RAL-92-079

Science and Engineering Research Council
**Rutherford Appleton Laboratory**
Chilton DIDCOT Oxon OX11 0QX

RAL-92-079



# Some Results using the BECAUSE Benchmark Suite on the Intel iPSC/2 and iPSC/860 Hypercubes

**R F Fowler B W Henderson and C Greenough**

December 1992

# Some Results using the BECAUSE Benchmark Suite on the Intel iPSC/2 and iPSC/860 Hypercubes

R.F. Fowler†, B.W. Henderson‡ and C. Greenough†

† *Mathematical Software Group*
‡ *Parallel Processing Group*

October 1992

**Abstract**

We present results for the BECAUSE Benchmark Set (BBS) on two machines, the Intel iPSC/2 and iPSC/860. These computers are of similar design and architecture, both being MIMD machines with processors connected together in a hypercube topology and supporting efficient through routing of messages. The software environment on the two machines is also very similar with support for a range of basic communication operations.

The main difference between the machines lies in the processor used on each node, which on the iPSC/2 is an Intel 386 (plus FPU) while the iPSC/860 uses a state of the art i860 chip. It is shown that the basic floating point performance of the iPSC/2 is no longer competitive with current workstations such as the IBM RS6000/320. On the other hand, a single node of iPSC/860 is found to give comparable performance to such workstations. The communication performance of the machines for long messages is very similar (2.8Mbytes/sec.), which is reasonable for the iPSC/2 but is found to present problems on the iPSC/860 in some of the benchmark tests. Results for other aspects of communication performance including overlap with computation and access to the concurrent file system are also presented.

The scalability of the performance when using multiple nodes on both machines is examined for the Class 2 and 3 BBS tests and comparisons made with the reference machine. Some aspects of the implementations used for the parallel versions of the tests are discussed.

Presented at the BECAUSE Workshop, INRIA, Sophia–Antipolis, France, 12–16 October 1992

---

Mathematical Software Group
Computational Modelling Division
Rutherford Appleton Laboratory
Chilton, Didcot
Oxfordshire OX11 0QX

# Contents

# 1 Introduction

The BECAUSE Benchmark Set (BBS) is a suite of test programs covering the range of numerical operations associated with typical finite element and finite volume codes used in field modelling problems. The origin and detailed specification of these tests is described in documents from the BECAUSE project, [4], [5]. The tests are aimed at parallel computers of both MIMD and SIMD type.

The tests within the BBS are classified into three classes with Class 1 covering basic operations (communication, DAXPY, etc.), Class 2 containing a range of standard operations on unstructured meshes and Class 3 including higher level routines such as linear solvers and FFT codes.

The aim of this paper is to present some of the results of the BBS tests that have been run on an iPSC/2 and an iPSC/860, and to discuss the usefulness of this data.

In section 2 we present a brief summary of the architecture of the machines and the methodology used in programming the BBS tests. Sections 3, 4 and 5 then preset some representative results from the test programs. A more comprehensive description of the test programs and the iPSC/2 and iPSC/860 results can be found in [3]. Finally in section 6 we comment on the results for the two machines and the value of the BBS tests.

# 2 Machine Architecture and Implementation Methodology

The iPSC/2 and iPSC/860 both share the same connection topology, namely hypercube, and the same communication hardware. A *Direct Connect Module* (DCM) on each node allows the rapid creation of communication paths between any two nodes in the network. The iPSC/2 machine that was used has 32 nodes each consisting of a 80386 processors plus a Weitek 1167 floating point processor, known as an SX node. Each node has 4 Mbytes of memory.

The iPSC/860 uses the i860 processor on each node for both integer and floating point calculations. The machine used in these tests has 64 nodes, each with 16 Mbytes of memory. Full technical descriptions of these machines is available in publications from Intel such as [1] and [2]. Access to both machines was via a SUN-4 host using an Ethernet connection.

In terms of software, both machines run the NX/2 operating system on all nodes. Fortran 77 was used to implement all the tests, with the Greenhills Fortran compiler on the iPSC/2 and the Portland Group Fortran compiler (Release 3.0) on the iPSC/860. All tests were carried out full optimisation of the compilers, except where noted in the results.

The BBS tests are defined in terms of a written description accompanied by a serial version of the test (where possible) in standard Fortran 77. The parallel implementations that we have used on the iPSC machines are in general based on minimal modifications to the existing serial software combined with direct use of the message passing library provided by Intel. In some tests we use library routines which call the Intel communication functions, but these routines are of a similar level to the Intel ones. They are used to allow parallel implementations designed for another machines (the PARSYS SN1000) to be moved more easily to the iPSCs. In some cases, such as the Gaussian elimination test, it proved necessary to discard the serial version provided and write a new program, though still keeping to standard Fortran 77.

The basic measure that we make for all tests is the elapsed time since this this is key value which can be directly compared to other machines and does not hide the real performance, as measures such as speed up do. Where possible we also report the MFLOP/s rates since this measure is widely used and is convenient for the display of results. Results are compared with the performance of an IBM

RS6000/320 to put them in perspective with current high performance workstations.

# 3 Class 1 Test Results

Within Class 1 of the BBS there exist three types of basic tests: interprocessor communication, basic computational operations and disk I/O performance. Some representative examples of these are presented here.

## 3.1 BBS.1.1.1: Neighbour to neighbour communication

This test looks at the basic communication between neighbouring nodes in an MIMD machine. The test involves sending data from one node to another and then back again using varying message size. Timing is performed on the first node and we assume symmetry so that the time to send in one direction is half the total time. On the iPSC machines this test is implemented via the CSEND and CRECV functions described in the Intel documentation, [1]. Figure 1 shows the results of this test for the iPSC/2 and iPSC/860, in terms of Mbytes/s as a function of the message size.

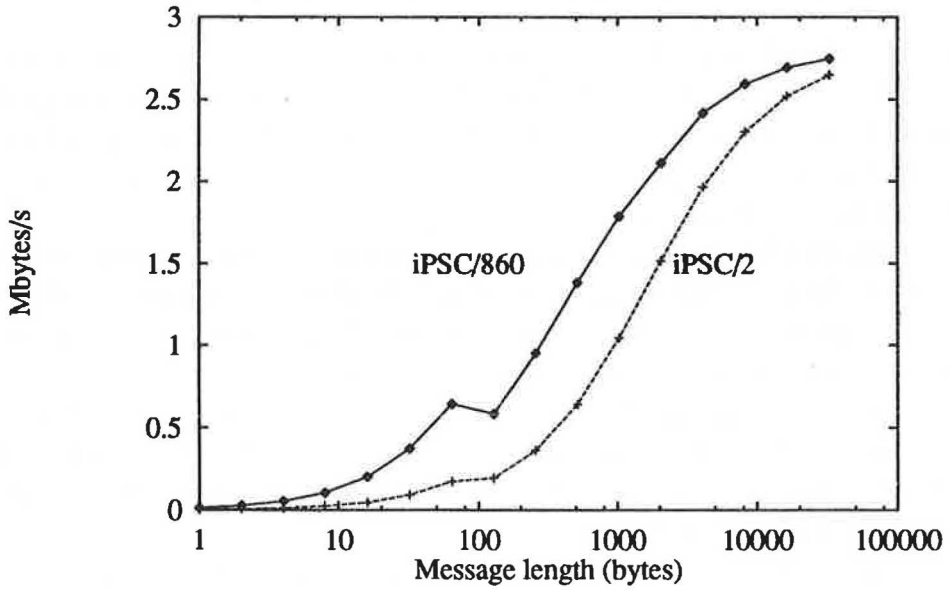


Figure 1: Results of BBS.1.1.1 on the iPSC/2 and iPSC/860 showing the data transmission rate as a function of message size.

From the results we see that both machines approach the theoretical rate of 2.8 Mbytes/s of the hardware for long messages. However, the iPSC/860 does perform significantly better than the iPSC/2 for short messages. The discontinuity at message lengths close to 100 is due to the different manner in which short and long (> 100 bytes) messages are handled. In fact for the iPSC/860 there is also some evidence of a second discontinuity in the performance for messages of the order of 4000 bytes, the size of the FIFO buffer used on the message routing hardware.

The data form this benchmark can be used to determine the values of the communication parameters $R_\infty$, the asymptotic data transfer rate and $N_{1/2}$, the message length for half the peak performance. These are related to the parameters $T_{start}$ and $T_{send}$, the start up time for communication and the time to send one byte of data respectively. Both sets of parameters are given in Table 1.

| Parameter | iPSC/860 | | iPSC/2 | |
|---|---|---|---|---|
| | (<100 bytes) | (>100 bytes) | (<100 bytes) | (>100 bytes) |
| $T_{start}$ ($\mu$S) | 74 | 175 | 328 | 612 |
| $T_{send}$ ($\mu$S/byte) | 0.35 | 0.36 | 0.59 | 0.36 |
| $N_{1/2}$ (bytes) | 211 | 486 | 555 | 1748 |
| $R_{\infty}$ (Mbytes/sec) | 2.85 | 2.77 | 1.69 | 2.78 |

Table 1: Communication parameters for the iPSC/860 and iPSC/2.

Other BBS tests look at the cost of communication over several "hops", i.e. between non-neighbouring processors, the ability to send messages in both directions at once along a link and the data rate to the host processor. We will not present these results here for reasons of space, though they are given in [6]. We just note here the following points:

- The direct routing of the iPSC machines means that communication over many links involves a very small additional overhead compared to nearest neighbour communication.
- The communication links are bi-directional and can give close to the single message throughput in both directions simultaneously.
- Though a host program can be run on either the SRM (the front end machine for the iPSCs) or a remote machine such a Sun-4, this mode of working is no longer encouraged. A single i860 node can out perform both these hosts and on the machine tested had 16 Mbytes of memory. The data rate to the SRM from any node is close to 2.8 Mbytes/s for long messages.

### 3.2 BBS.1.1.6: Overlap of communication with computation

This test is aimed at determining the ability of two nodes to overlap calculation and communication. It is run in three parts, the first of which is to determine the time required to send a fixed amount of data from one node to a neighbour. A short DAXPY type calculation is then timed and we determine how many such operations can be performed in the same time as that taken by the communication operation. Finally we time how long it takes to run both the computation and the communication together, and hence determine the percentage of overlap that is possible.

The specification for this test suggests that the data should be sent by one process while another is responsible for running the calculation on each node. However, since the iPSC/860 does not support multiple processes on each node, we could not follow this implementation. Instead, a single large message was initiated using the asynchronous routines `ISEND/IRECV` and then the computational phase performed, followed by a wait for any communication to complete. This implementation only gives the overlap of communication with very long messages.

For the iPSC/860, an overlap of approximately 84% was measured for combined computation and calculation, with a message length of 1 Mbyte. The iPSC/2 had an overlap of 76% on the same test.

While we would expect lower performance with shorter message length, since the set up time will remain constant, this shows that a significant amount of the communication time can be hidden on these machines by use of the `ISEND` and `IRECV` routines.

### 3.3 Basic floating point tests

A number of floating point tests are included in Class 1 of the BBS tests such as BBS.1.2.1 and BBS.1.2.2 which look at some basic vector operations. With appropriate distribution of the data, the DAXPY type operations are fully parallel, as long as we have fewer processors than elements in the vectors. Other operations, such as vector dot products, require some global communication which limits the overall speed up of parallel evaluation. Table 2 compares some of the results of the iPSC machines (one node) with the reference machine (IBM RS6000/320).

| Operation | iPSC/860 | iPSC/2 | RS6000/320 |
|---|---|---|---|
| $z_i = 2x_i + 1$ | 12.1 | 0.258 | 6.27 |
| $y_i = x_i z_i + 1$ | 10.2 | 0.224 | 5.52 |
| $w_i = x_i y_i + z_i$ | 11.7 | 0.194 | 4.43 |
| $s = \sum w_i x_i$ | 18.1 | 0.232 | 6.79 |
| $x_{max} = \max(x_i)$ | 2.47 | 0.127 | 1.30 |

Table 2: MFLOP/s for basic floating point tests on the iPSC/860, iPSC/2 (single processor results) and RS6000/320 (vector length $\approx$ 9000).

It is found that the performance of the DAXPY type operations do scale linearly with the number of processors on both machines. In fact, due to the increased amount of cache available with multiple processors, it is sometimes possible to achieve greater than linear speed up.

Both the dot product and maximum value test show sub-linear speed up due to the cost of communication. Typical results for the latter (BBS.1.2.2) on the iPSC/860 are shown in Figure 2.

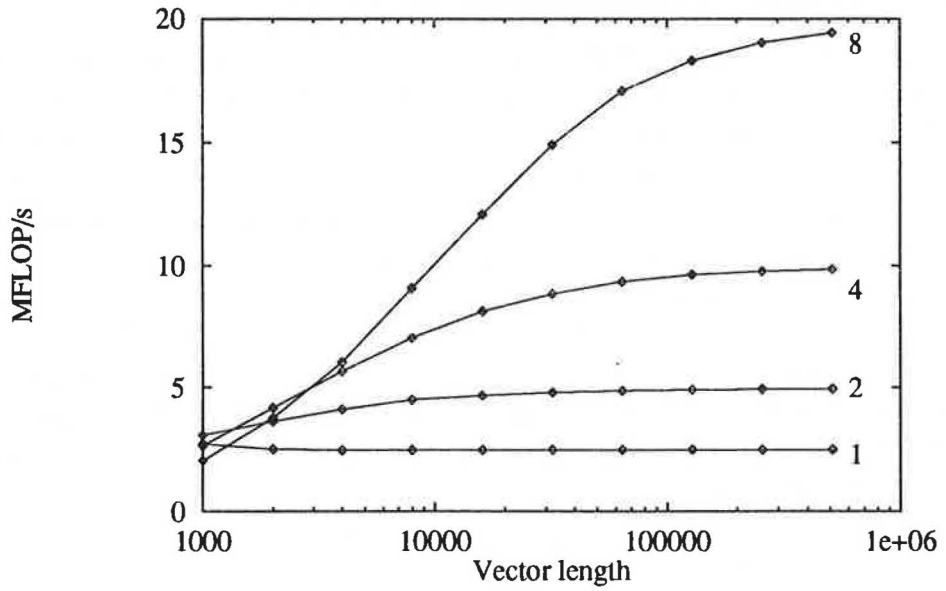


Figure 2: Results of BBS.1.2.2 on the iPSC/860 showing the computation rate as a function of vector length using 1 to 8 processors.

A simple analysis of the results for these tests shows that the communication cost is independent of the vector length but scales as $\log_2 p$ with the number of processors $p$. This is as expected for a hypercube architecture. If we ignore the effects of the cache memory on the results for short vector

lengths, we can assume that the asymptotic rate for the operations using $p$ processors is given by $r_\infty^p = pr_\infty^1$, where $r_\infty^1$ is the asymptotic MFLOP/s rate for 1 processor.

The computation rate for finite vector length $n$ in the dot product test can be modeled by

$$r_n^p = \frac{2n}{\frac{2n}{pr_\infty^1} + 170\log_2 p} \tag{1}$$

in the case of the iPSC/860 (the communication cost is approximately $170\log_2 p\ \mu$s in this equation). For this machine $r_\infty^1 = 18.1$ in double precision. This equation can also be used to give an estimate of the half performance length for this operation,

$$n_{1/2}^p = 85 r_\infty^1 p\log_2 p \tag{2}$$

Thus we can predict that that for a full iPSC/860 of 128 processors, that half maximum performance for the dot product operation would require a vector length of $n_{1/2}^{128} \approx 1400000$. A similar analysis can be made for the maximum value test, where the actual communication cost is approximately the same as for the dot product.

# 4 Class 2 Test Results

The tests within Class 2 of the BBS cover basic kernels that are associated with many mesh based calculations. In almost all cases we need to make some decomposition of the physical mesh in terms of either elements or nodes so that parts of the work can be mapped to each processor. Different approaches for the decomposition have been employed in the tests and the results presented are examples of the performance that can be obtained, but do not claim to be the most efficient possible.

One of the difficulties in specifying a benchmark for a parallel machine based on kernels extracted from an application code is that the distribution of data among the processors is not clearly specified. If one selects the optimum distribution of data for one operation it may well not be good for another kernel and the cost of data movement will only become clear when the whole application is implemented on the parallel machine. Nevertheless we have tried to make reasonable assumptions for the data distribution within the tests, but accept that these are not unique.

Results of two representative Class 2 tests are discussed in the following sections.

## 4.1 Contribution loop tests

Tests within subclass 1 of Class 2 represent some of the basic operations that occur in calculations on unstructured meshes in 2D and 3D. For all these tests we have used a simple decomposition of the mesh into slices and assigned one slice to each processor. This has the advantages of being easy to implement and requiring only communication with one other processors for the correct assembly of terms on the interface between two subdomains. A more complex decomposition strategy could give fewer nodes on the subdomain interfaces, at the expense of having to deal with many neighbouring processors.

A typical example is BBS.2.1.5 which computes flux terms on a hexahedral mesh. This test has been implemented as outlined above with slices of the mesh assigned to each processor. Each processor performs the flux calculation on the elements assigned to it. Once this is complete, data for the boundary nodes is exchanged so that each processor can have the correct results for all nodes within its subdomain. We do not try to overlap communication and computation in the current

implementations, though clearly this could be done and the results of BBS.1.1.6 show that this would reduce the communication costs.

Some results are shown in Figure 3 on the iPSC/860 for a range of mesh sizes and number of processors.

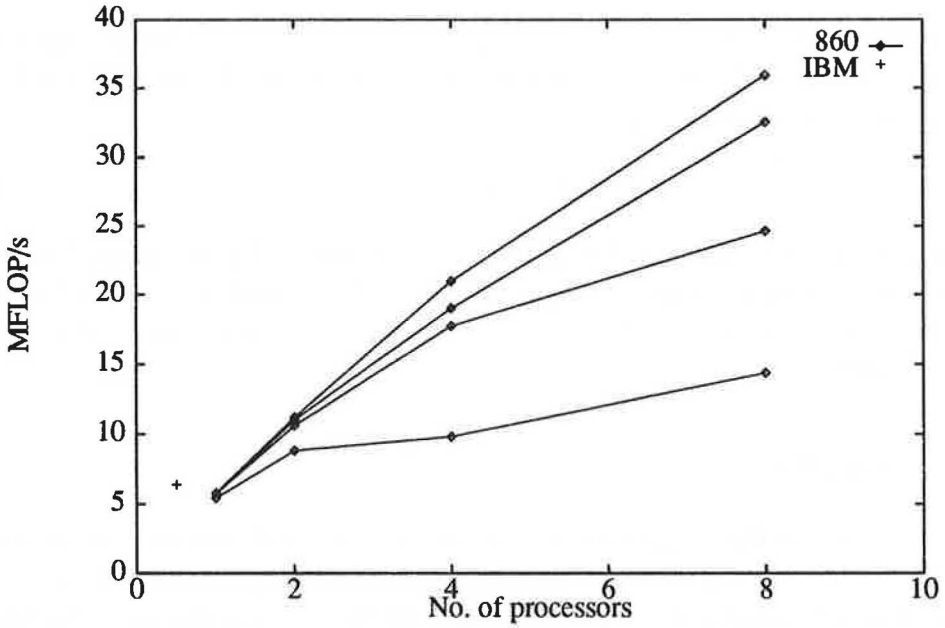


Figure 3: Results of BBS.2.1.5 on the iPSC/860 showing the computation rate (MFLOP/s) as a function of number of processors. The 4 curves correspond to mesh sizes of $10^3$, $20^3$, $30^3$ and $40^3$ nodes, from bottom to top. The performance of the reference machine (+) is also shown.

We find that the performance of a single node of the iPSC/860 is similar to that of the RS6000/320. As is expected the efficiency [1] of the multiprocessor version increases as the mesh size is increased, since the amount of computation in this test is proportional to $n^3$ (the number of nodes in the $n \times n \times n$ mesh) while the communication costs vary as $n^2$. Using 8 processors a mesh of about 8000 nodes is required to achieve an efficiency of 50% while the largest example (64000 nodes) gives about 77% efficiency.

Results for the iPSC/2 will show much better efficiency, but only because the computational power of each processor is so low that the communication costs are hidden. Even using 16 processors the iPSC/2 does not reach the speed of either one node of the iPSC/860 or the RS6000/320.

## 4.2 Forward substitution on a sparse matrix

A common problem in iterative linear algebra is the solution of systems of the form $L\mathbf{x} = \mathbf{y}$ where $L$ is a sparse lower triangular matrix and $\mathbf{x}$ and $\mathbf{y}$ are vectors. BBS.2.4.2 is based on this operation with the non-zero structure of the sparse matrix arising from the connectivity of a mesh composed of tetrahedral and hexahedral elements.

An implementation of this test has been developed following the methods described by Pommerell *et al* [8]. The method used is based on first mapping mesh nodes to processors followed by a renumbering of the nodes using "colouring" of the mesh. The implementation of the decomposition

[1] The efficiency is defined as $E_p = S_p/p$ where we have $p$ processors and $S_p$ is the speed up of the multiprocessor version over the same code on one processor, with fixed problem size.

method is again the simple slicewise approach which will give more interface nodes (and hence communication) than box type partitions. The colouring algorithm used is a greedy method with the extra condition that we want each processor to have the same number of nodes of each colour (referred to as a balanced colouring in [8]).

Communication is necessary to swap solution values corresponding to interface nodes. As before this is done by synchronous methods with no attempt to overlap communication with computation. Typical results for the iPSC/860 are shown in figure 4.

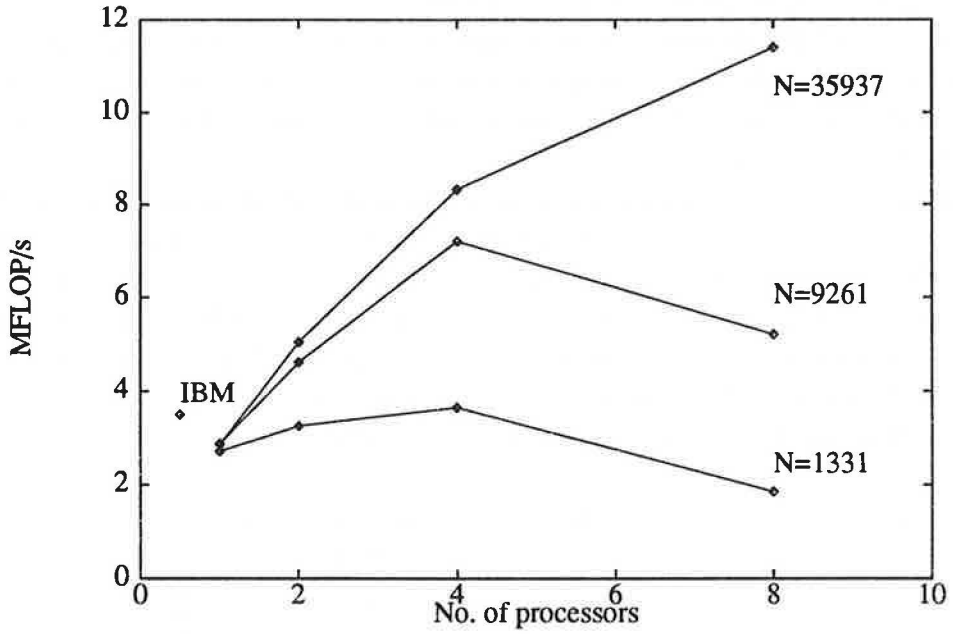


Figure 4: Results of BBS.2.4.2 on the iPSC/860 showing the computation rate (MFLOP/s) as a function of number of processors. The 4 curves correspond to different mesh sizes ($N$). The performance of the reference machine (IBM) is also shown.

Even for the largest problem (35937 nodes) we find the speed up with 8 processors is only 4.0, i.e. 50% efficient. For the 1331 node mesh the 8 processor version is actually slower than the uniprocessor version. In fact the results are significantly better in terms of actual speed up on the iPSC/2, e.g. $S = 6.3$ on 8 processors for $n = 9261$ compared to $S = 1.8$ on the iPSC/860. However even using 16 processors the best performance on the iPSC/2 was 1.4 MFLOP/s which is half the single processor result on the iPSC/860.

We conclude that the iPSC/860 suffers from insufficient communication bandwidth compared to its computational rate to treat this test well, except with when using a large mesh and few processors.

## 5 Class 3 test results

In the Class 3 BBS we look at whole library type routines. On many machines such operations would be provided in system library packages. While Intel do market packages of this type (e.g. the ProSolver family of linear algebra routines for iPSC machines), these were not available on the Daresbury machine. Hence we present results for some of our own implementations, which will not be as highly optimised as as those provided by the vendor. In this paper we present some typical results for the 2D FFT test and for the dense Gaussian elimination problem (BBS.3.3.1 and BBS.3.4.1).

## 5.1 Two dimensional FFT

This test (BBS.3.3.1) looks at the operation of performing an FFT on a set of values on a 2D grid. Since no library routine was available, we implemented the method described in [7].

The data in the 2D array is assumed to be divided amongst the processors such that each processor has $m/p$ rows assigned to it, where $m$ is the total number of rows in the array ($m \times n$) and $p$ is the number of processors in use. This distribution of data takes place before the timing measurement is started.

Each processor then performs 1D FFTs along the rows on the matrix that are available to it. These operations are independent and no communication is required.

Having completed the FFTs on rows, it is then necessary to perform the same operations on columns. This requires a matrix transpose amongst the processors so that the data is then distributed by columns. After the transpose is complete, the processors can then perform the FFTs on the columns (now stored as rows) independently.

The matrix transpose is performed with a public domain transpose subroutine that is described in [7]. The complete $m \times n$ matrix $A$ can be thought of as being made up of $p \times p$ sub-matrices $a_{ij}$, each of size $(m/p) \times (n/p)$. To perform the transpose, a processor has to exchange each sub-matrix, except the one on the diagonal, with the appropriate destination processor. After the sub-matrices have been exchanged, a local transpose must be made on each of them. The real 1D FFTs were performed (in double precision) with routines from the public domain FFTPACK code.

Figure 5 shows the results for 4 different problem sizes on the iPSC/860.

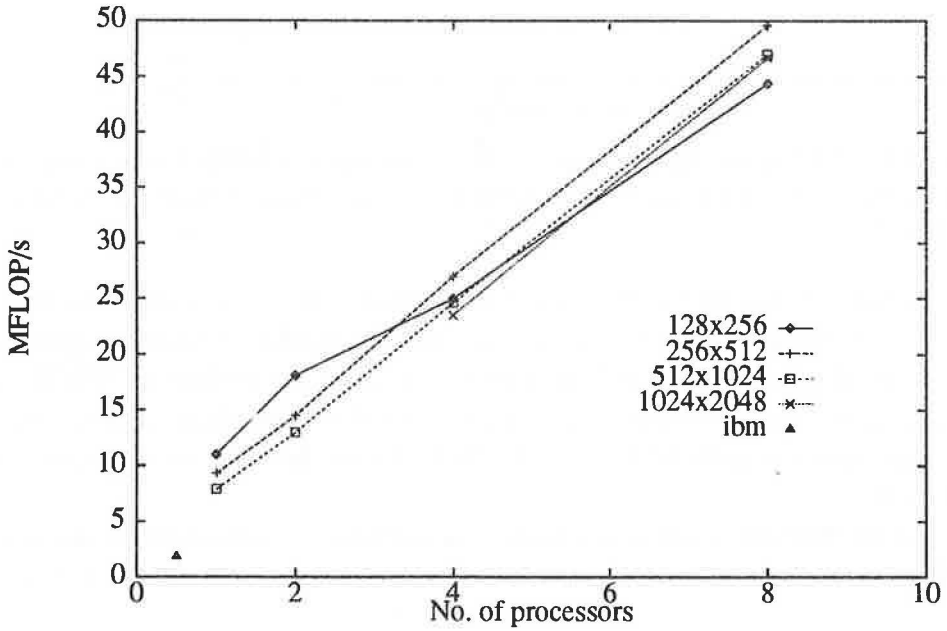


Figure 5: Results of BBS.3.3.1 on the iPSC/860 showing the computation rate (MFLOP/s) as a function of number of processors. The 4 curves correspond to different mesh sizes. The performance of the RS6000/320 is also shown.

There is significant variation in the single processor performance rate which may be due to the effects of the cache memory. The speed up that is obtained again varies with problem size. For the case of a $512 \times 1024$ array the speed up with 8 processors is 5.9 (74%) whereas the $128 \times 256$ problem gives a speed up of only 4 on 8 processors. The results for the iPSC/2 show greater speedup, but the

overall performance is always less than a single i860 node.

The result reported for the RS6000/320 is rather low and may be due to the choice of routine for the 1D FFTs, which was not the same as that used on the iPSC machines.

## 5.2 Dense matrix solution

BBS.3.4.1 involves the solution of the linear system $A\mathbf{x} = \mathbf{b}$ where $A$ is a dense matrix. For stability reasons at least partial pivoting is required in the solution process.

The implementation used for the iPSC machines is based on a cyclic mapping of rows to the available processors, e.g. with 2 processors the first will be assigned rows 1,3,5... etc. This should allow reasonable load balancing as the elimination proceeds.

Each time a row is to be eliminated the processor that holds it broadcasts it to all other processors. All processors then scan this row to find the pivot element and use the permuted row to update all the remaining rows that they are responsible for.

The backward substitution step is performed in a similar manner with each processor broadcasting solution values as they are found so that the others can use them in the row calculations assigned to them. Where possible we have used asynchronous communication so that computation can be performed while data is being sent.

Figure 6 shows the elapsed time to solve three problem sizes on the iPSC/860, the largest being a matrix of $400 \times 400$. Also shown are the time to solve on the IBM RS/6000, though these were obtained using a solver with full pivoting rather than partial pivoting, and so are not directly comparable.

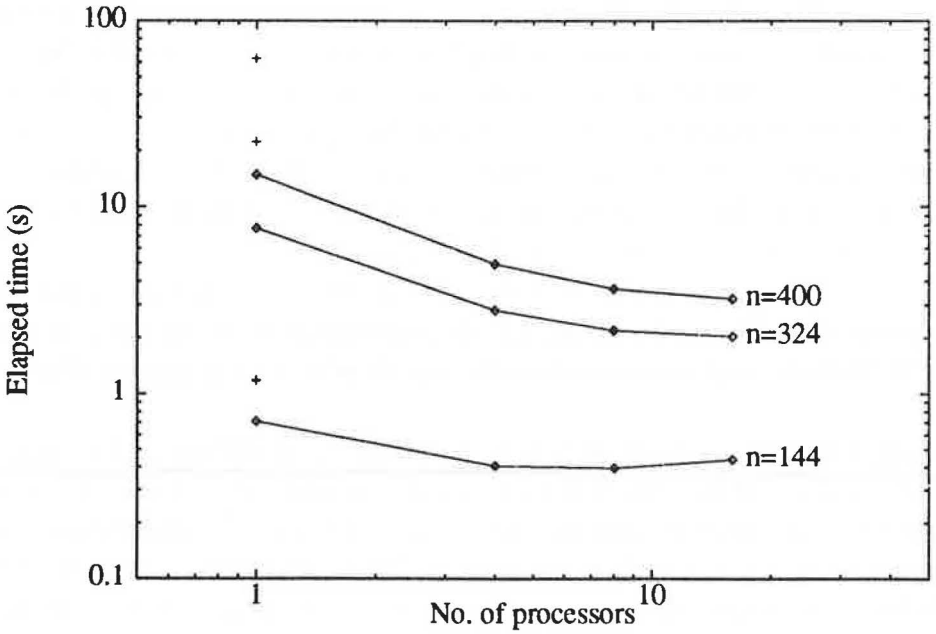


Figure 6: Results of BBS.3.4.1 on the iPSC/860 showing the computation time as a function of number of processors. The 3 curves correspond to different matrix sizes ($n \times n$). The time for the reference machine (+) is also shown.

These times can be converted to MFLOP/s rates using the approximation that the total number of FLOPS in the Gaussian elimination is $2n^3/3$. This gives about 2.9 MFLOP/s for the single i860 processor and a peak of 13.2 MFLOP/s for the 16 processor system for the $400 \times 400$ matrix. The speed up of the multiprocessor version is shown in Table 3.

| Matrix size | Number of processors | | | |
|---|---|---|---|---|
| | 1 | 4 | 8 | 16 |
| 144 | 1.0 | 1.7 | 1.8 | 1.6 |
| 324 | 1.0 | 2.7 | 3.5 | 3.8 |
| 400 | 1.0 | 3.0 | 4.1 | 4.6 |

Table 3: Speed up for BBS.3.4.1.

Again the performance of the iPSC/860 seems to be limited by the the communication and very large problems would have to be solved to use more than 8 processors efficiently with this implementation. We expect that a highly tuned library routine for Gaussian elimination should be able to give significantly better performance than 3 MFLOP/s on a single node, but the cost of communication would remain a serious bottleneck.

# 6 Conclusions

A selection of results from the BBS tests have been presented for the iPSC/2 and iPSC/860. In fact, we have focused mainly on the iPSC/860 since the iPSC/2 represents an older generation of machine and is no longer competitive in speed with current workstations which are also cheaper and easier to program.

The iPSC/860 is capable of offering very high levels of performance for highly parallel calculations since a single node is similar in performance to that of the reference machine (RS6000/320) and a full system may have up to 128 such nodes. However in several of the tests using problem sizes that are typical of many calculations performed today, we find that communication on the iPSC/860 is a serious bottleneck. This is not surprising since its communication hardware has the same peak rate as that of the iPSC/2 and yet the computation rate of each node has been greatly increased. A reasonable aim for a well balanced machine is that the time to send one word of data (8 bytes) is similar to the time it takes to compute such a value. The iPSC/860 takes of the order of 30 times longer to send a word than to calculate it (assuming 10 MFLOP/s and 2.8 Mbytes/s).

The Intel Paragon (the successor to the iPSC/860) should address this problem, since though there is a modest improvement in floating point performance, the communication rate has been increased almost 100 fold to 200 Mbytes/s. Such a machine should be significantly easier to program efficiently than the iPSC/860.

The BBS tests cover a range of operations which are related to calculations on unstructured meshes, though of course many do have much wider relevance, such the basic tests of Class 1 and operations such as the Gaussian elimination and the 2D FFT. The current set of tests does have some limitations, for example there is no test of all-to-one or all-to-all communication. Also some of the tests are rather similar to each other, which is why the project partners have only implemented a subset of the tests. Nevertheless, implementation of the BBS has given us a good understanding of the performance potential and limitations of the iPSC machines. We have also gained experience of using the communication routines provided by Intel which are quite comprehensive and straightforward to use.

One problem with the BBS which might be better dealt with in another release is the specification of the distribution of the data for the tests. At present the implementor of the test has considerable freedom in a few tests as to where the initial data is and where the answers are left. This makes results

difficult to compare fairly if two people have made different assumptions in their implementations. Part of this problem may be due to the different approaches that tend to be used on SIMD machines as opposed to MIMD machines.

## References


[1] Intel Corporation, "iPSC/2 and iPSC/860 User's Guide", April 1991, Doc. No. 311532-007.

[2] Intel Corporation, "iPSC/2 and iPSC/860 Programmer's Reference Manual", 1991.

[3] R.F. Fowler, B.W. Henderson and C. Greenough, "BECAUSE Benchmark Results for the iPSC/2 and iPSC/860", RAL Report (to be published), (1992).

[4] "Elementary loops and Elementary Tasks", BECAUSE Project Report, WP2D1 Part II (1991).

[5] "Identification of Critical Parts", BECAUSE Project Report, WP1D1 Part III (1991).

[6] "BECAUSE Benchmark Implementations and Results for the iPSC/2 and iPSC/860", BECAUSE Project Report, BECAUSE.RAL.92.5 (1992).

[7] D.Moody, "Benchmarking Intel iPSC processors", p. 115, in *Evaluating Supercomputers*, (ed. A. van der Steen), Chapman and Hall (1990).

[8] C. Pommerell *et al*, "A set of new mapping and colouring heuristics for distributed memory parallel processors", Technical Report No 90/1, ETH Zurich, 1990.