

Ngix* HTTPS优化指南 - 基于第三代英特尔® 至强® 可扩展处理器及英特尔® QuickAssist 技术



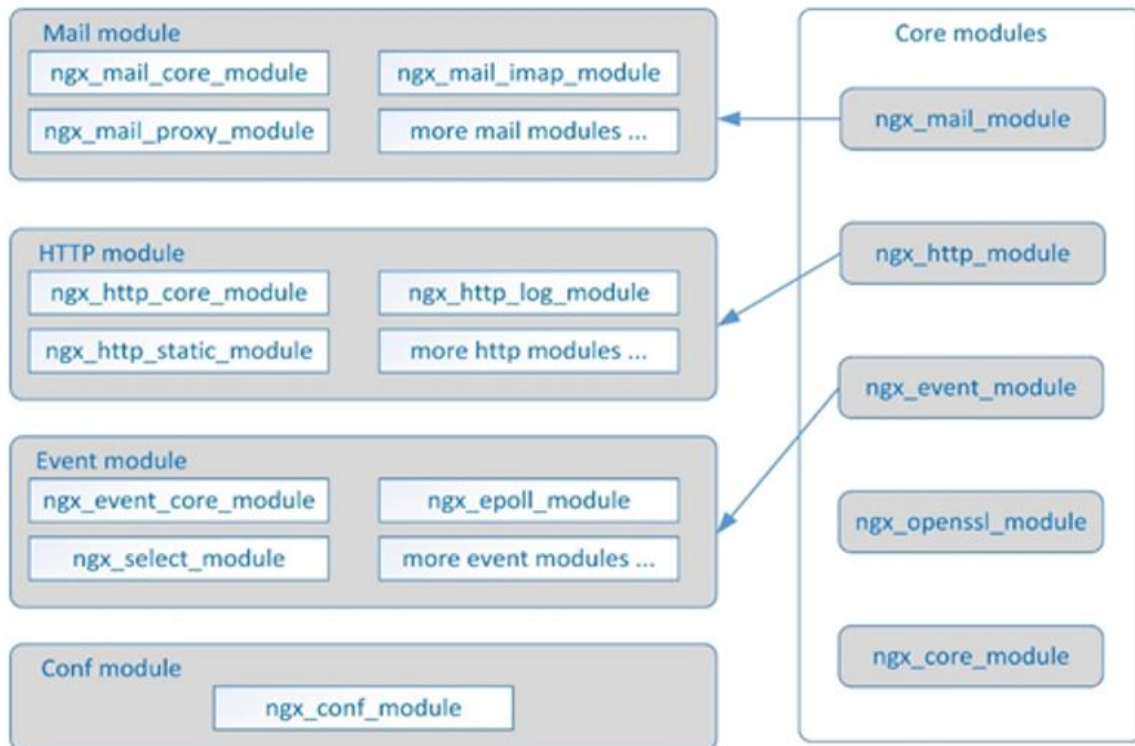
Revision Record	2
1. 介绍.....	3
2. BIOS 设置	5
3. Linux*优化.....	5
4. Intel® QAT安装和设置	5
5. OpenSSL*和QAT_Engine安装	6
6. Ngix安装及优化设置.....	7
6.1. Ngix安装.....	7
6.2. Ngix优化设置	8
7. 性能测试工具介绍.....	10
8. 意见反馈.....	11

Revision Record

Date	Rev.	Description
05/27/21	1.1	Added feedback section
04/22/2021	1.0	Initial public release.

1. 介绍

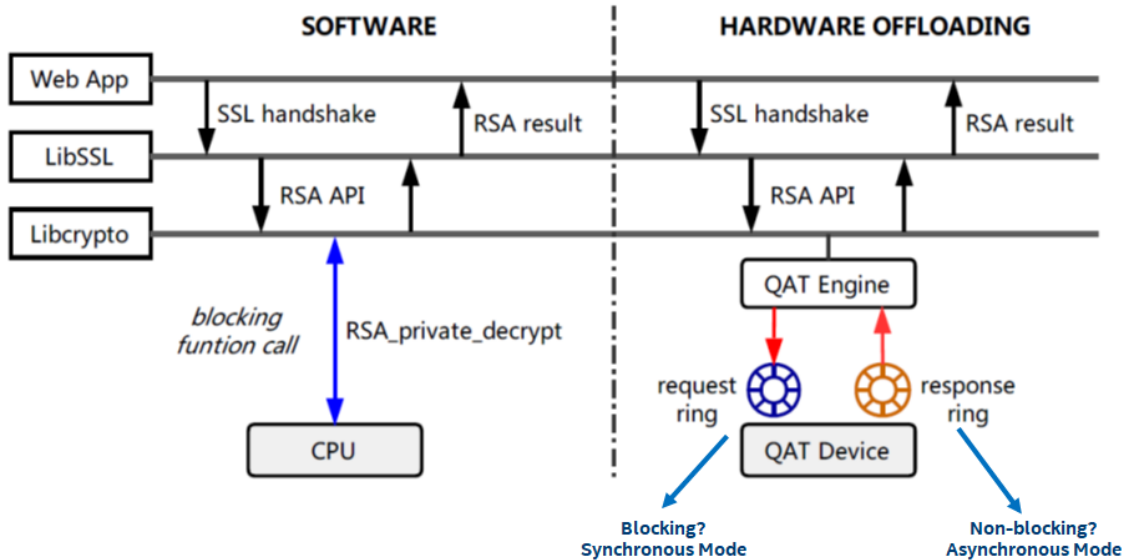
Nginx*是一款由俄罗斯软件工程师Igor Sysoev编写的基于类BSD许可证的高性能HTTP和反向代理web服务器，同时也可提供IMAP/POP3/SMTP服务。自2004年第一版发布以来市场普及率逐年攀升，目前在许多一线互联网公司和IT企业中都得到广泛使用。Nginx架构设计非常灵活，以一个非常微小和简洁的内核搭载核心模块、基础模块、三方模块，并通过文件静态映射和可配置化的指令进行模块协同处理，在HTTP代理、动静分离、负载均衡、虚拟主机、反向代理、缓存提速、授权访问等各个应用场景中都表现出高性能、高并发、低内存的显著优势。



Nginx更多的信息请见：<http://nginx.org>。

Nginx遵循SSL/TLS来加强web站点的访问安全性。Intel推出了基于Intel® QuickAssist Accelerator（后文简称QAT）的硬件offload解决方案，可有效提升web站点的安全访问性能。

该解决方案需要Intel® QAT卡硬件，以及QAT驱动程序、QAT Engine、OpenSSL*、异步Nginx等软件支持，提供了卸载批量SSL请求到专用硬件以及应用层并行异步处理机制以大幅提升性能。



Intel® QAT专有硬件的详细信息可参考：<https://01.org/sites/default/files/downloads/intelr-quickassist-technology/337003-001-intelquickassisttechnologyandopenssl-110.pdf>

系统所需软硬件环境说明：

服务器配置	硬件配置	CPU	Intel® Xeon® PLATINUM 8360Y CPU @ 2.20GHz
		内存	16*32GB DDR4, 3200 MT/s
		硬盘	Intel® SSD S4610, 960G
		网卡	Intel® Ethernet Controller XXV700 25GbE SFP28
		Intel QAT卡	Intel® QuickAssist Accelerator 8970
	软件配置	操作系统	CentOS* 7.8
		内核	3.10.0-1127.el7.x86_64
		Nginx	asynch_mode_nginx v0.4.4
		OpenSSL	v1.1.1i
		Intel QAT driver	QAT v1.7L4.12.0.00011
		Intel QAT Engine	v0.6.4

说明：

- 本文阐述的方案是基于Intel® QuickAssist Accelerator的硬件。内存、硬盘、网卡可视客户用量需求确定。
- 方案中所列软件可参考https://github.com/intel/QAT_Engine。链接中阐述了各软件的最小支持版本，本文所列软件版本均为符合上述要求的软件版本，经过验证测试。
- Asynch_mode_nginx是Intel为支持nginx软硬件加速所使用的优化版nginx，下载后可直接编译使用。相关说明可参照以下链接：https://github.com/intel/asynch_mode_nginx。
- QAT_Engine链接：https://github.com/intel/QAT_Engine。

2. BIOS 设置

BIOS可优化配置项及推荐值如下:

配置项	推荐值
Hyper-Threading	Enable
CPU C6 report	Auto
SpeedStep (Pstates)	Enable
Turbo Mode	Enable
PCIe Link Speed	Gen4
Energy Efficient Turbo	Disable
Boot Performance Mode	Max Performance

3. Linux*优化

1. 设置相应的系统启动参数

```
intel_iommu=off
processor.max_cstates=1 idle=poll pcie_aspm=off
```

2. 关闭cpupower服务

```
systemctl stop cpupower.service
```

3. 关闭防火墙

```
systemctl disable firewalld.service
```

4. 用户进程设置

```
ulimit -c unlimited    #生成core dump
ulimit -n 1000000      #设置最大文件打开句柄数
```

4. Intel® QAT安装和设置

Intel® QuickAssist Accelerator是一块PCIe卡，需要首先将其装入服务器的PCIe插槽中。以下为QAT加速卡的图片



卡安装完毕后，需要确认BIOS设置，将PCIe link speed设置到Gen4，X16，以获取最高性能。

用户可以从以下网址下载Intel® QAT加速卡的驱动，解压缩，配置并安装。本文以/QAT为驱动安装目录：

<https://01.org/intel-quickassist-technology/intel-quick-assist-technology/downloads/intel%C2%AE-quickassist-technology-driver-linux-hw-version-1.7>

```
$cd /  
$mkdir QAT
```

把qat1.7.l.4.12.0-00011.tar.gz拷贝到/QAT目录下

```
$cd /QAT  
$tar xvzf ./ qat1.7.l.4.12.0-00011.tar.gz  
$./configure  
$./make install
```

驱动安装完成后，命令输出中可以看到有3个QAT加速设备（qat_dev）处于启动状态。

然后用如下命令启动QAT服务。

```
$service qat_service start
```

5. OpenSSL*和QAT_Engine安装

1. OpenSSL源代码下载

```
$ cd /usr/local/src  
$ wget https://www.openssl.org/source/openssl-1.1.1i.tar.gz
```

2. OpenSSL编译安装

```
$ tar xvzf openssl-1_1_1i.tar.gz
```

```
$ cd ./openssl-1_1_1i
$ ./config --prefix=/usr/local/ssl -Wl,-rpath, /usr/local/ssl/lib
#本文将openssl安装在/usr/local/ssl
$ make
$ make install
```

3. QAT Engine源代码下载

```
$ export OPENSSL_ENGINES=/usr/local/ssl/lib/engines-1.1
$ cd /usr/local/src
$ git clone https://github.com/intel/QAT_Engine.git
```

4. QAT Engine编译安装

```
$ cd ./QAT_Engine
$ ./autogen.sh
$ ./configure \
    -- with-qat_dir=/QAT \
    -- with-openssl_install_dir=/usr/local/ssl
$ make
$ make install
```

5. QAT服务配置文件替换。

为了使得QAT Engine能和QAT service联合工作，还需要把QAT Engine目录下的实例配置文件拷贝到/etc目录下，替换掉QAT原来的配置文件。

```
$ cd /usr/local/src/QAT_Engine/qat/config/c6xx/multi_process_event-driven_optimized
$ cp * /etc/ (在执行该操作之前，可以将原来QAT的配置文件改名存档，比如c6xx_dev0.conf重命名为c6xx_dev0.conf.bak)
```

在替换配置文件后，重新启动qat_service以加载新的配置文件：

```
$ service qat_service restart
```

用OpenSSL加载QAT Engine:

```
/usr/local/ssl/bin/openssl engine -t -vvvv qatengine
```

6. 可通过openssl speed命令查看使用qat engine前或后的sign和verify信息。

```
$/usr/local/ssl/bin/openssl speed rsa2048
$/usr/local/ssl/bin/openssl speed -engine qatengine -async_jobs 8 rsa2048
```

6. Nginx安装及优化设置

6.1. Nginx安装

1. 完成QAT、OpenSSL、QAT_Engine的安装和配置。

2. Nginx源代码下载

```
$ git clone https://github.com/intel/asynch_mode_nginx.git
```

3. Nginx编译安装

```
$ cd asynch_mode_nginx
$ export NGINX_INSTALL_DIR=<Nginx安装目录>
$ export OPENSSL_LIB=<SSL安装目录>
$ ./configure \
    --prefix=$NGINX_INSTALL_DIR \
    --with-http_ssl_module \
    --add-dynamic-module=modules/nginx_qat_module \
    --with-cc-opt="-DNGX_SECURE_MEM -I$OPENSSL_LIB/include -Wno-error=deprecated-
declarations" \
    --with-ld-opt="-Wl,-rpath=$OPENSSL_LIB/lib -L$OPENSSL_LIB/lib -lz"
$ make
$ make install
```

4. 启动Nginx服务

在完成Nginx相关设置后，在Nginx安装目录下执行 `./sbin/nginx` 命令即可启动Nginx服务。

6.2. Nginx优化设置

1. 生成RSA自签名证书和密钥文件。

```
$ openssl req -x509 -sha256 -nodes -days 365 -newkey rsa:2048 -keyout server.key -out
server.crt -subj "/C=CN/ST=Beijing/L=Beijing/O=Example Inc./OU=Web Security/CN=example.com"
```

在Nginx安装目录下新建cert目录，将server.crt和server.key拷贝到该目录下。

2. 关键nginx配置。在nginx.conf文件中进行如下参数设置：

- 设置工作进程数量和事件模式

```
worker_processes <工作进程数量>; #建议与服务器CPU core数量相同
events {
    use epoll; #epoll是select/poll的增强版本，改进对大批量文件描述符的处理
    ...
}
```

- 加载和设置SSL Engine模块

```
load_module modules/nginx_ssl_engine_qat_module.so;
ssl_engine {
    use_engine qatengine;
    default_algorithms ALL;
    qat_engine {
        qat_offload_mode async;
        qat_notify_mode poll;
        qat_poll_mode heuristic;
    }
}
```



```

        qat_shutting_down_release off;
    }
}

```

- 设置Https Server

```

server {
    listen 443 ssl reuseport so_keepalive=on ...; #如要验证SSL握手性能则设为off
    sendfile on;
    server_name example.com; #与证书CN一致
    ssl_asynch on;
    ssl_session_tickets on; #如要验证SSL握手性能则设为off
    ssl_certificate <Nginx安装目录>/cert/server.crt;
    ssl_certificate_key <Nginx安装目录>/cert/server.key;
    ssl_ciphers AES256-SHA:AES128-SHA;
    #以上可根据客户需要设置其他Cipher Suite以验证不同认证、加解密协议的性能
    ssl_prefer_server_ciphers on;
    ...
}

```

3. nginx.conf 完整的配置样例，该配置适合RSA2K握手性能测试:

```

user root;
worker_processes 8;
load_module modules/ngx_ssl_engine_qat_module.so;
events {
    use epoll;
    worker_connections 8192;
    multi_accept on;
    accept_mutex on;
}

ssl_engine {
    use_engine qatengine;
    default_algorithms ALL;
    qat_engine {
        qat_offload_mode async;
        qat_notify_mode poll;
        qat_poll_mode heuristic;
        qat_shutting_down_release off;
    }
}

http {
    server {
        listen example:443 ssl reuseport backlog=131072 so_keepalive=off rcvbuf=65536
        sndbuf=65536;
        keepalive_timeout 0s;
        ssl_verify_client off;
        ssl_session_tickets off;
        access_log off;
        ssl_asynch on;
        ssl_session_timeout 300s;
        ssl_protocols TLSv1.2;
        ssl_ciphers AES128-SHA;
        ssl_prefer_server_ciphers on;
    }
}

```

```

    ssl_certificate      server2048.crt;
    ssl_certificate_key  server2048.key;
    location / {
        root    html;
        index  index.html index.htm;
    }
}

```

7. 性能测试工具介绍

本文介绍两款web服务器压测工具：wrk 和 ab。

- wrk 是一款针对Http/Https协议的基准测试工具。它能够在单机多核CPU的条件下，使用系统自身的高性能I/O机制，如epoll, kqueue等，通过多线程和事件I/O模式，对目标机器产生大量的负载。

使用示例: (where users would replace the URL with their own Nginx URL)

```
$ wrk -t 10 -c 1000 -d 30S https://example.com:443/index.html
```

说明：命令使用10个线程1000个并发连接取文件，对服务器进行30秒的压测。更多的信息请见：

<https://github.com/wg/wrk> 。

- ab 工具是apache*自带的压力测试工具。它不仅可以对apache服务器进行网站访问压力测试，也可以对或其它类型的服务器进行压力测试。比如Nginx、tomcat、IIS等。用户如果安装了Apache，那么ab工具已经随着Apache一并安装了，如果没有安装Apache，用户可以通过如下命令简单安装（CentOS系统）：

```
$ yum -y install httpd-tools
```

使用示例: (where users would replace the URL with their own Nginx URL)

```
$ ab -n 100000 -c 100 -Z AES128-SHA -f TLS1.2 https://example.com:443/index.html
```

说明：命令使用100个并发发送100000个请求get index文件，对服务器进行压测。更多的信息请见：

<https://httpd.apache.org/docs/2.4/programs/ab.html> 。

8. 意见反馈

我们非常重视您的反馈。如果您对本指南有意见和建议（肯定的建议或不足的改进指正），或正在查找本指南没有包含的内容，请通过后面链接与我们联系：

<https://community.intel.com/t5/Software-Tuning-Performance/bd-p/software-tuning-perf-optimization>

Notices & Disclaimers

Intel technologies may require enabled hardware, software or service activation.

No product or component can be absolutely secure.

Your costs and results may vary.

Code names are used by Intel to identify products, technologies, or services that are in development and not publicly available. These are not "commercial" names and not intended to function as trademarks

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.