行政院國家科學委員會專題研究計畫　成果報告

異質叢集中的同步通訊近似演算法

計畫類別：個別型計畫
計畫編號：NSC93-2213-E-002-115-
執行期間：93 年 08 月 01 日至 94 年 07 月 31 日
執行單位：國立臺灣大學資訊工程學系暨研究所

計畫主持人：劉邦鋒

計畫參與人員：莊中豪，張瑞賢

報告類型：精簡報告
報告附件：出席國際會議研究心得報告及發表論文
處理方式：本計畫可公開查詢

中　華　民　國 94 年 12 月 26 日

附件一

行政院國家科學委員會補助專題研究計畫 ■ 成 果 報 告 □期中進度報告

異質叢集中的同步通訊近似演算法

計畫類別：■ 個別型計畫　□ 整合型計畫
計畫編號：NSC　93-2213-E-002-115-
執行期間：　93 年 8 月 1 日至　94 年 7 月 31 日

計畫主持人：劉邦鋒
共同主持人：
計畫參與人員：莊中豪, 張瑞賢

成果報告類型(依經費核定清單規定繳交)：■精簡報告　□完整報告

本成果報告包括以下應繳交之附件：
□赴國外出差或研習心得報告一份
□赴大陸地區出差或研習心得報告一份
□出席國際學術會議心得報告及發表之論文各一份
□國際合作研究計畫國外研究報告書一份

處理方式：除產學合作研究計畫、提升產業技術及人才培育研究計畫、列管計畫及下列情形者外，得立即公開查詢
□涉及專利或其他智慧財產權，□一年□二年後可公開查詢

執行單位：國立臺灣大學資訊工程學系暨研究所

中　華　民　國　94　年　10　月　31　日

An Approximation Algorithm for Broadcast Scheduling in Asynchronous Heterogeneous Clusters

Abstract

Network of workstation (NOW) is a cost-effective alternative to massively parallel supercomputers. However, a cluster may consist of different types of processors and this heterogeneity complicates the design of efficient collective communication protocols. This paper shows that a simple fastest-node-first (FNF) heuristic is very effective in reducing broadcast time for heterogeneous cluster systems. Despite the fact that FNF heuristic does not guarantee optimal time, we prove that FNF always gives near optimal broadcast time for a special case of cluster, and guarantees performance for general clusters. We show that FNF gives a total broadcast time of 2T + h, where T is the optimum time and h is a constant. This improves over the previous bound on 2gT +h, where g is a theoretically unbounded ratio of processor speed. We also describe the experimental results in which we compare the completion time of FNF with the optimal solutions found by an exhaustive search. The experimental results indicate that FNF gives solutions that are most of the time within 10% of the optimum, and with a probability up to 0.65 FNF actually finds the optimal solution. Our theoretical results also improve the efficiency of the exhaustive search by 33%.

Keyword

heterogeneous cluster systems, data broadcast, scheduling.

摘要

工作站叢集是一種有效的計算平台. 一個工作站叢集中可能包含不同種類的工作站, 所以如何在這些異質工作站之間快速傳布資料就成爲工作站叢集效能的重要因素之一. 本計畫成果顯示一簡單的'快速者優先'方法可有效減少異質工作站叢集中資料由一部工作站傳布至其餘所有工作站的總時間. 如果最佳總傳布時間爲T, 則'快速者優先'方法可保證在2T+h的時間內將資料由一部工作站傳布至其餘所有工作站. 這比過去文獻中的分析結果更爲精準, 而且根據實驗顯示, 有百分之六十五的時候'快速者優先'方法可達成最佳總傳布時間.

關鍵字

異質工作站叢集, 資料傳布, 排程

1 Introduction

Network of workstation (NOW) is a cost-effective alternative to massively parallel supercomputers [1]. As commercially available off-the-shelf processors become cheaper and faster, it is now possible to build a PC or workstation cluster that provides high computing power within a limited budget. High performance parallelism is achieved by dividing the computation into manageable subtasks, and distributing these subtasks to the processors within the cluster. These off-the-shelf high-performance processors provide a much higher performance-to-cost ratio so that high performance clusters can be built inexpensively. In addition, the processors can be conveniently connected by industry standard network components. For example, Fast Ethernet technology provides up to 100 Mega bits per second of bandwidth with inexpensive Fast Ethernet adaptors and hubs.

Parallel to the development of inexpensive and standardized hardware components for

NOW, system software for programming on NOW is also advancing rapidly. For example, the Message Passing Interface (MPI) library has evolved into a standard for writing message-passing parallel codes [9, 8, 13]. An MPI programmer uses a standardized high-level programming interface to exchange information among processes, instead of native machine-specific communication libraries. An MPI programmer can write highly portable parallel codes and run them on any parallel machine (including network of workstation) that has MPI implementation.
Most of the literature on cluster computing emphasizes on homogeneous cluster – a cluster consisting of the same type of processors. However, we argue that heterogeneity is one of the key issues that must be addressed in improving parallel performance of NOW. Firstly, it is always the case that one wishes to connect as many processors as possible into a cluster to increase parallelism and reduce execution time. Despite the increased computing power, the scheduling management of such a heterogeneous network of workstation (HNOW) becomes complicated since these processors will have different performances in computation and communication. Secondly, since most of the processors that are used to build a cluster are commercially off-the-shelf products, they will very likely be outdated by faster successors before they become unusable. Very often a cluster consists of "leftovers" from the previous installation, and "new comers" that are recently purchased. The issue of heterogeneity is both scientific and economic.
Every workstation cluster, be it homogeneous or heterogeneous, requires efficient collective communication [2]. For example, a barrier synchronization is often placed between two successive phases of computation to make sure that all processors finish the first phase before any can go to the next phase. In addition, a scatter operation distributes input data from the source to all the other processors for parallel processing, then a global reduction operation combines the partial solutions obtained from individual processors into the final answer. The 2 efficiency of these collective communications will affect the overall performance, sometimes dramatically. Heterogeneity of a cluster complicates the design of efficient collective communication protocols. When the processors send and receive messages at different rates, it is difficult to synchronize them so that the message can arrive at the right processor at the right time for maximum communication throughput. On the other hand, in homogeneous NOW every processor requires the same amount of time to transmit a message. For example, it is straightforward to implement a broadcast operation as a series of sending and receiving messages, and in each phase we double the number of processors that have received the broadcast message. In a heterogeneous environment it is no longer clear how we should proceed to complete the same task.
This paper shows that a simple heuristic called fastest-node-first (FNF), introduced by Banikazemi et. al. [2], is very effective in designing broadcast protocols for heterogeneous cluster systems. The fastest-node-first technique schedules the processors to receive the broadcast in the order of their communication speed, that is, the faster node should be scheduled earlier. Despite the fact that the FNF heuristic does not guarantee optimal broadcast time for every heterogeneous network of workstations, we show that FNF does give near optimal broadcast time when the communication time of any slower processor in the cluster is a multiple of any faster processor. Based on this result, we show that FNF is actually an approximation algorithm that guarantees a broadcast time within $2T + t$, where T is the optimal broadcast time and t is the maximum difference between two processors. This improves over the previous bound $2gT + h$ [17] where g

is the maximum ratio between receiving and sending costs, and can be arbitrarily large theoretically. In a previous paper [19] we show a similar result for a communication model where the communication cost is determined by the sender only. This paper shows that FNF can still achieve guaranteed performance when the model determines the communication costs based on both the sender and the receiver. We also conduct experiments on the performance of the fastest-node-first technique. The cluster we construct in our simulation consists of three types of processors, and the number of nodes is 100. We construct the schedules from a random selection and FNF, and apply them on the heterogeneous cluster model. Experimental results indicate that FNF gives superior performance over random selection, for up to 2 times of throughput. This simulation result indicates that FNF is very effective in reducing broadcast time in practice.

2 Communication Model

There have been two classes of models for collective communication in homogeneous cluster environments. The first group of models assumes that all the processors are fully connected. As a result it takes the same amount of time for a processor to send a message to any other processor. For example, both the Postal model [5] and LogP model [15] use a set of parameters to capture the communication costs. In addition the Postal and LogP model assume that the sender can engage in other activities after a fixed startup cost, during which the sender injects the message into the network and is ready for the next message. Optimal broadcast scheduling for these homogeneous models can be found in [5, 15]. The second group of models assume that the processors are connected by an arbitrary network. It has been shown that even when every edge has a unit communication cost (denoted as the Telephone model), finding an optimal broadcast schedule remains NP-hard [10]. Efficient algorithms and network topologies for other similar problems related to broadcast, including multiple broadcast, gossiping and reduction, can be found in [7, 11, 12, 14, 18, 21, 22, 23].

Various models for heterogeneous environments have also been proposed in the literature. Bar-Nod et al. introduced a heterogeneous postal model [4] in which the communication costs among links are not uniform. In addition, the sender may engage another communication before the current one is finished, just like homogeneous postal and LogP model. An approximation algorithm for multicast is given, with a competitive ratio logk where k is the number of destination of the multicast [4]. Banikazemi et al. [2] proposed a simple model in which the heterogeneity among processors is characterized by the speed of sending processors, and show that a broadcast technique called fastest-node-first works well in practice. We will refer to this model as the sender-only model. Based on the sender-only model, an approximation algorithm for reduction with competitive ratio 2 is reported in [20], and the fastest-node-first technique is shown to be also 2-competitive [19]. Despite the fact that the sender-only model is simple and has a high level abstraction of network topology, the speed of the receiving processor is not accounted for. In a refined model proposed by Banikazemi et al. [3], communication overheads consists of both sending and receiving time, which we will refer to as the sender-receiver model. For the sender-receiver model the same fastest- node-first is proven (Libeskind-Hadas and Hartline [17]) to have a total time of no more than $2hT + g$, where h is the maximum ratio between receiving and sending time, g is the maximum difference between two receiving time, and T is the optimal time. We adopt the sender- receiver model in this paper and improve this

bound to 2T + g. Other models for heterogeneous clusters include [6, 16].

2.1 Model Definition

The model is defined as follows: A heterogeneous cluster is defined as a collection of processors $p_0, p_1, ..., p_{n-1}$, each capable of point-to-point communication with any other processor in the cluster. Each processor is characterized by its speed of sending and receiving messages, and the network is characterized by the speed to route a message from the source to the destination. Formally, we define the sending time of a processor p, denoted by s(p), to be the time it needs for p to send a unit of message into the network. The network is characterized by its latency L, which is the time for the message to go from its source to its destination. Finally we define the receiving time of a processor p, denoted by r(p), to be the time it takes for p to retrieve the message from the network interface. We further assume that the processor speed is consistent – if a processor p can send messages faster than another processor q, it can also receive the messages faster. Formally we assume that for two processors p and q, $s(p) \leq s(q)$ if and only if $r(p) \leq r(q)$. The communication model dictates that the sender and receiver processors cannot engage in multiple message transmissions simultaneously – a sender processor must complete its data transmission to the network before sending the next message. Also a processor can only inject messages into the network at an interval specified by its sending time. This restriction is due to the fact that processor and communication networks have limited bandwidth, therefore we would like to exclude from our model the unrealistic algorithms in which a processor simply sends the broadcast message to all the other processors simultaneously. Similarly, the model prohibits the simultaneous receiving of multiple messages by any processor. That is, the model disallows the unrealistic implementation of a reduction operation by having one processor to receive the messages from all the other processors simultaneously. Many message passing libraries provide non-blocking send and receive primitives, but these simultaneous message transmissions are eventually serialized in the hardware level.

The communication model allows asynchronous send. In asynchronous message passing the sender only needs to wait for its sending time before initiating the next send. The sender can immediately start preparing the next transmission without waiting for the receiver to complete. This asynchronous send has been implemented in various communication libraries, including MPI, and it is consistent with the behaviors of communication hardware.

2.2 Simplified Model Description

We can simplify the model as follows: Since a receiving node p always has to wait for L+r(p) time steps before it actually receives the message, we add the network latency L into its receiving time. As a result we simply state that the new receiving time of p_1 in the previous example is 2. The processor p_1 therefore receives its message at time $s(p_0)+r(p_1) = 1+2 = 3$, and p_3 receives its message from p_0 at time $2s(p_0)+r(p_3) = 8$. From now on we will use this simplified model.

After simplifying the communication model, we can define other terminologies for the broadcast problem in a heterogeneous system. Assume that a processor q sends a message to another processor p at time t, then p becomes ready to receive at time t + s(q), since p now can start receiving the message, and we denote the ready to receive time of p by R(p). At time t + s(q) +

r(p) p becomes ready to send because it can start sending its own message now, and we use S(p) to denote the ready to send time of p. A processor p can finish sending messages into the network at time S(p) + s(p), S(p) + 2s(p), ..., S(p) + i * s(p), where i is a positive integer, until the broadcast finishes.

3 Fastest-node-first Technique

It is difficult to find the optimal broadcast tree that minimizes the total broadcast time in a heterogeneous cluster, therefore a simple heuristic called fastest-node-first (FNF) is proposed in [2] to find a reasonably good broadcast schedule for the original sender-only heterogeneous model [2].

3.1 Fastest-Node-First Scheduling for Broadcast

The FNF heuristic works as follows: In each iteration the algorithm chooses a sender from the set of processors that have received the broadcast message (denoted by A), and a receiver from the set that have not (denoted by B). The algorithm picks the sender s from A because, as the chosen one, it can inject the message into the network as early as possible. The algorithm then chooses the fastest processor in B as the destination of s. That is, r is the processor that has the minimum sending time in B. After the assignment, r is moved from B to A and the algorithm iterates to find the next sender/receiver pair. Note that this same technique can be applied to both models – the sender only and the sender-receiver heterogeneous models – since we assume that the sending and receiving times are consistent among processors. The intuition behind this heuristic is that, by sending the message to those fast processors first, it is likely that the messages will propagate more rapidly.

The fastest-node-first technique is very effective in reducing broadcast time [2, 17, 19]. The FNF has been shown in simulation to have a high probability to find the optimal broadcast time when the transmission time is randomly chosen from a given table [2]. The FNF technique also delivers good communication efficiency in actual experiments. In addition, FNF is simple to implement and easy to compute.

3.2 FNF Does not Guarantee Optimal Broadcast Time

Despite its efficiency in scheduling broadcast in heterogeneous systems, the FNF heuristic does not guarantee optimal broadcast time [2, 6] in sender-only model. Since the sender-only model is a special case of the sender-receiver model, FNF is not optimal in the sender-receiver model either. For example, in the situation of Figure 1 FNF will not achieve optimal time,

4 Theoretical Results

Despite the fact that FNF cannot guarantee optimal broadcast time, we show that FNF is optimal in some special cases of heterogeneous clusters. Based on the results of these special cases, we show that the fastest-node-first algorithm produces a schedule with guaranteed performance.

We first consider the following theorem from [2].

Theorem 1 [2]

There exists an optimal schedule in which all processors sends messages without delay. That is, for all processor p in T, starting from its ready to send time, p repeatedly sends a message with a

period of its sending time until the broadcast ends.
With Theorem 1, we can simply discard those schedules that will delay messages, and still find the optimal one. Since there is no delay, we can characterize a schedule as a sequence of processors sorted in their ready to receive time. Recall the set A and B in the description of fast-node-first method. Since no delay is allowed, any scheduling method must schedule s, the processor in A that could have completed the sending at the earliest time, to send a message immediately. Formally we define $P = (p_0, ..., p_{n-1})$ to be a sequence of n processors sorted in their ready to receive time and the processors appear in P in non-decreasing sending speed, except for the source s_0. That is, the processors will be ready to receive the broadcast according to their sending time. The total broadcast time of P (denoted by T(P)) is by definition $\max_{n-1\ i=1} S(p_i)$, the latest ready to send time among all the processors1. A broadcast sequence P is optimal if and only if for any other permutation of P (denoted by P′), $T(P) \leq T(P')$.Let p be a processor and $NS_P(p, t)$ be the number of messages successfully sent at and before time t by p in the sequence P.

Formally, $NS_P(p, t) = \lfloor \frac{t-S(p)}{s(p)} \rfloor$, for $t \geq S(p)$. We can define ready to receive time $R(p_i)$ and

ready to send time $S(p_i)$ recursively (Eqn. 1). that is, the ready to receive time of the i-th processor in P is the earliest time when the total number of messages sent by the first $i - 1$ processors reaches i.

$R(p_0) = 0$ and $S(p_0) = 0$
$R(p_i) = \min\{t \mid {}_{i-1}X_{j=0} NS_P(p_j, t) \geq i\}$, $1 \leq i \leq n - 1$
$S(p_i) = R(p_i) + r(p_i)$, $1 \leq i \leq n - 1$ (1)

4.1 Power 2 clusters

In this section we consider a special case of heterogeneous clusters in which all the sending and receiving costs are power of 2, and we refer to such clusters as power 2 clusters [19]. Similar notation is also used in [17]. We show that FNF technique does guarantee minimum ready to receive time for the last processor receiving the broadcast message in a power 2 cluster, and this is the foundation of our competitive ratio analysis.
Henceforth we will focus on minimizing the ready to receive time of the last processor in a sequence $P = (p_0, ..., p_{n-1})$, which is denoted as $TR(P) = R(p_{n-1})$. We will later relate our finding with the latest ready to send time among all the processors, denoted by $TS(P) = \max_{n-1\ i=0} S(p_i)$, which is the time the broadcast actually takes. We choose this approach since TR(P) is much easier to handle in our mathematical analysis than TS(P). Note that the processor that has the latest ready to receive time may not have the latest ready to send time.

We first establish a lemma that it is always possible to switch a processor p with a slower processor q that became ready to receive right ahead of p (with the exception that q is the source) so that p and q will contribute more on the NS function after the switch. We then use an induction to show that this modification will not increase the ready to receive time of the processors thereafter, including the last one in the sequence. That is, the switch of p and q will not increase the final ready to receive time. This leads to the optimality of FNF for the last ready to receive time in a power 2 cluster.

Lemma 2 Let p be a first faster processor that became ready to receive right after a slower processor q in a sequence P, that is, $R(p) = t_1 > R(q) = t_0$, and $s(p) < s(q)$. By switching p with q in P we obtain a new sequence P'. Then, in this new sequence P', R(p) is moved forward from t_1 to t_0, and R(q) is delayed from t_0 to no later than t_1, and $NS_{P'}(p, t) + NS_{P'}(q, t) \geq NS_P(p, t) + NS_P(q, t)$, for $t \geq t_0$.

Let us consider the change of NS function from q's point of view. q is delayed by only one time step, so $NS_S(q)$ is at most greater than $NS_{S'}(q)$ by 1, which only happens at time interval $[t_0 + r(q) + ks(q), t_0 + r(q) + ks(q) + 1)$, where k is a positive integer, r(q) is the receiving time of q, and s(q) is the sending time of q. Note that this interval includes the time $t_0+r(q)+ks(q)$ but not $t_0+r(q)+ks(q)+1$. See Figure 4 for an illustration. However, during this interval $NS_{P'}(p)$ will be larger than $NS_P(p)$ by one since s(q) is a multiple of s(p), and r(q) is a multiple of r(p) due to speed consistency. This increase compensates the decrease due to q and the Lemma follows. After establishing the effects of exchanging the two processors on the NS function, we argue that the ready to receive time of the processors after p and q will not be delayed from P to P'. We prove this statement by an induction and the following lemma serves as the induction base:

Lemma 3 Let p and q be the $(j-1)_{th}$ and j_{th} processor in P, then the ready to receive time of p_{j+1} in P' is no later than in P.

Lemma 4 The ready to receive time of p_l in P' is no later than in P, for $j + 1 \leq l \leq n - 1$.

One immediate result from Lemma 3 and 4 is that for any processor sequence of a power 2 cluster, including the optimal ones, the final ready to receive time will never be increased by making the faster processors ready to receive earlier than slower ones. Now we have the following theorem: Theorem 5 The fastest-node-first algorithm gives optimal final ready to receive time for a power 2 cluster.

4.2 An approximation algorithm

Theorem 5 by itself is not very useful in practice since most clusters do not have such property on their transmission speed. We can use Theorem 5 to show that FNF is actually an approximation algorithm of competitive ratio 2 for the final ready to receive time. By increasing the transmission time of processors, we can transform any heterogeneous cluster into a power 2 cluster. We increase the sending and receiving time of each processor p to be $2^{\lceil \log s(p) \rceil}$ and $2^{\lceil \log r(p) \rceil}$ respectively, that is, the smallest power of 2 that is no less than the original value.

We will show that FNF, optimal for the transformed cluster, also gives a schedule at most twice that of the optimal final ready to receive time for the original cluster. Theorem 6 The fastest-node-first scheduling has a final ready to receive time no greater than twice that of the optimal final ready to receive time. Now we establish the relation between the final ready to receive time and the total broadcast time.

Theorem 7 The total broadcast time from fast-node-first technique is at most 2T +_, where T is the optimal total broadcast time, and is $\max\{r(p_i)\} - 2\min\{r(p_i)\}$.

References

[1] T. Anderson, D. Culler, and D. Patterson. A case for networks of workstations (now). In IEEE Micro, Feb 1995.
[2] M. Banikazemi, V. Moorthy, and D.K. Panda. Efficient collective communication on heterogeneous networks of workstations. In Proceedings of International Parallel Processing Conference, 1998.
[3] M. Banikazemi, J. Sampathkumar, S. Prabhu, D. Panda, and P. Sadayappan. Communication modeling of heterogenous networks of workstations for performance characterization of collective operations. In Proceedings of International Workshop on Heterogeneous Computing, 1999.
[4] A. Bar-Noy, S. Guha, J. Naor, and Schieber B. Multicast in heterogeneous networks. In Proceedings of the 13th Annual ACM Symposium on theory of computing, 1998.
[5] A. Bar-Noy and S. Kipnis. Designing broadcast algorithms in the postal model for message-passing systems. Mathematical Systems Theory, 27(5), 1994.
[6] P.B. Bhat, C.S. Raghavendra, and V.K. Prasanna. Efficient collective communication in distributed heterogeneous systems. In Proceedings of the International Conference on Distributed Computing Systems, 1999.
[7] M. Dinneen, M. Fellows, and V. Faber. Algebraic construction of efficient networks. Applied Algebra, Algebraic Algorithms, and Error Correcting codes, 9(LNCS 539), 1991.
[8] J. Bruck et al. Efficient message passing interface(mpi) for parallel computing on clusters of workstations. Journal of Parallel and Distributed Computing, Jan 1997.
[9] Message Passing Interface Forum. MPI: A message-passing interface standard. Technical Report UT-CS-94-230, 1994.
[10] M. R. Garey and D. S. Johnson. Computer and Intractability: A guide to the theory of NP-Completeness. W. H. Freeman, 1979. 20
[11] L. Gargang and U. Vaccaro. On the construction of minimal broadcast networks. Network, 19, 1989.
[12] M. Grigni and D. Peleg. Tight bounds on minimum broadcast networks. SIAM J. Discrete Math., 4, 1991.
[13] W. Gropp, E. Lusk, N. Doss, and A. Skjellum. High-performance, portable implementation of the MPI Message Passing Interface Standard. Parallel Computing, 22(6):789–828, 1996.
[14] S. M. Hedetniemi, S. T. Hedetniem, and A. L. Liestman. A survey of gossiping and broadcasting in communication networks. Networks., 18, 1991.
[15] R. Karp, A. Sahay, E. Santos, and K. E. Schauser. Optimal broadcast and summation in the logp model. In Proceedings of 5th Ann. Symposium on Parallel Algorithms and Architectures, 1993.
[16] R. Kesavan, K. Bondalapati, and D. Panda. Multicast on irregular switch-based networks with wormhole routing. In Proceedings of International Symposium on high performance computer architecture, 1997.
[17] R. Libeskind-Hadas and J. Hartline. Efficient multicast in heterogeneous networks of wrokstations. In Proceedings of 2000 International Workshop on Parallel Processing, 2000.
[18] A. L. Liestman and J. G. Peters. Broadcast networks of bounded degree. SIAM J. Discrete Math., 1, 1988.

[19] P. Liu. Broadcast scheduling optimization for heterogeneous cluster systems. Journal of Algorithms, 42, 2002.
[20] P. Liu and D. Wang. Reduction optimization in heterogeneous cluster environments. In Proceedings of the International Parallel and Distributed Processing Symposium, 2000.
[21] D. Richards and A. L. Liestman. Generalization of broadcast and gossiping. Networks, 18, 1988. 21
[22] J.A. Ventura and X. Weng. A new method for constructing minimal broadcast networks. Networks, 23, 1993.
[23] D. B. West. A class of solutions to the gossip problem. Discrete Math., 39, 1992. 22