CATEGORY: ASTRONOMY/ASTROPHYSICS - 01

POSTER **P9283**

CONTACT NAME
Ronald Caplan: caplanr@predsci.com

NVIDIA GTC GPU TECHNOLOGY CONFERENCE

Accelerating the Solar Wind with OpenACC

Ronald M. Caplan, Jon A. Linker, Zoran Mikić, Roberto Lionello, Pete Riley, and Cooper Downs

Predictive Science Inc.
9990 Mesa Rim Rd. Suite 170
San Diego, CA 92121
www.predsci.com

INTRODUCTION

The solar wind is a varied stream of charged particles continually emitted from the Sun out into the heliosphere. Modeling this emergent wind is an active area of scientific research, and a vital component of space weather forecasting.

The Coronal and Heliospheric Model Suite (CORHEL) is an in-production software suite that generates solar wind solutions and is publicly hosted at NASA's Community Coordinated Modeling Center. It contains multiple model options for the solar wind including 3D magnetohydrodynamic simulations.

Here, we show our progress integrating GPU acceleration within CORHEL and its potential application towards a continuously-running solar wind model. The goal of the GPU implementation is to allow high-resolution real-time runs normally requiring multiple CPU nodes on a supercomputer to be done with a single multi-GPU stand-alone server. To minimize issues with portability and compatibility, all GPU acceleration is done using OpenACC directives to ensure a single source-code base.

CORHEL

Coronal Model (A)

Potential Field Source Surface

$$\nabla^2\Phi = 0 \qquad \left.\frac{\partial\Phi}{\partial r}\right|_{R_S} = B_r\Big|_{R_S}$$

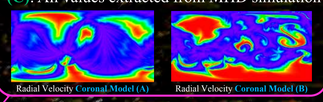
$$\vec{B} = \nabla\Phi \qquad \Phi|_{r_{ss}} = 0 \qquad r_{ss} = 2.5$$

POT3D

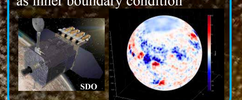
Heliospheric Solar Wind Boundary Conditions

(A) & **(B)**: Empirical values computed from extracted magnetic field

(C): All values extracted from MHD simulation



Spacecraft Observations
Surface radial magnetic field used as inner boundary condition



CORONA

SUN

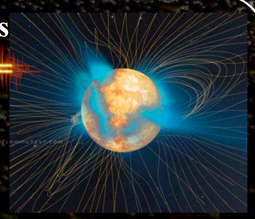
$r = 30R_S$

HELIOSPHERE

EARTH

Coronal (B) & (C) and Heliospheric Models

MAS — MAGNETOHYDRODYNAMIC ALGORITHM OUTSIDE A SPHERE



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

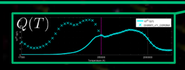
$$\frac{\partial \rho}{\partial t} = -\nabla\cdot(\rho\,\mathbf{v})$$

$$\mathbf{q}_1 = -f(r)\,\beta_{rm}(T)\,\kappa_0 T^{5/2}\hat{\mathbf{b}}\hat{\mathbf{b}}\cdot\nabla T$$

$$\mathbf{q}_2 = (1-f(r))\frac{k}{(\gamma-1)}\frac{\rho}{m_p}T\,\mathbf{v}\,\hat{\mathbf{b}}\hat{\mathbf{b}}$$

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

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



H Heating models include empirical and/or wave turbulence driven

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

$\nabla\cdot\mathbf{B} = 0$ $\quad \mathbf{B} = \nabla\times\mathbf{A}$ $\quad \mathbf{J} = \frac{c}{4\pi}\nabla\times\mathbf{B}$ $\quad p = 2\,k\,T\,\rho/m_p$ $\quad \mathbf{g} = -g_0 R_S^2\hat{\mathbf{r}}/r^2$ $\quad g_0 = 6.00\,G$ $\quad \gamma = 5/3$ $\quad \hat{\mathbf{b}} = \mathbf{B}/|\mathbf{B}|$ $\quad \mathbf{v}_A = \mathbf{B}/\sqrt{4\pi\rho}$ $\quad R_S = 6.00\,G$ $\quad c_s^2 = \gamma\,p/\rho$

$$\beta_{rm}(T) = \begin{cases} (T/T_m)^{-5/2} & T < T_m \\ 1, & T \geq T_m \end{cases} \qquad T_m = 3.5\times10^5\,K \qquad f(r) = 1 - 0.5\tanh[(r-10\,R_S)/R_S]$$

$$S = (\Delta t^2 k^2)^{-1}\,(C_0^2/(1-C_f)^2 - 1) \qquad C_f = \Delta t\,k\cdot\mathbf{v} \qquad C_0^2 = 0.25\,\Delta t^2 k^2\,(c_s^2 + |\mathbf{v}_A|^2)$$

$$k^2 = 4\,(\Delta x^{-2} + (r\,\Delta\theta)^{-2} + (r\,\Delta\phi\,\sin\theta)^{-2})$$

$\gamma = 1.05$ $\quad \gamma = 5/3$ $\quad \gamma = 3/2$

Boundary Solves Avoids spurious currents caused by interpolation errors of boundary values.

$$\nabla_t^2\Psi = \partial B_r/\partial t$$

$$\nabla_t^2\Phi = -\nabla_t\cdot(\mathbf{v}\times\mathbf{B})_t$$

$$\mathbf{E_t} = \nabla_t\times\Psi\,\hat{\mathbf{r}} + \nabla_t\Phi$$

IMPLEMENTATION

Coronal Model (A) POT3D [arXiv:1709.01126]

Algorithms: Finite-difference leading to sparse matrix solve using Preconditioned Conjugate Gradient (PCG) with two PC options: **PC1** (diagonal scaling) and **PC2** (Block-Jacobi w/ ILU0). For most cases, **PC2** much faster than **PC1**.
OpenACC: - Only able to portably accelerate **PC1**. - CUDA-aware MPI for multi-GPU capability. - Unstructured data regions to keep data on GPUs.

Coronal Model (B) MAS [arXiv:1811.02605]

https://devblogs.nvidia.com/solar-storm-modeling-gpu-openacc

Algorithms: Finite-difference with explicit and implicit time-stepping. Implicit steps use same PCG solver as in **Coronal Model (A)**.
OpenACC: - Manual deep-copy for derived types. - Vector-friendly vs. cache-friendly memory layout for stencil operations. - Array reductions with atomics.

Coronal Model (C) *(GPU acceleration is under current development.)*

Heliospheric Model MAS

Algorithms: Same as **Coronal Model (B)** with additional 2D boundary condition PCG solvers. PCG solvers more efficient with **PC1**,
OpenACC: 2D boundary solves can be too small for efficient GPU acceleration, so acceleration made optional.

Portability: For all implementations, we exclusively use OpenACC directives resulting in a portable, single source code.

CPU & GPU Single Source!

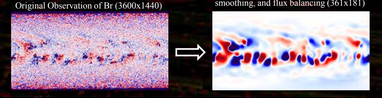
Code modification summary:

```
(A)       < 1% !$acc lines added out of  ~8,000 total.
(B) & (H) < 2% !$acc lines added out of ~60,000 total.
```

REAL-WORLD TESTS

Input Data

Carrington Rotation 2144
(11/21/2013 - 12/19/2013)
Magnetic field from HMI.



Coronal Model (A)

- Single PCG solve.
- Solver is faster with **PC2** (GPU uses **PC1**).

(A1) Default resolution: (101x92x182)
(A2) High resolution: (201x721x1441)

Coronal Model (B)

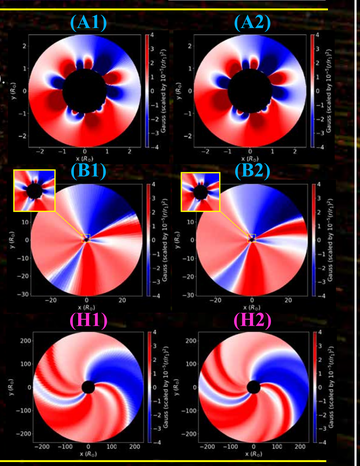
- Physical relaxation time: 48 hours.
- Solvers faster with **PC2** (GPU uses **PC1**).

(B1) Default resolution : (101x101x129)
(B2) MHDweb* resolution: (201x151x257)
*www.predsci.com/mhdweb

Heliospheric Model

- Uses solution from **(B1)** for inner BC.
- Physical relaxation time: 10 days.
- Solvers faster with **PC1**.
- Requires 2D boundary solves.

(H1) Default resolution: (141x111x129)
(H2) High resolution: (551x165x360)



Computational Environments

CPU

Compiler	Intel 18.0.2
MPI Library	Intel MPI 18.0.2
Instruction Set	AVX-512
CPU Model	Platinum 8160
Clock Rate	2.1 GHz
#CPUs x #Cores per Node	2 x 24
Total Memory Bandwidth	128 GB/s

TACC's Stampede2 Skylake

GPU

Compiler	PGI 18.10
MPI Library	OpenMPI 3.1.3
Driver/CUDA Version	410.48/10.0
GPU Model	V100
Clock Rate	1.38 GHz
#GPUs x #FP64 Cores per Node	4 x 2560
Memory Bandwidth per GPU	900 GB/s

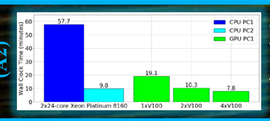
NVIDIA's PSG Test Cluster

TIMING RESULTS

Coronal Model (A)

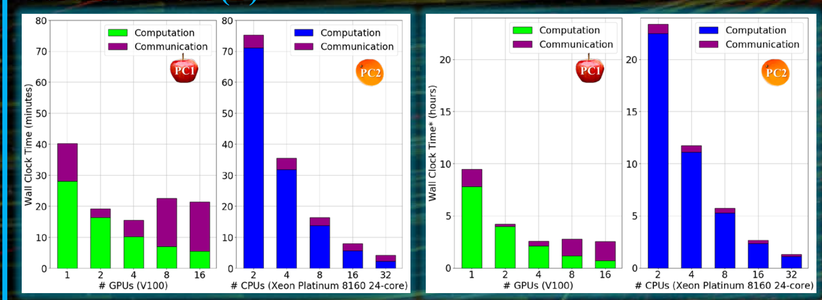
(A1) Solve completes in under 1 second on a 2x24-core CPU node (problem too small for efficient GPU acceleration)

(A2)



- CPU: **PC2** ~6x faster than **PC1**.
- For **PC1**, single V100 GPU ~3x faster than single CPU node.
- Multi-GPU server with 4xV100 faster than single CPU node.

Coronal Model (B)



(B1) ~ 1.3 million pts

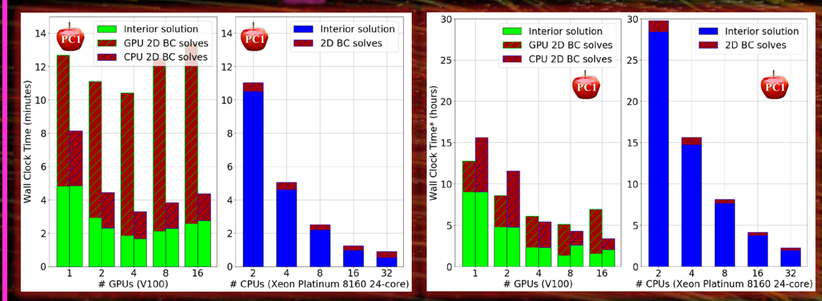
- Single V100 GPU ~2x faster than single two-socket CPU node.
- Multi-GPU exhibits poor inter-node scaling – likely due to lack of CUDA-aware MPI support on test cluster and small problem size.

(B2) ~ 7.8 million pts

- Larger problem size yields better GPU speedup and scaling but still exhibits poor multi-node scaling.
- Single 4xV100 server ~10x faster than single dual-socket CPU server!

* Timings extrapolated from running 1/6 of the total time steps

Heliospheric Model



(H1) ~ 2.0 million pts

(H2) ~ 32.7 million pts

- Boundary condition 2D solves too small for efficient GPU acceleration and creates load imbalance for both GPU and CPU runs.
- GPU runs decomposed in only theta-phi to avoid load imbalance - but this hurts many-GPU performance for the main algorithms.
- To try to avoid inefficient 2D solves on GPU runs, we test computing them on the CPU. For the smaller **(H1)** run, this drastically improves performance, but on the larger **(H2)** run, its effectiveness depends on the level of parallelism. Very few threads are used here (as we have 1 MPI rank per GPU), so future OpenACC features may alleviate this by switching to multi-core for the BC solves.
- Even with these difficulties, a single 4xV100 server performs ~3x faster than a single CPU server for **(H1)**, and ~6x faster for **(H2)**.

DISCUSSION

Summary: Larger problems are best for GPU performance. Achieving scaling and performance from the **Heliospheric Model** is challenging due to 2D boundary solves. Hardware/software support for direct CUDA-aware MPI is important for multi-GPU runs. OpenACC allows the code to remain single-source and portable. **Current results:** the tested single multi-GPU server always outperformed the single dual-socket CPU server (often several times faster).

Prospects for continually-running solar wind model: At present operational resolutions, all models run faster than real-time on a single CPU or GPU server. Adding real-time file processing and boundary driving will impact performance similar to the boundary challenges of the **Heliospheric Model**. For larger resolutions, the **Coronal Model (B)** runs only twice as fast as real-time on the single CPU server (less than ideal). Using a multi-GPU server improves this by a large factor. Runs requiring even larger resolutions would require the use of a multi-GPU system for single server real-time simulations.

Next steps: We are currently implementing GPU acceleration for **Coronal Model (C)**. As it is much more computationally intensive than all the other models in **CORHEL**, we expect it to benefit the most from a GPU implementation. We are also looking into ways to either avoid or speedup the 2D boundary solves in the models that currently require them.

Funding provided by
NASA NSF

Background image: NASA Goddard Media Studios