Harnessing heterogeneous systems for **n**-body problems

Felipe A. Cruz
Tsuyoshi Hamada

Nagasaki Advanced Computing Center
Nagasaki University, Japan

Long Title

Controlling and making use of

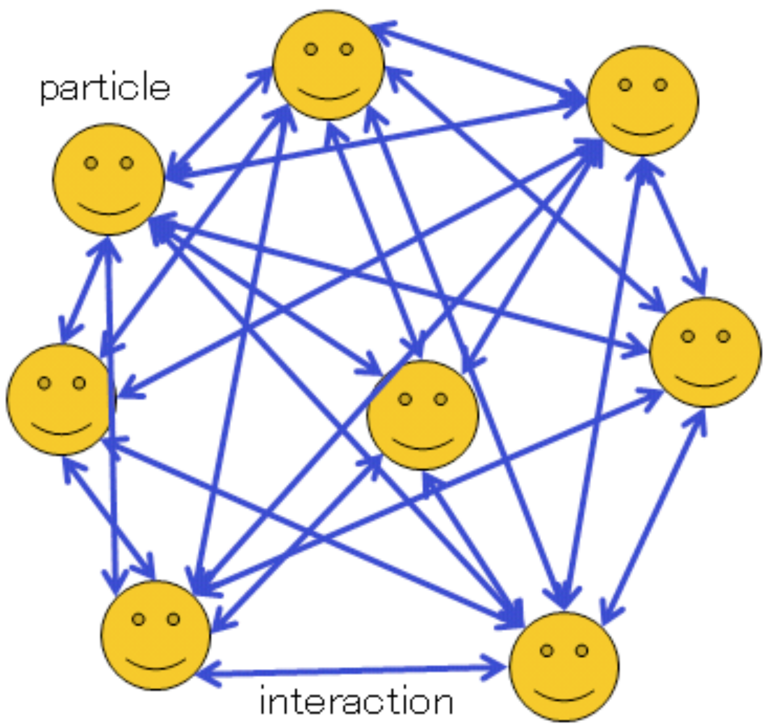
computing systems of diverse characteristics

for the N-body problem.

Application

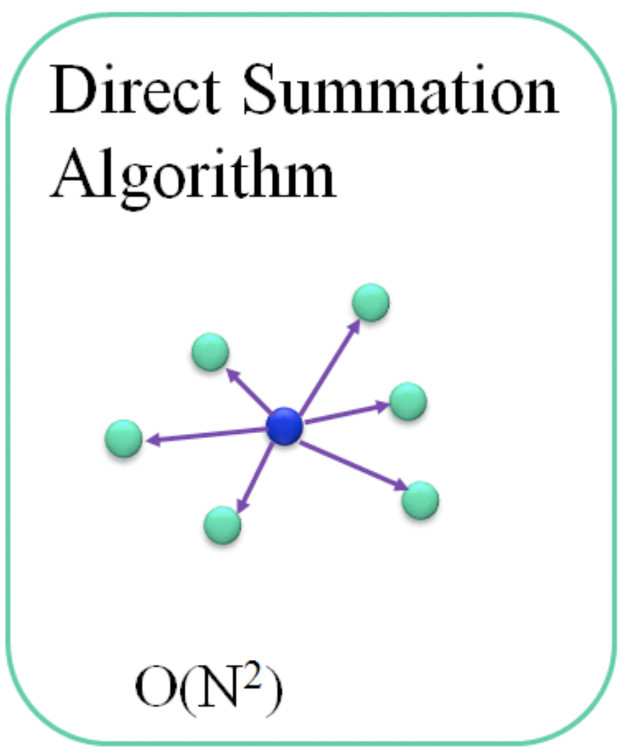
Felipe A. Cruz

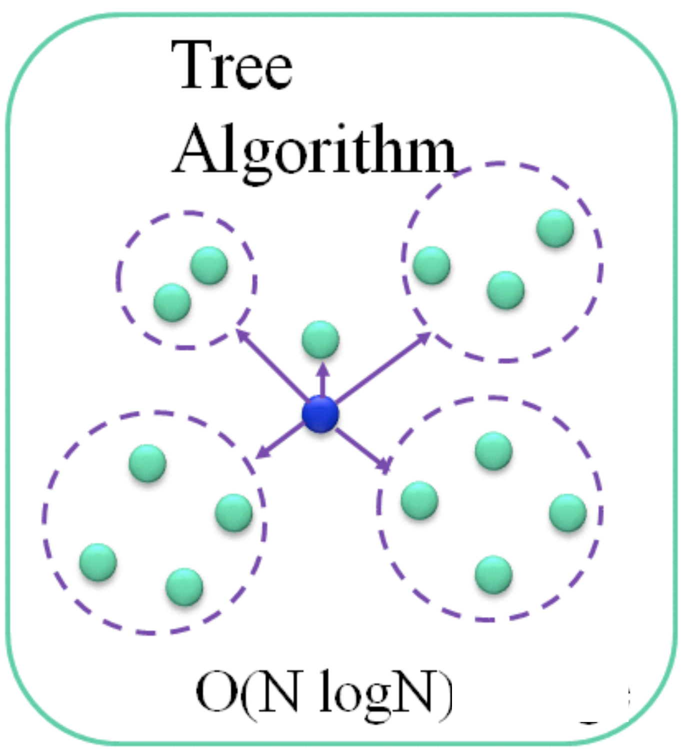
N-body simulation



- All pair-wise **N** particle interactions
  - Stars, Galaxies, Atoms, etc.

- Computational cost
  - Direct sum - O($\mathbf{N^2}$)
  - Tree algorithm - O(**NlogN**)
  - FMM - O(**N**)





Applications of N-body

Poisson

$$\nabla^2 u = -f$$

Astrophysics

$$\nabla^2 \phi = 4\pi G M$$

Electrostatics

$$\nabla^2 \phi = -\frac{q}{\epsilon_0}$$

Fluid Mechanics

$$\nabla^2 p = -\nabla \cdot \{\mathbf{u} \cdot (\nabla \mathbf{u})\}$$

$$\nabla^2 \mathbf{u} = -\nabla \times \boldsymbol{\omega}$$

Helmholtz

$$\nabla^2 u + k^2 u = -f$$

Acoustics

$$\frac{1}{c^2}\frac{\partial^2 \phi}{\partial t^2} = \nabla^2 \phi$$

Electromagnetics

$$\mu_0 \epsilon_0 \frac{\partial^2 \mathbf{E}}{\partial t^2} = \nabla^2 \mathbf{E}$$

$$\mu_0 \epsilon_0 \frac{\partial^2 \mathbf{H}}{\partial t^2} = \nabla^2 \mathbf{H}$$

Quantum Mechanics

$$\frac{1}{c^2}\frac{\partial^2 \phi}{\partial t^2} = \nabla^2 \phi - \frac{m^2 c^2}{\hbar^2}\phi$$

The need for **performance**

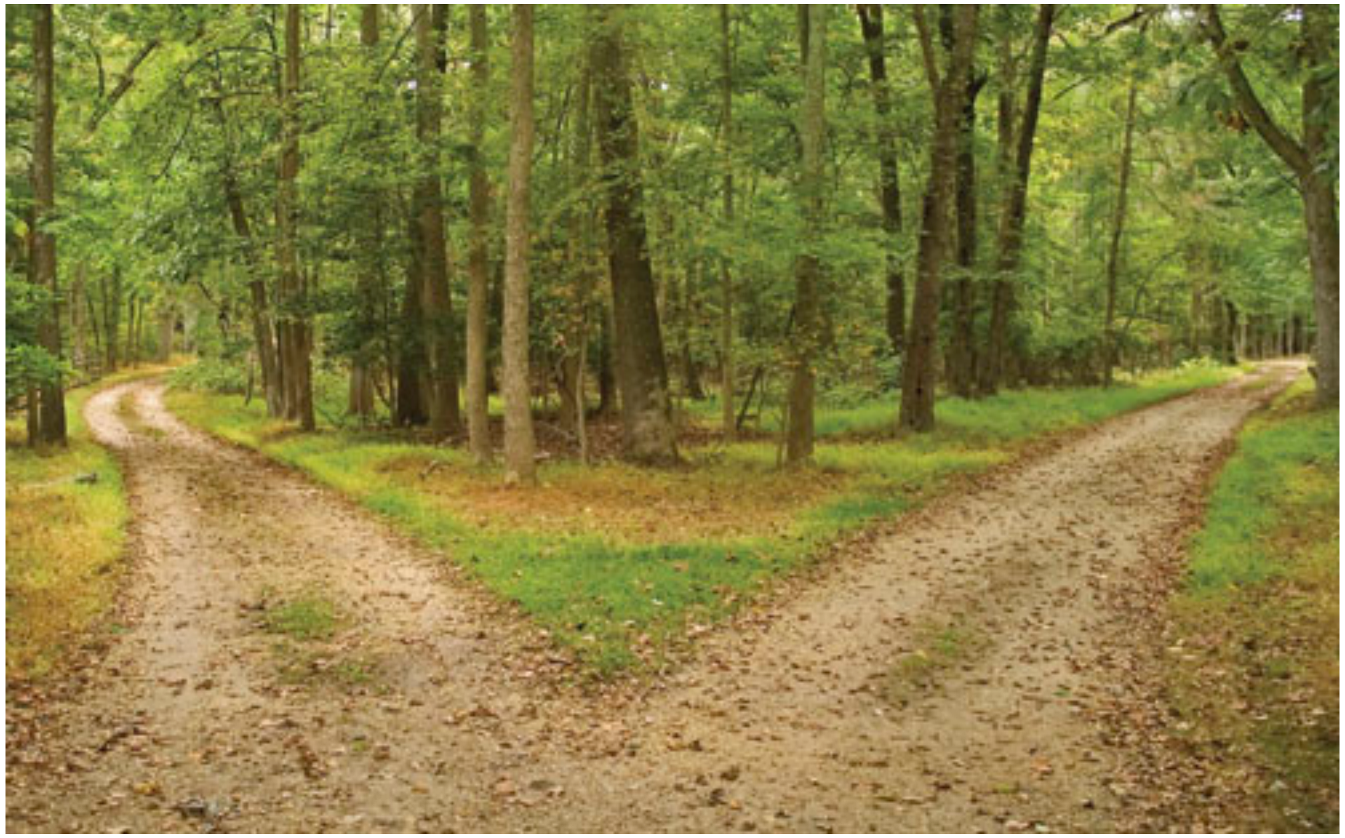
Felipe A. Cruz

To generate new knowledge that allow us to understand, predict, and control our environment.

We use computers to solve increasingly **bigger** and **more complex problems** with **higher accuracy** and in **less time.**

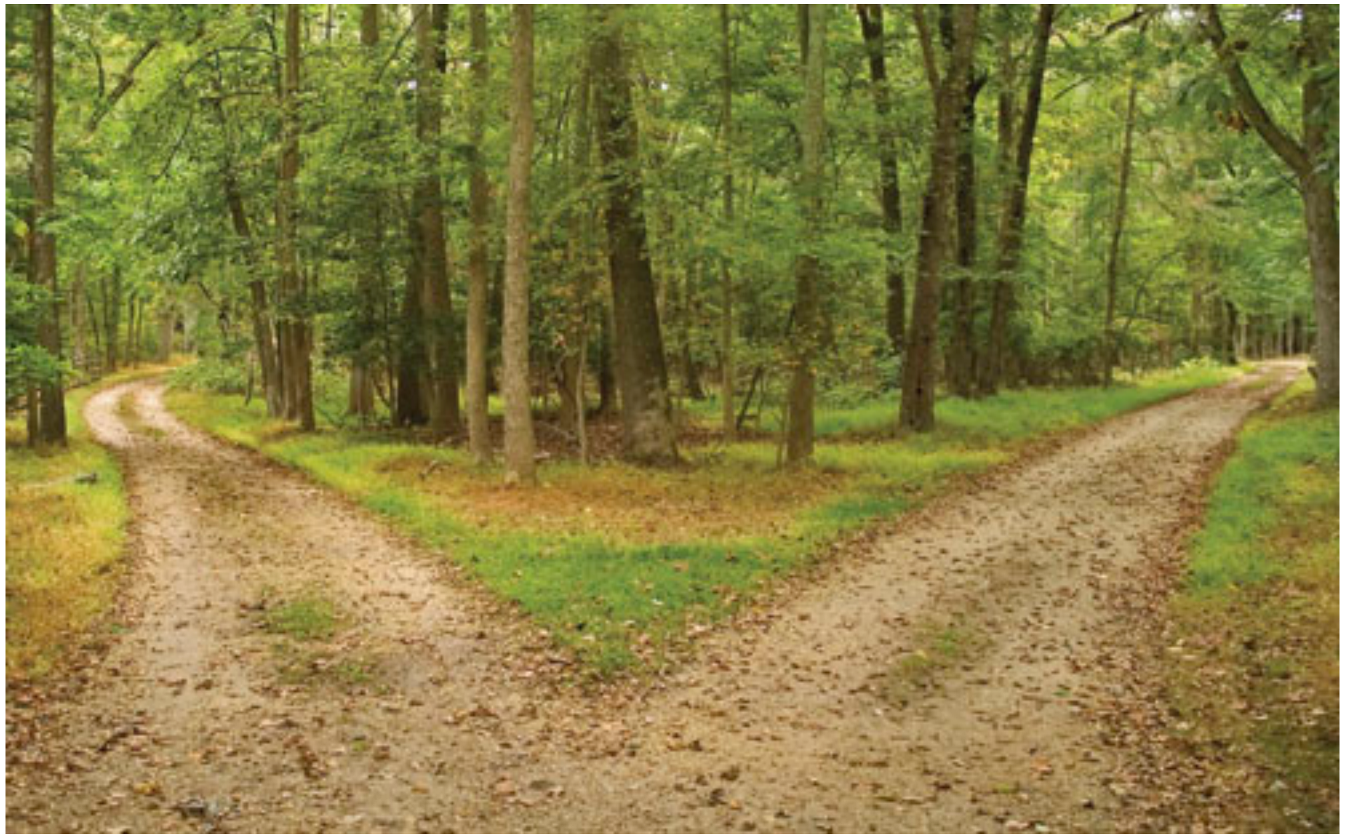
Therefore, we **need high performance** systems**.**

Paths to performance



Paths to performance

Faster processors

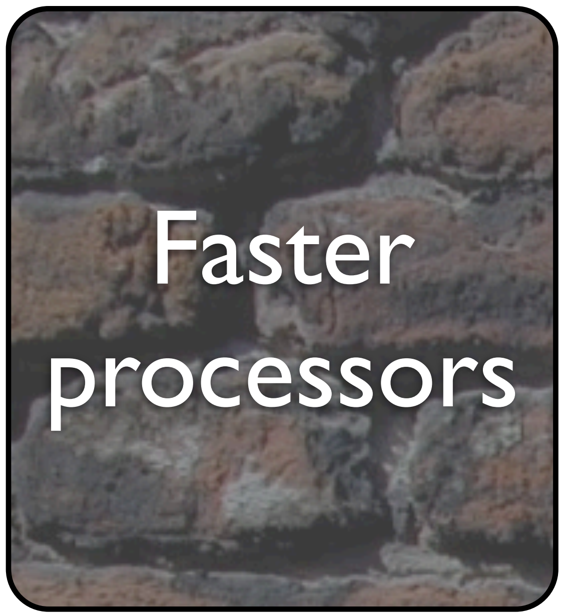


Important method for **improving performance**.

Processors speed improved from **1 MHz** (1980s) to over **4GHz** (2000s)

Over **1,000 fold** faster clock rates in **30 years**!

Paths to performance



Memory wall: large gap between memory and processor speed.

Power wall: higher clock speeds require exponential power increase.

Paths to performance

Faster processors

Use concurrency

Processors with many components that ***may execute*** in parallel.

Benefit from ***aggregated performance***.

Performance

=

effective use of concurrency

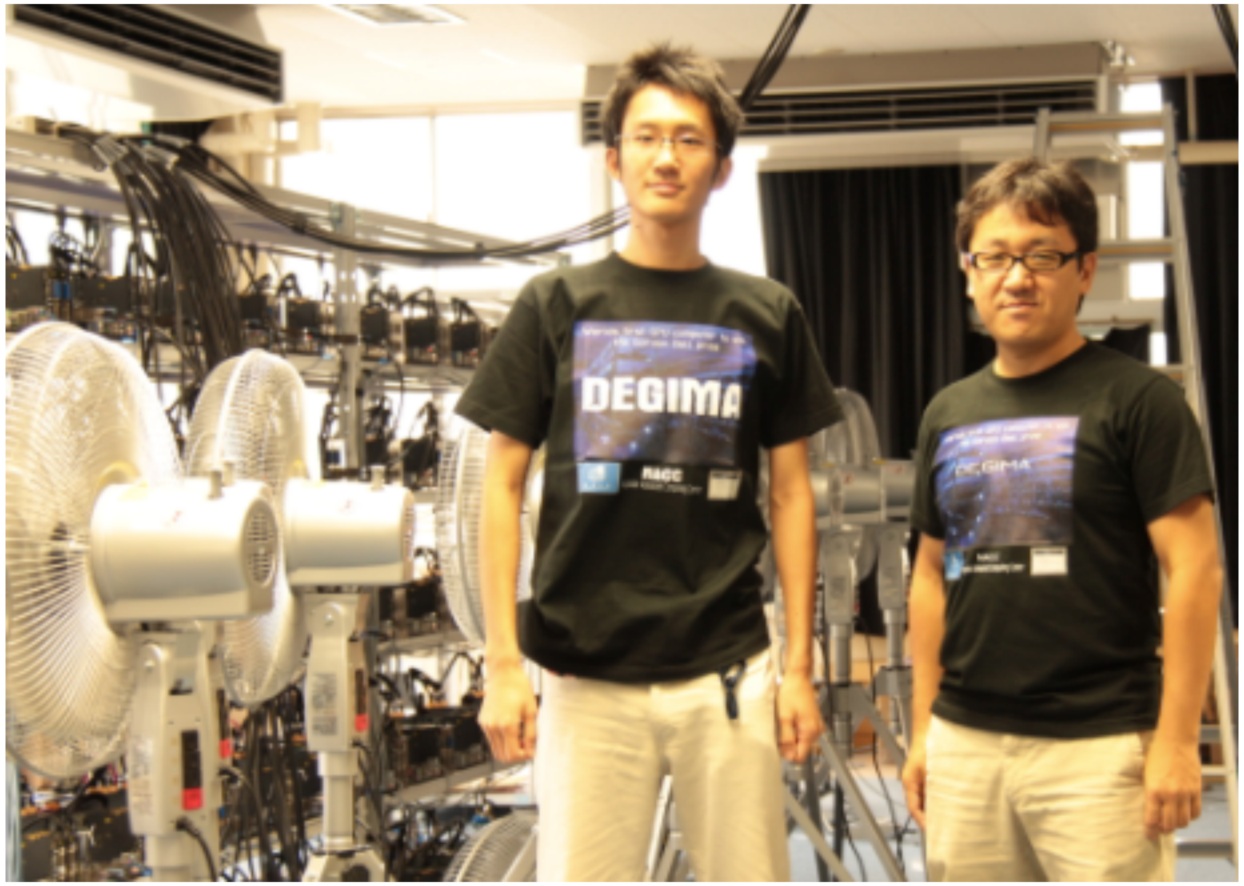
DEGIMA

Felipe A. Cruz

DEGIMA project

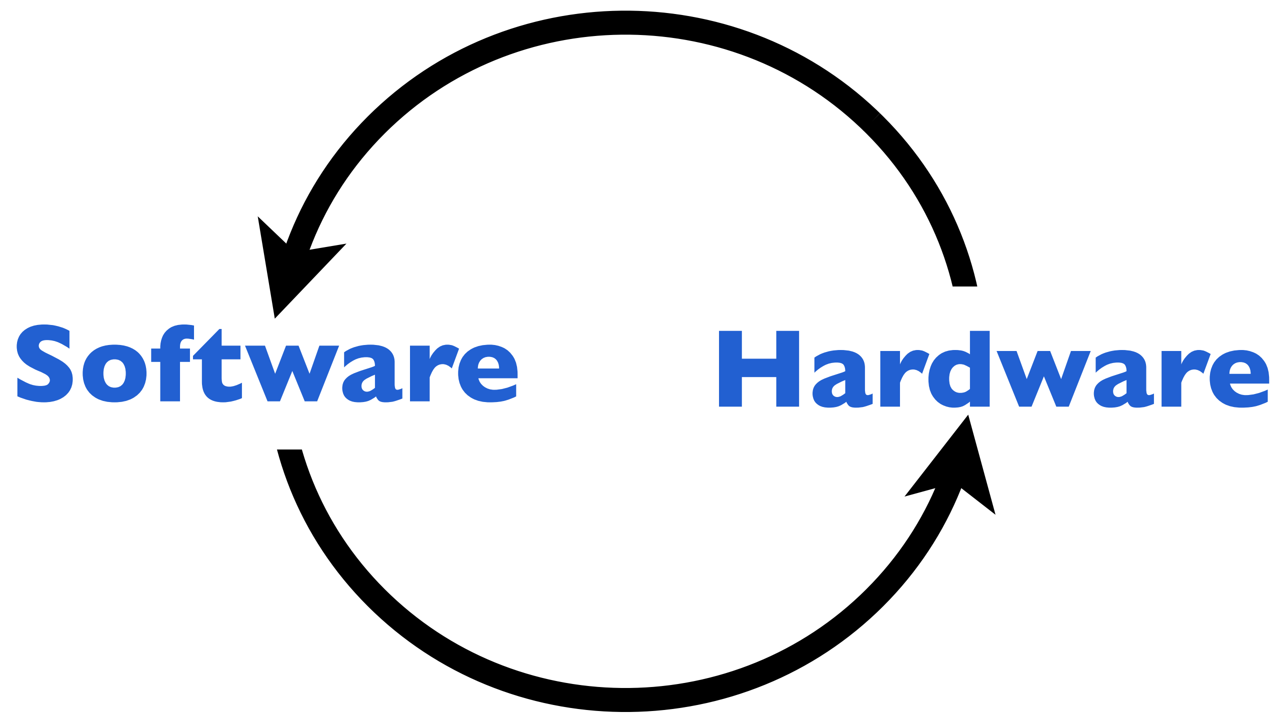
The name DEGIMA remind us of our goal:

DEstination for
GPU
Intensive
MAchine



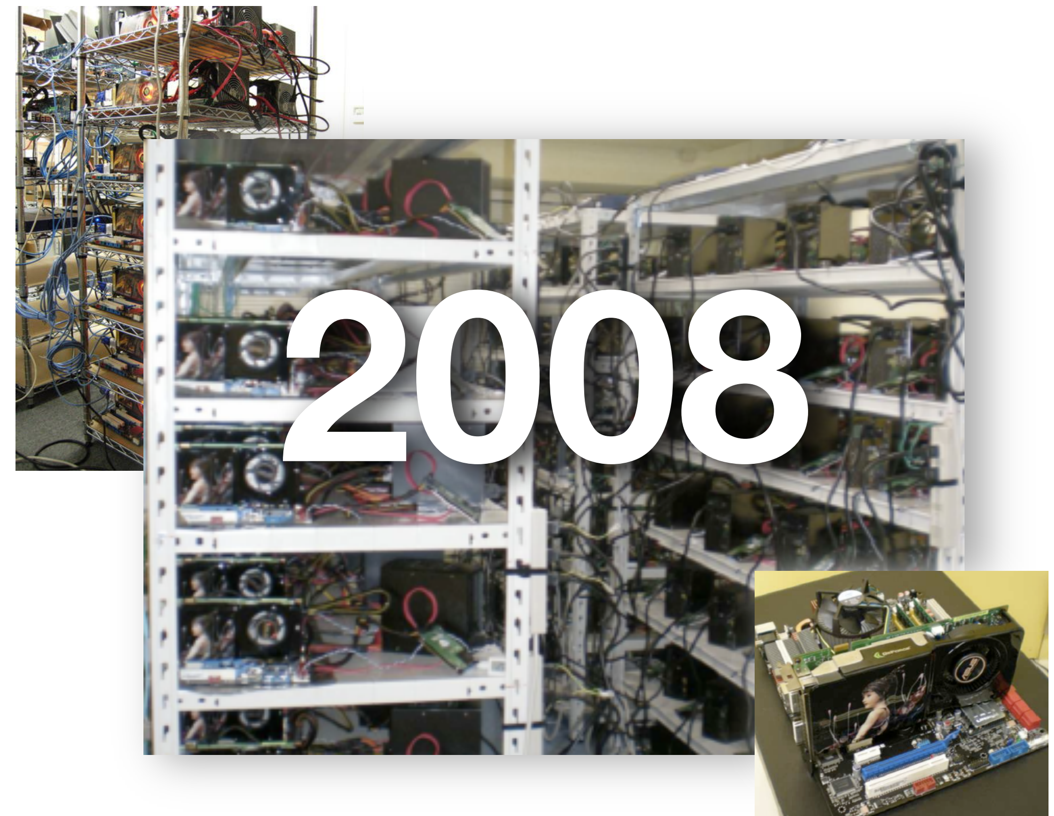
DEGIMA development

- **Driven by** software and hardware iterations



- Each update **overcomes a system bottleneck.**
- Each update **uncovered a new challenge!**

DEGIMA 2007 ~ 2009



DEGIMA 2010

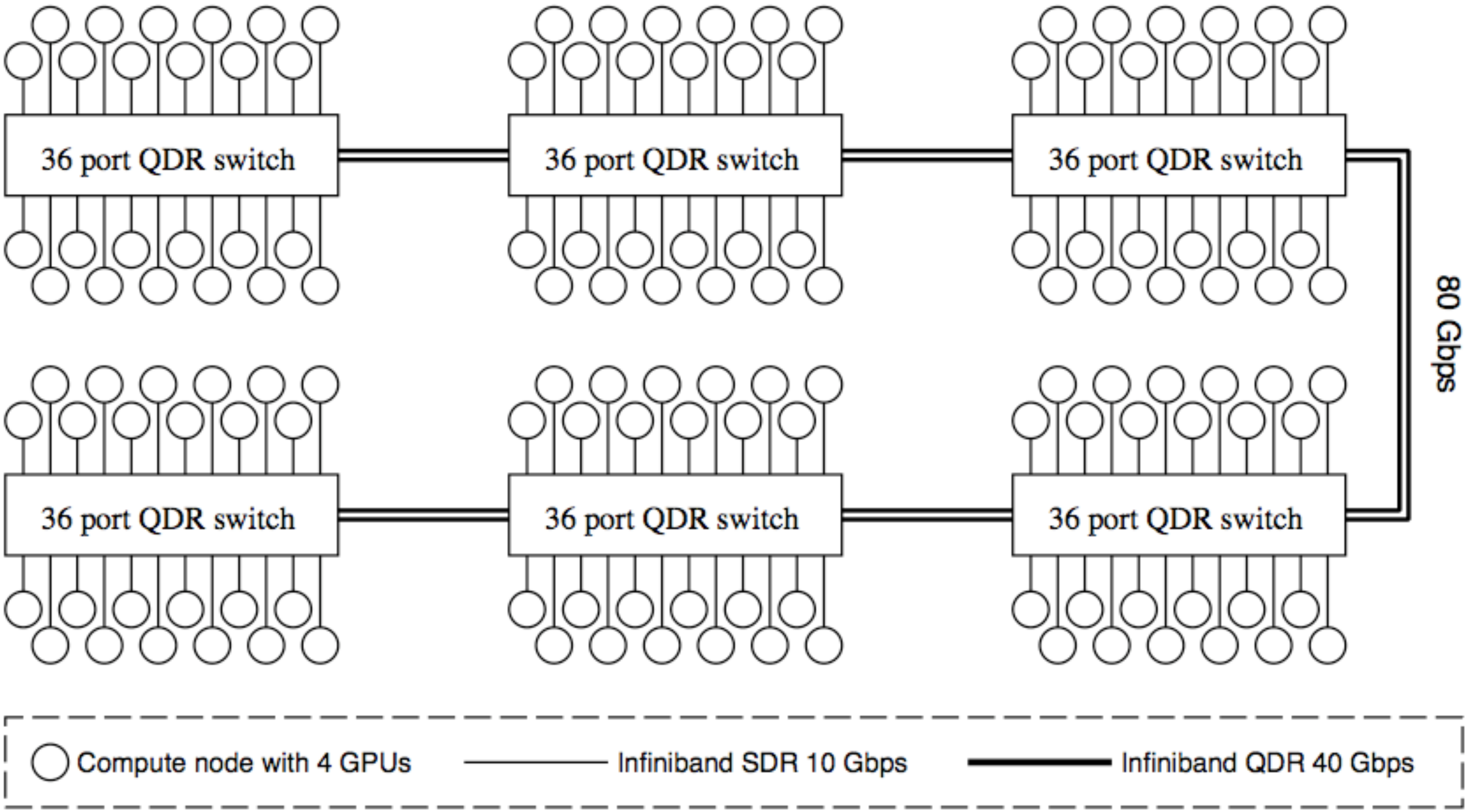
Current system:

- **144** Core i7 920
- **576** GeForce **GT200**
- **144** x **12** GB DDR3 ram
- **10** Gbps **IB** interconnect

The path to DEGIMA

Year	CPU	GPU	Memory	Interconnect	Performance
2007	80 Xeon 3.2 Ghz	40 GeForce G80	80 GB DDR2	1Gbps GbE	1Tflops
2008	128 Core2Quad 2.4 GHz	128 GeForce G92	1 TB DDR2	1Gbps GbE	20 Tflops
2009	128 Core2Quad 2.4 GHz	256 GeForce G92	1 TB DDR2	1Gbps GbE	40 Tflops
2010	**144 Core i7 920**	**576 GeForce GT200**	**1.7 TB DDR3**	**10 Gbps IB**	**190 Tflops**

DEGIMA system configuration



Challenges

- Computing systems present a **hierarchy of parallelism**: distributed memory, shared memory, task parallel, data parallel, superscalar.
- Computing nodes are **heterogeneous** and **compute dense**: CPU, GPU, FPGA.
- An **optimal implementation** depends on the architecture.

Moore’s law at the computer history museum, mountain view.
Recently renovated!

How do we keep up with Moore’s law?

- Distributed systems.
- Compute dense nodes.
- Heterogeneous!

Expose all concurrency in your algorithm!

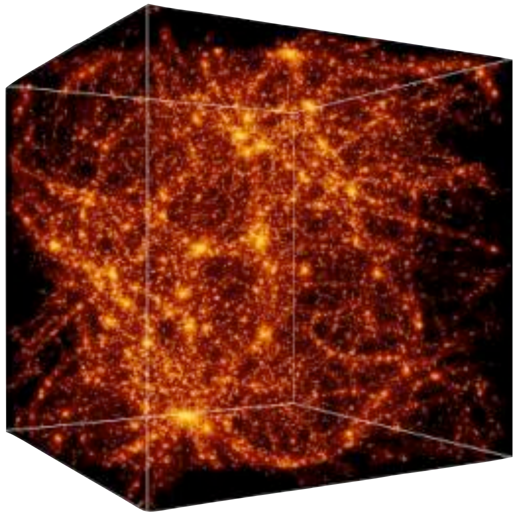
NACC
Nagasaki Advanced Computing Center

Nagasaki University

Fast summation algorithms

Felipe A. Cruz

The summation algorithms



N-body simulation: millions of pair-wise particle interactions.

- Accelerate the evaluation of problems of the form:

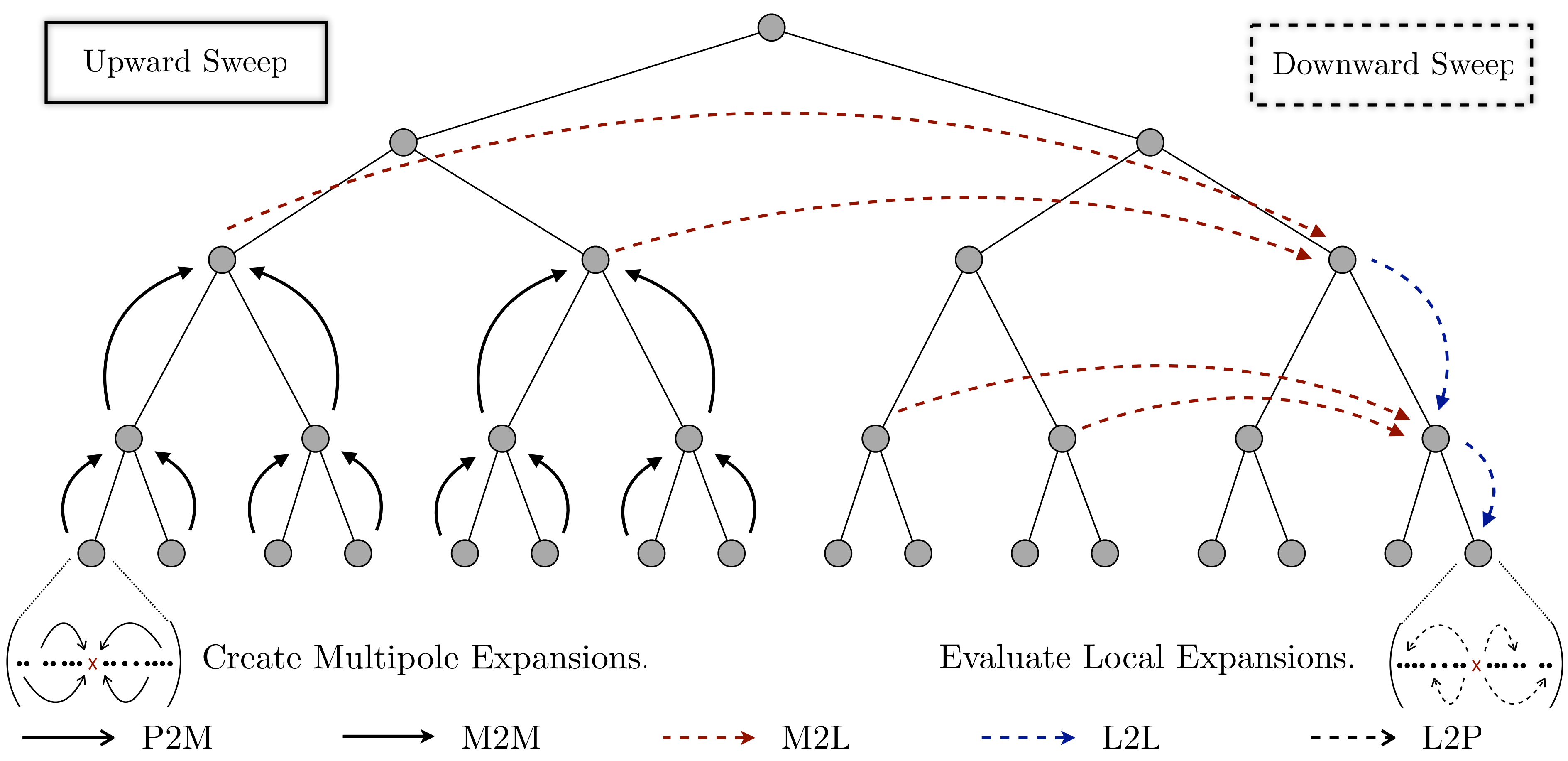
$$f(y) = \sum_{i=1}^{N} c_i \mathbf{K}(y - x_i) \qquad y \in [1 \ldots N]$$

  - For N evaluations the **total amount of work** is proportional to $\mathbf{N^2}$

- The FMM and tree-codes exchange **accuracy for speed**.

Complex algorithms

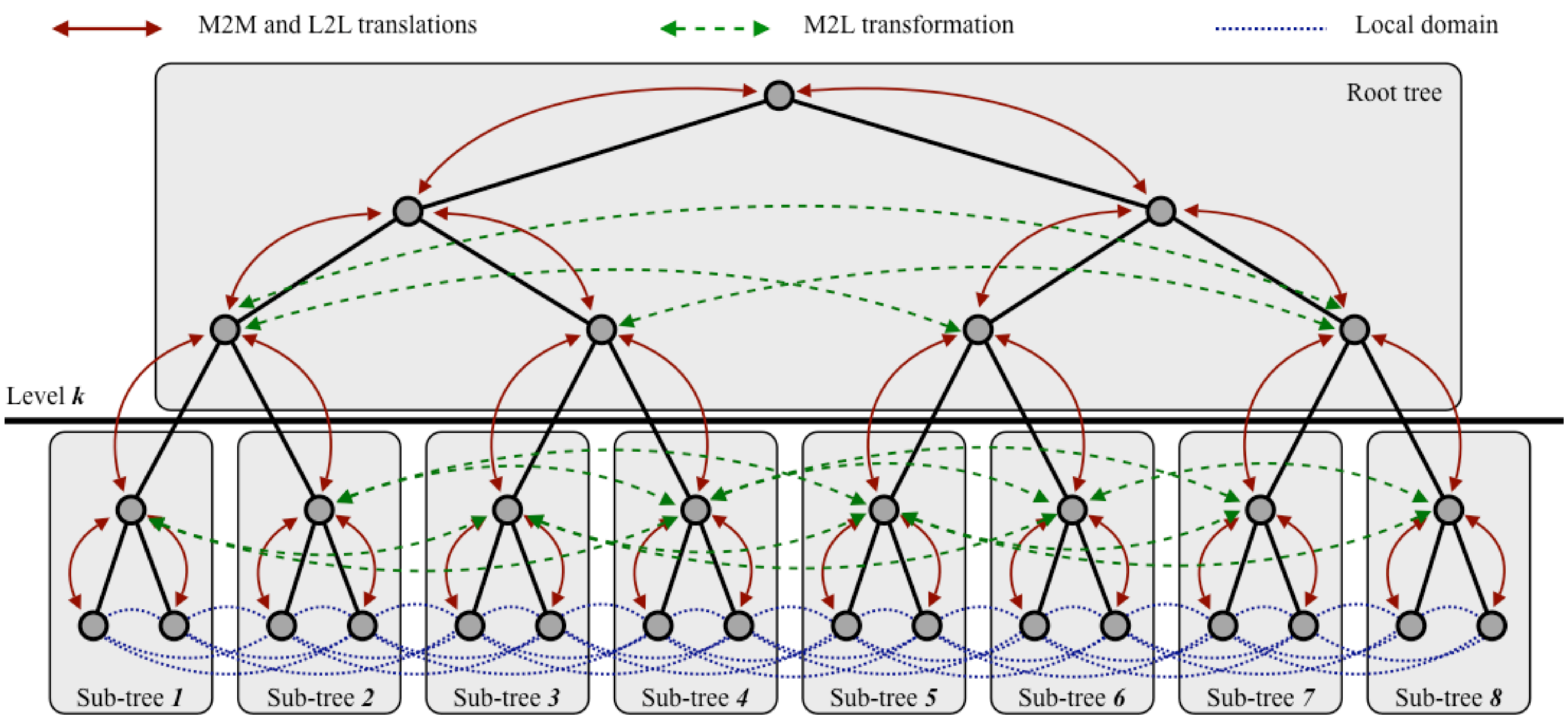
- Algorithm: traverse the tree to find spatial relations between clusters.
- Performs computations while traversing tree (Sweeps).



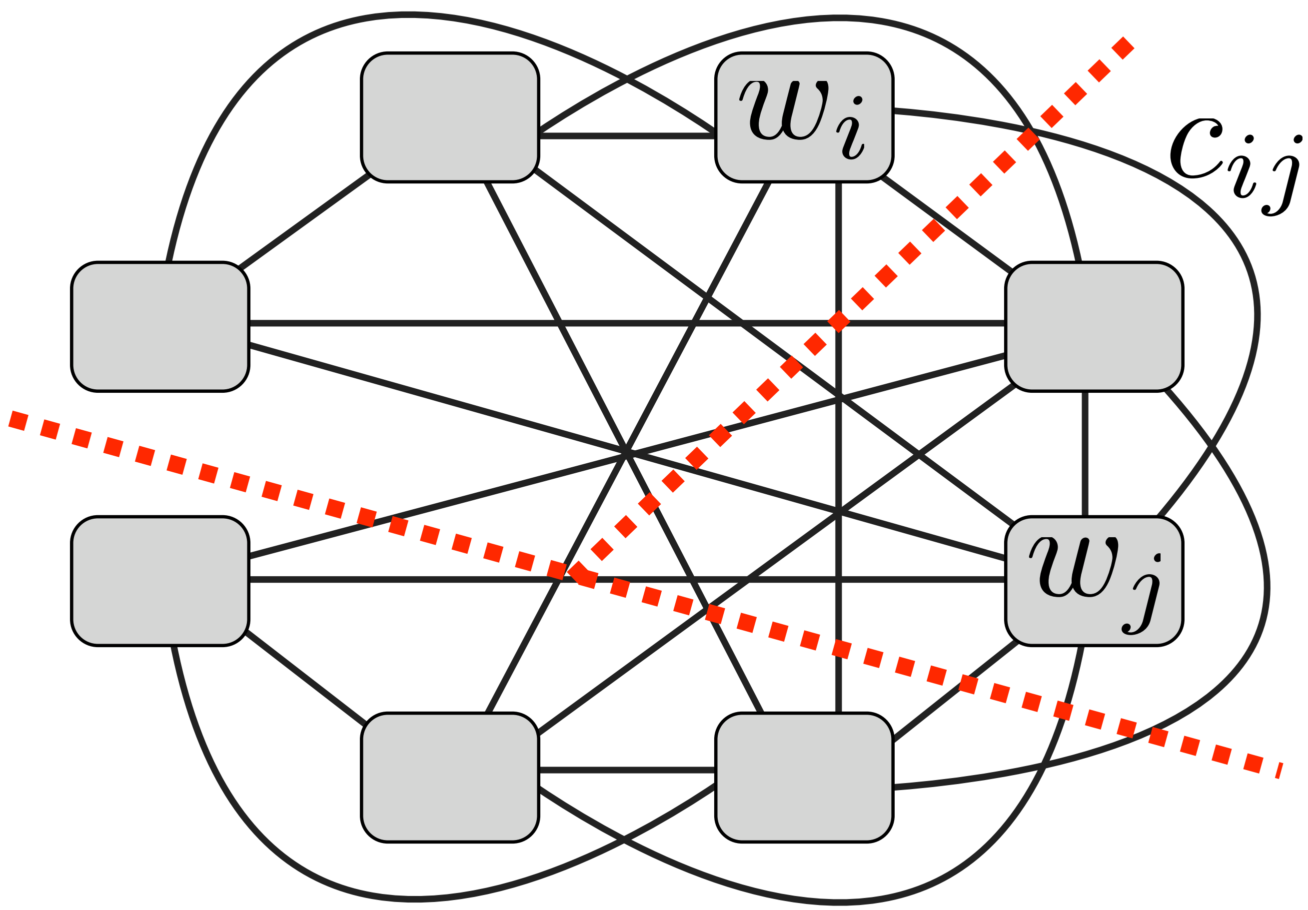
Distributed FMM

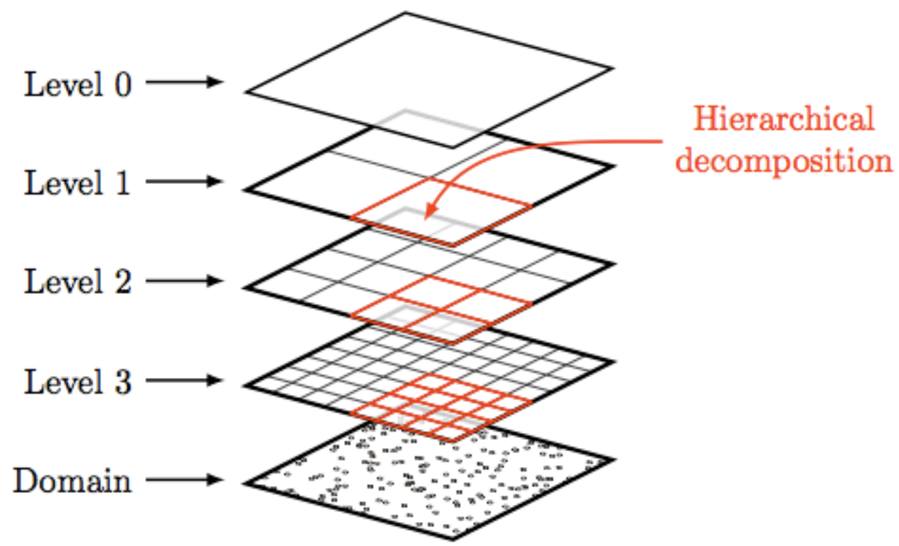
Felipe A. Cruz

Algorithm subdivision



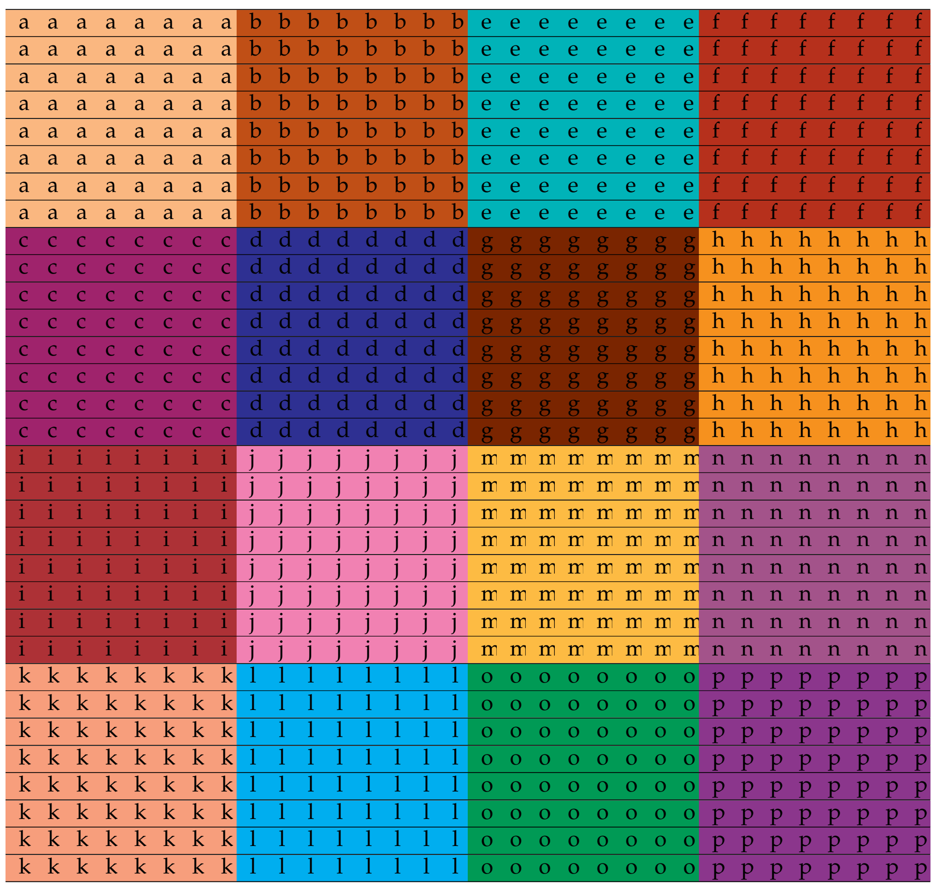
Load balancing



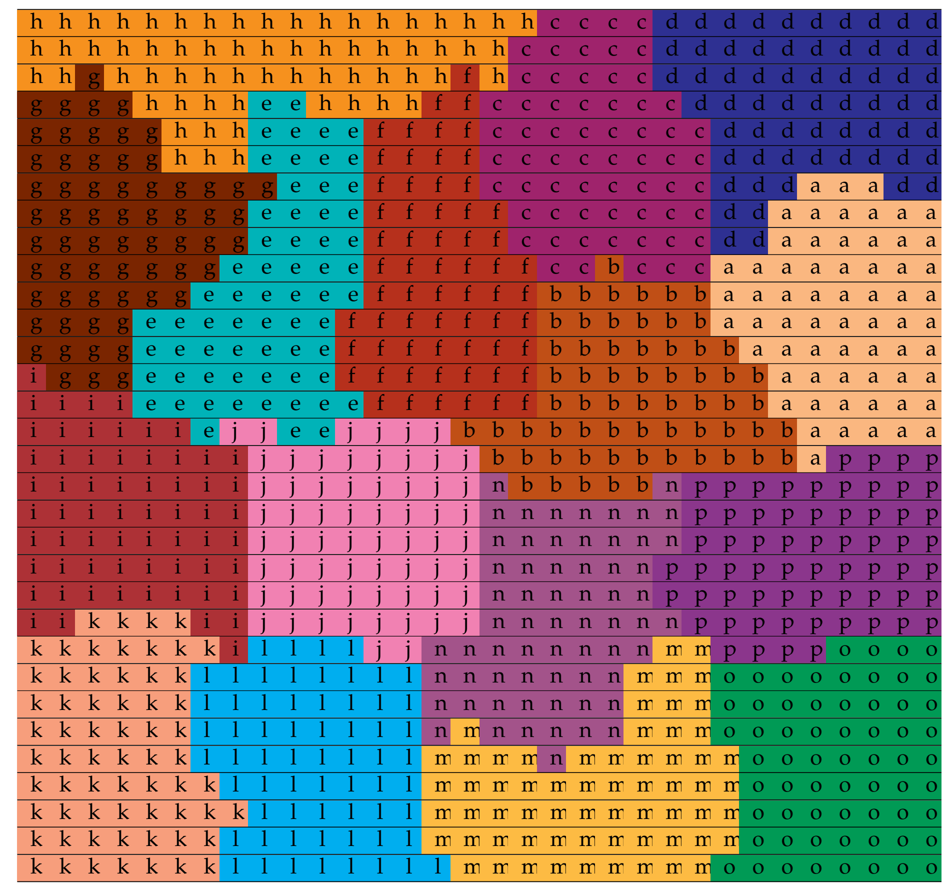


Partition strategy

- Partitioning strategy: load balancing (k=5, L=9)
- MPI implementation.



Simple distribution of subtrees into 16 partitions.
Partition generated using box numbering (z-order).
Space filling curve: exploit numbering locality.



Optimized distribution of subtrees into 16 partitions.
Partition generated using ParMETIS.
ParMETIS: balanced partitions that minimize communications.

NACC
Nagasaki Advanced Computing Center

Nagasaki University

Heterogeneous FMM

Felipe A. Cruz

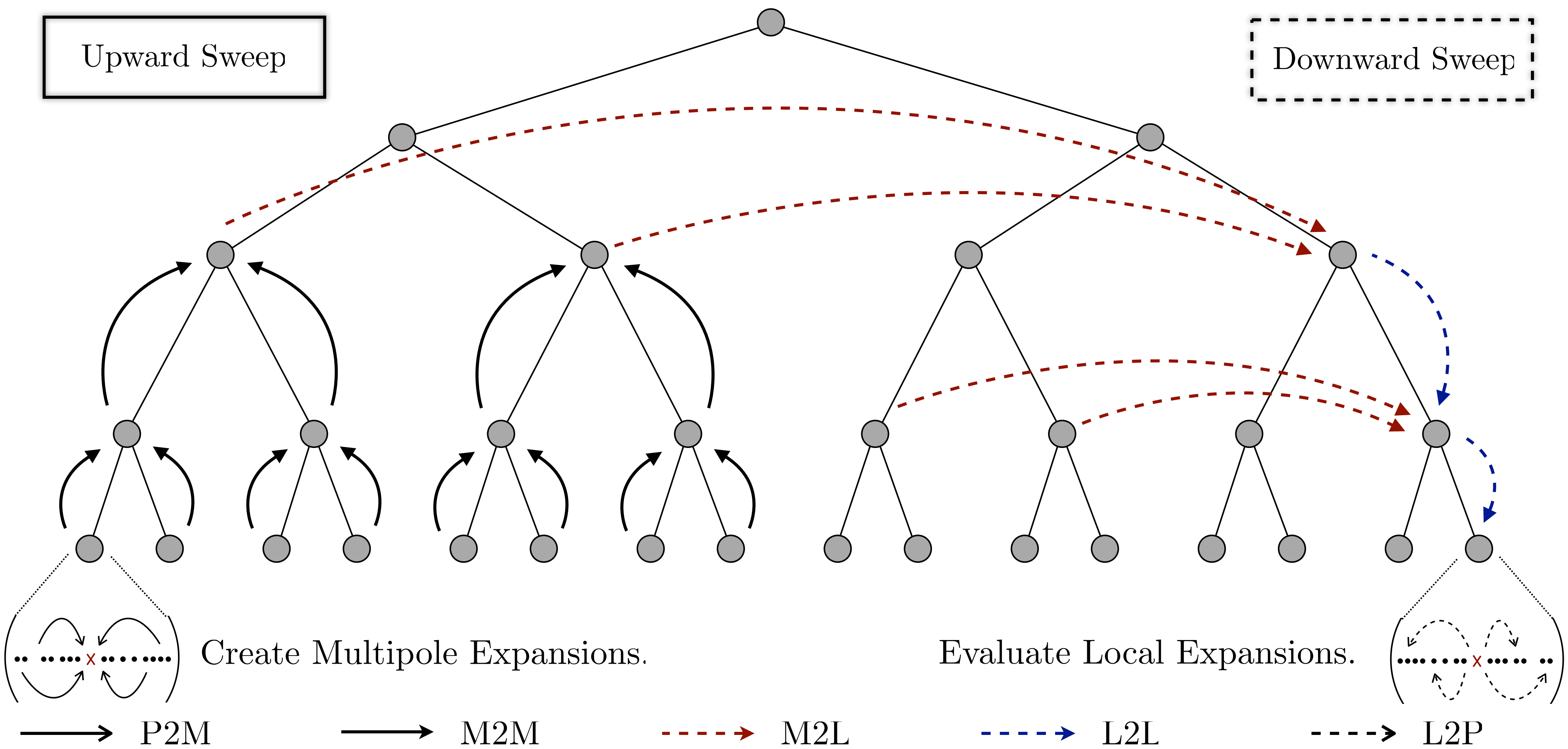
Heterogeneous strategy

Concurrent Queues:

- **Expose task level** and **fine-grained** parallelism.
- Make use of **data-temporal locality**.
- Abstraction from **work declaration** and **work execution**.
- Allow **heterogeneous queue execution**: multicore and GPU.
- Allow **parallel queue execution**.

Work **reorganization**: From hierarchical structure to ...

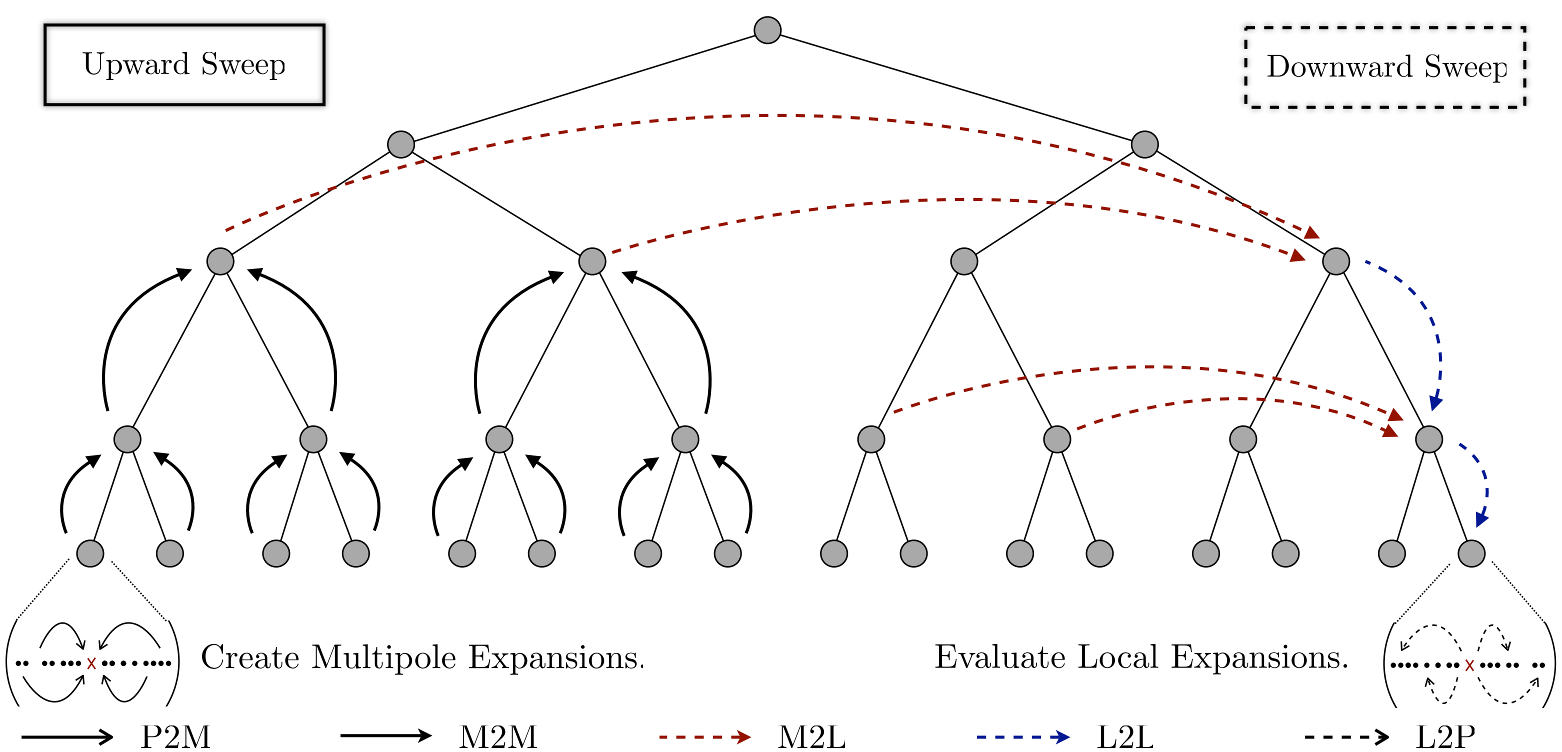
Bad access pattern.



Bird's eye view represents the algorithm: computations are performed while traversing the tree.

Work **reorganization**: From hierarchical structure to a Queue.

- **Homogeneous** units of work.
- Improved **temporal data locality**.
- Tree traversal: find spatial relations and queue work.

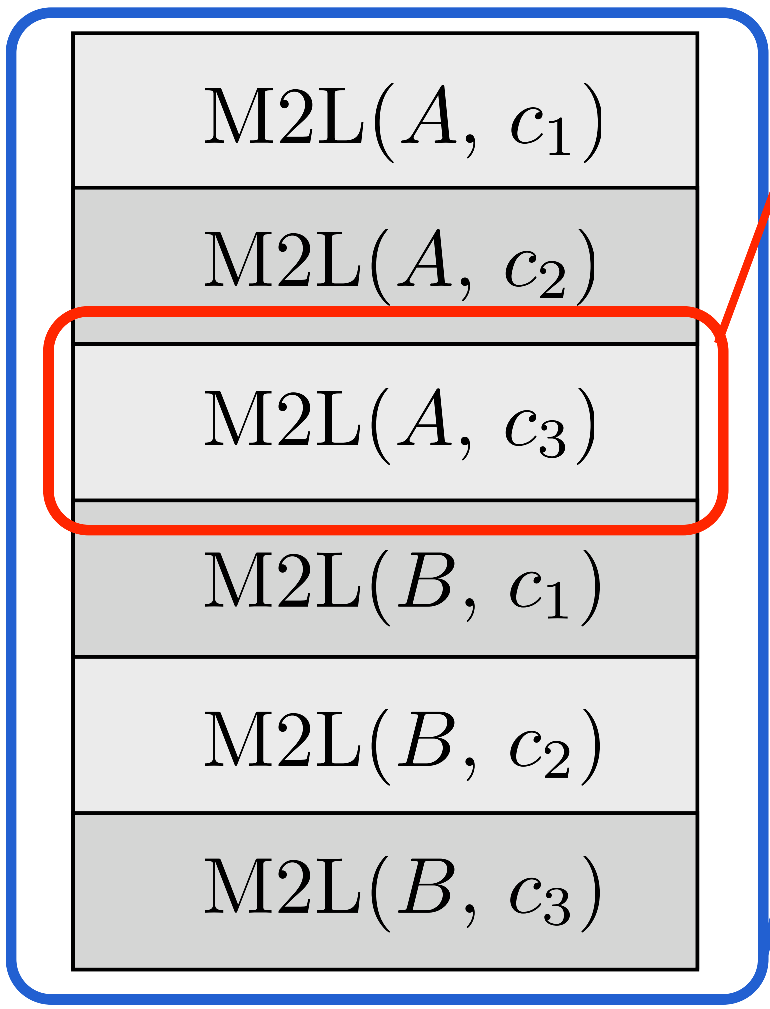


Bird's eye view of the FMM algorithms: Sketch of the FMM algorithm that relates computations to the hierarchical algorithmic and data structure.

Reorganize computations →

Reorganized Task Queue
M2L(A, c_1)
M2L(A, c_2)
M2L(A, c_3)
M2L(B, c_1)
M2L(B, c_2)
M2L(B, c_3)

Queue of tasks: Computational intensive tasks are queued and reorganized.



Queue component description: Example shows multiple queued task, regrouped for execution.

Task — **Parallel work unit**

- Sets task Input/Output.
- Defines data dependencies.
- Specifies concurrent and homogeneous unit of work.

Queue — **Work organization**

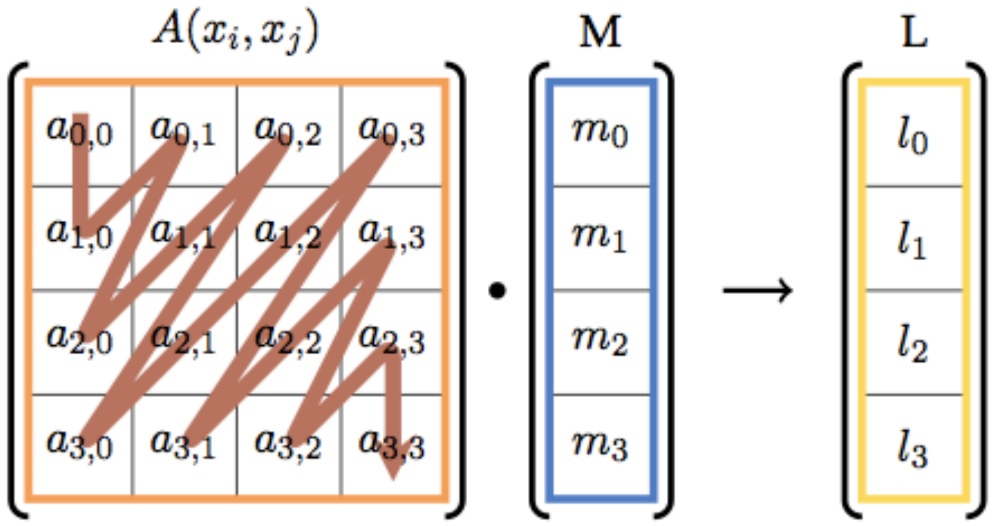
- Group tasks for temporal data-locality.
- Abstracts work declaration from execution.
- Implement queue execution: multicore, gpu.
- Improved access pattern.

Targeted optimization

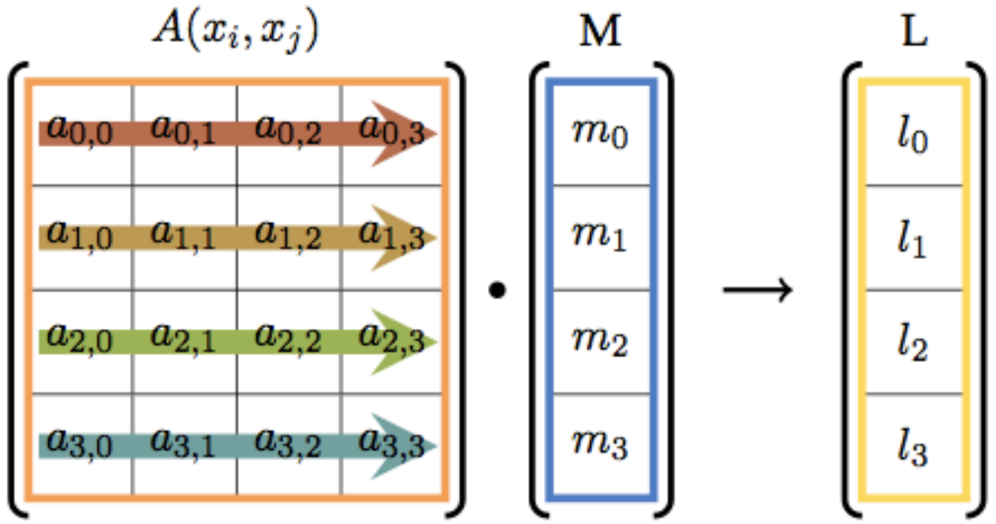
The Multipole to Local (M2L) stage:

- M2L stage **over 99%** of computation time
- **Hundreds of thousands** of M2L translations
- **Matrix free**, and **computationally intensive**
- Top performance: **548 Gflops** per C1060 GPU
- GPU code: **Less serial-efficient but more parallel!** not the same CPU code! 20x faster!

Right: Reformulating the calculations of a single translation to better map the GPU massively parallel architecture



(a) Traversing the matrix diagonals.



(b) By rows.

Conclusion

Conclusions:

- No silver bullet!
- Concurrency is your friend!
- Break the problem into smaller simpler problems.
- Use algorithms to optimize parameters:
  - Dynamic load balance, work distribution, auto-tuning.
- No heroes, we need interdisciplinary collaborations!

Future work:

- Automatic load-balancing between available hardware.
- Automatic optimization of the critical path execution.

Thank you