Investigations into OpenMP with EPOCH

Paddy Gillies

paddy.gillies@awe.co.uk

www.awe.co.uk

Why OpenMP?

- Many-core is here
- Performance through increased core count, not clock
- MPI won’t keep scaling
- Increasing MPI ranks impacts load-balance
- Xeon Phi: 60+ cores, 240+ threads
- GPUs: 1000s of threads

OpenMP

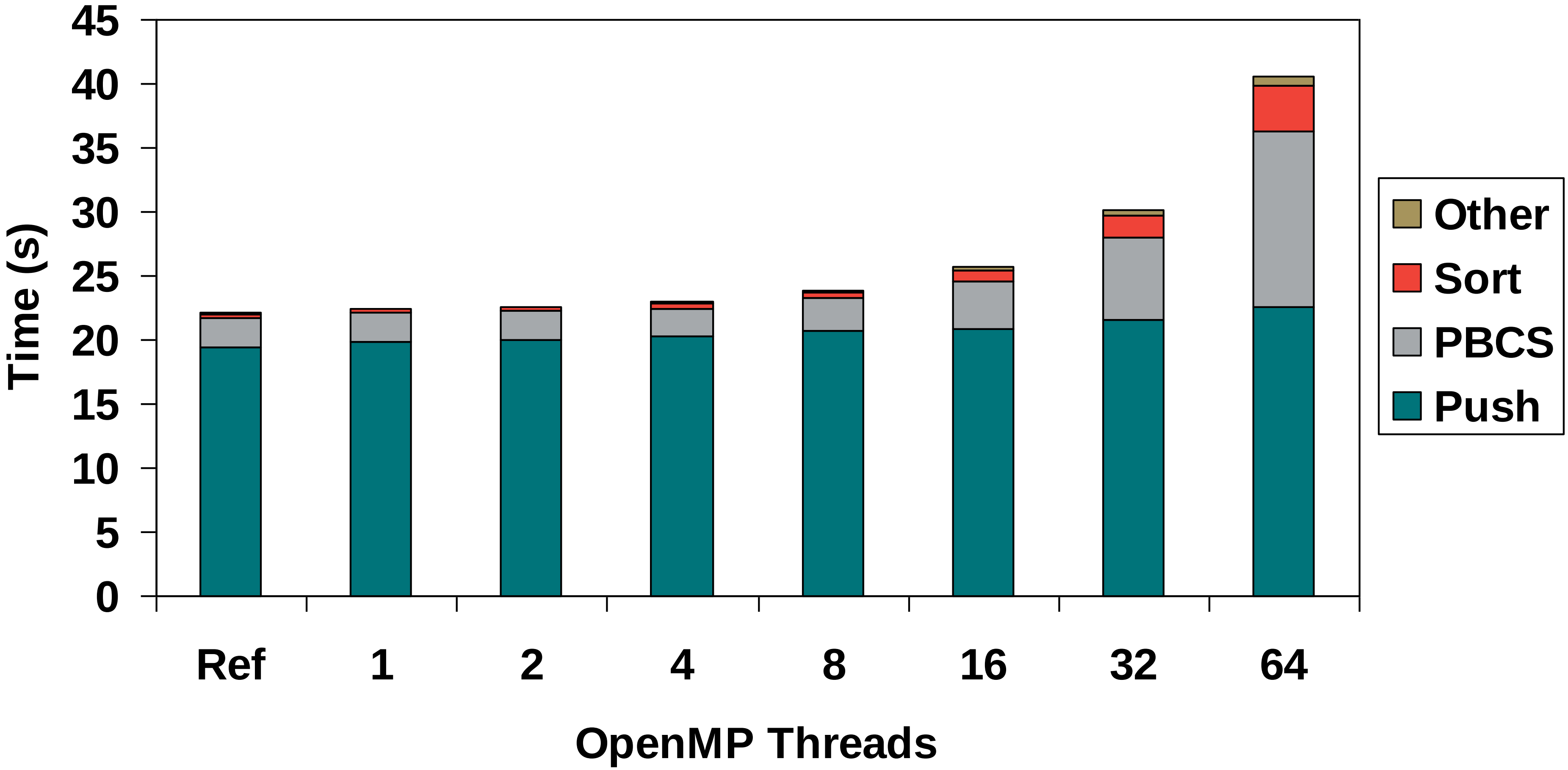
- Each process spawns a set of threads
- Threads of a single process share the same data
- Use in EPOCH for any loop over particles
  - Particle ‘push’
  - Particle boundaries
  - Sort
- Current deposition needs special treatment

First implementation

- Add OpenMP pragmas to particle push
- Need to manage current deposition
  - Use private copy of current array
  - Sum over threads after particle loop

```
$!OMP PARALLEL DO
DO
ipart=1,species(i)%npart
  ...
END DO
!$OMP END PARALLEL DO
```

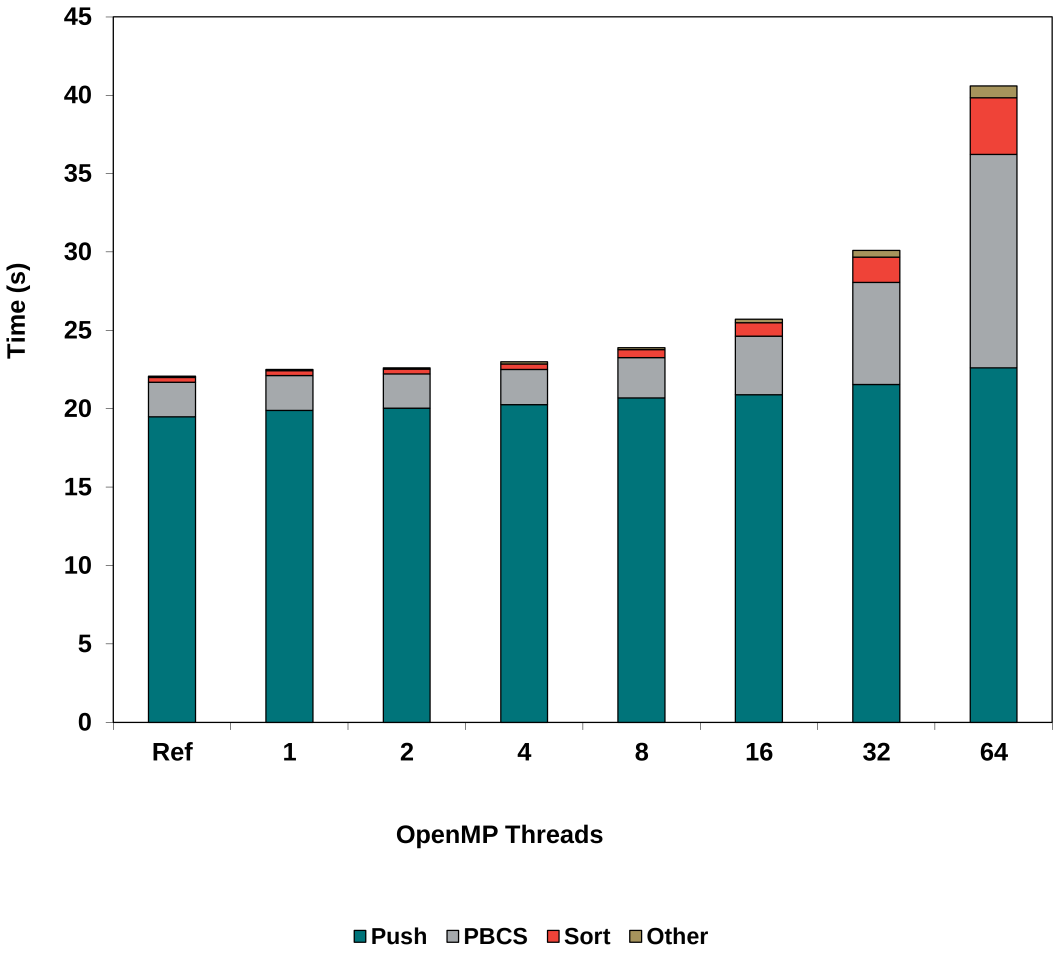
- Also add to particle boundaries and sort
  - Limited by data movement



MPI ranks x OpenMP threads=64
Dual 16 core 2.3GHz Intel E5-2698v3 with hyper-threading, Cray compiler

Scaling issues

- Works well for small numbers of threads
- Particle push scales well
- Performance becomes dominated by particle boundaries and sort

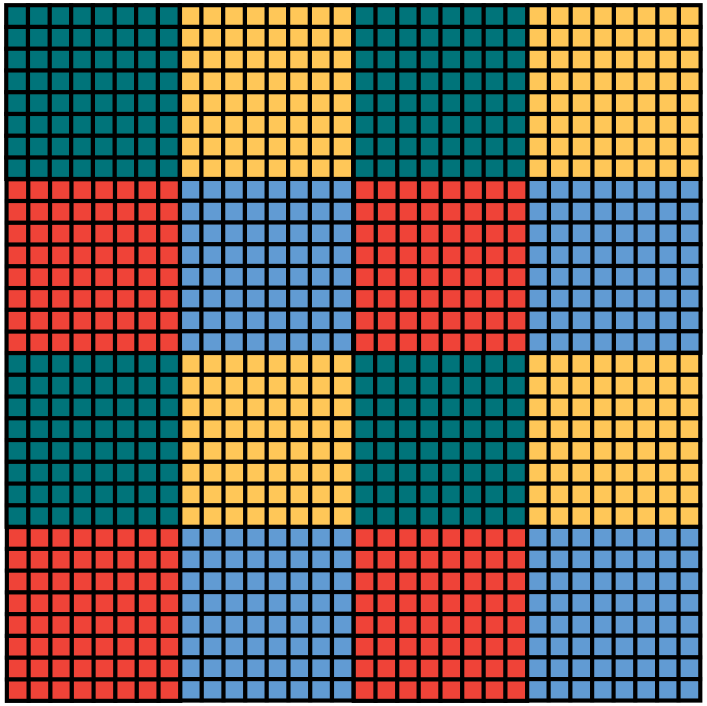


Colouring implementation

- Want to improve scaling
- Still need to avoid conflicts in current deposition
- Use a colouring scheme
- Particles in tiles of the same colour are pushed at the same time
- Each thread pushes separate tiles

Colouring scheme

- Grid is divided into tiles
- Tiles are ‘coloured’ so that like-coloured tiles are separate
- Particles are sorted by cell

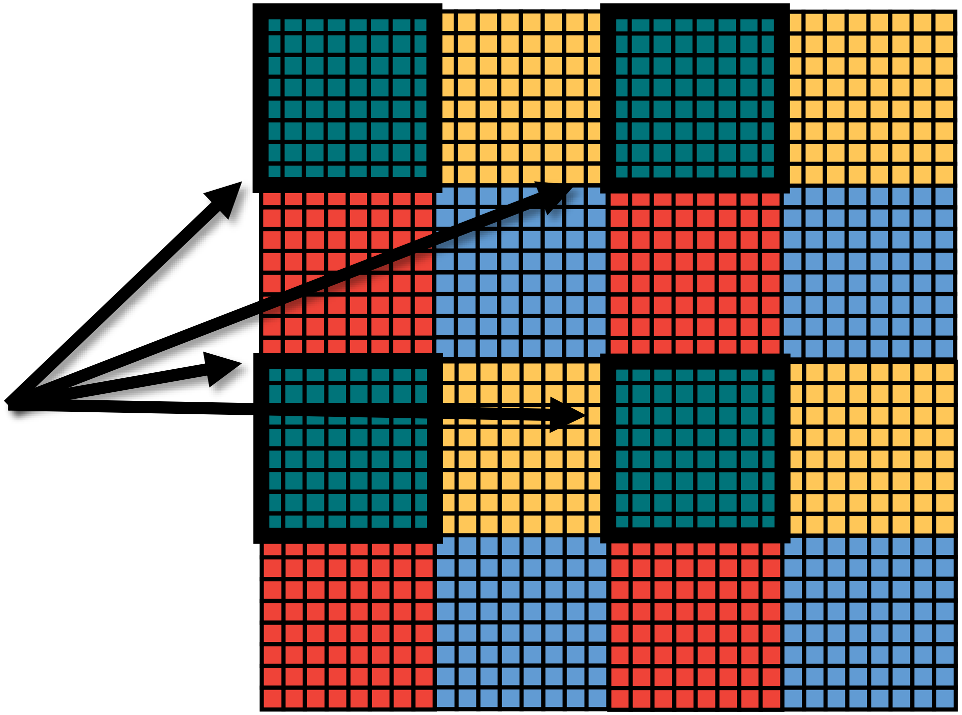


Colouring scheme

- Process each ‘colour’ in turn

- Each thread processes a different tile

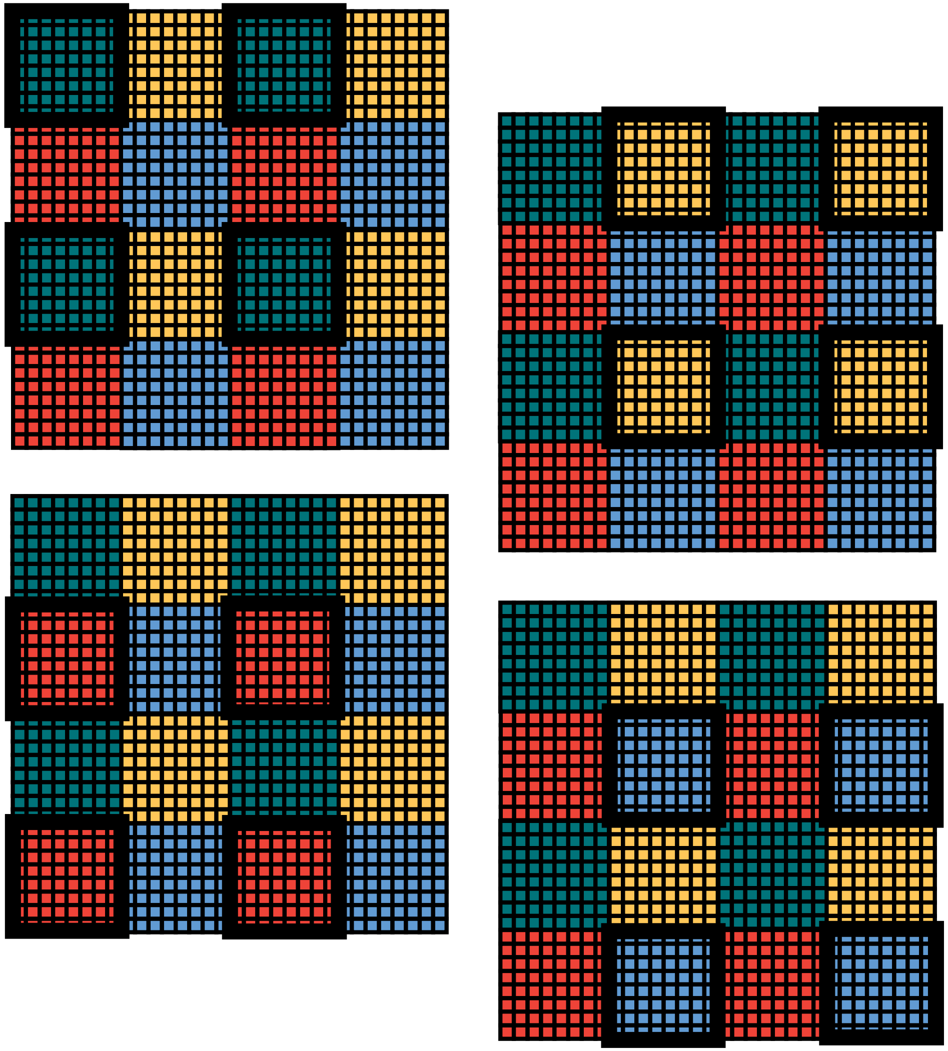
- Guarantees that different threads do not access same part of grid



4 threads

Colouring scheme

- Process each ‘colour’ in turn
- Each thread processes a different tile
- Guarantees that different threads do not access same part of grid



Colouring implementation

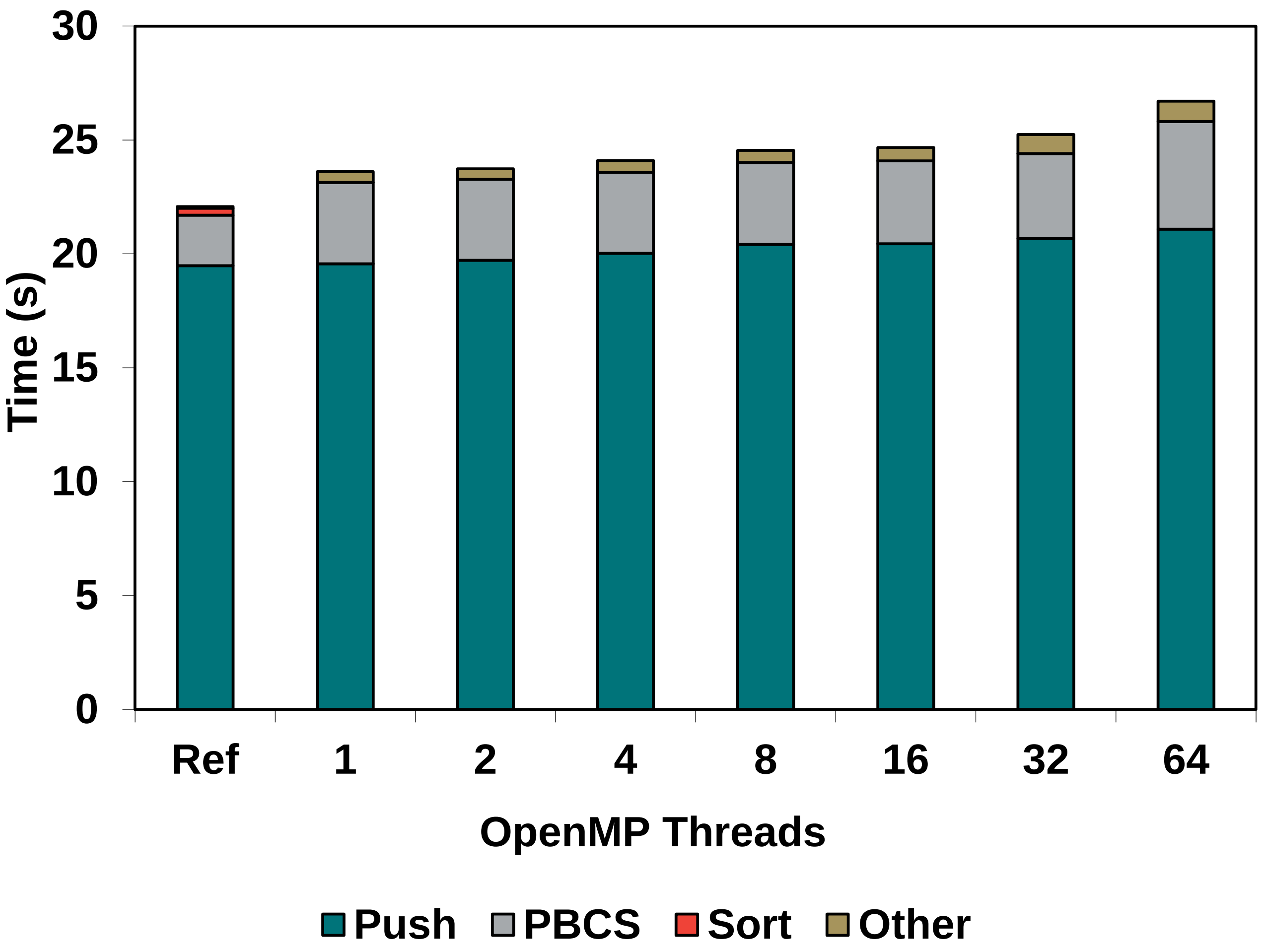
```
DO icol=1,n_colours

  $!OMP PARALLEL DO PRIVATE(istart,iend,ipart)
  DO itile=icol,ntiles,n_colours

    istart=tile_start(itile)
    iend=tile_end(itile)

    DO ipart=istart,iend
      ...
    END DO

  END DO
  !$OMP END PARALLEL DO

END DO
```

Sort and Particle boundaries

- Both change the particle data
  - Based on particle position

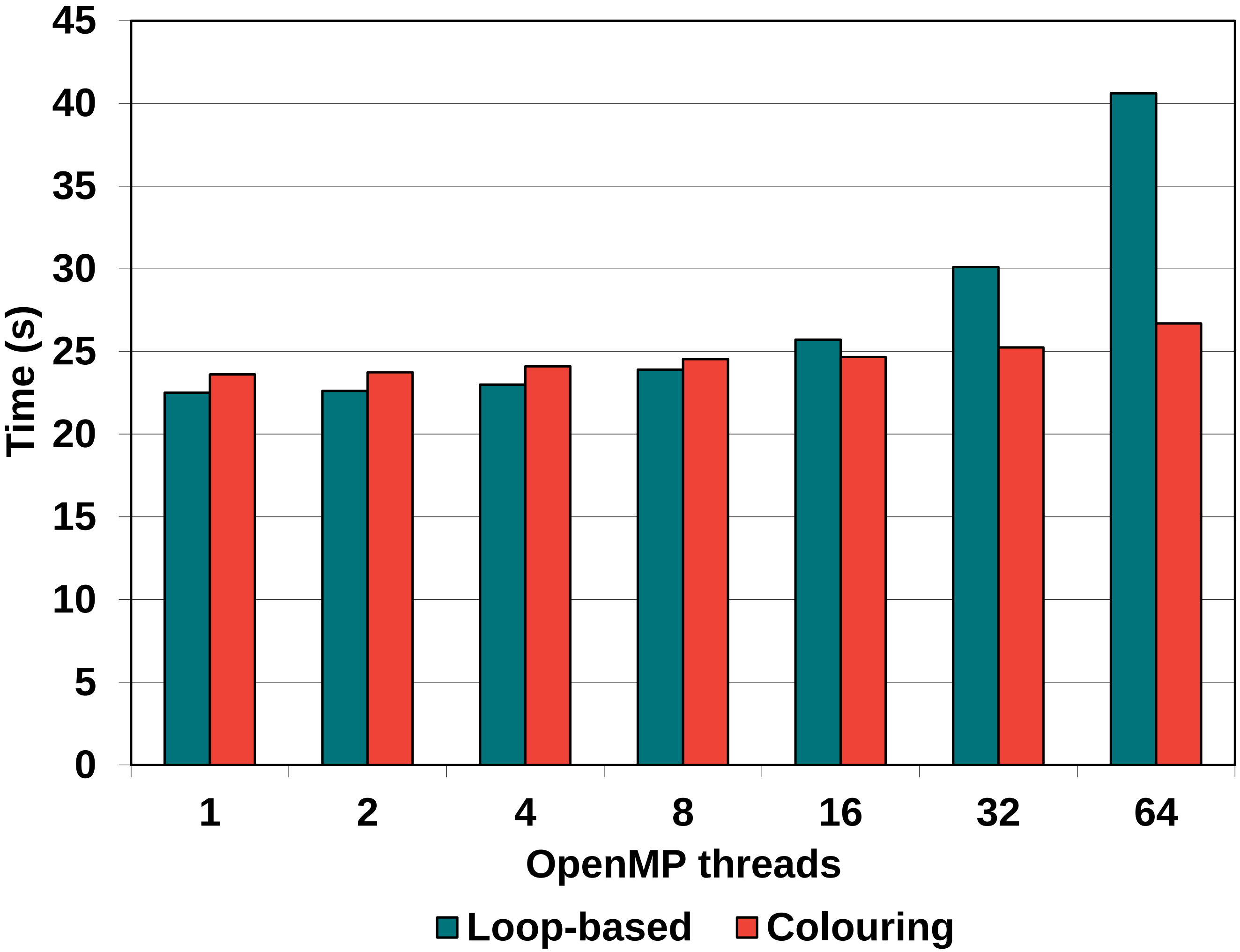
- Can merge these routines

- The colouring scheme can be used here
  - Count particles in each cell
  - Calculate particles in each tile
  - Send & receive particles from neighbour ranks
  - For each colour in turn
    - Place particles in their new location in the particle array



- Improved scaling to large numbers of threads
- Better particle boundary and sort scaling

MPI ranks x OpenMP threads=64
Dual 16 core 2.3GHz Intel E5-2698v3 with hyper-threading, Cray compiler



- Colouring scheme required for large number of threads
- Will help load-balance & I/O

Conclusions

- Many-core is here
- MPI scaling is limited
- OpenMP is one way to use many-core
- Efficient OpenMP method developed for EPOCH