



**Northern Illinois
University**

Electron Cooling for an Electron Ion Collider: Computational Methods and Code Development

**Herman D. Schaumburg
Béla Erdélyi**

DOE-NP Accelerator R&D PI Meeting
November 13, 2018

Main Goals



- Proof of principle **demonstration of electron cooling** with the **first particle-level detailed** simulations (Jones Report:: line 3: High, A)
- **Develop a high-performance code** with these capabilities, and other beam dynamics challenges beyond cooling (Jones Report:: line 4: High, A)
- **Applications** to electron-ion collider (JLEIC) modeling, design, and optimization (Jones Report:: line 39: High)

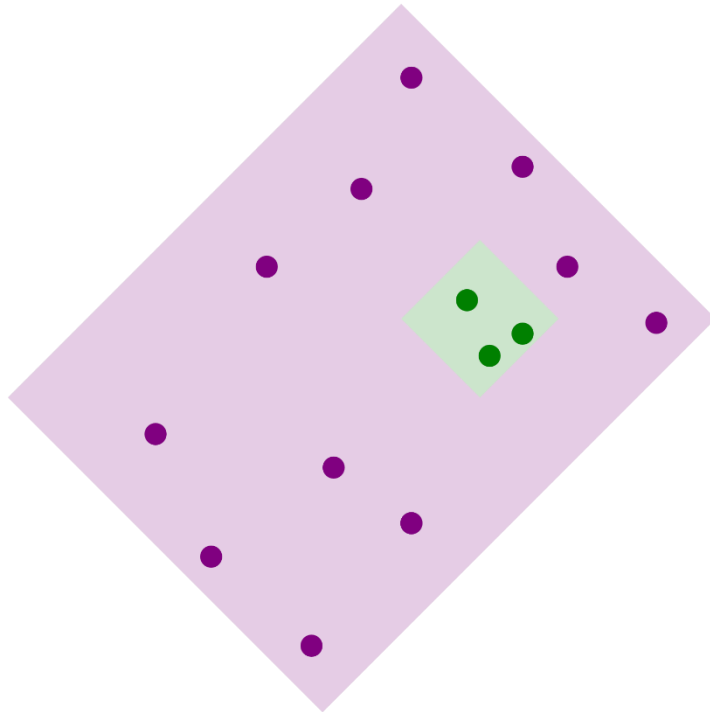
Expenditures and Milestones



	FY10+FY11	FY12+FY13	FY14+FY15	FY16+FY17	TOTALS
Funds Allocated	0+56	55+52=107	50+54=104	50+50=100	\$367K
Actual Costs to Date	56	107	104	100	\$367K

	FY17	FY18
Quarter 1	Shared memory parallelization of FMM data structures and integration with parallel FMM	Identification of speed and parallel efficiency bottlenecks in the current version of PHAD
Quarter 2	Variable order Picard integrator with automatic step size control parallelization	Amelioration of speed and efficiency bottlenecks in PHAD
Quarter 3	Binned time step implementation in parallel	DC electron cooling initial rate estimations for different ion species (charge states) at low energy
Quarter 4	Parallel PHAD integration, benchmarking and optimizations	Low energy electron cooling initial rate estimations for different initial distributions and external fields

PHAD Overview



N_i

N_i^c

- Accurate at expense of speed.
- No approximations (exact treatment).
- Algorithms employed are the most efficient available.

Main Outcomes



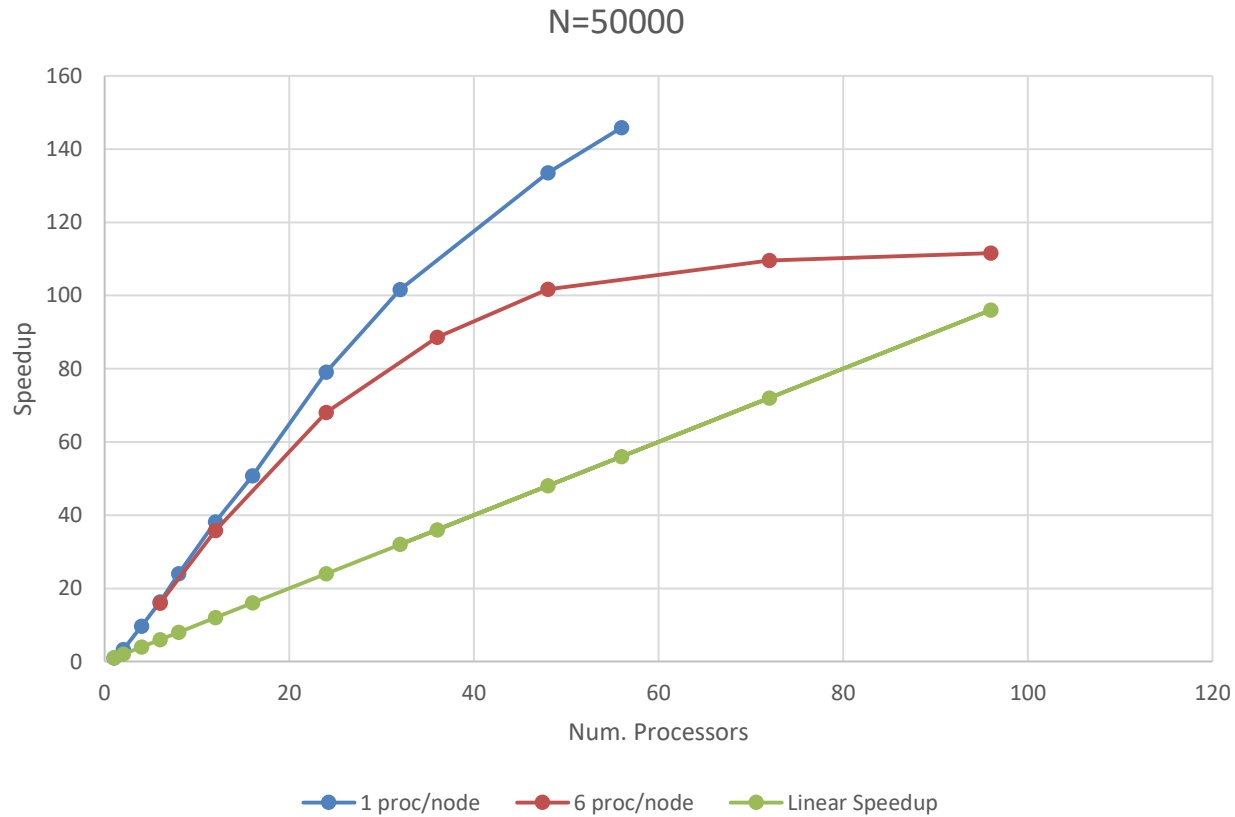
Stability & Performance

- Improved FMM.
- Improved local solver (Simo integrator).
- Memory improvements.
- Avoidance of digit cancellation.

User friendliness

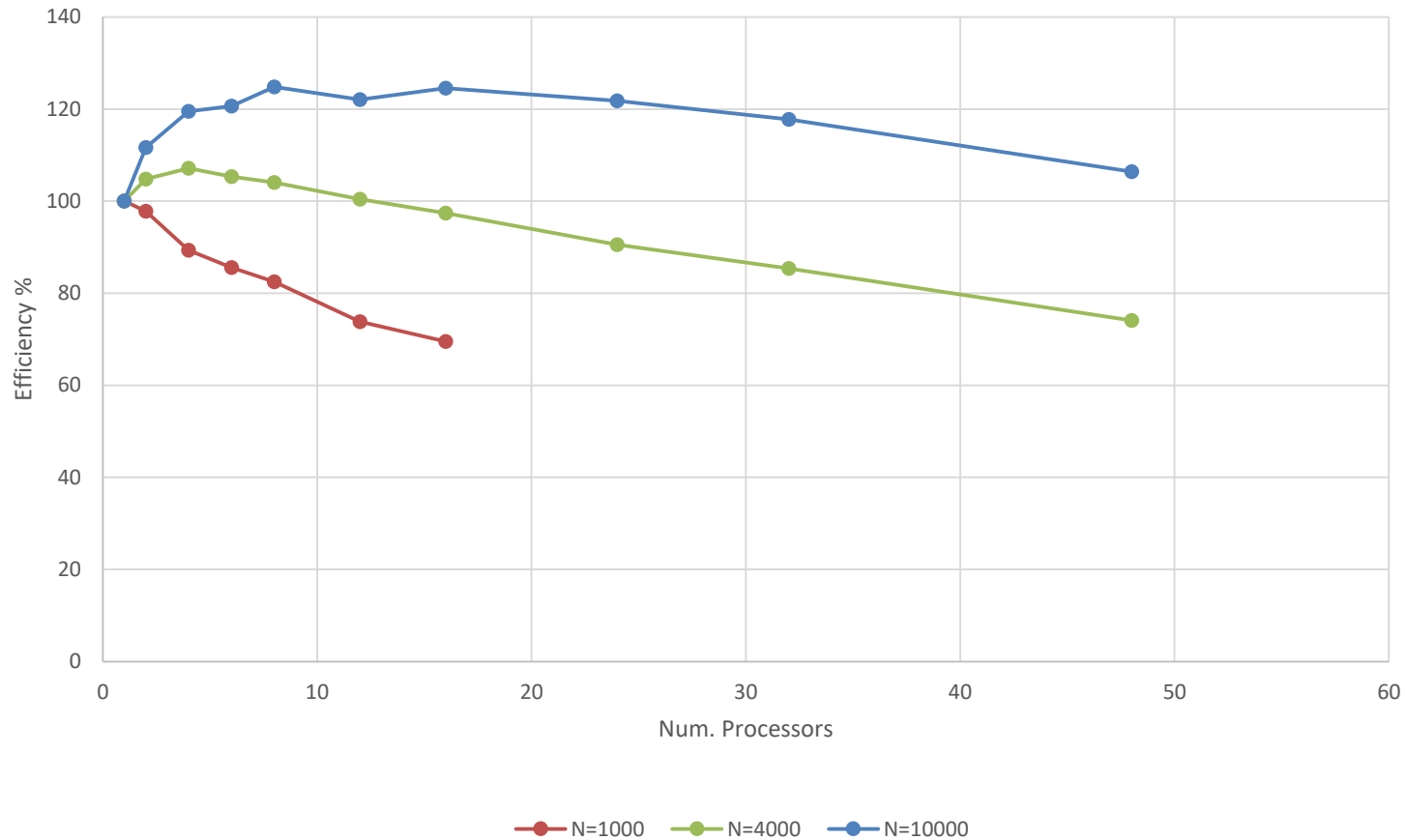
- Parameter tracking.
- Additional timers.
- Error messages.
- Relaunch capability.
- New example.

Parallel Simo Integrator



Superlinear speedup

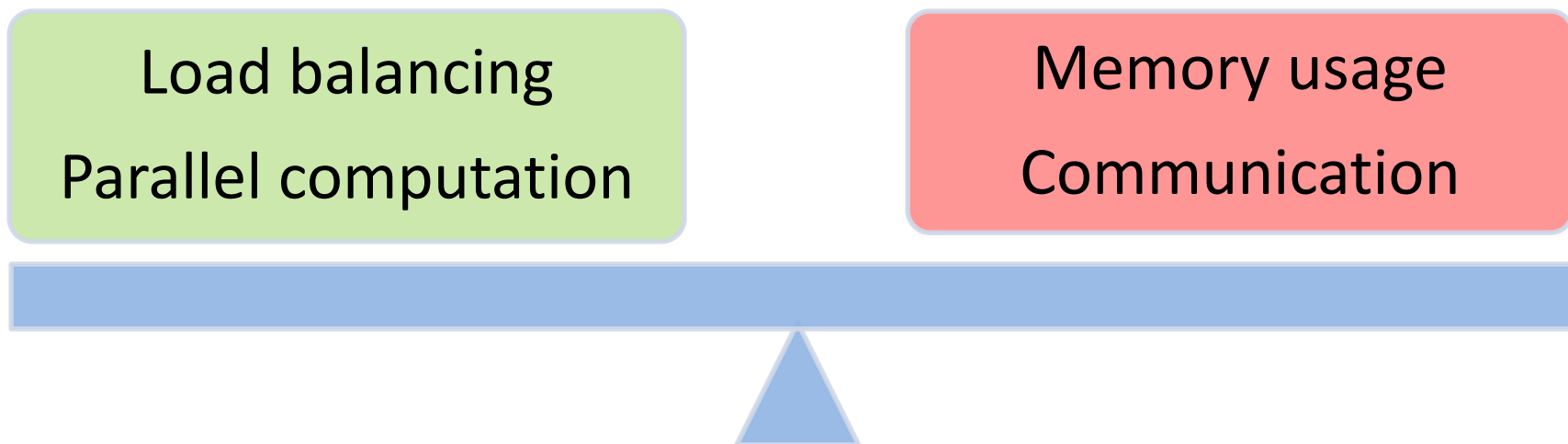
Efficiency of weak scaling



Identifying Bottlenecks



- Implemented several different parallelization strategies.
- Strategies have competing benefits and costs.



PHAD Parallelization Strategies



$$N_1 \left\{ \begin{array}{l} \text{Process 1} \\ \text{Process 2} \\ \vdots \\ \text{Process } p \end{array} \right. \quad N_2 \left\{ \begin{array}{l} \text{Process 1} \\ \text{Process 2} \\ \vdots \\ \text{Process } p \end{array} \right. \quad \dots \quad N_k \left\{ \begin{array}{l} \text{Process 1} \\ \text{Process 2} \\ \vdots \\ \text{Process } p \end{array} \right.$$

$$\left\{ \begin{array}{cccccc} N_1 & N_{p+1} & N_{2p+1} & \dots & & \text{Process 1} \\ N_2 & N_{p+2} & N_{2p+2} & \dots & \vdots & \text{Process 2} \\ \vdots & \vdots & \vdots & & N_k & \vdots \\ N_p & N_{2p} & N_{3p} & \dots & & \text{Process } p \end{array} \right.$$

$$\left\{ \begin{array}{cccccc} N_{\sigma(1)} & N_{\sigma(p+1)} & N_{\sigma(2p+1)} & \dots & & \text{Process 1} \\ N_{\sigma(2)} & N_{\sigma(p+2)} & N_{\sigma(2p+2)} & \dots & \vdots & \text{Process 2} \\ \vdots & \vdots & \vdots & & N_{\sigma(k)} & \vdots \\ N_{\sigma(p)} & N_{\sigma(2p)} & N_{\sigma(3p)} & \dots & & \text{Process } p \end{array} \right.$$

Balanced strategy



$$\left\{ \begin{array}{llll} N_{\sigma(1)} & N_{\sigma(p+1)} & N_{\sigma(2p+1)} & \cdots & \text{Process 1} \\ N_{\sigma(2)} & N_{\sigma(p+2)} & N_{\sigma(2p+2)} & \cdots & \text{Process 2} \\ \vdots & \vdots & \vdots & & \vdots \\ N_{\sigma(p)} & N_{\sigma(2p)} & N_{\sigma(3p)} & \cdots & \text{Process } p \end{array} \right.$$

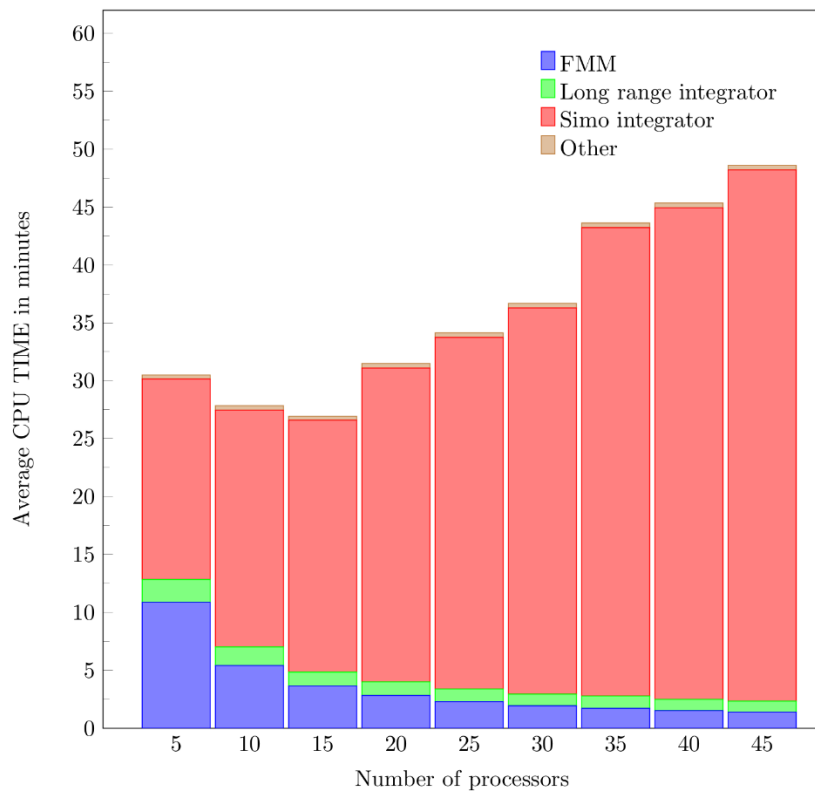
$$\left[\begin{array}{cc} |N_1| & 1 \\ |N_2| & 2 \\ \vdots & \vdots \\ |N_k| & k \end{array} \right] \xrightarrow{\text{indexed quicksort}} \left[\begin{array}{cc} |N_{\sigma(1)}| & \sigma(1) \\ |N_{\sigma(2)}| & \sigma(2) \\ \vdots & \vdots \\ |N_{\sigma(k)}| & \sigma(k) \end{array} \right]$$

$$|N_{\sigma(1)}| \leq |N_{\sigma(2)}| \leq \cdots \leq |N_{\sigma(k)}|$$

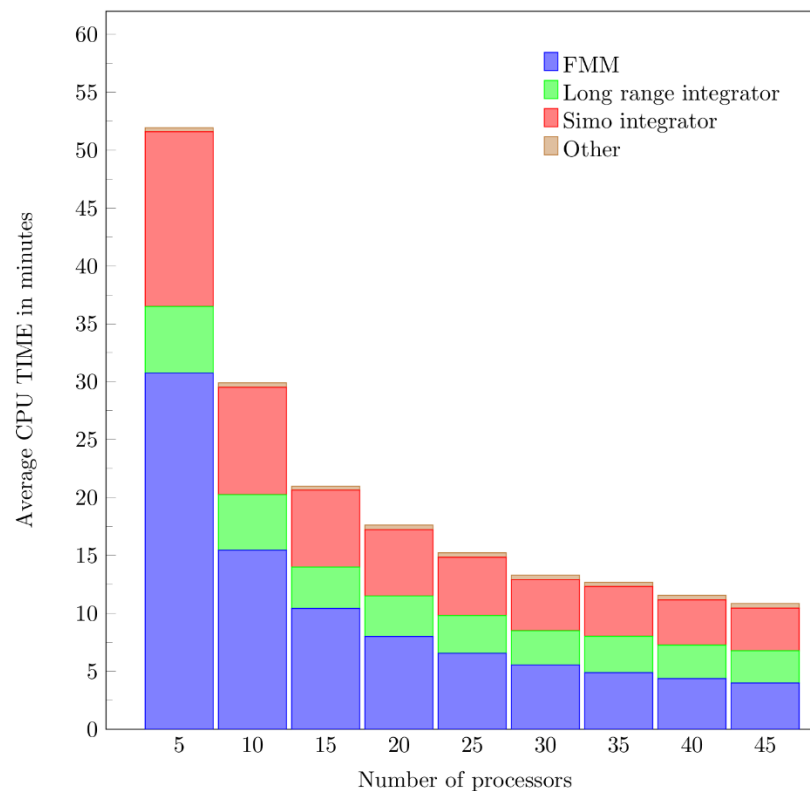
Parallel Simo Vs. Unbalanced



“Parallel Simo” scheme – 1 timestep



“Unbalanced” scheme – 5 timesteps

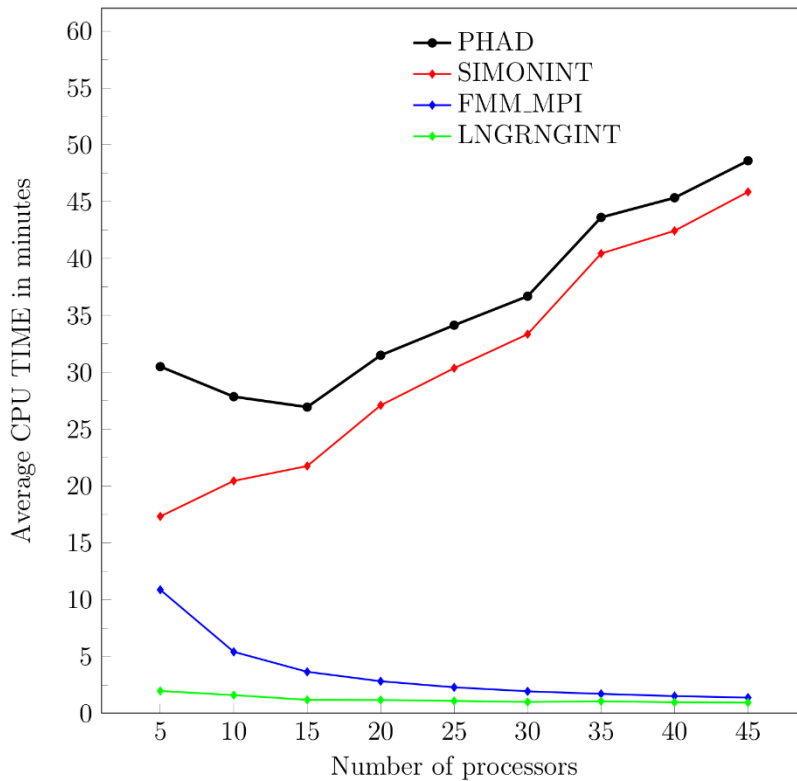


49.9K Particles, FMM order 6, $q=60$

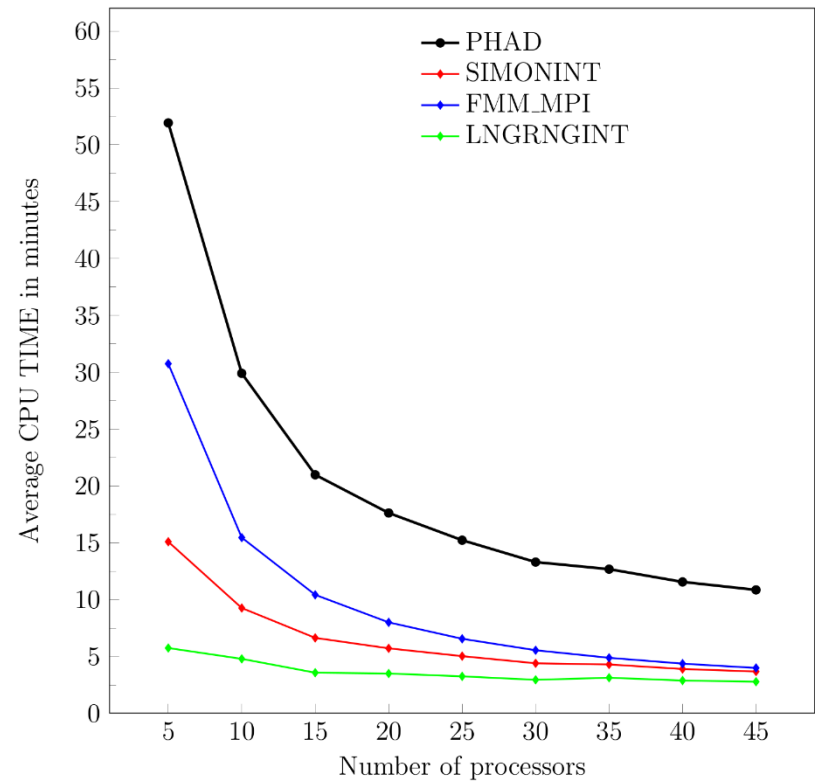
Subprocedure Breakdown



“Parallel Simo” scheme – 1 timestep



“Unbalanced” scheme – 5 timesteps

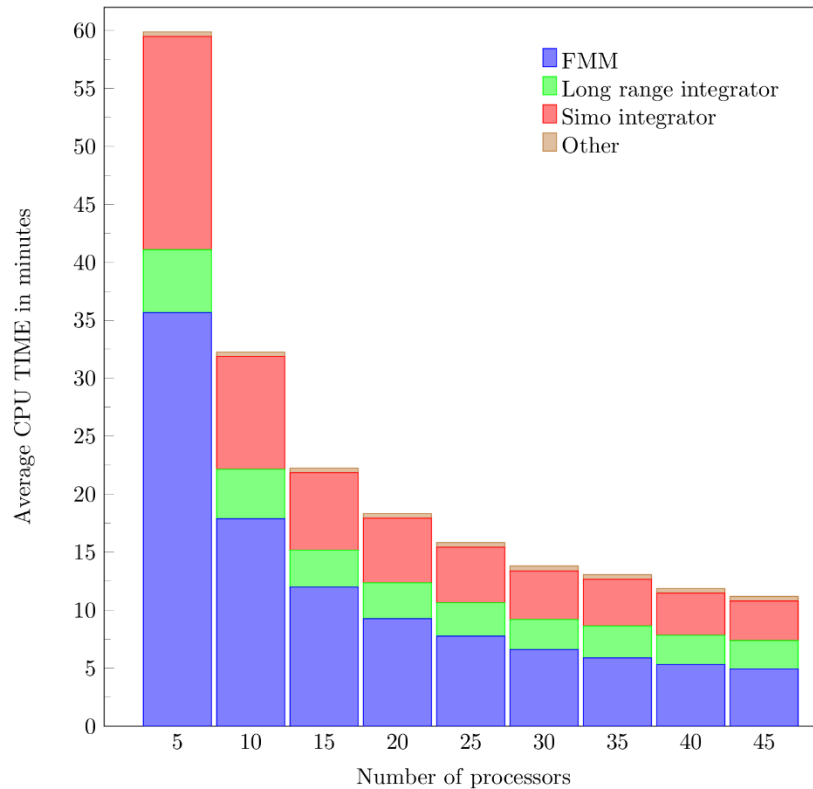


49.9K Particles, FMM order 6, $q=60$

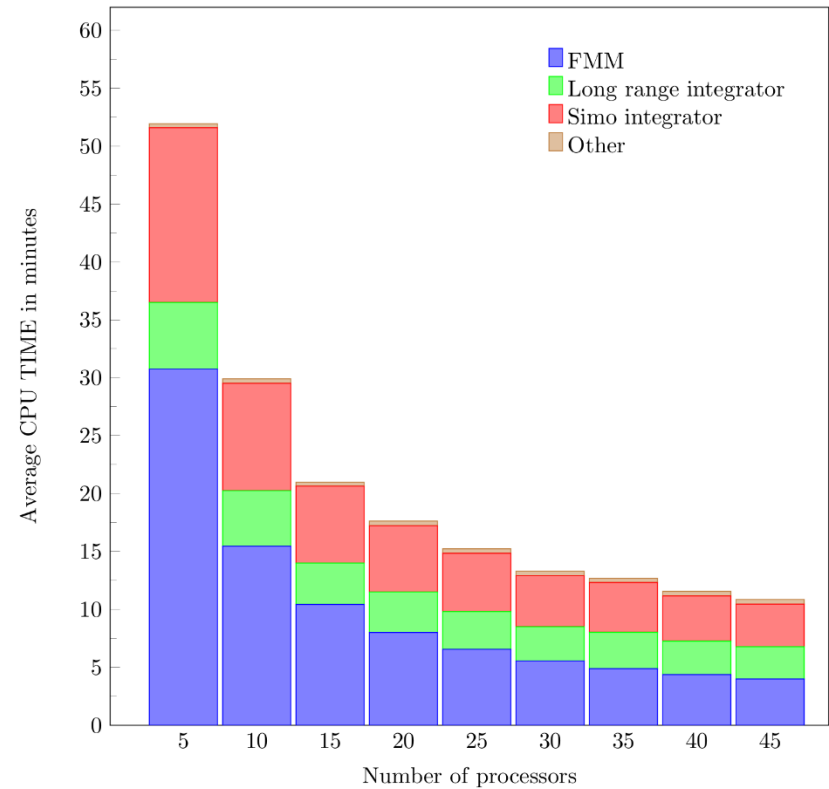
Balanced Vs. Unbalanced



“Balanced” scheme – 5 timesteps



“Unbalanced” scheme – 5 timesteps

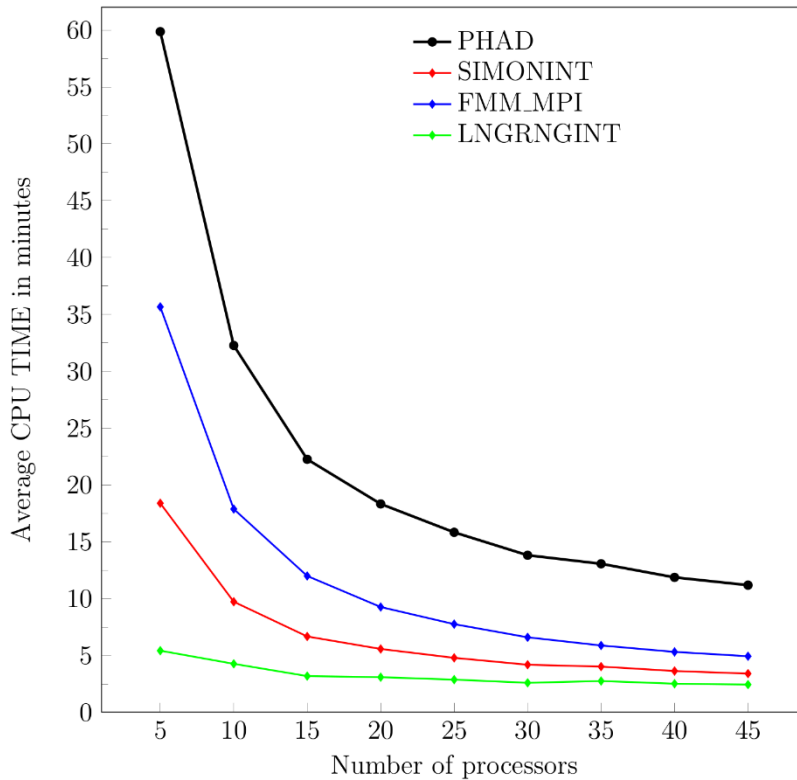


49.9K Particles, FMM order 6, q=60

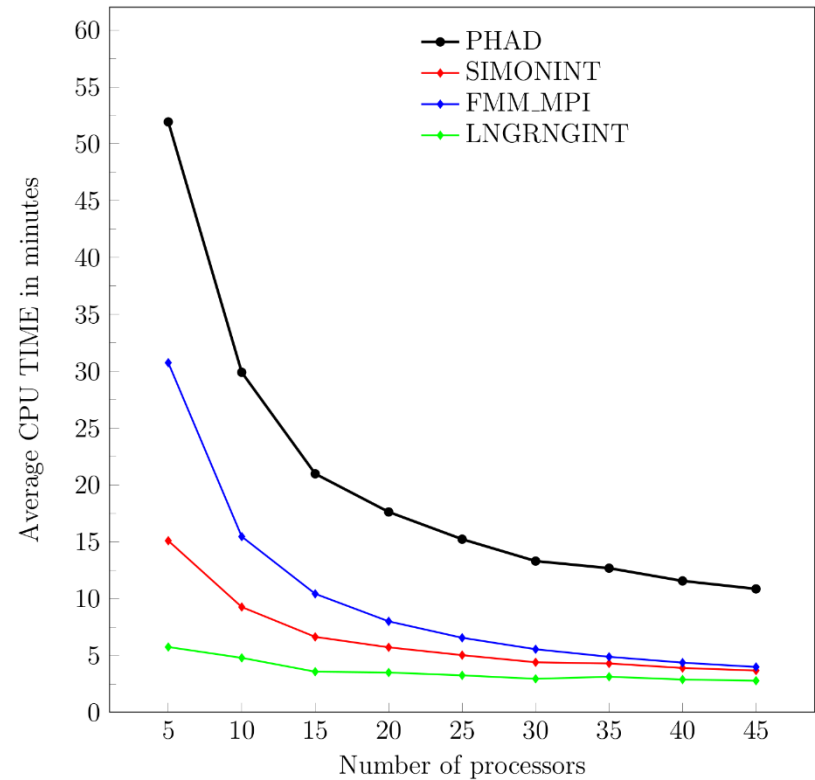
Subprocedure Breakdown



“Balanced” scheme – 5 timesteps



“Unbalanced” scheme – 5 timesteps



49.9K Particles, FMM order 6, $q=60$

Stability & Performance Improvements



- Simo integrator ensures large timesteps leading to nonphysical results are not taken.
- Relativistic gamma computed to avoid digit cancelation.
- Modification allows smaller Simo order for small PHAD timesteps.
- Several memory improvements.

Memory improvements



- FMM memory requirements do not scale with the number of processors.
- Memory for Taylor series in Simo integrator depends on q parameter instead of N .
- Memory improvement in indices storage.

Index Memory Improvements



$$\begin{array}{l}
 \text{number} \\
 \text{of} \\
 \text{neighborhoods}
 \end{array}
 \left\{ \begin{array}{cccccc}
 & \overbrace{\hspace{10em}}^q & & & & \\
 n_{11} & n_{12} & \cdots & n_{1|N_1|} & \times & \times \\
 n_{21} & n_{22} & \cdots & n_{2|N_2|} & \times & \times \\
 & & \vdots & & &
 \end{array} \right. \quad \text{2D}$$

$$n_{11} \ n_{12} \ \cdots \ n_{1|N_1|} \ n_{21} \ n_{22} \ \cdots \ n_{2|N_2|} \ n_{31} \ \cdots \quad \text{1D}$$

Bonuses: Communication efficiency.
 Larger N with same RAM.

User friendliness



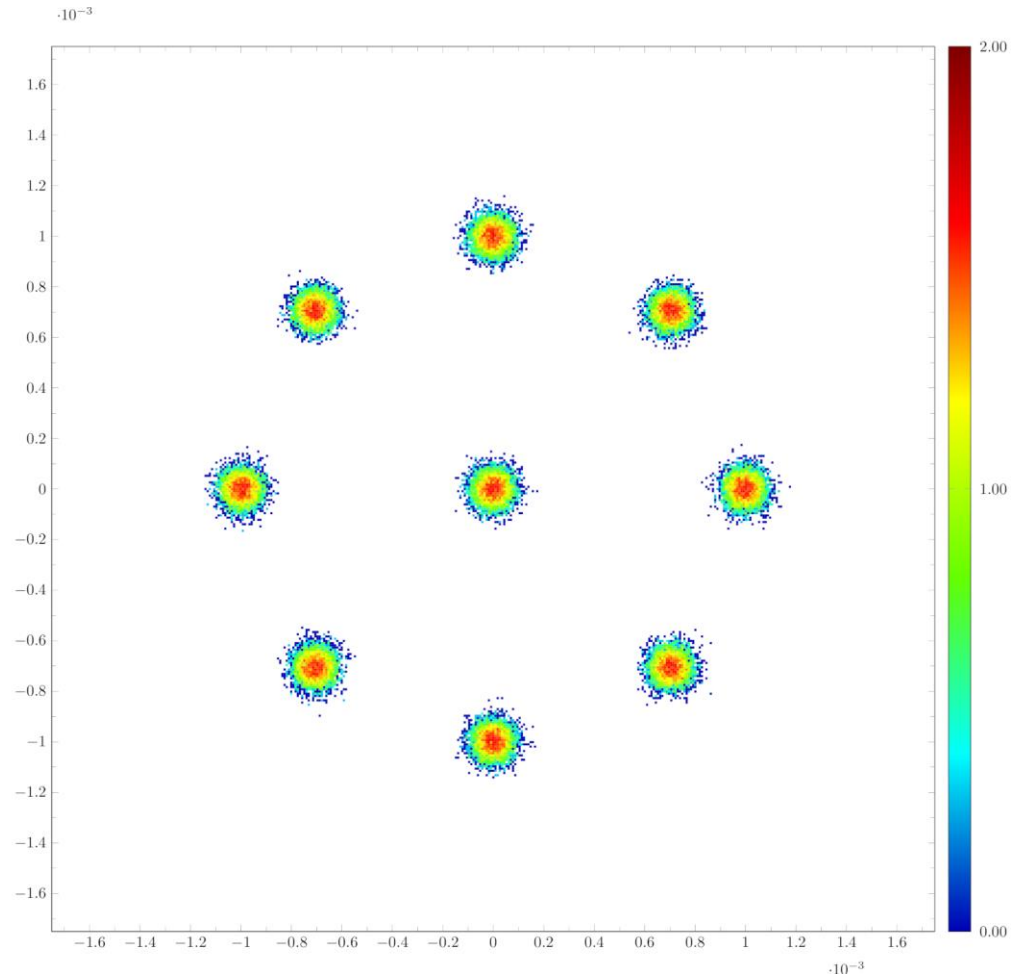
- Parameter tracking - Simo integrator largest order and minimum timestep used.
- Separate timers for read/write and some communications.
- Descriptive error messages.
- Relaunch capability.
- Website improvement (in progress).
- New example – beamlets.

Beamlets example script

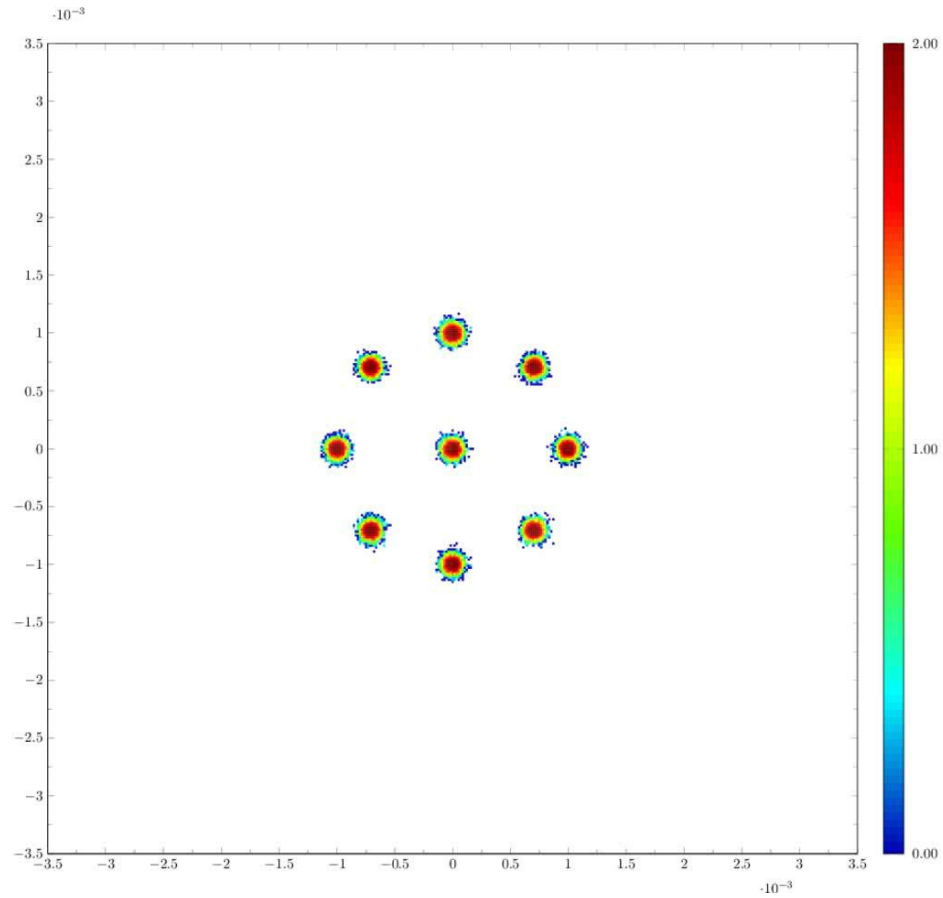


- Generates all PHAD input files.
- Positions generated using 2D Gaussian and uniform beam size.
- Momentum generated from Maxwell-Boltzmann distribution.
- Detailed documentation.

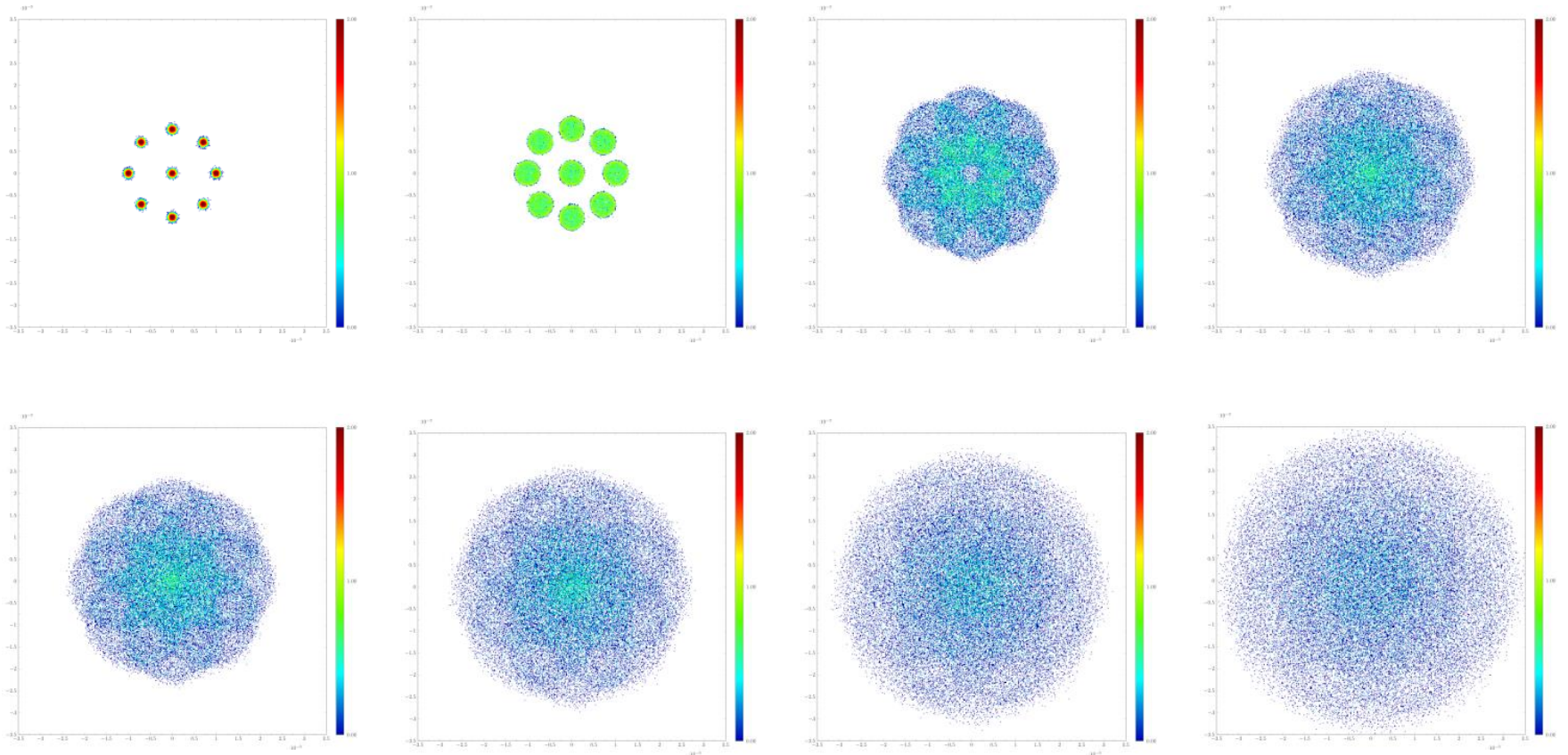
Beamlet Setup



Beamlet Simulation



Snapshots



Summary and Conclusions

- PHAD is stable.
- Improvements to the FMM have improved its performance.
- Among the three tried strategies for parallelizing PHAD, the unbalanced strategy seems to be the best.
- Overall memory requirements have been significantly reduced.
- Several user friendly capabilities have been added (Relaunch, error messages, additional timers, parameter tracking)
- Further electron cooling simulations have begun.
- Use our code; we are here to help.

<https://niu.edu/beamphysicscode/>



NIU Beam Physics Code Repository

Home

PHAD

PISCS

COMFY

N Body Simulations

3D FMM

Picard Integrator

Updates

Group Members

Contact Us

MSU COSY Infinity

NIU Physics

NIU NICADD

NIU Beam Physics and Accelerator

Research

Technical Users

DEPARTMENT OF
ENERGY | Office of
Science



Beam Physics Code Repository

This website is dedicated to sharing our group's code development results with anyone interested in charged particle beam dynamics. Most of our codes are COSY Infinity-based, the general purpose nonlinear dynamics code with an underlying arbitrary order Differential Algebraic (DA) engine. It has its own minimalist scripting language that even novice users can master quickly. Moreover, any application developed in COSY can be retrofitted with a nice Java-based custom-GUI from within the code with only a few additional commands.

Check out our projects listed in the menu to the left!

Each menu item contains a project/code that can be downloaded with a dropdown menu for easy access to its subsections. The Updates section will contain the latest news for each project. In addition to an overview page describing the code in broad lines, there are pages allowing easy downloading of the code(s), viewing its documentation/manual, and downloading examples/application running the code(s). Always, only the latest versions will be posted. You may contact us by clicking the **Contact Us** menu item.

Guiding Principles

There are a few principles that guide(d) the development of these codes:

- **Beam Physics:** although beam physics is often used synonymously with accelerator physics, we are modeling beams, their dynamics, and develop new simulation techniques for beam physics, with main applications to accelerator science, but not only
- **Computational Science:** in addition to "mere" manipulation of numbers, we use computers to handle functions, more precisely their truncated Taylor expansions, in pretty much the same way as we do with numbers (hence the DA methods), and much more
- **Accuracy and Efficiency:** what is deemed accurate and efficient is problem dependent. Given the "right" circumstances, one can break any beam dynamics code. Using any code as a black box, without understanding the underlying approximations, limitations and domain of applicability is not wise, to say the least.
- **High Performance Computing:** first, algorithmic improvements usually beat hardware improvements, but improvements in both multiply. Second, although there are pseudo-methods to this effect, strictly speaking time doesn't parallelize. Therefore, efficient time stepping at varying accuracies is paramount. Unfortunately, there is still no completely satisfactory solution to this problem; much remains to be done.
- **Modeling of particle interactions:** the preceding principles naturally lead to conclusions regarding our approach to particle interactions, namely that modeling them depends on the purpose of the applications (want to run on a laptop, only bulk transverse effects are interesting, and want a low accuracy approximate result fast, or as realistic as possible, with a gazillion particles, and want to run for a very long time, and I want to get right every minute detail of the structure). Some of our modest attempts at these issues are our **PHAD** code. In our opinion, this topic looks much better than the time