



Solution Brief

libNthumb, the NHN* performance primitive for real-time creation of thumbnail image with Intel® IPP Library

Sungwhan Jung, NHN Corporation
Tai Ha, Intel® Corporation
Chao Yu, Intel® Corporation

August, 2010

NHN Overview

NHN Corporation is Korea's premier Internet company, operating the nation's top search portal, Naver (www.naver.com), the leading online game portal, Hangame (www.hangame.com) the nation's largest children's portal, Jr.Naver (jr.naver.com) and the first online donation portal, Happybean (happybean.naver.com)

Starting from the business pillars of search and games, NHN has rolled out a wide range of innovative and convenient online services. A number of surveys demonstrate that the company is regarded as an undisputed leader in the online services industry worldwide.

NHN has emerged as Korea's largest Internet Company in terms of net profit. This outstanding growth is largely attributable to the company's efforts to create and introduce stable revenue streams in its core services; for instance, search-oriented advertising services and fee-based games.

NHN outgrew the confines of Korea's borders. NHN built a good reputation in Japan and China, and also set up NHN USA, forging towards becoming a world-class internet company.

Project Overview

Internet portal sites such as Naver generally use the thumbnail image, which is a reduced-size version of an original image, throughout their services. Naver has three processes to create a thumbnail image; decoding original images, resizing, and encoding to a thumbnail image. In this process, the ImageMagick*, which is famous for Linux*-based image process library, is broadly used.

The latest CPU provides the newest Streaming SIMD Extensions (SSE, SSE, SSE2, SSE3, SSSE3, SSE4, SSE4.1, SSE4.2, and Intel® AVX) instruction set to accelerate the signal and multimedia data processing. However, the ImageMagick doesn't fully utilize these instruction sets. In those cases where we use the SSE instruction set during creating a thumbnail image, we can improve the performance.

The NHN Performance Engineering team develops the thumbnail creation library "libNthumb", which uses the SIMD instruction set through Intel® Integrated Performance Primitives (Intel® IPP). Intel® IPP provides optimized software building blocks for multimedia, data and image processing application. Intel® IPP is available as a component of Intel® Parallel Studio, Intel® Parallel Studio XE, and as a standalone product.

In this paper, we will show the performance benefit of libNthumb against ImageMagick through following two benchmark tests.

- Batch Process: The unit performance test of decoding, encoding and resizing
- Real-time Process: The real-time performance test of thumbnail image creation on a web-server

Sample Libraries for Benchmark Tests

1. ImageMagick

ImageMagick, which is a widely-used open source library, has functions to read, write and convert in more than 100 image formats. It provides various types of API according to the abstraction level as well as command-line tools. It is taken as the performance baseline.

2. libNthumb

libNthumb, which is the library specialized in thumbnail image creation, has the following functions.

- Fast jpeg resizing
 - Using Inverse Discrete Cosine Transform (IDCT) scale factor

- SSE instruction set via Intel® IPP
- Multi-step resizing
- Sharpen filter
- Lossless auto rotation
- Metadata removal
- Cropping with Rectangle Regions of Interest (ROI)

libNthumb uses the encoders/decoders from both Intel® IPP JPEG sample code and ImageMagick. The Intel® IPP JPEG encoder/decoder is used to adjust IDCT scale factor and the ImageMagick encoder/decoder is used to support various image formats and recover partially-damaged images.

libNthumb uses the Intel® IPP because it improves the performance of data stream operation by utilizing SIMD instruction set.

libNthumb delivers additional performance boost when creating thumbnail images from jpeg image files. A jpeg image file has several decoding steps and the IDCT process is one of them. In the IDCT process, we get a resized image by adjusting a scale factor. The resized image reduces the size of data set for next operations and it improves the performance.

3. libNthumbIppOnly

This is a variant of the libNthumb that removes IDCT scale factor feature out of the jpeg resizing feature. The reason we use this library in the benchmark tests is to measure performance gains that only Intel® IPP contributes to. The performance difference between ImageMagick and this library shows the performance gain only through Intel® IPP. Similarly, the performance difference between this library and libNthumb shows the performance gain only through IDCT scale factor.

Test Environment

The followings describe test environment for both H/W and S/W.

System	Qty	CPU	Mem	Linux Kernel
SUT	1	L5420 2.56GHz	16GB	2.6.18-164.11. 1.el5x86_64
Load Generator	3	3060 2.4GHz	4GB	2.6.18-164.el5 PAEi686
Console	1	3060 2.4GHz	4GB	2.6.9-78.0.22 ELsmpi686

- OS: CentOS* 5.3
- Libraries
 - ImageMagick 6.5.9-3
 - Intel® IPP 6.1.2.051
- Benchmark tools for Batch Process
 - Multi-threaded brew tool
- Benchmark tools for Real-time Process
 - Grinder* 3.2 (Load Generator)
 - Nginx* 0.7.65 (Web Server Container)

Benchmark: Batch Process

1. Workload

The benchmark repeatedly runs a transaction and the transaction consists of decoding, resizing and encoding for the below image.

The test data is 12 Mega-pixel JPEG Image (4000 x 3000, 5.5 MB). It will be resized to a 400 x 300 thumbnail image.



For the scalability test, benchmark results are measured for different workloads. The workload is controlled by the number of worker threads; 1, 2, 4, or 8. Each worker thread runs ten transactions.

2. Performance Metrics

There are 6 performance metrics in this benchmark as follows.

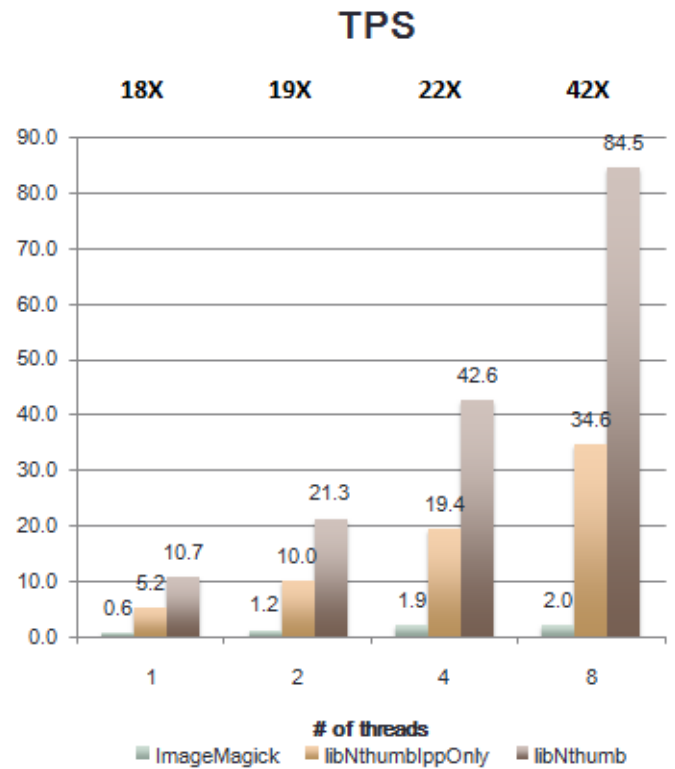
- Transactions Per Second (TPS)
- Average Elapsed Time for Decoding
- Average Elapsed Time for Resizing
- Average Elapsed Time for Encoding
- Average Elapsed Time for Transactions
- Scalability

Processor utilization is not selected as a performance metric because there is no much difference between ImageMagick and libNthumb.

3. Benchmark Result

- Transactions Per Second (TPS)

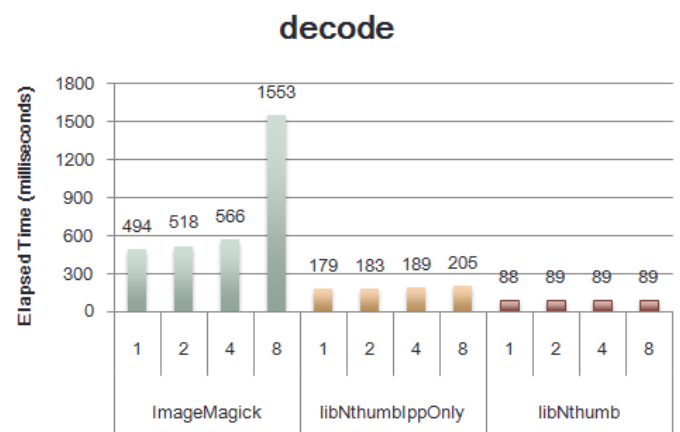
The figure below shows average TPS according to the number of thread during the average elapsed time.



libNthumb shows 42X performance gain over ImageMagick at maximum in direct proportion to the number of thread.

- Average Elapsed Time for Decoding

The figure below shows average elapsed time for the decoding process.

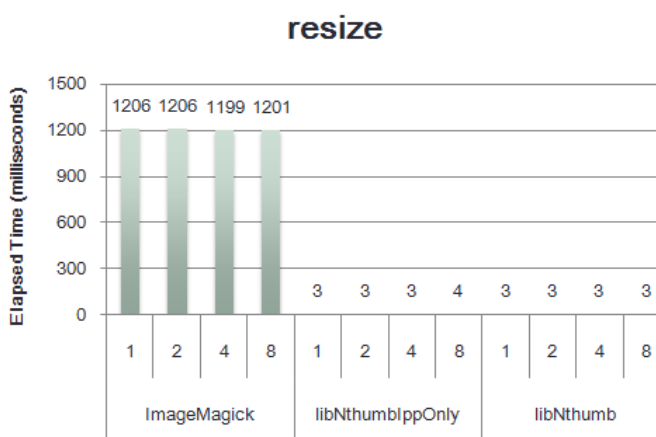


When the number of worker threads is 1, libNthumb shows 5.6X performance gain over ImageMagick. libNthumb also shows 17X performance gains when the number of work threads is 8.

The decoding process consists of reading image file from a disk and decoding it into byte stream. This kind of process doesn't have dependency between worker threads and can avoid parallelization overhead as the number of worker threads increases. However, ImageMagick shows performance degradation when the number of worker threads reaches 8.

- Average Elapsed Time for Resizing

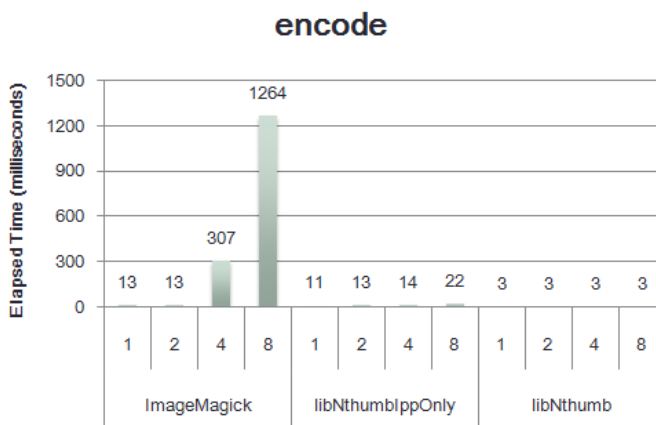
The figure below shows average elapsed time for resizing process.



Regardless of the number of worker threads, libNthumb shows about 400X performance gain over ImageMagick. It is because libNthumb has smaller data set through IDCT scale factor during decoding process.

- Average Elapsed Time for Encoding

The figure below shows average elapsed time for encoding process.

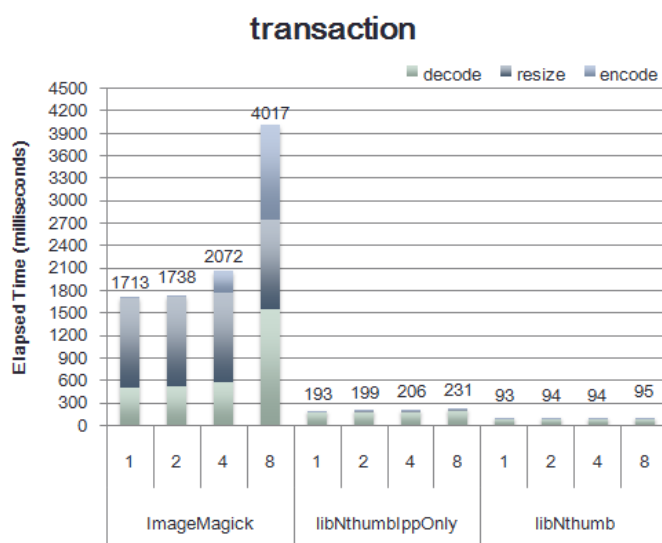


Encoding time is relatively shorter than decoding time because the data set is reduced through resizing process.

When the number of worker threads is 1 or 2, libNthumb shows 4.3X performance gain over ImageMagick. libNthumb also shows 421X performance when the number of worker threads reaches 8. It is because ImageMagick performance drops as the number of worker threads increases above 4, while libNthumb performance doesn't change.

- Average Elapsed Time for a Transaction

The figure below shows average elapsed time for a transaction.

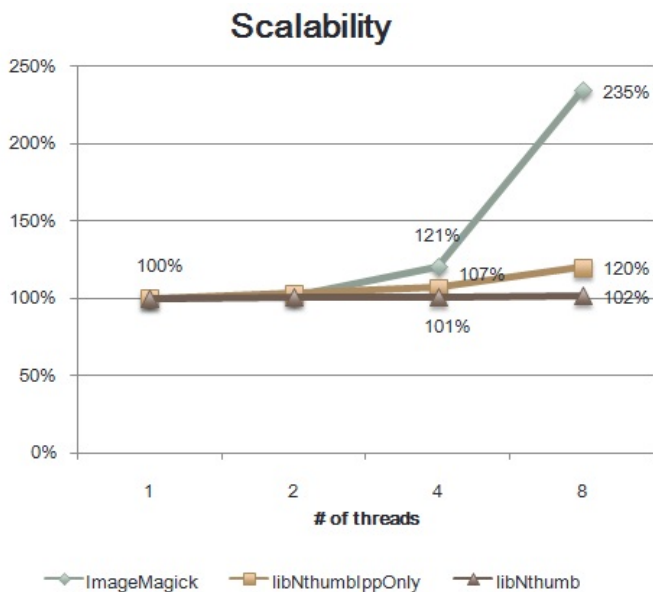


In cases where the number of worker threads is 1, libNthumb shows an 18X performance gain over ImageMagick. libNthumb also shows a 42X performance gain when the number of work threads is 8. As mentioned above, ImageMagick displays performance degradation in decoding process and encoding process as the number of worker threads increases. In the following "Scalability" section, we will mention the main cause of this scalability issue.

- Scalability

When seeing the graph titled "TPS" on page 3, libNthumb scales up almost linearly as the number of worker threads increases, while ImageMagick rarely scales.

Also, the figure below shows the scalability from the viewpoint of "Average Elapsed Time for Transactions". The graph takes the result from 1 thread as the baseline, 100%. Then, it shows the other results of 2, 4, 8 threads in increased percentages over the baseline.



libNthumb shows consistent results without performance degradation, while ImageMagick shows 2.4X performance degradation at maximum.

The main cause of the performance degradation is the lock that ImageMagick uses for thread safety. Looking into source code, "magick/semaphore.c", there is the following mutex lock.

```
static pthread_mutex_t semaphore_mutex = PTHREAD_MUTEX_INITIALIZER;
```

In the decoding and encoding process, LockSemaphoreInfo() is called to acquire the lock even though a separate image is under processing. Consequently, this lock contention causes performance degradation as the number of worker thread increases.

The compile option, "--without-threads" on building ImageMagick can be used to remove this locking mechanism. Then it becomes unsafe for multi-threading, leaving synchronization mechanism to its users.

The scalability of ImageMagick doesn't meet the requirements of NHN where lots of concurrent users demand short response times as with other Internet Portals.

Benchmark: Real-time Process

The previous section, Batch Process, focuses on performance unit test. So, a large image, 12 million pixels (4000 X 3000), is used for the benchmark test. Also it runs batch work with same operations.

However, there are lots of requests for different operations on smaller images in real world environments. In this real world environment, the performance difference between libNthumb and ImageMagick seems to get smaller.

Hence, we will talk about another benchmark, Real-time Process where a real web server is requested to call sample libraries for thumbnail image creation.

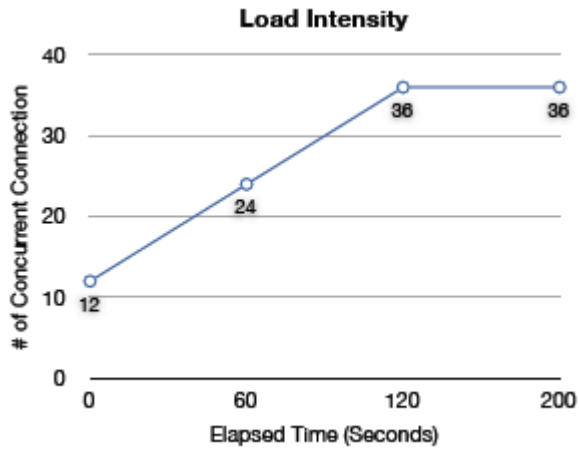
1. Workload

NHN currently uses an in-house solution as a thumbnail image creation platform. After analyzing the log files of it, only JPEG files are taken for test data set. JPEG files are 90% out of original images files in the log files. Nginx web server is used for a container to call sample libraries creating thumbnail images corresponding to original images in real-time.

One transaction is composed of the following steps

- A load generator randomly selects a JPEG image among test data set. Then, it makes a HTTP request to create a thumbnail image of the JPEG image.
- The load generator sends the HTTP request to NginX web server.
- NginX web server transfers the requests to a library module; ImageMagick, libNthumbIppOnly or libNthumb.
- The library module creates 300 x 200 thumbnail images.
- NginX web server makes a HTTP response with the thumbnail image and returns it.

The figure below shows changes of workload intensity during a benchmark test.



processing a big image file or when a load generator does process-forking to increase workload.

The figure below shows average TPS between 150 and 200 second range where concurrent connections become the 36 at maximum.

2. Performance Metrics

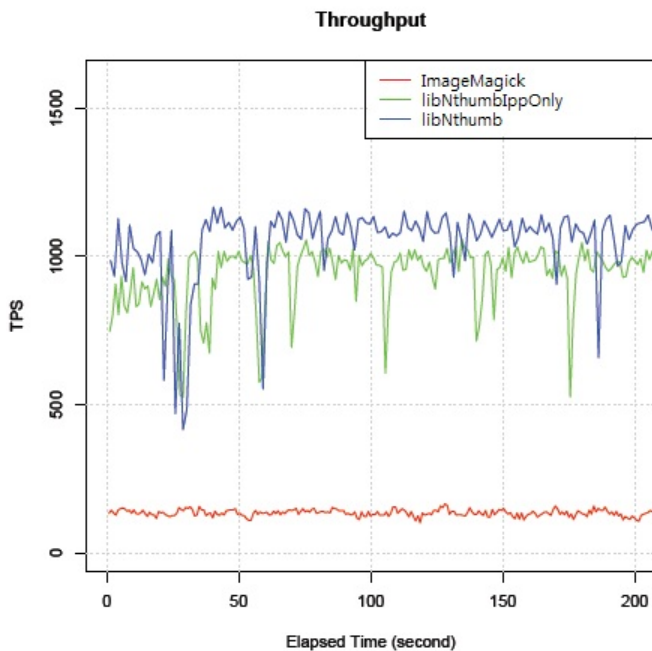
There are 3 performance metrics in this benchmark as follows.

- Transactions Per Second (TPS) /Throughput
- Average Response Time
- Processor Utilization

3. Benchmark Result

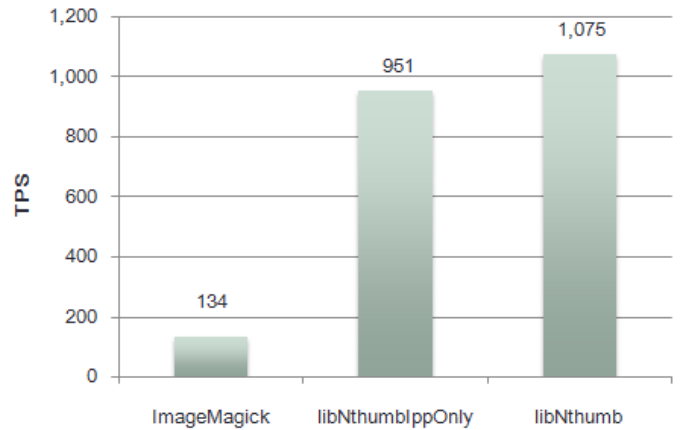
- Transactions Per Second (TPS)

The figure below shows TPS changes during a benchmark test.



Generally, libNthumb shows higher TPS than ImageMagick. There are fluctuations in the libNthumb result graph. They happen when

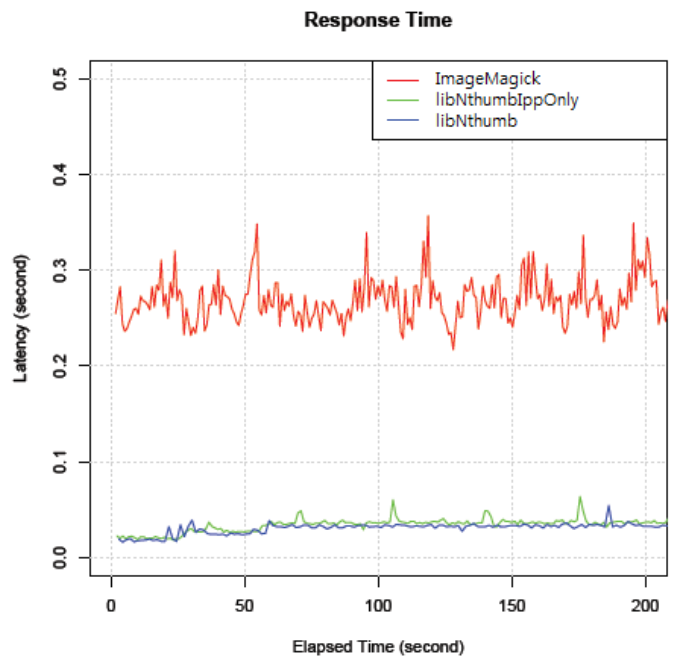
Throughput



libNthumb shows 8X performance over ImageMagick in terms of average TPS.

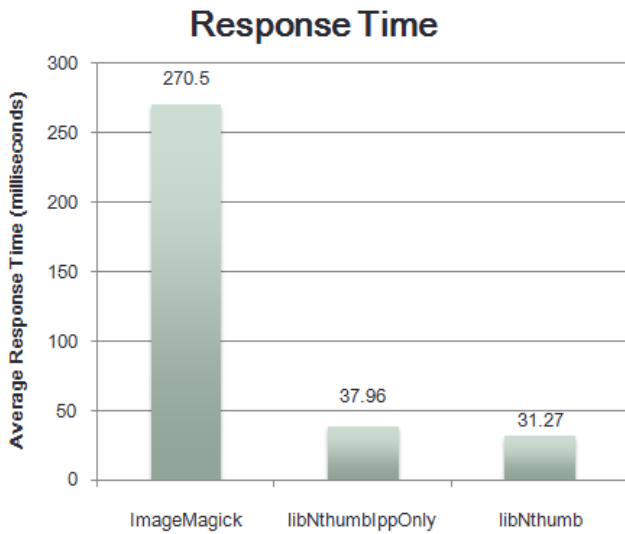
- Average Response Time

The figure below shows changes of response time during a benchmark test.



libNthumb shows shorter response time and less fluctuation than ImageMagick.

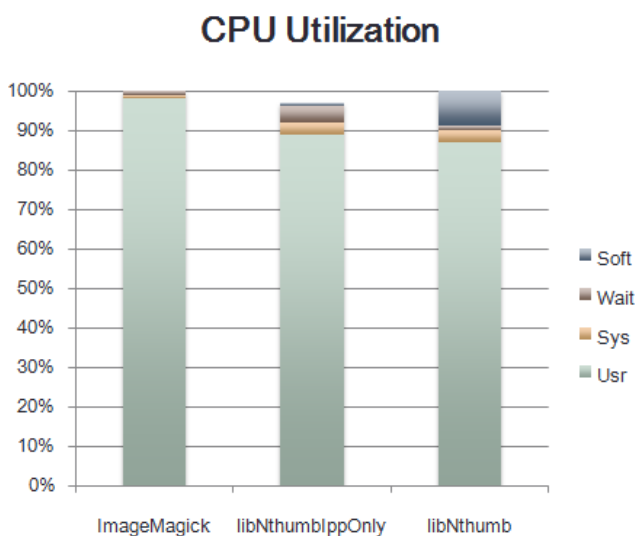
The figure below shows average response time between 150 and 200 second range



libNthumb shows 8.7X performance over ImageMagick in terms of average response time.

- Processor Utilization

The figure below shows changes of processor utilization between 150 and 200 second range.



	Usr	Sys	Wait	Soft
imk	98%	1%	1%	0%
libnthumb (ipp)	89%	3%	4%	1%
libnthumb (ipp + idct)	87%	3%	1%	9%

Both libNthumb and ImageMagick show 100% processor utilization. However, there are differences in a detailed analysis. libNthumb has higher processor utilization in the System and soft

IRQ domain, because it makes higher TPS utilizing higher network bandwidth.

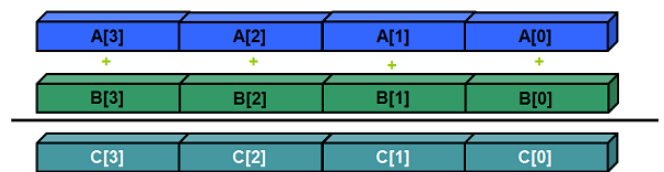
Performance Improvement Factors

As seen in previous sections, libNthumb performance is far better than ImageMagick in creating thumbnail images. There are two main factors to improve the performance of libNthumb.

- using SIMD instruction with Intel® IPP
- using IDCT scale factor

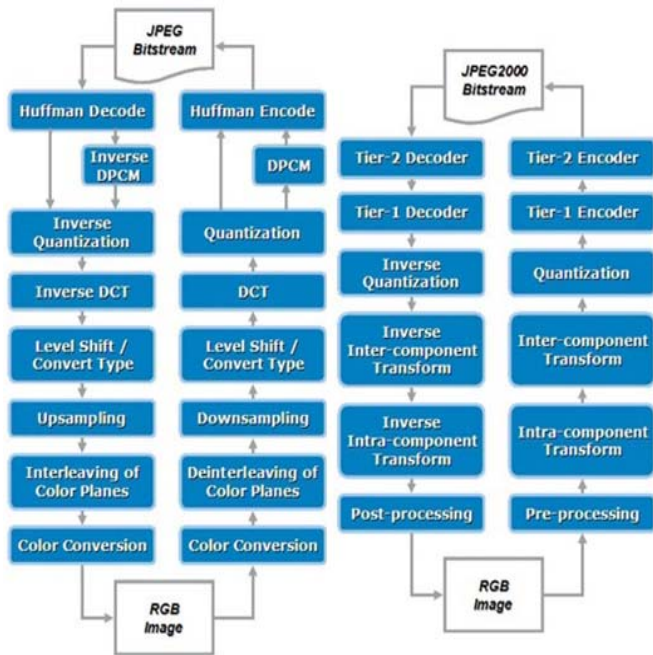
1. SIMD instructions with Intel® IPP

Intel® Streaming SIMD Extensions is a new set of Single Instruction Multiple Data (SIMD) instructions designed to improve the performance of various applications. They are available on Intel® and Intel® compatible processors. One SIMD instruction can process several data elements at the same time. For example, a SSE2 instruction can compute two 64bit integer data, or four 32bit integer data concurrently, shown below.



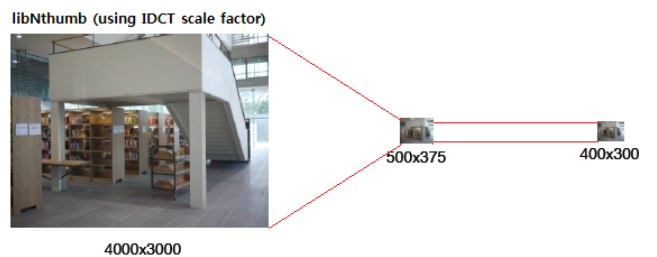
Intel® IPP functions are designed to deliver performance by matching the function algorithms to low-level optimizations based on the processor's available features such as Streaming SIMD Extensions and other optimized instruction sets.

The libNthumb uses Intel® IPP JPEG encoding sample code to complete the JPEG encoding and decoding transform. Intel® IPP has optimized the key algorithmic components for JPEG Codec, shown below.



2. IDCT scale factor

JPEG image data can be resized by adjusting block size during IDCT process. libNthumb improves the performance by utilizing this IDCT process. It scales down the image size to be as close as thumbnail image size during IDCT process. So there is reduced and decoded image before resizing process.

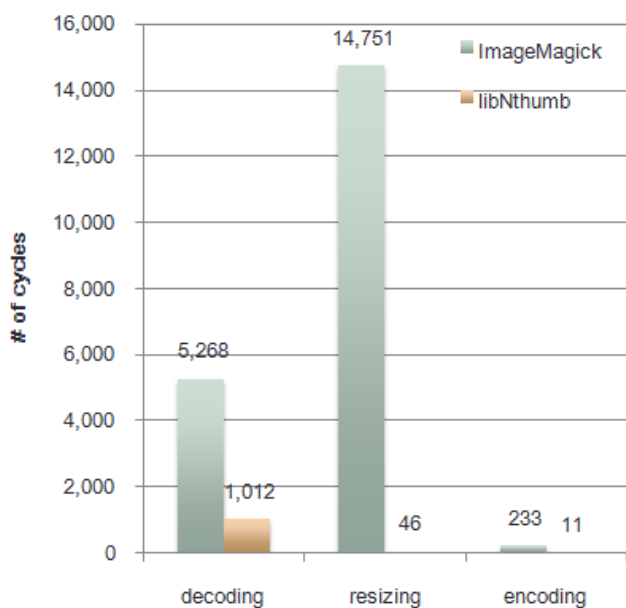


(a) original file (JPEG encoded) (b) decoded image (c) resized image

Besides the JPEG encoding and decoding function, the libNthumb takes advantage of the Intel® IPP image processing resize functions. The resizing process is the loop of calculation on image pixels. Each pixel includes 24 bit data (R,G,B each 8 bits). By utilizing Intel® IPP, which is optimized by SIMD instructions, resizing process is well performed in libNthumb.

The picture below is a result of profiling CPU cycles for each step in batch process test. We can see significant performance improvement, especially in the image resizing part. This is because of two reasons: 1) the resize function uses optimized Intel® IPP functions 2) by controlling IDCT scale factor in the JPEG decoding, libNthumb only resize a contracted image.

of cycles during operation



The picture above is a diagram for each process of JPEG resize where the JPEG image is reduced by 1/10 size.

ImageMagick does not change IDCT scale factor in decoding process. Therefore, the result image of decoding has the same size as the original image, while libNthumb obtained the result image which is reduced by 1/8 size by handling IDCT scale factor in decoding process.

Because the resizing process is repeated integer-intensive operation, the resizing process with reduced image costs less than one with full-size image.

Image Quality

The thumbnail image should have a certain level of quality. When it comes to the quality of thumbnail image from libNthumb, the quality difference from ImageMagick is invisible to the naked eye. The below pictures are thumbnail images generated by ImageMagick and libNthumb, respectively.



<Thumbnail Image by ImageMagick>



<Thumbnail Image by libNthumb>

There are various methods of resizing. Image quality will differ depending on the filter used. libNthumb improves image quality through multi-level resizing and sharpening filters, each having a different look and feel.

Conclusion

libNthumb is the performance-optimized library to focus on thumbnail image creation. It also provides additional features that are useful for thumbnail image creation in Internet portal services. For example, the auto rotation feature supports the EXIF format, which is a metadata used for image rotation in digital camera and is not supported by most web browsers. It also has metadata retention and removal features.

It improves the performance by using Intel® IPP library for utilizing SSE instructions and IDCT scale factor. In the Batch Process benchmark, libNthumb shows 42X performance improvement over ImageMagick that is widely used library today. In the Real-time Process, it also shows 8X performance improvement.

Lastly, libNthumb has better scalability than ImageMagick. libNthumb minimizes locking overhead, while ImageMagick uses big lock to support multi-threading causing less scalability.

Notices

Performance results are based on certain tests measured on specific computer systems. Any difference in system hardware, software or configuration will affect actual performance. **For more information go to <http://www.intel.com/performance>**

Copyright © 2010 Intel Corporation. All rights reserved.

Intel is a trademark of Intel Corporation in the U.S. and other countries. *Other names and brands may be claimed as the property of others.

Intel does not control or audit the design or implementation of third party benchmark data or Web sites referenced in this document. Intel encourages all of its customers to visit the referenced Web sites or others where similar performance benchmark data are reported and confirm whether the referenced benchmark data are accurate and reflect performance of systems available for purchase.

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Optimization Notice

Intel® compilers, associated libraries and associated development tools may include or utilize options that optimize for instruction sets that are available in both Intel® and non-Intel microprocessors (for example SIMD instruction sets), but do not optimize equally for non-Intel microprocessors. In addition, certain compiler options for Intel compilers, including some that are not specific to Intel micro-architecture, are reserved for Intel microprocessors. For a detailed description of Intel compiler options, including the instruction sets and specific microprocessors they implicate, please refer to the "Intel® Compiler User and Reference Guides" under "Compiler Options." Many library routines that are part of Intel® compiler products are more highly optimized for Intel microprocessors than for other microprocessors. While the compilers and libraries in Intel® compiler products offer optimizations for both Intel and Intel-compatible microprocessors, depending on the options you select, your code and other factors, you likely will get extra performance on Intel microprocessors.

Intel® compilers, associated libraries and associated development tools may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include Intel® Streaming SIMD Extensions 2 (Intel® SSE2), Intel® Streaming SIMD Extensions 3 (Intel® SSE3), and Supplemental Streaming SIMD Extensions 3 (Intel® SSSE3) instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors.

While Intel believes our compilers and libraries are excellent choices to assist in obtaining the best performance on Intel® and non-Intel microprocessors, Intel recommends that you evaluate other compilers and libraries to determine which best meet your requirements. We hope to win your business by striving to offer the best performance of any compiler or library; please let us know if you find we do not. Notice revision #20101101