

# Dinámica Molecular

**Una pequeña introducción a la  
computación de alto desempeño**

Pablo N. Alcain

WTPC 2018 - UNQ

# ¿Qué es la dinámica molecular?

Uno de los *benchmarks* informales en HPC

Física simple: partículas que interactúan entre sí con Newton

Mucha versatilidad

Núcleo de ejecución crítico en tiempo de fácil identificación

# ¿Qué es la dinámica molecular?

Uno de los *benchmarks* informales en HPC

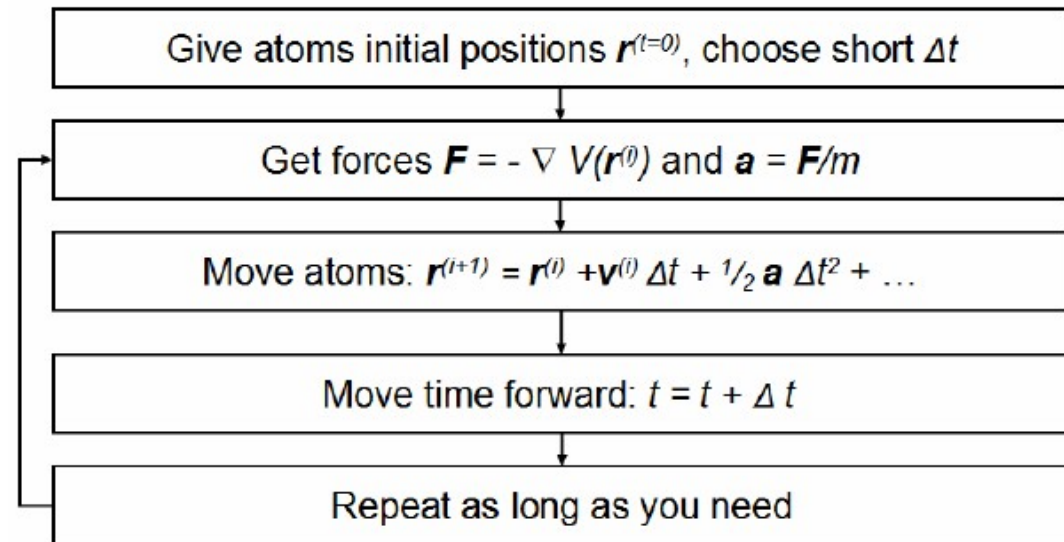
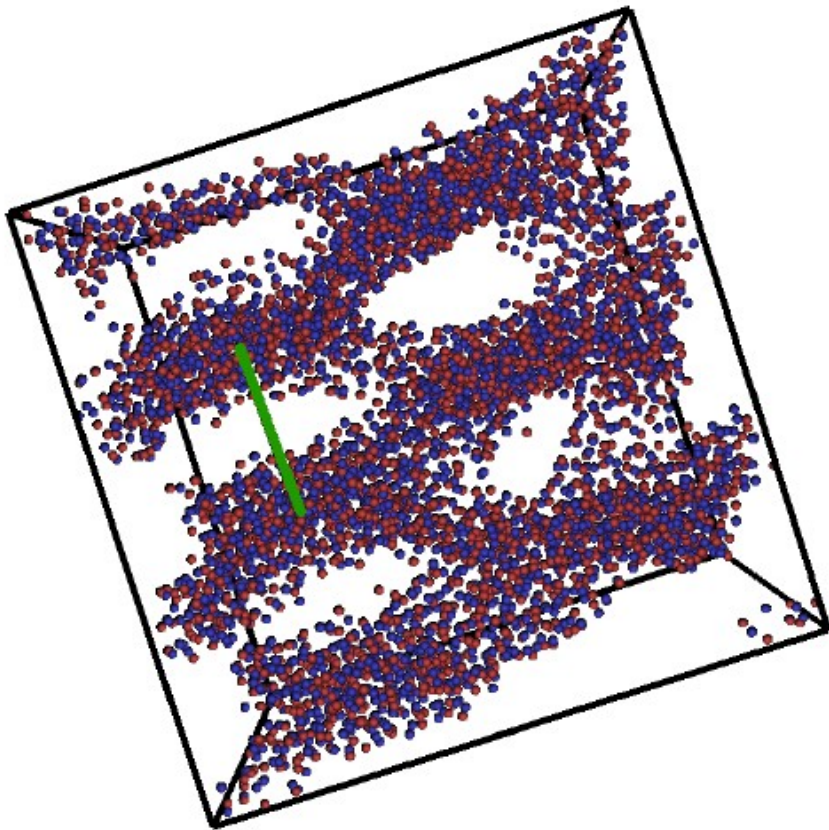
Física simple: partículas que interactúan entre sí con Newton

Mucha versatilidad

Núcleo de ejecución crítico en tiempo de fácil identificación

Ideal para hacer una **interfaz de C/Python**

# ¿Qué es la dinámica molecular?



# Con qué arrancamos

## Primera opción

Núcleo crítico escrito en C

Separado en estructuras

~340 líneas de código

Buena performance en rt

```
File Edit View Search Terminal Help
File Edit Options Buffers Tools C Help
1#include "force.h"
2
3void newton(System *sys, CellList *clist) {
4    double epot = 0.0;
5    for (int i = 0; i < 3 * sys->nthreads * sys->n_particles; i++)
6        sys->force[i] = 0.0;
7    #pragma omp parallel reduction(+:epot)
8    {
9        for (int cc = 0; cc < clist->ncells; cc+=sys->nthreads) {
10            int tid = omp_get_thread_num();
11            int c = cc + tid;
12            if (c >= clist->ncells) continue;
13            Cell cell = clist->list[c];
14            for (int ii = 0; ii < cell.n_particles; ii++) {
15                for (int jj = ii + 1; jj < cell.n_particles; jj++) {
16                    int i = cell.particles[ii];
17                    int j = cell.particles[jj];
18                    double dr[3];
19                    for (int k = 0; k < 3; k++)
20                        dr[k] = sys->position[3*i+k] - sys->position[3*j+k];
21                    minimum_images(sys, dr);
22                    epot += calculate_force(sys, i, j, dr, tid);
23                }
24            }
25            for (int d = 0; d < cell.n_neigh; d++) {
26                Cell cell2 = clist->list[cell.neigh[d]];
27                for (int ii = 0; ii < cell.n_particles; ii++) {
28                    for (int jj = 0; jj < cell2.n_particles; jj++) {
29                        int i = cell.particles[ii];
30                        int j = cell2.particles[jj];
31                        double dr[3];
32                        for (int k = 0; k < 3; k++)
33                            dr[k] = sys->position[3*i+k] - sys->position[3*j+k];
34                        minimum_images(sys, dr);
35                        epot += calculate_force(sys, i, j, dr, tid);
36                    }
37                }
38            }
39        }
40    }
41    sys->potential = epot;
42    for (int i = 1; i < sys->nthreads; i++) {
43        int offset = 3 * i * sys->n_particles;
44        for (int j = 0; j < 3 * sys->n_particles; j++) {
45            sys->force[j] += sys->force[j] + offset;
46        }
47    }
48 }
49 #attribute__((always inline,pure))
50 inline void minimum_images(System *sys, double *dr) {
- 2.7k BU - force.c C:/W3C AC FlyC:0/1 Abbrev < Git-master
```

# Con qué arrancamos

## Segunda opción

Interfaz con el usuario en Python

Separado en clases

~400 líneas de código

~300 líneas de test

Interfaz versátil

```
File Edit View Search Terminal Help
File Edit Options Buffers Tools C Help
1#include "force.h"
2
3void newton(System *sys, CellList *clist) {
4  double epot = 0.0;
5  for (int i = 0; i < 3 * sys->nthreads * sys->n_particles; i++)
6    sys->force[i] = 0.0;
7  #pragma omp parallel reduction(+:epot)
8  {
9    for (int cc = 0; cc < clist->ncells; cc+=sys->nthreads) {
10     int tid = omp_get_thread_num();
11     int c = cc + tid;
12     if (c >= clist->ncells) continue;
13     Cell cell = clist->list[c];
14     for (int ii = 0; ii < cell.n_particles; ii++) {
15       for (int jj = ii + 1; jj < cell.n_particles; jj++) {
16         int i = cell.particles[ii];
17         int j = cell.particles[jj];
18         double dr[3];
19         for (int k = 0; k < 3; k++)
20           dr[k] = sys->position[3*i+k] - sys->position[3*j+k];
21         minimum_images(sys, dr);
22         epot += calculate_force(sys, i, j, dr, tid);
23       }
24     }
25     for (int d = 0; d < cell.nneigh; d++) {
26       Cell cell2 = clist->list[cell.nneigh[d]];
27       for (int ii = 0; ii < cell.n_particles; ii++) {
28         for (int jj = 0; jj < cell2.n_particles; jj++) {
29           int i = cell.particles[ii];
30           int j = cell2.particles[jj];
31           double dr[3];
32           for (int k = 0; k < 3; k++)
33             dr[k] = sys->position[3*i+k] - sys->position[3*j+k];
34           minimum_images(sys, dr);
35           epot += calculate_force(sys, i, j, dr, tid);
36         }
37       }
38     }
39   }
40 }
41 sys->potential = epot;
42 for (int i = 1; i < sys->nthreads; i++) {
43   int offset = 3 * i * sys->n_particles;
44   for (int j = 0; j < 3 * sys->n_particles; j++) {
45     sys->force[j] += sys->force[j] + offset;
46   }
47 }
48 }
49 #attribute__((always inline,pure))
50 inline void minimum_images(System *sys, double *dr) {
- 2.7k BU - force.c C:\WSC AC FlyC:0\1 Abbrev < Git-master
```

# Con qué arrancamos

## Segunda opción

Interfaz con el usuario en Python

Separado en clases

~400 líneas de código

~300 líneas de test

Interfaz versátil

```
File Edit View Search Terminal Help
File Edit Options Buffers Tools C Help
1#include "force.h"
2
3void newton(System *sys, CellList *clist) {
4  double epot = 0.0;
5  for (int i = 0; i < 3 * sys->nthreads + sys->n_particles; i++)
6    sys->force[i] = 0.0;
7  #pragma omp parallel reduction(+:epot)
8  {
9    for (int cc = 0; cc < clist->ncells; cc+=sys->nthreads) {
10     int tid = omp_get_thread_num();
11     int c = cc + tid;
12     if (c >= clist->ncells) continue;
13     Cell cell = clist->list[c];
14     for (int ii = 0; ii < cell.n_particles; ii++) {
15       for (int jj = ii + 1; jj < cell.n_particles; jj++) {
16         int i = cell.particles[ii];
17         int j = cell.particles[jj];
18         double dr[3];
19         for (int k = 0; k < 3; k++)
20           dr[k] = sys->position[3*i+k] - sys->position[3*j+k];
21         minimum_images(sys, dr);
22         epot += calculate_force(sys, i, j, dr, tid);
23       }
24     }
25     for (int d = 0; d < cell.nneigh; d++) {
26       Cell cell2 = clist->list[cell.nneigh[d]];
27       for (int ii = 0; ii < cell.n_particles; ii++) {
28         for (int jj = 0; jj < cell2.n_particles; jj++) {
29           int i = cell.particles[ii];
30           int j = cell2.particles[jj];
31           double dr[3];
32           for (int k = 0; k < 3; k++)
33             dr[k] = sys->position[3*i+k] - sys->position[3*j+k];
34           minimum_images(sys, dr);
35           epot += calculate_force(sys, i, j, dr, tid);
36         }
37       }
38     }
39   }
40 }
41 sys->potential = epot;
42 for (int i = 1; i < sys->nthreads; i++) {
43   int offset = 3 * i * sys->n_particles;
44   for (int j = 0; j < 3 * sys->n_particles; j++) {
45     sys->force[j] += sys->force[j] + offset;
46   }
47 }
48 }
49 #attribute__((always inline,pure))
50 inline void minimum_images(System *sys, double *dr) {
- 2.7k BU - force.c C:\WSC AC FlyC:0\1 Abbrev < Git-master
```



# Qué pueden hacer

Generar la interfaz de C con Python

Graficar (¿en tiempo real?) energías [magnitudes colectivas]

Hacer una linda interfaz Python/usuario (¿gráfica?)

Agregar otros tipos de interacciones entre partículas

Medir otras magnitudes colectivas



# Dinámica Molecular

**Una pequeña introducción a la  
computación de alto desempeño**

Pablo N. Alcain

WTPC 2018 - UNQ