Split Molecular Dynamics

Aiichiro Nakano

Collaboratory for Advanced Computing & Simulations
Department of Computer Science
Department of Physics & Astronomy
Department of Chemical Engineering & Materials Science Science
University of Southern California

Email: anakano@usc.edu

Goal: Learn MPI communicator concept using in situ data analysis of molecular dynamics simulation
MPI_Comm_split(

MPI_Comm mothercomm, daughtercomm;
int color, key;
MPI_Comm_split(mothercomm, color, key, &daughtercomm);

• MPI_Comm_split() subdivides a communicator, mothercomm, into a set of daughter communicators, where processes of the same color belong to the same daughter communicator. Processes within each color are ranked according to key, or if key is the same, according to the rank in mothercomm. It returns a pointer to a daughter communicator, daughtercomm, to which the process belongs.

• MPI_Comm_split() is a simpler, more high-level function to construct communicators, instead of using MPI_Comm_create() combined with MPI_Gorup_excl() or MPI_Group_incl().
• Split MPI_COMM_WORLD into two communicators; one performs molecular dynamics (MD) simulation, whereas the other analyzes simulation data on the fly in background.

```c
int gid, sid, md;
MPI_Comm workers;
MPI_Comm_rank(MPI_COMM_WORLD, &gid);  // Global rank
md = gid % 2;  // = 1 (MD workers) or 0 (analysis workers)
MPI_Comm_split(MPI_COMM_WORLD, md, 0, &workers);
MPI_Comm_rank(workers, &sid);  // Rank in workers
```

Run as mpirun –np 2× nproc

# of processors needed for MD (specified in pmd.h)
Analysis: Velocity Probability Density

\( P(\mathbf{v}) \): Probability density function of atom velocity \( \mathbf{v} \)

```c
#define VMAX 5.0  // Max. velocity value to construct a velocity histogram
#define NBIN 100   // # of bins in the histogram

void calc_pv() {
    double lpv[NBIN], pv[NBIN], dv, v;
    int i;

    dv = VMAX/NBIN;   // Bin size
    for (i=0; i<NBIN; i++) lpv[i] = 0.0;  // Reset local histogram
    for (i=0; i<n; i++) {
        v = sqrt(pow(rv[i][0], 2)+pow(rv[i][1], 2)+pow(rv[i][2], 2));
        lpv[v/dv < NBIN ? (int)(v/dv) : NBIN-1] += 1.0;
    }
    MPI_Allreduce(lpv, pv, NBIN, MPI_DOUBLE, MPI_SUM, workers);
    MPI_Allreduce(&n, &nglob, 1, MPI_INT, MPI_SUM, workers);
    for (i=0; i<NBIN; i++) pv[i] /= (dv*nglob);  // Normalization
    if (sid == 0) {
        for (i=0; i<NBIN; i++) fprintf(fpv, "%le \%le\n", i*dv, pv[i]);
        fprintf(fpv, "\n");
    }
}
```
Main Program: Initialization

```c
init_params();
if (md) {
    set_topology();
    init_conf();
    atom_copy();
    compute_accel();
}
else
    if (sid == 0) fpv = fopen("pv.dat","w");
```

- All processes read input parameters, `init_params()`. The `nproc` processes of MD workers (`md == 1`) perform MD initialization tasks, whereas only rank 0 among the other `nproc` analysis workers (`md == 0`) opens a file to output the calculated velocity probability density function.
Main Program: Main MD Loop

for (stepCount=1; stepCount<=StepLimit; stepCount++) {
    if (md) single_step();
    if (stepCount%StepAvg == 0) {
        if (md) {
            Send # of atoms, \( n \), to rank gid-1 in MPI_COMM_WORLD
            Send velocities of \( n \) atoms to rank gid-1 in MPI_COMM_WORLD
            eval_props();
        }
        else {
            Receive # of atoms, \( n \), from rank gid+1 in MPI_COMM_WORLD
            Receive velocities of \( n \) atoms from rank gid+1 in MPI_COMM_WORLD
            calc_pv();
        }
    }
}

- MD workers perform MD simulation. Every \( \text{stepAvg} \) steps, MD workers send their atom velocities to corresponding analysis workers (i.e., those with the same ranks in respective daughter communicators). Upon receiving the velocities, analysis workers calculate the velocity probability density function.
Main Program: Finalization

```c
if (md && sid == 0)
    printf("CPU & COMT = %le %le\n", cpu, comt);
if (!md && sid == 0)
    fclose(fpv);
```

- **Rank 0 of MD workers** reports the computing & communication times, whereas **rank 0 of analysis workers** closes the probability density output file.

- **Finally:** Change all `MPI_COMM_WORLD`’s in the original MD functions to workers ~ it’s only a matter in the small MD world!

---

**Diagram:**

- **Time**
- **Message**
- **MD**
- **Analysis**
Results

Graphs showing the distribution of $P(v)$ for different values of $t$: $t = 0.05$, $t = 0.10$, $t = 0.15$, $t = 0.20$, and $t = 0.25$. The graphs are labeled with steps 10, 20, and 30.
Maxwell-Boltzmann Distribution

\[ P_{\text{Maxwell-Boltzmann}}(v) = \frac{4}{\sqrt{\pi}} \left( \frac{m}{2k_B T} \right)^{3/2} v^2 \exp \left( -\frac{mv^2}{2k_B T} \right) \]