*Journal of Statistical Physics, Vol. 43, Nos. 5/6, 1986*

# Advanced Large-Scale and High-Speed Multiprocessor System for Scientific Applications

# CRAY X-MP-4 Series

Steve S. Chen
Cray Research, Inc.
Chippewa Falls, Wis.

## Overview of CRAY X-MP-4 System

1. General-purpose multiprocessor system for multitasking applications.
    - Run independent tasks of different jobs on multiple processors. Program compatibility (with CRAY-1) is maintained for all tasks.
    - Run related tasks of single job on multiple processors.
        - Loosely-coupled tasks communicating through shared memory.
        - Tightly-coupled tasks communicating through shared registers.
    - Small overhead of task initiation for multitasking, O(1μs) to O(1ms), depending on granularity of the tasks and software implementation techniques.
    - Flexible architecture concept for processor clustering.
        - All processors are identical and symmetric in their programming functions, i.e., there is no permanent master/slave relation existing between all processors.
        - A cluster of k processors ($0 \leq k \leq p$) can be assigned to perform a single task, where $p = 4$ is the number of physical processors in the system.
        - Up to $p + 1$ processor clusters can be assigned by the operating system.
        - Each cluster contains a unique set of shared data and synchronization registers for the inter-communication of all processors in a cluster.
        - Each processor in a cluster can run in either monitor or user mode controlled by the operating system.
        - Each processor in a cluster can asynchronously perform either scalar or vector operations dictated by user programs.
        - Any processor running in monitor mode can interrupt any other processor and cause it to switch from user mode to monitor mode.
        - Built-in detection of system deadlock within the cluster.
    - Faster exchange for switching machine state between tasks.
    - Hardware supports separation of memory segments for each user's data and program to facilitate the concurrent programming.

2. General-purpose multiprocessor system for compute-bound and I/O-bound applications.

   - All processors share a central bipolar memory (16MW), organized in m=64 interleaved memory banks (four times that of CRAY-1). All banks can be accessed independently and in parallel during each machine clock period. Each processor has four parallel memory ports (four times that of CRAY-1) connected to this central memory, two for vector fetches, one for vector store, and one for independent I/O operations. The multiport memory has built-in conflict resolution hardware to minimize the delay and maintain the integrity of all memory references to the same bank in the same time, from all processor's ports. The interleaved and efficient multiport memory design, coupled with shorter memory cycle time, provide a high-performance and balanced memory organization with sufficient bandwidth (sixteen times that of CRAY-1) to support simultaneous high-speed CPU and I/O operations.

   - New, large, CPU-driven Solid-state Storage Device (SSD) is designed as an integral part of the mainframe with very high block transfer rate. This can be used as a fast-access device for user large pre-staged or intermediate files generated and manipulated repetitively by user programs, or used by the system for job "swapping" space and temporary storage of system programs. The SSD design with its large size (128MW), very fast data transfer speed (maximum rate exceeds 2000 MB/sec, 500 times faster than current disk), and much shorter access time (less than .5 ms, 100 times shorter than current disk), coupled with the high-performance multiprocessor design, will enable the user to explore new application algorithms for solving bigger and more sophisticated problems in science and engineering which they could not attempt before.

   - The I/O Subsystem, which is an integral part of the CRAY X-MP-4 System, also contributes to the system's overall performance. The I/O Subsystem (compatible with CRAY-1/S and 1/M) offers parallel streaming and striping of disk drives, I/O buffering (8MW max.) for disk-resident and Buffer Memory-resident datasets, high-performance on-line tape handling, and common device for front-end system communication, networking, or specialized data acquisition. The IOP design enables faster and more efficient asynchronous I/O operations for data access and deposition of initial and final outputs through high-speed channels (each channel has maximum rate 100 MB/sec, and more than 10 times faster than current disk), while relieving the CPUs to perform computation-intensive operations.

   - A new disk storage device is available and attached to the I/O Subsystem, which has 2 1/2 the performance with respect to the transfer rate and access time (maximum rate 10 MB/sec, and average access time 20 ms), and twice the storage capacity (1.2 GB) of the current disk.

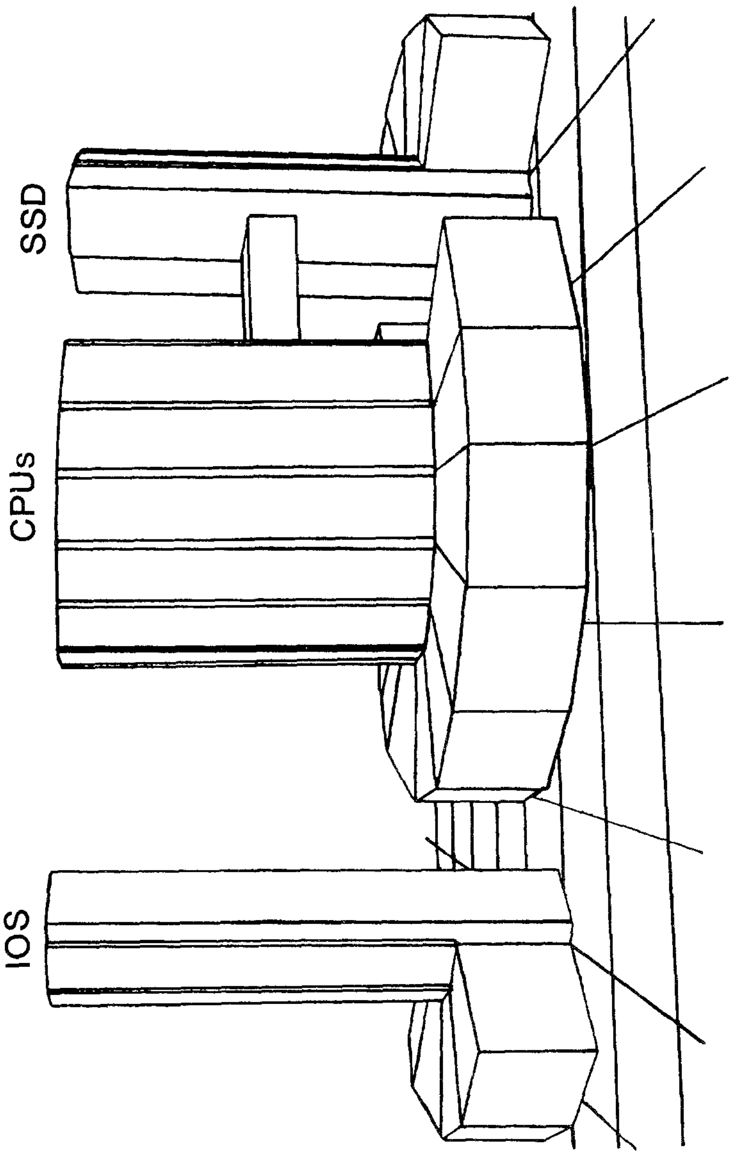
3. General-purpose multiprocessor system for scalar and vector applications.

   - All processors are controlled synchronously by a central clock with improved cycle time = 9.5 ns (vs. 12.5 ns of CRAY-1).

   - The scalar performance of each processor is improved through faster machine clock, shorter memory access time, and larger instruction buffers (twice that of CRAY-1).

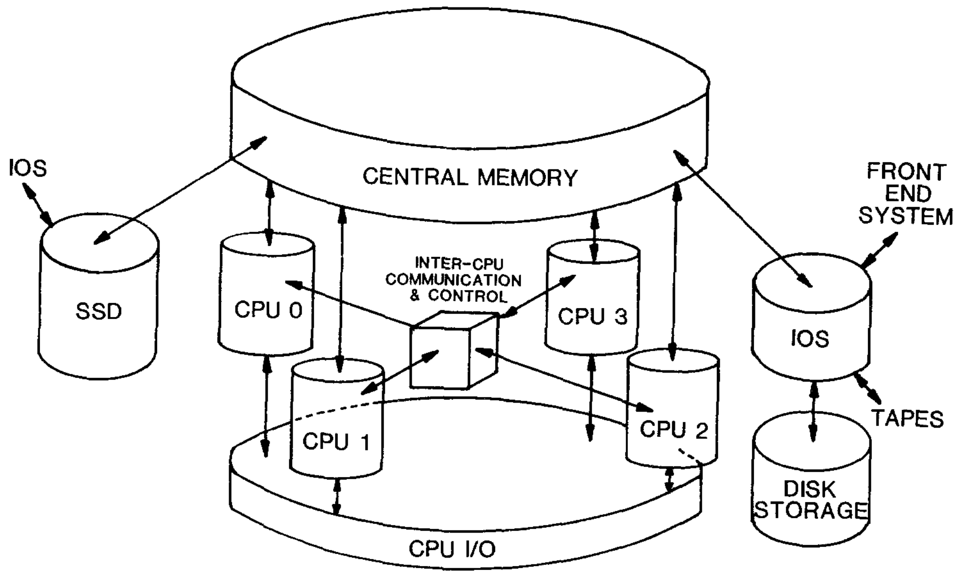
   - The vector performance of each processor is improved through faster machine clock, parallel memory ports, and hardware automatic "flexible chaining" features. These new features allow simultaneous memory fetches, arithmetic, and memory store operations in a series of related vector operations (this contrasts to the "fixed chaining" and uni-directional vector fetch/store in CRAY-1). As a result, the processor design provides higher speed and more balanced vector processing capabilities for both long and short vectors, characterized by heavy register-to-register or heavy memory-to-memory vector operations. In addition, hardware support of gather/scatter operations (chainable from other vector memory fetches and stores), and compressed index generation are also provided to facilitate and speedup the execution of various conditional vector operations, realized from ordinary user programs.

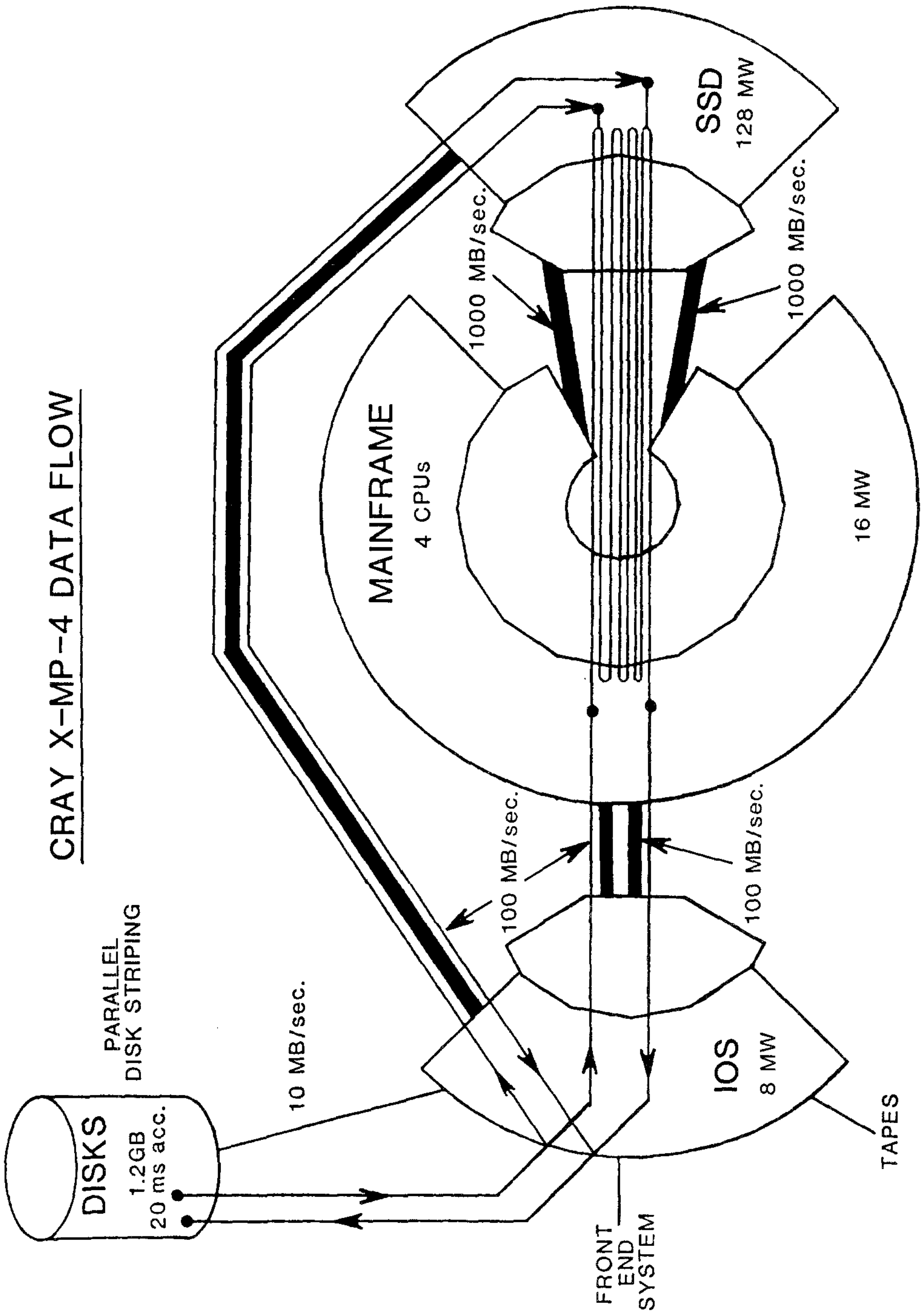
   - The processor design is well-balanced for processing both scalar and vector codes. The overall effective performance of each processor in execution of typical user programs with interspersing scalar and vector codes (usually short vectors) is ensured through fast data flow between scalar and vector functional units, short memory access time for vector and scalar references, as well as small start-up time for scalar and vector operations. As a result of this unique design characteristic, the machine can perform very well in real programming environments using standard compiler, without resorting to enormous amounts of hand-coding or even restructuring of the original application algorithms. Certainly, as the code is more vectorized, and vector length is becoming longer, an even better performance can be achieved.



CRAY X-MP-4 MULTIPROCESSOR

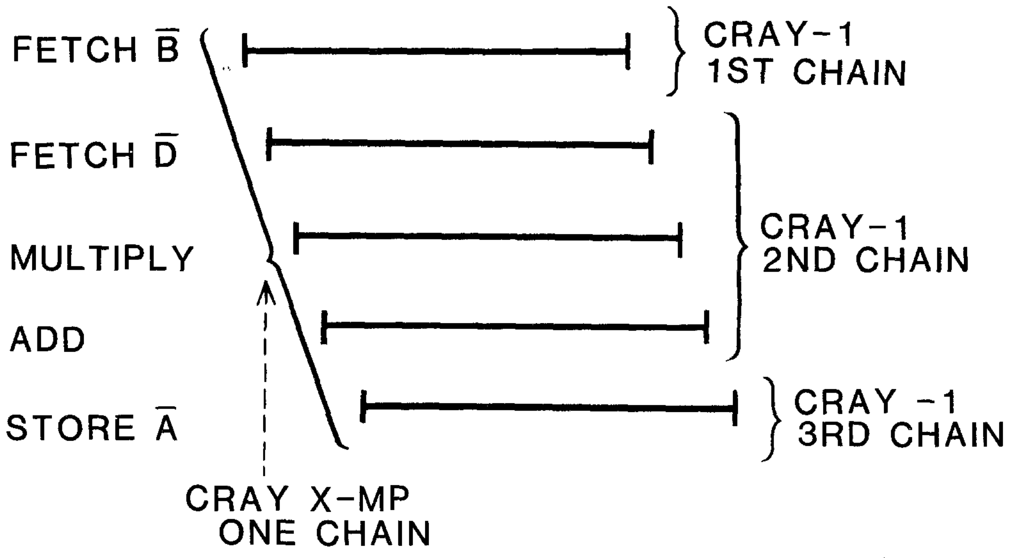
## CRAY X-MP-4 OVERALL SYSTEM ORGANIZATION




CRAY X-MP-4 DATA FLOW
DISKS
1.2GB
20 ms acc.
PARALLEL
DISK STRIPING
10 MB/sec.
100 MB/sec.
100 MB/sec.
100 MB/sec.
IOS
8 MW
TAPES
FRONT
END
SYSTEM
MAINFRAME
4 CPUs
16 MW
1000 MB/sec.
1000 MB/sec.
SSD
128 MW

## VECTOR COMPUTATIONS

$\bar{A} = \bar{B} + s * \bar{D}$



NOTES:

1. Using 3 memory ports per processor

2. Hardware automatically "chains" through all five vector operations such that one result per clock period can be delivered

3. Support conditional vector operations: Gather/scatter, Compressed Index

## VECTOR LOOP FAMILIES
## BENCHMARK TIMINGS ON X-MP-4

1-CPU

| | SHORT VECTOR (VL = 8) | MEDIUM VECTOR (VL = 128) | LONG VECTOR (VL = 1024) |
|---|---|---|---|
| A=B | 1.1 | 1.8 | 2.1 |
| A=B+C | 1.2 | 2.2 | 2.7 |
| A=B*C | 1.5 | 2.6 | 3.3 |
| A=B/C | 1.5 | 1.9 | 2.0 |
| A=B+C+D | 1.5 | 2.7 | 3.2 |
| A=B+C*D | 1.4 | 2.9 | 3.6 |
| A=B+s*D | 1.3 | 3.0 | 4.0 |
| A=B+C+D+E | 1.3 | 2.3 | 2.7 |
| A=B+C+D*E | 1.6 | 2.5 | 2.9 |
| A=B*C+D*E | 1.3 | 2.5 | 3.1 |
| A=B+C*D+E*F | 1.5 | 2.1 | 2.2 |
| | 1.5 | 2.5 | 3.0 |

(Unit based on compiler generated code running on 1/S)

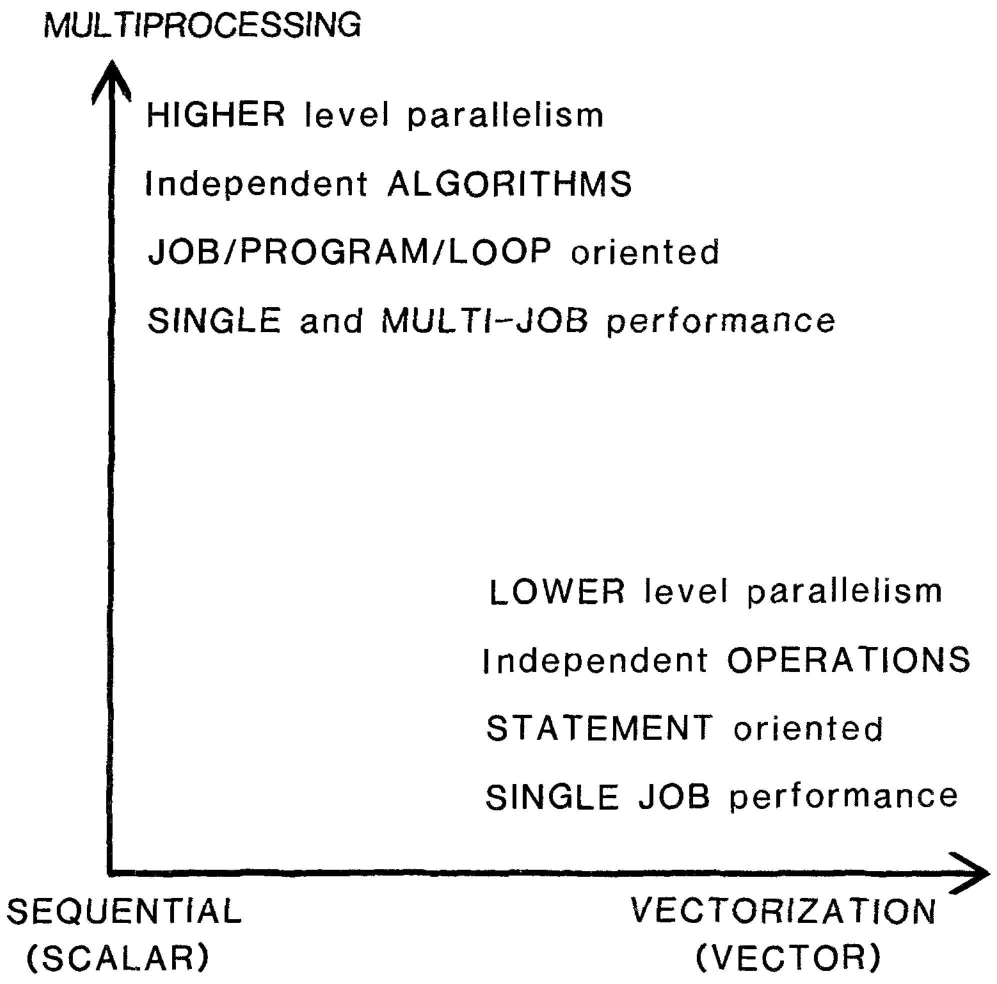
## X-MP-4 GATHER/SCATTER

```
CDIR$ IVDEP
      DO 10 J = 1,N
         A(I(J)) = A(I(J)) + S * B(J)
10    CONTINUE
```

I is a vector of distinct subscripts

| N | (1-CPU) X-MP-4 Compiled Code vs. CRAY-1 Compiled Code Speedup | (1-CPU) X-MP-4 Compiled Code vs. CRAY-1 Optimized Library Code Speedup |
|---|---|---|
| 8 | 6.9 | 4.4 |
| 128 | 17.1 | 5.5 |
| 1024 | 16.2 | 4.6 |

## TWO DIMENSIONS OF PARALLELISM



## MULTITASKING ON THE CRAY X-MP-4 MULTIPROCESSOR

OFFERS:

- The exploitation of another dimension of parallelism beyond vector processing

- A natural way for scientists and engineers to view their problems

- An opportunity for numerical analysts to explore new and faster parallel algorithms

- A convenient way for programmers to express concurrency in their programs

- Improved performance at several levels

## MULTITASKING VS. VECTORIZATION FOR THE CRAY X-MP-4

-Vectorization offers a speedup of up to 10-20 over scalar processing, depending on actual code and vector length

- Multitasking offers an additional speedup of $Sp \leq 4$ , depending on task size and relative multitasking overhead

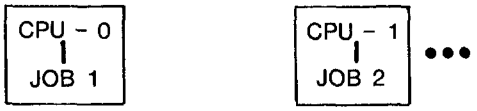
- Total speedup over scalar processing
  = Speedup (Multitasking) $*$ Speedup (Vectorization)
  e.g. = $Sp * (10-20) = 34-72$ : Assuming $Sp = 3.4-3.6$

- A general guideline: first, partition tasks at the highest possible level to apply multitasking, and then vectorize each task as much as possible

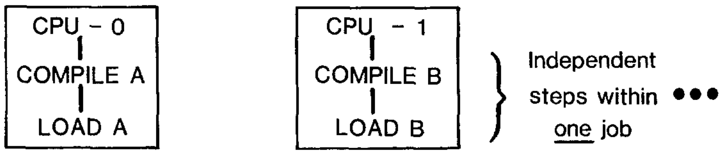
## MULTITASKING – EXPLOITING PARALLELISM AT SEVERAL LEVELS

(Conceptual Examples)

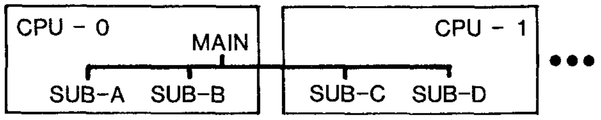
1. Multitasking at the job level (1)



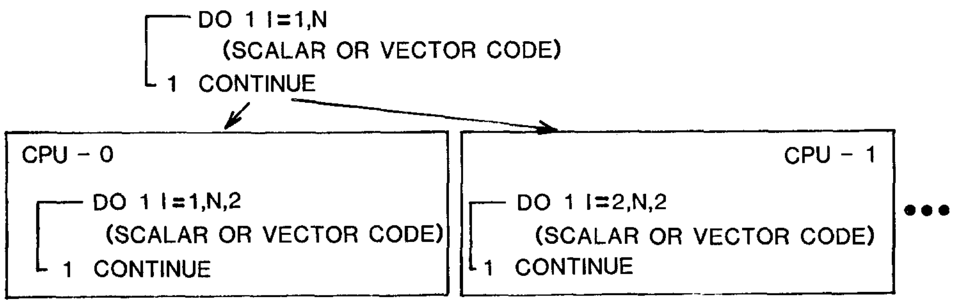
2. Multitasking at the job-step level (2)



3. Multitasking at the program level (3)



4. Multitasking at the loop level (4)



```
DO 1 I=1,N
  (SCALAR OR VECTOR CODE)
1 CONTINUE
```

CPU – 0

```
DO 1 I=1,N,2
  (SCALAR OR VECTOR CODE)
1 CONTINUE
```

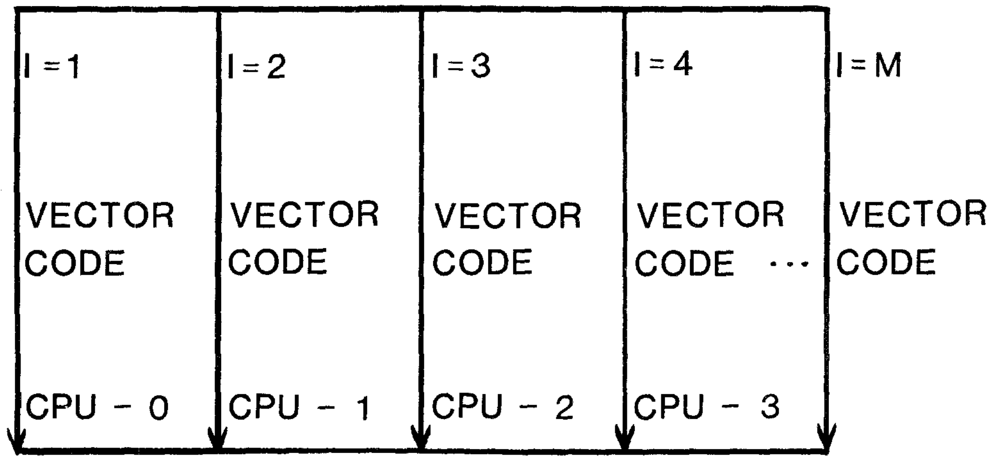
CPU – 1

```
DO 1 I=2,N,2
  (SCALAR OR VECTOR CODE)
1 CONTINUE
```

NOTES: 1. SW support available now

3., 4. SW support for user-directed multitasking available now

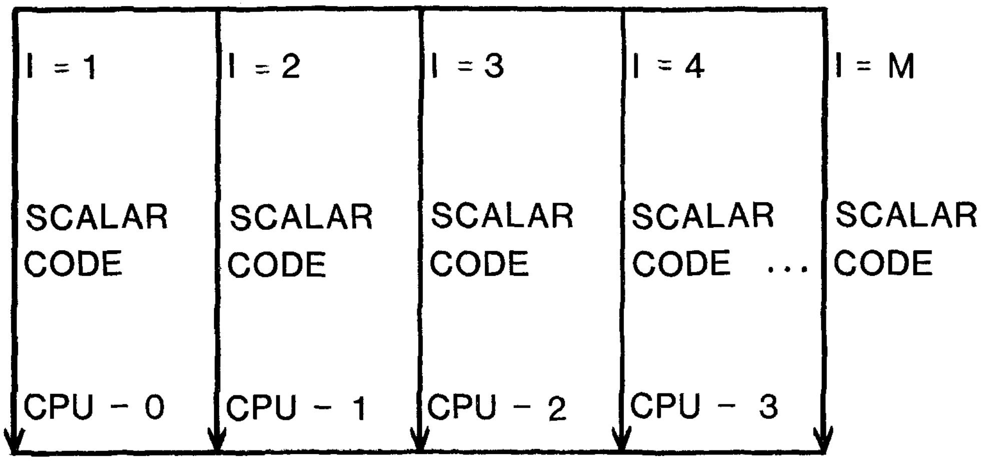
## MULTITASKING OF VECTOR CODE



Example:

```
        DO 2 I = 1,M
             :
           DO 1 J = 1,N                    } VECTOR
    1        A(I,J) = B(I,J) + C(I,J)      } CODE
             :
    2   CONTINUE
```

MULTITASKING OF SCALAR CODE



Example:

```
      DO 2 I = 1,M
        :
      DO 1 J = 1,N                      } SCALAR
1     A(I,J) = A(I,J-1)*A(I,J)          } CODE
        :
2     CONTINUE
```

## X-MP-4 MULTITASKING PERFORMANCE

- Multitasking is running on the X-MP.

- Multitasking has been demonstrated with various application codes.

- For four processors the speedup of 3.5 - 3.8 over one processor are observed.

- Parallelism is at a high level - program modification is minimal.

- Multitasking overhead with the X-MP hardware/software is negligible.

## SPECTRAL

- This is a benchmark code for short term weather forecasting
- Parallelism occurs on the latitude level, i.e., the outermost loop inside each time step
- Multitasking accounts for 98% of the execution time of a model experiment involving a global grid structure with 160 latitude by 192 longitude points, and 200 time steps

| CPUs | Execution time | Actual speedup | Theoretical speedup |
|---|---|---|---|
| 1 | 333.7 sec. | 1.00 | 1.00 |
| 2 | 174.4 sec. | 1.91 | 1.96 |
| 3 | 122.5 sec. | 2.72 | 2.88 |
| 4 | 94.0 sec. | 3.55 | 3.77 |

## PICF

- A particle-in-cell simulation program for electro/magneto static interaction between collisionless beams of plasma

- Parallelism occurs in the independent tracking of particles, and the evaluation of total charge distribution

- Multitasking accounts for 97% of the execution time of a model experiment involving 37,000 particles, and 100 time steps

| CPUs | Execution time | Actual speedup | Theoretical speedup |
|---|---|---|---|
| 1 | 72.3 sec. | 1.00 | 1.00 |
| 2 | 37.9 sec. | 1.91 | 1.94 |
| 3 | 26.6 sec. | 2.72 | 2.83 |
| 4 | 20.7 sec. | 3.48 | 3.67 |

## GAMTEB

- A Monte Carlo code which transports gamma rays in a carbon cylinder

- Parallelism occurs in the independent tracking of gamma rays

- The program uses a new technique which allows reproducibility of results for codes whose execution flow is determined by a random number generator, irrespective of the number of tasks or processors

- Multitasking accounts for 99% of the execution time of a model experiment involving 1 million original rays

| CPUs | Execution time | Actual speedup | Theoretical speedup |
|---|---|---|---|
| 1 | 202. sec. | 1.00 | 1.00 |
| 2 | 103. sec. | 1.96 | 1.98 |
| 3 | 69.5 sec. | 2.91 | 2.94 |
| 4 | 53.8 sec. | 3.75 | 3.88 |

## MG3D

- A seismic 3-D migration code to construct underground reflector structure.
- Parallelism occurs in the decoupled frequency domain at each depth level, after Fourier Transform over time.
- Multitasking accounts for 98.7% of the execution time of a model experiment involving 200 x 200 traces, with 1024 samples for each trace, and 1000 depth level.
    - Total computation = 1.5 x $10^{12}$ FLOPS
    - Total I/O = 40 x $10^{9}$ Words
- Without using SSD, it takes <u>24 hours</u> on a single processor X-MP-4 with DD-29 disks.
- With 3.67 MWs of central memory space and 40 MWs of SSD space.

| CPUs | Elapsed Time | Actual Speedup | Theoretical Speedup |
|---|---|---|---|
| 1 | 3.49 hr. | 1.00 | 1.00 |
| 2 | 1.85 hr. | 1.89 | 1.97 |
| 4 | 1.01 hr. | 3.45 | 3.85 |

## X-MP-4 THROUGHPUT TEST

- A vectorized test job requires 32.6 CPU seconds.

- Twenty copies of the test job in memory make up a job mix workload of 20 $*$ 32.6 = 652 CPU seconds.

- Wall clock time is measured from the time the first job begins execution until the last job completes execution.

| CPUs | Wall clock time | Actual speedup | Theoretical speedup with assumed 1% HW/SW overhead |
|---|---|---|---|
| 1 | 652. sec. | 1.00 | -- |
| 2 | 328. sec. | 1.99 | 1.98 |
| 4 | 171. sec. | 3.81 | 3.88 |

## CRAY X-MP-4 OVERALL PERFORMANCE

1-CPU RATE[1] (210 MFLOPS) (PEAK 315 MFLOPS)

| | |
|---|---|
| MINIMUM: | 1.25 |
| TYPICAL: | 1.5 - 2.5 |
| MAXIMUM: | 4 |

4-CPU RATE[2] (840 MFLOPS) (PEAK 1260 MFLOPS)

| | |
|---|---|
| MINIMUM: | 5 |
| TYPICAL: | 6 - 10 |
| MAXIMUM: | 16 |

I/O RATE[3]

| | ACCESS TIME | TRANSFER RATE |
|---|---|---|
| DISK | 1 | 1 |
| SSD | .01 | 500 |

NOTES:

1. Unit based on compiler generated code running on 1/S. TYPICAL refers to small-to-medium size vectors encountered in typical programs

2. Assuming four CPUs are dedicated to multitasking of a single large job

3. Unit based on measured time per sector on current disk

The author would like to thank Chris Hsiung, John Larson, and Philippe de Forcrand for their significant contribution in applying the multitasking concept to various practical applications and performance analysis.