2006－HPC－107（24）
2006／8／1

ストリーミングデータ処理のためのタスクスケジューリング

吉永一美
九州工業大学大学院
情報工学研究学科

小出 洋
九州工業大学情報工学部
知能情報工学科

概　要

本研究では，ストリーミングデータ処理を含むアプリケーションプログラムを広域並列分散環境で効率的に実行することを目的として，広域ネットワークの性能の変動に着目するタスクスケジューリング手法を提案し，その実現のための予備的な評価実験を行った．その結果，提案手法を用いた場合，全体の実行時間の短縮が図られ，動的にネットワーク性能が変動している環境でも，実行時間のばらつきが抑えられた．この結果はより複雑なタスクスケジューリング手法と組み合わせる場合に有用な特性である．

A New Taskscheduling for Processing of Streaming Data

Kazumi YOSHINAGA
Graduate School of Computer
Science and Systems Engineering
Kyushu Institute of Technology

Hiroshi KOIDE
Faculty of Computer Science of
Systems Engineering
Kyushu Institute of Technology

This paper proposes a new task scheduling method which considers changing performance of networks to execute application programs processing streaming data. Preliminary validation experiments for implementation of the method have been conducted. In the result, the proposed method reduces the execution time. The dispersion of the execution time is suppressed even if the network bandwidth is dynamically changing. This characteristic is very useful when this method combines more complicated task scheduling methods.

1　はじめに

本研究では,「マルチメディアデータの処理」や「実験装置から連続的に大量に生成されるデータの解析」などのストリーミング処理が含まれる分散プログラム (以下，ストリーミング型分散プログラム) を Grid や PC クラスタ等の分散コンピューティング環境に含まれている多くのプロセッサや記憶領域等の計算機資源を有効に利用することにより，効率良く実行することを目的としている．

ストリーミング処理はデータの入力，処理，出力を繰り返し行う．通常のタスク処理ではこれらを一度しか行わない．ストリーミング処理では到着する処理依頼よりも処理能力の方が高い必要がある．現在までのところ，ストリーミング型分散プログラムは，分散コンピューティング環境で効率的な実行を行うことができない．その大きな理由は，従来から良く研究されているワークフロー型 [1, 2] やタスクファーミング型の分散プログラム向けのタスクスケジューリング手法 [3] がそのままでは適用できないため，分散プログラミング環境の計算機資源を効率良く割り付けることができないからである．

そこで本研究では，ストリーミング型分散プログラムを分散コンピューティング環境上で実行しても，効率良く動作させることができ，その実行時間を短縮できるタスクスケジューリングを提案する．ストリーミング型分散プログラムでは，それ以外の分散プログラムよりも，タスク間でネットワークを介して大量のデータ転送が発生する．バンド幅や遅延時間といった広域ネットワークの性能はその普及とともに飛躍的に向上しているが，依然として計算機内部のデータ転送速度と比較するとかなり遅いため，タスク間のデータ転送がボ

トルネックとなり実行速度の低下に直結する．さらに広域ネットワークは，大勢の他人と共有された資源であるため，その性能は常に時間的に変動しており，その変動の予測も難しい．その上で実行される分散プログラムのタスクの実行時間は，実行の度に変動する．特に，頻繁に通信が行われるストリーミング型分散プログラムのタスクの実行時間の変動はとても大きなものになる．特にネットワーク通信とタスクが行う計算を時間的に重ね合わせやワークフロー型のタスクスケジューリング手法等で，その効率に与える影響が大きい．

本手法では，これらのネットワーク性能とその変動に着目したタスクスケジューリング手法を提案し，その実現に向けた予備的な評価実験を行った．その結果，提案手法を用いた場合，プログラム全体の実行時間の短縮が図られ，動的にネットワーク性能が変動する環境においても，実行時間のばらつきを抑えることができた．

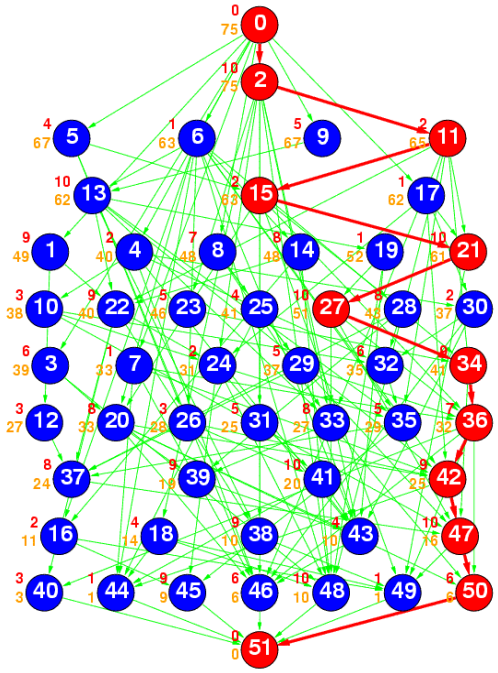

図 1: 分散プログラムのタスクグラフの例.
Fig. 1: An Example of Task Graph.

2　提案するタスクスケジューリング

2.1　対象とする問題

本タスクスケジューリング手法の目的は，ストリーミング型分散プログラムを，分散コンピューティング環境で効率良く実行することである．そのような分散コンピューティング環境は，さまざまな性能の計算機やネットワークが接続されており，負荷の変動，特に広域分散環境ではネットワークにおける負荷変動や性能格差が大きい．本タスクスケジューリング手法の対象であるストリーミング型分散プログラムは，ストリーミングデータ処理を行うタスクを含んでいる．こうしたタスクを含まない通常のアプリケーションと比較して，タスク間で連続したデータ通信が発生し続けるため，通信が多く，長時間に渡るという特徴がある．

以上の考察から，ネットワーク性能の変動を考慮して，ストリーミング処理を含むタスクを効率良く実行するようにするタスクスケジューリング手法を設計する．

分散プログラムをタスクグラフ (図 1，[4] として表すとすると，ストリーミングデータ処理を含むタスクとしては，

1. ストリーミングデータの入力がある
2. ストリーミングデータの出力がある
3. ストリーミングデータの入出力ともに持ち合わせている

の 3 種類がある．今回提案するタスクスケジューリング手法では，これらのうちストリーミングデータの入力があるタスク，つまり 1 と 3 のタスクを扱う．

2.2　ワーカの選択

スケジューリング手法が実際に実行可能なタスクを割り付ける際，タスク t_x を割り付けるものとすると，t_x に対してストリーミングでデータを送信するタスクを $t_{s_1}, \cdots, t_{s_n}$ とし，それぞれに割り付けたワーカを $w_{s_1}, \cdots, w_{s_n}$ とする．このとき，

$$\sum_{i=1}^{n} bw\_pred(w_{s_i}, w_x)$$

の値を最大にするような w_x を，タスク t_x の割り付けるべきワーカとする．ここで，$bw\_pred(w_m, w_n)$ は，ワーカ m とワーカ n の間のバンド幅の予測値である．

2.3 タスク・マイグレーション

2.1 で述べたように，ストリーミングデータ処理を含むタスクは長期間に渡りデータを流し続ける．本研究が対象とする並列分散環境えは，ネットワーク性能が変動することから，2.2 で決めたワーカがずっと良い効率を維持するとは考え難い．そのため，ワーカの効率が低下した場合の対策を講じる必要がある．

そこで，私たちは，ネットワーク性能が低下してきた場合に，タスクを他のワーカに移動する (マイグレーションする) ようにした．具体的には，入力ストリームの流量を監視しておき，その値が著しく減少した場合に，そのタスクの実行を中断して，別のワーカに移動する．移動した先のワーカは，ストリームの再接続など必要な処理を行った後，中断した時点から実行を再開する．

マイグレーション時にどのワーカにタスクを移動するかについては，2.2 で述べた方法を用いる．

3 予備的な評価実験

3.1 実験に用いたハードウェア環境

実験に用いたネットワークの構成を図 2 に示す．また，ネットワークに接続されている計算機の構成を表 1 に示す．

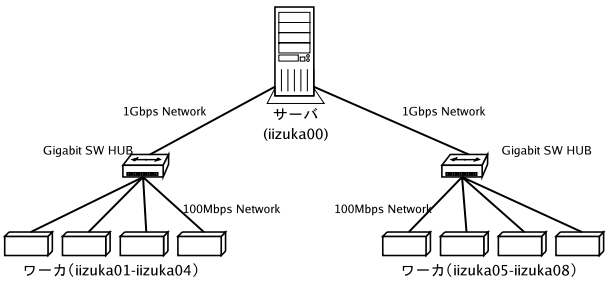

図 2: 実験に用いた計算機とネットワーク．
Fig. 2: The Networks for the Experimentations.

本研究が対象にしている環境は，多数の利用者が使用しており，負荷が動的に変動する分散コンピューティング環境である．この環境を模擬するため，MacOS X の標準ファイヤウォールである ipfw と，帯域制限ソフトウェアの thorottled [5] を用いて，他の利用者により負荷がかかった状態を模擬した．なお，実際に別のトラフィックを流して帯域幅を制限する実験も行ったが，あまり帯域幅に変化が見られなかった．この原因のひとつとして，MacOS X がプロセス毎に帯域の確保等のなんらかの高負荷時対策のための通信の制御をしている可能性がある．

以下のような挙動をする帯域制限プログラムを `iizuka01` と `iizuka08` で実行した．

1. 帯域制限する通信相手 l_h を，`iizuka02`〜`iizuka07` からランダムに 1 つ選択する
2. 帯域制限する時間 l_t を，60〜119 秒の間でランダムに設定する
3. 自ホストと l_h の間の通信帯域を l_t 秒間だけ 512 KB/s に制限する
4. 帯域制限を解除して 1 に戻る．

この帯域制限プログラムにより，`iizuka01` と `iizuka02`〜`iizuka07` の中の 1 ホスト間，および `iizuka08` と `iizuka02`〜`iizuka07` の中の 1 ホスト間に，つねに帯域制限が掛かっている状況になる．

3.2 並列分散処理プラットホーム

実験に用いたソフトウェアプラットホームの構成を図 3 に示す．これは大きく分けて，タスクスケジューリングシステムと資源情報サーバ (RIS) [6] のふたつの部分から構成されており，すべて Java を用いて実装されている．各モジュール間の通信には JavaRMI および TCP/IP が用いられている．

3.2.1 タスクスケジューリングシステム

タスクスケジューリングシステムは，タスクの組み合わせで定義されたユーザ・プログラムをスケジューラからの指示に基づいてワーカに割り付けることにより，並列分散コンピューティングを実現するものである．システムは，タスクコントローラ，タスクマネージャ，タスクスケジューリング GUI の 3 つのモジュールと，タスクコントローラが使用するスケジューラクラスから構成されている．各コンポーネントの役割は以下の通りである．

表 1: 実験環境計算機のスペック
Table 1: The Specification of machines.

	サーバ (iizuka00)	ワーカ (iizuka01-08)
OS	Momonga Linux (kernel2.6.10-26msmp)	MacOS X 10.3.9
CPU	Intel® Pentium® 4 3.20GHz	PowerPC G4 1.42GHz
メモリ	1024MBytes DDR SDRAM	512MBytes DDR SDRAM
その他	Java 1.5.0_04	Java 1.4.2_09 throttled 0.4.2

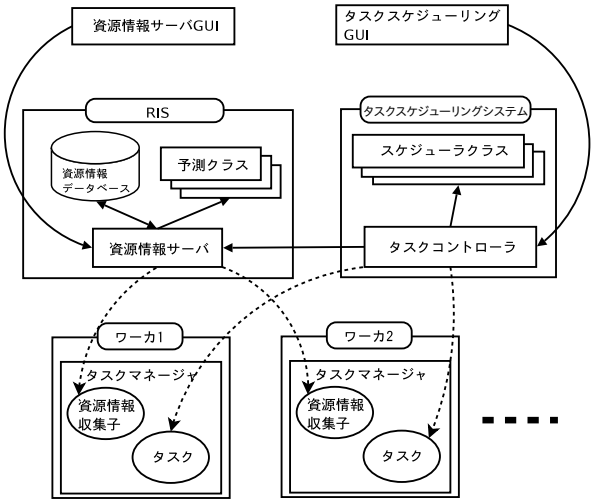

図 3: ソフトウェアプラットホーム.
Fig.3: The Software Platform.

タスクコントローラ：ユーザにより投入されたユーザ･プログラムを構成するタスクを，ユーザが指定したスケジューラクラスに基づき，タスクマネージャに割り付ける．

タスクマネージャ：各ワーカ上で動作しており，タスクコントローラにより割り付けられたタスクを実行する．

タスクスケジューリング GUI：ユーザがタスクコントローラを制御するための GUI フロントエンドである．タスクを組み合わせてユーザ・プログラムを構成したり，使用するタスクスケジューラを選択して実行したりできる．スクリーンショットを図 4 に示す．

タスクやタスクスケジューラは，あらかじめ用意されている Java で書かれたインターフェースクラスを実装することにより，ユーザが自由に作成することができる．タスク間のデータ交換とし

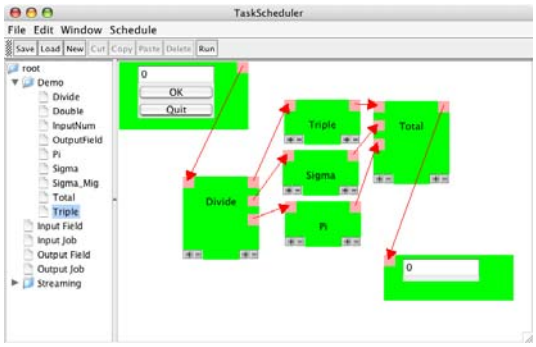

図 4: タスクスケジューリング GUI.
Fig.4: The GUI for Task Scheduling.

ては，ひとつのタスクが終了する際に次のタスクに情報が受け渡される (返り値と引数) タイプの他に，タスクが終了せずにストリーミングデータとして次のタスクに情報を受け渡す (ストリーミング) タイプの両方をサポートしている．前者は Java RMI を用いて実装されており，後者は TCP/IP ソケット通信を用いて実装されている．本研究では，このタスクスケジューリングシステム上で動作するタスクスケジューラを開発し，ストリーミングデータ処理を含む検証実験用ユーザ・プログラムを作成することにより，評価を行った．

3.2.2 資源情報サーバ (RIS)

資源情報サーバ (RIS: Resource Information Server) は，各ワーカのプロセッサ負荷やネットワーク負荷を含む資源情報を計測・蓄積し，過去の情報の提示や予測を実現している [6]．この種の資源情報の予測システムでは，NWS(Network Weather Service [7] が有名だが，直後の予測値だけではなく，今後の負荷変動の仕方なども予測する必要があるため，そのための予測手法が組み込

まれている RIS を使用した．今回使用した版は，[6] と同じ基本設計を用いて，タスクスケジューリングシステム上のユーザ・プログラムとして新たに実装したものである．[6] では C 言語 (一部は Fortran) と MPI を用いて実装されていたが，今回使用した版は，Java 言語と JavaRMI を用いている点が異なっている．ユーザがこれを操作するための GUI のスクリーンショットを図 5 に示す．

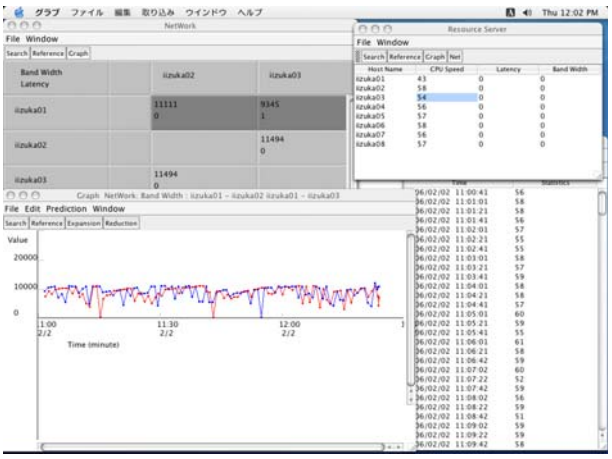

図 5: 資源情報サーバ GUI.
Fig.5: The GUI for RIS.

3.3 手法の実装

2 章において提案した手法を評価するため，2 種類のタスクスケジューラの実装を行った．ひとつは，タスクマイグレーションを行わずに，タスクの実行開始時に 2.2 で示した選択方法によりワーカを決定し，一度ワーカに割り付けられたタスクはそこでユーザプログラム終了まで実行される．ふたつ目は，タスクマイグレーションを行う．実行開始時に 2.2 の方法によりワーカを決定し，実行を開始した後，入力ストリームの流量を監視する．データ流量は 10 秒間を単位時間として測定し，その値が

- そのワーカに割り付けられてから測定された流量の最大値の 1/10 以下となった
- 80kbit/秒以下となった

のいずれかの条件となった場合，マイグレーションを行うものとした．

今回は RIS の高精度予測手法 [6] を行わずに，過去 1 分間のネットワーク帯域幅の平均値を予測値として用いた．高精度予測手法を用いなかった理由は，実験環境が普段あまり利用されてない環境であり，予測に必須となる有効なログが無いことと，実験の為に加えた負荷が帯域制限という特殊なものであるためである．

3.4 評価と考察

評価用のユーザ・プログラムとして，図 6 に示す簡単なものを用いた．このユーザ・プログラムは，`iizuka01` でバイナリファイルを読み込み (`FileRead`)，ストリームを用いて `iizuka02` から `iizuka07` のいずれかの計算機を中継し (`FileThrough`)，最終的に `iizuka08` にストリームでデータを送りファイルに書き込む (`FileWrite`).

図 6: 評価用ユーザ・プログラム.
Fig.6: The User Program for The Evaluation.

図 2 に示す通り，`FileRead` を実行する `iizuka01` と，`FileWrite` を実行する `iizuka08` はルータを介して別のネットワークで実行される．これは同一ネットワークで実行すると，中継タスクである `FileThrough` も通信特性が良好な同一ネットワーク内に配置されるため，結果が偏るからである．

評価するタスクスケジューラが扱うタスクはデータストリーミングの入力を持つ `FileThrough` と `FileWrite` である．ワーカを選択する 2.2 で示した式にこれらのタスクを適用した．ファイルアクセスが含まれる `FileWrite` はマイグレーションの対象とすることができないので，マイグレーションの対象となるタスクは `FileThrough` のみである．

この評価用ユーザ・プログラムをマイグレーションを行うスケジューラと行わないスケジューラのそれぞれを用いて実行し，処理が終了するまでに要した時間を計測した．コピーするファイルのサイズはファイル A が 588MB，ファイル B が 2.6GB であり，各 10 回実行した．結果を表 2 に示す．実行時間と転送速度は平均値である．

結果より，どちらのファイルを用いた場合でも，

表 2: 実験結果.
Table 2: The Result of the Experimentations.

	ファイル A		ファイル B	
マイグレーションの有無	無	有	無	有
平均実行時間 [秒]	682.8	455.5	3130.5	2106.9
(最長)[秒]	1160	654	4959	2855
(最短)[秒]	371	294	2475	1524
実行時間の標準偏差	230.0	134.6	693.8	357.1
平均転送速度 [K バイト/秒]	882	1322	858	1275

マイグレーションを行うことで全体の実行時間を約 33%短縮できた．また，マイグレーションを行うことにより，実行時間のばらつきを抑え，安定した性能を得られることが確認できた．

4 おわりに

本研究では，ストリーミングデータ処理を含む分散プログラムの実行時間を短縮できるタスクスケジューリング手法の提案を行い，その実現に向けて予備的な評価実験を行った．その結果，タスク間のストリーミング通信の効率が良くなるようにタスクマイグレーションを行うことにより，計算時間が短縮化され，計算時間のばらつきも小さくなることが確認できた．

今後の課題として，ネットワーク負荷やプロセッサ負荷の予測手法との連携，タスクマイグレーションのコストを考慮することによるさらなる効率化を行い，マルチメディア処理などの実際的なアプリケーションによる評価を行っていく予定である．

参考文献

[1] Kwok, Y. and Ahmad, I.: Static Scheduling Algorithms for Allocating Directed Task Graphs to Multiprocessors, *ACM Computing Surveys*, Vol. 31, No. 4, pp. 406-471 (1999).

[2] Koide, H. Hirayama, H., Murasugi, A. Hayashi, T. and Kasahara, H.: Meta-scheduling for a Cluster of Supercomputers, *Proceedings of International Conference on Supercomputers Workshop, Scheduling Algorithms for Parallel/Distributed Computing - From Theory to Practice* -, 1999, pp. 63-69.

[3] Yamamoto, H., Kawahara, K., Takine, T. and Oie, Y.: Investigation and Fundamental Analysis of Process Allocation Scheme on Grid Computing Environment, *Techical Report of IECE*, IN2002-8, pp.43-48 (2002).

[4] Standard Task Graph Set: Kasahara Laboratory, Waseda University, `http://www.kasahara.elec.waseda.ac.jp/schedule`.

[5] InterArts Creative Media – Throttled Pro –, `http://www.intrarts.com/throttled.html`.

[6] 小出洋, 山岸信寛, 武宮博, 笠原博徳, 情報資源サーバにおける資源情報予測の評価, 情報処理学会論文誌:プログラミング, Vol. 42, No.SIG3(PRO 10), pp.65-73 (2001).

[7] 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).