卫士通PCIE密码卡B型号性能测评

|  |  |  |
| --- | --- | --- |
| 编写 | 朱林峰 | 2019年02月22日 |
| 审核 |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 变更人 | 变更日期 | 变更项 |
| v1.0 | 朱林峰 | 2019-02-22 | 完成初稿 |
|  |  |  |  |

**目录**

1 测试目的 1

2 测试环境 1

2.1 硬件环境 1

2.2 软件环境 1

3 测试方法 1

3.1 加密卡层测试 1

3.2 protobuf 层测试 2

4 测试数据 2

4.1 加密卡层测试数据 2

4.2 protobuf 层测试数据 3

5 结论 4

测试目的

目前公司各类项目都依赖于卫士通的加密卡硬件进行数据加解密或数字签名，加密卡的性能至关重要，其决定着最后服务的性能极限。在对加密卡的使用过程中，会对加密卡进行多层封装和调用，只有在知道各个封装层次下对应的加密卡的性能，才能发现性能主要损耗在什么环节，并能够有针对性的进行性能优化。

目前加密卡的封装层次分别如下:

1) 卫士通加密卡自带的 C 语言函数调用库，这一层可称为加密卡层。

2) 云京自己使用 C 语言对加密卡函数调用库进行了一层封装，提供调用更加简洁的 protobuf 形式的 API。这一层可称为 protobuf 层。

3) 云京继续使用 java 语言封装 C 语言的 protobuf API，得到能直接通过 java 进行调用的 API。这一层可称为 java 层。

4) 后端服务消费 java 封装的加密卡库，对外提供 REST API。这一层可称为服务层。

目前暂时仅对前两层进行性能测试，一方面得到加密卡层在我们的生产环境中的实际性能值，另一方面测评通过 C 语言 protobuf 进行封装后，其性能损耗情况如何。

后期其他开发者可对第三层和第四层进行性能测试。

测试环境

2.1 硬件环境

本次测试使用的是 ip 为 192.168.20.142 的机器，不清楚需要关心哪些硬件配置，故留待运维进行补全。

2.1.2 加密卡

型号: 卫士通商用PCI-E密码卡，B 卡

SM1 性能: 加解密速率不低于150Mbps

SM2 性能: 签名速率不低于1800次/秒

SM2 性能: 验证速率不低于1300次/秒

SM3 性能: 摘要计算速率不低于270Mbps

SM4 性能: 加解密速率不低于250MGbps

随机数性能: 真随机数产生速率：5Mbps

2.2 软件环境

* 操作系统: centos7
* 内核版本: 3.10.0-862.14.4.el7.x86\_64
* docker: 18.06.1-ce
* GCC: 4.8.5 20150623

测试方法

3.1 加密卡层测试

通过 docker 构建编译环境，在 docker 容器内进行测试。写 C 语言测试用例，直接调用卫士通加密卡调用库，循环调用并计时，测试三轮。测试用例代码示例如下:

int errors = 0;

int counts = 10000;

long start\_timestamp = current\_timestamp();

int i;

for (i = 0; i < counts; i++) {

const char \*data = "abcabcabcabcabcabcabcabcabc";

char out[1024] = {0};

int out\_len = sizeof(out);

int error\_code = ctx\_digest(0, 0, data, strlen(data), out, out\_len);

if (error\_code != YERR\_SUCCESS) errors++;

}

long stop\_timestamp = current\_timestamp();

printf("digest performance test result: \n");

printf("errors: %d\n", errors);

printf("counts: %d\n", counts);

printf("time: %ld\n", stop\_timestamp - start\_timestamp);

3.2 protobuf 层测试

通过编写 C 语言代码，使用 protobuf 对加密卡调用库进行封装，得到一层 protobuf 的 API。在 docker 容器内，编写 C 语言测试用例，循环调用 protobuf API 并计时，测试三轮。测试用例代码示例如下:

uint8\_t out[1024 \* 32] ={0};

int errors = 0;

int counts = 10000;

long start\_timestamp = current\_timestamp();

int i;

for (i = 0; i < counts; i++) {

char \*data = "abcabcabcabcabcabcabcabcabc";

int l = api\_digest(0, 0, data, strlen(data), out);

Response \*response = response\_\_unpack(NULL, l, out);

if (response->code != 0) errors++;

response\_\_free\_unpacked(response, NULL);

}

long stop\_timestamp = current\_timestamp();

printf("digest performance test result: \n");

printf("errors: %d\n", errors);

printf("counts: %d\n", counts);

printf("time: %ld\n", stop\_timestamp - start\_timestamp);

测试数据

4.1 加密卡层测试数据

digest performance test result:

errors: 0

counts: 10000

time: 609

sign performance test result:

errors: 0

counts: 10000

time: 7607

sign performance test result:

errors: 0

counts: 10000

time: 8432

digest performance test result:

errors: 0

counts: 10000

time: 611

sign performance test result:

errors: 0

counts: 10000

time: 7575

sign performance test result:

errors: 0

counts: 10000

time: 8310

digest performance test result:

errors: 0

counts: 10000

time: 610

sign performance test result:

errors: 0

counts: 10000

time: 7601

sign performance test result:

errors: 0

counts: 10000

time: 8339

平均而言:

* 摘要: 16393.5 次每秒
* 签名: 1316.8 次每秒
* 验签: 1196.2 次每秒

4.2 protobuf 层测试数据

digest performance test result:

errors: 0

counts: 10000

time: 657

sign performance test result:

errors: 0

counts: 10000

time: 7627

verify performance test result:

errors: 0

counts: 10000

time: 8469

digest performance test result:

errors: 0

counts: 10000

time: 645

sign performance test result:

errors: 0

counts: 10000

time: 7607

verify performance test result:

errors: 0

counts: 10000

time: 8497

digest performance test result:

errors: 0

counts: 10000

time: 666

sign performance test result:

errors: 0

counts: 10000

time: 7648

verify performance test result:

errors: 0

counts: 10000

time: 8536

平均而言:

* 摘要: 15246.5 次每秒
* 签名: 1311.1 次每秒
* 验签: 1176.4 次每秒

结论

加密卡层性能数据如下:

* 摘要: 16393.5 次每秒
* 签名: 1316.8 次每秒
* 验签: 1196.2 次每秒

protobuf 层性能数据如下:

* 摘要: 15246.5 次每秒
* 签名: 1311.1 次每秒
* 验签: 1176.4 次每秒

经过 protobuf 层封装后性能损耗率如下:

* 摘要: (16393.5 - 15246.5) / 16393.5 = 0.07 = 7%
* 签名: (1316.8 - 1311.1) / 1316.8 = 0.0043 = 0.43%
* 验签: (1196.2 - 1176.4) / 1196.2 = 0.01655 = 1.655%

综合而知，经过 C 语言 protobuf 对加密卡进行封装之后，存在一定的性能损耗，但损耗值在可接受范围之内。而经过 protobuf 进行封装之后，java 或者其他语言调用加密卡将更加简单。

然而实际测试的性能，与厂商声称的性能存在一定的差距。该差距并不是因为对加密卡进行封装而产生的，原因可能如下:

1) 可能实际上加密卡并没有达到其声称的性能。

2) 硬件配置或许还不够高，可能比加密卡厂商测试用的硬件配置低。

3) 测试方法不同，比如可能加密卡厂商能够直接对芯片进行测试，而不需要经过操作系统进行处理。