GUETZLI 图片压缩算法分析

李梂忆

摘要：如何平衡图片大小和质量是图片压缩的核心问题。本文通过与传统编码器 Libjpeg-turbo 在压缩率、压缩时间、占用内存等方面的对比，另用 Intel 研发的 VTune 优化软件进行热点分析，来评估 Google 的 jpeg 图片压缩算法 Guetzli。测试表明，Guetzli 能在已有压缩算法基础上再将 jpeg 图片压缩 20%到 30%，且肉眼观察图片质量并无改变。但于此同时，压缩图片的时间大为增加。若能将压缩时间缩短，Guetzli 将为图片压缩提供新的可能。

关键词：Guetzli；Libjpeg-turbo；图片压缩；VTune；

1. Guetzli 介绍

Guetzli 是 Google 于 2017 年发布的一种 JPEG 编码器，旨在在高视觉质量下实现出色的压缩密度。 Guetzli 生成的图像通常比其他压缩算法生成的等效质量图像小 20-30%。 目前 Guetzli 的计算非常缓慢。

1.1 Guetzli 安装

Guetzli 为 open source，Google 将所有代码发布于 GitHub。

https://github.com/google/guetzli

详见附录。

1.2 Guetzli 特性

Guetzli 使用迭代优化过程。为了使问题更简单，优化器不受文件大小的指导。相反，它仅以感知质量为目标驱动。其目的是创建一个具有低于且尽可能接近给定阈值的感知距离的 JPEG 编码。每次迭代产生一个候选输出 JPEG，最后选出最好的一个。

Guetzli 使用闭环优化器对图像进行两方面的调整：优化 JPEG 全局量化表和每个 JPEG 块中的 DCT 系数值。具体优化过程见下图：

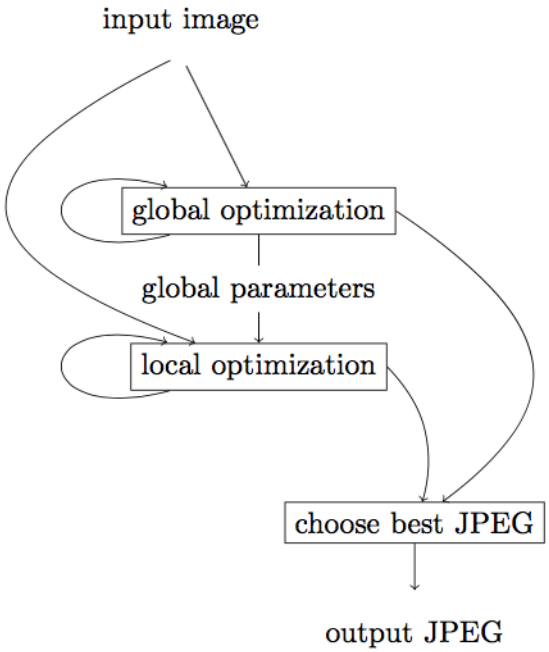

图 1 Guetzli 优化过程（https://arxiv.org/pdf/1703.04421.pdf）

1.3 Butteraugli 度量标准

Guetzli 使用同样来自 Google 的感知距离度量标准 Butteraugli 作为其优化过程中的反馈来源。Butteraugli 是一种比其他编码器“以更彻底和更详细的方式评估颜色感知和视觉遮蔽”的模型。其目标是找到人眼不能与原始图像区分的最小的 JPEG。

Butteraugli 考虑到大多数 JPEG 编码器没有使用的三个视觉特性。首先，由于锥体的灵敏度谱的重叠，伽马校正不应该分别应用于每个 RGB 通道。例如，人眼看到的黄光量和对蓝光的敏感度有关，所以对于黄色附近的蓝色变化可以不太精准地编码。YUV 色彩空间被定义为伽马压缩 RGB 的线性变换，因此不足以对这种现象进行建模。其次，人眼在蓝色中的分辨率比红色和绿色都要低，在视网膜的高分辨率区域几乎没有蓝色受体，因此，可以较不精确地编码蓝色的高频率变化。最后，图像中精细结构的可视性取决于附近的视觉活动量，即我们可以更不精确地编码具有大量视觉噪声的区域。

以上方面的考量使得 Guetzli 能保证图像的均匀损失。

2.Libjpeg-turbo 介绍

为测试 Guetzli 性能，本文将 Guetzli 与另一常用 jpeg 编码器 Libjpeg-turbo 对比。

2.1 Libjpeg-turbo 安装

https://github.com/libjpeg-turbo/libjpeg-turbo

详见附录。

2.2 Libjpeg-turbo 特性

libjpeg-turbo 是 libjpeg 的一个分支，它使用 SIMD 指令来加速基线 JPEG 编码和解码，将 bmp 或 ppm 图像压缩为 jpg 格式。

3.性能测试

3.1 测试目的

分别对 Guetzli 压缩的图像和 Libjpeg-turbo 压缩的图像进行对比测试，对比在不同 CPU 和不同质量参数下各自的压缩时间与压缩率。

3.2 测试环境

系统硬件环境

Platform	**Broadwell**
Processor	E5-2699 v4
Frequency	2.20 GHz
Max Turbo Frequency	3.50 GHz
Memory	8 * 32GB 2133 MHz
FSB/QPI Frequency	9.6 GT/s
Thread(s) per Core	2
Sockets	2
Number of Core per	22
L1d Cache	32KB
L1i Cache	32KB

L2 Cache	256KB
L3 Cache (Total)	56320KB
SMT/MUNA/TURBO	ON

表 1 CPU1 信息

Platform	**Skylake**
Processor	8180
Frequency	2.5 GHz
Max Turbo Frequency	3.5 GHz
Memory	12 * 16GB 2666 MHz
FSB/QPI Frequency	10.4 GT/s
Thread(s) per Core	2
Sockets	2
Number of Core per	28
L1d Cache	32KB
L1i Cache	32KB
L2 Cache	1024KB
L3 Cache (Total)	39424KB
SMT/MUNA/TURBO	ON

表 2 CPU2 信息

系统软件环境

OS	CentOS 7.3.1611
kernel	3.10.0
Compiler	gcc:4.8.5 20150623

表 3 系统软件环境

3.3 测试实施

图片	像素	大小（字节）
nightshot_iso_100.bmp	192*144	82998
head.bmp	444*600	799254
lagochungara.bmp	871*573	1499022

ahom3.bmp	1024*768	2359350
earth.bmp	2048*1024	6291510

表 4 图片信息

依次测试五张大小不一的 bmp 照片。在同一 CPU 下，先用 Libjpeg-turbo 压缩为 jpg 格式，再用 Guetzli 以不同质量系数二次压缩。换 CPU 重复操作。

CPU 选取 Intel 的 E5-2688 V4 和 Skylake8180, Skylake 是性能更高的处理器。

因 Guetzli 质量参数最低支持 84，故选取 84，90，95 三个参数进行对比。

命令行：

```
time -p ./cjpeg -outfile test.jpg image.bmp
time -p ./bin/Release/guetzli --quality 84 test.jpg output.jpg
```

最后换单进程为多进程测试。

多进程代码：

```
#include <stdlib.h>
#include <omp.h>
int main()
{
    #pragma omp parallel for
    for(int i=0; i<=1000; i++)
    {
        ./bin/Release/guetzli —quality 84 test.jpg output.jpg;
    }
}
```

命令行：

```
g++ omp.cc -fopenup
./a.out
```

3.4 测试结果

单进程

CPU	图片大小(字节)	libjpeg-turbo消耗内存(字节)	libjpeg-turbo压缩时间(秒)	libjpeg-turbo压缩后大小(字节)	libjpeg-turbo压缩率	guetzli质量系数	guetzli消耗内存(字节)	guetzli压缩时间(秒)	gueztli压缩后大小(字节)	gueztli压缩率
E5-2699 V4	82998	25368	0.03	4357	94.75%	84	19196	0.89	3366	22.75%
						90	19196	0.83	3624	16.82%
						95	19196	0.83	3879	10.97%
	799254	25314	0.04	39063	95.11%	84	36800	12.85	33795	13.49%
						90	34848	11.28	35832	8.27%
						95	36800	9.04	36900	5.54%
	1499022	25330	0.04	116070	92.26%	84	56040	26.12	95496	17.73%
						90	55868	25.06	102766	11.46%
						95	55868	21.28	109555	5.61%
	2359350	25316	0.04	40584	98.28%	84	78748	37.26	31022	23.56%
						90	77084	27.26	32562	19.77%
						95	74344	23.87	33908	16.45%
	6291510	25314	0.05	237097	96.23%	84	182472	104.36	190895	19.49%
						90	182472	93.76	206747	12.80%
						95	174792	93.27	220427	7.03%

表5 单进程E5-2699 V4测试结果

CPU	图片大小(字节)	libjpeg-turbo压缩时间	guetzli压缩时间
SKL8180	82998	0.01	0.74
			0.66
			0.66
	799254	0.01	10.43
			8.89
			7.13
	1499022	0.01	21.73
			20.19
			18.57
	2359350	0.02	30.68
			22.30
			19.63
	6291510	0.02	91.95
			81.50
			81.46

表6 单进程Skylake8180测试结果

多进程

CPU	图片大小(字节)	guetzli质量系数	guetzli压缩时间(秒)	吞吐(张/秒)
E5-2699 V4	82998	84	22.69	44.07
		90	20.55	48.66
		95	20.20	49.50
	799254	84	326.55	3.06
		90	280.34	3.57
		95	220.03	4.54
	1499022	84	658.49	1.52
		90	618.13	1.62
		95	561.95	1.78
	2359350	84	892.16	1.12
		90	653.18	1.53
		95	575.17	1.74
	6291510	84	2468.88	0.41
		90	2192.57	0.46
		95	2180.96	0.46

CPU	图片大小(字节)	guetzli质量系数	guetzli压缩时间(秒)	吞吐(张/秒)
SKL8180	82998	84	20.19	49.53
		90	17.96	55.68
		95	15.76	63.45
	799254	84	274.21	3.65
		90	222.96	4.49
		95	188.18	5.31
	1499022	84	539.56	1.85
		90	510.93	1.96
		95	468.20	2.14
	2359350	84	781.23	1.28
		90	569.02	1.76
		95	429.91	2.33
	6291510	84	2272.84	0.44
		90	1989.99	0.50
		95	2002.44	0.50

表7 多进程测试结果

3.5 结果分析

分析 3.4 节的测试数据，可以得出如下图表。

- 原图、Libjpeg-turbo 和 Guetzli 压缩后的图片大小对比

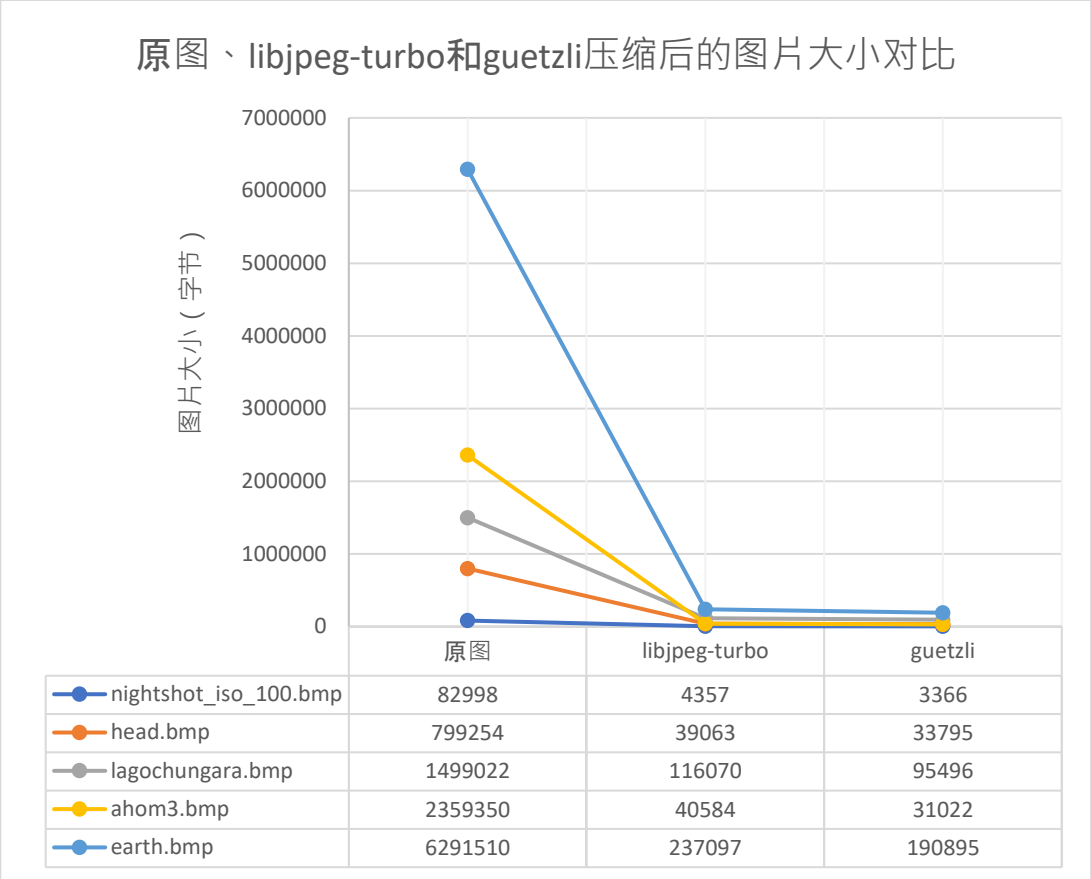

图 2 原图、Libjpeg-turbo 和 Guetzli 压缩后的图片大小对比

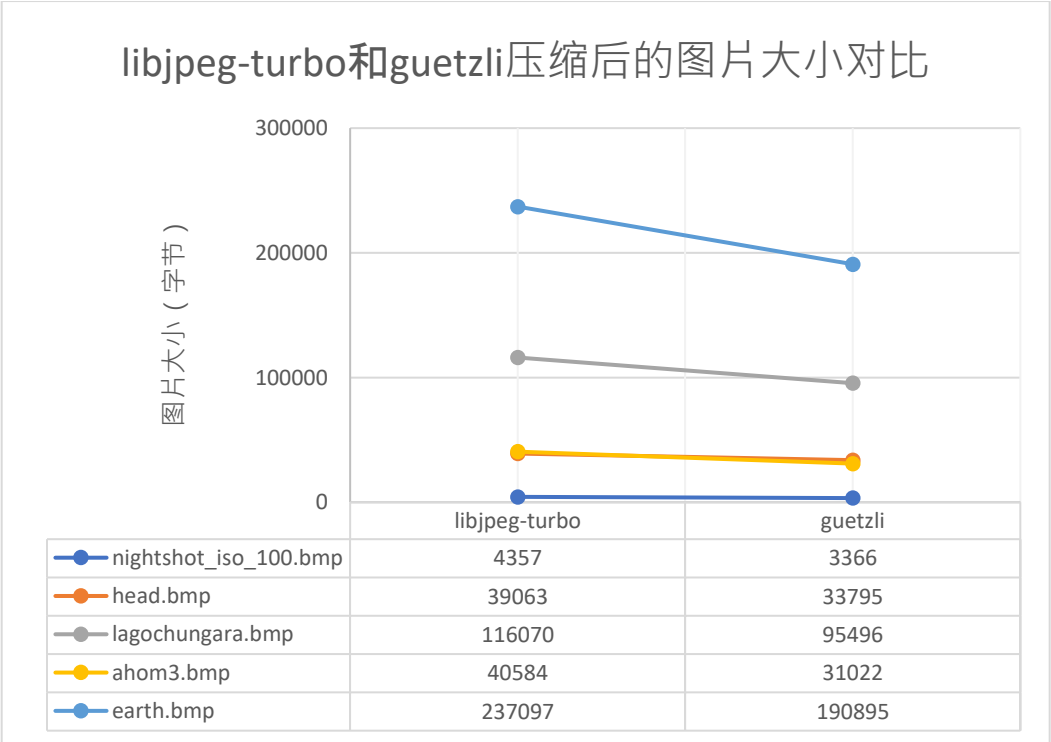

图 3 Libjpeg-turbo 和 Guetzli 压缩后的图片大小对比

由图 2、3 可以看出，Guetzli 在 Libjpeg-turbo 压缩的基础上能再压缩 15%左右。

- 单进程 guetzli 在不同 CPU 下的压缩时间对比

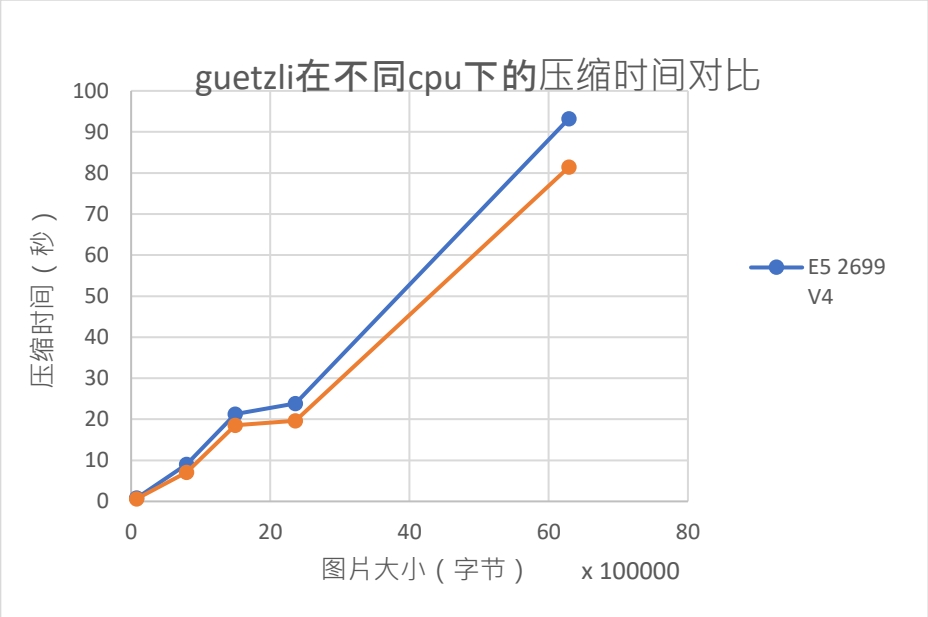

图 4 单进程 Guetzli 在不同 CPU 下的压缩时间对比

由图 4 可以看出，图片大小越大，Guetzli 压缩时间越长。Skylake8180 能将压缩时间减少 20%左右。

- guetzli 在不同质量系数下的压缩时间对比

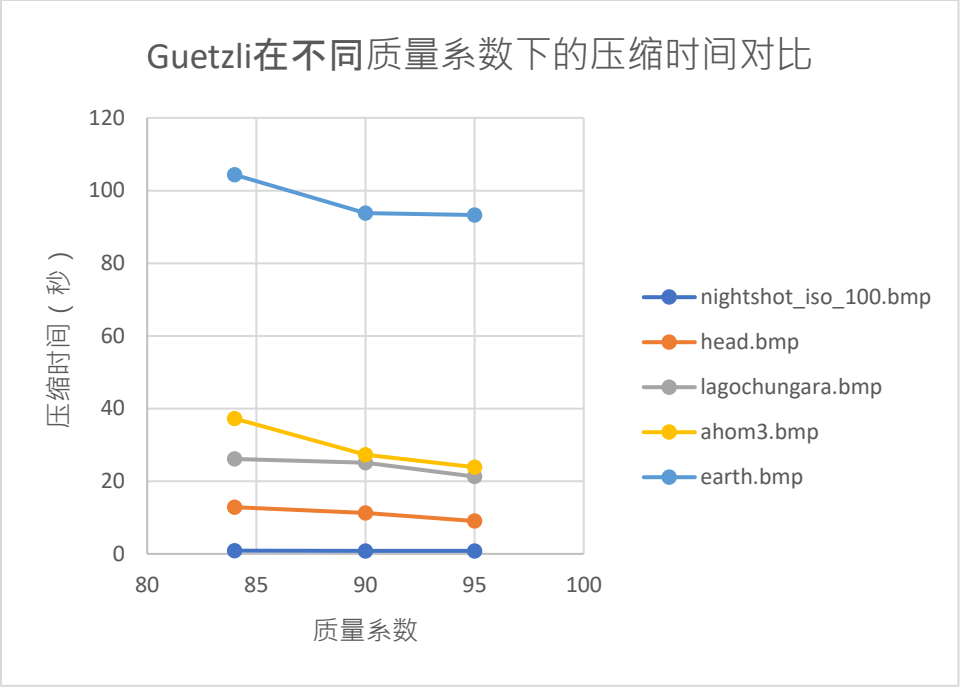

图 5 Guetzli 在不同质量系数下的压缩时间对比

由图 5 可以看出，质量系数越大，Guetzli 压缩时间越短。

- Guetzli 在不同质量系数下的压缩率对比

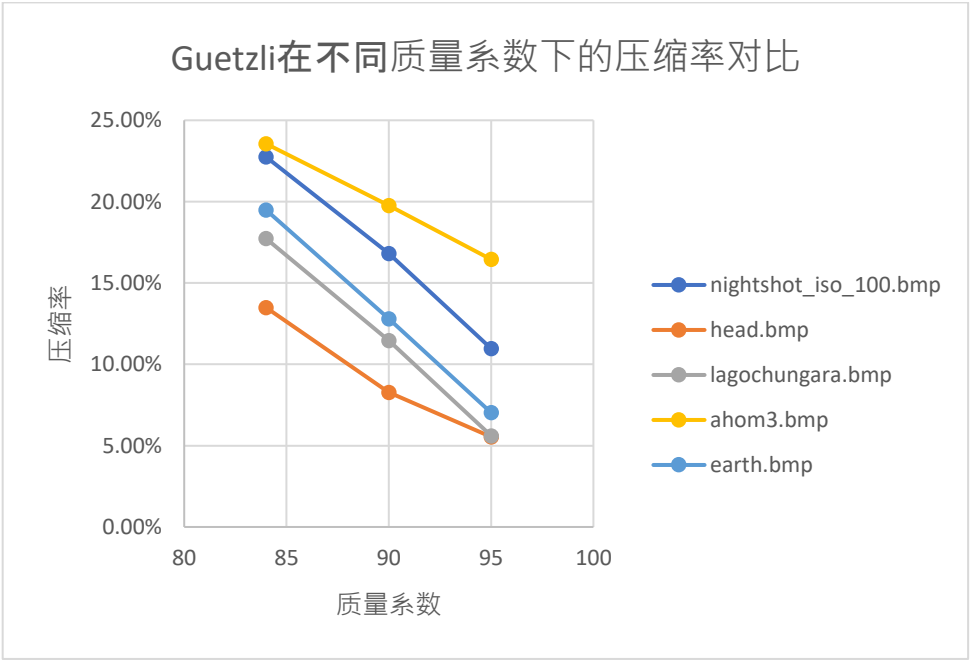

图 6 Guetzli 在不同质量系数下的压缩率对比

由图 6 可以看出，质量系数越大，压缩率越低。

- Guetzli 与 Libjpeg-turbo 在不同图像下的内存消耗对比

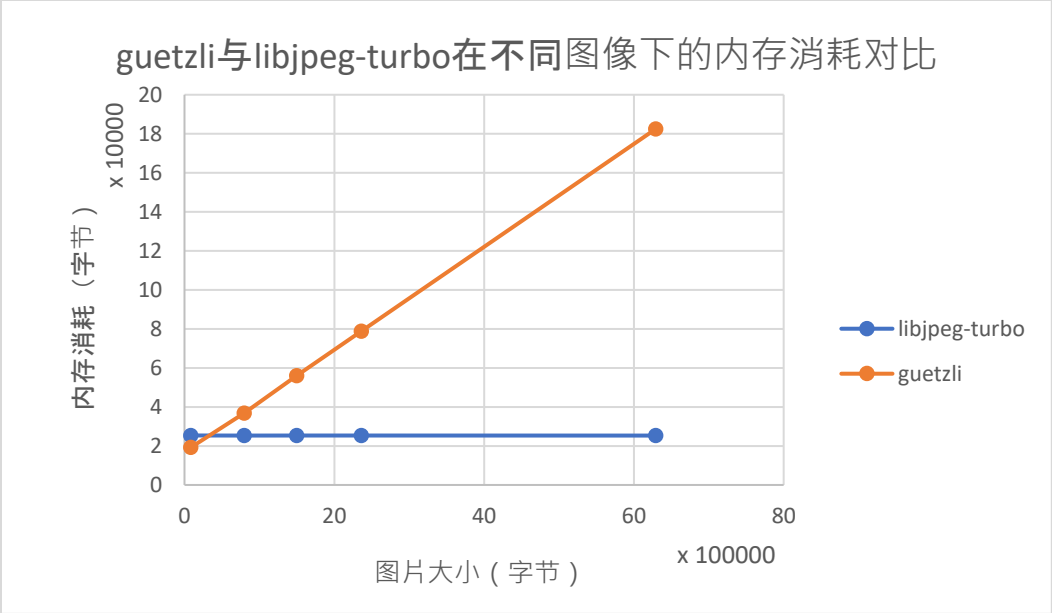

图 7 Guetzli 与 Libjpeg-turbo 在不同图像下的内存消耗对比

由图 7 可以看出，图片大小越大，guetzli 消耗内存越多，libjpeg-turbo 消耗内存较少且基本不变。

- 多进程 Guetzli 在不同 CPU 下的吞吐对比

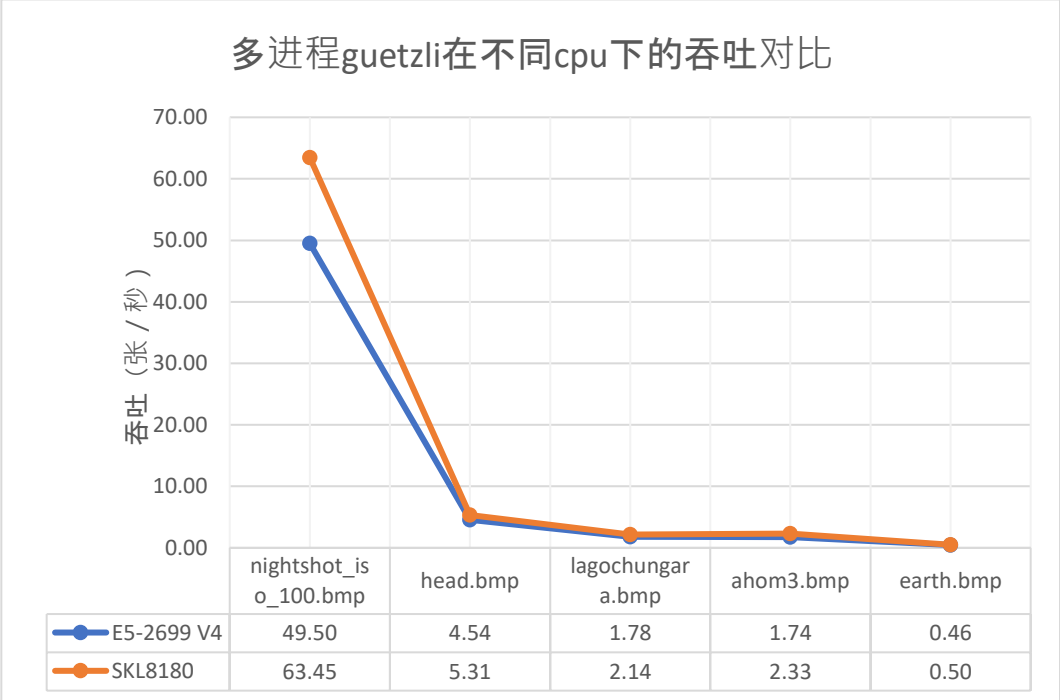

图 8 多进程 Guetzli 在不同 CPU 下的吞吐对比

由图 8 可以看出，Skylake8180 能将吞吐增加 20%左右。

4. VTune 分析

4.1 VTune 安装

https://software.intel.com/en-us/-getting-started-with-intel-vtune-amplifier-xe-2017

详见附录。

4.2 VTune 介绍

VTune Amplifier 性能分析器是 Intel Parallel Studio 的产品，是用于基于 32 位和 64 位 x86 的机器的软件性能分析的商业应用。它具有 GUI（图形用户界面）和命令行，并提供 Linux 或 Microsoft Windows 操作系统的版本。

VTune Amplifier 协助各种代码分析，包括堆栈采样，线程分析和硬件事件采样。分析器结果包括例如每个子程序中花费的时间等细节。

本文主要通过 VTune 热点分析 guetzli 处理时间长的原因。

4.3 VTune 实施

在 root 上使用命令行进行测试，将结果复制到 spark 上 gui 显示。

```
source amplxe-vars.sh
amplxe-cl -collect hotspots bin/Release/guetzli --quality 84 output1.jpg output2.jpg
amplxe-cl -report hotspots
```

4.4 VTune结果

- Summary

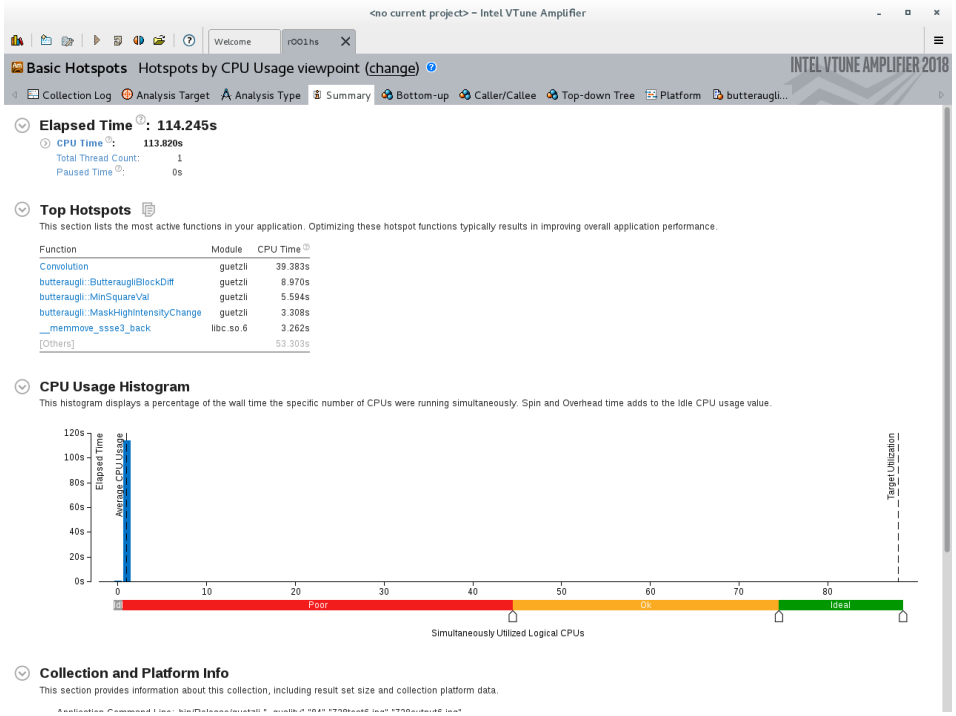

图 9 Summary: CPU 使用柱状图

- Bottom-up

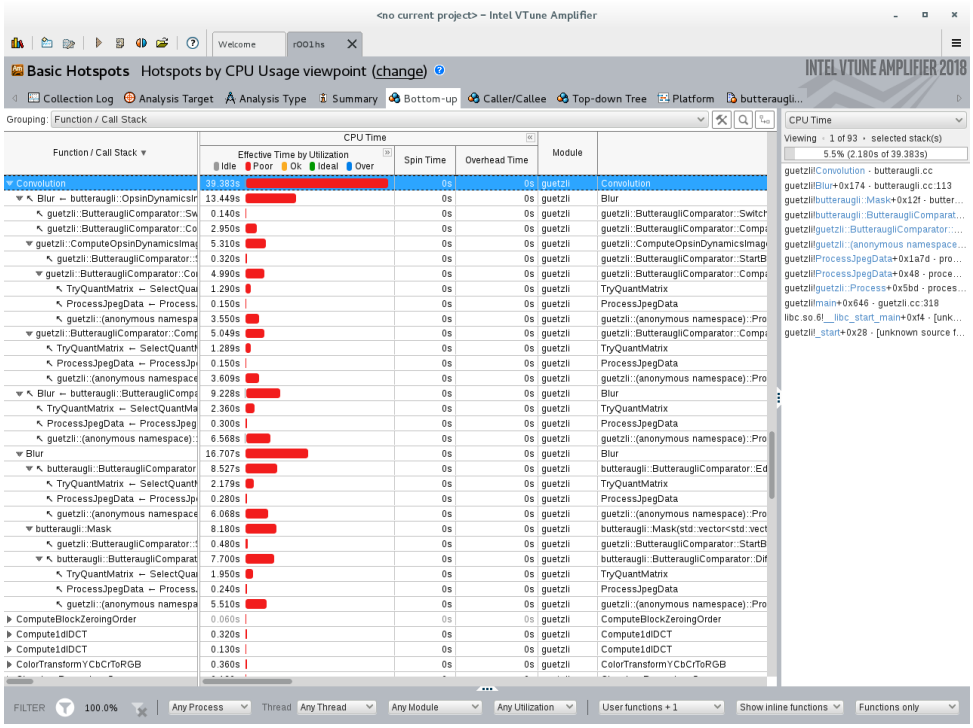

图 10 Bottom-up：convolution 占用 CPU 时间

由图 9 可以看出，convolution 是耗时最长的 function。

- Caller/Callee

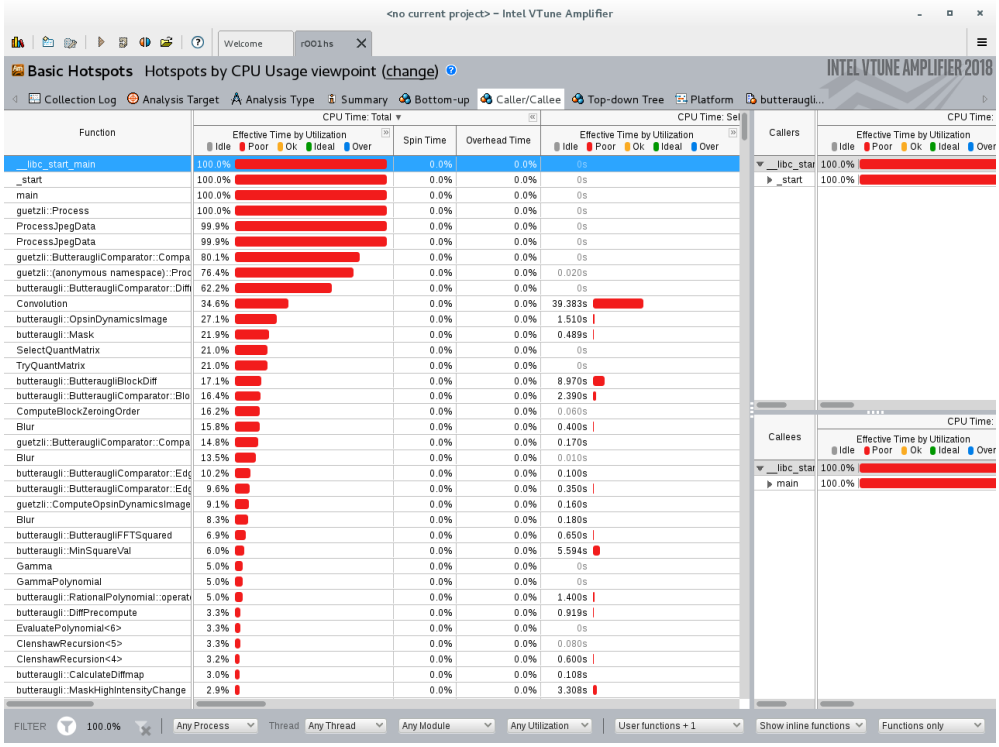

图 11 Caller/Callee：convolution 函数的 caller

- 具体代码

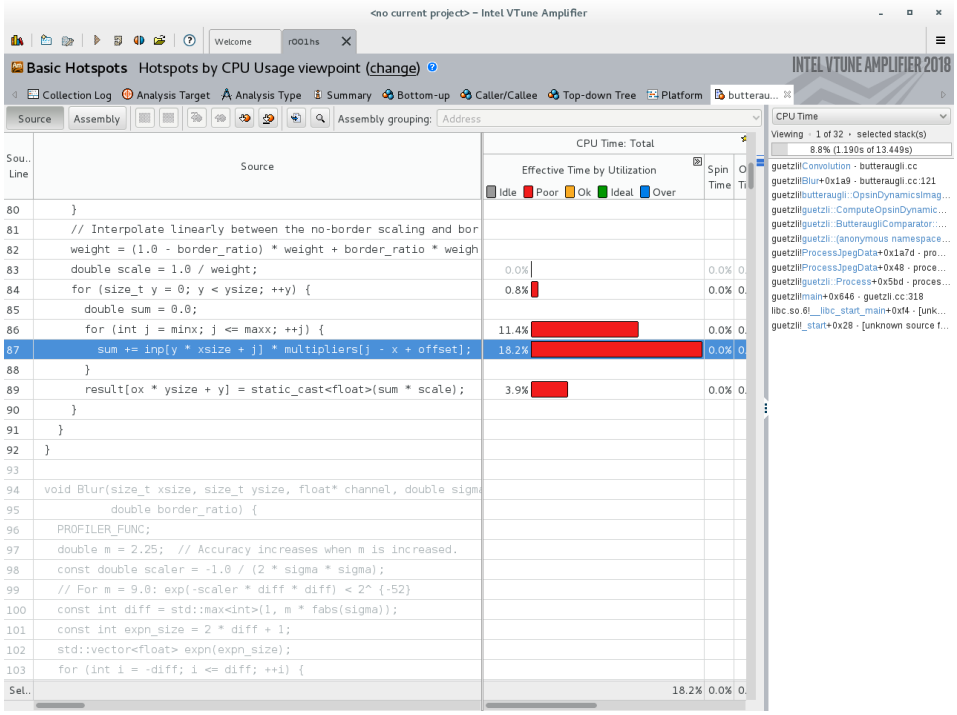

图 12 具体代码

5. 结论

Guetzli 能在普通 jpeg 图片的基础上额外压缩 20%到 30%，且肉眼观察图片质量无改变。但因为其压缩时间也大为加长，性能难达商用。经 VTune 热点分析，若提高 convolution 的耗时，则 Guetzli 的可用性增强。

6. 参考文献

[1] https://arxiv.org/pdf/1703.04421.pdf

7. 附录

7.1 Guetzli 安装

1.复制源代码

```
git clone git@github.com:google/guetzli.git
```

2.安装 libpng

3.make

7.2 Libjpeg-turbo 安装

1.复制源代码

```
git clone https://github.com/libjpeg-turbo/libjpeg-turbo.git
```

2.安装 nasm

```
yum install nasm
```

3.mkdir build

4.autoreconf –fiv

5.cd build

6.sh ../configure

7.make

7.3 VTune 安装

1.scp spark@dl-bj:/home/jimin/parallel_studio_xe_2018_beta_update1_cluster_edition.tgz .

7.4 图片示例

earth.bmp	2048*1024	6291510

原图

libjpeg-turbo

guetzli quality 84