Finding Regularity in Irregularities

Assumptions

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

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

*array is x10 faster than list*
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**
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.