MPI 二维并行化

任广智

December 13, 2019

首先考虑二维 MPI 通信域的建立，此处并不不涉及物理坐标，待通信域建立好之后，再将物理坐标对应进去就可以了。

MPI_DIMS_CREATE 生成笛卡尔拓扑下的进程分布，需要注意的是，当 dim_xy 均为 0 时，系统自动划分进程分布。系统会尽量使每个维度上的进程数接近，而且 dim_xy 将以递减的顺序排序。当 dim_xy 的某个维度为正数时，这个维度不会被修改，系统仅修改其余为 0 的部分。

```
call MPI_DIMS_CREATE(N_proces, 2, dim_xy(0:1), ierr)
```

以下的函数调用生成二维笛卡尔拓扑下的通信子域，并且生成相应的笛卡尔坐标便于使用。

```
call MPI_CART_CREATE(MPI_comm_world, 2, dim_xy(0:1), period_xy(0:1), .false., &
        MPI_comm_cart_xy, ierr)
call MPI_CART_COORDS(MPI_comm_cart_xy, rank, 2, c_rank(0:1), ierr)
```

接下来我们考虑当考虑确切物理坐标以及网格数时候系统给出的二维 MPI 并行化方案。考虑实空间网格为

$$(N_x, N_y, N_z) = (64, 128, 62)$$

的情况，此时对应的经过 P3DFFT 变换之后的谱空间网格应为

$$(N_x, K_y, K_z) = (64, 128, 32)$$

二维网格划分的关系则有

$$(\frac{N_x}{M_1}, \frac{N_y}{M_2}, N_z) \sim (N_x, \frac{N_y}{M1}, \frac{N_z+2}{2M_2}) = (N_x, \frac{K_y}{M1}, \frac{K_z}{M_2})$$

令 M1=dim_xy(1)，M2=dim_xy(2) 我们可以方便地得到每个进程中的网格数。也就是说我们将实空间中 (Nx,Ny)，谱空间中 (Ky,Kz) 分别对应到了 dim_xy 中。考虑 8 进程时候的情况，此时 MPI 给出的 dim_xy=(4,2)，相应的进程分布为：

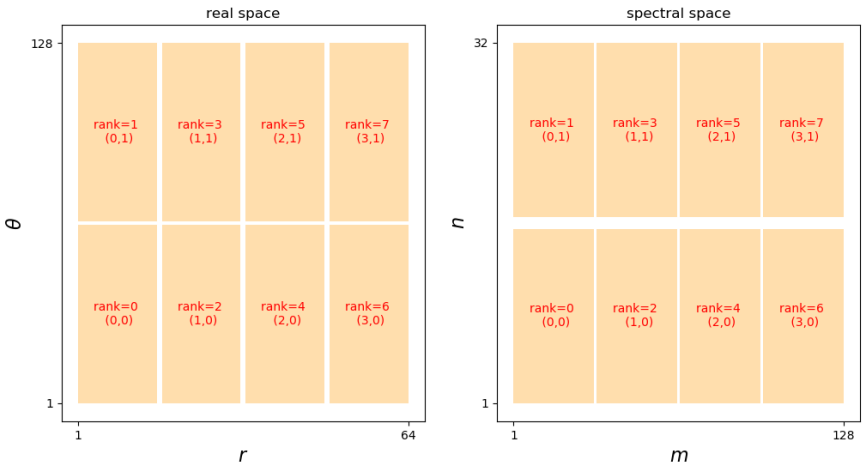

Figure 1:

可以看出，网格的划分符合预期，实空间中径向为四进程，极向二进程，而谱空间中极向模数为四进程，环向模数为二进程。c_rank 每个维度对应 dim_xy 的维度，也就是 c_rank(0) 对应径向，c_rank(1) 对应极向。另外 rank 的排序为列主导，也就是说沿着 c_rank(1) 进行填充，这一点是很重要的。通过上述图片可以比较清楚地看到二维下的进程分布。相邻的进程互相交换数据，这个在程序中通过 MPI_CART_SHIFT 来完成。一维的通信子域的生成通过 MPI_COMM_SPLIT 来完成。

通过上述图像，我们知道内边界进程为 c_rank(0)=0 的进程，而外边界进程为 c_rank(0)=dim_xy(0)-1 的进程。对于需要交换关于圆心对称的内边界进程数据，我们将进程划分映射到极坐标下来观察。8 进程下我们只需要交换进程 0 和进程 1 的边界数据即可。

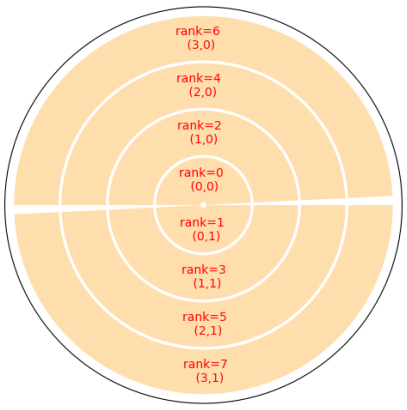

Figure 2:

我们设置 dim_xy=(2,16) 下的情况。可以看到内边界需要交换的数据在 c_rank(1) 的数值上相差 8，即 dim_xy(1)/2。所以关于内边界的需要交换的进程我们定义为 z_rank 时，z_rank 的定义应如下：

```
if(c_rank(0) == 0) then
    if(c_rank(1) <= dim_xy(1)/2-1) then
        z_rank = rank + dim_xy(1)/2
    else
        z_rank = rank - dim_xy(1)/2
    end if
end if
```

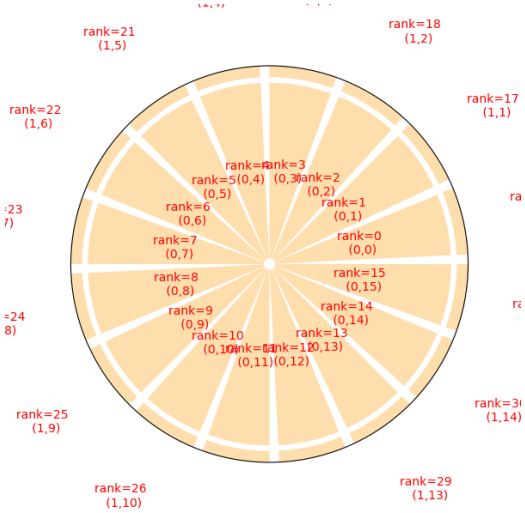

Figure 3: