

Generic fusion codes?

- We have encountered several codes with similar structure.
- The code has some sort of grid structure in which particles move and interact with the grid.

Generic fusion code - structure

- The codes are parallelized over non-interacting particles because it is the easy way.
 - Particles interact with the grid cells in a more or less random fashion.

Generic fusion code - problem

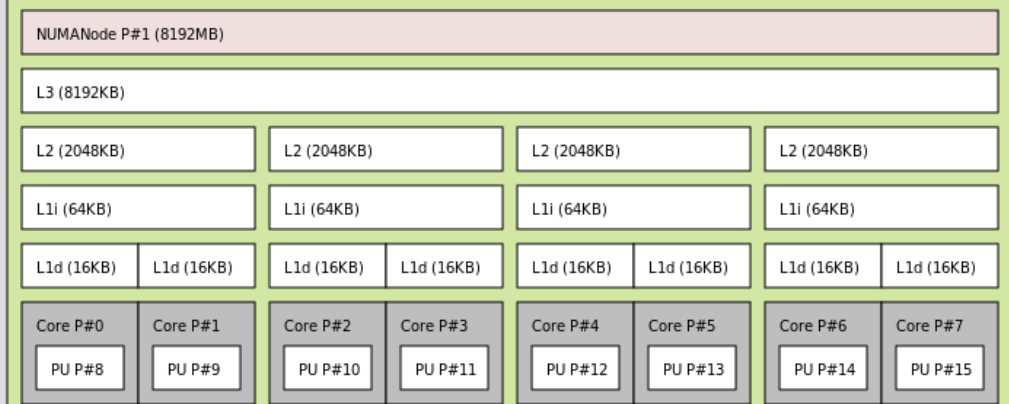
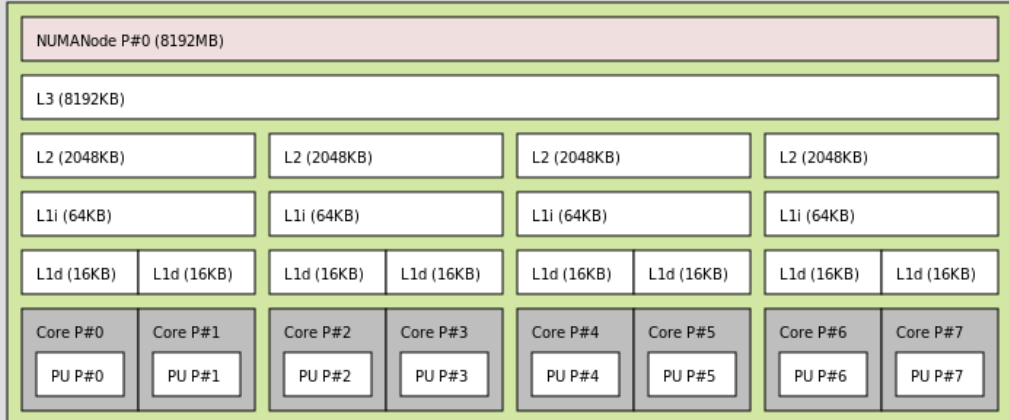
- As particles interact with the grid cells in a more or less random fashion it means random memory access.
- Random memory access for a large grid structure is computationally very inefficient - cache misses at very large rate.

Modern hardware

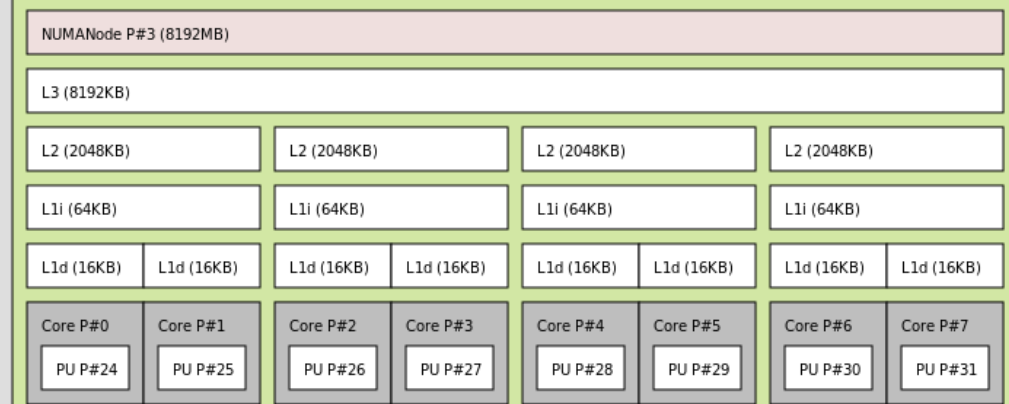
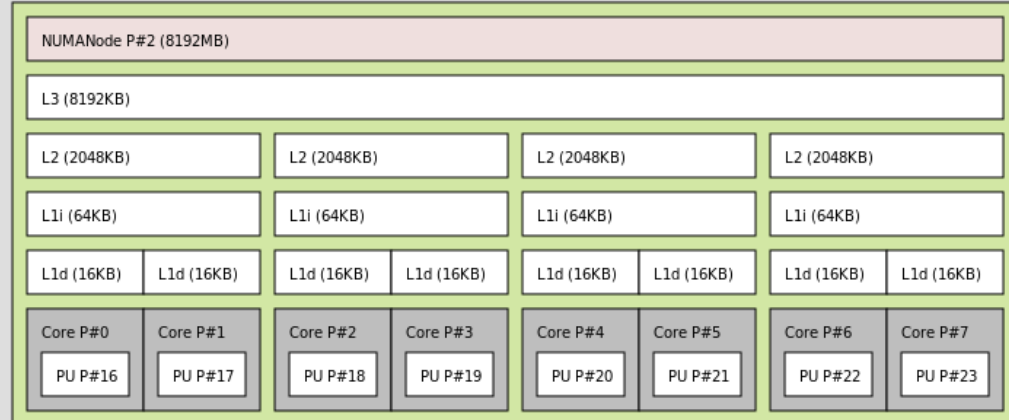
(https://en.wikipedia.org/wiki/CPU_cache)

Machine (32GB)

Socket P#0 (16GB)



Socket P#1 (16GB)



Generic fusion code - solution

- Exchange the parallelization over particles to be over grid domains. Use MPI.
 - Typically means that code need to be rebuilt from scratch.

Example – sputtering code

- Rebuilt it with a square grid – 2D domain-decomposition
- Projectiles generated per domain in a ‘swarm’ inducing recoils.
- Particles that are about to pass domain boundary are collected in arrays.

Example – sputtering code 2

- Boundary passing particles sent by MPI-messages to neighbor domains, and are considered as a secondary swarm.
 - The process is repeated until, `activeParticles=0`.

Example – sputtering code 3

- The domains can be small enough to fit in cache.
- The code now scales way better and cache misses practically vanish.
 - Code may become 100-fold faster.

The tabGap for GPUs

- Not explicit ACH project. Done in cooperation with ÅA.
Good example of code optimization.
 - MD code for radiation damage in solids.
- tabGap uses MD potentials with triplet potentials instead of more traditional pair-potentials only.

tabGap – the case for GPUs

- At every MD timestep calculate forces for all relevant triplets in the solid. Calculation use tabulated data and spline interpolation in an (r_1, r_2, θ) -space.
- This is perfect for GPUs – same instructions for a very large set of independent data.

Miscellaneous

- One task was a compression of data – Just applying standard tools ~ 30% reduction in size.
 - EUDAT project – did not work well.
- One task I went through an ancient solver done by some russians in the 80's. Code was a complete mess. I cleaned out about 25% of the code lines that did absolutely nothing.

Miscellaneous 2

- One task was to parallelize a code built with object-oriented Fortran. Code parallelization with custom made MPI-type as to not destroy the object-structure of the code. Worked fine – good scaling achieved.
- One code had a few lines that constantly read a large set of constants from the main memory. Changed to only fetch needed constants. 95% of compute time vanished + trivial parallelization with MPI.