Main Points

- **Main Phases of PIC codes**
- **Approach to Parallelization**
- **Implementation Details**
- **Performance Results**
- **Conclusion**

Main Phases of PIC codes

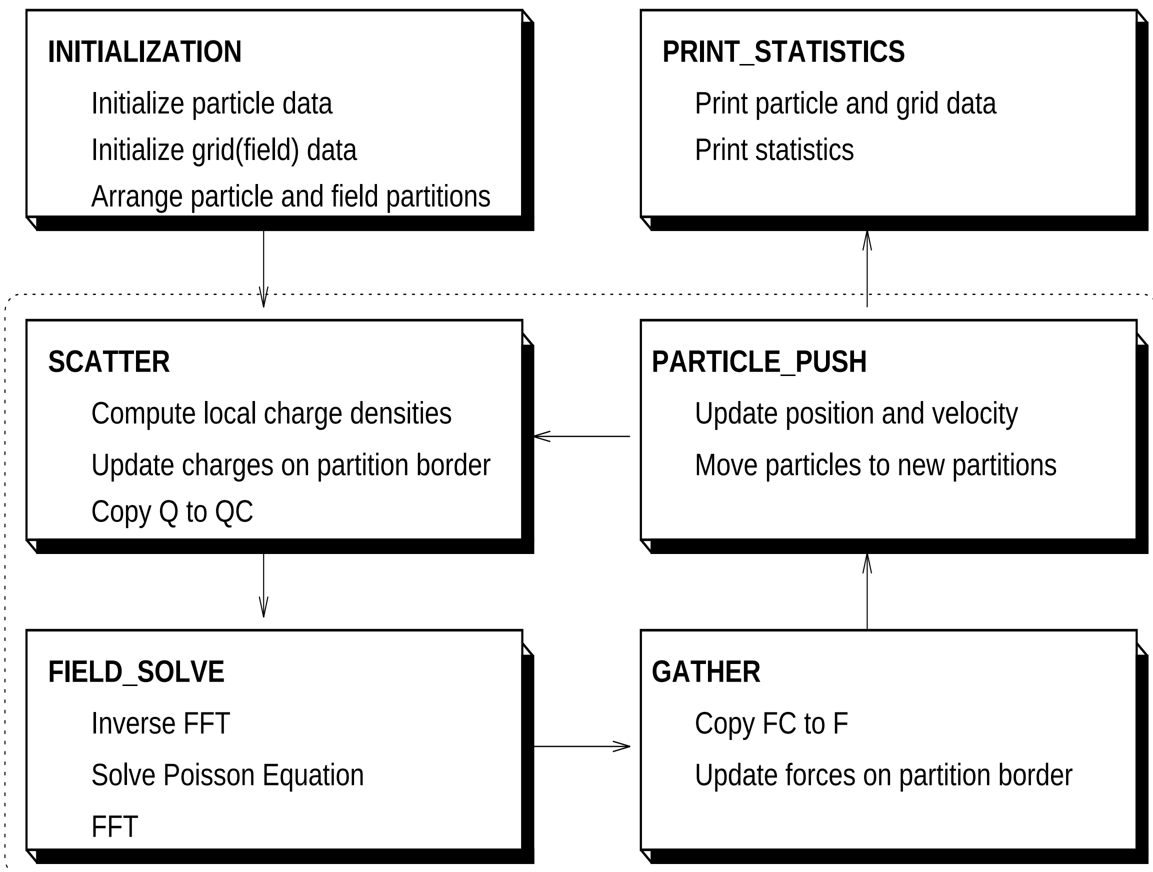

Figure 1: Main phases of a typical PIC code.

Approach to Parallelization

- **Decomposition of Principal Data Structures**
- **Loop Level Parallelization(FORALL and INDEPENDENT DO)**
- **Communication Optimizations(CSHIFT)**
- **Passing Distributed Data to Subroutines(Descriptive Directive)**
- **Extrinsic Procedures**
- **Compiler Restrictions(Nonlinear Dependency)**

Decomposition of Principal Data Structures

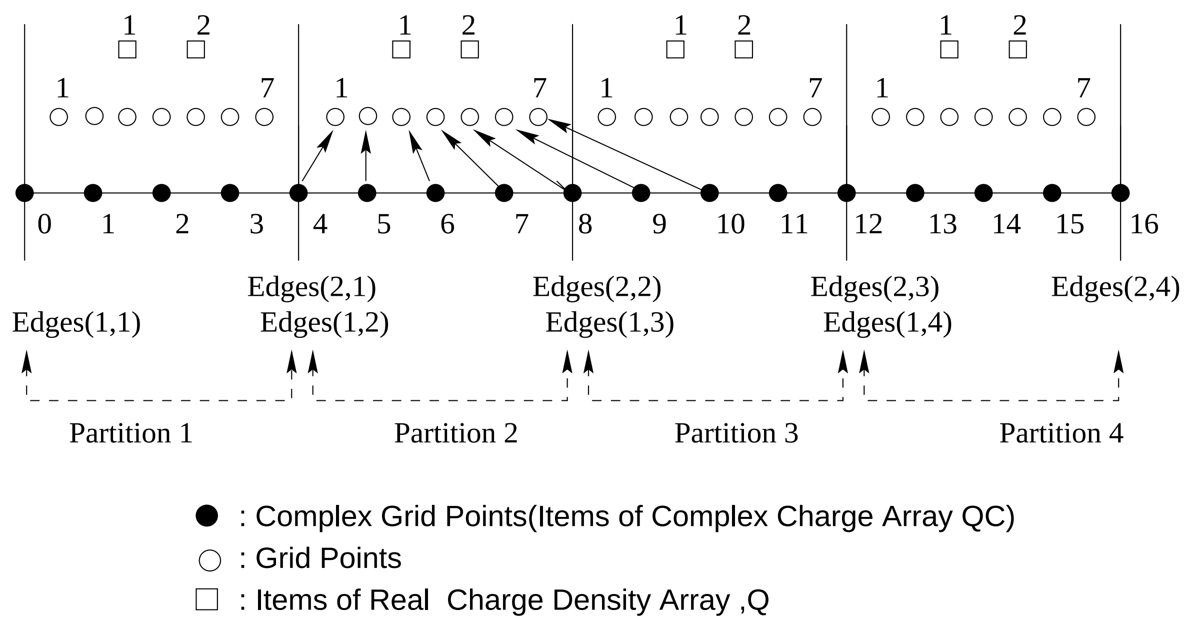

Figure 2: Illustration of particle and grid decompositions (Eulerian Decomposition) and associated charge densities at each grid point and interaction among them.

- **Particle Array (and Particle Partition(PP))**
- **Grid Array (and Field Partition(FP))**
- **Guard Points**

Extrinsic Procedures

We wrote :

- **Fast Fourier Transform (FFT)**
- **Each Nested Loop**

as Extrinsic Subroutine

The version of hpf compiler we used could not have parallelized some of loops in PIC code. Therefore we used extrinsic subroutines to make parallel these type of loops.

We will give example about how we wrote these extrinsic functions later.

Implementation Details

- **How did we write down extrinsic functions**
- **Scatter Phase**
- **Field Solve Phase**
- **Gather Phase**
- **Particle Push Phase**

How did we write down extrinsic functions

We distributed particles and fields onto processors statically. **PART** holds the information of particles and Q the charge densities on grid points.

- DIMENSION PART(DIM=2,NUMBER OF PARTICLES, NUMBER OF BLOCKS)
- DIMENSION Q(NUMBER OF GRID POINTS,NUMBER OF BLOCKS)

We distributed these arrays in a block-wise of last dimension. That is ,

- **!HPF$ DISTRIBUTE (*,BLOCK) :: Q**
- **!HPF$ DISTRIBUTE (*,*,BLOCK) :: PART**

Here are how we constructed our extrinsic functions:

- **Analyze the nested loop to find the list of variables used there or computed**
- **Build one extrinsic function whose body is the body of loop and whose parameters are those found at previous step**

ONE EXTRINSIC FUNCTION EXAMPLE

Here is one nested loop to compute charge densities on grid points

```
DO k = 1, number_of_particle_partitions
      DO i = 1, number_of_particles_in_partition
            dx = part(POS,j,k) -⌊part(POS,j,k) + 0.5⌋
            n = ⌊ part(POS,j,k) +0.5 ⌋ -noff(k) +2
            q(n, k) = q(n,k) + qm*(0.75-dx)^2
            q(n+1, k) = q(n+1,k) +0.5*qm*(0.5+dx)^2
            q(n-1, k) = q(n-1,k) +0.5*qm*(0.5-dx)^2
      END DO
END DO
```

Figure 3: Calculating the particle charge densities at each grid point.

Here is the extrinsic function to compute above nested loop in parallel

```
      Subroutine Scatter1(part,q,noff)
      Dimension part(idimp,npmax,nblok),q(nxpmx,nblok),noff(nblok)
!HPF$ Distribute *(*,*,block) :: part
!HPF$ Distribute *(*,block) :: q
!HPF$ Distribute *(block) :: noff
!HPF$ INDEPENDENT, NEW(k,i,n, dx)
k = 1
DO i = 1, number_of_particles_in_partition
      dx = part(POS,j,k) -⌊part(POS,j,k) + 0.5⌋
      n = ⌊ part(POS,j,k) +0.5 ⌋ -noff(k) +2
      q(n, k) = q(n,k) + qm*(0.75-dx)^2
      q(n+1, k) = q(n+1,k) +0.5*qm*(0.5+dx)^2
      q(n-1, k) = q(n-1,k) +0.5*qm*(0.5-dx)^2
EDNDO
```

Figure 4: Calculating the particle charge densities at each grid point.

Scatter Phase

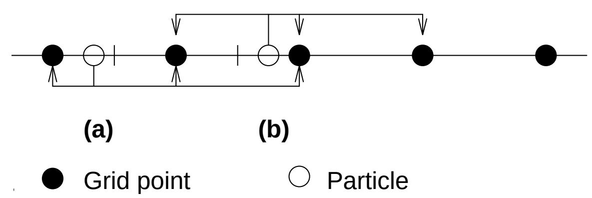

Figure 5: For a one-dimensional simulation, interaction of a particle (a) on the lower half of a grid interval; (b) on the upper half of a grid interval with the neighboring grid points.Inpedendent Do or Extrinsic function

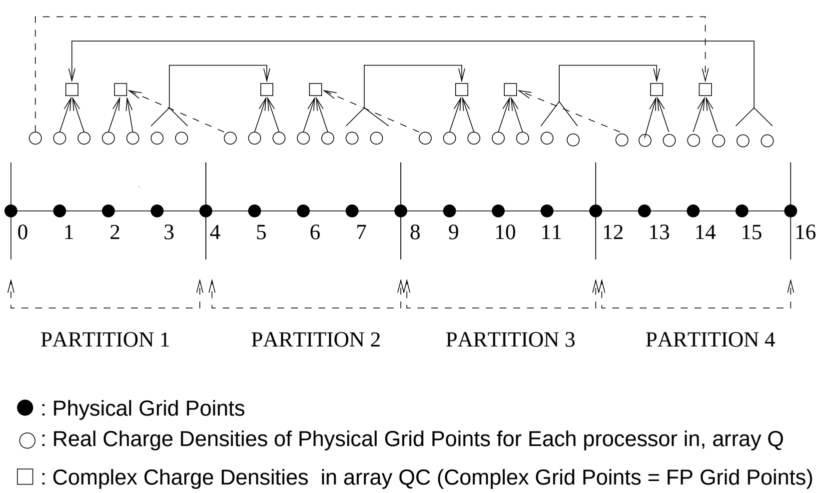

Figure 6: Copying data from real charge density to complex charge density. Inpedendent Do or Extrinsic Function

Particle Push Phase

- **Each processor fills its send buffers,SbufR and SbufL**
- **Each processor also calculates the number of outgoing particles in each send buffer**
- **Using CSHIFT function,shift these send buffers and number of outgoing particles to neighbors**
- **Each processor deposits the appropriate particles with position between its boundaries ,if any,and constructs again from others ,if any, the send buffers**
- **Repeat above cycle until each particle reaches into appropriate region**

Above steps,except third one, can be calculated in local.

- **We wrote extrinsic function for each step above,except third one**
- **For third one, we use CSHIFT function to shift trough third dimension**

Performance Results

We got approximately same results as MPL version of PIC code with HPF implementation. Here are the speedup graphics of 1D and 2D PIC codes for both HPF and MPL versions.

Here are the timing results for 1D and 2D PIC codes for MPL and HPF versions If we looked the tables following, we can see also every subroutine of PIC code approximately shows linear speedup

Phase	**Subphase**	**Number of Processors**					
		1	2	4	8	16	32
Scatter	Phase 1	2.910	1.450	0.730	0.360	0.188	0.094
	Phase 2	0.001	0.004	0.003	0.003	0.005	0.007
Field Solve	IFFT	0.123	0.075	0.045	0.030	0.023	0.024
	Poisson	0.003	0.003	0.003	0.002	0.003	0.004
Gather	Copy	0.001	0.004	0.003	0.003	0.005	0.007
Particle Push	Push	4.880	1.930	0.970	0.480	0.248	0.136
	Migrate	0.000	0.800	0.410	0.210	0.115	0.070
Loop Iteration Time		1818.0	999.0	513.0	261.0	144 .0	94.5

Table 1: Timing Results(in seconds) for 1D PIC code in HPF using 4505600 Particles and 16K Grid Points.

Phase	**Subphase**	**Number of Processors**					
		1	2	4	8	16	32
Scatter	Phase 1	2.910	1.450	0.730	0.360	0.189	0.090
	Phase 2	0.001	0.004	0.003	0.003	0.005	0.007
Field Solve	FFT	0.123	0.075	0.029	0.030	0.029	0.009
	Poisson	0.003	0.003	0.003	0.002	0.005	0.004
Gather	Copy	0.001	0.004	0.003	0.003	0.005	0.007
Particle Push	Push	4.880	1.930	0.980	0.490	0.250	0.136
	Migrate	0.00	0.81	0.40	0.20	0.10	0.07
Loop Iteration Time		1818.0	1004.0	505.6	263.8	132.1	85.5

Table 2: Timing Results(in seconds) for 1D PIC code in MPL using 4505600 Particles and 16K Grid Points.

Phase	Subphase	Number of Processors				
		1	4	8	16	32
Scatter	Phase 1	5.450	1.300	0.650	0.329	0.167
	Phase 2	0.0009	0.004	0.004	0.004	0.005
Field Solve	FFT	0.005	0.015	0.009	0.009	0.015
	Poisson	0.009	0.003	0.003	0.003	0.004
Gather	Copy	0.0009	0.0040	0.0040	0.0040	0.0050
Particle Push	Push	8.150	1.890	0.950	0.486	0.251
	Migrate	0.000	0.380	0.230	0.178	0.204
Loop Iteration Time		4453.8	1196.0	624.0	325.0	24 7.0

Table 3: Timing Results(in seconds) for 2D PIC code in HPF using 3571712 Particles and 128 * 256 Grid Points.

Phase	**Subphase**	**Number of Processors**					
		1	2	4	8	16	32
Scatter	Phase 1	5.45	2.66	1.32	0.66	0.33	0.15
	Phase 2	0.000	0.000	0.009	0.009	0.004	0.005
Field Solve	FFT	0.005	0.025	0.009	0.009	0.009	0.015
	Poisson	0.009	0.009	0.009	0.009	0.003	0.004
Gather	Copy	0.000	0.009	0.009	0.009	0.004	0.005
Particle Push	Push	8.150	3.820	1.880	0.940	0.479	0.250
	Migrate	0.000	0.689	0.370	0.180	0.100	0.200
Loop Iteration Time		4453.8	2368.6	1206.5	614.2	303.0	237.9

Table 4: Timing Results(in seconds) for 2D PIC code in MPL using 3571712 Particles and 128 * 256 Grid Points.

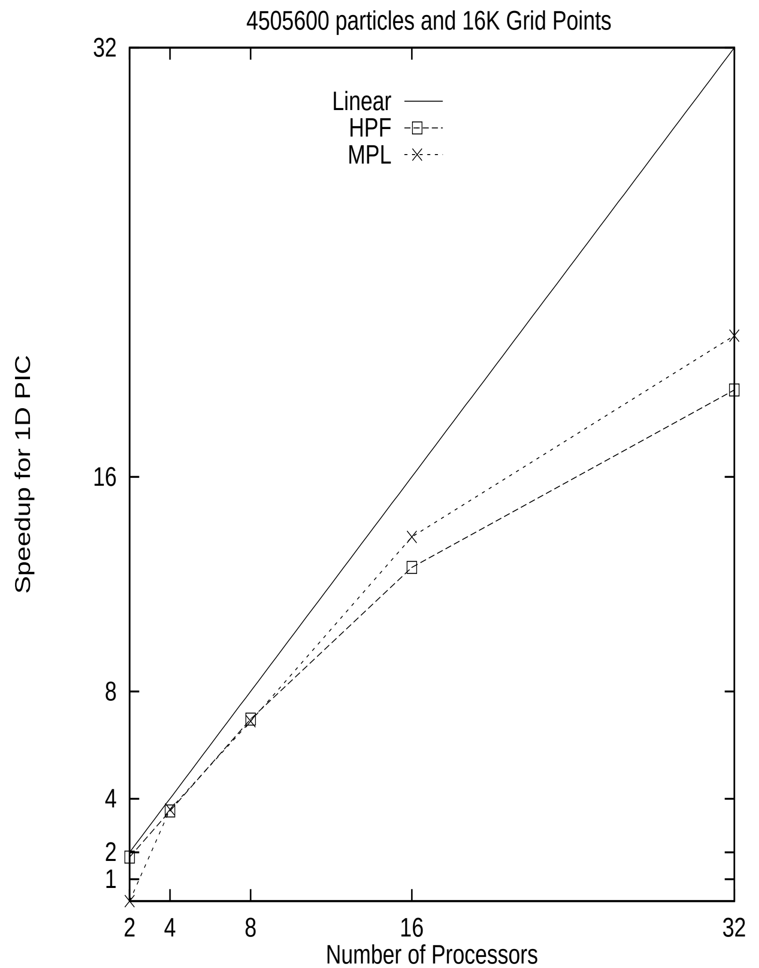

Figure 7: Comparison of Speedup Graphics of 1D MPL and HPF PIC Codes

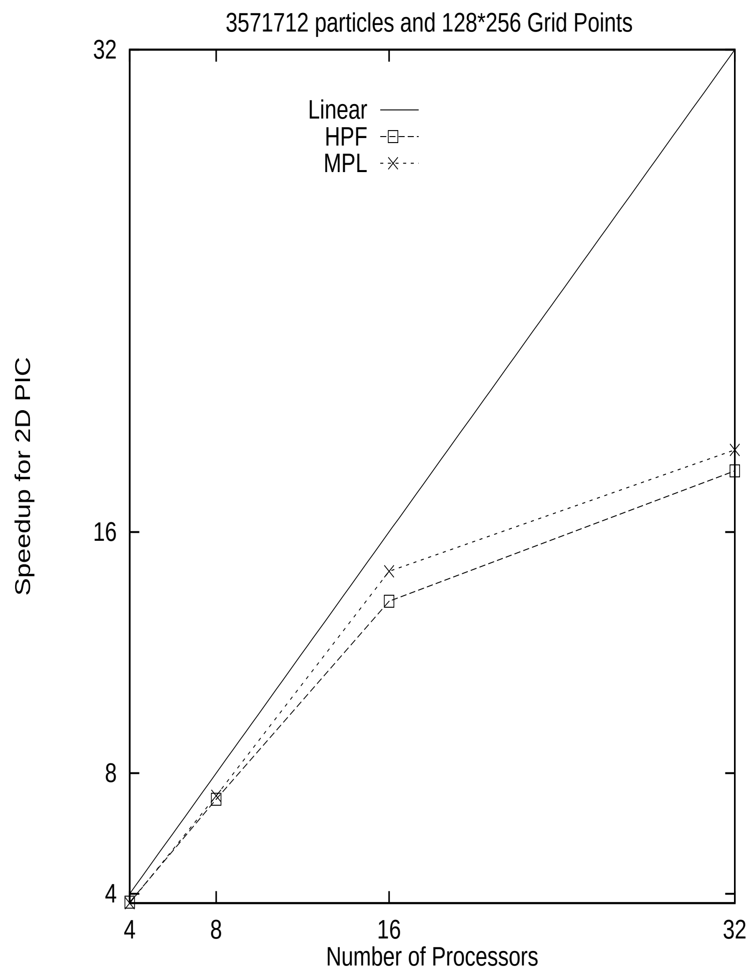

Figure 8: Comparison of Speedup Graphics of 2D MPL and HPF PIC Codes

CONCLUSION

- We implemented PIC in HPF succesfully
- If one HPF compiler can support our extrinsic functions as independent loop, then we can change these extrinsics with them
- We got linear speedup from our application
- HPF can compete with Message Passing Systems
- Here is very uptodate news. We implemented all of the extrinsics in independent loop with PGI HPF compiler successfully
- If one HPF compiler can not support our extrinsics, user can use our ideas about how the loops can be transformed to extrinsics