CASE STUDY

intel Software

Speeding MD5 Image Identification by 2x

Intel® Integrated Performance Primitives

High-Performance Computing

Tencent 腾讯

TenCent Optimizes Image Identification

Tencent, Inc., is China's largest and most-used Internet service portal. It owns both the largest online game community and the largest Web portal (qq.com), as well as the No. 1 and No. 2 applications (WeChat*, QQ*) in China.

Every day, Tencent needs to process billions of new user-generated images from WeChat, QQ, and QQ Album*. Some hot applications even have hundreds of millions of images to be uploaded, stored, processed, and downloaded in a single day—which consumes vast computing resources.

To manage, store, and process these images, Tencent developed Tencent File System* (TFS*). But even with compression, the image volume reached hundreds of petabytes. Moreover, it is still growing explosively—and the supported cluster has more than 20,000 servers.

Technical Background

Based on TFS, the image processing system provides uploading, scaling, encoding, and downloading services. As an image uploads, TFS scales it into a different resolution and creates the related ID by Message Digest Algorithm 5 (MD5).[1] Next, the image is transcoded into WebP* format for storage. While downloading an image, the system must find the right place to read the image, and then transcode it into the user-required image format and resolution (Figure 1).

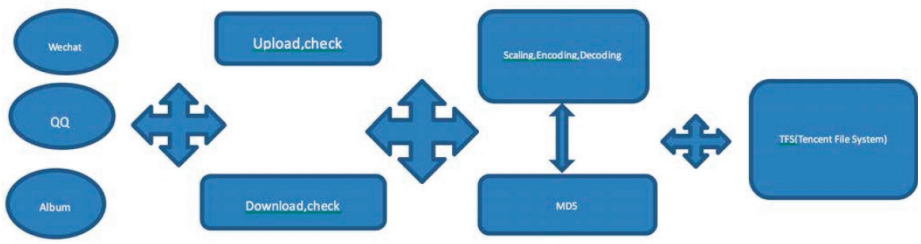

Figure 1. TenCent File System* Image Processing

Higher Performance and More Efficient Data Management Across a Wide Range of Applications

"Through close collaboration with Intel engineers, we adopted the Intel® Integrated Performance Primitives library for the image identification component in our online image storage and processing application. The application's performance improved significantly, and our cost of operations reduced greatly. We really appreciate the collaboration with Intel and are looking forward to more collaboration."

—Nicholas,
Leader of the TFS-Based Image Storage and Processing Team, TenCent

Because the website has tons of visits each second, there's a small possibility that the image download component will read the wrong image. Avoiding this kind of error requires an MD5 calculation and check. However, this is a huge computing workload—so Tencent needed to maximize MD5 computing performance.

Originally, Tencent used the md5sum* utility tool along with the Operator* OS to compute the MD5 value for each image file. Intel worked closely with Tencent engineers to help them optimize performance with Intel® Integrated Performance Primitives (Intel® IPP)—which helped Tencent achieve a 100 percent performance improvement on the Intel® architecture-based platform.

Intel® Streaming SIMD Extensions and Software Optimization

Intel introduced an instruction set extension with the Intel® Pentium® III processor called Intel® Streaming SIMD Extensions (Intel® SSE). This was a major redesign of an earlier single-instruction, multiple-data (SIMD) instruction set called MMX®, introduced with the Intel Pentium processor.

Intel evolved the Intel SSE instruction set along with Intel architecture, extending it by wider vectors and adding a new extensible syntax and rich functionality. The latest SIMD instruction set, Intel® Advanced Vector Extensions 2 (Intel® AVX2), can be found in the Intel® Core™ i7 processor.

Most of the Intel® Xeon® processors in the TFS system support Intel SSE2, one of the Intel® SIMD processor supplementary instruction sets. Intel SSE2 is supplemented by Intel SSE3, Intel SSE4.x, and Intel Advanced Vector Extensions (Intel AVX).

Intel AVX is a 256-bit instruction set extension to Intel SSE, designed to provide even higher performance for applications that are compute-intensive. Intel AVX adds new functionality to the Intel SIMD instruction set (based on Intel SSE) on floating-point and integer computing, and it includes a more compact SIMD instruction set.

Figure 2 shows one SIMD operation on eight data (32-bit integer type, floating point type) instructions.

Intel AVX improves performance by extending the breadth of vector processing capability across floating-point and integer data domains. This results in higher performance and more efficient data management across a wide range of applications such as image and audio/video processing, scientific simulations, financial analytics, and 3D modeling and analysis.

Algorithms That Benefit from Intel SSE

Algorithms that can benefit from Intel SSE[2] include those that employ logical or mathematical operations on data sets larger than a single 32-bit or 64-bit word. Intel SSE uses vector instructions, or SIMD architecture, to complete operations such as bitwise XOR, integer or floating-point multiply-and-accumulate, and scaling in a single clock cycle for mul-

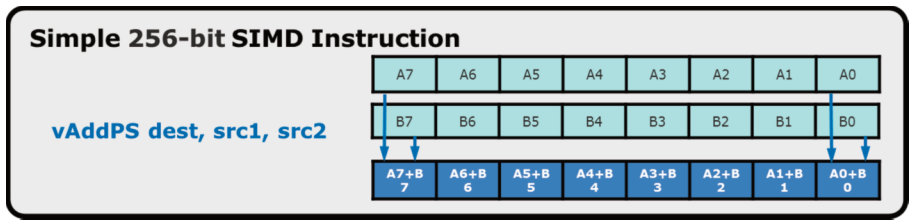

Figure 2. SIMD Operation on 8 Data Instructions

Table 1. Intel® IPP Features

Optimized for Performance and Power Efficiency	Intel Engineered and Future-Proofed to Shorten Development Time	Wide Range of Cross-Platform and OS Functionalities
• Highly tuned routines • Highly optimized using SSSE4, SSSE3, Intel® SSE, and Intel® AVX2, Intel® AVX12 instruction sets • Performance beyond what an optimize compiler produces alone	• Fully optimized for current and past processors • Saves development, debug, and maintenance time • Code once now, receive future optimizations later	• Thousands of highly optimized signal, data, and media functions • Broad domain support • Supports Intel® Quark™, Intel® Core™, Intel® Xeon®, and Intel® Xeon Phi™ platforms

Table 2. Processor-Specific Codes

Associated with Power-Specific Libraries		
px	mx	Generic code optimized for processors with Intel® Streaming SIMD Extensions (Intel® SSE)
W7		Optimized for processors with Intel SSE2
	m7	Optimized for processors with Intel SSE3
v8	u8	Optimized for processors with Supplemental Streaming SIMD Extensions 3 (SSSE3), including Intel® Atom™ processor
P8	y8	Optimized for processors with Intel SSE4.1
g9	e9	Optimized for processors with Intel® Advanced Vector Extensions (Intel® AVX) and Intel® Advanced Encryption Standard New Instructions
h9	i9	Optimized for processors with Intel AVX2

tiple 32-bit or 64-bit words. Speed-up comes from the parallel operation and the size of the vector (multiword data) to which each mathematical or logical operator is applied.

Examples of algorithms that can significantly benefit from SIMD vector instructions include:

- **Image processing and graphics.** Both scale in terms of resolution (pixels per unit area) and the pixel encoding (bits per pixel to represent intensity and color) and both benefit from speedup relative to processing frame rates.
- **Digital signal processing (DSP).** Samples digitized from sensors and instrumentation have resolution like images as well as data acquisition rates. Often, a time series of digitized data that is one-dimensional will still be transformed using algorithms, like a DFT (Discrete Fourier Transform) that operate over a large number of time series samples.
- **Digest, hashing, and encoding.** Algorithms used for security, data corruption protection, and data loss protection such as simple parity, CRC (cyclic redundancy check), MD5, SHA (secure hash algorithm), Galois math, Reed-Solomon encoding, and CBC (cypher-block-chaining) all make use of logical and mathematical operators over blocks of data, often many kilobytes in size.
- **Data transformation and data compression.** Most often, simulations in engineering and scientific computing involve data transformation over time and can include grids of data that are transformed. For example, in physical thermodynamic, mechanical, fluid-dynamic, or electrical-field models, a grid of floating-point values is used to represent the physical fields as finite elements. These finite element grids are then updated through mathematical transformations over time to simulate a physical process.

Optimized for Performance

Intel IPP is a performance building block for all kinds of image and signal processing, data compression, and cryptography needs. These ready-to-use, royalty-free functions are highly optimized using Intel SSE and Intel AVX and Intel AVX2 instruction sets, which often outperform what an optimized compiler can produce alone.[3]

Table 1 summarizes the features of Intel IPP.

The Intel IPP library is optimized for a variety of SIMD instruction sets. Besides the optimization, Intel IPP also provides an automatic “dispatching” mechanism, which can detect the SIMD instruction set that is available on the running processor and select the optimal SIMD instructions for that processor.

Table 2 shows processor-specific codes that Intel IPP uses.

See Understanding CPU Dispatching in the Intel® IPP Library for more information on dispatching. For more information on Intel IPP functions optimized for Intel AVX, read the article Intel® IPP Functions Optimized for Intel® AVX.

MD5 in Intel IPP

Hash functions are used in cryptography with digital signatures and for ensuring data integrity. When used with digital signatures, a publicly available function hashes the message and signs the resulting hash value. The party that receives the message can then hash the message and check if the block size is authentic for the given hash value.

Hash functions are also referred to as "message digests" and "one-way encryption functions." To ensure data integrity, hash functions are used to compute the hash value that corresponds to a particular input. Then, if necessary, you can check if the input data has remained unmodified. You can recompute the hash value again using the available input and compare it to the original hash value. Intel IPP has implemented the following hash algorithms for streaming messages:

- MD5 [RFC 1321]
- SHA-1
- SHA-224
- SHA-256
- SHA-384
- SHA-512 [FIPS PUB 180-2]

These algorithms are widely used in enterprise applications.

A Closer Look at MD5

MD5 is a widely used cryptographic hash function producing a 128-bit (16-byte) hash value, typically expressed in text format as a 32-digit hexadecimal number.

Although MD5 was considered as "cryptographically broken and unsuitable" in a strict environment, it has been widely used in the software world to provide some assurance that a transferred file has arrived intact. For example, file servers often provide a precomputed MD5 (known as md5sum) checksum for the files, so that a user can compare the checksum of the downloaded file to it.

Most Linux*-based operating systems include md5sum utilities in their distribution packages.

The Intel IPP MD5 functions apply hash algorithms to digesting streaming messages. It uses a state context (for example, ippsSHA1State) as an operational vehicle to carry all necessary variables to manage the computation of the chaining digest value. For example, the primitive implementing the MD5 hash algorithm must use the ippsMD5State context. The function Init initializes (MD5Init) the context and sets up specified initialization vectors. Once initialized, the function Update (MD5Update) digests the input message stream with the selected hash algorithm until it exhausts all message blocks. The function Final (MD5Final) is designed to pad the partial message block into a final message block with the specified padding scheme. It then uses the hash algorithm to transform the final block into a message digest value.

Here is an example illustrating how the application code can apply the implemented MD5 hash standard to digest the input message stream:

1. Call the function MD5GetSize to get the size required to configure the ipps MD5State context.
2. Ensure that the required memory space is properly allocated. With the allocated memory, call the MD5Init function to set up the initial context state with the MD5-specified initialization vectors.
3. Keep calling the function MD5Update to digest the incoming message stream in the queue until its completion. To determine the current value of the digest, call MD5GetTag between the two calls to MD5Update.
4. Call the function MD5Final for padding the partial block into a final MD5-1 message block and transform it into a 160-bit message digest value.
5. Clean up secret data stored in the context.
6. Call the operating system memory free service function to release the ippsMD5State context.

Intel engineers optimized Intel IPP functions mainly by the vectorization, or SSE instruction, and by extracting Intel architecture such as cache utilization, registers reutilization, etc. With respect to Intel IPP MD5 implementation, the optimized technique is used:

- Fully unrolled code instead of tiny loop
- Using cyclic registers permutation instead of memory operations

Table 3. Code Example

md5sum code	IPP MD5

md5sum performance.PNG > hash.md5
cat hash.md5
0e5e74555b68db366c85b1b194f258fe performance.PNG.
 	
The md5sum is along with Cent OS distribution. The source code can be download from
	char* intel_md5sum(const char *p, unsigned uiLen)
	{
	static Ipp8u MD[32];
	int size;
	IppsMD5State *ctx;
	ippsMD5GetSize(&size);
	ctx = (IppsMD5State*)malloc(size);
	IppStatus st = ippsMD5Init(ctx);
	st = ippsMD5Update((const Ipp8u *)p, (int)uiLen, ctx);
	st = ippsMD5Final(MD, ctx);
	free(ctx);
	return (char*)MD;
	}

- Coding rotations immediately instead of general parameterized 32-bit rotation

The Intel IPP code replaced the md5sum. With the code shown in Table 3, no more manual optimization was needed.

Invoking Intel IPP to Accelerate MD5

The Intel IPP MD5 code, `md5test.cpp`, is compiled using gcc as follows:

```
[root@localhost code]# make -f Makefile.gcc

g++ -O2 ipp_md5.cpp -o ipp_md5

-I/opt/intel/compilers_and_libraries_2016.0.109/linux/ipp/include

/opt/intel/compilers_and_libraries_2016.0.109/linux/ipp/lib/intel64/

libippcp.a /opt/intel/compilers_and_libraries_2016.0.109/linux/ipp/lib/

intel64/libippcore.a
```

This integrates the IPP crypto library into the program and extracts performance from the computing resources automatically. Figure 3 is a screen shot of Intel® VTune™ Amplifier XE running the ipp_md5 program. It shows the ipp function e9_ippsMD5Update takes most of the CPU time of the program where e9 (AVX-optimized) code was running.

Performance Data

The test was run based on different sizes of image files using Intel IPP and md5sum provided by the Linux OS. Using 10,000 iterations resulted in the performance shown in Table 4 and Figure 4.

On the Intel® Xeon® processor E5-2620 (15M Cache, 2.00 GHz, 7.20 GT/s Intel® QuickPath Interconnect, Intel AVX-supported), comparing the md5sum along with Linux showed a 100 percent performance improvement. Tencent engineers also implemented Intel IPP MD5 for their online system. Their test showed about a 60 percent performance improvement compared to the original MD5.

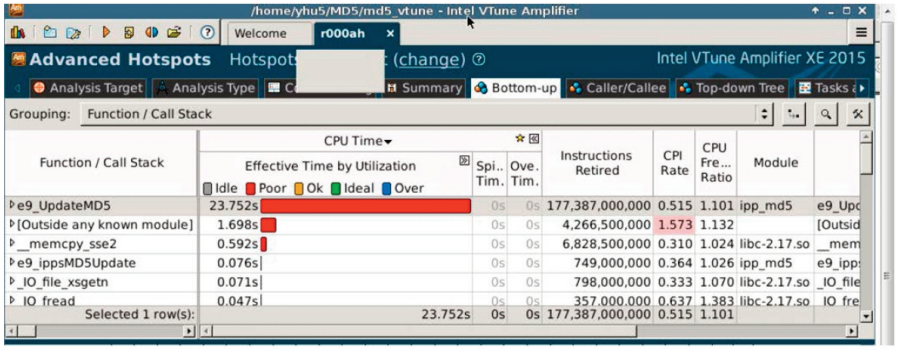

Figure 3. Intel® vTune™ Amplifier running the ipp_md5 program

Table 4. Test Run Performance Results

Processor OS	Test Image Size	md5sum (Average Time for 10,000 Iterations)	IPP_Md5 (Average Time for 10,000 Iterations)
Intel® Xeon® processor E5-2620 (15M cache, 2.0 GHz, 7.20 GT/s, Intel® QuickPath Interconnect, Intel® Advanced Vector Extensions-supported CentOS 6.5)i9	4K 8K 16K 32K 64K 128K	206 ms 406 ms 789 ms 1,574 ms 2,420 ms 6,273 ms	95 ms 189 ms 369 ms 740 ms 1,183 ms 2,943 ms

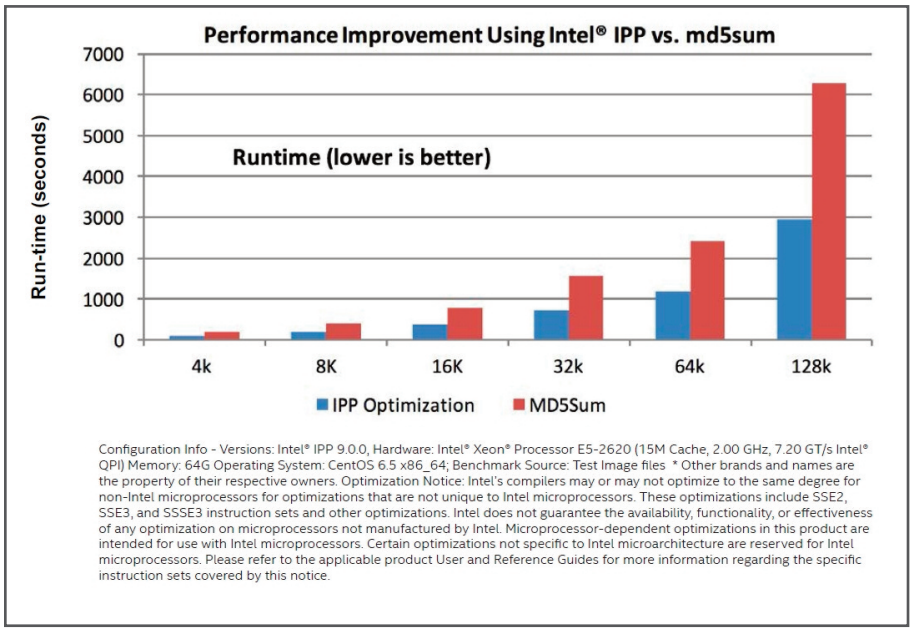

Figure 4. Test run performance

Conclusion

Tencent has billions of new user-generated images to process every day from WeChat, QQ, and QQ Album. All images are handled by the TFS-based image storage and processing system. Tencent has to give each image a unique ID by MD5 hash. Intel worked with Tencent engineers to optimize this function component using Intel IPP, achieving a 2x performance improvement.

Methods for improving the speed of computing the md5sum of images is straightforward with Intel IPP. This work demonstrates significant progress toward being able to handle these computationally intensive methods by optimizing them for the latest Intel® hardware using the Intel IPP and performance-tuning methodologies.

.Learn more about Intel® Integrated Performance Primitives

[1]md5sum on Wikipedia.

[2]Using Intel® Streaming SIMD Extensions and Intel® Integrated Performance Primitives to Accelerate Algorithms.

[3]Intel® AVX Realization of IIR Filter for Complex Float Data.