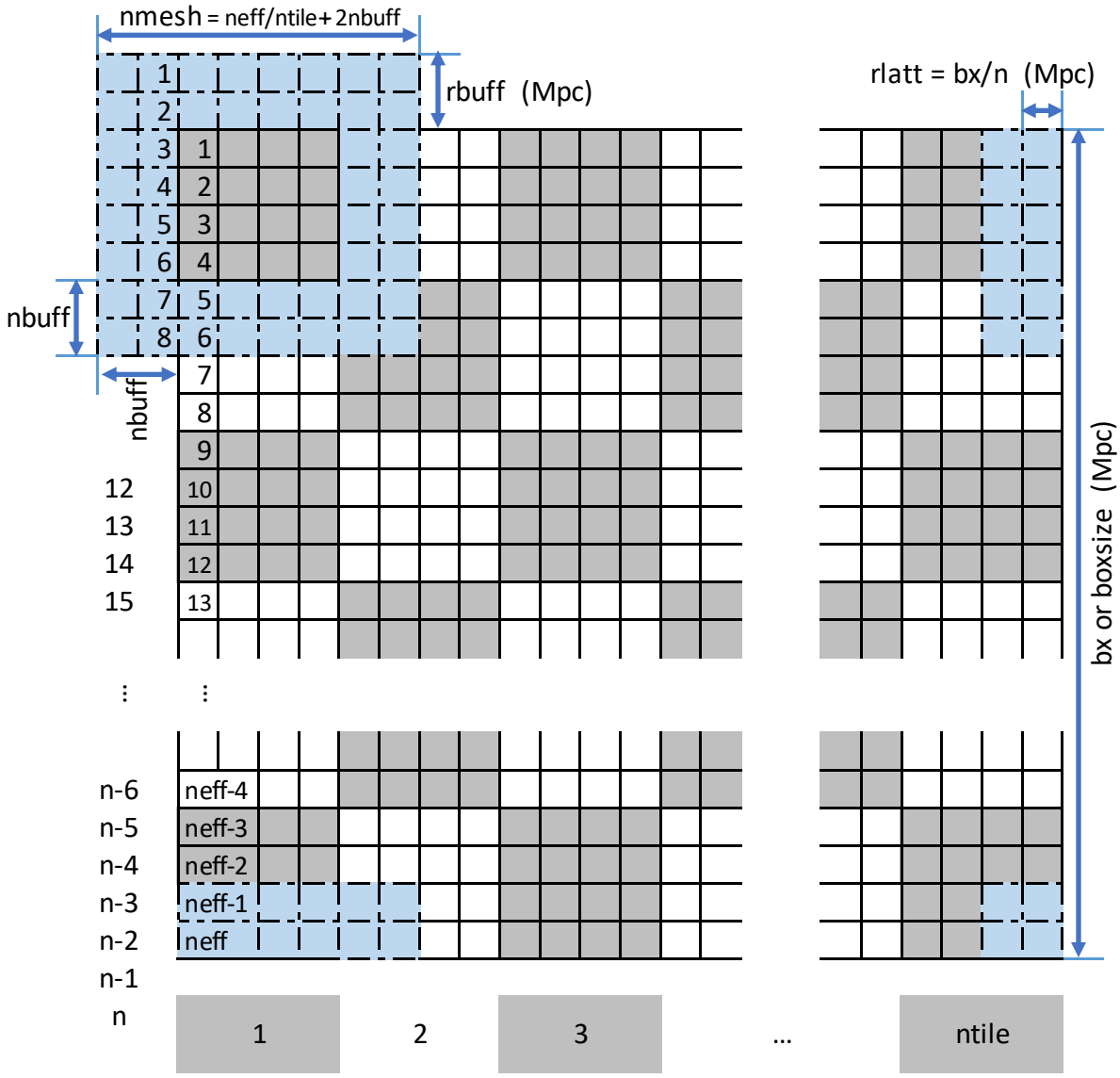

The python script `peak-patch/tools/setup-run.py` determines the optimal tiling for a run given *e.g.* if you wanted to simulate a cubic volume with Lagrangian (initial) side length 256 Mpc and use a cubic lattice of side length 512 cells:

```
cd peak-patch/tools
python2.7 setup-run.py 512 256
```

The general usage is of the form

```
python setup-run.py <ngrid> <boxsize> [cluster:<(niagara, gpc)>
nproc per node:<(32, 8)> <rbuff>]
```

Where

```
boxsize: the sidelength of the simulation volume in Mpc
ngrid: the number of gridpoints to be used in the simulation
```

The simulation volume is then split into cubic tiles to be run in parallel for speed. Each tile must have a buffer region (shown in blue above) to capture short range dynamical effects, and thus if the number of tiles is large, you end up double-counting a lot of cells. So this is an optimization problem, which `setup-run.py` solves for us, an outputs:

```
ntile: the number of tiles per side
nbuff: the thickness of the buffer around each tile in cells
nmesh = (n-2nbuff)/ntile + 2nbuff: the sidelength of a tile
(including the buffers) in cells
nnodes: the number of compute nodes to run the simulation on
```

One thing to note is that the chosen value of `n` sets the sidelength (in terms of a number of cells) of the effective simulation volume plus buffers on each end, which means that the effective simulation volume is actually:

```
neff = n - 2nbuff
```