Webサービスベースのワークフロー管理における信頼性と負荷を考慮したスケジューリングパラメタ調整法

Parameter Adjustment Methods for Activity Instance Scheduling in Web Services Based Workflow Management Considering the Reliability and Workload

加藤 英之♡ 小林 隆志♢ 横田 治夫♢

Hideyuki KATOH Takashi KOBAYASHI Haruo YOKOTA

我々はこれまでに，近年注目されているWebサービスを用いたワークフローに着目し，その特徴を考慮して効率よくアクティビティの割り当てを行う手法であるOXTHASを提案し，その有効性を示してきた．Webサービスを前提とすると，複数のワークフローエンジンがWebサービスを共用する場合が考えられるため負荷状況を正確に把握できない．OXTHASでは，Webサービスの過去の処理履歴を利用して負荷状況を推定し，処理負荷サイズを考慮してアクティビティを割り当てることにより効率の良い処理を実現している．本稿では，より実際の環境に対応させるべく，ネットワークやハードウェア等の障害を考慮に入れたスケジューリング方法を提案し，シミュレーションによりその有効性を示す．提案手法では，タイムアウトを設定し，スケジューリングのパラメタとして個々のエグゼキュータ毎のタイムアウト値と処理能力の推定値を調整する方法を取ることで，障害の影響を抑えることを可能にしている．

In a workflow management system, appropriate allocation of activity instances (AIs) greatly contributes to improve its efficiency. We have proposed OXTHAS, a load-balancing method of scheduling the AIs in workflow management systems using Web services. The OXTHAS makes the AIs be allocated to appropriate executors based on the estimation of their processing capacity using execution histories in workflow engines. In this paper, we propose a failure-aware re-scheduling methods for the OXTHAS using process timeouts under the network and system failures, and evaluate the effectiveness of the proposed methods thorough simulations. We demonstrate that methods of adjusting the estimated processing capacity and timeout duration as scheduling parameters in each executor are effective to allocate AIs appropriately with holding down the influence of failures.

1. はじめに

近年，ネットワークを介してHTTPとSOAPを用いて，ソフトウェアシステム間で通信を行うオープンな技術であるWebサービスが注目されている[1]．Webサービスは，実行環境に依存しないアプリケーションの連携が可能なため，CORBAやDCOMなどの分散オブジェクト技術よりも利用が容易である．現在，BPEL4WS[1]などの仕様の登場により，Webサービスを利用したワークフローの実現に関する研究が行われている．しかしながら，それらの仕様では負荷分散や障害対策に関して十分に言及されていない．

この問題を解決するために，我々はこれまでに，Webサービスを用いたワークフローにおける障害対策手法として集中管理に適した手法[2]，分散管理，エージェントによる実現に適した手法[3]を提案してきた．また，我々はWebサービス利用を前提としたワークフロー管理における負荷分散手法であるOXTHAS(Observed Execution Time History and Activity Size based scheduling)[4, 5]を提案してきた．しかし，[4, 5]では，障害時におけるスケジューリングについては言及していなかった．本稿では，我々が提案したスケジューリング戦略OXTHASに関して，システムの故障やネットワークの障害等を考慮し，それらに対応するために，タイムアウトを用いたOXTHAS再スケジューリング戦略を提案する．OXTHASでは，エグゼキュータ毎のワークフロー・エンジンから観測した各アクティビティ・インスタンスの過去の実行時間の履歴と，そのアクティビティ・インスタンスの処理負荷サイズから推定処理能力を算出し，アクティビティ・インスタンスを適切なエグゼキュータに割り当てることで負荷分散を実現する．このとき，ネットワークやエグゼキュータの障害に対応するためにOXTHASにタイムアウトの概念を導入する．タイムアウトが発生した場合には，そのエグゼキュータで発生したタイムアウトの回数も考慮に入れて推定処理能力を再計算，もしくはタイムアウトの値を調整し，再スケジューリングを行う．

2. 想定するワークフロー管理モデル

2.1 想定するワークフロー管理アーキテクチャ

我々の想定するワークフロー管理アーキテクチャは主にワークフロー・エンジンとエグゼキュータから成り立つ．ワークフロー・エンジンはプロセス・インスタンスやアクティビティ・インスタンスを制御，管理する．プロセスは一連の工程から成り立つ作業を表し，アクティビティは1つ1つの工程を表す[6, 7]．ワークフロー・エンジンとエグゼキュータの機能は以下のようになる．

ワークフロー・エンジン ワークフロー･エンジンはクライアントがプロセス・インスタンスを入れるためのキューとエグゼキュータが処理を終えたアクティビティ・インスタンスを入れるためのキューを持っている．ワークフロー・エンジンはプロセス・インスタンスの情報を持つ．

ワークフロー・エンジンはアクティビティ・インスタンスをキューから取り出し，処理を行うエグゼキュータのキューに入れる．さらに，エグゼキュータから帰ってきたアクティビティ・インスタンスに対しACK/NACKをエグゼキュータに知らせる．もし，同一のアクティビティ・インスタンスに対して複数のリクエストが発生した場合，最後に発生したリクエストを採用することとする．

エグゼキュータ エグゼキュータはワークフロー・エンジンがアクティビティ・インスタンス入れるためのキューを持つ．エグゼキュータはキューからアクティビティ・インスタンスを1つずつ取り出して，処理を行う．処理が終わったら，ワークフロー・エンジンのキューにそのアクティビティ・インスタンスを入れる．

上記の手順の中で，各アクティビティ・インスタンスをどのエグゼキュータに割り当てるかの戦略によって，システム全体の負荷分散を行うことが可能となる．本稿では，2つ以上のワークフロー・エンジンが存在し，いくつかのエグゼキュータを共用し，各ワークフロー・エンジンは自身が管理するアクティビティ・インスタンスの情報のみ知ることができると仮定する．

2.2 Webサービスベースのワークフロー管理

前述のワークフロー管理アーキテクチャを元としたWebサービスベースのワークフロー管理を対象とするとき，以下のような特徴が考えられる．

♡ 学生会員 東京工業大学 大学院 情報理工学研究科 計算工学専攻
katoh@de.cs.titech.ac.jp

♢ 正会員 東京工業大学 学術国際情報センター
{tkobaya,yokota}@cs.titech.ac.jp

[1] http://www-128.ibm.com/developerworks/library/specification/ws-bpel/

(1) 1 つのエグゼキュータの行うアクティビティ・インスタンスの処理が 1 つの Web サービスである．

(2) 個々のワークフロー・エンジンエグゼキュータの内部情報を知ることができない．

1 つ目に関しては，エグゼキュータが 1 種類以上のアクティビティ・インスタンスの処理を Web サービスとして提供することが考えられるため，個々のアクティビティ・インスタンス毎にエグゼキュータの処理能力を考慮して割り当てを行う必要がある．また，個々のアクティビティ・インスタンスの入力データのファイルサイズが一定でないことが処理時間に影響を与える場合を考慮する必要がある．本稿では，処理時間はファイルサイズに比例すると考える．さらに，Web サービスを提供するプロバイダの環境やネットワークの良否による遅延やタイムアウトの発生などを考慮する必要がある．

2 つ目に関しては，複数のワークフロー・エンジンが共存することを想定する場合，いくつかのワークフロー・エンジンがエグゼキュータを共用する可能性がある．そのため，ワークフロー・エンジンはエグゼキュータのキュー長を知ることができず，ワークフロー・エンジンが知ることのできる情報は，ワークフロー・エンジンがエグゼキュータにリクエストを出してからエグゼキュータから処理が終わってそのワークフロー・エンジンに戻ってくるまでの時間になる．そして，この時間には処理時間の他にキューでの待ち時間やネットワークの遅延時間なども含まれる．

3. OXTHAS 再スケジューリング戦略

3.1 OXTHAS スケジューリング戦略

ここでは，我々が以前に提案した OXTHAS スケジューリング戦略について簡潔に説明する．詳細は [4, 5] を参照されたい．

OXTHAS-N は各アクティビティ・インスタンスを処理能力が上位 N 台のエグゼキュータに割り当てる．しかし，前述の通り，ワークフロー・エンジンはエグゼキュータの内部情報を知ることができないため，リクエストを出してから処理が終わって戻ってくるまでの時間からエグゼキュータの処理能力を推定しなくてはならない．そこで，アクティビティ・インスタンス a_j に対するエグゼキュータ e_k の推定処理能力 $c_{k,j}$ を以下ように定義する．

$$c_{k,j} = \frac{\sum_{i=1}^{l}(\frac{s_{i,j}}{t_{i,j}} \cdot m_{i,j,k})}{\sum_{i=1}^{l} m_{i,j,k}}$$

この推定処理能力 $c_{k,j}$ は単位時間当たりの仕事量の平均値を過去の処理時間から求めている．i は $P = \{p_1 \dots p_l\}$ におけるプロセス・インスタンス番号であり，$t_{i,j}$ は p_i 中の j 番目のアクティビティ・インスタンスの観測実行時間，$m_{i,j,k}$ は p_i 中の j 番目のアクティビティ・インスタンスが e_k において実行可能かどうかを表す変数である．もし，そのエグゼキュータへのマッピングが可能の場合 $m_{i,j,k}$ は 1 であり，そうでない場合は 0 になる．$c_{k,j}$ の値が高い場合，そのエグゼキュータの処理能力が高いか，そのエグゼキュータへのリクエスト数が少ないことを意味する．アクティビティ・インスタンスの割り当てを行う度に推定処理能力を計算するため，その時点におけるエグゼキュータの処理能力と負荷を考慮することが可能となる．

アクティビティ・インスタンスを $c_{k,j}$ の値が上位 N 台のエグゼキュータに分散するために，処理負荷サイズに閾値を設け，N 個の領域に分割する．そして，処理負荷サイズがどの領域にあるかにより，上位 N 台のエグゼキュータのうちのどのエグゼキュータに割り当てるかを決定する．N 個の領域に分割する方法は，整数の比による分割や推定処理能力の値の比による分割，等分割などが考えられる．ここでは，予備実験の結果より，整数の比による分割方法 (整数比分割) を用いることとする [8].

整数比分割では，アクティビティ・インスタンスの処理負荷サイズを分割するための閾値は，w_i = (処理負荷サイズのボーダーライン)，n = (分割数)，S_{max} = (過去の履歴の中でのアクティビティ・インスタンスの最大サイズ) とし，次のように定める．

$$w_i = \left(\sum_{j=1}^{i} j \Big/ \sum_{k=1}^{n} k\right) \cdot S_{max}$$

3.2 障害の考慮

エグゼキュータからの応答がない場合，その理由としてネットワークの混雑とネットワークもしくはエグゼキュータの故障が考えられる．ネットワークの混雑が理由の場合，処理が遅くなるだけで，処理自体はそのうち行われ，その混雑の程度に関してはすでに推定処理能力で対応されている．しかし，ネットワークやエグゼキュータの故障が理由の場合，処理が停止してしまうことが予想される．そこで，タイムアウトの概念を導入しネットワークやエグゼキュータの故障に対応する．このとき，タイムアウト発生時の再割り当て方法を OXTHAS 再スケジューリング戦略と呼ぶ．

OXTHAS 再スケジューリング戦略では，タイムアウトが発生してアクティビティ・インスタンスを再び割り当てる時に，再割り当てを行うアクティビティ・インスタンスをすでに割り当てたエグゼキュータには基本的には割り当てないようにする．そのため，もしエグゼキュータが 10 台の場合，同じアクティビティ・インスタンスで再割り当てが発生する度に割り当て対象となるエグゼキュータの数も 9 台，8 台 … と減っていく．そして，再割り当て対象エグゼキュータ台数 N_{exe} と OXTHAS-N の N の関係が，

$$N_{exe} < N$$

となった場合，OXTHAS-N_{exe} で再割り当てを行う．

3.3 調整法

タイムアウトを導入する際，故障の影響をなるべく抑える必要がある．故障の影響を抑える方法としては，故障の可能性があるエグゼキュータにアクティビティ・インスタンスをなるべく割り当てないようにする方法と，たとえ故障の可能性のあるエグゼキュータに割り当てられてもすぐにタイムアウトを発生させることで故障の影響を抑える方法が考えられる．前者を推定処理能力調整法，後者をタイムアウト値調整法とし，以下のように定義する．

推定処理能力調整法

先ほど定義した推定処理能力にタイムアウトの発生時を考慮に入れて，$c'_{k,j}$ を以下のように再定義する (R_k はエグゼキュータ e_k でタイムアウトが発生した回数．α は定数)．

$$c'_{k,j} = \frac{c_{k,j}}{1 + \alpha \cdot R_k}$$

タイムアウトの発生時，その推定処理能力が減少し，そのエグゼキュータにアクティビティ・インスタンスを割り当てないようになる．

タイムアウト値調整法

タイムアウトが発生した場合はそれ以降に割り当てるアクティビティ・インスタンスに関して，タイムアウトの値 T_k を次のように定める．(β, T_{init} は定数)

$$T_k = T_{init} - \beta \cdot R_k$$

タイムアウトが発生した場合はそれ以降に割り当てるアクティビティ・インスタンスに対して設定されるタイムアウトの値が低くなるため，故障したエグゼキュータに割り当てられてもすぐにタイムアウトが発生し，再割り当てが起こる．

もし，タイムアウトが発生した後にそのエグゼキュータから処理が返ってきた場合，故障検知ミスか故障からの復旧を意味するため，$c'_{k,j}$ や T_k の値を最初に定義した値に戻す．そのため，一時的な故障に対応することが可能であり，故障傾向が高い場合は，タイムアウトのペナルティを大きくするという効果がある．

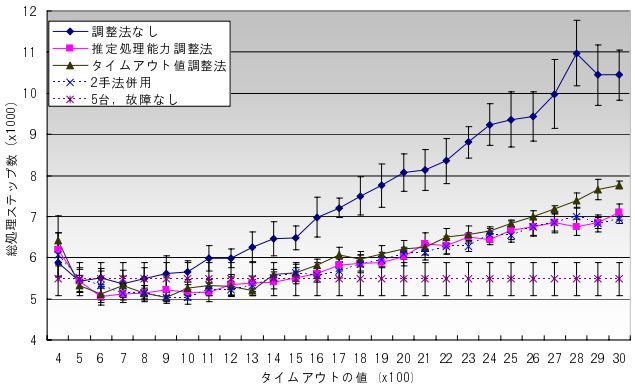

図 1: 故障時における再割り当て手法の違いによる
総処理ステップ数の比較
Fig.1 Total processing step of each adjustment method for time-out step setting

4. 評価

4.1 シミュレーション内容

前述した OXTHAS 再スケジューリング戦略における調整法の有効性を示すために [4] のシミュレータを用いて，シミュレーションを行った．エグゼキュータは 6 台，ワークフロー・エンジンは 2 台とし，各ワークフロー・エンジンでは同じ数のプロセス・インスタンス (プロセス・インスタンス数は 200 個) を処理し，プロセス・インスタンスは 10 ステップに 1 つ到着し，それぞれのプロセス・インスタンスをどこで処理するかなどはそのワークフロー・エンジンしか管理していない．ビジネスプロセスは分岐などのない単純なものとした．エグゼキュータの処理性能や，プロセス・インスタンス，アクティビティ・インスタンスのサイズなどはランダムで決定される．また，すべてのプロセス・インスタンスが持つアクティビティ・インスタンスの数は 5 として，すべて同じにしてある．そして，各エグゼキュータではすべてのアクティビティ・インスタンスに関して処理可能であるとする．また今回は，エグゼキュータの部分的な故障は考えず，故障の時は，そのすべてのアクティビティ・インスタンスの処理が不可能であるとする．

最初に，6 台のエグゼキュータのうち，1 台のエグゼキュータを途中で故意に故障させたときのタイムアウトの効果を，OXTHAS-N(N=3) において通常の OXTHAS-N と提案した調整法を用いた OXTHAS-N を比較する．また，故障させたエグゼキュータを除いて故障をしない 5 台で処理を行ったときとの比較も行う．さらに，推定処理能力調整法，タイムアウト値調整法の効果の検証を行う．

推定処理能力は過去の履歴から求められるものであるため，初めの 1000 ステップまでは，ランダム割り当てを用いる．それぞれ 10 回測定を行い，その平均を取る．実験結果の各グラフの誤差棒は 95 パーセント信頼区間を表す．

4.2 結果と考察

図 1 は，故障が発生する場合にタイムアウトを変化させていったときの調整法を用いない OXTHAS と 2 つの調整法を比較したものである．エグゼキュータ 6 台中の 1 台 (エグゼキュータ 2) を，2000 ステップ目に故意に故障させて測定を行った．推定処理能力調整法は $\alpha = 1$ とし，タイムアウト値調整法は $\beta = T_{init}/100$ とした．「5 台，故障なし」の線は故障させた 1 台を最初から抜かして故障をしない 5 台のエグゼキュータで測定したときの結果である．調整法を用いない OXTHAS よりも調整法を用いた場合の方が，タイムアウトの値が長くなっていった時の総処理ステップ数の低下を抑えていることがわかる．さらに，タイムアウトの値が 500 ステップから 1400 ステップの間では，「5 台，故障なし」よりも提案した調整法の方が効率よく処理が行われている．このことは，低信頼なエグゼキュータを含んだ 6 台のエグゼキュータの処理が高信頼な 5 台のエグゼキュータの処理に匹敵，もしくは勝る場合もあるということを示している．また，2 つの調整法の間では差が見られず，推定処理能力調整法とタイムアウト値調整法の両手法を併用した場合にも差は見られないため，2 つの調整法の間に相乗効果は見られない．

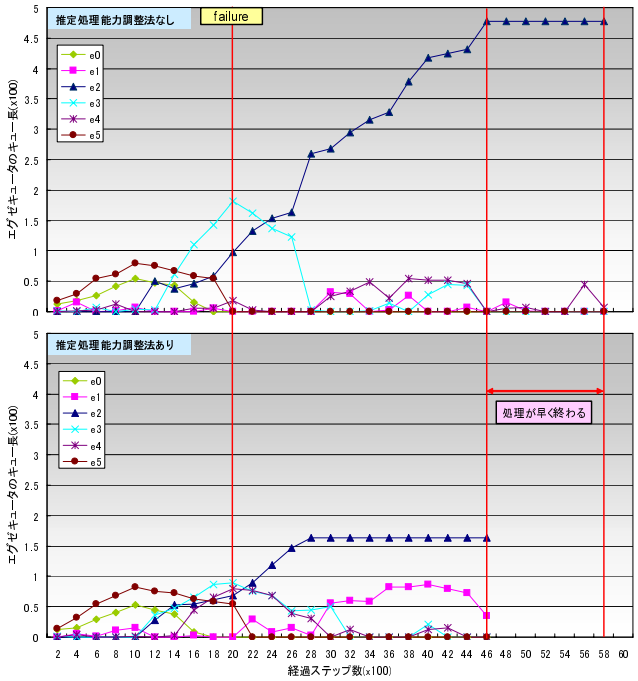

図 2: 推定処理能力調整法の有無による
OXTHAS の各エグゼキュータのキュー長の変化
Fig.2 Queue length transition of the crashed executor

図 2 は各エグゼキュータのキュー長の変化を表し，図 3 は故障したエグゼキュータ 2 のタイムアウト発生回数の変化を表す．これらの図はタイムアウトの値が 1000 ステップのときの 10 回の試行のうちのある 1 回の試行についてのグラフである．まず，推定処理能力調整法の効果について説明する．推定処理能力調整法を用いない OXTHAS の場合の図 2 では，故障に対して早期に反応できていないため，故障後も長いステップ数の間，エグゼキュータ 2 のキュー長が増加している．一方，推定処理能力調整法を用いた場合では，故障に対して早く反応し，そのエグゼキュータにアクティビティ・インスタンスが割り当たらず，別のエグゼキュータに割り当てられ (この図の場合はエグゼキュータ 1，3，4)，全体の処理が早く終わる．また，図 3 におけるタイムアウト値調整法を用いない場合では，タイムアウト発生回数が全処理が終わるステップ数まで増加している．一方，タイムアウト値調整法を用いる場合では，故障したエグゼキュータ 2 におけるタイムアウト発生回数の増加は 4000 ステップほどで止まっている．これは，タイムアウト値調整法により，タイムアウトの値が短くなりエグゼキュータ 2 から他のエグゼキュータに早期に再割り当てが行われ，その結果全体の処理も早く終わる．

以上の結果より，2 つの調整法はそれぞれ，推定処理能力とタイムアウトの値を適切に調節し，エグゼキュータの故障の影響を抑えていることが分かる．

5. 関連研究

Lie-jie Jin らは，プロセス・インスタンスを一定時間毎にワークフロー・エンジンに投入する際にワークフロー・エンジンでの処理前の待ち時間を減らすようにどのワークフロー・エンジンに入れるかについての負荷分散手法を提案している [9]．その中で分散ワークフロー管理システムにおけるワークフロー・エンジン間の

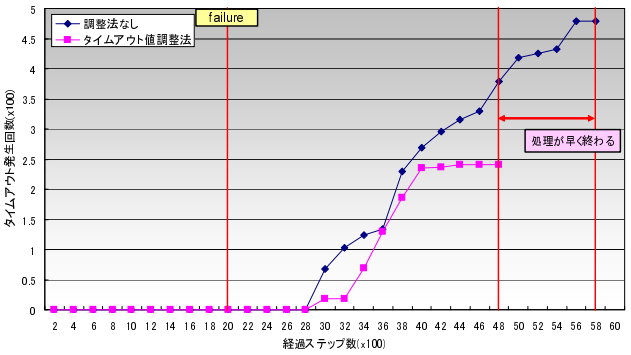

図 3: タイムアウト値調整法の有無による
OXTHAS の e2 のタイムアウト発生回数の変化
Fig.3 Time-out number transition of the crashed executor

負荷分散を行うために負荷指標を定義している．しかし，今回のように複数のワークフロー・エンジンがエグゼキュータを共用する場合には，それぞれのエグゼキュータの負荷を計算し，その負荷に応じて処理を振り分ける負荷分散が重要となってくるが，この研究ではエグゼキュータ間の負荷分散に関しては対応していない．

また，Koji Nonobe らは，資源制約付きスケジューリング問題に関するアルゴリズムを提案している [10]．これは，ワークフロー・エンジンが 1 つであるような閉じたワークフロー管理システムにおける仕事の割り当てに関しては最短で処理が完了する割り当て方を提供してくれる．しかし，複数のワークフロー・エンジンがあり，それぞれのワークフロー・エンジンが自身の仕事にのみ割り当てや監視を行っているような場合や，ネットワークの環境が動的に変化するような場合，このアルゴリズムをそのまま利用することはできない．

さらに，ワークフローにおいてプロセス・インスタンスの処理の順番を動的に変更することにより期日に遅れるプロセス・インスタンスの数を減らすスケジューリング手法を提案している [11]．これは処理時間とルートを推測しプロセス・インスタンスの処理の順番を適切に決定している．しかし，本稿で目的としている負荷分散や障害対策については述べていない．

その他に，分野は異なるが，Web サーバにおいて，小さなファイルを求めるリクエストから優先的に処理することによりキューでの待ち時間を減らし，パフォーマンスを改善する研究 [12] がある．しかし，Web サービスベースのワークフロー管理システムにこの手法をそのまま適用することはできない．

6. 終わりに

本稿では，Web サービスベースの集中管理型ワークフロー管理における障害を考慮した負荷分散手法を提案した．具体的には，ワークフロー管理モデルを説明し，ワークフローと Web サービスの特徴を考慮に入れたスケジューリング戦略 OXTHAS について説明した．さらに，OXTHAS に基づいて，エグゼキュータの障害を考慮に入れ，タイムアウトを利用した推定処理能力調整法とタイムアウト値調整法の 2 つの調整法を提案した．そして，シミュレーションにより，推定処理能力とタイムアウトの値を調整することで障害の影響を抑えた処理が可能なことを示した．

今後の課題としては，より効率的な再スケジューリング戦略にするための提案手法の改良を行うこと，様々なプロセス・インスタンスの到着率やプロセス・インスタンス毎にアクティビティ・インスタンスの数や種類が異なる時，アクティビティ・インスタンスの処理負荷サイズに偏りがある場合等に対応することや，エグゼキュータが複数のアクティビティ・インスタンスを同時に処理できる場合，エグゼキュータの故障が多発する場合やエグゼキュータが一部の種類のアクティビティ・インスタンスの処理ができないような部分故障が発生する場合などの様々な状況に対応するようにすることなどが挙げられる．また，実際にこのワークフロー管理システムを実現し，実際のシステムでも同様の有効性が言えるかを検証することが重要である．

[謝辞]

本研究の一部は，文部科学省科学研究費補助金特定領域研究 (16016232)，独立行政法人科学技術振興機構 CREST，および 21 世紀 COE プログラム「大規模知識資源の体系化と活用基盤構築」の助成により行なわれた．

[文献]

[1] 株式会社日本ユニテック DigitalXpress 編集部. SOAP UDDI WSDL Web サービス技術基礎と実践 徹底解説. 技術評論社, 2002.

[2] 加藤英之, 小林隆志, 横田治夫. ワークフローの自動実行における障害対策. In *DEWS2004*. 電子情報通信学会, Mar. 2004. I-12-02.

[3] Neila Ben Lakhal, Takashi Kobayashi, and Haruo Yokota. Throws: An architecture for highly available distributed execution of web services compositions. In *Proc. of RIDE WS-ECEG'2004*, pp. 103–110. IEEE, Mar 2004.

[4] 加藤英之, 小林隆志, 横田治夫. Web サービスを用いたワークフローにおける負荷分散手法. 信学技報 DE2005-46(2005-7), 電子情報通信学会, Jul. 2005.

[5] 加藤英之, 小林隆志, 横田治夫. Web サービスを用いたワークフロー管理における負荷分散手法のシミュレーションによる評価. *DBSJ Letters*, Vol. 4, No. 2, pp. 25–28, Sep. 2005.

[6] Workflow Management Coalition. http://www.wfmc.org/.

[7] 戸田保一, 飯島淳一, 速水治夫, 堀内正博. ワークフロー -ビジネスプロセスの変革に向けて-. 株式会社日科技連出版社, 1998.

[8] 加藤英之. Web サービスベースのワークフロー管理におけるアクティビティスケジューリングに関する研究. 修士論文, 東京工業大学, Jan. 2006.

[9] Li jie Jin, Fabio Casati, Mehmet Sayal, and Ming-Chien Shan. Load balancing in distributed workflow management system. In *Proc. of SAC2001*, Aug. 2001.

[10] Koji Nonobe and Toshihide Ibaraki. Formulation and tabu search algorithm for the resource constrained project scheduling problem. In C.C. Ribeiro and P. Hansen, editors, *Essays and Surveys in Metaheuristics*, pp. 557–588. Kluwer Academic Publishers, 2002.

[11] Gregorio Baggio, Jacques Wainer, and Clarence Ellis. Applying scheduling techniques to minimize the number of late jobs in workflow systems. In *Proc. of SAC2004*, Mar. 2004.

[12] Mor Harchol-Balter, Bianca Schroeder, Nikhil Bansal, and Mukesh Agrawal. Size-based scheduling to improve web performance. In *Proc. of TOCS2003*, May 2003.

加藤 英之　Hideyuki KATOH
平 16 東工大・工・情工卒．平 18 同大大学院・情報理工・計算工・修士課程了．日本データベース学会学生会員．

小林 隆志　Takashi KOBAYASHI
平 9 東工大・工・情報工学卒．平 11 同大大学院・情報理工・計算工学・修士課程了．平 16 同大大学院・同専攻・博士課程了．平 14 同大学術国際情報センター・助手．工博．日本データベース学会，日本ソフトウェア科学会，情報処理学会，ACM 各会員．

横田 治夫　Haruo YOKOTA
昭 55 東工大・工・電物卒．昭 57 同大大学院・情報・修士課程了．同年富士通 (株)．同年 6 月 (財) 新世代コンピュータ技術開発機構研究所．昭 61(株) 富士通研究所．平 4 北陸先端大・情報・助教授．平 10 東工大・情報理工・助教授．平 13 東工大・学術国際情報センター・教授．工博．日本データベース学会理事, 電子情報通信学会，情報処理学会，人工知能学会，IEEE，ACM 各会員．