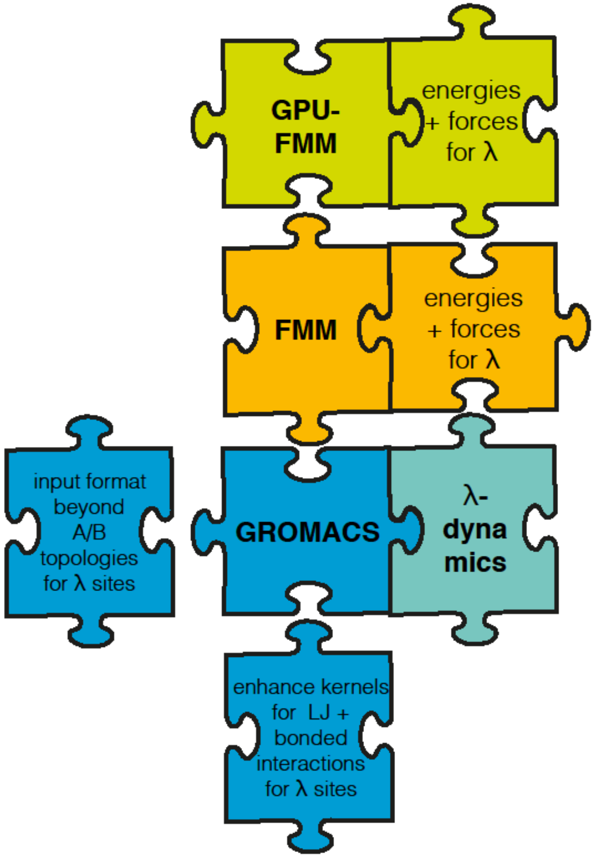
FMM –
GPU implementation
and λ-dynamics

Bartosz Kohnke

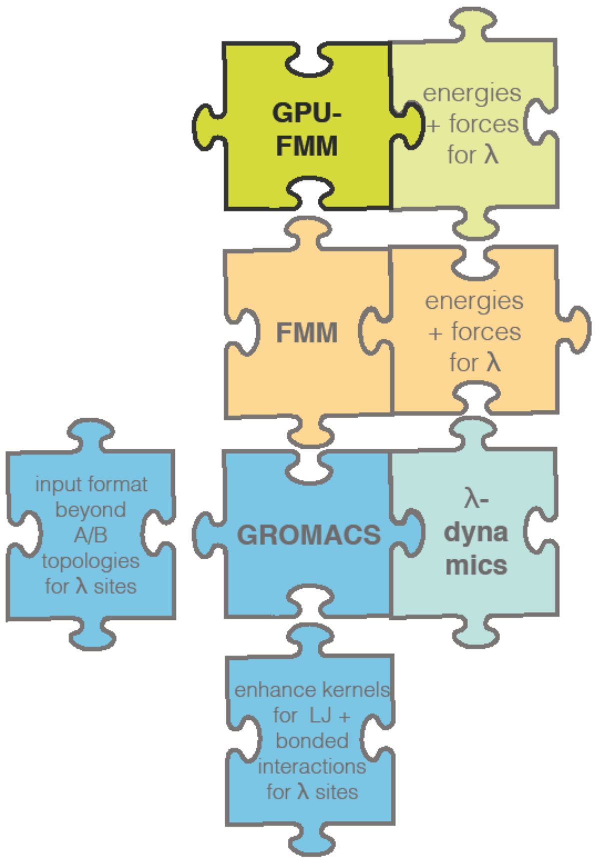
FMM

GPU implementation and λ-dynamics FMM



FMM

GPU implementation and λ-dynamics FMM

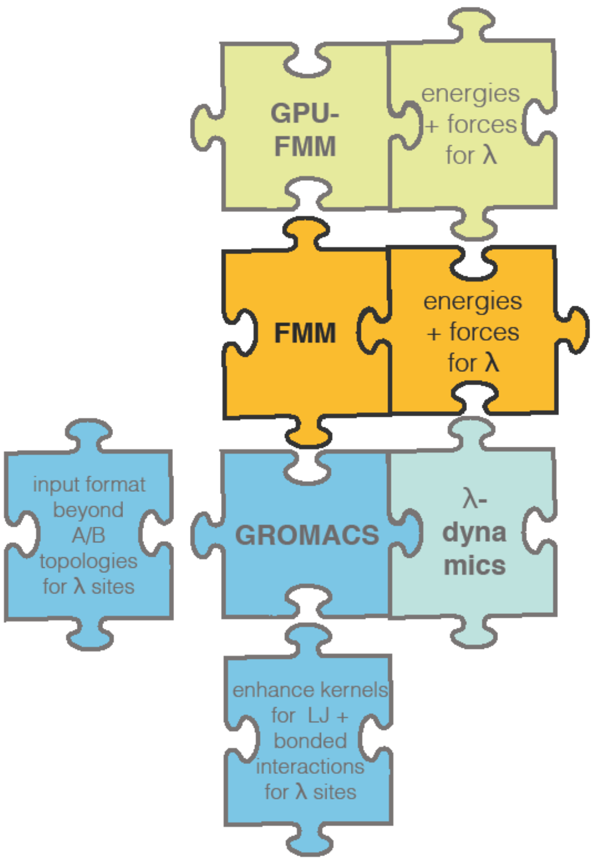


Outline

- FMM Parallelization
  - Data structures
  - Parallelization approaches
  - Results

FMM

GPU implementation and λ-dynamics FMM



Outline

- FMM Parallelization
  - Data structures
  - Parallelization approaches
  - Results
- λ-dynamics FMM
  - λ-dynamics – electrostatics
  - How λ-dynamics affects the FMM performance
  - Scaling of λ-dynamics FMM

FMM – Data Structures

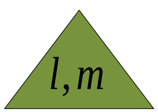
Farfield

Coefficient Matrix, Generalized Storage Type

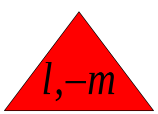
- Multipole order p
- Matrix size $\mathcal{O}(p^2)$
- Triangular shape

Multipole/Taylor Moments and Operators

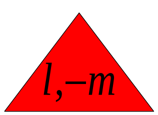
- ω, μ, M
- Stored as coefficient matrix

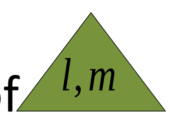


- one complex value

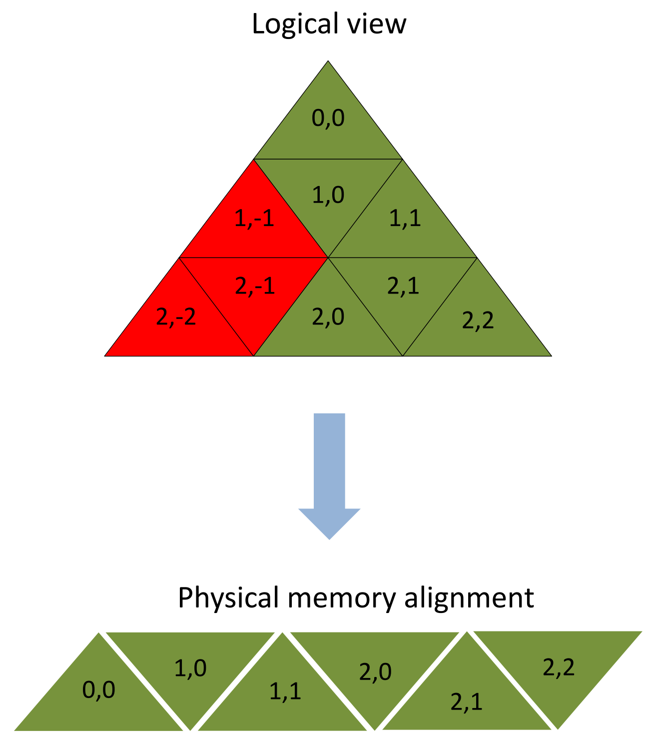


- does not need to be stored



is a function of


Coefficient Matrix



Multipole to local translation (M2L)

Tree loop and Box – Neighbor Structure, ws=1

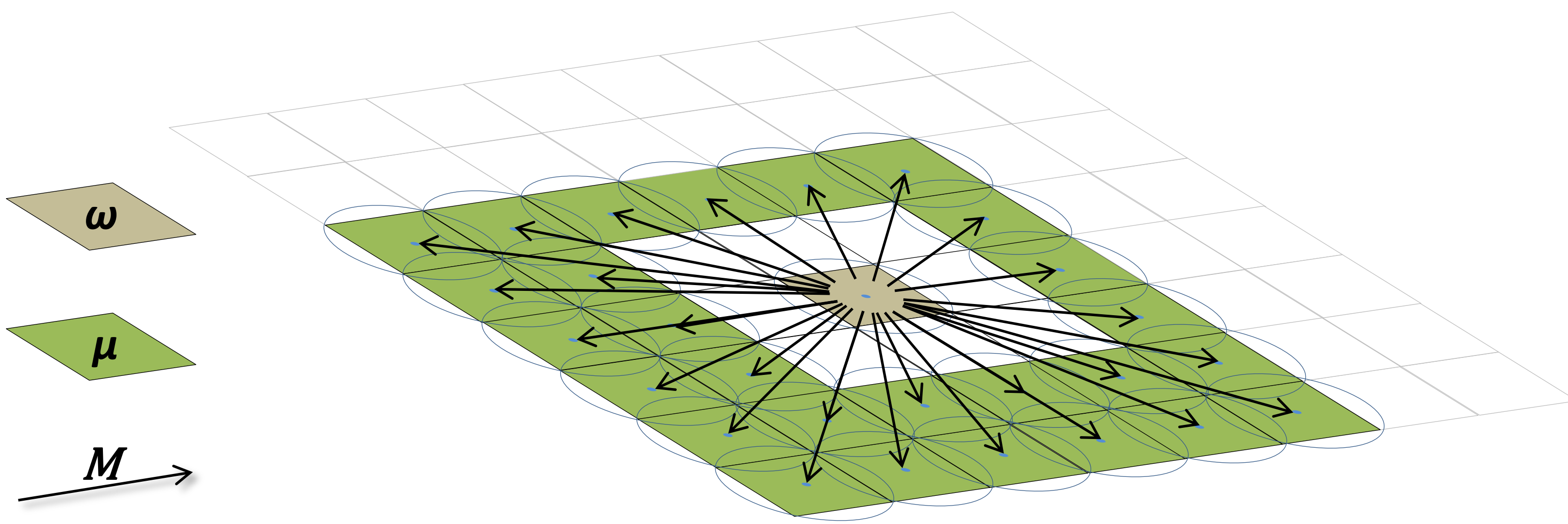
M2L Operation

- On each depth of the tree
  - Compute position of $\boldsymbol{\omega}$
  - Determine valid neighbors $\boldsymbol{\mu}$
  - Compute index of the operator $\boldsymbol{M}$
  - Compute one p^4 M2L-Operation for each valid $\boldsymbol{\mu}$ and $\boldsymbol{\omega}$ pair

M2L – operations extent

- d = 4, ws = 1, p = 10
- 4.6 x 10^9 global memory reads
- 3.8 x 10^7 global memory writes

$$\mu(\mathbf{b}-\mathbf{a}) = \sum_{l=0}^{p}\sum_{m=0}^{l}\sum_{j=0}^{p}\sum_{k=-j}^{j} M_{l+j,m+k}(\mathbf{b})\,\omega_{jk}(\mathbf{a})$$



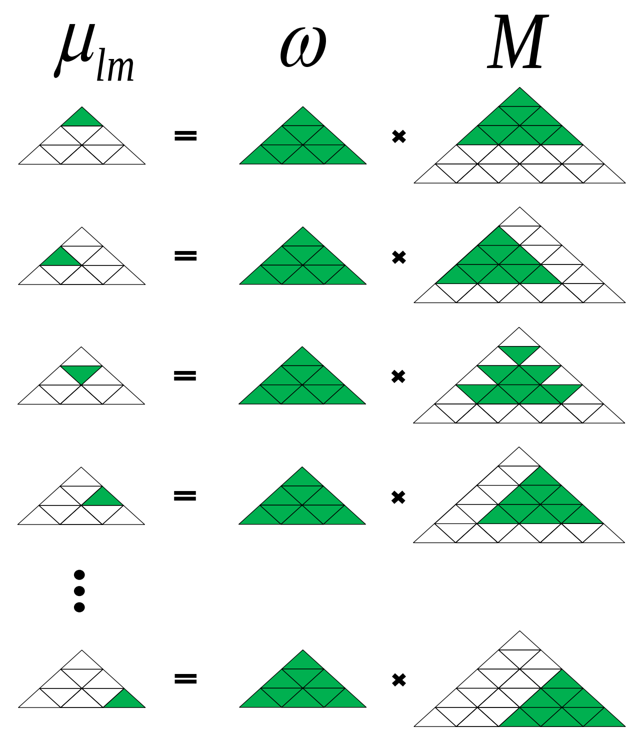
M2L Operation – Internal Structure

Translating multipole expansion to local expansion, p^4 loop structure

One M2L operation

$$\mu(\mathbf{b}-\mathbf{a}) = \sum_{l=0}^{p}\sum_{m=0}^{l}\sum_{j=0}^{p}\sum_{k=-j}^{j} M_{l+j,m+k}(\mathbf{b})\omega_{jk}(\mathbf{a})$$

```
for (int l = 0; l <= p; ++l)
  for (int m = 0; m <= l; ++m)
  {
    omega_l_m=0;
    for (int j = 0; j <= p; ++j)
    {
      for (int k = -j; k <= j; ++k)
      {
        omega_l_m += M[m_idx](l + j, m + k)
                     *
                     omega[o_idx](j, k);
      }
    }
    mu[mu_idx](l, m) += omega_l_m
  }
```

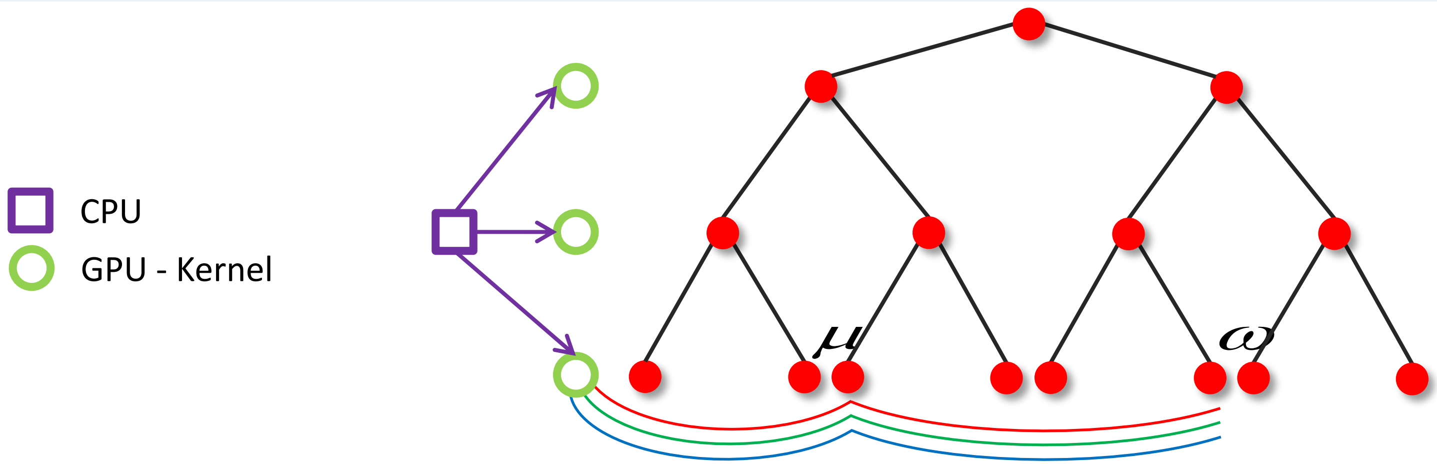


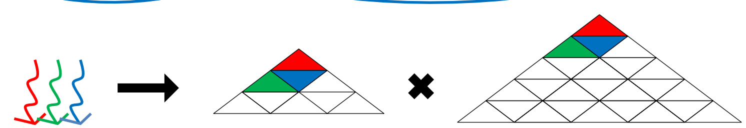
M2L Operation – Parallelization

First approach

Full parallel kernel

- Parallelize all loops





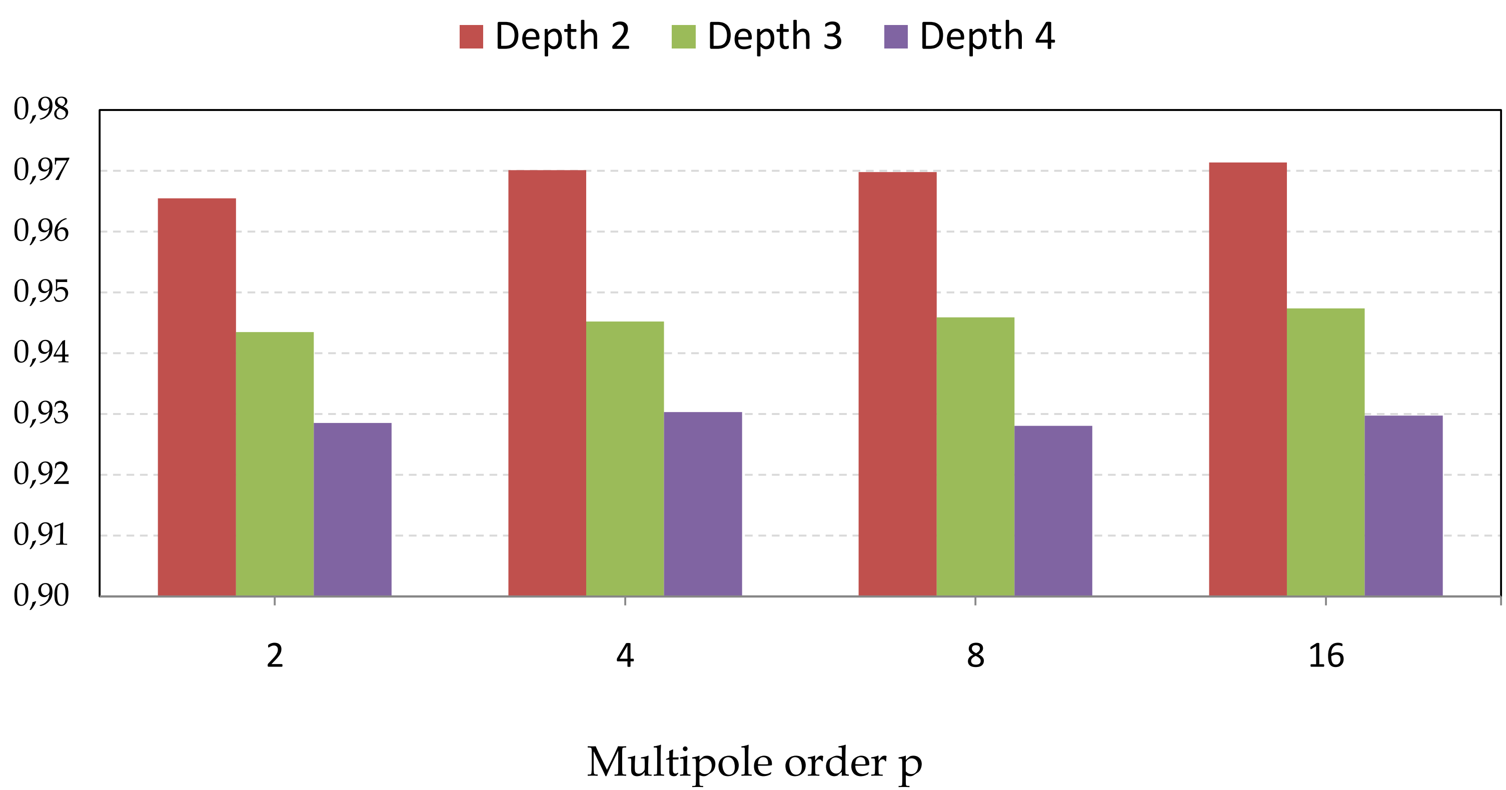
Drawbacks

- p^4 threads compute redundant information
  - computing index of valid neighbor ω for μ and operator M index
  - checking boundaries
- very divergent
- execution time dominated by many integer divisions/modulo

Full Parallelization Costs

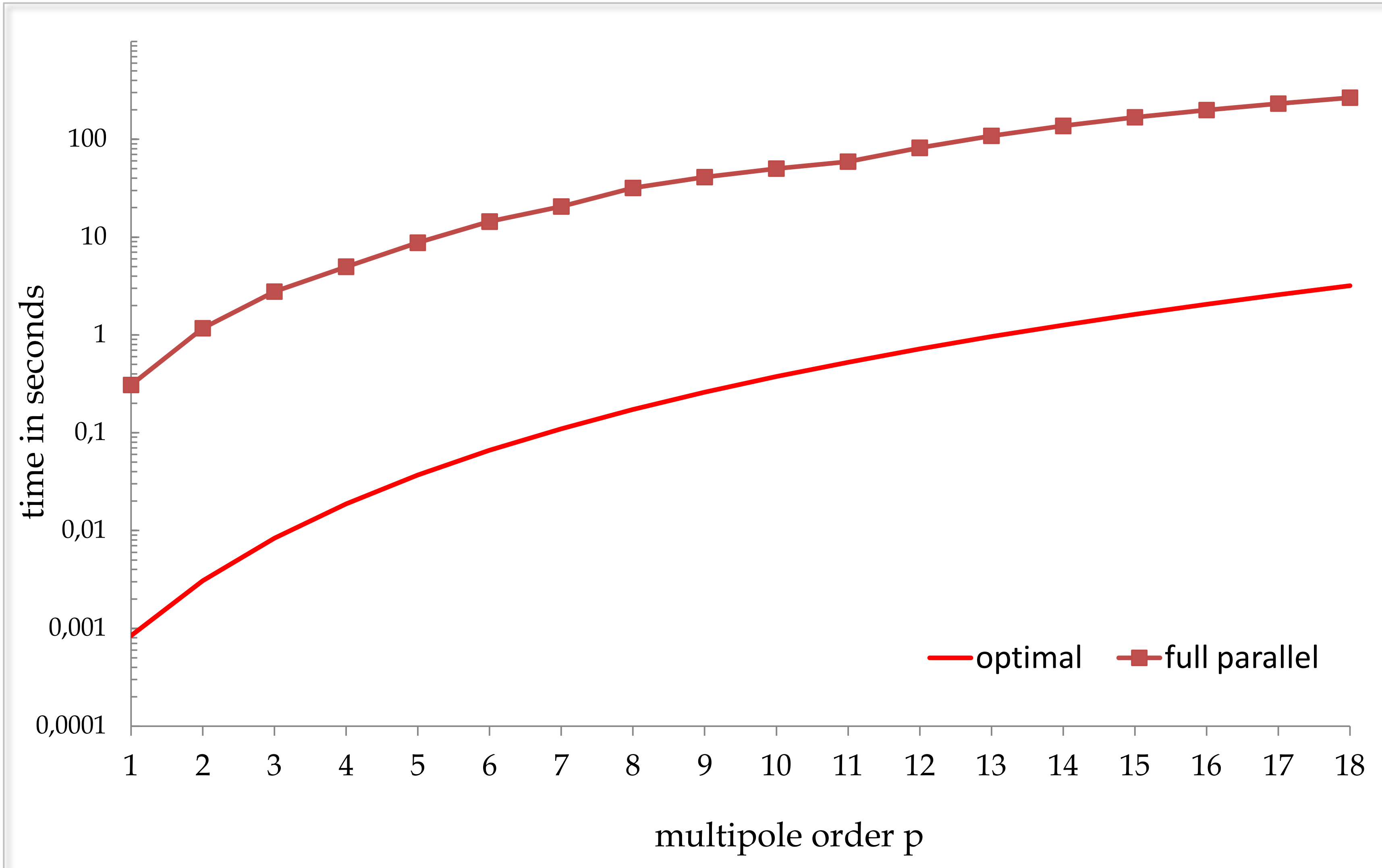
M2L full parallel kernel

Relative costs of index computation



M2L Runtime – Full Loop Parallelization

Depth 4, 4096 Boxes

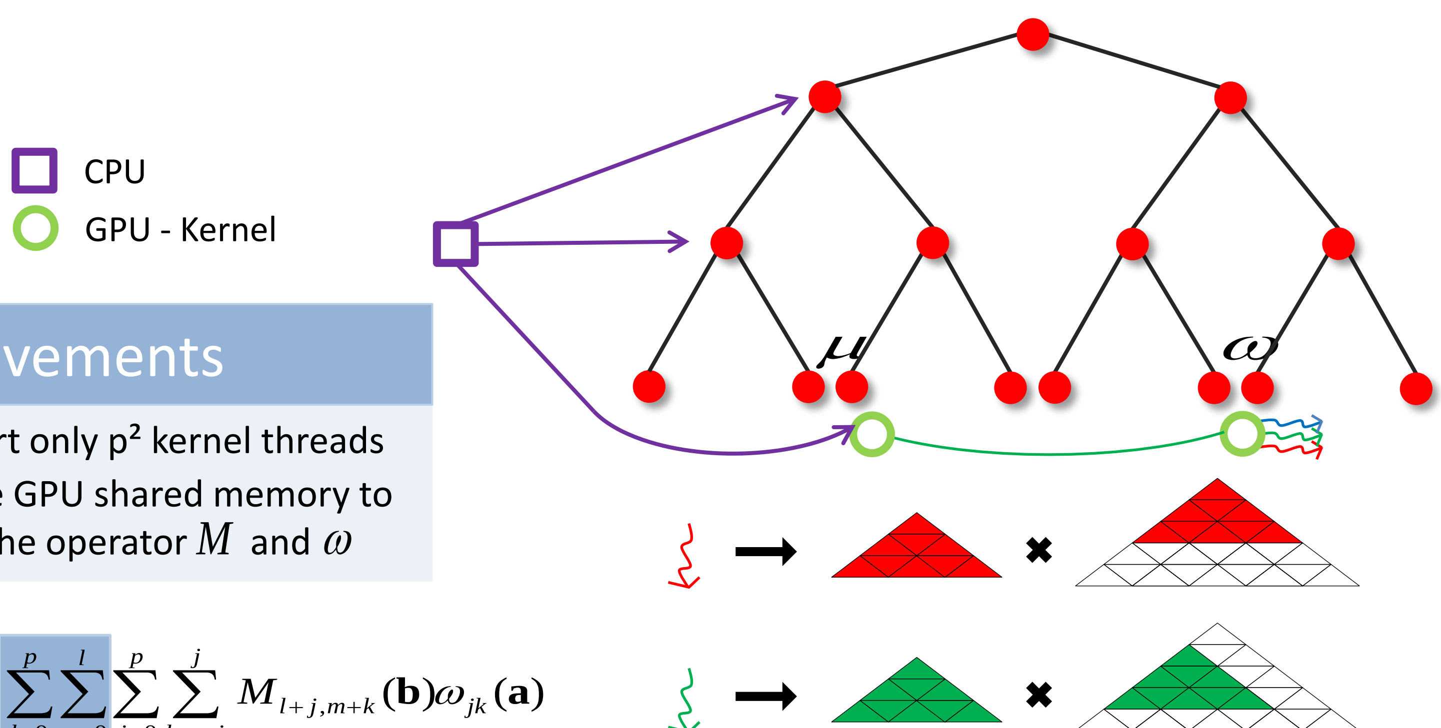


M2L Dynamic Scheme + Shared Memory

Dynamic Scheme Overview

Launching Kernels

- loop over all tree levels @ host CPU
- loop over boxes on a certain level (64, 512, 4096…) @ host CPU
- loop over all neighbor boxes (216) – kernels launched from host
- perform a single M2L operation – kernels launched from GPU (dynamically)



Improvements

- Start only p² kernel threads
- Use GPU shared memory to cache operator M and ω

$$\mu(\mathbf{b}-\mathbf{a}) = \boxed{\sum_{l=0}^{p}\sum_{m=0}^{l}}\sum_{j=0}^{p}\sum_{k=-j}^{j} M_{l+j,m+k}(\mathbf{b})\,\omega_{jk}(\mathbf{a})$$

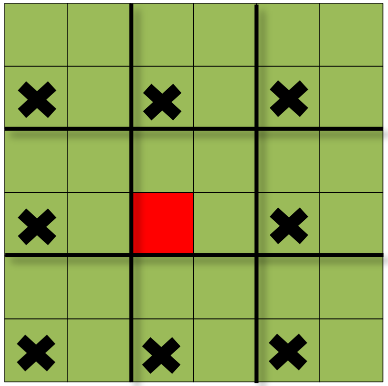
M2L Dynamic Scheme + Shared Memory

Dynamic Scheme Details

Launching Kernels

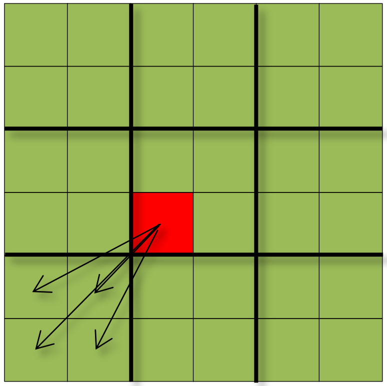
Parent kernel

- Computes interaction set
- Checks boundaries
- Computes indices of target box and $\boldsymbol{M}$ operator
- Launches a child kernel for each group of eight (3D) M2L operations
- Very small non blocking kernels <<<(1,1,1),(3,3,3)>>>



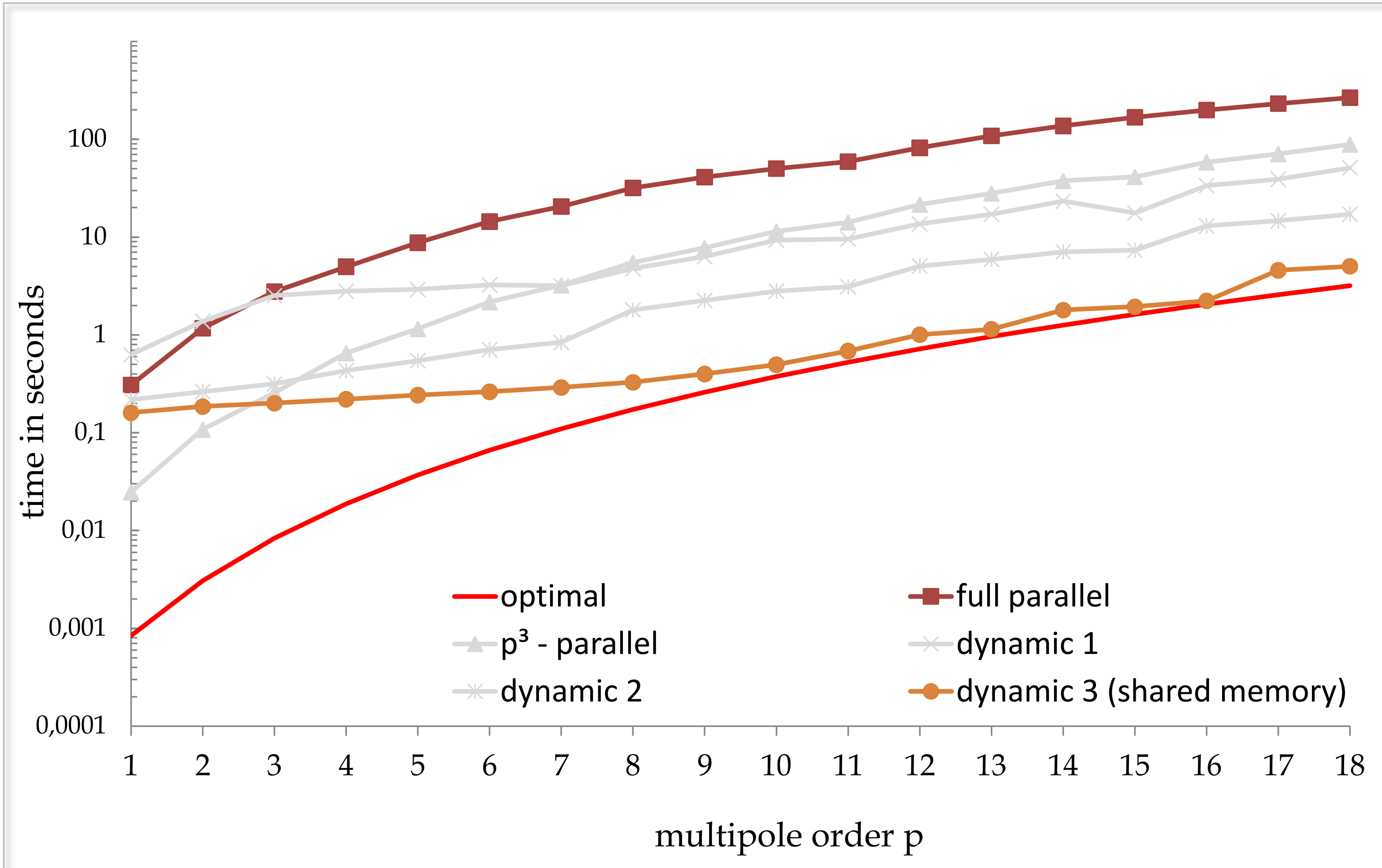
Child kernel

- Performs M2L operations between one source and eight target boxes sharing the same parent
  - Easy computaion of indices
- Caches the $\boldsymbol{\omega}$ and $\boldsymbol{M}$ values into shared memory
- Non blocking kernels <<<(2,2,2),(p, p, 1)>>>



M2L Dynamic Scheme + Shared Memory

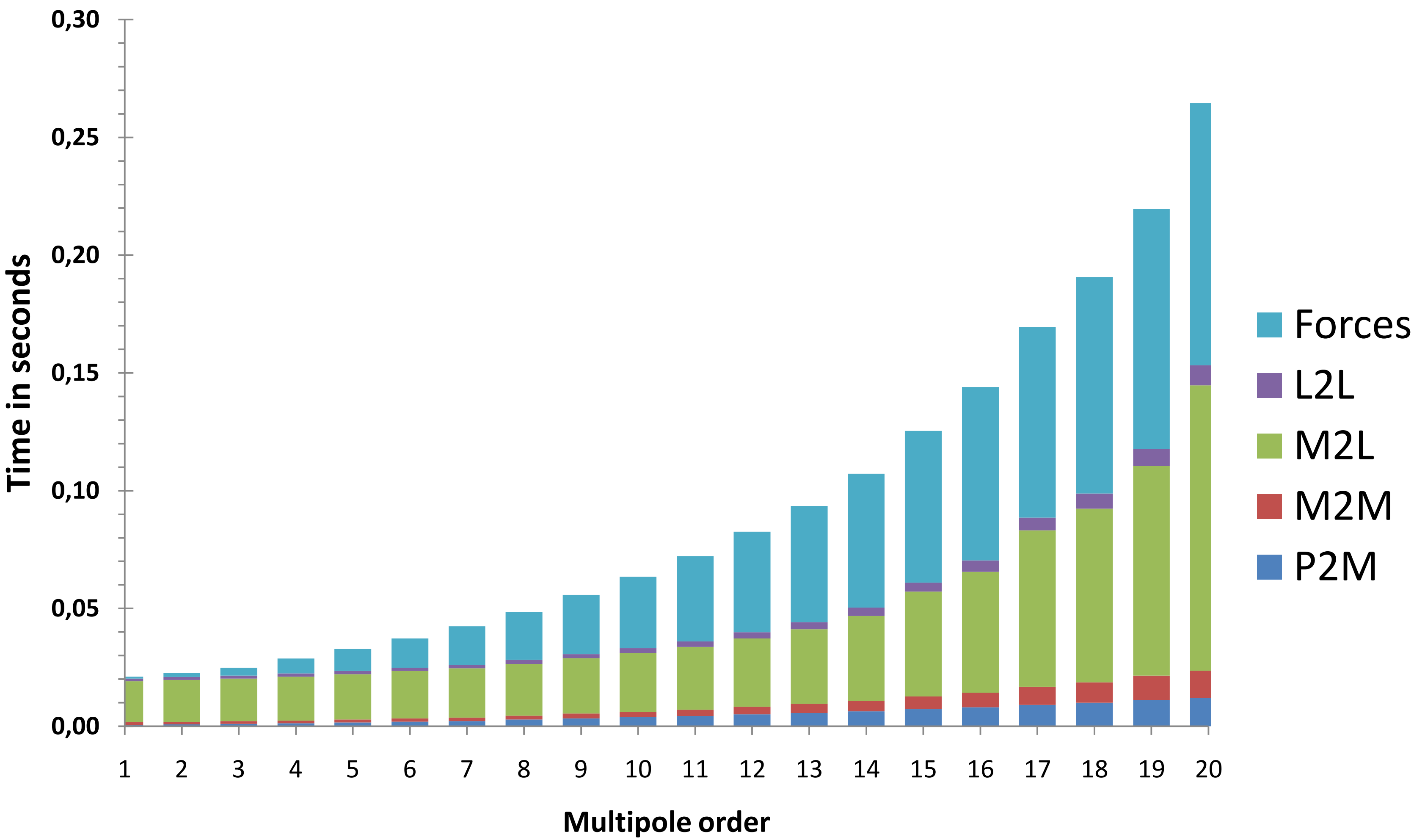
Depth 4, 4096 Boxes



FMM runtime

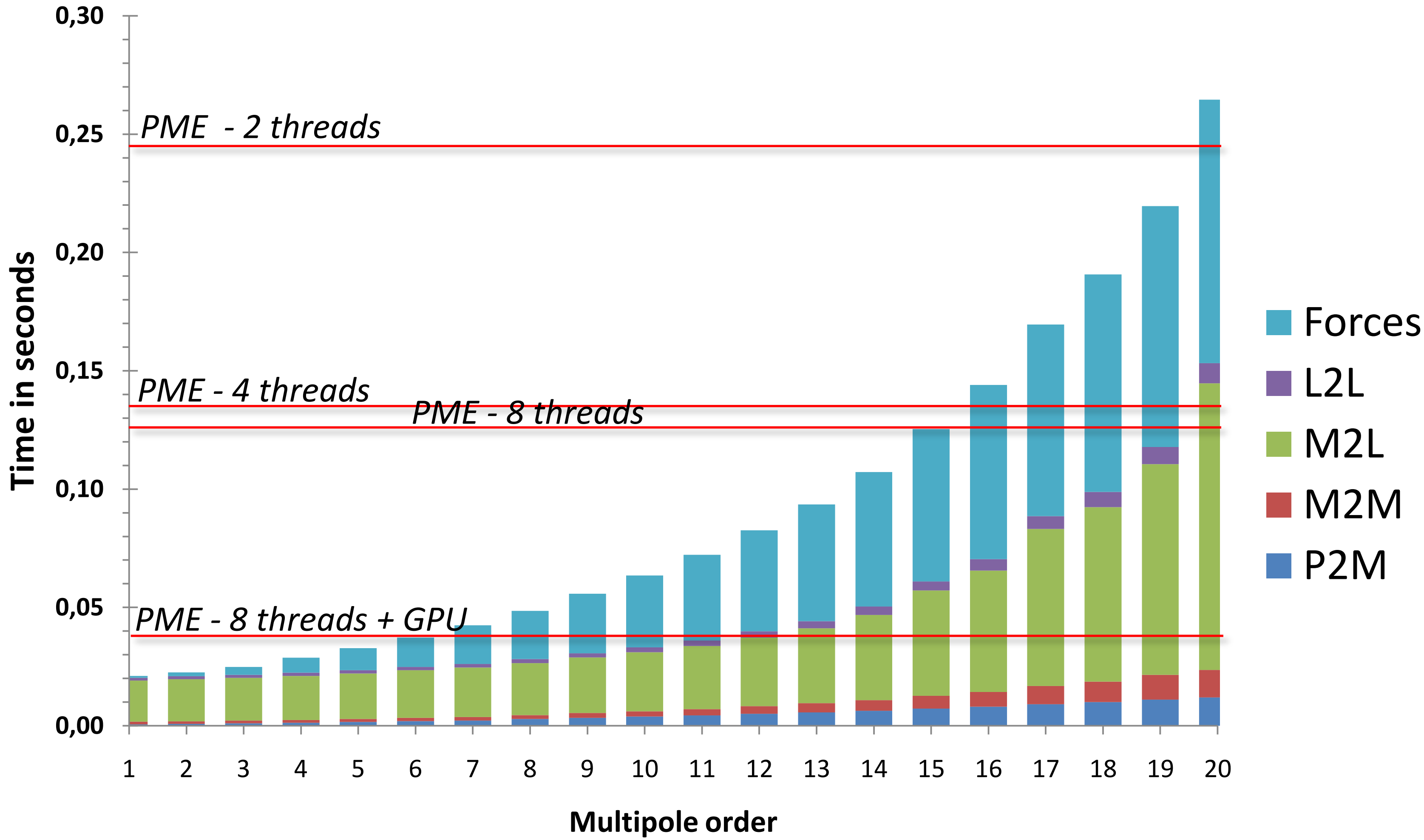
285119 Particles – channel protein (TehA), Intel E5-1620 CPU, 4 Cores (8 Hyperthreads)

FMM – Depth 3, 512 Boxes@Dept(3), GeForce GTX Titan



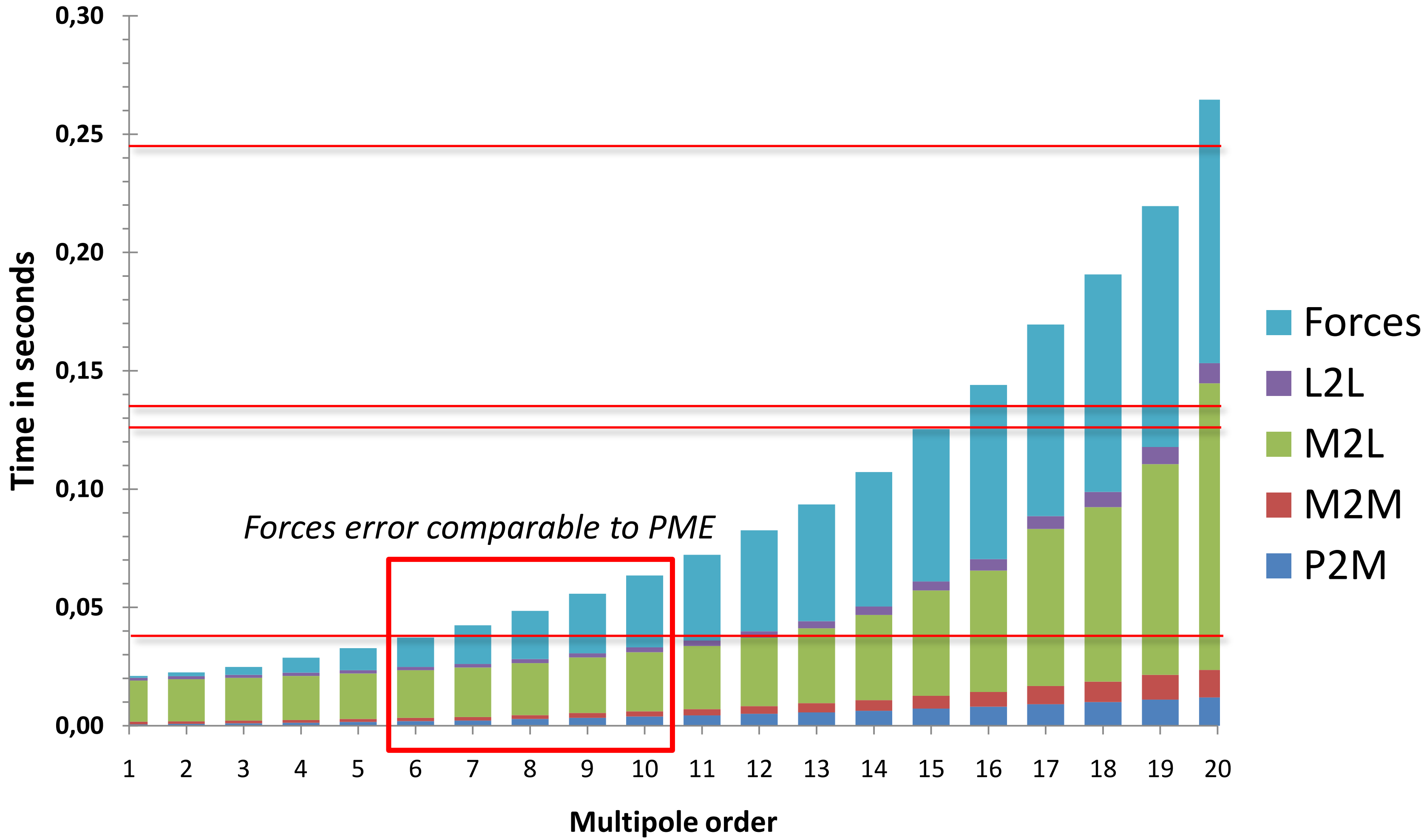
FMM runtime

285119 Particles – channel protein (TehA), Intel E5-1620 CPU, 4 Cores (8 Hyperthreads)
FMM – Depth 3, 512 Boxes@Dept(3), GeForce GTX Titan
GROMACS – PME grid 0.12 nm, cutoff 1.0 nm, GeForce GTX 680 GPU



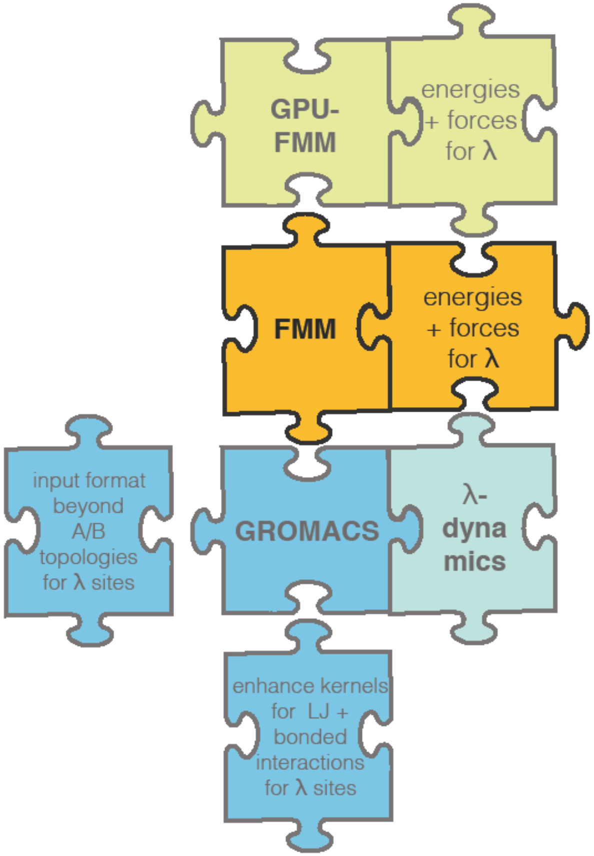
FMM runtime

285119 Particles – channel protein (TehA), Intel E5-1620 CPU, 4 Cores (8 Hyperthreads)
FMM – Depth 3, 512 Boxes@Dept(3), GeForce GTX Titan
GROMACS – PME grid 0.12 nm, cutoff 1.0 nm, GeForce GTX 680 GPU



FMM

GPU implementation and λ-dynamics FMM



Outline

- FMM Parallelization
  - Data structures
  - Parallelization approaches
  - Results
- λ-dynamics FMM
  - λ-dynamics – electrostatics
  - How λ-dynamics affects the FMM performance
  - Scaling of λ-dynamics FMM

λ-dynamics and FMM

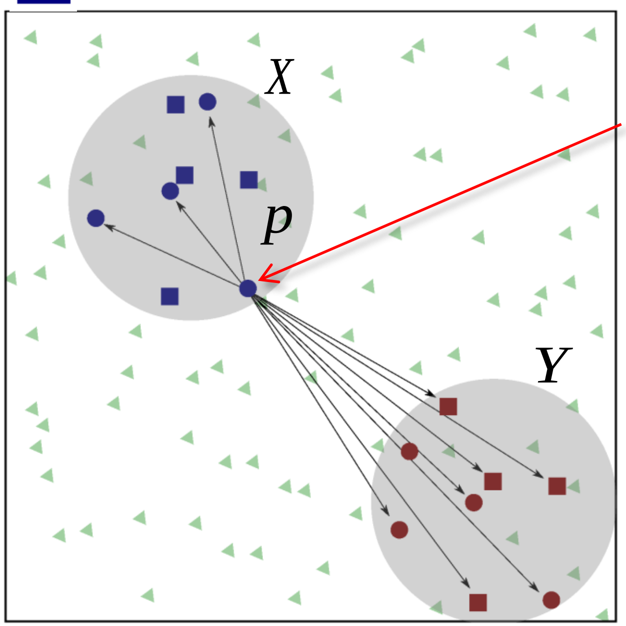
Introduction to λ-dynamics electrostatics computation

λ-dynamics

- The particles in the system can build variable groups called *Sites*
- Sites can have multiple forms *Forms*
  - The number of particles and charges can vary within a site
  - The variability occurs locally

● Particles of *Site X*, *Form π*

■ Particles of *Site X*, *Form κ*



Force acting on particle p

$$\mathbf{F}_p = q_p \underbrace{\sum_{\substack{a \\ a \neq p}}^{Particles(X,\pi)} f}_{\text{same site}} + \underbrace{\lambda_{(X,\pi)} q_p \sum_{\substack{S \\ S \neq X}}^{Sites} \sum_{\alpha}^{Forms(S)} \lambda_{(S,\alpha)} \sum_{a}^{Particles(S,\alpha)} f}_{\text{other sites}}$$

f - *Coulomb force kernel*

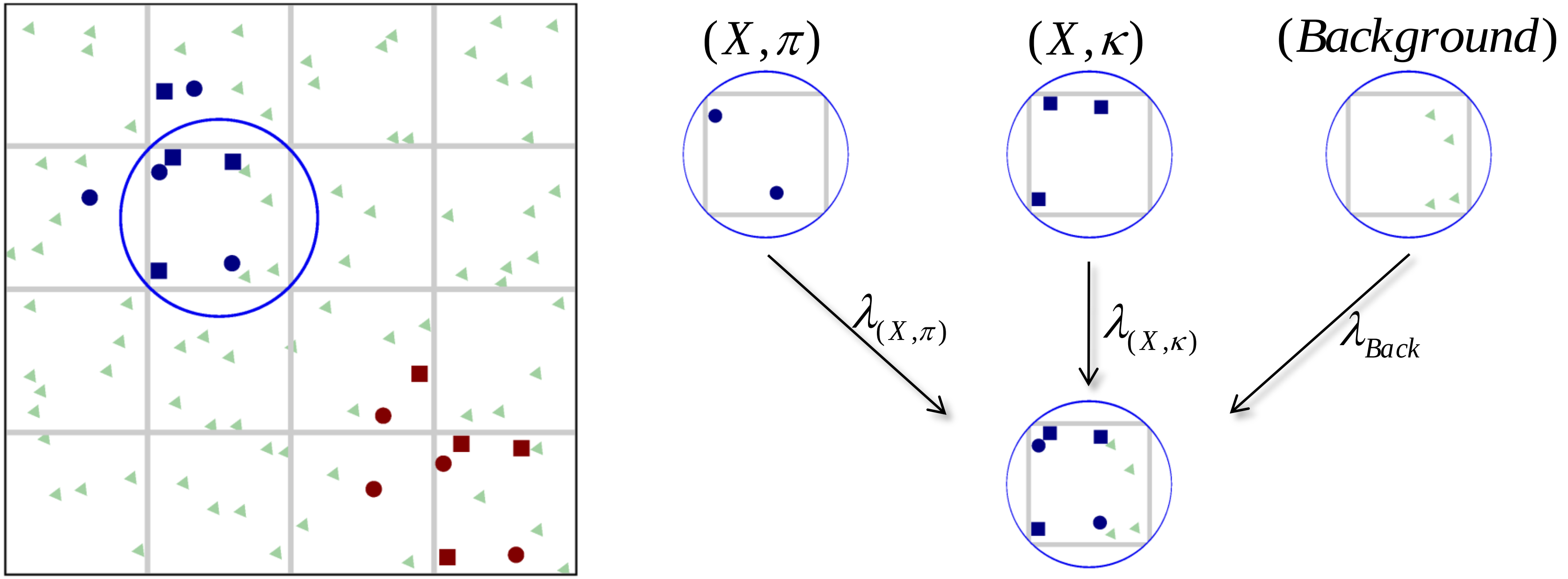
λ-dynamics and FMM

How the λ-dynamics affects the FMM workflow

P2M (particle to multipole)

- For each particle $\mathcal{O}(p^2)$- operation

- For each particle + **build average multipole** $\mathcal{O}(p^2)$
  - Number of P2M operations depends on the number of particles
  - Average multipole building depends on number of forms per box



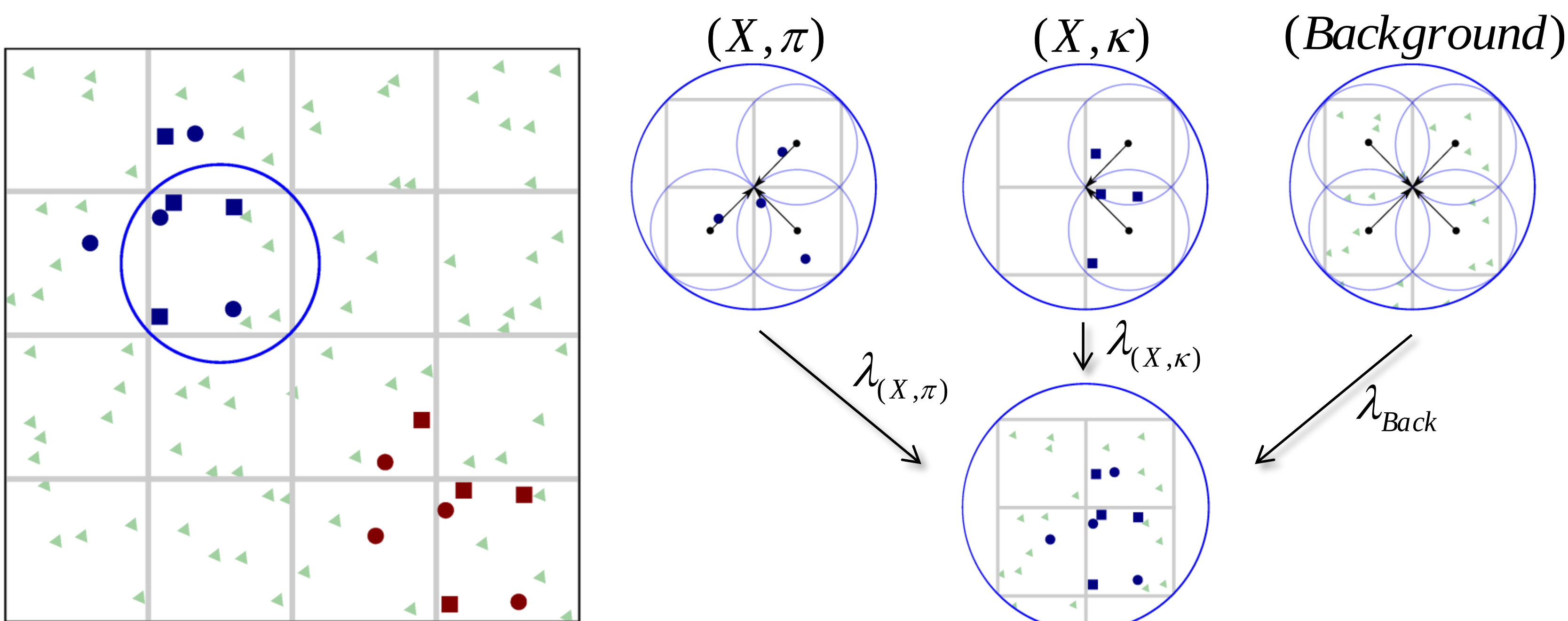
λ-dynamics and FMM

How the λ-dynamics affects the FMM workflow

M2M (multipole to multipole)

- For each box in the tree $\mathcal{O}(p^4)$

- For **each form** in each box in the tree + **build average multipole** $\mathcal{O}(p^2)$
  - Overhead of one p^4 operation for each form of all sites per box!
  - But, impact of M2M to the overall FMM performance ≈ 0.03



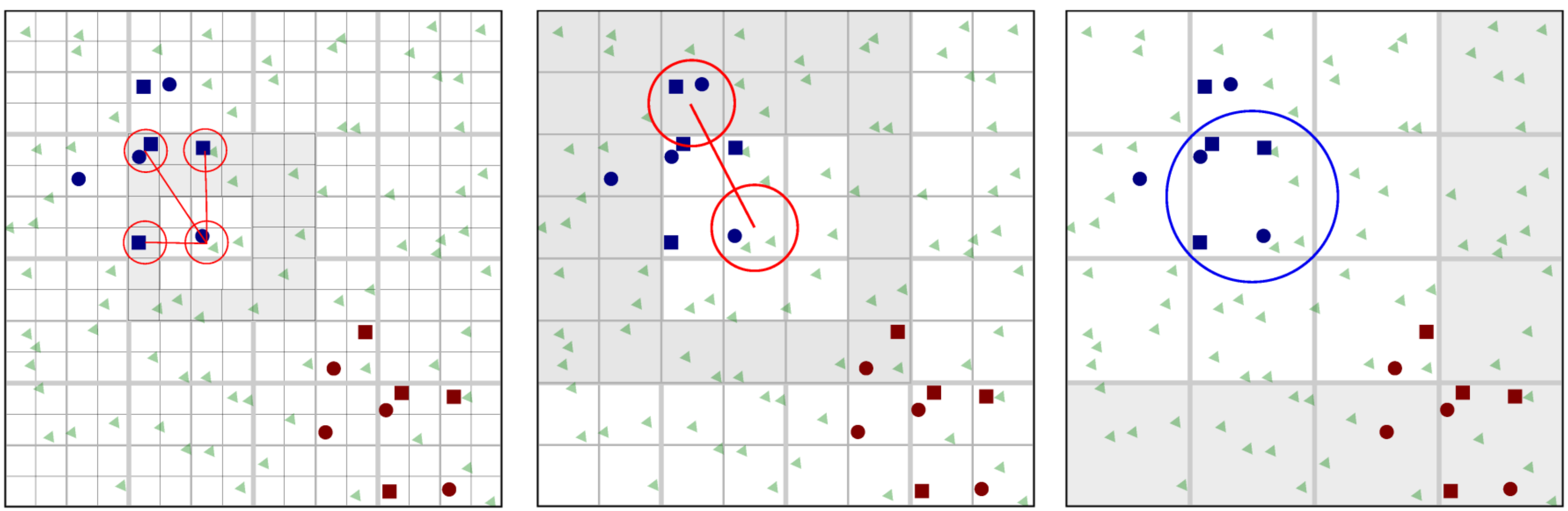
λ-dynamics and FMM

How the λ-dynamics affects the FMM workflow

M2L (multipole to local)

- For each box in the tree ➔ 189 (ws=1) single M2L operations

- For each box in the tree + **depends on site particles distribution**
  - Only intra-site-form interactions need special treatment
  - Occurs only if the same sites interact via multipoles (farfield)



λ-dynamics and FMM

How λ-dynamics affects the FMM performance

N *total #particles,* ***F*** *total #site forms,* ***B*** *# FMM-Boxes,* ***f*** *# forms per box,* ***n*** *# particles per box*

P2M

$N \cdot \mathcal{O}(p^2) + f \cdot \mathcal{O}(p^2)$

P2P (ws=1)

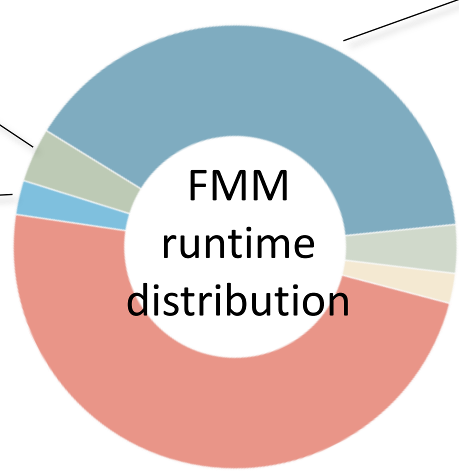
$B(depth) \cdot 27 \cdot \mathcal{O}(n^2)$

M2M

$(B\text{-}1) \cdot \mathcal{O}(p^4)$

→

$(B\text{-}1) \cdot f \cdot \mathcal{O}(p^4) + f \cdot \mathcal{O}(p^2)$



Forces

$N \cdot \mathcal{O}(p^2)$

M2L(ws=1)

$(B\text{-}1) \cdot 189 \cdot \mathcal{O}(p^4)$

→

$(B\text{-}1) \cdot 189 \cdot \mathcal{O}(p^4)$

+

$(B\text{-}1) \cdot x \cdot \mathcal{O}(p^4)$

M2M

$(B\text{-}1) \cdot \mathcal{O}(p^4)$

→

$(B\text{-}1) \cdot f \cdot \mathcal{O}(p^4)$

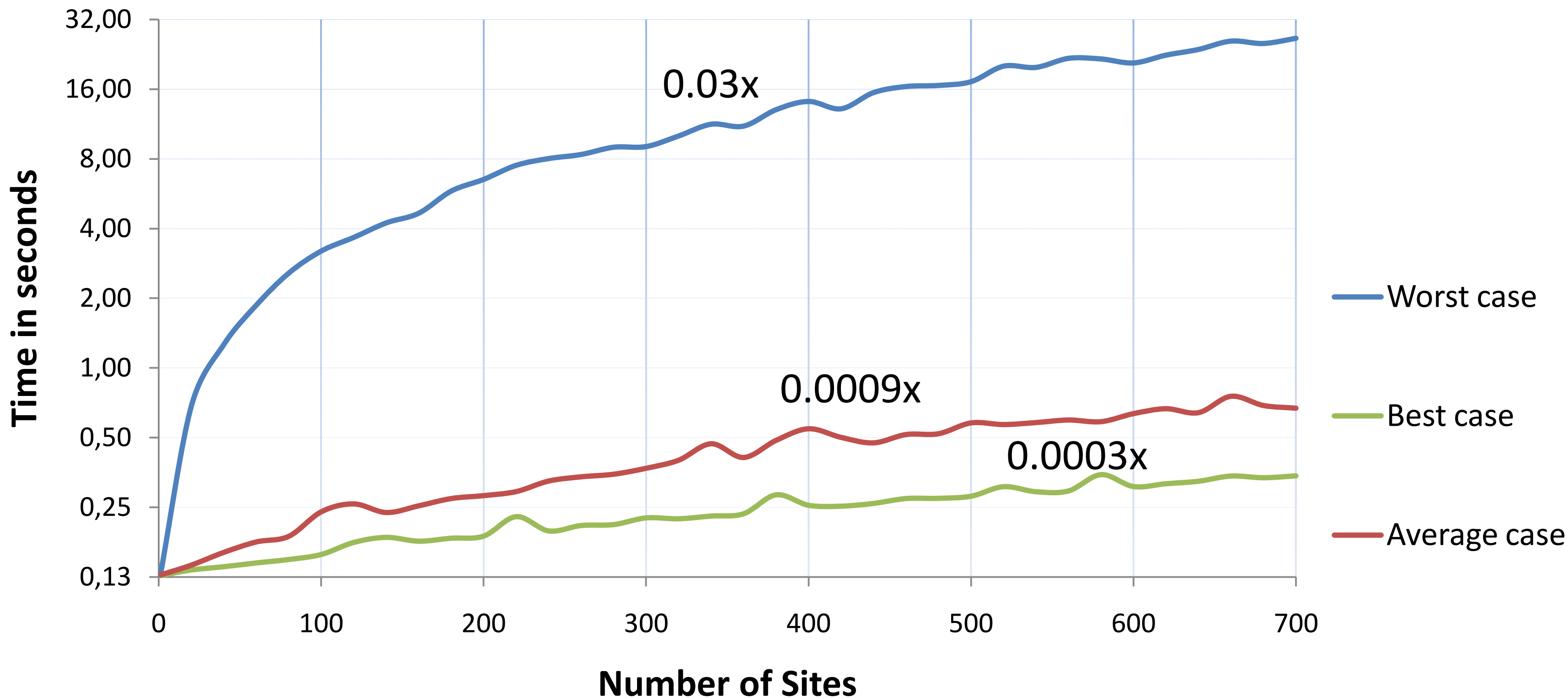
FMM runtime dependency on site distribution

Testrun on a system with random particle distribution (5 forms per Site on average)

Average case – forms span 5 boxes on the deepest level

Best case – every forms of each site span one box on the deepest level

Worst case – every form of each site spans the whole simulation box



Thank You

Questions ?