Performance Characterization of SPEC CPU2006 Integer Benchmarks on x86-64 Architecture

Dong Ye†, Joydeep Ray‡, Christophe Harle‡, and David Kaeli†
†ECE Department, Northeastern University, Boston, MA 02115
{dye, kaeli}@ece.neu.edu
‡Advanced Micro Devices, Austin, TX 78741
{joydeep.ray, christophe.harle}@amd.com

EXTENDED ABSTRACT

As x86-64 processors become the CPU of choice for the PC (personal computer) market [1], it becomes increasingly important to understand the performance benefits we should expect as we migrate applications from 32-bit environments to 64-bit environments. For applications that require the additional memory addressing capability provided by 64-bit computing (e.g., commercial databases and digital content authoring tools), it is not surprising to see that x86-64 has become the platform of choice. However, for less-demanding desktop applications that can fit in a 32-bit address space, we would like to know whether we can obtain any performance benefit by moving to this wider instruction set architecture.

In this work [2], we report on the performance differences obtained when benchmarks are compiled as 32-bit binaries versus compiled as 64-bit binaries (and both binaries are run natively on an x86-64 based system.) Using the experimental system shown in Table I, we run 64-bit binaries in 64-bit mode and 32-bit binaries in compatibility mode (we refer to this mode as *32-bit mode* in this paper.)

CPU	AMD Athlon™ 64 X2 4400+, 2.2 GHz, dual-core, with an integrated dual-channel DDR memory controller
Memory	2 DIMMs of 1GB DDR400 memory modules, with peak memory bandwidth 6.4 GB/s
OS	Novell SUSE® Linux Professional 9.3 x86-64 Edition [3], run level 3, kernel 2.6.11
Compiler	GCC 4.1.1, "-O3" turned on to build all benchmarks except 400.perlbench, "-O2" turned on to build 400.perlbench

TABLE I
THE EXPERIMENTAL SYSTEM

We studied the integer benchmarks taken from the SPEC CPU2006 suite [4]. We used the GCC 4.1.1 C/C++ compiler [5] to generate both 64- and 32-bit binaries. For each benchmark, we used the same optimization flags to generate both binaries. In addition to optimization flags, we used the *-64* and *-m32* switches to generate the respective 64-bit and 32-bit binaries. We run all the experiments on the single platform shown in Table I. In order to reduce the variation introduced by an operating system with SMP support on a multi-core system, we run benchmarks exclusively on one core by wrapping around the benchmark invocation command inside the operating system's CPU affinity binding command. To minimize the variations introduced by multitasking, we run the operating system at level 3 and disable some heavy-weight daemon processes while we collected performance data.

We have observed that for the SPEC CPU2006 integer benchmarks, 64-bit mode offers a sizable performance advantage over 32-bit mode (7% on average) as shown in Figure 1. However, the advantage (or disadvantage) varies from benchmark to benchmark. For a handful of programs, running in 64-bit mode is significantly slower than running in 32-bit mode. We collected a number of performance characteristics of these benchmarks (e.g., code size, dynamic instruction count, runtime memory footprint, cache behavior as well as memory controller utilization). In these studies, we have focused on the difference of various performance characteristics between these two modes.

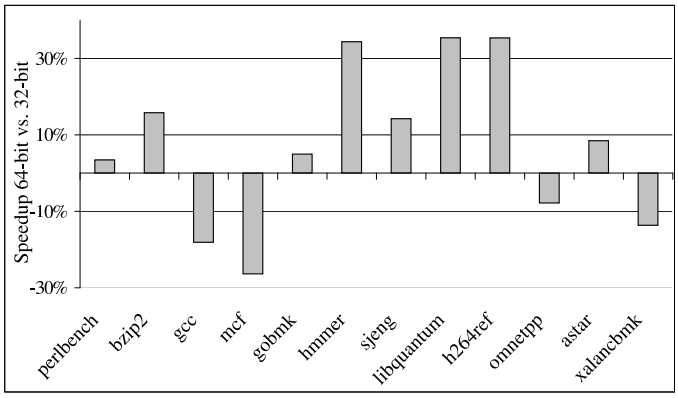

Fig. 1. 64-bit vs. 32-bit speedup in CPU2006int.

We further analyze five benchmarks: 429.mcf, 456.hmmer, 462.libquantum, 464.h264ref, and 483.xalancbmk. These benchmarks have shown some biggest differences between the performance observed under these two modes. We aim to better understand the program characteristics that favor or disfavor a 64-bit architecture.

Table II presents some key performance metrics (64-bit vs. 32-bit mode differences) of these selected five benchmarks and a summary of the reasons responsible for the observed performance differences.

Benchmark	Run time increase	Memory footprint increase	Dynamic instruction count decrease	Data cache request rate increase	Cause of performance difference
429.mcf	26.35%	100.12%	5.74%	33.40%	Larger memory footprint due to use of long and pointer data types in 64-bit mode.
456.hmmer	-34.34%	9.84%	8.72%	15.80%	More registers available in 64-bit mode.
462.libquantum	-35.38%	-31.44%	53.71%	62.50%	Native 64-bit integer arithmetic in 64-bit mode.
464.h264ref	-35.35%	5.73%	9.96%	22.94%	Faster calling convention (because of more registers) in 64-bit mode.
483.xalancbmk	13.65%	33.87%	7.60%	28.32%	Larger memory footprint due to pointers in 64-bit mode.

TABLE II
PERFORMANCE DIFFERENCES OF FIVE SPEC CPU2006 INTEGER BENCHMARKS (64-BIT MODE RELATIVE TO 32-BIT MODE.) AS WELL AS THEIR FIRST-ORDER REASONS

Some of the common traits of these programs that lead to performance benefits for 64-bit mode are: (1) The use of 64-bit integer arithmetic; (2) The presence of loop bodies that require many registers (note that loop unrolling is a common compiler optimization that can also increase register pressure); (3) Many calls to small functions that can be economically inlined.

Some major traits of these programs that present potential performance degradation for 64-bit mode are: (1) Memory intensive applications; especially those that have already experienced a high data cache miss rate in a 32-bit environment; (2) Intensive use of long and pointer data types in terms of the amount of memory allocated for such data types and the frequency of their access.

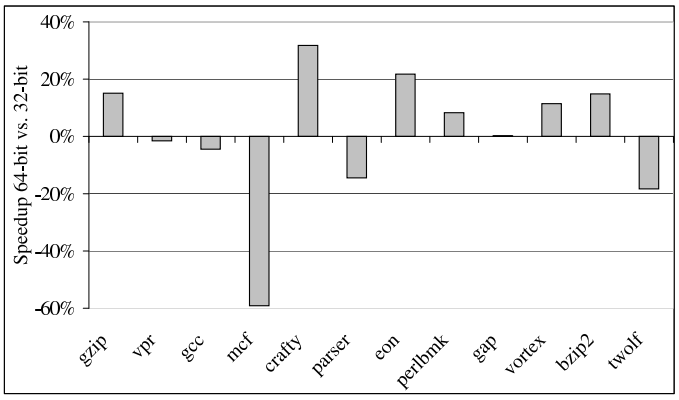

Fig. 2. 64-bit vs. 32-bit speedup for CPU2000int.

Finally, we take a look at the performance difference between these two modes as observed by the integer benchmarks of the previous generation of SPEC CPU suite: CPU2000 [6]. On average, CPU2000 integer benchmarks obtain very little (less than 1%) performance gain when run in 64-bit mode compared to 32-bit mode, as shown in Figure 2. This lack of performance improvement from 32-bit mode to 64-bit mode is largely due to one benchmark, *mcf*. As included in both suites, this particular benchmark underwent some source code changes when moving from CPU2000 to CPU2006. Interestingly, the dip in performance in 64-bit mode is much smaller for *mcf* in CPU2006 than for *mcf* in CPU2000. When moving to CPU2006, several heavily populated data structures in this benchmarks were changed to use an *int* data type instead of a *long* data type. This change helps to reduce the impact of the increased memory footprint when moving to 64-bit, which is one major reason why *mcf* experiences a cache performance hit in 64-bit mode.

ACKNOWLEDGMENT

This work was done when Dong Ye worked in AMD Austin as a co-op engineer. AMD and AMD Athlon 64 are trademarks of Advanced Micro Devices, Inc. SUSE® is the registered trademark of Novell, Inc. Linux® is the registered trademark of Linus Torvalds.

Disclaimer: All performance numbers reported in this paper are estimates because they not fully compliant with SPEC run and reporting rules [7]. It is expected, though not proven, that results from a fully compliant run would be very close.

REFERENCES

[1] K. J. McGrath and D. Christie, "The AMD x86-64 ISA: Extending the x86 to 64-bits," in *Hot Chips 14*, August 2002.
[2] D. Ye, J. Ray, C. Harle, and D. Kaeli, "The Performance Characterization of SPEC CPU2006 Integer Benchmarks on x86-64 Architecture," in *2006 IEEE International Symposium on Workload Characterization (IISWC 2006)*, October 2006, pp. 120–127.
[3] Novell. SUSE Linux. [Online]. Available: http://www.novell.com/products/suselinux
[4] J. L. Henning, "SPEC CPU2006 Benchmark Descriptions," *ACM SIGARCH Computer Architecture News*, vol. 34, no. 4, pp. 1–17, September 2006.
[5] Free Software Foundation. GCC, GNU Compiler Collection. [Online]. Available: http://gcc.gnu.org
[6] J. L. Henning, "SPEC CPU2000: Measuring CPU Performance in the New Millennium," *IEEE Computer*, vol. 33, no. 7, pp. 28–35, July 2000.
[7] SPEC. SPEC CPU2006 Documentation. [Online]. Available: http://www.spec.org/cpu2006/Docs