TITLE: A PERFORMANCE COMPARISON OF THREE SUPERCOMPUTERS: FUJITSU VP-2600, NEC SX-3, AND CRAY Y-MP

AUTHOR(S): Margaret L. Simmons
Harvey J. Wasserman
Olaf M. Lubeck
Christopher Eoyang
Raul Mendez
Hiroo Harada
Misako Ishiguro

SUBMITTED TO: Supercomputing '91

CONF-911164--1

A Performance Comparison of Three Supercomputers: Fujitsu VP-2600, NEC SX-3, and CRAY Y-MP.

Margaret L. Simmons, Harvey J. Wasserman and Olaf M. Lubeck

Computing and Communications Division
Los Alamos National Laboratory
Los Alamos, New Mexico 87545

Christopher Eoyang and Raul Mendez

Institute for Supercomputing Research
1-13-1 Kachidoki, Chuo-ku, Tokyo 104, Japan

Hiroo Harada and Misako Ishiguro

Japan Atomic Energy Research Institute
Tokai - Mura, Maka - Gun, Ibaraki, 319-11, Japan

ABSTRACT

The performance of two second-generation supercomputers, the NEC SX-3 and the Fujitsu VP2600, is analyzed using the Standard Los Alamos Benchmark Set, the Mendez Fluid Dynamics Codes, and some highly vectorizable production-type codes from Los Alamos. For comparison, data are also given for a single processor of the CRAY Y-MP8/264. Factors affecting performance such as memory bandwidth, vector register organization, and the effects of multiple vector pipelines are examined. On a highly vectorizable code that can take advantage of multiple vector pipes, the SX-3 and VP2600 are faster than the single CRAY Y-MP processor by factors of seven to eight.

Introduction

The first vector supercomputers manufactured by a non-U.S. company were the Fujitsu VP-200, the Hitachi S810/20, and the NEC SX-2, which appeared in 1984. An early performance evaluation of the VP-200 and the S810/20, using benchmark codes designed to represent the Los Alamos National Laboratory (LANL) computing workload, showed that the VP-200 could be two to three times as fast as a (9.5-ns) CRAY X-MP/24 processor for highly vectorized codes, although the scalar performance of the two machines was comparable.[1] The S810/20 did not perform as well as the X-MP and the VP-200, largely because of a longer central processing unit (CPU) clock period and poorer scalar performance.

In 1985 the LANL codes were used to test the NEC SX-2, which executed "as fast or faster than any other existing single-processor" computer.[2] SX-2 performance on short vectors was 1.5 to 3 times that of a single (9.5 ns) CRAY X-MP/48 processor, and 2 to 4 times that of the X-MP on longer vectors. Scalar performance of the two processors was comparable when the X-MP used the CFT77 compiler, although the X-MP was half as fast as the SX-2 when it used the CFT1.14 compiler.[2]

The Hitachi S820/80, introduced in 1987 and regarded as a second-generation supercomputer, had essentially the same architecture as the

S810/20, but used improved chip and wiring technology. Because of its improved clock rate, it performed almost twice as fast as the NEC SX-2 on highly vectorized codes in the LANL set,[3] although its scalar performance was slightly worse than that of the SX-2 and about equal to that of a single CRAY X-MP/416 (8.5-ns) processor.

In this paper we examine the performance of two more second-generation supercomputers, the Fujitsu VP2600 and the NEC SX-3, introduced in 1990. For comparison purposes, we include results obtained using a CRAY Y-MP. The Y-MP is a third generation Cray computer introduced by Cray Research in 1988 and first benchmarked by the Los Alamos group in 1988[5]. Although the Cray C-90, a successor to the Y-MP, is expected soon, it has not yet been announced by Cray Research; therefore, we use data on the older Y-MP.

Machine Architecture

Although the architecture of the SX-3 and the VP2600 have been described in numerous publications[6-7], we include a short summary here. We omit a detailed explanation of the CRAY Y-MP architecture, since it is not significantly different from that of the X-MP, descriptions of which also have appeared.[8-9]

NEC SX-3 The general configuration of the SX-3 system is similar to that of the SX-2, in that a dual arrangement of control processor and arithmetic processor (AP) is used. However, unlike the SX-2, system functions in the SX-3 are now handled by the arithmetic processors, which previously only handled computation. The SX-3 control processors are now dedicated to I/O functions.

An SX-3 AP consists of separate scalar and vector sections, each with its own arithmetic functional units. The scalar section employs a RISC architecture in combination with 64-KBytes of cache memory and 128 64-bit scalar registers. The SX-3 clock period is 2.9 ns, although scalar instructions issue in 5.8 ns, or every other clock period. Note that the SX-2 issued 32-bit operand instructions every clock period (6.0 ns) and issued 64-bit operand instructions every two clock periods (12.0 ns).

The SX-3 vector processor is equipped with one or more functional unit sets, each of which contains two add/shift pipelines and two multiply/logical pipelines operating at a clock period of 2.9 ns. Since each pipeline is fully segmented, a single pipeline set is capable of producing a total of four floating-point results every 2.9 ns. We tested an SX-3 configured with four functional unit sets yielding a maximum of 16 results (eight add and eight multiply) in each minor clock period. In fully-configured models of the SX-3 there are 72 vector registers, each with a capacity of 256 64-bit words.

The main memory of the SX-3 has a maximum capacity of 2 GBytes and a maximum (64-bit) interleave factor of 1,024. There are scalar cache load and store ports. For vector memory references there are three ports (two load and one store), each capable of transferring four 64 bit words per clock period. In comparison, the SX-2 had one port to memory, which could be used either for loads (eight words wide), or stores (four words wide). On a four-processor SX-3, there are a total of six ports; thus three ports are shared by two processors, and the effective bandwidth can be halved when both processors are active. Our benchmark was carried out on an SX-3 containing only one active processor and 1 GByte of memory.

The SX-3 we tested was running a beta-release version (r1.01) of the Super-UX operating system, a version of System V UNIX with bsd 4.3 extensions. We conducted the entire benchmark at the NEC Fuchu, Japan, facility, by accessing the SX-3 remotely (using telnet) from a Sun Sparcstation and several X-terminals. The compiler was f77sx rev. 005. Initial tests were run during the week of December 10, 1990. Subsequently, some results were discovered to be tainted by the presence of compiler directives; therefore some tests were rerun during May, 1991.

Fujitsu VP2600. Members of the Fujitsu VP-2000 series[7] also have separate vector and scalar processors, as did their predecessors, the VP-200 and VP-400. The VP-2600 model can contain two scalar units and one vector unit, and the entire system operates with a clock period of 3.2 ns. We benchmarked a machine containing a single scalar unit. Two types of vector arithmetic pipelines are included, one for multiply and add/logical, and one dedicated to divide. In the 2600 model, four of the multiply/add pipelines are present and may be operating in parallel; thus, four multiply/add results can be produced each clock period. As did its predecessor, the VP2600 has dynamically reconfigurable vector register sets, although the capacity is increased to twice that of the VP-200. The 128 KBytes of vector register storage may be configured as 32-bit or 64-bit words and further configured into equal-length registers composed of whole words. For example, assuming 64-bit words, the range is from 8 registers holding 2048 elements to 256 registers holding 64 elements. The VP-2600 has a main memory capacity of 2 GBytes; the machine we used contained 1 GBytes. There are two data paths between the CPU and memory, each four (64-bit) words wide, and they are capable of simultaneously loading and storing data.

The VP-2600 tests were carried out using the Fujitsu OS IV/MSP batch operating system, which is similar to the IBM MVS system. Initial VP2600 tests were run in December, 1990, and March, 1991. The results were updated in May, 1991, using Fortran77/VP version V10L30. All VP-2600 tests were carried out at the Japan Atomic Energy Research Institute in Tokai. Cray Y-MP date were obtained in March, 1991, using CFT77 version 5.0 on LANL Machine R (SN 1054) and SN1033 at the Cray Research Facility in Mendota Heights, MN.

RESULTS: Primitive Vector Operations

Tables 1-3 list rates in millions of floating-point operations per second (MFLOPS) for selected vector operations as a function of vector length. For various reasons, data are not available at all vector lengths for all machines; hence the blanks in the tables.

These data are collected by running an inner loop over the vector length of interest and an outer loop that repeats the inner loop to minimize systematic error in the timing measurement. (Typically a total of one million operations are timed.) However, both the VP-2600 and the SX-3 have optimizations that defeat the intent of the benchmark by removing redundant computation or preloading operands. We elected to block the optimization by including a subroutine call in the outer loop that references the vectors involved in the operation. The time to make this call was measured separately and then subtracted from the operation time. We are

Table 1. Rates (MFLOPS) for Selected Vector Operations as a Function of Vector Length on the Fujitsu VP2600

Operation	*10*	*50*	*100*	*256*	*257*	*1000*	*5000*
V=V+S	23.0	108.3	195.2	399.8		773.4	
V=V+S(I=1,N,8)	18.8	63.3	95.6	126.8		146.7	
V=V+S(I=1,N,23)	19.8	89.6	165.2	325.1		500.3	
V=V*V	19.5	88.7	162.7	302.3		488.1	
V=V+S*V	39.0	177.4	354.8	643.4		1001.6	
V=V*V+V	33.9	162.5	300.3	540.7		887.4	
V=V*S+V*S	53.3	254.3	509.3	966.7		1501.9	
V=V*V+V*V	46.0	201.8	389.7	732.0		1046.0	
V=V(IND)+S	12.6	55.7	90.7	149.3		208.4	
V(IND)=V*V	16.3	69.7	111.5	163.9		198.9	

Table 2. Rates (MFLOPS) for Selected Vector Operations as a Function of Vector Length on One Processor of the NEC SX-3

Operation	*10*	*50*	*100*	*256*	*257*	*1000*	*5120*
V=V+S	24.6	121.4	233.1	469.0	327.7	798.7	1014.2
V=V+S(I=1,N,8)	14.0	60.2	102.5	168.7		276.1	
V=V+S(I=1,N,23)	14.4	71.8	140.0	311.0		526.9	
V=V*V	23.0	113.4	218.2	428.6	294.7	711.2	896.9
V=V+S*V	44.2	218.2	420.6	874.1	586.5	1401.5	1792.1
V=V*V+V	40.6	202.8	391.9	700.8	497.5	1181.3	1468.4
V=S*V+S*V	62.3	307.8	594.4	1203.4	815.0	1965.9	2500.0
V=V*V+V*V	55.0	269.3	502.0	946.4	661.1	1548.8	1904.8
V=V(IND)+S	11.0	46.1	70.9	104.8	79.2	130.1	158.3
V(IND)=V*V	12.8	50.4	74.3	104.8	76.8	115.2	134.3

aware that this fix may have prevented other optimizations, thereby reducing the measured speed of the machine. Nevertheless, it is necessary to do so, in order to provide a consistent basis for comparison.

Contiguous Memory Access. All three machines provide nearly equivalent contiguous-access performance at vector length 10 for all operations. However, the rates for the VP2600 and SX-3 increase rapidly with increasing vector length, and at vector length 5000, the VP-2600 is about 4-9 times faster than the Y-MP, and the SX-3 is about 6-12 times faster than the Y-MP, depending on the operation. Averaged over all the contiguous memory operations, SX-3 performance is about 2.8 times that of the Y-MP at vector length 100 and about 4.4 times that of the Y-MP at vector length

Table 3. Rates (MFLOPS) for Selected Vector Operations as a Function of Vector Length on One Processor of the CRAY Y-MP

Operation	*10*	*50*	*100*	*256*	*257*	*1000*	*5000*
V=V+S	21.9	88.4	100.2	126.6	117.8	142.8	148.1
V=V+S(I=1,N,8)	21.1	77.0	87.2	107.4	99.9	119.4	123.1
V=V+S(I=1,N,23)	21.1	85.9	96.3	124.0	115.6	141.1	146.7
V=V*V	20.3	82.3	93.4	119.6	110.8	138.5	144.2
V=V+S*V	40.2	156.8	187.9	233.7	209.7	262.1	271.5
V=V*V+V	33.7	109.3	122.0	157.8	148.4	169.0	172.2
V=V*S+V*S	51.5	153.0	177.7	203.4	188.1	214.3	217.5
V=V*V+V*V	50.5	145.6	167.2	200.9	183.5	213.1	216.7
V=V(IND)+S	20.5	53.6	58.1	75.5	68.3	83.4	89.0
V(IND)=V*V	17.9	46.4	51.0	65.7	61.6	70.4	74.2

256; these lengths dominate some of the applications codes discussed later.

It is instructive to compare the three machines' performance for the SAXPY operation (V3 = V2 + S * V1). This operation requires two vector loads and one store per arithmetic functional unit set, or two chimes on all three machines. Performance of the SX-3 on SAXPY at asymptotic vector lengths is about 1.7 GFLOPS, which is less than one-half the peak potential performance of the processor. Such degradation relative to peak is observed because the SX-3 does not have enough memory bandwidth to support SAXPY on eight sets of functional units if results are stored to memory, and therefore, four sets of functional units are not used. Observed SAXPY performance is so about 35% of peak performance. The Y-MP, on the other hand, can supply operands at a rate that keeps its single set of functional units fully loaded, and it achieves about 70% of its peak performance.

The VP2600 also lacks sufficient memory bandwidth to support eight sets of functional units, because it can only process one load and one store per functional unit set per clock period. Asymptotic performance of the VP2600 on SAXPY is about 1.1 GFLOPS, or about 25% of potential peak.

Note that of all the operations studied, the Y-MP achieves its highest processing rate on SAXPY. In contrast, the SX-3 achieves its highest processing rate for the operation V = S * V + S * V. Here, the SX-3 can use an additional set of functional units without making additional demands on memory. Thus, two chimes are required for three FLOPS and the processing rate increases to 2.5 GFLOPS. The Y-MP, however, being functional-unit limited with this operation, requires three chimes, and its rate decreases to about 217 MFLOPS.

Constant, Non-Unit Stride Memory Access. Comparison of rates for the operation V = V + S for vectors accessed contiguously with rates using stride-23 shows the effect of non-unit strides. With odd strides, the Y-MP achieves full performance for this operation, at short as well as asymptotic vector lengths. VP2600 performance degrades to about 85% of the contiguous rate at short vector lengths, and the degradation becomes worse as vector lengths increase, decreasing to about 60% at vector length 1000. On the SX-3, non-unit stride effects are nearly constant for all vector lengths, with a degradation to about 60% of the contiguous rate. The observed rates using stride 8 may be used to estimate the effect of memory bank conflicts. Of the three machines, the Y-MP is the least affected by memory bank conflicts, and the VP2600 is the most affected, with rates dropping to less than 20% of the contiguous case. However, even with the memory bank conflicts, rates for V = V + S using stride 8 on the VP-2600 are still slightly higher than those of the Y-MP, and SX-3 rates for this operation are significantly better than those of the Y-MP.

Irregular Memory Access. The Y-MP also maintains the highest relative performance for scatter/gather operations, with rates degrading to about 50-60% of the corresponding contiguous cases. VP-2600 performance degrades to about 25-40%, and SX-3 performance degrades to about 15% of contiguous performance. Absolute rates are highest for the VP2600, which suffers less degradation than the SX-3.

Effect of Stripmining. In the Tables, both the SX-3 and the Y-MP show the effect of stripmining, i.e., the overhead associated with reloading the vector registers after the maximum number of elements has been processed. For the Y-MP, this occurs every 64 elements; for the SX-3 it occurs every 256. Tables 2 - 3 show the effect at vector lengths 256 and 257. This effect results in about a 10% decrease in Y-MP rates for each strip, and a 30% decrease in SX-3 rates. The data given in Table 2 do not show the effect of stripmining on the Fujitsu VP-2600 because when the code measuring the vector rates is run, the maximum vector length is known to the compiler, which can then configure the VP2600 vector registers to their maximum length. Thus, all the vector lengths run will fit within the vector registers, and no stripmining is required. However, running the operations with a maximum vector length of 5000, for example, does affect the rates, because the stripmining software must then be included. With stripmining software included, rates at vector lengths 10 - 256 decrease by about 35%.

RESULTS: Application Benchmarks.

Table 4 lists execution times in seconds for the larger, more integrated benchmarks representing application codes; descriptions of the codes are given in Appendix A . Unless noted otherwise, only results from untuned codes are reported. Untuned means that only those changes necessary to get the codes to run are allowed, along with any compiler switches. A full discussion of tuned results is to be published in a separate paper.

The benchmarks may be divided into four basic classes of codes, depending on the importance of vectorization.

Non-vectorizable Codes. The SX-3 is about 1.3 times faster than the Y-MP on ESN and PHOTON, while the machines are basically about the same on GAMTEB. The similarity in the performance is probably because the SX-3 scalar-issue period of 5.8 ns is essentially the same as the Y-MP clock period. The VP2600 is faster than the Y-MP on all three codes, by factors of about 1.3 - 1.5; this performance advantage is probably due to the faster VP2600 clock period.

Table 4. Comparison of Benchmark Execution Times on Single Processors of NEC SX-3, CRAY Y-MP, and Fujitsu VP2600.

Code	NEC SX-3	CRAY Y-MP	Fujitsu VP2600
intmc	7.7	7.5	5.4
esn	5.9	7.6	5.8
photon	46.9	62.4	40.6
gamteb	2.3	2.8	1.6
twodant915	14.7	11.6	19.5
fft	1.4	2.2	1.5
twodant93	52.5	50.9	68.9
lss	1.2	2.2	3.4
matrix	11.3	11.9	16.1
hydro	6.0	9.1	4.5
wave	33.4	56.7	27.5
lss300	12.1	33.2	31.6
vortex	2.8*	10.0	nr
mhd2d	0.4*	1.0	10.8
baro	8.4*	40.1	10.9
vgam	0.46	0.38	0.42
x3d	15.7	29.6	21.7
neut32	75.0*	326.2	83.9
neut64	150.0*	700.5	166.2
pueblo32	1.6*	11.6	8.0
pueblo64	11.7*	97.4	160.6

* = Results are from December, 1990 measurements
no = No tuning done
nr = Not run

Partially Vectorizable or Vectorizable Codes with Short Vector Lengths. Although performance on the eight codes in this group is mixed, we wish to draw attention to four codes that most closely represent production codes in use at the Laboratory. Interestingly, although these codes were originally developed on Cray systems, performance of the SX-3 and VP2600 is significantly faster than the Y-MP on HYDRO and WAVE. This is consistent with the vector kernel results (above) for vector lengths 100 and 256, using constant strides and indirect addressing, which showed both the SX-3 and VP2600 to be faster than the Y-MP. The SX-3 also equals the performance of the Y-MP on TWODANT93. On TWODANT915, Y-MP performance exceeds that of the SX-3 and VP2600. The Y-MP is also faster than the VP2600 on TWODANT93; however, we are unable to account for these TWODANT trends at this time.

Vectorizable Codes with Intermediate Vector Lengths. Three codes in the Mendez suite have vector lengths that average about 200 - 500. On these codes the SX-3 runs about 2.5 - 5 times faster than the Y-MP. Note that the primitive vector operation tests mentioned above showed the SX-3 to be about four times faster than the Y-MP at vector length 256. On MHD2D the VP2600 is slower than the Y-MP by a factor of 11, because the VP2600 compiler fails to vectorize several critical loops. On BARO, the VP2600 is able to vectorize the code, and runs it almost 4 times faster than the single Y-MP processor.

Fully Vectorizable Codes with Very Long Vector Lengths. We begin with PUEBLO, for which a more detailed discussion of optimization levels is required. PUEBLO contains 24 loops whose bounds are unknown to the compiler. Without any additional information, the Cray CFT77 compiler vectorizes these loops conditionally, which is known to cause significant performance degradation as the bounds are checked each time the loop is run. As it turns out, Cray has a compile line option that directs the compiler to ignore the possible dependences created by the unknown loop bounds. We used this option, which comes under the "untuned" level of optimization. Without directives, the Fujitsu compiler does not vectorize these loops at all. The code as originally supplied to NEC contained Cray compiler directives that forced vectorization of these loops. Either the NEC compiler recognized the Cray directives or it was able to vectorize the loops anyway. The net result of all of this is that "untuned" results for Cray and NEC reflect vectorization of these loops while "untuned" for Fujitsu does not. At the untuned level, the SX-3 is about 7 - 8 times faster than the single Y-MP processor, while the VP2600 results are problem-size dependent; the VP2600 runs the smaller problem about 1.5 times faster than the Y-MP, but the Y-MP runs the bigger problem 1.6 times faster, again, because of lack of vectorization on the VP2600. The VP2600, using compiler directives in a "tuned" version of the code, runs 6 - 7 times faster than the (single-processor) Y-MP.

The SX-3 runs NEUT about four times faster than a single Y-MP processor. PUEBLO involves more floating-point computation than NEUT, which may account for its larger performance ratio relative to the Y-MP. On X3D, the SX-3 is about twice as fast as the Y-MP. This smaller performance ratio relative to the Y-MP may be due to the large number of scatter / gather memory references in X3D. The VP2600 shows comparable performance on NEUT, where it is also four times faster than the YMP. Its performance on X3D is also comparable; it is faster than the Y-MP (factor of 1.4) and slower than the SX-3 (factor of 1.5).

Summary

Although the NEC SX-3 and the Y-MP are both multiple processor machines, and the Fujitsu VP2600 has multiple scalar units, our comments in this report relate only to single processor results.

As we have known for some time, Japanese supercomputer manufacturers have chosen to upgrade their first generation machines by expanding their vector processing capabilities, i.e., by adding additional vector functional units. This is in contrast with the route taken by the American manufacturer, Cray Research, who retained the same number of vector pipelines found in the original CRAY-1 and increased the total number of processors instead. As a result of the work reported here, we see that, for single processors, the multiple vector pipes provided in the Japanese machines do provide impressive performance on our codes that are highly vectorizable and have long vector lengths. We have not yet assessed the effect of putting additional processors to work on these codes.

We observed a single NEC SX-3 processor running a code nearly eight times faster than a single CRAY Y-MP processor. On another code the SX-3 ran about four times faster than a single Y-MP processor. On these two codes, the Fujitsu VP2600 ran about 1.5 and 4 times faster, respectively, than the single-processor Y-MP, and thus the SX-3 was about 1.1 - 4 times faster than the VP2600. These codes are both well suited for multi-pipeline architectures: they are very highly vectorizable (>99%), and they have very long vector lengths (~32,000). Vectorizable codes in our benchmark suite that do not have very long vector lengths were apparently not able to take advantage of the multiple pipelines. Thus, their SX-3 performance relative to the Y-MP was about a factor of two better, which is probably more related to the difference in cycle time between the two machines.

Once again, the reader should be aware of the age differences between the Y-MP and the VP2600 and SX-3. Furthermore, note that any conclusions that may be drawn from the results presented in this report must pertain specifically to the workloads represented by the benchmark codes we used. Comparison with results based on other workloads must be made with caution.

References

1. Lubeck, O. M., Moore, J. W., and Mendez, R., "A Benchmark Comparison of Three Supercomputers: Fujitsu VP-200, Hitachi S810/20, and Cray X-MP/2," IEEE Computer, December 1985.

2. Lubeck, O. M.,, Moore, J. W., and Mendez, R., "The Performance of the NEC SX-2 and CRAY X-MP Supercomputers," Los Alamos National Laboratory document LA-UR-87-227 (1987).

3. Eoyang, C., Mendez, R. H., and Lubeck, O. M., "The Birth of the Second Generation: The Hitachi S-820/80," Proc. Supercomputing '88, IEEE Computer Society, pp. 296-303, 1988.

4. Lubeck, O. M., "Supercomputer Performance: The Theory, Practice, and Results," Adv. in Computers, 27, 1988.

5. H. J. Wasserman, "Los Alamos National Laboratory Computer Benchmarking 1988," Los Alamos National Laboratory report LA-11465-MS (December, 1988).

6. Watanabe, T., Matsumoto, H., and Tannenbaum, P.D., "Hardware Technology and Architecture of the NEC SX-3/SX-X Supercomputer System," Proc. Supercomputing '8, IEEE Computer Society, pp. 842-846, 1989.

7. Uchida, N., Hirai, Yoshida, N., and Hotta, K., "Fujitsu VP2000 Series," Proc. Compcon Spring '90, IEEE Computer Society, 1991.

8. M. L. Simmons and H. J. Wasserman, "Los Alamos National Laboratory Computer Benchmarking 1986", Los Alamos National Laboratory report LA-10898-MS (January, 1987).

9. Hockney, R. W. and Jesshope, C. R., "Parallel Computers 2," (Bristol, England:Adam Hilger, 1988).

Appendix A. Description of Benchmark Codes

INTMC: An integer Monte Carlo code containing almost no floating point arithmetic. The random number generator requires at least 32-bit integer operations. There is no I/O and all data are internally generated.

FFT: A fast Fourier transform (FFT) code that is highly vectorizable. This code measures the speed of single Fourier transformations. Because it executes many operations with short vector lengths, it is very sensitive to vector start-up times. FFT library routines supplied by all supercomputer manufacturers generally perform multiple FFTs at much higher execution rates than this benchmark code. No I/O is performed.

VECOPS: Tests rates of primitive vector calculations as a function of vector length. Vector operands and results are fetched from and stored to contiguous memory locations, except for four operations that involve gather/scatter. Typically one million floating point operations are timed.

VECSKIP: Performs the same operations as VECOPS. The vectors are accessed in noncontiguous memory locations with several values for the stride, which can be adjusted to test for performance during memory conflicts.

MATRIX: Basic matrix operations, including multiplication and transpose, on matrices of order 100. The code is highly vectorizable but not optimized for vector computers.

GAMTEB: A Monte Carlo photon transport code. This is a relatively small model code with a simple source and straightforward geometry. It is only slightly vectorizable.

VGAM: A fully vectorized version of GAMTEB written originally for the CDC CYBER-205. The code runs the same size problem and the same geometry as GAMTEB, but a different random-number generator is used. VGAM uses scatter/gather operations extensively. The initial vector length is 1000, but this grows and then shrinks as the computation progresses.

PHOTON: Monte Carlo photon transport code that uses the methods of GAMTEB, but with more complicated geometry, more materials, and more statistics gathered. It requires 64-bit arithmetic for its random number generator as does GAMTEB. It also does not vectorize.

LSS: A linear system solver from LINPACK for systems of equations of order 100. It uses the method of Gaussian elimination. Although it is fully vectorizable, it is not optimized for supercomputers. Library routines supplied by supercomputer manufacturers will achieve considerably higher execution rates.

LSS300: LSS using systems of equations of order 300.

ESN: A one-dimensional, discrete ordinates, particle transport code that solves the transport equation by the discrete ordinates method. The current algorithm implemented in ESN was developed by Wienke and Hiromoto. Particles are described by a flux, defined at each point in space and time, and the flux is a function of particle energy and direction of flight. The discrete ordinates method involves discretizing all these variables (space, time, energy, and angle) and applying an iterative solution scheme. There are 16 energy groups involved. The code does not vectorize.

HYDRO: A two-dimensional Lagrangian hydrodynamics code based on an algorithm by W. D. Schultz. HYDRO is representative of a large class of codes in use at the Laboratory. The code is 100% vectorizable. A typical problem is run on a 100 X 100 mesh for 100 time steps. An important characteristic of the code is that most arrays are accessed with a stride equal to the length of the grid.

WAVE: A two-dimensional, relativistic, electromagnetic particle-in-cell simulation code used to study various plasma phenomena. WAVE solves Maxwell's equations and particle equations of motion on a Cartesian mesh with a variety of field and particle boundary conditions. The benchmark problem involves 500,000 particles on 50,000 grid points for 20 timesteps; about 4 MW of memory are required. The predominant vector length is 256 and indirect addressing is an important mode of memory access.

NEUT: A highly vectorizable Monte Carlo neutron transport code. Two problem sizes may be run, one starting with 32k neutrons, the other with 64k neutrons. NEUT represents a Fortran77 version of Eldon Linnebur's (LANL Group X-7) Connection Machine Fortran code.

PUEBLO: A Lagrangian hydrodynamics code used to model point explosions in space. The code is highly vectorizable, although Cray compiler directives are currently included. A 32X32X32 grid or 64X64X64 grid may used.

X3D: A highly vectorizable 3-dimensional hydrodynamics code that uses the free-Lagrange method. X3D is a Fortran77 port of the Connection Machine version, developed by Harold
Trease of Los Alamos.

TWODANT: A two dimensional discrete ordinates particle transport code used for neutral particle transport. It includes a multigrid solver and is vectorizable to some extent.

BARO: Fluid dynamics code from the Mendez suite of benchmark codes.

VORTEX: Fluid dynamics code from the Mendez suite of benchmark codes.

MHD2D: Fluid dynamics code from the Mendez suite of benchmark codes.