情報処理学会論文誌 Vol. 52 No. 2 780–791 (Feb. 2011)

分散処理の効率化のための実行時間予測手法

菅谷至寛[†1] 丹野祐樹[†1,*1]
大町真一郎[†1] 阿曽弘具[†2]

並列処理環境の利用効率を向上させるためには，効率的なタスクスケジューリングや負荷分散システム等が必要である．しかしそれらの中には，システムに投入されるプロセスの実行完了時間が既知であるという前提のもとで構築されている手法が多々存在する．よって，それらを実際にシステムに導入する際，投入されるプロセスの実行時間予測が必要となる．過去の実行履歴から予測対象プロセスに類似したプロセスを発見し，その情報を用いることで実行時間予測が可能である．ここで，プロセス間の類似性をどのように規定するかという問題が生じる．本論文で提案する予測手法では，プロセス間の類似性を判定するために有用なプロセス情報を，相互情報量を用いて選択する．それによって得られる複数の類似判定用テンプレートを用いて複数回の仮予測を行い，t 分布によって定義される信頼度とプロセス情報数に基づく予測選択を行う．予測実験の結果，提案手法が従来手法よりも高い予測精度を示すことを確認した．

Run Times Prediction for Load Balancing Efficiently

Yoshihiro Sugaya,[†1] Yuki Tanno,[†1,*1]
Shinichiro Omachi[†1] and Hirotomo Aso[†2]

Task runtime prediction is often required for task scheduling algorithms and load balancing methods because some scheduling methods exploit execution time of tasks. Task runtime can be predicted by using historical information of “similar” task runs. That is because the execution time of similar task tends to be more similar than execution time of tasks that have no same task property. In this strategy, an important problem is how we can define similarity between task runs. In this paper, we propose a runtime prediction method which uses mutual information to define similarity measure and predictor selection based on confidence index and the number of task property in the set for measuring similarity. Experimental results indicate that the proposed method can predict task runtimes more accurately than previous methods.

1. はじめに

近年，計算機クラスタやグリッドコンピューティングといった，並列分散処理環境の利用がますます拡大している．それらの利用効率を向上させるためには，効率的なタスクスケジューリングや負荷分散システム等が必要になるが，提案されている手法の中には，投入されるプロセスの実行完了時間情報が得られることを前提として構築されているものが存在する[1),2)]．プロセス実行時間情報を用いることで，待ち時間に実行可能な後続タスクを詰め込むことが可能となり，応答時間を悪化させずに利用効率を改善できることが知られている．

それらを実際に並列分散システムに導入する際，どのようにしてプロセス実行前に実行完了時間の情報を用意するかが問題となる．この問題に対する解決策の 1 つとして，ユーザが実行時間の見積りをシステムに提出するという方法がある．しかし，Tsafrir ら[2)] の解析によって，一般にユーザはあまり正確な見積りを行えない傾向があることが確認されている．不正確な見積りの利用はシステム利用効率の向上を鈍らせ，あるいは逆に利用効率を悪化させてしまう．また，正確な見積りが得られればスケジューリング性能が向上することがいくつかの研究によって示されており[2)–4)]，より正確な実行時間情報を得ることは重要だと考えられる．

そのために，システム側で実行時間予測を行うことが考えられている．類似した特徴を持つプロセスどうしは，実行時間も類似していることが多い．予測対象のプロセスと類似したプロセスが過去に実行されていれば，その情報を用いて実行時間の予測を行うことが可能であり，ユーザの見積りよりも正確な場合があることが知られている．そのため，この考えに基づく実行時間予測手法がいくつか提案されている[3)–7)]．

これらの手法での基本的な枠組みはおおむね共通であり，次のように予測が行われる．まず，ジョブ管理システムや OS のプロセスアカウンティング等によって，プロセスの実行時間やプロセス情報を含む実行履歴が保存されているものとする．予測対象プロセスが指定されると，実行履歴から予測対象プロセスに似ているプロセスを複数選択する．それらの実行

†1 東北大学大学院工学研究科電気・通信工学専攻
Department of Electrical and Communication Engineering, Graduate School of Engineering, Tohoku University
†2 日本大学工学部
College of Engineering, Nihon University
*1 現在，日本電気株式会社
Presently with NEC Corporation

時間の統計量を予測値とする．統計量としては平均値や中央値等が考えられるが，文献 5) では平均値が良い予測結果を与えることが報告されている．

ここで，プロセス間の類似性をどのように規定するかということが非常に重要であり，予測精度を左右する．類似性はプロセス情報，すなわちユーザ名，プロセス名，使用メモリ量等の，プロセスに付随するプロセスを特徴付ける情報を利用して計られている．類似性の判断基準が適切ではない場合，本来は類似していないプロセスを類似プロセスと判断することになり，結果として実行時間予測精度を悪化させる．

Gibbons[3] は手動で決定された類似判定用テンプレートを用いた．類似判定用テンプレート（以降，単にテンプレートとも表記）とは，プロセス間の類似性判断に有用となるプロセス情報の部分集合，たとえば {ユーザ名, プロセス名, ノード数} であり，すべての属性値が一致したときに類似したプロセスと見なす．しかし，Gibbons の手法ではテンプレートが固定されているため，実行環境によっては予測が適切に行えないといった事態が生じる．

Smith ら[5] は，正解の得られている多数の予測結果から遺伝的アルゴリズムによってテンプレートを作成する手法を提案し，それを用いて実行時間予測を行っている．この手法では，予測のためにアドホックな知識を用いないため，実行環境に応じてテンプレートを生成することが可能で，それによって予測精度が向上することが報告されている．しかし，テンプレート作成には非常に時間がかかるため前もって行っておく必要がある．そのため，予測システム動作中の環境変化に対する検討は行われておらず，そのような場合には対応できないことが懸念される．

一方，Senger ら[4] の手法では，テンプレートを用いる代わりにプロセス情報に基づくプロセス間距離を定義し，一定数の近傍プロセスに関して距離の指数加重平均を求めることによって予測値を算出する．この手法は，従来手法のうちで最も予測精度が高いが，すべてのプロセス情報を同等に扱って距離を求める点に問題がある．実際には各プロセス情報の重要性には差があるはずであり，非常に重要なプロセス情報が重要でないプロセス情報と同等に扱われてしまうことで，結果的に真に類似しているプロセスを取得できない可能性がある．

また，複数の予測を行い，その中から有効な予測を選択する手法として，丹野ら[7] は信頼度による予測選択を導入している．信頼度による選択は，予測精度の向上にある程度の効果があるが十分ではない．さらに，この手法は固定されたテンプレートを使用しているため，Gibbons の手法と同じ問題がある．

本論文では，先に述べたような基本的な予測の枠組みを踏襲するが，テンプレートの新しい自動生成法，および複数の仮予測からもっともらしい予測を選択する新しい手法を導入することによって，より高精度な予測を目指す．まず，テンプレートの生成では，個々のプロセス情報の重要性を相互情報量を用いて計り，重要度の高いものから順に選んでテンプレートを自動的に生成する．その際，環境やタスクの多様性を考慮して複数のテンプレートを用意し，各テンプレートごとに予測値（仮予測値）を求める．求まった複数の仮予測値の中からより適切なものを選択するため，t 分布に基づく信頼度とプロセス情報の数を利用した予測選択を行う．テンプレートを自動生成することで，利用環境に応じた類似性判定基準を定めることができ，様々な環境への対応が可能となる．また，複数の仮予測を行い，効果的な予測選択を導入することで，最終的な実行時間予測精度の向上を図る．本論文では信頼度と予測精度の関係について改めて評価し，プロセス情報の数を信頼度とあわせて用いる予測選択手法を提案する．

なお，本研究では予測結果を CPU 資源の管理およびスケジューリングに応用することを想定しているため，実際のプロセス実行時間よりも十分に短い時間で予測処理を行う必要がある．実行時間が 60 秒以上で実行時間に最も影響を与える資源が CPU であるような計算タスクのみを予測の対象としており，対話的プロセスや極端に実行時間が短いプロセスは扱わない．

本論文の構成は以下のとおりである．2 章では提案する CPU 実行時間予測手法の概要について述べる．3 章では類似判定用テンプレートの生成法について述べる．4 章では予測実験結果とそれに基づく考察を示し，5 章ではまとめと今後の課題を述べる．

2. CPU 実行時間予測

提案する実行時間予測手法の概略は以下のとおりである．まず，プロセス間の類似性を判定するために重要となるプロセス情報を統計的に選択し，類似判定用テンプレートを複数生成する．1 章で述べたように，類似判定用テンプレートはプロセス情報の組であり，その値がすべて一致するプロセスを類似と見なす．なお，テンプレートの生成は予測のつど行うのではなく，事前に，または，ある間隔で行う．その後，テンプレートを用いて実行履歴から集められた類似プロセスを基に仮予測を実行する．1 つのテンプレートによって収集された類似プロセス群から，1 つの仮予測が生成される．最後に，仮予測の中から最適なものを 1 つ選択し，その仮予測値を最終的な予測値として出力する（**図 1**）．仮予測選択の基準として，t 分布によって定義される信頼度と，テンプレートのプロセス情報数に基づくヒューリスティックを用いる．

本章では，準備として実行履歴とプロセス情報について説明した後，主に仮予測の算出手

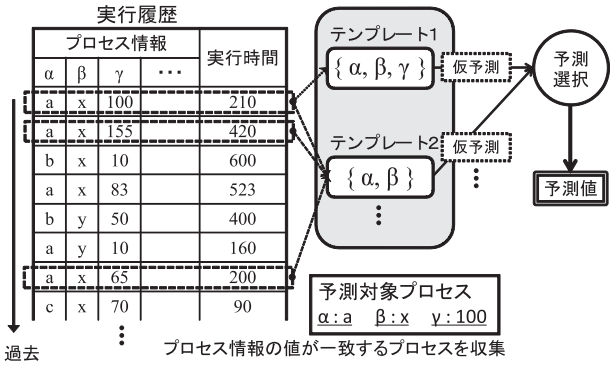

図 1　CPU 実行時間予測の概略図．ただし，テンプレート作成手順は含まない
Fig. 1　Outline of CPU run time prediction except constructing templates.

順と予測選択について述べる．類似判定用テンプレートの生成手法およびそれに関連した予測選択の改良については次章で詳しく述べる．

2.1　実行履歴とプロセス情報

実行履歴を記録するシステムの詳細については本論文の範囲を超えるので扱わないが，ジョブ管理システムまたは OS のプロセスアカウンティング等によって，プロセスの実行履歴が保存されているものとする．プロセスが終了するたびに，システムはそのプロセスの実行時間とプロセス情報を記録する．

プロセス情報とは，ユーザ名，プロセス名，グループ名，使用メモリ量，使用 CPU 数等各プロセスに付随する属性であり，その値は「文字列」または「数値」である．使用メモリ量，使用 CPU 数等，連続的で大小比較に意味があるものについては「数値」として扱い，それ以外のプロセス情報については「文字列」として扱う．

2.2　仮　予　測

複数のテンプレートを用いて，各テンプレートごとに以下の処理を行う．

(1)　類似プロセス候補の取得
(2)　仮予測値の算出
(3)　仮予測の信頼度の算出

2.2.1　類似プロセス候補の取得

実行履歴を探索し，類似プロセス候補を取得する．類似プロセス候補とは，テンプレートに含まれるすべてのプロセス情報について，予測対象プロセスと値が一致するものを指す．「値が一致する」かどうかは，次のように判定する．プロセス情報が文字列の場合は，完全一致した場合に値が一致と見なす．数値の場合は，予測対象プロセスのプロセス情報値で正規化した値の差を計算し，計算値が閾値以内ならば一致と見なす．なお，この閾値はプロセス情報ごとに設定することも考えられるが，本論文の実験においては実行時間もプロセス情報の 1 つと考え，想定する許容誤差に合わせてすべてのプロセス情報で 20%とした．ただし，プロセス実行時の途中の状態も記録されている場合には，実行途中の値の変動から差の累積値を計算し，累積値が閾値以内ならば一致と見なす[7]．

探索は予測時点から過去方向に行い，予測時点から過去一定期間探索をするか，類似プロセス候補を規定個数取得したならば探索を終了する．

2.2.2　仮予測値の算出

取得した類似プロセス候補に対してさらに絞り込みを行う．先の処理で取得したプロセス群は，ある程度類似したプロセスではあるが，重要なプロセス情報に関してさらに絞り込むことで，予測精度の向上が見込まれる．また，類似プロセス候補間においてプロセス情報の値が等しい場合でも，実行開始時刻の最新性に着目して類似プロセスを絞り込むことが望ましい．たとえば，デバッグを繰り返した後に本動作するようなプロセスは CPU 実行時間が大きく変化する可能性があり，後期に投入されたプロセスが予測対象プロセスにより類似していると考えられる．

まず，類似プロセス候補を以下の基準でソートする．テンプレートにおいて値が数値のプロセス情報がある場合は，その中で最も順位が高いプロセス情報の値に着目し，値に基づいて昇順に類似プロセス群をソートする．テンプレート中に数値のプロセス情報がない場合は，実行開始時刻が新しい順にソートする．ソート後，類似プロセス候補の CPU 実行時間において，クラス間分散とクラス内分散比が最大となるようにプロセス群を 2 分割する．2 集合のうち，ソート基準において予測対象プロセスに近いほうの集合を，最終的な類似プロセスとして取得する．ここで，プロセス情報の順位とは，次章で述べる類似判定用テンプレートを求める際に付けられた順位を指す．

以上の絞り込み処理の後，得られた類似プロセス群の CPU 実行時間平均値を仮予測値とする．

2.2.3　仮予測の信頼度の算出

複数の仮予測から最適なものを選択することで予測精度の向上が期待できるが，最適性をどのように判断するかが問題となる．本手法では，指標の 1 つとして文献 7) の手法を用い，各仮予測に対して t 分布を用いて信頼度を定義する．

t 分布は標本集団から母集団を検定する場合に用いられる確率分布である[8]．もし，真に類似したプロセスの集合が得られたならば，そのばらつきは測定誤差と考えられるため，正規分布に従う．そこで，予測対象プロセスに類似したプロセスの CPU 実行時間集合を母集団とし，それらは正規分布に従うと仮定する．予測対象プロセスの CPU 実行時間は母集団の平均値となる．そして，予測処理によって抽出された類似プロセスの CPU 実行時間集合は，母集団から標本抽出された集合と見なすことができる．よって，標本平均の母平均に対する検定の考え方が適用でき，抽出された類似プロセスの CPU 実行時間集合の t 値は t 分布に従う．

取得した類似プロセス集合と仮予測値に対して t 分布を用い，各仮予測の信頼度 *Confidence* を設定する．信頼度は，取得した類似プロセス集合を標本集団，仮予測値を標本平均，求めたい実測値（将来分かる値）を母平均と見なし，標本平均と母平均の差が標本平均の 10%以内に収まる確率と定める．すなわち，t 分布の密度関数 $g(t)$ を用い，次式で与えられる．

$$Confidence = 2\int_0^A g(t)dt. \tag{1}$$

ここで，$A = \frac{P_t \cdot 0.1}{\sqrt{S/n}}$ であり，P_t は算出した仮予測値，S は取得した類似プロセス群における CPU 実行時間の不偏分散，n は取得した類似プロセス数である．0.1 は 10%に対応する．なお，この 10%という数値は，想定する許容誤差や類似プロセス集合における実行時間の分布を考慮して経験的に定めたものである．ただし，想定する許容誤差が多少違っても多くの場合変更しなくてよい．t 分布の密度関数の性質から，この値を多少変化させても信頼度の順位は変わらない．また，自由度は $k = n - 1$ となる．取得した類似プロセス数が多いほど，CPU 実行時間がまとまっているほど信頼度は高くなる．すなわち，信頼度が高いということは，「この仮予測を生成したテンプレートは，似た CPU 実行時間を持つプロセスを集められる」ということを意味する．なお，自由度 k の t 分布の密度関数 $g(t)$ はベータ関数 $B(\lambda_1, \lambda_2)$ を用いて次の式で定義されている．

$$g(t) = \frac{1}{\sqrt{k}B(k/2, 1/2)}\left(1 + \frac{t^2}{k}\right)^{-\frac{k+1}{2}} \tag{2}$$

2.2.4　信頼度評価実験

類似プロセスの CPU 実行時間が正規分布に従うことは，あくまで仮定であるので，t 分布による信頼度が本当に予測精度に関係する値になっているかを確認しておく必要がある．CPU 実行時間の予測実験を行い，信頼度がある値以上となった予測結果の精度がどのような傾向にあるのかを検証した．

実験には，Parallel Workloads Archive[9] で提供されている LANL[10] CM-5 システムの実行履歴を使用した．このシステムで利用できるプロセス情報として，ユーザ名，プロセス名，グループ名，引数，メモリ使用量，使用 CPU 数，待ち時間の 7 つがある．

テンプレートは 2 種類用意した．予測選択は行わず，各テンプレートの予測精度と信頼度の関係を調べる．各テンプレートに含まれるプロセス情報を以下に示す．

テンプレート A　グループ名，引数

テンプレート B　グループ名，引数，CPU 数，メモリ使用量，ユーザ名

実行履歴探索期間は予測時点から過去 3 カ月とし，その期間に達するか類似プロセス候補を 10 個取得したならば探索を終了する．ランダムに選択した 1,079 個のプロセスについて予測実験を行い，誤差率による評価を行った．誤差率は次式で求められる．

$$error[\%] = \frac{|P_t - R_t|}{R_t} \cdot 100. \tag{3}$$

ここで，P_t は予測値，R_t は実測値を表す．誤差率が低い予測結果が多いほど，また，誤差率が高い予測結果が少ないほど，予測精度が良いということを意味する．

予測結果から信頼度がある値（信頼度閾値）以上の予測結果を取り出し，抽出された予測結果の中で誤差率が 20%以内の予測がどの程度の割合あるのかを調べた．その結果を信頼度閾値に対する割合の変化として**図 2** に示す．図 2 から，2 つのテンプレートによる予測とも，信頼度閾値が高いほど低誤差率の割合が高くなり，信頼度が高い場合は予測精度が非常に良いということが確認できる．

これらの結果から，計算された信頼度は，予測結果のもっともらしさを表す指標として有用だと考えられる．しかし，信頼度閾値が同じでも，2 つのテンプレートで誤差率 20%以下の割合に差があることから，絶対的な指標ではないことに注意する必要がある．

2.3　予 測 選 択

信頼度に基づく予測選択は，信頼度が最大となる仮予測値を最終的な CPU 実行時間予測結果とするものとした．ただし，1 つのテンプレートで類似プロセス候補が 2 つ未満の仮予測は信頼性が低いためリジェクトし，最終的な予測結果として選択しない．また，すべてのテンプレートに対する仮予測がリジェクトされた場合は予測を行わない．

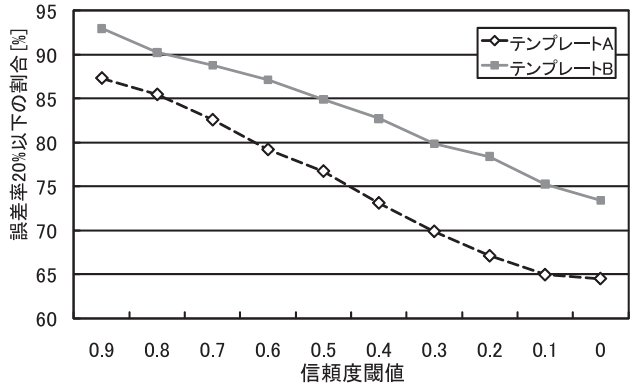

図 2　誤差率 20%以内となった予測結果の割合の変化
Fig. 2　Change of the ratio when error rates are not exceeding 20%.

3. 類似判定用テンプレート

本手法では，類似判定用テンプレートごとに仮予測を行う．類似判定用テンプレートとは，プロセス間の類似性を判断するために用いられるプロセス情報の集合である．ある 2 つのプロセスを比較するとき，テンプレートに含まれるすべてのプロセス情報について値が合致しているならば，その 2 つのプロセスは類似していると判断する．

テンプレートが類似性の判定に有効なプロセス情報からなっているならば，取得されたプロセスは，基準となったプロセスと非常に類似していると期待できる．ここで，有効なプロセス情報とは，似た CPU 実行時間値を持つプロセスを集めるのに役立つプロセス情報のことである．

テンプレートを生成するために，プロセス情報の CPU 実行時間に対する重要度を計算し，それに基づいて順位付けをする．その後，順位を考慮してテンプレートを複数個生成する．

3.1 プロセス情報の重要度順位付け

実行履歴に記録されているプロセス情報に関して，CPU 実行時間への影響力が高い順に順位付けを行う．影響力の評価は相互情報量[11] を利用する．相互情報量 $I(X;Y)$ は Y という条件を知ることで減少した X の不確かさの量を意味し，X と Y の依存性が強いほど値は大きくなる．プロセス情報と CPU 実行時間との間の相互情報量は，そのプロセス情報がどれほど CPU 実行時間に影響を与えるかを表している．その平均値を重要度と呼ぶ．

重要度第 k 位までを要素とするプロセス情報集合 D_k が求まっているとき，D_k のもとでの，あるプロセス情報 $a \notin D_k$ の重要度を次のように求める．

(1)　訓練データ中のあるプロセス i を基準とし，残りのプロセスとの比較を行う．各プロセスの CPU 実行時間値は，基準となるプロセスの実行時間で正規化しておく．2 つのプロセスの正規化実行時間の差が閾値（実験では想定する許容誤差に合わせて 20%）以内であれば類似と見なす．CPU 実行時間の値が i と類似している（事象 x_1）プロセスと，類似していない（事象 x_2）プロセスの数を調べ，それぞれを合計で割った数を生起確率とすることで $p_i(x)$，$x \in X$ を求める．$X = \{x_1, x_2\}$ は実行時間が基準プロセス i と「類似している」，または「類似していない」という事象からなる事象系である．

(2)　プロセス情報 a を，D_k に仮に加えた集合を $D_k^+(a)$ とする．$D_k^+(a)$ に含まれる全プロセス情報に関して，基準プロセス i と値が一致する（事象 y_1）プロセスと，一致しない（事象 y_2）プロセスの数を調べ，それぞれの生起確率 $p_i(y)$，$y \in Y$ を算出する．ここで，$Y = \{y_1, y_2\}$ は $D_k^+(a)$ のもとでプロセス情報が基準プロセス i と「一致する」，または「一致しない」という事象からなる事象系である．なお，「値が一致」するかどうかの判定は，類似プロセス候補の取得（2.2.1 項）の場合と同一である．

(3)　同様にして，(1)，(2) を同時に考慮した条件についてプロセスの数を数え，同時確率 $p_i(x, y)$，$x \in X$，$y \in Y$ を求める．

(4)　(1)，(2)，(3) で得られた確率を用いて，基準プロセス i のもとでの，事象系 X と Y の相互情報量 $I_i(X;Y)$ を以下のように計算する．

$$I_i(X;Y) = \sum_{y \in Y} \sum_{x \in X} p_i(x,y) \ln \frac{p_i(x,y)}{p_i(x)\, p_i(y)}.$$

(5)　基準プロセス i を変えながら同様に相互情報量を計算し，その平均を D_k のもとでのプロセス a の重要度 $Dependency(a,k)$ とする．

(6)　$Dependency(a,k)$ が最大のプロセス情報を第 $k+1$ 位のプロセス情報と決定し，$D_{k+1} = D_k \cup a$ とする．

$D_0 = \emptyset$ から始めて，逐次的にプロセス情報の順位を決定していき，最終的にすべてのプロセス情報について順位を定める．順位ごとに重要度を再計算する必要があるが，これは順位決定済みのプロセス情報集合を補完するプロセス情報を優先的に選択するためである．

3.2　重要度順位に基づいたテンプレート生成

求めたプロセス情報の重要度順位を考慮し，テンプレートを生成する．1 つの訓練データから N 個のテンプレートを生成し，テンプレート k（$1 \leq k \leq N$）には，前節で求めた重要度順に $k+1$ 位までのプロセス情報を含むようにする．すなわち，テンプレート $k = D_{k+1}$ とする．この生成法を TpN と呼ぶ．$N = 4$ のとき，Tp4 で得られるテンプレート集合は $\{D_2, D_3, D_4, D_5\}$ である．

これは，大まかな判断基準を持つテンプレートを基に，徐々に判断基準が厳しくなるようにテンプレートを生成することを意図している．判断基準が厳しいテンプレートによって取得された類似プロセス群は，非常に近い CPU 実行時間値を持つ可能性が高い．しかし，判断基準の厳しさから，類似プロセスをまったく取得できない場合も考えられる．そういった事態に対処するため，判断基準の厳しさに変化を付けてテンプレートの生成を行っている．N は利用できるプロセス情報を考慮しつつ，実験的に決定する．

3.3　重要度評価実験

相互情報量による重要度順位付けがうまく機能するか確認するため，生成されたテンプレートを用いて予測実験を行った．Tp4 に加えて Tp4 reverse というテンプレート生成法による予測を行い，比較を行う．

Tp4 reverse は比較のために用意したテンプレート生成法である．Tp4 と同様に 4 つのテンプレートを生成するが，1 つ目は上位 2 プロセス情報を使用し，以降は重要度順位が下位のプロセス情報から順に 1 つずつテンプレートに加えていく．Tp4 と Tp4 reverse の予測結果を比較すれば，本手法で用いた相互情報量による重要度順位付けが有効かどうか推測できる．

実験では 2.2.4 項で述べた実行履歴を用いた．重要度計算のために用いる訓練データには，予測開始前の 1,000 プロセスを用いる．それらを除いた 2,163 プロセスについて予測を行い，予測数と誤差率分布，正予測率による評価を行った．予測時の実行履歴探索期間も 2.2.4 項と同じである．

3.3.1　実験結果および考察

重要度順位付け結果は，上位から順に，ユーザ名，プロセス名，グループ名，CPU 数，メモリ使用量，引数，待ち時間となった．実験に用いたデータにおいて，待ち時間は CPU 実行時間とほぼ関連性がないが，本手法でも正しく最下位になっていることが確認できる．

この順位付け結果からそれぞれ 4 つのテンプレートを生成し仮予測を行う．Tp4 における各仮予測の予測結果と，信頼度による選択後の最終的な予測結果に関する誤差率分布を**表 1** に示す．表 1 において，仮予測 k とはテンプレート k を単独で用いた予測を示し，Tp4 は 4 つの仮予測から信頼度による選択を行った結果を表す．誤差率分布は，誤差率がある範囲内となった予測の，各手法で予測が得られた回数（表 1 の予測数）に対する割合で，誤差率 0 から 20%は値が高いほど，90%以上は値が低いほど予測精度が良いことを意味する．前者は予測がおおむね成功したケース，後者は予測が失敗したケースといえる．また，表 1 の正予測率は，誤差率 0 から 20%の予測数の全予測対象（2,163）に対する割合で，予測できなかった場合も予測失敗と考えたときの予測成功の割合である．

表 1　Tp4，Tp4 reverse による予測誤差率の分布と正予測率
Table 1　Distribution of prediction error rate and correct prediction rate using Tp4 and Tp4 reverse.

		誤差率分布 [%]		
予測手法	予測数	0–20%	90– %	正予測率
仮予測 1	2,115	67.18	9.98	65.70
仮予測 2	2,109	67.42	9.96	65.74
仮予測 3	2,069	70.28	9.09	67.22
仮予測 4	1,983	74.79	7.26	68.29
Tp4	2,073	72.03	9.50	69.02
Tp4 reverse	2,073	69.27	10.95	66.38

表 1 から，仮予測 1 から仮予測 4 の順に予測数は減少していくが，誤差率は改善されていくということが分かる．仮予測 1 から仮予測 4 の順でテンプレート中に含まれるプロセス情報は増加していく．追加された下位のプロセス情報が上位プロセス情報では判断できないものをうまく補い，類似性の判定性能が強化された結果であり，それによって正確な予測が可能となったと考えられる．予測選択後の予測結果を比較すると，Tp4 と Tp4 reverse の予測数は等しいが，誤差率分布は Tp4 の方が良い．正予測率も Tp4 が上回っており，重要度順位付けに従ってテンプレートを生成した方が良いといえる．

以上より，本手法で提案した相互情報量に基づく重要度順位付けは有効に機能していると考えられる．

3.4　複数の訓練データによるテンプレート生成と予測選択

1 つの訓練データから生成されるテンプレート群は，単一の重要度順位付け結果に基づいて生成されるため，判断基準の多様性に乏しい．すなわち，システムの実行時期によって投入されるプロセスの類似性の傾向が異なる場合，あるいはプロセスごとに類似性の傾向が変化する場合，1 つの訓練データベースから生成されたテンプレートでは，真に類似したプロセスを集めることができない可能性がある．

この問題への対処として，訓練データをある程度の大きさに分割し，それぞれテンプレートを作成するようにテンプレート生成法を拡張する．各訓練データは独自に重要度順位付けを行い，自身の順位結果のみを用いてテンプレートをそれぞれ生成する．これにより，プロセスごとに類似性の傾向の違いがあったとしても，対応できる可能性が高まると期待される．

以上のテンプレート生成法を用いた 2 つの選択法を提案する．1 つは次の予測手法である．

TpNxM　訓練データを時系列順に M 個に分割する．各訓練データは，自身の重要度順位結果から，3.2 節の方法で N 個ずつテンプレートを生成する．そして，計 $N \times M$ 個のテンプレートから信頼度による選択を行い，最終的な予測値を出力する．

ところで，2.2.4 項で述べたように，信頼度は仮予測の良さを表す指標として有用ではあるが，絶対的な指標ではない．また，予備実験により，プロセス情報を多く含み類似性の判断基準が厳しいテンプレートによる仮予測が，高い予測精度を示すことが確認されている．したがって，プロセス情報数が多いテンプレートによる仮予測を優先的に選択すれば，予測精度は向上すると考えられる．そこで，次の選択法を考えた．

TpNxMp　TpNxM と同様に $N \times M$ 個のテンプレートを生成する．予測選択では，初めは $N+1$ 個のプロセス情報を含んだテンプレート N による仮予測 M 個から信頼度による選択を行う．もし，そのすべてで予測ができなければ，判断基準を 1 段階緩くし，上位 N 個のプロセス情報を含むテンプレート $N-1$ による仮予測群から選択する．それでも予測ができなかった場合は，判断基準をさらに 1 段階ずつ緩めながら繰り返す．

3.5　テンプレート更新処理

予測を行う実行環境において，時期によって投入されるプロセスの類似性傾向が変化するならば，古いデータを用いて生成されたテンプレートでは予測がうまく行えない可能性がある．そこで，プロセス処理の合間にテンプレートを更新することを考える．

実行が完了したプロセスをある程度確保できたならば，それらを訓練データとして更新候補のテンプレート群を生成する．テンプレート更新処理は，プロセス情報数が同一のテンプレート集合ごとに行う．更新候補と同じものが既存のテンプレート集合に含まれている場合は，そのテンプレートの生成時刻の更新のみを行う．含まれていない場合は，生成時刻が最も古いテンプレートを更新候補で置き換える．

4. 評 価 実 験

提案手法（TpNxM，TpNxMp）の有効性を確認するため，CPU 実行時間予測実験を行った．実験では，2.2.4 項および 3.3 節と同じ LANL CM-5 の実行履歴を使用し，利用できるプロセス情報の数と訓練データの数を考慮して $N=4$，$M=4$ とした．この実行履歴は 24 カ月間，122,060 個のプロセスを含むが，初期の 4,000 個をテンプレート生成のための訓練データとする．残りの期間（3 カ月目以降）からランダムに 500 の時刻をサンプリングし，そのときに実行中だった 2,163 プロセスを予測対象プロセスとして，予測精度の評価に使用した．なお，これはランダムな時刻に予測要求があることを想定したものである．予測時の類似プロセス探索期間は，予測要求時刻からさかのぼって 3 カ月，または 10 個の類似プロセスを発見するまでの早いほうとした．

Tp4x4，Tp4x4p では，前述の訓練データから時系列順に 1,000 個ずつ 4 セット（計 4,000 個）を用いてテンプレートを生成する．この 2 手法のほかに，前述の Tp4 に加え，Tp4-4000 というテンプレート生成法での予測も行う．Tp4-4000 は，4 つの訓練データをまとめ，4,000 プロセスを 1 つの訓練データと見なしてテンプレートを 4 つ生成する．この手法では 4 つの訓練データの類似性傾向が統合され，“平均的な” テンプレートが生成されることになる．この手法と比較することで訓練データを分割することの意義が確認できる．

また，既存手法のうちで最も予測精度の高い Senger らの手法[4]（以後，IBL と表記する）と予測性能を比較する．IBL は次のように予測値を求める．予測対象プロセス x_q と過去のプロセス x_i とのプロセス間距離 $E(x_q, x_i)$ を求める．プロセス間距離は，各プロセス情報の不一致度の二乗和平方根として定義される．過去のプロセスをプロセス間距離が小さい順に k 個取得し，重みを

$$w(x_i) = \exp \frac{-E(x_q, x_i)}{\sigma^2}, \tag{4}$$

と定める．σ^2 は定数である．予測値 $f'(x_q)$ を各 x_i の実行時間 $f(x_i)$ を基に，次のような加重平均で求める．

$$f'(x) = \frac{\sum_{i=1}^{k} w(x_i) f(x_i)}{\sum_{i=1}^{k} w(x_i)}. \tag{5}$$

実験では，Senger らの論文で最良と判断されているパラメータ，$\sigma^2 = 0.125$，$k=5$ を用いた．

4.1　重要度順位付け結果および考察

各訓練データにおける重要度順位付け結果を**表 2** に示す．訓練データ欄の数値は，訓練データ中で新しいほうから数えて何番目から何番目までのプロセスを使用したかを表す．

表 2　重要度順位付けの結果
Table 2　Ranking results of task properties.

訓練データ	重要度順位付け結果
1–1,000	ユーザ名，プロセス名，グループ名，CPU 数，メモリ使用量，引数，待ち時間
1,001–2,000	引数，グループ名，ユーザ名，プロセス名，CPU 数，メモリ使用量，待ち時間
2,001–3,000	CPU 数，プロセス名，グループ名，ユーザ名，メモリ使用量，引数，待ち時間
3,001–4,000	引数，プロセス名，CPU 数，グループ名，メモリ使用量，ユーザ名，待ち時間
1–4,000	引数，プロセス名，グループ名，CPU 数，メモリ使用量，ユーザ名，待ち時間

各訓練データにおける重要度順位付け結果には違いがあり，訓練データとして指定する期間によって類似性の傾向が異なっているということが確認できる．また，4,000 プロセスを 1 つの訓練データとして使用した場合，その中に含まれる各訓練データのどれとも異なる結果となった．これは，訓練データをまとめることで，個々の訓練データにおける類似性の傾向が平均化されてしまった影響だと考えられる．

なお，テンプレートを生成するための処理時間は，Core^{TM} Duo T2300 を搭載した計算機（1 コアのみ使用）上で，1,000 プロセスの訓練データにおいては 1 つにつき 2.73 秒，4 つ合計で 10.90 秒であり，4,000 プロセスを 1 つの訓練データとして用いた場合は約 42.22 秒であった．訓練データを分割すると処理時間が大幅に短縮される．

4.1.1　CPU 実行時間予測結果および考察

予測結果に関する誤差率分布を**図 3** に示す．横軸が誤差率の範囲，縦軸が予測数に対する各誤差率の割合を表す．なお，1 回の予測値算出にかかる処理時間は，テンプレートを 4 個使用した場合は平均 0.73 ミリ秒，16 個使用時は 2.30 ミリ秒であった．

グラフから，Tp4，Tp4-4000，Tp4x4，Tp4x4p は，IBL よりも誤差率 0 から 10%の割合が高いことが分かる．しかし，この誤差率分布そのものでは，各手法間での定量的な比較がしにくい．そこで，許容できる誤差率と許容できない誤差率を想定し，前者を「予測成功」，後者を「予測失敗」の割合として評価を行う．まず後者について，システムによっては要求時間の 2 倍程度の経過時間でペナルティをかけることがあるので，これにマージンを入れて，90%以上の誤差率を「予測失敗」とする．次に，前者について検討する．もちろん予測誤差は少ないにこしたことはないが，予測には自ずから限界はあるはずなので，現実的で実現可能なマージンを設定する必要がある．図 3 を見ると，誤差率 20%以上はそれ未満と比べて平坦になっていることが分かる．この履歴に対し誤差率 20%未満はある程度高い割合で実現可能であり，また，多くの場合，応用上でも問題のない誤差率と考えられるため，これを「予測成功」と想定することにする．その結果を**表 3** に示す．なお，表の見方は，表 1 と同じである．

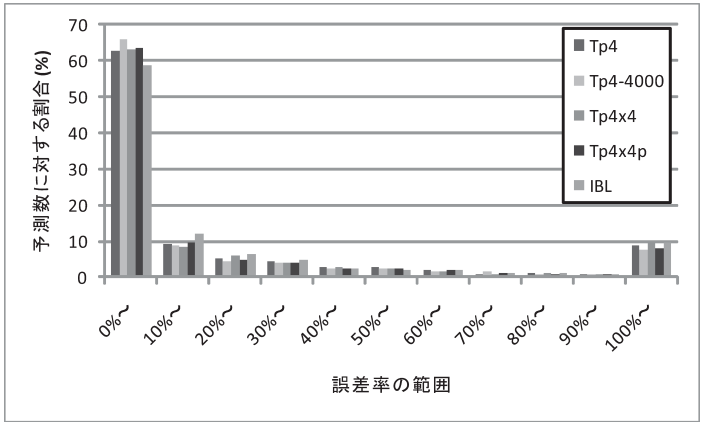

図 3　予測誤差率の分布
Fig. 3　Distribution of prediction error rate.

表 3　予測誤差率の分布と正予測率
Table 3　Distribution of prediction error rate and correct prediction rate.

		誤差率分布 [%]		
予測手法	予測数	0–20%	90– %	正予測率
Tp4	2,073	72.03	9.50	69.02
Tp4-4000	1,890	74.76	8.31	65.33
Tp4x4	2,158	71.31	10.43	71.15
Tp4x4p	2,158	73.26	8.71	73.09
IBL	2,163	70.50	10.31	70.50

まず，テンプレート作成に用いた訓練データ数とテンプレート数が異なる Tp4x4 と Tp4 を比較する．表 3 から，誤差率分布は Tp4 の方がわずかに良いが，予測数は Tp4x4 のほうが多いことが分かる．正予測率も Tp4x4 の方が良いことから，単に予測数が増加しただけでなく，その中に正確に予測できたケースが多数含まれていたことが分かる．Tp4x4 は複数の訓練データから多数のテンプレートを生成しているため，より多様なプロセスに対応できるテンプレートが得られたと考えられる．また，多数のテンプレートを用いることでより多くの仮予測を得ても，良い仮予測を選択できなければ正予測率は向上しないはずなので，信頼度による選択がある程度うまく機能しているということもいえる．

次に，同数の訓練データを用いている Tp4x4 と Tp4-4000 を比較する．誤差率分布では Tp4-4000 が Tp4x4 より勝っているが，Tp4-4000 の予測数は著しく少ない．その予測数の少なさがネックとなって，正予測率はすべての予測手法中で最も低い値であり，訓練データ数が少ない Tp4 よりも劣っている．訓練データのデータサイズを 4,000 プロセスとした場合，平均的なテンプレートが生成されたため，一部のプロセスの予測が行えず予測数が著しく減少したと考えられる．複数の訓練データを用いた場合は，様々な類似性判断基準を持つテンプレート群が生成される可能性が高く，多くのプロセスに対応できる．予測の安定性を考えると，データサイズの大きい 1 つの訓練データを用いるよりも，複数の訓練データに分割してテンプレートを生成すべきだと考えられる．

仮予測の選択手法が異なる Tp4x4 と Tp4x4p を比較すると，Tp4x4p の方が誤差率分布，正予測率がともに良いことが分かる．2.2.4 項で確認したように，信頼度は予測の尤もらしさの指標として有用ではあるが，絶対的な指標ではない．したがって，信頼度のみによる選択では不十分であり，プロセス情報数を用いた選択の導入によって厳しい条件のテンプレートを優先することで，適切な仮予測が選択できたと考えられる．

最後に Tp4x4p と従来手法である IBL とを比較する．表 3 より，両手法で予測数に大きな差がないことが分かる．提案手法は類似プロセスを規定個数以上取得できなければ予測が行えないが，複数の訓練データから多様なテンプレートを生成することで，予測不能となる事態をほとんど回避できている．誤差率分布および正予測率について比較すると，ともに Tp4x4p が IBL よりも良い結果となった．IBL では個々のプロセス情報の重要性の違いは考慮されないが，提案手法では個々の重要度を考慮してテンプレートを生成し，予測を行う．どのプロセス情報を重視するかといった選別が，誤差率分布の改善に作用したと推測される．Tp4x4p と IBL の正予測率の差は 2.59 ポイントだが，2,163 個のプロセスのうち 56 個が，許容誤差率（20%を想定）外から内に移行したことになる．スケジューリング手法やシステムの仕様によるが，実行時間が許容誤差を超えた場合に実行停止等，何らかのペナルティを受けたり，再スケジューリングが必要になったりする場合があり，無視できない影響を与えることがあると考えられる．

4.2　テンプレート更新時の予測性能評価

提案手法において，予測の合間にテンプレート更新を行った場合の予測性能評価を行った．Tp4，Tp4x4，Tp4x4p は，予測対象でなかったものも含めて 1,000 プロセスが実行完了するごとに，Tp4-4000 は 4,000 プロセスごとに，それらを訓練データとしてテンプレート更新処理を行う．テンプレート更新処理を行った場合の予測結果に関する誤差率分布および正予測率を，**表 4** に示す．

表 4　テンプレート更新時の予測誤差率の分布と正予測率
Table 4　Distribution of prediction error rate and correct prediction rate with template update.

予測手法	予測数	誤差率分布 [%] 0–20%	誤差率分布 [%] 90– %	正予測率
Tp4	2,074	73.00	9.84	70.00
Tp4-4000	2,051	71.72	9.95	68.00
Tp4x4	2,149	72.03	10.47	71.57
Tp4x4p	2,149	73.75	9.31	73.28

表 3 と比較すると，Tp4-4000 においては誤差率分布は悪くなっているものの，予測数は非常に大きく増加していることが確認できる．これは，類似プロセスを十分に収集できないテンプレートが初期設定されていたとしても，予測の合間にテンプレート更新処理を行うことで適したテンプレートが生成され，予測ができるようになる場合があるということを示唆している．

次に，Tp4-4000 以外の予測結果について，同様に更新処理の影響を確認する．予測数がわずかに減っているものがあるが，正予測率に悪影響を与えておらず大きな変化ではない．正予測率はいずれも増加しているものの，特に Tp4x4，Tp4x4p では向上分がわずかであることから，更新処理の有効性が確認できたとはいえない．これらのケースでは，初期テンプレートでも有効な予測が行える状態であったと考えられる．しかし，Tp4x4，Tp4x4p において特に不利益がなく，Tp4-4000 では予測数の大幅な増加に貢献していることから，不適切な初期テンプレートや特徴的なタスクの大量投入等にともなう類似性の変化に備えて，テンプレート更新処理を導入することは検討に値する．

4.3　他のシステムでの評価

前節では LANL の履歴による評価と議論を行ってきたが，本節では他のシステムの履歴による評価を行う．Parallel Workloads Archive [9] で提供されている，SDSC（San Diego Supercomputer Center）の Intel Paragon（SDSC96），CTC（Cornell Theory Center）の IBM SP2，HPC2N（High-Performance Computing Center North in Sweden）の Linux クラスタ，SHARCNET クラスタを用いて実行時間予測の実験を行った．

それぞれのシステムで使用したプロセス情報は**表 5** のとおりである．SDSC96 では利用できるプロセス情報が 3 つしかないため，$N = 2$，$M = 4$ とした．他の実験条件は前節の実験と同じである．**表 6**，**表 7**，**表 8**，**表 9** にそれぞれ結果を示す．なお，表中の（更新）は，テンプレート更新を行ったものを示す．

表 5　実験に使用したプロセス情報．ただし，SD，CT，HP，SH は SDSC96，CTC，HPC2N，SHARCNET をそれぞれ表す
Table 5　Task properties used for the experiments. SD, CT, HP and SH are abbreviations of SDSC96, CTC, HPC2N and SHARCNET, respectively.

プロセス情報	SD	CT	HP	SH
待ち時間		○	○	○
使用 CPU 数	○	○	○	○
メモリ使用量			○	○
要求 CPU 数			○	
要求時間		○	○	
要求メモリ量			○	
ユーザ名	○	○	○	○
グループ名			○	
プロセス名		○		○
キュー	○	○		
パーティション			○	○

表 6　SDSC96 での予測精度
Table 6　Prediction accuracy for SDSC96.

		誤差率分布 [%]		
予測手法	予測数	0–20%	90– %	正予測率
Tp2x4	5,404	62.82	5.69	62.22
Tp2x4p	5,404	62.06	5.61	61.47
Tp2x4（更新）	5,404	62.82	5.69	62.22
Tp2x4p（更新）	5,404	62.06	5.61	61.47
IBL	5,456	63.69	5.85	63.69

SDSC96 では，誤差率分布，正予測率ともに，提案手法は IBL よりやや悪い．提案手法では，相互情報量を用いて各プロセス情報の重要度を算出し，重要なプロセス情報からなるテンプレートを複数生成することに特徴があるが，SDSC96 ではわずか 3 つのプロセス情報しか利用できず，異なる有効な組合せを作成し，選択する余地がなかったことが主な原因と考えられる．実際，Tp2x4 では最大で 8 つのテンプレートが生成されるはずだが，多くとも 4 つのテンプレートしか生成されなかった．

CTC では，誤差率分布，正予測率ともに，提案手法が IBL を上回っている．CTC では 6 つのプロセス情報が利用でき，前述の SDSC96 と比べて利用できるプロセス情報が多いため，重要なプロセス情報を含むテンプレートを複数生成する戦略がうまく働いたものと考えられる．

表 7　CTC での予測精度
Table 7　Prediction accuracy for CTC.

		誤差率分布 [%]		
予測手法	予測数	0–20%	90– %	正予測率
Tp4x4	5,211	63.62	8.52	63.56
Tp4x4p	5,211	65.59	7.85	65.53
Tp4x4（更新）	5,211	63.89	8.42	63.83
Tp4x4p（更新）	5,211	65.63	7.89	65.57
IBL	5,216	61.52	7.62	61.52

表 8　HPC2N での予測精度
Table 8　Prediction accuracy for HPC2N.

		誤差率分布 [%]		
予測手法	予測数	0–20%	90– %	正予測率
Tp4x4	5,118	80.74	2.87	80.43
Tp4x4p	5,118	80.64	2.46	80.33
Tp4x4（更新）	5,132	79.11	3.99	79.02
Tp4x4p（更新）	5,132	79.39	2.65	79.30
IBL	5,138	75.87	3.19	75.87

表 9　SHARCNET での予測精度
Table 9　Prediction accuracy for SHARCNET.

		誤差率分布 [%]		
予測手法	予測数	0–20%	90– %	正予測率
Tp4x4	5,127	77.53	1.58	58.59
Tp4x4p	5,127	77.55	1.68	58.61
Tp4x4（更新）	5,127	79.00	1.10	59.70
Tp4x4p（更新）	5,127	96.86	1.17	73.20
IBL	6,784	37.75	3.02	37.75

HPC2N，SHARCNET でも，誤差率分布，正予測率ともに，提案手法が勝っている．特に SHARCNET では，テンプレート更新ありの Tp4x4p の誤差率分布が非常に優れており，IBL の予測精度がきわめて悪い．プロセス情報の重要性に大きな偏りがあり，また，時間的な変化もあったためである．IBL はすべてのプロセス情報を同じように扱うため，重要性に偏りがある場合，重要なプロセス情報を過小評価し，反対に重要でないプロセス情報を過大評価することで問題を引き起こす．この影響が，SHARCNET では非常に強く出たものと思われる．たとえば，SHARCNET は構成の異なる複数の Opteron クラスタからなってお

り，全体としては非均質であるが，プロセスがどのクラスタで実行されたかはプロセス情報「パーティション」で示されており，特に重要なプロセス情報といえる．

5. ま と め

本論文では，相互情報量を用いてプロセス間の類似性を判断し，t 分布信頼度とテンプレートのプロセス情報数に基づく予測選択を導入したプロセス実行時間予測法（TpNxM, TpNxMp）を提案した．実験の結果，利用できるプロセス情報が十分にある場合，既存の手法で最も性能の高い IBL よりも良い予測性能を示すことが確認できた．また，テンプレート生成（更新）にかかる時間は 2.73 秒，予測にかかる時間は 2.30 ミリ秒である．本論文では 60 秒以上のタスクを対象としているが，それと比較して予測は十分に高速であり，粗粒度タスクスケジューリングや資源管理システム等での利用が期待できる．テンプレート更新は予測 1 回ごとに行えるほど高速ではないが，数秒のアイドル時間が確保できれば実行可能なので，ある程度の間隔ごとに行える程度の処理時間だと思われる．

本論文では，テンプレート生成に使用する訓練データとして，時間的に連続した 1,000 プロセスを用いている．これは，経過時刻によって投入されるプロセスの傾向が変化する，いい換えれば，投入されるプロセスの傾向には時間的局所性があるという仮定に基づいているが，その仮定が成り立つかどうかは環境に依存すると思われる．時刻に依存しない場合でも，他の属性によって傾向が変化するかもしれない．その場合，プロセス群を同じ傾向を持つ集合にあらかじめクラスタリングし，得られた集合を訓練データとして指定する方法等が考えられる．パラメータの設定方法についても，今後検討を行う必要がある．

また，本論文では高精度の実行時間予測手法を構築することに注力し，予測結果の利用法について具体的には検討していない．より高精度な予測は，より効率的なスケジューリングに役立つと期待されるが，実際にスケジューリング等に適用し効果を確認すること，および本手法に適したスケジューリング手法を開発することは，今後の課題である．本手法では信頼度として予測の自信度のようなものが得られるため，これをスケジューリングに利用していくことも考えられる．

謝辞　本研究の一部は，科学研究費補助金 若手研究（B）（課題番号：19700052）による補助のもとで行われた．

参 考 文 献

1) Kim, S.C., Lee, S. and Hahm, J.: Push-Pull: Deterministic Search-Based DAG Scheduling for Heterogeneous Cluster Systems, *IEEE Trans. Parallel and Distributed Systems*, Vol.18, No.11, pp.1489–1502 (2007).
2) Tsafrir, D., Etsion, Y. and Feitelson, D.G.: Backfilling Using System-Generated Predictions Rather than User Runtime Estimates, *IEEE Trans. Parallel Distributed Systems*, Vol.18, No.6, pp.789–803 (2007).
3) Gibbons, R.: A Historical Application Profiler for Use by Parallel Schedulers, *Proc. Job Scheduling Strategies for Parallel Processing*, Feitelson, D.G. and Rudolph, L. (Eds.), pp.58–77, Springer Verlag (1997).
4) Senger, L.J. and Santana, M.J.: An Instance-based Learning Approach for Predicting Execution Times of Parallel Applications, *Proc. 3rd International Information and Telecommunication Technologies Symposium*, pp.9–15 (2004).
5) Smith, W., Foster, I. and Taylor, V.: Predicting application run times with historical information, *Journal of Parallel Distributed Computing*, Vol.64, No.9, pp.1007–1016 (2004).
6) Krishnaswamy, S., Loke, S. and Zaslavsky, A.: Estimating computation times of data-intensive applications, *Distributed Systems Online, IEEE*, Vol.5, No.4, pp.1–12 (2004).
7) 丹野祐樹，菅谷至寛，阿曽弘具：プロセス情報を利用した実行時間予測と信頼度による予測選択手法，情報科学技術レターズ，Vol.6, pp.21–23 (2007).
8) 鈴木義也ほか：概説数理統計，共立出版 (1994).
9) Parallel Workloads Archive. http://www.cs.huji.ac.il/labs/parallel/workload/
10) LANL (Los Alamos National Lab). http://www.lanl.gov/
11) 瀧　保夫：情報論 I，岩波書店 (1978).

(平成 22 年 7 月 14 日受付)
(平成 22 年 11 月 5 日採録)

菅谷　至寛（正会員）

1995 年東北大学工学部通信工学科卒業．2002 年同大学大学院工学研究科電気・通信工学専攻博士後期課程修了．2000 年より同大学院工学研究科助手．2007 年同助教．博士（工学）．並列処理アーキテクチャ，並列ソフトウェア，分散処理システム等の研究に従事．パターン認識，画像処理等にも興味を持つ．電子情報通信学会会員．

丹野　祐樹

2007 年東北大学工学部通信工学科卒業．2009 年同大学大学院工学研究科電気・通信工学専攻博士前期課程修了．同年日本電気株式会社入社．在学中，並列・分散処理に関する研究に従事．

大町真一郎（正会員）

1988 年東北大学工学部情報工学科卒業．1993 年同大学大学院工学研究科情報工学専攻博士後期課程修了．同年同大学情報処理教育センター助手．1996 年同大学工学部助手．1999 年同大学大学院工学研究科助教授．2009 年同教授．博士（工学）．その間，2000〜2001 年米国ブラウン大学客員准教授．パターン認識，コンピュータビジョン，並列処理，文字認識システムの開発等の研究に従事．2007 年 MIRU 長尾賞，IAPR/ICDAR Best Paper Award 各受賞．IEEE，電子情報通信学会，人工知能学会等各会員．

阿曽　弘具（正会員）

1968 年東北大学工学部電気工学科卒業．1974 年同大学大学院工学研究科電気および通信工学専攻博士後期課程修了．1973 年同大学工学部助手．1979 年名古屋大学工学部講師．1982 年同大学助教授．1986 年東北大学工学部助教授を経て，1991 年同大学教授．2009 年日本大学工学部教授．工学博士．その間，学習オートマトン，セル構造オートマトン，並行処理理論，シストリックアルゴリズム設計論，文字認識，音声認識，ニューラルネットワーク等の研究に従事．1991 年度電子情報通信学会業績賞受賞．IEEE，ACM，EATCS，電子情報通信学会，人工知能学会，LA 各会員．