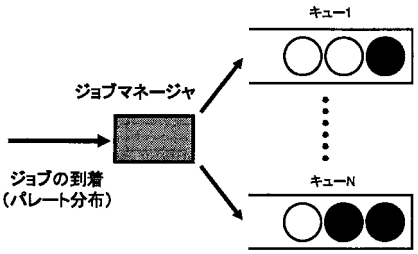

図 1 シミュレーションに用いたモデル

MAQL法と同様に，過去のデータを考慮したキュー選択が期待できる．オフセット中に処理したジョブ数が最も多いキュー，すなわち，平均サービス時間が最短のキューを選択することで，全体の平均サービス時間を低減できると考える．また，MAQL法と違い，十分に古い状態の情報を考慮しないことでより動的なキュー選択が期待できる．

ジョブが新たに到着したとき，ジョブマネージャはオフセットジョブ数の最も少ないキューをジョブの投入先として選択する．ただし，シミュレーション開始後，時間がオフセット未満の内に到着したジョブはランダム法によって投入先が決定されるものとした．

4. ジョブ投入システムのモデル化

モンテカルロ法によるシミュレーションを行うにあたって，プログラムの挙動を定義するため，実際のバッチ処理型プログラム実行環境をモデル化した．図1にシミュレーションで用いたモデルを示す．通常，複数のノードが相互連結して構成されたスーパーコンピュータではユーザとの通信を代表して行うゲートウェイノードが存在する．このため，本モデルは実際のジョブ投入システムと類似しており，妥当なモデルと考える．ゲートウェイノードは，ユーザからシェルスクリプトでジョブ投入のリクエストを受け取り，指定されたキューへとジョブを投入する．図1ではジョブマネージャがゲートウェイノードの役割を果たし，それ以外のノードはそれぞれキューを設置して投入されたジョブを受け入れる．

到着時間間隔分布には式 (1) で示すパレート分布を用いた．パレート分布は所得やインターネットトラフィックのデータサイズなどの分布を表現する際に用いられる．X[min] はジョブの到着時間間隔をあらわ

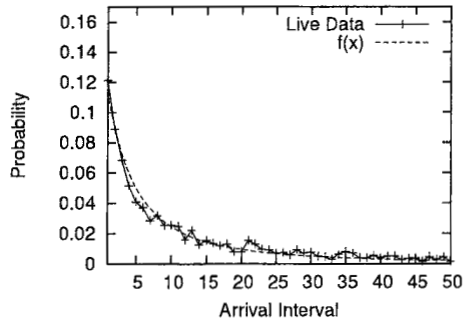

図 2 ジョブ到着時間間隔の実データとパレート分布による近似

す確率変数である．パラメータ a，b は，実データの採取から得られた結果より最小二乗法を用いた近似によって決定した．実データの採取は京都大学大型計算機センター所有の HPC2500 を利用した．今回採取した約 2ヶ月間の実データのうち最大の到着間隔は 1023 分であり，これを基に，X の実現値 x で 1000 を超えるものは廃棄した．

実データと近似したパレート分布の比較図を図 2 に示す．

$$\Pr(X \leq x) = F(x) = 1 - (1 + \frac{x}{b})^{-a} \qquad (1)$$

$$\text{p.d.f} \quad f(x) = \frac{ab^a}{(x+b)^{a+1}}$$

$$a = 1.1, \quad b = 7.34927$$

$1, \ldots, N$ はシステムに存在するキューの識別番号であり，ジョブマネージャによって各キューへ投入されたジョブは FCFS(First Come First Serve) よってサービスを開始される．それぞれのジョブは，到着時にサービス終了時刻 (キュー離脱時刻) を割り当てられ，先に到着したジョブが必ずしも先に終了するとは限らない．サービス終了の時刻は到着時点の現在時刻 T_{now}[min] と平均 p_i[min](i はキューの識別番号) の指数分布に従う確率変数 $Y(p_i)$[min] の和，$T_{now} + Y(p_i)$[min] で与えられる．

今回の実験では，キュー数のパラメータを $N = 6$ とした．これは，京都大学の大型計算機 HPC2500 のバックエンドプロセッサで実行されるキュー数から決定した．それぞれのキューにおける指数分布の平均値 p_i[min] を次に与える．

(1) パラメータセット 1
- (a) $p_i = 30$ $(i = 1, \ldots, N)$
- (b) $p_1 = p_2 = p_3 = 15$, $p_4 = p_5 = p_6 = 45$
- (c) $p_1 = p_2 = 15$, $p_3 = p_4 = 30$, $p_5 = p_6 = 45$
- (d) $p_i = 5 + 10(i-1)$ $(i = 1, \ldots, N)$

(2) パラメータセット 2
- (a) $p_i = 60$ $(i = 1, \ldots, N)$
- (b) $p_1 = p_2 = p_3 = 30$, $p_4 = p_5 = p_6 = 90$
- (c) $p_1 = p_2 = 30$, $p_3 = p_4 = 60$, $p_5 = p_6 = 90$
- (d) $p_i = 35 + 10(i-1)$ $(i = 1, \ldots, N)$

(3) パラメータセット 3
- (a) $p_i = 90$ $(i = 1, \ldots, N)$
- (b) $p_1 = p_2 = p_3 = 45$, $p_4 = p_5 = p_6 = 135$
- (c) $p_1 = p_2 = 45$, $p_3 = p_4 = 90$, $p_5 = p_6 = 135$
- (d) $p_i = 65 + 10(i-1)$ $(i = 1, \ldots, N)$

以降では上記のパラメータセットをそれぞれ (1)(a), (2)(c) のように記述する．(1)(a), (2)(a), (3)(a) はそれぞれ，全てのキューで処理性能が均一である場合を想定しており，パラメータセットが (1), (2), (3) となるにつれて処理性能が弱くなっていることを意味している．他の (b), (c), (d) はそれぞれのキューで処性能に差がある場合を考えている．(b) はこの中で最も分散が大きく，(c) では分散が小さくなる．(d) の場合は，(1), (2), (3) と平均サービス時間が大きくなっていくときに，それぞれのキューで増加する平均サービス時間が等しくなるよう設定した．

オフセット法におけるオフセットの値 M は M=150[min], 1500[min], 3000[min], 5000[min] をを用い，(1)(a) から (3)(d) まで全てのパラメータにおいて実験を行った．シミュレーションはモンテカルロ法にて行い，シミュレーションの設定時間は一週間とした．利用する乱数の生成にはメルセンヌツイスター[6] を用いた．

5. 評価実験

本章では，モンテカルロ法によるシミュレーションの結果を示す．

5.1 オフセット値による性能比較

図 3 に，パラメータセットが (1)(d) であるときの M 毎のオフセット法のサービス時間確率分布を示す．この図より，M=1500, 3000, 5000 の場合にはほとんど差が無く，M=150 の時には他の場合よりもジョブが短時間で完了しやすいことが分かる．図 3 ではパラメータ (1)(d) を用いているが，(1)(b), (1)(c), (2)(b), (2)(c), (3)(b), (3)(c) でもこれと同様の傾向

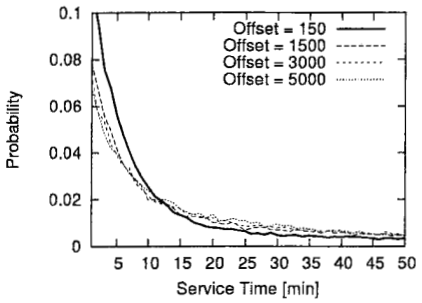

図 3　オフセット値 M によるオフセット法の比較 ((1)(d))

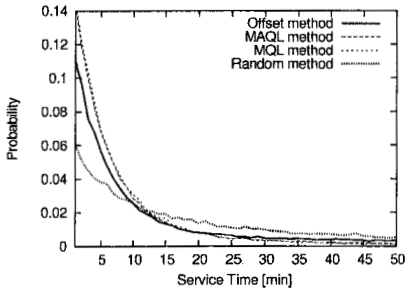

図 4　手法ごとのサービス時間確率分布 (M=150, (1)(d))

が得られ，ここで述べた以外のパラメータでは M がどの値をとってもほぼ等しい確率分布が得られた．

5.2　手法間の性能比較

図 4 に M=150 でパラメータ (1)(d) の場合における各手法のサービス時間確率分布を示す．グラフ中の線はそれぞれのパラメータにおけるジョブのサービス時間確率分布である．この図より，MQL 法と MAQL 法はほとんど同じ分布を示し，他の手法よりも短時間でジョブを処理できていることが分かる．また，オフセット法がランダム法に比べて短時間でジョブを処理できていることが読み取れる．図 4 ではパラメータ (1)(d) を用いているが，パラメータ (1)(a), (2)(a), (3)(a) の時，それぞれの手法が描く確率分布のグラフはほぼ等しくなった．また，これ以外のパラメータでは図 4 と類似した傾向を示した．

5.3　パラメータ変化による性能比較

図 5, 7, 6, 8 はそれぞれが各手法同士でパラメータ (1)(d), (2)(d), (3)(d) の場合におけるサービス時間確率分布を示している．これらのグラフから，パラメータ (1)(d) の時にジョブが短時間で完了できる確率は (2)(d), (3)(d) の場合を大きく上回っていることが分かる．この傾向は他のパラメータ ((1)(a), (2)(a), (3)(a) など) でも得られたが，最も顕著に差が現れ

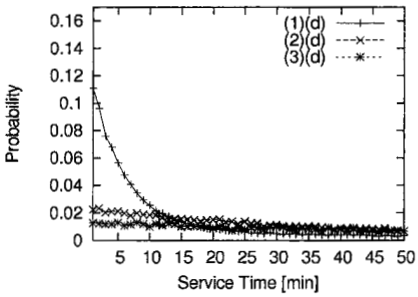

図 5　オフセット法 (M=150) の比較

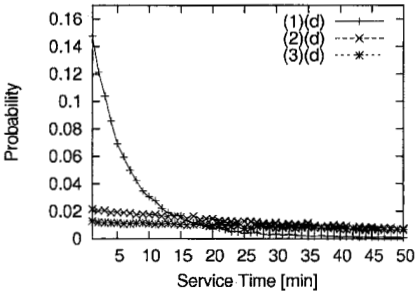

図 6　MQL 手法の比較

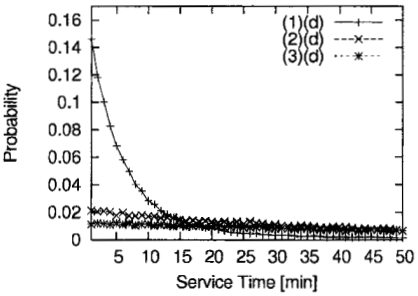

図 7　MAQL 法の比較

たのは上記の (1)(d), (2)(d), (3)(d) の組み合わせであった．したがって図から，サービス時間を短くするという観点からは，MAQL 法と MQL 法が最も性能が高く，オフセット法，ランダム法の順番となる．特に，非常に高速な処理を行えるキューと，比較的遅いキューが共存している場合に MAQL 法と MQL 法の効果が現れた．

6.　考　　察

6.1　M の変化で生じる差の原因

図 3 から，パラメータ (1)(d) の時，M=150 は他の M よりもサービス時間の期待値が低く，ジョブを短時間で処理できていることが分かる．M=1500, 3000,

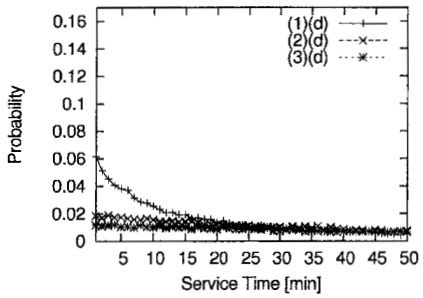

図 8 ランダム法の比較

5000 がほぼ等しい確率分布を示しているが，この理由は次のように考えられる．パラメータセット (1)(d) では各キューでの平均サービス時間 p_i に偏りが生じているため，十分な時間が経過した時，この偏りが処理の完了したジョブ数へ正確に反映される．この結果，p_i が最も小さいキューへジョブが投入されやすくなる．

6.2 手法の優劣関係

図 4 では，パラメータ (1)(d) の時，ランダム法以外の三手法がランダム法よりもジョブを短時間で処理できていることが分かる．また，図 6，図 7 の結果と合わせて，MQL 法と MAQL 法の間に明確な差が生じていないことが読み取れる．この傾向は (1)(d) 以外のパラメータセットでも現れており，これらのことから，あるキューについてキュー長が最小である確率と平均キュー長が最小である確率は互いに近い値であると考えられる．

6.3 平均サービス時間 p_i の増加による性能変化

図 5，6，7，8 からは，パラメータが (2)(d)，(3)(d) と変化するにつれてサービス時間が大きくなる確率が著しく増加していることが分かる．また，類似した傾向が (2)(a)-(2)(c)，(3)(a)-(3)(c) でも現れている．これは単に，キューの平均サービス時間 p_i が上昇した影響であると考えられる．

7. まとめ

本研究では，バッチ処理環境におけるリソース利用率における不均衡の発生を従来より単純な手法で回避することを目的とし，いくつかの自動キュー選択手法を提案した．また，提案手法について様々なパラメータセットでモンテカルロ法によるシミュレーションを行い，その性能を評価した．この結果，ジョブのサービス時間確率分布という観点からは，MQL 法と MAQL 法の間にはほとんど差がないことが分かった．また，オフセット法では，実験で用いたパラメータ M=150, 1500, 3000, 5000 の内，M=150 の場合が最も短時間でジョブを処理することができるとわかった．それぞれの手法を比較した結果，MQL 法と MAQL 法が最も性能が高く，オフセット法がこれに続き，ランダム法は最も性能が低かった．本研究では評価基準をサービス時間確率分布に絞っており，異なった基準による実験結果の評価が更に必要であると考えられる．

今後の課題として，前述の新たな評価基準による評価と，新規キュー選択手法の提案，提案したキュー選択手法の発展などが考えられる．

謝　辞

本研究の着想段階から実験設定，論文執筆まで多大なご協力を頂いた先生方に感謝し，ここに謝意を表する．また，本研究の一部は，文部科学省科学研究費補助金（若手研究（B））「選択型コンテンツの放送型配信に関する研究」（課題番号：18700085）の研究助成によるものであり，ここに記して謝意を表す．

参考文献

1) HPC Systems, http://www.hpc.co.jp/
2) Usage of SCS MAFFIN, http://www.affrc.go.jp/ja/info/scs/index.html
3) 産業総合技術研究所, http://www.aist.go.jp/
4) 学術情報メディアセンター, https://web.kudpc.kyoto-u.ac.jp/lsc/
5) 東京工業大学, http://www.titech.ac.jp/home-j.html
6) Mersenne Twister with improved initialization, "http://www.math.sci.hiroshima-u.ac.jp/m-mat/MT/"
7) Karl Czajkowski, Ian Foster, Nick Karonis, Carl Kesselman, Stuart Martin, Warren Smith and Steven Tuecke, "A Resource Management Architecture for Metacomputing Systems," in *Proc. IPPS/SPDP'98 Workshop on Job Scheduling Strategies for Parallel Processing*, page 4–18. IEEE-P, March 1998.
8) Yoav Etsion and Dan Tsafrir, "A Short Survey of Commercial Cluster Batch Schedulers," Technical Report 2005-13, He-brew University, May 2005.
9) Louis H. Turcotte, "A Survey of Software Environments for Exploiting Networked Computing Resources," Draft Report, Engineering Research Center for Computational Field Simulations, Mississippi State, January 1993.
10) Mary Papakhian, "Comparing Job-Management Systems: The User's Perspective," *IEEE Computational Science & Engineering*, vol. 5, Issue 2, April 1998.