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
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
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 $k \cdot \nu$ (for some $k$, say 10$\nu$)
    • $\nu$: 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 $k \cdot m \cdot \nu$ (say 100$\nu$)

• Important corollary:
  – You can be at close to optimal grainsize without having to think about $p$, the number of processors
Grainsize in a common setting

Jacobi3D running on JYC using 64 cores on 2 nodes

- 2 MB/chare,
- 256 objects per core
Grainsize: Weather Forecasting in BRAMS

- BRAMS: Brazilian 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

Charm Tutorial
Baseline: 64 Objects

Profile of Usage for Processors 0-63
Time per Step: 46s
Overdecomposition: 1024 Objects

Profile of Usage for Processors 0-63
Time per Step: 33s

Benefits from communication/computation overlap
With Load Balancing: 1024 objects

Usage Profile for Processors 0-63
Time per Step: 27s

No overdecomp (64 threads)  46 sec
+ Overdecomposition (1024 threads)  33 sec
+ Load balancing (1024 threads)  27 sec