SV1ex Memory Performance in a Multi-user Environment

Ulrich Detert
Research Centre Juelich (FZJ)
Institute for Applied Mathematics (ZAM)
D-52425 Juelich
Germany
Phone: (++49)2461-61-6434
Fax: (++49)2461-61-6656
U.Detert@fz-juelich.de

Abstract

After replacing a Cray T90 by an SV1ex system at FZJ we observed variations in the CPU time of user applications that were unexpectedly high. The paper describes details of our observations, gives some information on the reasons for the performance variations and outlines measures taken to reduce the observed effects. Components involved in this process include the SV1ex memory hardware, system software and also user application codes.

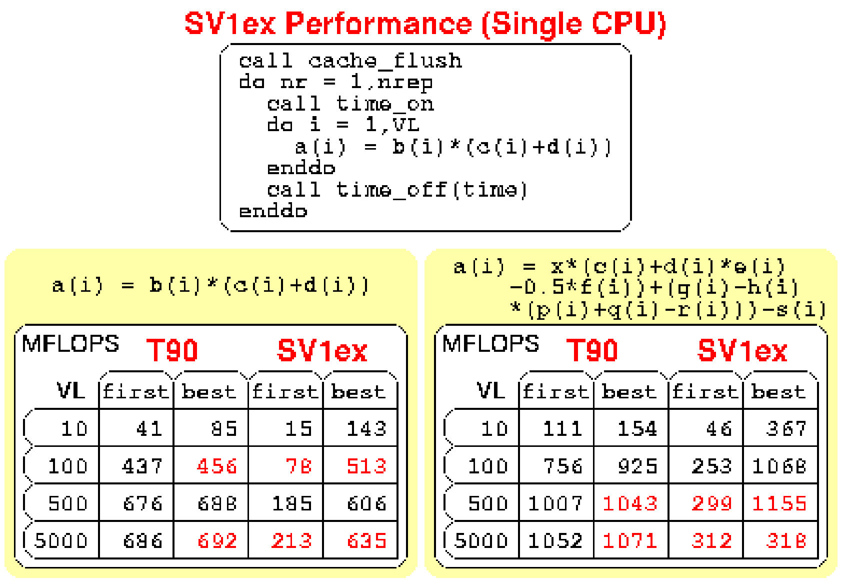
Introduction

In January 2002 FZJ replaced a Cray T90/10 and a Cray J90/16 by an SV1ex-1A system with the following characteristics:

- 16 CPUs
- 500 MHz clock
- 4 flops/clock per CPU
- 256 KB cache per CPU
- 32 GB shared memory
- 2048 memory banks
- 4x4 backplane

Performance Observations

Single CPU performance measurements showed that the performance of the SV1ex CPUs is comparable to the performance of T90 CPUs if the cache utilization is high (Fig 1). On the other hand, if the data access pattern doesn't allow for good cache reuse, the CPU performance drops significantly due to memory bandwidth constraints.



SV1ex Performance (Single CPU)

```
call cache_flush
do nr = 1,nrep
  call time_on
  do i = 1,VL
    a(i) = b(i)*(c(i)+d(i))
  enddo
  call time_off(time)
enddo
```

a(i) = b(i)*(c(i)+d(i))

MFLOPS	T90		SV1ex	
VL	first	best	first	best
10	41	85	15	143
100	437	456	78	513
500	676	688	185	606
5000	686	692	213	635

```
a(i) = x*(c(i)+d(i)*e(i)
    -0.5*f(i))+(g(i)-h(i)
    *(p(i)+q(i)-r(i)))-s(i)
```

MFLOPS	T90		SV1ex	
VL	first	best	first	best
10	111	154	46	367
100	756	925	253	1068
500	1007	1043	299	1155
5000	1052	1071	312	318

Fig. 1: Single CPU Performance

When the SV1ex system was taken into production soon after installation there were complaints from users about great variations in CPU time. Users claimed to see variations in the range up to factor 5. The variation of CPU time of user jobs was especially unpleasing, since job accounting is essentially done on the basis of CPU time. Furthermore, there were situations were identical user jobs reached their time limit in some runs but ran to completion in other runs.

In order to evaluate the extent of the CPU time variations and to find out about the reasons we decided to periodically run a probe job and plot the CPU time of the probe against the workload of the SV1ex system (Fig. 2). The probe job exhibited indeed CPU time variations in the described range. However, there was no clear correlation between the measured CPU time and the system workload.

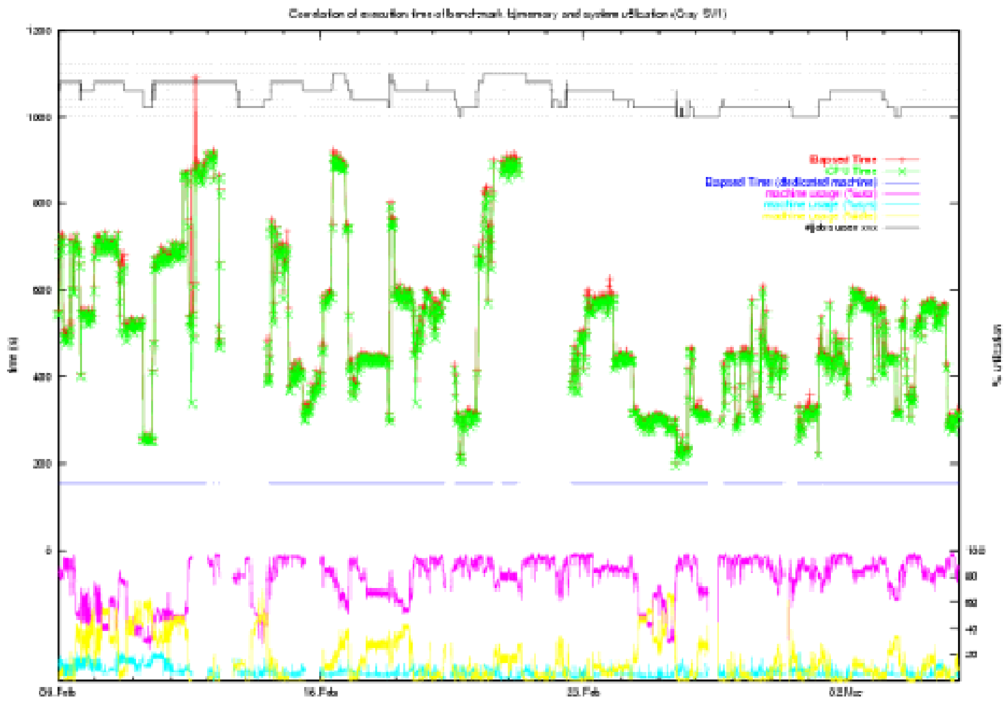


Fig. 2: CPU Time Variations of Probe Job

Memory Benchmark

For further analysis we decided to use a kernel benchmark that specifically stresses memory and cache and to run multiple copies of the benchmark on a dedicated system. The first observation was that on a dedicated system as well as under normal production load there is a strong correlation between CPU time and the number of memory conflicts per memory reference (Fig. 3). The measurements were done with hpm (Hardware Performance Monitor, group 2).

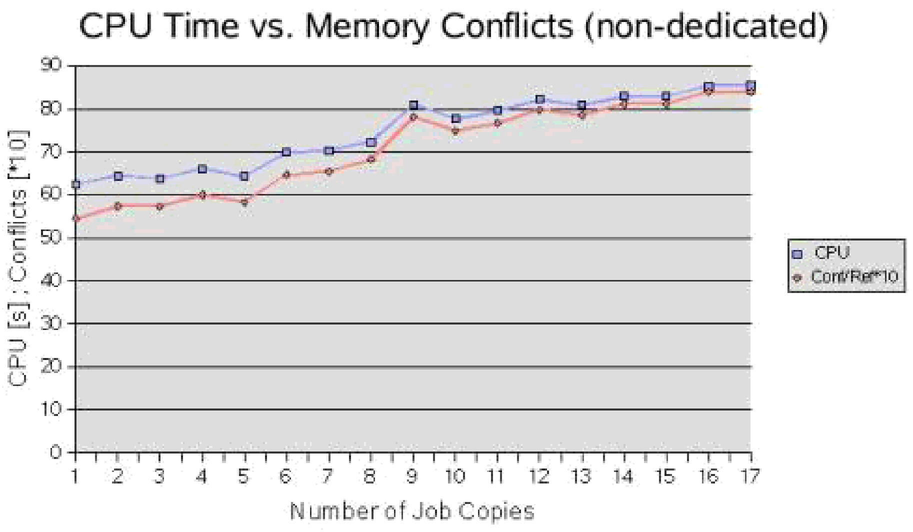


Fig. 3: CPU Time vs. Memory Conflicts

On a dedicated system the kernel benchmark loops were executed in 1 to 16 copies to produce memory load proportional to the number of job copies. The various kernel loops exhibited a different behavior with respect to CPU time and memory conflicts (Fig. 4):

- The ’cache’ loop exploits a 100% cache hit rate and thus doesn’t produce memory conflicts. The resulting CPU time curve is a straight line independent on the number of job copies.
- The ’mem’ loop has a 0% cache hit rate and its CPU time increases by factor 2 from 1 to 16 job copies.
- The ’cachemix’ loop shows approx. 35% cache hit rate for memory loads, but contains a significant number of memory stores that bypass the cache due to the write through cache architecture of SV1ex. The CPU time raises by factor 4 from 1 to 16 job copies.
- The ’mix’ loop is a combination of the other three loops and thus shows a moderate increase in CPU time (factor 1.6).

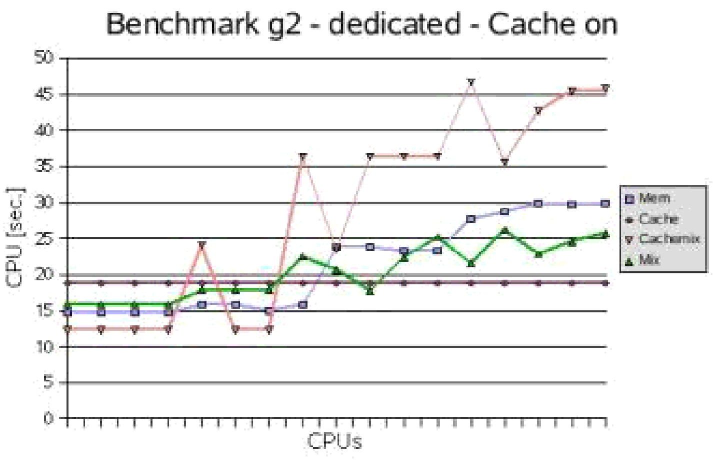


Fig. 4: CPU Time of Benchmark Loops

A remarkable property especially of the ’cachemix’ curve in Fig. 4 is the great variation in CPU time with peaks for 5 and 8 job copies e.g. These peaks hint at great variations in the CPU time for identical jobs, since in Fig. 4 the curves have been plotted from randomly selected job copies, not from the average CPU time of multiple runs.

Fig. 5 depicts this observation in more detail. There are significant differences in the CPU time of identical runs depending on the CPU numbering. When stepping from 4 to 5 job copies the CPU time of two jobs increases by factor 2 while 3 jobs remain at the previous performance level. Similar occurs when going from 5 to 6 job copies. 16 jobs finally exhibit a balanced performance at a low performance level, however.

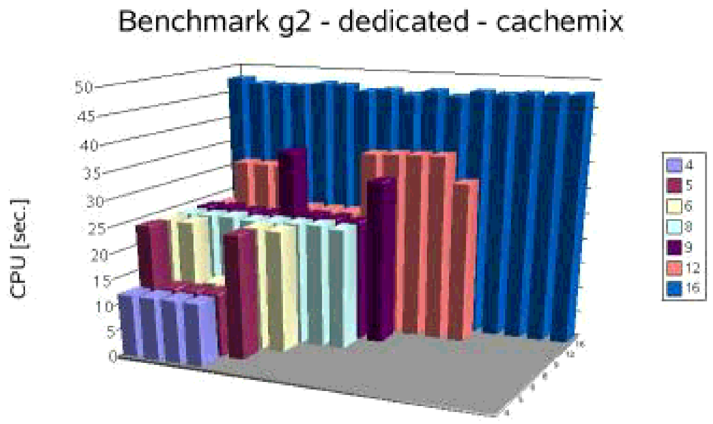
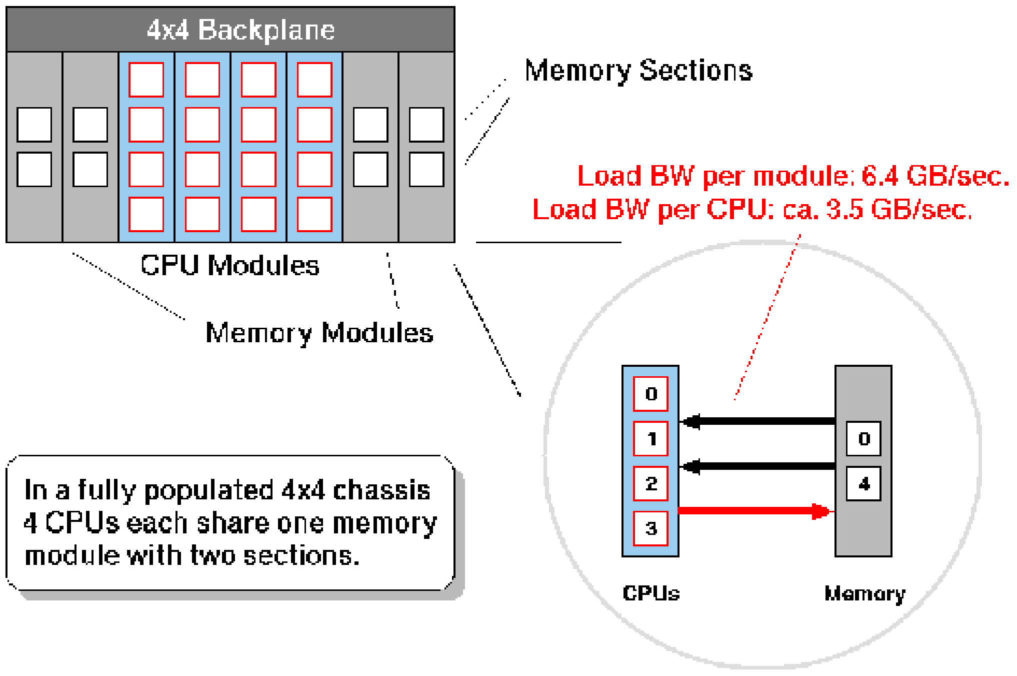


Fig. 5: CPU Time Variation for ’cachemix’ Loop

Memory Architecture

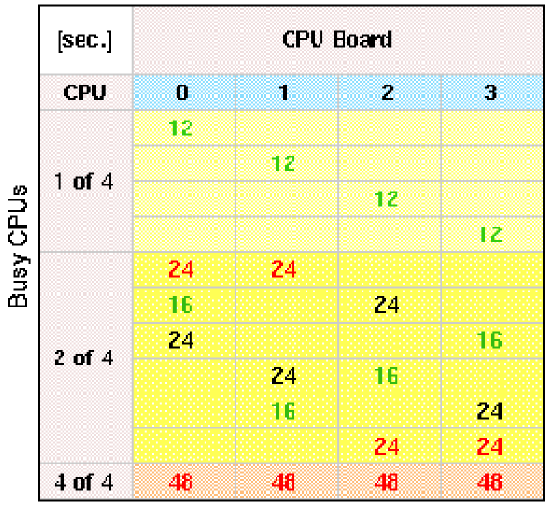
The reason for the observed performance behavior can be found in the SV1ex memory architecture. Fig. 6 shows the basic principal of the SV1ex memory with a 4x4 backplane. 4 CPU modules share 4 memory modules, so, in a fully populated system, 4 CPUs each share one memory module with two sections. On memory loads each CPU can transfer data at a rate of approximately 3.5 GB/s. Since one memory module is capable of delivering data at a rate of 6.4 GB/s, two CPUs can absorb the entire memory bandwidth. On memory stores the situation is even worse, since the available memory bandwith is only half the bandwidth of loads.



4x4 Backplane
Memory Sections
Load BW per module: 6.4 GB/sec.
Load BW per CPU: ca. 3.5 GB/sec.
CPU Modules
Memory Modules
In a fully populated 4x4 chassis 4 CPUs each share one memory module with two sections.
0
1
2
3
0
4
CPUs
Memory

Fig. 6: SV1ex-1A Memory Architecture

The memory bandwith that is available to a given CPU is furthermore dependent on the CPU numbering on the module. Fig. 7 shows the CPU time measurements of the ’cachemix’ loop for 1, 2, and 4 copies. All jobs ran on CPUs in the same module. If just one job is run it requires 12 sec. CPU time independent of the CPU’s location on the board. If four copies are run, the situation is balanced as well (48 sec. for each job). If two jobs are run, however, the required CPU time depends on the location of the involved CPUs. There are bad combinations (CPUs 0+1 or CPUs 2+3 with 24 sec. per job) and not so bad combinations like CPUs 0+2 or 0+3 e.g. For two jobs per module there are no good and balanced combinations, however.



	[sec.]	CPU Board			
	CPU	0	1	2	3
Busy CPUs	1 of 4	12			
			12		
				12	
					12
	2 of 4	24	24		
		16		24	
		24			16
			24	16	
			16		24
				24	24
	4 of 4	48	48	48	48

Fig. 7: CPU Time Dependent on CPU Location

It is worthwhile noting that the Unicos kernel scheduler recognizes this situation and places jobs on empty CPU modules if possible and - as of Unicos 10.0.1.1 - avoids bad CPU combinations, if there are two jobs per module.

User Codes

Unfortunately, the above results do not explain the observed CPU time variations during production on a fully loaded system. Additional investigations revealed that high CPU time measurements in our probe job were strongly correlated with the existance of one specific user job in the system workload.

Investigations showed that this job used stride 2048 memory access for a large number of array computations (Fig. 8).

```
      PARAMETER (NN=1024)
      PARAMETER (WW=2*NN)
      ...
      REAL*8 EMA(WW,WW),EVE1(WW),ZEVE1(WW)
      ...
      DO 22 L1=M1+1,WW
        VOLDA=VOLDA+EMA(M1,L1)
        VOLDC=VOLDC+CMA(M1,L1)
22    CONTINUE
      ...
      DO 52 L1=M1+1,WW
        EMA(M1,L1)=EVE1(L1)
        CMA(M1,L1)=CHIVE1(L1)
52    CONTINUE
      ...
```

Fig. 8: User Code with Stride 2048 Memory Access

Since the SV1ex memory utilizes 2048 memory banks, the above code segment produces a tremendous amount of memory bank conflicts. Different from the Cray T90 memory system e.g. these conflicts not only slow down the originating program, but also all other jobs running in the system. So, on Cray SV1ex systems program optimization has become of even greater importance than before.

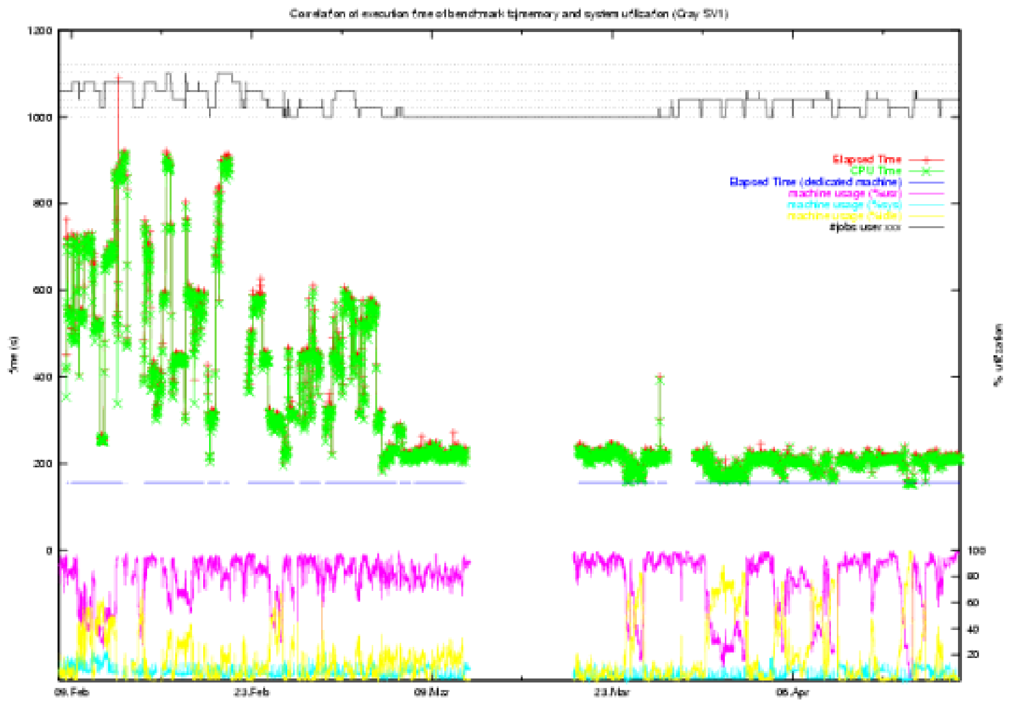


Fig. 9: CPU Time Variations Before and After Program Optimization

Fig. 9 shows the CPU time variations of our probe job before and after the optimization of the user code that caused the memory conflicts. It will be an issue for the future to ensure that no other job in the system workload undermines the performance of the entire system.

Acknowledgement

The author would like to thank Wolfgang Frings at FZJ and all people at Cray Inc. who helped analysing and solving the observed problems. Special thank is devoted to Wilfried Oed, Thomas Erdmann and Ralph Krotz.