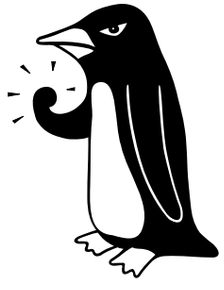
EPOCH specific

"The Angry Penguin", used under creative commons licence from Swantje Hess and Jannis Pohlmann.



Warwick RSE

Why Fortran?

- Not dead language
  - Dominates in HPC sector
  - >70% of used core hours on ARCHER are Fortran
  - New standard being released this year
- Easier to develop in than C
- Faster execution than C++ (in general)

Why Not Feature X?

- EPOCH has a very large user base
- Lots of people want lots of things
- Developer time is very limited

- Make feature requests on the forums
  - Remember that this is a *request*
  - It'll go on the list of things that we consider for future development

Meant for users

User interaction

- src/user_interaction/...
- ic_module - Do manual setup of particles to e.g. produce non-Maxwellian distribution function
  - Runs after the deck is parsed so particles are in state after deck has been parsed, can change
- custom_parser.f90 - add functions and constants to the deck
  - Useful for functions that are common to you but not common enough for us to write

User interaction

- custom_deck.f90 - Add new blocks to the deck, mostly useful if you've edited the core code but don't want to write a full deck block yet
- custom_laser.f90 - Mostly legacy, but does give you more control over the properties of a laser driver than you can have through the deck
    - Not much really, mostly left over from when the deck was less powerful.

EPOCH core bits

EPOCH variables

- nx, ny, nz - Number of cells in the domain on the current processor

- ng - Number of guard cells used for boundary conditions

- x, y, z - Variables holding the local grid on the current processor

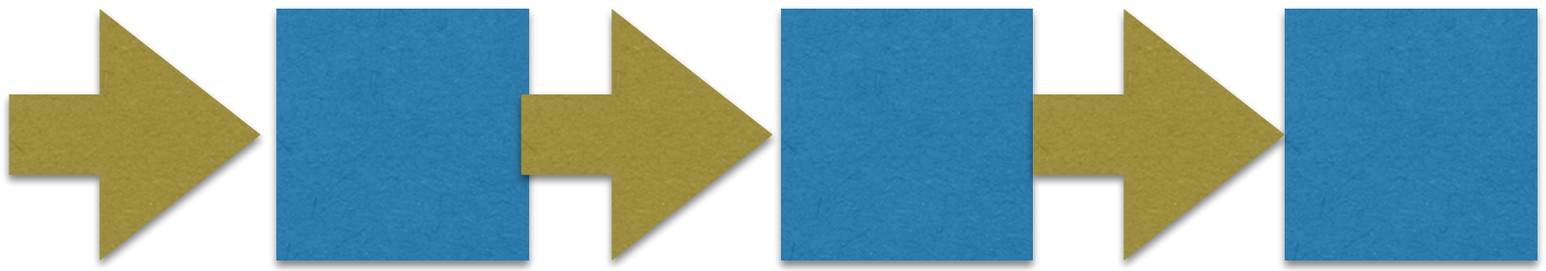
- species_list - List of all species (correspond to each species block)

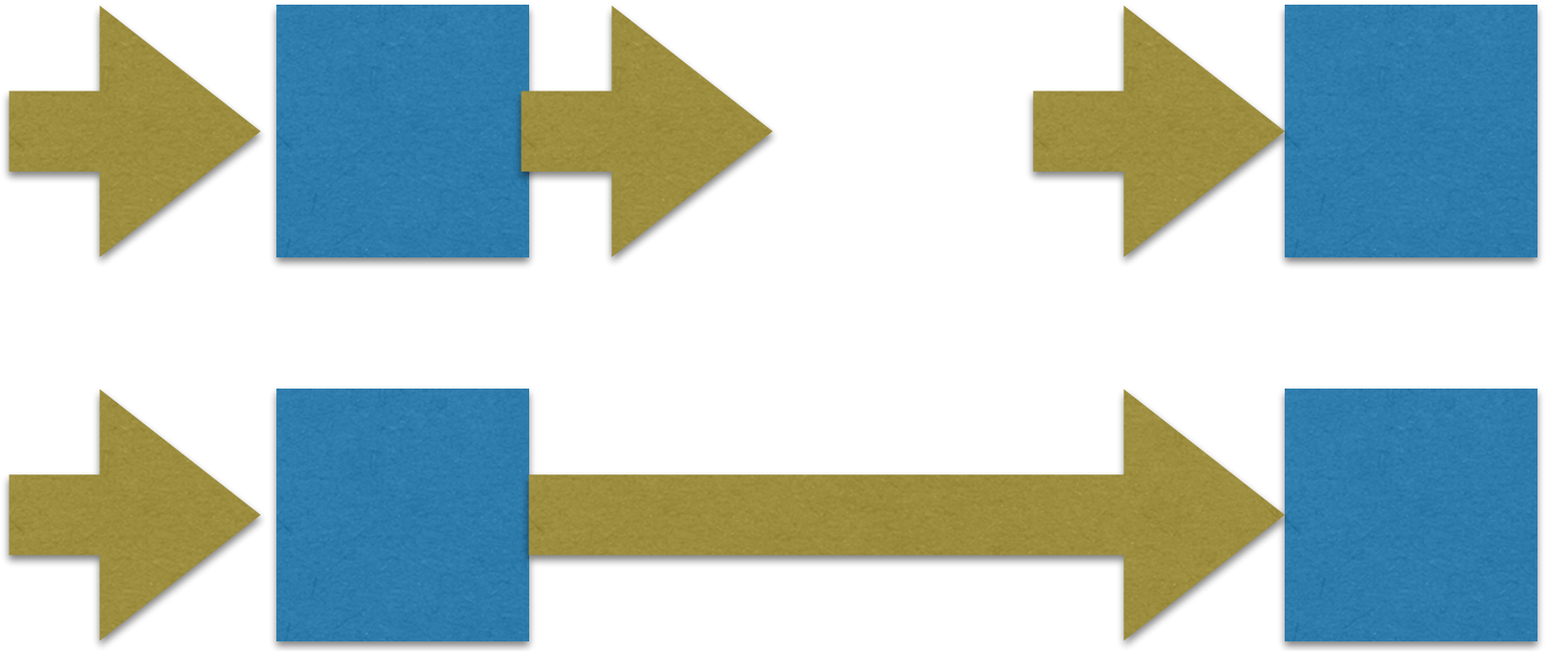
- TYPE(particle) - Type describing a single PIC particle

EPOCH particles

- EPOCH particles are stored in a linked list
- Each particle knows how to get to the next particle in the list
- First particle in the list is stored separately in a ***particle list(partlist)*** object
- Cannot access individual particle directly
- That's fine for EPOCH because you pretty much always go through all particles
- Can break the chain and reconnect it to remove particles

Linked list





Parallelism in EPOCH

Domain Decomposition

- Split up the spatial domain so that each processor has it's own part

- Boundaries between processors are just like normal boundary conditions

- Just have to get the data from the neighbouring processor to populate the boundary

- Want to do as little communicating as possible (maximise computation done)

- Minimise surface area to volume ratio to get best scaling

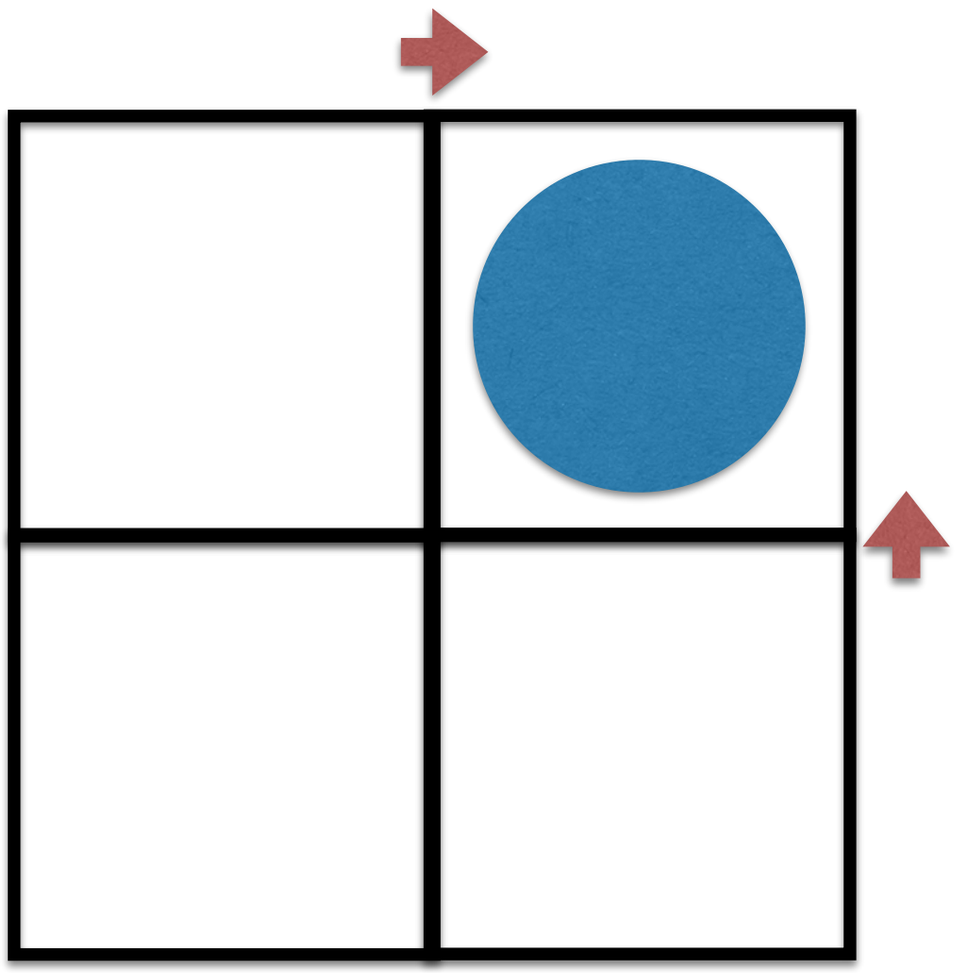
Domain Decomposition

- For fluid code (like CFSA's LARE code) job done
- Every processor has same sized part of domain
  - Finishes work at the same time
- For EPOCH have to worry about particles
  - Particles are free to move around
  - Can have more particles in one bit of grid than another

Domain Decomposition

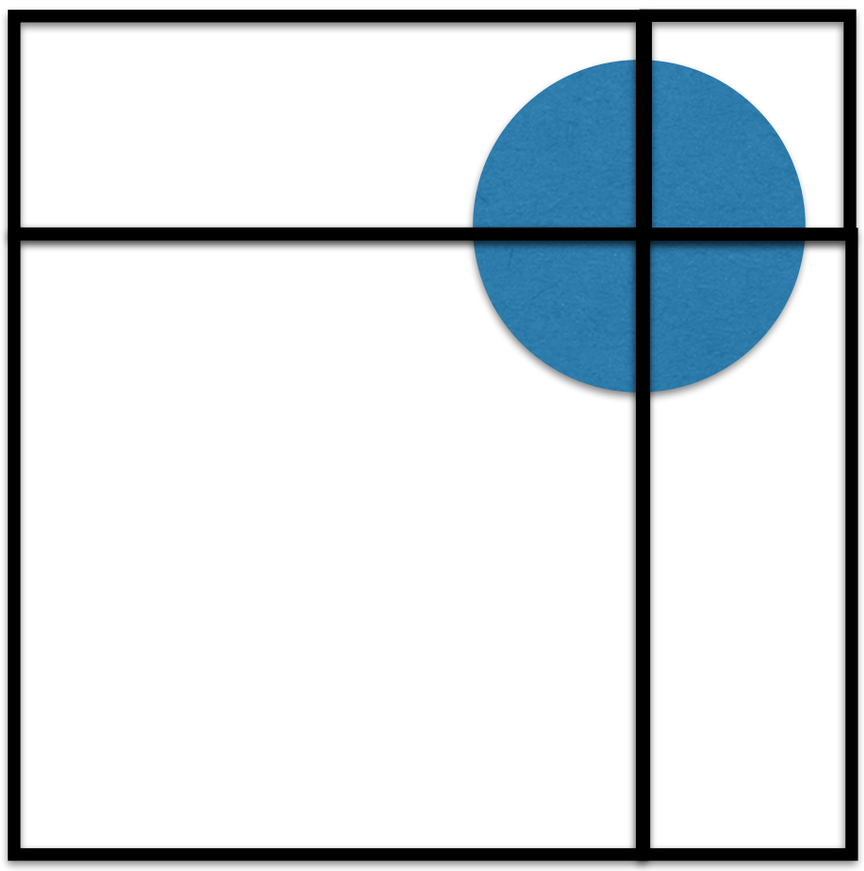
- Leads to a load balancing problem
- Have to adjust the amount of space on each processor so that they have the same number of particles, not the same amount of space
  - Roughly anyway, things like the EM solver also takes work and that depends on number of grid cells
- How do you do load balancing?

Load Balancing



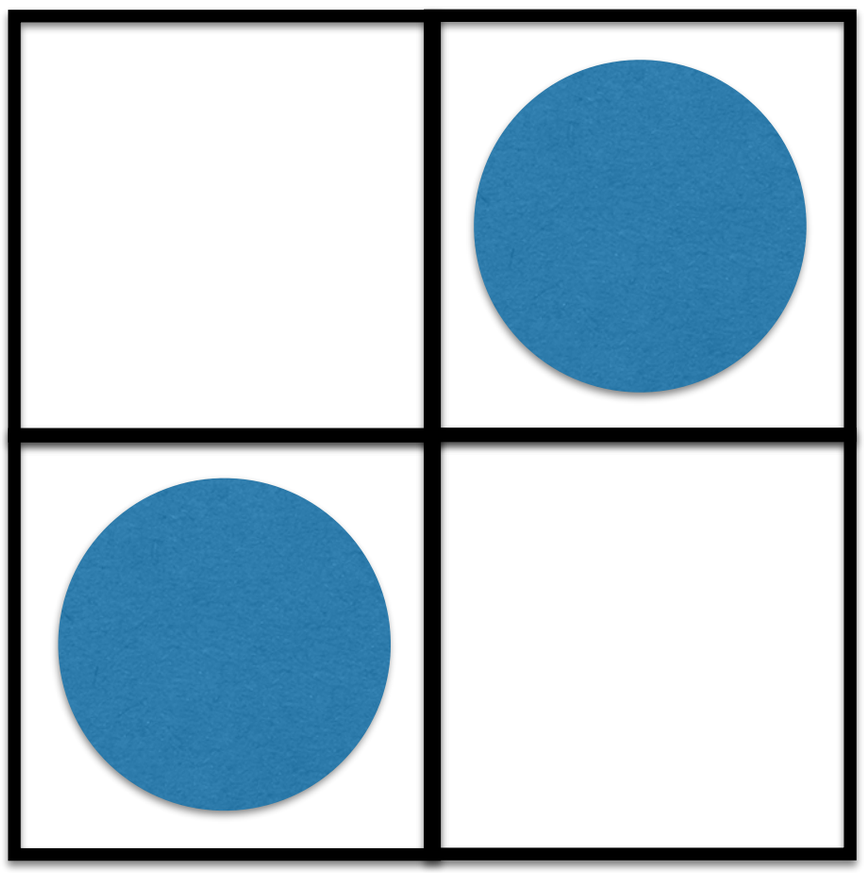
- Only 1 processor has any work
- Have to load balance
- Simplest solution is to move processor edges "rigidly"
- Y range in domain can only depend on Y coordinate of processor
- And similarly for X

Load Balancing



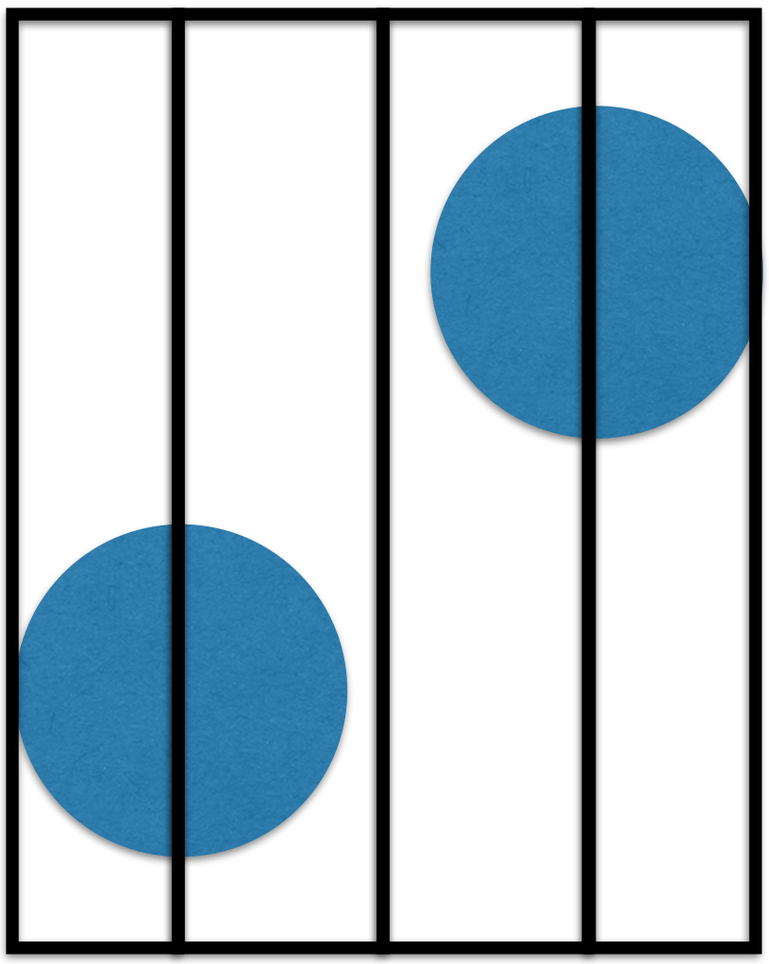
- Now all processors have same load
- Problem solved?
- Only for some cases
- This is EPOCH's current algorithm

Load Balancing



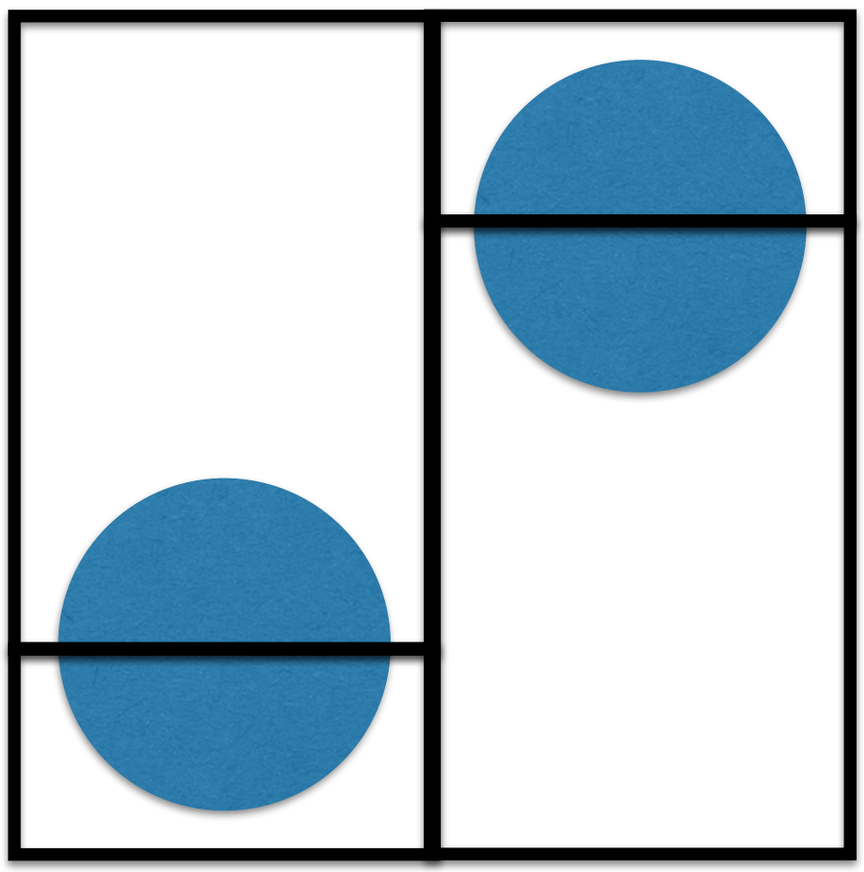
- Not for this case!
- Two processors do almost no work at all
- Using the algorithm before load as balanced as possible
- How to fix this?

Load Balancing



- Do it only in 1 dimension
- Can always split domain like this
- Performance drops as you have fewer grid cells on a processor
  - Communication dominates
- Limits maximum number of CPUs
- Is possible in current EPOCH

Load Balancing



- Break the requirement that we move edges rigidly
- Load is now balanced much better for wider range of conditions
- Work starting to implement in EPOCH soon

IO in EPOCH

SDF

- The SDF IO system is a low level self describing wrapper over the MPI-IO parallel output system

- File output is not dependent on number of processors that EPOCH was run on

- Performance is very high on well set up machines

- Gives us more control than using netCDF and a more seamless user experience than HDF5