

# FPGA证券柜台API调优手册

为了提升客户的交易效率，本文档从环境配置，api配置及使用方式等方面给出了不同的优化建议，客户端可根据实际使用场景进行优化，以达到最优交易条件。

## 1、系统环境配置

### 1.1、CPU优化

极速交易场景对CPU延迟与稳定性要求极高，建议进行如下优化：

#### 1.1.1、绑核与NUMA绑定

- **描述：**策略程序应使用和报单网卡同一NUMA节点上的cpu核来设置亲和性，以提升内存和I/O访问效率。若服务器中只有一个NUMA节点，则无需关注此优化。
- **操作：**
  - 执行lscpu命令查看NUMA节点数量及每个NUMA节点下的cpu列表
  - 查看网卡NUMA节点

```
cat /sys/class/net/ens1f0/device/numa_node （以网卡ens1f0为例）
```

假设查询结果为0，则应使用NUMA node0的cpu列表作为待隔核绑核列表

#### 1.1.2、CPU核隔离处理

- **描述：**提前将服务器进行隔核处理，为策略程序的核心业务线程分配专用物理CPU核心，避免被其他进程打扰，假设选取**8，9，10**三个cpu核心进行隔核绑核。
- **操作（以下以redhat为例，若操作失败，请查找相关系统对应操作）：**
  - 编辑 /etc/default/grub 文件：

```
vim /etc/default/grub
```

- 找到 GRUB\_CMDLINE\_LINUX 行在内容最后添加 isolcpus 参数，例如隔离CPU 8-10：

```
GRUB_CMDLINE_LINUX="crashkernel=auto resume=/dev/mapper/rhel-swap rd.lvm.lv=rhel/root rd.lvm.lv=rhel/swap rhgb quiet hugepagesz=1GB hugepages=32 default_hugepagesz=1GB isolcpus=8,9,10"
```

- 更新GRUB配置：

```
sudo update-grub
```

- 重启系统使更改生效

#### 1.1.3、关闭不必要的CPU特性

- **超线程（Hyper-Threading）：**关闭，减少L1/L2资源争用，降低抖动。
  - 重启计算机，进入BIOS/UEFI设置界面
  - 找到"Hyper-Threading"或"HT Technology"选项
  - 将其设置为"Disabled"
  - 保存设置并退出
- **虚拟化（VT-x/AMD-V）：**关闭，减少硬件虚拟化带来的延迟。
  - 重启计算机，进入BIOS/UEFI设置界面
  - 找到虚拟化相关选项（名称可能为）：
    - Intel VT-x (Intel Virtualization Technology)
    - AMD-V (AMD Virtualization)
    - SVM (Secure Virtual Machine)

- VT-d (Intel Virtualization Technology for Directed I/O)
- 将相关选项设置为"Disabled"
- 保存设置并退出
- **性能最大化（BIOS/UEFI设置、操作系统）：**
  - **重启服务器，进入BIOS/UEFI设置界面**（开机时按F2/Del/Esc等，根据主板厂牌）。
  - 常见设置项建议如下：

```
CPU Power Management – 关闭 (Disable)
Intel SpeedStep (EIST) /AMD Cool'n'Quiet – 关闭 (Disable)
C-States/C1E Support – 关闭 (Disable)
Intel Turbo Boost – 开启 (Enable)（极端低延迟下如需完全稳定波动可慎重，通常开启）
Performance/Power Profile/Power Policy – 选择“Maximum Performance”或“Performance”
Package C State limit – 设为C0/C1或最低
Hardware P-states – 选择“Disable”或“Native OS Controlled”
说明：具体名称以你服务器品牌实际显示为准，常见厂商（如Dell、HP、Lenovo、Supermicro等）都有“Performance”或“High Performance”相关预设选项。
```

- **开启睿频（Turbo Boost）：**保持开启，可提升单核性能，但建议监控功耗与热管理。

#### 1.1.4、其他优化（建议）

- 保持CPU频率恒定，关闭CPUIDLE、CPUFREQ动态频率调整。
- 启用高精度定时器（HPET），提高定时准确度。

## 1.2、内存优化

内存性能影响数据访问、网络收包延迟等，建议以下优化措施：

#### 1.2.1、预分配大页内存（HugePages）

- 开启大页内存，以减少页表开销与TLB Miss，提升大数据吞吐性能。
- 方法：linux下通过 `vm.nr_hugepages` 配置，或支持Transparent Hugepage（尽量设为always）。

```
echo 'vm.nr_hugepages = 2147483648' >> /etc/sysctl.conf
sysctl -p
```

- RecvSendDetach=0的情况下API使用了大页内存，需要系统相应的支持

## 2、API使用与配置

api提供的高性能模式有两种，一种是onload模式和tcp direct模式，两种模式所需的环境相同，但api参数配置不同，高性能模式需要高频机器方能达到最优性能。建议使用tcp direct模式，实测效果较onload更优。

### 2.1、配置说明

配置文件中的参数说明

#### 2.1.1、Tcp Direct模式配置

- **SolarflareTradeEthName配置**
  - 配置SolarflareTradeEthName参数为交易链路sfc网卡名称
  - 配置RecvSendDetach=0
  - 配置ApiMode=1
  - 配置OrderWarm=1
  - 配置CpuCore=8,9

### 2.1.2、Onload模式

- 打开RecvSendDetach配置

- 配置 RecvSendDetach=1，此参数开启，意味着用户侧报单发送线程需要绑核，api只需绑定回报线程即可
- 配置 CpuCore = -1, 9，此处仅需绑回报线程即可
- 使用onload启动进程
- 用户线程绑核示例如下

```
#include <sched.h>
int core = 8; //核心号8
cpu_set_t cpus;
CPU_ZERO(&cpus);
CPU_SET(core, &cpus);
auto ret = sched_setaffinity(0, sizeof(cpus), &cpus);
```

### 2.1.3、普通模式

- 关闭RecvSendDetach配置

- 配置 RecvSendDetach=0
- 配置 CpuCore=8,9
- 配置 OrderWarm=1
- 使用onload启动进程

### 2.1.4、UDP交易模式

- 通用UDP配置

- 配置 TradeProtocol=2，使用UDP协议进行交易；
- 配置 RecvSendDetach=1，此参数开启，意味着用户侧报单发送线程需要绑核，api只需绑定回报线程即可；
- 配置 CpuCore = -1, 9，此处仅需绑回报线程即可；
- 配置 ApiMode=0；

- EFVI交易配置：仅适用于Solarflare网卡；

- 配置 TradeProtocol=2，使用UDP协议进行交易；
- 配置 RecvSendDetach=1，此参数开启，意味着用户侧报单发送线程需要绑核，api只需绑定回报线程即可；
- 配置 CpuCore = -1, 9，此处仅需绑回报线程即可；
- 配置 ApiMode=0；
- 配置 SolarflareTradeEthName 参数为交易链路sfc网卡名称；
- 配置 EnableUdpEfviTrade=1；

## 2.2、报单用户侧warm

- api库是无法warm用户侧的，用户可以在空闲时间持续发送用户侧WARM单，以warm用户侧报单路径

- 代码示例

```
CXeleReqOrderInsertField inputField{};
inputField.Direction = XELE_ORDER_WARM; //API识别到XELE_ORDER_WARM会直接返回，不会发送订单
UserApi->reqInsertOrder(inputField, nRequestID++);
```

- 不要在回调函数中进行复杂耗时的业务逻辑处理，否则会阻塞接收线程导致无法接收后续的回调信息
- 尽量使用单线程进行报单，减少线程间的耗时