One Year with an iPSC/860

Eric Barszcz
NASA Ames Research Center
Moffett Field, CA 94035

Abstract

This paper describes experiences over the past year with an the Intel iPSC/860, a distributed memory MIMD parallel computer based on the Intel i860 floating point processor. The system at NASA Ames Research Center has 128 nodes, and a theoretical peak performance of over seven GFLOPS. This paper describes the system at Ames Research Center, talks about system stability, compiler performance measured by the NAS kernels, and results from a two-dimensional computational fluid dynamics application.

Intel iPSC/860 System

In spring of 1989 DARPA and Intel Scientific Computers announced the "Touchstone" project. This project calls for the development of a series of prototype machines by Intel Scientific Computers, based on hardware and software technologies being developed by Intel in collaboration with research teams at CalTech, MIT, UC Berkeley, Princeton, and the University of Illinois. One of the milestones is the "Gamma" prototype. On December 29, 1989, the Numerical Aerodynamic Simulation (NAS) Systems Division at NASA Ames Research Center took delivery of one of the first two Intel Touchstone Gamma prototypes. The system is marketed commercially as the Intel iPSC/860 and will be referred to as the iPSC/860 for the remainder of the paper. For a review of early experiences with the iPSC/860 at Ames Research Center and Oak Ridge National Laboratory, see [1] and [5] respectively.

The iPSC/860 system is based on the 64 bit i860 microprocessor by Intel [6]. The i860 runs at 40 MHZ (the initial system was delivered with 33 MHZ processors, which were upgraded to 40 MHZ). The theoretical peak speed is 80 MFLOPS for 32 bit floating point and 60 MFLOPS for 64 bit floating point operations. There are thirty-two 32-bit integer address registers, and sixteen 64-bit floating point registers (which may be used as thirty-two 32-bit floating point registers). Floating point register 0 is hardware wired to zero. This implies there are only fifteen 64-bit floating point registers that can hold non-zero values. The i860 also has an 8 kilobyte data cache and a 4 kilobyte instruction cache on-chip. The data path between cache and registers is 128 bits wide. The data path between main memory and registers is a 64 bits wide.

The i860 has a number of features to facilitate high execution rates. First of all, a number of important operations, including floating point add, multiply, and loads from main memory, can be pipelined. When pipelined floating point operations are used, they are segmented into three stages, and a new operation can be initiated every 25 nanosecond clock period (except for the 64 bit floating multiply instruction, which has two stages and is initiated every other clock period). Pipelined loads also are initiated every other clock period due to the speed of the dynamic random access memory (DRAM) used for main memory. Another advanced feature is that the i860 is a "super scalar" chip; multiple instructions can be executed in a single clock period. For example, a memory fetch, a floating point add and a floating point multiply can all be initiated in a single clock period.

A single node of the iPSC/860 system consists of the i860, eight megabytes (MB) of 70 nanosecond DRAM, and hardware for communication to other nodes. For every 16 nodes, there is also a unit service module to facilitate node diagnostics. The iPSC/860 system at NASA Ames consists of 128 computational nodes. The theoretical peak performance of this system is approximately 7.3 GFLOPS on 64 bit data.

The 128 nodes are arranged in a seven dimensional hypercube using the direct connect routing module and the hypercube interconnect technology of the iPSC/2 [7]. The point to point bandwidth of the interconnect system is 2.8 MB/sec per channel, the same as the iPSC/2. However the latency for message passing is reduced from about 350 microseconds to about 90 microseconds.

Following Bomans and Roose [4] we model the communication time T_{comm} by a least squares fit of the

Computer System	Length (bytes)	Latency (μ sec)	Time/word (μ sec)
iPSC/2	< 100	350	1.60
	> 100	660	2.88
iPSC/860	< 100	90	1.50
	> 100	180	2.88

Table 1: Linear Regression Messing Passing Parameters

data according to the model

$$T_{comm}(k) \quad = \quad t_{startup} + k * t_{send}$$

where k is the number of 8 byte words, $t_{startup}$ is the latency and t_{send} is the time per word. We obtain the data in Table 1 (the iPSC/2 numbers are from [4]). Thus we are able to confirm a considerably reduced message passing latency, which is mainly obtained through the increased speed of the i860 on the iPSC/860, when compared to the Intel 386/387 on the iPSC/2. These results are also confirmed in [3].

Attached to the 128 computational nodes are ten I/O nodes, each can store approximately 700 MB. So, the total capacity of the I/O system is thus about 7 GB. These I/O nodes operate concurrently for high throughput rates [8].

The complete system is controlled by a system resource module (SRM), which is based on an Intel 80386 processor. This system handles compilation and linking of source programs, as well as loading the executable code into the hypercube nodes and initiating execution.

The software environment of the iPSC/860 system is similar to that of the iPSC/2. The SRM runs Unix System V/386 and features the usual networking facilities including support for the Network File System (NFS). Also available is remote host software that allows a user to compile and run from a workstation. The individual nodes run a stripped down Unix-like kernel. Fortran-77 and C compilers, as well as an assembler and linker, are provided on the SRM. The system supports Fortran message passing commands for control of multiple processor execution.

System Stability

The Touchstone Gamma prototype arrived at the Numerical Aerodynamic Simulation (NAS) Systems Division at NASA Ames Research Center on December 29, 1989. Technicians from Intel Scientific Computers arrived on January 3, 1990 to install the system. On January 5, 1990 the system was made available to selected users for testing.

Month	Boards Replaced	Avg. Reboots per Day
January	6	
February	2	
March	3	
April	1	
May	7	2.45
June	several	2.87
July	1	N/A
August	0	2.41
September	2	N/A
October	0	0.90
November	1	1.88
December	0	1.56

Table 2: Stability of the iPSC/860 System

The initial system was installed with 33 MHZ i860 chips, UNIX System V/386 Version 3.2 software, and pre-production release 3.2 Fortran and C compilers from Green Hills. From March 12-15, the system was upgraded to 40 MHZ i860s.

Table 2 shows the number of node boards replaced by month and the average number of reboots per day by month. Typically, when a board was suspect, the system administrator would switch the board to a new position to see if it would be detected by diagnostics. If so, then it would be replaced. If diagnostics did not find it, the board would be marked as potentially bad and wait for future complaints. Defective boards are sent to Intel Scientific Computers for postmortem analysis.

On the iPSC/860 system, users can reboot the system. Occasionally, a user would reboot the system when their code deadlocked due to programmer error rather than a system problem or when they wanted "clean" timings. So not all reboots shown in the table are due to the system problems. January through April are blank because we did not start collecting reboot information until May.

New operating system software was installed March 19th which eliminated some problems that had been seen in user applications. It should be noted that not all codes that run correctly on and iPSC/2 compile and run on the iPSC/860. This includes some codes developed on the simulator.

On April 16th, the direct connect module (DCM) on the system resource module (SRM) failed and the boot block on the SRM hard disk was destroyed. The system was back up the next day and a second compile

engine was installed on the 20th to ease the load on the SRM.

In June, the system became very unstable. Some of this could be due to the influx of summer employees using the machine. The number of users who had logged in and attached to nodes went from 17 in May to 24 in June.

In early July the system also had problems. On July 4th, the concurrent file system (CFS) was corrupted and 'crestore' failed to work. The CFS was back up on the 6th without any files being restored. Technicians from Intel Scientific Computers arrived to stabilize the system. A comment in the log for July 16th remarks that the number of reboots is the same as for June, even though the number of active users increased to 26.

The only major events of August are that production 3.2 software was installed on the 20th and the alpha Fortran compiler from Portland Group was installed on the 23rd.

In October, the version 1.1 Fortran compiler from Portland Goup was installed. It is a major enhancement to the system because it is a cross compiler. Local users with a Sun 3 or Sun 4 can cross compiler and run on the iPSC/860. This frees up cycles on the compile engines for off site users and almost eliminates compiling on the SRM.

Currently, there are 84 user accounts on the SRM, of which 30 are activily being used. The average number of cubes allocated at any time is 6-7. This is averaged over 24 hours per day, 7 days per week. This is very high considering that a maximum of 9 cubes can be allocated at any time.

One of the problems that is still outstanding is that the SRM crashes with "kernel panics". Nothing shows up when diagnostics are run and it only seems to occur when the system is heavily loaded.

Fortran Compiler Performance

The Fortran compiler provided on the initial NASA Ames iPSC/860 system was the pre-production release 3.2 produced by Green Hills. Although it had some scalar optimization options, it did not take advantage of advanced features of the i860 such as the pipelining of floating point operations and the utilization of multiple functional units. As a result, single node Fortran performance was poor.

Some results of tests using the NAS Kernel Benchmark Program are shown in Table 3. This benchmark assesses the performance of a computer on seven subroutines that are typical of computational fluid dynamics computations done at NASA Ames [2]. The overall single node performance figure of 0.98 MFLOPS (64 bit), which is only about 1.6% of the theoretical peak performance of the i860 on 64 bit data, indicates there is considerable room for improvement.

Program	Error	Time	MFLOPS
MXM	3.43E-15	3.106	1.35
CFFT2D	1.26E-13	4.029	1.24
CHOLSKY	2.90E-12	1.737	0.64
BTRIX	5.53E-13	10.889	1.48
GMTRY	8.63E-14	138.550	0.82
EMIT	1.48E-16	7.909	2.86
VPENTA	1.19E-14	0.550	1.18
TOTAL	3.68E-12	166.770	0.98

Table 3: Single Node NAS Kernel Performance Results

These figures were obtained using the pre-production 3.2 compiler from Green Hills and compiled with no optimization. When compiled with all optimizations enabled (-OLM), the first three figures increased to 5.39 MFLOPS, 3.77 MFLOPS, and 1.71 MFLOPS, respectively, but the remaining tests did not complete, most likely due to a bug in the compiler. By comparison, the overall single node performance figure on the Cray Y-MP for this benchmark is 97 MFLOPS with no tuning and 160 MFLOPS with minor tuning.

It is important to note that the average MFLOPS, given on the last line in Table 3, is based on the total number of floating point operations performed and the total time to perform them and not by averaging the MFLOPS rating for each kernel.

With the installation of production release 3.2 of the operation system software in August, the kernels ran about 17% slower than with the previous operation system software. It turns out there are different alignment requirements for the 80386 and i860 processors and some sites have hybrid systems composed of i860 and 80386 compute nodes. When compiling for a homogeneous i860 system, the -Z618 compiler flag assumes the proper alignment and most of the lost performance is recovered. Also, the intrinsic libraries were changed. The new libraries are "safe" (give correct results all of the time) but slower, whereas the old libraries were "unsafe" but faster.

Table 4 contains single node performance results of the NAS Kernels when compiled using the Green Hills Fortran compiler with -OLM and -Z618 compiler flags. As one can see from the table, performance has improved and all of the kernels now compile and run with full optimization. There are two anomalies in the table. First, some of the error values have changed. This is probably caused by changes in the intrinsic li-

Program	Error	Time	MFLOPS
MXM	3.43E-15	0.691	6.07
CFFT2D	1.26E-13	1.247	3.99
CHOLSKY	2.88E-12	0.846	1.31
BTRIX	2.50E-13	6.063	2.66
GMTRY	2.60E-13	112.703	1.00
EMIT	1.33E-15	9.530	2.37
VPENTA	8.45E-15	0.427	1.52
TOTAL	3.53E-12	131.507	1.24

Table 4: Single Node Green Hills Compiler Results (-OLM -Z618)

Program	Error	Time	MFLOPS
MXM	3.43E-15	0.711	5.90
CFFT2D	1.26E-13	1.206	4.13
CHOLSKY	2.88E-12	0.885	1.25
BTRIX	2.50E-13	6.524	2.47
GMTRY	3.96E-14	113.279	1.00
EMIT	1.33E-15	8.798	2.57
VPENTA	8.45E-15	0.458	1.42
TOTAL	3.31E-12	131.861	1.24

Table 5: Single Node Alpha Portland Group Compiler Results (-O2)

braries. Second, EMIT actually runs slower with the new software and optimizations. When compiled with -OLM and without -Z618 it takes only 8.656 seconds compared to the 9.530 seconds given in the table. No explanation for this is known at this time.

Included with the production release 3.2 was an alpha version of a Fortran compiler from Portland Group. The compiler has three levels of optimization, no optimization specified, -O2, and -O3. Table 5 contains single node performance results from the NAS Kernels compiled with -O2 compiler flag. Results using -O2 and -O3 compiler options are mixed, neither option is best for all kernels with -O2 having better overall results. As before, the overall MFLOPS rating is based on the total number of floating point operations performed and the total time for all kernels.

Comparing the Green Hills compiler to the Portland Group compiler, there is no clear winner, they have the same overall MFLOPS rating. Even if one weights the MFLOPS ratings of all kernels equally, the Green Hills compiler averages 2.70 MFLOPS per kernel compared to the Portland Group average of 2.68 MFLOPS per kernel. Looking at individual kernels, the Green Hill compiler did better on 4 kernels, worse on 2 and tied 1. The only major difference is the Portland Group compiler only compiles for the i860 where the Green Hills compiler can generate code for the i860 and 80386.

Program	Error	Time	MFLOPS
MXM	3.43E-15	0.619	6.78
CFFT2D	1.26E-13	1.023	4.87
CHOLSKY	2.88E-12	0.948	1.17
BTRIX	2.50E-13	5.640	2.85
GMTRY	3.96E-14	116.422	0.97
EMIT	1.33E-15	8.851	2.55
VPENTA	8.45E-15	0.455	1.43
TOTAL	3.31E-12	133.958	1.22

Table 6: Single Node Beta Portland Group Compiler Results (-O3)

In October, the beta version of the Portland Group compiler was installed. It represented a major step forward, not in terms of performance, but for system usability. The Portland Group beta compiler is a cross compiler. Programs can be compiled on Sun 3s and Sun 4s and run on the iPSC/860. Combined with NFS mounted workstations, cross compilation removes much of the burden from the two 80386 compile engines (includes the SRM).

Some of the potential and difficulty of compiling for the i860 can be seen from the results of a double precision dot product coded three different ways, as shown in Figure 1. The top two curves are assembly coded where one vector is loaded from cache and the other from main memory. Curve three is also assembly coded but bypasses cache and loads both vectors from main memory. The fourth curve is a Fortran coded dot product compiled, using the pre-production Green Hills compiler in March, with full optimization.

When one vector is loaded from cache (i.e. the top two curves), the dot product peaks at about 27 MFLOPS. As the vector length exceeds the cache size (8 KB = 1000 words), performance drops off dramatically. With a stride of two, only half of the data in cache is usable and so performance drops off when the vector length exceeds 512 words. The Fortran coded dot product also shows the effect of cache and peaks at 8.7 MFLOPS. Since the effect of cache is seen at a vector length of 512 with a stride of 1, both vectors must be loaded from cache implying the compiler does not use pipelined loads. The current versions of the Fortran compilers use pipelined loads.

Curve three is the assembly coded version that bypasses cache. It remains flat after an initial startup and

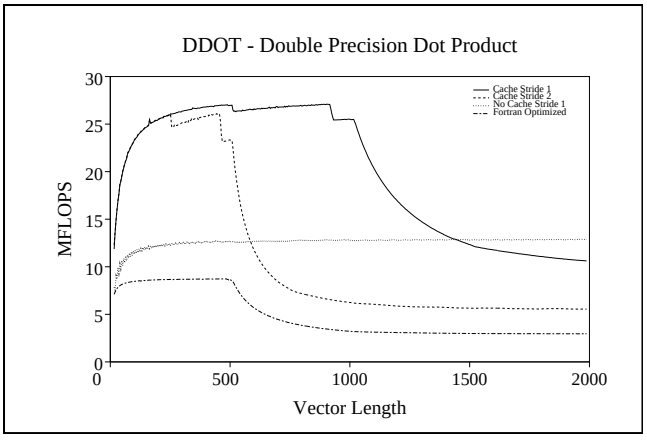


Figure 1: Double Precision Dot Product.

runs about 13 MFLOPS independent of vector length. In fact, it is the fastest dot product for vectors longer than 1500.

From the above results, it is clear that the compiler does not yet take advantage of all of the advanced features of the i860. Certainly there is much work to be done on the Fortran compiler to enable it to effectively utilize these features but it will be difficult or impossible for the compiler to always implement the best algorithm.

Multiprocessor Application Performance

In this section, performance of ARC2D, a computational fluid dynamics (CFD) program that has been ported to the Intel iPSC/860 at NASA Ames, is discussed and analyzed. This version of ARC2D solves 2-D Euler equations based on the diagonal form of the Beam and Warming implicit approximate factorization algorithm [9] and is capable of treating general 2-D geometries in either time accurate mode or accelerated non-time accurate steady-state mode. It was ported to the iPSC/860 by Sisira Weeratunga, a CSC contractor to the NAS Systems Division at Ames Research Center who also provided the information for this section.

Due to less stringent stability bounds and the consequent ability to obtain solutions which require fine grid spacing for numerical resolution, an implicit time integration technique is implemented in ARC2D. Although time differencing can be either first or second order accurate, only the former is required if steady state solutions are of interest.

The diagonalized form of the 2-D Euler equations has scalar tridiagonal or pentadiagonal inversions in place of block tridiagonal or block pentadiagonal inversions, without sacrificing the accuracy of the steady-state solution. In order to overcome the numerical instability due to nonlinear interactions, artificial dissipation terms are added to the implicit scheme. The nonlinear artificial dissipation model chosen is a one in which second and fourth order dissipation are combined with appropriate coefficients to produce a scheme with good shock capturing capabilities [10]. A linearized form of the artificial dissipation model is added to the diagonal factors, which necessitates the use of scalar pentadiagonal solvers. This produces an efficient, stable and convergent form of the implicit factored algorithm for steady state solution of 2-D Euler equations.

The concurrent implementation of the ARC2D algorithm is achieved by decomposing the computational domain into rectangular subdomains, and mapping them onto the hypercube using a 2-D binary reflected Gray code. Pipelined Gaussian elimination is then used to solve the systems of equations in both directions.

Pipelined Gaussian elimination with rectangular subdomains is found faster than in-processor Gaussian elimination with stripwise subdomains followed by a transpose to rearrange the data for solution in the second dimension.

Global exchanges are required for computing the 2-norm of the residual of the continuity equation and the number of supersonic points in the flow field, which are used to monitor convergence.

Performance of the resulting implementation is summarized in Table 7 [12]. For each problem size the following is given, seconds per time step, MFLOPS based on the CRAY hardware performance monitor, efficiency where $Efficiency(\%) = ((MFLOPS$ on one processor$)/N*(MFLOPS$ on N processors$))*100$, and single node performance of a CRAY-2 and a CRAY-YMP on the same problem. As the problem size increases, 64 nodes of the iPSC/860 performs as well as or better than the CRAY-2 and achieves over 70% of the performance of the CRAY-YMP on the 320x128 problem.

Conclusion

With the Intel iPSC/860 system, multi-GFLOPS peak floating point performance is now available on a MIMD hypercube computer system. Initial performance results indicate that a significant fraction of this peak performance may be obtained on some specialized applications, particularly those that can be implemented with algorithms and techniques that possess a high degree of data locality [11]. For the larger class of applications that do not possess high degrees of data locality, performance rates will be limited by both the restricted bandwidth between the processor and main memory on the individual nodes and by the restricted

Problem Size		Intel iPSC/860 No. of Processors							CRAY-2	CRAY-YMP
		1	2	4	8	16	32	64		
(192x64)	Sec./Step	4.1	2.1	1.11	0.61	0.35	0.22	0.14	0.15	0.08
	MFLOPS	3.0	5.9	11.2	20.3	35.6	56.7	86.4	82.0	163.0
	Efficiency(%)		98	92	83	73	58	44		
(256x80)	Sec./Step		3.49	1.82	0.98	0.54	0.33	0.20	0.26	0.13
	MFLOPS		6.0	11.4	21.0	39.0	64.0	101.0	79.0	161.0
	Efficiency(%)		99	95	88	81	67	53		
(320x128)	Sec./Step			3.62	1.90	1.02	0.57	0.34	0.48	0.24
	MFLOPS			11.6	22.0	41.0	73.0	123.0	86.0	172.0
	Efficiency(%)			97	92	85	76	64		

Table 7: ARC2D Performance

communication bandwidth between nodes. For both classes of applications, performance rates for the time being are considerably lower than ideal due to immature Fortran compilers.

On the other hand, such limitations are typical of an young system. Hopefully future developments, both hardware and software, will alleviate some of these bottlenecks and permit broad classes of scientific computations to run at supercomputer speeds.

Acknowledgment

The author acknowledges the valuable information and assistance provided by Victor Jackson and David Scott of Intel Scientific Computers, and Leigh Ann Tanner of NAS for keeping the machine running and access to her system logs.

References

1. Bailey, D. H., et al., "Performance Results on the Intel Touchstone Gamma Prototype", *Proceedings of the 5th Distributed Memory Computing Conference*, April 1990, p. 1236 - 1245.

2. Bailey, D. H., and Barton, J. T., "The NAS Kernel Benchmark Program", *NASA Technical Memorandum* 86711 (August 1985).

3. Bokhari, S., "Communication Overhead on the Intel iPSC-860 Hypercube", *ICASE Interm Report* 10 (May 1985).

4. Bomans, Luc and Roose, Dirk, "Benchmarking the IPSC/2 Hypercube Multiprocessor", *Concurrency*, vol. 1 (1989), p. 3 - 18.

5. Heath, M. T., Geist, G. A., and Drake J. B., "Early Experience with the Intel iPSC/860 at Oak Ridge National Laboratory", *ORNL Technical Memorandum* 11655 (September 1990).

6. *i860 64-Bit Microprocessor Programmer's Reference Manual*, Intel Corporation, Santa Clara, CA, 1990.

7. *IPSC/2 User's Guide*, Intel Scientific Co., Beaverton, OR, 1989.

8. Lou, Z. C., "A Summary of CFS I/O Tests", *NASA Ames Research Center, Applied Research Office Report* RNR-90-020 (October 1990).

9. Pulliam, T. H. and Chaussee, D. S., "A Diagonal Form of an Implicit Approximate Factorization Algorithm", *Journal of Computational Physics*, vol. 39 (1981), p. 347 - 363.

10. Pulliam, T. H., "Efficient Solution Methods for the Navier-Stokes Equations", Lecture Notes for The Von Karman Institute for Fluid Dynamics Lecture Series: Numerical Techniques for Viscous Flow Computation in Turbomachinery Bladings, Jan. 20 - 24, 1986.

11. Scott, D. S., Castro-Leon, E., and Kushner, E. J., "Solving Very Large Dense Systems of Linear Equations on the iPSC/860", *Proceedings of the 5th Distributed Memory Computing Conference*, April 1990, p. 286 - 289.

12. Weeratunga, S., Private Communication.