Track B: Particle Methods – Part 3

PRACE Spring School 2012

Maciej Cytowski (ICM UW)

CAPACITIES

e-infrastructure

PART 3: Exchange of particles

- Computing interactions between particles ~ 30 min
- Hands-on Zoltan neighbourhood assignment ~ 1 h

Computing interations between particles

- **We consider only short range interactions**
- **For each particle:**
  - Compute its distance to each other local particle
  - Compute forces if two particles are close enough
  - Determine which processors own geometry that intersects the neighbourhood of the particle
  - Send particle to other processors based on the export list
  - Compute distance between particle and all remote particles
  - Compute forces if two particles are close enough
  - Send results back

Computing interations between particles

Gravitational like forces between two close particles

```
/* This function computes forces between two given particles */
int force(struct particle_data *p1,struct particle_data *p2) {

  float dist;

  /* Compute distance between particles */
  dist = sqrt(pow(p1->position.x-p2->position.x,2)+
              pow(p1->position.y-p2->position.y,2)+
              pow(p1->position.z-p2->position.z,2));

  /* Only short range interactions */
  if(dist<epsilon && dist>0.01) {
    p1->force.x-=G*(p1->position.x-p2->position.x)/pow(dist,3);
    p1->force.y-=G*(p1->position.y-p2->position.y)/pow(dist,3);
    p1->force.z-=G*(p1->position.z-p2->position.z)/pow(dist,3);
  }

  return 0;
}
```

Computing interactions

```
/* This function computes interactions between all particles in the
        simulation */
int compute_forces(){

  int i,j,k;
  float dist;
  int numprocs;
  int procs[size];
  struct export_list_data export_list[4*lnp];
  int nexp=0;

  /* Compute local interaction */
  for(i=0;i<lnp;i++) {
    for(j=0;j<lnp;j++) {
      if(i==j) continue;
      force(&particles[i],&particles[j]);
    }
  }

  compute_remote_forces(nexp,export_list);

  return 0;

}
```

Computing local interactions

Computing remote interactions

Computing remote interactions

Remote interactions = interactions with particles that belong to other processes

```
int compute_remote_forces(int nexp, struct export_list_data *export_list);
```

nexp – number of particles to be exported
export_list – list of particles to be exported together with process numbers

```
struct export_list_data{
  int particle;
  int proc;
};
```

Exercise 4 – construct the export list for the compute_remote_forces() function.

Computing remote interactions

Please look at compute_remote_forces() function.

1. Preparation of the export buffers (one buffer for each process) with the use of export_list.
2. Send information about message sizes – implemented with MPI_Alltoall() function
3. Send and receive particles – implemented with MPI_Sendrecv() function
4. Compute interaction between imported particles and all particles that belongs to the process
5. Send and receive results – implemented with MPI_Sendrecv() function
6. Update forces

compute_remote_forces()

```
int compute_remote_forces(int nexp,struct export_list_data *export_list) {

  int i,j,k;
  struct particle_data export[size][2*lnp];
  struct particle_data import[size][2*lnp];
  int nexport[size];
  int nimport[size];
  float dist;
  MPI_Status status;

  if(nexp==0) return 0;

  for(i=0;i<size;i++) {
    nexport[i]=0;
    nimport[i]=0;
  }
  /* Prepare export buffers */
  for(i=0;i<nexp;i++) {
    int p=export_list[i].proc;
    export[p][nexport[p]]=particles[export_list[i].particle];
    export[p][nexport[p]].force.x=0.0;
    export[p][nexport[p]].force.y=0.0;
    export[p][nexport[p]].force.z=0.0;
    nexport[p]++;
  }
```

Export and import buffers

Preparation of the export buffers (one buffer for each process) with the use of export_list

compute_remote_forces()

```
/* Send information about message sizes */
MPI_Alltoall(nexport,1,MPI_INT,nimport,1,MPI_INT,MPI_COMM_WORLD);

/* Send and receive particles */
for(i=0;i<size;i++) {
  if(i==rank) continue;
  MPI_Sendrecv(&export[i],nexport[i]*sizeof(struct
   particle_data),MPI_BYTE,i,rank,&import[i],nimport[i]*sizeof(struct
   particle_data),MPI_BYTE,i,i,MPI_COMM_WORLD,&status);
}
/* Compute remote interaction */
for(i=0;i<size;i++) {
  for(j=0;j<nimport[i];j++) {
    for(k=0;k<lnp;k++) {
      force(&import[i][j],&particles[k]);
    }
  }
}
/* Send and receive results */
for(i=0;i<size;i++) {
  if(i==rank) continue;
  MPI_Sendrecv(&import[i],nimport[i]*sizeof(struct
   particle_data),MPI_BYTE,i,rank,&export[i],nexport[i]*sizeof(struct
   particle_data),MPI_BYTE,i,i,MPI_COMM_WORLD,&status);
}
```

Send information about message sizes

Send and receive particles

Compute interaction between imported particles and all particles that belongs to the process

Send and receive results

compute_remote_forces()

```
   for(i=0;i<size;i++) {
    nexport[i]=0;
  }
  /* Update forces */
  for(i=0;i<nexp;i++) {
    int p=export_list[i].proc;
    particles[export_list[i].particle].force.x+=export[p][nexport[p]].force.x;
    particles[export_list[i].particle].force.y+=export[p][nexport[p]].force.y;
    particles[export_list[i].particle].force.z+=export[p][nexport[p]].force.z;
    nexport[p]++;
  }

  return 0;

}
```

Update forces

Remarks

- Both local and remote interactions are computed with the use of **force()** function
- Different interactions and simulation results can be produced by modifications to **force()** function and adjusting **epsilon** parameter
- Resulting codes can have different parallel performance (e.g. scalability)

How to prepare the export list?

- export_list – list of particles to be exported to other processes
- In order to prepare the export_list each process needs to run a loop over all local particles and:
  - determine possible intersections with other processes' geometry
  - add particles to the export list if the intersection was determined

How to determine possible intersections?

Zoltan box assignment

```
int Zoltan_LB_Box_Assign (struct Zoltan_Struct * zz, double xmin, double
ymin, double zmin, double xmax, double ymax, double zmax, int *procs,
int *numprocs);
```

- Given a geometric decomposition of space (currently only RCB, RIB, and HSFC are supported), and given an **axis-aligned box** around the geometric object, **Zoltan_LB_Box_PP_Assign** determines which processors and parts own geometry that intersects the box
- To use this routine, the parameter **KEEP_CUTS** must be set to TRUE when the decomposition is generated

Hands-on data exchange – Exercise 4

- Uncomment the compute_forces() and move() functions in main.c
- Execute the code – only local interactions are computed
- Copy and visualize the results

- Implement the preparation of the export_list
- Execute the code – all interactions are computed
- Copy and visualize the results

- Increase the number of steps (e.g.128)