CASE STUDY

# Optimizing Image Processing

## Intel® C++ Compiler, Intel® Integrated Performance Primitives

High-Performance Computing

### JD.com Speeds Image Processing 17x

As China's largest online direct sales company, JD.com handles several billion product images every day. To do this, the company developed its own distributed file system, JD File System* (JFS*). By using Intel® C++ Compiler and Intel® Integrated Performance Primitives (Intel® IPP), JD.com sped up its image processing 17x—handling 300,000 images in 162 seconds instead of 2,800 seconds.

### Business Requirements

JD.com processes more online transactions than any other company in China, with a market share of 54.3 percent in the second quarter of 2014, according to iResearch, a third-party market research firm. The company's business has grown rapidly, from offering approximately 1.5 million SKUs in 2011 to approximately 25.7 million in 2013. Today, JD.com must handle petabytes of data, which takes an efficient, robust, distributed file system.

### Image System Overview

To handle its massive data, JD.com developed JFS*, a distributed cache system and high-speed, key-value storage system that formed a solid foundation for its fast-growing e-commerce business. As a distributed storage system designed for excellent performance, reliability, and scalability, JFS provides three types of interfaces: binary large object storage, file system storage, and block storage. JFS supports many of JD.com's core services including a public/private cloud, an image system, a logistics exchange platform, and instant messaging file sharing storage. Figure 1 shows the overall architecture.

Product images might be the single most important aspect of an e-commerce website like JD.com. Without the ability to touch, hold, smell, taste, or otherwise handle the products they're interested in, potential customers have only images to interact with. In JD.com, every product is displayed and described using high-, medium-, and low-resolution images in diverse formats (e.g., JPG, GIF, and PNG). Every day, billions of product images must be processed and stored in the JFS, which requires great performance and capacity. To cut storage costs, JD.com usually compresses large images into smaller ones.

## Analyzing and Optimizing Performance to Remove Bottlenecks and Cut Response Time

"Through close collaboration with Intel engineers, we adopted the Intel® C++ compiler and Intel® Integrated Performance Primitives library in our online image processing application. The application performance improved significantly, and our cost of operations reduced accordingly. This is a typical case demonstrating how technology creates value."

—Liu Haifeng
Chief Architect of Cloud Platform and Director of the System Technology Department
JD.com

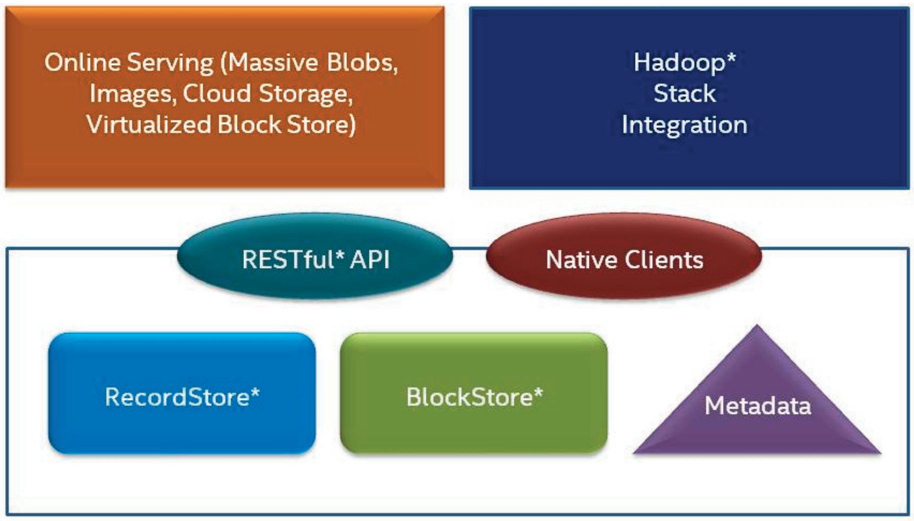

Online Serving (Massive Blobs, Images, Cloud Storage, Virtualized Block Store)
Hadoop* Stack Integration
RESTful* API
Native Clients
RecordStore*
BlockStore*
Metadata

**Figure 1.** JFS architecture

A big challenge for the company is meeting image processing requirements, such as resizing, sharpening, color reducing, or adding special effects to images. Originally, all image processing was achieved using GraphicsMagick* (GM*) and the GNU C Compiler* (GCC*), which consumed a lot of CPU time and seriously impacted JD.com's business resources.

Intel worked closely with JD.com engineers to help them analyze the bottlenecks in their image processing application and optimize them by using Intel® Integrated Performance Primitives and Intel C++ Compiler on the Intel® architecture platform.

### Intel Tools Boost Image Processing

Intel C++ Compiler delivers strong performance on compatible processors. It provides extensive optimizations for the latest Intel® processors including Intel® Xeon® processors and Intel® Xeon Phi™ coprocessors. Its patented, automatic CPU dispatch feature optimizes code for the current running processor and runs code optimized for specified processors identified at application runtime. Intel C++ Compiler also comes with broad support for current and previous C and C++ standards, plus popular extensions including C++11, C99, and OpenMP* 4.0.

Intel® IPP is an extensive library of software functions for media and data processing. It provides thousands of frequently used functions in various domains, including image processing. These functions are highly optimized for performance using Intel® Single Instruction Multiple Data instruction sets.

### Tuning Performance

GM is an open-source image processing system that includes a robust, efficient collection of tools and libraries to support reading, writing, and manipulating of images in more than 88 major formats. As the fundamental part of JD.com's image processing application, GM processes a large number of images on the fly every hour. Working with Intel, JD.com put great effort into optimizing GM and relevant image processing libraries with Intel® Software Development Tools.

The JD.com engineers wanted to identify both performance bottlenecks and potential areas of optimization for their

**Table 1.** Code comparison of the ScaleImage and ScaleImage_ipp functions

**Original Version**

```
MagickExport Image *ScaleImage(const Image *image,const
unsigned long columns,
          const unsignedlong rows,ExceptionInfo *exception)
{
 for (y=0; y < (long) scale_image->rows; y++)
  {
   q=SetImagePixels(scale_image,0,y,scale_image->columns,1);
   if (q == (PixelPacket *) NULL)
    break;
   if (scale_image->rows == image->rows)
    {
     /*
      Read a new scanline.
     */
     p=AcquireImagePixels(image,0,i++,image-
>columns,1,exception);
     if (p == (const PixelPacket *) NULL)
      break;
     for (x=0; x < (long) image->columns; x++)
      {
       x_vector[x].red=p->red;
       x_vector[x].green=p->green;
       x_vector[x].blue=p->blue;
       x_vector[x].opacity=p->opacity;
       p++;
      }
    }
    ...
 }
 ...
 return(scale_image);
}
```

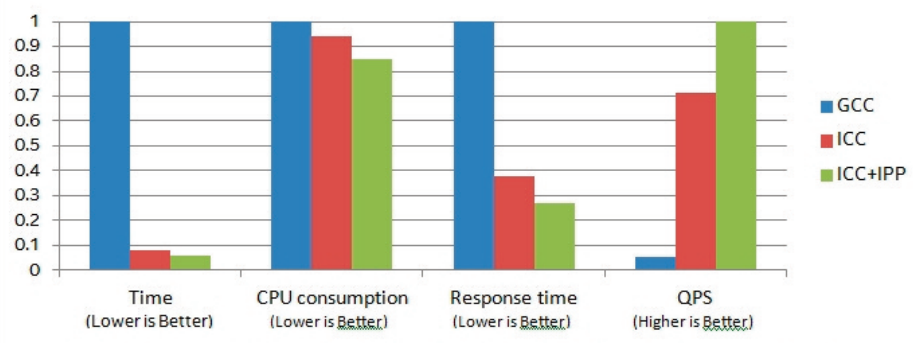
**Optimized Version Using Intel IPP**

```
MagickExport Image *ScaleImage_ipp(const Image *image,const
unsigned long columns,
          const unsignedlong rows,ExceptionInfo *exception)
{
 ...
 /*use ipp to resize*/
 /*only works with RGB image*/
 IppStatus st;
 int channel = 4;
 blobSrc = ippsMalloc_8u(channel*(image->columns) * image-
>rows);
 dstBuf = ippsMalloc_8u(channel*columns * rows);

 DispatchImage(image, 0, 0, image->columns, image->rows,
"RGBA", CharPixel, blobSrc, &srcExp);
 ...
 int interpolation =IPPI_INTER_NN;
 /*Interpolation*/
 st = ippiResizeGetBufSize(srcRoi, dstRoi, channel,interpolation ,
&bufSize);
 pBuffer = ippsMalloc_8u(bufSize);
              ...
              st= ippiResizeSqrPixel_8u_C4R(blobSrc, srcSize,
srcWidthStep, srcRoi, dstBuf, dstWidthStep, dstRoi, x_factor,
y_factor, 0.0, 0.0,interpolation, pBuffer);
              scale_image = ConstituteImage(dstSize.width,
dstSize.height, "RGBA", CharPixel, dstBuf, &dstExp);
 ...
 return scale_image;
}
```

**Table 2.** JD.com benchmark results

| | GCC* | Intel® C++ Compiler | Intel C++ Compiler and Intel® IPP |
|---|---|---|---|
| Time (seconds) | 2,800 | 226.867 | 161.589 |
| CPU consumption | 100 percent | 94 percent | 85 percent |
| Response time (ms) | 2,943 | 378.112 | 269.315 |
| QPS | 100 | 1,322 | 1,856 |



Configuration: Hardware: Inspur NF5270M3 with Intel® Xeon® E5620 @ 2.4GHz, 2 sockets, 64 GB RAM, Hyperthreading is on. Software: Intel® C++ compiler 14.0.2, Intel® Integrated Performance Primitives 8.1, GCC 4.4.7. Linux OS: CentOS release 6.5 (Final)

**Figure 2.** Performance gains from Intel C++ Compiler and Intel IPP

image processing application running on Intel Xeon processor E5-2620-based systems. Since image resizing is one of the most popular and time-consuming operations in the application, they used a benchmark to measure application performance on Intel Xeon processors. The benchmark applies to situations in which 300,000 image files are resized in 500 concurrent instances. The metrics recorded include the time the application took to complete the resizing, CPU consumption, the number of image files resized per second, and average response time.

The JD.com engineers recompiled GM and some image processing libraries—including libturbojpeg*, libpng*, and libwebp*—with GCC and the Intel C++ Compiler. They then benchmarked the application on Intel Xeon processor-based systems. The user times showed that the libraries generated using Intel C++ Compiler performed 12x faster than the GCC-generated libraries. The Intel C++ Compiler-generated libraries were observed to reduce CPU consumption from 100 percent to 94 percent. In addition, average response time was reduced 62 percent. Response time dropped from 2,943 ms to 378 ms . Queries per second (QPS) increased from 100 to 1,322.

JD.com engineers identified that image resizing is a hotspot in GM and implemented a new function, ScaleImage_ipp, which is an IPP-optimized version of the ScaleImage function in GM , which changes the size of an image to specified dimensions. Its functionality is essentially the same as ScaleImage. A few IPP function calls were added to replace original code snippets. Since more than 95 percent of the images processed by GM in JD.com are BMPs, JPEGs, or PNGs, the current implementation of the ScaleImage_ipp function supports only these three formats. A conditional jump at caller sites was added to the ScaleImage function that determines the format of images to be resized. The ScaleImage_ipp function is executed for BMP, JPEG, and PNG images, while the ScaleImage function is executed for all other formats. Table 1 shows a code snippet in the ScaleImage and ScaleImage_ipp functions side-by-side.

The ScaleImage_ipp function led to a considerable performance improvement, according to JD.com's benchmark results (Table 2). CPU consumption dropped to 85 percent; QPS increased to 1,856. This means that JD.com can process more requests using the same number of servers. In addition, JD.com customers see images resized on Web browsers much more quickly.

## Summary

JD.com got extraordinary performance for its image processing application with Intel C++ Compiler. The compiler improved the performance of JD.com image resizing code 12x on Intel Xeon processor-based servers. And, after Intel IPP library was added, the application achieved a speedup of 17x compared to the original GCC-generated code.

With accelerated performance, JD.com can process more images faster and at a lower latency. JD.com achieved significant cost savings through efficient system utilization and speeding up resource-intensive code.

In August 2015, Intel announced community licensing for Intel® Performance Libraries, including Intel IPP. Community licensing allows individuals, companies, and organizations to use these powerful and award-winning performance libraries to create better, more reliable, and faster software applications at no cost.

## Learn More

Try Intel® C++ Compiler and Intel® Integrated Performance Primitives, available as part of Intel® Parallel Studio XE >

Explore Intel's free tools and libraries >