The Journal of Supercomputing, 4, 153–167 (1990)

Performance Comparison of the CRAY-2 and CRAY X-MP/416 Supercomputers[1]

MARGARET L. SIMMONS and HARVEY J. WASSERMAN
Computing and Communications Division, Los Alamos National Laboratory, Los Alamos, New Mexico 87545

(An earlier version of this paper was presented at Supercomputing '88. Received December 1988; final version accepted February 1990.)

Abstract. The serial and parallel performance of one of the world's fastest general purpose computers, the CRAY-2, is analyzed using the standard Los Alamos Benchmark Set plus codes adapted for parallel processing. For comparison, architectural and performance data are also given for the CRAY X-MP/416. Factors affecting performance, such as memory bandwidth, size and access speed of memory, and software exploitation of hardware, are examined. The parallel processing environments of both machines are evaluated, and speedup measurements for the parallel codes are given.

Key words: performance, benchmark, supercomputer.

1. Introduction

In 1985, the first Cray Research Incorporated CRAY-2 supercomputer was installed at the National Magnetic Fusion Energy Computational Center (NMFECC). Since that time this series of machine has undergone many changes, both in hardware and software. This paper evaluates some of these changes by observing their effect on a series of computationally intensive benchmark codes. We measured the performance of three models of the CRAY-2 that differ in their common memory hardware. The first two models we measured had common memory implemented with dynamic random-access memory (DRAM) with chip access times of 120 and 80 nanoseconds (ns). These machines are Serial 2003, located at the University of Minnesota, and Serial 2011, located at the Air Force Weapons Laboratory in Albuquerque, New Mexico. The third model, Serial 2012, located at Cray Research, uses static random-access memory (SRAM) with a chip access time of 55 ns.

In Section 2, we present a brief outline of the architectural and functional features of the CRAY-2, with emphasis on those features that affect performance. For comparison, corresponding architectural features from another Cray Research product, the X-MP/416, are included. Later sections present benchmark data with single-processor and multiprocessor results discussed separately.

2. Comparison of Architectures

The CRAY-2 is a general-purpose parallel/vector supercomputer system. There are four central processing units (CPUs), each with vector and scalar capabilities. Up to 256 million

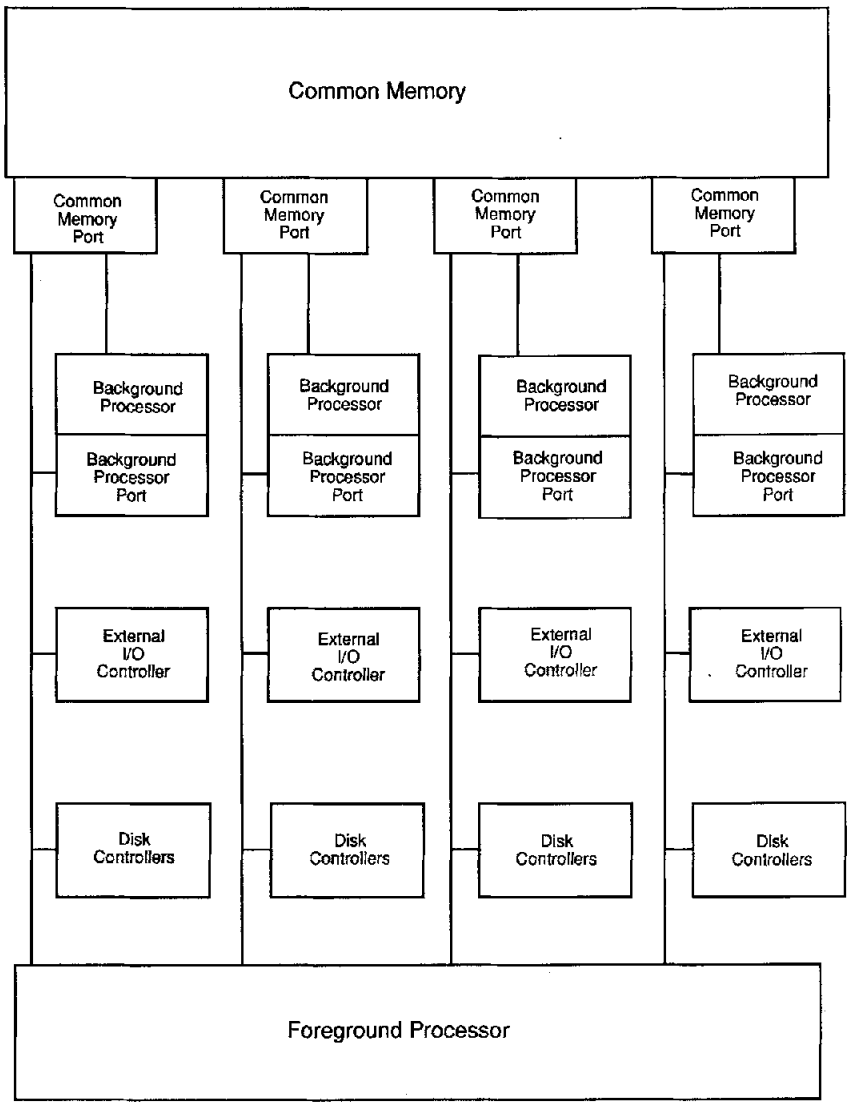

Figure 1. CRAY-2 mainframe configuration.

words of dynamic CMOS memory give the CRAY-2 one of the largest memory capacities of any supercomputer on the market today. For a schematic of the mainframe configuration, see Figure 1. For comparison, the X-MP is also a 4-CPU machine, each CPU having vector and scalar capabilities, but with a common memory of up to 16 million 64-bit words of static bi-polar memory.

2.1 CPUs

The CPU clock period on the CRAY-2 is 4.1 ns, while on the X-MP/416, the CPU clock period is 8.5 ns. The effect of this difference is not always as large as it at first seems. Instructions can issue from the instruction buffer on the X-MP every clock period (CP), while on the CRAY-2 the rate is one every other clock period. This gives the CRAY-2 an effective clock period of 8.2 ns with respect to instruction issue, nearly equal to that on the X-MP. After an appropriate start-up, however, arithmetic results are produced every CP on both machines.

The CPUs on both machines contain three sets of registers that serve as source and destination for computations in the functional units. These are address registers, scalar registers, and vector registers, referred to as A-, S-, and V-registers, respectively. In addition, the CRAY-2 has 16K words of local (or fast) memory that can be used by these registers as temporary storage. The access time between local memory and A- and S-registers is 5 and 4 CPs, respectively. The access time for V-registers is 8 CPs + length of the vector. Instead of local memory, the X-MP has an extra set of 72 temporary storage registers called B- and T-registers. Access times for these registers is 1 CP. The V-registers on the X-MP have no corresponding temporary registers. In addition to these registers, the CRAY-2 has eight semaphore flags to enable synchronization of common memory during multitasking. Only one of these semaphores can be assigned to a job. In contrast, the X-MP has five sets of shared registers (shared among the four CPUs), including 32 semaphores. Arithmetic on both machines is done in fully segmented (pipelined) functional units. This pipelining allows the functional units, some of which can also operate in parallel, to deliver a result every clock period, after a suitable start-up time. Chaining, which allows the output of one arithmetic operation to serve as the immediate input to a subsequent operation, is not available on the CRAY-2. There are also different numbers of functional units on the two machines: the X-MP has 14 while the CRAY-2 has 9, including the reciprocal square root unit. Table 1 gives some representative times for arithmetic operations. For a more complete explanation of the CPU organization, see [Kampe and Nguyen 1986].

2.2 Memory

As mentioned previously, the CRAY-2 has up to 256 million 64-bit words of common, or shared, memory, interleaved up to 128 ways. The memory is organized into quadrants with 32 banks in each quadrant. Each quadrant has a data path to four common memory ports, one for each processor. The four quadrants are accessed by the four processors in *phase time*. This means that each processor can access one particular quadrant every fourth clock period. The quadrants are accessed in a round robin fashion; that is, processor 1 can access

Table 1. Scalar floating point operation times.

Operation	CRAY-2	X-MP/416
Add	76 ns (19 CP)	51 ns (6 CP)
Multiply	76 ns (19 CP)	59.5 ns (7 CP)
Reciprocal	88 ns (22 CP)	119 ns (14 CP)

quadrant 1 at cycle 1, quadrant 2 at cycle 2, and so on, back to quadrant 1 at cycle 5. This arrangement has important implications for strided memory references, as discussed below. With one port per processor, the CRAY-2 can do one load or one store at a time. The X-MP, with its four ports per processor, can do two vector loads, one vector store, and one I/O memory reference simultaneously.

While the large size of the memory on the CRAY-2 is an asset, the memory cycle time on the early CRAY-2 models of 234 ns (57 CPs) was slow enough to be a detriment. Cray's first solution to the memory speed problem was pseudobanking, a technique that allows access to a physical memory in less time than the memory chip cycle time. Cycle time is made up of two parts, access time and off-chip time. The logic chips are busy for a time equal to the access time, while the memory chips are busy for an additional time equal to the off-chip time. Pseudobanking uses the simple trick of addressing alternate planes of chips within the module. This can be done in a time equal to the access time, effectively reducing the cycle time by nearly half. Using this approach the effective cycle time on the 256-Mword DRAM decreased from 57 CPs to 33 CPs [Numrich 1985]. Pseudobanking is only needed on CRAY-2s with DRAM. Later solutions have involved the use of faster memory chips and static rather than dynamic memories. The X-MP uses chips with a scalar memory access time of 14 CPs (119 ns).

Several factors affect the rate of data transfer between common memory and the vector registers on the CRAY-2. The first is the rate of instruction issue for vector reads and writes. With only one common memory port per processor, each read or write instruction must wait for the port to be free before it can issue. If one word transfers per clock period, then the next instruction can issue $VL + 8$ CPs later [Numrich 1985], where VL is the requested vector length. The minimum transfer time per word, $T(\text{min})$, is approximated by

$$T(\text{min}) = (VL + 8)/VL \ CPs. \tag{1}$$

For a vector length of 64, this time is 72/64 or 1.125 CP/word, giving a maximum transfer rate of 217 Mwords/s (assuming a 4.1-ns clock). This rate is highly optimistic and assumes no quadrant or bank conflicts.

Memory conflicts also reduce the data transfer rate. Memory conflicts on the CRAY-2s occur at two levels: quadrant conflicts and bank conflicts. Quadrant conflicts are caused by the difference in the rate at which memory-request addresses arrive at the port and the rate at which the port can process these requests. Recall that each memory quadrant on the CRAY-2 can be addressed by each processor only once every 4 CPs. The time between memory requests to the same quadrant is called the quadrant period. Quadrant periods of four cause no conflicts to occur, while periods of two or one can cause conflicts. Vectors with odd strides, including strides of one, have a quadrant period of four, and thus cause no conflicts. Even strides, however, cause conflicts of varying severity. The worst case is a stride divisible by four; here the quadrant period is one. Even in the absence of other conflicts, memory quadrant conflicts can cause performance degradation.

Bank conflicts, like quadrant conflicts, are caused by attempts to access data in the same bank within too small a time period. The bank conflict effect is a function of bank cycle time and number of banks. See Table 2 for a list of cycle times for the machines we tested.

Table 2. Memory characteristics for three tested models of the CRAY-2.

Model	Chip Access Time (ns)	Memory Cycle Time (CP)	Memory Type	Memory Size (Mword)
2003	120	57	Dynamic	256
2011	80	45	Dynamic	256
2012	55	41	Static	128

When a bank conflict occurs, the address in the quadrant buffer requires more than the 4-CP quadrant access time to clear. Memory backup then occurs because these quadrant buffers remain full until the requested bank is free.

3. Description of Benchmark Programs

The Computing and Communications Division at Los Alamos National Laboratory maintains a set of portable benchmark programs representing characteristic tasks that a large supercomputer would be required to run at the Laboratory. This benchmark set has been run on a wide range of both scalar and vector machines [Brickner et al. 1986; Griffin and Simmons 1984; Lubeck et al. 1985, 1987; Simmons and Lubeck 1986; Simmons and Wasserman 1987; Wasserman et al. 1987]. A database is maintained containing results of past runs of these programs on a variety of computers. A report from the National Research Council [1986] has characterized supercomputer benchmarks in terms of a hierarchy. Using the Council's characterization, the Los Alamos benchmark set consists of tests at the levels of hardware demonstration programs, basic routines, and stripped-down applications. A description of the codes is given in the appendix. Additional information can be found in [Wasserman 1988]. The programs are coded in ANSI Fortran for portability and typically can be run on a new machine with little or no change. Execution rates will be indicative of the potential initial usefulness of a new machine.

4. Single-Processor Results

The benchmark of CRAY-2 Serial 2003 took place in July 1987, while Serial 2011 was measured in October 1987, and Serial 2012 was measured in January 1988. The X-MP results were obtained in November 1987. Two of the CRAY-2s, Serial 2003 and 2012, ran UNICOS, a UNIX-like operating system; Serial 2011 and the X-MP/416 ran the Cray Timesharing System (CTSS). All measurements were made during dedicated time on a single processor.

4.1. Comparison of Three Types of CRAY-2 Hardware

Table 3 shows the effect of the faster memory hardware on our benchmark codes. The results for the 80-ns DRAM CRAY-2 are not always consistent with results from the other two machines. That is, the times for some benchmarks increase in going from the 120-ns

Table 3. Comparison of benchmark execution times (in seconds) for CRAY-2 (one processor) showing effect of hardware[a]

Code	120-ns DRAM (SN 2003)	80-ns DRAM (SN 2011)	55-ns SRAM (SN 2012)
FFT	10.7	10.5	9.6
GAMTEB	5.3	7.5	4.7
SCALGAM	104.6	100.9	92.2
LSS	9.5	9.5	8.9
MATRIX	61.9	59.4	57.3
INTMC	20.8	20.6	18.6
HYDRO	79.7	79.6	68.8
WAVE	186.6	203.0	174.6
ESN	NR	21.5	22.9
MCNP	NR	94.1	77.8

[a]CFT77 version 1.3 compiler.

CRAY-2 to the 80-ns CRAY-2. We believe this is due to different implementations of the CFT77 compiler and, in particular, the implementation of CFT77 under CTSS on Serial 2011. For this reason, and also because we wish to illustrate the maximum performance gain that could be realized from faster memory, in this discussion we focus on the difference in performance between the 120-ns DRAM CRAY-2 and the 55-ns SRAM CRAY-2S. Speedups due to the static memory are in the range of 7–16%. The two scalar codes SCALGAM and GAMTEB show identical speedups of 13%. HYDRO, which is nearly 100% vectorizable, shows the largest speedup. Note that a twofold change in memory chip access time should not yield anything close to a twofold speedup in the codes. The more pertinent hardware feature is the memory latency, which is the time to do loads from common memory. On the DRAM machine, the scalar access latency is 59 CPs, while on the SRAM machine, the latency for scalar loads is 43 CP. Thus, the maximum speedup we could observe here is about 37%. That the maximum observed speedup is still smaller than this may suggest that the compiler could hide some of the memory latency, perhaps by more use of the local memory.

4.2. Comparison of Compilers

Table 4 shows a comparison of execution times on the fast memory CRAY-2S (Serial 2012) using the two CFT77 compilers (1.3) and (2.0).[2] Version 2.0 yields a dramatic improvement on some of the codes. The FFT code speeds up by a factor of 2.3 relative to CFT77 1.3. All previous versions of CFT77 vectorized several loops in FFT *conditionally*; the repeated execution of the conditional code at run time caused much slower execution rates. In FFT the conditional code is generated because a loop bound is passed as an argument to a subroutine. These loops are now fully vectorized in version 2.0.

Using CFT77 2.0, HYDRO speeds up by 35%. HYDRO contains one minor loop that conditionally vectorized with CFT77 1.3 and now fully vectorizes with CFT77 2.0. HYDRO also has three loops, in the time-consuming subroutines VSETUV and VQTERM, that had

Table 4. Comparison of benchmark execution times (in seconds) for Serial 2012 CRAY-2S (55-ns memory, one processor) showing effect of compiler.

Code	CFT77 1.3	CFT77 2.0	Speedup
FFT	9.6	4.1	2.34
GAMTEB	4.7	4.4	1.04
SCALGAM	92.2	92.8	0
LSS	8.9	8.9	0
MATRIX	57.3	57.0	0
INTMC	18.6	18.2	1.02
HYDRO	68.8	51.1	1.35
WAVE	174.6	109.7	1.6
ESN	21.5	20.4	1.05
MCNP[a]	77.8	78.1	0

[a]Four thousand source particles.

not vectorized at all in version 1.3 and now vectorize in version 2.0. Each of these three loops did not vectorize with version 1.3 because of function references in conditional blocks. However, another loop in HYDRO that consumes a considerable portion of the execution time does not vectorize with any Cray Research compiler, although we know of other compilers that are successful in vectorizing this loop.

The WAVE code also derives an impressive gain from the use of CFT77 2.0 on the CRAY-2S. The 60% decrease in execution time results from the full vectorization of over 50 loops that had been conditionally vectorized in version 1.3. However, 16 loops in WAVE remain conditionally vectorized.

So far we have focused on CRAY-2 results obtained with the CFT77 compiler. A version of the Cray Research CFT compiler, called CFT2, is also available on the CRAY-2. A comparison of benchmark execution times for code produced using both compilers is given in Table 5. The CFT77 2.0 compiler produces much better code in all cases.

4.3. Comparison of CRAY-2S with CRAY-MP/416

In this section we examine the performance of only the CRAY-2S with that of the CRAY X-MP/416 results. We used a pre-release of CFT77 2.0 (BF185) on a CRAY X-MP/416

Table 5. Comparison of benchmark execution times (in seconds) for CRAY-2 (one processor) using CFT2 and CFT77 compilers on SN-2012.

Code	CFT2	CFT77 2.0
FFT	4.9	4.1
GAMTEB	10.0	4.4
SCALGAM	144.7	92.8
LSS	12.4	8.9
MATRIX	[a]	57.0
INTMC	20.6	18.2
HYDRO	107.0	51.1
WAVE	[a]	109.7
ESN	27.0	20.4
MCNP	[a]	78.1

[a]Did not compile.

running the CTSS operating system at Los Alamos National Laboratory (LANL). (The version of CFT77 2.0 we used on the CRAY-2S was also a pre-release, BF184.)

4.3.1. Primitive Vector Operations. First, we examine the performance of selected elementary vector operations, listed in Tables 6 and 7. These tables contain rates, in Mflops, for various elementary vector operations as a function of vector length. All operations were carried out with unit stride except for the second and third operations in both tables. All measurements were done on a dedicated system using a single processor. On the X-MP, an in-register infinite loop was also used to keep the idle processors occupied. The X-MP/416 tests used bidirectional memory.

For the nonstrided, nonscatter/gather operations in Tables 6 and 7, the differences between the two machines at vector length 1000 can generally be reconciled with the rate at which each machine is capable of producing results. For example, on the first operation, $V = V + S$, we expect comparable rates, and we observe 83 Mflops for the CRAY-2S and 100 Mflops for the X-MP. As another example, on the fourth operation, $V = V * V$, we expect the asymptotic rate on the CRAY-2S to be less than that of the X-MP by about a factor of 1.5; at vector length 1000, the observed ratio is 1.78 (51 Mflops for the CRAY-2S and 93 Mflops for the X-MP). However, the CRAY-2S compiler has to unroll all these loops (to a depth of four) to achieve this performance. At shorter vector lengths the X-MP is faster than the CRAY-2 by about a factor of 2.

Table 6. Rates (Mflops) on the CRAY-2S for selected vector operations as a function of vector length (single processor; CFT77 2.0).

Operation	10	50	100	200	1000
$a(i) = b(i) + s$	10	37	44	47	83
$a(i) = b(i) + s(i=1, n, 23)$	9	35	42	46	84
$a(i) = b(i) + s(i=1, n, 8)$	8	16	16	16	18
$a(i) = b(i)*c(i)$	8	28	33	36	51
$a(i) = b(i) + s*c(i)$	15	54	65	72	113
$a(i) = b(i)*c(i) + d(i)*e(i)$	19	60	66	76	90
$a(i) = b(j(i)) + s$	7	19	21	22	29
$a(j(i)) = b(i)*c(i)$	8	28	29	34	35

Table 7. Rates (Mflops) on the CRAY-X-MP/416 for selected vector operations as a function of vector length (single processor; CFT77 2.0; bidirectional memory).

Operation	10	50	100	200	1000
$a(i) = b(i) + s$	20	64	70	83	100
$a(i) = b(i) + s(i=1, n, 23)$	19	65	74	83	99
$a(i) = b(i) + s(i=1, n, 8)$	19	65	74	81	100
$a(i) = b(i)*c(i)$	17	58	67	75	93
$a(i) = b(i) + s*c(i)$	36	113	127	151	181
$a(i) = b(i)*c(i) + d(i)*e(i)$	43	105	122	128	148
$a(i) = b(j(i)) + s$	17	41	51	51	59
$a(j(i)) = b(i)*c(i)$	16	38	41	42	50

Comparison of the first and second operations in Table 6 shows that, as expected, the CRAY-2S suffers no performance degradation with odd strides. However, with stride 8, performance on the CRAY-2S is about one-fourth of the nonstrided rate. The minimum time for memory transfer on the CRAY-2S is slightly more than 1 CP/word. However, with stride 8 all words of data reside in the same quadrant. Therefore, the minimum transfer time, delayed by quadrant conflict only, is about 6.5 CP/word. With a stride of 8, there are no bank conflicts on the machine we used.

Scatter/gather operations, the last two rows in Tables 6 and 7, are much more efficient on the X-MP than they are on the CRAY-2, over the entire range of vector lengths. The gather operation on the CRAY-2 is subject to a special hardware delay so that references are allowed roughly once every 4 CPs.

4.3.2. Benchmark Codes. A comparison of the current CRAY-2S results with the CRAY X-MP/416 for the rest of the benchmark codes is shown in Table 8. Two sets of results are given for the X-MP: one from a pre-release of CFT77 2.0 and one from the production compiler, CFT 1.14. The first thing to notice in Table 8 (comparing columns two and three) is that on the X-MP, CFT77 2.0 now produces better code than CFT 1.14 (with no compiler options) for all but one benchmark. The only (minor) exception is MATRIX, for which CFT 1.14 with the BTREG option (shown in parenthesis in Table 8) is slightly faster than CFT77 2.0.

The X-MP has a significant performance advantage over the CRAY-2S on seven of the ten codes. Of the seven, four are highly vectorizable: HYDRO, LSS, MATRIX, and WAVE. In HYDRO, LSS, and MATRIX, the predominant loop length is about 100. The VECOPS data in Tables 6 and 7 showed that the X-MP ran loops at vector length 100 nearly twice as fast as the CRAY-2S did. In WAVE, the predominant loop length is 256. WAVE also involves many gathers for which, as shown above, the X-MP is superior.

Interestingly, in contrast with the VECOPS data, the X-MP is only about 5% faster on the FFT code, a highly vectorized code with short vector lengths on which the X-MP should be fastest.

Table 8. Comparison of benchmark execution times (in seconds) for CRAY-2S (Serial 2012) and CRAY X-MP/416 (single processor).

Code	CRAY-2 (CFT77 2.0)	X-MP/416 (CFT77 2.0)	X-MP/416 (CFT 1.14)	
FFT	4.1	3.9	4.3	
GAMTEB	4.4	5.2	7.6	
SCALGAM	92.8	76.2	88.4	
LSS	8.9	7.6	11.6	(6.7)[a]
MATRIX	57.0	49.2	54.7	(33.2)[a]
INTMC	18.2	12.1	40.2	
HYDRO	51.1	39.8	48.9	
WAVE	109.7	86.0	111.2	
ESN	20.4	15.9	18.2	
MCNP[b]	78.1	73.2	76.7	

[a]Time in parenthesis is with bidirectional memory and OPT=BTREG for the CFT 1.14 compiler.

[b]Four thousand source particles started.

An important aspect of vectorization on the CRAY-2S concerns the way in which arrays are dimensioned. Because of quadrant conflicts that can have a noticeable effect on performance, arrays with even dimensions will suffer performance degradations relative to arrays with odd dimensions. This fact is highlighted in the performance of the codes LSS and MATRIX relative to the X-MP. Both codes spend most of their time in SAXPY, and both have loop lengths of 100. Yet MATRIX runs nearly 63% faster on the X-MP than it does on the CRAY-2S, whereas LSS runs about 45% faster on the X-MP. In MATRIX, two of three critical arrays have even dimensions, while in LSS all critical arrays have odd dimensions. Thus, relative to the X-MP, one must be far more careful of program array dimensions on the CRAY-2S.

The relationship between the X-MP and the CRAY-2S on codes not overwhelmingly vector in nature is harder to explain. Of the two Monte Carlo photon transport codes, one, SCALGAM, runs about 28% faster on the X-MP, while the other, GAMTEB, runs about 18% faster on the CRAY-2S. ESN, a totally scalar code, runs about 28% faster on the X-MP. But on MCNP, the X-MP is only 7% faster than the CRAY-2. The reason for this is not clear.

4.3.3. X-MP/416 External Storage Performance. The larger central memory on the CRAY-2 is an important asset for this machine. However, the X-MP can be equipped with an external solid-state storage device (SSD) that can also offer potential for large codes. An obvious question is: if a problem can be programmed with an *out-of-core* algorithm, how does the X-MP with SSD perform relative to the same problem run *in-memory* on the CRAY-2?

The WAVE code can be so programmed. We ran a job requiring about 20 Mwords of storage on the CRAY-2 (Serial 2011, 80-ns memory). We ran the same code on an X-MP/416 runnning CTSS and equipped with a 512-Mword SSD using one channel (1250 Mbyte/s). Both machines used the CFT77 version 2.0 compiler. The X-MP/416 version transferred to the SSD in block sizes of 204,800 words. The CRAY-2 ran the job in 461 seconds, while the X-MP required 355 seconds of CPU time and 360 seconds of elapsed (wall-clock) time. Although we did not run this code on the 55-ns CRAY-2S, we can approximate what the performance will be. Using the CFT77 version 2.0 compiler, the standard WAVE benchmark runs about 12% faster on the 55-ns CRAY-2S than it does on the 80-ns CRAY-2, so the best CRAY-2S time for the 19-Mword job would be about 411 seconds. This value is still larger than the X-MP wall clock time. Note that although I/O to the SSD does not require particularly difficult coding (as might I/O to a disk) other than insuring a large block size, the CRAY-2 version requires no extra coding.

5. Multitasking Results

The four processors of the CRAY-2 can simultaneously be brought to bear on a single job through the multitasking environment. We ran our large Monte Carlo transport code, MCNP, in this environment on the Serial 2011 CRAY-2 at the Air Force Weapons Laboratory in February 1988. The compiler was CFT77 2.0, the operating system was CTSS, and the multitasking library was Multilib. We ran a problem size of 60,000 source particles. For comparison, we ran the same problem on an X-MP/416 at Los Alamos using the CFT77 2.0 compiler, the CTSS operating system, and a multitasking library that is a local system.

Table 9. Multitasking execution times (in seconds) for MCNP on the CRAY-2 (Serial 2011) and the CRAY X-MP/416.

	CRAY-2 Total	CRAY X-MP/416 Total
1 Processor	1343.3	963.6
2 Processor	697.4	486.3
3 Processor	477.2	330.9
4 Processor	351.5	253.0
Serial	1240.9	923.8

We used a parallelization method called macrotasking developed at Cray Research and adapted for CTSS on the CRAY-2 by the NMFECC. This method operates at the granularity level of the subroutine.[3] Multitasking runs on both machines were done during dedicated time. The times are given in Table 9. Note that the X-MP is about 40% faster than the CRAY-2 for one to four processors. The serial times differ by 34%, which is comparable to the differences observed for the other scalar serial codes.

Speedup is defined as

$$S = T_s/T_n, \tag{2}$$

where T_s is the serial execution time and T_n is the execution time using n processors. The speedups for MCNP are plotted in Figure 2. The CRAY-2 shows a speedup of 3.53 for four processors, while on the X-MP speedup is 3.65. This difference might be attributed to several factors, one of which is the availability of only a single semaphore per job on the CRAY-2. The X-MP has 32 semaphores available to a job. Another factor affecting speedup is the implementation of synchronization primitives. The Los Alamos system has implemented spin-wait locks while the Cray Research/NMFECC implementation is somewhat less efficient. Since Monte Carlo algorithms are considered to be ideal candidates for parallel processing, one might expect a speedup for four processors that is somewhat closer to four. One reason that we do not see this for this set of runs is that the time spent in the serial sections, such as the setting up of the problem, is constant and independent of the number of source particles. This means that as more and more processors are brought to bear on a problem of fixed size, the serial portion takes a larger percentage of the time. This is, of course, Amdahl's law [Amdahl 1967].

If we interpret Ware's model [Ware 1972] (of Amdahl's law) of vector performance as applying to multiprocessor performance, we can also define speedup as

$$S = [(1 - f) + (f/p)]^{-1}, \tag{3}$$

where f is the fraction of the code that can be executed in parallel and p is the number of processors. For MCNP, which is about 98% parallel, we get a predicted speedup of 3.77 for four processors. This is somewhat higher than our measured speedup of 3.53 on the CRAY-2S.

There are several reasons for the difference in these two speedups. One is the effect of multiprocessor synchronization overhead [Buzbee 1984]. Another is the additional time

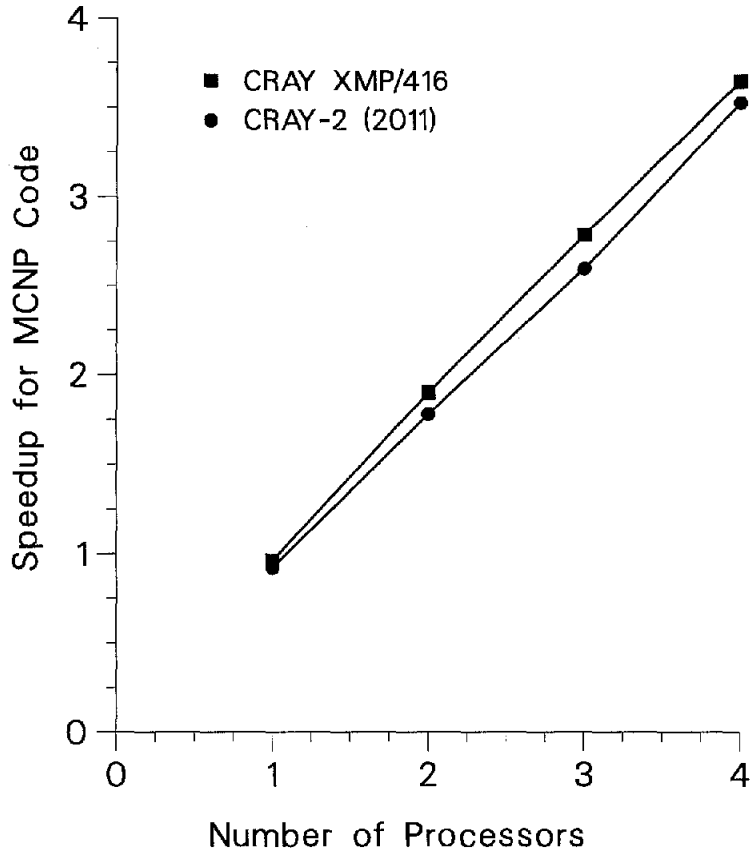

Figure 2. Speedup on a CRAY-2.

required for system overhead in the multiple processor runs. This serial version of MCNP, for example, is not stack based and so incurs no overhead associated with stack management.

6. Conclusions

The faster memory chips on recent models of the CRAY-2 provide some improvement on our benchmarks, but do not, by themselves, allow the CRAY-2S to perform better than the X-MP/416 in single-processor mode. This is because too much of the memory bottleneck on the CRAY-2 is due to factors other than chip access time.

The biggest improvements we have observed during the evolution of the CRAY-2 are derived from compiler changes, not hardware changes. In particular, HYDRO and WAVE, two benchmark codes that closely resemble production codes at the Laboratory, benefit significantly from the combination of new hardware and a new version of CFT77 on the CRAY-2S.

The X-MP has a clear performance advantage over the CRAY-2S on our codes that are highly vectorized. However, the difference between these machines is less clear on codes

that are not overwhelmingly vector. The significant factor here appears to be the longer memory latency on the CRAY-2. Although the CRAY-2 provides more central memory than the X-MP, we have shown that on one code that takes advantage of the X-MP SSD, the faster processor and high I/O rates can overcome the lack of X-MP memory.

In multitasking mode, the CRAY-2 performs about as well as the X-MP on the problem that we ran. While the overall times are not as fast as the X-MP, the speedups are comparable. The overhead observed for the problem we ran could be reduced either by running a larger problem or by using more efficient synchronization (microtasking).

Acknowledgments

This work was performed under the auspices of the United States Department of Energy. We are grateful to Dr. Larry Rapagnani of the Air Force Weapons Laboratory for a generous allocation of resources on the CRAY-2 (Serial 2011). Ann Hayes, Rebecca Koskela, Olaf Lubeck, and James Moore of Los Alamos National Laboratory helped collect some of the benchmark data. We thank Ken Lord of Cray Research for his assistance during all phases of the benchmark process.

Appendix: Description of the Codes in the Benchmark Set

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 [Swartztrauber 1984] 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. The 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.

SCALGAM: 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 [Dongarra et al. 1979] 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.

MCNP: MCNP is a general-purpose Monte Carlo code [Booth et al. 1986], heavily used at the Laboratory and elsewhere, that does neutron, photon, or coupled neutron/photon transport. It includes the ability to calculate eigenvalues for critical systems. The code treats an arbitrary three-dimensional configuration of materials in geometric cells bounded by first- and second-degree surfaces and some special fourth-degree surfaces. Point-wise cross sections are used throughout. The test problem includes a fair sample of the commonly used features of the code. The code has been parallelized for several different parallel processors. Typically 100,000 source particles are started. The code does not vectorize.

ESN: ESN is a one-dimensional, discrete ordinates, particle transport code that solves the transport equation by the discrete ordinates method [Wienke 1982]. The current algorithm implemented in ESN was developed by Wienke and Hiromoto [1985]. 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: HYDRO is a two-dimensional Lagrangian hydrodynamics code based on an algorithm by W.D. Schultz [1964]. 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 100×100 mesh for 100 time steps.

WAVE: WAVE is a two-dimensional, relativistic, electromagnetic particle-in-cell simulation code used to study various plasma phenomena [Morse and Neilson 1971]. 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.

Notes

1. This work was performed under the auspices of the U.S. Department of Energy.
2. A later version of CFT77 became available after this study was completed, but the CRAY-2 SN 2012 was no longer available to us. Because hardware on the CRAY-2 can differ significantly from machine to machine, we elected not to test the later versions of the compiler.
3. Another approach, called microtasking, operates at the granularity of the DO loop and can be much more efficient. However, since MCNP does not contain many DO loops, a moderate amount of reprogramming would be required in order to take advantage of microtasking.

References

Amdahl, G. 1967. The validity of the single processor approach to achieving large-scale computing capabilities. In *Proc., Amer. Fed. of Information Processing Societies*, Vol. 30, Washington, D.C., pp. 483–485.

Booth, T.E., et al. 1986. MCNP—A general Monte Carlo code for neutron and photon transport, version 3a. Los Alamos Nat. Laboratory Unclassified Release LA-7396-M Rev 2.

Brickner, R.G., Wasserman, H.J., Hayes, A.H., and Moore, J.W. 1986. Benchmarking the IBM 3090 with vector facility. Los Alamos Nat. Laboratory Unclassified Release LAUR-86-3300.

Buzbee, B.L. 1984. The efficiency of parallel processing. *Computer Design.*

Dongarra, J.J., Moler, C.B., Bunch, J.R., and Stewart, G.W. 1979. LINPACK user's guide. *Society for Industrial and Applied Mathematics.*

Griffin, J.H., and Simmons, M.L. 1984. Los Alamos National Laboratory computer benchmarking 1983. Los Alamos Nat. Laboratory Unclassified Rept. LA-10151-MS.

Kampe, F.C., and Nguyen, T.M. 1986. Performance comparison of the CRAY-2 and CRAY X-MP on a class of seismic data processing algorithms. *Parrallel Computing*, 559–570.

Lubeck, O., Moore, J., and Mendez, R. 1985. A benchmark comparison of three supercomputers: Fujitsu VP-200, Hitachi S810/20, and CRAY X-MP/2. *IEEE Comp.*, 18: 10–29.

Lubeck, O., Moore, J., and Mendez, R. 1987. The performance of the NEC SX/2 and CRAY X-MP supercomputers. Los Alamos Nat. Laboratory Unclassified Release LAUR-87-227.

Morse, R.L., and Neilson, C.W. 1971. Numerical simulation of the Weibel instability in one and two dimensions. *The Physics of Fluids*, 14, 4.

National Research Council. 1986. *An Agenda for Improved Evaluation of Supercomputer Performance.* National Academy Press, Washington D.C.

Numrich, R. 1985. CRAY-2 Memory. Private communication.

Schultz, W.D. 1964. *Methods of Computational Physics*, 3.

Simmons, M.L., and Lubeck, O. 1986. Benchmark of the Convex C-1 mini supercomputer. Los Alamos Nat. Laboratory Unclassified Release LAUR-86-2890.

Simmons, M.L., and Wasserman, H.J. 1987. Los Alamos National Laboratory computer benchmarking 1986. Los Alamos Nat. Laboratory Rept. LA-10898-MS.

Swartztrauber, P.N. 1984. FFT algorithm for vector computers. *Parallel Computing*, 1, 45.

Ware, W.H. 1972. The Ultimate Computer. *IEEE Spectrum*, 9, 3:84–91.

Wasserman, H.J. 1988. Los Alamos National Laboratory computer benchmarking 1988. Los Alamos Nat. Laboratory Rept. LA-11465-MS.

Wasserman, H.J., Simmons, M.L., and Hayes, A.H. 1987. A benchmark of the SCS-40 computer: A mini supercomputer compatible with the CRAY X-MP/24. Los Alamos Nat. Laboratory Unclassified Release LAUR-87-659.

Wienke, B. 1982. ESN: One dimensional S_n transport module for electrons. *J. Quant. Spect. Rad. Trans.*, 28:311.

Wienke, B., and Hiromoto, R. 1985. Parallel S_n iteration schemes. *Proc., Internat. Meeting on Advances in Nuclear Engineering Computational Methods.*