ハイパフォーマンス
コンピューティング 68－6
（1997．10．17）

# 実用並列計算機モデルLogPQによる並列アルゴリズムの動作解析

北陸先端科学技術大学院大学 情報科学研究科

當山 孝義　　堀口 進

近年，さまざまな商用並列計算機が開発され，幅広い用途に普及しつつある．並列プログラムの開発基盤として，現実の制約を考慮した実用並列計算モデルの研究が必要とされている．Cullerらは，実用並列計算モデルとして，通信レイテンシ，通信オーバヘッド，通信ギャップおよびプロセッサ数の4パラメータを導入したLogPモデルを提案した．しかし，並列計算機の通信路はさまざまな制約があり，プログラムの開発環境の観点では，LogPモデルは十分に実用的な計算モデルではない．たとえば，並列計算機の多くは，通信にバッファを用いて，通信遅延をできるだけ少なくすることができる．そこで筆者らは，LogPモデルの通信路にキューと4つのパラメータを追加したLogPQモデルを提案した．本論文では，並列計算機CM5を用いた実験により，LogPモデルとLogPQモデルの実用性を比較検討した．その結果，通信バッファにより通信遅延の隠蔽がモデル化でき，並列行列乗算を効率的に実行できることが分った．また，LogPQモデルは，並列計算機上での実行時間を，LogPモデルより正確に表すことができることが分った．

# Evaluation of Parallel Algorithms by Parallel Computation Model LogPQ

Graduate School of Infomation Science,
Japan Advanced Institute of Science and Technology

Takayoshi TOUYAMA *and* Susumu HORIGUCHI

The parallel computation model LogP takes account of latency $L$, overhead $o$, gap $g$, and the number of processors $P$ and is well known as a realistic parallel computational model. However, most of parallel computers has a lot of constraints for communication channels and LogP does not consider this communication constraints. Thus, we have proposed the LogPQ model which takes account of four queue parameters into the LogP model. This paper addresses performance comparisons between the LogP and LogPQ models by implementing parallel matrix multiplication algorithms on a parallel machine CM5. The experimental results confirm that the LogPQ model estimate more accurately parallel performances of algorithms than the LogP model.

## 1　はじめに

近年，さまざまな商用並列計算機が開発され，幅広い用途に普及しつつある．しかし，並列計算機のプログラム開発は，アルゴリズムの性能解析が難しく，経験的に行われているのが実情である．そこで，各種の並列計算機で効率的に実行可能な並列プログラムの開発環境や実用的な並列計算モデルが必要とされている．

今まで，実際の並列計算機ハードウェアの制約を考慮した各種の並列計算モデルが提案されている．しかし，現在の商用並列計算機の主流はメッセージパシング型計算機で，nCUBE2[1]やCM-5[2]は，プロセッサ間通信をメッセージの送受信で行うSend-Receive

モデルを用いている．J-Machine[3]は，各プロセッサがメッセージ受信に対し処理を行うMessage Driven Modelを用いる．

Eickenら[4]は，受理側プロセッサがメッセージ受信に対して割り込みによるハンドラ実行を行うActive Messageモデルを提案した．Cullerら[5]は，実用的な並列計算モデルとしてLogPモデルを提案した．しかし，並列プログラムの動作解析等の並列プログラム開発環境に組み入れるには，Active MessageモデルやLogPモデルの通信路制約は不十分である．実際の並列計算機では，通信メッセージサイズの通信遅延に与える影響を考慮する必要がある．また，従来考慮されなかった非同期通信は，通信路のバッファ動作をモデルに組み込むことにより，並列処理の動作解析が可能になる．これらを簡潔な形で並列計算モデルに入れることにより，実際の並列計算機で従来より効率的な並列プログラムの構築が可能になる．

筆者らは，通信路のバッファ動作を考慮した実用並列計算モデルLogPQを提案した．LogPQモデルは，LogPモデルの通信路にキューを付加し，その制約として3つのキューパラメータを追加した実用並列計算モデルである．LogPQモデルは，通信メッセージサイズの通信遅延に対する影響を考慮できる．また，通信路のバッファ動作のオーバーフロー制御を明示的にすることにより，実用的な通信処理の解析が可能である．

本論文では，並列計算機CM5を用いた実験により，LogPモデルとLogPQモデルの実用性を比較検討した．その結果，通信バッファにより，通信遅延の隠蔽がモデル化でき，並列行列乗算を効率的に実行できることが分かった．また，LogPQモデルは，並列計算機上での実行時間を，LogPモデルより正確に表すことができることが分かった．

# 2 LogPとLogPQモデル

## 2.1 LogPモデル

LogPモデルは，分散メモリ型並列計算機を主な対象としたモデルであり，各プロセッサはメッセージ通信によりコミュニケーションを行う．LogPモデルは，並列計算機の物理的制約を，通信レイテンシ$L^*$，通信オーバヘッド$o^*$，通信ギャップ$g^*$およびプロセッサ数$P$の4パラメータで表す．図1に，LogPモデルのメッセージ通信動作を示す．$L^*$はメッセージ通信の送信から受信までの通信遅延の上限，$o^*$は各プロセッサにおけるメッセージ送受信処理のオーバヘッド，$g^*$は各プロセッサにおけるメッセージ送受信の最小間隔である．パラメータ$L^*$，$o^*$，$g^*$の値は，プロセッサの局所命令の実行時間を1クロックとして，クロックを単位として表す．

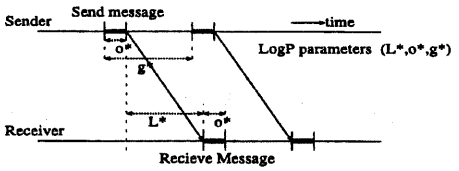



図1: Communication behavior in the LogP model.

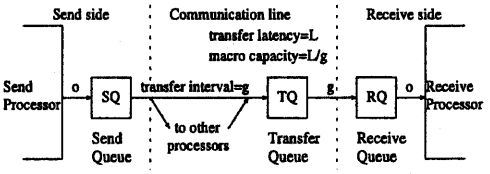



図2: A structure of LogPQ model.

## 2.2 LogPQモデル

LogPQモデルは，LogPモデルに通信路のバッファ動作を付加したものである．図2に，LogPQモデルの構造を示す．LogPQモデルは，各プロセッサの送信部に送信キューを，受信部に受信キューを，受信部直前の集結点に通信キューを付加する．そして，パラメータとして，各キューサイズの上限を与える．すなわち，送信キュー，受信キュー，通信キュー各々のサイズの上限を$SQ$，$RQ$，$TQ$とする．

## 2.3 LogPQモデルにおけるメッセージ通信

図3に，通信メッセージとLogPQモデルのパラメータの関係を示す．商用並列計算機は，プロセッサ間通信に固定ワード長のパケットを用いる場合が多い．LogPQモデルは通信を固定サイズのメッセージで表す．通信路の物理的制約を，LogPモデルの1ワード通信のパラメータ$L_0$，$o_0$，$g_0$で表したと

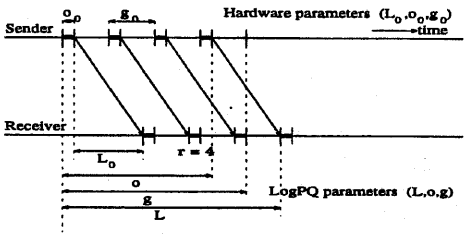



図 3: Message passing in LogPQ model.

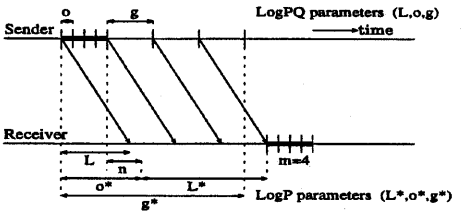



図 4: Relation between LogP and LogPQ.

き，LogPQ パラメータ $L$，$o$，$g$は次式で表される．

$$\begin{aligned} L &= L_0 + o_0 + (r-1)\cdot g_0 \\ o &= o_0 + (r-1)\cdot g_0 \\ g &= r \cdot g_0 \end{aligned} \qquad (1)$$

但し，ワード長の比 $r$は，通信路のワード長 $w_c$をプロセッサのワード長 $w_p$で割った値である．

## 2.4 LogP と LogPQ モデルの関係

図 4に，LogP モデルと LogPQ モデルにおける $m$ ワードのメッセージ通信を示す．$m$ ワードのメッセージ通信に対する LogP パラメータを $L^*$，$o^*$，$g^*$ とする．LogPQ モデルは，$m$ ワードのメッセージの通信を，1 ワードのメッセージ通信を $m$ 回実行することにより表す．図 4の $L$，$o$，$g$は LogPQ パラメータである．さらに，LogP モデルの通信オーバヘッドの固定成分に対応する値として，LogPQ にパラメータ $n$ を付加する．LogP パラメータ $L^*$，$o^*$，$g^*$ と LogPQ パラメータ $L$，$o$，$g$，$n$ の関係を，次式で表す．

$$\begin{aligned} L &= L^* + o^* \\ o &= (o^* - n)/m \\ g &= g^*/m \end{aligned} \qquad (2)$$

# 3 並列行列乗算による実験的評価

## 3.1 Cannon のアルゴリズム

Cannon アルゴリズム [7] は，行列 $A$ と $B$の積 $C = A \cdot B$を求める並列行列乗算アルゴリズムである．行列サイズを $N \times N$，プロセッサ数を $P = p^2$ とする．なお，行列サイズ $N$は $p$ の倍数とする．先ず，プロセッサをサイズ $p \times p$ の行列に配置し，$P_{ij}$ $(0 \le i, j < p)$ と表す．そして，各プロセッサ $P_{ij}$に，部分行列 $A_{ij}$，$B_{ij}$，$C_{ij}$を割り当てる．各 $A_{ij}$，$B_{ij}$，$C_{ij}$は，それぞれ行列 $A$，$B$，$C$の $i$ 行目で $j$列目のブロックである．各ブロックのサイズは $l \times l$，但し $l = N/p$ である．積 $C$は以下のように算出される．

1. 各プロセッサ $P_{ij}$は，$A_{ij}$を $P_{i((j-i) \bmod p)}$に，$B_{ij}$を $P_{((i-j) \bmod p)j}$に送信
2. 各プロセッサ $P_{ij}$は，$C_{ij}$に $(A_{ij} \cdot B_{ij})$ を加える
3. 以下を $(p-1)$ 回繰り返す．
   (a) 各プロセッサ $P_{ij}$は，$A_{ij}$を $P_{i((j-1) \bmod p)}$に，$B_{ij}$を $P_{((i-1) \bmod p)j}$に送信
   (b) 各プロセッサ $P_{ij}$は，$C_{ij}$に $(A_{ij} \cdot B_{ij})$ を加える

このアルゴリズムの計算時間は $O(N^3/P)$ であり，通信時間は $O(N^2/P^{1/2})$ である．

## 3.2 プロセッサ間通信

Cannon の並列アルゴリズムを C 言語を用いて並列計算機 CM5 に実装した．プロセッサ間通信は，CM5 の通信ライブラリ CMMD[2] の非同期通信命令を用いる．非同期通信命令は，以下のように用いられる．

1. 非同期メッセージの送信/受信コマンドを呼び出す Message Control Block（MCB）が確保され，そのメッセージの状態が保存される
2. MCB をチェックし，メッセージ通信の完了を待つ
3. MCB を解放する
   これ以降，CMMD はこの MCB を再利用できる
4. 受信したメッセージを利用する

並列アルゴリズムにプロセッサ間通信方式が及ぼす影響を解析するため，単純通信方式と隠蔽通信方式を実装した．図 5に，各種通信方式を用いた Cannon アルゴリズムの実装を示す．なお，Cannon アルゴリズムの各部分行列のサイズを $r = l^2$ とする．このとき，アルゴリズムにおける一回の繰り返し処理で，$2r$ ワードのデータが送受信される．

| [Source processor] | [Destination Processor] |
|---|---|
| loop: | loop: |
| Call send command | Call receive command |
| Wait until send finishes | Wait until receive finishes |
| Free its MCB | Free its MCB |
| Exec. the computation | Exec. the computation |
| goto loop | goto loop |

(a)Simple communication.

| [Source processor] | [Destination Processor] |
|---|---|
| loop: | loop: |
| Call send command | Call receive command |
| Exec. the computation | Exec. the computation |
| Wait until send finishes | Wait until receive finishes |
| Free its MCB | Free its MCB |
| goto loop | goto loop |

(b)Hidden communication.

| [Source processor] | [Destination Processor] |
|---|---|
| | Call plural receive |
| loop: | commands in advance |
| Call send command | loop: |
| Exec. the computation | Exec. the computation |
| Wait until send finishes | Wait until receive finishes |
| Free its MCB | Free its MCB |
| goto loop | goto loop |

(c)Hidden with using the receive buffer.

| [Source processor] | [Destination Processor] |
|---|---|
| loop: | loop: |
| Call send command | Call receive command |
| Exec. the computation | Exec. the computation |
| goto loop | Wait until receive finishes |
| Wait until all sends finish | Free its MCB |
| Free all MCB | goto loop |

(d)Hidden with using the send buffer.

図 5: Strategies of communication in Cannon's algorithm.

**(a) 単純通信方式**
単純通信は，通信処理が終了するまで計算処理を実行しない．この場合，通信処理による遅延は，並列処理全体の実行時間に直接影響する．一方，通信処理がハンドシェイクで制御されるので，各プロセッサは $2r$ ワード長の送受信バッファを持てばよい．

**(b) 隠蔽通信方式**
隠蔽通信は，通信処理の実行中に計算処理を行い，通信遅延を隠蔽する．この場合もハンドシェイクにより，通信に必要な送受信バッファサイズは $2r$ ワードでよい．しかしこのとき，二つの遅延が発生する可能性がある．先ず，各プロセッサはメッセージ送信の前に以前の送信の完了を待たねばならない．また，各プロセッサは他プロセッサと非同期的に動作するので，受信側プロセッサの開始が遅ければ送信側プロセッサは送信を待たされる．

**(c) 受信バッファ付加**
隠蔽通信方式は，受信バッファの付加により，受信側プロセッサ開始遅れによる遅延を隠蔽できる．

**(d) 送信バッファ付加**
隠蔽通信方式は，送信バッファの付加により，以前の送信の完了待ちによる遅延を隠蔽できる．

**(e) 送受信バッファ付加**
隠蔽通信方式は，図 5(c)(d) を合わせた送受信バッファの付加により，上記の両方の遅延を隠蔽できる．

### 3.3 通信遅延と実行時間

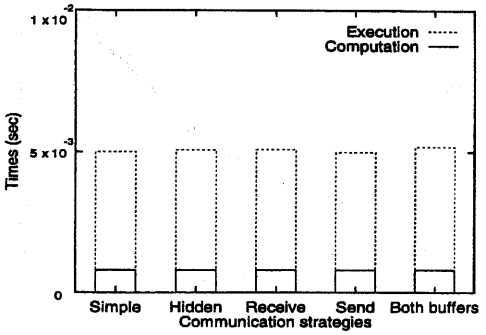
図 6に，CM5 の 64 プロセッサを用いた Cannon の並列乗算アルゴリズムの実行時間と通信方式の関係を示す．図 6(a)，(b)，(c) は，それぞれ行列サイズ $N = 8$，64，1024 に対する，単純通信，隠蔽通信，受信バッファ付加，送信バッファ付加，送受信バッファ付加を用いた場合の実行時間を示す．図の波枠部は実行時間，実枠部は計算処理時間である．通信遅延時間は，この両時間の差で表される．

行列サイズが小さい場合は，通信遅延時間が計算処理時間より大きいため，通信は隠蔽できない．一方，行列サイズが大きい場合は以下のようになる．単純通信の通信遅延時間は，計算処理時間より大きく，並列処理性能は小さくなる．隠蔽通信は，遅延を少なくできるが，行列サイズが大きくなるに従い増加するため，その効果は限定される．隠蔽通信に受信バッファを付加すると，遅延をさらに小さくすることができる．また，送信バッファを付加すると，行列サイズが大きな場合に遅延はほぼ隠蔽され，実行時間は計算処理時間にほぼ等しくなる．送受信双方のバッファを付加した場合，行列サイズが大きな場合に通信遅延は完全に隠蔽され，誤差の範囲内程度になる．これらの実験結果から，$4r$ ワード長の送受信バッファがあれば，CM5 上で Cannon アルゴリズムを効率的に実行できることが分かる．

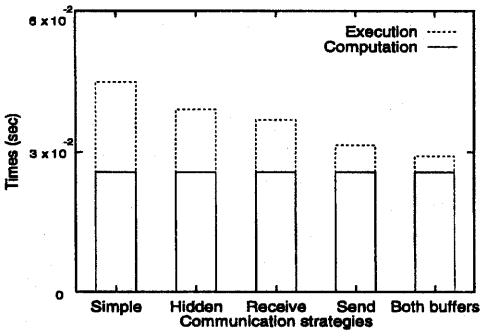
## 4 LogP と LogPQ モデルによる解析

### 4.1 並列処理時間の解析

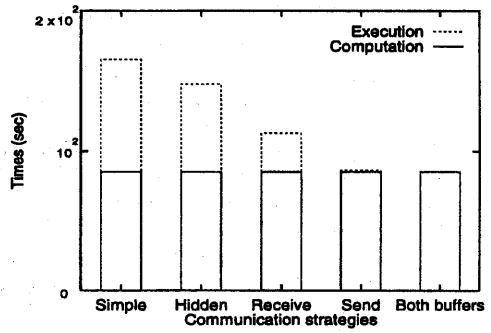
Cannon アルゴリズムの送受信バッファ付き隠蔽通信を用いて行う並列アルゴリズムを，LogP と LogPQ モデルにより解析する．




(a) Matrix size $N = 8$.




(b) Matrix size $N = 64$.




(c) Matrix size $N = 1024$.

図 6: Computation and execution times of Cannon's algorithm by CM5 with 64 PEs.

Cannon アルゴリズムの並列処理時間 $T$ について考える．サイズ $l$ の部分行列乗算一回の実行時間 $t_M$ は，

$$t_M = t_f + t_s \cdot l^3 \tag{3}$$

で与えられる．ここで，$t_f$ は計算処理時間の固定部分，$t_s$ は一回のスカラ乗算の（周辺処理を含めた）実行時間である．計算処理全体の実行時間 $t_c$ は，

$$t_c = p \cdot t_M. \tag{4}$$

したがって，並列処理時間 $T$ は，

$$T = t_c + p \cdot t_{com} \tag{5}$$

となる．ここで，$t_{com}$ は通信処理の実行時間である．

並列処理時間を LogP と LogPQ モデルにより解析する．LogP モデル（1 メッセージを用いる場合）では，各プロセッサは各部分行列 $A_{ij}$，$B_{ij}$ を 1 個の $r = l^2$ ワード長メッセージで送信する．このとき，通信処理の実行時間 $t_{LogP_1}$ は，次式で表される．

$$\begin{aligned} t_{LogP_1} &= o^* + 3max(o^*, g^*) \quad (t_M \geq L^*) \\ &= (L^* - ot) + (o^* + 3max(o^*, g^*)) \\ &\qquad \text{(otherwise).} \end{aligned} \tag{6}$$

LogP モデル（1 ワード長メッセージを用いる場合）では，各プロセッサが各部分行列を $r$ 個の 1 ワード長メッセージで送信する．このとき，通信処理の実行時間 $t_{LogP_r}$ は，次式で表される．

$$\begin{aligned} t_{LogP_r} &= o_0 + max(o_0, g_0) \cdot (4r - 1) \\ &\qquad (t_M \geq (L_0 - (2r - 1) \cdot max(o_0, g_0))) \\ &= (L_0 - (2r - 1) \cdot max(o_0, g_0) - ot) \\ &\quad + (o_0 + max(o_0, g_0) \cdot (4r - 1)) \\ &\qquad \text{(otherwise).} \end{aligned} \tag{7}$$

LogPQ モデルでは，各プロセッサは各部分行列を $r$ 個の 1 ワード長メッセージで送信する．このとき，通信処理の実行時間 $t_{LogPQ}$ は，次式で表される．

$$\begin{aligned} t_{LogPQ} &= 4(o \cdot r + n) \\ &\qquad (t_M \geq (L + (2r - 1)g - 2r \cdot o - n)) \\ &= (L + (2r - 1)g - 2r \cdot o - n - ot) \\ &\quad + 4(o \cdot r + n) \qquad \text{(otherwise).} \end{aligned} \tag{8}$$

## 4.2 LogP と LogPQ パラメータ

並列計算機 CM5 上での各実行時間 $t_s$，$t_f$ を測定した結果，$t_s = 5.98 \cdot 10^{-6}(sec)$，$t_f = 1.55 \cdot 10^{-4}(sec)$ となる．

| LogP (1 messgae) | LogP (1-word length) | LogPQ |
|---|---|---|
| $L^* = 1.25 \cdot 10^{-4}$ | $L_0 = 1.77 \cdot 10^{-4}$ | $L = 2.96 \cdot 10^{-4}$ |
| $o^* = 1.71 \cdot 10^{-4}$ | $o_0 = 1.19 \cdot 10^{-5}$ | $o = 4.59 \cdot 10^{-6}$ |
| $g^* = 5.50 \cdot 10^{-5}$ | $g_0 = 3.44 \cdot 10^{-6}$ | $g = 3.44 \cdot 10^{-6}$ |
| | | $n = 9.75 \cdot 10^{-5}$ |

(sec)

表 1: LogP and LogPQ parameters of CM5.

また，CM5 に対する LogP と LogPQ モデルのパラメータを表1に示す．ここでは，通信パラメータをクロックでなく秒で表す．通信パラメータの基準点として，部分行列サイズ $l = 4$ の場合，すなわち 16 ワード長メッセージに対する実験により，LogP および LogPQ パラメータを求めた．

### 4.3 LogP と LogPQ モデルの比較

図7に，並列計算機 CM5 上で Cannon アルゴリズムを実行した並列処理時間と，LogP と LogPQ モデルにより求めた並列処理時間を示す．

LogP モデル（1 メッセージ）による解析結果は，$N < 32$ では実際より大きく，$N > 32$ ではその逆になる．LogP モデル（1 ワード長メッセージ）では，通信路のバッファリングとパラメータ $n$ を考慮しないため，LogP モデル（1 メッセージ）と逆の結果になる．一方，LogPQ モデルは，LogP モデルに比べ実際の並列処理時間に近い結果を得ることができる．

## 5 まとめ

本論文では，通信バッファを考慮した実用的な並列計算モデル LogPQ について述べた．並列行列乗算アルゴリズムを並列計算機 CM5 上で実行し，従来の LogP モデルより LogPQ モデルが実用性が高いことを示した．また，通信にバッファを用いることにより，アルゴリズムの通信遅延をより隠蔽できることを明らかにし，並列アルゴリズムの効率を LogPQ モデルを用いて検討することができることを示した．

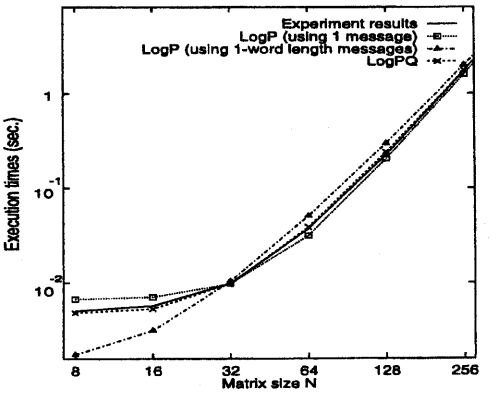



図 7: Execution times and analytical times of the LogP and LogPQ models for Cannon's matrix multiplication as a function of submatrix size on 64 PEs of CM5.

## 参考文献


[1] "nCUBE2: Technical Overview", nCUBE Corporation (1992).

[2] "Connection Machine CM-5 Technical Summary ", Thinking Machines Corporation (1992).

[3] William J.Dally, Andrew Chien, Stuart Fiske, Waldemar Horwat, John Keen, Michael Larivee, Rich Lethin, Peter Nuth and Scott Wills, "The J-Machine: A Fine-Grain Concurrent Computer", Information Processing 89, pp.1147-1153 (1989).

[4] Thorsten von Eicken, David E.Culler, Seth Copen Goldstein and Klaus Erik Schauseret, "Active Messages: a Mechanism for Integrated Communication and Computation", Proceeding of the 19th Annual International Symposium on Computer Architecture pp.256-266 (1992).

[5] David Culler, Richard Karp, David Pattersom, Abhijit Sahay, Klaus Erik Schauser, Eunice Santos, Ramesh Subramonian and Thorsten von Eicken, "LogP: Towards a Realistic Model of Parallel Computation", Proceeding of the 4th ACM SIGPLAN Symposium on Principles and Parallel Programming (1993).

[6] Takayoshi Touyama and Susumu Horiguchi, "Evaluation of Parallel Computer Model LogPQ (in Japanese)", IPSJ SIG Notes, 94-ARC-109-8, pp.57-64 (1994).

[7] Vipin Kumar, Ananth Grama, Anshul Gupta and George Karypis, "Introduction to Parallel Computing", Benjamin/Cummings Publishing (1994).