```
!-----------------------------------------------------------------------
! Skeleton 2D Electrostatic GPU PIC code
! written by Viktor K. Decyk, UCLA
      program gpufpic2
      use fgpupush2
      use fgpulib2
      use fgpufft2
      use push2_h
      implicit none
      integer, parameter :: indx =   9, indy =   9
      integer, parameter :: npx =  3072, npy =   3072
      integer, parameter :: ndim = 2
      real, parameter :: tend = 10.0, dt = 0.1, qme = -1.0
      real, parameter :: vtx = 1.0, vty = 1.0, vx0 = 0.0, vy0 = 0.0
      real :: ax = .912871, ay = .912871
! idimp = dimension of phase space = 4
      integer :: idimp = 4, ipbc = 1
      real :: wke = 0.0, we = 0.0, wt = 0.0
! sorting tiles
      integer :: mx = 16, my = 16
! fraction of extra particles needed for particle management
      real :: xtras = 0.2
! declare scalars for standard code
      integer :: np, nx, ny, nxh, nyh, nxh1, nxe, nye, nxeh, nxyh, nxhy
      integer :: mx1, my1, mxy1, ntime, nloop, isign
      real :: qbme, affp
      real, dimension(1) :: sum
!
! declare scalars for GPU code
      integer :: nblock = 128
! nscache = (0,1,2) = (no,small,big) cache size
      integer :: nscache = 1
      integer :: mmcc, nppmx, nppmx0, ntmax, npbmx
      integer :: nxhd
      integer, dimension(1) :: irc
!
! declare arrays for standard code
      real, dimension(:,:), pointer :: part
      complex, dimension(:,:), pointer :: ffct
      integer, dimension(:), pointer :: mixup
      complex, dimension(:), pointer :: sct
!
! declare arrays for GPU code
      real, device, dimension(:,:), allocatable :: g_qe
      real, device, dimension(:,:,:), allocatable :: g_fxye
      complex, device, dimension(:,:), allocatable :: g_ffct
      integer, device, dimension(:), allocatable :: g_mixup
      complex, device, dimension(:), allocatable :: g_sct
      complex, device, dimension(:,:), allocatable :: g_q, g_qt
      complex, device, dimension(:,:,:), allocatable :: g_fxy, g_fxyt
      real, device, dimension(:), allocatable :: g_wke, g_we
      real, device, dimension(:,:,:), allocatable :: g_ppart, g_ppbuff
      integer, device, dimension(:), allocatable :: g_kpic
      integer, device, dimension(:,:), allocatable :: g_ncl
      integer, device, dimension(:,:,:), allocatable :: g_ihole
      real, device, dimension(:), allocatable :: g_sum
      integer, device, dimension(:), allocatable :: g_irc
      complex, dimension(:,:), pointer :: qt
      complex, dimension(:,:,:), pointer :: fxyt
      real, dimension(:,:,:), pointer :: ppart
      integer, dimension(:), pointer :: kpic
!
! declare and initialize timing data
      real :: time
      integer, dimension(4) :: itime
      double precision :: dtime
      real :: tdpost = 0.0, tguard = 0.0, tfft = 0.0, tfield = 0.0
      real :: tpush = 0.0, tsort = 0.0
!
! initialize scalars for standard code
      np = npx*npy; nx = 2**indx; ny = 2**indy; nxh = nx/2; nyh = ny/2
      nxh1 = nxh + 1; nxe = nx + 2; nye = ny + 1; nxeh = nxe/2
      nxyh = max(nx,ny)/2; nxhy = max(nxh,ny)
      mx1 = (nx - 1)/mx + 1; my1 = (ny - 1)/my + 1; mxy1 = mx1*my1
      nloop = tend/dt + .0001; ntime = 0
      qbme = qme
      affp = real(nx*ny)/real(np)
! set size for FFT arrays
      nxhd = nxh1
!
! allocate and initialize data for standard code
      allocate(part(idimp,np))
      allocate(ffct(nyh,nxh))
      allocate(mixup(nxhy),sct(nxyh))
      allocate(kpic(mxy1))
      allocate(qt(ny,nxh1),fxyt(ny,ndim,nxh1))
!
! set up GPU
      irc = 0
      call fgpu_setgbsize(nblock)
      call init_cuf(0,irc(1))
      if (irc(1) /= 0) then
         write (*,*) 'CUDA initialization error!'
         stop
      endif
! obtain compute capability
      mmcc = fgetmmcc()
      if (mmcc < 20) then
         write (*,*) 'compute capability 2.x or higher required'
         stop
      endif
! set cache size
      call fgpu_set_cache_size(nscache)
!
! allocate data for GPU code
      allocate(g_qe(nxe,nye),g_fxye(ndim,nxe,nye))
      allocate(g_ffct(nyh,nxh),g_mixup(nxhy),g_sct(nxyh))
      allocate(g_q(nxhd,ny),g_qt(ny,nxh1))
      allocate(g_fxy(nxhd,ndim,ny),g_fxyt(ny,ndim,nxh1))
      allocate(g_wke(mxy1),g_we(nxh1))
      allocate(g_sum(1))
!
! prepare fft tables
      call WFFT2RINIT(mixup,sct,indx,indy,nxhy,nxyh)
! prepare NVIDIA ffts
      call fgpufft2rrcuinit(nx,ny,ndim)
      call fgpufft2cuinit(nx,ny,ndim)
! calculate form factors
      isign = 0
      call POIS22T(qt,fxyt,isign,ffct,ax,ay,affp,we,nx,ny,nxh1,ny,nxh,  &
     &nyh)
! copy in solver arrays to GPU
      g_mixup = mixup
      g_sct = sct
      g_ffct = ffct
! initialize electrons
      call DISTR2(part,vtx,vty,vx0,vy0,npx,npy,idimp,np,nx,ny,ipbc)
!
! find number of particles in each of mx, my tiles: updates kpic, nppmx
      call DBLKP2L(part,kpic,nppmx,idimp,np,mx,my,mx1,mxy1,irc)
      if (irc(1) /= 0) then
         write (*,*) 'DBLKP2L error, irc=', irc
         stop
      endif
! allocate vector particle data
      nppmx0 = (1.0 + xtras)*nppmx
      ntmax = xtras*nppmx
      npbmx = xtras*nppmx
! align data to warp size
      nppmx0 = 32*((nppmx0 - 1)/32 + 1)
      ntmax = 32*(ntmax/32 + 1)
      npbmx = 32*((npbmx - 1)/32 + 1)
!
      allocate(g_ppart(nppmx0,idimp,mxy1))
      allocate(g_ppbuff(npbmx,idimp,mxy1))
      allocate(g_kpic(mxy1))
      allocate(g_ncl(8,mxy1),g_ihole(2,ntmax+1,mxy1))
      allocate(g_irc(1))
      allocate(ppart(nppmx0,idimp,mxy1))
!
! copy ordered particle data for GPU code: updates ppart and kpic
      call PPMOVIN2LT(part,ppart,kpic,nppmx0,idimp,np,mx,my,mx1,mxy1,irc&
     &)
      if (irc(1) /= 0) then
         write (*,*) 'PPMOVIN2LT overflow error, irc=', irc
         stop
      endif
! sanity check
      call PPCHECK2LT(ppart,kpic,idimp,nppmx0,nx,ny,mx,my,mx1,my1,irc)
      if (irc(1) /= 0) then
         write (*,*) 'PPCHECK2LT error: irc=', irc
         stop
      endif
! copy to GPU
      g_irc = irc
      g_ppart = ppart
      g_kpic = kpic
!
! * * * start main iteration loop * * *
!
  500 if (nloop <= ntime) go to 2000
!     write (*,*) 'ntime = ', ntime
!
! deposit charge with GPU code: updates g_qe
      call dtimer(dtime,itime,-1)
      call fgpu_zfmem(g_qe,nxe*nye)
      call fgpu2ppost2l(g_ppart,g_qe,g_kpic,qme,nppmx0,idimp,mx,my,nxe, &
     &nye,mx1,mxy1)
      call dtimer(dtime,itime,1)
      time = real(dtime)
      tdpost = tdpost + time
!
! add and copy guard cells with GPU code: updates g_q
      call dtimer(dtime,itime,-1)
      call fgpucaguard2l(g_q,g_qe,nx,ny,nxe,nye,nxhd,ny)
      call dtimer(dtime,itime,1)
      time = real(dtime)
      tguard = tguard + time
!
! transform charge to fourier space with GPU code: updates g_q, g_qt
      call dtimer(dtime,itime,-1)
      isign = -1
      call fgpuwfft2rcs(g_q,g_qt,isign,g_mixup,g_sct,indx,indy,nxhd,ny, &
     &nxhy,nxyh)
! NVIDIA fft
!     call fgpufft2rrcu(g_q,g_qt,isign,indx,indy,nxhd,ny)
      call dtimer(dtime,itime,1)
      time = real(dtime)
      tfft = tfft + time
!
! calculate force/charge in fourier space with GPU code:
! updates g_fxyt, g_we
      call dtimer(dtime,itime,-1)
      call fgpupois22t(g_qt,g_fxyt,g_ffct,g_we,nx,ny,nxh1,ny,nxh,nyh)
      call dtimer(dtime,itime,1)
      time = real(dtime)
      tfield = tfield + time
!
! transform force to real space with GPU code: updates g_fxy, g_fxyt
      call dtimer(dtime,itime,-1)
      isign = 1
      call fgpuwfft2rcsn(g_fxy,g_fxyt,isign,g_mixup,g_sct,indx,indy,ndim&
     &,nxhd,ny,nxhy,nxyh)
! NVIDIA fft
!     call fgpufft2rrcun(g_fxy,g_fxyt,isign,indx,indy,ndim,nxhd,ny)
      call dtimer(dtime,itime,1)
      time = real(dtime)
      tfft = tfft + time
!
! copy guard cells with GPU code: updates g_fxye
      call dtimer(dtime,itime,-1)
      call fgpuccguard2l(g_fxy,g_fxye,nx,ny,nxe,nye,nxhd,ny)
      call dtimer(dtime,itime,1)
      time = real(dtime)
      tguard = tguard + time
!
! push particles with GPU code:
      call dtimer(dtime,itime,-1)
! updates g_ppart, g_wke
      call fgpuppush2l(g_ppart,g_fxye,g_kpic,qbme,dt,g_wke,idimp,nppmx0,&
     &nx,ny,mx,my,nxe,nye,mx1,mxy1,ipbc)
! updates g_ppart, g_ncl, g_ihole, g_wke, g_irc
!     call fgpuppushf2l(g_ppart,g_fxye,g_kpic,g_ncl,g_ihole,qbme,dt,    &
!    &g_wke,idimp,nppmx0,nx,ny,mx,my,nxe,nye,mx1,mxy1,ntmax,g_irc)
      call dtimer(dtime,itime,1)
      time = real(dtime)
      tpush = tpush + time
!
! reorder particles by tile with GPU code:
      call dtimer(dtime,itime,-1)
! updates g_ppart, g_ppbuff, g_kpic, g_ncl, g_ihole, and g_irc
      call fgpuppord2l(g_ppart,g_ppbuff,g_kpic,g_ncl,g_ihole,idimp,     &
     &nppmx0,nx,ny,mx,my,mx1,my1,npbmx,ntmax,g_irc)
! updates g_ppart, g_ppbuff, g_kpic, g_ncl, and g_irc
!     call fgpuppordf2l(g_ppart,g_ppbuff,g_kpic,g_ncl,g_ihole,idimp,    &
!    &nppmx0,mx1,my1,npbmx,ntmax,g_irc)
      call dtimer(dtime,itime,1)
      time = real(dtime)
      tsort = tsort + time
!
! sanity check
      irc = g_irc
      if (irc(1) /= 0) then
         write (*,*) 'push or reorder error: ntmax, irc=', ntmax, irc
         stop
      endif
!
! energy diagnostic
      if (ntime==0) then
         call fgpu_zfmem(g_sum,1)
         call fgpusum2(g_we,g_sum,nxh1)
         we = g_sum(1)
         call fgpu_zfmem(g_sum,1)
         call fgpusum2(g_wke,g_sum,mxy1)
         wke = g_sum(1)
         write (*,*) 'Initial Field, Kinetic and Total Energies:'
         write (*,'(3e14.7)') we, wke, wke + we
      endif
      ntime = ntime + 1
      go to 500
```

```
 2000 continue
!
! * * * end main iteration loop * * *
!
      write (*,*) 'ntime = ', ntime
! energy diagnostic
      call fgpu_zfmem(g_sum,1)
      call fgpusum2(g_we,g_sum,nxh1)
      we = g_sum(1)
      call fgpu_zfmem(g_sum,1)
      call fgpusum2(g_wke,g_sum,mxy1)
      wke = g_sum(1)
      write (*,*) 'Final Field, Kinetic and Total Energies:'
      write (*,'(3e14.7)') we, wke, wke + we
!
      write (*,*)
      write (*,*) 'deposit time = ', tdpost
      write (*,*) 'guard time = ', tguard
      write (*,*) 'solver time = ', tfield
      write (*,*) 'fft time = ', tfft
      write (*,*) 'push time = ', tpush
      write (*,*) 'sort time = ', tsort
      tfield = tfield + tguard + tfft
      write (*,*) 'total solver time = ', tfield
      time = tdpost + tpush + tsort
      write (*,*) 'total particle time = ', time
      wt = time + tfield
      write (*,*) 'total time = ', wt
      write (*,*)
!
      wt = 1.0e+09/(real(nloop)*real(np))
      write (*,*) 'Push Time (nsec) = ', tpush*wt
      write (*,*) 'Deposit Time (nsec) = ', tdpost*wt
      write (*,*) 'Sort Time (nsec) = ', tsort*wt
      write (*,*) 'Total Particle Time (nsec) = ', time*wt
      write (*,*)
!
! close down NVIDIA fft
      call fgpufft2cudel()
      call fgpufft2rrcudel()
! deallocate memory on GPU
      deallocate(g_irc,g_ihole,g_ncl,g_kpic,g_ppbuff,g_ppart)
      deallocate(g_sum,g_we,g_wke,g_fxyt,g_fxy,g_qt,g_q)
      deallocate(g_sct,g_mixup,g_ffct,g_fxye,g_qe)
! close down GPU
      call end_cuf()
!
      stop
      end program
```