卒業論文

題目

タスク並列スクリプト言語MegaScript処理系におけるホストの管理方法の改良

指導教員

大野 和彦 講師

2014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

田中　宏明(410831)

内容梗概

近年，遺伝子解析や環境シミュレーションなどの様々な分野で大規模計算が求められている．その一方で，単一プロセッサの性能向上は頭打ちの状態にあり，広域分散環境での並列処理に注目が集まっている．
しかし，大規模な並列処理のプログラミングを行うことは専門的な知識や高いプログラミング力が必要なため困難である．そこで，我々は大規模な環境下において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 を開発している．本研究では MegaScript におけるホストの管理方法の改良を行った．

Abstract

In recent years, large-scale calculations are required in various fields such as gene analysis and environmental simulation. On the other hand, performance improvement of a single processor is in a state of ceiling, and attention is being paid to parallel processing in a wide area distributed environment.
However, it is difficult to program large scale parallel processing because it requires expert knowledge and high programming ability. Therefore, we are developing a task parallel script language MegaScript to easily realize parallel processing under a large scale environment. In this research, we improved the management method of host in MegaScript.

目 次

図 目 次

表 目 次

1　はじめに

近年，遺伝子解析や環境シミュレーションなどの様々な分野で大規模計算が求められている．その一方で，単一プロセッサの性能向上は頭打ちの状態にあり，広域分散環境での並列処理に注目が集まっている．

しかし，大規模な並列処理のプログラミングを行うことは専門的な知識や高いプログラミング力が必要なため困難である．そこで，我々は大規模な環境下において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 を開発している．本研究では MegaScript におけるホストの管理方法の改良を行った．

2　背景

2.1　タスク並列スクリプト言語 MegaScript

MegaScript は，大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使用し，タスク間の通信をストリームと呼ばれる論理的な通信路を通して行うことでワークフロー型の並列処理を実現する．

2.1.1　階層化動作モデル

現状の MegaScript の動作モデルは fig. 2.1 のようになる．動作モデルには管理・制御等を行うマスターホスト，及びサブマスターホストと，タスクの処理等を行うスレーブホストから成る階層型のマスタースレーブシステムが採用されている．ホストが階層的に管理されているため，マスターホスト１台で全ホストに制御メッセージを送信する必要がなく，サブマスターホストでそれぞれ管理しているホストにメッセージを送信すればよい．従って，より多くの計算資源を扱うことができ、大規模並列処理の高い実行効率を実現できる．本稿においては議論を簡単にするため，fig.2.1 のような 2 階層のモデルを前提とする．

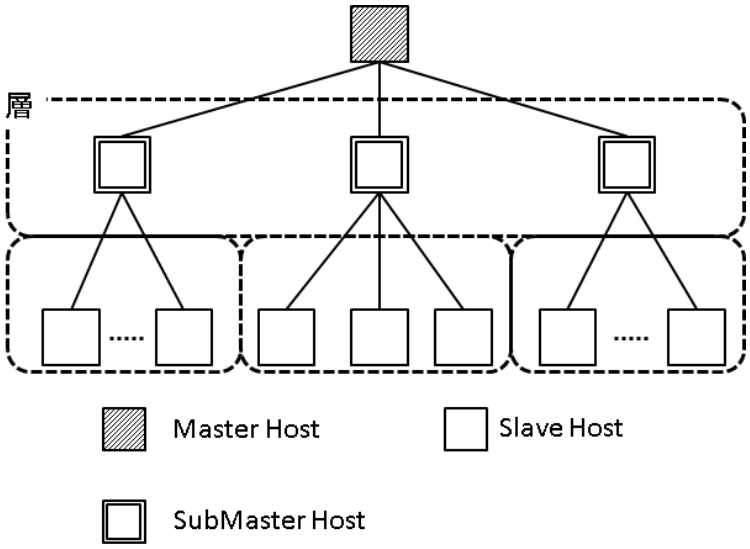

fig . 2.1: 階層化動作モデル

2.1.2 Host クラス

MegaScript ではホストを表すクラスとして Host クラスが実装されている．Host クラスはメンバとして名前や ID，マスターやスレーブなどの役割，ホストの性能，スケジューラにより割り当てられたタスク・ストリームを格納するキューを持っており，スケジューラは生成されたオブジェクトを利用しタスクの割り振りを行う．ホストの性能とは，ホストの計算能力をスカラー値で表したものである．現状マスターホストは全てのホストに対応するオブジェクトを持っており，全てのスレーブホストの情報を使う事で精度の高いスケジューリングを可能としている．しかしながら，環境が大規模になるに従い，マスターホストのメモリが足

らなくなり破綻する，スレーブの情報収集に時間が掛かるという問題がある．

3　提案手法

2.3 節で述べた問題を解決するためにマスターホストの管理するホストオブジェクトをサブマスターホストだけのものに絞ることを考えた．これによりマスターホストはすべてのスレーブホストを管理することなくスケジューリングを行う．よってマスターホストのメモリ不足は解決する．例えば，マスターホストの下に 1,000 台のサブマスターホスト，そして，それぞれのサブマスターホストの下に 1,000 台ずつのスレーブホストがある環境を想定する．この時，従来のままではマスターホストは 1,000,000 台のスレーブホストの情報を管理することになるが，提案手法ではマスターホストは直下の 1,000 台のホストを管理すればよい．

しかし，スレーブホストの情報を削除したことでメモリは軽減できるが，ホストの性能を考慮したスケジューリングが不可能となり，スケジューリングの精度が低下してしまう．これは，マスターホストが持っているのはサブマスターホストの性能情報で，実際にタスクを実行するスレーブホストの性能情報は持っていないからである．そこで，サブマスターホストの性能を直下のスレーブホストの性能の合計として与え，マスターホストへ伝えることを考えた．これにより，低下したスケジューリングの精度を改善する．

サブマスターホストへ直下のスレーブホストの性能の和を与えることにより，マスターホストは各サブマスターホストの下のスレーブホストがどれだけ性能を持っているかを把握できる．マスターホストはサブマスターホストを子の合計性能分を持った高性能なスレーブホストと見なし，タスクを割り当てる．その後，サブマスターホストは自身の子ホストへタスクを割り当てる．

しかし，直下に性能 P_i のスレーブホストを n 台持ったホストと，性能 $\sum_{i=1}^{n} P_i$ のホストでは割り振るべきタスクは異なる．そこで，直下のスレーブホストの台数を性能の和と別にサブマスターホストへ持たせる．台数を持たせることで，合計性能と合わせて直下のスレーブホストの平均性能を算出でき，マスターホストはサブマスターホストが直下にどのくらいの性能のホストを何台持っているか，というところまで考慮し，スケジューリングを行うことができる．これらによりスケジューリング結果をより最適なものへ近づけられる．

fig.3.2 のように，サブマスターホスト $M_1 ... M_m$ の m 台，サブマスターホスト M_i の子ホストにスレーブホストが $H_{i_1} ... H_{i_{ni}}$ の n_i 台あるとする．それぞれのスレーブホストの計算能力が S_{i_j} のとき，スレーブホストに与えられる性能はそれぞれ $P_{i_j} = S_{i_j}$，サブマスター M_i に与えられる性能

は $P_i : (\sum_{j=1}^{n_i} S_{i_j}, n_i)$ で表される.

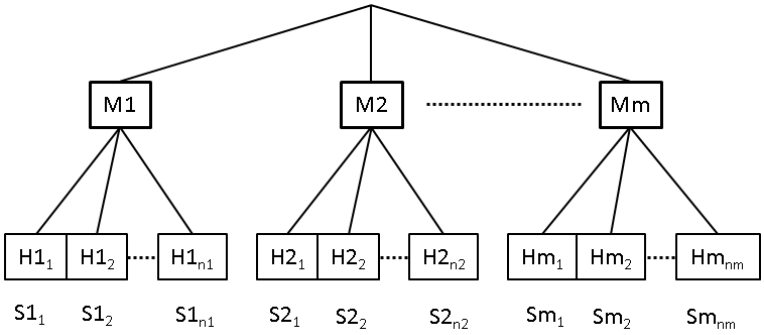

fig . 3.2: 想定する動作モデルの一般系

4　評価

今回の実装により，スケジューリングの精度の低下をどれだけ抑えられたか確認するため，提案手法でのスケジューリング結果と最適に近い解と考えられるフラットな環境でのスケジューリング結果，及びスレーブの情報がない状態でのスケジューリングの結果を求め，ワークフロー全体の実行時間であるメイクスパンの評価をそれぞれ行った．スケジューリングアルゴリズムにはDAGスケジューリングの一種であるHEFTを用いた．ただし，サブマスターホストのスケジューラにはタスクをコストに従い均等に割り振るアルゴリズムを用いた．評価に用いる実行環境としてfig.4.3のような環境を仮想的に用意した．サブマスターホスト5台の下に16/32/64/128からランダムに選んだ台数のクラスタを構成した，スレーブホストの総数は304台であり，クラスタ内の性能は均質である．また，評価に用いたワークフローは遺伝子解析分野で用いられる Epigenomics workflow と呼ばれる実ワークフローであり，その構造を fig.4.4 に示す．ワークフローの個々のタスクとして，1,000〜10,000のランダムなコストを持ったタスクを仮想的に用意した．これらの環境でスケジューリングを実行した．その結果について，タスクのコストをc，ホストの性能をpとすると，タスクの実行にc/p秒かかり，依存タスクが違うホストにあ

る場合はタスク間のデータ転送に 0.1 秒かかるものとし，シミュレートを行いメイクスパンを求めた．評価は全て fig.4.4 の太枠で囲われている部分のタスク数を 1,000, 5,000, 10,000 と変化させて行う．結果を Table.4.5 に示す．

タスク数 1,000 の結果において，フラットな環境でのスケジューリング結果に比べ，スレーブの情報なしのメイクスパンが約 177%増加したのに対し，提案手法ではメイクスパンの増加を 53%に抑えられている．同様にタスク数 5,000 ではそれぞれ 235%,11%の増加, タスク数 10,000 ではそれぞれ 246%,3%の増加という結果となった．提案手法はフラットの環境に比べメイクスパンの増加が認められるが，タスクが多くなるに従い数%の増加に留まっている．また，スレーブの情報なしのものが 200%程度増加していることと比較すると，提案手法はスケジューリングの精度の低下を十分抑えられたと言える．

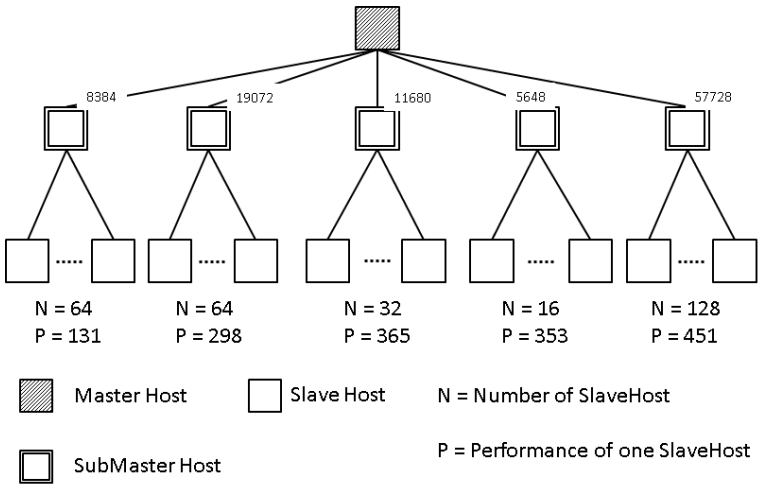

fig . 4.3: 評価に用いたホストの構成

5　おわりに

本研究では，階層動作モデルにおいてマスターホストからスレーブホストの情報を削除し，それによるスケジューリングの精度の低下を抑えるための手法の提案，評価を行った．その結果，大規模な環境においてマスターホストの深刻なメモリ不足による破綻を解決できた．また，スケジューリング精度の低下も無視できる範囲に抑えることができた．

今後の課題として，ホストの性能変動に対応した動的な性能管理などが挙げられる．また，本研究ではタスク数が 10,004 の場合までの評価しか行っていないが，実際ではタスク数が 10 万を越える場合も少なくない．

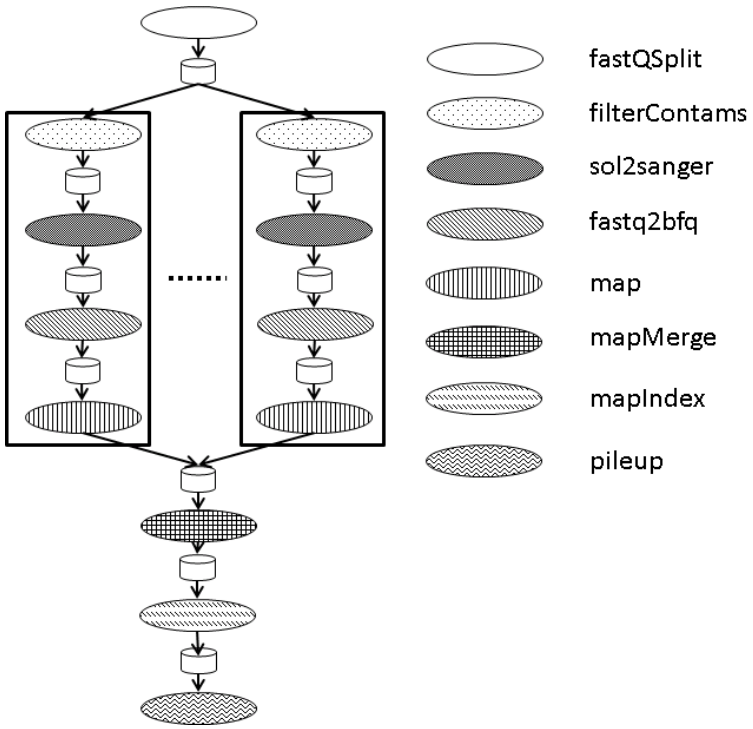

fig . 4.4: Epigenomics workflow

従って，その場合のスケジューリング精度を検証する必要もある．

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山田俊行講師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.

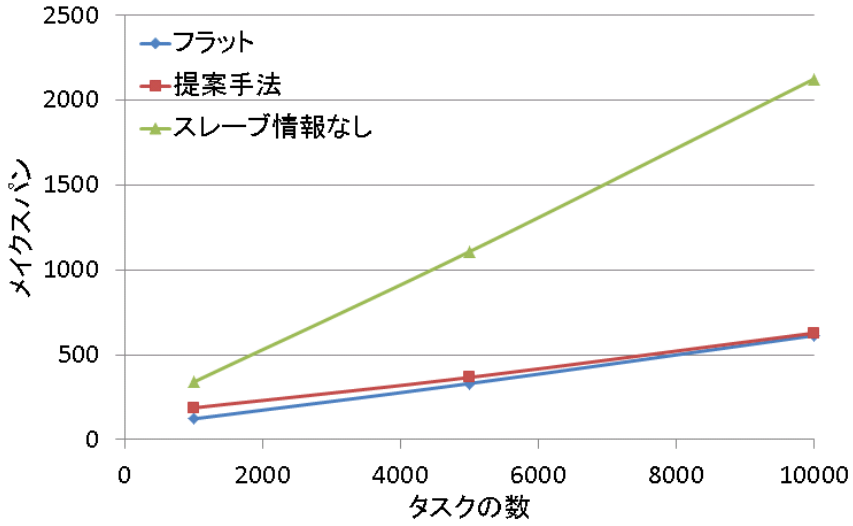

fig . 4.5: スケジューリング長

参考文献

[1] 大塚保紀，野佑公，西里一史，大野和彦，and 中島浩．タスク並列スクリプト言語 megascript の構想．In 先進的計算基盤システムシンポジウム SACSIS，pages73-76，2003

A　プログラムリスト

B　評価用データ