Insitu Visualization and Analysis of Ion Accelerator Simulations using Warp and VisIt

O. Ruebel, B. Loring, J. Vay, D. P. Grote, R. Lehe, S. Bulanov, H. Vincenti, W. Bethel

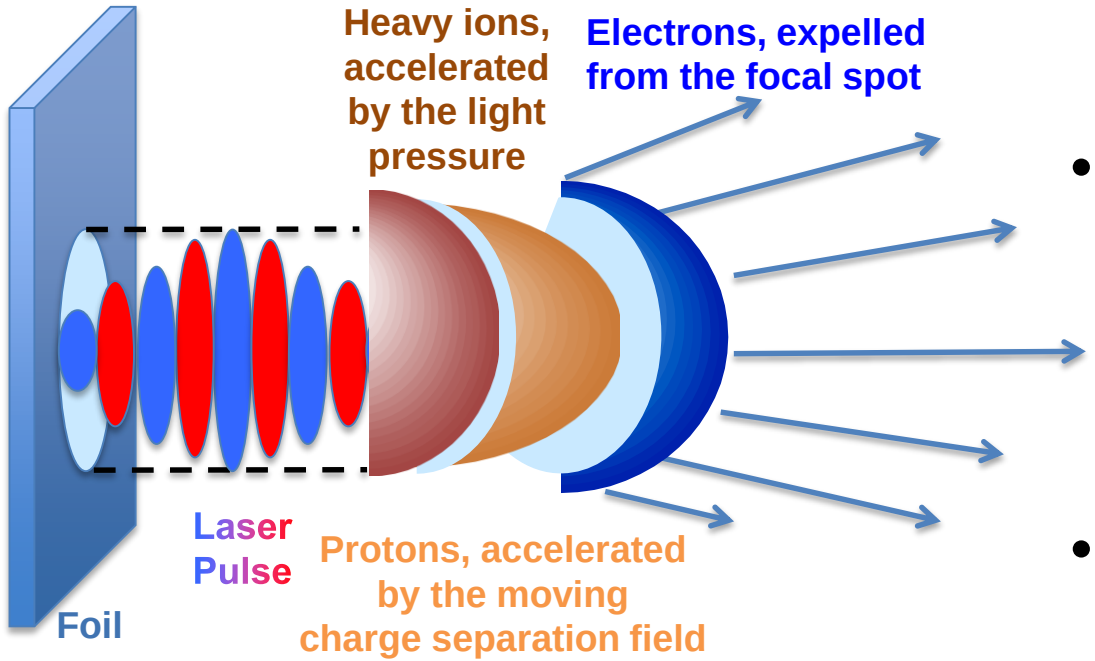
WarpIV

- a state of the art insitu vis and analysis application
- couples Warp laser plasma simulation to insitu visualization tools
- provides optimized i/o and analysis
- provides domain specific functionality, eg. filters, Yee cell support
- written in a mix of Python and C++. User only sees the Python side.
- provides a structured/uniform interface to Warp, includes a number of example.
- can be used “live” or in batch
- runs on any system from laptop to Cray SuperComputer

Why insitu? What are the challenges?

- increase temporal resolution and at the same time realize I/O reduction
- rendering, reduces problem sized data to image sized data
  - fast, small, serial I/O, doesn’t impact file system.
  - but hard to get it right without a lot fiddling and fine tuning
- “extracts”, derived quantities, reduced geometry etc
  - focused analysis, do exactly what you need, write exactly what you need
  - often smaller, but not always
  - different I/O properties than simulation itself
  - render it later, so you can get the best result

Emerging Science: Ion beam acceleration



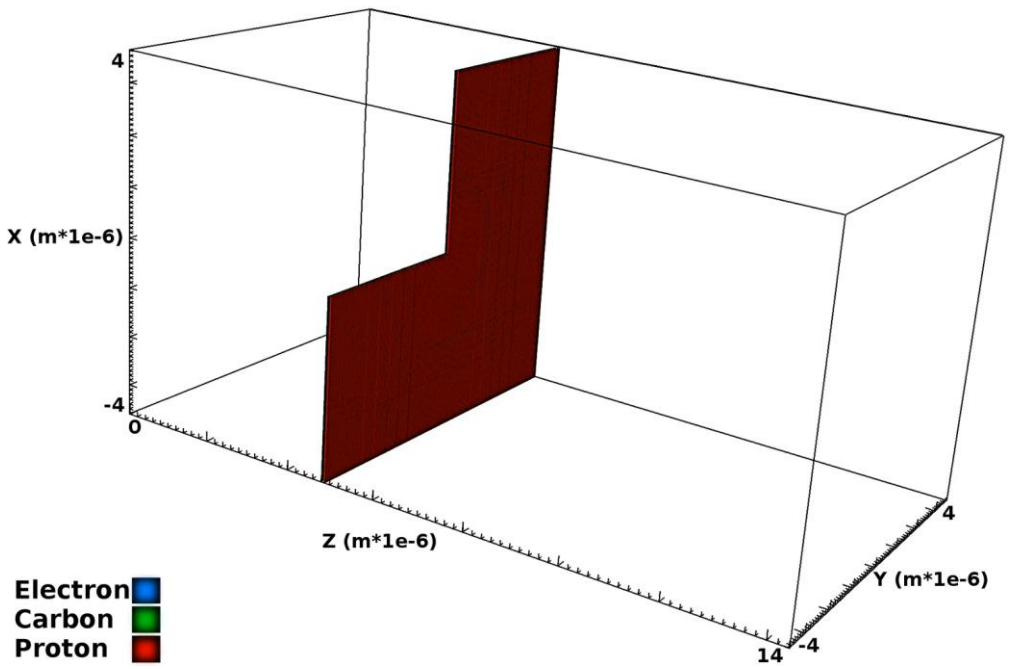
simple idea: use a laser to create a plasma. charge separation accelerates ionized hydrogen

- can be used in many applications!
  - including cancer therapy
- potential advantages cancer therapy
  - precise control over depth at which energy is deposited results in less tissue damage
  - relatively compact and cheap
- An active area of research, simulations are used to develop and prove theory and will eventually model experimental apparatus

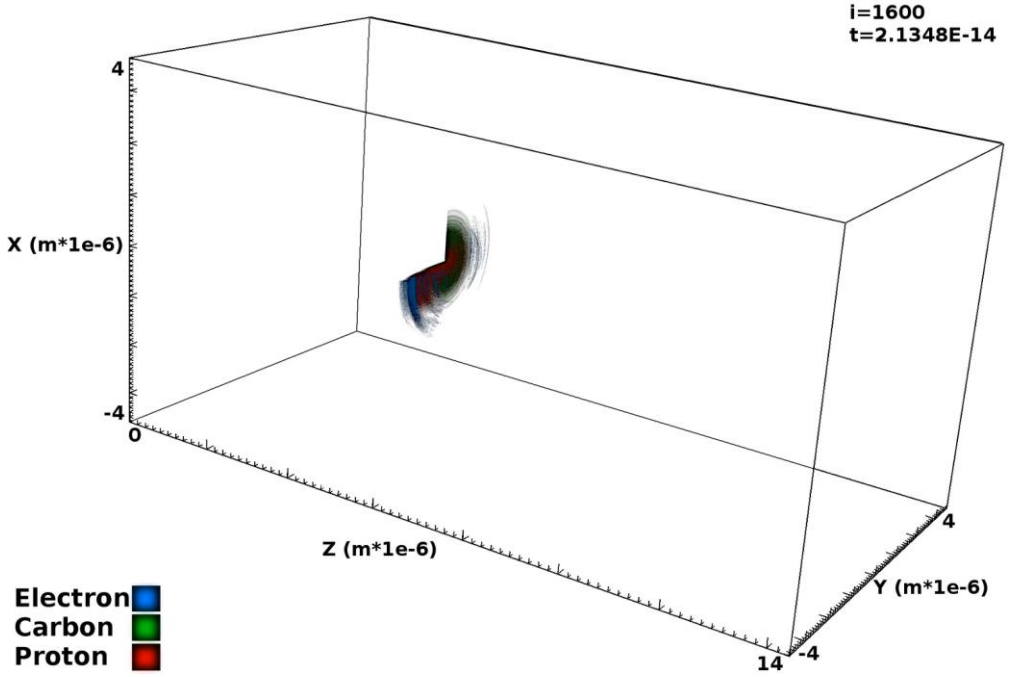
*S. S. Bulanov, et al. "**Accelerating monoenergetic protons from ultrathin foils by flat-top laser pulses in the directed-coulomb-explosion regime**," Phys. Rev. E, vol. 78, p. 026412, Aug 2008.*

Visualization: Iso-surfaces

Particle Density



Kinetic Energy



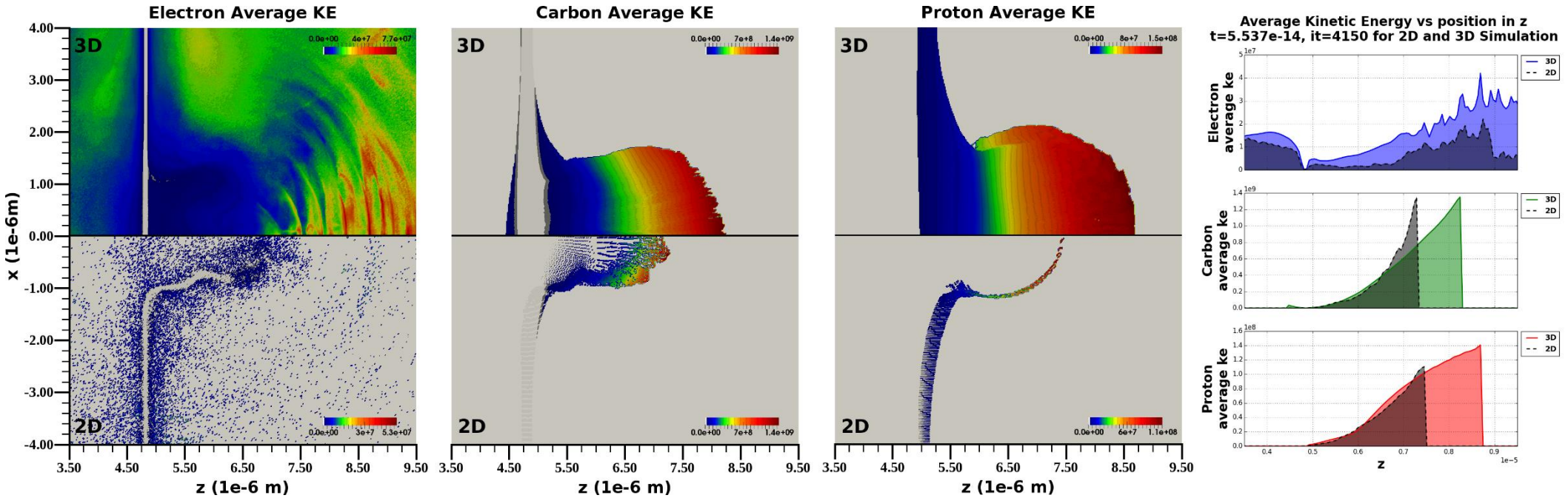
Focus: Cost Benefit of 3D Simulation

- 2D simulations are often used in theoretical development
  - computational costs are low
  - 2D datasets are small
  - easy to analyze on a workstation using familiar tools
- 3D simulations more accurately model the situation
  - very computationally expensive
  - very large datasets are produced
  - analysis is challenging, need parallel vis tools, and supercomputing

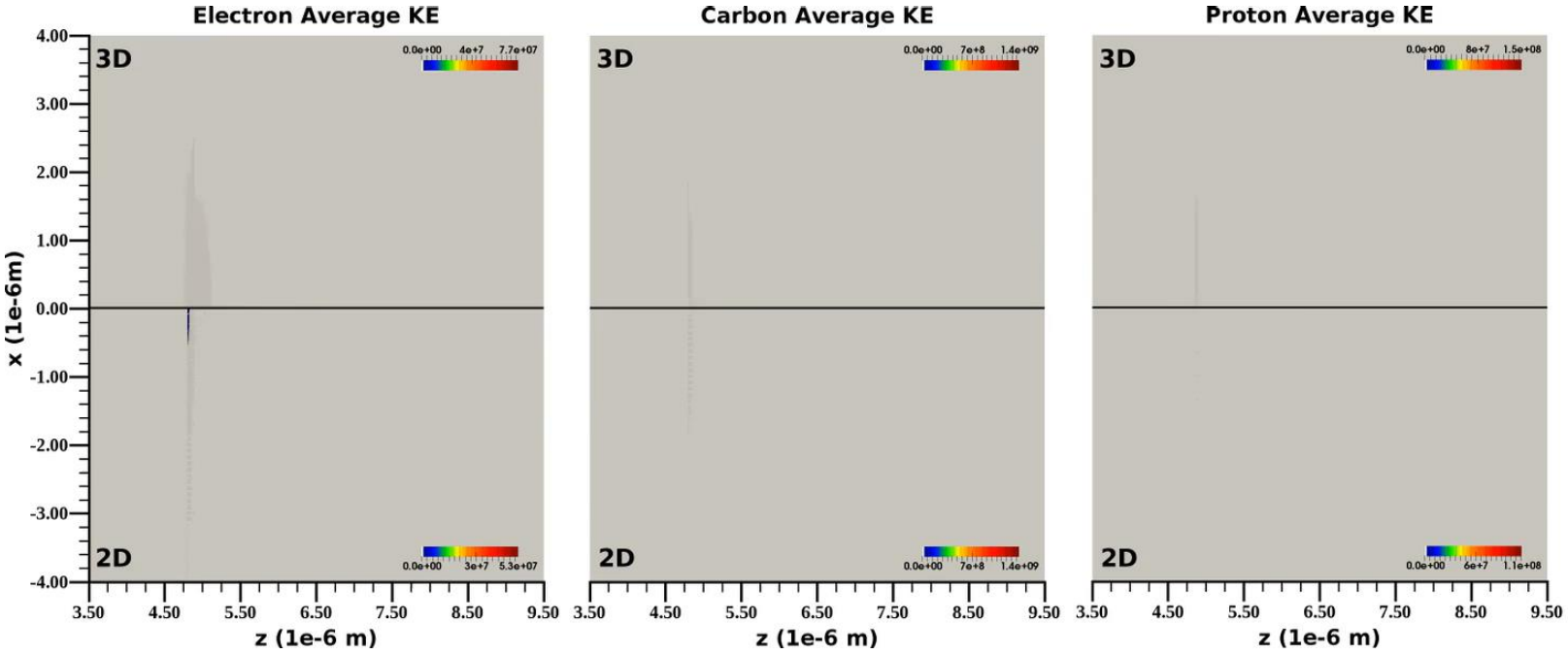
Are 2D simulations of the ion beam accelerator good enough?

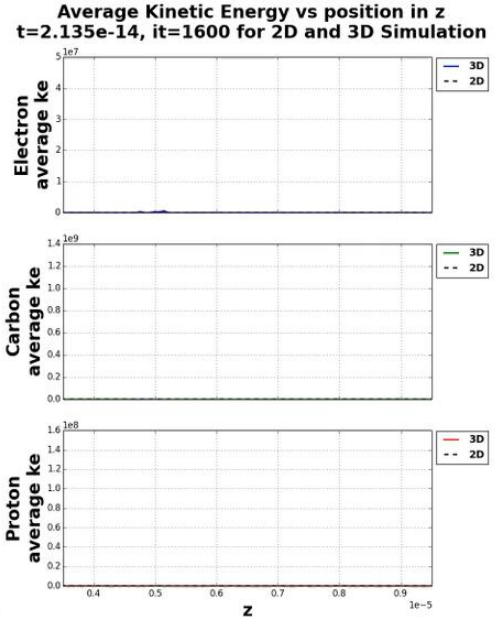
*O. Rubel et al, "**In situ Visualization and Analysis of Ion Accelerator Simulations using Warp and VisIt**", IEEE CG&A, SPECIAL ISSUE ON HIGH PERFORMANCE VISUALIZATION AND ANALYSIS, SEPTEMBER 2015 (submitted)*

Are 2D simulations good enough?

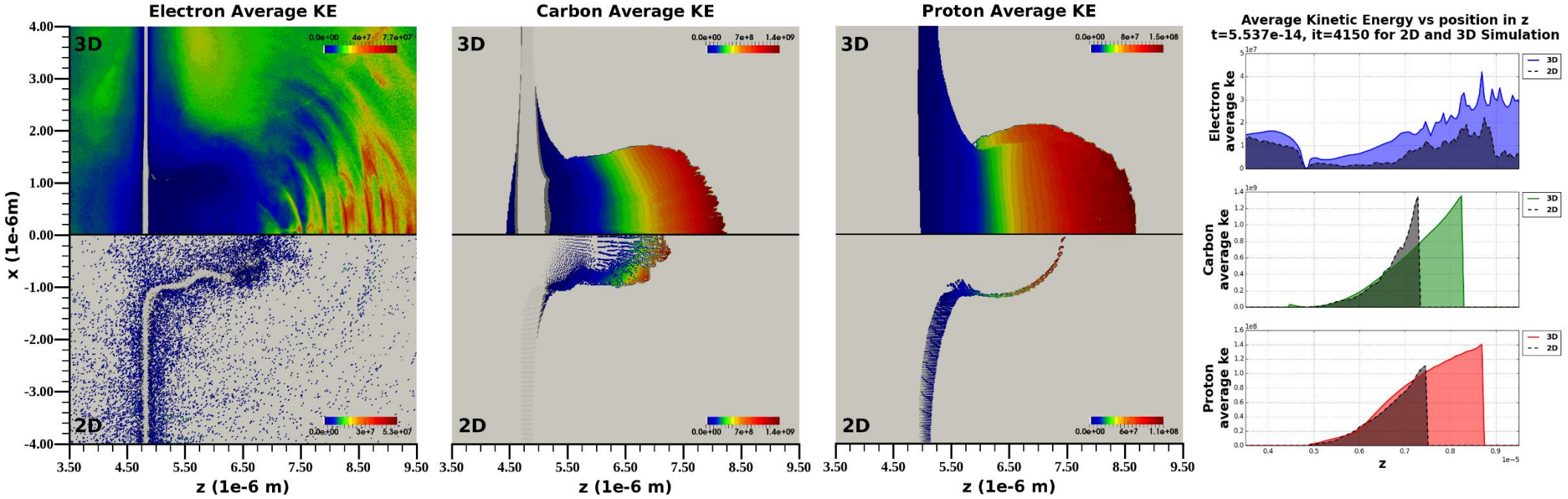


Analysis: Compare 3D and 2D simulation



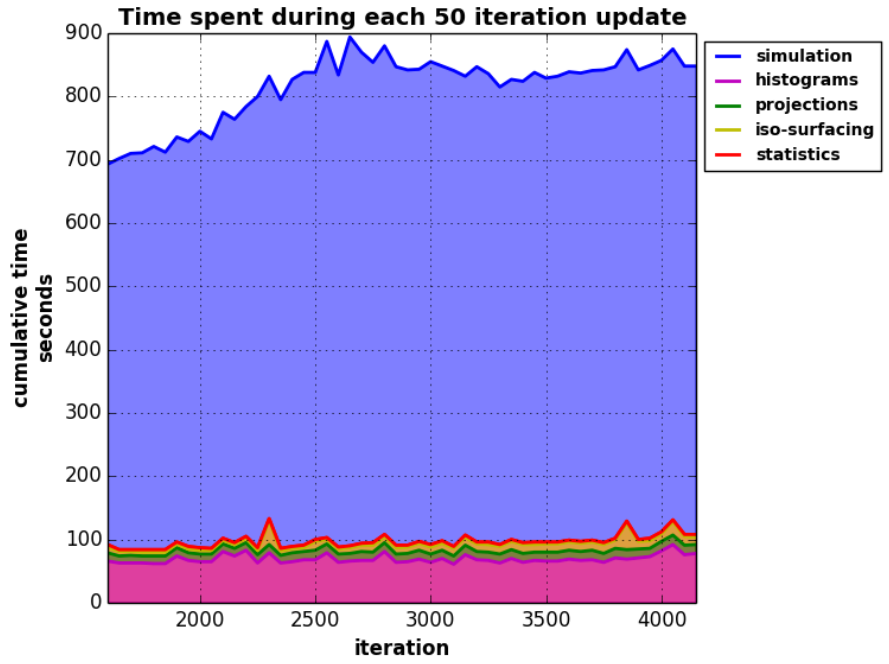


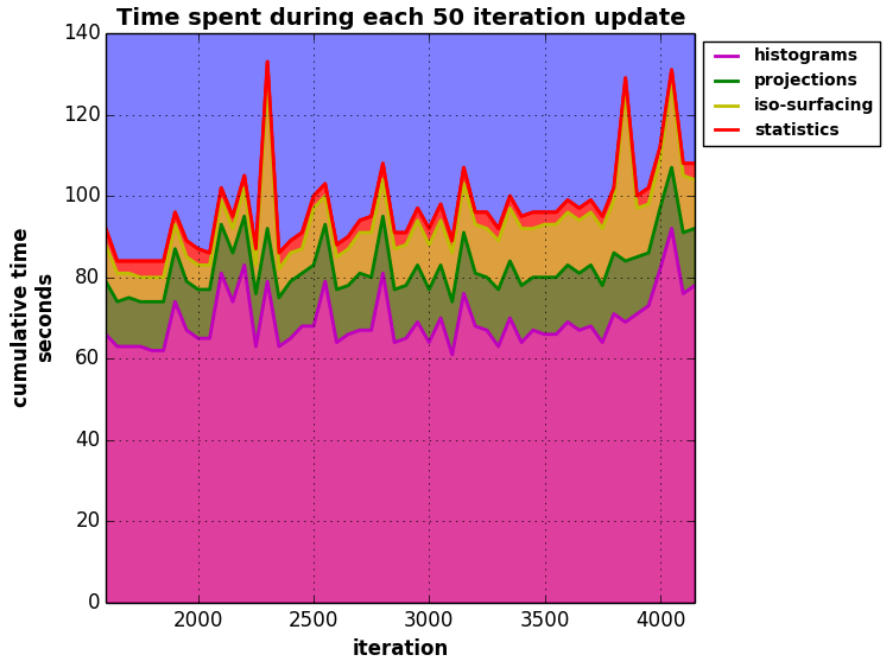
Are 2D simulations good enough?



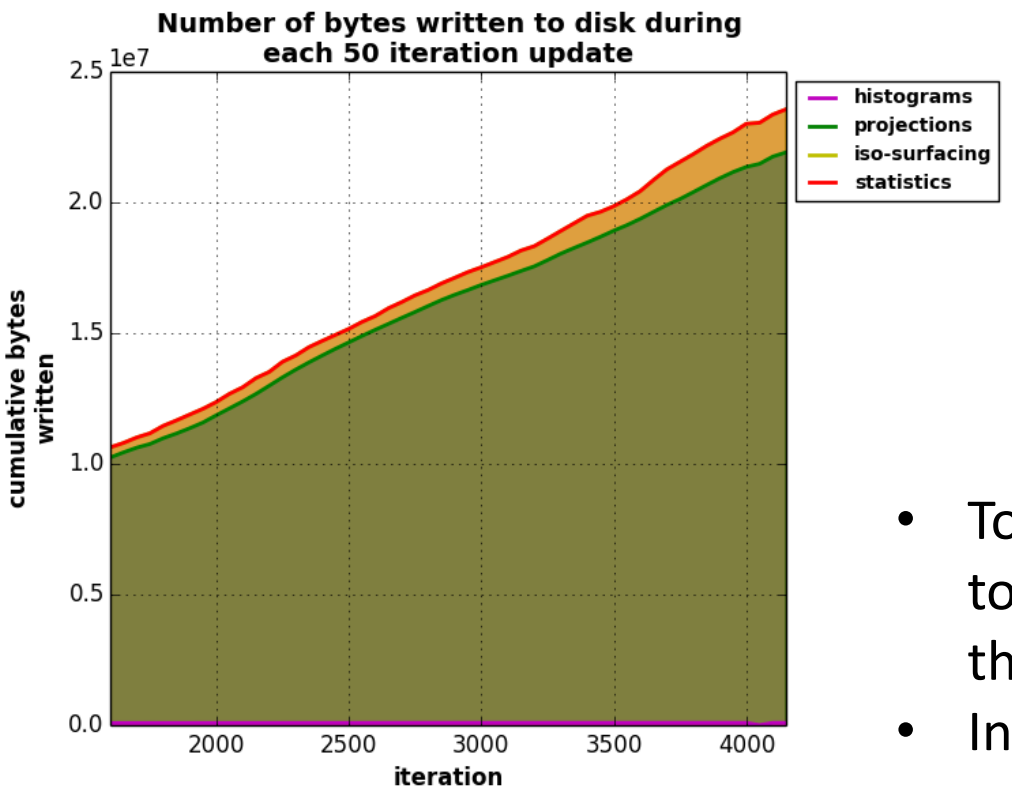
- 2D is qualitatively similar, but in 3D higher energies are reached, the beam propagates faster and further over the same time period.

The Big Question: How did we do?





What about I/O?



Category	Actually written to disk
histograms	4.48984 MB
projections	795.969 MB
iso-surfaces	40.7662 MB
statistics	3484 b
total	**841.228 MB**

- To do this post process we would have written **3.2 TB** to disk to capture the **minimum** amount of data to do the same analysis.
- Insitu bought us a **4033** times reduction in I/O.

Rendering performance is critical insitu

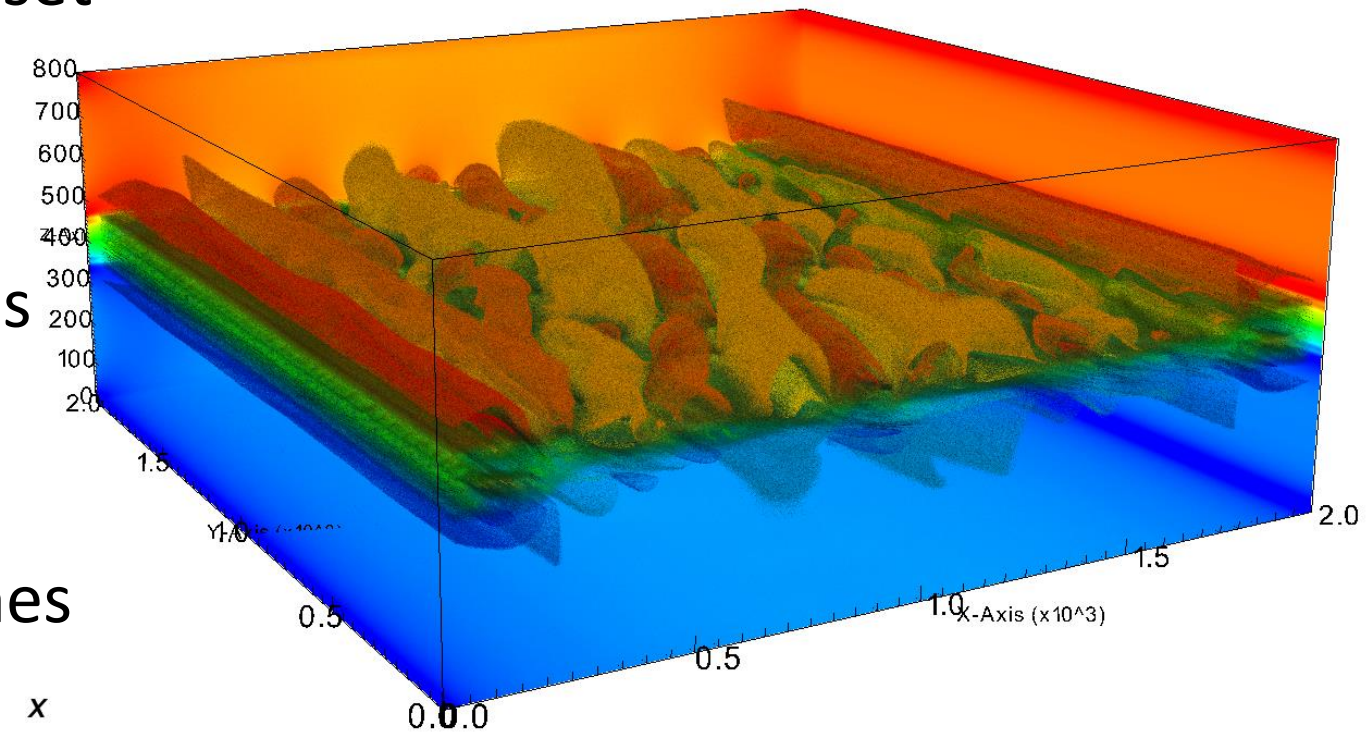
- Rendering is challenging insitu! Many parameters to fine tune to get a nice looking(usable) result.
  - it’s hard to fine tune rendering in advance
  - you are essentially working blind
- Rendering has some of the biggest opportunity for data reduction while capturing 3D nature of the data
- With insitu rendering you may need to render more to ensure you get the result
  - need to look at more fields
  - maybe from different angles

Rendering improvements for VisIt & libsim

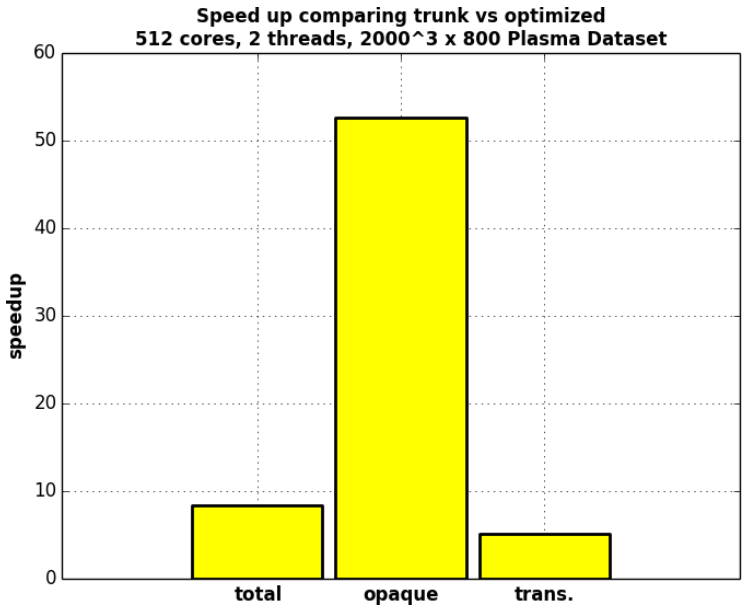
- Alpha compositing capability, and improved depth compositing
  - SIMD vectorization, threads overlap comm and composite
- Ordered compositing optimization, eliminates the need for global redistribution of data. lets you render in place.
- Optimized camera order depth sort in VTK
  - transform qsort to std::sort lets comparisons be inlined
  - Use GetCellPoints instead of GetCell. minimal overhead to compute cell depth, and eliminates calls through VTK's virtual API
  - SIMD vectorization, templatization

Testing the improvement

- 2000^2 x 800 plasma dataset
- 512 cores on Edison
- 3 opaque slices
- 20 transparent iso-surfaces
- data is rotated in ½ deg increments
- average time over 20 frames



The result



Data rendered, Plasma		
type	**desc.**	**zones**
opaque	3 slice	7190403
trans.	20 iso	464815961

Speedup vs trunk	
total	8.34
opaque	52.65
transparent	5.12

Render and Composite Time				
code	**configuration**	**total**	**trans.**	**opaque**
opt.	global sort	21.0305	20.6666	0.354182
	ordered comp.	4.55346	4.06768	0.326197
trunk	global sort	37.9864	20.8121	17.1737

Conclusion

- With post process analysis if you could get your images out in your work day it was *“fast enough”*
- With insitu analysis we are under intense scrutiny from the science community
- If this is going to be viable we have to have ***efficient*, *fast* and *reliable* code**