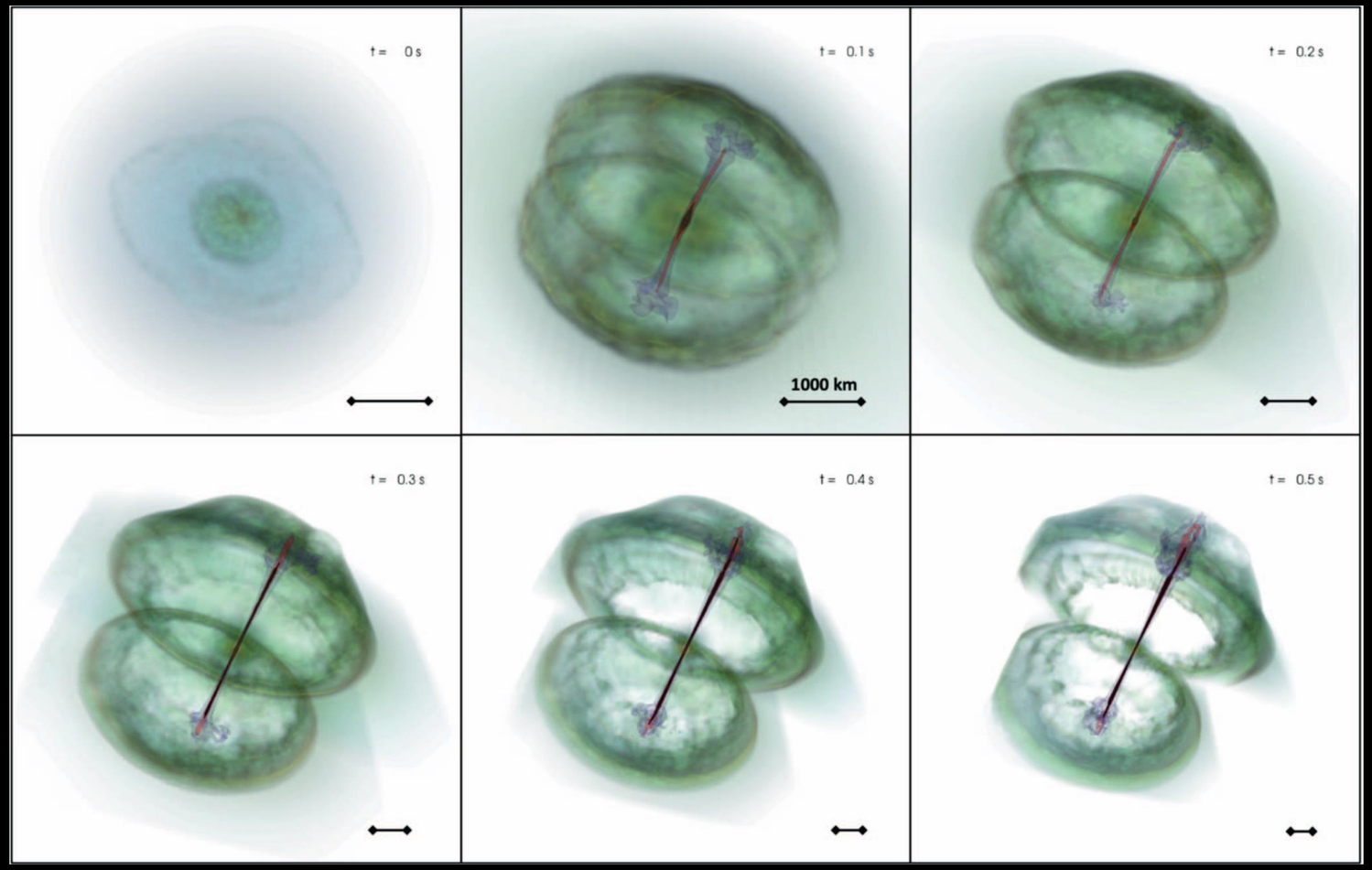
Efficient Utilization of Resources in GRB Simulations at Multiple Scales

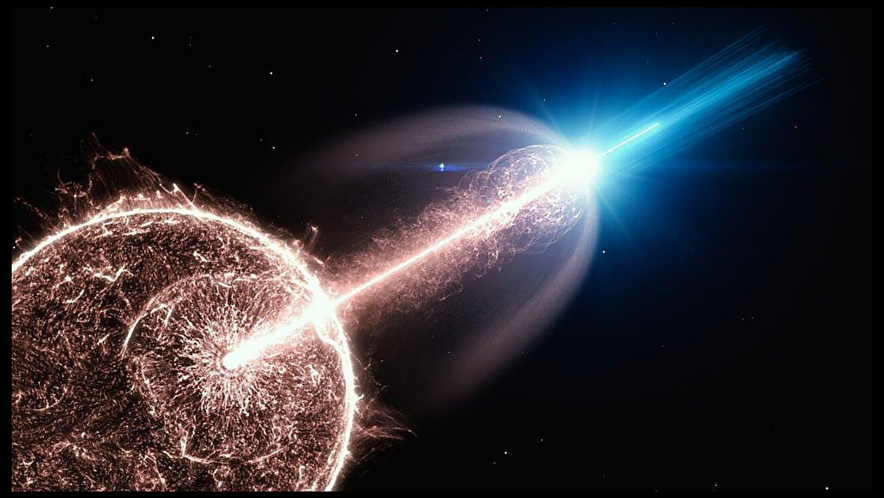


Nathan Walker

Lazzati, Davide, Rosalba Perna, Riccardo Ciolfi, Bruno Giacomazzo, Diego López-Cámara, and Brian Morsony. 2021. "Two Steps Forward and One Step Sideways: The Propagation of Relativistic Jets in Realistic Binary Neutron Star Merger Ejecta." The Astrophysical Journal Letters 918 (1): L6. https://doi.org/10.3847/2041-8213/ac1794.

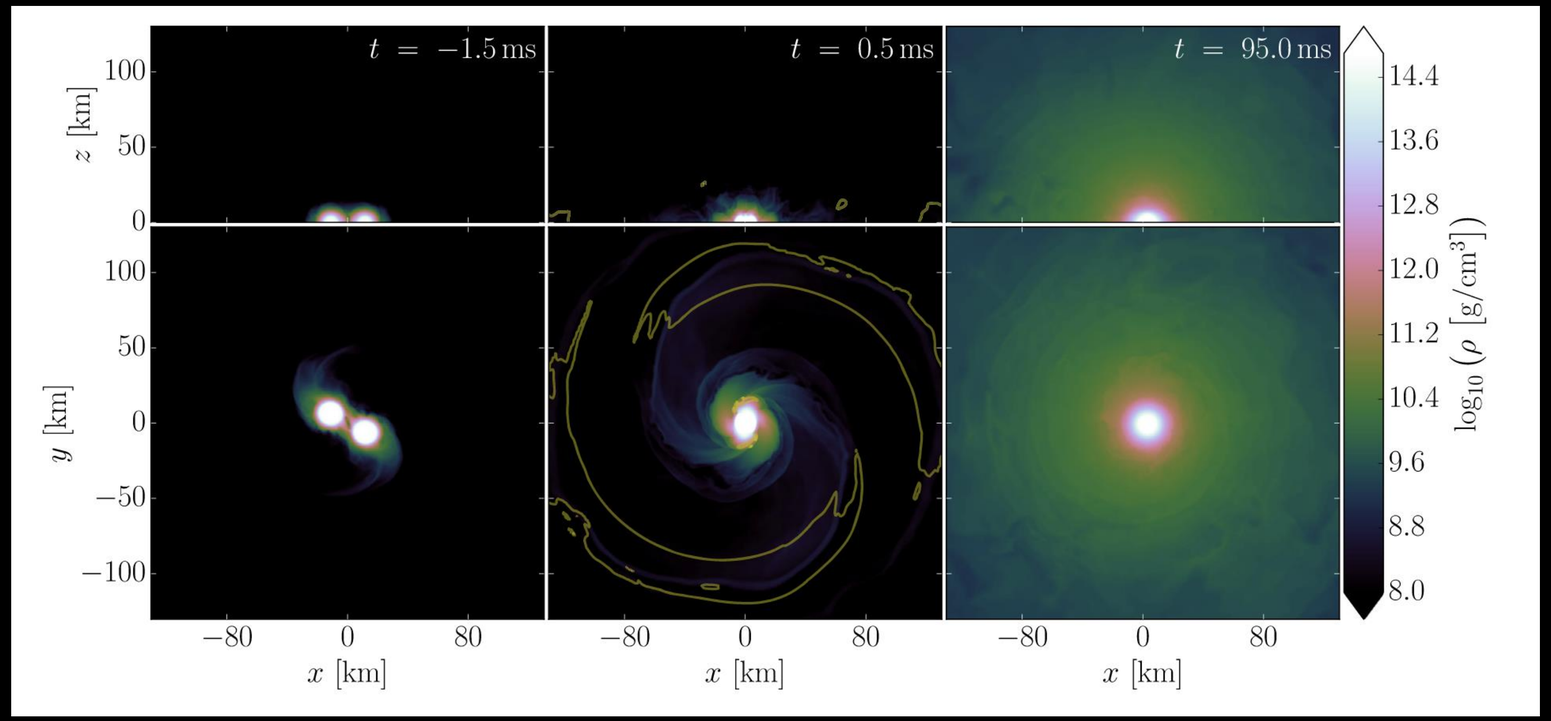
Gamma-Ray Bursts

- First Discovered in the late 60s
- The most energetic explosions ever observed
- Elusive mystery:
  - Progenitors?
  - Cosmological Origin?
  - Emission Mechanism?
- Two classes:
  - LGRBs - Supernovae
  - SGRBs - Compact Object Mergers

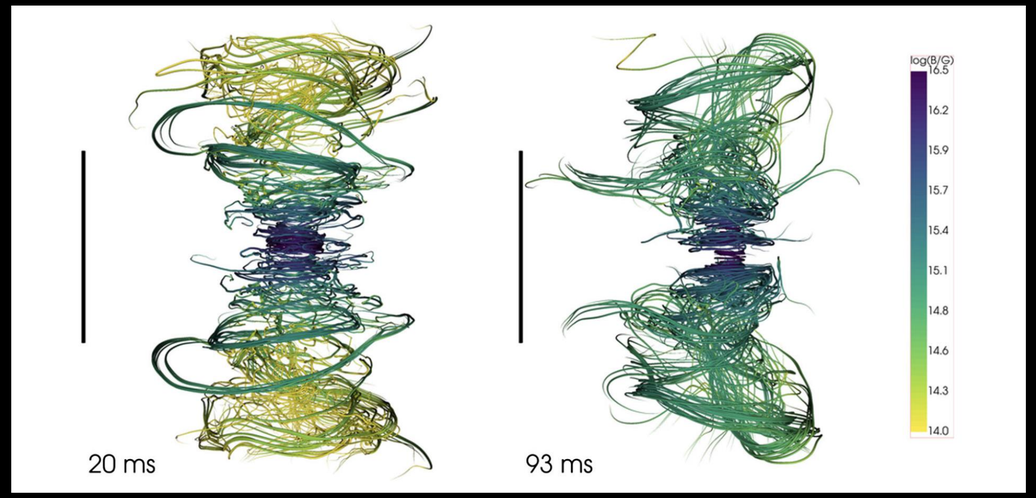


Compact Mergers

Ciolfi et al, 2019. doi:10.1103/PhysRevD.100.023005

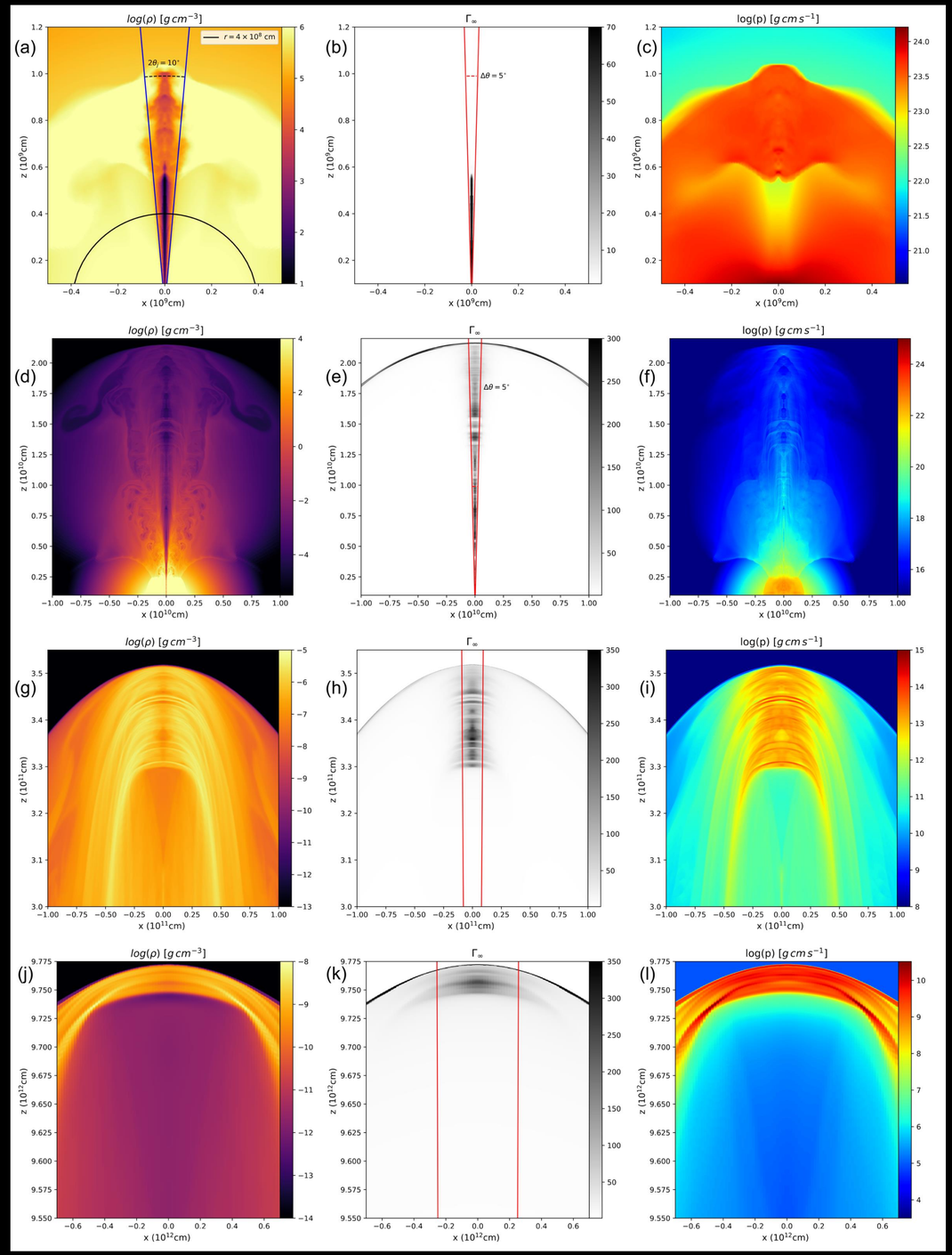


- GRMHD Simulation - Binary Neutron Star Merger (Ciolfi et al, 2019)
  - Neutron star remnant collapses into a black hole
- Inject a relativistic jet into merger ejecta



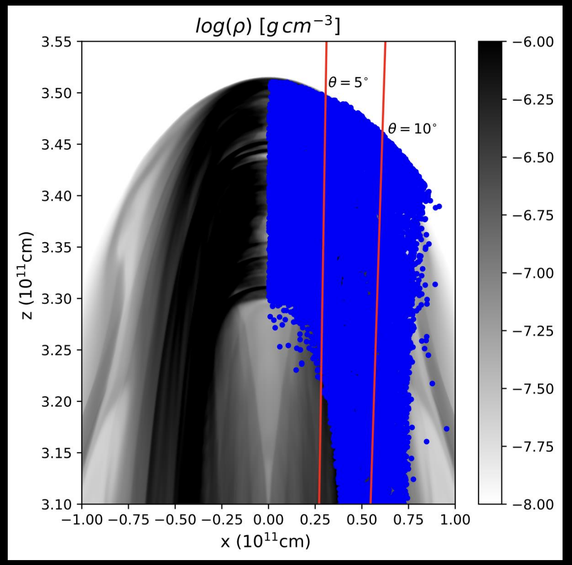
Hydrodynamics - Jet Injection and Evolution

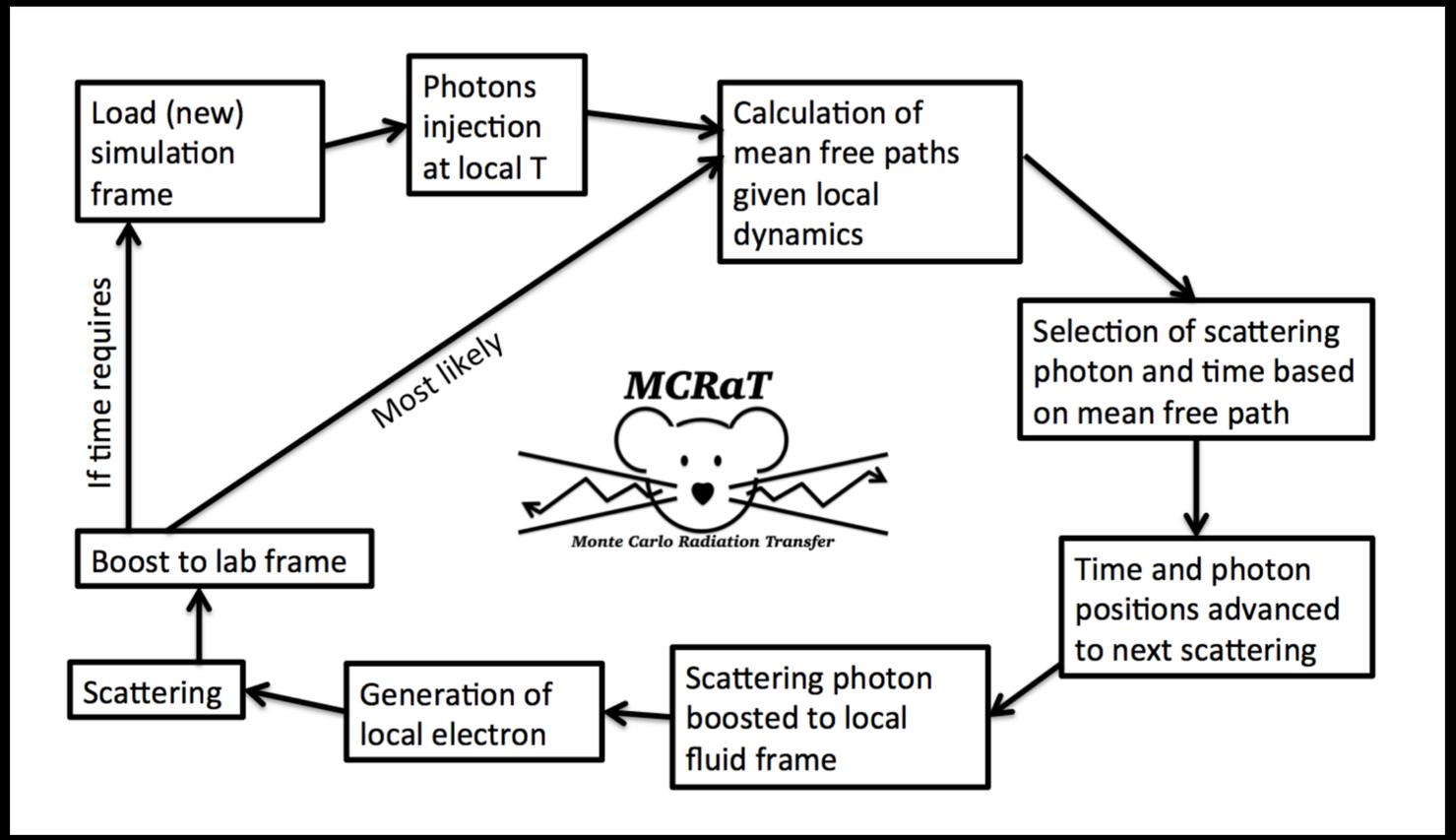
- PLUTO hydrodynamical simulation
  - 2D spherical coordinates
  - Static grid
  - Jet injection lasts ~ 2 s
- The jet becomes highly structured (radially and angularly), and becomes a pancake near the photosphere



Radiation Transfer

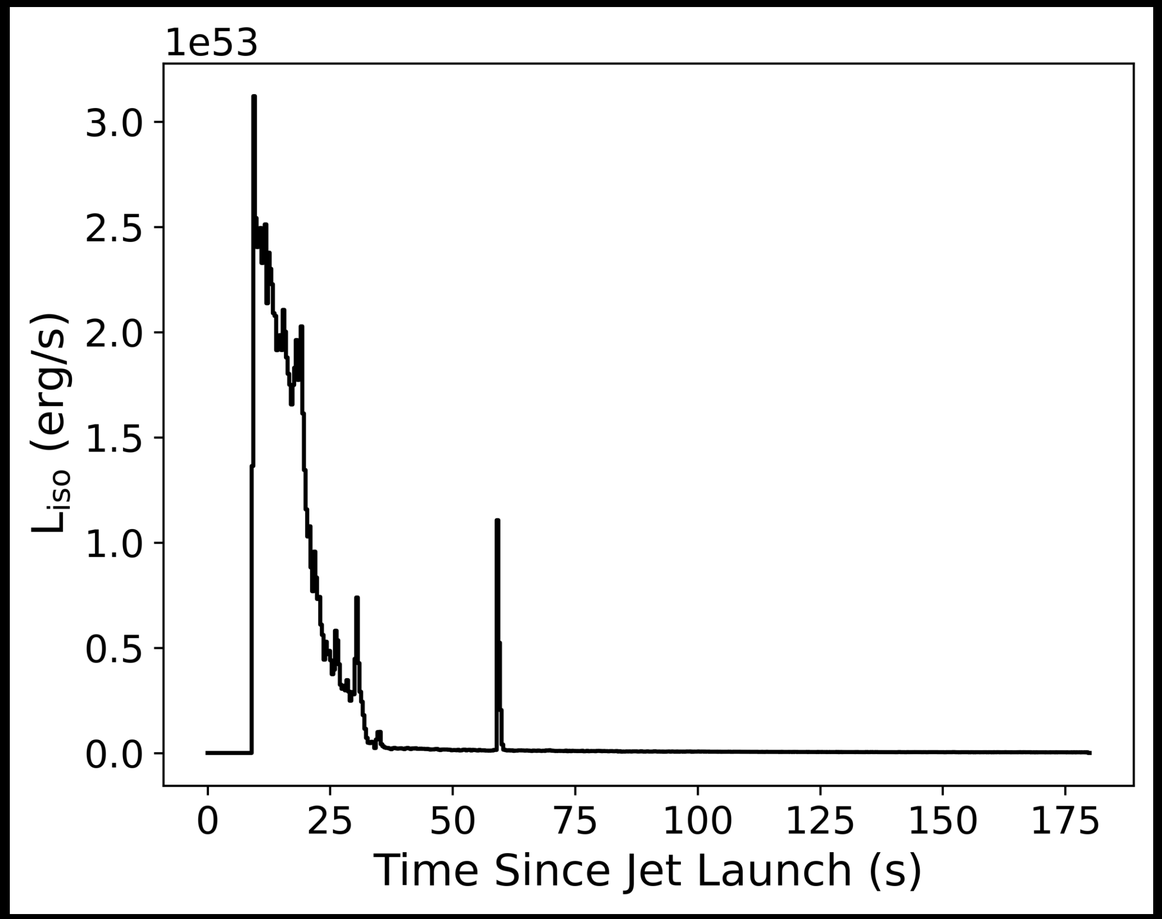
- Takes hydro frames as input
- Injects and scatters photons
  - Compton scatters photons with fluid (including Klein-Nishina)



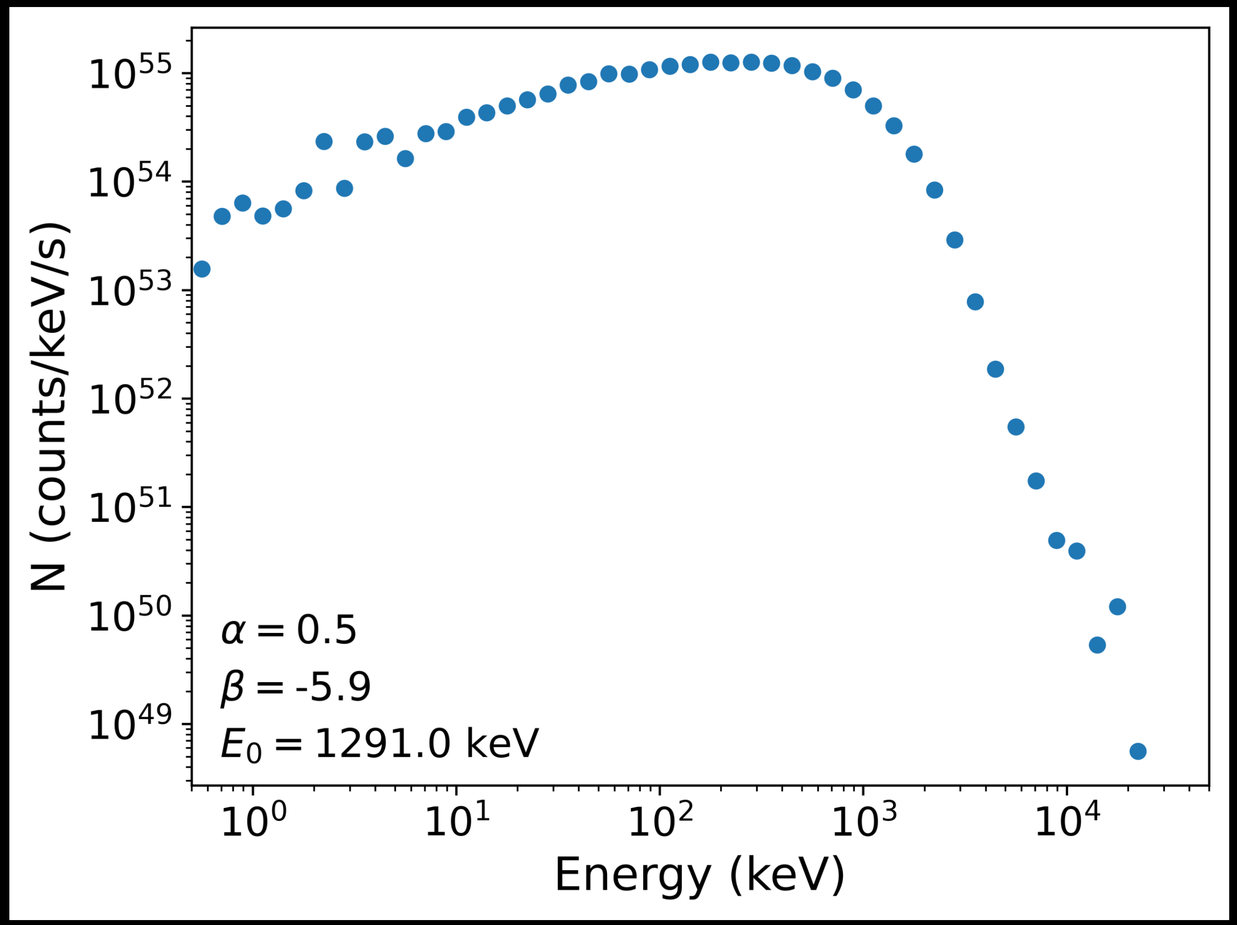


GRB Observables

Lightcurve



Spectrum



Modeling GRBs Involves Physics at Multiple Scales

- Stellar Evolution/Compact Mergers
- Compact Object Formation and Jet Launching
- Jet Propagation
- Radiation and Afterglow

How do we do all of this efficiently?

How do these scale?

Numerical Hydrodynamics

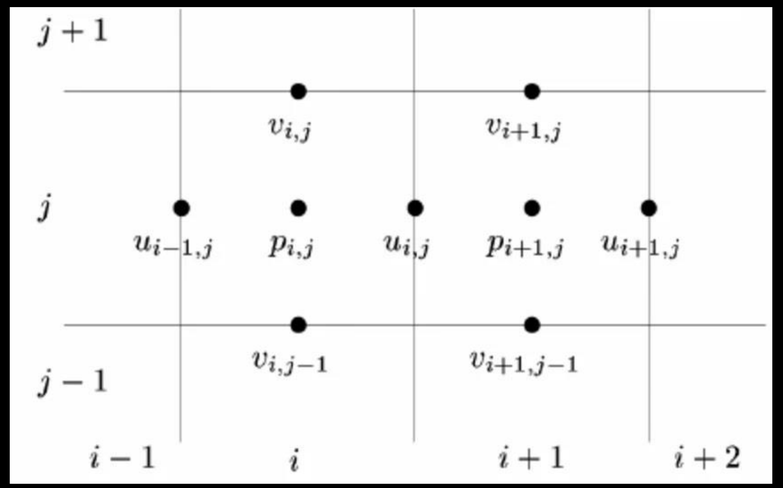
- Solve hydrodynamics on a discretized grid

$$\partial_t \mathbf{Q}(x,t) + \partial_x \mathbf{F}(\mathbf{Q}(x,t)) = \mathbf{S}(\mathbf{Q}(x,t))$$

$$\mathbf{Q} = \begin{pmatrix} q_1 \\ q_2 \\ \dots \\ q_m \end{pmatrix}; \mathbf{F}(\mathbf{Q}) = \begin{pmatrix} f_1 \\ f_2 \\ \dots \\ f_m \end{pmatrix}; \mathbf{S}(\mathbf{Q}) = \begin{pmatrix} s_1 \\ s_2 \\ \dots \\ s_m \end{pmatrix}$$

$$\mathbf{Q}^{\mathrm{new}} = \mathbf{Q}^{\mathrm{old}} - \frac{\Delta t}{\Delta x}\left(\mathbf{F}_{\mathrm{right}} - \mathbf{F}_{\mathrm{left}}\right) + \Delta t \mathbf{S}_{\mathrm{vol}}$$

- Each $\mathbf{Q}^{\mathrm{new}}$ can be calculated independently
  - Each time-step can be vectorized

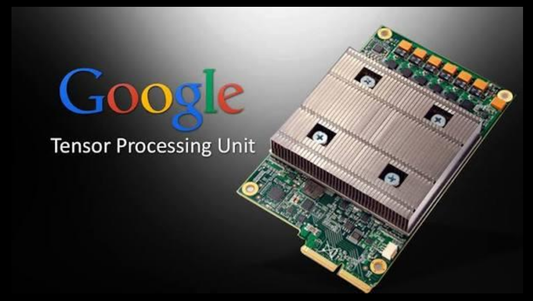


Hardware Acceleration

Run Computations on Specialized Hardware

- Cut up the grid and do calculations in parallel
  - TPU
    - Tensor operations
  - GPU
    - SIMD shaders
  - NPU
    - Matrix operations
  - CPU
    - Regular computer stuff

Simulation Data



Effectively Using Resources

Scaling Up Simulations

Personal Computers

Apple Silicon

intel

nVidia

AMD

metal

cuda

mlx

mpi

openmp

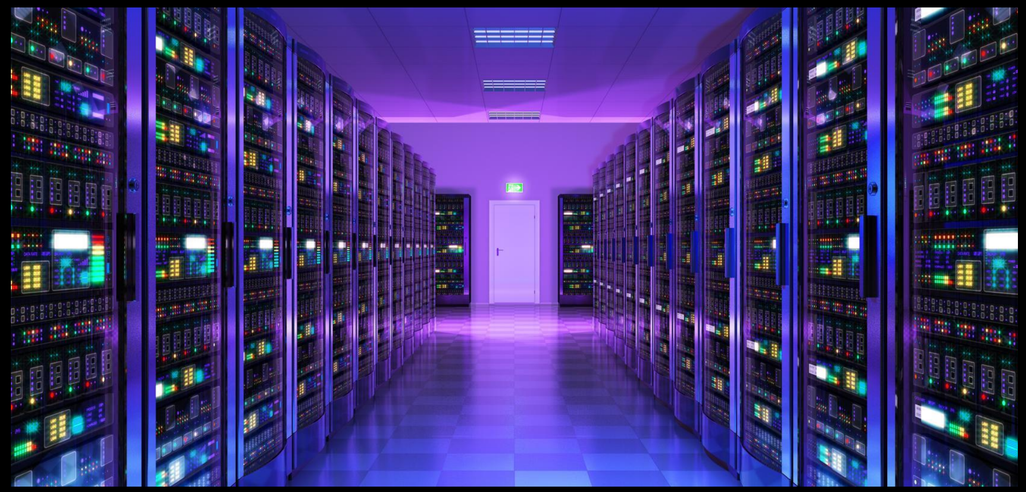
jax

openacc

opengl

opencl

Supercomputers and Clusters



CPU

intel

nVidia

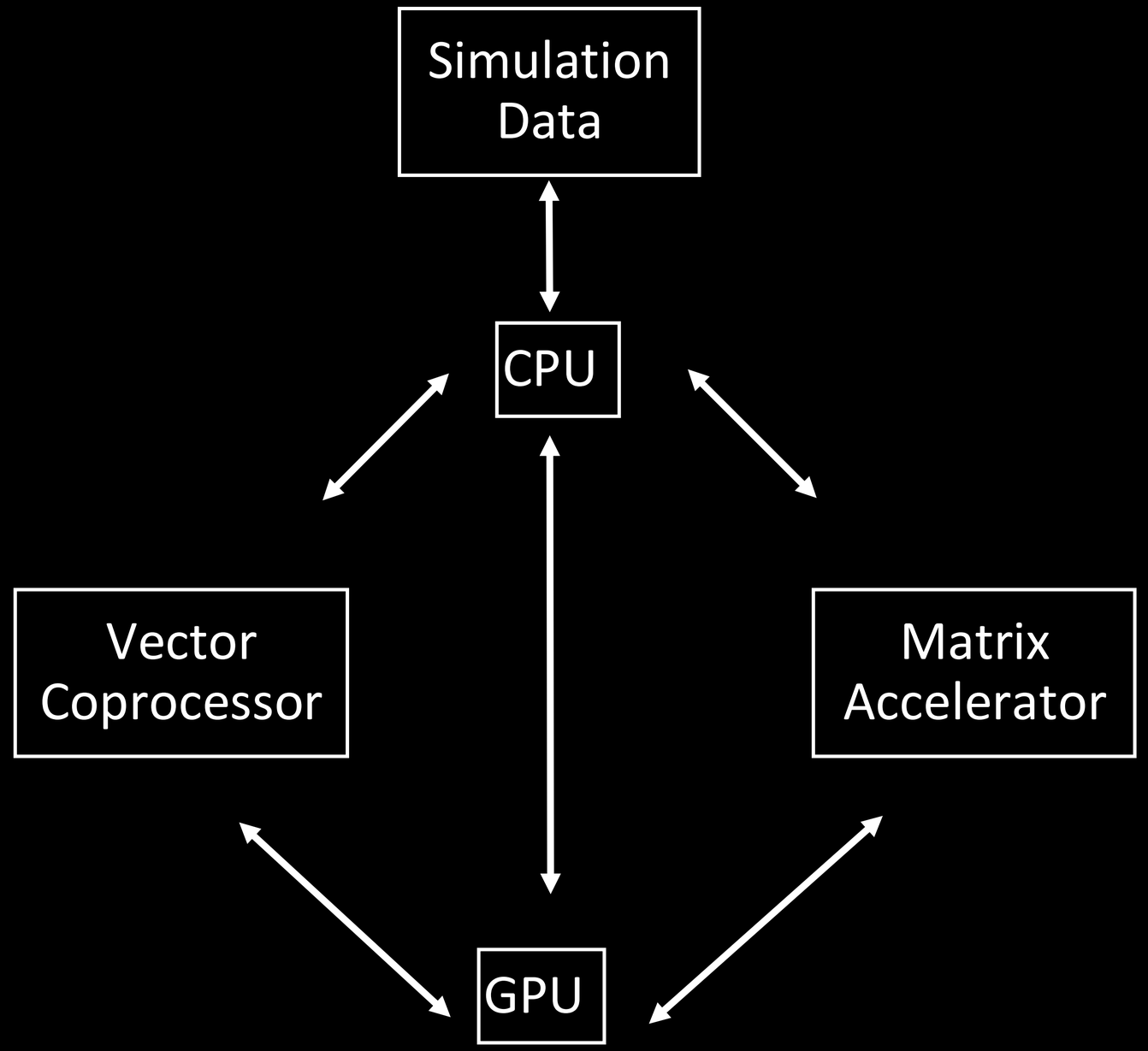
GPU

AMD

Multi-Architecture Design

Accelerator-focused vs. GPU focused

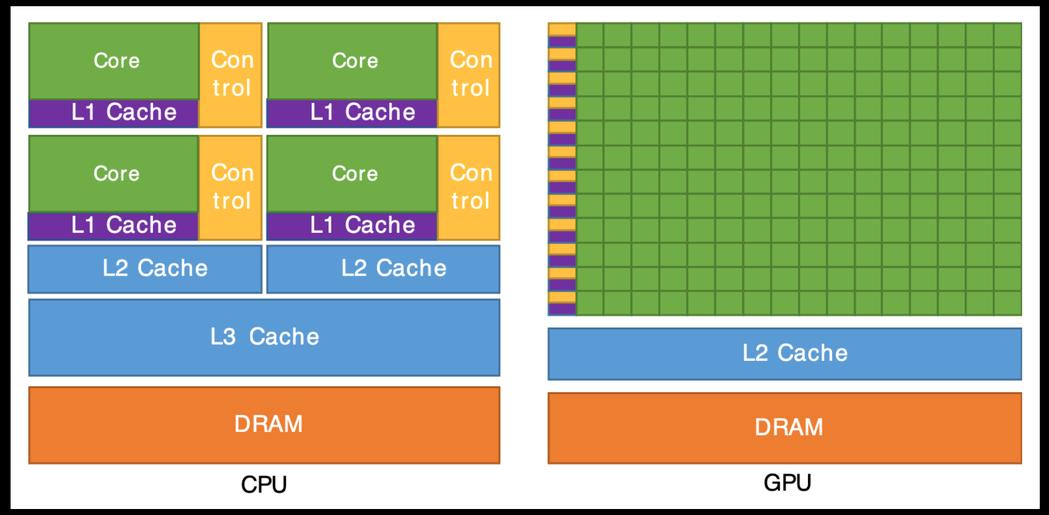
- Many current software is optimized for CPU *or* GPU
  - i.e. you compile and run the entire thing on a CPU cluster or a GPU cluster
- A lot of hardware has a variety of resources
  - e.g. Apple Silicon:
    - CPU p-cores
    - CPU e-cores
    - CPU vector extensions
    - CPU matrix accelerators
    - GPU Cores



Architecture Agnostic Strategies

Which Accelerator Do We Need?

- CPU cores are advanced control logic machines
  - most modern CPUs also have SIMD units for vectorization
  - native support for 64 bit precision
- GPUs are “dumb” SIMD machines
  - native support for 64 bit precision? only sometimes
    - Which operations ***need*** 64 bit precision?



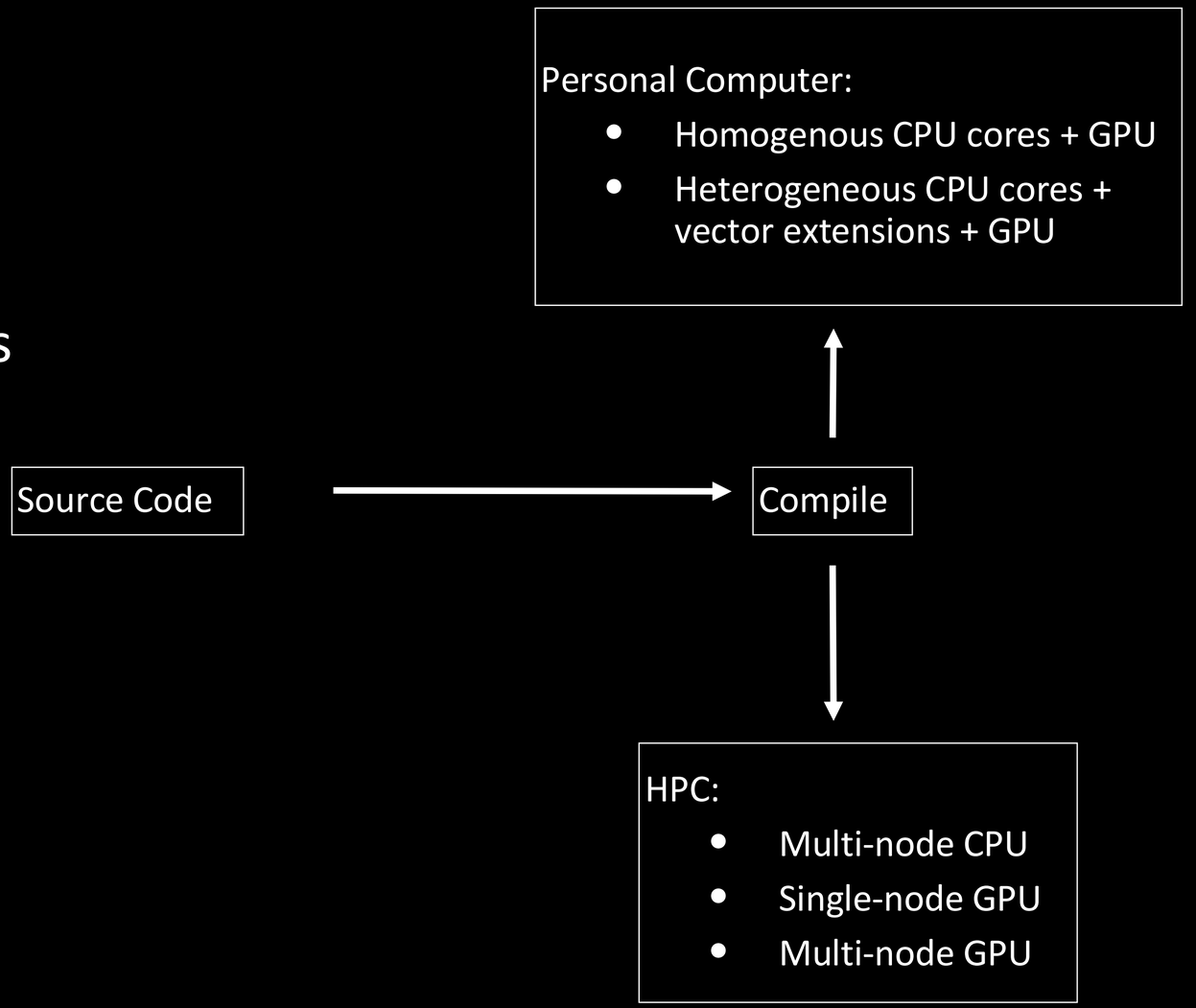
Hydro variables? Absolutely

AMR? Maybe not

Architecture Agnostic Strategies

How to implement vectorization?

- Auto-vectorization: gcc -o2 or -o3
  - How reliable is auto-vectorization?
  - Compilers are good, but vectorizing loops is hard to automate
- Explicit low-level vectorized functions BLAS, LAPACK, etc.
  - cuBLAS for GPUs should make porting relatively smooth



Conclusions

- Modern GRB simulations require vast computational resources
- Modern computer technologies are vast in their variety
- We need to standardize software techniques to utilize different hardware/software platforms
- Much of the core technology exists
- We need consistency and transparency
- Consider which operations need precision