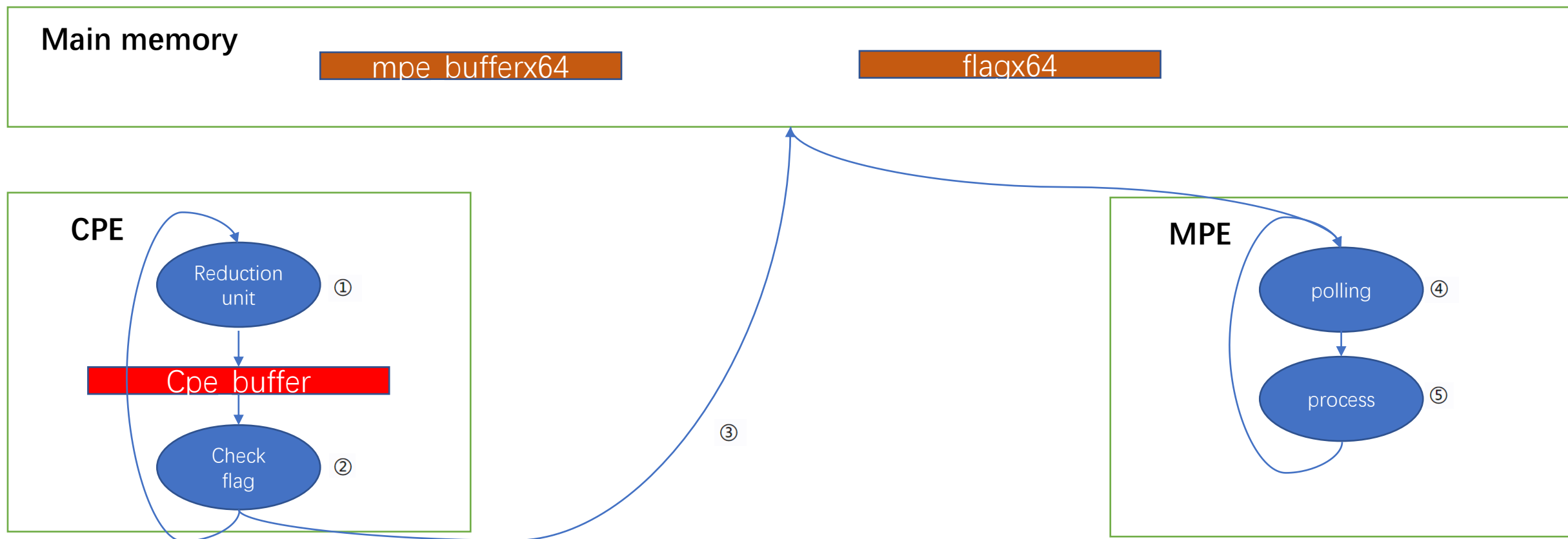


# 主核集中规约&小kernel DMA

罗海文

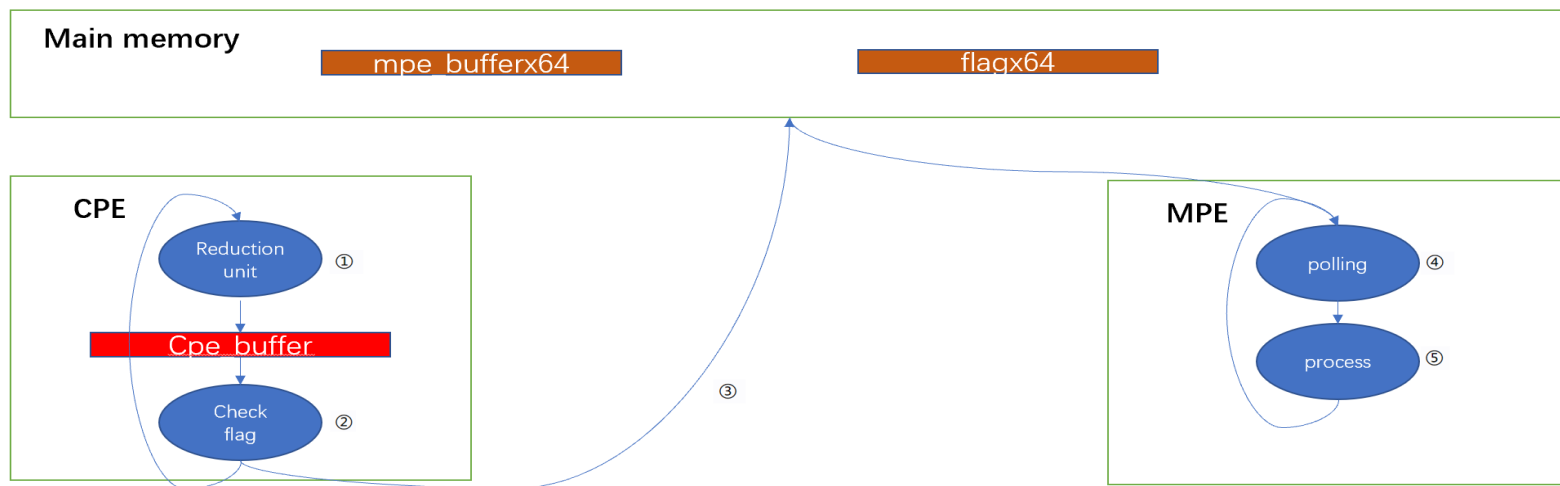
2021.10.18

# 主核集中规约-流程图



`First_order_H[i_place] += first_order_H_dense[i_compute][j_compute];`

# 主核集中规约-文字描述



主存中申请的数组有两个，一个用于接收cpe buffer中的数据，由于有64个从核，所以大小是cpe\_buffer的64倍；一个用于保存64个flag的值，flag用于从核与主核之间的同步。以cpe\_id=0的从核为例，当flag[0]=0时，意味着主核已经处理完了该从核对应的buffer，因此该从核可以向主存中传输cpe\_buffer中的数据，而主核在轮询时不进行规约处理；当flag[0]=1时，意味着该从核已经准备好了数据，主核可以进行规约处理。Cpe\_buffer是ldm上的数组。

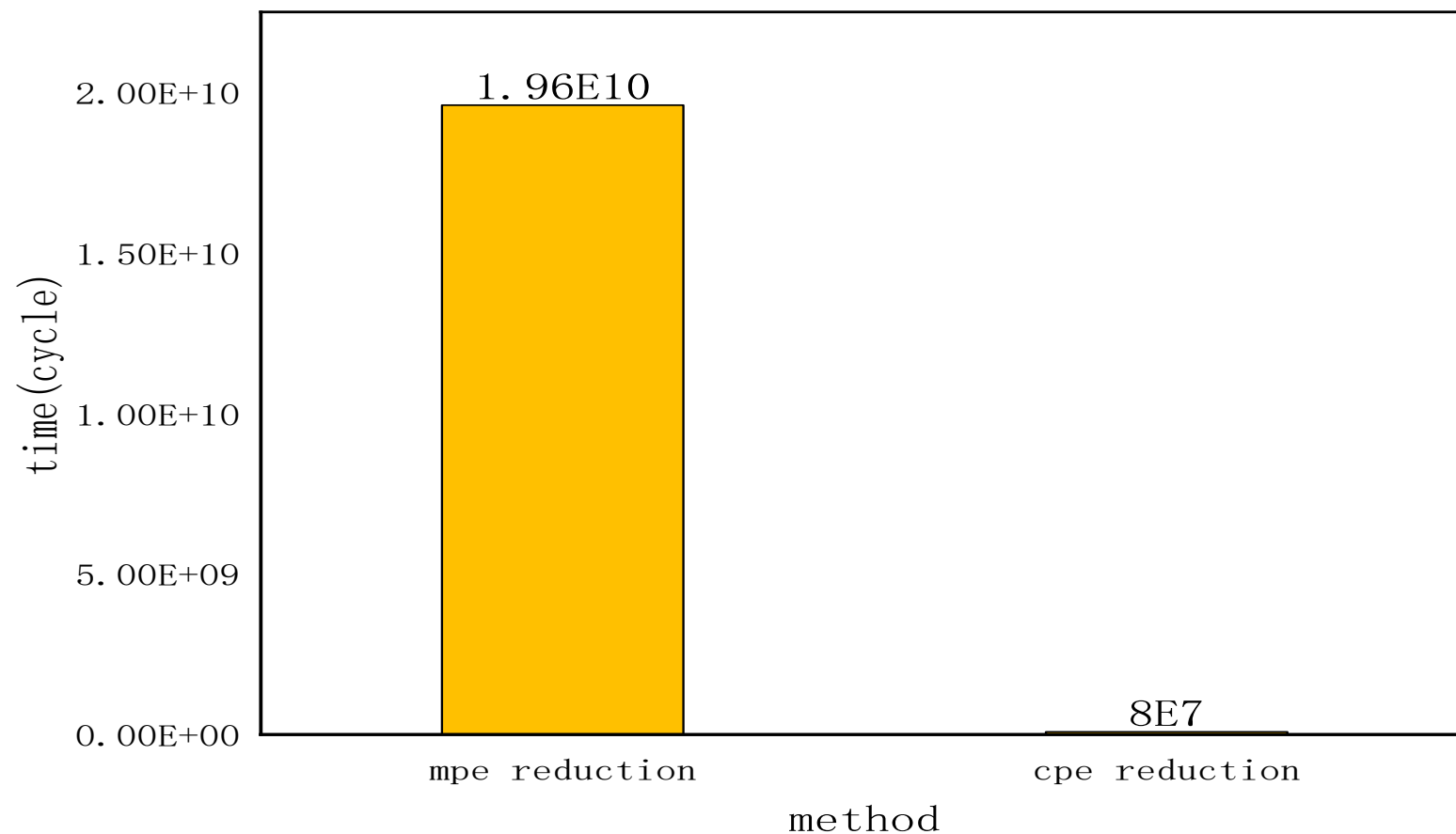
- ①从核的reduction unit获取i\_place和first\_ordre\_H\_dense[i\_compute][j\_compute]的值，暂存到cpe\_buffer中
- ②当cpe\_buffer满时，检查flag的值，如果flag=0，则③将cpe\_buffer中的数据用dma的方式传输到主存，然后将flag置为1；如果flag=1，则等待flag的值变成0，然后执行③的操作。从核重复①②③操作。
- ④查看每个从核对应的flag的值，如果flag的值为1，则⑤执行规约操作，执行完成之后将flag置为0；如果flag的值为0，则不处理，进行在一轮的轮询。主核重复执行④⑤操作。

# 主核集中规约-遇到的bug

```
--Type <return> to continue, or q <return> to quit--
core<51> (stopped) 0x4ffff06ce25c in slave_integrate_first_order_h_dielectric_to_polar_c_()
core<52> (stopped) 0x4ffff06ce25c in slave_integrate_first_order_h_dielectric_to_polar_c_()
core<53> (stopped) 0x4ffff06ce25c in slave_integrate_first_order_h_dielectric_to_polar_c_()
core<54> (stopped) 0x4ffff06ce25c in slave_integrate_first_order_h_dielectric_to_polar_c_()
core<55> (stopped) 0x4ffff06ce25c in slave_integrate_first_order_h_dielectric_to_polar_c_()
core<56> (stopped) 0x4ffff06ce25c in slave_integrate_first_order_h_dielectric_to_polar_c_()
core<57> (stopped) 0x4ffff06ce25c in slave_integrate_first_order_h_dielectric_to_polar_c_()
core<58> (stopped) 0x4ffff06ce25c in slave_integrate_first_order_h_dielectric_to_polar_c_()
core<59> (stopped) 0x4ffff06ce25c in slave_integrate_first_order_h_dielectric_to_polar_c_()
core<60> (stopped) 0x4ffff06ce25c in slave_integrate_first_order_h_dielectric_to_polar_c_()
core<61> (stopped) 0x4ffff06ce25c in slave_integrate_first_order_h_dielectric_to_polar_c_()
core<62> (stopped) 0x4ffff06ce25c in slave_integrate_first_order_h_dielectric_to_polar_c_()
core<63> (stopped) 0x4ffff06ce25c in slave_integrate_first_order_h_dielectric_to_polar_c_()
(swpdb) t cg 0 slave 8
[Switching to thread 3 (CG<0>PE<8>)]
#0 0x00004ffff06cdcd0 in slave_integrate_first_order_h_dielectric_to_polar_c_ (partition_t
lattice_vector=<optimized out>, coords_center=0x500063253d40, first_order_H_sparse=<opt
r_grid_min=0x5000597b03c0, log_r_grid_inc=0x5000597b0000, basis_wave_ordered=0x5000634d
batches_work_size=0x5000597e2d80, centers_basis_integrals=0x5000637f1d80, batches_work_
Cbasis_to_center=0x500063f878a0, species_center=0x500063265720, basis_fn_start_spl=0x50
center_to_atom=0x500063268620, basis_fn=0x50006377a900, basis_l=0x50005efel280, Cbasis_
write_buffer_to_memory=0x500058666950, cpes_done=0x500058666a50, buffer_size=0x50005866
inspectors_sparse_from_dense=0x5000597e3500, batches_work_i_basis=0x5000597e3200, mpi_i
n_centers_integrals=<optimized out>, prune_basis_once=0x3ff2c, n_centers_basis_i=0x3ff3
n_atoms=0x3ff40, n_centers=0x3ff44, n_max_spline=0x3ff48, l_ylm_max=0x3ff4c, n_max_comp
first_order_potential=0x500065236820, j_coord=1) at integrate_first_order_H_polar_c.c:6
670 while(write_buffer_to_memory[_MYID]==1); //wait mpe process
(swpdb) p write_buffer_to_memory[_MYID]
$1 = 0
(swpdb) █
```

- 64个从核有两个卡住了，使用swpdb调试打印出flag的值是0，但是程序却卡在了while(flag==1);这条语句。

# 主核集中规约-单个从核性能提升



# 小kernel cache换DMA- tab\_atom\_centered\_coords

- Loop tiling

- 外层循环通过调度把任务分配给从核

- 对内层循环进行分块，由于内层循环中需要访问数组

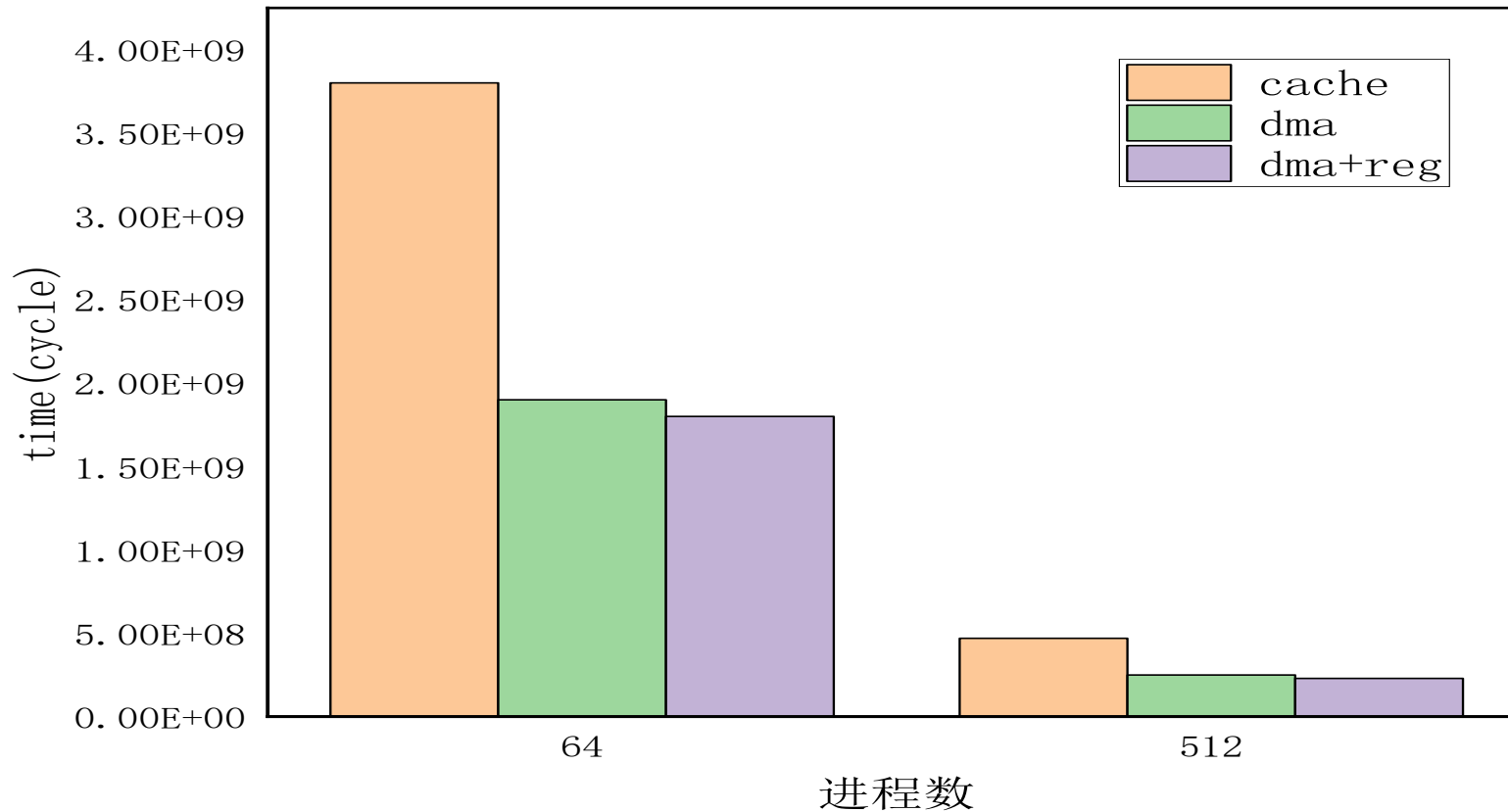
dir\_tab, dir\_tab\_index, dist\_tab\_sq的大小都是n\_centers\_integrals，所以分块的大小为MAX\_ALLOCATABLE/5

- 充分利用寄存器

- 每一次内层循环dir\_tab都需要写入一次和读入一次，可以先把dir\_tab的值存在寄存器中，写入dir\_tab之后需要读入时可以直接读寄存器中的值

```
for(int i_point=0;i_point<n_point;i_point++)
{
    for(int i_center=0;i_center<n_centers_integrals;i_center++)
    {
        dir_tab(0,i_center,i_point)=coord_current(0)-dir_tab_index(0,i_center);
        dir_tab(1,i_center,i_point)=coord_current(1)-dir_tab_index(1,i_center);
        dir_tab(2,i_center,i_point)=coord_current(2)-dir_tab_index(2,i_center);
        dist_tab_sq(i_center,i_point)=
            dir_tab(2,i_center,i_point)* dir_tab(2,i_center,i_point)
            +dir_tab(2,i_center,i_point)*dir_tab(2,i_center,i_point)
            +dir_tab(2,i_center,i_point)*dir_tab(2,i_center,i_point);
    }
}
```

# 小kernel cache换DMA- tab\_atom\_centered\_coord性能提升



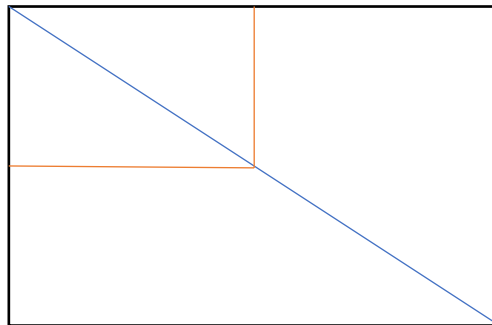
# 小kernel cache换DMA-prune\_density\_matrix\_sparse

- **loop tiling**

- 第一层循环通过调度把任务分配给从核
- 每一个任务中i\_basis\_index可以一直保留在ldm中
- 对column\_index和density\_matrix\_sparse进行分块，分块大小为i\_start-i\_end。

- **对称矩阵的写入**

- 先把结果暂存在ldm中，然后通过dma写入内存，列使用跨步dma，行使用连续dma



```
for(i=0;i<n_compute_c;i++)
{
    i_basis=i_basis_index(i);
    i_start=index_Hamiltonian(1,1,i_basis);
    i_end=index_Hamiltonian(2,1,i_basis);

    for(j=0;j<i;j++)
    {
        j_basis=i_basis_index(j);
        for(k=i_start;k<i_end;k++)
        {
            if(column_index(k)==j_basis)
            {
                density_matrix_con(i,j)=
                    density_matrix_sparse(k);
                density_matrix_con(j,i)=
                    density_matrix_sparse(k);
            }
        }
    }
}
```



# 小kernel cache换DMA-prune\_density\_matrix\_sparse性能提升

