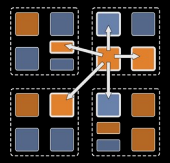
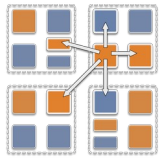
GRAINSIZE



Grainsize

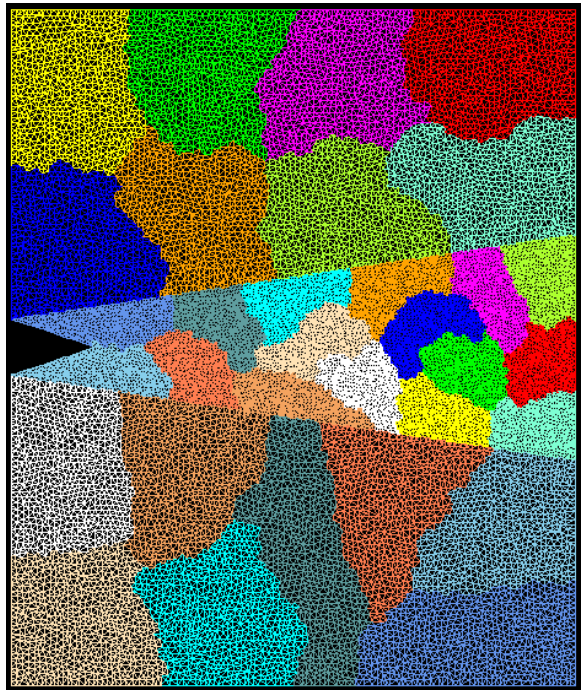
- Charm++ philosophy:
  - Let the programmer decompose their work and data into coarse-grained entities
- It is important to understand what I mean by coarse-grained entities
  - You don't write sequential programs that some system will auto-decompose
  - You don't write programs when there is one object for each *float*
  - You consciously choose a grainsize, **but** choose it independently of the number of processors
    - Or parameterize it, so you can tune later

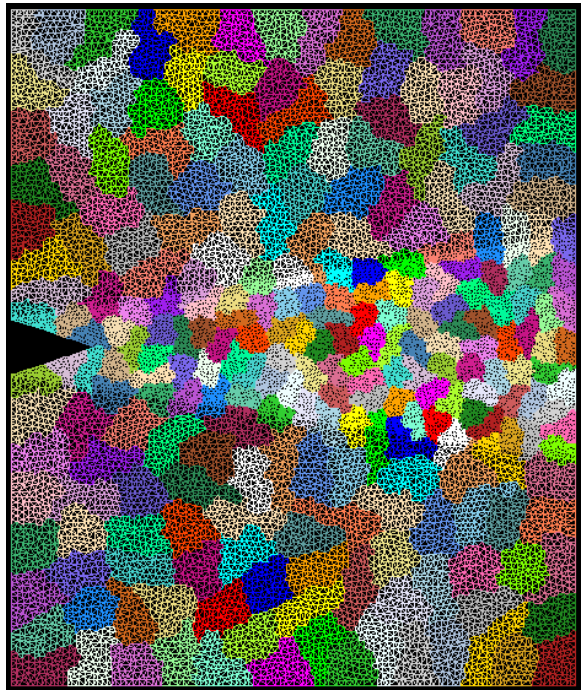


PPL UIUC

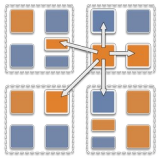
Crack Propagation

This is 2D, circa 2002...
but shows overdecomposition for unstructured meshes





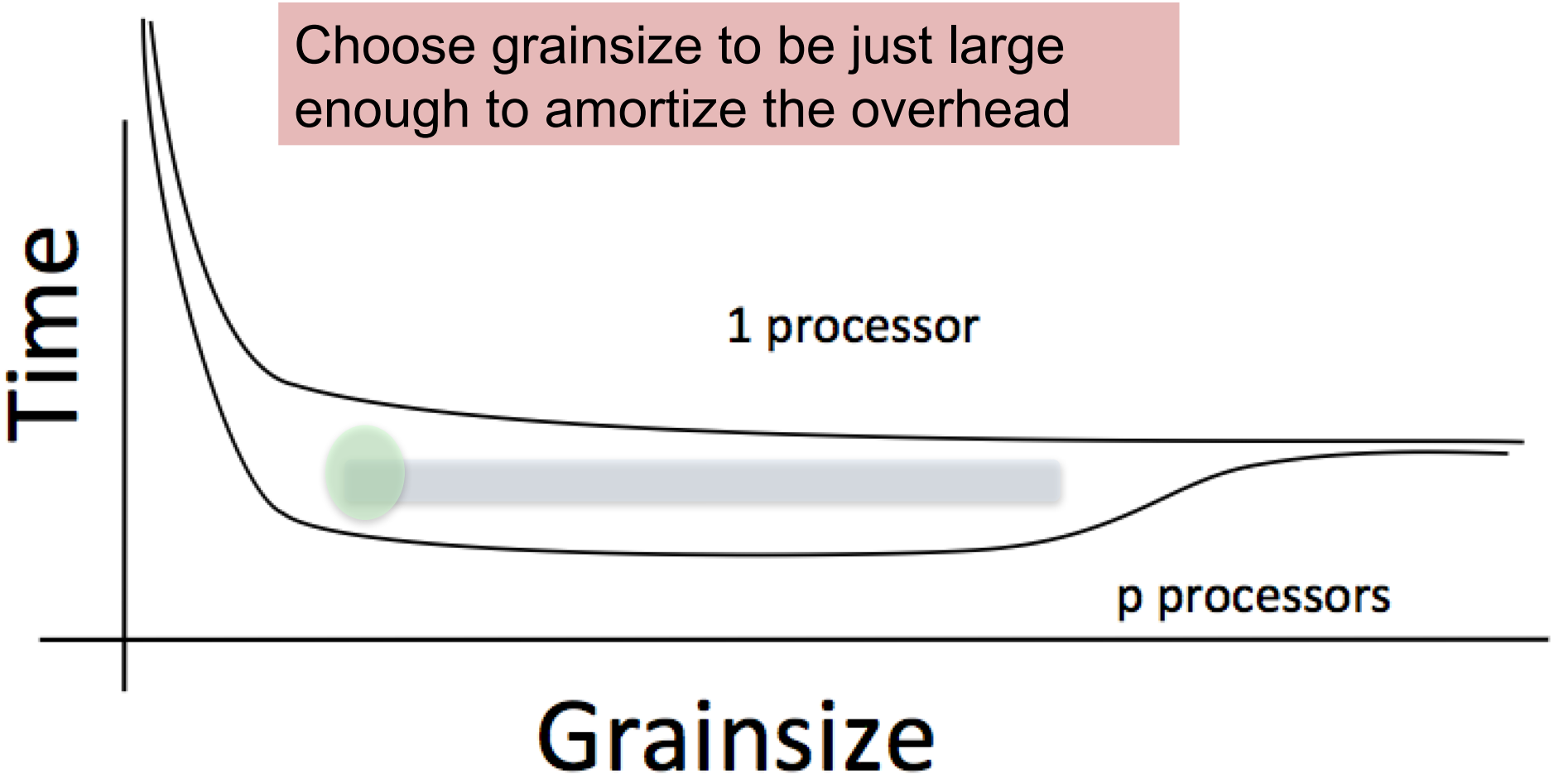
Decomposition into 16 chunks (left) and 128 chunks, 8 for each PE (right). The middle area contains cohesive elements. Both decompositions obtained using Metis.
Pictures: S. Breitenfeld, and P. Geubelle

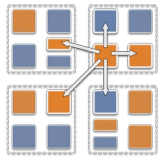


PPL UIUC

Working definition of grainsize:
amount of computation per remote interaction

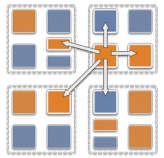
Choose grainsize to be just large enough to amortize the overhead





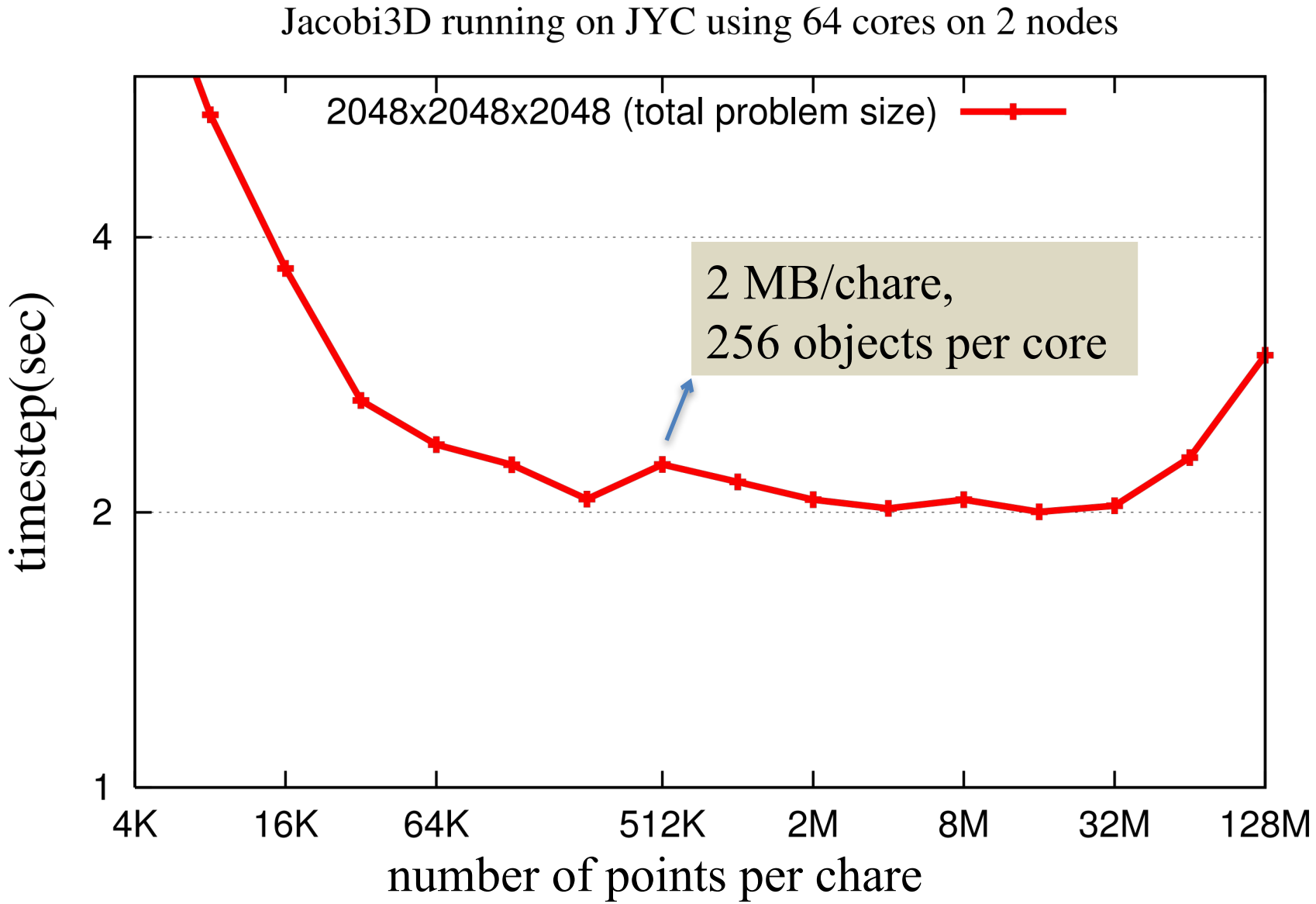
Rules of Thumb for Grainsize

- Make it as small as possible, as long as it amortizes the overhead
- More specifically, ensure:
  - *Average* grainsize is greater than $k \cdot v$ (for some k, say $10v$)
    - v: overhead per message
  - No single grain should be allowed to be too large
    - Must be smaller than T/p, where p: number of processors, T: sequential execution time
    - Can generalize by saying must be smaller than $k \cdot m \cdot v$ (say $100v$)
- Important corollary:
  - You can be at close to optimal grainsize without having to think about p, the number of processors



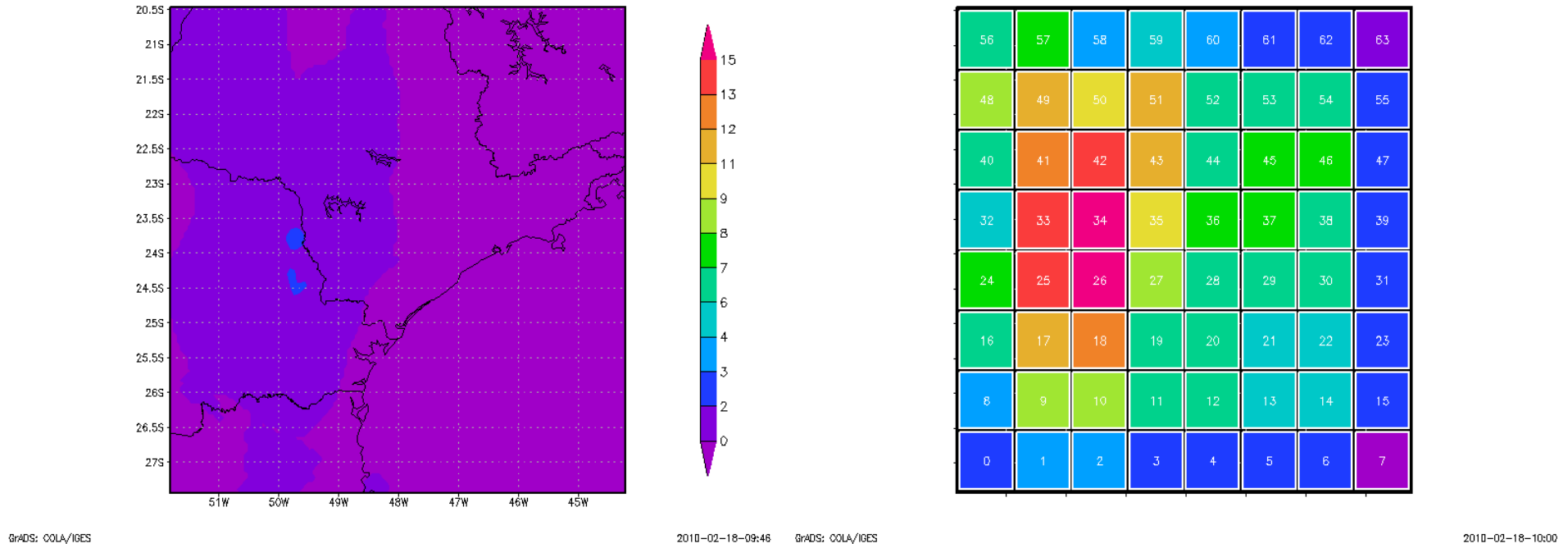
PPL UIUC

Grainsize in a common setting



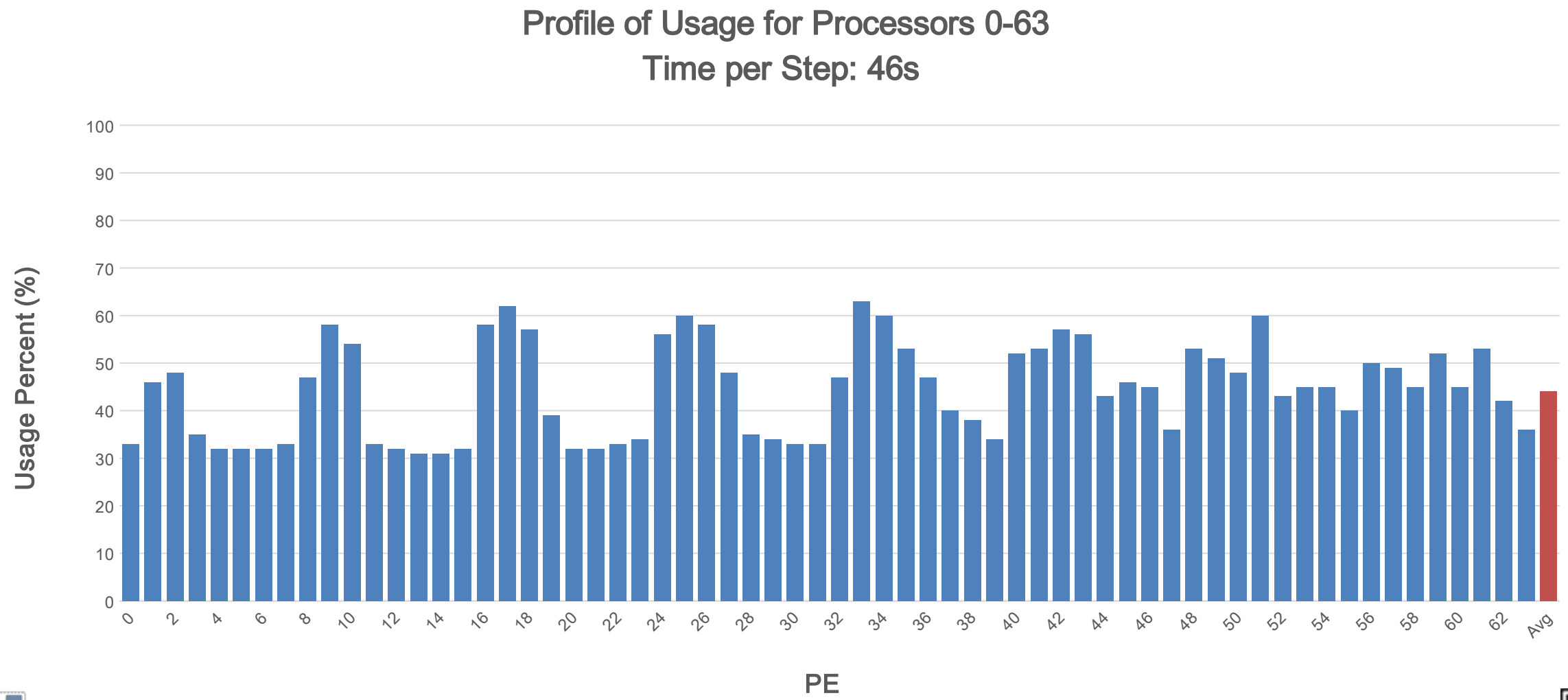
Grainsize: Weather Forecasting in BRAMS

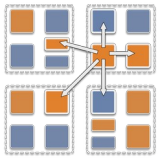
- BRAMS: Brazillian weather code (based on RAMS)
- AMPI version (Eduardo Rodrigues, with Mendes, J. Panetta, ..)



Instead of using 64 work units on 64 cores, used 1024 on 64

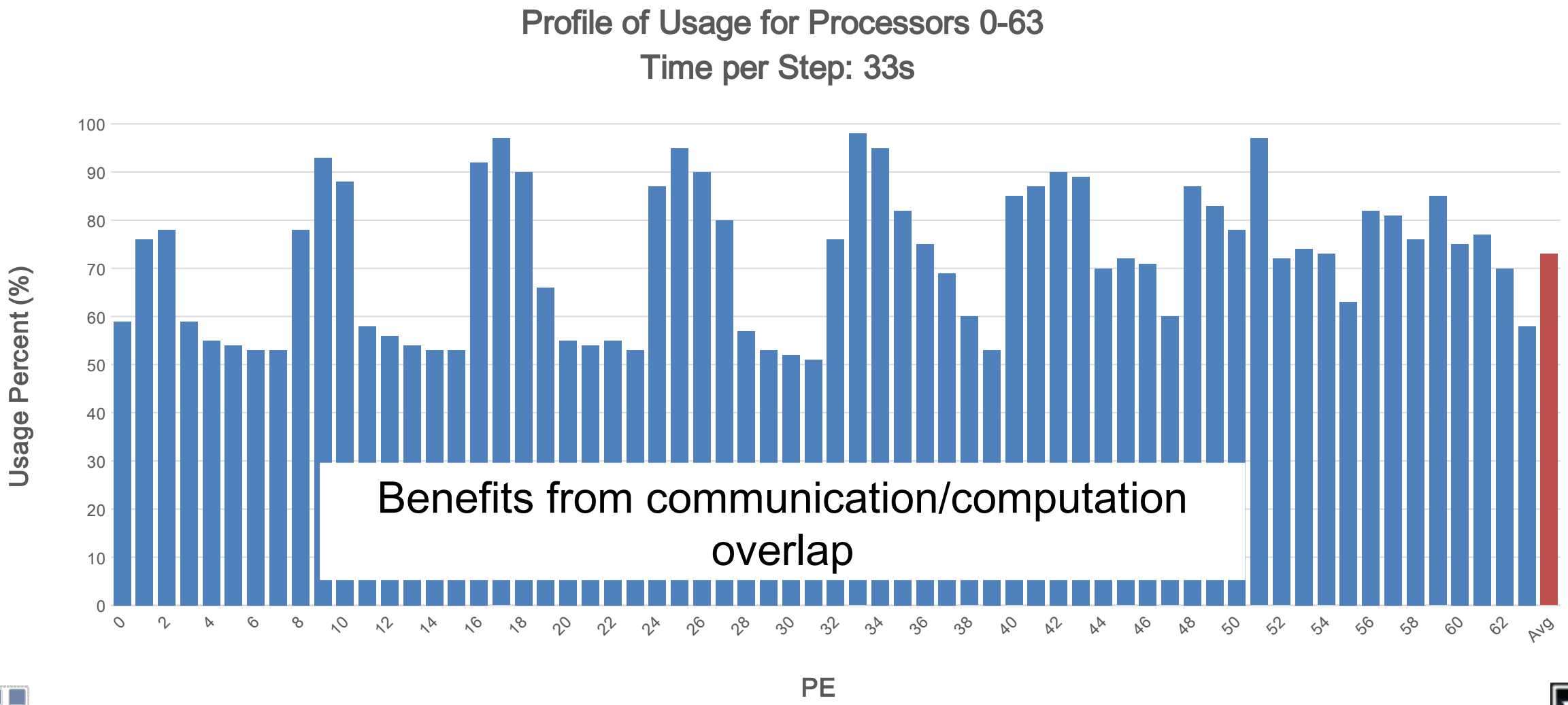
Baseline: 64 Objects

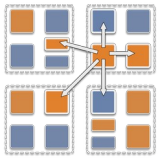




PPL UIUC

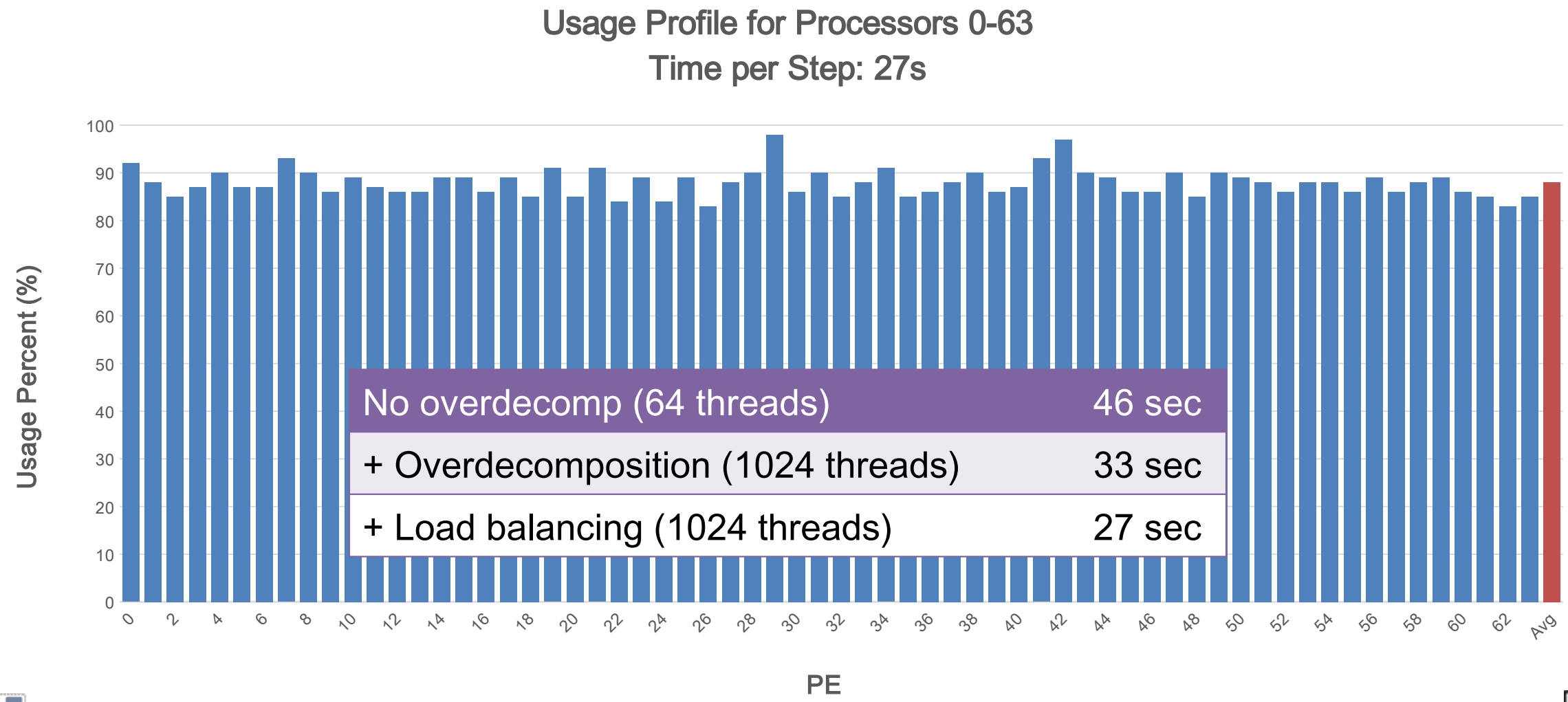
Overdecomposition: 1024 Objects

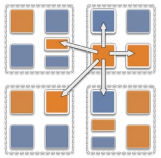




PPL UIUC

With Load Balancing: 1024 objects





PPL UIUC