A Study of the System Memory Bandwidth on a SV1 Using an Automotive Production Workload

Alex Akkerman
Ford Motor Company

Chuck Schwab
Ford Motor Company

Joe Kaminski
Cray Inc.

Dave Strenski
Cray Inc.

Abstract

It is a common perception that the memory bandwidth on the SV1 is a bottleneck that limits the number of processors the system memory can support. For example, according to the STREAM SUM benchmark, the SV1 memory bandwidth saturates at about 20 processors. We questioned whether this saturation point was true for a production system running real automotive applications. These applications might not require the full memory bandwidth allowing us to effectively utilize more than 20 processors. A small scale experiment was performed comparing the system's performance between an SV1 with 18 and 24 processors. Since results from this experiment proved favorable, we expanded it to 32 processors. The design and results of these experiments are presented in this paper.

Introduction and Background

The "Benchmarker's Guide for the CRAY SV1 System", published in July 2000 by the Cray benchmarking group (pages 5-8) explains the peak and measured bandwidths for the CRAY SV1. In that document the Cray benchmark group goes through all the details of the processor to cache and processor to memory bandwidth rates. The document goes on to show a table that measures the actual memory bandwidth between the processors and main memory as a function of the number of processors in the system and the number of processors per module using the Stream SUM benchmark (Table 1 and Figure 1). The data shows that for the Stream SUM benchmark, which is very memory intensive, there is a drop in the memory bandwidth when all 4 processors on the module are moving data at the same time. The data also shows that the SV1 saturates the memory bandwidth at about 20 processors. The first SV1 at Ford was configured with only 16 processors (4 modules with 4 processors per module) based on the assumption that additional processors would be memory bandwidth limited. After running the system for more than a

Table 1: Processor to Memory rate as measured by the Stream SUM benchmark (Gbyte/sec)

Processors	Modules	Mem 512	Mem 128
1	1	2.52	2.51
2	1	4.71	4.68
2	2	5.00	4.97
4	1	5.08	5.07
4	2	9.25	9.10
4	4	9.89	9.76
8	2	9.95	9.74
8	4	16.76	15.00
8	8	18.89	18.26
12	4	14.22	13.51
12	6	21.72	19.58
12	8	22.93	20.62
16	4	17.69	16.30
16	8	25.04	21.66
20	8	24.85	21.65
24	8	24.98	21.84
28	8	25.37	21.83
32	8	25.44	21.93

year, the question was raised as to whether the capacity of the system could be increased to 18 processors (6 modules with 3 processors per module) or 24 processors (8 modules with 3 processors per module). A small set of automotive applications (MSC Nastran and Mecalog's Radioss) was gathered and a small scale test was performed by running the application set with 18, 20 and 24 processors and comparing the elapsed times.

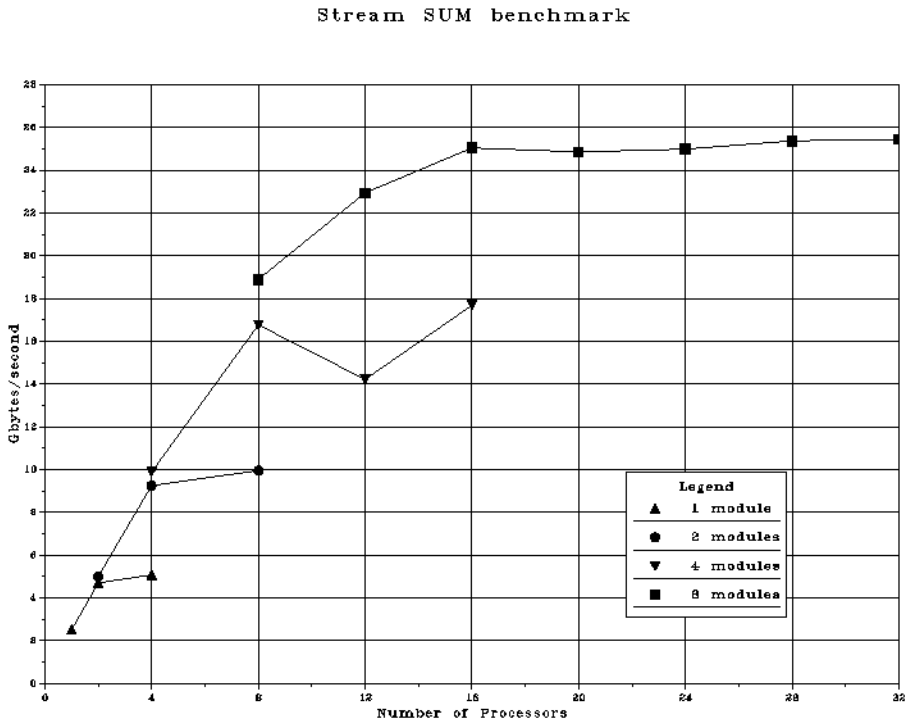

Figure 1 - Stream SUM benchmark

Since the results from that small experiment showed that there was only a small increase in the elapsed times for the jobs when going from 16 to 18, 20 or 24 processors, the system's capacity was increased to 24 processors. After running the system with 24 processors without any problems, we decided to run a similar experiment to verify the last results and expand the test to 32 processors.

32 Processor SV1 Memory Bandwidth Experimental Procedure

The purpose of this experiment was to see if a Cray SV1 had enough memory bandwidth to support 32 processors worth of work when running a typical automotive production workload of a mixture of MSC Nastran and Mecalog's Radioss. This experiment looked at running those applications on a 8 module SV1 using 1, 2, 3, and 4 processors per module. We also compared running with 4 processors per module with 2, 4, 6, and 8 modules in the system. The tests were performed by the Cray on-site team at Ford. Using the /etc/cpu command along with a map of where each processor was on each module, we where able to dynamically turn processors off and on between tests. All tests were run as root on a system running Unicos 10.0.0.8

The tests involved a combination of MSC Nastran jobs and Mecalog Radioss jobs. The jobs are typical automotive NVH and crash analysis. The Nastran jobs where always run with a single processor and the Radioss jobs were always run with 4 processors, with the exception of a few Radioss jobs that were run with 2 processors to get the total processor demand correct for the test. The 2 processor Radioss timing results were thrown away.

The the purpose of this paper, we will define “fully loaded” as a system that has the number of jobs running on the system equal to the number of processor in the system. Conversely, an overcommitted system is one in which the number of jobs running is greater then the number of processor in the system. Two sets of tests were performed; once in a fully loaded system and once in a overcommitted system. The overcommitted tests had the system over loaded by 25%. This is to reflect a typical production environment.

A typical job mix for a 24 processor test in the fully loaded case would be 4 single processor Nastran jobs and 5 jobs running Radioss, each asking for 4 processors. The total processor demand would be 24 processors, equalling the number of processors in the system. In the 25% overcommitted case, there were 4 single processor Nastran jobs and 6 jobs running Radioss with each asking for 4 processors. One additional Radioss job asking for 2 processors, is added to the test to fill up the demand to 30 processors, 25% greater then the number of processors in the system.

To facilitate the running of the different job mixes, the system always had 8 fully populated modules for a total of 32 processors. Before each run, all the processors were enabled, then the processors that need to be turned off were downed, and the configuration was then confirmed with “/etc/cpu -i”. Since the /etc/cpu command can only be run by root, the entire test suite was run as root with all the data, input and output, being written to /tmp. Here is a sample of the test script:

```
/etc/cpu -n 0-31 -u
/etc/cpu -n 2,3,6,7,10,11,14,15,18,19,22,23,26,27,30,31 -d
/etc/cpu -i
  ( cd rad_j1 ; timex ${RAD} < TESTD01 ; cd .. ) >& C16L16T1R1.log &
  ( cd rad_j2 ; timex ${RAD} < TESTD01 ; cd .. ) >& C16L16T1R2.log &
  ( cd rad_j3 ; timex ${RAD} < TESTD01 ; cd .. ) >& C16L16T1R3.log &
  ( cd nas_j1 ; timex ${NAS}           ; cd .. ) >& C16L16T1N1.log &
  ( cd nas_j2 ; timex ${NAS}           ; cd .. ) >& C16L16T1N2.log &
  ( cd nas_j3 ; timex ${NAS}           ; cd .. ) >& C16L16T1N3.log &
  ( cd nas_j4 ; timex ${NAS}           ; cd .. ) >& C16L16T1N4.log &
  wait
  ( cd rad_j1 ; timex ${RAD} < TESTD01 ; cd .. ) >& C16L20T1R1.log &
  ( cd rad_j2 ; timex ${RAD} < TESTD01 ; cd .. ) >& C16L20T1R2.log &
  ( cd rad_j3 ; timex ${RAD} < TESTD01 ; cd .. ) >& C16L20T1R3.log &
  ( cd rad_j4 ; timex ${RAD} < TESTD01 ; cd .. ) >& C16L20T1R4.log &
  ( cd nas_j1 ; timex ${NAS}           ; cd .. ) >& C16L20T1N1.log &
  ( cd nas_j2 ; timex ${NAS}           ; cd .. ) >& C16L20T1N2.log &
  ( cd nas_j3 ; timex ${NAS}           ; cd .. ) >& C16L20T1N3.log &
  ( cd nas_j4 ; timex ${NAS}           ; cd .. ) >& C16L20T1N4.log &
  wait
```

The Nastran and Radioss jobs were designed such that they had approximately the same run time length. This was done so that the system would have a near uniform load during the test. As a result of running multiple jobs, the elapsed times between them were compared against each other and were found to have very little variation, so the job elapsed times were averaged and presented in the tables and graphs as a single number.

Results

The results showed that even though the elapsed times for the jobs increase as more processors are turned on, either by turning on more processors per module or by adding more modules to the system, and the workload proportionally increased, there appears to be no dramatic slow down for a fully populated SV1 with 32 processors when running this application workload.

In the fully loaded case the elapsed times degrade (get longer) with a near linear function when more processors are turned on within each modules. This was unexpected since the Stream SUM benchmark showed a saturation point around 20 processors, so the expected elapsed times should have grown non-linearly beyond that point. Even more surprisingly was that in the overcommitted case the degradation decreased. See Figures 2 and 3.

The other interesting result from this test is that there doesn't appear to be any bottle neck getting off the module since the elapsed times for the applications are similar whether we add more processors per module or more modules to the system. See Figures 4 and 5.

Another observation showed variations in the Radioss times when running with just a few processors in an overcommitted system. The times are excessively long and it's suggested that there were just too few processors for running the jobs and the operating system overhead and the scheduler interrupted that job too much and cause the excessive elapsed time. We did not look into these results to any great length because the focus of interest was on the other end of the graph where we have far more the just 8 processors in the system.

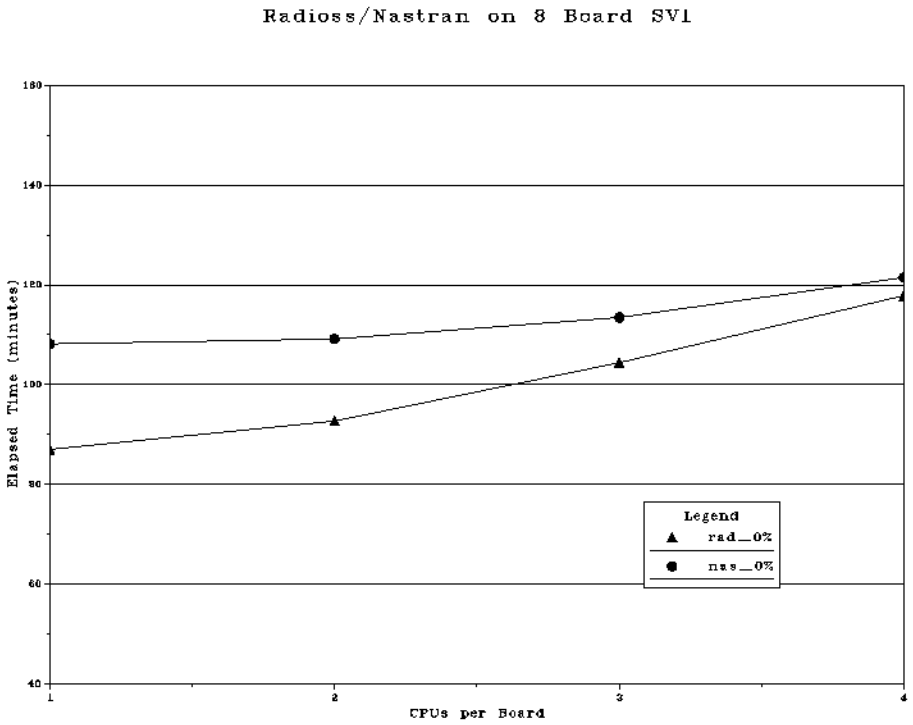

Figure 2 - Elapsed time as more processors are turned on per module (fully loaded)

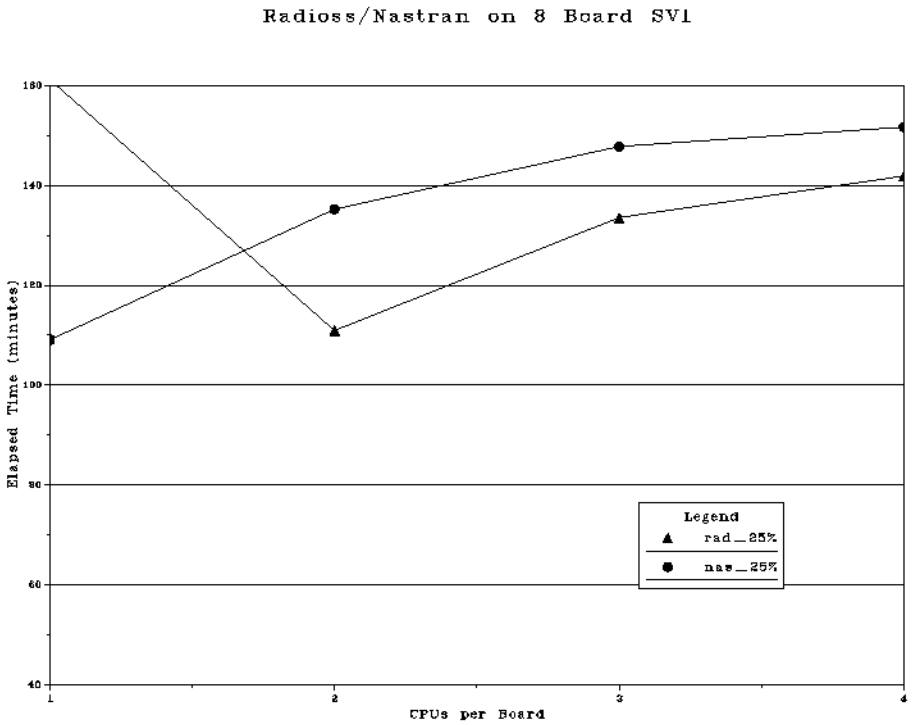

Figure 3 - Elapsed time as more processors are turned on per module (25% overcommitted)

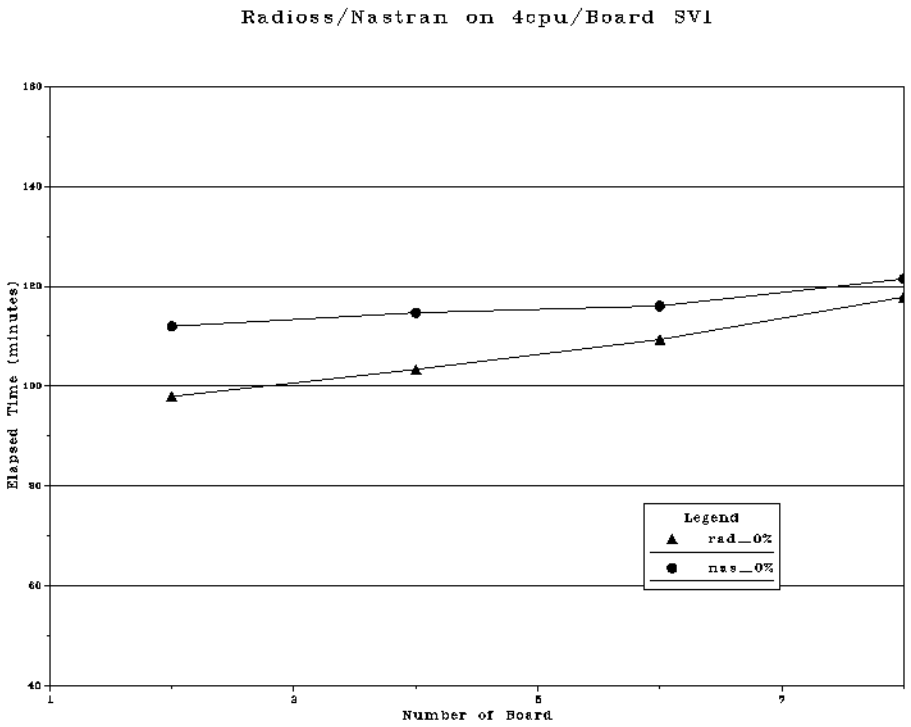

Figure 4 - Elapsed time as more modules are turned on (fully loaded)

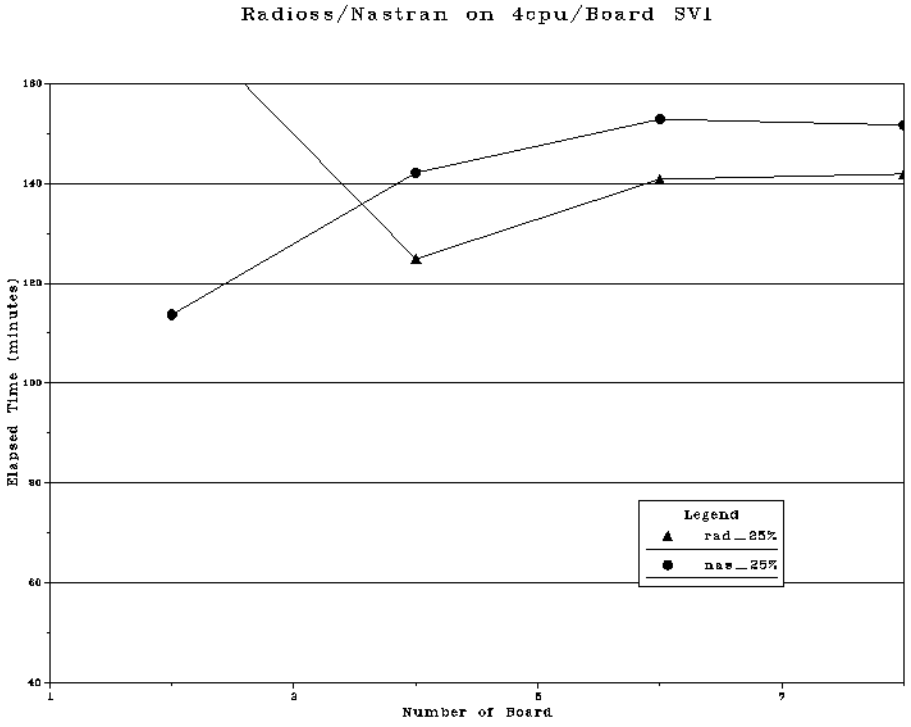

Figure 5 - Elapsed time as more modules are turned on (25% overcommitted)

Explanation of Results

Before explaining the results, it is useful to understand the characteristics of Nastran and Radioss. The Radioss code scales fairly well but gets about 3x speedup on 4 processors. So when multiple copied of Radioss are running in parallel there can be quite a bit of sharing between the processors. Therefor a system that is overcommitted with parallel Radioss jobs might really only be fully load. For example a system with 18 processors running 6 copies of Radioss each asking for 4 processors, is really only effectively using 3 processors per jobs and the system is only fully loaded. Since the Nastran jobs are run with a single processor there are no scaling effect to consider, but Nastran does a lot of I/O and there can be considerable I/O wait time where the processors can do other work while waiting for the I/O to complete.

The best explanation for the linear results in the fully loaded test cases is that the application mix of Radioss and Nastran was limit by something other then memory bandwidth of the system. The 4 processors Radioss jobs got between 400 and 600 MFLOPS with the best job hitting 641 MFLOPS. The single processor Nastran jobs got between 250 to 375 MFLOPS.

One possible explanation as to why the overcommitted results don't degrade as much is because they are already running slower then the fully loaded results and the additional processors and work allow the scheduler to divide the work up better. In this case by moving from 24 to 32 processors, the system has added 25% more capacity and yet the elapsed times for the jobs degraded by only about 6%.

Conclusions and Future Experiments

From these experiments it appears worth while to load the SV1 systems up with 32 processor. We would like to preform this experiment again with SV1ex processors and memory. The SV1e is a faster processors and should make the over all elapsed times faster but the slope should increase because more data will be trying to get to memory. The SV1ex has a faster (wider?) memory and would make not only the elapsed time faster, but should also flatten the slope of the curve due to the faster memory bandwidth.

Table 1: Radioss on 8 modules SV1 (0% overcommitted)

PE/Module	Time (min)	% increase
1	86.97	
2	92.68	6.57%
3	104.37	12.61%
4	117.83	12.90%

Table 1: Nastran on 8 modules SV1 (0% overcommitted)

PE/Module	Time (min)	% increase
1	108.15	
2	109.12	0.90%
3	113.45	3.97%
4	121.48	7.08%

Table 2: Radioss on 8 modules SV1 (25% overcommitted)

PE/Module	Time (min)	% increase
1	161.18	
2	110.90	
3	133.53	20.41%
4	141.87	6.25%

Table 2: Nastran on 8 modules SV1 (25% overcommitted)

PE/Module	Time (min)	% increase
1	109.08	
2	135.20	23.95%
3	147.80	9.32%
4	151.68	2.63%

Table 3: Radioss on 4 cpus/module (0% overcommitted)

Modules	Time (min)	% increase
2	97.9	
4	103.32	5.54%
6	109.33	5.82%
8	117.83	7.77%

Table 3: Nastran on 4 cpus/module (0% overcommitted)

Modules	Time (min)	% increase
2	111.98	
4	114.68	2.41%
6	116.18	1.31%
8	121.48	4.56%

Table 4: Radioss on 4 cpus/module (25% overcommitted)

Modules	Time (min)	% increase
2	174.70	
4	124.87	
6	140.82	12.77%
8	141.87	0.75%

Table 4: Nastran on 4 cpus/module (25% overcommitted)

Modules	Time (min)	% increase
2	113.72	
4	142.12	24.97%
6	152.85	7.55%
8	151.68	-0.77%