to be combined and communicated are partitioned into d (the dimension of the cube) equal parts, where one part is exchanged in each direction. This kind of approach to increasing the utilization of the communication network is discussed in [4], Chapter 21. The effect is to reduce the con multiplying the β term in the time complexities of Versions 2 and 3 by a factor d, w carried through to reduce the execution time of the hybrid algorithm as well. the scope of this paper.

Acknowledgements

The author would like to thank Dr. Israel Nelken and Dr. Bernar comments.

References

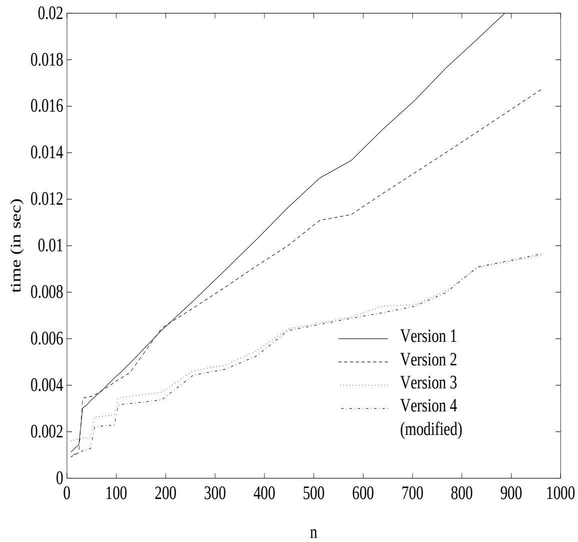
[1] E. Anderson, A. Benzoni, J. Dongarra, S. Moulton, S. O R. van de Geijn. LAPACK for distributed memory arch *allel Processing for Scientific Computing, Fifth S*

[2] J. J. Dongarra and R. A. van de Geijn. Reducti on distributed memory architectures. LA University of Tennessee, 1991.

[3] T. H. Dunigan. Performance of the 11491, Oak Ridge National Laborat

[4] G. Fox, M. Johnson, G. Lyzeng *Concurrent Processors*, volu

[5] C. Moler, J. Little, an



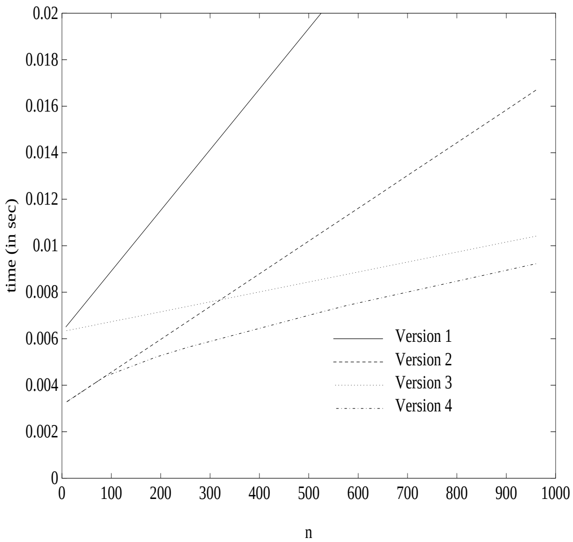
0.02
0.018
0.016
0.014
0.012
0.01
0.008
0.006
0.004
0.002
0
time (in sec)
0
100
200
300
400
500
600
700
800
900
1000
n
Version 1
Version 2
Version 3
Version 4
(modified)

Figure 8: Observed time as a function of vector length n on 64 nodes. In this version, the length of vectors being communicated has not been adjusted, and the algorithm has been modified as described in Section 6, with $\alpha = 480\mu$sec, $\beta = 2\mu$sec, and $\gamma = .35\mu$sec.

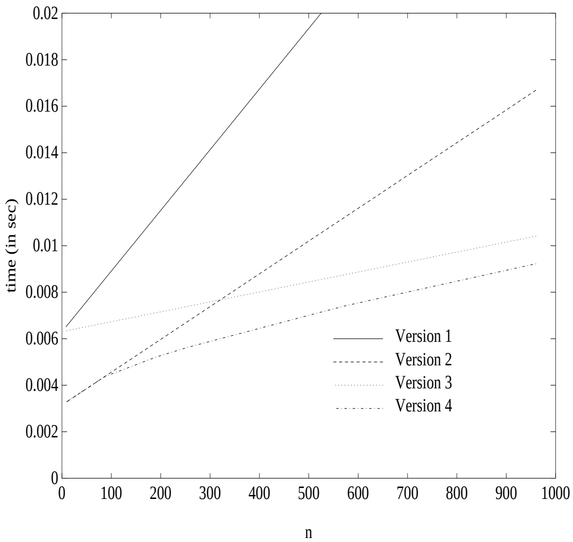
communicated is less than 25. While there is a dramatic drop in execution time for Versio
2 when $n \leq 25$, Version 3 is particularly aided by the reduction in startup time, si
being communicated are halved at each step. For example, on a cube of dimens
vector is of length 512, the vectors communicated during the last tw
8, respectively, thereby reducing the execution time. This di
in Section 4. The following modification appears to
optimal choice as described in Section 4
each stage until the vector lengt
vectors thereafter, is us
method shows si

7 Conc

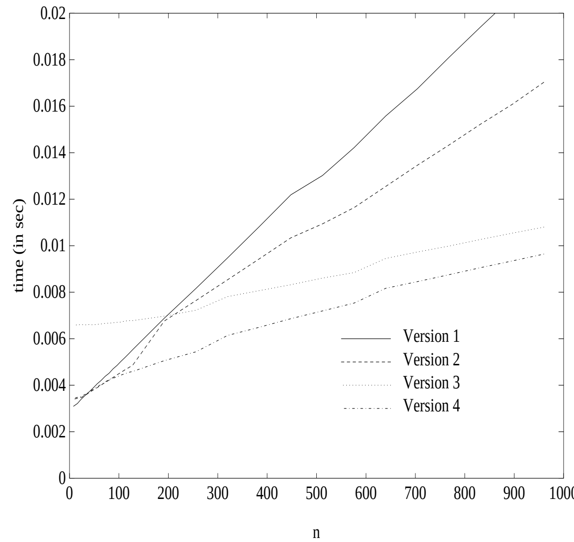
The impleme
the ope
ou



time (in sec)
n
Version 1
Version 2
Version 3
Version 4
0.02
0.018
0.016
0.014
0.012
0.01
0.008
0.006
0.004
0.002
0
0
100
200
300
400
500
600
700
800
900
1000



time (in sec)
n
Version 1
Version 2
Version 3
Version 4



time (in sec)
n
Version 1
Version 2
Version 3
Version 4

Figure 7: Predicted (*left*) and observed (*right*) time as a function of vector length n on 64 nodes when $\alpha = 525\mu$sec, $\beta = 2\mu$sec, and $\gamma = .35\mu$sec.

6 Experiments on the Intel iPSC/860

To test our theoretical results, we implemented the various global combine operations on tl
Intel iPSC/860. Our experiments centered around a specific global combine operation, th
summation of single precision floating point vectors.

The Intel iPSC/860 is a commercial parallel processor that consists of l
connected in a hypercube topology. Although this machine is somewhat
in Section 2, it can be programmed in such a way that all assumpti
In [3] it is shown that the cost for communicating a flo
iPSC/860 is roughly given by Assumption 5, with $\alpha = 136$
to single precision floating point numbers is $\gamma = .$
25 or less are communicated, the communica
this complication, we padded all communi
floating point numbers. There is so
general bookkeeping, yielding
overhead: $\beta = 2.0\mu$sec.
The first series of
proaches descr
communi
t

```
hybridCOMB2R(n, x, y, d)
begin
     if n < 2α/((d-1)(β+γ)+γ) then
          if bit(i,d-1)=1 then
               send (n, x, ngbr(i,d-1))
          else
               recv (n, y, ngbr(i,d-1))
               combine (n, x, y)
               if d-1 > 0 call hybridCOMB2R(n, x, y, d-1)
     else
          let x0 = x[0,...,n/2-1], x1 = x[n/2,...,n]
          if bit(i,d-1)=0 then
               send(n/2, x1, ngbr(i,d-1))
               recv(n/2, y, ngbr(i,d-1))
               combine(n/2, x0, y)
               if d-1>0 call hybridCOMB2R(n/2, x0, y, d-1)
               recv(n/2, x1, ngbr(i,d-1))
          else
               send(n/2, x0, ngbr(i,d-1))
               recv(n/2, y, ngbr(i,d-1))
               combine(n/2, x1, y)
               if d-1>0 call hybridCOMB2R(n/2, x1, y, d-1)
               send(n/2, x1, ngbr(i,d-1))
end
```

Figure 6: Optimal hybrid global combine-to-root routine

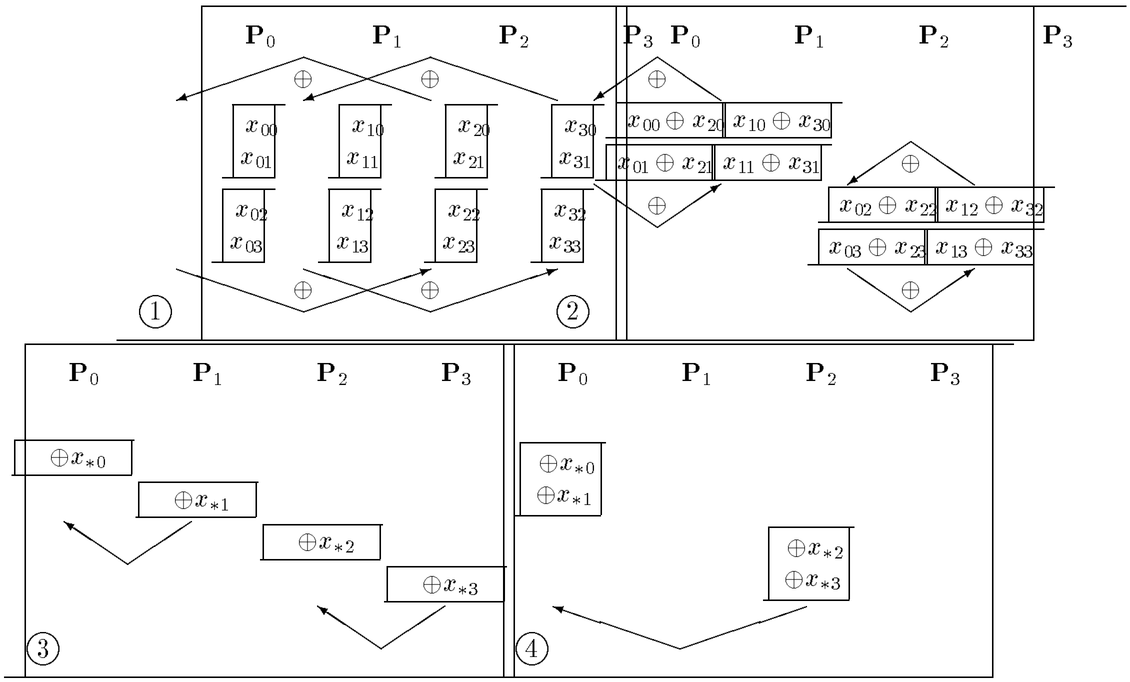


Figure 5: Second approach to combine-to-root on 4 nodes

5 Combine-to-Root

Some algorithms require the result of the global combine to only be known
loss of generality, we can take this node to be $\mathbf{P}_0$.
The first approach prog

```
hybridCOMB(n, x, y, d, S)
begin
    if S_{d-1} = 0 [a] then
        send (n, x, ngbr(i,d-1))
        recv (n, y, ngbr(i,d-1))
        combine (n, x, y)
        if d-1 > 0 call hybridCOMB(n, x, y, d-1, S)
    else
        let x0 = x[0,...,n/2-1], x1 = x[n/2,...,n]
        if bit(i,d-1)=0 then
            send(n/2, x1, ngbr(i,d-1))
            recv(n/2, y, ngbr(i,d-1))
            combine(n/2, x0, y)
            if d-1>0 then call hybridCOMB(n/2, x0, y, d-1, S)
            send(n/2, x0, ngbr(i,d-1))
            recv(n/2, x1, ngbr(i,d-1))
        else
            send(n/2, x0, ngbr(i,d-1))
            recv(n/2, y, ngbr(i,d-1))
            combine(n/2, x1, y)
            if d-1>0 call hybridCOMB(n/2, x1, y, d-1, S)
            send(n/2, x1, ngbr(i,d-1))
            recv(n/2, x0, ngbr(i,d-1))
end
```

[a] In Section 4 it will be shown that an optimal hybrid strategy can be obtained by deleting S from the calling sequence and replacing this condition by

$$n < 2\alpha/((d-1)(\beta+\gamma)+\gamma)$$

Figure 4: Hybrid global combine routine

Since $T(S, 2^{-(d-j-1)}n, j) > T(S, 2^{-(d-j)}n, j) \geq 0$ and $T(S, n, d) \leq T(R, n, d)$, we conclude th

$$\alpha + 2^{-(d-j-1)}n(\beta+\gamma) < 2\alpha + 2^{-(d-j)}n(2\beta+\gamma)$$

and hence

$$n < 2^{d-j}\frac{\alpha}{\gamma} < 2^{d-k}\frac{\alpha}{\gamma} \leq 2^{d-k}\frac{\alpha}{k(\beta+\gamma)+\gamma}$$

which contradicts the definition of k.

Case 2b: $j \leq k$. Then $S = (S_0, \ldots, S_k, 1, \ldots, 1)$ and

$$T(S, n, d) = \sum_{i=1}^{d-k-1} (2\alpha + 2^{-k}n(2\beta +$$

However,

$$T(\bar{S}, n, d) = \sum_{i=1}^{d-k-1} \Big(2\alpha + 2^{-k}$$

and, by Case 1 above,

$T($

which ag

Proof: The first four results follow immediately from the definition of $T(S, n, d)$. Th
from the observation that the time of the first j steps of
time for the last $d-j$ steps. Ho
wher

combines of the partial result on subcubes of dimension $d-1$. This suggests a fa
strategies that combine Versions 2 and 3.
We can generate 2^d separate strategies by co
indicates that for the s

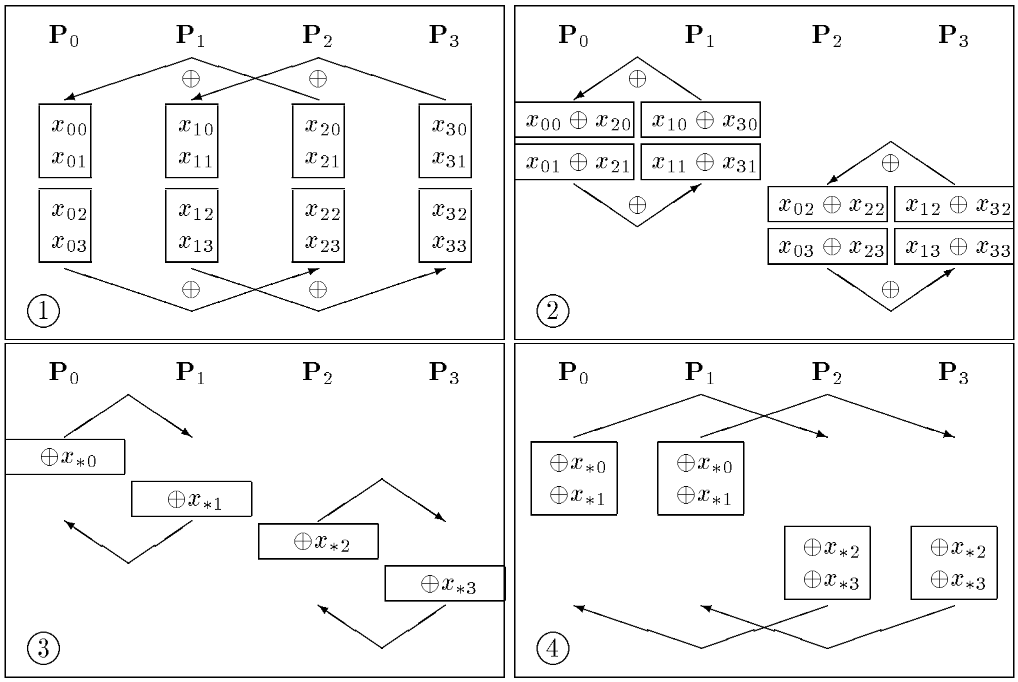


Figure 3: Version 3 on 4 nodes

not sent. This process proceeds for direction $d-2, \ldots, 0$, where the size o
communicated and combined is halved at each s
vector that results fr

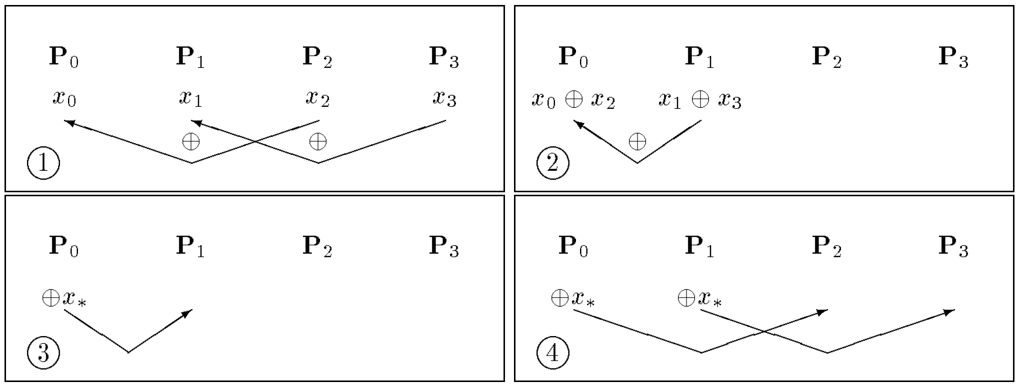


Figure 1: Version 1 on 4 nodes

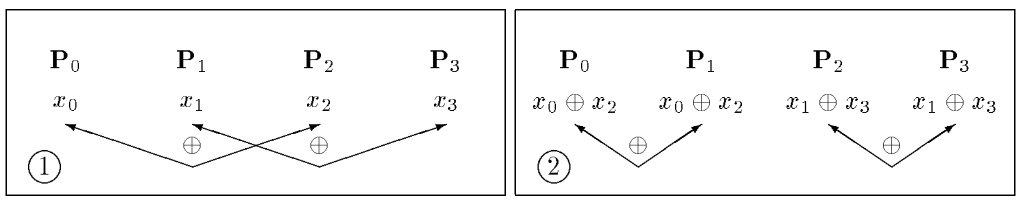


Figure 2: Version 2 on 4 nodes

3.2 Version 2

A second approach, described in [4], Section 14-5.4, starts by
contents of vector x_i with its neighbor in di
with the content

2 Assumptions

Target architectures for our algorithmare distributed memory hypercube mu
ing Multiple Instruction Multiple Data (MIMD) r
NCUBE2 and Transputer base
For our theoreti

LAPACK Working Note 29
On Global Combine Operations*

Robert A. van de Geijn

Department of Computer Science
University of Tennessee
Knoxville, Tennessee 37996-1301

and

Department of Computer Sciences
University of Texas at Austin
Austin, Texas 78712

na.vandegeijn@na-net.ornl.gov

Abstract

We discuss a hybrid strategy for implementing global combine operations on distributed memory MIMD multicomputers. A theoretical analysis is given and results from its implementation on the Intel iPSC/860 are reported.

1 Introduction

this paper, we address the implementation of the combine operation when vectors of data to
be combined are distributed among the processors (nodes) of a MIMD hy
Several solutions to this problem have appeared
that combines two of th
Ou