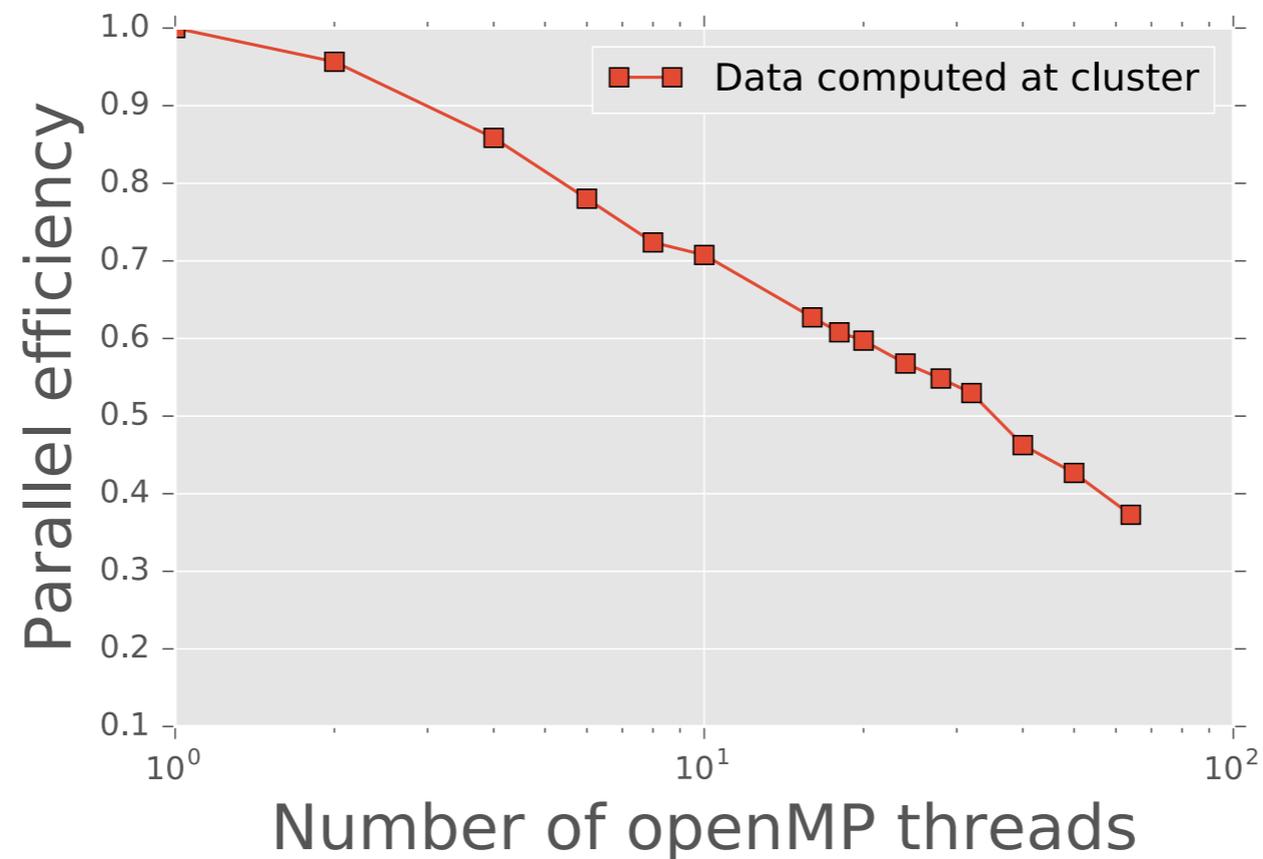


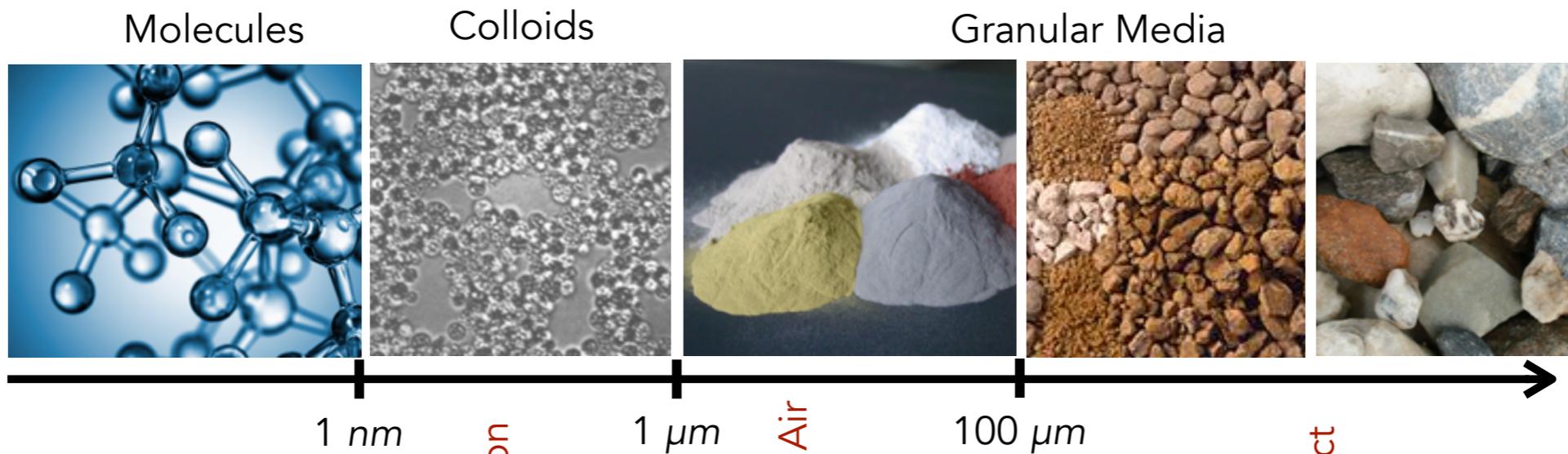
# From scratch to optimal number of threads: DEM and openMP

William Oquendo, [woquendo@gmail.com](mailto:woquendo@gmail.com)



# **THE PROBLEM**

**Why did I use my code?**

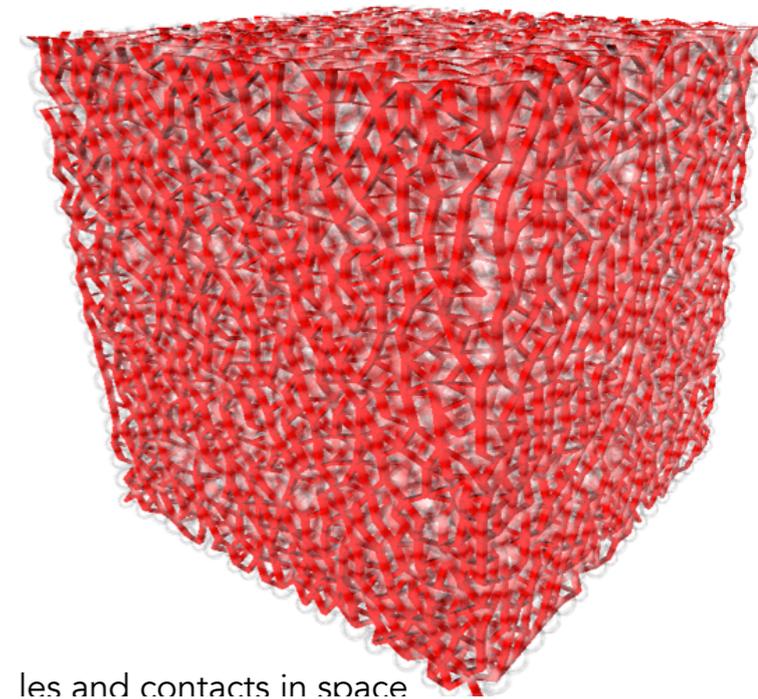
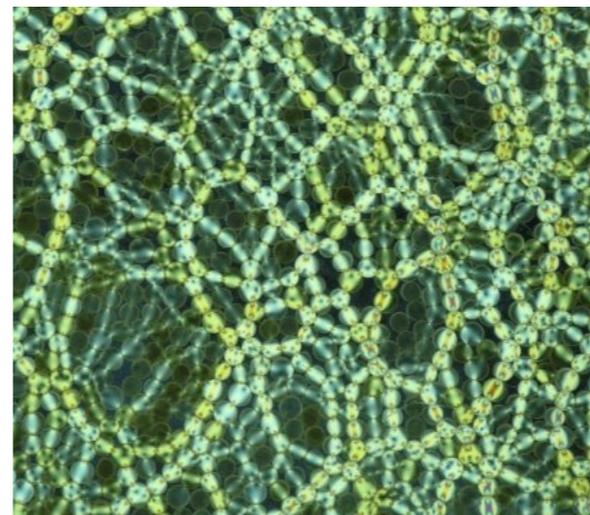
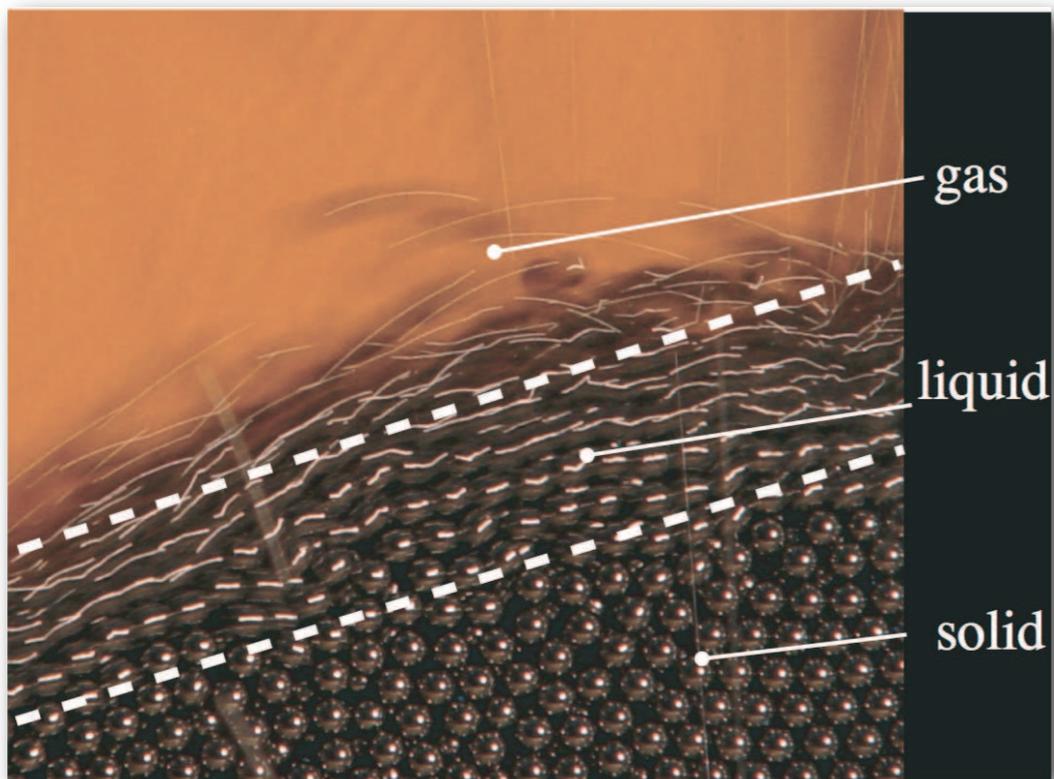


Thermal agitation  
VdW Forces

Interactions with Air  
VdW Forces  
Humidity

Frictional contact

**Frictional  
Characteristic particle size**



les and contacts in space

# **THE SERIAL VERSION**

**My code is complex, let's implement a "simpler" version**

From scratch ... Maybe not a good idea!  
(about 400 loc)

```
2 #include "random_helper.h"  
3 #include "config.h"  
4 #include "types_dem.h"  
5 #include "prepro.h"  
6 #include "helper_dem.h"  
7 #include "time_evolution.h"  
8 #include "forces.h"
```

From scratch ... Maybe not a good idea!  
(about 400 loc)

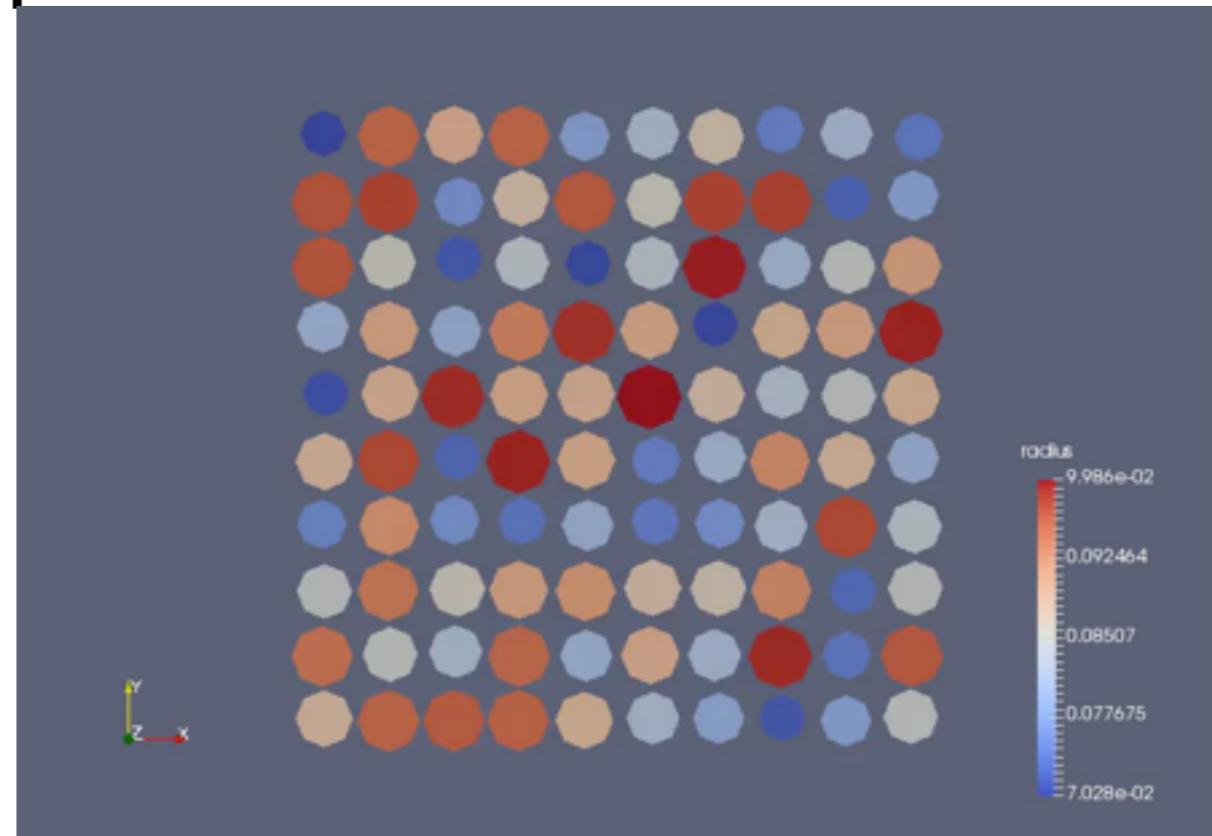
```
2 #include "random_helper.h"  
3 #include "config.h"  
4 #include "types_dem.h"  
5 #include "prepro.h"  
6 #include "helper_dem.h"  
7 #include "time_evolution.h"  
8 #include "forces.h"
```

- **Preprocessing : Put particles on a grid, disorder, random vets, assign radii, set fixed quantities**
- **Fix Vx and Fy on top wall (simulates simple shear)**
- **Use leap frog for time evolution**
- **Periodic boundary conditions (horizontal)**
- **Postprocessing: Visualization with python + paraview**

# From scratch ... Maybe not a good idea! (about 400 loc)

```
2 #include "random_helper.h"  
3 #include "config.h"  
4 #include "types_dem.h"  
5 #include "prepro.h"  
6 #include "helper_dem.h"  
7 #include "time_evolution.h"  
8 #include "forces.h"
```

- **Preprocessing : Put particles on a grid, disorder, random vets, assign radii, set fixed quantities**
- **Fix Vx and Fy on top wall (simulates simple shear)**
- **Use leap frog for time evolution**
- **Periodic boundary conditions (horizontal)**
- **Postprocessing: Visualization with python + paraview**





# PARALLEL (OMP) OPTIMAL NUMBER OF THREADS

# First parallel approach : openmp

## What is the ideal number of threads?

```
22 #pragma omp parallel for private(idx)
23     for (idx = 0; idx < ngrains; ++idx) {
24         for (int jdx = idx+1; jdx < ngrains; ++jdx) {
```

# First parallel approach : openmp

## What is the ideal number of threads?

```
22 #pragma omp parallel for private(idx)
23     for (idx = 0; idx < ngrains; ++idx) {
24         for (int jdx = idx+1; jdx < ngrains; ++jdx) {
```

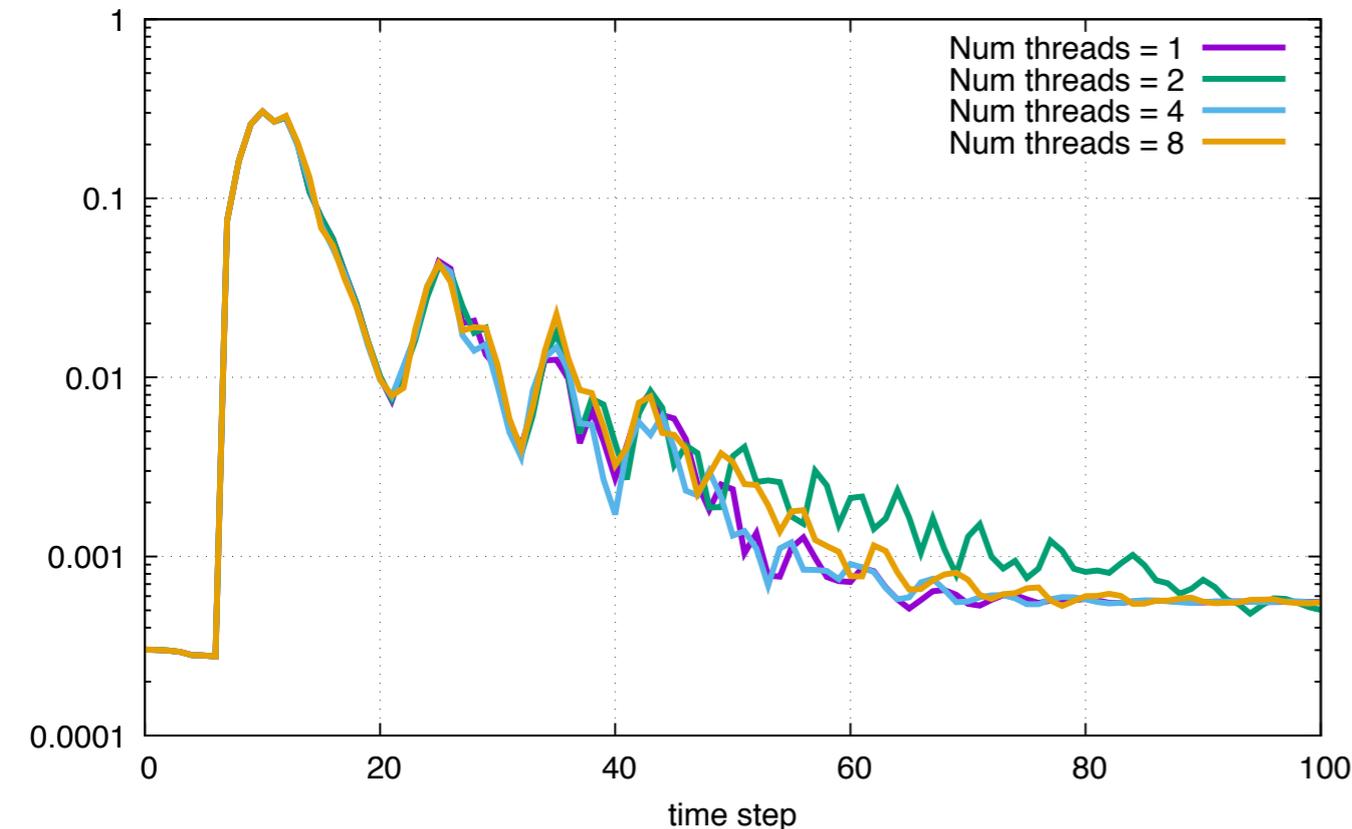
```
21     double dx, dy, rij, urij, delta, dVx, dVy, tmp, Fnx, Fny;
22     int ii, jj;
23 #pragma omp parallel for shared(grains, conf) private(ii, jj, dx, dy, rij, urij, delta, dVx, dVy, tmp, Fnx, Fny)
24     for (ii = 0; ii < ngrains; ++ii) {
25         for (jj = ii+1; jj < ngrains; ++jj) {
26             dx = grains[ii].R[0] - grains[jj].R[0];
27             //if (dx <= conf.LX/2) dx += conf.LX;
```

# First parallel approach : openmp

## What is the ideal number of threads?

```
22 #pragma omp parallel for private(idx)
23   for (idx = 0; idx < ngrains; ++idx) {
24     for (int jdx = idx+1; jdx < ngrains; ++jdx) {
```

```
21 double dx, dy, rij, urij, delta, dVx, dVy, tmp, Fnx, Fny;
22 int ii, jj;
23 #pragma omp parallel for shared(grains, conf) private(ii, jj, dx, dy, rij, urij, delta, dVx, dVy, tmp, Fnx, Fny)
24   for (ii = 0; ii < ngrains; ++ii) {
25     for (jj = ii+1; jj < ngrains; ++jj) {
26       dx = grains[ii].R[0] - grains[jj].R[0];
27       //if (dx <= conf.LX/2) dx += conf.LX;
```



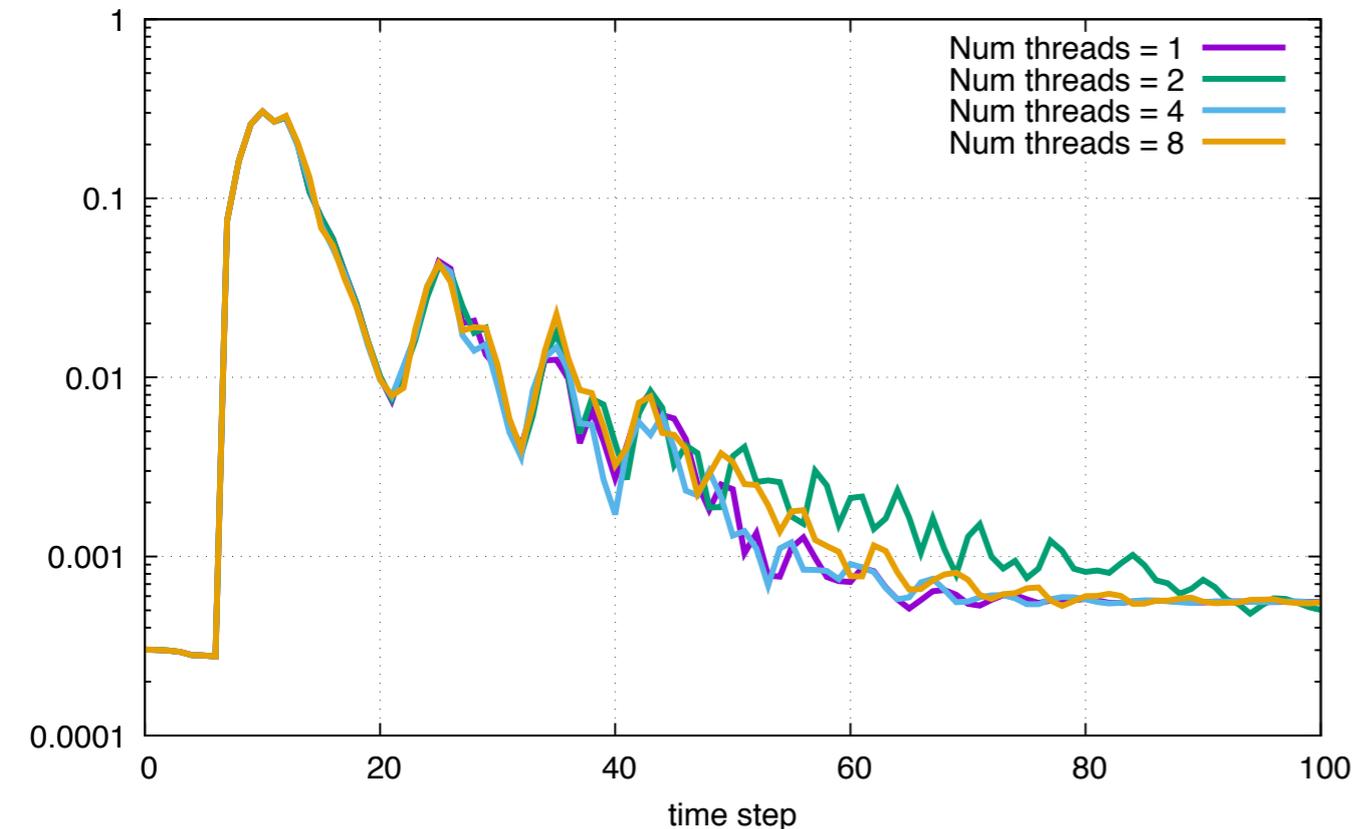
# First parallel approach : openmp

## What is the ideal number of threads?

```
22 #pragma omp parallel for private(idx)
23   for (idx = 0; idx < ngrains; ++idx) {
24     for (int jdx = idx+1; jdx < ngrains; ++jdx) {
```

```
21 double dx, dy, rij, urij, delta, dVx, dVy, tmp, Fnx, Fny;
22 int ii, jj;
23 #pragma omp parallel for shared(grains, conf) private(ii, jj, dx, dy, rij, urij, delta, dVx, dVy, tmp, Fnx, Fny)
24 for (ii = 0; ii < ngrains; ++ii) {
25   for (jj = ii+1; jj < ngrains; ++jj) {
26     dx = grains[ii].R[0] - grains[jj].R[0];
27     //if (dx <= conf.LX/2) dx += conf.LX;
```

**But time appears to be slower for more threads :(**



# First parallel approach : openmp

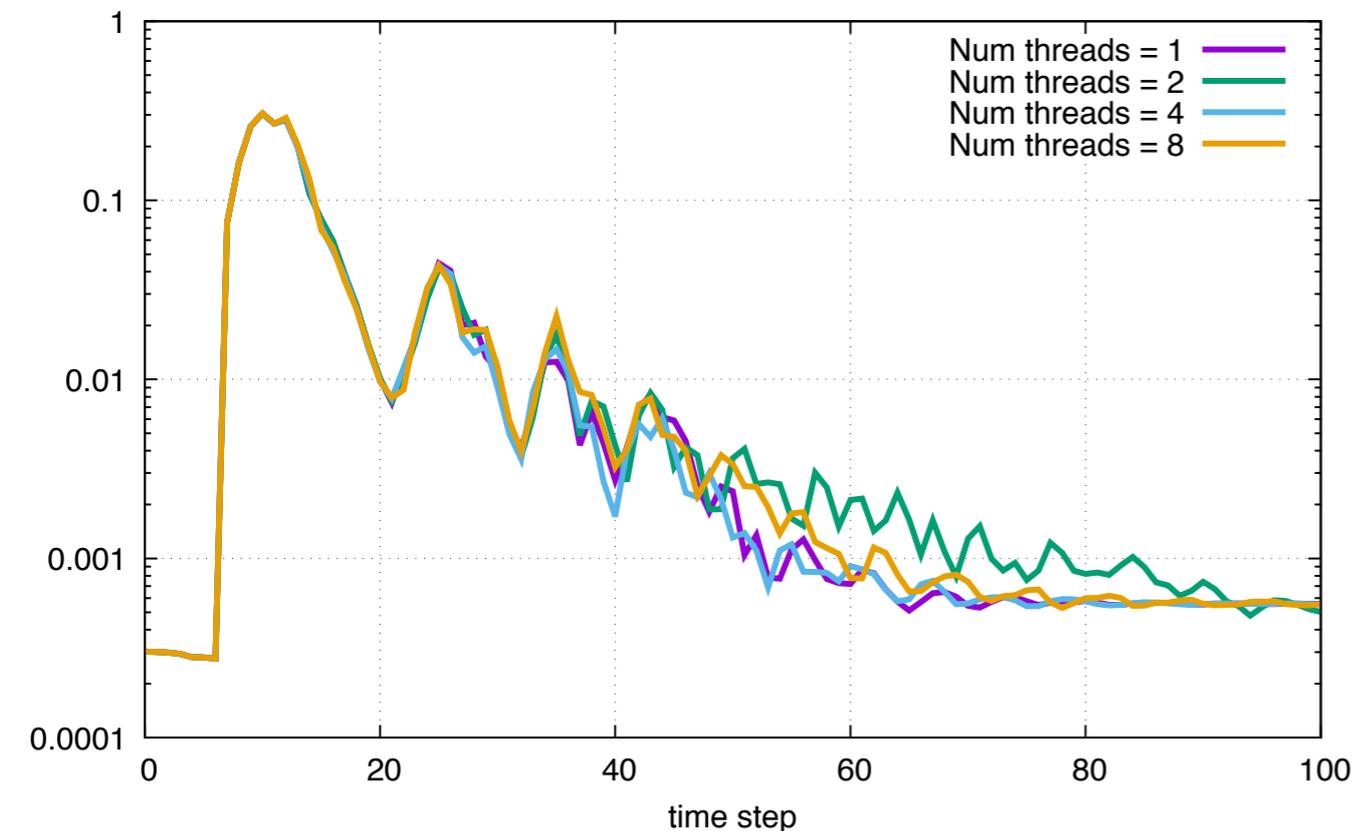
## What is the ideal number of threads?

```
22 #pragma omp parallel for private(idx)
23   for (idx = 0; idx < ngrains; ++idx) {
24     for (int jdx = idx+1; jdx < ngrains; ++jdx) {
```

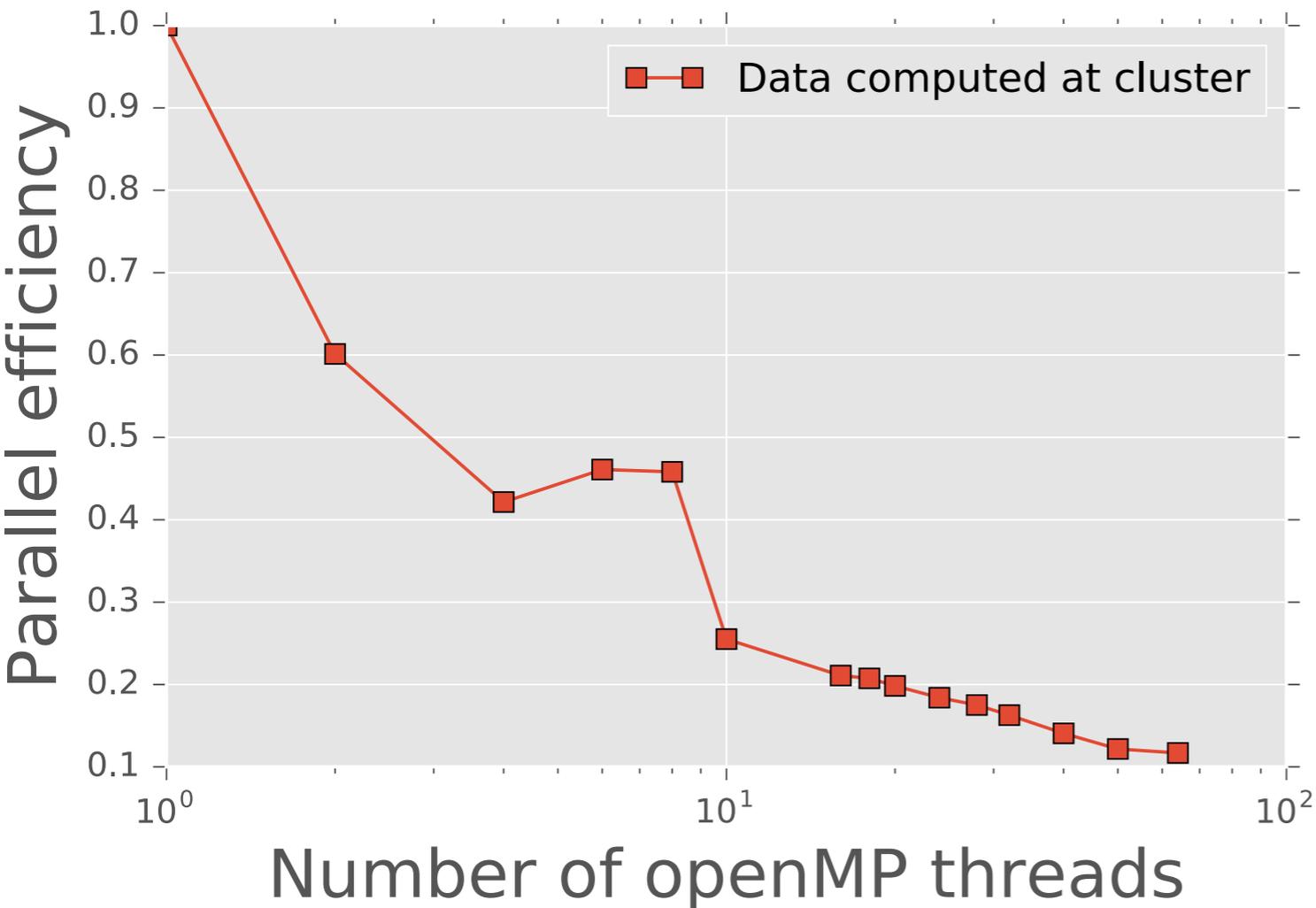
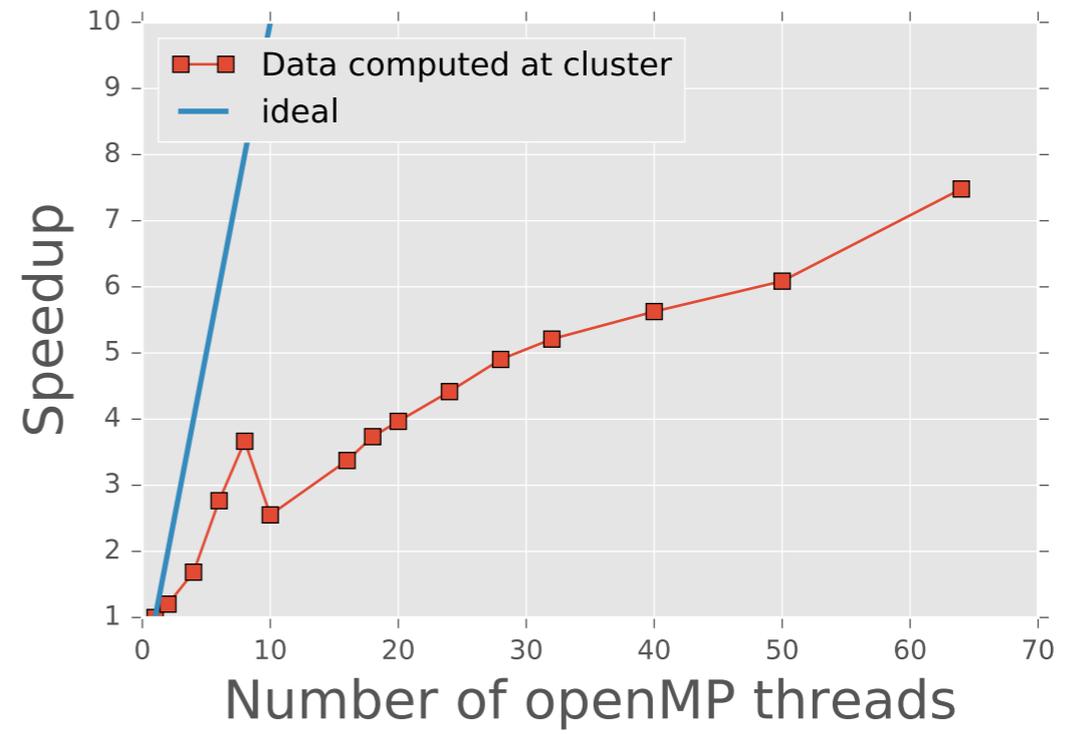
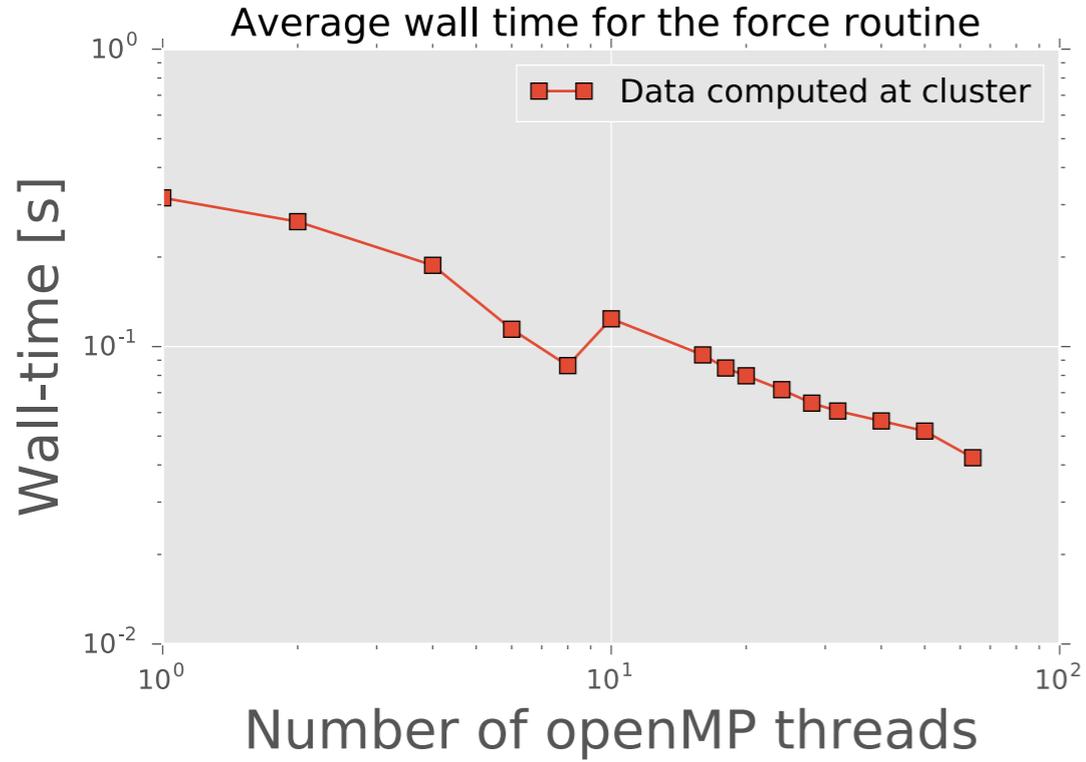
```
21 double dx, dy, rij, urij, delta, dVx, dVy, tmp, Fnx, Fny;
22 int ii, jj;
23 #pragma omp parallel for shared(grains, conf) private(ii, jj, dx, dy, rij, urij, delta, dVx, dVy, tmp, Fnx, Fny)
24   for (ii = 0; ii < ngrains; ++ii) {
25     for (jj = ii+1; jj < ngrains; ++jj) {
26       dx = grains[ii].R[0] - grains[jj].R[0];
27       //if (dx <= conf.LX/2) dx += conf.LX;
```

**But time appears to be slower for more threads :(**

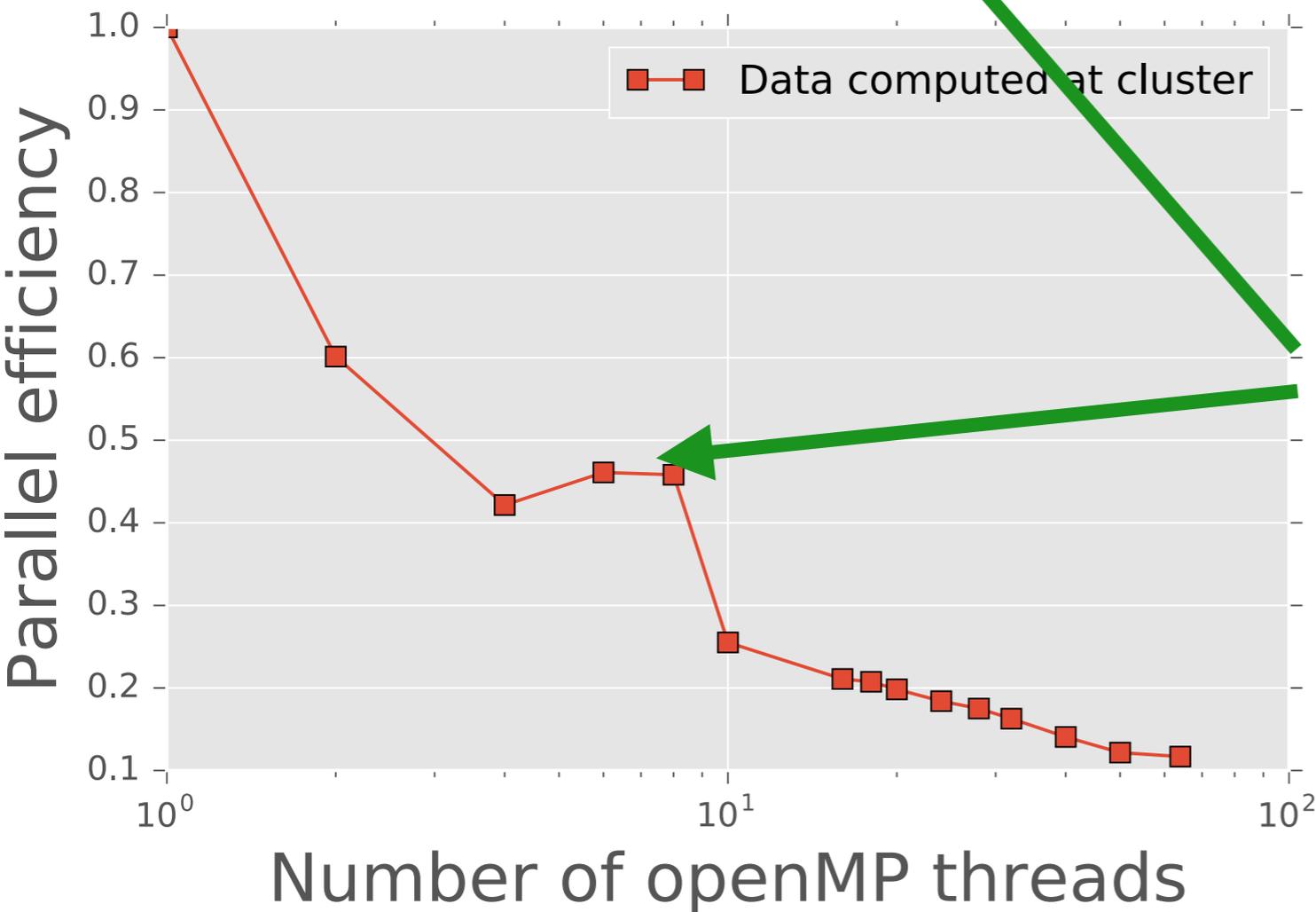
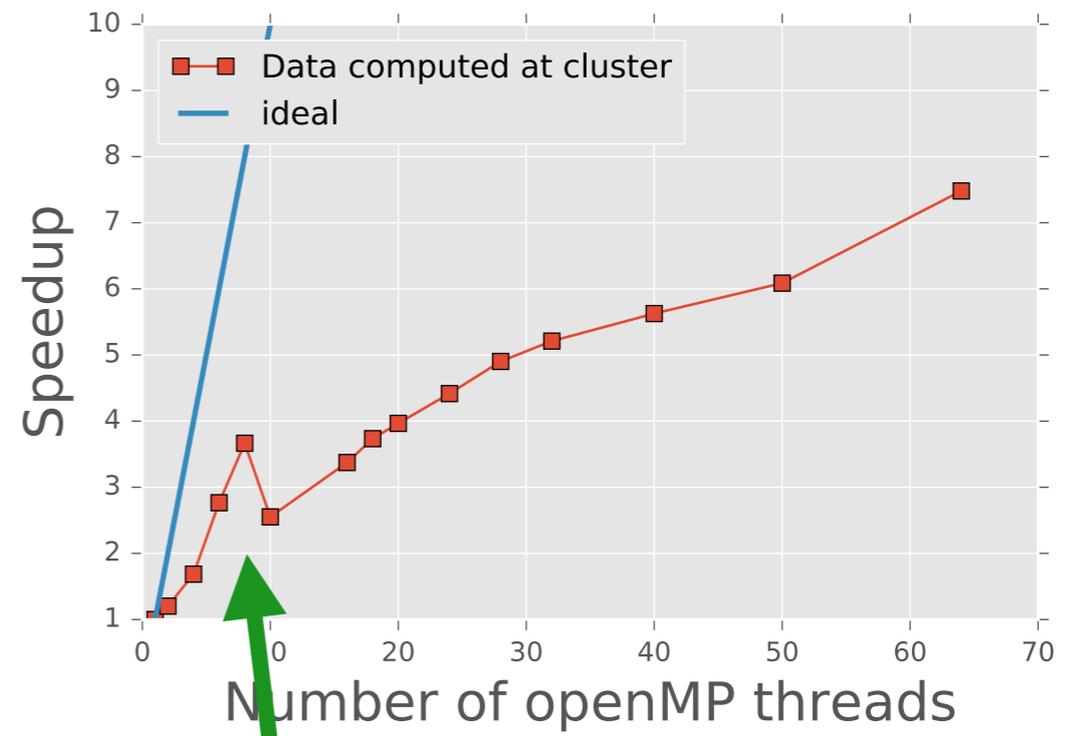
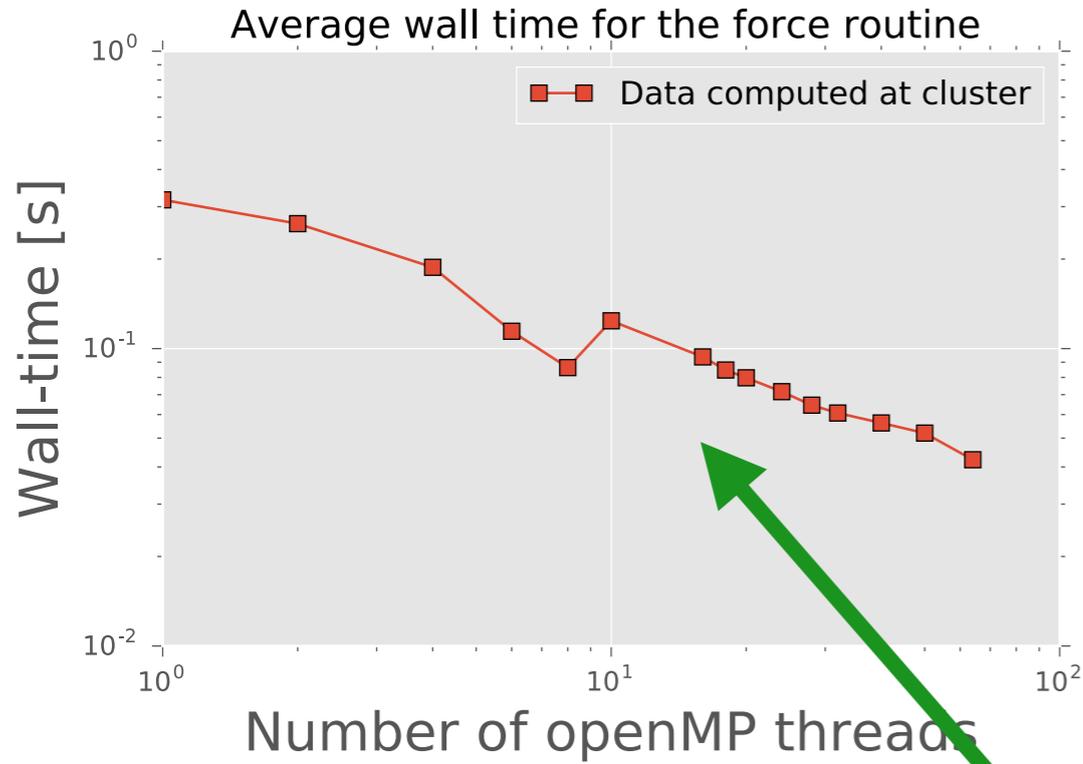
**Be ambitious! Increase the system size (decreases the ratio of communication or thread creation/destruction to computation)!!!**



# Several runs

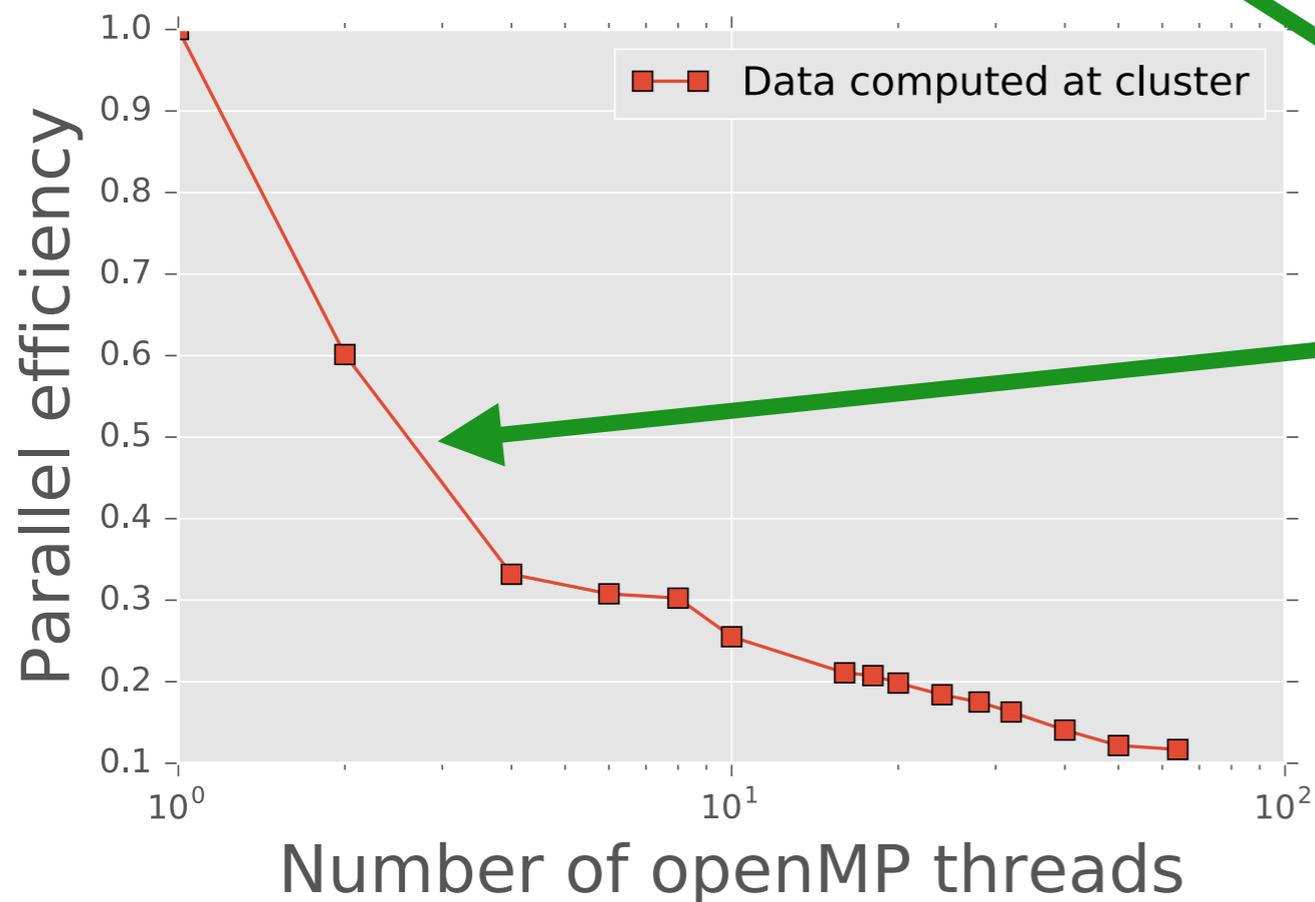
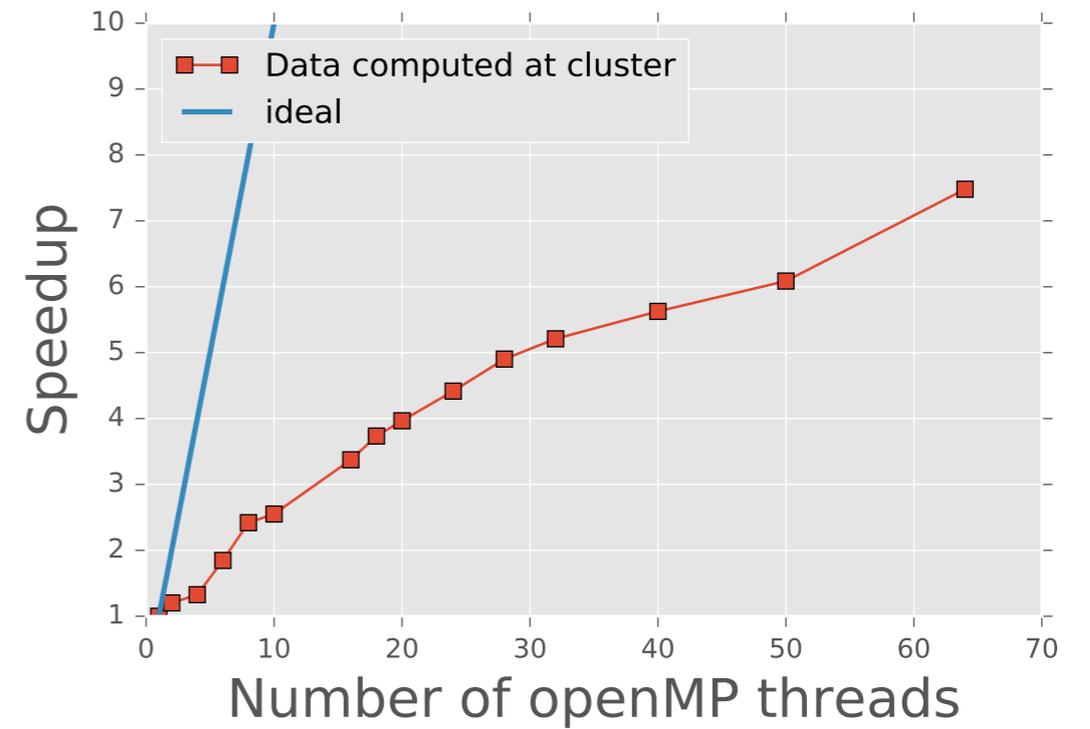
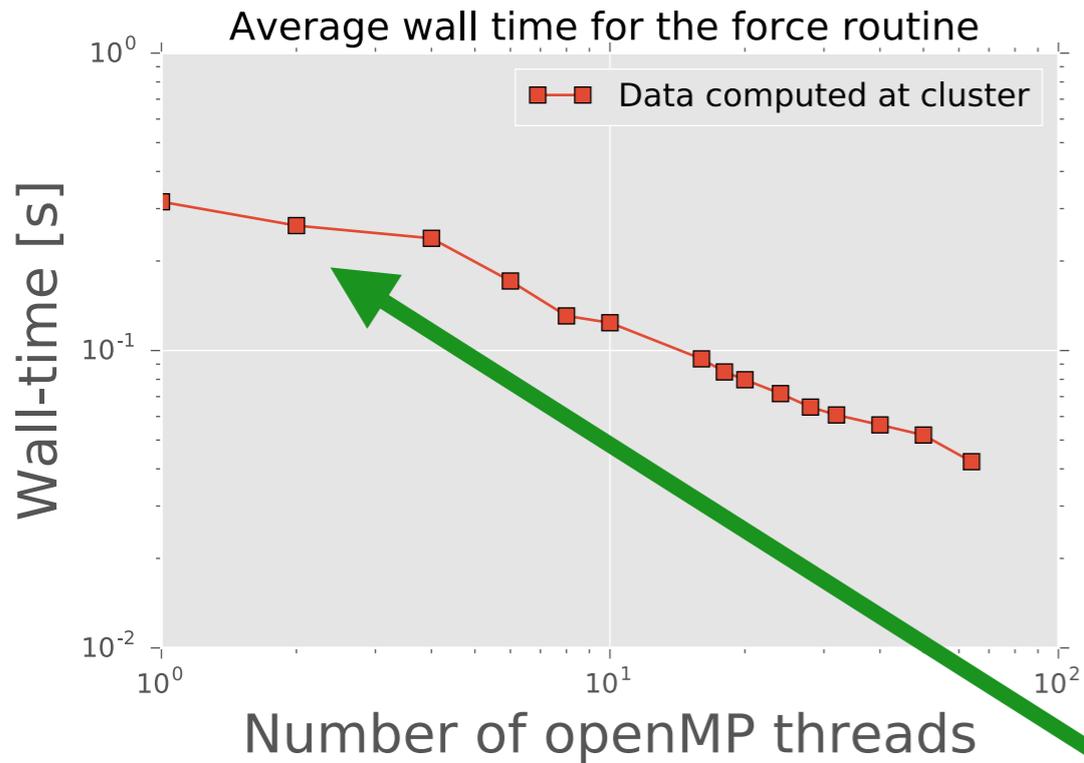


# Several runs



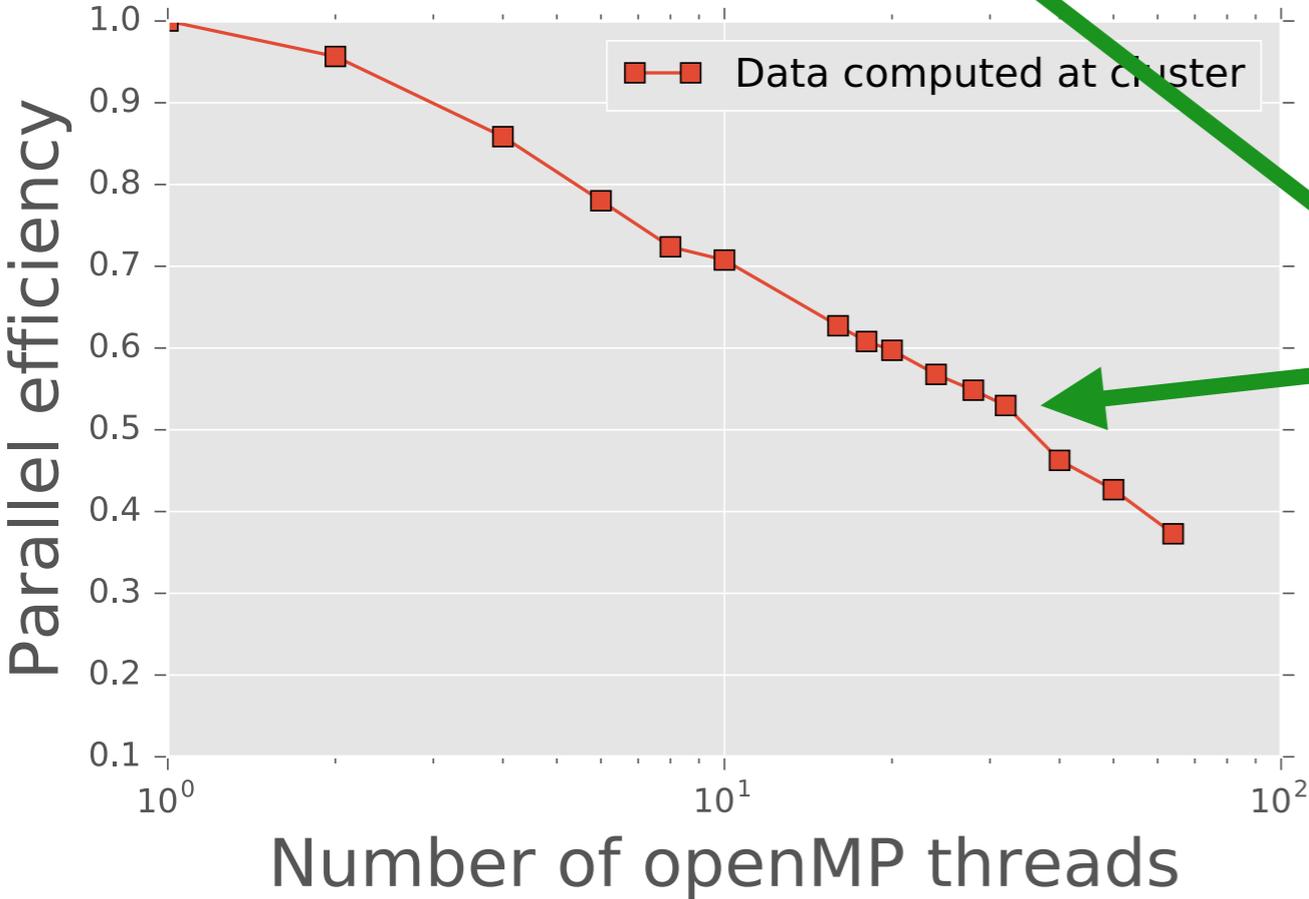
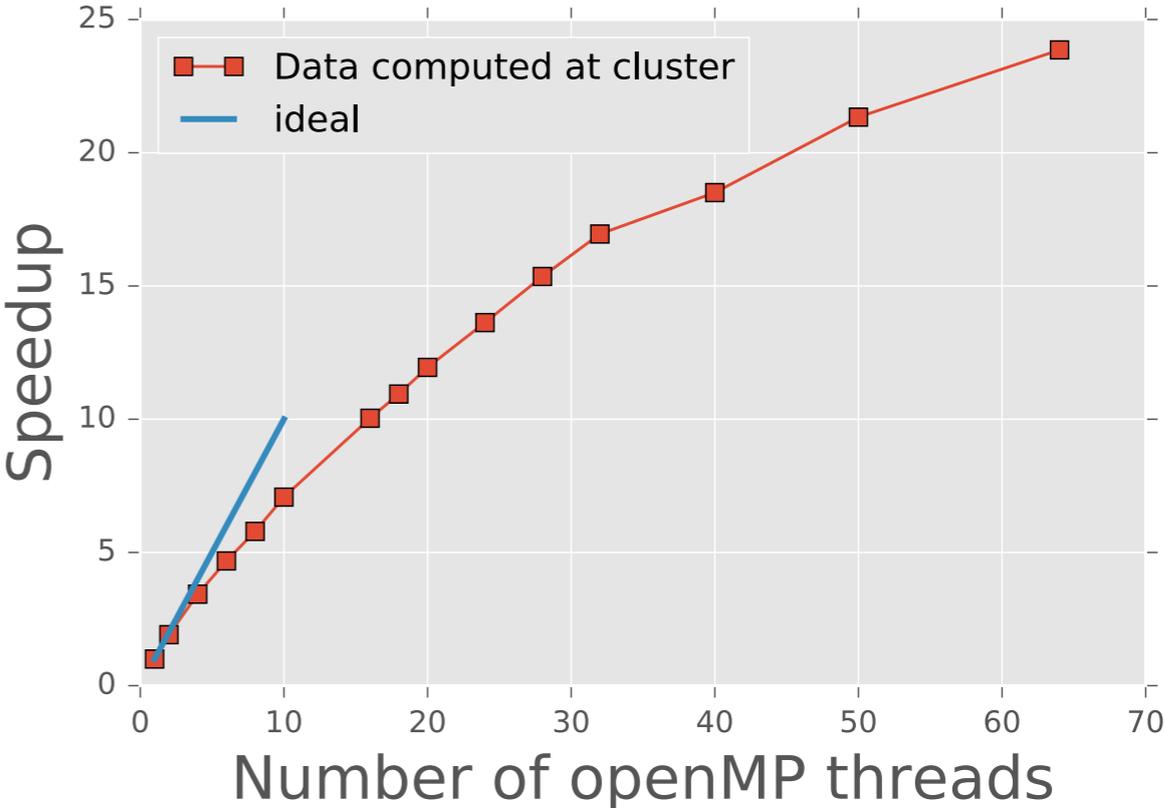
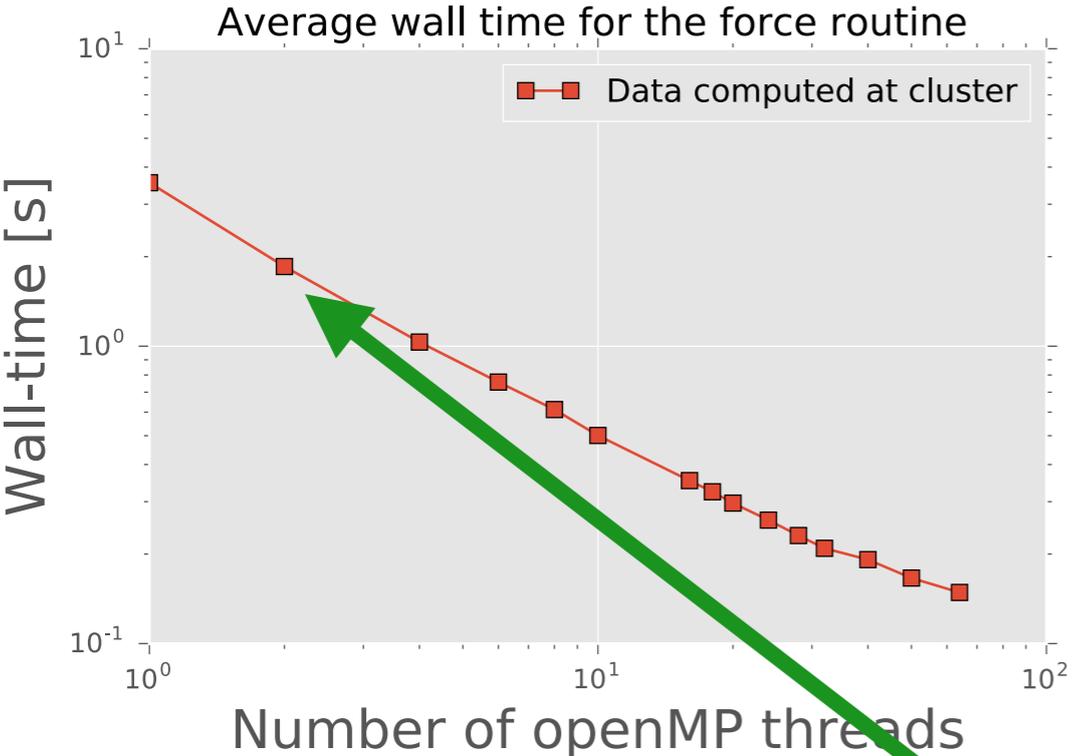
There are some outliers, I need to repeat the simulations

# Several runs



**The optimal number of cores seems to be 2!!! (wall time equal to 0.25s)**

# Competing threads because of 3th Newton law



**The optimal number of cores seems to be 32!!! Maybe because the actual wall time increased by a factor of 10. (2 seconds)**

# Conclusions

- Always think if your system size is really testing the parallel approach.
- Be explicit about what is shared and private inside an omp parallel for .
- Always test and debug with the right tools.
- Compute the parallel efficiency to really get an idea of the ideal number of cores