API入口：

nova\api\openstack\compute\migrate_server.py

MigrateServerController：_migrate_live

拿到的REQUEST：

```
{
"os-migrateLive": {
"block_migration": "false",
"host": "node6"
    }
}
```

Nova_API

1.nova\compute\api.py API：live_migrate

a.生成request_spec.request_destination信息，该信息主要保存指定的hostname值（如果有的话）；

b.清空host_name的值，保证无论是否指定host_name，流程都会通过scheduler调度（如果指定了force参数迁移流程会跳过scheduler）；

2.conductor/api.py ComputeTaskAPI:live_migrate_instance

a.根据API的版本信息选择调用的分支:live_migrate_instance/migrate_server

Nova_Conductor、Nova_Computer

3.conductor/manager.py ComputeTaskManager：live_migrate_instance

a.创建migration表记录；

b.创建迁移任务(LiveMigrationTask)：

c.运行迁移任务，并且捕获迁移过程中的异常；

LiveMigrationTask：

1.选择target，或者对target进行检查（经过Scheduler），如果指定了force参数那么检查不经过scheduler。

生成migrate_data参数（LiveMigrateData）

生成LiveMigrateData时需要target节点和source节点的driver共同作用：

target节点:virt/libvirt/driver.py LibvirtDriver：check_can_live_migrate_destination

1.target磁盘剩余空间、CPU信息比较；

2.创建一个共享文件:迁移前创建一个共享文件（目的是判断CONF.instances_path目录是否是共享目录）

source节点：virt/libvirt/driver.py LibvirtDriver：check_can_live_migrate_source

1. 判断is_volume_backed，指定系统盘是否为cinder盘，block_device_mapping表中如果是cinder盘destination_type字段的

2. 判断is_shared_instance_path，该参数指定了source和target的共享目录是否是共享目录（手段就是判断target创建的临时

3. 判读is_shared_block_storage，指定系统盘是否是一个共享盘，如cinder盘，或者NAS机制的镜像文件，都是共享的

4. 判断block_migration，not（is_shared_instance_path or is_shared_block_storage） 这个参数可以通过API指定，也可

2.调用compute的迁移接口，compute/manager.py ComputeManager：live_migration

compute/manager.py ComputeManager：_do_live_migration ---> 迁移过程的入口函数

```
try:

    ComputeManager：pre_live_migration --> 调用_get_instance_block_device_info获取target信息（iqn，lun-id等），并且刷新到数据库！！！！
        --> LibvirtDriver：pre_live_migration  --->  在target节点为块设备建立连接，并且向source返回新的连接信息。
except:
        _rollback_live_migration :
                1.断开source的磁盘连接信息，因为这时候已经要把磁盘设定为ERROR了！？
                2.调用rollback_live_migration_at_destination：清理磁盘连接，销毁target上的libvirt实例？

                # 这里没有回滚block_device_mapping表的信息

try：
    LibvirtDriver：live_migration(_rollback_live_migration)
except：  Exception
    此处无需要执行_rollback_live_migration，_rollback_live_migration方法作为一个回调函数传递给live_migration，recovery在live_migration内
```