資源情報サーバにおける資源情報予測の評価

小出 洋[†] 山岸信寛[†]
武宮 博[†,††] 笠原博徳[†††]

分散コンピューティング環境における効率的なタスクスケジューリング実現の一環として，我々は，分散コンピューティング環境上のプロセッサやネットワークの負荷など計算資源に関する情報（資源情報）を収集し，それをもとに将来の値を予測する資源情報サーバ（Resource Information Server；RIS）を開発している．1つのプログラムの実行時間の最小化を目的とするスケジューラは，プロセッサやネットワークの負荷予測に基づいて，動的にタスクを割り付けるため，RIS は必須である．現在，高速と高精度の 2 種類の資源情報予測を行うモジュールが RIS に実装されている．高速予測モジュールは，予測を行う時点の最近接過去に記録された資源情報だけを使用し，将来の資源情報の値を迅速に予測する．高精度予測モジュールは，予測時点の最近接過去の資源情報の変化と類似した負荷パターンを過去のデータから検索するため，予測時間を要するが，より高い精度で資源情報を予測することができる．要求精度と予測時間に応じて，これらのモジュールを使い分けることにより，将来の資源情報を効率的に予測することができる．本論文では，資源情報の計測と予測方法，RIS のシステム構成，予測時間と予測精度に関する評価について述べる．

Evaluation of the Resource Information Prediction in the Resource Information Server

Hiroshi Koide,[†] Nobuhiro Yamagishi,[†] Hiroshi Takemiya[†,††] and Hironori Kasahara[†††]

For the purpose of realizing effective task scheduling in the distributed computing environment, we have developed the resource information server (RIS) which measures the resource information, CPU and network loads, on a cluster of heterogeneous supercomputers and predicts the future resource information. RIS is indispensable for a scheduler to dynamically allocate tasks in order to minimize the execution time of a single program based on dynamically predicted CPU and network loads. The prototype system of RIS has two kinds of modules to predict future resource information. The first module quickly predicts future resource information by using only the most recent past resource information. The second module predicts it with much smaller error, but with use of much more time by searching the most similar sequence of resource information to a recent trend of resource information. The prototype system effectively forecasts resource information any time in the future by switching these modules according to required accuracy and given computation time. In this paper, the measuring and forecasting methods of the resource information, the implementation of the system, and evaluations of computation time and accuracy of the resource information forecast are described.

1. はじめに

近年，さかんに研究されているメタコンピューティングは，ネットワークで接続された並列計算機，ワークステーション，データベース等から構成される分散コンピューティング環境を，科学技術計算などの高速化や大規模化等のために効率的に利用する技術である[1)~4)]．

メタコンピューティング環境は，複数の異なるアーキテクチャの計算機から構成されており，一般に多くの利用者により共有されるため，計算機やネットワークの負荷が時間的に変動する．このような特徴を持つメタコンピューティング環境においては，計算機やネッ

† 日本原子力研究所計算科学技術推進センター
Center for Promotion of Computational Science and Engineering, Japan Atomic Energy Research Institute
†† 日立東北ソフトウェア
Hitachi Tohoku Software, Ltd.
††† 早稲田大学電気電子情報工学科
Department of Electrical, Electronics and Computer Engineering, Waseda University

トワークの負荷変動を考慮したプログラミングを行う必要がある．このためのインフラストラクチャとして，メタコンピューティング環境でタスクやジョブを計算資源に効率的に割り当てることができるスケジューリングシステムが必要となる．一例をあげると，早稲田大学と日本原子力研究所では，メタコンピューティング環境における実行時間最小化を目的としたタスクスケジューリングの問題に焦点を絞り，課金，セキュリティ，フォールトトレランス等の問題を扱わなくて済む分散コンピューティング環境である並列計算機クラスタ COMPACS（*COMplex PArallel Computer System*，日立 SR2201，SR2201 ディスクサイドモデル，NEC SX-4，富士通 VPP300，CRAY T94，IBM SP-3，SGI Power Onyx 10000 等から構成される）上で，この環境に対応したメタスケジューリング[5]の開発，評価を行っている．

利用者は，逐次プログラムさえ用意すれば，メタスケジューリングにより，異機種構成の並列計算機クラスタ上で自動並列分散実行することができ，実行時間を短縮化することができる．現在，この目的のために使用されているスケジューリングアルゴリズムの 1 つである CP/ETF/MISF for Meta-scheduling は，制御およびデータ依存関係を持つ粗粒度タスク集合を，プログラム全体の実行時間がなるべく短くなるように，クラスタ上の各並列計算機に自動的に割り付けることができる[5]．

資源情報サーバ（Resource Information Server；RIS）[6]は，分散コンピューティング環境上のプロセッサやネットワークの負荷など計算資源に関する情報（資源情報）を収集し，それをもとに将来の値を予測する．CP/ETF/MISF for Meta-scheduling は，RIS が予測した資源情報をもとに各計算機上での各タスクの実行時間を見積もり，そのタスクの処理を最も早く終了すると予測される計算機に割り当てることによりタスクスケジューリングを行っている[5]．

現在の RIS の実装システムでは，高速と高精度の 2 つの資源情報の予測を行うモジュールを実装している．高速予測モジュールは，予測を行う時点の最近接過去に記録された資源情報だけを使用し，近い将来の資源情報の値をできるだけ迅速に予測する．高精度予測モジュールは，最近接過去の資源情報の変化と類似した負荷パターンを過去の資源情報を記録したデータベースから検索し，それをもとに予測するため，高速予測モジュールと比較してより長い計算時間が必要だが，より高い精度で資源情報を予測することができる．現在の RIS の実装システムは，要求精度と予測時間に応じて，これらのモジュールを使い分けることにより，任意の将来の資源情報を効率的に予測することができる．

本論文では，RIS のシステム構成，RIS における資源情報の計測と予測方法，予測時間と予測精度に関する評価について述べる．

2. 資源情報サーバ：RIS

2.1 RIS のシステム構成

RIS システムは，資源情報 GUI，資源情報 API ライブラリ，RIS 本体から構成されている（**図 1**）．

資源情報 GUI は，利用者からの資源情報の確認や RIS の動作に関する設定等の要求を RIS に伝え，RIS からの出力を利用者に提示するグラフィカル・ユーザ・インタフェースである．この GUI は，STA 基本システム[7]の GUI サブシステム上で稼働する Java アプレットとして実現されている．

資源情報 API ライブラリは，利用者プログラムが RIS に資源情報の予測要求を行い，予測結果等を利用者プログラムに返すライブラリである．

RIS 本体は，以下のモジュールから構成されている．

資源情報モニタ： ネットワーク上の各計算機におけるすべてのプロセッサに関する資源情報，各計算機間のネットワークに関する資源情報を収集する．

資源情報プローバ： RIS から各並列計算機に生成されるプロセスであり，実際に資源情報の計測を行う．

資源情報 API： 利用者プログラム，または，資源情報 GUI（以下，クライアントと呼ぶ）からの要求を受け付け，資源情報プレディクタに資源情報の予測，または，資源情報データベースに検索を指示する．その予測結果や検索結果を要求元に返答する．

資源情報プレディクタ： 資源情報 API からの指示に従い，指定された方法で資源情報を予測する．その結果を資源情報 API に通知する．

資源情報データベース： 資源情報モニタが収集した資源情報をデータベースに蓄積する．資源情報プレディクタ，資源情報 API からの要求に従い，収集済みの資源情報を要求元に通知する．

2.2 資源情報の予測方法

現在のところ，RIS が収集した過去の資源情報をもとに資源情報を予測する，前述の高速予測モジュールと高精度予測モジュールの 2 つの資源情報予測モジュールのプロトタイプを実装している．

高速予測モジュール： 予測を行う時点の最近接過去に記録された資源情報のみをもとに近い将来の資源情報を迅速に予測することを目的としている．プロ

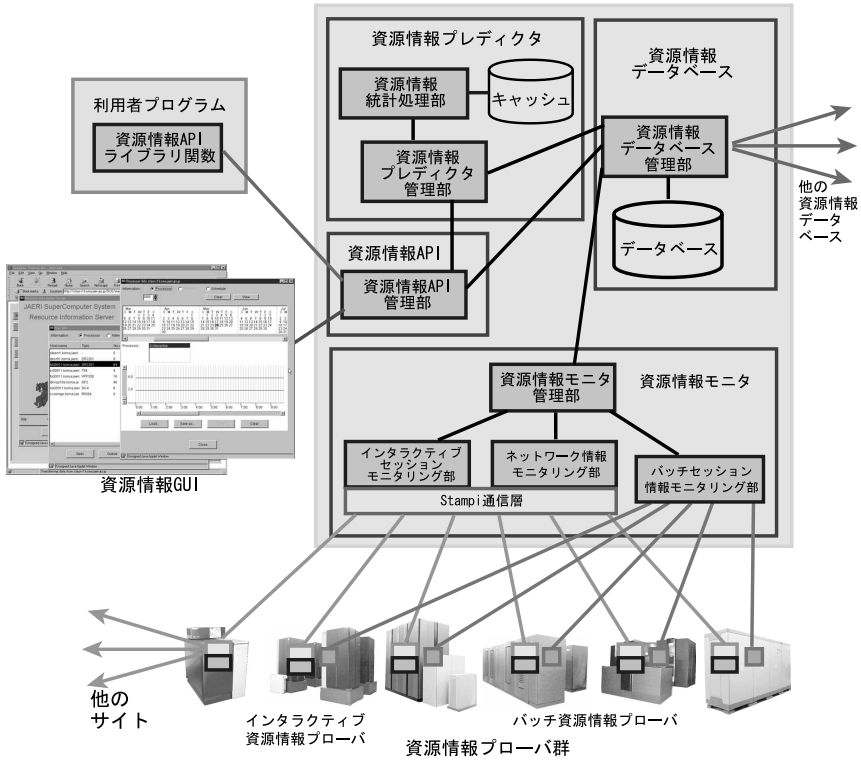

図 1 RIS システムの構成
Fig. 1 Configuration of RIS system.

トタイプでは，最近接過去の p 秒間のデータの平均値を将来の資源情報の予測値として出力する．

高精度予測モジュール： 予測を行う時点の資源情報の変動パターンと類似したパターンを過去の資源情報データベースから検索し，これをもとに予測を行う．現在実装しているプロトタイプは，以下の手順により，将来の資源情報の予測値を計算する（**図 2**）．

(1) 事前に，過去の資源情報の q 秒ごとの平均値を計算し，時間順に $v_0, v_1, \cdots, v_{n-1}$ とする．なお，資源情報データベースには，nq 秒間の資源情報の記録があるものとしている．

(2) 最近の wq 秒間，q 秒ごとの資源情報の平均値を w 個分計算し，時間順に $u_0, u_1, \cdots, u_{w-1}$ とする．

(3) 以下で定義される類似度 $R(x)$ が最小になるインデックス x を求める．

$$R(x) = \sum_{i=0}^{w-1} (v_{x+i} - u_i)^2$$

(4) 今後の資源情報の変化の様子は，過去 wq 秒間の資源情報の変化と最も類似した過去のその後の変化と同様と仮定する．つまり，今後 q 秒ごとの予測値は，時間順に $v_{x+w}, v_{x+w+1}, v_{x+w+2}, \cdots$ とする．

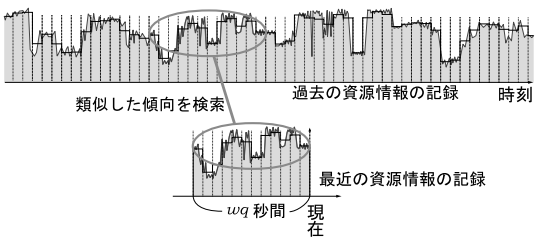

図 2 高精度予測モジュールのアルゴリズム
Fig. 2 Algorithm of the accurate prediction module.

3. RIS の実装

3.1 提供する資源情報

現在の RIS の実装システムでは，**表 1** に示す資源情報を扱っている．このほか，データベースに記録された動的に変化しない資源情報として，運用時間や運用方法等に関する情報，主記憶領域情報（使用可能な主記憶の総量），ファイル領域情報（使用可能なファイル総量とファイルの有効期限）も提供している．

3.2 各モジュールの連携した動作

RIS を構成する各モジュールは，以下のように連

表 1 現在の RIS が取り扱う資源情報
Table 1 Resource informations handled by the current RIS.

プロセッサ負荷	その時点で，あるプログラムを実行すると，ほかに負荷がない場合の何倍実行時間を要するかを表す.
プロセッサ遅延	その時点でのプロセス起動要求から実際にプロセスが起動されるまでに要する時間を表す.
ネットワーク負荷	その時点での各並列計算機間のネットワークバンド幅（K バイト/秒）を表す.
ネットワーク遅延	その時点でのパケットの先頭データが相手に届くまでの時間を表す.

携して動作することにより，資源情報の収集と予測を行う．

資源情報の収集：

(1) 資源情報モニタは，異機種並列計算機間通信ライブラリ Stampi[8]を使用して，資源情報を実測するためのプロセス（プローバ）を測定対象の並列計算機に 1 つずつ起動し，資源情報の実測値を収集する（図 1）.

(2) 起動された各プローバは，以下の実測を行い，表 1 に示す資源情報を収集する．

(a) 起動したプローバから，Stampi を使用して，並列計算機内の各プロセッサに測定用プロセスを起動し，プロセッサ負荷とプロセッサ遅延を測定する．

(b) 測定対象リストに登録された 2 台の並列計算機間のネットワーク負荷，および，遅延を測定する．起動したプローバからリストにある 2 台のうち一方の並列計算機に，Stampi を使用して測定用プロセスを起動し，2 台の計算機間のネットワークの負荷と遅延を測定する．

(3) 収集された資源情報の実測値は，資源情報データベースに送られ，そこで記録される．

資源情報の予測：

(1) クライアントは，資源情報 API ライブラリを使用して，資源情報 API に対し資源情報の予測要求を行う．この際，予測に必要な情報（利用する予測モジュール，予測対象のプロセッサやネットワーク，予測対象の期間）も資源情報 API に送る．

(2) 資源情報 API は，クライアントからの予測要求の受付けを行い，予測要求をキューに入れ管理する．また，資源情報プレディクタに予測要求をキューから順に転送する．

(3) 資源情報プレディクタは，指定された方法に対応する予測モジュールにより，指定された資源情報の予測を行い，予測結果を資源情報 API に返答する．予測に必要な過去の資源情報の記録は資源情報データベースから得る．

(4) 資源情報 API は，資源情報プレディクタから返された予測結果を，要求元クライアントに返答する．

4. 予測精度の性能評価実験

資源情報の予測に必要な計算時間と予測精度を評価するため，RIS が実際に収集した資源情報をもとに各予測モジュールが資源情報の予測を行う性能評価実験を行った．性能評価実験の手順は，以下のとおりである．

(1) あらかじめ，各並列計算機の各資源情報の約 1 分ごとの資源情報を，長期間にわたり収集する．以下，資源情報が収集されている期間を資源情報収集期間と呼ぶ．

(2) RIS が収集した資源情報のうち，予測対象となる計算機，資源情報を抜き出し，q 秒ごと（たとえば，300 秒ごと）の平均値を計算する．

(3) 資源情報の各予測モジュールによる資源情報の予測を行う．資源情報収集期間中のある特定の期間（以下，予測期間と呼ぶ）を選び，その期間中の q 秒ごとの時点において，その時点から先の資源情報をそれ以前の資源情報を使用して予測する．

(4) 実測値と各予測モジュールの予測結果から，予測精度を計算する．

性能評価実験では，COMPACS を構成する 1 台である日立 SR2201 における 64 プロセッサのうち，インタラクティブ運用に割り当てられている 16 プロセッサの負荷の最大値を予測対象の資源情報として選択した．複数プロセッサの負荷の最大値を予測対象とする理由は，すべてのプロセッサを使用して並列計算を行う場合，一般に負荷が最大のプロセッサがボトルネックになり，計算時間が決まることが多いからである．

ここで，本評価実験におけるプロセッサ負荷と予測誤差を定義しておく．まず，プロセッサ負荷は，(経過時間)/(CPU 時間) であり，ある単位時間で終了するプログラムが無負荷のときの何倍の計算時間で終了するかを表す値である．また，本評価実験で評価する各モジュールの予測誤差は，実測値 $u_w, u_{w+1}, u_{w+2}, \cdots$ と予測値 $v_{x+w}, v_{x+w+1}, v_{x+w+2}, \cdots$ から以下の式で与えられるプロセスに割り当てられる CPU 時間の差として求める．なお，以下の式における x は，2.2 節の類似度 $R(x)$ を最小にする過去の実測値のインデックスである．

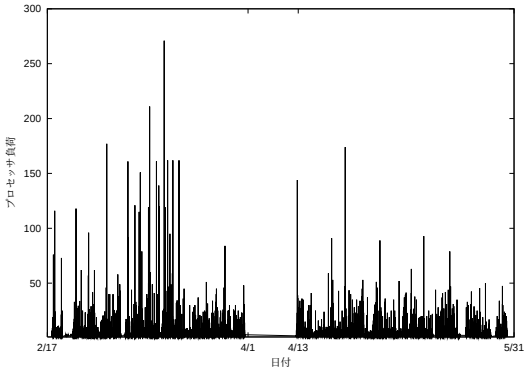

図 3 評価実験に使用した資源情報（横軸は “月/日”，縦軸はプロセッサ負荷）

Fig. 3 The resource information which is used for the performance evaluation (the horizontal and vertical axes mean “‘Month/Date” and the processor load respectively).

$$\text{予測誤差 (\%)} = \left| 1 - \sum_{i=0}^{(\text{時間間隔}/q)-1} \frac{1}{u_{w+i}} \Big/ \sum_{i=0}^{(\text{時間間隔}/q)-1} \frac{1}{v_{x+w+i}} \right| \times 100$$

あらかじめ，2000 年 2 月 17 日から 2000 年 5 月 31 日まで，RIS が資源情報を収集してある（**図 3**）．図 3 で値が示されていない日（同年 4 月 1 日から 4 月 13 日等）は，COMPACS が運用されていなかった，あるいは，RIS を再構成していた等の理由により，資源情報を収集できなかった．したがって，図 3 の値が示されている期間が資源情報収集期間である．

まず，高精度予測モジュールで使用する w および q の各定数を決定するため，2000 年 5 月 22 日午前 0 時から 28 日午前 0 時までの予測期間を選び，その期間の q 秒おきの時刻において，高精度予測モジュールを使用して 1680 秒後から q 秒間の予測を行った．

まず，$w = 11$ と固定して q を変化させたときの予測誤差を**図 4** に示す．この図から，q が大きいと予測誤差が少なくなることが分かる．実際，この傾向は w を変化させても一般的に成立している．q を大きくすればより広い範囲を平均することになるので，変動が小さくなり予測しやすくなるためと考えられる．しかし，RIS の予測値をタスクスケジューリング等の目的に使用することを想定すると，q はできるだけ小さい方が予測対象時間を細かく指定できるようになり，使いやすくなる．そこで，予測誤差がある程度小さくなり，計算時間が数分間程度の粗粒度タスクを並列計算機クラスタ上でスケジューリングする目的で予測を使用できる $q = 300$ を選択することにした．

次に，$q = 300$ と固定して w を変化させたときの予測誤差を**図 5** に示す．$w = 11$ としたとき，高精度予測モジュールの予測誤差が最小となった．また，今回，実験に使用した RIS のプロトタイプ版では，運用上設定されている CPU 時間の使用制限を回避するため，1 時間ごとに再立ち上げしているので，再立ち上げ中に要する約 3 分間は資源情報を収集できないという制限がある．図 5 に規則正しい周期が現れているのは，この 1 時間がちょうど wq で分割できるか否かが周期的に変化するため，それにより予測誤差が変化するためではないかと考えられる．

高速予測モジュールの予測結果例を**図 6** に示す．図 6 は，2000 年 5 月 27 日 9 時 33 分から，p 秒おきに高速予測モジュールを使用して $2p$ 秒後の予測を行い，予測値と実測値を連続的にグラフに表したものである（$p = 300$）．この図から，高速予測モジュールは，変動が少ない場合，予測誤差が少なく予測を行うことができるが，予測対象が短い時間で大きく変動する場合，

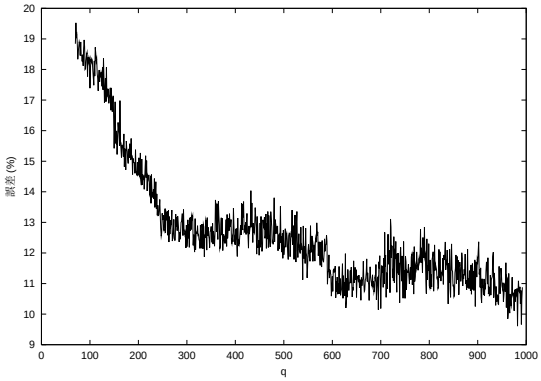

図 4 q を変化させたときの高精度予測モジュールの予測誤差（$w = 11$）

Fig. 4 The error of the accurate prediction module when q is varied ($w = 11$).

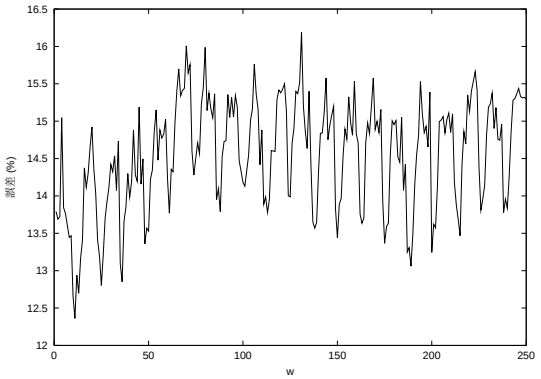

図 5 w を変化させたときの高精度予測モジュールの予測誤差（$q = 300$）

Fig. 5 The error of the accurate prediction module when w is varied ($q = 300$).

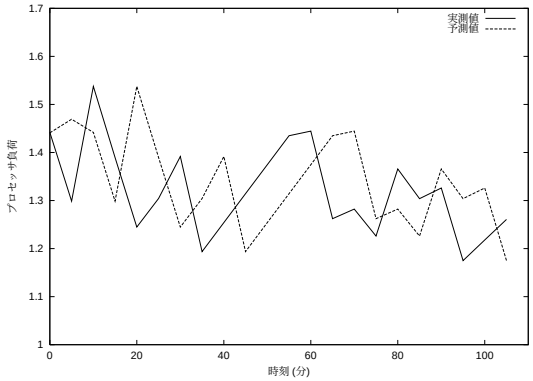

図 6 高速予測モジュールの予測結果例
（p 秒間の平均を点で表示）

Fig. 6 An example of the fast prediction module result (every average per p seconds is shown by a point).

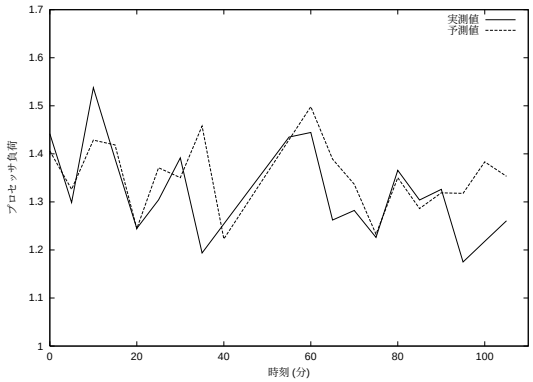

図 7 高精度予測モジュールの予測結果例
（q 秒間の平均を点で表示）

Fig. 7 An example of the accurate prediction module result (every average per q seconds is shown by a point).

予測誤差が大きくなることが分かる．

高精度予測モジュール（$w = 11$，$q = 300$）の予測結果例を**図 7** に示す．この図は，2000 年 5 月 27 日 9 時 33 分（横軸の時刻 0）における高精度予測モジュールがその時刻から先にプロセッサ負荷の変動を予測した 1 回の予測結果を表したものである．図 7 の時刻 0 から最近 wq (= 3300) 秒間のプロセッサ負荷の実測値に最も類似した過去の記録は同年 4 月 27 日 1 時 45 分からの 3300 秒間であった．したがって，高精度予測モジュールは，それから先の同日 2 時 40 分からの記録を予測結果としている（図 7 の点線）．

図 8 は，2000 年 5 月 22 日午前 0 時から 28 日午前 0 時までの 6 日間にわたる予測期間中の 300 秒ごとの時点において，その時点から先の資源情報を予測したとき，各予測モジュールの予測対象時間別の予測誤差の平均値を示す．たとえば，最上段の左から 3 番目は，予測期間中の 300 秒ごとの時点において，600 秒先の 300 秒間のプロセッサ負荷を予測したとき，高精度予測モジュールの予測誤差の平均値が 11.96%，高速予測モジュールの予測誤差の平均値が 42.17%であったことを示している．下段の数値は高速予測モジュールの予測誤差を表している．これらの数値から，高速予測モジュールが直後を予測する場合，21%程度の予測誤差で予測を行うことができるが，時間的に将来になればなるほど，予測誤差が大きくなることが分かる．また，上段の数値は高精度予測モジュールの予測誤差を表している．これらの数値から，高精度予測モジュールの過去の類似した負荷パターンを探索する方法により，直後以外であっても資源情報の予測が可能であることが分かる．また，その予測誤差は約 12%程度であり，高速予測モジュールよりも良い．予測対象が将来になるほど予測誤差は増加するが，実験の範囲では予測誤差の増加はわずかであった（たとえば，高精度予測モジュールの 2 時間後の予測誤差は約 15%に増加する）．

1 回の予測に要する計算時間は，高速予測モジュールで 3.54×10^{-7} 秒，数 10 日程度を探索する高精度予測モジュールで 6.18×10^{-3} 秒であった〔Sun Enterprise 450（Ultra SPARC-II 296 MHz）を使用〕．最近の資源情報だけを使用して，迅速に近い将来の資源情報を予測する高速予測モジュールの目標は達成されている．

なお，利用者プログラムが実際に資源情報の要求，および，結果を取得するためには，資源情報 API ライブラリによる RIS 本体と利用者プログラム間の通信が必要である．それに要する通信時間は，RIS 本体と利用者プログラムが同一の SMP 型並列計算機上にある場合，RIS 本体と利用者プログラム間の通信は共有メモリを利用して行われ，それ以外の場合，TCP/IP を利用して行われる．RIS のプロトタイプ版での要求と結果取得を合計した通信時間は，前者の場合で 4.3×10^{-5} 秒，後者の場合で 1.9×10^{-1} 秒である〔RIS を AT 互換機（FreeBSD-2.2.8R，Pentium 166 MHz），利用者プログラムを AT 互換機（FreeBSD-4.0R，Pentium III Xeon 750 MHz）で実行し，100 baseTX ネットワークを使用〕．

メタスケジューリングのようなタスクスケジューリングでは，スケジューラから RIS に対し頻繁に資源情報の予測要求が行われるため，この通信時間はできるだけ短時間で済む必要がある．計算時間が数秒程度のタスクを並列計算機クラスタ上で，メタスケジューリ

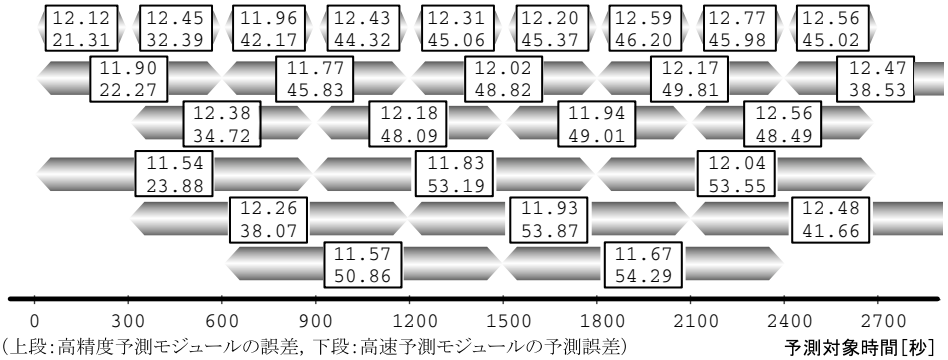

図 8 各予測モジュールにおける予測誤差（%）
Fig. 8 The error of the resource information prediction of each module.

ングにより並列分散実行する場合，効率の良いタスクスケジューリングを行うために，タスクスケジューラと RIS 間の通信は前者の共有メモリを利用して行うことが必須であった[5]．

5. 関連研究との比較

Dinda らは，線形時系列モデル（linear time series models）をプロセッサ負荷の予測に適用している[9]．この線形時系列モデルは，プロセッサ負荷は特定の線形フィルタに完全にランダムで予測不可能なホワイトノイズを入力したときの出力であるとする予測モデルである．線形フィルタの各係数を過去のプロセッサ負荷の測定値から計算し，計算された係数を持つフィルタを使用することにより，線形時系列モデルに基づく負荷予測を行うことができる．

Dinda らは，プロセッサ負荷は過去の振舞いから実用的な精度でいつも予測できることを指摘している．RIS の高精度予測モジュールの過去の類似したパターンを検索する方法は，このことを利用して予測を行っているが，線形時系列モデルとは異なる独自の方法である．なお，高速予測モジュールは，Dinda らが予測のために採用した線形時系列モデルの 1 つである BM モデルと同一である．

NWS（Network Weather Service）[10)~12)]は，RIS と同様のサービスを提供する代表的なソフトウェア・システムである．NWS は，Dinda らの線形時系列モデルに基づく複数の予測モジュール[11),12)]を組み込み，最近の予測で最も誤差が少なかった予測モジュールを使用して予測を行う[10)]．

NWS は，以下の要件を満たすべく実装されている．

高精度： 資源情報の予測精度が良いこと

軽量性： 実質的な計算に影響を与えないように極力負荷が軽いこと

普遍性・安定性： 一般的なシステムサービスとして，いつでも，どの計算機でも，サービスを受けられること

RIS においてもこれらの要件を満たすことは重要である．精度の高い予測を行うために，過去の資源情報をできるだけ詳細に記録する必要があり，プロセッサとネットワークに負荷を与える実測を頻繁に行う必要がある．しかし，頻繁に実測を行うと軽量性が犠牲になり，実質的な計算に影響を与える．また，RIS が安定した長時間動作を行うことは，いつでもサービスを受けられるため，普遍性・安定性に直接結び付くうえ，長期間の連続した過去の資源情報が得られるため，資源情報の予測精度を高くすることが可能になる．したがって，相反する軽量性と高精度の要件をバランス良く満たし，普遍性と安定性を高める実装を行う必要がある．RIS と NWS のこの問題に対する実装上のアプローチの相違点を以下に示す．

取り扱う資源情報に関する相違点： RIS は，並列分散プログラムに資源情報を提供することを想定している．並列分散プログラムは，TCP/IP 等プリミティブな通信ライブラリのみを使用して構築されることはまれであり，Stampi 等並列分散処理に適した通信ライブラリを使用して構築される．このため，資源情報を測定する際，通信ライブラリ等のオーバヘッドを考慮する必要がある．考慮すべきオーバヘッドには，以下が考えられる．

1. 計算機の種類や組合せによっては，通信経路の途中にルータを起動しメッセージ中継を行わないと通信することができないことがあり，このルータを経由するためのオーバヘッド
2. 計算機の組合せによっては，バイトオーダの変換や（IEEE と CRAY など）数値フォーマットの変換を行う必要があり，この変換のためのオーバ

ヘッド

実際に，Stampi 等の通信ライブラリは，1. で必要とされるルータの起動やメッセージの中継，2. のデータフォーマットの変換を自動的に行っている．これらのオーバヘッドのため，TCP/IP を直接測定して得た性能と Stampi 等の通信ライブラリの性能は 30 パーセント以上異なる場合がある[13]．

RIS は，これらのプログラムが利用する異機種並列計算機間通信ライブラリを利用した場合のプロセッサやネットワークに関する資源情報を直接測定しているため，これらのオーバヘッドが考慮されている．NWS は，汎用に使用されることを想定しており，汎用の通信プロトコルである TCP/IP に関する性能を直接測定しているため，これらのオーバヘッドは考慮されていない．

逆に，現在の RIS はプローバの起動やネットワーク性能の測定のため，測定対象の計算機で共通の異機種並列計算機間通信ライブラリが使用できることを前提としている．この条件は NWS では不要である．

資源情報の測定に関する相違点： 普遍性を高めるため，すべての計算機のあらゆる組合せを結ぶネットワークに関する資源情報を測定する必要があるが，同時に多くの測定のための通信を行うとネットワークにかかる負荷が大きくなる．RIS では，動的に生成して同時に実行するプローバの数を制限することにより，ネットワークにかかる負荷を制御している．NWS はプローバ（NWS Sensor）のグループを階層化し，グループ間のネットワーク性能の測定を制限することにより，ネットワークにかかる負荷を制限している．

6. ま と め

本論文では，分散コンピューティング環境上の資源情報を収集し，それをもとに将来の資源情報の予測を行う RIS における，資源情報の計測と予測方法，システム構成について述べた．資源情報の予測を行う高速，および，高精度予測モジュールを提案し，それらの予測時間と予測精度に関する評価を行った．

その結果，高速予測モジュールでは，平均 3.54×10^{-7} 秒の予測時間で直後を予測する場合，約 21%の予測誤差で負荷予測可能であり，高精度予測モジュールでは，平均 6.18×10^{-3} 秒の予測時間で 12%の予測誤差で負荷予測可能であった．すなわち高精度予測モジュールは高速予測モジュールに比べ約 17000 倍の時間を要するが，約 9%予測精度を改善できることが確かめられた．また，高速予測モジュールは，最近の資源情報だけを使用する方法により，迅速に近い将来の資源情報を予測することができ，高精度予測モジュールは，過去の類似した負荷パターンを探索する方法により，直後以外の資源情報であっても予測が可能であった．

RIS はそのプロトタイプ版がすでに実装されており，COMPACS における日立 SR2201，NEC SX-4 等，複数の並列計算機の資源情報を収集・予測できる．

今後の予定として，我々は，次の 3 点を考えている．第 1 に，w は高精度予測モジュールの予測誤差に影響を与える重要な定数であるが，予測誤差を最小にする値は予測対象の計算機や利用の仕方などに依存すると考えられる．今回，$w = 11$ のときに高精度予測モジュールの予測誤差が最小となったが，w と予測誤差の関係に関する調査を行い，その理由も見つけたい．第 2 に，図 3 によると，通常約 2 から 10 程度の SR2201 のプロセッサ負荷が頻繁に数 10 を超えていることが分かる．このような負荷のピークをそれ以前の記録から予測することができる予測モジュールを設計し，その予測結果を利用して事前にそのプロセッサを避けてタスクをスケジューリングすることは，実行効率の向上のために重要である．第 3 に，Smith ら[14]は，利用者名やプロセス名，プロセスに与える引数は，そのプロセスの実行時間の予測に有意義であることを示す実験結果を得ている．今後，RIS でもこれらの情報を使用できるように高精度予測モジュールの改良を行い，予測精度をさらに向上させていく予定である．

さらにその後，プロセッサ負荷ばかりでなく通信性能に関する予測も含めて RIS を，メタスケジューリングに応用して評価していく予定である．

謝辞 査読者の方々，ならびにプログラミング研究会において貴重なご意見をくださった方々に深く感謝いたします．

参 考 文 献

1) Foster, I. and Kesselman, C.: The Globus Project: A Status Report, *Proc. IPPS/SPDP '98 Heterogeneous Computing Workshop*, pp.4–18 (1998).
2) Karonis, N., Martin, S., Smith, W., et al.: A Resource Management Architecture for Metacomputing Systems, *Proc. IPPS/SPDP '98 Workshop on Job Scheduling Strategies for Parallel Processing* (1998).
3) Grimshaw, A. and Wulf, W.: The Legion Vison of a Worldwide Virtual Computer, *Comm. ACM*, Vol.40, No.1, pp.39–45 (1997).
4) International Grid Research Demonstrations : *SC'98 Performance Networking and*

Computing Conference, `http://www.startap.net/igrid` (1998).
5) Koide, H., Hirayama, T., Murasugi, A., Hayashi, T. and Kasahara, H.: Meta-scheduling for a Cluster of Supercomputers, *Proc. International Conference on Supercomputing Workshop, Scheduling Algorithms for Parallel/Distributed Computing – From Theory to Practice*, pp.63–69 (1999).
6) 小出 洋ほか：メタスケジューリングのための資源情報サーバの構築，計算工学会講演会論文集，Vol.5, No.1, pp.357–360 (2000).
7) 武宮 博，今村俊幸，小出 洋：並列分散科学技術計算を支援するソフトウェア・システム（STA）の構築，情報処理，Vol.40, No.11, pp.1104–1109 (1999).
8) `http://ssp.koma.jaeri.go.jp/stampi.html`
9) Dinda, P. and O'Hallaron, D.: An Evaluation of Linear Models for Host Load Prediction, *Proc. 8th IEEE International Symposium on High Performance Distributed Computing*, `http://www.computer.org/proceedings/hpdc/0287/0287toc.htm` (1999).
10) Wolski, R., Spring, N. and Hayes, J.: The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing, *Journal of Future Generation Computing Systems*, Vol.15, No.5-6, pp.757–768 (1999).
11) Wolski, R., Spring, N. and Hayes, J.: Predicting the CPU Availability of Time-shared Unix Systems, *Proc. 8th IEEE High Performance Distributed Computing Conference* (1999).
12) Wolski, R.: Forecasting Network Performance to Support Dynamic Scheduling Using the Network Weather Service, *Proc. 6th High-Performance Distributed Computing Conference* (1997).
13) Takemiya, H., et al.: Software Environment for Local Area Metacomputing, *Proc. 4th International Conference on Supercomputing in Nuclear Applications* (2000).
14) Smith, W., Taylor, V. and Foster, I.: Using Run-Time Predictions to Estimate Queue Wait Times and Improve Scheduler Performance, *Proc. Workshop on Job Scheduling Strategies for Parallel Processing* (1999).

(平成 12 年 7 月 14 日受付)
(平成 13 年 1 月 22 日採録)

小出 洋（正会員）

平成 3 年電気通信大学電気通信学部計算機科学科卒業．平成 9 年同大学院電気通信学研究科情報工学専攻博士後期課程修了．博士（工学）．平成 8 年より日本原子力研究所計算科学技術推進センター勤務．並列分散処理，メタスケジューリング手法の研究に従事．日本ソフトウェア科学会会員．

山岸 信寛（正会員）

日立東北ソフトウェア（株）ソフトウェア開発本部 PSB 事業グループ技師．平成元年東北大学工学部情報工学科卒業．平成 3 年同大学院工学研究科情報工学専攻博士前期課程修了．同年日立東北ソフトウェア（株）入社．平成 12 年より日本原子力研究所計算科学技術推進センター出向．

武宮 博（正会員）

日立東北ソフトウェア（株）ソフトウェア開発本部 PSB 事業グループ主任研究員．昭和 61 年東北大学大学院理学研究科天文学博士前期課程修了．平成元年同博士後期課程中退．同年日立東北ソフトウェア（株）入社．平成 12 年より日本原子力研究所計算科学技術推進センター受託研究員．

笠原 博徳（正会員）

昭和 55 年早稲田大学理工学部電気工学科卒業．昭和 60 年同大学院博士課程修了．工学博士．昭和 58 年～60 年早稲田大学理工学部助手．昭和 60 年カリフォルニア大バークレー短期客員研究員，日本学術振興会第 1 回特別研究員．昭和 61 年早稲田大学電気工学科専任講師．昭和 63 年同助教授．平成 9 年同大電気電子情報工学科教授，現在に至る．平成元年～2 年イリノイ大学 Center for Supercomputing R&D 客員研究員．昭和 62 年 IFAC World Congress 第 1 回 Young Author Prize 受賞．平成 9 年情処坂井記念特別賞受賞．主な著書「並列処理技術」（コロナ社）．電子情報通信学会，電気学会，シミュレーション学会，ロボット学会，IEEE，ACM 等会員．