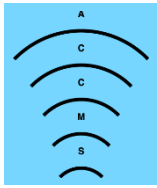


Finding Regularity in Irregularities

- Assumptions

	HPC	Big-Data
compute on	double x;	int n; int *p;
huge //ism	definitely	why not
memory wall	high&thick	= or >
lower layer	SIMD+wide L/S	(SIMD+)wide L/S
code	for() {...}	while() {...if...if...}
data	A[i][j][k++]	p->q->r; p=p->s
SIMD friendly	yes hopefully	no *in general*

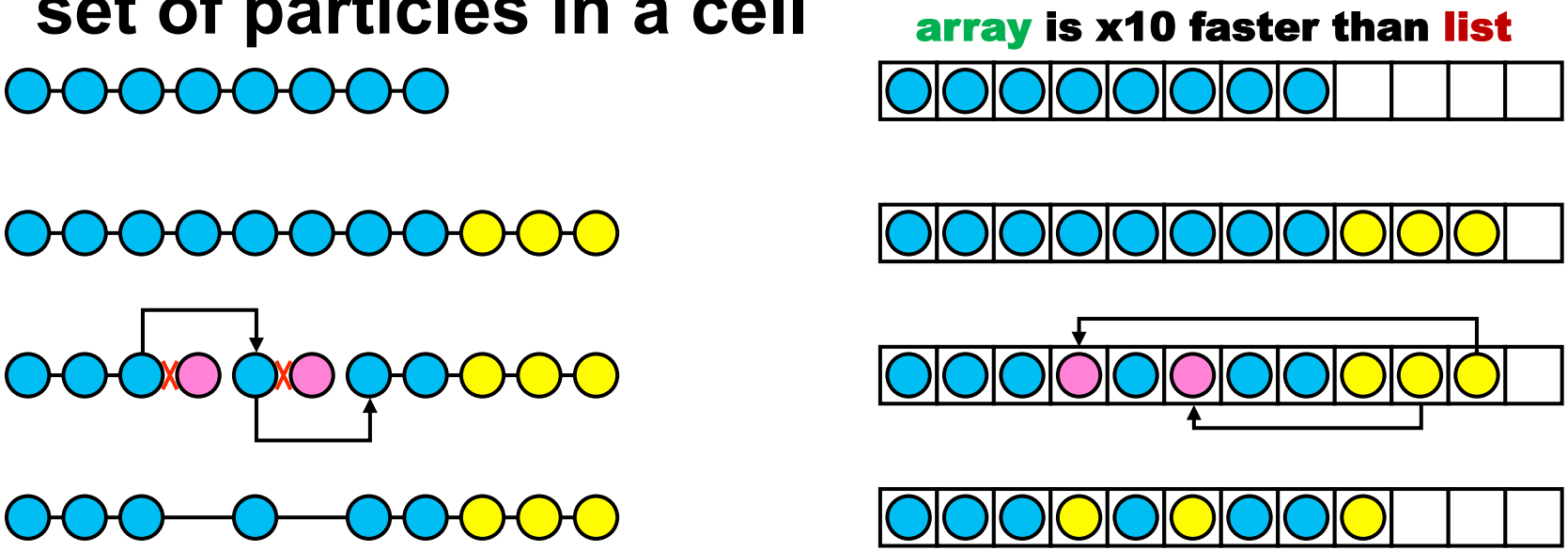
- Does *in-general* irregularity always hold for *in-practice* executions?

Linear List in General

- Why do we love a *list* to represent a set?
  - The set is created dynamically (with other sets).

    Q: But once created, cannot it be an array?
  - Elements are added/deleted to/from the set.

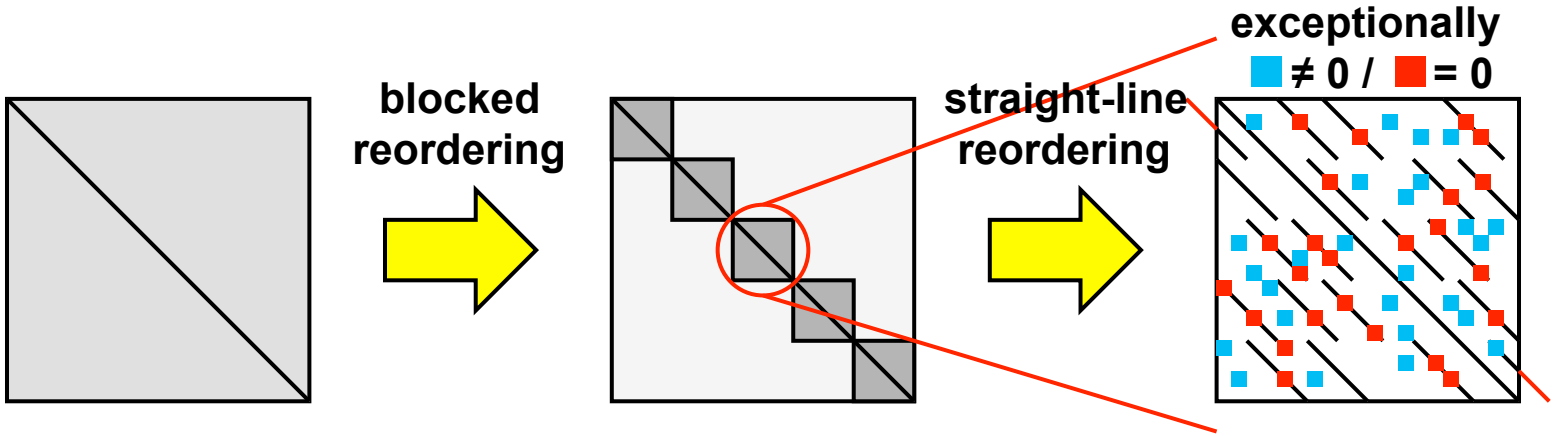
    Q: But how often? How many?
- HPC example of *array in practice*:
  set of particles in a cell

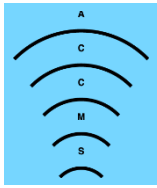


Pointer-rich `struct` in General

- Trees, graphs, etc. are much tougher than lists to represent by simple arrays.
  - Neighbors of a node cannot be contiguously placed in an array.

    Q: Must be so, but cannot a set of neighbor pairs be contiguous?
- HPC example:
  SIMD-aware sparse matrix reordering





Regularity-aware Programming

- I *don't love* such programming *definitely !!*
  - Even the relatively simple array-based implementation has been a nightmare for me having programming experiment of 30+ years.
  - Such effort is rewarded only with performance (i.e., not with money or ... ☺).
- We have to remember we need;
  - *Sets* rather than *lists* or *arrays*.
  - *Graphs* rather than set of `struct` or *CRS*.
- Why don't we (or you hopefully) challenge this issue to provide BD&HPC community of (a kind of) *library* with proper abstraction and hidden efficient implementation.