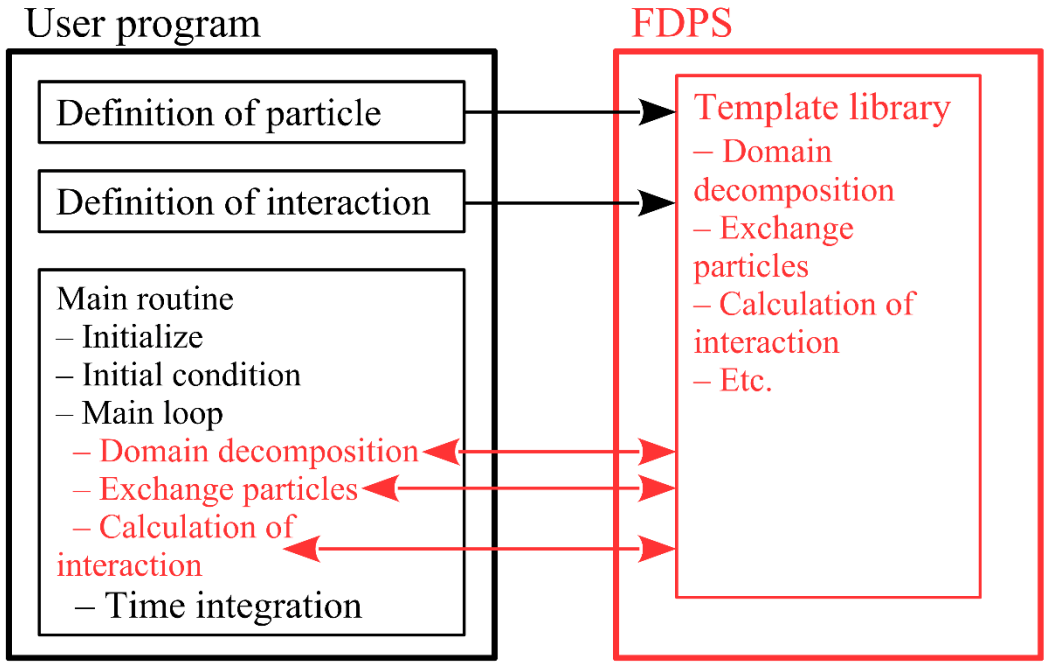
FDPS ~Framework for Developing Particle Simulator~

- C++ header library to develop parallelized particle simulation code (e.g. N-body, molecular dynamics)
- FDPS APIs provide dynamic load balancing, communication among nodes, and force calculation
- Users only define particles class, interaction kernel, and particle integration
- Fortran interface and accelerator are available

FDPS ~ Framework for Developing Particle Simulator~



Iwasawa et al., PASJ, 68, 54, 2016.

Available at
https://github.com/FDPS/FDPS

Example of FDPS code

```
00 #include <particle_simulator.hpp>
01 using namespace PS;
02
03 class FP{
04 public:
05   F64vec r,v,f;
06   void clear(){r = v = f = 0.0;}
07   void copyFromForce(const FP& fp){f = fp.f;}
08   F64 getRSearch() const {return 4.5;}
09   F64vec getPos() const {return r;}
10   void setPos(const F64vec &_r){r = _r;};
11   void copyFromFP(const FP &fp){(*this) = fp;}
12 };
13
14 struct Kernel{
15   void operator()(const FP *epi,const S32 ni,
16                   const FP *epj,const S32 nj,
17                   FP *force){
18     for(S32 i=0; i<ni; i++){
19       F64vec ri = epi[i].r; F64vec fi = 0.0;
20       for(S32 j=0; j<nj; j++){
21         const F64vec rij = ri - epj[j].r;
22         const F64 r2 = rij * rij;
23         if(r2==0.0 || r2>4.5*4.5) continue;
24         const F64 r2i = 1.0/r2;
25         const F64 r6i = r2i * r2i * r2i;
26         fi += r6i*(48.0*r6i-24.0)*r2i * rij;
27       }
28       force[i].f = fi;
29     }
30   }
31 };
32
33
34 int main(int argc,char **argv){
35   Initialize(argc,argv);
36   ParticleSystem<FP> ps;
37   ps.initialize();
38   if(Comm::getRank()==0){
39     ps.setNumberOfParticleLocal(1000);
40     S32 count = 0;
41     for(S32 x=0;x<10;x++)
42       for(S32 y=0;y<10;y++)
43         for(S32 z=0;z<10;z++){
44           ps[count].r = F64vec(x,y,z)-5.0;
45           ps[count++].v = 0.0;
46         }
47   }else ps.setNumberOfParticleLocal(0);
48   DomainInfo di;
49   di.initialize(0.3);
50   di.decomposeDomainAll(ps);
51   ps.exchangeParticle(di);
52   TreeForForceShort<FP,FP,FP>::Scatter t;
53   t.initialize(1000,0.0,64,256);
54   t.calcForceAllAndWriteBack(Kernel(),ps,di);
55   S32 nl = ps.getNumberOfParticleLocal();
56   const F64 dt = 0.005; const F64 dth = 0.5*dt;
57   for(int s=0;s<1000;s++){
58     for(int i=0;i<nl;i++) ps[i].v+=ps[i].f*dth;
59     for(int i=0;i<nl;i++) ps[i].r+=ps[i].v*dt;
60     di.decomposeDomainAll(ps);
61     ps.exchangeParticle(di);
62     nl = ps.getNumberOfParticleLocal();
63     t.calcForceAllAndWriteBack(Kernel(),ps,di);
64     for(int i=0;i<nl;i++) ps[i].v+=ps[i].f*dth;
65   }
66   Finalize();
67 }
```

67-line code is compiled to hybrid MPI /OpenMP molecular dynamics simulation program