NOTE

ON THE TIME REQUIRED TO SUM n SEMIGROUP ELEMENTS ON A PARALLEL MACHINE WITH SIMULTANEOUS WRITES

Ian PARBERY

Department of Computer Science, Whitmore Laboratory, The Pennsylvania State University, University Park, PA 16802, U.S.A.

Communicated by M. Nivat

Abstract. Suppose we have a completely-connected network of random-access machines which communicate by reading and writing data from their neighbours, with simultaneous reads and writes allowed. In the case of write conflicts, we allow any protocol which results in one of the competing values being written into the target register. We consider the semigroup summation problem, that is, the problem of summing n semigroup elements. If the semigroup is finite, we find that it can be solved in time \(O(\log n / \log \log n)\) using only \(n\) processors, regardless of the details of the write-conflict resolution scheme used. In contrast, we show that any parallel machine for solving the summation problem for infinite cancellative semigroups must take time \(\lceil \log_3 n \rceil\), again, regardless of the details of the conflict resolution scheme. We give an example where it is possible to sum \(n\) 'polynomial-sized' elements in less than \(\lceil \log_3 n \rceil\) time using only polynomially many processors. We are also able to show that such a machine must obey the \(\lceil \log_3 n \rceil\) lower bound for elements which are only polynomially larger. Our upper bounds are for a machine with a reasonable local instruction set, whilst the lower bounds are based on a communication argument, and thus hold no matter how much computational power is available to each processor. Similar results hold for a parallel machine whose processors communicate via a shared memory.

1. Introduction

Suppose we have a completely-connected network of processors. Each processor has a distinguished register called the communication register. In a single time-step, each processor may perform one or both of the following operations in the correct sequence:

(1) Perform a communication step. Either:
   (i) read the communication register of some processor, or
   (ii) write a value into the communication register of some processor.

(2) Perform an internal computation.

We allow an arbitrary number of processors to read a communication register simultaneously and, in the case of simultaneous writes, we allow any convention in which a single processor is allowed to write its value, whilst all other contending data items are lost. The winner in each case may depend in an arbitrary fashion...
upon the values being written, the target processor, the processors competing, or even the time. This includes most conflict-resolution schemes found in the literature, for example, in [6] the lowest numbered processor wins, and in [12] the processor attempting to write the largest value wins, with ties broken in favour of the lowest numbered processor. In the case of simultaneous attempts to read from and write into the same communication register, the reads are serviced before the writes. While deriving lower bounds we shall allow any kind of internal instruction, whilst upper bounds will be given on a machine with limited local instruction-set (including semigroup addition, logical operations, flow of control etc.).

Suppose \( P : N \rightarrow N \), and \( P(n) \geq n \) for all \( n \geq 1 \). A computation on an input of size \( n \) is defined as follows. The input is broken up into \( n \) unit-size pieces, and one piece is given to each of the first \( n \) processors. \( P(n) \) processors are activated. They execute their local programs synchronously, subject to the rules in the previous paragraph. \( P(n) \) is called the processor bound. Note that, throughout this paper, the number of processors used in any individual computation is finite. The time bound \( T(n) \) is the maximum, over all inputs of size \( n \), of the number of instructions executed before all processors have halted. The output is encoded in the final state of the processors, according to some reasonable output convention.

Suppose \((S, +)\) is an infinite cancellative semigroup (that is, \( S \) is closed under \(+\); \(+\) is associative; and, for all \( a, b, c \in S \), if either \( a + b = a + c \) or \( b + a = c + a \), then \( b = c \)). Examples include finite sets under disjoint set union, the natural numbers under addition, the integers under addition and finite bit strings under bit-wise exclusive-or (note that all groups are cancellative semigroups). We wish to ask the following question: how fast can the parallel machine described above add \( n \) arbitrary elements of a cancellative semigroup? We shall show that a network of \( P(n) \) processors requires time \( \Omega(n) \) to add \( n \) elements drawn from a subset of \( S \) of size \( P(n) \). Thus, for example, for every network \( M \) of \( n^c \) processors (where \( c \) is a constant), there is a constant \( d \) (dependent on \( c \)) such that \( M \) requires time \( \Omega(n) \) to add \( n \) \( n^d \)-bit integers.

Lower bounds for parallel models which communicate by reading and writing can be surprisingly subtle, even if multiple writes are not allowed [3, 16]. Previous lower bounds on models with simultaneous writes have only been shown for severely restricted machines. For example, Vishkin and Wigderson [18] bound the total amount of successful communication in each time-step, and Meyer auf der Heide and Reischuk [7] give a lower bound of \( \Omega(n) \) for integer summation on a machine with limited local instruction set. Our lower bound subsumes this result, and in contrast uses only elementary techniques. Recently, Meyer auf der Heide and Wigderson [8] have removed this restriction and extended the lower bound to parallel machines which compute strongly nonconstant symmetric functions using the 'lowest-numbered processor wins' convention for handling multiple writes. The proof is a complicated argument based on Ramsey theory. Our proof technique generalizes easily to a large class of strongly nonconstant functions whose intersection with the strongly nonconstant symmetric functions includes semigroup summa-
time bounds for parallel summations of semigroup elements

More recently, an improved lower bound for integer summation has been obtained by Beame [1].

We also briefly investigate upper bounds for the problem of summation in finite semigroups, and find that fast algorithms on a superlinear number of processors can be used to produce fast algorithms on processors. As a corollary, we deduce that the summation of \( n \) elements of a finite semigroup can be computed in time \( O(\log n) \) on \( n \) processors. This is an improvement of the result described by Vishkin and Wigderson [18], where they observe that the same upper bound on polynomially many processors follows from the work of Chandra, Stockmeyer and Vishkin [2].

The remainder of this paper is divided into two sections. Section 2 contains the lower bound for infinite cancellative semigroups and briefly discusses extensions to other models (including shared memory machines). Section 3 shows that the lower bound can be broken if polynomially-smaller sized input elements are used and contains upper bounds for finite semigroups.

2. The lower bound

**Theorem 1.** Let \((S, +)\) be an infinite cancellative semigroup. Then a network of \( P(n) \) processors requires time \([\log_3 n]\) to add together \( n \) elements of \( S \) drawn from some subset \( S' \subseteq S \) with \( P(n)^{O(\log n)} \) elements.

**Proof.** An upper bound of \([\log_3 n]\) steps is obvious. The base of three in the logarithm stems from the ability of a processor to take a value from its local memory, a value which it reads from a neighbouring processor and a value which is written into its communication register by a second neighbour and sum the three values in a single step. For the lower bound, suppose \( M \) is a \( P(n) \) processor parallel machine which can sum \( n \) elements in time \( T(n) \), and let \( x = (x_0, \ldots, x_{n-1}) \) be an input string consisting of \( n \) symbols, each of which is a member of \( S \). We assume that the processors are numbered \( 0, 1, \ldots, P(n) - 1 \), and that the output will be found in processor 0 at the end of the computation. Let \( G_x \) be the directed graph with vertices \((p, t), 0 \leq p < P(n), 0 \leq t < T(n)\), and with an edge from \((p_1, t_1)\) to \((p_2, t_2)\) if \( t_2 = t_1 + 1 \) and either \( p_1 = p_2 \) or, during time-step \( t_1 \) of the computation of \( M \) on input \( x \), either processor \( p_2 \) reads a value from \( p_1 \), or \( p_1 \) successfully writes a value into \( p_2 \). \( G_x \) is called the computation graph of the parallel machine on input \( x \). The \( i \)th symbol of \( x \) is said to be reachable if there is a path from vertex \((i, 0)\) to vertex \((0, T(n))\) in \( G_x \).

Note that it is not necessarily the case that all of the symbols of any particular input are reachable. For example, suppose that \( v \) is a fixed semigroup element. Suppose processor \( A \) wishes to communicate some semigroup element to processor \( B \). First, processor \( B \) writes the value \( v \) into its own communication register. In the next time-step, if \( A \) wants to write some other value to \( B \), it simply writes that value.
If it wishes to communicate \( v \), it writes nothing. In either case, in the next time-step the communication register of processor \( B \) contains the correct value. Using this technique, it is easy to construct a parallel machine which sums \( n \) semigroup elements in such a manner that any input string containing the element \( v \) has unreachable symbols.

Suppose the elements to be added together are drawn from a set \( S' \subseteq S \) of size \( N \). We claim that (provided \( N \) is sufficiently large) there is an input string in which all symbols are reachable. For a contradiction, suppose that every input has at least one unreachable symbol. Define the reachable set of \( x \) to be \( R_x = \{ i \mid x_i \text{ is reachable} \} \).

Let \( Q_x \subseteq \{ 0, 1, \ldots, n-1 \} \) be such that \( |Q_x| = n-1 \) and \( R_x \subseteq Q_x \). Then there is a unique \( i \) such that \( 0 \leq i < n \) and \( i \in Q_x \). For definiteness, suppose \( Q_x \) is chosen so that \( i \) is minimal. Call \( Q_x \) the critical set of \( x \), and \( x_i \) the unreachable symbol of \( x \).

Suppose we fix an input \( x \). How many input strings \( y \) are there such that \( G_x = G_y \)? If \( G_x = G_y \), then, clearly, \( R_x = R_y \) and the critical set for \( x \) is also critical for \( y \). Suppose there are two inputs \( y_1 \) and \( y_2 \) with identical critical strings such that \( G_{x_1} = G_{y_1} = G_{y_2} \). Then, by a simple cut-and-paste argument, both \( y_1 \) and \( y_2 \) must sum to the same value since (once the computation graph is fixed) the unreachable symbol cannot affect the output. But, by the cancellation law, if we have two inputs \( y_1 \) and \( y_2 \) with identical critical strings and identical sums, then they must have identical unreachable symbols. Therefore, \( y_1 = y_2 \), from which we deduce that there are at most as many candidates for \( y \) as there are different critical strings. Since there are at most \( N^{n-1} \) different critical strings, we can conclude that at most \( N^{n-1} \) different inputs can give rise to the same computation graph.

Let \( G(n) \) be the number of possible computation graphs on \( n \) inputs. By the pigeonhole principle, at least one graph must be used for at least \( N^n / G(n) \) input strings. If \( N \) is chosen such that \( N > G(n) \), then this value is greater than \( N^{n-1} \), which contradicts the result of the previous paragraph. Thus there must be an input string for which all symbols are reachable. Since, for all \( x \), \( G_x \) has in-degree 3, this implies that \( T(n) \geq \lceil \log_3 n \rceil \).

Exactly how large can \( G(n) \) be? Each computation graph has \( T(n) \) layers, each corresponding to a single step of the parallel machine. How many different choices for each layer are there? Clearly there are \( P(n)^{P(n)} \) choices for the subgraph corresponding to the read operations. The subgraph corresponding to the write operations forms a bipartite matching (if \( X \) and \( Y \) are finite, disjoint sets of vertices, a bipartite matching is a graph \( G = (V, E) \) where \( V = X \cup Y, E \subseteq X \times Y \), and each element of \( X \cup Y \) appears in at most one edge of \( E \)). Let \( M(x, y) \) be the number of bipartite matchings from a set of size \( x \) to a set of size \( y \geq x \). Then \( M(1, y) = y+1 \) and, for all \( x > 1 \), \( M(x, y) = M(x-1, y) + yM(x-1, y-1) \). Therefore, \( M(x, y) \leq (2y - x + 3)2^{y-2}y! \) (proof by induction on \( x \)). If we further define \( M(x) = M(x, x) \), we see (by Stirling's approximation) that \( M(x) \leq x^{x+O(1)} \). Thus there are \( P(n)^{P(n)+O(1)} \) choices for the subgraph corresponding to the write operations, and so \( G(n) \leq (P(n)^{2P(n)+O(1)}T(n)) \).
Note that if we insist that the elements to be added be described in $T(n)^{O(1)}$ bits, (which according to [9, 10] is one of the prerequisites for the parallel computation thesis), then the lower bound is not valid. However, the result does hold for semigroup elements which can be described in polynomially many bits, which at least ensures that the input encoding is 'concise' in the sense of [5].

**Corollary 2.** For every infinite cancellative semigroup $(S, \cdot)$ and $n^c$-processor network $M$, there is a set $S' \subseteq S$ with $\log |S'| = O(n^c \log^2 n)$ such that $M$ requires time at least $[\log n]$ to sum $n$ elements of $S'$. The addition of $n$ arbitrary elements of $S$ requires time at least $[\log n]$ regardless of the processor bound.

Note that the proof of Theorem 1 (and hence, Corollary 2) works equally well for parallel machines which compute functions on $n$ inputs with the property that fixing $n - 1$ of the inputs and the output fixes the remaining input.

A CRCW-PRAM is a variant of the shared memory machine (see, for example, [4, 6]) in which concurrent reads and writes are allowed. Theorem 1 extends to CRCW-PRAMS in the obvious fashion since a CRCW-PRAM with $S(n)$ words of shared memory and $P(n)$ processors can be simulated without asymptotic time loss by a network of $P(n) + S(n)$ processors. With a little thought, an exact upper and lower bound of $[\log_2 n]$ can be derived. Thus we can deduce the following corollary.

**Corollary 3.** For every infinite cancellative semigroup $(S, \cdot)$ and shared memory machine $M$ with $n^c$ processors and shared memory cells, there is a set $S' \subseteq S$ with $\log |S'| = O(n^c \log^2 n)$ such that $M$ requires time at least $[\log_2 n]$ to sum $n$ elements of $S'$. The addition of $n$ arbitrary elements of $S$ requires time at least $[\log_2 n]$ regardless of the processor bound or amount of shared memory available.

Meyer auf der Heide and Reischuk [7] prove a restricted version of Corollary 3 for integer addition in a model in which the type of internal computation which can be done in one step is severely limited. Our lower bound subsumes their result, and has a much simpler proof. Recently, Meyer auf der Heide and Wigderson [8] have removed this restriction (on the 'lowest-numbered-processor-wins' model), and have made the above-mentioned extensions to strongly nonconstant symmetric functions.

Our lower bound technique is based on the fact that communication with any particular processor must be channeled through a single register. This model is called a restricted access network in [9]. The lower bound still holds if each processor is allowed to read any register of its neighbours. If we also allow each processor to write into any register of its neighbours, then the lower bound no longer holds since a machine which has in its local instruction set an $n$-ary addition operator can sum $n$ elements in only 2 steps. However, if we restrict local instructions to binary operations, then our lower bound naturally extends to this model since a $P(n)$ processor network with $S(n)$ words of local memory per processor can be
simulated without asymptotic time loss on a $P(n)S(n)$-processor restricted-access network.

3. Some upper bounds

The lower bound of Section 2 is slightly unsatisfying because it holds only for inputs consisting of $n$ large elements. However, it holds for any infinite cancellative semigroup, and any reasonable convention for handling multiple-writes. We shall show that such a powerful result requires the use of large elements by giving an example in which the lower bound can be broken if polynomially-smaller sized input elements are used.

Suppose we wish to perform a bit-wise exclusive-or of $n$ bit-strings. It is possible to perform the exclusive-or of $n$ bits in only one step using $n + 1$ processors on a machine with the following protocol for dealing with simultaneous writes. If the number of processors attempting to write a non-zero value is odd, then the smallest-numbered processor attempting to write a non-zero value succeeds; otherwise, the largest-numbered processor succeeds. To ‘exclusive-or’ together $n$ bit-strings distributed one-per-processor in processors $0, 1, \ldots, n - 1$, processors $0$ through $n - 1$ write their input values to processor $0$, while processor $n$ writes a zero to processor $0$. The value received by that processor is the exclusive-or of the $n$ inputs. (Note that the same technique can be used to add $n$ elements of a finite semigroup in constant time and with $n + O(1)$ processors.)

Thus, $n$ bit strings can be bit-wise exclusive-or ed together in only $[\log_2 b] + 2$ steps using a $b(n + 1)$ processor network with the above register-access convention and a sufficiently powerful local instruction set. We use one team of $n + 1$ processors for each bit position and take one step to fan out the $n$ inputs to the $b$ teams using simultaneous reads, one step for each team to perform the exclusive-or of its bits and $[\log_2 b]$ steps to fan in the $b$ results. Thus, if $b \leq n/27$, $[\log_3 b] + 2 \leq [\log_3 n] - 1$ steps suffice. Yet a lower bound of $[\log_3 n]$ holds on $n^3 + O(n)$ processors when $b = \Omega(n^2 \log^2 n)$. An upper bound less than $[\log_3 n]$ can be obtained on $n^{3/2}$ processors using any reasonable convention for multiple writes, with $b = c\sqrt{n}$ for a carefully chosen constant $c$.

We can conclude from the above that the lower bound does not hold for even polynomially-smaller sized input elements. In the remainder of this section, we shall see that it is possible to do asymptotically better when summing $n$ elements of a finite semigroup, even if only $n$ processors are available and the multiple-write convention is arbitrary. In what follows, we assume that the $i$th processor of the parallel machine, $i \geq 0$ has a distinguished read-only register called the processor identity register, or PID, which is preset to $i$. The results hold for both networks and shared-memory machines. Let $T(n, P(n))$ be the time required to sum $n$ elements of an arbitrary (finite or infinite) semigroup using $P(n)$ processors. For convenience we shall write $T(n)$ for $T(n, n)$. 
Lemma 4. Suppose: \( f: \mathbb{N} \to \mathbb{N}, \ f(n) \leq n \) for all \( n \geq 0 \). Then \( T(n) \leq T([n/f(n)]) + T(f(n), n) + O(1) \).

Proof. Suppose we are to sum \( n \) elements with \( n \) processors. Each processor computes \( f(n) \). The processors divide themselves into \( f(n) \) teams. The \( i \)th team, \( 0 \leq i < f(n) \), consists of those processors with \( \text{PID} \mod f(n) = i \). Thus there are \( f(n) - 1 \) teams of \( \lfloor n/f(n) \rfloor \) processors and one team of at most \( \lfloor n/f(n) \rfloor \) processors. Each team independently computes the sum of its inputs, in time \( T(\lfloor n/f(n) \rfloor) \) (we assume that \( T(n) \) is monotone nondecreasing). This leaves \( f(n) \) partial sums, which can be added together in time \( T(f(n), n) \).

A similar result can be proved for \( n^e \) processors. However, Lemma 4 is sufficient for our purposes.

Corollary 5 (Shiloach and Vishkin [17]). The maximum of \( n \) integers can be found in time \( O(\log \log n) \) with \( n \) processors, and in constant time with \( n^{1+\varepsilon} \) processors (for any constant \( \varepsilon > 0 \)).

Proof. It is easy to compute the maximum of \( n \) elements in constant time using \( n^2 \) processors (hint: construct an \( n \times n \) matrix with the \( (i,j) \)th entry one if the \( i \)th input element is less than the \( j \)th, and zero otherwise). Thus, by using Lemma 4 with \( f(n) = \lfloor \sqrt{n} \rfloor \), the time required to find the maximum of \( n \) elements using \( n \) processors is given by

\[
T(n) \leq T(\lfloor \sqrt{n} \rfloor) + O(1) = O(\log \log n).
\]

By limiting the depth of recursion to \( \lfloor \log_k n \rfloor \), for some constant \( k \geq 1 \), the time can be reduced to \( O(\log k) \) with \( n^{1+1/k} \) processors.

The following result is well-known and can be obtained using the techniques of \([9, 10]\).

Theorem 6. The finite semigroup summation problem can be solved in constant time with \( 2^{O(n)} \) processors.

A similar result is reported by Vishkin and Wigderson [18]. It is possible to solve the finite semigroup summation problem in constant time with only \( 2^{n^{1+\varepsilon}} \) processors, for any real number \( \varepsilon > 0 \) (see, for example, [14]). This matches the lower bound of Yao for computing the parity of \( n \) bits [19] on a machine with limited instruction set. However, Theorem 6 is sufficient for our needs.
The finite semigroup summation problem can be solved in time $O(\log n/\log \log n)$ using $n$ processors.

Proof. By Lemma 4 with $f(n) = O(\log n)$, and Theorem 6. \qed

Thus, for example, the exclusive-or of $n$ bits can be computed in time $O(\log n/\log \log n)$ using $n$ processors. This matches the lower bound of Yao [19] for polynomially many processors with limited instruction-set. Theorem 7 has been reported independently by Reif [15].

Theorem 7 can be generalized to the case of summing $n$ constant-bit elements of an infinite semigroup with the property that any $n$-element sum can be described in $O(\log n)$ bits (since Theorem 6 holds with the processor bound modified to $2^{O(n \log n)}$, the proof follows from Lemma 4 with $f(n) = O(\log n/\log \log n)$.) Thus, for example, $n$ single-bit integers can be added together in time $O(\log n/\log \log n)$ using $n$ processors.

Acknowledgment

A preliminary version of Theorem 1 appears in the author's Ph.D. Thesis [9], where it is restricted to the addition of natural numbers. I would like to thank Mike Paterson for pointing out a serious flaw in an earlier version of that result, and Hans-Ulrich Simon for several stimulating discussions which led to the current formulation. Working drafts of the current paper have appeared in [11, 13]. I would also like to thank Janos Simon for his comments on an earlier draft of this paper, and an industrious referee for helpful and much-appreciated advice on presentation and phrasing.

References


