卒業論文

題目

並列スクリプト言語MegaScriptの実行時情報視覚化

指導教員

大野 和彦 講師

2009年

三重大学 工学部 情報工学科
計算機アーキテクチャ研究室

松下圭吾 (405847)

内容梗概

近年，複雑な物理計算を要する多くの分野においては Pflops 以上の計算能力が求められており，100 万台規模のプロセッサを用いたメガスケールコンピューティングが必要となった．

そのため，メガスケール規模での並列処理を実現するためのタスク並列スクリプト言語 MegaScript が開発されている．MegaScript は，並列プログラムや逐次プログラムをタスクとして扱い，タスクと各タスクの通信路であるストリームの定義，接続等を記述して，並列処理を行わせる．

しかし，現在 MegaScript の統合開発環境が存在しないため，特に大規模な並列処理のプログラム作業は容易ではない．こういった理由から本研究室では，MegaScript 統合開発環境の構築をしているがコーディング支援部分の実装のみとなっており，デバッグ等に関する支援が存在しない．

デバッグやチューニングという作業は並列処理において欠かせない作業である．しかし，効率の良い作業を行うためにはプログラムに関する情報が必要になる．特にデバッグやチューニングにとって重要な情報は実行時に関する情報である．

そこで本研究では，実行時情報の中からユーザにとって有益度の高い情報を選定し，視覚化を行った．これによって，ユーザがプログラムの実行過程やホストに割り当てられる負荷の分散具合やホスト間で通信するメッセージに関する情報を把握する事を可能とした．また，視覚化した情報がプログラムのどのコードに関する情報なのかを分かりやすく視覚化する事で，プログラムの開発効率を高める事が出来た．

Abstract

In recent years, many reserch fields with complex physical calculations requires over Pflops computational capacity, and is needed mega-scale computing that uses 1 millions scale proceeors.

Therefore, in an environment marked by mega-scale, a task-parallel script language MegaScript is developed for the mega-scale computing. MegaScript treat with parallel or sequential program as a task, describe define the stream that is the channel of each task as the task, the connection etc. ,and the parallel processing is done.

However, because the integrated development environment of MegaScript doesn't exist now, the program work of an especially large-scale parallel processing is not easy.In this laboratory for such reasons, The MegaScript integration development environment is constructed. But, support concerning only mounting the coding support part and debugging, etc. doesn't exist.

Debugging and work of tuning are indispensable work in the parallel processing. However, to do efficient work, information on the program is needed. Especially, important information for debugging and the tuning is information on execution.

Therefore, in this study, I chose high information of the useful degree from run-time information for a user and visualized it. As a result, user can understand execution process of program, decentralized condition of load allocated by host, and information on the message communicated among hosts. Moreover, the development efficiency of the program was able to be improved by plainly visualizing whether visualized information was information on which code of the program.

目 次

図 目 次

表 目 次

1　はじめに

1.1　背景

近年，地球環境/災害シミュレータのような複雑な物理計算を要する分野においてはPflops以上の計算能力が求められている．しかし，逐次処理の限界や従来の専用並列計算機の設置には，莫大なコストや巨大な施設が必要となるといった理由から，100万台規模のプロセッサを用いたメガスケールコンピューティングが必要とされるようになった．そこで，オブジェクト指向言語Rubyをベースとしたタスク並列スクリプト言語MegaScript[1]が開発が行われている．効率の良い並列言語のプログラムを作成する事は容易ではないにも関わらず，MegaScript開発支援ソフトは存在しない．そういった理由から，開発効率を向上させるためMegaScript統合開発環境[2]の構築を行っている．

統合開発環境とは，コーディングからデバッグまで一貫して行える物である．しかし，現状のMegaScript統合開発環境はコーディング支援のみ行われている．

1.2 研究の目的

並列プログラムは高速性が求められる．しかし，並列処理において処理時間のボトルネックとなる部分は通信遅延や，負荷分散の不均等による遅延である可能性があるのでプログラムの出力結果だけでは，ユーザはボトルネックを特定出来ない．そこで，本研究ではタスク並列スクリプト言語 MegaScript を対象とした実行時情報表示機構を提案し，ユーザが効率の良い並列プログラムを作成するために必要な情報を選定して，明確にその情報の重要性を表示出来る手法を考案する事でユーザの補助を行う．これにより，深く並列処理を理解していなくてもボトルネックの可能性がある箇所を推測出来る事を可能とし，幅広いユーザが高速な並列プログラムを作成する支援を行う．

1.3 本文構成

本文の構成は以下のようになっている．まず，2 章では現在構築中の MegaScript 統合開発環境を紹介した後，情報視覚化という分野について説明する．3 章では提案する実行時情報視覚化機能の概要について述べ，4 章では提案手法の設計を説明し，5 章で提案手法の実装方法を述べる．6 章では，本設計に関する評価について述べ，7 章でまとめと課題を述べる．

2 背景

2.1 MegaScript

MegaScript は，100 万プロセッサ規模の並列処理を目的とするスクリプト言語である．MegaScript では，逐次プログラムや小規模な並列プログラムのような独立性の高い外部プログラムをタスクとして利用し，並列に実行することで大規模な並列性を生み出す．また，タスク間通信は，ストリームと呼ばれる論理的な通信路を通して互いにメッセージ交換を行う．タスクのスケジューリングは MegaScript が自動で行うので, ユーザはメガスケール規模でのプログラミングを少ない負担で行える．

2.2 MegaScript 統合開発環境

2.2.1 特徴

MegaScript はタスクネットワークを記述する言語であるため，図式表現によって視覚的にプログラムを作成するビジュアルプログラミング (VP) を用いる事でその構造を素早く正確に入力でき，容易にプログラミングを行えると期待できる．しかし，コードを一切記述出来ない事は使いづらい環境になってしまう．そこで，MegaScript のプログラミング支援として VP とテキストベースプログラミングの併用を採用し，より柔軟な

プログラミングを可能とした．MegaScript 統合開発環境は，Eclipse[3] のプラグインとして構築されており，タスクネットワーク表示は GEF[4] を用いている．

2.2.2 現在の状況

統合開発環境とは，エディタ，コンパイラ，デバッガなど，従来別々に利用されていたプログラムに必要なツールを一つのインタフェースで結合して利用できるようにしたものである．現状の MegaScript 統合開発環境は，コーディング支援のみ行われている．本研究では，MegaScript プログラム並列実行機能は不可欠なので，MegaScript 統合開発環境に並列実行機能を実装した．これにより，統合開発環境上での並列プログラム実行を可能とした．

2.3 情報視覚化

2.3.1 概要

情報視覚化とは，得られる情報をテキストやグラフを用いてユーザが最も認識し易い形で表示する研究である．ユーザは千差万別なので，人によって好き嫌いがあり，全ての人に最適な物を作り上げるのは困難である．こういった点から，ユーザが状況によって表示方法を選択出来る

ツールが多く存在する．

2.3.2 関連研究

私が本研究の前段階として調べた関連研究を紹介する．まず一つ目は，OminiOpenMP コンパイラ用並列プログラム可視化 [5]．これは，OpenMP の欠点であるスレッド操作の隠蔽を改善するためにバリア同期やクリティカル区間情報等を可視化するツールである．これは，for ループにおけるスケジューリング方法を視覚的に表示する事や全体の性能に大きな影響を及ぼす SCASH バリア同期に要する時間を他のバリア同期と区別して表示可能であるという特徴を持つ．このツールは確かに視覚化を行う情報の選択は問題ないと考えられる．しかし，MegaSctipt と OmniOpenMP コンパイラでは特徴が異なり，ユーザが得たい情報は同じではないのであくまで参考にしかならない．

次は，NaraView[6] を紹介する．これは，自動並列コンパイラ Parafrase-2 に対応する支援ツールで，Parafrase-2 が出力する並列化情報の一つである hierarchical task graph(HTG) の視覚化を行う．HTG を視覚化する事で，プログラムの構造を表示し，その中でユーザが指定した部分のデータ依存等の情報を表示する．この視覚化を，コンパイラの並列化処理とインタラクティブに行うことで，ユーザと自動並列化コンパイラの対話形

式の情報のやりとりが可能となる．しかし，このツールは自動並列化の情報を表示するので，本研究で得たい情報である実行時情報とは異なる．また表示方法として 3 次元の木構造を用いているが，MegaScript はデータフロー型のグラフを形成するので応用するのは難しいと考えられる．

最後に，VisuaLinda[7] を紹介する．これは，並列言語 Linda の実行状態を 3 次元視覚化を行う支援ツールで，時間変化・データフロー・CPU への割り当て等の情報の視覚化を行う．これにより，従来の 2 次元では情報別に表示画面が必要だったが，3 次元で表す事で多くの情報を一画面に納める事が可能となりユーザは一点に集中出来るといった利点を持つ．しかし，一画面に多くの情報が集まることで用途別での使用が困難になった．ユーザがツールを深く理解していれば問題はないが，ライトユーザにとっては情報別に画面を用意した方が得たい情報を的確に確認出来る．

3　提案手法の概要

3.1　目的

並列プログラムは高速に動作することが期待される．しかし，そういったプログラムを作成するには，多くの情報が必要となる．例えば，ホストに割り当てられる負荷が最適でない場合，あるプロセスがボトルネッ

クとなり処理時間に大きく影響を及ぼす可能性がある．これを解消するためには，各プロセスがどのホストで実行されてるかを知る必要がある．しかし，この情報を単純にテキスト出力で表示してしまうと，ユーザは把握しにくい．

そこで，情報をユーザが把握しやすいように視覚的に表示する事で，理解する時間を短縮しプログラミングの効率向上を図る．

3.2 特徴

現状の並列プログラミング環境において，実行時情報の視覚化を行うツールは存在しているが，それ自身を開発環境に含んだ物がない．また，MegaScript はタスクネットワークを形成する言語であり，開発環境上でプログラムをグラフィカル表現している．そのタスクネットワークグラフに実行時の情報を加える事で，他の視覚化ツールよりもプログラムと実行時情報の関連が把握しやすくデバッグ・チューニング作業効率向上に繋がると考えられる．

4　提案手法

4.1　収得情報の選定

情報の視覚化を行うに当たって，まず，どの情報を視覚化するかを決めなくてはならない．それには，MegaScript の並列モデルや並列処理の特性等を知る必要がある．そこで，4.1.1 で MegaScript の並列モデルについて述べ，4.1.2 でチューニング・デバッグについて述べる．

4.1.1　並列モデル

ここで MegaScript の並列モデルについて簡単に述べる．MegaScript は内部に組み込まれているスケジューラによって各ホストへとタスクを割り当てる．プログラムが実行されると，各ホストは自分に割り当てられているタスクを全て実行する．このため，他の並列言語だと上手く同期が取れないと実行が破綻する可能性がある．しかし，MegaScript はデータフロー型のネットワークであるタスクネットワークを形成する言語で，メッセージの通信方向が一定になっており循環する事がないので，デッドロックが起きない．

4.1.2　チューニング・デバッグ

並列プログラムは逐次プログラムより高速な処理である事が期待される．高速な並列プログラムを作成する時，必然的にチューニングを行わなければならない．負荷分散や通信ボトルネック等を初めから考慮してプログラムを作成する事はなかなか難しいので，チューニングは必要である．デバッグもユーザが一度で完璧なプログラムを作成するのであれば必要ではないが，現実的ではない．こういった理由から，ユーザは作成したプログラムに対してチューニング・デバッグを行うと仮定できる．

並列プログラムをチューニングする際，何を考慮するべきか．並列プログラムが逐次プログラムに比べて複雑な理由は，処理を行うホストが複数存在する事である．これにより，タスクの割り当てやホスト間の通信を考慮してプログラムを作成する必要性が出てくる．

4.1.3　取得情報の決定

4.1.2 でも述べたように，並列プログラムは高速でなければならない．そこで，チューニングの支援に重点をおいて情報を選択した．それには，プログラムのボトルネックを特定出来る情報を選択しなければならない．並列プログラムの処理時間が延びる原因になる可能性が高い要因とは何

かを考えた．

まずは，スケジューラの性能による処理能力の低下である．スケジューラの性能が悪いと，各ホストに最適なタスクの割り当てが出来ず，ホストに暇を与えてしまう可能性がある．そこで，各ホストの負荷分散具合を知るためにホストのCPU使用率情報とタスクの割り当て先情報である実行ホスト情報を表示する事にした．

次に考えられる要因として，プロセスの中断がある．これは，何かメッセージを受け取って処理を行う場合，そのメッセージを受信するまでプロセスは処理を行う事が出来ずに中断してしまう事である．MegaScriptにおいてプロセスはタスクに当たる．タスクは他の言語で書かれた実行ファイルであるので，タスクによって処理時間は様々である．これよって，タスクネットワークの各分岐で実行過程の推移に差が生まれてしまう．そのため，分岐の合流地点でタスクの中断が発生してしまう．これを解消するためには，プログラムの実行過程の推移を把握する必要がある．そこで，タスクCPU使用率情報とタスク実行完了情報を表示する事にした．

他の要因として通信遅延がある．ネットワークを介する通信はあまり速くなく，処理時間を下げる原因になりやすい．可能な限り通信は少なく

する方が良いと言われている．そこで，タスクから送信されるメッセージの情報を表示する事にした．

4.2 情報視覚化方法

情報の表示方法には，様々な方法がある．情報をテキスト表示するだけでは，ユーザはその情報の意味を理解出来ない可能性がある．そこで，4.1.3 で選択した情報をどのように表示すれば，その情報を表示した意図を理解してくれるかを考えた．

4.2.1 タスク CPU 使用率

並列処理において，実行過程の推移を把握する事は重要と言える．並列に実行されたプロセスの動作 (送信のタイミング等) によっては，他のプロセスを中断させる可能性もある．それによる遅延が，高速処理を妨げる一つの要因と言える．そこで，プログラムをグラフで表したタスクネットワークに各タスクの現在の CPU 使 用率を図 4.1 のように表示した．図中の Task にはそのタスクの変数名が表示される．実行中断したタスクは CPU 使用率が著しく低下していると考えられる．元々タスクネットワークはデータフロー型のネットワークで，プログラムの推移が上から下へ流れる事を表してので，並列性を見る場合は横の繋がりを見るだ

けで良い．横に並ぶタスクとの比較を行い，実行中断されているタスクを把握させる事でプログラムのボトルネック解消を支援する．

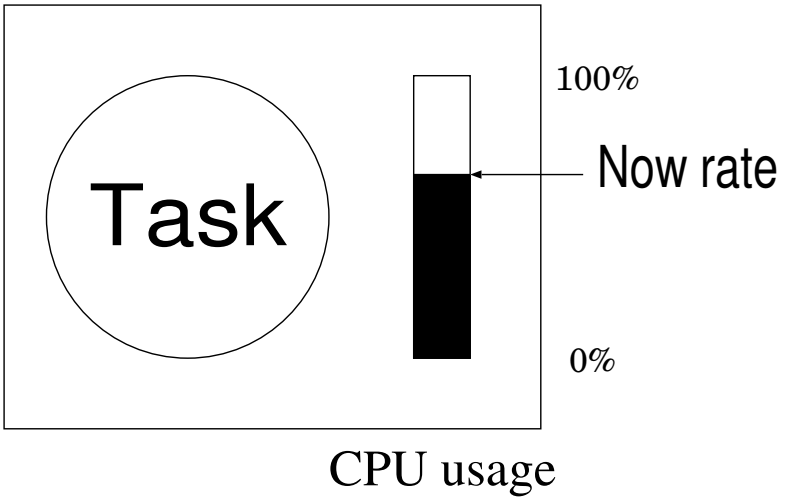

図 4.1: タスク CPU 使用率表示

4.2.2 ホスト CPU 使用率

複数のホストを使用する並列処理では，各ホストに割り当てられる仕事量が重要である．一台のみ集中したり，均等に割り振りされているがホストの性能が不均質である場合は，プログラムの実行時間に大きく影響する可能性がある．こういった理由から，各ホストの仕事量の割り当ては重要であると言える．そこで，各ホストの CPU 使用率を図 4.2 のように表示した．タスクの CPU 使用率表示とは異なり，一定期間の CPU 使用率の変化が確認出来る．各ホストの CPU 使用率を縦に並べて表示を行

う事で，同じ時間帯での比較が容易に出来るので負荷分散を確認出来る．

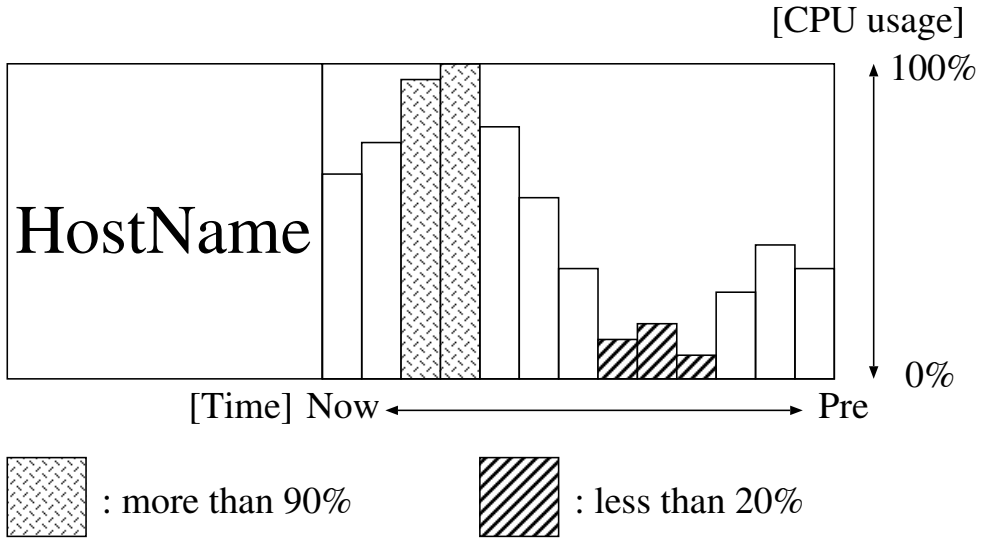

図 4.2: ホスト CPU 使用率表示

4.2.3 タスク実行完了情報

4.1 で説明した様に，実行過程を確認出来る事は有益である．データフロー型のタスクネットワークを用いているので，実行が完了したタスクを判別出来れば実行の推移を把握出来る．表示方法としては，色を用いる．完了したタスクは価値が無いとして画面から消す方法やタスクを大きく表示する方法等も考えられるが，前者は元のネットワークと対応が取りにくく，後者は密集して見にくい．

4.2.4 実行ホスト情報

MegaScript において，ホストに割り当てられる仕事量に相当するのはタスクであるので，ホストとタスクの繋がりを表示する事は有益であると言える．これにより，タスク数単位での仕事量の偏りが確認出来る．方法としては，色を用いて分別している．全タスクをホスト別に分別した表示を図 4.3 に示す．しかし，プログラム実行時にはタスク実行完了もタスクの色を変えて表示しているため競合してしまう．そこで，注目しているタスクに関連したホスト等のみ色で分別するピックアップ型の表示が必要だと考えられる．マウスポインタを注目しているタスクに合わせると，図 4.4 のように表示される．タスク単位のホストの偏りが知りたいのであれば，実行後にオリジナル型を用いる．また，実行中に CPU 使用率等と関連させながら見るのであればピックアップ型を用いる．これらを用途により使い分ける事で状況に応じた表示が実現できる．

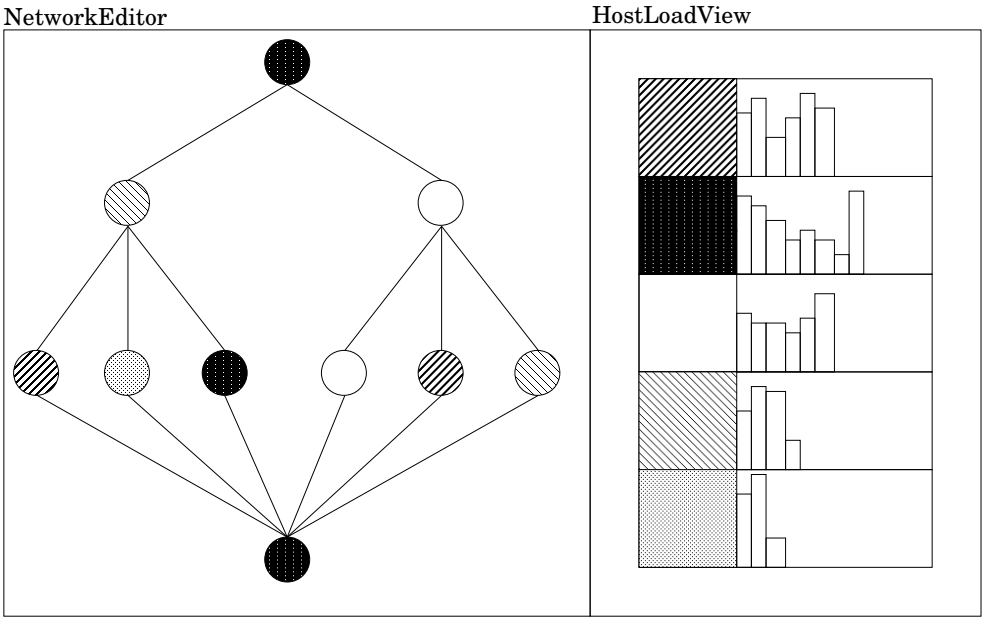

図 4.3: オリジナル型同一ホスト情報表示

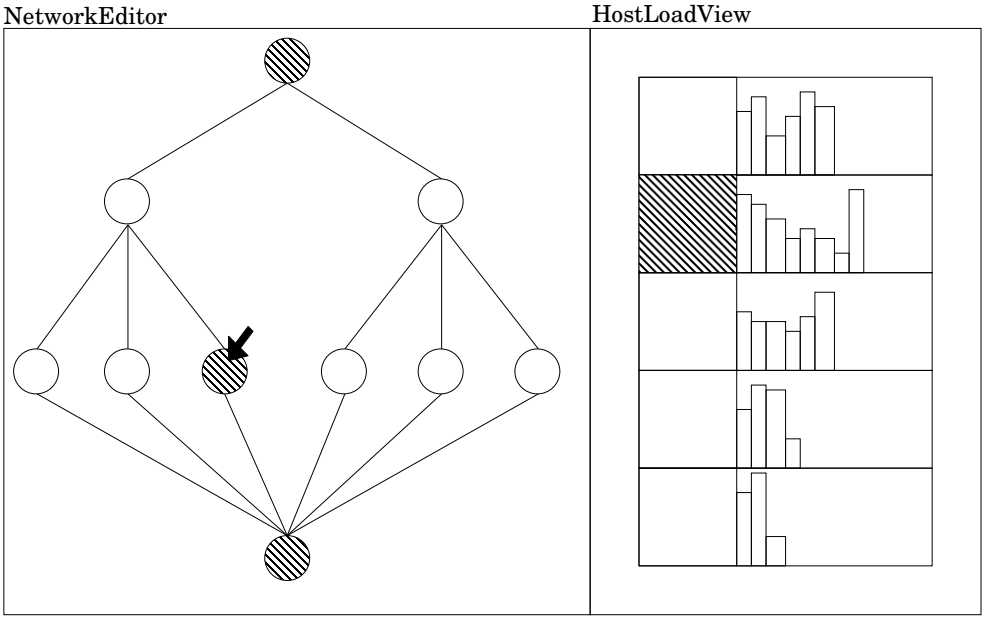

図 4.4: ピックアップ型同一ホスト情報表示

4.2.5 通信情報

並列処理は，逐次処理には無いホスト間の通信が行われる．この通信による遅延は，並列処理の大きなボトルネックとなっている．これを改善するためには，ボトルネックを特定しなければならない．一番単純な原因として，通信データ量が考えられる．膨大なデータを一度に出力したり，微少なデータを何度も出力する事は場合によって，遅延に繋がることがある．そこで，タスクから送信されるデータに関する情報から，送信回数・平均バイト数・最大バイト数を表示した．

5 提案手法の実装

5.1 Eclipse

Eclipse は，オープンソースで IBM によって開発された統合開発環境 (IDE) の一つである．Java をはじめとするいくつかの言語に対応していて，Eclipse 自体は Java で記述されている．Eclipse は機能統合環境にプラグインとしてさまざまな機能を組み込むことができるよう設計されているため，その拡張性は非常に高い．プラグインも Java で記述されている．MegaScript 統合開発環境は Eclipse のプラグインとして構築しているので，Eclipse の高い拡張性を利用することが出来る．

5.1.1 GEF

Eclipse のプラグインの一つ．GEF はモデルをグラフィカルに編集するアプリケーションを作成する為に使用するフレームワークで，UML エディタや GUI ビルダ等の複雑なアプリケーションをより少ない労力で作成することが出来る．

5.1.2 グラフ表示画面

Eclispe にはグラフ表示用の画面として，エディタとビューが用意されている．エディタはプログラム等のデータの作成・編集・保存を行う機能を持つ．ビューは様々な情報を表示し、かつそれらに対して操作を行うことのできるウィンドウである．これらを使い分ける事で，統合開発環境全体を構築する．MegaScript のプログラムを表すタスクネットワークはエディタを用いて表示する事で，保存を可能としている．今回の手法は，実行時情報の視覚化であるので情報を保存する必要がない．そこで，実行時情報の表示にはビューを用いた．

5.2 情報表示構成

5.2.1 MegaScriptNetworkEditor

MegaScript プログラムと同等のタスクネットワークをグラフィカルに表示するためのエディタである．このエディタは，MegaScript プログラムを作成するためのエディタであるので，こちらを操作する事でもプログラムを作成する事が可能である．本研究では，このエディタに表示する情報を増やすため改良を加えた．GEF は，描写を行う領域や描写されるグラフの形状を各オブジェクトで管理している．MegaScript 統合開発環境では，オブジェクトはタスクやストリームに当たる．今回の手法では，プログラム実行時のみ描写領域を拡大し情報を追加する事にした．これにより，プログラム作成時には余計なスペースを占有する事がなく表示スペースを効率良く利用できる．

5.2.2 HostLoadView

ホストの CPU 使用率を表示するために作成したビューである．ホストの CPU 使用率はプログラム実行時の情報なので，保存する必要がない．エディタでの作成は，ビューを作成するよりも労力が掛かる．そこで今回はビューでの表示した．

5.2.3 HostNetworkView

物理的なホストネットワーク環境を表示するためのビューである．しかし，現在ホスト間の繋がりを結ぶ表示しか行っていないため用途はあまりない．将来的な使用用途としては，ホスト間通信の競合等を確認出来るようにしたいと考えている．それには，バンド幅・レイテンシ等の通信環境情報や他のユーザによる通信負荷等を知る必要があるので，これらを実現出来る手法を考えなければならない．

5.3 情報収得構成

5.3.1 Adapter

実行時情報収得方法として，Adapter[8] を採用した．これは，当研究室で提案されたMegaScript の拡張機能である．MegaScript オブジェクトであるタスクやストリーム等に付加する事で，様々なイベントをトリガーとして処理を行える．

今回の手法では，一定時間に処理を行う Adapter とメッセージ出力をトリガーとして処理を行う Adapter を用いた．他にも様々な Adapter が存在するので，収得する情報増やす際も Adapter を利用できると考えられる．Adapter の一覧を表 5.1 に示す．

収得対象オブジェクト	収得タイミング	収得可能情報
ホスト	一定時間	ホスト CPU 使用率
		タスク CPU 使用率
タスク	メッセージ入力	受信回数
		受信データ量 (合計・平均)
	メッセージ出力	送信回数
		送信データ量 (合計・平均)
	指定ファイル OPEN	タスク書き込み状態
	終了時	タスク完了

表 5.1: 収得可能情報一覧

5.3.2 ログファイル

Adaoter が収集した情報は一度ファイルに書き込んで，そこから情報を読み込む形式にした．このファイルをログファイルと呼び，提案手法で用いる形式を表 5.2 に示す．

ログファイル内では，”キーワード:対象:収得情報”のようにそれぞれの情報を”:”で区切って記述されている．収得情報が複数ある場合も”:”で区切って記述されている．

キーワード	対象	収得情報		
0	タスク名	CPU 使用率 (整数)		
1	ホスト名	CPU 使用率 (整数)		
2	タスク名	送信回数 (整数)	平均データ量 (整数)	最大データ量 (整数)

表 5.2: ログファイル形式

キーワードの”0”はタスクの CPU 使用率である事を示し，”1”はホ

ストのCPU 使用率である事を示し，”2”はタスクの通信情報である事を示している．キーワードは，統合開発環境側でその情報が何の情報を表しているのかを知るために記述している．タスク名とは，そのタスクの実体である実行ファイル名である．現在では，タスクの変数名は実行ファイル名で無ければ情報を表示する事が出来ない．

今回の手法で用いる情報は3つだが，将来異なる情報を用いる際もこの形式で記述すれば変更箇所が少ないという利点がある．

5.3.3　情報読み取り

情報はログファイルから読み込むのだが，いつ書き込みが行われログファイルが更新されるか統合開発環境側からは認識出来ない．そこで，読み込み関数をスレッド化し，実行中は常にそのファイルを監視する事にした．これにより常にログファイル格納された最新の実行時情報を収得可能となる．

5.4　色の管理

使用できる色はColorManagerクラスによって管理されている．タスク等の色を変更する場合，タスク自身が持つ色用の変数を用いてColorManagerを参照して色を決定している．そのため，repaint部分を修正する事無く色

の変更が可能である．また，色の種類を増やしたい場合はColorManager を変更するだけで良いので，開発効率の向上を図れる．

実行ホスト視覚化に関しては，その機能を解除した時に元の色に戻らなければならない．そのため，タスク等には変更前の色を記憶する変数も用意した．

5.5 タスクCPU使用率視覚化構成

タスクの表示がGEF を用いて描写されているため，グラフが描かれる描写範囲 (レイヤー) はタスク毎に異なる．そのため，タスクに関連するグラフを描写する場合はそのタスクが描写されているレイヤーに描写する事が自然だと考えられる．そこで，実行中の情報を表示するためにプログラムが実行された時のみタスク毎のレイヤーの範囲を拡張する事にした．実行中のみ範囲を拡張する事で，無駄なスペースの確保を防ぎ，タスクの移動等の操作が行いづらくなってしまうのを軽減させている．

5.6 ホストCPU使用率視覚化構成

ホストCPU 使用率の描写には，GEF に付属する様々な形状を描画するためのプラグインDraw2D を利用した．現在，全てのホスト情報を同一スレッドで表示している．全てのホストが同様に仕事をする訳ではな

いので，ホスト毎にスレッドを用意する方法も考えた．比較を行い易くするため同一キャンパスにグラフを描写する事を考えていたので，別々のスレッドでホストを表示するにはキャンパスを共有しなければならなくなる．その方法考える余裕が無かった事から，今回は同一スレッドで表示を行っている．

5.7 実行ホスト視覚化構成

同一ホストの識別は，タスクをホスト別に異なる色で表示することで実現させている．プログラムが実行されると，ファイルにスケジュール結果が書き込まれる事を想定しているが，現在は予め自分でファイルを用意しなければならない．そのファイルから情報を読み取り各タスクに割当先のホスト名情報を記憶させる．この情報を用いて，同じ名前のホストを持つタスクは同色で表示すれば結果的にホスト別に表示出来ることになる．さらに，処理を速くするためホスト名の他に hostId という変数を持たせた．この変数を用いて ColorManager を参照すれば，比較を行う事無くホスト別に識別する事が可能になり処理時間を短縮できる．

しかし，色で識別するには限界がある．並列言語 MegaScript はメガスケール規模のプログラムを想定しているので，色で識別するには困難で

あると考えられる．そういった理由も含め，タスクネットワークを階層的に表現する方法を考えていかなければならない．

5.8 ピックアップ型実行ホスト視覚化構成

上記であげた実行ホスト視覚化の問題点を補助するために実装した．これは，Java に用意されているイベントクラス MouseEvent を利用している．使用しているメソッドは mouseEntered,mouseExited である．mouseEntered は，範囲外から範囲内にマウスカーソルが入るとイベントが発生となる．反対に mouseExited は，範囲内から範囲外にマウスカーソルが出るとイベント発生になる．これらを用いる事で，ユーザが注目しているタスクが判別できる．ここでの範囲とはタスクの描写範囲 (レイヤー) の事である．

まず，マウスカーソルがタスクのレイヤーに入ったら，そのタスクと同じホストに割り当てられているタスクを検出する．比較には変数 hostGroup を用いる．変数 hostName を用いても検出できるが，文字列比較よりも整数比較の方が速い．検出したタスクの色用変数を変更する．これにより，マウスカーソルが入ったタスクと同じホストの割り当てられているタスクのみ色が変更される．

次に，マウスカーソルがタスクのレイヤーから出た場合，上記と同様に同じホストに割り当てられているホストを検出し，色用変数を変更前の値に戻す．

全体を識別する方法が同時に動作しないように，フラグを用意して片方の実行ホスト視覚化表現のみ動作するように実装している．

6 性能評価

6.1 評価方法

実行時情報視覚化では情報をリアルタイムで表示する事が重要と言える．そこで，タスクの再描写に掛かる時間を計測し，リアルタイムでの表示が行えているか評価を行った．Java の標準関数である currentTimeMillis() を利用した．この関数を用いて再描写関数の前後で時間の収得を行い差分を取る事で処理時間の計測を行った．currentTimeMillis 関数は，ミリ秒単位の正確さを保障していないので十ミリ秒単位で評価を行えるようにしなければならない．そのため，タスク数を 50000 から開始した．情報の更新間隔は約 500ms なので，500ms 内に収まっていればリアルタイムに表示を行えていると言える．

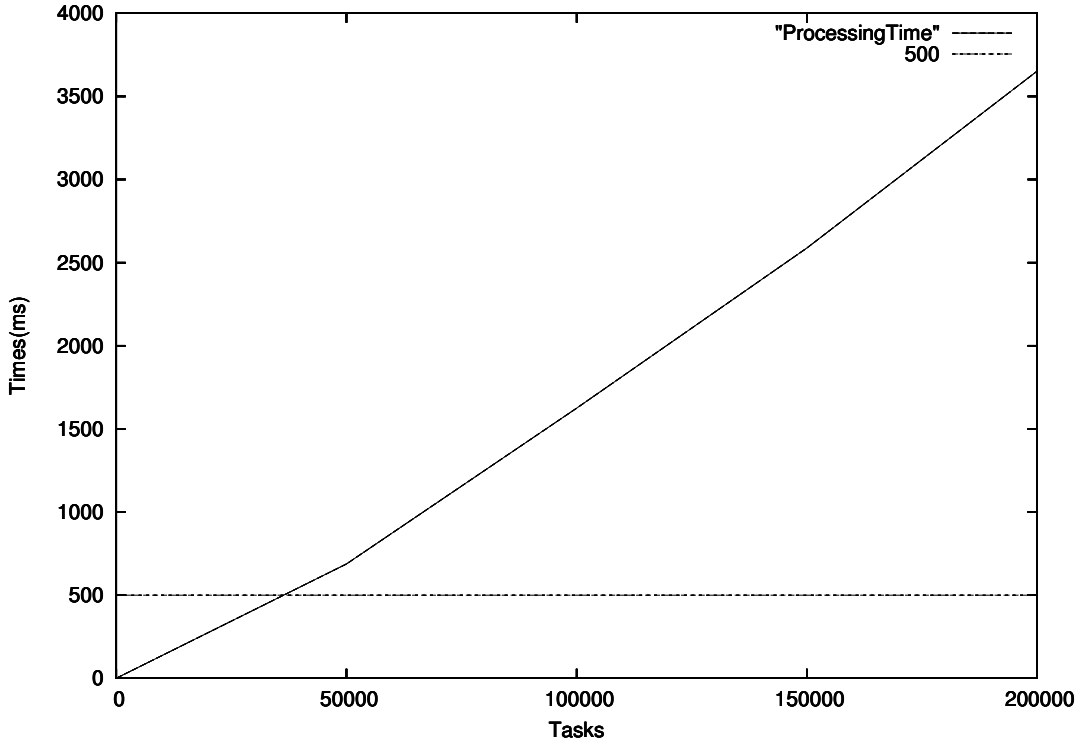

図 6.5: 評価結果

6.2 評価結果

図 6.5 で示した評価結果から，3 万タスク程度の規模ならリアルタイムに表示を行える事が解った．実際，大規模になると視覚的に見やすい大きさを保つことが出来ないため 3 万タスク程度の描写が可能なら問題ないと言える．

7　おわりに

並列プログラム作成等に有効な情報をグラフィカルに表示する事で，ユーザが情報を容易に認識しやすくなり効率的な利用が可能となった．また描写時間も更新間隔内で収まっているため，リアルタイムで情報の表示が行えていると言える．

しかし，MegaScript は大規模なプログラムを想定しているのでタスクネットワーク全体を表示するのは困難であると考えられる．ネットワークを階層的に表示を行い段階的に注目箇所を絞る手法が必要であると考えられる．

謝辞

本研究を行うにあたり，御指導，御助言頂きました大野和彦講師，並びに多くの助言を頂きました近藤利夫教授，佐々木敬泰助教に深く感謝致します．また，様々な局面にてお世話になりました研究室の皆様にも心より感謝いたします．

参考文献

[1] 大塚保紀，深野佑公，他：タスク並列スクリプト言語 MegaScript の構想，先進的計算基盤システムシンポジウム SACSIS2003，pp.73-76，2003

[2] 谷口和也，松本真樹，他：タスク並列スクリプト言語のビジュアル開発環境の構築，情報処理学会研究報告書 Vol.2007No.80，pp91-96，2007

[3] 宮本信二：Eclipse3.1 完全攻略，ソフトバンククリエイティブ，2005

[4] Eclipse Graphical Editing Framework (GEF)，http://www.eclipse.org/gef/

[5] 上嶋明，小畑正貴，他：Omni OpenMP コンパイラ用並列プログラム可視化ツール，先進的計算基盤システムシンポジウム SACSIS2005，pp53-60，2005

[6] 羽田昌代，笹倉万里子，他：NaraView を利用した実アプリケーションの並列化事例，HPCVol2000No.57，pp39-44，2000

[7] 高田哲司，小池英樹：VisuLinda:並列言語 Linda のプログラムの実行状態の３次元視覚化，日本ソフトウェア科学会 WISS'94，pp215-223，1994

[8] 阪口祐輔，大野和彦，他：タスク並列スクリプト言語処理系におけるユーザレベル機能拡張機構，情報処理論文誌：コンピューティングシステム，pp296-307,2006