NVIDIA GTC 2024

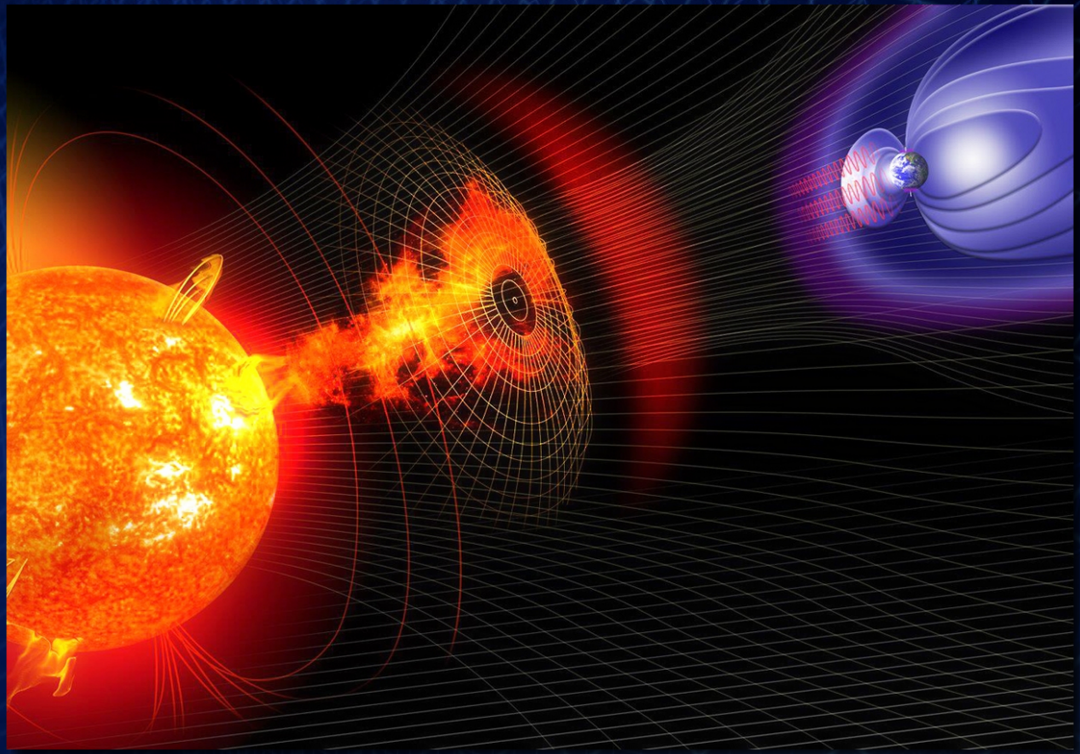
Simulating Solar Storms on GPUs with Fortran Standard Parallelism

Ronald M. Caplan, Miko M. Stulajter, Jon A. Linker,
Tibor Torok, Cooper Downs, Andres Reyes, Viacheslav S. Titov,
Roberto Lionello, and Pete Riley

Predictive Science Inc.

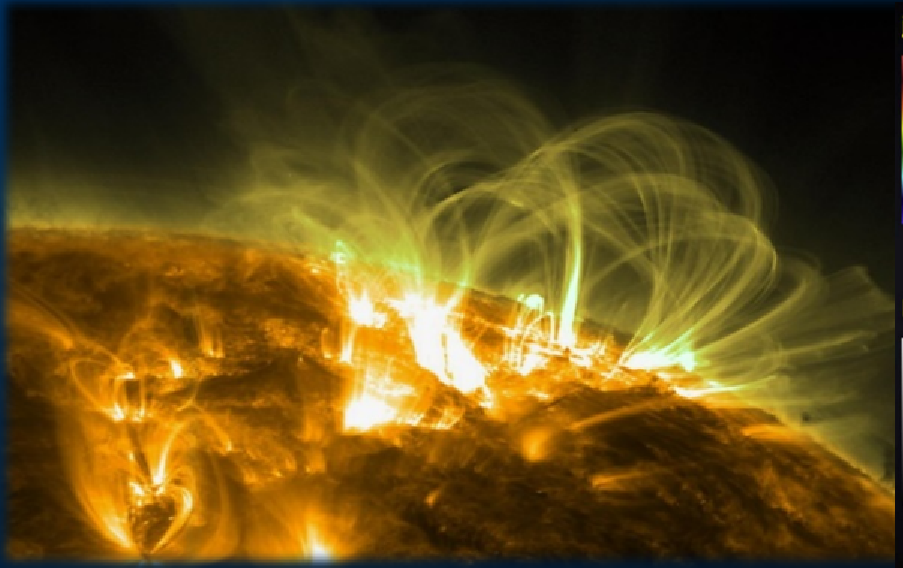
Outline

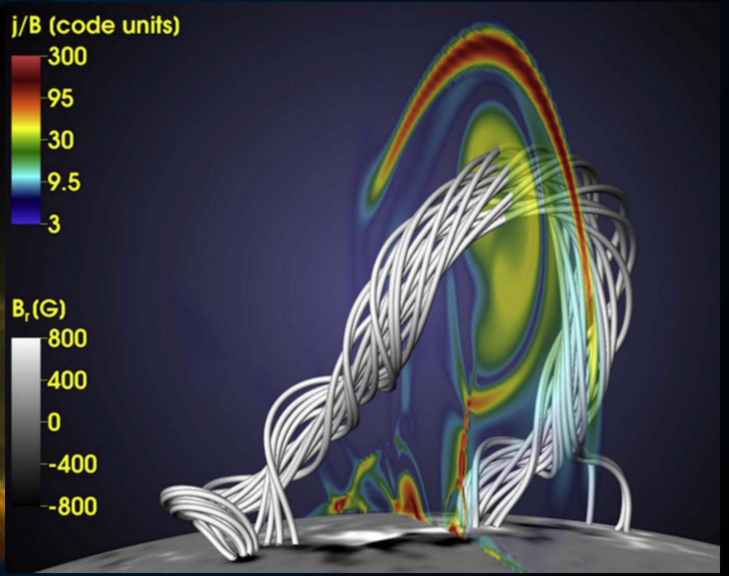
- The `what’ & `why’ of Solar Storms
- How *you* can model Solar Storms
- Run on GPUs with “just Fortran”?
- Let’s see it!



What are Solar Storms?

- Large explosive events on the Sun such as solar flares and coronal mass ejections (CME)
- CMEs can eject billions of tons of magnetized million-degree plasma out into space
- They originate in regions of strong magnetic field on the solar surface called “active regions”
- Their structure can be mathematically modeled with a twisted “flux rope” magnetic field





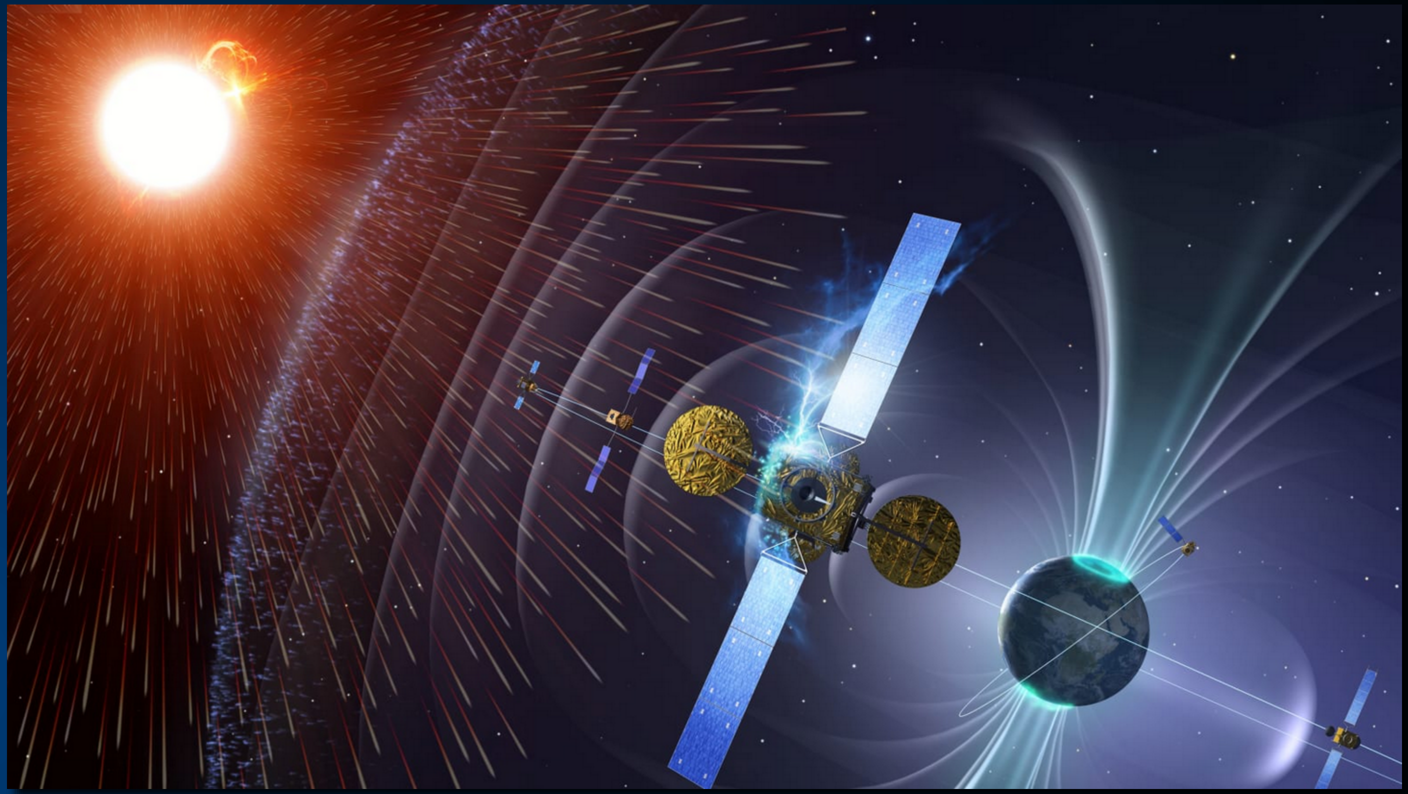
Why Study Solar Storms?

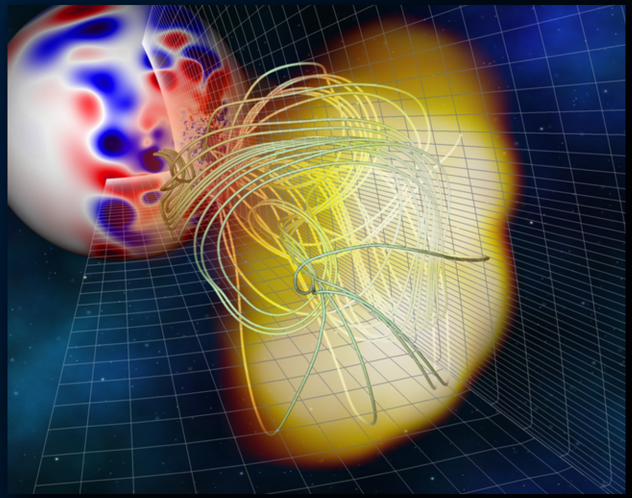
- Interesting! CMEs involve multiple levels of physical scales and processes

- Having many observations allows validation of physical models

- Important! CME impacts at Earth can cause interference & damage to our electronic infrastructure including GPS satellites and the power grid

- Numerical models are a key tool in solar storm analysis and prediction

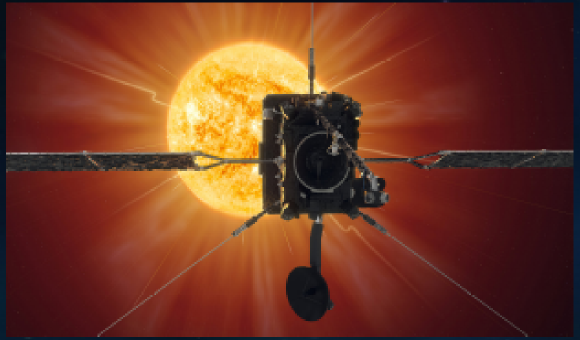




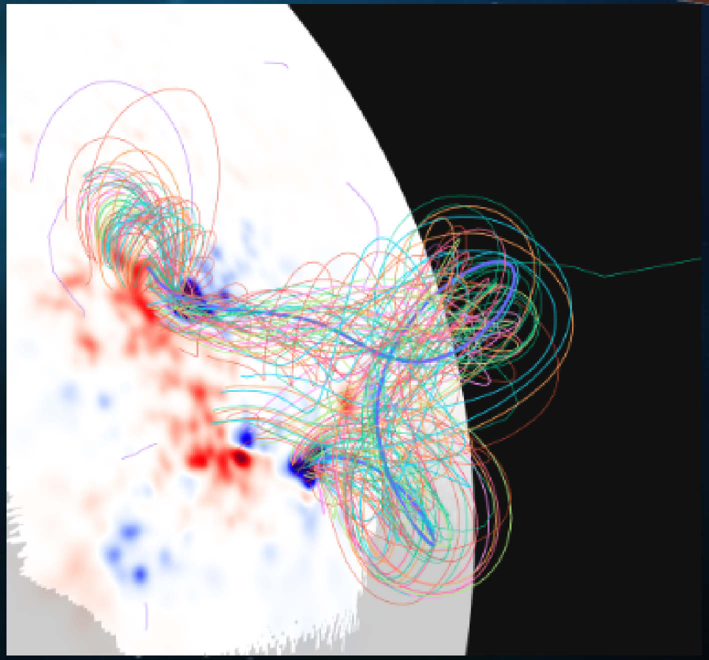
How *you* can Model Solar Storms: Overview

CORHEL-CME *Recipe for Making Solar Storms*

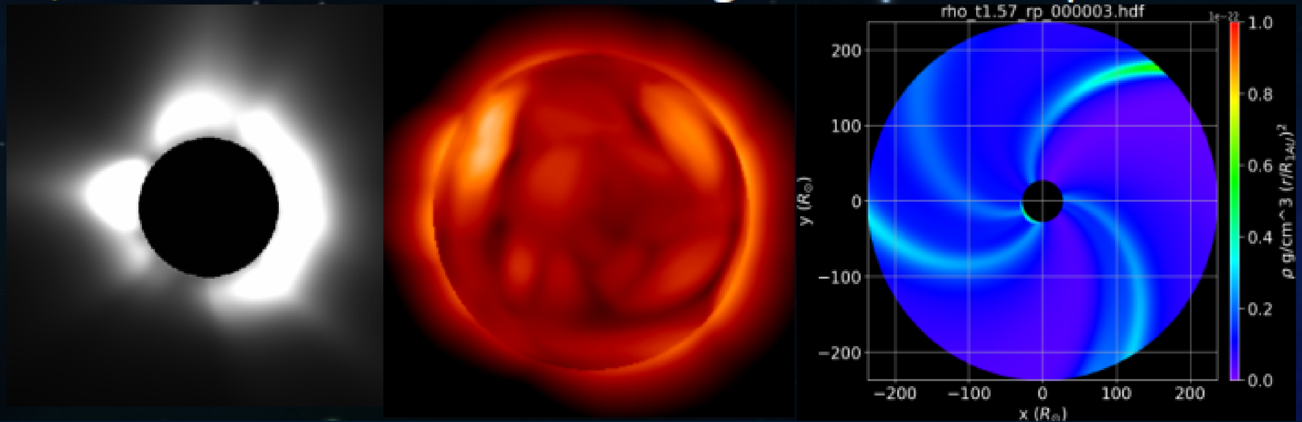
1) Get the Sun's surface magnetic field from satellite observations:



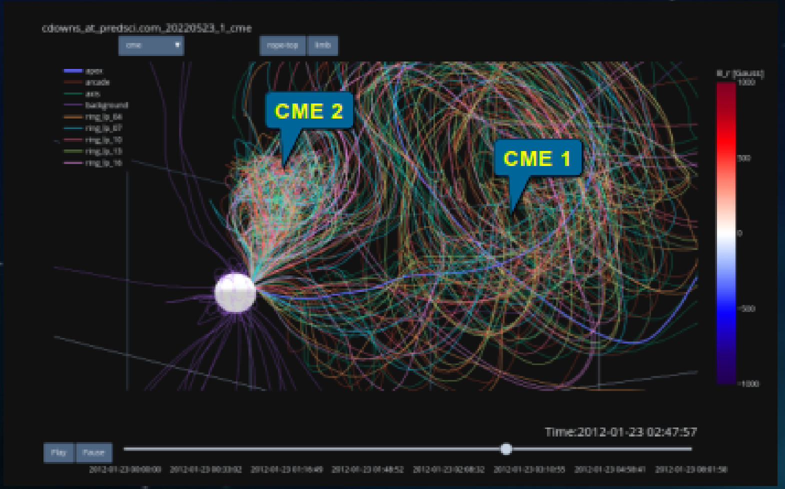
2) Design twisted magnetic rope(s) to erupt:



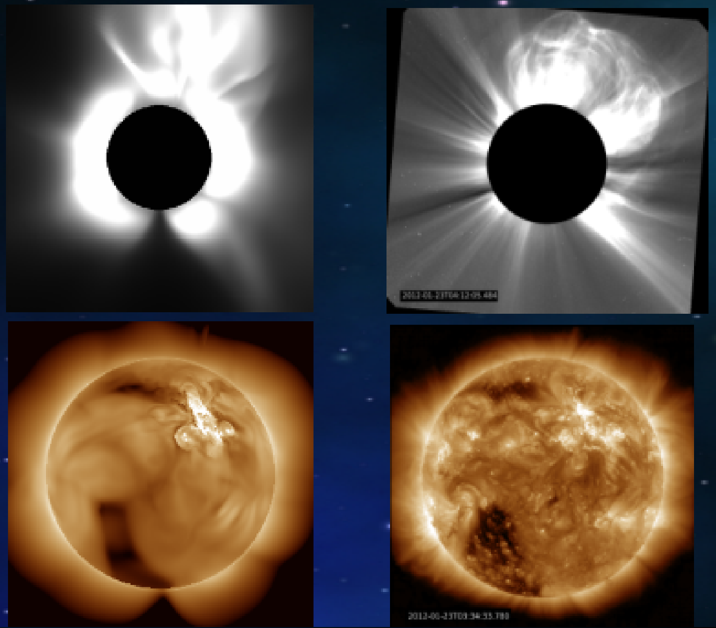
3) Simulate the Sun's background atmosphere:



4) Insert the rope(s) and run a simulation to make them erupt and travel to Earth!



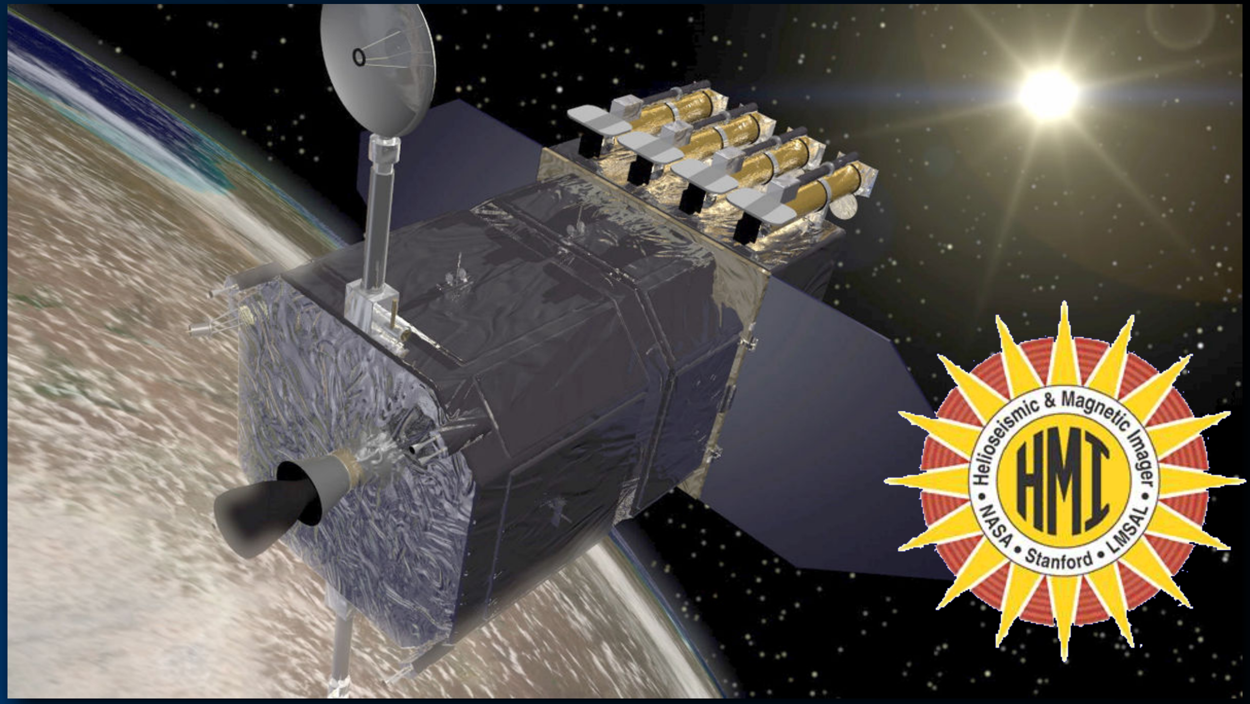
Simulation | Actual Sun



How *you* can Model Solar Storms: (1) Read the Sun

CORHEL-CME

- Grab observations of the solar surface magnetic field
- Most observations along Sun-Earth line, but need full Sun data!
  - "Synoptic"/Diachronic: Take band of data over ~28 day solar rotation (default)
  - Synchronic: Surface flux transport models can simulate the flow of the field behind the Sun
- Automatic processing of full-Sun data including binning, flux balancing, and smoothing



HipFT

DC Fortran

github.com/predsci/hipft

DIFFUSE

DC Fortran

Stulajter, et. al. Lec Notes in Comp Sci, 13194, 3-21 (2022)

How *you* can Model Solar Storms: (2) Calm before the storm

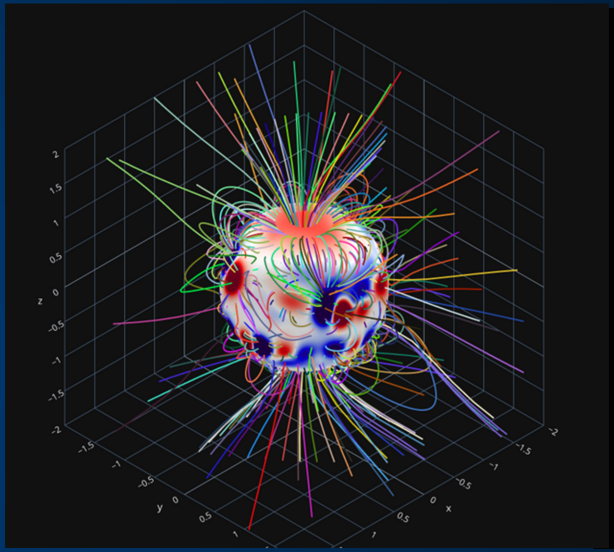
CORHEL-CME

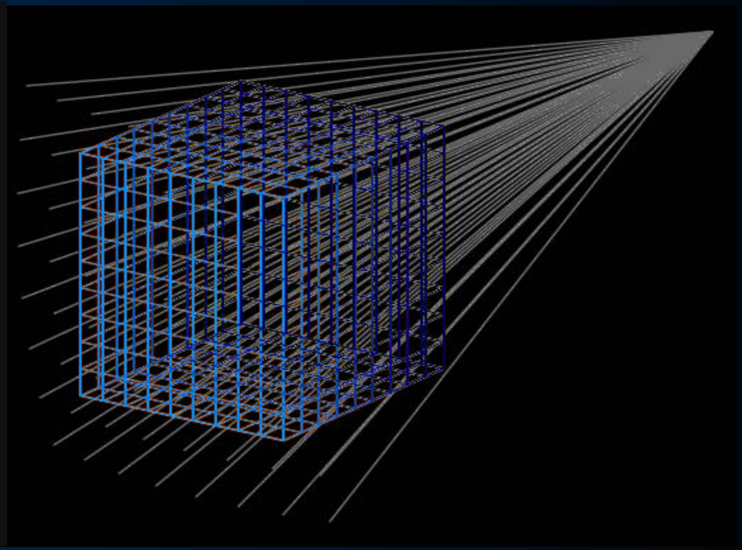
- Use surface field as lower boundary condition for a magnetohydrodynamic (MHD) simulation of the Sun's atmosphere
- Start with a "potential field" solve for the initial 3D magnetic field (similar to POT3D)
- Run the MHD simulation long enough to reach a quasi-steady background solution
- Trace through solution (with a physics model) to create synthetic observations directly comparable to real ones

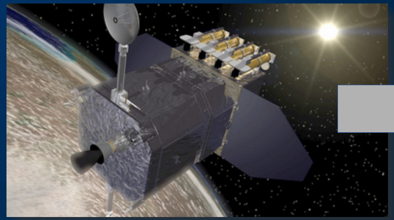
POT3D

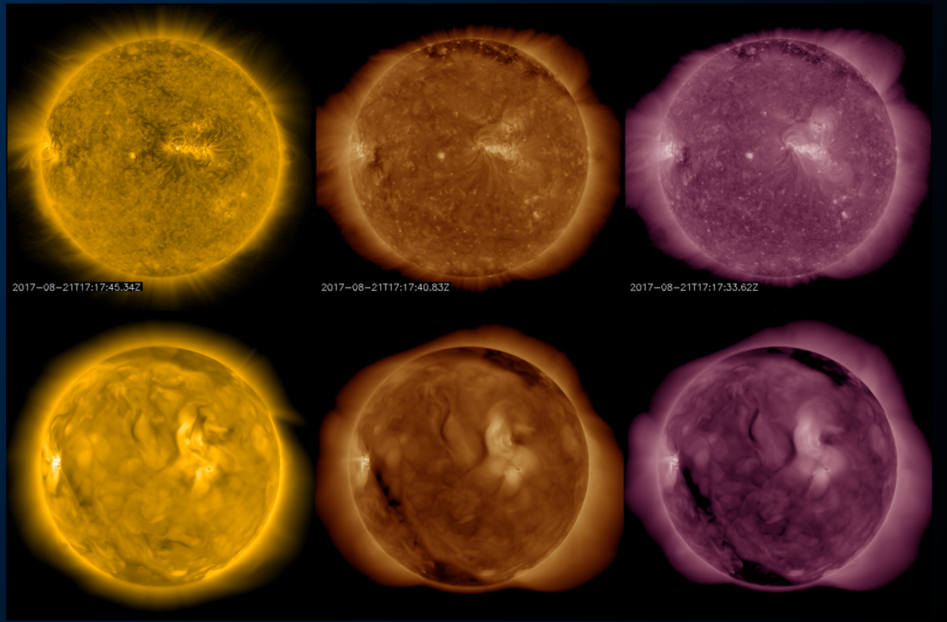
DC Fortran

github.com/predsci/pot3d









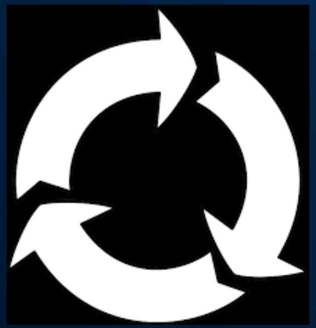
MAS

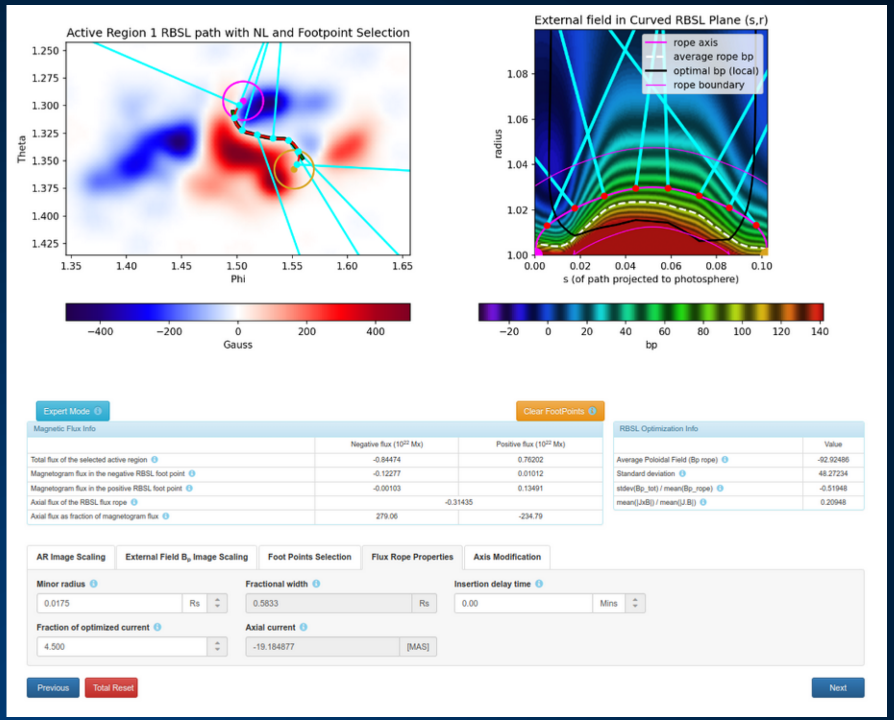
NVIDIA GTC

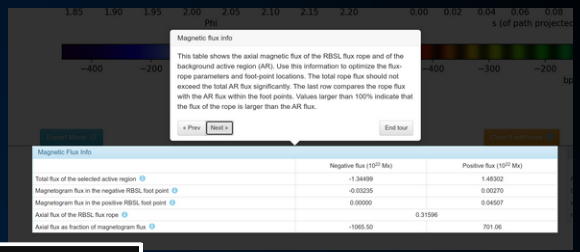
How *you* can Model Solar Storms: Design your perfect storm

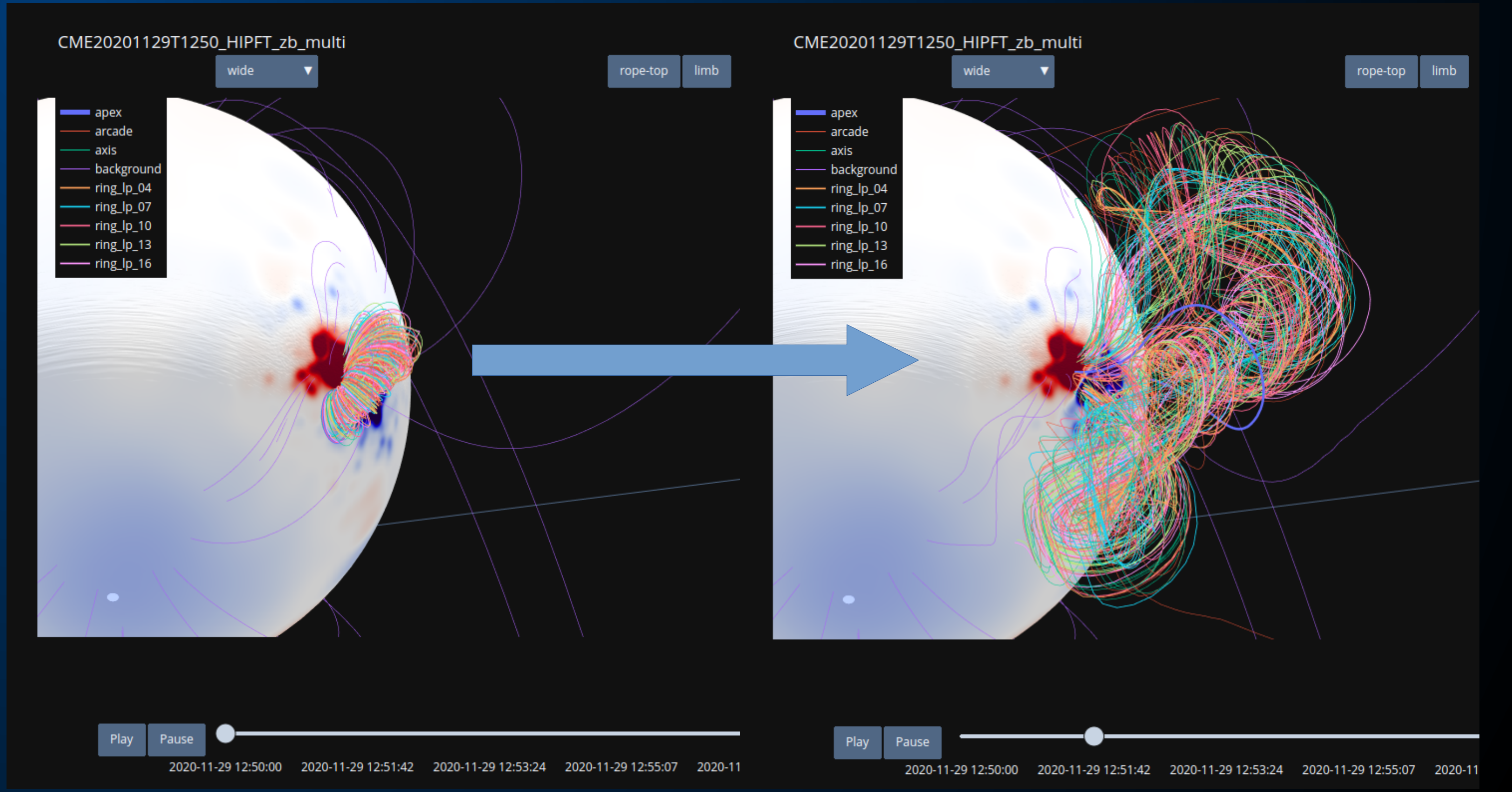
CORHEL-CME

- Design flux rope(s) with GUI
- Step-by-step guides through tool-tips and tutorial videos
- Test if flux rope(s) are eruptive with reduced-model MHD simulation (quick ~20 minute turnaround time on 4xGPUs)
- Refine rope(s) parameters and repeat!







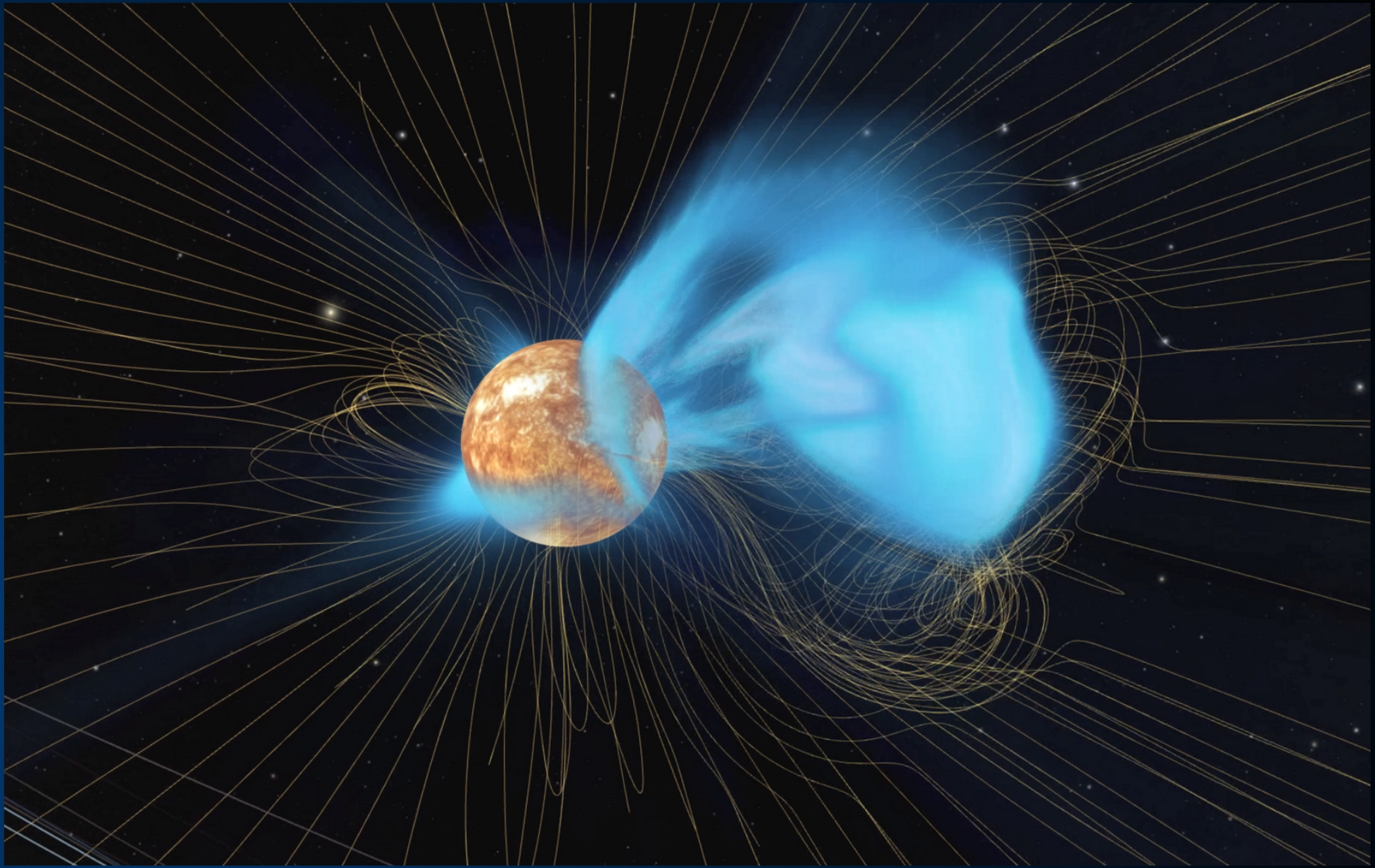


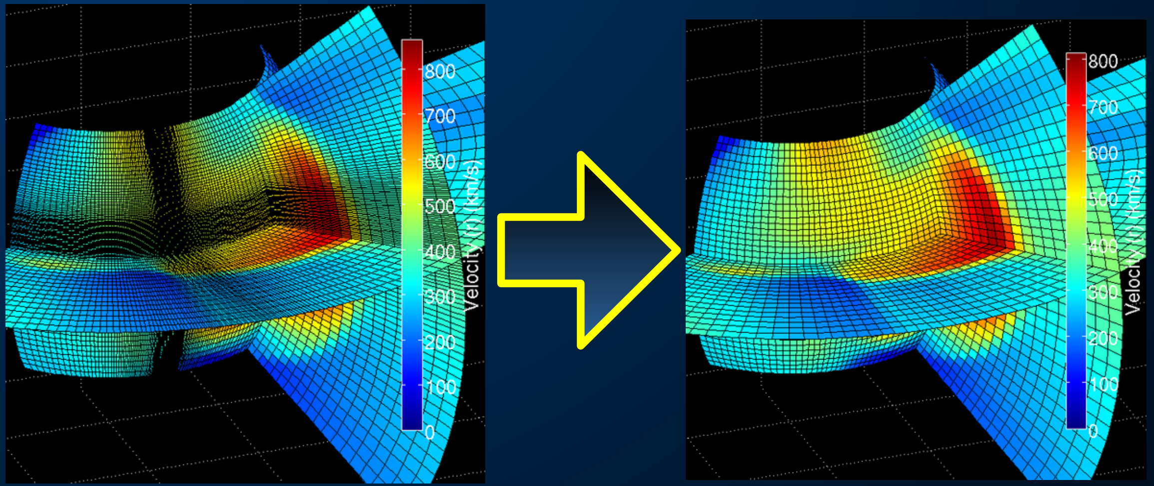
Predictive Science Inc.

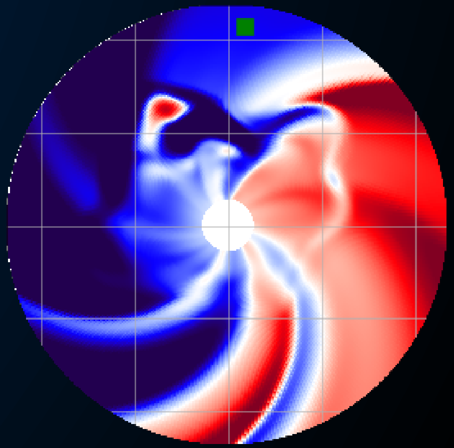
How *you* can Model Solar Storms: Release the storm!

CORHEL-CME

- Use MHD background solution as initial condition
- Insert flux rope(s)
- Run full MHD CME simulation from Sun to Earth
- Re-mesh after initial eruption to reduce compute time



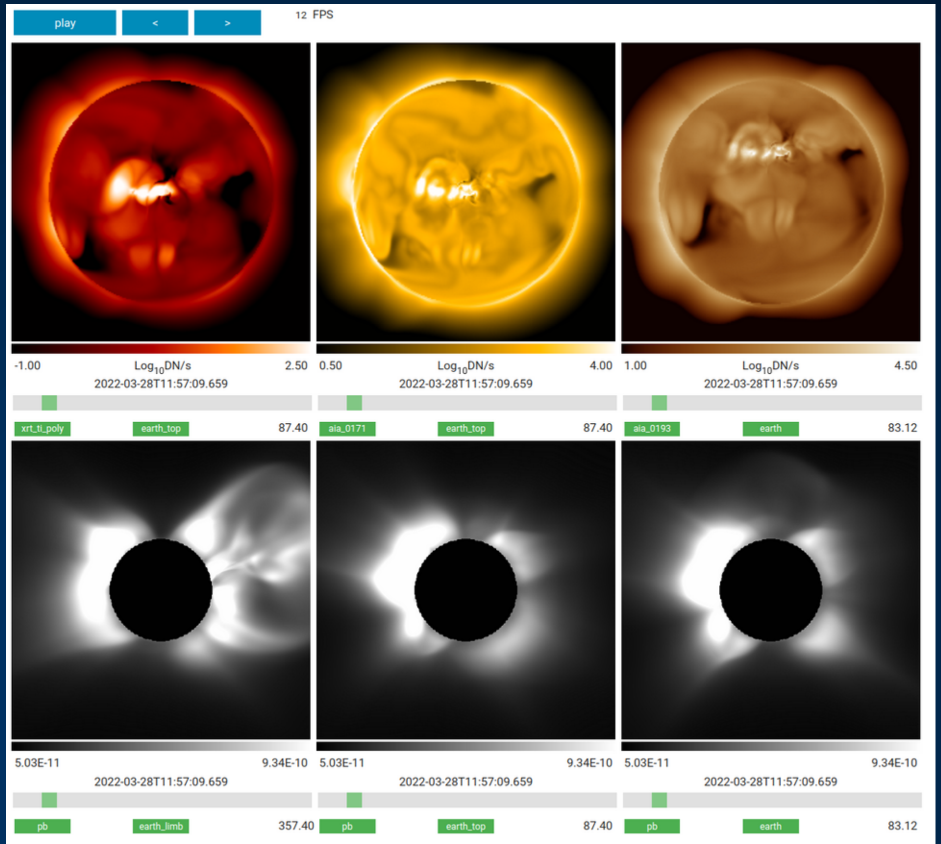


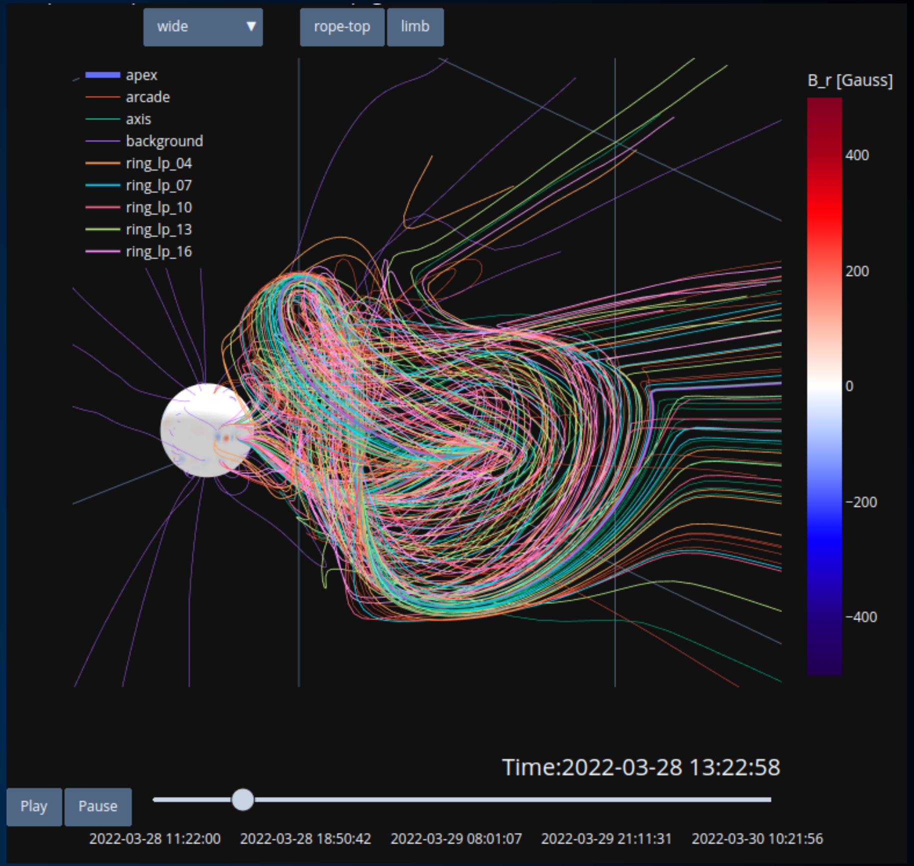


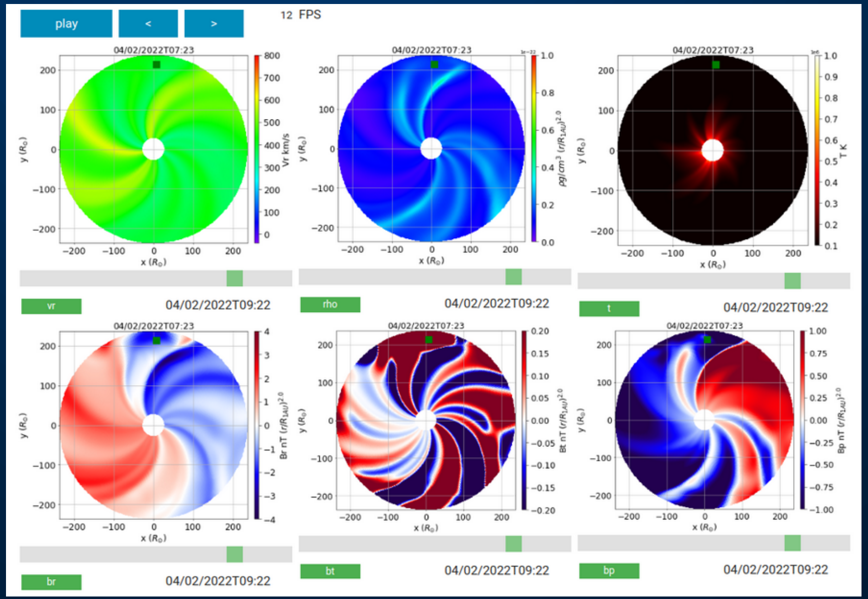
How *you* can Model Solar Storms: What now?

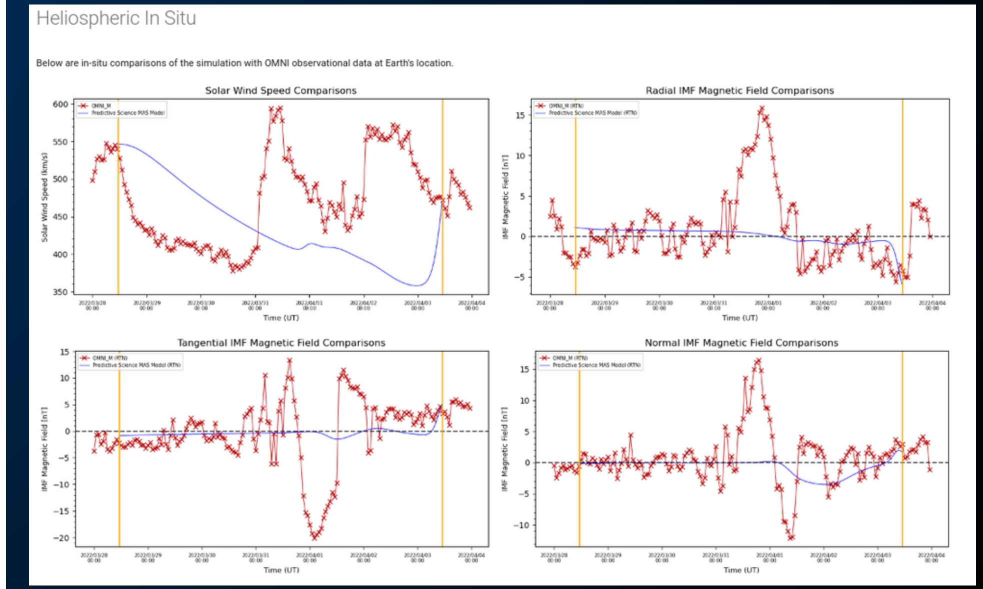
CORHEL-CME

- Auto-generated reports to analyze CME
- Simulation can be used as input to other models such as the STAT solar energetic particle model









How *you* can Model Solar Storms: Where? How does it work?

CORHEL-CME

- Web GUI hosted at NASA's CCMC
- Simulations are run on a **single** AWS GPU instance
- Computationally expensive MHD simulations run using the MAS code

MAS

ccmc.gsfc.nasa.gov

ccmc.gsfc.nasa.gov/models/CORHEL-CME~1

MAS: Magnetohydrodynamic Algorithm Outside a Sphere

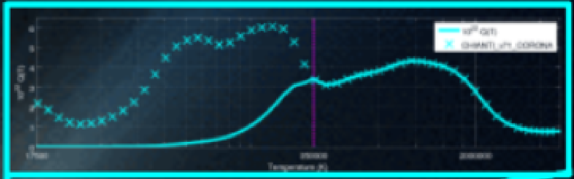
MAS
MAGNETOHYDRODYNAMIC ALGORITHM OUTSIDE A SPHERE

$$\frac{\partial \mathbf{A}}{\partial t} = \mathbf{v}\times(\nabla\times\mathbf{A}) - \frac{c^2\,\eta}{4\,\pi}\nabla\times\nabla\times\mathbf{A}$$

RESISTIVITY

$$\frac{\partial \rho}{\partial t} = -\nabla\cdot(\rho\,\mathbf{v})$$

$$\frac{\partial T}{\partial t} = -\nabla\cdot(T\mathbf{v}) - (\gamma-2)\,(T\,\nabla\cdot\mathbf{v}) + \frac{\gamma-1}{2\,k}\frac{m_p}{\rho}\left[-\nabla\cdot(\mathbf{q}_1+\mathbf{q}_2) - \frac{\rho^2}{m_p^2}Q(T) + H\right]$$



RADIATIVE COOLING

CORONAL HEATING

THERMAL CONDUCTION

$$\mathbf{q}_1 = -f(r)\,\beta_{T_{\mathrm{cut}}}(T)\,\kappa_0\,T^{5/2}\,\hat{\mathbf{b}}\hat{\mathbf{b}}\cdot\nabla T$$
$$\mathbf{q}_2 = (1-f(r))\,\frac{k}{(\gamma-1)}\frac{\rho}{m_p}\,T\,\mathbf{v}\,\hat{\mathbf{b}}\hat{\mathbf{b}}$$

$$H = H^* + \frac{\rho}{4\,\lambda_\perp}\left[|z_-|\,z_+^2 + |z_+|\,z_-^2\right]$$
$$\lambda_\perp = \lambda_0\sqrt{\frac{B_w}{|\mathbf{B}|}} \qquad |z_\pm(r=R_\odot)| = z_0$$

ALFVEN WAVES

$$\frac{\partial \epsilon_\pm}{\partial t} = -\nabla\cdot(\epsilon_\pm[\mathbf{v}\pm\mathbf{v}_A]) - \frac{\epsilon_\pm}{2}\nabla\cdot\mathbf{v}$$

$$\frac{\partial \mathbf{v}}{\partial t} = -\mathbf{v}\cdot\nabla\mathbf{v} + \frac{1}{\rho}\left[\frac{1}{c}\mathbf{J}\times\mathbf{B} - \nabla p - \nabla\left(\frac{\epsilon_+ + \epsilon_-}{2}\right) + \rho\,\mathbf{g}\right] + \frac{1}{\rho}\nabla\cdot(\nu\rho\nabla\mathbf{v}) + \frac{1}{\rho}\nabla\cdot\left(S\,\rho\,\nabla\frac{\partial\mathbf{v}}{\partial t}\right)$$

VISCOSITY

SEMI-IMPLICIT OPERATOR

WAVE TURBULENCE

$$\frac{\partial z_\pm}{\partial t} = -(\mathbf{v}\pm\mathbf{v}_A)\cdot\nabla z_\pm - \frac{z_\pm\,|z_\mp|}{2\,\lambda_\perp} + \frac{z_\pm}{4}(\mathbf{v}\mp\mathbf{v}_A)\cdot\nabla(\ln\rho) + \frac{z_\mp}{2}(\mathbf{v}\mp\mathbf{v}_A)\cdot\nabla(\ln|\mathbf{v}_A|)$$

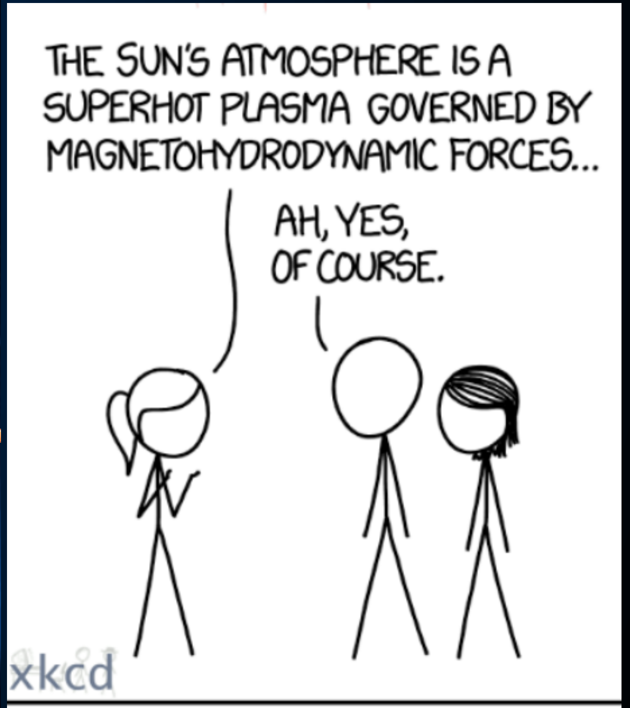
$$\nabla\cdot\mathbf{B}=0 \qquad \mathbf{B}=\nabla\times\mathbf{A} \qquad \mathbf{J}=\frac{c}{4\pi}\nabla\times\mathbf{B}$$
$$p = 2\,k\,T\,\rho/m_p \qquad \mathbf{g} = -g_0\,R_\odot^2\,\hat{\mathbf{r}}/r^2 \qquad \gamma = 5/3$$
$$\hat{\mathbf{b}} = \mathbf{B}/|\mathbf{B}| \qquad \mathbf{v}_A = \mathbf{B}/\sqrt{4\,\pi\,\rho} \qquad B_w = 6.09\,G \qquad v_c^2 = \gamma\,p/\rho$$
$$\beta_{T_m}(T) = \begin{cases}(T/T_m)^{-5/2} & T < T_m\\ 1, & T \ge T_m\end{cases} \qquad T_m = 3.5\times10^5\,K \qquad f(r) = 1 - 0.5\tanh\left[(r - 10\,R_\odot)/R_\odot\right]$$
$$S = (\Delta t^2\,\tilde{k}^2)^{-1}\,(C_w^2/(1-C_f)^2 - 1) \qquad C_f = \Delta t\,\tilde{k}\cdot\mathbf{v} \qquad C_w^2 = 0.25\,\Delta t^2\,\tilde{k}^2\,(v_c^2 + |\mathbf{v}_A|^2)$$
$$\tilde{k}^2 = 4\,(\Delta r^{-2} + (r\,\Delta\theta)^{-2} + (r\,\Delta\phi\,\sin\theta)^{-2})$$

predsci.com/mas

GPU Implementation:

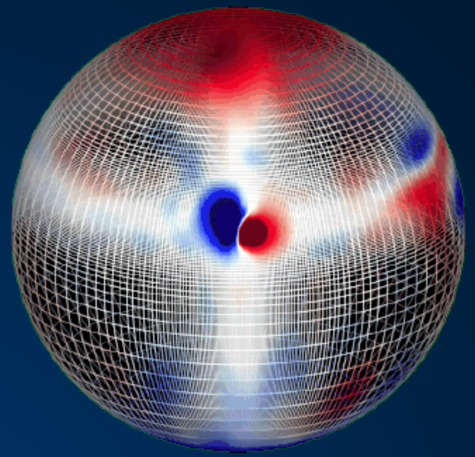
Caplan et. al. J. of Phys.: Conf. Series. ASTRONUM 2018. 1225,1 (2019) 012012

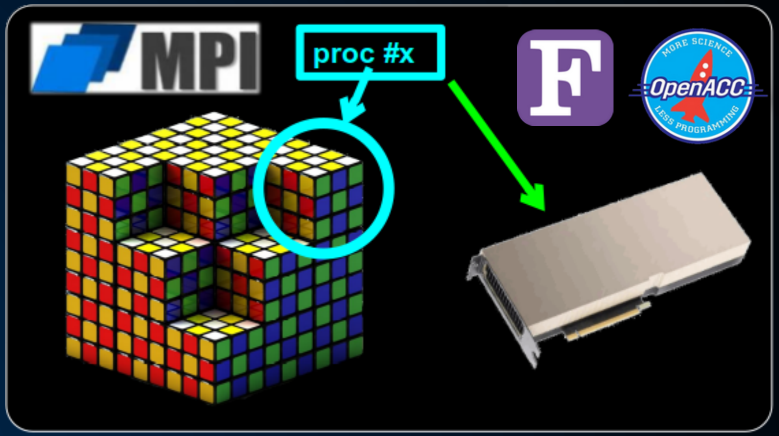
Caplan et. al. IEEE IPDPSW Proceedings., (2023) 582-590.

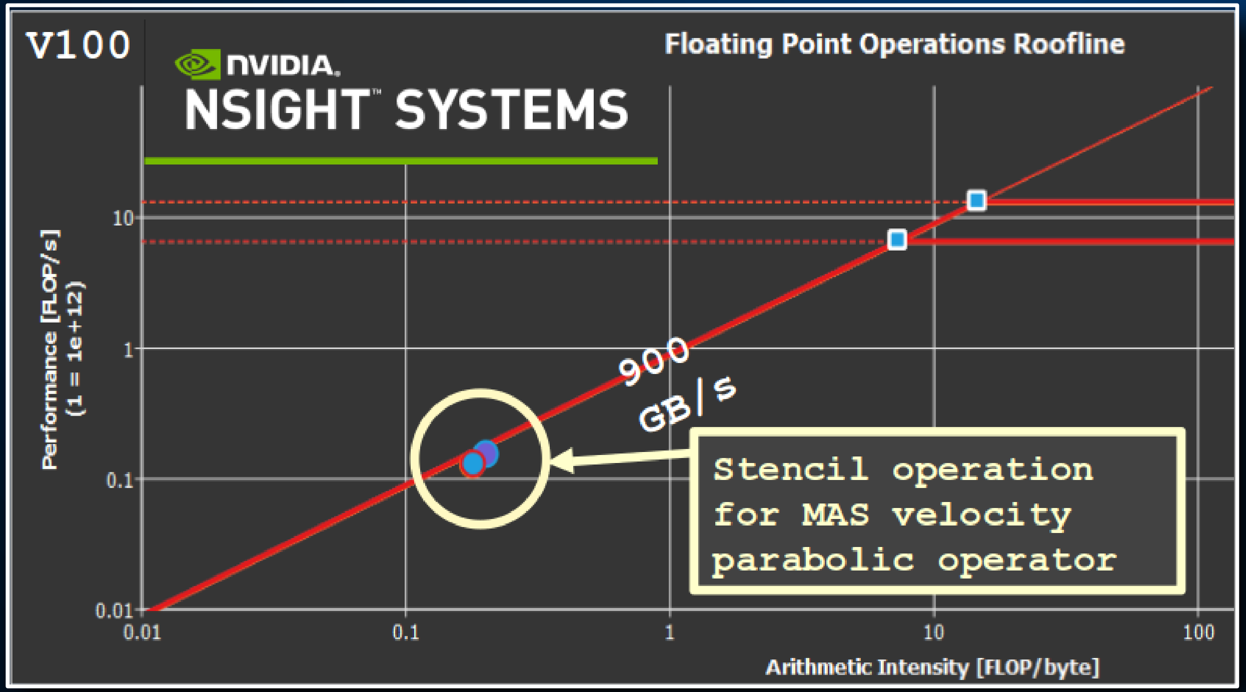


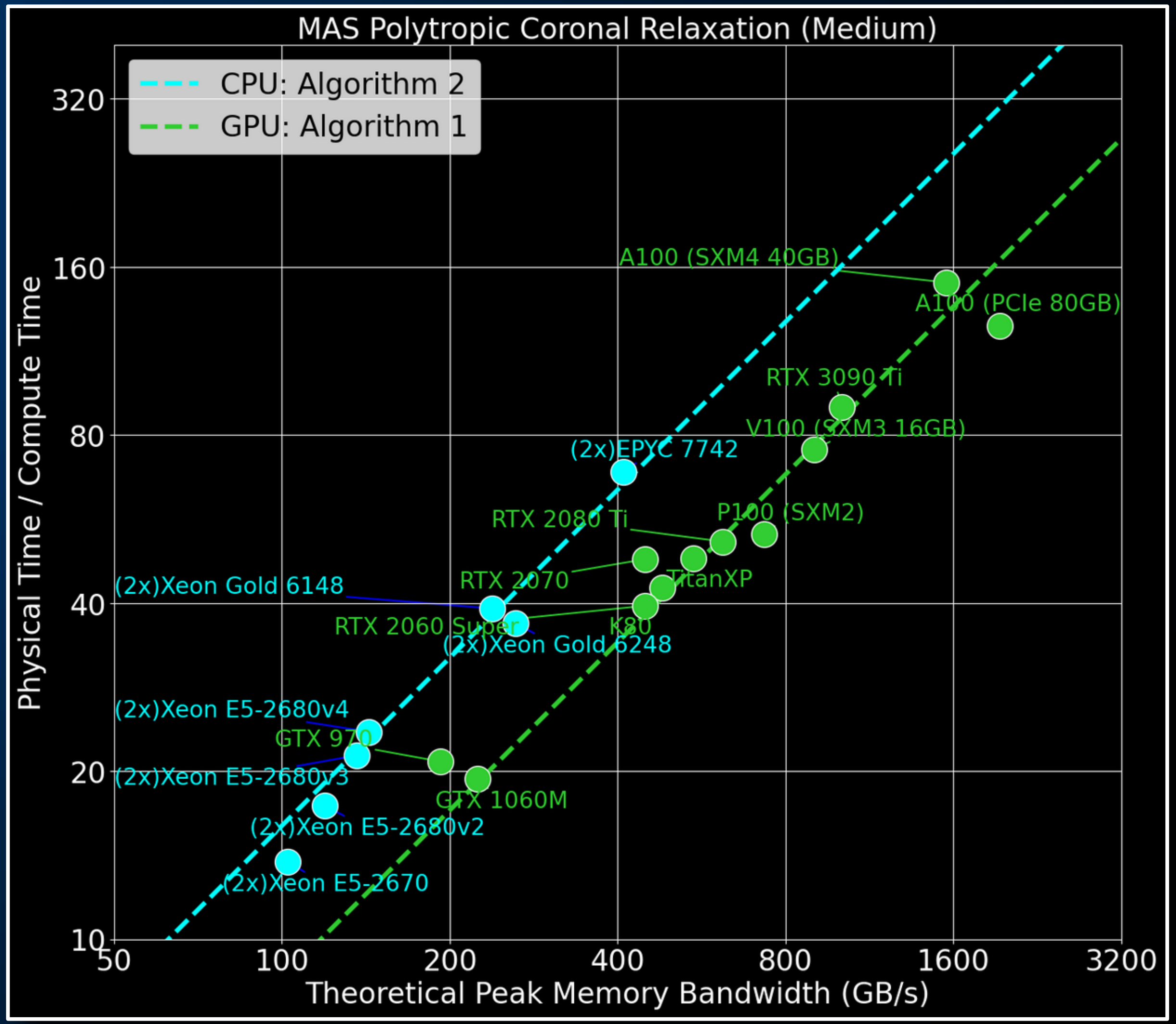
MAS: Numerical Methods and Code Details

- Finite difference on a logically rectangular non-uniform spherical grid
- Preconditioned (PC) Conjugate Gradient solvers with two PCs, **PC1**: GPU & CPU, and **PC2** (better!): CPU-Only (for now)
- **Fortran**, parallelized with **MPI**, later GPU-accelerated with **MPI+OpenACC**
- Highly memory bandwidth bound! (low AI)









Fortran on GPUs: How?

Libraries	Standard Language	Directive APIs	Low-Level APIs
cuSPARSE	DC	OpenMP OpenACC More Science, Less Programming	CUDA FORTRAN cass-support/ clfortran OpenCL

Programmer Productivity ⟶ Programmer Control

```fortran
do concurrent (i=1:N,j=1:M)
  Computation
enddo
```

- Introduced in ISO Standard Fortran 2008
- Indicates loop can be run with **out-of-order** execution
- Can be hint to the compiler that loop **may** be parallelizable
- No current support for atomics, device selection, conditionals, etc.
- Fortran 2023 specification added reductions

Fortran on GPUs: What works where? (currently)

Compatibility Matrix

[A. Herten] github.com/AndiH/gpu-lang-compat

	NVIDIA	AMD	intel
Fortran DC		(HPE)	(Early)
OpenACC More Science, Less Programming		(GCC/HPE)	(Converter*)
OpenMP®			

*github.com/intel/intel-application-migration-tool-for-openacc-to-openmp

Flang?

Fortran on GPUs: Why Just Fortran?

Why use Fortran standard parallelism?

- Longevity (ISO)
- Smaller code footprint
- More familiar to domain scientists
- Parallelism on CPUs too!
- Currently less portable than directives (may change)

These also apply to legacy codes!

Original Non-Parallelized Code

```
do k=1,np
  do j=1,nt
    do i=1,nrm1
      br(i,j,k) = (phi(i+1,j,k)-phi(i,j,k))*dr_i(i)
    enddo
  enddo
enddo
```

OpenACC Parallelized Code

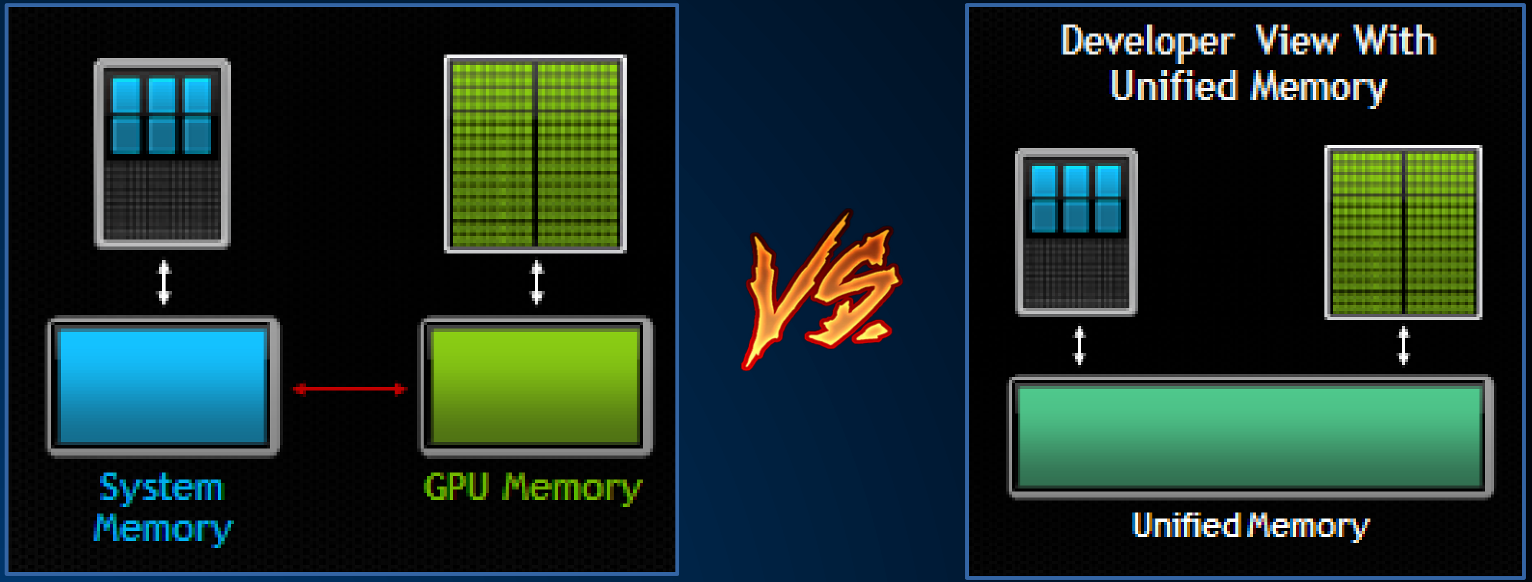
```
!$acc enter data copyin(phi,dr_i)
!$acc enter data create(br)
!$acc parallel loop default(present) collapse(3) async(1)
do k=1,np
  do j=1,nt
    do i=1,nrm1
      br(i,j,k) = (phi(i+1,j,k)-phi(i,j,k))*dr_i(i)
    enddo
  enddo
enddo
!$acc wait
!$acc exit data delete(phi,dr_i,br)
```

Fortran Standard Parallelized Code

```
do concurrent (k=1:np,j=1:nt,i=1:nrm1)
  br(i,j,k) = (phi(i+1,j,k)-phi(i,j,k))*dr_i(i)
enddo
```

Fortran on GPUs: Getting data where it needs to be

- CPU and GPU have separate memories
- Transfer between memories is slow, so want to keep data on GPU
- OpenMP/ACC have explicit data movement directives
- Unified memory (UM) auto pages data so can make DC efficient without data directives
- **Grace-Hopper has fast CPU - GPU memory sharing, so it can perform as well with UM as manual data management!**




```
!$acc enter data copyin(x)
!$acc exit  data copyout(x)
```

```
-gpu=managed
-gpu=unified
```

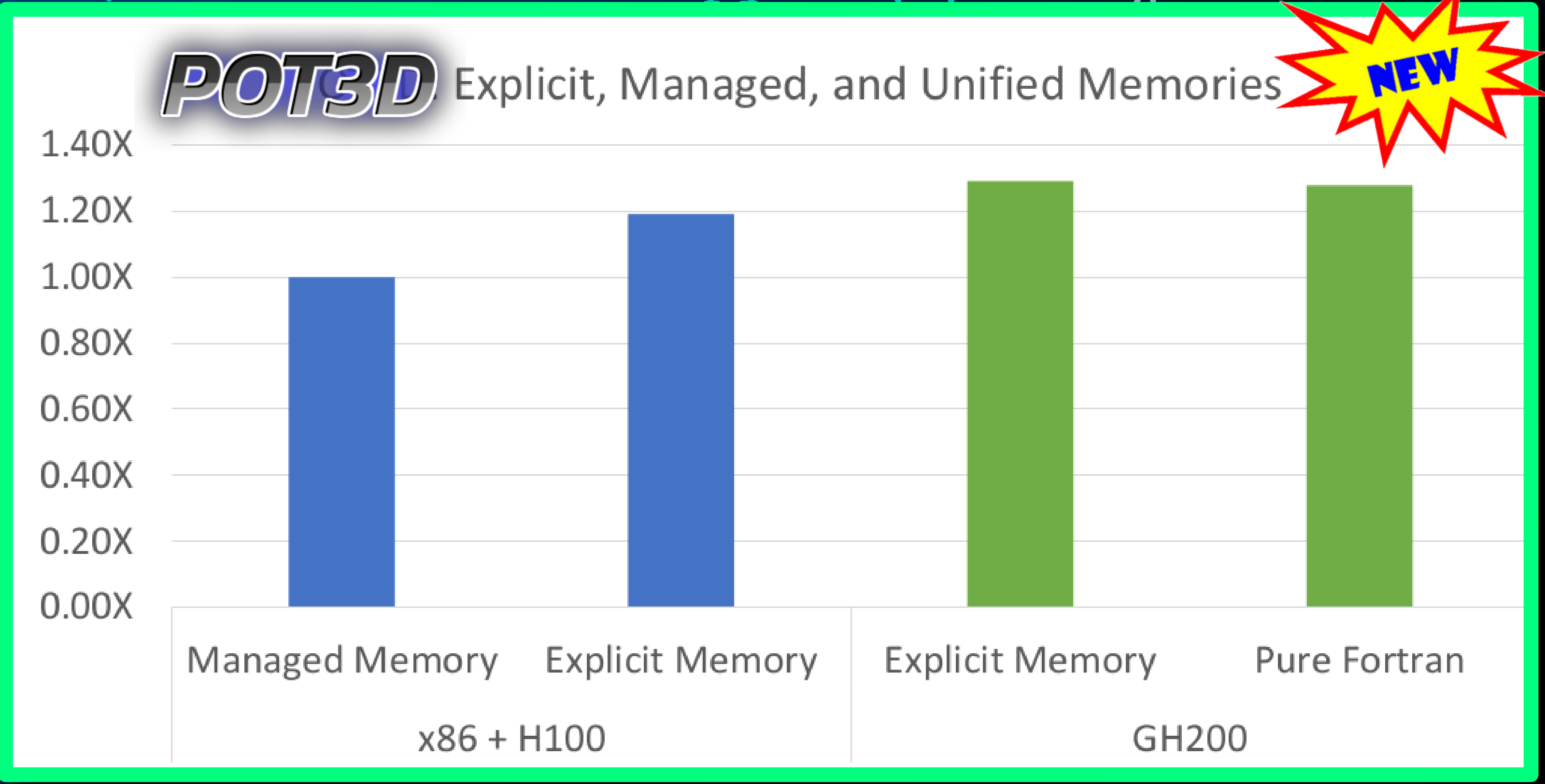
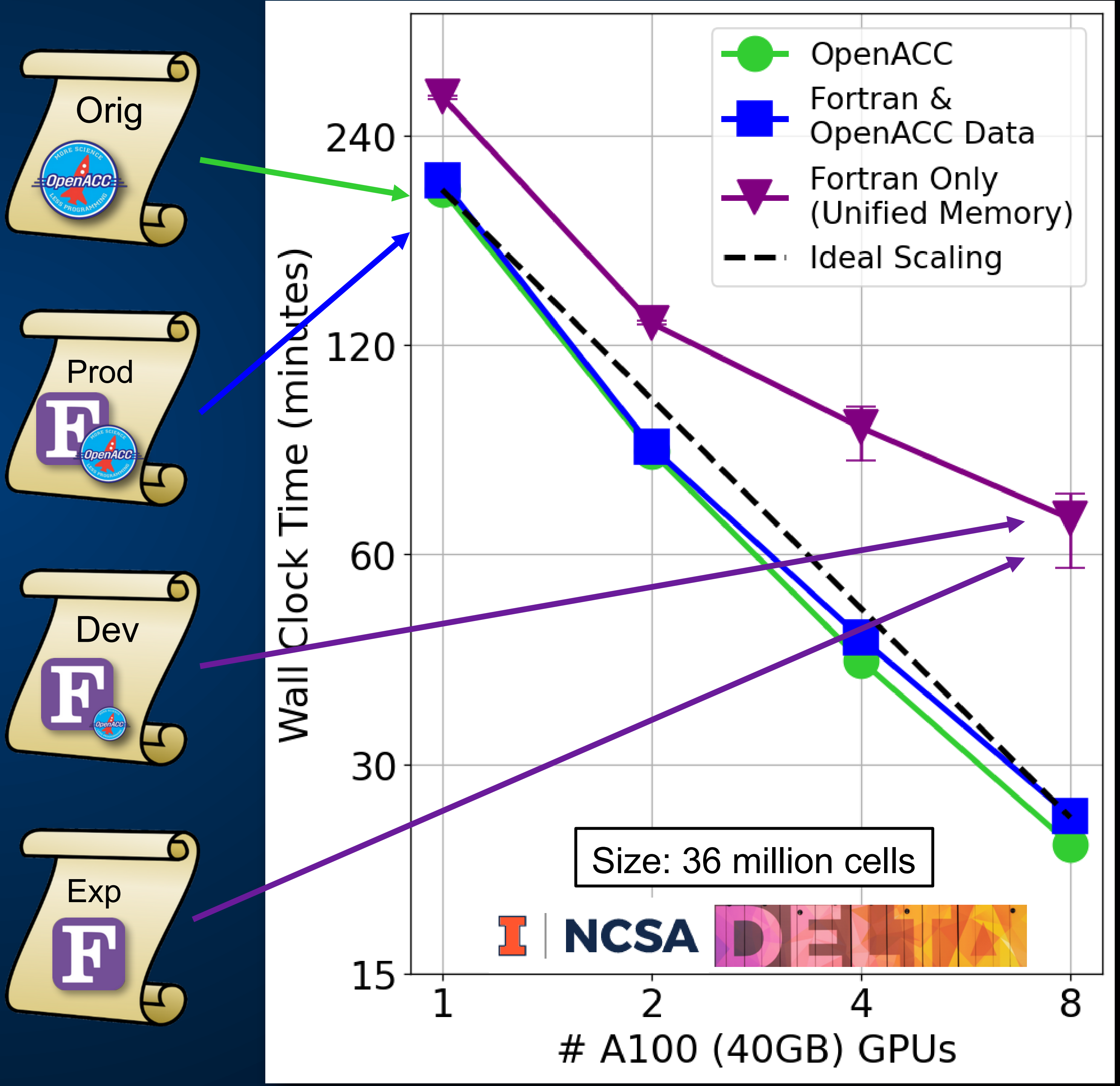


Figure courtesy of Jeff Larkin, NVIDIA

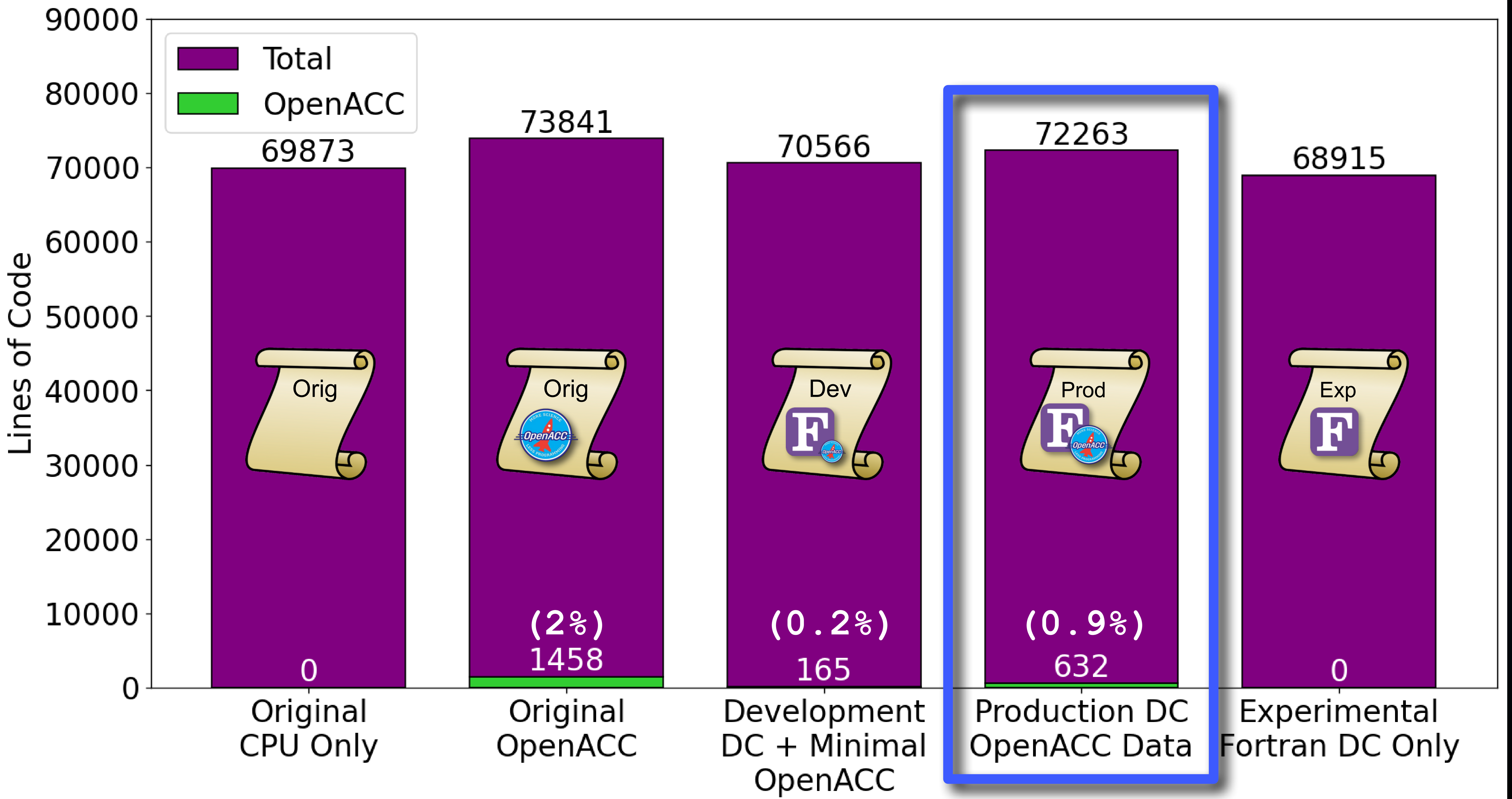
Fortran on GPUs: MAS Implementation

Caplan et. al., 13th AsHES Workshop, IEEE IPDPSW Proceedings., (2023) 582-590

- Replaced OpenACC loops with DC
- Left OpenACC for reductions (DC “reduce” too new!), and for minimal needs (routine, device selection, etc.)
- Two branches:
  - Development: Minimal OpenACC
  - Production: Development with OpenACC added for data movement
- Experimental version with ZERO directives
- Performance of Production branch similar to original OpenACC implementation
- Performance of Development and Experimental branch slower due to non-optimal UM with MPI (should get better with updates (e.g. GH)



Fortran on GPUs: MAS Implementation



Let's see it!

Real case: Solar storm on 3/28/2022:

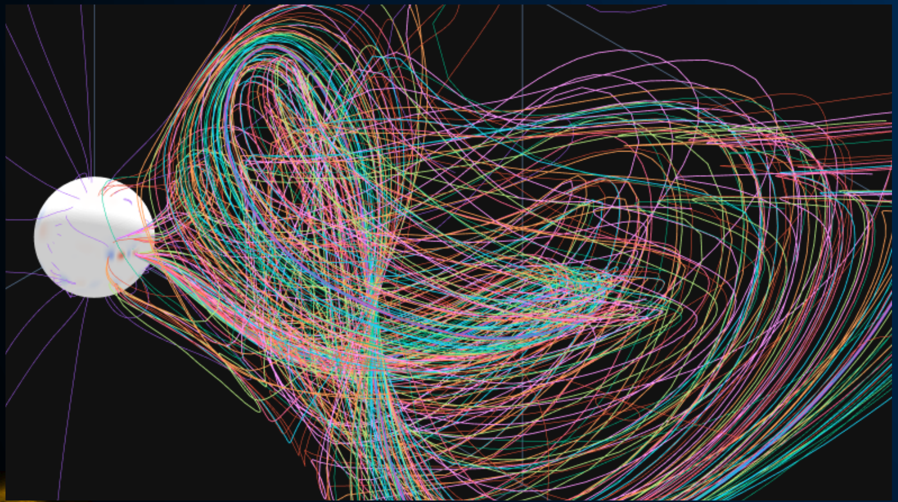
A Cannibal CME is Approaching Earth

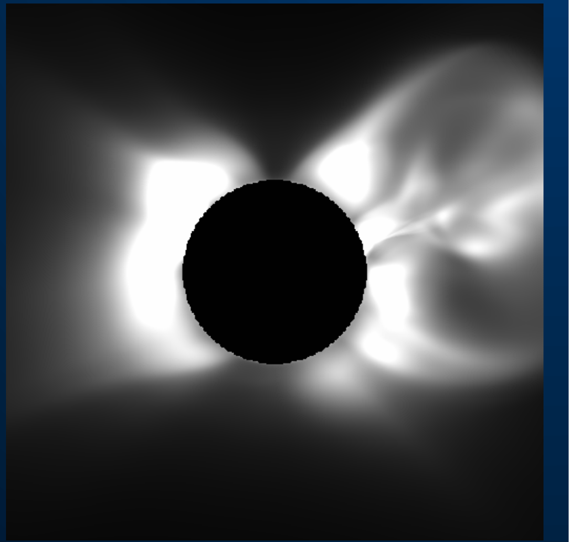
MARCH 30, 2022 / DR.TONY PHILLIPS

March 29, 2022: On March 28th, sunspot AR2975 unleashed a frenzy of solar flares– more than 17 in all. There were 11 C-class flares and 6 M-flares. At least two full-halo CMEs emerged from the chaos:

CORHEL-CME

Simulated CME:





Computational Environment:

DELTA | I NCSA

NCSA Delta 8xGPU Node

# CPUs x Model	**(2x)** EPYC 7742
# GPUs x Model	**8x** A100-40GB SXM4
Peak DP FLOP/s / GPU	9.8 TFLOP/s
Memory / GPU	40 GB
Memory Bandwidth/GPU	1555 GB/s

```
nvhpc    24.1 OpenMPI 4
gfortran 10.2 OpenMPI 4
```

ACCESS

EXPANSE COMPUTING WITHOUT BOUNDARIES | SDSC

# CPUs x Model	**(2x)** EPYC 7742
# Total Cores	128 *(we use 64)*
Peak FLOP/s	7.0 TFLOP/s
Memory	256 GB
Total Memory Bandwidth	381.4 GB/s

Let’s see it!

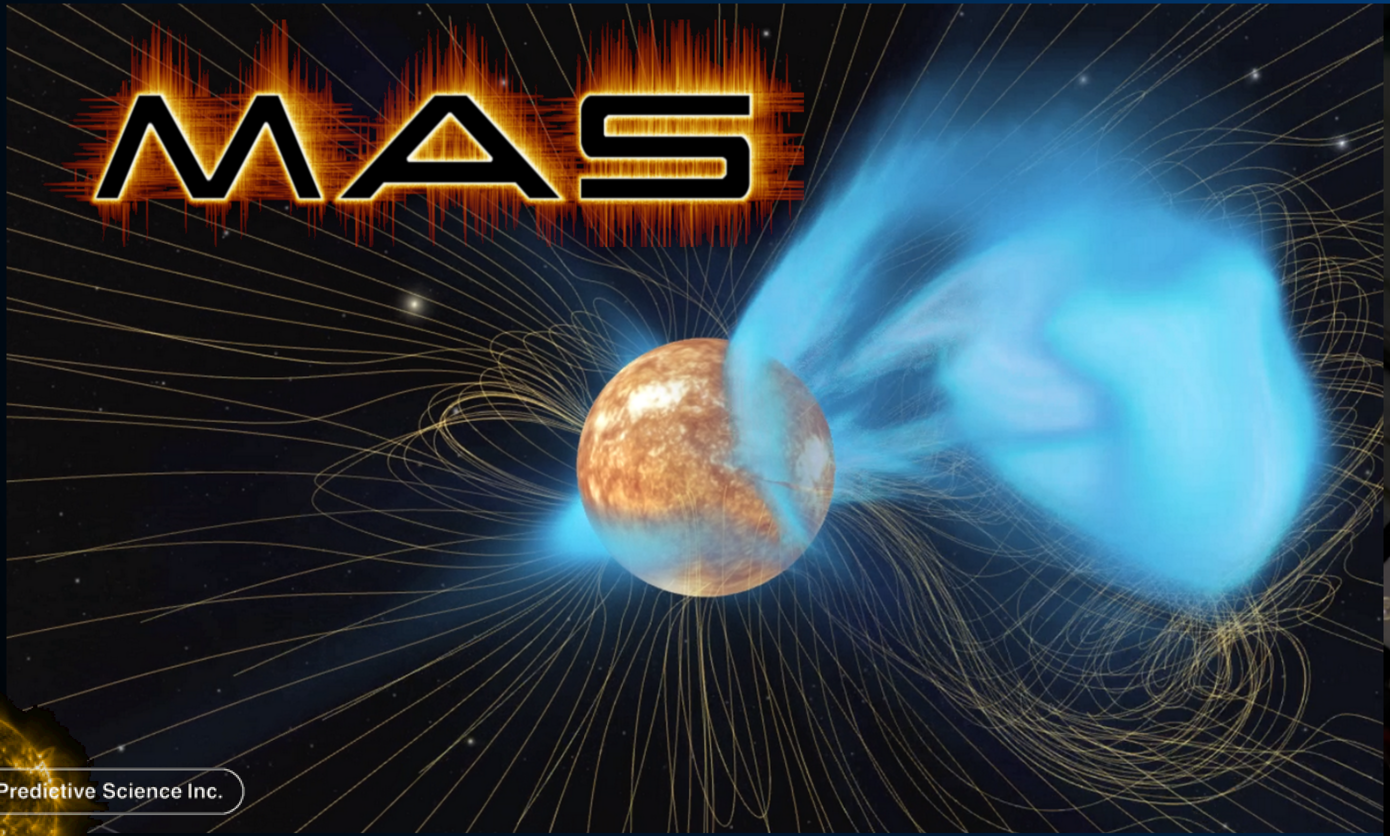
- Full design and simulation of the CME uses six runs of MAS, each with various grid sizes and run times:

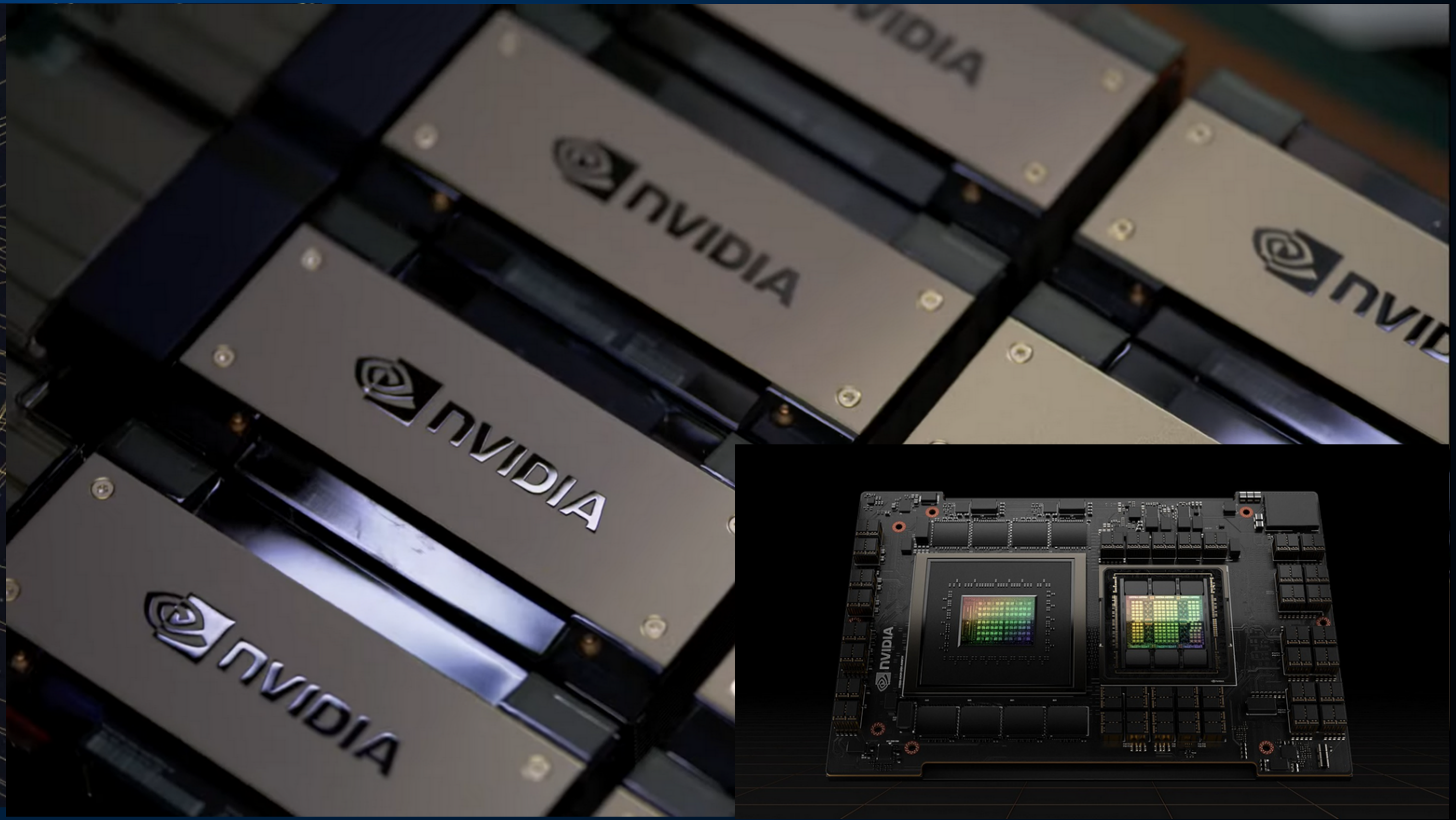
CORHEL-CME: CME 3/28/2022 Run Type	Size (Millions of Cells)	CPU Expanse 2xEPYC7742 4 Nodes [1.6 TB/s MMB]		GPU AWS P3 8xV100 1 Node [7.2 TB/s MMB]		GPU Delta 8xA100 (40GB) 1 Node [12.4 TB/s MMB]	
		Computation	**Processing**	**Computation**	**Processing**	**Computation**	**Processing**
Flux Rope Eruption (ZBMHD)	8.3	0.5	0.01	0.4	0.01	0.2	0.01
Background Relaxation Corona (TMHD)	3.1	2.1	0.6	4.1	0.6	3	0.6
Background Relaxation Heliosphere (PMHD+2D)	32.7	1.1		1.1		0.8	
CME Eruption (TMHD)	29.6	7.6		3.7		2.4	
CME Coronal Propogation (TMHD)	9.3	25.2	1	22.1	1	14.8	1
CME Heliosphere Propogation (PMHD+2D)	32.7	4.4		1.2		0.8	
Total		*40.9*	*1.7*	*32.6*	*1.7*	*22*	*1.7*
Total Wall Clock:		42.6 Hours		34.3 Hours		23.7 Hours	

EXPANSE aws EC2 P3 DELTA

Take away

The MAS code, GPU-accelerated with Fortran standard parallelism (do concurrent) and minimal OpenACC data movement directives, allows us to achieve a one day turn-around for realistic CME simulations on a single multi-GPU compute node





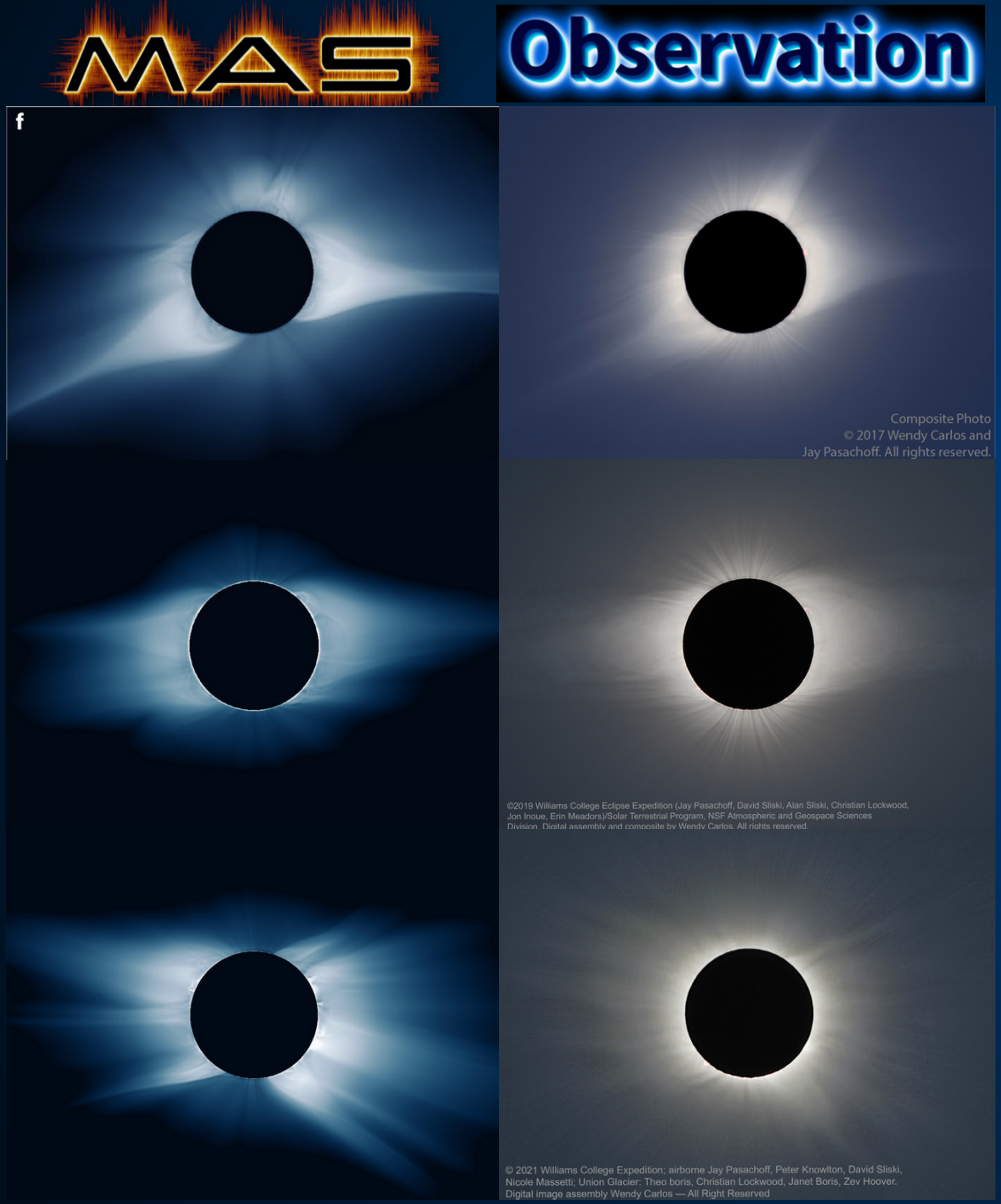
Coming Soon! Total Solar Eclipse

- PSI has a tradition to use our MAS MHD model to predict the appearance of upcoming total solar eclipses:

2017

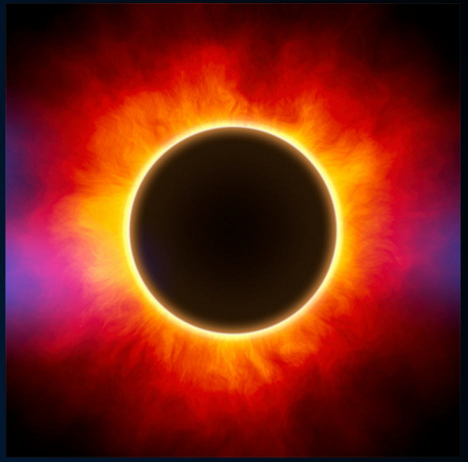
2019

2021



April 8, 2024, Solar Eclipse

On April 8, 2024, a total solar eclipse will cross North America, passing over Mexico, United States, and Canada.



Predictive Science Inc.

predsci.com/eclipse