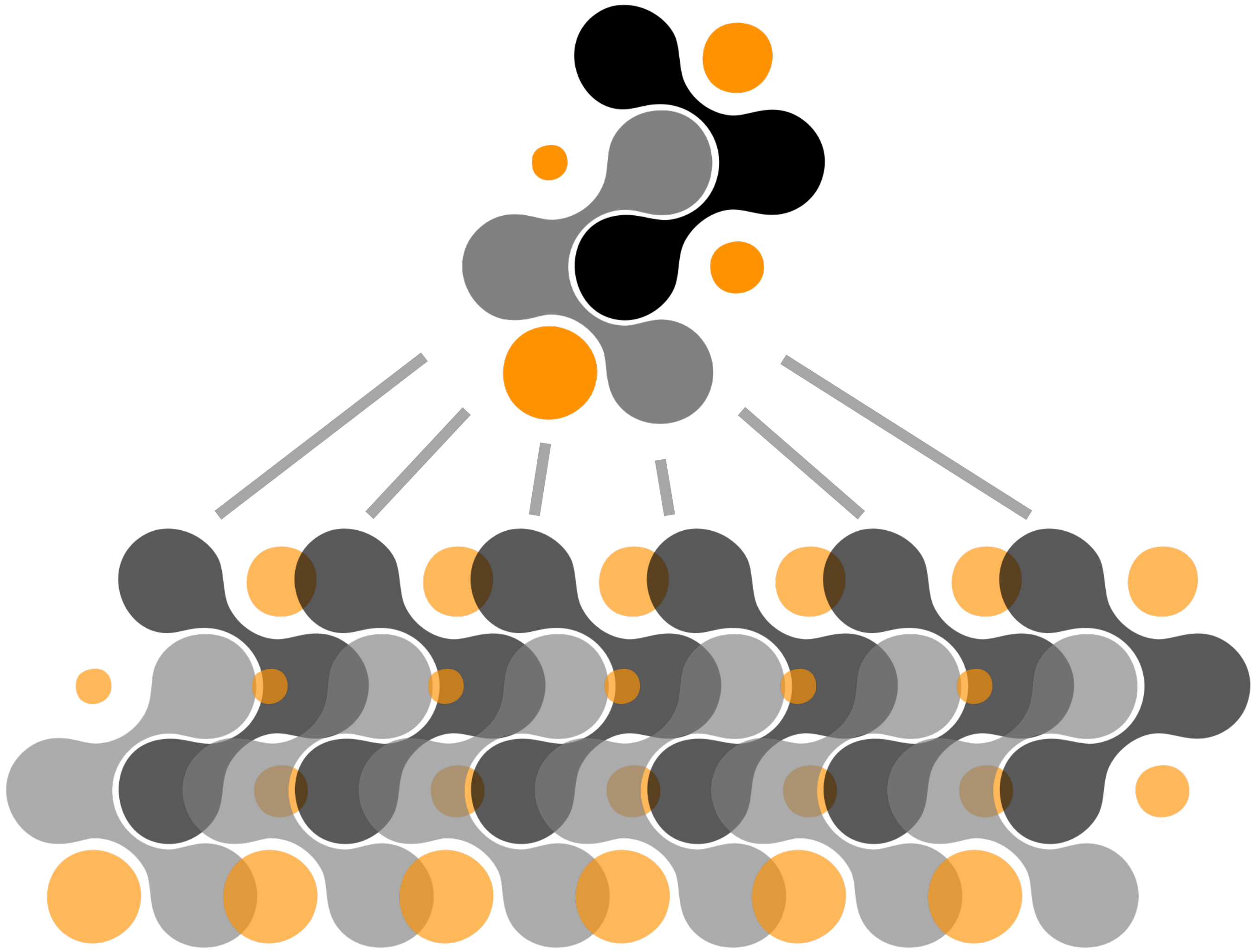
Parallelizing MDAnalysis



Why? Speed!

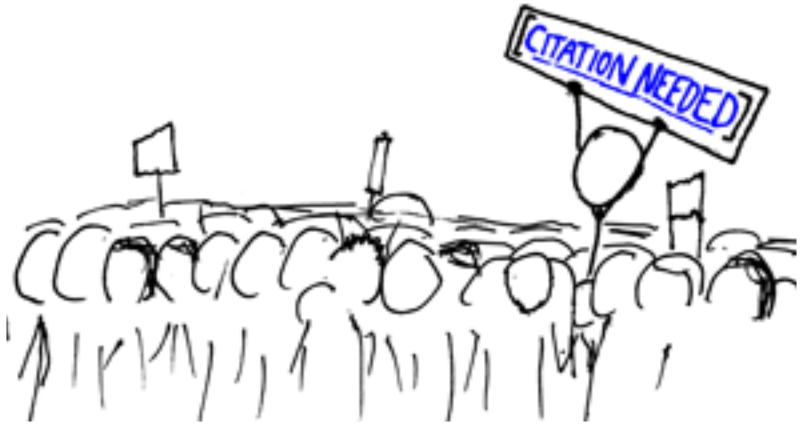
How? Over frames!

Most MD analyses are over trajectory frames

Most of those involve calculations independent from other frames

Almost embarrassingly parallel (linear scaling vs. serial)

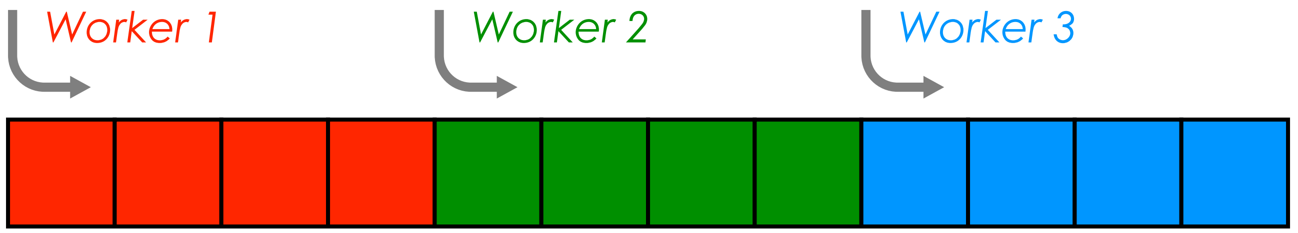
Generic implementation



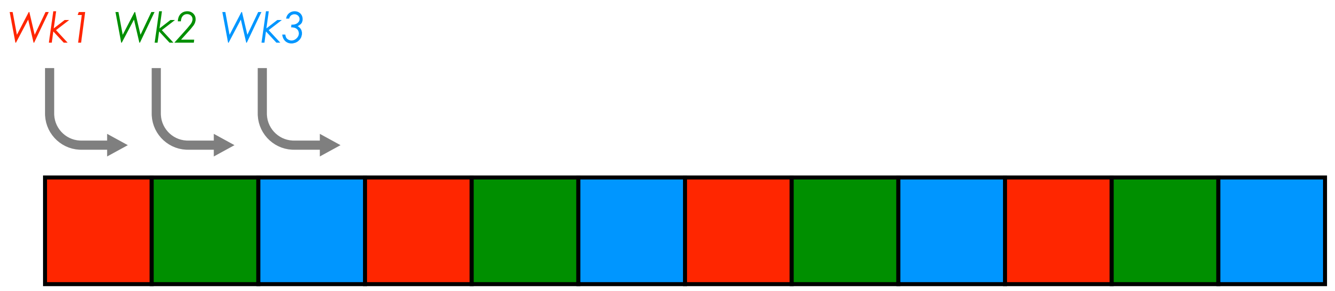
From https://xkcd.com/285/

Parallelization over frames

Block-wise



Interleaved



Simpler to implement

Better load-balanced

May require sorting of results

Parallelization Modes

Thread-based
(full memory sharing)

Thread safety can be complex

CPython implementation is *very* inefficient because of the GIL (Global Interpreter Lock)

Process-based
(requires inter-process communication)

Fork
(child processes)

Sub processes start from the state in the code when they were spawned

`multiprocessing`

Cannot parallelize over different nodes (computers)

Independent processes

Each process starts from scratch

`MPI (mpi4py)`

Can parallelize over different nodes (computers)

Typical flow `multiprocessing`

Parent process

1. import modules
2. define functions/Analyses
3. load Universe
4. `multiprocessing.Pool()`
5. communicate work/parallelization data

Children processes

1. receive work/parallelization data (**worker id**) as target function arguments
2. **iterate/compute over respective frames**
3. if needed, do some post-processing
4. communicate results back to parent process
5. quit

6. receive list of results per worker
7. combine results
8. output and quit

Typical flow

`MPI`

All 'ranks'

1. import modules
2. define functions/Analyses
3. load Universe
4. determine parallelization division from rank id and pool size
5. **iterate/compute over respective frames**
6. if needed, do some post-processing
7. communicate results to rank 0

Other ranks

8. quit

Rank 0

8. receive list of results per worker
9. combine results
10. output and quit

How much to parallelize?

More is usually faster

but

Avoid oversubscribing cores
`multiprocessing` uses all cores by default
`os.get_schedaffinity` shows you available cores
(or `cat /proc/cpuinfo`, or `htop`, or `lstopo`)

Beware of disk I/O bottlenecks
worse if the computation time per frame is fast compared to frame reading

RAM may become limiting
depending on the analysis algorithm

Respect imposed limits
HPC use may impose core limits

Advanced strategies (not in the practical)

`multiprocessing`:
avoid initial communication by finding own worker id and taking advantage of access to parent memory
(requires reopening open filehandles to trajectories)

`MPI:`
avoid initializing one Universe per rank by initializing only for rank 0 and then broadcasting to all ranks.

Implementations

Pure MDAnalysis + `multiprocessing/MPI`
Example in the practical!

PMDA https://github.com/MDAnalysis/pmda
Implementation of a `ParallelAnalysisBase` using `dask`, a flexible and scalable parallel scheduler.

MDreader https://github.com/mnmelo/mdreader
Implementation of per-frame parallelization using either `multiprocessing` or `MPI`, plus some command-line convenience.