システムソフトウェアと
オペレーティング・システム 77－17
マルチメディア通信と分散処理 87－17
（1998. 2. 26）

大域並行計算とそのメッセージ評価スケジューリング

前川博俊 千葉哲央 斉藤隆之
(株)ディジタル・ビジョン・ラボラトリーズ

我々は，広域のネットワークをひとつの統合的な計算環境として捉え，その上で大域的な並行計算を行える大域計算方式を提案している．この方式では，計算オブジェクト間で送信されるメッセージの評価は，その時間属性と優先度に基づいて大域的な枠組みでスケジューリングすることができる．そのスケジューリングは，OS 独立で大域的な時間制御と負荷均衡化の機能を提供する長期スケジューリングと，OS 機能を用いてメッセージの評価実行を行う短期スケジューリングで実現する．本稿では，このメッセージ評価スケジューリングの方式とその基本実時間特性について述べる．

Global Concurrent Computation and Its Message Evaluation Scheduling

Hirotoshi MAEGAWA, Tetsuhiro CHIBA, and Takayuki SAITO
Digital Vision Laboratories Corporation

We have proposed a global computation scheme capable of developing a global concurrent computation over synthesized environment of wide-area networks. In this scheme, evaluation of messages passed between computation objects can be scheduled in a global fashion based on those timing-attributes and priorities. The scheduling scheme is composed of the long-term scheduling that is independent of OS and provides functions of timing control and load balancing, and the short-term scheduling that executes message-evaluations using OS features. In this paper, we describe the message evaluation scheduling scheme and evaluate its fundamental real-time performance.

1. はじめに

我々は，広域のネットワークを統合的なひとつの計算環境として捉え，その上で大域的な並行計算を行うことが可能な大域計算方式を提案している[1-3]．これは，分散マルチメディア処理や分散問題解決といった多様な分野での広域計算を，様々な計算リソースをネットワークワイドな多様な局面で柔軟に利用した上で，容易に構築できるような枠組みの提供を目的としたものである．提案する方式は，次の特長を持つ：

- 広域のネットワーク環境を統合的に捉え，その環境上で大域的並行計算が可能な計算パラダイムを有する．
- プログラミング言語や OS，ネットワーク等の違いによる多様な実行環境に適応し，広域環境での大域並行計算を可能とするメッセージパッシング機構を提供する．
- 計算空間上のオブジェクトは大域参照で管理し，大域参照を用いた効率のよい空間管理とオブジェクト間メッセージ送信を実現する．
- 連続メディアデータを含むマルチメディア構造体を，ネットワーク上で大域的に扱うことができる．
- ネットワーク上のリソースを動的に探索して利用することが可能なメディエーション機能を持つ．

この方式をさらに，対話的連続メディア配送制御や知的分散信号解析といった問題領域に適用するとき，広域分散環境での実時間処理が，実現すべき重要な技術要素となる．そのような領域では，ネットワーク応答や時間共有の特性あるいは問題自体の動特性に起因した不確定要素が多く，決定論的な予測に基づくスケジューリングは難しい．分散環境のための実時間処理の方法として，タスクの大域管理[4]やメッセージ連鎖の解析的制御[5,6]，階層的なタスク配置管理[7]などが提案されているが，それらは広域環境での計算や非決定的処理を包括しているものではない．また，実時間性の実現において考慮すべき負荷分散の方式[8-10]や，実時間制御機能を備えた並行オブジェクト言語系[11-14]などが提案されているが，それらの方式はさらに，大域的なスケジューリングの枠組みの中で活用したい．

我々が提案する計算方式では，計算のスケジューリングは，オブジェクト間メッセージの評価の実行を制御することによって行う．スケジューリング機能は，OS 独立で大域的な時間制御と負荷分散を行う長期スケジューリングと，OS 機能を用いてメッセージ評価を並行に実行する短期スケジューリングとからなる．

本稿の以下においては，提案する計算方式における

大域的な計算の基本パラダイムについて述べ，そのパラダイムに基づいたオブジェクト間メッセージの評価スケジューリングの方式を提案する．また，その方式による大域的な実時間処理の基本性能を評価する．

2. 大域並行計算の方式

大域計算(Global Computation)とは，広域のネットワーク上の計算であって，そこでの分散リソースを論理的に一体のものとして捉え，そしてその上で実行される計算であるとする[1]．従来の広域(wide area)計算は，統合環境上での一体化された計算ではないと解釈し，大域計算とは区別する．大域計算を実現する基本パラダイムについて述べる．

2.1 並行計算の基本パラダイム

基本となる計算の方式は，分散モジュールが非同期のメッセージ交換によって並行計算を実行し，大域的にはそのメッセージの連鎖によって計算が進行するものである．広域の分散・協調計算への適合性を考慮すれば，同期性の少ない計算の枠組みの提供が望ましい[15]．

大域計算を実行する分散モジュールを，並行オブジェクトと呼ぶ．並行オブジェクトは，ある大域計算空間上で表される大域参照で管理する．大域計算空間は，並行オブジェクトの集合と，大域参照を用いて示されるオブジェクト間関係との組である．

並行オブジェクトは，計算空間上の計算実体であり，リモート呼出可能な分散オブジェクト，分散関数，あるいは，大域共有変数である．これらによる次のパラダイムを持った並行計算を行う：

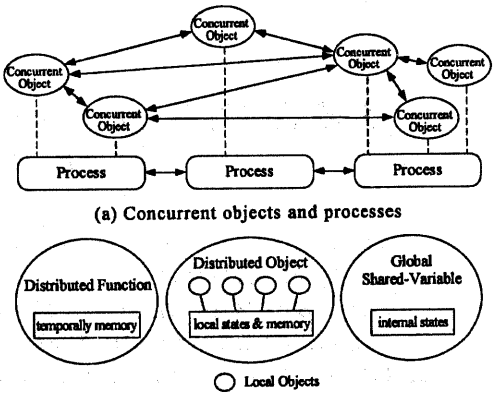
- 分散オブジェクト指向計算
- 分散手続き(関数)呼出
- 大域共有変数

この計算を行うための並行オブジェクト間の通信(分散オブジェクトメソッドや分散関数の呼出，大域共有変数へのアクセス)は，次の方法により行う．

- 非同期メッセージ送信(返値なしの一方向の通信)
- 遅延評価型同期呼出(返値にアクセスしたときブロックする)
- 完全同期呼出(返値到着まで呼出側がブロックする)

これらの通信を総称してメッセージと呼ぶ．メッセージ上では並行オブジェクト間で，任意のデータ構造体を送ることができる．

メッセージの評価のうち分散オブジェクトメソッドの計算と大域共有変数アクセスは，そのオブジェクトにおいて排他的かつアトミックに実行する．同一のオブジェクト上で複数のメッセージを並行して評価したい場合や，メソッド内で遅延処理をしたい場合は，それらを非同期メッセージに分割した形で記述し処理する．また，実際に稼働させるプログラミング言語によって，メソッドがオブジェクトに属するのであれば，その実行はそのオブジェクトで行う．メソッドが汎関数から派生するものであれば，実行はそのメソッドの所在で行う．計算の演算対象のスコープは，並行オブジェクトの内部である．リモートオブジェクトのオペランドを直接見ることはできない．



(a) Concurrent objects and processes

(b) Concurrent objects and their internal states

Fig. 1 Concurrent computation environment.

2.2 オブジェクト管理とメッセージパッシング機能

大域計算空間に配置する並行オブジェクトは，ネットワークノードにプロセスを生成して管理する．計算空間は，同一のネットワーク環境上に複数のものをそれぞれ個別に形成し管理することができる．プロセスは，並行オブジェクトのための計算環境である．並行オブジェクトとプロセスの関係と，並行オブジェクトの内部状態構成を，Fig. 1に示す．

プロセスは，大域参照とその管理の機能[3]と，並行オブジェクト間のメッセージパッシングの機構[2]を提供する．メッセージパッシング機構は，アプリケーション言語インタフェース(API)と，OS やハードウェアに対するシステムインターフェースを備える．オブジェクトとプロセスは，計算空間上をマイグレートすることが可能である．

これらの機能は，プログラミング言語や OS，ハードウェア，ネットワークといった実行環境の違いに適応して稼働できる．メッセージは，異なる言語間でも送信可

能であり，また，任意のデータ構造体を引数として扱える．プロセスの集合として得られる実行環境は，大域空間上で共通の仮想的計算環境として機能する．

3. メッセージ評価スケジューリング

計算は，オブジェクト間のメッセージの連鎖によって実行される．従って，その計算のスケジューリングは，並行オブジェクト間で送信されるメッセージの評価の実行を，その時間制約を示す属性と優先度に基づいて制御することによって行う．スケジューリングは，次のふたつの処理に分けて実現する：

- 時間的長期性と空間的広域性の尺度から，メッセージ評価の順序を，OS 等のシステム機能と独立して制御する．
- 順序付けられたメッセージ評価を，短期的時間尺度によって局所的に，システム機能を有効に活かしながら実行する．

これらの処理をそれぞれ，長期スケジューリング，短期スケジューリングと呼ぶ．ここでは，メッセージ評価スケジューリングの方式と機能について述べる．その方式の機能構成を，Fig. 2に示す．

3.1 大域的スケジューリングの方式

実時間処理を扱うスケジューリングは，「時間」に基づいた計算の規範を与えるものである．メッセージの連鎖によって生起される計算の場合，「時間」はその連鎖の中で整合していればよい．スケジューリングを行う局所的な場のそれぞれが，大域的に一貫した時間を共有している必要はない．メッセージには必要に応じて，その評価における時間制約が属性として記述される．

スケジューリングの対象となるタスクの単位は，オブジェクト間で送信されるメッセージの評価である．スケジューリングの処理は，プロセス毎に局所的に行う．プロセス間では，大域的な負荷分散を支援するメタ機能を提供する．プロセスは，現在の実装では「OS のプロセス」に対応している．

タスクの実行は，長期と短期のスケジューリングによって行う．長期スケジューリングでは，メッセージに付帯した時間属性と優先度に基づいて，評価すべきメッセージを選択して実行可能状態とする．大域的な時間制約は，長期スケジューリングで解決する．短期スケジューリングでは，可動状態にある複数のメッセージ評価を並行して処理するための制御を行う．

メッセージが属性として持つのは，時間属性と優先度である．時間属性は，スケジューリングポリシと，データタイムやメッセージ送信タイム，評価の許容時間(デッドライン)，メッセージの周期性，実時間性の厳密度，計算の非精密性の許容度といったものを含む．ここで，データタイムとは，生起時間や外界時間など，処理対象のデータに局所的に与えられた時間を大域的に扱うものである．スケジューリングポリシとして現在扱っているのは，データタイム，メッセージ送信時間，同到着時間，デッドライン時間のそれぞれ時間順と，デッドライン長と計算周期のそれぞれ長さ順である．

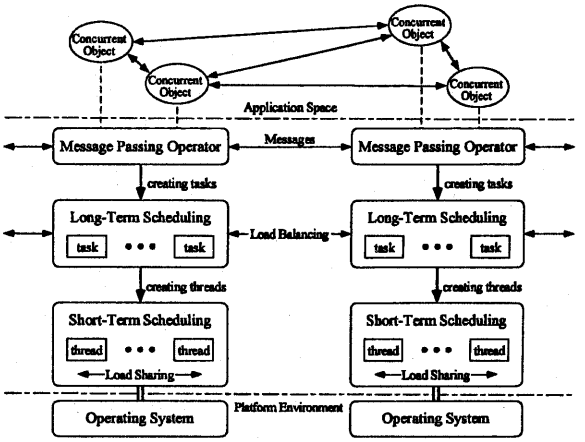


Fig. 2 Message evaluation scheduling scheme.

実時間処理では，プロセス間あるいはノード間での負荷の均一性が，系の実時間性を高める上で重要である．長期スケジューリングではノード間での負荷均衡を，短期スケジューリングでは，処理リソースが割り当てられた空間内での負荷共有を，スケジューリング方式の中に採り入れる．負荷分散の処理はしかし，負荷を動的に把握した上でのシステム機能として実現するのは得策ではない．その機能は，アプリケーションの実行において負荷分散の実現を支援するメタ機能として提供する．

メッセージ評価としてのタスクの実行は，OS のスレッド機能によって行う．分散オブジェクト計算系の多くでは，個々のオブジェクトがメソッド実行のスレッドを有している[11,12]．我々の方式では，スレッドはメッセージ評価タスクに対して動的に割り付ける．これはまた，スレッド資源を有効に活用する実践的手法でもある．

メッセージパッシング機構におけるメッセージ受信機能の一部とスケジューラの構成を，Fig. 3に示す．メッセージディスパッチャは，受信したメッセージから，その評価の実行情報と属性を有する関数記述子を生成す

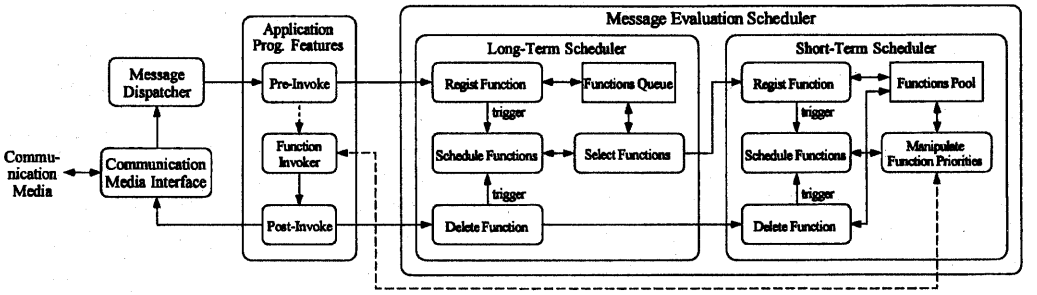


Fig. 3 Message evaluation scheduler configuration.

る. スケジューラは, 関数記述子をタスクとして受け取り, その実行を制御する. スケジューラは, それぞれ長期と短期のスケジューリングを行う長期スケジューラと短期スケジューラとからなる.

3.2 長期スケジューリング

長期スケジューラは, 受け取ったタスク(関数記述子)をキューに格納し, 評価を起動すべきタスクをそのメッセージ属性に基づいて選択する. 選択したタスクはキューから取り出し, 短期スケジューラに送る. 長期スケジューリングは, メッセージの受信とメッセージ評価の終了をトリガーとして動作させる.

長期スケジューリングで扱うのは, 大域的尺度での対象タスクの評価順序である. 評価順序は, メッセージの優先度と時間属性およびプロセスでの現在時間から, スケジューリングポリシに基づいて算出するタスク優先度で表す. 長期スケジューリングでは, 時間順のポリシをより重視する. タスクは, その優先度別のリストをさらに優先度順にリストにしたキューで管理する. タスク優先度は, 算出時のプロセス時間を反映した値で, 時間の経過とともにキュー上での順位が動的に変化する. タスク優先度順位の大域性は, 時間属性のデータタイムなどの大域的時間要素から得ることが出来る. 優先度の算出が局所時間からのみによる場合, その順序の確度は, 確率的にノード間での局所時間の相対差のオーダである. [1]

同一の分散オブジェクトでのメッセージ評価は排他的に行う. 分散オブジェクトにはメッセージ実行状態を保持し, その状態に基づいて関数記述子の選択を行う. 従って, 同一の分散オブジェクトへのメッセージが輻輳したとき, 長期スケジューリングでのタスク優先度の順序性が保てなくなる. これは, 負荷分散による解決の対象である(3.4節参照).

3.3 短期スケジューリング

長期スケジューラから送られたタスクは, 実行スレッドを割り付けてプールで保持する. 実行スレッドは, その関数記述子の情報をもとに実際の関数やメソッドを呼び出して処理するものである. 短期スケジューリングは, 長期スケジューラからのタスクの受取とメッセージ評価の終了をトリガーとして動作させる.

短期スケジューリングでは, プール上で可動状態となっている関数記述子の評価を, OS のスレッド機能を用いて, 実行制御する. スレッド優先度は, メッセージの時間属性と優先度から再度, 短期的尺度に基づいて算出する. 短期スケジューラは, デッドラインまでの余裕度に従った優先度の動的変更といった処理をする. これは, OS のスケジューリングに対するメタ機能として位置付けることができる.

スレッドの実行を OS 機能によって行うのは, メッセージ評価以外のタスク [2] の実時間処理や, 実行環境への適応性の観点からも妥当である. OS によってリソースロックなどに伴う優先度継承などの処理が行われた場合, そのスレッド優先度の操作は一時的であって, 短期スケジューラによる優先度の操作は, それとは非干渉で行えるとする. ノード上に複数の PE (processing element)がある場合, プロセスやオブジェクト配置に基づく PE 空間の分割や, スレッドの PE への割付などは, 短期スケジューラによるチューニングの対象である(3.4節参照).

3.4 負荷分散

負荷分散には大別して, 負荷均衡化と負荷共有化の手法がある. これらを, その特性に基づいてそれぞれ, 長期と短期のスケジューリングの枠組みに採り入れる.

長期スケジューリングでは, プロセス間およびノード

---

[1] ノード間での局所時間の差は, メッセージ送信の最小時間のオーダで認識することが可能である.

[2] 例えば, 我々の計算方式においても連続メディア処理を採り入れており[1,2], 試作システムにおいてもその機能が稼働している. その局所制御は, プロセス内のドメインタスクとして, 短期スケジューラの処理対象に組み込む予定である.

間で負荷を均衡させ，非同期性の発生を抑制するとともに，系全体の応答性を高める．広域分散環境上での負荷均衡化は，[9]や[15]で実現しているような負荷拡散型の制御を行うのが好適である．そのためアプリケーションに対して，複製オブジェクト間でのメッセージ分配や，新たなオブジェクトとプロセスを生成する際の配置決定の支援機能を提供する．アプリケーションでは同時に，そのドメイン知識によって，オブジェクトとプロセスのマイグレーションの処理をしてもよい．我々の計算方式では，大域参照を活用した効率のよいマイグレーション機能を提供する．

短期スケジューリングでは，生起したメッセージ評価タスクを遅滞なく実行しなければならない．実時間処理における負荷共有化では，非周期的なタスク入力に対応するため，[8]で述べられているようなタスク充填型の処理が求められる．この処理はしかし，負荷モニタのコストの観点から，ひとつのプロセスが複数の PE に共有されているときに行う．

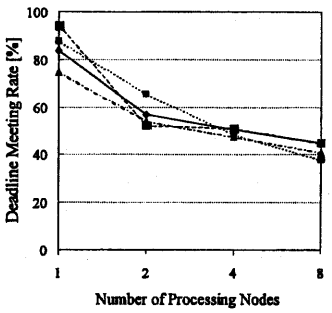
4. 実時間特性の評価

提案する方式に基づいた大域並行計算の試作システムが現在，Solaris 2.4/SparcStation と WindowsNT 4.0/IBM PC/AT 互換機，10Base-T Ethernet / 156Mbps ATM の環境上で，C++, Java を API 言語として稼働している．短期スケジューリングでは現在，Solaris 2.4 と Windows NT 4.0 が提供する以上の実時間化の処理はしていない．

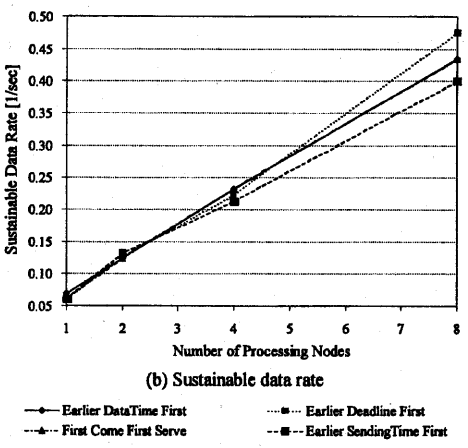
本稿では，システムの実時間特性評価の予備検証として，メッセージ伝播の時間応答性に関する基本的性能を評価する．次の問題において，その処理能力上限値付近での応答性を，ノード数とスケジューリングポリシを変えて観測した．実験は，C++/Solaris/SpacStation (HyperSarc/125MHz)/10BaseT Ethernet 上で行った．

実験問題は，64 分散オブジェクトの系に一定頻度で 8 メッセージを並行入力し，入力に伴ってオブジェクト間でメッセージをそれぞれ 32 段伝播させ，その応答性を観測するものである．メッセージは全て非同期で，その評価の計算粒度は均一とした．複数ノードの場合はメッセージの送信先を，発信元とは別のノードを中心にガウス分散させた．デッドラインは，処理能力上限での系全体の応答時間を 32 等分した時間を，各メッセージに均等に与えた．スケジューリングポリシは系全体で同一とした．長期スケジューリングのみを行い，短期スケジューリングではスレッド優先度を均一にした．

処理能力は，メッセージ(データ)入力の上限頻度で



(a) Deadline meeting rate



(b) Sustainable data rate

Fig. 4 Performance of response.

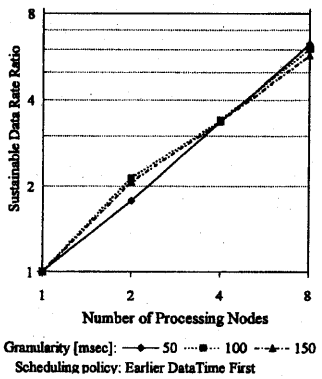


Fig. 5 Performance Curve

ある Sustainable Data Rate (SDR)によって測定した．SDR は，メッセージやデータの伝達頻度であって，所定の観測レーテンシが安定していて時間経過とともに増大することのない最大値である[15]．

ノード数を変えて系の応答性を観測した結果を，Fig. 4に示す．(a)はスケジューリングポリシ毎の SDR でのデッドライン充足率，(b)は(a)の各観測値での SDR であ

る. 処理限界(SDR)では, ネットワーク通信負荷による内部処理の乱れが認められるが, 系全体の応答能力は, ノード数に対してほぼリニアに増加している. さらにメッセージ評価粒度を変えたときのノード数に対するSDR 増加比を, Fig. 5に示す. 分散処理では, 本実験のように非同期で均質な制御を行えば, 理想的な台数と応答性の効果を得ることができる. 我々の計算方式も, 分散処理の特質を活かした基本性能を備えているといえる.

本実験は分散系全体での実時間応答性を観測したものであり, Fig. 4においてもスケジューリング性能の詳細を顕現化するには至っていない. 今後, この予備実験を基に, より複雑な問題による評価を行う予定である.

5. まとめ

本稿では, 我々が提案している広域ネットワーク環境上での大域並行計算の方式と, そこでのメッセージ評価スケジューリングの方式と機能について報告した. そのスケジューリング機能は, 次の特長を持つ:

- 広域のネットワーク環境環境上での大域的な計算スケジューリングが可能である.
- そのスケジューリングは, 評価すべきメッセージをその時間属性と優先度によって選択する長期スケジューリングと, 選択された評価を並行して処理する短期スケジューリングとで実現する.
- 長期スケジューリングでは, 系全体の応答性向上のための負荷の均衡化の処理に, また, 短期スケジューリングでは, 実時間性の充足のための負荷の共有化の処理に対応する.
- 長期スケジューラはタスク生成器として, 短期スケジューラはタスク実行器として機能する. 実効的なタスク処理により, 計算実行リソースの効果的な利用が可能である.
- 長期スケジューリングは, 実時間制御のための OS 独立なミドルウェア機能として, また, 短期スケジューリングは, 計算実行のためのOS機能インタフェースとして位置付けることができる.

提案している大域並行計算のシステムは現在, Solaris2.4, WindowsNT4.0とEthernet/ATMの環境上で, C++, JavaをAPI言語として, プロトタイプが稼働中である. プロトタイプ上で, メッセージ評価のスケジューリングに関する基本性能を測定し, システムの実時間性評価のための予備検証を行った. 提案する広域環境上での大域的スケジューリングは, その方式の提示と試験実装に留まっている. 今後, アルゴリズム構築とそれに基づいた解析による方式の吟味, および, 種々の実時間性能の評価が必要である.

広域環境でのマルチメディア処理や分散知識処理といった分野は, 実時間性の要求を高めながら, 今後ますますその重要性を増すと考えられる. 大域並行計算システムの開発を通じて, 提案するスケジューリング方式の有効性を検証していく予定である.

参考文献

[1] 前川博俊, 斉藤隆之, 千葉哲央:大域計算アーキテクチャ——広域環境での並行計算とマルチメディア処理の統合的実現, 情処論文誌, 39 (2), 1998 to appear.

[2] 前川博俊ほか:ネットワーク環境におけるマルチメディア並行計算プラットフォームの実現, 情処ワークショップ論文集, 96 (1), 417-424, 1996.

[3] 小早川雄一, 斉藤隆之, 前川博俊:ネットワークスケーラブルな分散オブジェクト空間管理方式, 情処研究報告, 97-DPS-80, 211-216, 1997.

[4] Kao, B and Garcia-Molina, H.: Deadline Assignment in a Distributed Soft Real-Time System, *IEEE Trans. PDS*, 8 (12), 1268-1274, 1997.

[5] Abdelzaher, T. F. and Shin, K. G.: Optimal Combined Task and Message Scheduling in Distributed Real-Time Systems, *RTSS*, 162-170, 1995.

[6] Natale, M. D. and Stankovic, J. A.: Dynamic End-to-end Guarantees in Distributed Real Time Systems, *RTSS*, 216-227, 1994.

[7] Feitelson, D. G. and Rudolph, L.: Distributed Hierarchical Control for Parallel Processing, *Computer*, 23 (5), 65-77, 1990.

[8] Bestavros, A.: Load Profiling in Distributed Real-Time Systems, BUCS-TR-96-017, Boston Univ., 1996.

[9] Hailperin, M.: *Load Balancing Using Time Series Analysis for Soft Real Time Systems with Statistically Periodic Loads*, PhD thesis, Comp. Sci., Stanford, 1993.

[10]Kurose, J. F. and Chipalkatti, R.: Load Sharing in Soft Real-Time Distributed Computer Systems, *IEEE Trans. Computers*, C-36 (8), 993-1000, 1987.

[11]Takashio, K., Shitomi, H., and Tokoro, M.: Constructing Distributed Real-Time Systems with DROL Real-Time Objects, *RTCSA*, 142-149, 1995.

[12]Ishikawa, Y., Tokuda, H., and Mercer, C. W.: An Object-Oriented Real-Time Programming Language, *Computer*, 25 (10), 66-73, 1992.

[13]Nigro, L. and Tisato, F.: RTO++: A Framework for Building Hard Real-Time Systems, *J. OOP*, 6 (2), 35-47, 1993.

[14]Gehani, N. and Ramamritham, K.: Real-Time Concurrent C: A Language for Programming Dynamic Real-Time Systems, *J. RTS*, 3, 377-405, 1991.

[15]Maegawa, H.: ConClass: A Framework for Real-Time Distributed Knowledge-Based Processing, *IEEE Trans. KDE*, 6 (6), 909-919, 1994.