TD-2038: API CPU affinity guide ••• 193d5520
Changfeng Chen authored 4 minutes ago

api_cpu_affinity_guide.md 3.27 KB

Xele-Trade TraderAPI绑核说明

version 1.0

Xele-Trade TraderAPI, 简称**TraderAPI**, 用于与Xele-Trade进行通信。

本文旨在解答如何设置TraderAPI的工作线程的亲和性(CPU Affinity). 不同版本的TraderAPI, 实现设置亲和性方式不同。

编程环境

Linux x86_64 操作系统

工具函数

获取Linux线程的Linux Thread ID

Linux提供了`gettid`系统调用获取Linux线程的Linux Thread ID, **注意不是`pthread`库的Thread ID。**

```
 #include <sys/types.h>

pid_t gettid(void);
```

应为`gettid`没有对应的glibc包装函数，所以我们要自己包装一下：

```
pid_t getltid(void)
{
  pid_t tid;
  tid = syscall(SYS_gettid);
  return tid;
}
```

配置Linux线程的CPU亲和性

Linux提供了`sched_setaffinity`系统调用配置Linux线程的CPU亲和性

```
#define _GNU_SOURCE             /* See feature_test_macros(7) */
#include <sched.h>

void CPU_ZERO(cpu_set_t *set);

void CPU_SET(int cpu, cpu_set_t *set);

int sched_setaffinity(pid_t pid, size_t cpusetsize,
                      cpu_set_t *mask);
```

TraderAPI 2.0之前

这个时期的TraderAPI只有一个工作线程。设置亲和性的方法就是，在TraderAPI回调函数中(`OnFrontConnected()`除外)获取该线程的**Linux Thread ID(LTID)**, 把LTID和 CPU核心代号传递给相应的系统调用即可.

举例

要把工作线程绑定到CPU核心`3`上,先要获取工作线程的LTID:

```
///用户登录应答
    virtual void OnRspUserLogin(CXeleFtdcRspUserLoginField *pRspUserLogin,
                                CXeleFtdcRspInfoField *pRspInfo,
                                int nRequestID,
                                bool bIsLast) {
```

```
pid_t tid = getltid();
}
```

配置工作线程的CPU亲和性:

```
///用户登录应答
    virtual void OnRspUserLogin(CXeleFtdcRspUserLoginField *pRspUserLogin,
                                CXeleFtdcRspInfoField *pRspInfo,
                                int nRequestID,
                                bool bIsLast) {
pid_t tid = getltid();
cpu_set_t cpus;
CPU_ZERO(&cpus);
CPU_SET(core, &cpus);
sched_setaffinity(0, sizeof(cpus), &cpus);
}
```

TraderAPI 2.0 之后

这个时期的TraderAPI有两个工作线程。设置亲和性的方法就是，调用TraderAPI的接口`RegisterWorkerAffinity()`即可。

```
///注册接口线程亲和性
///@param cores 每个工作线程被分配的处理器核心序号, 有效范围: [0, 系统总核心数-1]
///                目前接口总共两个工作线程:  性能路径线程，吞吐量路径线程
///@param size cores数组的大小，代表需要配置亲和性的线程数
///@return 注册亲和性的结果， 0为成功， 其他为失败
///@remark 亲和性在操作系统中子进程或线程会继承，在接口线程上下文中起新的线程要注意。
///         要在Init()之前调用
virtual int RegisterWorkerAffinity(int *cores, int size)
```

举例

要把工作线程分别绑定到CPU核心`3`和`4`上:

```
CXeleTraderApi *api = CXeleTraderApi::CreateTraderApi();
...
int cores[2];
int core_size = 2;
cores[0] = 3;
cores[1] = 4;
int ret;
ret = api->RegisterWorkerAffinity(cores, core_size);
...
api->Init();
```

引用

- Xele-Trade交易API接口说明
- man 2 gettid
- man 2 sched_setaffinity
- man 2 CPU_ZERO