Scalfmm Periodic Model

A description of the Scalfmm library

Berenger Bramas

Problem description

Implementing the periodic boundary condition

- We want to stay generic
- We have different kernels
- We would like to make in 1/2/3 directions if needed
- We would like to keep the current accuracy
- Of course we want it running fast

Using the FMM

Using FMM kernels to create a periodic system looks good:

- One kernel should propose the functions for the periodicity
- It keeps its constraints and characteristics
- It keeps its accuracy
- Kernels are optimized for FMM

But the ideal system should re-use what each kernel already proposes

Using the FMM

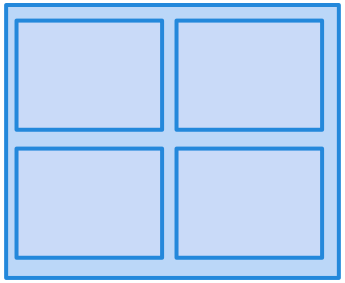
We want to use the current kernels as they are to perform periodic simulations:

→ It means that this system should be hidden in the FMM core and independant from the kernel

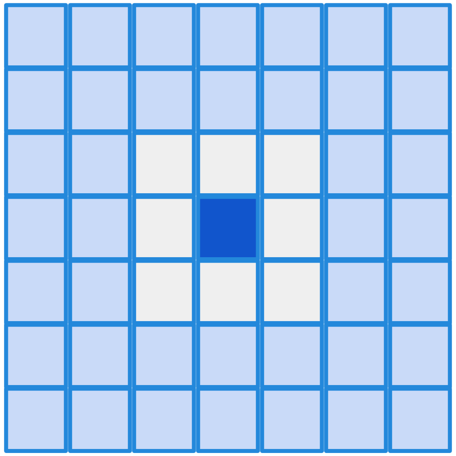
→ It is an algorithmic problem constrained (limited) by the kernels possibilities

What a kernel can do

M2M/L2L (1/8 children)



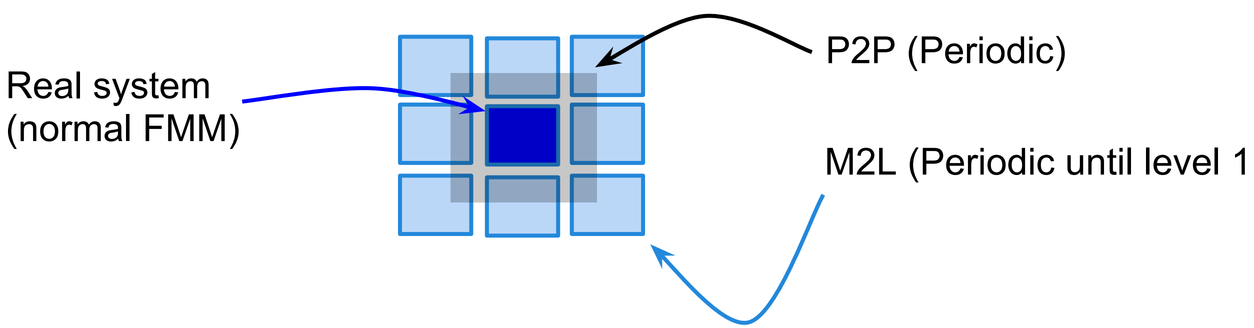
M2L (-3/+3 in x/y/z)



Periodicity - First Step

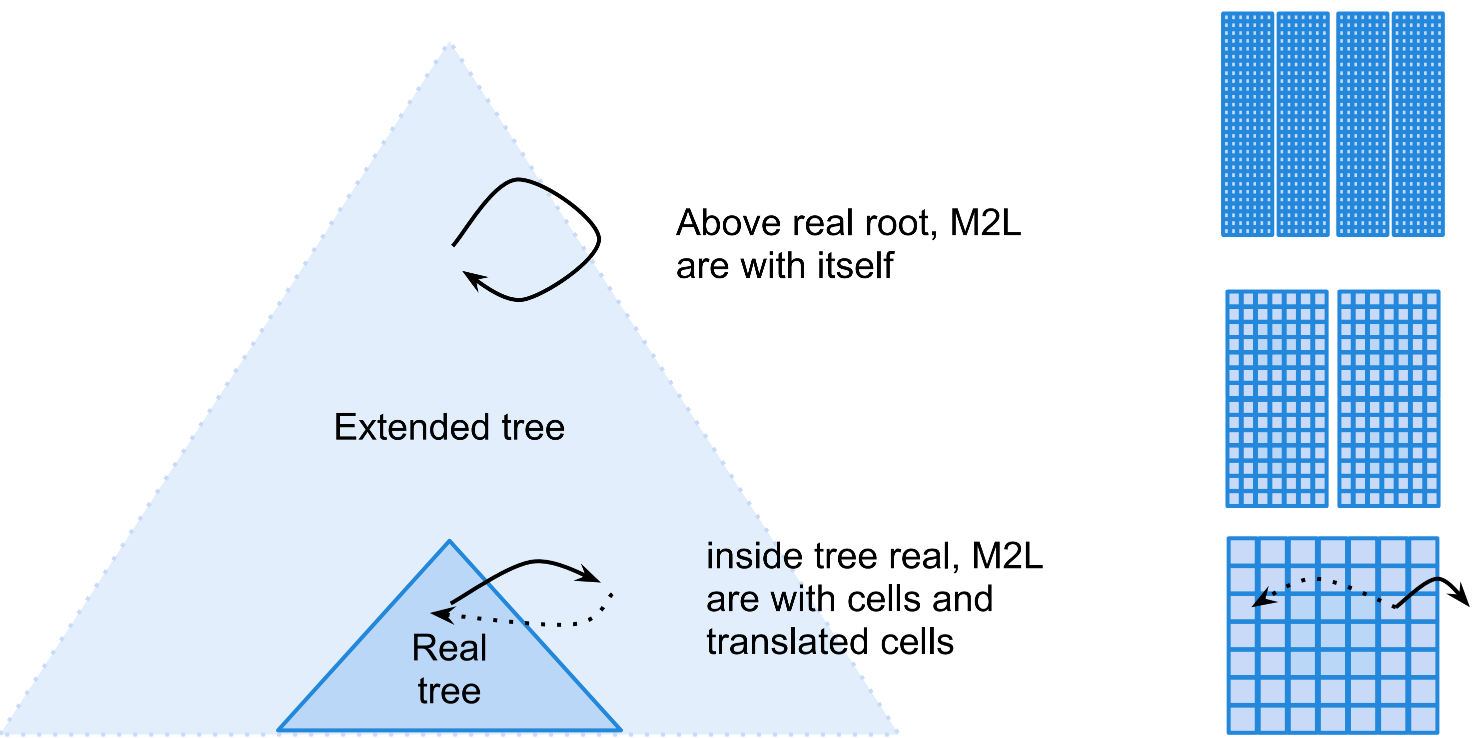
As we are processing the FMM, what are changed if it is periodic?

M2L/P2P : on the border use data from the other side of the box



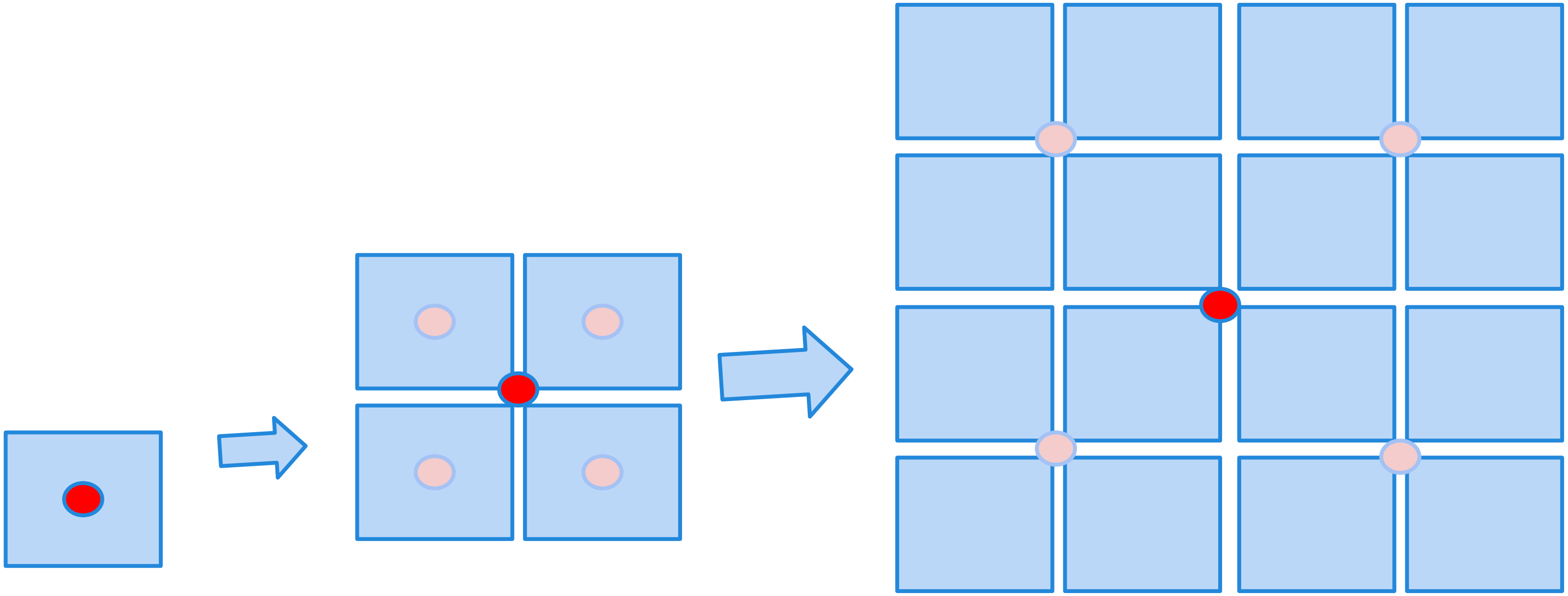
Imaginary tree

From the original tree we can imagine a repetition of the box in each dimension and have in mind a biggest tree



Imaginary tree

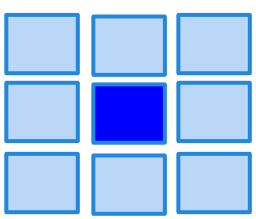
- Moving "up", above the real tree root requires a M2M operation by using the cell of the previous level as the 8 children
- We can represent a cell at any level above the real root

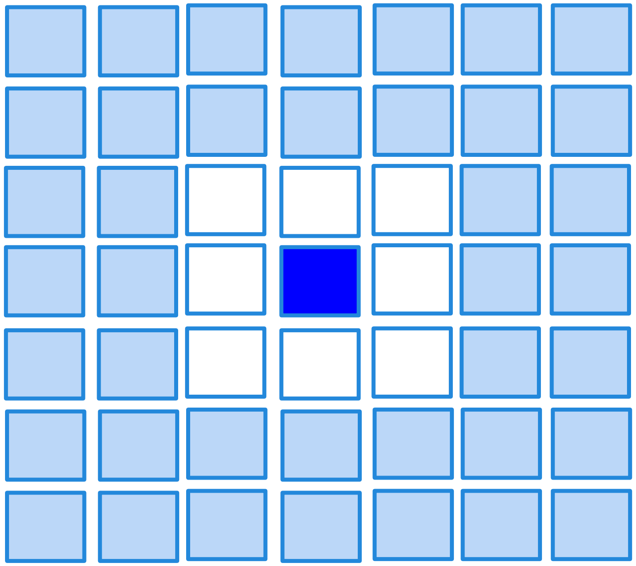


Small periodicity

- Using standard operators: M2L and M2M/L2L already implemented by the kernels, we can create basic periodicity
- Let be D the deeper of the periodicity

D = -1' 0'





And if D > 0?

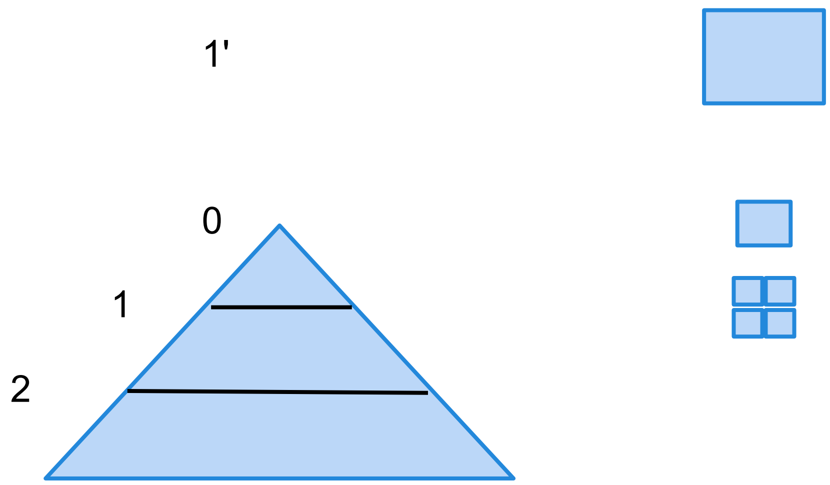
- What if we want to go above?
- We can construct any cell above the real root with the M2M

- But what kind of M2L do we need to do? To work correctly, the FMM needs M2L (from -2 /+3 or -3/+2)
- And then, to do the downward pass, what L2L do we have to compute?

Example

With D = 1'

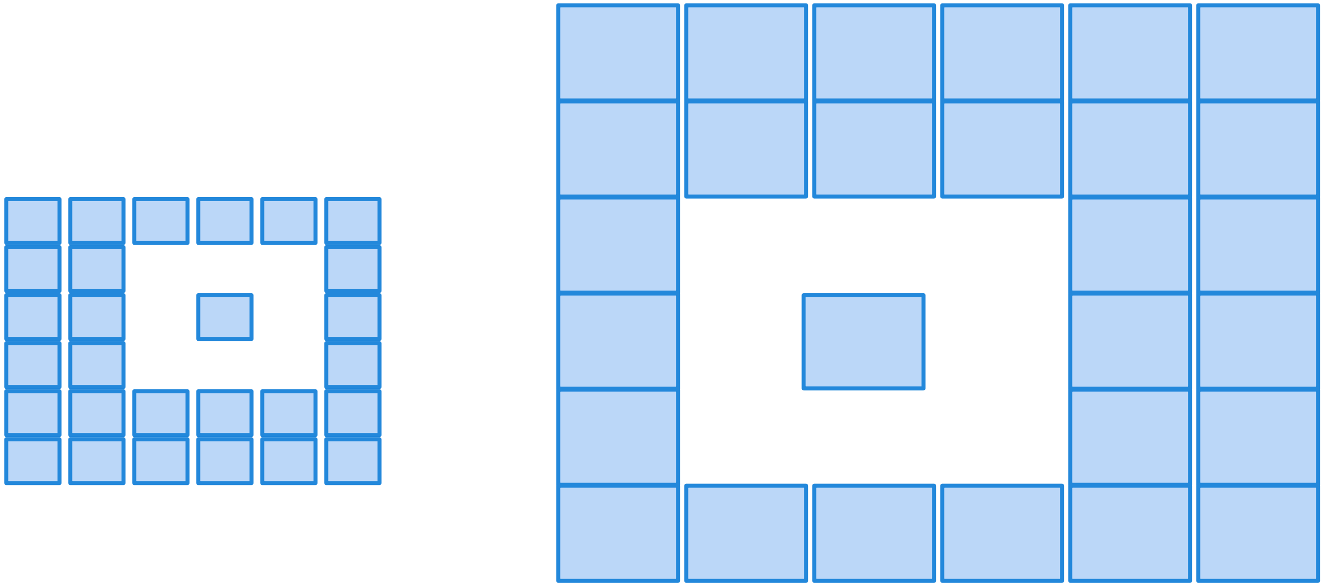
- We compute cells using M2M for level 1 to +1' (usual FMM stop M2M at level 2/3)



Example

D = 1'

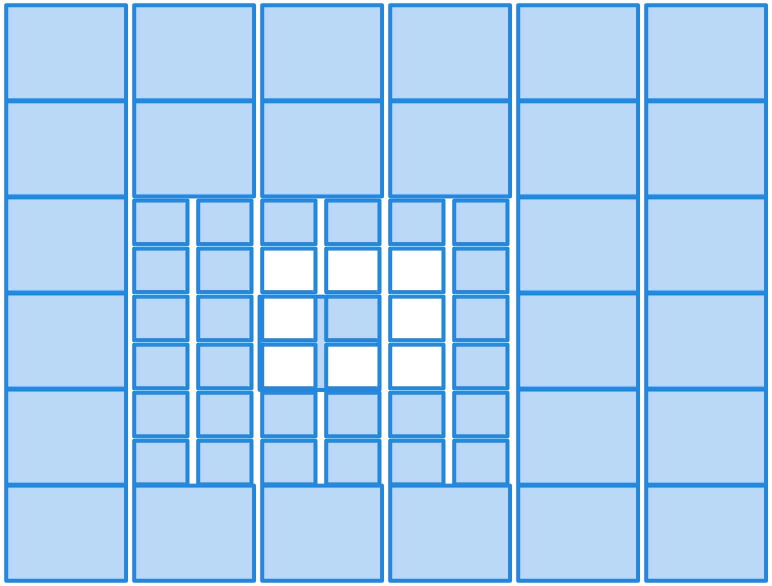
- We do M2L (-3/2 for level 0 to D-1', and -2/3 for level D') using previously computed cells



Example

D = 1'

- Then we do a L2L by considering that our cell is in position +1/+1/+1



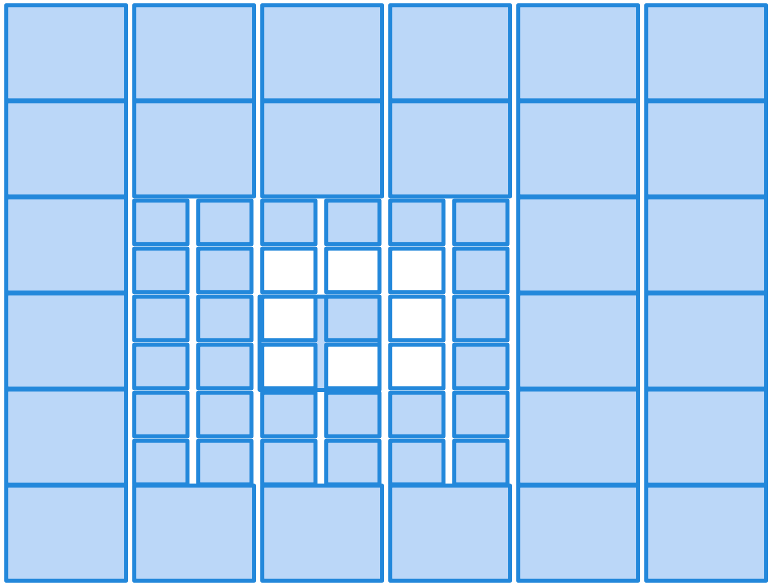
Example

D = 1'

- By doing this we have a grid of :

R = 3*2^(D+1) of original system (D=1';R=12)

- It is not symmetric! We need to add a "border"

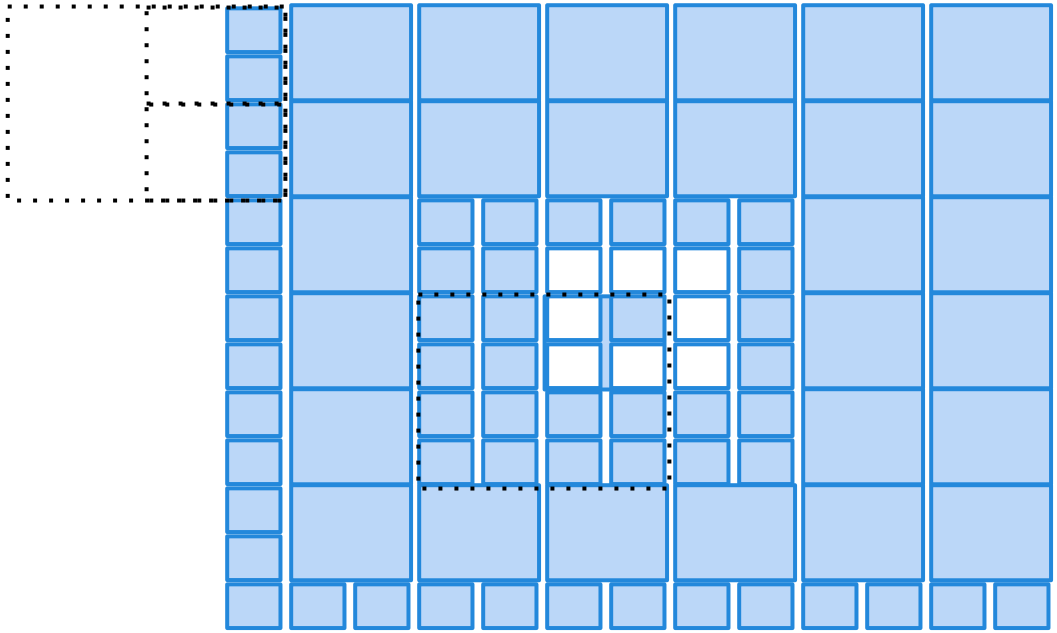


Example

D = 1'

- We add boxes and compute M2L at D+1'

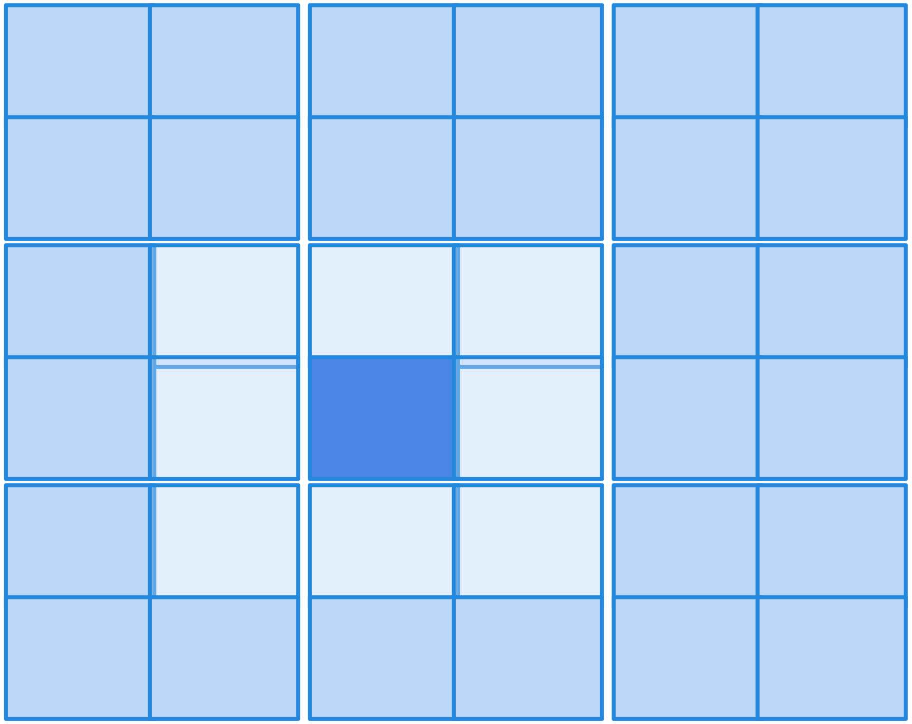
Then R = 2^(D+1) + 1 of the original system



Another example

Another point of view is to start from the top

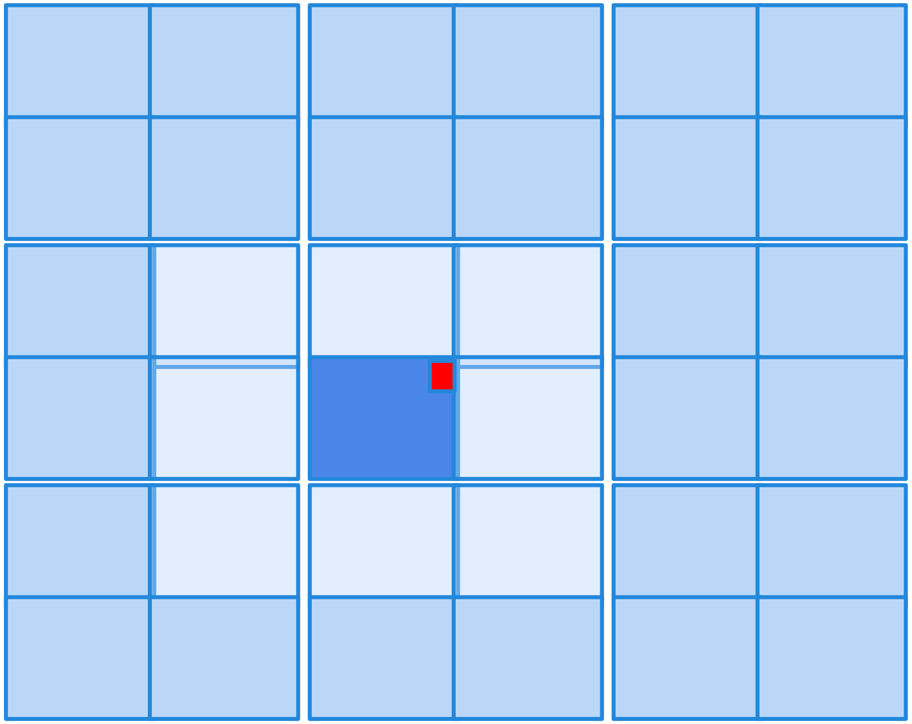
- We do a M2L at a very high level(-2/+3)



Another example

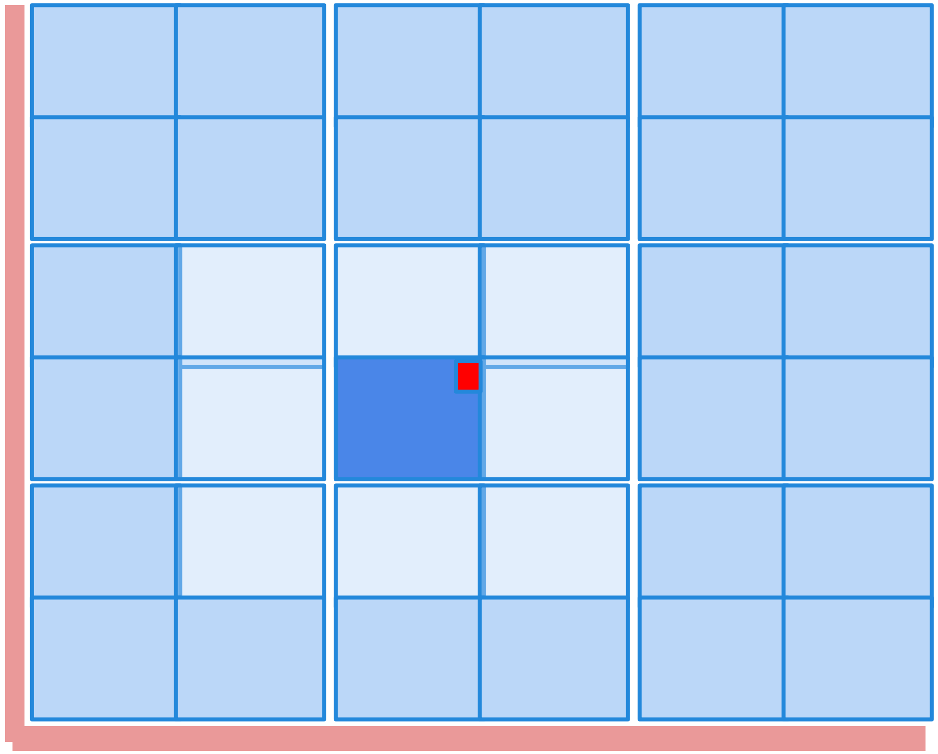
Our objective is to be in the middle

- The original box is centered



Another example

That is why we consider to be at position (1,1,1) for the L2L and that we need a border for the symmetry



Algorithm - Summary

1. Compute M2M for l = 0 : D
2. Compute M2L (-3:2/-3:2/-3/2) from 0 to D-1'
3. Compute M2L at D (-2:+3/-2:+3/-2/+3)
4. Compute the border (and M2L at D+1')
5. Go down with L2L using position 7 from D+1' to 0 (or 1 because usual FMM do not do L2L between 0 and 2)

Conclusion

- We add a periodic system independent from the kernel (as long as the kernel work for the FMM)
- It uses standard FMM operators (no special M2L needed)
- It can be extended to go only in some directions
- It repeats the original grid per:$3 * 2^{(D+1)} + 1$
- It already works with all our operators since no changes are needed, we cheat by giving wrong information to the kernel about the

Cost

- For each level above 2 we do:

(189 M2L + 8 M2M + 1 L2L) * D

- And the "border":

7 M2L

(X M2M + X M2L) * 7 * D

With X [1:4]