

LogGPS：メッセージ通信プロトコルの切替えを考慮した 高水準通信ライブラリ向けの並列計算モデル

伊野文彦[†] 藤本典幸[†] 萩原兼一[†]

LogGP モデルを拡張することで高水準通信ライブラリを用いた並列プログラムを解析するための並列計算モデル LogGPS を提案する。LogGP モデルは並列計算機の通信時間をモデル化する。しかし、このモデルはメッセージを送信するときに同期が発生することを考慮していない。MPI などの高水準通信ライブラリでは長いメッセージを送信するときに同期を必要とする通信プロトコルを用いるため、LogGP モデルは同期のための待ち時間が長い並列プログラムの実行時間を正確に見積もることができなかった。そこで、我々は同期を必要とせずに送信できるメッセージ長の上限值などをパラメータとして LogGP モデルへ追加した。本稿では、両モデルに基づいて MPI を用いた並列プログラムを解析し、各々の解析結果を比較する。結果として、並列プログラムの性能改善過程において、同期のための待ち時間を LogGPS モデルで解析することの重要性を確認できた。

LogGPS: Modeling Message-passing Protocols in High-level Communication Libraries

FUMIHIKO INO,[†] NORIYUKI FUJIMOTO[†] and KENICHI HAGIHARA[†]

We present a new parallel computational model, named LogGPS, that captures message-passing protocols in high-level communication libraries. The LogGPS model is an extension of the LogGP model, which abstracts the communication performance on parallel machines. Although the LogGP model captures long messages with the bandwidth parameter G , it has not been taken care of the synchronization which is needed before sending a long message by high-level communication libraries. Our model has one additional parameter S , which is the borderline length between synchronous send and asynchronous send. We also show a comparison between measured and predicted times of an MPI program using both models. The results indicate that the LogGPS model is more precise than the LogGP model, and analyzing synchronization wait-time is important when improving parallel programs.

1. はじめに

より多くの実行環境に対して良い性能が期待できる並列プログラムを開発することは重要である。あらかじめ実行環境の性能を並列計算モデルとして抽象化しておけば、モデルのパラメータを変更することで並列プログラムの性能をさまざまな実行環境に対して解析できる。このような実用性を重視した並列計算モデルとして LogP モデル¹⁾ や LogGP モデル²⁾ がある。LogP モデルは4つのパラメータ L, o, g, P (3章参照)を用いて通信時間をモデル化する。LogGP モデルは LogP モデルを拡張したモデルであり、並列計算機が長いメッセージを効率良く送信するための通信機

構を持つことに着目し、メッセージ長に応じて2つの通信バンド幅 ($1/g, 1/G$) を使い分ける。両モデルは Active Messages³⁾ や Elan library⁴⁾ などの低水準通信ライブラリに対して良い解析精度を示した^{1),2),5)}。

ここで、LogGP モデルは通信機構の長所(通信バンド幅の向上)のみをモデル化していて、その短所(同期の発生)をモデル化していない。このことは MPI⁶⁾ や MPL⁷⁾ などの高水準通信ライブラリを用いた並列プログラムに対する解析精度を悪化させる可能性がある。たとえば、MPI の多くの実装^{8)~11)} ではメッセージ長に応じてライブラリ内部でメッセージ通信プロトコル(以下通信プロトコル)を切替えていて、それらは同期を必要とするものと同期を必要としないものに分類できる。この切替えは、通信機構の長所だけをパラメータとして持つ LogGP モデルでは考慮できない。したがって、高水準通信ライブラリを用いた並列プログラムを対象とする並列計算モデルは長いメッセージ

[†] 大阪大学大学院基礎工学研究科情報数理系専攻
Department of Informatics and Mathematical Science,
Graduate School of Engineering Science, Osaka University

向けの通信機構の長所だけでなく、短所をパラメータとして持つ必要がある。

一方、MPIを用いた並列プログラム（以下**MPIプログラム**）に対して LogGP モデルを適用する研究が行われている^{12)~15)}。文献 12), 13) では、MPI の送信命令を用いて長いメッセージを送信するとき、送信命令の内部で同期が発生することを指摘している。彼らは同期を最短時間で実現できると仮定し、受信バッファの確保などの同期のための時間（**同期コスト**）を通信オーバーヘッド o に含め、 o をメッセージ長ごとの定数としている。しかし、一般に同期がつねに最短時間で実現できるとは限らないため、彼らの手法は同期のための**待ち時間**が大きく変動する並列プログラムに対して良い解析精度が得られない。同期が発生することを考慮しない文献 14) も同様である。

そこで、文献 15) では待ち時間を LogGP モデルとは別にモデル化している。待ち時間を通信遅延 L およびプロセッサ番号を含む数式で表し、LogGP モデルに基づく通信時間に加算している。この手法は解析精度が良いが、あらかじめ待ち時間を数式で導くことのできない並列プログラムに適用できない。

これらの研究より、MPI プログラムに対して良い解析精度を得るためには、並列計算モデルが通信プロトコルの切替えを考慮し、待ち時間を正確に見積もる必要があるといえる。また、実行環境から LogGP モデルのパラメータ値を取得する方法、さらに各パラメータを用いて MPI 通信命令の**処理時間**を定義する方法は文献ごとに異なり、どの方法が良い解析精度を得られるのか明らかでない。

以上をふまえ、本稿では以下を目標とする。

(G1) 高水準通信ライブラリ向けの実用的な並列計算モデル **LogGPS** の提案

(G2) 高水準通信ライブラリへの LogGPS モデルの適用およびその適用方法の評価

我々は (G1) を実現するために、同期を必要とせずに送信できるメッセージ長の上限値をパラメータ S として LogGP モデルへ追加した。さらに、モデルの解析精度を向上するために、通信オーバーヘッド o を線形関数で表すなどの修正を加えた。また、(G2) に対しては、多くの実行環境で動作する高水準通信ライブラリ MPI⁶⁾ に対して LogGPS モデルを適用、さらに LogGP モデルに基づく適用方法^{12),13),15)} と比較検討した。

本稿では、まず MPI の各実装における通信プロトコルについて整理する。次に、提案する並列計算モデル LogGPS について説明し、LogGPS モデルにおけ

る主な MPI 通信命令の処理時間を定義する。その後、MPI プログラムを用いて LogGPS モデルを評価する。評価の結果、LogGPS モデルは MPI プログラムを解析するうえで有用であり、特に MPI プログラムの性能を改善する過程において有益な情報を引き出せることを確認できた。

2. MPI 実装の通信プロトコル

本章では MPI の各実装^{8)~11)} における通信プロトコルについて整理する。

以下、多くの実装の基となる MPICH⁸⁾ について述べる。MPICH はハードウェア非依存の層とハードウェア依存の層 (ADI, Abstract Device Interface) を持つ⁸⁾。表 1 に、デフォルトの ADI (P4) が用いる 3 種類の通信プロトコルを示す。なお、以降では送信プロセッサを Ps, 受信プロセッサを Pr とする。

Short プロトコルおよび **Eager** プロトコルでは、Pr がメッセージ内容をバッファに複製するため、Ps は対応する受信命令の呼び出しを待たずに次の処理を開始できる (図 1(a))。一方、**Rndv.** (Rendezvous) プロトコルでは、Ps は送信要求 REQ を送信してから Pr からの送信許可 ACK が返ってくるまで待つ (同期する) 必要がある (図 1(b))。このとき、送信命令がブロッキング送信 (MPI_Send) であれば Ps は送信命令の中でアイドル状態となる。送信命令がノンブロッキング送信 (MPI_Isend) であれば、Ps は REQ を送信したのち次の処理を開始できる (図 1(c))。ただし、対応する通信完了命令 (MPI_Wait) を呼び出したときに ACK が返っていなければ、Ps は通信完了命令の中でアイドル状態となる。Rndv. プロトコルは Pr との同期を必要とするが、同期後はまとめてメッセージを送信できるため、長いメッセージを効率良く送信できる。また、NEC Cenju-4¹⁶⁾ や Myrinet¹⁷⁾ のように、リモート DMA 転送が行える場合は、さらに送信効率を向上させることができる。

表 2 に各実装^{8)~11)} におけるメッセージ長と通信プロトコルの関係を示す。各々、短いメッセージには同期が不要な通信プロトコル (Short, Eager) を、長いメッセージには同期が必要だが送信効率の良い通信プロトコル (Rndv.) を採用している。このように、メッ

表 1 MPICH の通信プロトコル
Table 1 MPICH message-passing protocols.

Protocol	Packet	Synchronization
Short	single	no
Eager	multiple	no
Rendezvous	multiple	yes

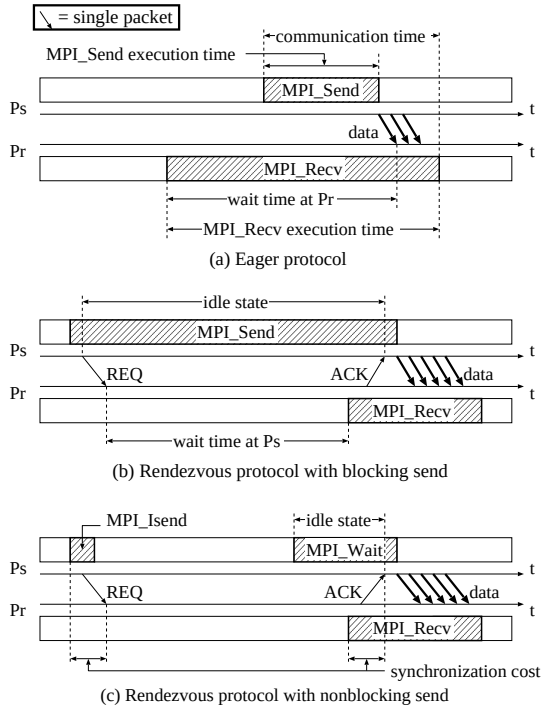


図 1 MPICH の通信プロトコルの動作

Fig. 1 Behavior of MPICH message-passing protocols.

表 2 MPI 実装におけるメッセージ長と通信プロトコルの関係
Table 2 Relationship between message length and message-passing protocols of MPI implementations.

Implementation	Message length (Bytes)		
	Short	Eager	Rndv.
MPICH P4	~1023	~127999	128000~
MPICH Cenju-4	~1028	—	1029~
MPICH-SCore	~8191*	~16383**	16384** ~
LAM C2C	~65536	—	65537~
IBM PE	~4095	—	4096** ~

*: on Myrinet **: changeable by runtime option

セージ長に応じて 3 つの通信プロトコルを使い分けることで、任意の長さのメッセージを効率良く送信できる。

なお、Rndv. プロトコルへ切り替わるメッセージ長の上限値を変更するためには、各実装のソースコードを修正し、通信ライブラリを再コンパイルする必要がある。ただし、MPICH-SCore⁹⁾ および IBM PE¹¹⁾ では MPI プログラムを実行するときのオプションで上限値を変更できる。なお、MPI プログラムの実行中に上限値を動的に変更することはできない。

以降では、REQ の到着から受信命令の呼び出しまでの Ps における 1 待ち時間を**送信待ち時間**と呼び (図 1 (b)), 受信命令の呼び出しからメッセージの到

着までの Pr における待ち時間を**受信待ち時間**と呼ぶ (図 1 (a)). また、REQ の送信および ACK の受信などの同期のための時間を**同期コスト**と呼ぶ (図 1 (c)). さらに、送信命令の呼び出しから受信命令の完了までの時間を**通信時間**と呼び、MPI 通信命令の呼び出しから完了までの時間をその命令の**処理時間**と呼ぶ (図 1 (a)).

3. LogGPS モデル

提案する LogGPS モデルの基となる LogGP モデル²⁾ は以下の 5 つのパラメータを持つ。

- L : 通信遅延. 送信プロセッサから受信プロセッサまでのネットワーク転送に要する時間。
- o : 通信オーバーヘッド. メッセージを送信もしくは受信するときにプロセッサが占有される時間. この間プロセッサは他の処理を行うことはできない。
- g : メッセージを連続して送信もしくは受信するときの最短の時間間隔. 逆数 $1/g$ は短いメッセージ向けの通信バンド幅に対応。
- G : 長いメッセージを送信するときの 1 バイトあたりの時間間隔. 逆数 $1/G$ は長いメッセージ向けの通信バンド幅に対応。
- P : プロセッサ数。

これらのパラメータに加えて、LogGPS モデルは以下の相違点を持つ。なお、以降では送信するメッセージ長を k バイトとする ($k \geq 0$)。

(D1) パラメータ S の追加. S は同期を必要とせずに送信できるメッセージ長の上限値を表す. $k > S$ のとき, Ps は Pr からの ACK を待つ。

(D2) パラメータ s を追加し, 間隔 G を $k \leq s$ のとき G_s , $k > s$ のとき G_l とする. s は 1 つのパケットで送信できるメッセージ長の上限値を表す。

(D3) 通信オーバーヘッド o を Ps および Pr, $k \leq S$ および $k > S$ で区別し, 各々 k に関する線形関数 $o' + kO_{ss}$, $o' + kO_{sl}$, $o' + kO_{rs}$ および $o' + kO_{rl}$ とする. ここで, o' は 0 バイトのメッセージを送信もしくは受信するときのオーバーヘッドを表し, O_{ss} および O_{rs} (O_{sl} および O_{rl}) は $k \leq S$ ($k > S$) のときのメッセージ 1 バイトあたりの送信オーバーヘッドおよび受信オーバーヘッドを表す。

(D1) を設けることでメッセージごとに同期が発生するか否かをモデルが判定することができる。(D2) および (D3) は, モデルの解析精度を向上するための工夫である。なお, (D2) および (D3) を設ける根拠は 4.2 節で述べる。

表 3 LogGPS モデルにおける通信時間
Table 3 Communication times under the LogGPS model.

Condition	Communication time	$T_1 \sim T_5, T'_1 \sim T'_3$
$k \leq s$	$T_1 + T_2 + T_3$ (1)	$T_1 = o' + kO_{ss}$ $T'_1 = o' + kO_{sl}$
$s < k \leq S$	$T_1 + T'_2 + T_3$ (2)	$T_2 = kG_s + L$ $T'_2 = sG_s + (k-s)G_l + L$
$k > S$	$T_4 + T_5 + T'_1 + T'_2 + T'_3$ (3)	$T_3 = o' + kO_{rs}$ $T'_3 = o' + kO_{rl}$
		$T_4 = \max\{o' + L, t_r - t_s\} + o'$ $T_5 = o' + L + o'$

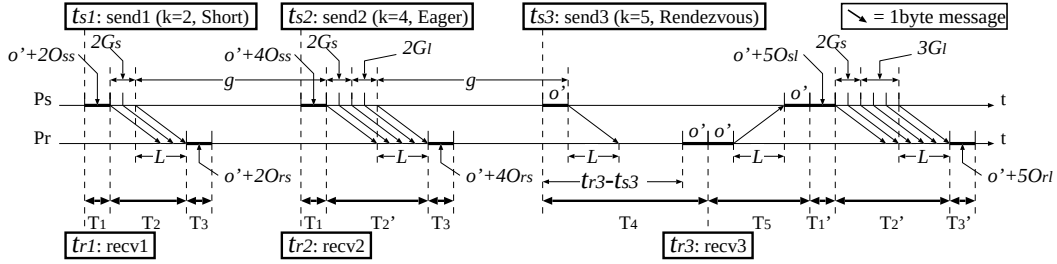


図 2 LogGPS モデルにおけるメッセージ送例 ($s = 2, S = 4$)
Fig. 2 An example of message sends under the LogGPS model ($s = 2, S = 4$).

また、LogP モデル同様、ネットワークのトポロジを隠蔽する¹⁾。さらに、 p 台のプロセッサが同一プロセッサへメッセージを送信するとき、 p 個のメッセージが受信プロセッサの通信バンド幅 $1/G_s$ および $1/G_l$ を共有すると考え、各々を p で除算した値を実効通信バンド幅とする¹⁸⁾。これらの想定は実際のネットワークを厳密にモデル化しているとはいえないが、多くの実行環境において許容できると考える (3.3 節参照)。

3.1 通信時間の定義

本節では、LogGPS モデルに基づいて k バイトのメッセージを送信するときの通信時間を定義する (表 3)。

図 2 に LogGPS モデルにおけるメッセージ送例 ($s = 2, S = 4$) を示す。この例では Ps から Pr へ 3 回の送信 (send1~send3) を行っている。

まず、 $k \leq S$ のとき、Ps は Pr と同期することなくメッセージを送信する (send1, send2)。Ps がメッセージを構成する先頭バイトをネットワークに流し出すまでに要する時間 T_1 は送信オーバーヘッド $o' + kO_{ss}$ である。次に、先頭バイトに続いて s バイトまでは 1 バイトあたり G_s の間隔でメッセージが流れ出る (send1)。 s バイトを超えると 1 バイトあたり G_l の間隔で引き続きメッセージが流れ出る (send2)。同様に、Pr は s バイトまでは G_s の間隔で、それ以降は G_l の間隔でメッセージを受け取る。ただし、メッセージを構成する各バイトは Ps を出発して Pr に到着するまでに通信遅延 L を必要とする。以上から、メッセージの先頭が Ps を出発してからメッセージの後尾が Pr に到着するまでの時間 T_2 (および T'_2) は、 $k \leq s$ のとき

$T_2 = kG_s + L$ 、 $k > s$ のとき $T'_2 = sG_s + (k-s)G_l + L$ となる。最後に、メッセージ全体が Pr に到着した後、Pr が次の処理を開始できるようになるまでに要する時間 T_3 は受信オーバーヘッド $o' + kO_{rs}$ である。 $k \leq S$ のときの通信時間は $T_1 \sim T_3$ の和で求まる (表 3 の式 (1), (2))。

次に、 $k > S$ のときを考える (send3)。このとき Ps は Pr と同期する必要があるため、先の $T_1 \sim T_3$ の O_{ss} および O_{rs} を、各々 O_{sl} および O_{rl} に置き換えた $T'_1 \sim T'_3$ に加えて、同期を実現するための時間の和を通信時間とする。まず、Ps は送信要求 REQ を送信する。LogGPS モデルでは、REQ の送信は 0 バイトのメッセージ送信と等価であると見なす。このとき、REQ が Pr に到着するまでに要する時間は $o' + L$ となる。次に、Ps は Pr から ACK が返ってくるまで待つ。一方、Pr は受信命令を呼び出したときに、REQ の到着を確認する。このとき要する時間は受信オーバーヘッド o' である。以上から、送信命令の呼出時刻を t_s 、受信命令の呼出時刻を t_r としたとき、送信命令の呼び出しから Pr における REQ の確認までに要する時間 T_4 は $\max\{o' + L, t_r - t_s\} + o'$ となる。項 $t_r - t_s$ と項 $o' + L$ のうち、値の大きい方を通信時間に含めることで、送信待ち時間を LogGPS モデルに反映できる。その後、Pr は ACK を Ps に返す。REQ 同様、ACK の送信を 1 バイトのメッセージ送信と見なすと、このとき要する時間 T_5 は $o' + L + o'$ となる。Ps は ACK を受け取った後、本来のメッセージを送信する。以降の処理に要する時間は $k \leq s$ のときと同様に算出できる。以上から、 $k > S$ のときの通信時

表 4 LogGPS モデルにおける主な MPI 通信命令の処理時間
Table 4 MPI routine execution times under the LogGPS model.

Routine	Condition	Execution time
MPI_Send	$k \leq S$	T_1 (4)
	$k > S$	$T_4 + T_5 + T'_1$ (5)
MPI_Isend		o' (6)
MPI_Recv	$k \leq s$	$\max\{T_1 + T_2 - (t_r - t_s), 0\} + T_3$ (7)
	$s < k \leq S$	$\max\{T_1 + T'_2 - (t_r - t_s), 0\} + T_3$ (8)
	$k > S$	$\max\{o' + L - (t_r - t_s), 0\} + o' + T_5 + T'_1 + T'_2 + T'_3$ (9)
MPI_Irecv		o' (10)
MPI_Wait		$\max\{T_{blk} - (t_i - t_w), o'\}$ (11)

間は T_4, T_5 および $T'_1 \sim T'_3$ の和で求まる (式 (3)).

3.2 MPI 通信命令への適用

MPI 通信命令の処理時間は 3.1 節で定義した通信時間と必ずしも一致しない (図 1(a)). たとえば、ブロッキング送信命令およびノンブロッキング送信命令を用いて長さ k のメッセージを送信する場合、双方の通信時間は同一だが、後者は MPI_Isend の完了から MPI_Wait の呼び出しまでの時間を他の処理に費すことができる (図 1(b), (c)). したがって、MPI プログラムを良い精度で解析するためには MPI 通信命令ごとに処理時間を定義する必要がある。

一般に、MPI の実装は低水準通信ライブラリを用いることが多い。たとえば、MPICH Cenju-4 は Paralib/CJ4¹⁶⁾ を、MPICH-SCore は PM¹⁹⁾ を用いている。したがって、MPI 通信命令の処理時間を定義するとき、内部の低水準通信命令のモデル化に関して、以下の 2 通りの方法が考えられる。

- 内部の低水準通信命令ごとにモデル化
- 内部の低水準通信命令を隠蔽してモデル化

前者は詳細なモデル化であるため、LogP モデルおよび LogGP モデルを利用する場合も精度が良い。しかし、MPI の実装ごとに内部処理をモデル化するため、実装独立な方法といえず、一般的には後者の方法をとることが多い^{12)~15)}。

一方、後者は MPI 通信命令の内部処理を隠蔽するため、実装独立な定義が得られる。ただし、内部処理を隠蔽しても矛盾が生じないモデルが必要となる。たとえば、MPI_Send の処理時間を定数 o とするモデルは、MPI_Send の内部でメッセージ全体をネットワークに流し出す実装に適用できない。なぜなら、メッセージ全体を流し出すときの時間がメッセージ長 k に関する線形関数 T_2 となり、定数 o と矛盾するためである。そこで、LogGPS モデルでは時間 $T_1 \sim T_3$ をすべて線形関数とすることで、メッセージを流し出す処理がどの時間に含まれてもよいようにしている。また、内部処理を隠蔽することで、モデルの各パラメー

タの持つ意味が変わる可能性がある。したがって、実効通信バンド幅を算出する際 (3 章参照)、どのパラメータが本来の通信バンド幅を含むのかを判断する必要がある。そこで、LogGPS モデルでは O_{ss}, O_{rs} および G_s (O_{sl}, O_{rl} および G_l) のうち最も大きな値を持つものが $k \leq s$ ($k > s$) のときの通信バンド幅を含むと考え、実効通信バンド幅を算出するときの除算対象とする。

以下、MPI の通信命令 MPI_Send, MPI_Isend, MPI_Wait の処理時間を定義する。他の主な通信命令については結果のみを示す (表 4)。

まず、MPI_Send の処理時間を定義する。 $k \leq S$ のとき、すなわち非同期でメッセージを送信するとき P_s は送信オーバーヘッド T_1 を経れば、MPI_Send を完了できる (式 (4))。一方、 $k > S$ のとき P_s は P_r と同期する必要がある。したがって、同期を実現するための時間 $T_4 + T_5$ を経たのち T'_1 を経れば、MPI_Send を完了できる (式 (5))。

次に、MPI_Isend の処理時間を定義する。 $k \leq S$ のとき P_s はメッセージを送信するための処理を開始すれば、MPI_Isend を完了できる (式 (6))。一方、 $k > S$ のとき P_s は P_r と同期する必要があるが、ノンブロッキング送信の場合は REQ を送信すれば MPI_Isend を完了できる (式 (6))。

なお、送信命令もしくは受信命令を繰り返し呼び出す場合、2 番目以降の各命令は間隔 g を保つ必要がある。したがって、各命令の処理時間は厳密には g を含む式であるべきだが、 g は省く (4.1 節で後述)。

最後に、通信完了命令 (MPI_Wait) の処理時間 T_{wait} を考える。一般に、ブロッキング通信命令はノンブロッキング通信命令と通信完了命令を続けて呼び出すことで実現できる。したがって、 T_{wait} はブロッキング通信命令の処理時間 T_{blk} から逆算することで求める。まず、通信完了命令に対応するノンブロッキング通信命令をブロッキング通信命令に置き換えたときの処理時間 T_{blk} を算出する。次に、通信完了命令の呼出時刻

を t_w , 対応するノンブロッキング通信命令の呼出時刻を t_i とすると, 時刻 t_w には T_{blk} のうち $t_i - t_w$ に相当する仕事を処理できたことになる. したがって, T_{wait} を式 (11) のように定める. ここで, 式 (11) の項 o' は, $T_{blk} - (t_i - t_w) < 0$ のとき, すなわち時刻 t_w 以前に通信処理が完了したときへの対策である. 厳密には通信処理の完了を確認するときに要する時間を用いるべきだが, モデルを簡略にするため o' を代わりとした.

3.3 ネットワークのモデル化に関する議論

LogP モデルではネットワークのトポロジを隠蔽した. そのうえで, ネットワーク全体の通信バンド幅が存在すると仮定し, 同時に $\lceil L/g \rceil$ 個のメッセージが任意のプロセッサ間で通信できると定義した¹⁾. しかし, より高速なネットワークが実現するにつれ, 彼らは現実的なシステムでは全体の通信バンド幅よりもネットワークインタフェース (NI) における通信バンド幅が性能ボトルネックになることを指摘した¹⁸⁾. 一方, LogGP モデル²⁾ ではネットワークが完全結合であることを想定し, 任意の時刻においてプロセッサは 1 個のメッセージのみを送信もしくは受信できるとした.

LogGPS モデルでは, 文献 18) 同様, NI が性能ボトルネックとなることが多いと考え, 通信バンド幅をメッセージ数 p で除算する. 3つのモデルは実際のネットワークを厳密にモデル化しないため, ネットワーク上のパケットを詳細に解析するなどの用途には向かない. しかし, Active Messages や MPI などの通信ライブラリを用いて並列プログラムを開発するユーザには容認できるモデル化であると考え.

4. 評価

本章では, 以下の観点で LogGPS モデルを評価する.

- (1) モデルの妥当性. 複数の実行環境において (D2) および (D3) の工夫がモデルの解析精度を向上させることを確認する (4.2 節).
- (2) モデルの有用性. LogGPS モデルに基づいて MPI プログラムを解析した結果, その性能ボトルネックが送信待ち時間にあった例を示す (4.3 節).

また, その性能ボトルネックの削減過程を示す (4.4 節).

- (3) LogGP モデルとの比較. LogGPS モデルおよび LogGP モデルに基づく解析結果を比較し, 前者の解析精度が優れていることを確認する (4.5 節).

MPI プログラムの実行には分散メモリ型並列計算機 NEC Cenju-4¹⁶⁾ および PC クラスタを用いた. 後者は 16 台の PC (Pentium II 450 MHz) を Myrinet¹⁷⁾ および Fast Ethernet により接続したクラスタである. また, 使用した MPI の実装は Cenju-4 が MPICH⁸⁾, PC クラスタが MPICH-SCore⁹⁾ である.

評価に用いた MPI プログラムは, メッセージの往復時間 (RTT, Round Trip Time) を計測する RTT 計測プログラムと並列ソフトウェアコンテスト PSC95²⁰⁾ の Cenju-3 部門で優勝したプログラム²¹⁾ である. 後者は, ガウスの消去法を用いて連立方程式 $Ax = b$ ($A: n \times n$ の密行列) を解く. なお, コンテストでは Cenju-3 で 64 台のプロセッサを用い, 行列サイズ $n = 1000 \sim 2000$ で解いている.

4.1 パラメータ値の取得

LogGPS モデルの各パラメータ値を取得する方法について述べる. まず, S および s は MPI の実装に依存するため, 各々のドキュメントおよびソースコードから調べた (表 2). 次に, g を除く各パラメータ値は RTT 計測プログラムを用いて往復時間を実測し, LogGPS モデルにおける往復時間と比較することで求めた (4.2 節). 最後に, g は MPI_Send を連続して呼び出し ($0 < k \leq s$), 2 回目以降の MPI_Send の処理時間から 1 回目の処理時間を減算することで求めた. しかし, 個々の処理時間は呼出回数に独立してほぼ一定であったため, 高水準通信ライブラリでは o' が g を含有すると判断し, 文献 13) 同様, その存在を無視した. o' が g を含有する理由の 1 つとして, MPI_Send 内での関数の呼出回数が多い点があげられる. 以降では, g を含まない, 表 4 の式 (4)~(11) を用いて各 MPI 通信命令の処理時間を算出した. 表 5 に取得したパラメータ値を示す.

表 5 各実行環境における LogGPS パラメータ
Table 5 LogGPS parameters for each machines.

Platform	L (ns)	o' (ns)	O_{ss} (ns)	O_{rs} (ns)	G_s (ns)	O_{sl} (ns)	O_{rl} (ns)	G_l (ns)	s (bytes)	S (bytes)
Cenju-4	0.51×10^3	4.26×10^3	19.61	1.98	0.04	7.02	0.28*		1028	1028
Myrinet	0.85×10^3	6.73×10^3	5.02	4.72	15.17	4.80	3.86	0.04	8191	16383
Fast Ethernet	24.58×10^3	22.34×10^3	12.37	11.53	188.77	9.73	9.89	65.09	1023	16383

*: $O_{rl} + G_l = 0.28$

4.2 LogGPS モデルの検証

RTT 計測プログラムを用いて 3 章の (D2) および (D3) の妥当性を検証する. 作成した RTT 計測プログラムはブロッキング通信命令を用いて k バイトのメッセージを 2 つのプロセッサ間で往復させるプログラムである (図 3). ただし, プロセッサの計算負荷を不均等に設定できるように, 片方のプロセッサの処理にダミーループを追加した. 図 3 に, ダミーループの実行時間を w としたときの実行の様子 ($k \leq s$) を示す.

以下, LogGPS モデルに基づく往復時間 RTT_1 を導出する (表 6). 紙面の都合上, $k \leq s$ のときについてのみ述べる. P0 における MPI_Recv の処理時間を T_{Recv} とすると, 図 3 から $RTT_1 = T_1 + w + T_{Recv}$ とできる. ここで, w を大きくするとプロセッサ P1 はプロセッサ P0 よりも負荷が相対的に軽くなるため, P0 が P1 の通信命令の呼び出しを待つことはないかと仮定できる. このとき, P0 における MPI_Send の呼出時刻を 0 とし, P1 における MPI_Send の呼出時刻 t_s は $T_1 + T_2 + T_3$ となる. 一方, P0 における MPI_Recv の呼出時刻 t_r は $T_1 + w$ であるので, 式 (7) より

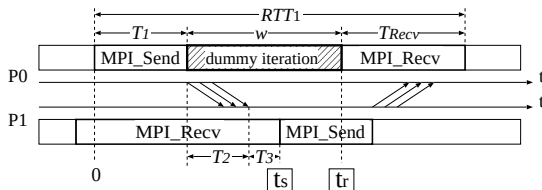


図 3 RTT 計測プログラムの動作 ($k \leq s$)

Fig. 3 Behavior of the RTT measurement program ($k \leq s$).

$T_{Recv} = \max\{T_1 + 2T_2 + T_3 - w, 0\} + T_3$ となる. 以上から, $k \leq s$ のときの往復時間 RTT_1 は表 6 の式 (12) のように求まる. 式 (12) から $w \geq T_1 + 2T_2 + T_3$ のとき, RTT_1 は T_2 に依存しない. つまり, 通信遅延を計算時間で隠蔽できる (通信と計算のオーバーラップ). 同様に $s < k \leq S$ および $k > S$ のときについて RTT_1 を求めた結果を表 6 に示す. また, 図 4 に実測に基づく往復時間 RTT_2 と k の関係を, 図 5 に $w = 0$ および $w = W$ (W は十分大きな値) のときに得られる RTT_1 と k の関係を示す.

以下, 図 5 について検証する. まず, $w = W$ かつ $k \leq S$ のとき, グラフの勾配は $O_{ss} + O_{rs}$ である (図 5). 仮に, 通信オーバーヘッドが定数 o' であるとすると ($O_{ss} = O_{rs} = 0$), この部分の勾配は 0 になる. ゆえに, $w = W$ かつ $k \leq S$ の勾配を見ることで, 通信オーバーヘッドが線形関数であるかどうかを判断できる. 図 4 をみると, どの実行環境においてもグラフの勾配は 0 でない. したがって, 通信オーバーヘッ

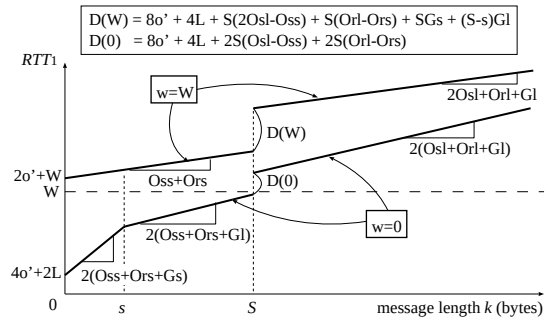
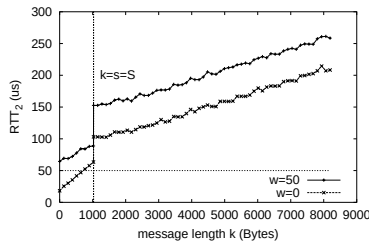


図 5 メッセージ長を変動させたときの往復時間 (LogGPS)
Fig. 5 LogGPS Round Trip Time for different message lengths.

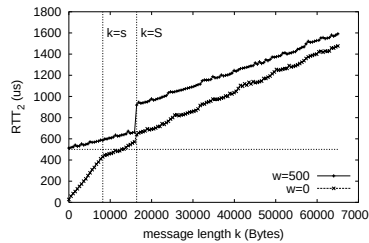
表 6 LogGPS モデルに基づく往復時間

Table 6 LogGPS Round Trip Time.

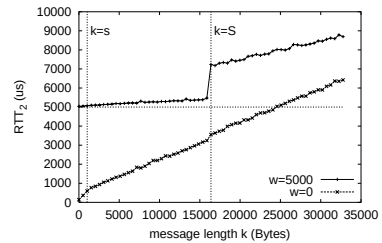
Condition	Round Trip Time: RTT_1
$k \leq s$	$\max\{T_1 + 2T_2 + T_3 - w, 0\} + w + T_1 + T_3$ (12)
$s < k \leq S$	$\max\{T_1 + 2T'_2 + T_3 - w, 0\} + w + T_1 + T_3$ (13)
$k > S$	$\max\{T'_2 + T'_3 + o' + L - w, 0\} + w + 2T'_1 + T'_2 + T'_3 + 2T_4 + 2T_5 - (o' + L)$ (14)



(a) Cenju-4 ($s=S=1028$)



(b) Myrinet ($s=8191, S=16383$)



(c) Fast Ethernet ($s=1023, S=16383$)

図 4 メッセージ長を変動させたときの往復時間 (実測)

Fig. 4 Measured Round Trip Time for different message lengths.

ドを線形関数で表す方がモデルの精度が良いといえる (D3).

次に, $w = W$ のとき, $k = s$ の前後においてグラフは連続である (図 5). このことは, $k > s$ においてメッセージがネットワークを流れる時間 T'_2 が $sG_s + (k - s)G_l + L$ と表せることを意味する. 仮に, T'_2 が $kG_l + L$ であるとする, 通信オーバーヘッド o' は $k \leq s$ のときと $s < k \leq S$ のときと異なる値を持つ必要がある ($w = 0$ かつ $k = s$ においてグラフが連続であるため). o' が異なる値を持つと, $w = W$ かつ $k = s$ においてグラフが不連続となる. これは図 4 の実行結果と矛盾する. ゆえに, $w = W$ かつ $k = s$ のときにグラフが連続であれば, T'_2 は $sG_s + (k - s)G_l + L$ と表せる. この式は先頭パケットに引続き後続パケットを連続して転送するワームホールルーティングの特性を表す. なお, 図 4 の $k \leq s$ および $k > s$ の勾配から G_s と G_l が異なる値を持つことも分かる (D2).

最後に, $w = W$ かつ $k \leq S$ のとき, グラフの勾配は G_s および G_l に依存しない (図 5). すなわち $k \leq S$ のとき, 計算負荷に偏りのある MPI プログラムではネットワークの性能よりも通信ライブラリの性能 (通信オーバーヘッド) がプログラム全体の性能に影響する. 実際に図 4(c) において $w = 5000$ かつ $k \leq 16383$ の勾配は 292 Mbit/s であり, この値は Fast Ethernet の理論値 100 Mbit/s を超える. この理由は, P0 が受信命令を呼び出したときには, P1 からのメッセージはすでに P0 のバッファに到着していて, P0 はバッファからメッセージを複製するだけでよかったことにある.

なお, LogGPS の各パラメータ値は図 4 の実測結果を図 5 と比較することで算出する. L および o' は縦軸の 2 切片から 2 つの方程式を作成し, それらを解くことで求める. 残り 6 つのパラメータ値はグラフの 5 カ所の勾配から得られる 5 つの方程式および $k = S$ のときの MPI_Send の処理時間 T_1 から得られる方程式を解くことで求める. Myrinet における各方程式を以下に示す (単位はナノ秒, $W = 5 \times 10^5$, $S = 16383$).

$$4o' + 2L = 2.862000 \times 10^4$$

$$2o' + W = 5.134580 \times 10^5$$

$$O_{ss} + O_{rs} = 9.733257$$

$$2(O_{ss} + O_{rs} + G_s) = 4.979819 \times 10^1$$

$$2(O_{ss} + O_{rs} + G_l) = 1.955259 \times 10^1$$

$$2(O_{sl} + O_{rl} + G_l) = 1.740265 \times 10^1$$

$$2O_{sl} + O_{rl} + G_l = 1.350428 \times 10^1$$

$$o' + SO_{ss} = 8.893013 \times 10^6$$

4.3 MPI プログラムの解析例

LogGPS モデルに基づいて MPI プログラムを解析した例として, ガウスの消去法を用いて連立方程式を解くプログラムをあげる. プログラムの詳細は文献 21) を参照していただきたい.

プログラムの解析はツールを作成することで実現した. ツールはトレースデータおよび LogGPS モデルの各パラメータ値を入力とし, モデルに基づく予測時間およびその内訳などを出力する. トレースデータは, 解析対象とする MPI プログラムを実行することでプロセッサごとに得られ, 各プロセッサが実行した MPI 通信命令の種類, 呼出時刻, 完了時刻および引数 (メッセージ長, 通信相手および通信タグなど) を時系列順に記録したファイルである. なお, 時刻計測には MPI の時刻計測命令 MPI_Wtime を用いた.

ツールはプロセッサごとに通信命令の処理時間 t_{com} および計算部分の処理時間 t_{cal} を時系列順に加算していき, 全プロセッサの中で最も長い処理時間を予測時間と見なす. ここで, 計算部分とは通信命令の完了 (完了時刻 t_r) から次の通信命令の呼び出し (呼出時刻 t_c) までを指し, トレースデータの記録 t_r および t_c を用いて $t_{cal} = t_c - t_r$ とする. 一方, t_{com} は LogGPS モデルの各パラメータ値および式 (4)~(11) を基に算出する. ただし, 3 章で述べた実効通信バンド幅はツール内の処理の進行とともに変動する値である.

以下, 解析の手順について述べる. まず, MPI プログラムが含むすべての MPI 通信命令をトレースデータ生成命令へ置換した. トレースデータ生成命令は元の MPI 命令の呼び出しを含む. 次に, 置換済のプログラムをトレースデータ生成ライブラリとリンクし, PC クラスタ (Myrinet, 16 台) で実行することで行列サイズ n ごとのトレースデータを生成した. 最後に, トレースデータおよび表 5 のパラメータ値 (Myrinet および Fast Ethernet) をツールへ入力し, プログラムを解析した. 表 7 左欄に, 実測によるプログラムの実行時間 T_{m1} , LogGPS モデルに基づく予測時間 T_{p1} および誤差 $\sigma_1 = 100 \times (T_{p1} - T_{m1})/T_{m1}$ を示す. また, Myrinet における T_{p1} の内訳を全プロセッサで平均したものを図 6(a) に示す.

σ_1 より, LogGPS モデルは実際の実行を 7%以内の精度で予測できている. また, $n = 1024$ のときを除いて待ち時間が T_{p1} のおよそ 50%を占めていることから (図 6(a)), このプログラムの性能ボトルネックは待ち時間にあるといえる. さらに, $n \leq 1024$ のときは受信待ち時間が大きいのに対し, $n > 1024$ のと

表 7 ガウスの消去法プログラムの実行時間と予測時間 (LogGPS)

Table 7 Measured and predicted times for the Gaussian elimination program (LogGPS).

Platform	matrix size n	original code, $S=16383$			original code, $S=65536$			fragmented code, $S=16383$		
		Meas.	Pred.	Err.	Meas.	Pred.	Err.	Meas.	Pred.	Err.
	n	T_{m1}	T_{p1}	σ_1	T_{m2}	T_{p2}	σ_2	T_{m3}	T_{p3}	σ_3
Myrinet	256	0.047	0.050	6.4	0.047	0.050	6.4	0.047	0.050	6.4
	512	0.121	0.129	6.6	0.121	0.129	6.6	0.121	0.129	6.6
	1024	0.612	0.603	-1.5	0.612	0.603	-1.5	0.612	0.603	-1.5
	2048	7.668	7.587	-1.1	4.403	4.365	-0.9	4.512	4.435	-1.7
	4096	62.038	62.137	0.2	35.902	33.324	-7.2	43.795	39.682	-9.4
Fast Ethernet	256	0.210	0.214	1.9	0.210	0.214	1.9	0.210	0.214	1.9
	512	0.489	0.492	0.6	0.489	0.492	0.6	0.489	0.492	0.6
	1024	1.436	1.347	-6.2	1.436	1.347	-6.2	1.436	1.347	-6.2
	2048	9.455	9.144	-3.3	6.426	5.988	-6.8	6.903	6.361	-7.9
	4096	68.101	67.129	-1.4	39.565	37.135	-6.1	47.826	43.875	-8.3

Times in seconds, errors in percentage

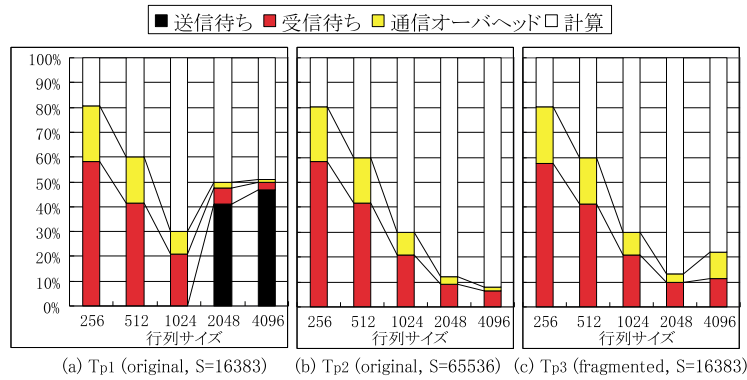


図 6 予測時間の内訳 (LogGPS, Myrinet)

Fig. 6 Breakdown of predicted times (LogGPS, Myrinet).

きは送信待ち時間が大きい。このプログラムでは送信するメッセージの長さが n とともに増加し、 $n = 1024$ のときの最長メッセージ長は $S (= 16383)$ をわずかに下回る 16320 バイトであった。したがって、 n が 1024 を超えると Rndv. プロトコルに基づく送信が多くなり、送信待ち時間が増加していた。

なお、コンテストで用いた MPI の実装 mini-MPI では $S = 128000$ である。したがって、コンテスト時には Rndv. プロトコルに基づく送信は生じず、送信待ち時間は性能ボトルネックにならなかったと考えられる。今回、LogGPS モデルに基づいて S の値が小さい実行環境のパラメータでプログラムを解析することで、潜在的な性能ボトルネックが露見したといえる。

このように、LogGPS モデルに基づいて MPI 通信命令の処理時間の内訳 (待ち時間および通信オーバーヘッド) を解析することで、プログラムの実行状況を詳細に調べることができる。

4.4 送信待ち時間の削減例

以下、4.3 節の性能ボトルネックを削減することを考える。送信待ち時間を削減する方法として、実現が

容易なものから順に 3 通りの方法を以下に示す。

- (1) MPI プログラムの実行時に S を大きな値へ変更。
- (2) メッセージを長さ S 以下に分割し、繰り返し送信。
- (3) アルゴリズムを変更。

(1) は MPI プログラムを実行するときに S の値を変更できる MPI の実装 (MPICH-SCore など) で実現でき、プログラムの修正を必要としない。(2) は送信命令および受信命令をメッセージ長に応じて繰り返し呼び出すようにプログラムを修正する必要がある。(3) は送信命令および受信命令の呼出時刻が大きくずれないアルゴリズムに基づいてプログラムを再実装する必要がある。以降では、(1) と (2) について述べる。

まず、 $n = 4096$ のときの最長メッセージ長をトレースデータから調べると 65472 バイトであった。そこで、 $S = 65536$ としてプログラムを修正することなくツールを用いて再び解析した。このときの実測および予測の結果を表 7 中欄および図 6 (b) に示す。

$n \leq 1024$ では S を変更した影響がないのに対し、 $n > 1024$ では T_{m1} を短縮できた (T_{m1} , T_{m2})。同様

表 8 LogGPS モデルと LogGP モデルの比較 (k : メッセージ長)Table 8 Comparisons between the LogGPS model and the LogGP model (k : message length).

項目	LogGP-1	LogGP-2	LogGPS
(D1) 同期のモデル化	同期コスト	同期コスト・待ち時間 (個別にモデル化)	同期コスト・待ち時間 (通信命令の呼出時刻に基づく)
(D2) 通信バンド幅	k ごとの定数	2 定数, 長さ s で区別	2 定数, 長さ s で区別
(D3) 通信オーバーヘッド	k ごとの定数, $P_s \cdot P_r$ で区別	2 定数, 長さ S で区別	k の線形関数, $P_s \cdot P_r$, 長さ S で区別

の結果が LogGPS モデルに基づく予測時間 T_{p1} および T_{p2} においても確認できる. T_{m1} を短縮できた理由は送信待ち時間を消去できたことにあり, $n = 4096$ のときには待ち時間の比率を 50%から 7%に削減できた (図 6(a), (b)).

次に, (2) について考える. 表 3 および表 5 より, メッセージ長が $S (= 16383)$ のときの Myrinet における通信時間は 298 マイクロ秒と算出できる. 一方, $n = 4096$ のときに長さが S を超えるメッセージは 1 プロセッサあたり計 22.5 MB であり, メッセージを分割すると計 1438 回の送受信を繰り返す必要がある. このときの通信時間は 0.43 秒と算出でき, これは $n = 4096$ のときの送信待ち時間 29.126 秒よりも十分小さい. したがって, メッセージを分割送信することでプログラムを改善できる見込みがある.

そこで, メッセージを長さ S 以下に分割, 繰り返し送信および受信するようにプログラムを修正した. その後, 修正済プログラムを再実行することでトレースデータを再生成しツールを用いて解析した. 実測および予測の結果を表 7 右欄および図 6(c) に示す.

なお, コンテストで用いた MPI の実装 mini-MPI では $S = 128000$ である. したがって, コンテスト時には Rndv. プロトコルに基づく送信は生じず, 送信待ち時間は性能ボトルネックにならなかったと考えられる. 今回, LogGPS モデルに基づいて S の値が小さい実行環境のパラメータでプログラムを解析することで, 潜在的な性能ボトルネックが露見したといえる.

このように, LogGPS モデルに基づいて MPI 通信命令の処理時間の内訳 (待ち時間および通信オーバーヘッド) を解析することで, プログラムの実行状況を詳細に調べることができる.

T_{m3} よりメッセージを分割送信することでも T_{m1} を短縮できた. 同様のことが T_{p3} についても確認できる. ここで, $n = 4096$ のときの σ_3 の値が他と比べて大きい. この理由は, LogGP モデル同様, LogGPS モデルでもメッセージをバッファに複製するときに遅延が発生しないと見なしているためである. (2) の場合, メッセージを分割送信することで非同期通信の実行回数が増加し, バッファの負荷が高まる. バッファ

表 9 Myrinet における LogGP パラメータ

Table 9 LogGP parameters for Myrinet.

Model	Parameter	Message length (Bytes)	
		≤ 8191	> 8192
LogGP-1	L (ns)	7.46×10^3	
	o (ns)	図 7(a), (b)	
	G (ns)	図 7(c)	
LogGP-2	L (ns)	7.00×10^3	
	o (ns)	3.83×10^3	72.88×10^3
	G (ns)	24.98	8.85

g は双方とも用いない.

が溢れるとメッセージ内容が壊れてしまうため, MPI の実装では送信命令の内部でバッファが溢れないようにメッセージを制御する. したがって, バッファ溢れを防ぐための待ち時間を考慮できず, $n = 4096$ のときの T_{p3} が T_{m3} よりもやや小さな値となった.

4.5 LogGP モデルとの比較

以下, LogGP モデルに基づく解析例を示し, LogGPS モデルと比較する. 比較のために用いたプログラムは, 4.3 節で用いた MPI プログラムである.

LogGP モデルにおける MPI 通信命令の処理時間の定義は文献 [12], [13] および [15] に基づいた. 以降では, 前者^{[12],[13]} を LogGP-1 モデル, 後者^[15] を LogGP-2 モデルと呼ぶ. 3 章で述べた (D1) ~ (D3) に着目して, 各モデルの相違点を表 8 にまとめる. なお, LogGP-1 モデルでは o および G はメッセージ長ごとに定める必要がある. さらに, o は MPI 通信命令ごとに定める.

まず, 両文献の方法に従い, Myrinet における両モデルのパラメータを計測した. 取得したパラメータを表 9 および図 7 に示す. 次に, ツールへ 4.3 節で用いたトレースデータを入力して各々の予測時間を算出した (表 10, 図 8). なお, LogGP-2 モデルでは待ち時間を数式から算出するが, より正確に解析するために LogGPS モデルと同様に MPI 通信命令の呼出時刻を基に待ち時間を算出した.

以下, 表 10 および図 8 について考察する. まず, $n = 4096$ のとき LogGP-1 モデルの誤差 σ_4 が大きい (-45.6%). この理由は, 同期コストのみを送信オーバーヘッド o_s に含め, o_s をメッセージ長ごとの

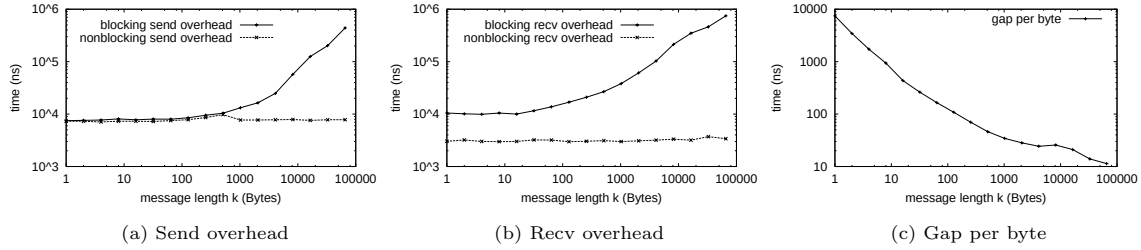


図7 Myrinet における LogGP-1 パラメータ
Fig. 7 LogGP-1 parameters for Myrinet.

表10 ガウスの消去法プログラムの予測時間 (LogGP, Myrinet)
Table 10 Predicted times for the Gaussian elimination program (LogGP, Myrinet).

matrix size n	original code, $S=16383$				original code, $S=65536$				fragmented code, $S=16383$			
	LogGP-1		LogGP-2		LogGP-1		LogGP-2		LogGP-1		LogGP-2	
	Pred. T_{p4}	Err. σ_4	Pred. T_{p7}	Err. σ_7	Pred. T_{p5}	Err. σ_5	Pred. T_{p8}	Err. σ_8	Pred. T_{p6}	Err. σ_6	Pred. T_{p9}	Err. σ_9
256	0.069	46.8	0.043	-8.5	0.069	46.8	0.043	-8.5	0.069	46.8	0.043	-8.5
512	0.165	36.4	0.110	-9.1	0.165	36.4	0.110	-9.1	0.165	36.4	0.110	-9.1
1024	0.668	9.2	0.551	-10.0	0.668	9.2	0.551	-10.0	0.668	9.2	0.551	-10.0
2048	4.528	-37.4	7.486	-2.4	4.528	8.9	4.225	-4.0	4.609	2.1	4.284	-5.0
4096	33.805	-45.6	61.591	-0.7	33.805	-5.8	32.764	-8.7	40.170	-8.3	39.129	-10.7

Times in seconds, errors in percentage.

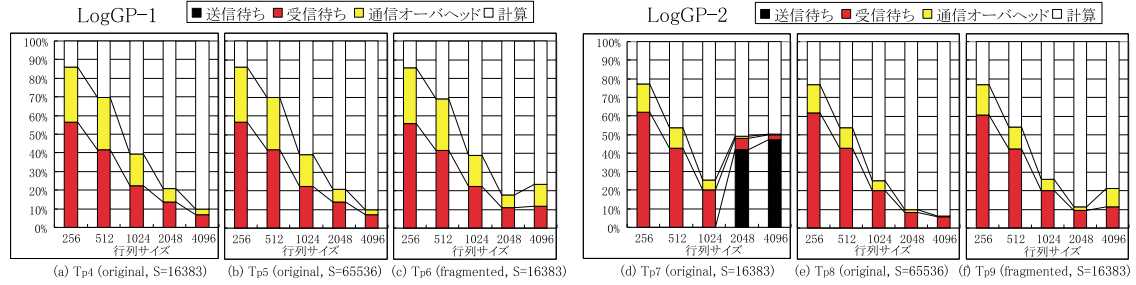


図8 予測時間の内訳 (LogGP, Myrinet)
Fig. 8 Breakdown of predicted times (LogGP, Myrinet).

定数としているためである。したがって、送信待ち時間が性能ボトルネックとなるプログラムに対しては、その性能ボトルネックを考慮できず、精度が悪化する(図8(a))。一方、LogGP-2モデルでは通信時間と別に送信待ち時間をモデル化するため、LogGPSモデル同様、誤差 σ_7 が小さい(-0.7%)。以上から、長いメッセージの通信時間を正確に見積もるためには、通信バンド幅の向上だけでなく、送信待ち時間を考慮する必要がある。

次に、 $n \leq 1024$ のとき LogGP-1 モデルの誤差 σ_4 は $n > 1024$ のときよりも大きい(最大46.8%)。この理由は、文献[13]のパラメータ値の取得方法にある。彼らは MPI 通信命令を1回呼び出すたびに時刻計測命令を呼び出している。この方法はメッセージ長が短いとき、通信命令の処理時間に対して計測オーバーヘッ

ドが大きくなる可能性があり、正確な計測とはいえない。たとえば、今回時刻計測に用いた MPI_Wtime を我々の PC クラスターで連続して呼び出したときの1命令あたりの処理時間は4マイクロ秒であり、LogGPSモデルにおける MPI_Send ($k=1$) の処理時間6.73マイクロ秒とほぼ同じであった。そこで、計測オーバーヘッドがより小さい、CPU のカウンタ (rdtsc 命令) を用いて LogGP-1 モデルのパラメータを再計測したが、誤差は最大27.7%であった。したがって、複数回の MPI 通信命令の呼び出しに対して計測する必要がある。一方、LogGP-2モデルの誤差 σ_7 は小さい(最大-10.0%)が、LogGPSモデルの誤差 σ_1 よりも少し大きい(最大6.4%)。LogGP-2モデルでは通信オーバーヘッド o を定数としているため、通信時間が実行時間に占める割合が大きいときは予測時間が実測時間よ

りも短くなる。以上から、短いメッセージの通信時間を正確に見積もるためには、通信オーバーヘッド ϕ をより正確に表す必要がある。

両モデルの違いは各パラメータ値の取得方法および MPI 通信命令の処理時間の定義方法のみであるため、これらの違いがモデルの予測精度に大きく影響することを確認できた。

5. ま と め

本稿では、高水準通信ライブラリを用いた並列プログラムを解析するための並列計算モデル LogGPS を提案した。LogGPS モデルでは、LogGP モデルに (D1)~(D3) の修正を加えることで、可能な限りライブラリ内部の実装を隠蔽でき、LogP モデルおよび LogGP モデルの低水準通信ライブラリに対する良い解析精度を高水準通信ライブラリに対しても実現できた。また、LogGPS モデルに基づいて連立方程式を解く MPI プログラムを解析した結果、実行環境によっては長いメッセージを送信するときの送信待ち時間が性能ボトルネックになりうることを確認できた。

今後の課題としては、非同期通信におけるバッファ溢れを防ぐための待ち時間のモデル化があげられる。

謝辞 本研究の一部は、日本学術振興会未来開拓学術研究推進事業 (JSPS-RFTF99I00903), PDC (並列・分散処理研究推進機構) および 栢森情報科学振興財団の補助による。並列計算機 Cenju-4 を利用させていただいた日本電気株式会社に感謝いたします。有益なご意見をいただいた査読者の方々に感謝いたします。

参 考 文 献

- 1) Culler, D., Karp, R., Patterson, D., Sahay, A., Schauer, K.E., Santos, E., Subramonian, R. and von Eicken, T.: LogP: Towards a Realistic Model of Parallel Computation, *Proc. 4th ACM SIGPLAN Symp. on Principles and Practice of Parallel Programming (PPoPP)*, pp.1-12 (1993).
- 2) Alexandrov, A., Ionescu, M.F., Schauer, K.E. and Scheiman, C.: LogGP: Incorporating Long Messages into the LogP Model—One, step closer towards a realistic model for parallel computation, *Proc. 7th Ann. ACM Symp. on Parallel Algorithms and Architectures (SPAA)*, pp.95-105 (1995).
- 3) von Eicken, T., Culler, D.E., Goldstein, S.C. and Schauer, K.E.: Active Messages: a Mechanism for Integrated Communication and Computation, *Proc. 19th Ann. International Symp. on Computer Architecture (ISCA)*, pp.256-266 (1992).
- 4) Homewood, M. and McLaren, M.: Meiko CS-2 Interconnect Elan-Elite Design, *Proc. IEEE Hot Interconnects '93 Symp.*, pp.95-105 (1993).
- 5) Dusseau, A.C., Culler, D.E., Schauer, K.E. and Martin, R.P.: Fast Parallel Sorting Under LogP: Experience with the CM-5, *IEEE Trans. Parallel and Distributed Systems*, Vol.7, No.8, pp.791-805 (1996).
- 6) Message Passing Interface Forum: MPI: A Message-Passing Interface Standard, *International Journal of Supercomputing Applications*, Vol.8, No.3/4 (1994).
- 7) IBM Corp.: IBM Parallel Environment for AIX, MPL Programming and Subroutine Reference, Document Number GC23-3893-00 (1995).
- 8) Gropp, W., Lusk, E., Doss, N. and Skjellum, A.: A High-Performance, Portable Implementation of the MPI Message Passing Interface Standard, *Parallel Computing*, Vol.22, No.6, pp.789-828 (1996).
<http://www.mcs.anl.gov/mpi/mpich/>.
- 9) O'Carroll, F., Tezuka, H., Hori, A. and Ishikawa, Y.: The Design and Implementation of Zero Copy MPI Using Commodity Hardware with a High Performance Network, *International Conference on Supercomputing '98 (SC)*, pp.243-250 (1998).
<http://pdswww.rwcp.or.jp/dist/score/>.
- 10) Burns, G., Daoud, R. and Vaigl, J.: LAM: An Open Cluster Environment for MPI, *Proc. Supercomputing Symposium '94 (SS)*, pp.379-386 (1994). <http://www.mpi.nd.edu/lam/>.
- 11) IBM Corp.: IBM Parallel Environment (PE). http://www.rs6000.ibm.com/software/sp_products/pe.html (2000).
- 12) Al-Tawil, K. and Moritz, C.A.: LogGP Quantified: The Case for MPI, *Proc. 7th IEEE International Symp. on High Performance Distributed Computing (HPDC)* (1998).
- 13) Moritz, C.A.: Cost Modeling and Analysis: Towards Optimal Resource Utilization in Parallel Computer Systems, Ph.D. Thesis, Royal Institute of Technology, Stockholm (1998). <http://www.cag.lcs.mit.edu/%7Eandras/phd.ps.gz>.
- 14) Kielmann, T., Bal, H.E. and Verstoep, K.: Fast Measurement of LogP Parameters for Message Passing Platforms, *Proc. 4th Workshop on Runtime Systems for Parallel Programming*, pp.1176-1183 (2000).
- 15) Sundaram-Stukel, D. and Vernon, M.K.: Predictive Analysis of a Wavefront Application Us-

ing LogGP, *Proc. 7th ACM SIGPLAN Symp. on Principles and Practices of Parallel Programming (PPoPP)*, pp.141–150 (1999).

- 16) 加納 健, 中村真章, 広瀬哲也, 細見岳生, 中田登志之: 並列計算機 Cenju-4 のユーザレベルネットワークインタフェース, *情報処理学会論文誌*, Vol.41, No.5, pp.1379–1389 (2000).
- 17) Boden, N.J., Cohen, D., Felderman, R.E., Kulawik, A.E., Seitz, C.L., Seizovic, J.N. and Su, W.-K.: Myrinet—A Gigabit-per-Second Local-Area Network, *IEEE Micro*, Vol.15, No.1, pp.29–36 (1995).
- 18) Culler, D.E., Liu, L.T., Martin, R.P. and Yoshikawa, C.O.: Assessing Fast Network Interfaces, *IEEE Micro*, Vol.16, No.1, pp.35–43 (1996).
- 19) O'Carroll, F., Tezuka, H., Hori, A. and Ishikawa, Y.: MPICH-PM: Design and Implementation of Zero Copy MPI for PM, Technical Report TR-97011, RWC (1998).
- 20) PSC95. <http://www.info.waseda.ac.jp/murao/ka/project/PSC95/> (1995).
- 21) 建部修見: SOFTWARE. <http://phase.etl.go.jp/%7Etatebe/software/index-j.html> (2000).

(平成 13 年 2 月 1 日受付)

(平成 13 年 5 月 30 日採録)



伊野 文彦 (学生会員)

平成 12 年大阪大学大学院基礎工学研究科博士前期課程修了。現在, 同大学院基礎工学研究科博士後期課程在学中。並列計算機のソフトウェア開発環境に関する研究に従事。



藤本 典幸 (正会員)

平成 4 年大阪大学基礎工学部情報工学科卒業。平成 6 年同大学院基礎工学研究科博士前期課程修了。平成 9 年同大学院基礎工学研究科博士後期課程単位取得退学。工学博士。現在, 同大学院基礎工学研究科助手。並列アルゴリズム, 並列言語の処理系・開発環境等に興味を持つ。



萩原 兼一 (正会員)

昭和 49 年大阪大学基礎工学部情報工学科卒業。昭和 54 年同大学院基礎工学研究科博士課程修了。工学博士。同大学基礎工学部助手, 講師, 助教授を経て, 平成 5 年奈良先端科学技術大学院大学教授。平成 6 年より大阪大学教授。平成 4~5 年文部省在外研究員 (米国メリーランド大学)。現在, 並列処理の基礎および応用に興味を持っている。