ThunderImage

Quick Start Guide

v1.0

2109/10

Revision History

Revision No.	Date	Remarks
1.0	2019/10	Initial version

Contents

ThunderImage Introduction

Overview

ThunderImage is a high-performance image processing accelerator developed by DeePoly Technology Inc. Based on the CPU+FPGA heterogeneous acceleration architecture, ThunderImage provides users with high-performance JPEG image decoding, image scaling, and JPEG image compression acceleration service support through task-level parallel optimization scheduling and hardware pipeline technology.

The ThunderImage architecture consists of 4 layers, as shown in Figure 1:

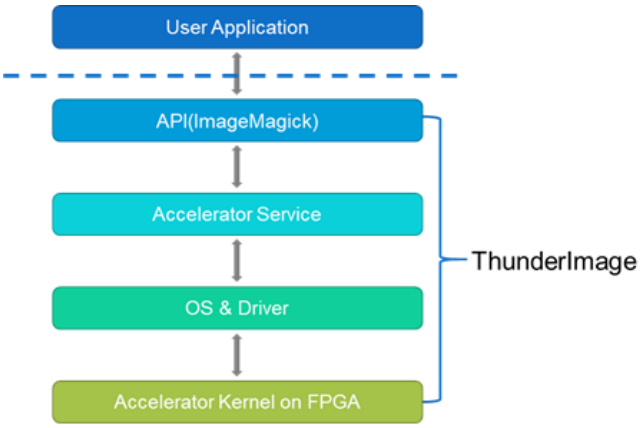

Figure 1 ThunderImage Architecture

By compatible with the ImageMagick API, ThunderImage simplifies the porting process of your existing code.

Support Info

To apply for a trial, please send Company Profile and Contact information to DeePoly product support: support@deepoly.com.

For any question about our ThunderImage , user can get our support through any of the way shown below.

ThunderImage Installation

ThunderImage Environment Requirements

ThunderImage runs on the Linux operating systems only, and does not support Windows. Following shows the recommend HW/SW requirements:

CPU - Intel E5-2630v3 x 2

RAM - 64 GB DDR4@1866MHz

OS - CentOS 7.4.1708

Accelerator Card - Xilinx Alveo U200

ThunderImage Setup

Install Required Packages

For CentOS or RedHat, you must install libtool-ltdl-devel and libjpeg-turbo-devel as prerequisite packages using following commands:

```
$ sudo yum install libtool-ltdl-devel
$ sudo yum install libjpeg-turbo-devel
```

After the installation is complete, confirm the installation status with the following commands:

```
$ sudo rpm -q libtool-ltdl-devel libjpeg-turbo-devel
```

If the installation is correct, the command will return the version of the package installed, as shown below:

```
$ rpm -q libtool-ltdl-devel libjpeg-turbo-devel
libtool-ltdl-devel-2.4.2-22.el7_3.x86_64
libjpeg-turbo-devel-1.2.90-5.el7.x86_64
```

ThunderImage Package Contents

ThunderImage Demo Release package contains 3 tarballs and 2 RPM packages as shown below:

1. ThunderImage-Rpms.tar.gz - RPM tarball for ThunderImage lib, example, service and license
2. ImageMagick.tar.gz - Installation file for ImageMagick
3. Benchmark.tar.gz - Benchmark scripts and input images for ThunderImage
4. xrt_201920.2.3.1301_7.4.1708-xrt.rpm - Xilinx Runtime (XRT) install package

5. xilinx-u200-xdma-201830.2-2580015.x86_64.rpm - U200 Deployment Shell

Install Xilinx Runtime (XRT) and U200 Deployment Shell

ThunderImage is a FPGA-accelerated software solution build on Xilinx Runtime and Alveo U200 accelerator card. As a prerequisite, Xilinx Runtime (XRT) should be installed and the Alveo U200 accelerator card should be flashed with the U200 Deployment Shell.

Command to install XRT and U200 Deployment Shell:

```
$ sudo yum install -y epel-release
$ sudo yum install -y ./xrt_201920.2.3.1301_7.4.1708-xrt.rpm
$ sudo yum install -y ./xilinx-u200-xdma-201830.2-2580015.x86_64.rpm
```

Command to flash the Alveo U200 Accelerator Card:

```
$ sudo /opt/xilinx/xrt/bin/xbutil flash -a xilinx_u200_xdma_201830_2 -t
1561465320
Probing card[0]: DSA on FPGA needs updating
DSA on below card(s) will be updated:
Card [0]
Are you sure you wish to proceed? [y/n]
y
```

If the flash command successes, **cold reboot machine** to load the new image on FPGA. If more than 1 card has been installed on the computer, using following command to determine which card to flash:

```
$ /opt/xilinx/xrt/bin/xbutil list
```

The output of this command is as follows:

```
$ /opt/xilinx/xrt/bin/xbutil list
INFO: Found total 1 card(s), 1 are usable
[0] 04:00.1 xilinx_u200_xdma_201830_2
```

Install ImageMagick-7.0.7-11Q16

ThunderImage is compatible with ImageMagick-7.0.7-11Q16.

Using following commands to install ImageMagick-7.0.7-11Q16:

```
$ tar -xzf ImageMagick.tar.gz
$ cd ImageMagick
$ sudo bash install-ImageMagick.sh
```

Install ThunderImage

Using following commands to install ThunderImage:

```
$ tar -xzf ThunderImage-Pkgs.tar.gz
$ cd ThunderImage-Pkgs
$ sudo bash ./install_rpm.sh .
```

The ThunderImage will be installed to */opt/ThunderImage-Premium*. Following shows the contents of the directory:

```
$ ls
etc/ examples/ include/ lib/ license/ server/
```

Using ThunderImage

Set Accelerator Card Slot

If more than 1 card has been installed on the computer, edit the value of $FPGA_SLOT in /opt/ThunderImage-Premium/etc/thunder-image.cfg to configure which accelerator card to use. Refer to *Section Install Xilinx Runtime (XRT) and U200 Deployment Shell* to determine the Card Slot on the computer. By default, ThunderImage will use Card Slot 0.

```
XILINX_XRT_ROOT=/opt/xilinx/xrt
FPGA_SLOT=0
```

ThunderImage Service Configuration

ThunderImage has been wrapped as a Linux service, using following command to start the ThunderImage service:

```
$ sudo systemctl start thunder-image.service
```

The startup process will be completed within 20s.

To stop the ThunderImage service:

```
$ sudo systemctl stop thunder-image.service
```

To check the service status:

```
$ sudo systemctl status thunder-image.service
```

To restart the service:

```
$ sudo systemctl restart thunder-image.service
```

ThunderImage APIs

Compare to the original ImageMagick APIs, which use independent API for JPEG decoding, image scaling, and JPEG encoding, ThunderImage combined JPEG decoding, scaling, and JPEG encoding to 1 API. Considering different user requirements, the following 3 APIs are provided according to different calling methods.

File API

The File API uses the file as an interface for input and output

```
MagickBooleanType ThunderScaleImage( MagickWand* magick_wand,
      const char* source_filename,
      const char* target_filename,
      const size_t resize_width,
      const size_t resize_height,
      const int keep_aspect_ratio);
@param magick_wand - Magick_wand object
@param source_filename - Filename of source JPEG
@param target_filename - Filename of target JPEG
@param resize_width - Resize width of target JPEG
@param resize_height - Resize height of target JPEG
@param keep_aspect_ratio - Keep the aspect ratio of the output image match
to the input image, 1 for keep

@return MagicTrue for success, MagicFalse for failure
```

Blob API

The Blob API uses the Blob as an interface for input and output

```
unsigned char* ThunderScaleImageBlob(
      MagickWand* magick_wand,
      const void* source_blob,
      const size_t source_length,
      size_t* target_length,
      const size_t resize_width,
      const size_t resize_height,
      const int keep_aspect_ratio);
@param magick_wand - Magick_wand object
@param source_blob - Source blob
@param source_length - Image length in source blob
@param target_length - Image length in target blob
@param resize_width - Resize width of target JPEG
@param resize_height - Resize height of target JPEG
@param keep_aspect_ratio - Keep the aspect ratio of the output image match
to the input image, 1 for keep

@return Target Blob
```

Shared Memory (SHM) API

The SHM API uses the Shared Memory as an interface for input and output. Input image should be copied to directory under '/dev/shm' path.

```
MagickBooleanType ThunderScaleImageSHM(
      MagickWand* magick_wand,
      const char* source_shmfile,
      const char* target_shmfile,
      const size_t source_length,
      const size_t target_shm_depth,
      size_t* target_img_size,
      const size_t resize_width,
      const size_t resize_height,
      const int keep_aspect_ratio);

@param magick_wand - Magick_wand object
@param source_shmfile - Filename of source SHM
@param target_shmfile - Filename of target SHM
@param source_length - Image length in source SHM
@param target_shm_depth - Depth of target SHM
@param target_img_size - File size of target image
@param resize_width - Resize width of target JPEG
@param resize_height - Resize height of target JPEG
@param keep_aspect_ratio - Keep the aspect ratio of the output image match
to the input image, 1 for keep

@return MagicTrue for success, MagicFalse for failure
```

Reference Designs

All reference designs have been installed at /opt/ThunderImage-Premium/examples. Following is the description:

1. Ref_FileAPI - Reference design using ThunderImage File API
2. Ref_BlobAPI - Reference design using ThunderImage Blob API
3. Ref_SHMAPI - Reference design using ThunderImage Shared Memory API
4. Ref_AllAPI_multithreads - High-performance reference design using all 3 APIs with multi-thread support.

ThunderImage Benchmark

Setup Benchmark Scripts and Images

Untar Benchmark.tar.gz to user-defined *$BenchmarkDir* with following command:

```
$ tar -xzf Benchmark.tar.gz $BenchmarkDir
```

Following shows the benchmark contents:

```
$ ls
copy.sh  cron.sh  default.cfg  function  J2J_cfg  performance.py
perfTest.sh  testimage
```

Run Benchmark

Untar Benchmark.tar.gz to user-defined *$BenchmarkDir* with following command:

```
$ ./cron.sh J2J_cfg/case1.cfg
```

default.cfg is a template of ThunderImage benchmark configuration file. Following shows the structure of the configure file

```
#!/bin/bash
CfgPath=$(dirname $(readlink -f "$0"))
# Input directory or file
Src=${CfgPath}/testimage/case/case1.jpg
# Target Resolution
TargetRes=( '768 576' '240 180' )
############################################################
# The following configure variable has default value.

# Copies - Copy source image directory or file $Copies times to the
# input directory for benchmark input
Copies=1
# Repeat - In order to avoid duplicate file I/O, the test program
# supports to repeat the process $Repeat times for each file of input
directory.
Repeat=20000
# APIType - Tell the test program to use which type of ThunderImage API.
# 0 for ImageMagick CPU flow
# 1 for ThunderImage File API flow
# 2 for ThunderImage Blob API flow
# 3 for ThunderImage SHM API flow
APIType=3
# WriteFile - Available except File API flow, set to 1 will force
ThunderImage to dump all
# result images to files.
WriteFile=0
# keepAR - Keep the aspect ratio of the output image match to the input
image.
KeepAR=0
# Rotate - Flag to rotate the target image from width x height to height x
width
Rotate=0
# CpuThreadNums - When $APIType is 0, the default thread numbers of
ImageMagick CPU flow.
CpuThreadNums=(8 16 32 40)
# FpgaThreadNums - When $APIType is not 0, the default thread numbers of
ThunderImage API flow.
FpgaThreadNums=(32 40 48 56 64)
# Performance - Sort performance by performance parameters
# 2-Thread, 3-Runtime(s), 4-Throughput(MBps), 5-CPUUsage, 6-QPS, 7-Latency
Performance=4
```

A *benchmark.sh* has been provided to run a basic benchmark of ThunderImage. Following shows the details of the basic benchmark:

1. Input Images
    1. 1 pic with 1024x768 resolution
    2. 1 pic with 4096x2160 resolution
    3. 1 pic with 4096x2160 resolution
    4. 1 pic with 1920x1080 resolution
2. Target resolutions
    1. For 4096x2160 input
        1. 1024x768
        2. 640x480
    2. For 4096x2160 input
        1. 1024x768
        2. 640x480
    3. For 1024x768
        1. 768x576
        2. 240x180
3. Number of source file copies - 1
4. Repeat number of input files
    1. 20000 for 1024x768 input
    2. 4000 for other input
5. Number of threads - 32 40 48 56 64
6. ThunderImage API Type - SHM API
7. Force to dump the output file - No
8. Keep aspect ratio of output image match to input image - No

Use following command to run the initial benchmark:

```
$ ./benchmark.sh
```

Check the Benchmark Report

The script will generate a directory named with *${CurTime}_${SrcName}_${Copies}_${Repeat}_${APIType}_${WriteFile}_${KeepAR}* to store all inputs and outputs of the benchmark.

Following shows the contents of the directory:

```
$ tree 20190530-221831_case1_jpg_1_20000_3_0_0/ -L 2
20190530-221831_case1_jpg_1_20000_3_0_0/
├── 240_180
│   ├── console.log
│   ├── output_32threads
│   ├── output_40threads
│   ├── output_48threads
│   ├── output_56threads
│   └── output_64threads
├── 768_576
│   ├── console.log
│   ├── output_32threads
│   ├── output_40threads
│   ├── output_48threads
│   ├── output_56threads
│   └── output_64threads
├── cron.log
└── cron_rpt.csv

12 directories, 4 files
```

All input images have been copied to *input* directory and output images could be found under *output_${ThreadNum}threads* directory under output directory of each resolution(eg, 240_180). After it's done, a *cron_rpt.csv* will be generated and shows QoR statistics including "*Runtime*", "*Throughput*", "*TPS*", "*Latency*" and "*CPU Usage*". Following shows the report:

```
Src: /localworkspace/ypwang/PhoenixBenchmark/0530BenchmarkVVDN/testimage/case/case1.jpg
Total size in MB: 2534.21
Total image amount: 20000

768x576
Resize, Thread, Runtime(s), Throughput(MBps), CPUUsage, QPS, Latency
768x576, 64, 9.227, 274.65, 9.93, 2167.55, 29.124
768x576, 56, 9.233, 274.47, 9.92, 2166.14, 25.523
768x576, 48, 9.248, 274.02, 9.62, 2162.62, 21.879
768x576, 40, 9.258, 273.73, 9.63, 2160.29, 18.278
768x576, 32, 9.328, 271.67, 9.95, 2144.08, 14.748

240x180
Resize, Thread, Runtime(s), Throughput(MBps), CPUUsage, QPS, Latency
240x180, 32, 3.981, 636.57, 16.55, 5023.86, 6.196
240x180, 64, 4.092, 619.3, 17.07, 4887.58, 12.73
240x180, 40, 4.189, 604.96, 16.32, 4774.4, 8.159
240x180, 56, 4.288, 591.0, 16.56, 4664.17, 11.688
240x180, 48, 4.346, 583.11, 16.28, 4601.93, 10.167
```

This *.csv* format report can be opened with Microsoft Excel on Windows as following shows:

Src: /localworkspace/ypwang/PhoenixBenchmark/0530BenchmarkVVDN/testimage/case/case1.jpg						
Total size in MB: 2534.21						
Total image amount: 20000						
768x576						
Resize	Thread	Runtime(s)	Throughput(MBps)	CPUUsage	QPS	Latency
768x576	64	9.227	274.65	9.93	2167.55	29.124
768x576	56	9.233	274.47	9.92	2166.14	25.523
768x576	48	9.248	274.02	9.62	2162.62	21.879
768x576	40	9.258	273.73	9.63	2160.29	18.278
768x576	32	9.328	271.67	9.95	2144.08	14.748
240x180						
Resize	Thread	Runtime(s)	Throughput(MBps)	CPUUsage	QPS	Latency
240x180	32	3.981	636.57	16.55	5023.86	6.196
240x180	64	4.092	619.3	17.07	4887.58	12.73
240x180	40	4.189	604.96	16.32	4774.4	8.159
240x180	56	4.288	591	16.56	4664.17	11.688
240x180	48	4.346	583.11	16.28	4601.93	10.167

ThunderImage Trouble Shooting

ThunderImage Status

Using following command to check ThunderImage service status:

```
sudo systemctl status thunder-image.service
```

If service status is ***inactive***, try to restart the service and check the status again.

Report Bugs

When creating a bug report, please include the following info:

1. ThunderImage runtime log at */var/log/thunder-image.log*
2. Output of *dmesg*
3. Output of *xbutil query*
4. Output of *xbutil scan*

And send this information to support@deepoly.com.