Objective: Basics of particle simulation

- Calculus (math) → simulation (science)
- Minimal knowledge to understand pmd.c
# Why MD: Dynamic Irregular Dwarf

## HPC 7 dwarfs

<table>
<thead>
<tr>
<th>Dwarf</th>
<th>Description</th>
<th>Communication Pattern (Figure axes show processors 1 to 256, with black meaning no communication)</th>
<th>NAS Benchmark / Example HW</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Dense Linear Algebra</td>
<td>Data are dense matrices or vectors. (BLAS Level 1 = vector-vector; Level 2 = matrix-vector; and Level 3 = matrix-matrix.) Generally, such applications use unit-stride memory accesses to read data from rows, and strided accesses to read data from columns.</td>
<td><img src="image1" alt="The communication pattern of MadBench, which makes heavy use of ScalAPACK for parallel dense linear algebra, is typical of a much broader class of numerical algorithms" /></td>
<td></td>
</tr>
<tr>
<td>(e.g., BLAS [Blackford et al 2002], ScaLAPACK [Blackford et al 1996], or MATLAB [MathWorks 2006])</td>
<td>Block Triangular Matrix, Lower Upper Symmetric Gaussian-Seidel / Vector computers, Array computers</td>
<td></td>
<td><a href="http://view.eecs.berkeley.edu/wiki">GraPE</a></td>
</tr>
<tr>
<td>2. Sparse Linear Algebra</td>
<td>Data sets include many zero values. Data is usually stored in compressed matrices to reduce the storage and bandwidth requirements to access all of the nonzero values. One example is block compressed sparse row (BCSR). Because of the compressed formats, data is generally accessed with indexed loads and stores.</td>
<td><img src="image2" alt="SuperLU (communication pattern pictured above) uses the BCSR method for implementing sparse LU factorization." /></td>
<td><a href="http://view.eecs.berkeley.edu/wiki">MD-GRAPE</a></td>
</tr>
<tr>
<td>(e.g., SpMV, OSKI [OSKI 2006], or SuperLU [Demmel et al 1999])</td>
<td>Conjugate Gradient / Vector computers with gather/scatter</td>
<td></td>
<td><a href="http://view.eecs.berkeley.edu/wiki">IBM 2006</a></td>
</tr>
<tr>
<td>3. Spectral Methods</td>
<td>Data are in the frequency domain, as opposed to time or spatial domains. Typically, spectral methods use multiple butterfly stages, which combine multiply-add operations and a specific pattern of data permutation, with all-to-all communication for some stages and strictly local for others.</td>
<td><img src="image3" alt="PARATEC: The 3D FFT requires an all-to-all communication to implement a 3D transpose, which requires communication between every link. The diagonal stripe describes BLAS-3 dominated linear-algebra step required for orthogonalization." /></td>
<td><a href="http://view.eecs.berkeley.edu/wiki">BlueGene/L</a></td>
</tr>
<tr>
<td>(e.g., FFT [Cooley and Tukey 1965])</td>
<td>Fourier Transform / DSPs, Zalink PDS [Zallink 2006]</td>
<td></td>
<td><a href="http://view.eecs.berkeley.edu/wiki">Edinburgh 2006</a></td>
</tr>
<tr>
<td>4. N-Body Methods</td>
<td>Depends on interactions between many discrete points. Variations include particle-particle methods, where every point depends on all others, leading to an O(N^2) calculation, and hierarchical particle methods, which combine forces or potentials from multiple points to reduce the computational complexity to O(N log N) or O(N).</td>
<td><img src="image4" alt="PMEMD's communication pattern is that of a particle mesh-Nek5000 calculation." /></td>
<td><a href="http://view.eecs.berkeley.edu/wiki">GraPE</a></td>
</tr>
<tr>
<td>(e.g., Barnes-Hut [Barnes and Hut 1986], Fast Multipole Method [Greengard and Rokhlin 1987])</td>
<td></td>
<td></td>
<td><a href="http://view.eecs.berkeley.edu/wiki">Tokyo 2006</a></td>
</tr>
<tr>
<td>5. Structured Grids</td>
<td>Represented by a regular grid; points on grid are conceptually updated together. It has high spatial locality. Updates may be in place or between 2 versions of the grid. The grid may be subdivided into finer grids in areas of interest (“Adaptive Mesh Refinement”); and the transition between granularities may happen dynamically.</td>
<td><img src="image5" alt="Communication pattern for Cactus, a PDE solver using 7-point stencil on 3D block-structured grids." /></td>
<td><a href="http://view.eecs.berkeley.edu/wiki">QCDDOC</a></td>
</tr>
<tr>
<td>(e.g., Cactus [Goodale et al 2003])</td>
<td>Multi-Grid, Scalar Pentadiagonal / QCDOC [Edinburgh 2006], BlueGene/L</td>
<td></td>
<td><a href="http://view.eecs.berkeley.edu/wiki">BlueGene/L</a></td>
</tr>
<tr>
<td>6. Unstructured Grids</td>
<td>An irregular grid where data locations are selected, usually by underlying characteristics of the application. Data point location and connectivity of neighboring points must be explicit. The points on the grid are conceptually updated together. Updates typically involve multiple levels of memory reference indirection, as an update to any point requires first determining a list of neighboring points, and then loading values from those neighboring points.</td>
<td><img src="image6" alt="Unstructured Adaptive / Vector computers with gather/scatter, Tera Multi Threaded Architecture Berry et al 2006 " /></td>
<td></td>
</tr>
<tr>
<td>(e.g., ABAQUS [ABAQUS 2006] or FIDAP [FLUENT 2006])</td>
<td>Unstructured Adaptive / Vector computers with gather/scatter, Tera Multi Threaded Architecture [Berry et al 2006]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7. Monte Carlo</td>
<td>Calculations depend on statistical results of repeated random trials. Considered embarrassingly parallel.</td>
<td><img src="image7" alt="Emarrassingly Parallel / NSF Teragrid" /></td>
<td><a href="http://view.eecs.berkeley.edu/wiki">Teragrid</a></td>
</tr>
<tr>
<td>(e.g., Quantum Monte Carlo [Aspuru-Guzik et al 2005])</td>
<td>Communication is typically not dominant in Monte Carlo methods.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[http://view.eecs.berkeley.edu/wiki](http://view.eecs.berkeley.edu/wiki)
System: A Set of Point Atoms

\[
\{ \vec{r}_i = (x_i, y_i, z_i) \mid x_i, y_i, z_i \in \mathbb{R}, i = 0, \ldots, N - 1 \}
\]

int n: \( N \), # of atoms
NMAX: Max # of atoms
double r[NMAX][3]: \( r[i][0\mid1\mid2] = x_i\mid y_i\mid z_i \)

pmd.h on the class home page
Trace of atom positions

Velocity

\[ \vec{v}_i(t) = \vec{r}_i'(t) = \frac{d\vec{r}_i}{dt} = \lim_{\Delta \to 0} \frac{\vec{r}_i(t + \Delta) - \vec{r}_i(t)}{\Delta} \]

\[
\text{double } \text{rv[NMAX][3]}: \text{rv[i][0|1|2]} = v_{ix}|y|z
\]

Acceleration

\[ \vec{a}_i(t) = \vec{r}_i''(t) = \frac{d^2\vec{r}_i}{dt^2} = \frac{d\vec{v}_i}{dt} = \lim_{\Delta \to 0} \frac{\vec{v}_i(t + \Delta) - \vec{v}_i(t)}{\Delta} \]

\[
\text{double } \text{ra[NMAX][3]}: \text{ra[i][0|1|2]} = a_{ix}|y|z
\]
Newton’s Equation of Motion

Newton’s 2nd law:

\[ m\ddot{r}_i(t) = \vec{F}_i(t) \]

Initial value problem: Given initial particle positions & velocities, \( \{(\vec{r}_i(0), \vec{v}_i(0))\} \)
Obtain those at later times \( \{(\vec{r}_i(t), \vec{v}_i(t)); t \geq 0\} \)

Potential energy:

\[ \vec{F}_k = -\frac{\partial}{\partial \vec{r}_k} V(\vec{r}^N) = -\left( \frac{\partial V}{\partial x_k}, \frac{\partial V}{\partial y_k}, \frac{\partial V}{\partial z_k} \right) \]

where the partial derivative is

\[ \frac{\partial V}{\partial x_k} = \lim_{h \to 0} \frac{V(x_0, y_0, z_0, \ldots, x_k + h, y_k, z_k, \ldots, x_{N-1}, y_{N-1}, z_{N-1}) - V(x_0, y_0, z_0, \ldots, x_k, y_k, z_k, \ldots, x_{N-1}, y_{N-1}, z_{N-1})}{h} \]

Pair potential:

\[ V(\vec{r}^N) = \sum_{i<j} u(r_{ij}) = \sum_{i=0}^{N-2} \sum_{j=i+1}^{N-1} u(|\vec{r}_{ij}|) \]
Molecular Dynamics Equations

Given initial atomic positions & velocities, \( \{(\vec{r}_i(0), \vec{\nu}_i(0)) \mid i = 0, \ldots, N - 1\} \),

obtain those at later times, \( \{(\vec{r}_i(t), \vec{\nu}_i(t)) \mid i = 0, \ldots, N - 1; t \geq 0\} \),

by integrating the ordinary differential equation,

\[
\ddot{\vec{r}}_k(t) = \vec{\ddot{a}}_k(t) = -\frac{\partial}{\partial \vec{r}_k} \sum_{i \neq j} u(r_{ij}) = \sum_{i \neq j} \vec{r}_{ij}(t) \left( -\frac{1}{r} \frac{du}{dr} \right)_{r=r_{ij}(t)} \left( \delta_{ik} - \delta_{jk} \right)
\]

where

\[
- \frac{1}{r} \frac{du}{dr} = \frac{48}{r^2 \left( \frac{1}{r^{12}} - \frac{1}{2r^6} \right)}
\]

(for the Lennard-Jones potential in a dimensionless unit)

Force calculation algorithm—\( O(N^2) \):

for \( i = 0 \) to \( N-1 \), \( \vec{a}_i = 0 \)

for \( i = 0 \) to \( N-2 \)

for \( j = i+1 \) to \( N-1 \)

compute \( \vec{a} = \vec{r}_{ij} \left( -\frac{1}{r} \frac{du}{dr} \right)_{r=|\vec{r}_{ij}|} \)

\( \vec{a}_{i+} = \vec{\alpha} \)

\( \vec{a}_{j-} = \vec{\alpha} \)
**Time Discretization**

**Snapshots with time interval Δ:**

\[
(\vec{r}_i(0), \vec{v}_i(0)) \mapsto (\vec{r}_i(\Delta), \vec{v}_i(\Delta)) \mapsto (\vec{r}_i(2\Delta), \vec{v}_i(2\Delta)) \mapsto \ldots
\]

**Question:** How to predict the next state, \((\vec{r}_i(t + \Delta), \vec{v}_i(t + \Delta))\), from the current state, \((\vec{r}_i(t), \vec{v}_i(t))\)？

**Solution:** Taylor expansion

\[
f(x_0 + h) = \sum_{n=0}^{\infty} \frac{h^n}{n!} \frac{d^n f}{dx^n} \bigg|_{x=x_0} = f(x_0) + \frac{df}{dx} \bigg|_{x=x_0} + \frac{h^2}{2} \frac{d^2 f}{dx^2} \bigg|_{x=x_0} + \frac{h^3}{3!} \frac{d^3 f}{dx^3} \bigg|_{x=x_0} + \ldots
\]

Verlet Discretization

**Position:**

\[
\begin{align*}
\vec{r}_i(t + \Delta) &= \vec{r}_i(t) + \vec{v}_i(t)\Delta + \frac{1}{2} \ddot{a}_i(t)\Delta^2 + \frac{1}{6} \dddot{r}_i(t)\Delta^3 + O(\Delta^4) \\
\vec{r}_i(t - \Delta) &= \vec{r}_i(t) - \vec{v}_i(t)\Delta + \frac{1}{2} \ddot{a}_i(t)\Delta^2 - \frac{1}{6} \dddot{r}_i(t)\Delta^3 + O(\Delta^4)
\end{align*}
\]

\[
\vec{r}_i(t + \Delta) + \vec{r}_i(t - \Delta) = 2\vec{r}_i(t) + \ddot{a}_i(t)\Delta^2 + O(\Delta^4)
\]

\[
\therefore \vec{r}_i(t + \Delta) = 2\vec{r}_i(t) - \vec{r}_i(t - \Delta) + \ddot{a}_i(t)\Delta^2 + O(\Delta^4)
\]

**Velocity:**

\[
\begin{align*}
\vec{r}_i(t + \Delta) &= \vec{r}_i(t) + \vec{v}_i(t)\Delta + \frac{1}{2} \ddot{a}_i(t)\Delta^2 + \frac{1}{6} \dddot{r}_i(t)\Delta^3 + O(\Delta^4) \\
\vec{r}_i(t - \Delta) &= \vec{r}_i(t) - \vec{v}_i(t)\Delta + \frac{1}{2} \ddot{a}_i(t)\Delta^2 - \frac{1}{6} \dddot{r}_i(t)\Delta^3 + O(\Delta^4)
\end{align*}
\]

\[
\vec{r}_i(t + \Delta) - \vec{r}_i(t - \Delta) = 2\vec{v}_i(t)\Delta + O(\Delta^3)
\]

\[
\therefore \vec{v}_i(t) = \frac{\vec{r}_i(t + \Delta) - \vec{r}_i(t - \Delta)}{2\Delta} + O(\Delta^2)
\]
Verlet Algorithm

Verlet discretization:

\[
\begin{align*}
\vec{r}_i(t + \Delta) &= 2\vec{r}_i(t) - \vec{r}_i(t - \Delta) + \vec{a}_i(t)\Delta^2 + O(\Delta^4) \\
\vec{v}_i(t) &= \frac{\vec{r}_i(t + \Delta) - \vec{r}_i(t - \Delta)}{2\Delta} + O(\Delta^2)
\end{align*}
\]

Verlet algorithm:

Given \( \vec{r}_i(t - \Delta) \) & \( \vec{r}_i(t) \),

1. Compute \( \vec{a}_i(t) \) as a function of \( \{\vec{r}_i(t)\} \)
2. \( \vec{r}_i(t + \Delta) \leftarrow 2\vec{r}_i(t) - \vec{r}_i(t - \Delta) + \vec{a}_i(t)\Delta^2 \)
3. \( \vec{v}_i(t) \leftarrow [\vec{r}_i(t + \Delta) - \vec{r}_i(t - \Delta)]/(2\Delta) \)

Drawback: Positions & velocities are not simultaneously updated for the same time step
Velocity Verlet Algorithm

**Theorem:** The following algebraic equation gives the same sequence of states, 
\((\vec{r}_i(n\Delta),\vec{v}_i(n\Delta))\), as that obtained by the Verlet discretization.

\[
\begin{align*}
\vec{r}_i(t+\Delta) &= \vec{r}_i(t) + \vec{v}_i(t)\Delta + \frac{1}{2}\ddot{a}_i(t)\Delta^2 \\
\vec{v}_i(t+\Delta) &= \vec{v}_i(t) + \frac{\ddot{a}_i(t) + \ddot{a}_i(t+\Delta)}{2}\Delta
\end{align*}
\]

**Velocity Verlet algorithm:**

Given \((\vec{r}_i(t),\vec{v}_i(t))\),

1. Compute \(\ddot{a}_i(t)\) as a function of \(\{\vec{r}_i(t)\}\)
2. \(\vec{v}_i(t + \frac{\Delta}{2}) \leftarrow \vec{v}_i(t) + \frac{\Delta}{2} \ddot{a}_i(t)\)
3. \(\vec{r}_i(t + \Delta) \leftarrow \vec{r}_i(t) + \vec{v}_i(t + \frac{\Delta}{2})\Delta\)
4. Compute \(\ddot{a}_i(t + \Delta)\) as a function of \(\{\vec{r}_i(t + \Delta)\}\)
5. \(\vec{v}_i(t + \Delta) \leftarrow \vec{v}_i(t + \frac{\Delta}{2}) + \frac{\Delta}{2} \ddot{a}_i(t + \Delta)\)
Velocity Verlet Algorithm for StepLimit Steps

Initialize \( (\vec{r}_i, \vec{v}_i) \) for all \( i \)

Compute \( \vec{a}_i \) for all \( i \) as a function of \( \{\vec{r}_i\} \) function computeAccel()

for stepCount = 1 to StepLimit

    do the following function singleStep()

        \( \vec{v}_i \leftarrow \vec{v}_i + \vec{a}_i \Delta \) / 2 for all \( i \)

        \( \vec{r}_i \leftarrow \vec{r}_i + \vec{v}_i \Delta \) for all \( i \)

    Compute \( \vec{a}_i \) for all \( i \) as a function of \( \{\vec{r}_i\} \) function computeAccel()

        \( \vec{v}_i \leftarrow \vec{v}_i + \vec{a}_i \Delta \) / 2 for all \( i \)

endfor

stepLimit+1 calls to function computeAccel()
Linked-List Cell Molecular Dynamics

- Computational complexity of `ComputeAccel()` in `md.c`:
  \[ \propto N(N-1)/2 = O(N^2) \]

- Data locality (cut-off length, \( r_c \)) reduces the complexity to \( O(N) \):
  \[ N \times \left(\frac{4\pi}{3}\right) r_c^3 \times \left(\frac{N}{V}\right) \]

- \( O(N) \) algorithm uses: (1) spatially localized cells; & (2) linked lists to keep track of atoms’ cell membership

  Data-local migration path to parallel MD

Prune the search space!
Cell Data Structures

Data-local spatial indexing scheme = cell + linked list

- Cell size
  \[ L_{c\alpha} = \lfloor L_\alpha / r_{c\alpha} \rfloor, \text{ int } lc[3] \]
  \[ r_{c\alpha} = L_\alpha / L_{c\alpha}, (\alpha = x, y, z), \text{ double } rc[3] \]

  where

  \( L_\alpha \): simulation box length
    \[(\text{double Region[3]})\]
  \( r_c \): Cut-off length (RCUT)

- Vector cell index, \( 0 \leq c_\alpha \leq L_{cx}-1 \)

- Serial cell index:
  \[ c = c_x L_{cy} L_{cz} + c_y L_{cz} + c_z \]
  or
  \[ c_x = \lfloor c / (L_{cy} L_{cz}) \rfloor \]
  \[ c_y = \lfloor c / L_{cz} \rfloor \mod L_{cy} \]
  \[ c_z = c \mod L_{cz} \]

- Atom-to-cell mapping:
  \[ c_\alpha = \lfloor r_\alpha / r_{c\alpha} \rfloor \]
Linked Lists

Data Structures:

lscl[NMAX]: Linked lists; lscl[i] holds the atom index to which the i-th atom points.

head[NCLMAX]: head[c] holds the index of the first atom in the c-th cell, or head[c] = EMPTY (= -1) if there is no atom in the cell.
/* Reset the headers, head */
for (c=0; c<lcxyz; c++) head[c] = EMPTY;

/* Scan atoms to construct headers, head, & linked lists, lscl */
for (i=0; i<nAtom; i++) {
    /* Vector cell index to which this atom belongs */
    for (a=0; a<3; a++) mc[a] = r[i][a]/rc[a];
    /* Translate the vector cell index, mc, to a scalar cell index */
    c = mc[0]*lcyz+mc[1]*lc[2]+mc[2];
    /* Link to the previous occupant (or EMPTY if you're the 1st) */
    lscl[i] = head[c];
    /* The last one goes to the header */
    head[c] = i;
}

where
lcyz = lc[1]*lc[2]
lcxyz = lcyz*lc[0]

**Single pass over atoms → O(N)!**
**O(N) Force Calculation Algorithm**

```plaintext
for (mc[0]=0; mc[0]<lc[0]; (mc[0])++)
for (mc[1]=0; mc[1]<lc[1]; (mc[1])++)
for (mc[2]=0; mc[2]<lc[2]; (mc[2])++) { /* Scan inner cells */
c = mc[0]*lcyz+mc[1]*lc[2]+mc[2]; /* Calculate a scalar cell index */
for (mc1[0]=mc[0]-1; mc1[0]<=mc[0]+1; (mc1[0])++)
for (mc1[1]=mc[1]-1; mc1[1]<=mc[1]+1; (mc1[1])++)
for (a=0; a<3; a++) { /* Unwrapping the periodic boundary condition */
    if (mc1[a] < 0)
        rshift[a] = -Region[a];
    else if (mc1[a]>=lc[a])
        rshift[a] = Region[a];
    else
        rshift[a] = 0.0;
}
    c1 = (((mc1[0]+1c[0])%lc[0])*lcyz 
        +((mc1[1]+1c[1])%lc[1])*lc[2] 
        +((mc1[2]+1c[2])%lc[2])); /* Scalar cell index of the neighbor cell */
i = head[c]; /* Scan atom i in cell c */
while (i != EMPTY) {
    j = head[c1]; /* Scan atom j in cell c1 */
    while (j != EMPTY) {
        if (i < j) { /* Avoid double counting of pair (i, j) */
            r_{ij} = r_i-(r_j+r_{shift}); /* Image-corrected relative pair position */
            if (r_{ij} < r_c^2)
                Compute forces on pair (i, j)
        }
    }
    j = lscl[j]; /* Follow the chain of linked atoms in neighbor cell */
}
i = lscl[i]; /* Follow the chain of linked atoms in central cell */
}
```

---

**O(N)**

**O(1)**
Scaling of Linked-List Cell MD

$\log(T_{md}) = -5.4606 + 1.9516 \log(n\text{Atom})$

$\log(T_{lmd}) = -2.8429 + 0.85896 \log(n\text{Atom})$