卒業論文

題目

階層型MegaScript処理系における通信機構の設計及び実装

指導教員

大野 和彦 講師

2011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

仲　貴幸 (407830)

内容梗概

近年，遺伝子解析や気象／災害のシミュレータなどでは複雑な数値計算を要するため，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ている．しかし，メガスケール規模の並列性を持つプログラムを記述するのは非常に困難であり，並列処理の記述に対する専門的な知識や，高い能力がプログラマに求められてしまう．そこで，メガスケールコンピューティング向け言語としてタスク並列スクリプト言語 MegaScript が開発されている．

現在 MegaScript の動作モデルにはタスクの処理等を行うスレイブホストと，管理・制御等を行うマスターホストの 2 種類から成る集中管理型が採用されている．しかし，この動作モデルでは，スレイブホストの数が膨大なとき，マスターホストはスレイブホスト全てを管理，制御する必要があり，また各スレイブホストからの応答を一台のホストで処理しなければならないという問題がある．そこで，マスターホストとは別にホストを管理するサブマスターホストを導入し，MegaScript の動作モデルを階層化する．これにより，通信回数が多くなりオーバヘッドが増加してしまうが，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．しかし，階層化動作モデルに基づく MegaScript 処理系全体の実装は作業量が多いため，まず通信機構の設計・実装し評価を行った．

実装した通信機構に対し，MegaScript で実際に行われるタスク間通信を疑似的に再現し，各タスクが 80Byte の文字列を 10 万回送信したときの通信時間及び LAN 通信，WAN 通信の回数を計測した．その結果，1 対 N の場合，WAN 通信回数を減らすことができた．しかし，新しい動作モデルは従来の動作モデルに比べ通信時間，LAN 通信回数が増加した．

Abstract

In recent years, many research fields with complex numeric calculations, such as gene analysis or simulator of weather/disaster, requires megascale computing that uses 1 million scale processors. However, it is very difficult to write programs with mega-scale parallelism. The programmer is also required deep knowledge and high ability for parallel programming. Therefore, a task-parallel script language MegaScript is developed for the mega-scale computing.

The current implementation of MegaScript is based on centralized control system which is consisted of two host type. One is a slave host, it carries on tasks and so on. The other is a mastar host, it controls and manages slave hosts. However, if the number of slave host expand, the load of the slave host control focus on a master host. So the present study establish a the hierarchical communication mechanism. In this way, it is able to contral more counting machine efficiently.

目 次

図 目 次

表 目 次

1　はじめに

1.1　背景

近年，遺伝子解析や環境シミュレーションなど複雑な物理計算を要する分野において，Pflops 以上の計算性能が求められている．Pflops を越える性能を得るためには，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．

しかし，既存の並列プログラミングモデルでメガスケール規模のプログラミングを行うことは困難である．そこで，我々はメガスケール規模の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[1] を開発している．

1.2　研究目的

現状の MegaScript の動作モデルにはタスクの処理等を行うスレイブホストと，管理・制御等を行うマスターホストの 2 種類から成る集中管理型が採用されている．この動作モデルでは，スレイブホストの数が膨大なとき，マスターホストはそのスレイブホスト全てに制御メッセージを送信する必要があり，また各スレイブホストからの返信メッセージを一台のホストで処理しなければならないという問題がある．

そこで，マスターホストとは別にホストを管理するサブマスターホストを導入し，MegaScript 処理系の動作モデルを階層化する．この動作モデルは，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の送信をすればよい．従って，通信回数が多くなりオーバヘッドが増加してしまうが，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．

1.3 本文構成

本文の構成は以下のようになっている．第 2 章でまず MegaScript 処理系の概要について述べ，第 3 章にサブマスターホストの実装，第 4 章に性能の評価を行い，最後に第 5 章にてまとめを行う．

2　概要

2.1　タスク並列スクリプト言語 MegaScript

2.1.1　基本設計

MegaScript は，100 万プロセッサ規模の並列処理を目的とするプログラミング言語である．MegaScript では，逐次プログラムや並列プログラムをタスクとして扱い，複数のタスクを制御して並列実行を行う．各タスクは独立性の高い処理モジュールであり，「ストリーム」と呼ばれる通信路を用いることによってタスク間のデータ送受信を行うことができる．MegaScript はメガスケール規模でのプログラミングを実現可能とし，広域に分散し複数の並列計算機を含むヘテロな環境下での動作を目指している．

2.1.2　タスク

MegaScript では，計算の主体となる外部プログラムのプロセスをタスクと呼ぶ．タスクとなるプログラムはユーザが任意の言語で作成することが可能であり，種々のタスクプログラムをそれぞれ別の言語で作成しても構わない．並列動作させる逐次／並列プログラムを MegaScript プログラム中に定義し，各ホストにおいてタスクの生成を行う．MegaScript では，

扱うホスト数が膨大であると考え，タスクを生成するホストはMegaScript スケジューラを呼び出すことにより自動的に決定される．

2.1.3 タスク間通信

MegaScript では，「ストリーム」と呼ばれる，各タスクの標準入出力を接続する通信路を用いてタスク間通信を行っている．(fig .2.1)

fig . 2.1: タスク間通信

ストリームの入出力端にはそれぞれ複数のタスクを接続することができ，これにより多対多通信が可能となる．入力端に複数のタスクが接続されている場合，各タスクの出力が行単位で結合されてストリームに流れる．また，出力端に複数のタスクが接続されている場合，出力端に接続されている全てのタスクにストリームからデータが送信される．

2.1.4 プログラミング

MegaScript のプログラミングモデルは，計算の主体となる外部プログラムをタスクとして定義し，これらの標準入出力間をストリームで接続したプロセスネットワークモデルである．

MegaScript はオブジェクト指向である Ruby を拡張した言語であるため，必要に応じて複雑な処理を書くことができる．MegaScript プログラム上では，タスクやストリームはそれぞれタスクオブジェクト，ストリームオブジェクトとして表現される (fig . 2.2)．これらのオブジェクトは物理的なタスク (プロセス) やストリーム (パイプやスレッド，プロセッサ間通信で実現された通信路) と一対一対応し，タスクやストリームの操作は対応するオブジェクトを使って行う．

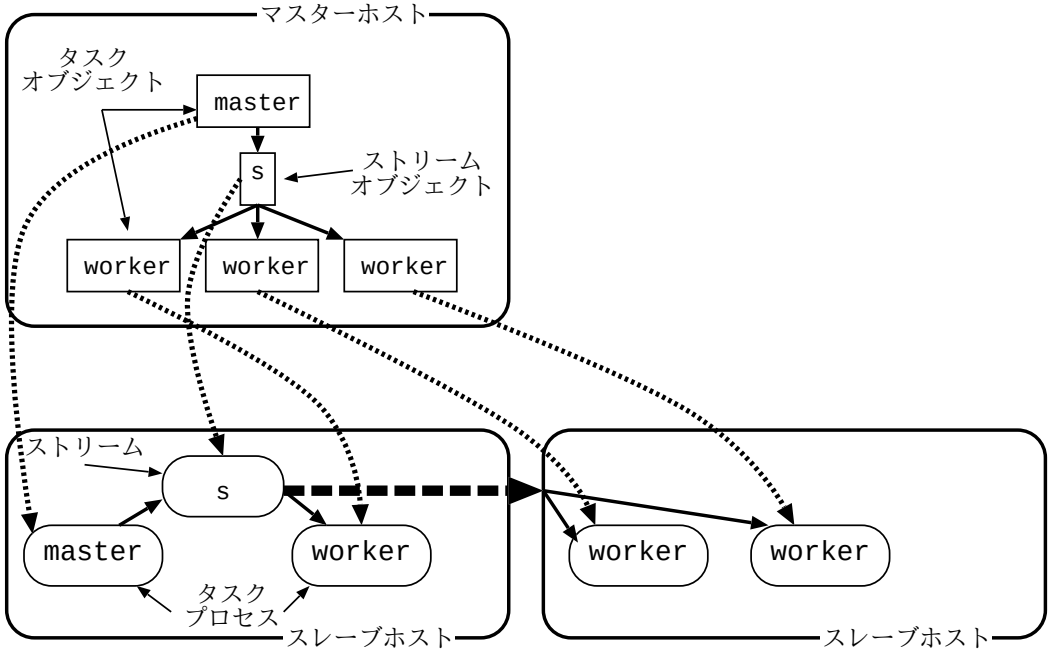

fig . 2.2: ストリーム接続

また，MegaScript では API 関数によるプログラミングとクラスライブラリによるプログラミングの二つの手法が提供される．API 関数によるプ

ログラミングは，MegaScript のもっとも低レベルな機能を提供する API 関数を用いることで，低水準ではあるが非常に柔軟で強力な並列処理の記述が可能である．また，並列処理の知識に乏しいエンドユーザ向けとしてタスクネットワーク構築の支援を行う階層化されたライブラリ構造 (fig . 2.3) が用意されている．このクラスライブラリを用いたプログラミングでは，ライブラリとして提供される基本的なタスクネットワーク構造やユーティリティ的な機能を用いることで，高度な並列処理を簡潔に書くことが出来る．

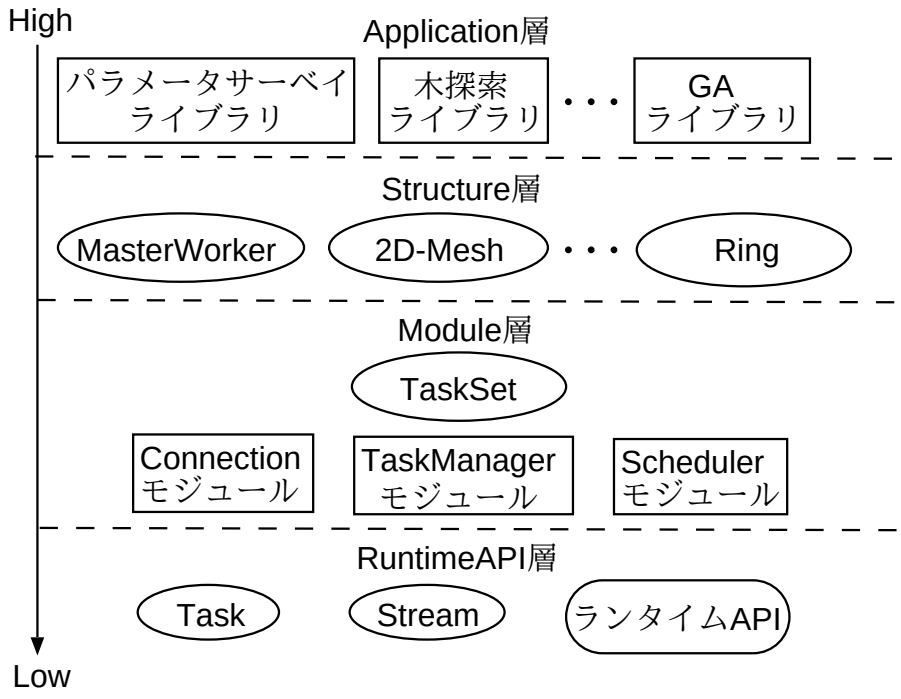

fig . 2.3: ライブラリの階層構造

2.2 集中管理型動作モデル

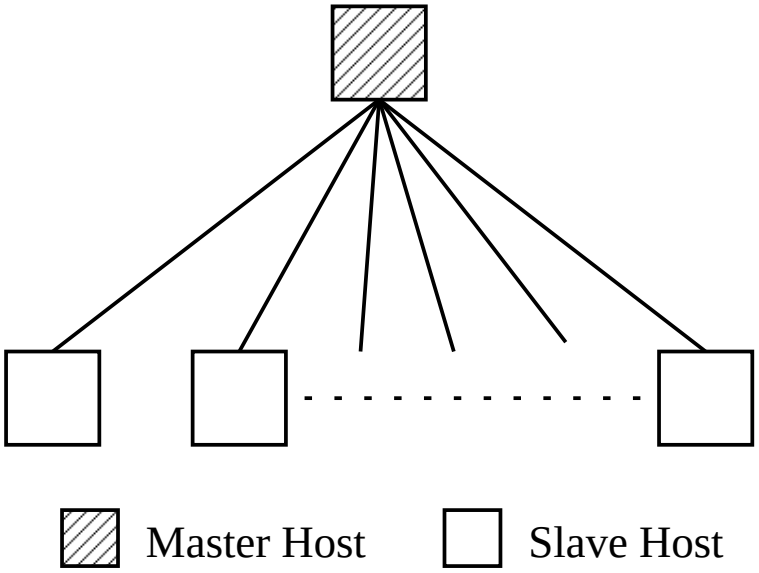

fig . 2.4: 集中管理型動作モデル

現状の MegaScript の動作モデルは fig. 2.4 のようになる．動作モデルにはタスクの処理等を行うスレイブホストと，管理・制御等を行うマスターホストの 2 種類から成る集中管理型が採用されている．

集中管理型動作モデルを構築する際の参加ホストの起動方法について述べる．ユーザはあらかじめ参加させるホストのネームリストを記述した設定ファイルを用意しておく．マスターホストは最初に起動させたプロセスが担当し，そのマスタープロセスが設定ファイルを読み込み，スレイブホストをバックグラウンドで起動している．

この動作モデルでは，スレイブホストの数が膨大なとき，マスターホストはそのスレイブホスト全てに制御メッセージを送信する必要があり，また各スレイブホストからの返信メッセージを一台のホストで処理しなければならないという問題がある．

3　実装

3.1　階層化動作モデル

問題を解決するためにマスターホストとは別にホストを管理するサブマスターホストを導入し，MegaScript の動作モデルを階層化する．新しい動作モデルは fig. 3.5 のようになる．この動作モデルは，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の送信をすればよい．従って，通信回数が多くなりオーバヘッドが増加してしまうが，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．

3.2　階層化動作モデルの構築

階層化動作モデルを構築する際の参加ホストの起動方法は集中管理型動作モデルとほぼ同様で，異なるのはマスターホストとスレイブホストの間にサブマスターホストを起動することである．次にホスト起動後の各ホスト間接続について述べる．

マスターホストは一つ下層のホスト (子ホスト (fig. 3.6)) を起動後，その子ホストからのソケットの接続を待つ．接続完了後，通信に必要なディ

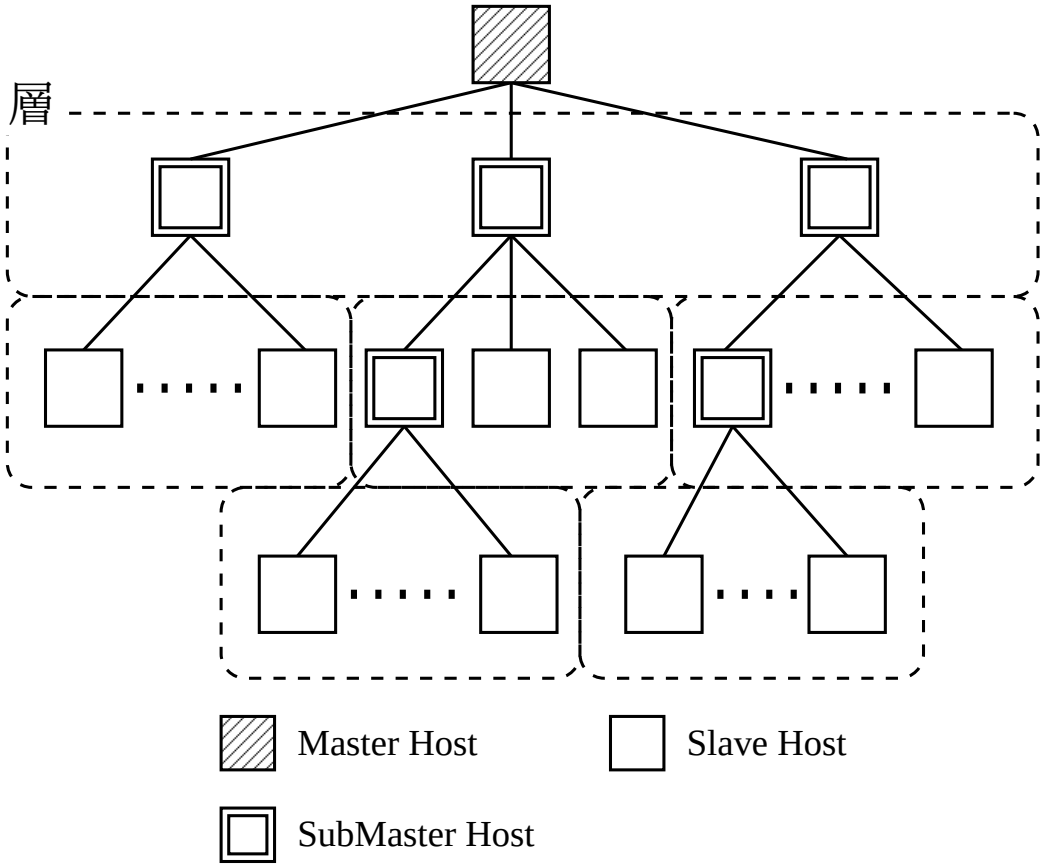

fig . 3.5: 階層化動作モデル

スクリプタとファイルポインタを保持する．そして子ホストに同じ層の別ホスト (兄弟ホスト (fig. 3.6)) 情報を送信し，接続処理を完了する．

サブマスターホストはまず，自分を起動したホスト (親ホスト (fig. 3.6)) との接続を行い，兄弟ホストの情報を読み込み，起動した子ホストからの接続を待つ．接続後はマスターホスト同様に子ホストの兄弟ホスト情報を送信し，親ホストから与えられたホスト情報を用いて兄弟ホストと

接続を行う．接続後はそれぞれディスクリプタ，ファイルポインタを保持し，サブマスターホストの接続処理を完了する．

スレイブホストは親ホストと兄弟ホストと接続し処理を完了する．

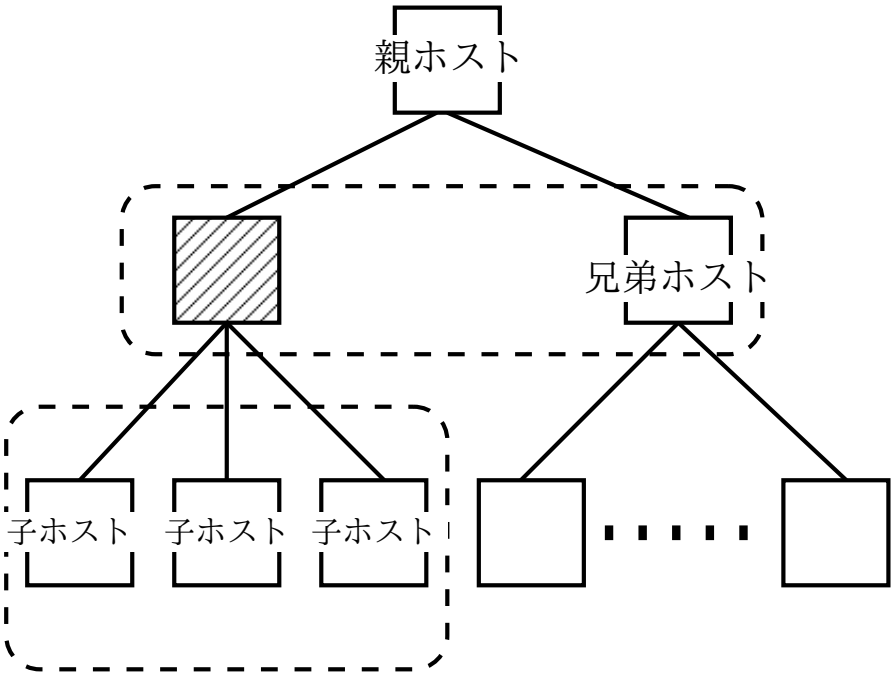

fig . 3.6: ホスト間の関係

3.3 ホスト間の通信

全てのホストは存在する親，子，兄弟ホストの情報を保持し，それぞれと通信が可能である (fig. 3.7)．例えば，タスク生成ならマスターホストから命令が子ホストへ送信される．子ホストがサブマスターホストの場合子ホストへ命令を転送し，スレイブホストの場合タスクの生成を行う．

メッセージの受信は接続時に得たディスクリプタを使用して管理を行っている．ディスクリプタの変化を監視し，変化のあったファイルからデータの読み込みを行う．

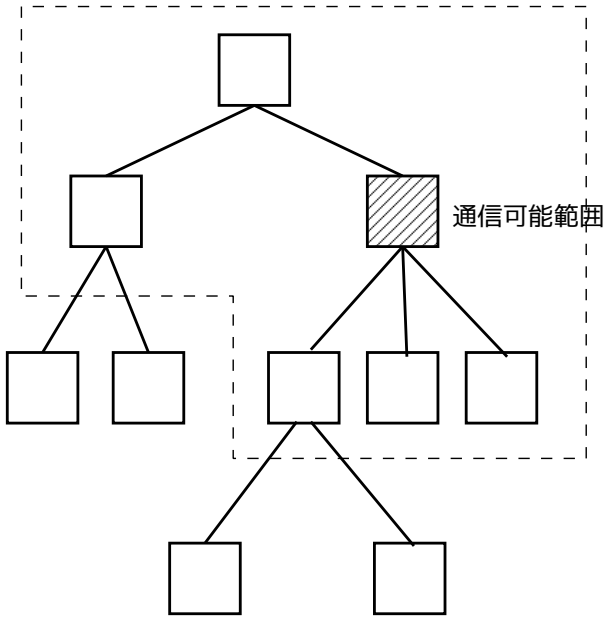

fig . 3.7: ホスト間の通信

3.4 階層化による問題点

サブマスターホストを導入することでホストを階層的に管理することができるようになる．しかし，異なるサブマスターホストに管理されるホスト間でストリーム通信が発生した場合 (fig. 3.8)，互いのホスト情報を保持していないため，送信元ホストは送信先ホストを探索，経由する必要がある (fig. 3.9)．従って，この処理がオーバヘッドとなる．

このオーバヘッドをホストの転送履歴を参照する等を行い軽減することが今後の課題となる.

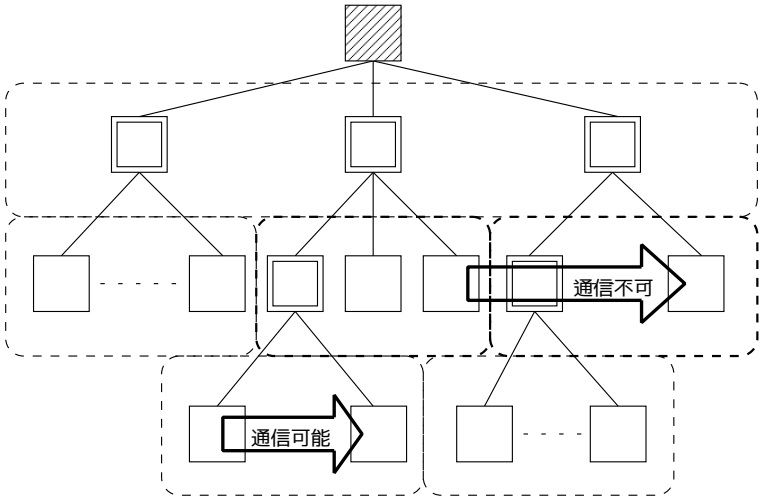

fig . 3.8: 階層化による問題点

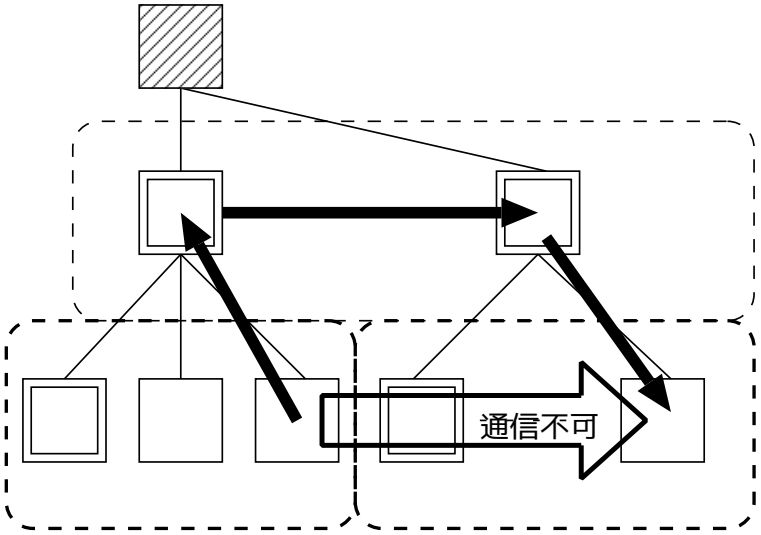

fig . 3.9: 階層化による問題点の解決

4　性能評価

4.1　評価環境

評価用の実験環境として Table．4.1 に示す PC13 台を接続した PC クラスタを用いた.

Table．4.1: 評価用 PC 性能

PC	DELL PowerEdge T100	DELL PowerEdge 840	DELL PowerEdge 800
CPU	Xeon X3330 2.66GHz	Xeon X3210 2.13GHz	Pentium4 2.88GHz
Memory	2GB	2GB	512MB
Network	Gigabit Ethernet	Gigabit Ethernet	Gigabit Ethernet
台数	8	4	1

4.2　評価方法

評価方法について記述します.

評価は TCP/IP 通信で行い，fig．4.10 のような MegaScript で実際に行われるタスク間通信を疑似的に再現し，fig．4.11，fig．4.12 に示すようにタスクの入出力がストリームに対して 1 対 1，1 対 N，N 対 1，N 対 N で接続される 4 パターンにおいて，各タスクが 80Byte の文字列をそれぞれ 10 万回送信した際の集中管理型動作モデルと階層化動作モデルでそれぞれ通信時間を計測した．また，LAN 通信と WAN 通信では通信速度に

大きな差があり，それぞれの通信回数によってワークフローの実行時間は大きく変化する．従って，各通信におけるデータ通信回数を計測した．しかし，本評価環境においてLAN通信とWAN通信の差がないため通信時間に影響はない．

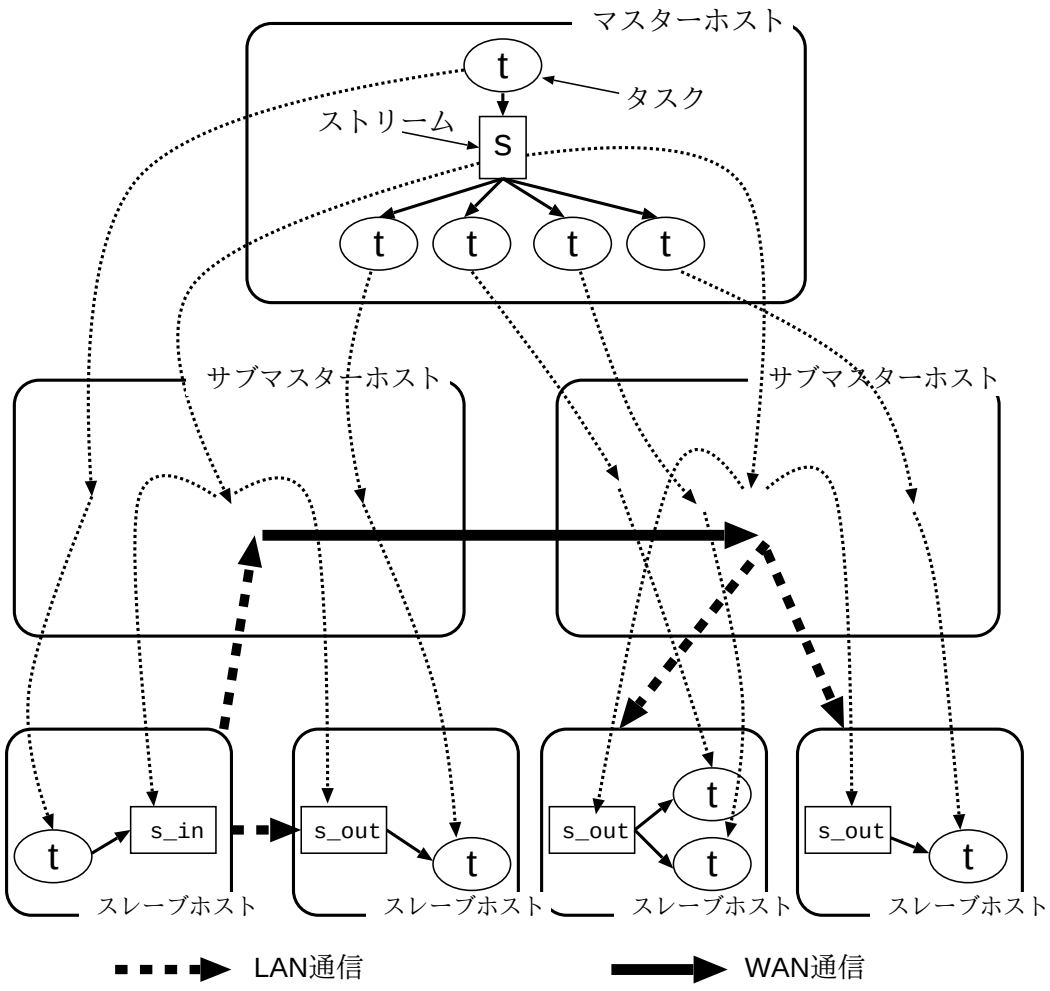

fig . 4.10: 評価時のモデル例

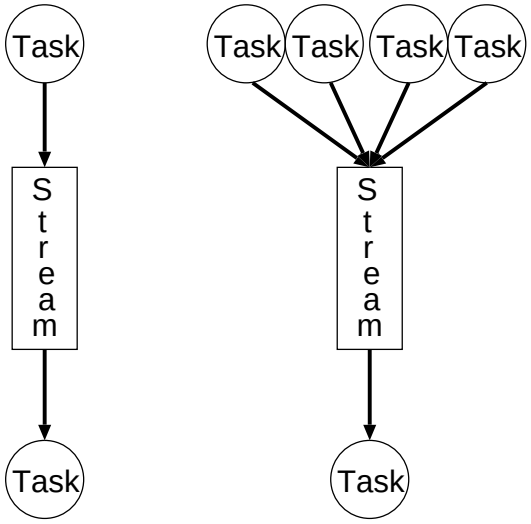

fig . 4.11: タスク間通信のパターン 1

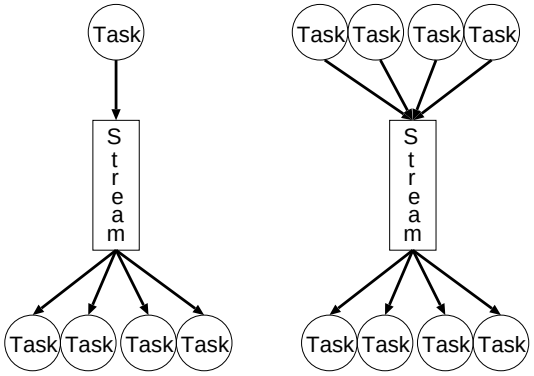

fig . 4.12: タスク間通信のパターン 2

4.3 評価結果

通信時間及び通信回数を計測した結果を 4.2 と 4.3 に示す。

Table. 4.2 からわかるように，階層化動作モデルでは集中管理型動作モデルに比べ通信時間が増加した．これは階層化によってスレイブホスト

Table . 4.2: 通信時間 (N=5)

	集中管理型動作モデル	階層型動作モデル
1 対 1	16.38	17.38
1 対 N	82.38	82.31
N 対 1	20.69	93.69
N 対 N	411.14	429.56

Table . 4.3: 通信種別のホスト間通信回数 (N=5)

	集中管理型		階層型	
	LAN	WAN	LAN	WAN
1 対 1	0	100,000	200,000	100,000
1 対 N	0	500,000	600,000	100,000
N 対 1	0	500,000	1,000,000	500,000
N 対 N	2,000,000	500,000	3,000,000	500,000

間の直接通信ができなくなり，サブマスターホストを中継する分，データの通信回数が増加したためだと考えられる．また，1 対 N の時の WAN 通信回数が減少していることも分かる．従って，実環境にて評価を行った場合，通信時間は階層型動作モデルにおいてより良い値が得られると考えられる．

従って今後は，データの通信回数や一回当たりのデータ通信量を減らすことで，階層化によるオーバヘッドを軽減する必要がある．そこで，冗長なデータ通信を削減したり，複数メッセージをまとめて送信することでデータの通信回数を減らしていく．また，一回当たりのデータ通信量

を減らすため，タスクの出力データの差分のみを送信する機能等を実装していく．

5　あとがき

本研究では，MegaScript 処理系における階層化動作モデルの通信部の実装，評価を行った．その結果，階層化によって大規模な環境を扱えるようになったが，無視できないオーバヘッドが発生した．

今後の課題として，発生したオーバヘッドの軽減手法を開発・実装を行い，実システムによる評価を行っていく必要がある．

また，本研究においては，計算機を 13 台用いて評価を行ったが，計算機台数がさらに増加した場合での通信時間への影響や有用性についても検証していく必要がある．

謝辞

本研究を行うにあたり，ご指導，ご助言いただきました下さいました大野和彦講師，並びに多くの助言をいただきました近藤利夫教授，佐々木敬泰助手に深く感謝いたします．また，様々な局面にてお世話になりました計算機アーキテクチャ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, 西里 一史, 大野 和彦, 中島 浩：タスク並列スクリプト 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] 大野 和彦, 大塚 保紀, 西里 一史, 阪口 裕輔, 高木 悠志, 中島 浩:タスク並列スクリプト言語 MegaScript ユーザマニュアル Ver0.05(2007).

[3] 西里　一史，大野　和彦，中島　浩：タスク並列スクリプト言語 MegaScript のランタイムシステムの設計と実装，情報処理学会研究報告，2003-HPC-95，pp.119-124(2003) .

A　プログラムリスト

B　評価用データ