MegaScript 処理系におけるスケジューラライブラリの設計

鈴木康平† 松本真樹† 大野和彦†
佐々木敬泰† 近藤利夫†

1. はじめに

近年，広域ネットワーク上に分散しているクラスタ群を利用した並列処理に注目が集まっている．そこで，我々は大規模な並列処理を容易に実現するためのタスク並列スクリプト言語 MegaScript[1] を開発している．本研究では MegaScript 上にスケジューラを容易に実装できるスケジューラライブラリを設計した．

2. 背景

MegaScript は，タスクの依存関係に従って処理を進めるワークフロー型の並列処理を実現するスクリプト言語である．大規模なワークフロー型並列処理で高い並列性を得るためには，タスクスケジューリングが重要になる．現在，スケジューリングアルゴリズムには膨大な種類がある[2] が，任意の並列アプリケーションに対し，常に最適解を出すような万能なものはない．そこで，盛んに新しいアルゴリズムの研究がされている．このため，ユーザは自分が実行したい並列アプリケーションに適したスケジューラを実装しなければならない場合がある．しかし，その場合に処理系の内部仕様を考慮しなくてはならないのでは，実装の負担が大きくなってしまう．さらに，スケジューラを複数実装する場合，アルゴリズムに共通する処理を何度も記述しなくてはならないため，非常に煩雑である．

3. スケジューラライブラリ

3.1 設計

スケジューラライブラリを設計するにあたり，次の 4 点が問題になる．(1) スケジューラを実装する際，MegaScript 処理系の内部仕様を考慮しなければならない．(2) 静的スケジューリングはユーザがスケジュール処理を呼び出すことができ，1 つのメソッドとして書ける．そして，そこから呼び出すメソッドにアルゴリズムの共通部分が多く，スケジューラを実装するたびにその共通する部分を書かなくてはならない．(3) アルゴリズムにより必要なタスクの情報が異なるため，個々のスケジューラがそれを保持するメンバ変数をタスクに追加したい場合がある．しかし，スケジューラは MegaScript 処理系の使用するタスクオブジェクトを扱うため，直接タスクオブジェクトにメンバ変数を追加すると，MegaScript 処理系拡張時に競合が発生する可能性がある．(4) 動的スケジューリングは．実行時のイベント発生に伴ってスケジュール処理を呼び出すため，1 つのメソッドとして書けない．

(1), (2) については，MegaScript がオブジェクト指向言語であることからクラス継承を利用することを考える．MegaScript の内部仕様に関わる部分やアルゴリズムに共通する部分を持つスーパークラスを用意し，スケジューラ実装者はそのクラスを継承してスケジューラを実装する．(3) については，タスクを間接的に拡張することを考えれば解決できる．(4) に対してはイベントドリブン方式を用いる．MegaScript 処理系は，タスクの終了やホストの性能変動等が発生した際，その旨をメッセージとしてスケジューラに送る．本ライブラリでは，スケジューラはそのメッセージをイベントトリガーとして，メッセージの種類によって処理を分岐させる．そして，各分岐先で予め用意したメソッドを呼び出す．スケジューラ実装者はそのメソッドを自作したサブクラス内でオーバライドすることで動的スケジューラを実装する．ただし，動的スケジューラに関する部分は現在構想段階であり具体的な設計は行っていない．

3.2 実装

提案するスケジューラライブラリのクラス図を**図 1** に示す．点線より上が本ライブラリのクラス図であり，本論文では独立型スケジューリングと DAG スケジューリングの 2 つのクラスを実装した．以下に示す 2 点に注意し，このようなクラス設計を行った．

- アルゴリズムの共通部分の階層性を利用する．
- スケジューラに対して，拡張性や可読性を高める．

† 三重大学大学院工学研究科情報工学専攻

ⓒ 2012 Information Processing Society of Japan

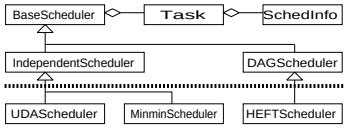

図 1 設計したスケジューラライブラリのクラス図

まず，独立型スケジューリングは，(1) タスク群を取得，(2) タスクのコスト値を取得，(3) タスクの計算機への割当という機能を持つ．DAG スケジューリングは，(1) タスク群を取得，(2) タスクのコスト値及び DAG 構造を取得，(3) タスクの計算機への割当という機能を持つ．(1) に関してはこの 2 つの分類だけでなく全てのスケジューラに共通する部分であるので BaseScheduler クラスに記述する．(2) に関してはそれぞれのアルゴリズムに共通する部分であるので，IndependentScheduler クラス，DAGScheduler クラスに記述する．(3) は，スケジューラ実装者がクラスを継承して記述する部分であるため，本ライブラリでは空メソッドとして BaseScheduler クラスに用意する．ここで，DAG スケジューリングの一種であるリストスケジューリングのスーパークラスを追加する場合を考える．この場合，DAGScheduler クラスを継承することで容易に追加できる．その上可読性も高い．

さらに，空クラス SchedInfo クラスを用意する．1 つの Task インスタンスに対し，1 つの SchedInfo インスタンスが自動的に生成される．Ruby には既に宣言されたクラスを拡張できるという特性があるため，スケジューラ実装者はそれぞれのスケジューラクラスに必要なメンバ変数やメソッドをスケジューラ実装時にこのクラスに持たせる．これにより，MegaScript 処理系と競合する心配がなくなる．

4. 評　価

設計したスケジューラライブラリの有用性を確認するために，本ライブラリを使用したスケジューラと不使用のスケジューラを実装し評価を行った．スケジューリングアルゴリズムには UDA[2)]，Minmin[2)]，HEFT[2)] を用いた．評価環境は Xeon X3330 2.66GHz，メモリ 2GB を搭載した計算機を使用した．評価には遺伝子解析の分野で利用される Epigenomics workflow を用いた．最初に本ライブラリを使用したスケジューラと不使用のスケジューラの記述量を比較した（**表 1**）．また，本ライブラリによって記述量が減少してもスケジューリング時間が大幅に増加した場合，有用であるとはいえない．そこで，比較対象のスケジューラのスケジューリング時間を計測した結果を**表 2** に示す．

表 1 スケジューラの記述量 (行数)

	UDA	Minmin	HEFT
不使用	82	106	176
不使用 (依存部分)	19	21	51
使用	44	72	121
使用 (依存部分)	0	0	0

依存部分とは MegaScript 処理系の内部仕様に依存する箇所

表 2 スケジューリング時間 (秒)

	UDA		Minmin		HEFT	
タスク数	不使用	使用	不使用	使用	不使用	使用
1004	2.01	2.01	5.88	6.60	0.17	0.18
5004	48.15	48.50	126.33	164.15	2.58	2.63
10004	197.85	198.04	518.48	663.85	9.69	9.78

表 1 から，行数は 30〜47 %削減されている．これは図 1 のように，クラスを継承することにより実装の大半がスケジュール部分のみでよくなるためである．そして，不使用の場合は処理系内部への依存部分が全体の 20〜30 %を占めていて，その部分の実装は負担が大きい．本ライブラリを使用することで，その部分を実装する必要がなくなるため，負担が軽減される．さらに，MegaScript の仕様変更の際にスケジューラ実装者は作成したスケジューラを書き換える必要がなくなる．スケジューリング時間においては，UDA，HEFT では数%の増加に止まっているが，Minmin に関しては 30 %程度増加していることがわかる．これは，Minmin スケジューラがスーパークラスのメソッドを呼び出す回数が多く，そのオーバーヘッドが大きいためである．

5. おわりに

本研究ではスケジューラの実装を容易にするためのスケジューラライブラリを設計し評価を行った．本ライブラリを用いることでスケジューラの記述量の削減を達成できた．またスケジューリング時間においては，UDA や HEFT で数%の増加に抑えることができた．しかし Minmin は 30 %程度増加してしまった．今後の課題として，動的スケジューラに関する部分を設計・実装することが挙げられる．またスケジューリング時間の増加を抑えていく必要がある．

参 考 文 献

1) 大塚保紀　他: タスク並列スクリプト言語 megascript の構想, In SACSIS2003, pp. 73–76 (2003).
2) 須田礼仁: ヘテロ並列計算環境のための タスクスケジューリング手法のサーベイ, 情報処理学会論文誌:コンピューティングシステム, Vol. 47, No. SIG 18(ACS 16), pp. 92–114 (2006).

ⓒ 2012 Information Processing Society of Japan