SPEC CPU2006 Memory Footprint

John L. Henning
Sun Microsystems.
Contact john dot henning at acm dot org

Memory Footprint Goal

The nominal goal for memory consumption by SPEC CPU2006 benchmarks is up to about 900 MB when compiled with 32-bit pointers. The 900 MB maxium was chosen so that a system with 1GB will have about 100MB available for the operating system and overhead processes. By comparison, the goal for SPEC CPU2000 was 200MB [1].

The memory goals are only approximate, because there are many factors that affect memory consumption. For example, some compilers might choose to pad arrays for performance reasons, while others do not. Some operating systems may allocate single pages of memory as they are touched, while others allocate larger regions earlier. The choice of page sizes may cause memory to be conserved or wasted. Perhaps most significantly, if the benchmarks are compiled for 64-bit pointers, memory consumption can be significantly higher than with 32-bit pointers.

Metrics

The observed memory footprint also depends on which metrics are used in the observation. This paper looks at two metrics commonly available with the Unix "ps" command:

- vsz - the virtual size, or total address space
- rss - the resident set size, or actual allocated physical memory.

Related papers discuss effects of differing page sizes [2] and memory locations that are not merely allocated, but also actually used by the code [3].

CPU2000 vs. CPU2006 rss and vsz

The tables below show virtual and resident sizes for both SPEC CPU2000 and SPEC CPU2006. The benchmarks were compiled using the Sun Studio 11 compiler set. The tuning used 32-bit pointers and "-fast -xpagesize=4M".

The benchmaks were run on a Sun Blade 2000 system with the Solaris 10 operatiing system, UltraSPARC-III+ processors, and 8GB physical memory. Memory usage was observed with 'ps -o rss,vsz' every 5 seconds. The reported values are the maximums seen for each benchmark.

In the far right column, CPU2006 benchmarks marked "(s)" are stable in their memory requirements. These benchmarks grow quickly to the listed sizes, and remain there.

SPECint2000			SPECint2006			
	Rss	vsz		rss	vsz	
164.gzip	188	188	400.perlbench	580	581	
175.vpr	44	45	401.bzip2	856	856	
176.gcc	148	149	403.gcc	932	933	
181.mcf	100	101	429.mcf	844	845	(s)
186.crafty	7	9	445.gobmk	28	29	
197.parser	32	37	456.hmmer	60	61	
252.eon	7	11	458.sjeng	180	180	(s)
253.perlbmk	120	121	462.libquantum	104	105	
254.gap	200	201	464.h264ref	68	69	
255.vortex	96	97	471.omnetpp	121	122	
256.bzip2	192	192	473.astar	313	314	
300.twolf	6	9	483.xalancbmk	340	342	

Table 1: Integer benchmarks memory usage (MB)

SPECfp2000			SPECfp2006			
	rss	vsz		rss	vsz	
168.wupwise	185	200	410.bwaves	881	894	(s)
171.swim	201	216	416.gamess	39	670	
172.mgrid	61	76	433.milc	676	677	
173.applu	196	212	434.zeusmp	525	1138	(s)
177.mesa	20	29	435.gromacs	25	38	(s)
178.galgel	129	174	436.cactusADM	879	1018	(s)
179.art	8	9	437.leslie3d	129	142	(s)
183.equake	48	49	444.namd	53	54	(s)
187.facerec	61	76	447.dealII	564	566	
188.ammp	20	21	450.soplex	457	626	
189.lucas	149	164	453.povray	9	10	(s)
191.fma3d	112	128	454.calculix	216	230	
200.sixtrack	64	84	459.GemsFDTD	838	850	(s)
301.apsi	198	212	465.tonto	47	65	
			470.lbm	416	417	(s)
			481.wrf	701	718	
			482.sphinx3	48	49	(s)

Table 2: Floating point benchmark memory usage (MB)

Observations

From the above tables, it can be noted that:

(a) About half of the CPU2006 benchmarks are larger than the maximum allowed for CPU2000 (200MB)

(b) All of the SPECint2006 benchmarks, and about half of the SPECfp2006 benchmarks, have resident and virtual sizes that are close to each other.

(c) The benchmark with the largest ratio between virtual and physical memory is 416.gamess, which allocates a large area of memory at startup, and then manages its own memory use internally during execution.

(d) Only one benchmark exceeds 900MB for its resident size, namely 403.gcc, which grows to 932MB. Approximately 7% of the 'ps' observations during the run of 403.gcc showed sizes over 900MB. See the graph of 403.gcc memory consumption vs. time later in this report.

(e) 434.zeusmp allocates 1138MB of virtual memory, but the resident size is only 525MB. The difference is due to a group of arrays that the setup routines allocate in COMMON blocks, /rado/ and /radr/, but which are not actually used. Because the pages are never written, the Solaris operating system never allocates physical memory for them.

(f) 436.cactusADM also allocates more than 900MB of virtual memory, while staying under 900MB of physical memory. In this case, the difference is due to virtual memory padding added by the C setup routines. The padding is not touched by the Fortran routines that consume most of the CPU time.

The routine SetupCache.c goes to considerable trouble to try to avoid cache conflicts for the many arrays that are used by the program, adding 8 MB of padding, plus an additional smaller pad, so that the virtual addresses of the start of each array end up on different physical cache lines.

Many real-world scientific codes attempt to be cache-friendly, as do many other SPEC benchmarks; but 436.cactusADM is probably the first SPEC CPU benchmark to go to so much trouble to influence array starting addresses. Although it could be argued that such attempts should be tuned to each platform where a program is to be run, it appears that the particular padding amounts chosen by the benchmark are reasonably harmless on several tested platforms.

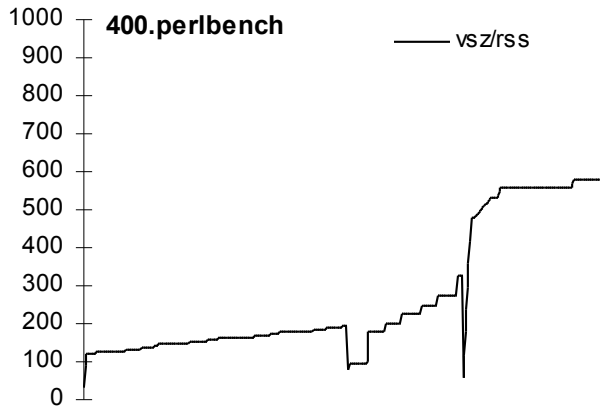
(g) Finally, it is perhaps worth emphasizing that memory measurements differ depending on systems and methods. If one compares the CPU2000 columns above to [1], there are differences, even though the same author used the same method (ps -ef -o rss,vsz). The most striking difference is for 181.mcf, which required nearly twice as much memory in study [1] than it does for the system tested here, because [1] used 64-bit pointers. Differences can also be observed when comparing the rss/vsz statistics between this paper vs. both [2] and [3], notably for 434.zeusmp and 436.cactusADM. On the IBM system, the extra virtual memory discussed above is not observed.

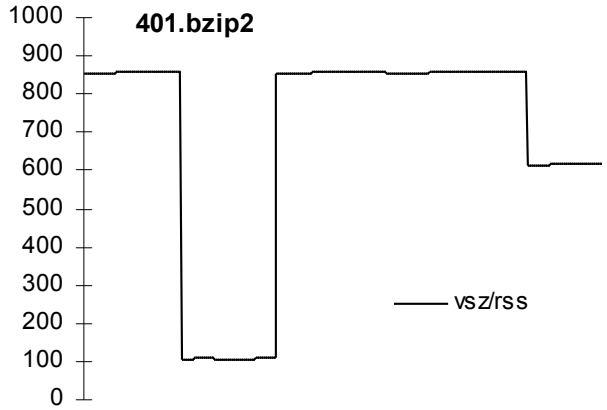
Integer benchmarks that vary over time – Part 1

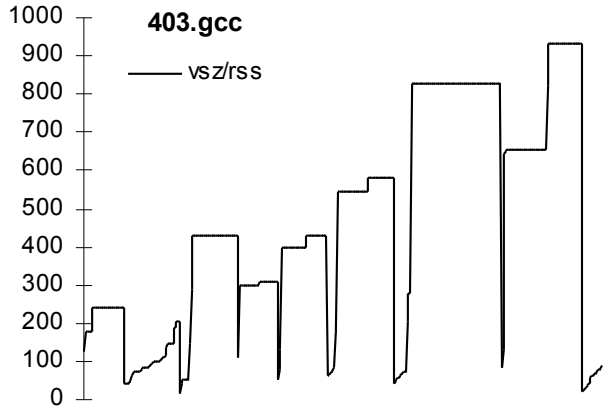
The tables above noted that some benchmarks are (s)table in their memory requirements. Other benchmarks vary in their memory requirements over time, and are shown in this and succeeding sections in graphic form. All graphs use the same conventions: the Y axis is the number of MB, and the X axis is time. One point is plotted for each of the 5-second interval observations.

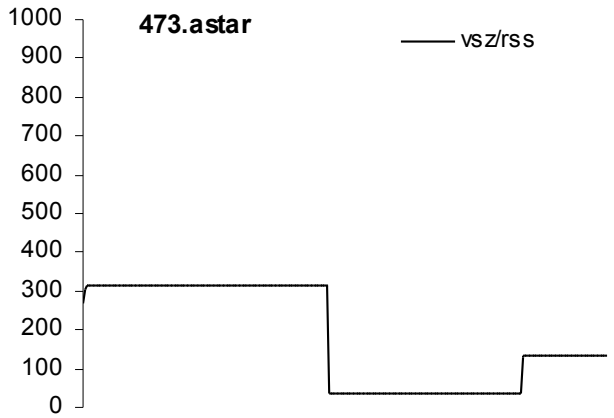
This section shows the 5 integer benchmarks that vary over time. For each of the benchmarks shown on this page, vsz and rss are indistinguishable at the scale shown here, i.e. 1000MB as the maximum Y value.

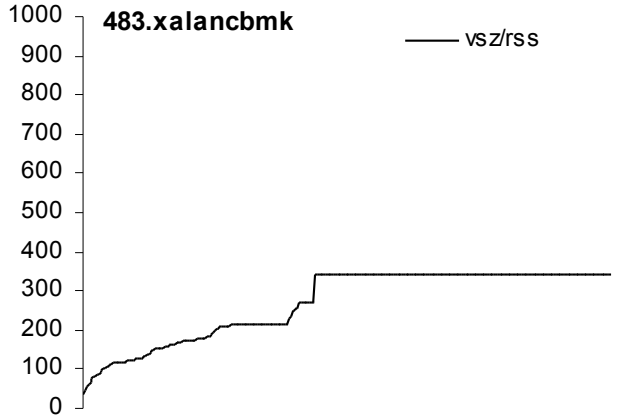
From the graphs, it is apparent that some benchmarks are invoked multiple times during the course of a run. For example, the description for 403.gcc says that it compiles 9 programs [4], and the eye can pick out about 9 different levels of memory consumption as it does so.







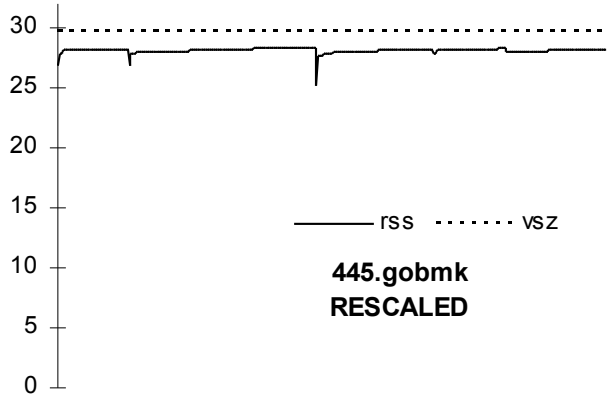


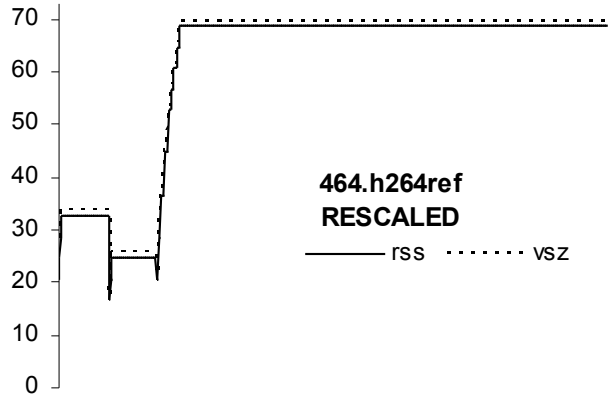


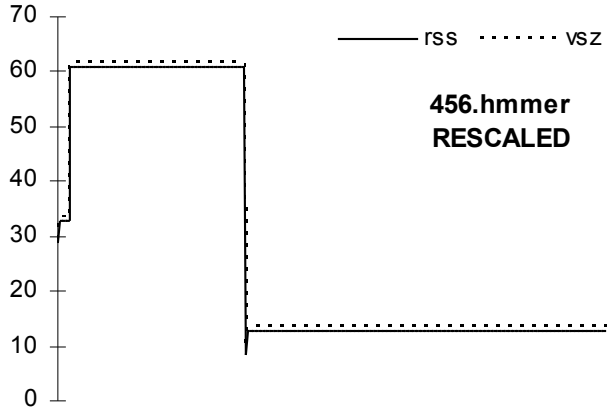
Integer benchmarks that vary over time – Part 2

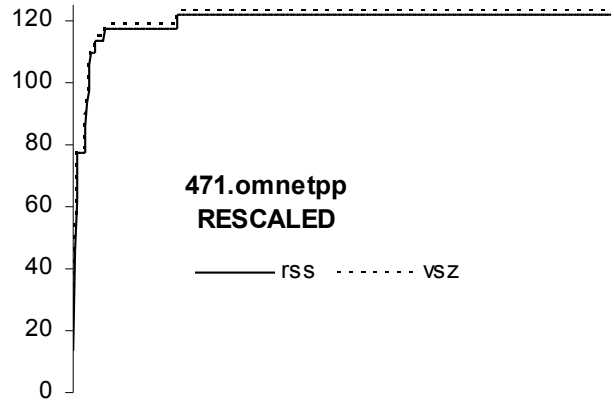
The integer benchmarks shown in this section vary over time, but the variation is too small to be shown at the same scale as the previous section. Therefore, these are shown rescaled

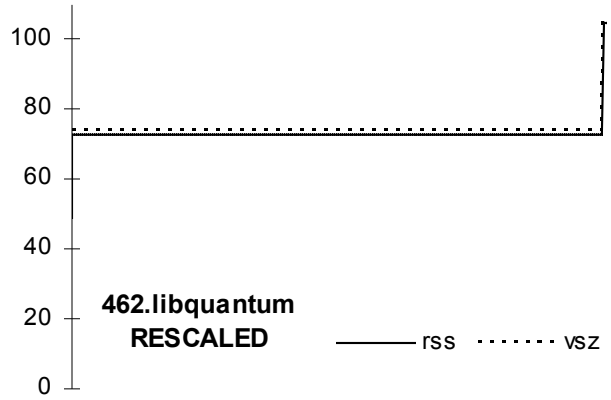
.





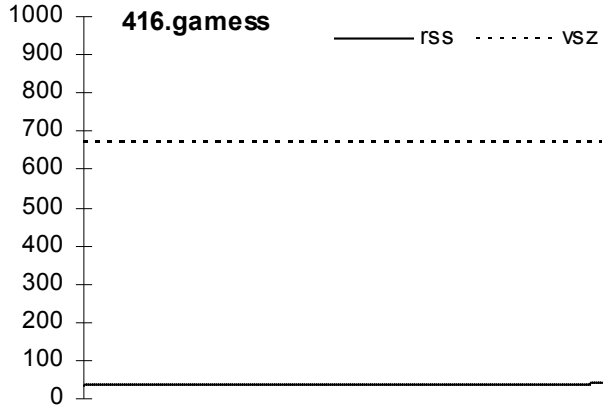


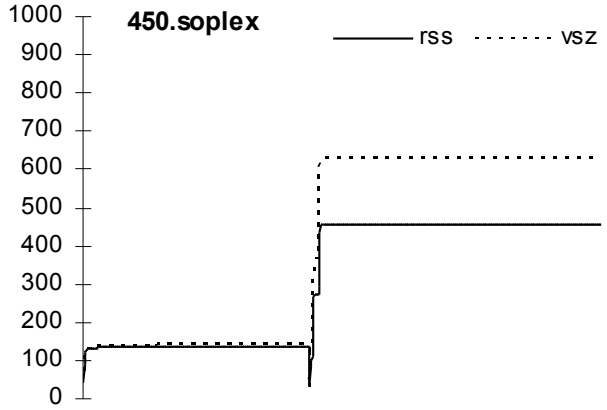


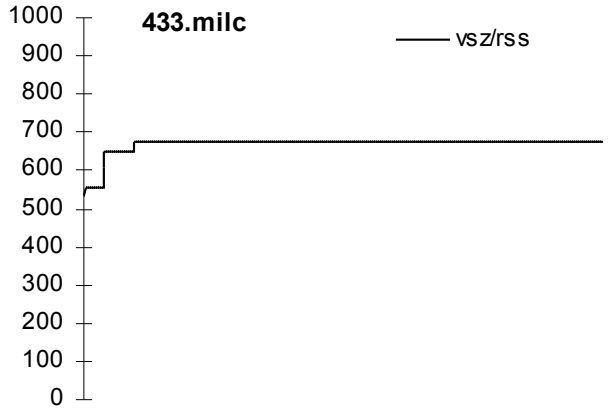


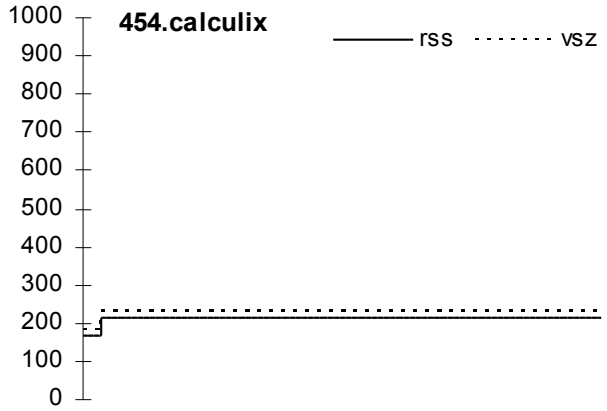
Floating Point Benchmarks that Vary Over Time – Part 1

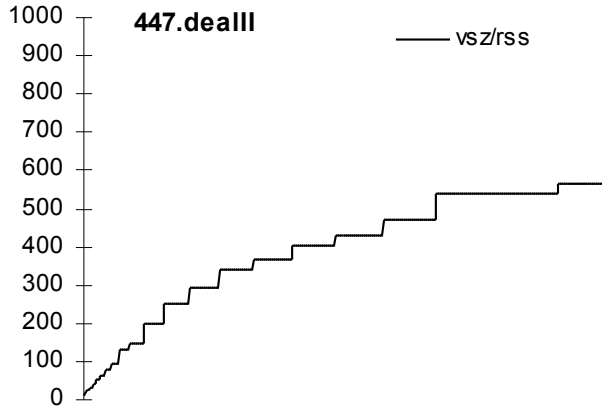
The floating point benchmarks shown in this section are all graphed on the same scale.

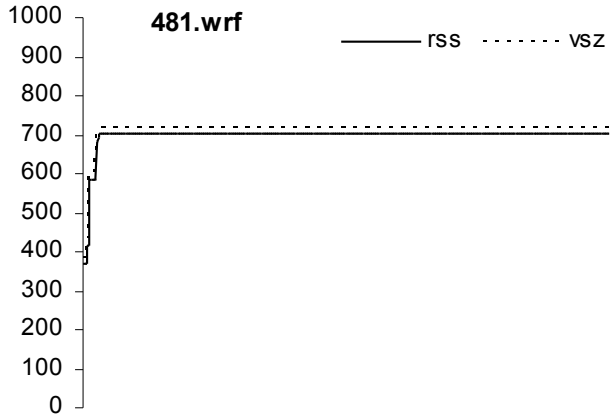






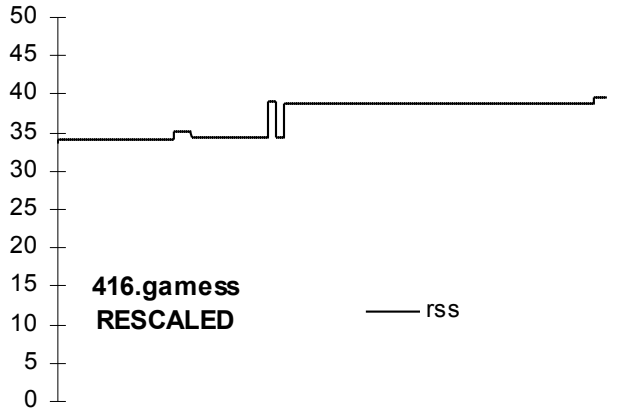


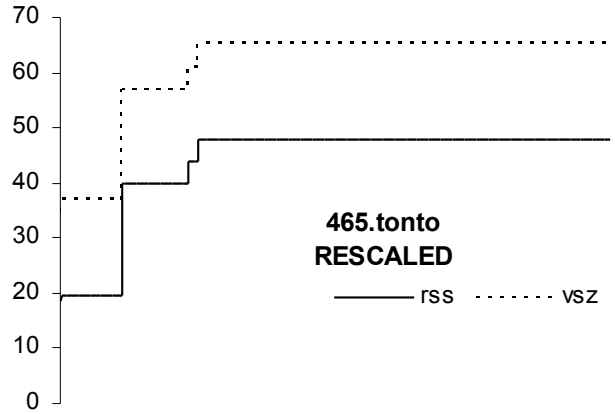




Floating Point Benchmarks that vary over time –part 2

Finally, the floating point benchmarks shown in this section have been rescaled for ease of viewing. Note that 416.gamess is shown in both the previous section and this section, because its rss behavior is so different from its vsz behavior.





Summary

Memory usage for the SPEC CPU2006 benchmarks is summarized both in tabular form and, for benchmarks that vary over time, in graphic form. The metrics used are from the Solaris 'ps' utility; other metrics would produce different results.

Acknowledgments

Thank you to Peter Farkas, Geetha Vallabhaneni, and Joel Williamson for contributions to the analysis of the benchmark memory consumption.

References

[1] John L. Henning, "SPEC CPU2000 Memory Footprint", www.spec.org/cpu2000/analysis/memory

[2] Wendy Korn and Moon S. Chang, "SPEC CPU2006 Sensitivity to Memory Page Sizes", Computer Architecture News, Volume 35, No. 1, March 2007.

[3] Darryl Gove, "SPEC CPU2006 Working Set Size", Computer Architecture News, Volume 35, No. 1, March 2007.

[4] SPEC's benchmark descriptions are posted at www.spec.org/cpu2006