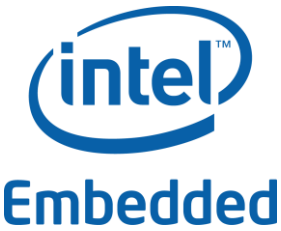


White Paper

Vinodh Gopal
Jim Guilford
Wajdi Feghali
Erdinc Ozturk
Gil Wolrich

IA Architects
Intel Corporation

High Performance DEFLATE Compression on Intel® Architecture Processors

November 2011

326352-001

Executive Summary

There is a critical need for lossless data compression in enterprise storage and applications such as databases and web servers, which process huge amounts of data. DEFLATE is a widely used standard to perform lossless compression, and forms the basis of utilities such as gzip and libraries such as Zlib. In these applications, compression imposes a large computational burden on the servers, and they could benefit from a highly optimized implementation. This paper describes the performance characteristics of fast prototype implementations of DEFLATE compression, on Intel® processors based on the 32-nm micro-architecture. As the performance of compression is data dependent, we report the performance on various industry standard corpora data sets.

This paper describes the performance characteristics of fast prototype implementations of DEFLATE compression. In terms of throughput, we are able to perform DEFLATE compression at the aggregate rate of ~**2.7 Gigabits/sec** on the Calgary Corpus data-set, on a **single core** of an Intel® Core™ i5 650 processor.[1]

*Our fastest DEFLATE compression implementation is ~**4.5 times** as fast as the fastest mode of the best open source version of Zlib compression,*

[1] Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Configurations: Refer to the Performance section on page 6. For more information go to http://www.intel.com/performance.

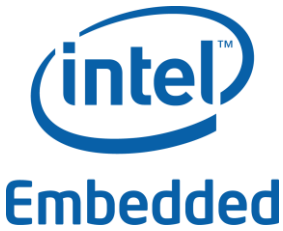
on the Intel® Core™ i5 processor 650. To achieve such large performance gains, we sacrifice a small amount of compressibility compared to Zlib-1.[2]

The Intel® Embedded Design Center provides qualified developers with web-based access to technical resources. Access Intel Confidential design materials, step-by step guidance, application reference solutions, training, Intel's tool loaner program, and connect with an e-help desk and the embedded community. Design Fast. Design Smart. Get started today. www.intel.com/embedded/edc.

[2] Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Configurations: Refer to the Performance section on page 6. For more information go to http://www.intel.com/performance.

Contents



Overview

There is a critical need for lossless data compression in enterprise storage and applications such as databases and web servers, which process huge amounts of data. DEFLATE [1] is a widely used standard to perform lossless compression, and forms the basis of higher level specifications such as gzip [5] and Zlib [6]. In these applications, compression imposes a large computational burden on the servers, and they could benefit from a highly optimized implementation.

This paper describes the performance characteristics of fast prototype implementations of DEFLATE compression, on Intel® processors based on the 32-nm microarchitecture. As the performance of compression is data dependent, we report the performance on various industry standard corpora data sets such as the Calgary [7], Canterbury [8] and Silesia Corpus [9]. Our fast implementations are completely compliant with the DEFLATE standard; thus their output can be decompressed by any publicly available compliant de-compressor such as Zlib.

DEFLATE Compression

Background

The DEFLATE compressed data format consists of a series of blocks, corresponding to successive blocks of input data. Each block is compressed using a combination of the LZ77 [3] algorithm and Huffman coding [2]. The LZ77 algorithm finds repeated substrings and replaces them with backward references (relative distance offsets). The LZ77 algorithm can use a reference to a duplicated string occurring in the same or previous blocks, up to 32K input bytes back.

A compressed block can have either static (fixed codes defined in the standard) or dynamic Huffman codes. Each dynamic block consists of two parts: a pair of Huffman code trees that describe the representation of the compressed data part, and the compressed payload. The compressed data consists of a series of elements of two types: literal bytes and pointers to replicated strings, where a pointer is represented as a pair <length, backward distance>. The DEFLATE format limits distances to 32K bytes and lengths to 258 bytes, with a minimum length of 3.

Each type of token or value (literals, distances, and lengths) in the compressed data is represented using a Huffman code, using one code tree for literals and lengths and a separate code tree for distances.

Our Implementations

We developed highly optimized prototype implementations of DEFLATE compression, that work efficiently on data buffers of different types and sizes. We refer to these implementations as "**igzip**". It is possible to get further performance gains for specific usages that target very small data buffers, work only on fixed sized buffers or have very different compression ratios. Such optimizations have not been attempted and are beyond the scope of the current paper.

Our optimizations focused on the hashing, longest prefix match of substrings for the LZ processing, and the Huffman code flow. There are different ways to design the data structures (compared to the Zlib method) for hash lookups as well as the Huffman encoding. We also manage the history buffer in various ways for efficient substring matching, given the size of the first level data cache, the size of other data structures and managing the input/output stream buffers. We sacrifice a small amount of compressibility to achieve very high performance gains. We developed 2 versions with varying speed/compressibility trade-offs, referred to as **igzip0** and **igzip1**.

In the gzip format, we also need to compute a 32-bit CRC of the uncompressed buffer. We do this efficiently using the PCLMULQDQ instruction as described in [4]. The CRC computation is a small proportion of the overall compression compute time.

Performance

The performance results provided in this section were measured on an Intel® Core™ i5 processor 650 at a frequency of 3.20 GHz, supporting Intel PCLMULQDQ instruction. The tests were run with Intel® Turbo Boost Technology off, and represent the performance with and without Intel® Hyper-Threading Technology (Intel® HT Technology) on a **single core**. We present the results with data in memory, excluding file I/O, to reflect a variety of usage models.

Since the compression performance depends on the type and size of the compressed data, we study the performance on a set of industry-standard corpus data sets.

To have a fair comparison of the performance of our "**igzip**" implementations to Zlib, we compiled the Zlib compressor with the ICC Compiler with aggressive optimizations. Our implementation is largely in optimized assembly code and thus less sensitive to compilers. We used version 1.2.5 of Zlib and generated 64-bit code. We built Zlib to generate 64-bit code with assembly support (i.e. we built it in such a way as to include the asm modules that are present in the zlib distribution).

The performance results represent all compute elements such as LZ String processing, the actual process of encoding the symbols, CRC, etc excluding File IO time.

Methodology

To measure the performance of a compressor on the given files in the data-set, we first read the files into a large memory buffer, and then run a warm-up sequence of unrelated computations to ensure that the processor core clock frequency has reached the rated frequency and is not low due to power-states. At this point, we call the function to compress the content of the files in this memory buffer and generate the output in another memory output buffer. We measure the cycles consumed by each function call and at the end of this first sequence, we get one set of timings for the compression on each file. We repeat this process a large number of times, collecting many timing measurements for the files.

For each file, we then sort the timings, discard the top and bottom quartiles and average the rest. This average gives a stable performance of the compressor over all files. The same methodology is followed for the other compressors.

The files in the data-set range from a few Kbytes to several Mbytes in uncompressed size. As the amount of memory required to process the larger files is significantly larger than the size of the last-level Cache, we expect cache misses to memory. A large number of the files have sizes that will fit in the last-level cache, but not entirely in the lower levels. As a result, we expect to see a mix of warm and cold data in these performance tests. However, since we ensure that we do not call a compress function on the same file back to back, the data is more likely to be evicted from the lower-level caches and to some extent from the last-level cache.

The timing is measured using the **rdtsc**() function which returns the processor time stamp counter (TSC). The TSC is the number of clock cycles since the last reset. The 'TSC_initial' is the TSC recorded before the function is called. After the function is complete, the **rdtsc**() is called again to record the new cycle count 'TSC_final'. The effective cycle count for the called routine is computed using

# of cycles = (TSC_final-TSC_initial).

A large number of such measurements are made for each file and then averaged as described above.

When 2 such identical threads are run simultaneously, the number of cycles measured for the function represents the cycles consumed in the core for 2 data buffers (from each thread) and we calculate the cycles/byte accordingly. Thus, if each thread was compressing buffers of size 1KB, then the net cycles/byte = # of cycles / 2*1KB

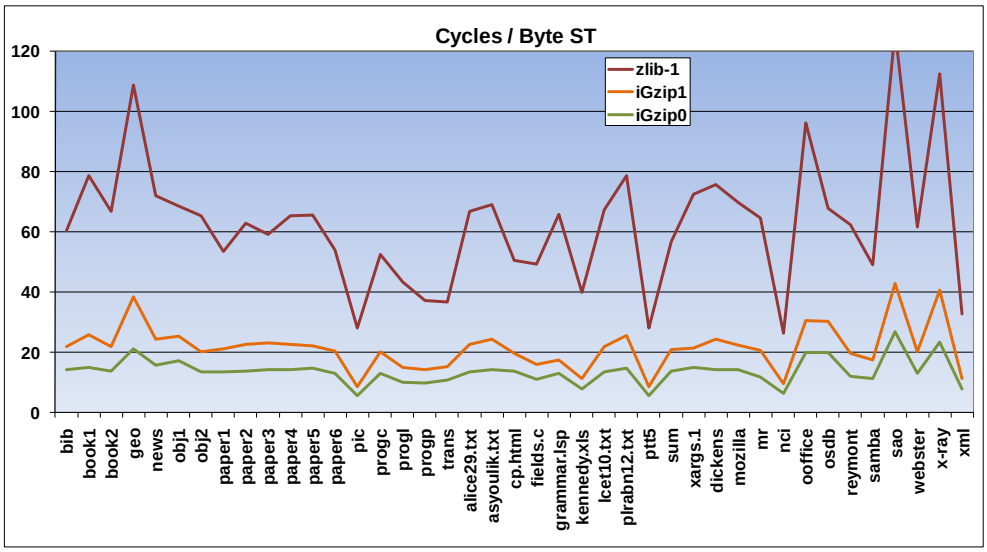
Note: Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Configurations: See previous page. For more information go to http://www.intel.com/performance.

Results

We show the compression performance across the various corpora, for all the compressors. We show performance with a single thread (ST) as well as with Intel® Hyper-Threading Technology (Intel® HT Technology).

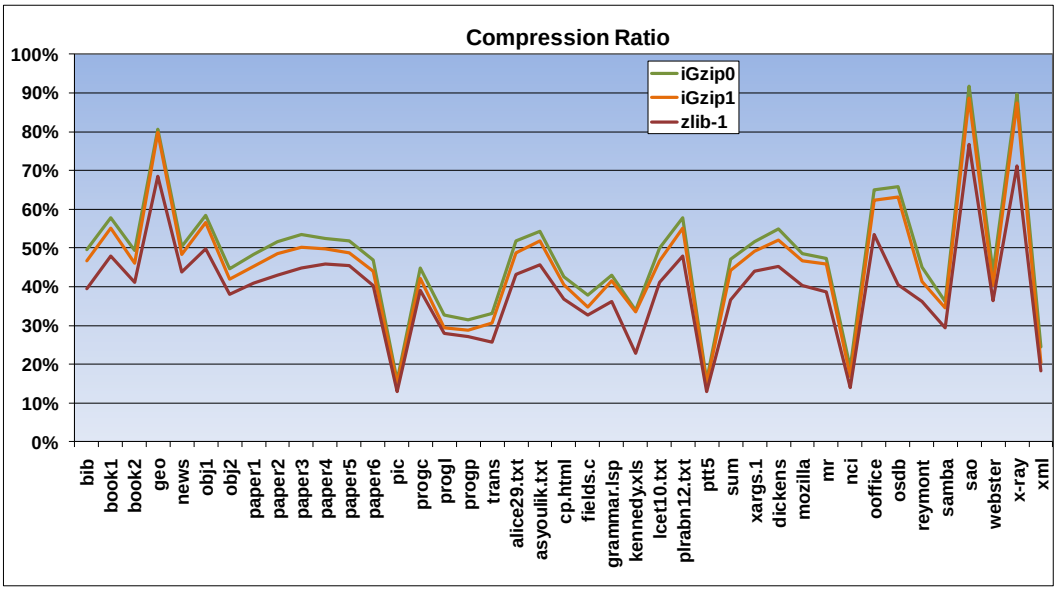
Figure 1: Performance (Cycles/Byte) of igzip/Zlib Compression with Single Thread[3]



[3] Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Configurations: Refer to the Performance section on page 6. For more information go to http://www.intel.com/performance.

Figure 2: Compression Ratio of igzip/Zlib Compression[4]



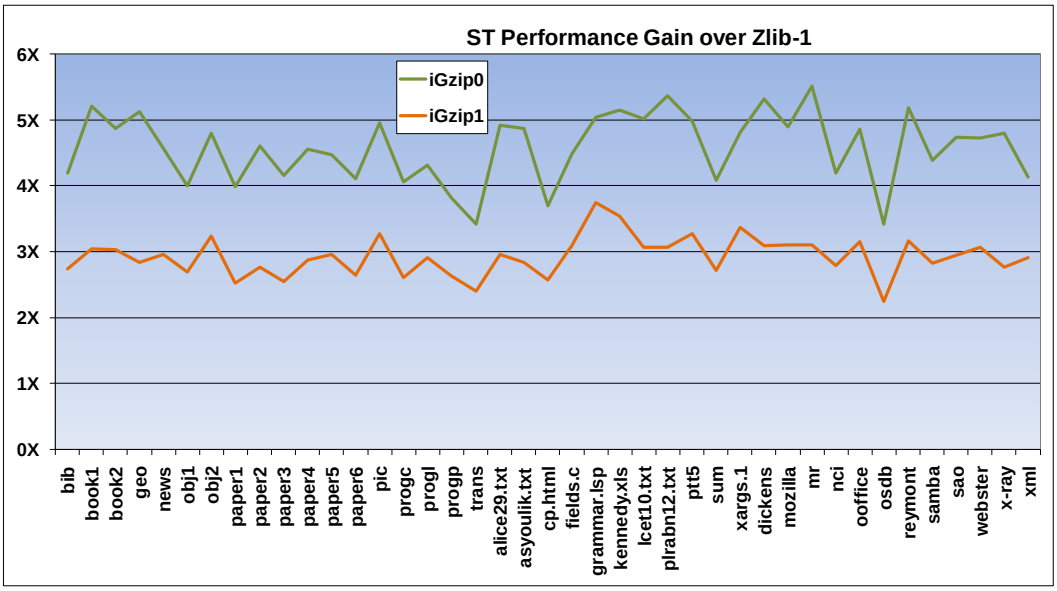
The figures show the compression performance in Cycles/byte and compression ratio, as a function of the data files in the Calgary, Canterbury and Silesia Corpora. The compression ratio is the compressed size as a percentage of the original size (i.e. lower/smaller is better).

---

[4] Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Configurations: Refer to the Performance section on page 6. For more information go to http://www.intel.com/performance.

Figure 3: Performance Gain of igzip over Zlib Compression with Single Thread[5]



Our optimized **igzip0** compression is ~**4.5 times** faster than Zlib-1 on the average without Intel® HT Technology. In terms of compression ratio, **igzip0** achieves an average of 45.5% which is a little worse than 36.6% achieved by Zlib-1.

The **igzip0** performance averages to ~**13.3 cycles/byte** on 1 core with a single thread.

We determine the averages based on summing the aggregate cycles and the aggregate bytes of the uncompressed files and computing the ratio. This method weights the average more towards bigger files. An alternate method would be to average the cycles/byte of each file, which weights all files evenly. However as both methods give close performance we chose the former method.

Excellent additional performance scaling can be achieved with Intel® HT Technology for **igzip** and Zlib compression. By performance scaling, we refer to how much additional performance is gained by using Intel® HT Technology

---

[5] Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Configurations: Refer to the Performance section on page 6. For more information go to http://www.intel.com/performance.

over a single thread for a given compressor on a single core. We show the detailed performance and compression ratios of the compressors on a single Corpus, with and without Intel® HT Technology. For brevity we also omit the details for **igzip1**.

Figure 4: Detailed Single-Thread Performance of Compression on Calgary Corpus[6]

		iGzip0				zlib -1			
File	Bytes In	Bytes Out	Comp ratio%	Cycles	Cycles /Byte	Bytes Out	Comp ratio%	Cycles	Cycles /Byte
bib	111,261	55,211	50%	1,600,108	14.4	44,021	39.6%	6,715,340	60.36
book1	768,771	443,775	58%	11,618,453	15.1	367,808	47.8%	60,536,273	78.74
book2	610,856	301,680	49%	8,373,321	13.7	250,628	41.0%	40,805,601	66.80
geo	102,400	82,563	81%	2,176,377	21.3	70,144	68.5%	11,152,660	108.91
news	377,109	189,632	50%	5,950,743	15.8	164,926	43.7%	27,195,119	72.11
obj1	21,504	12,561	58%	368,513	17.1	10,697	49.7%	1,474,180	68.55
obj2	246,814	110,326	45%	3,364,340	13.6	93,788	38.0%	16,156,057	65.46
paper1	53,161	25,702	48%	713,282	13.4	21,707	40.8%	2,845,157	53.52
paper2	82,199	42,407	52%	1,125,600	13.7	35,246	42.9%	5,178,816	63.00
paper3	46,526	24,834	53%	664,790	14.3	20,886	44.9%	2,760,057	59.32
paper4	13,286	6,956	52%	190,841	14.4	6,089	45.8%	870,262	65.50
paper5	11,954	6,206	52%	175,171	14.7	5,426	45.4%	784,232	65.60
paper6	38,105	17,887	47%	499,852	13.1	15,329	40.2%	2,055,806	53.95
pic	513,216	80,739	16%	2,906,618	5.7	65,949	12.9%	14,426,111	28.11
progc	39,611	17,748	45%	512,181	12.9	15,447	39.0%	2,078,846	52.48
progl	71,646	23,414	33%	720,913	10.1	20,046	28.0%	3,111,257	43.43
progp	49,379	15,490	31%	483,127	9.8	13,397	27.1%	1,842,008	37.30
trans	93,695	30,932	33%	1,003,987	10.7	24,004	25.6%	3,435,736	36.67

Figure 4 shows the compression ratios, total cycles to compress and the relative gain of **igzip0** over Zlib compression without Intel® HT Technology on the Calgary Corpus. The **igzip0** single-thread performance averages to ~**13.1 cycles/byte** on 1 core on the Calgary Corpus.

---

[6] Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Configurations: Refer to the Performance section on page 6. For more information go to http://www.intel.com/performance.

Figure 5: Detailed HT Performance of Compression on Calgary Corpus[7]

		iGzip0				zlib -1			
File	Bytes In	Bytes Out	Comp ratio%	Cycles	Cycles /Byte	Bytes Out	Comp ratio%	Cycles	Cycles /Byte
bib	111,261	55,211	50%	1,164,100	10.5	44,021	39.6%	5,014,278	45.07
book1	768,771	443,775	58%	8,677,439	11.3	367,808	47.8%	45,237,231	58.84
book2	610,856	301,680	49%	6,171,021	10.1	250,628	41.0%	30,297,886	49.60
geo	102,400	82,563	81%	1,624,338	15.9	70,144	68.5%	8,814,822	86.08
news	377,109	189,632	50%	4,276,990	11.3	164,926	43.7%	20,094,595	53.29
obj1	21,504	12,561	58%	263,361	12.2	10,697	49.7%	1,115,960	51.90
obj2	246,814	110,326	45%	2,400,836	9.7	93,788	38.0%	11,816,092	47.87
paper1	53,161	25,702	48%	520,365	9.8	21,707	40.8%	2,191,468	41.22
paper2	82,199	42,407	52%	837,880	10.2	35,246	42.9%	3,925,489	47.76
paper3	46,526	24,834	53%	491,144	10.6	20,886	44.9%	2,134,867	45.89
paper4	13,286	6,956	52%	139,296	10.5	6,089	45.8%	659,163	49.61
paper5	11,954	6,206	52%	127,178	10.6	5,426	45.4%	592,875	49.60
paper6	38,105	17,887	47%	368,014	9.7	15,329	40.2%	1,575,563	41.35
pic	513,216	80,739	16%	2,117,489	4.1	65,949	12.9%	10,792,283	21.03
progc	39,611	17,748	45%	371,530	9.4	15,447	39.0%	1,578,017	39.84
progl	71,646	23,414	33%	522,373	7.3	20,046	28.0%	2,326,970	32.48
progp	49,379	15,490	31%	351,358	7.1	13,397	27.1%	1,396,464	28.28
trans	93,695	30,932	33%	720,644	7.7	24,004	25.6%	2,585,046	27.59

The **igzip0** performance with Intel® HT Technology averages to ~**9.6 cycles/byte** on 1 core on the Calgary Corpus. Intel® HT Technology brings a performance gain of 1.36 times on **igzip0**. Similar performance scaling is observed for Zlib-1, which is 1.34 times faster with Intel® HT Technology.[7]

Our optimized **igzip0** compression is ~**4.5 times** faster than Zlib-1 on the average. In terms of compression ratio, **igzip0** achieves an average of 45.8% which is a little worse than 38.3% achieved by Zlib-1. [7]

---

[7] Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Configurations: Refer to the Performance section on page 6. For more information go to http://www.intel.com/performance.

Conclusion

The paper describes the performance of the best-known implementations of DEFLATE Compression, **igzip**, on an Intel® Core™ i5 processor 650 at a frequency of 3.20 GHz. We are able to perform compression at ~ **9.6 cycles/byte** on a single core with Intel® HT Technology, on the Calgary Corpus. In terms of throughput, a **single core** with Intel® HT Technology can perform compression on the Calgary corpus at the aggregate rate of ~**2.7 Gigabits/sec**.[8]

It is possible to get further performance gains for specific usages that target very small data buffers, work only on fixed sized buffers or have very different compression ratios. Such optimizations have not been attempted in the current versions of our code.

The Intel® Embedded Design Center provides qualified developers with web-based access to technical resources. Access Intel Confidential design materials, step-by step guidance, application reference solutions, training, Intel's tool loaner program, and connect with an e-help desk and the embedded community. Design Fast. Design Smart. Get started today. http://intel.com/embedded/edc.

Contributors

We thank Kirk Yap, Sean Gulley, Chengda Yang and Martin Dixon for their substantial contributions to this work.

References

[1] DEFLATE Compressed Data Format Specification - RFC1951: http://www.faqs.org/rfcs/rfc1951.html

[8] Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Configurations: Refer to the Performance section on page 6. For more information go to http://www.intel.com/performance.

[2] Huffman, D. A., "A Method for the Construction of Minimum Redundancy Codes", Proceedings of the Institute of Radio Engineers, September 1952, Volume 40, Number 9, pp. 1098-1101.

[3] Ziv J., Lempel A., "A Universal Algorithm for Sequential Data Compression", IEEE Transactions on Information Theory, Vol. 23, No. 3, pp. 337-343.

[4] Fast CRC Computation for Generic Polynomials using PCLMULQDQ Instruction http://download.intel.com/design/intarch/papers/323102.pdf

[5] RFC1952 - GZIP file format specification version 4.3 http://www.faqs.org/rfcs/rfc1952.html

[6] RFC1950 - ZLIB Compressed Data Format Specification version 3.3 http://www.faqs.org/rfcs/rfc1950.html

[7] Calgary Corpus http://www.data-compression.info/Corpora/CalgaryCorpus/index.htm

[8] Canterbury Corpus http://corpus.canterbury.ac.nz/descriptions/

[9] Silesia Corpus http://www.data-compression.info/Corpora/SilesiaCorpus/index.htm

Authors

Vinodh Gopal, Jim Guilford, Erdinc Ozturk, Gil Wolrich and **Wajdi Feghali** are IA Architects with the IAG Group at Intel Corporation.

Acronyms

IA	Intel® Architecture
ILP	Instruction level parallelism
SIMD	Single Instruction Multiple Data
SSE	Streaming SIMD Extensions