ISC Student Cluster Competition
HPC Application and Task Introduction: POT3D

Ronald M. Caplan
Computational Scientist
Predictive Science Inc.
caplanr@predsci.com

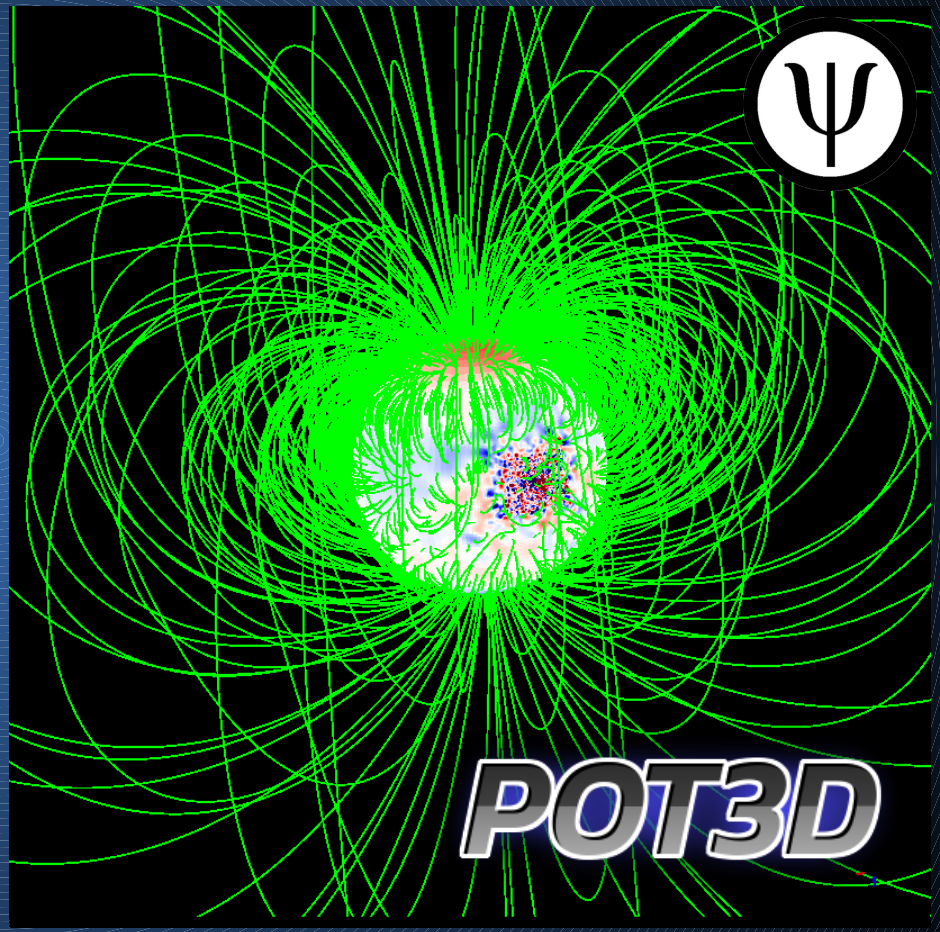
Predictive Science Inc.

ISC High Performance
STUDENT CLUSTER COMPETITION

HPC-AI
ADVISORY COUNCIL
NETWORK OF EXPERTISE

Outline

- POT3D
  - Usage in industry
  - Model
  - Numerical algorithm
  - Parallelism and Performance
- ISC23 task
- Cluster usage
- Tuning options
- Q&A



POT3D: Introduction

POT3D

- POT3D is a code that computes approximations of the solar coronal magnetic field (called potential fields) using observations of the solar surface magnetic field as a boundary condition
- The code is parallelized for use on CPUs and GPUs using MPI+OpenACC and StdPar (Standard Parallelism)
- The HDF5 file format is used for input/output

MPI

MORE SCIENCE
OpenACC
LESS PROGRAMMING

`do concurrent`

HDF

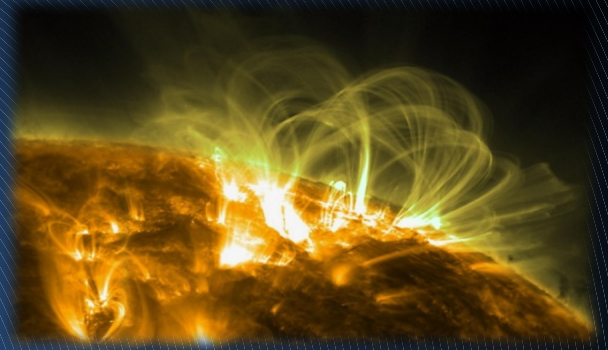
POT3D: Introduction

POT3D

- POT3D is included in the Standard Performance Evaluation Corporation's (SPEC) SPEChpc(TM) 2021 benchmark suite
- The current release is available on GitHub:
  `github.com/predsci/POT3D`
- Publications describing POT3D:
  - ***Variations in Finite Difference Potential Fields***
    Caplan, et. al., Ap.J. 915,1 (2021) 44
  - ***From MPI to MPI+OpenACC: Conversion of a legacy FORTRAN PCG solver for the spherical Laplace equation***
    Caplan, et. al., arXiv:1709.01126 (2017)

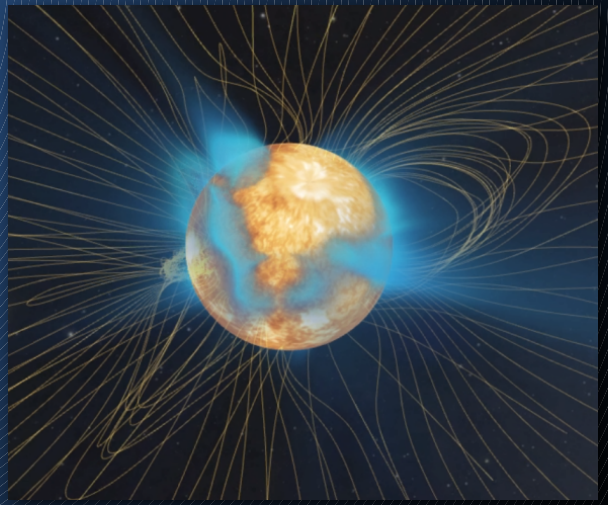
POT3D: Usage in Industry

- Potential field approximate solutions to the solar coronal magnetic field are used widely in the field of Solar/Heliophysics and space weather
- They are inexpensive to compute compared to full Magnetohydrodynamic models (MHD), but still can provide a fairly good approximation to the coronal magnetic fields
- The solutions are often used as initial conditions for large MHD codes
- POT3D is the potential field solver for the Wang-Sheeley-Arge (WSA) empirical solar wind model within the Corona-Heliosphere (CORHEL) software suite hosted at NASA's Community Coordinated Modeling Center (CCMC)
- It will also be used in an upcoming state-of-the-art open-source space weather modeling package as part of the joint NASA-NSF "Next Generation Software for Data-driven Models of Space Weather with Quantified Uncertainties" program



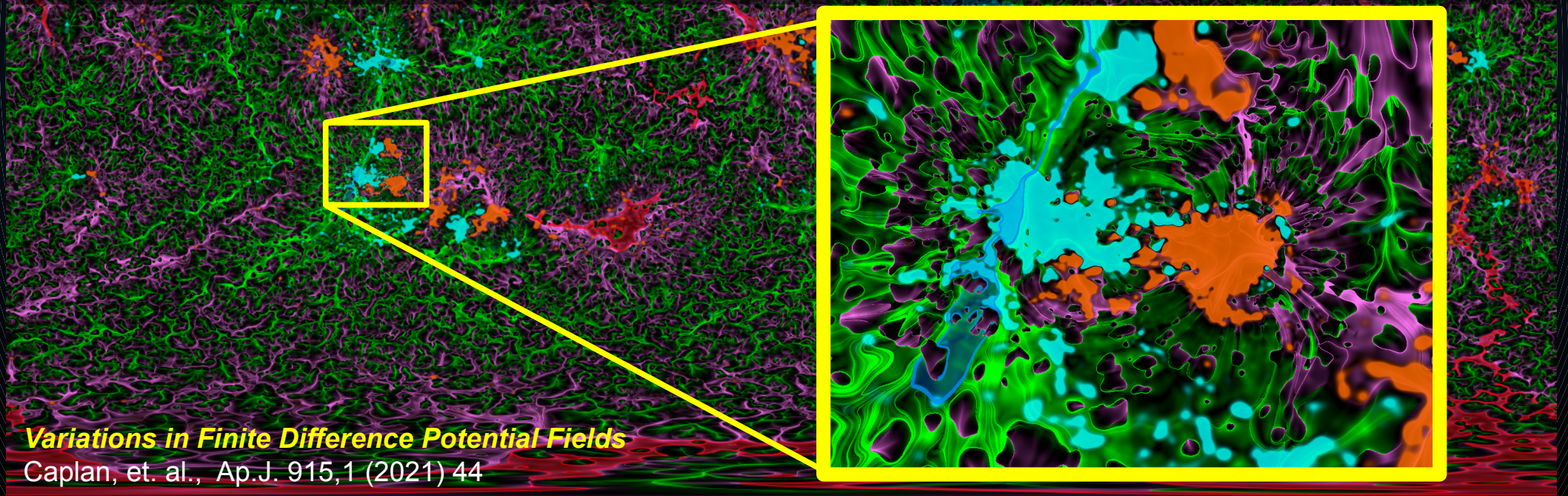
NASA

COMMUNITY COORDINATED MODELING CENTER



POT3D: Usage in Industry

- Potential field computations are fairly inexpensive at modest resolutions
- Due to this, it opens the door to computing solutions at very high resolutions
- At these high resolutions, the solution becomes much more computationally expensive, requiring a parallelized high-performance code
- POT3D provides this performance and has been used to compute the largest ever potential field solution (as of this writing) – 6.6 *billion* grid points



Variations in Finite Difference Potential Fields
Caplan, et. al., Ap.J. 915,1 (2021) 44

POT3D: Model

Maxwell equation: $\nabla \times \vec{B} = \mu_0 \vec{J}$

Assume no current: $\vec{J} \rightarrow \nabla \times \vec{B} = 0$

Solution form: $\vec{B} = \nabla \Phi$

Divergence-free condition: $\nabla \cdot \vec{B} = 0$

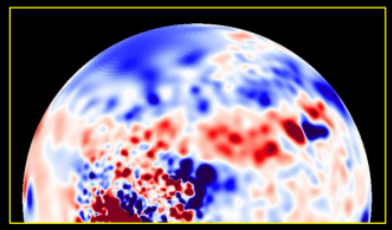
$$\boxed{\nabla^2 \Phi = 0}$$

Laplace equation

Boundary Conditions

$r = R_\odot$ Observed Surface

$$\left.\frac{\partial \Phi}{\partial r}\right|_{R_\odot} = B_r\Big|_{R_\odot}$$

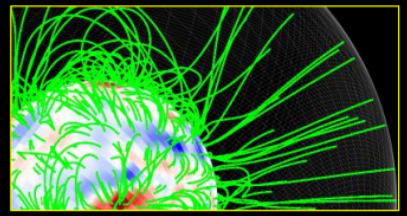


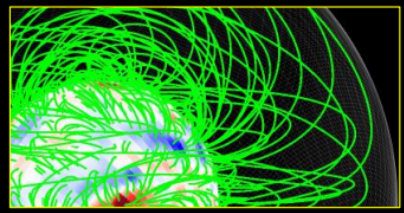
$r = R_1$ (a) Source Surface

$$\Phi|_{R_1} = 0$$

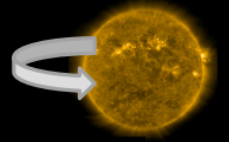
(b) Closed Field

$$\left.\frac{\partial \Phi}{\partial r}\right|_{R_1} = B_r\Big|_{R_1} = 0$$



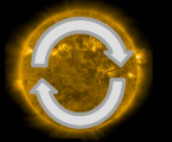


$\phi = 0, 2\pi$ Periodic $\Phi|_{\phi=0} = \Phi|_{\phi=2\pi}$



$\theta = 0, \pi$ Polar average

$$\Phi|_{\theta_0} = \frac{1}{2\pi}\int_{\phi=0}^{\phi=2\pi} \Phi|_{\theta_0 \pm \epsilon}\, d\phi$$

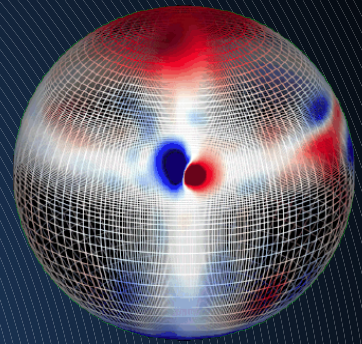


POT3D: Numerical Method

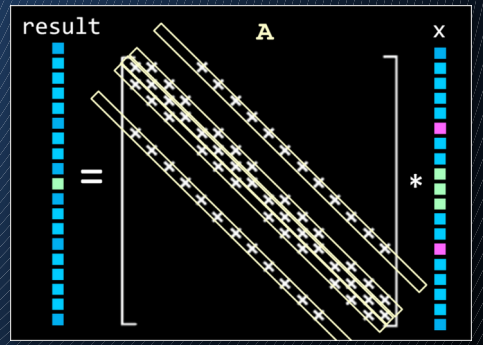
- Solves the 3D Laplace equation on a non-uniform logically-rectangular spherical grid
- The Laplacian operator is discretized using 2nd-order finite difference
- The coefficients for all grid points' local stencils are stored as a sparse 7-banded matrix in a modified DIA storage format
- The equation is then solved using an iterative Preconditioned Conjugate Gradient (PCG) solver, consisting of array operations (axpy), matrix-vector products, and dot products

$$\nabla^2 \Phi = 0$$

$$\mathbf{A}\,\vec{x} = \vec{b}$$

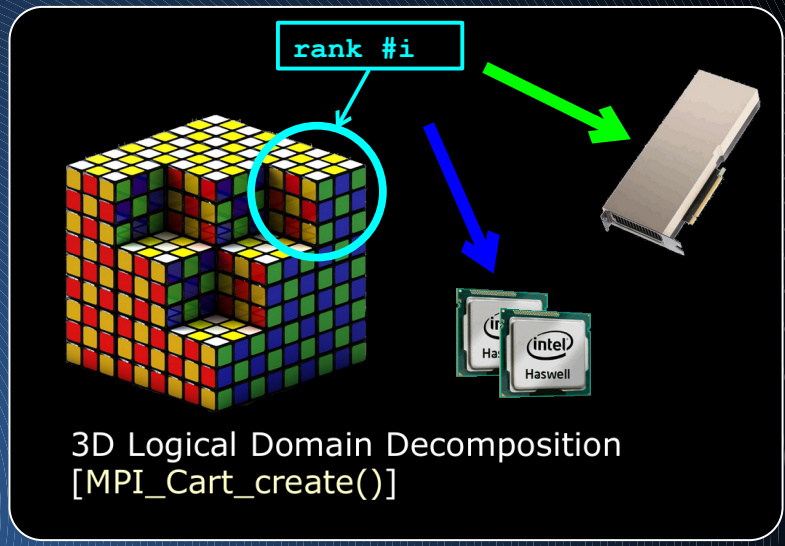


```
do j=2,ny-1
  do i=2,nx-1
    result(i,j) =   A(1,i,j)*x(i  ,j-1)
                  + A(2,i,j)*x(i-1,j  )
                  + A(3,i,j)*x(i  ,j  )
                  + A(4,i,j)*x(i+1,j  )
                  + A(5,i,j)*x(i  ,j+1)
  enddo
enddo
```



POT3D: Parallelism

- The logical grid is broken up into 3D blocks split as evenly as possible across all MPI ranks
- For operations that require neighbors (matrix-vector products), asynchronous point-to-point MPI communication is used (iSend/iRecv)
- For dot products and other collectives, global MPI "Allgather" routines are used
- Each MPI rank's local block of grid points computed by
  - 1 CPU thread (MPI-only)
  - 1 GPU (multi-GPUs)
    `-stdpar=gpu -acc=gpu -gpu=ccall,nomanaged`
  - Many CPU threads (hybrid-CPU)
    `-stdpar=multicore -acc=multicore -mp`
    Uses: `OMP_NUM_THREADS & ACC_NUM_CORES`
- The local block parallelism is achieved through the use of Fortran's standard parallelism (DC) along with OpenACC for loops that are not yet supported with DC as well as manual GPU-CPU data movement



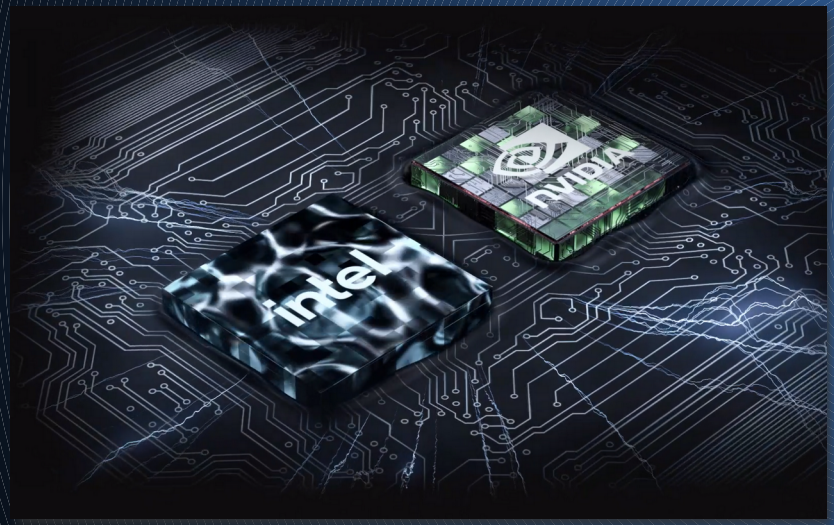
3D Logical Domain Decomposition
[MPI_Cart_create()]

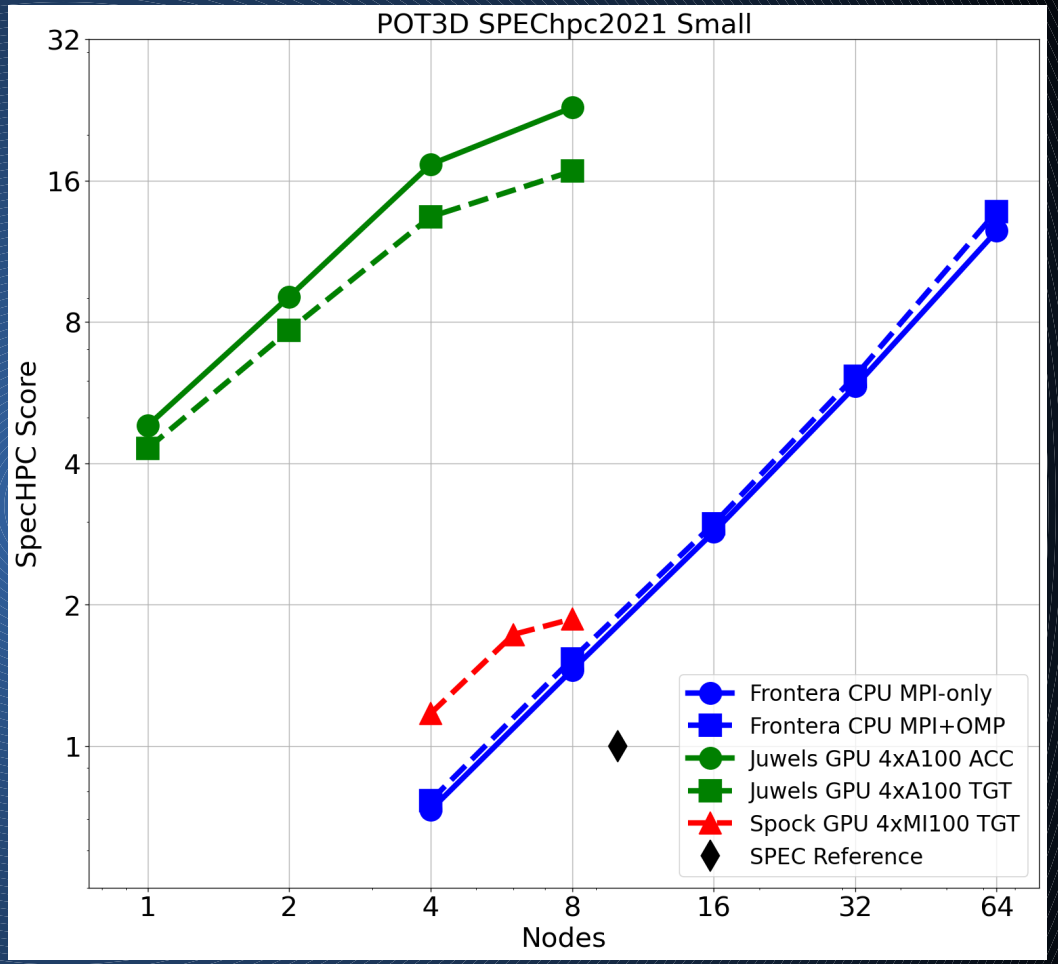
```
do concurrent (i=1:N)
  y(i) = a*x(i) + y(i)
enddo
```

```
!$acc parallel default(present)
!$acc loop
  do i=1,n
    y(i) = a*x(i) + y(i)
  enddo
```

POT3D: Performance

- POT3D scales well to many MPI ranks on both CPUs and GPUs





SPEChpc 2021 “Small” test (300 million points)
Brunst et. al. (2022)

POT3D: Performance

POT3D is *highly* memory bandwidth bound

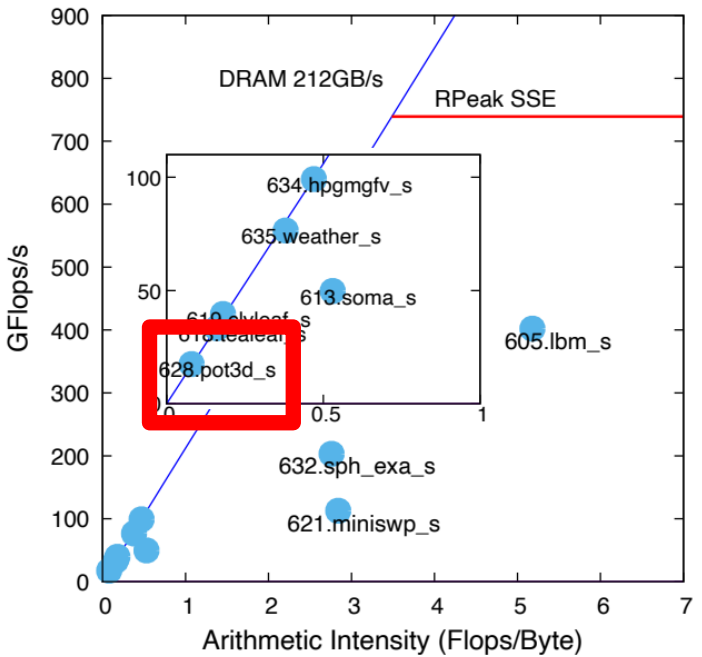
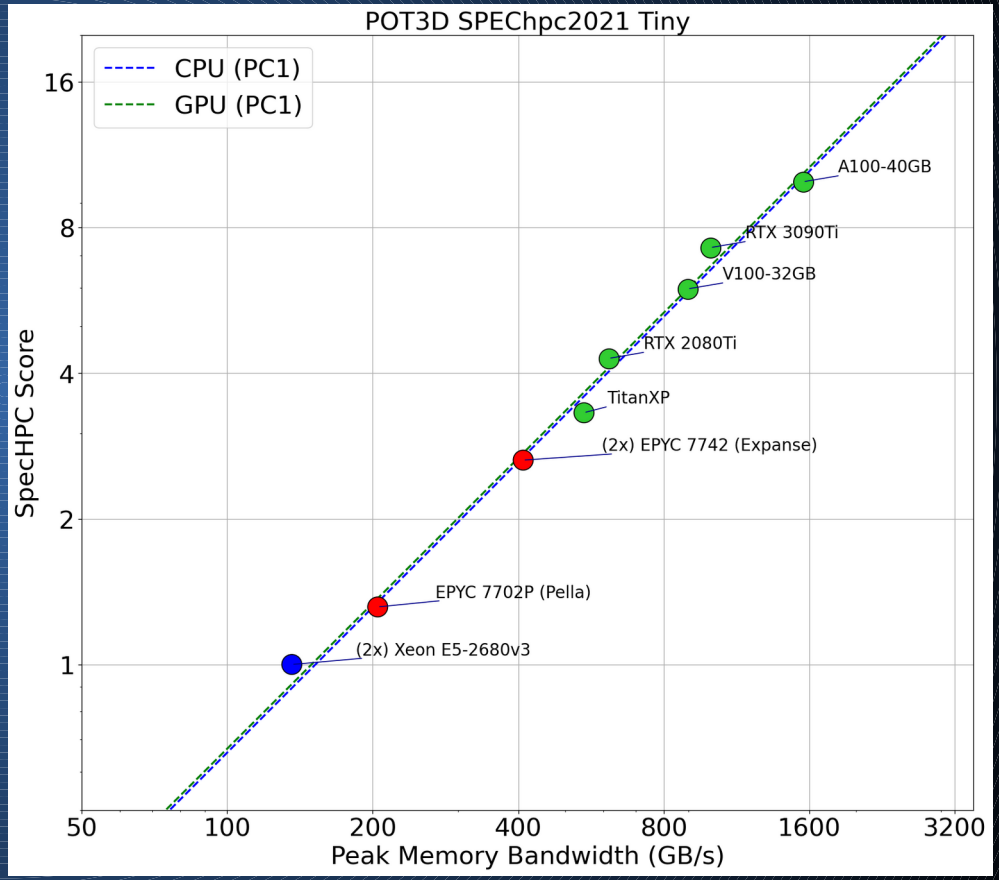


Fig. 4: Roofline plot for the *small* suite. Data collected for MPI-only versions using 4 nodes (224 ranks on Frontera). The roofline plots for the tiny, medium, and large suites are similar. Arithmetic intensity and memory bandwidth are collected for the entire duration of each program.



Brunst et. al. (2022)

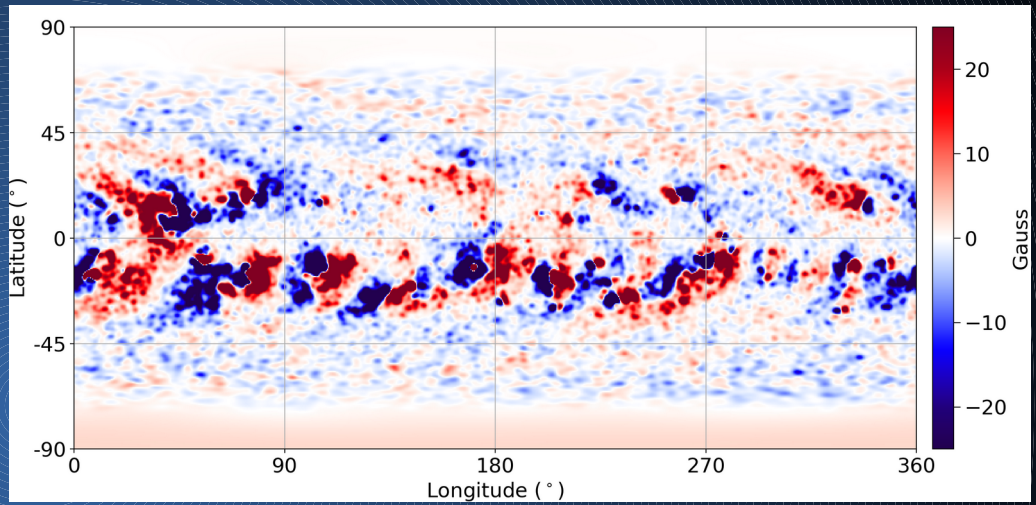
ISC23 SCC Task: Test Problem

- "ISC2023" – Test derived from the "small" POT3D benchmark run from the SPEChpc™ 2021 benchmark Suite

- Reference CPU timing on 10 (2x) 12-core Intel Xeon E5-2680v3 nodes (Total of 240 cores and 680 GB/s memory bandwidth): 1,650 seconds

- In POT3D GitHub, the test is located in: `/testsuite/isc2023`

Input surface radial magnetic field map



	NR	NT	NP	Total Points
Resolution	325	450	2,050	**~300 million**

Solver Iterations	**25,112**

ISC23 SCC Task

Task and Submission

1. **Use the input under `testsuite/isc2023` folder.**
2. **Run POT3D with `isc2023` input on both PSC bridges-2 and FAU Fritz CPU clusters using 4 nodes.**
   Experiment with number of ranks per socket/numa domains to get the best results.
   Your job should converge at 25112 steps and print outputs like below:

```
### The CG solver has converged.
Iteration:    25112   Residual:   9.972489313728662E-13
```

3. **Profile a run.**
   a. Use any of the remote clusters to run an MPI profile (such as IPM profile or any other profiler) for a run using 4 nodes with full PPN.
   b. Submit the profile as a PDF to the team's folder.
   c. In your presentation, also indicate the 3 main MPI calls that are being used and their run times, as well as the total MPI time for the test.
4. **Bonus task: Run POT3D on the PSC cluster using the V100 GPU partition.**
   a. Use only 4 GPUs for the run. It is recommended to use one rank per GPU.
   b. Submit the results to the team's folder.
   c. NOTE: To compile and run POT3D with the `nvfortran` compiler, you must load and/or build the HDF5 library compiled with `nvfortran`. The code is known to work with HDF5 1.8.21 (http://portal.hdfgroup.org/display/support/HDF5+1.8.21)
   d. An example build script for POT3D with the NVIDIA compiler can be found in `build_examples/build_gpu_nv22.3_ubuntu20.04.sh`
   e. Note that you do NOT need to enable the `cusparse` option because the test case is not set up to use the algorithm that requires `cusparse`. Therefore, if linking `cusparse` is causing difficulties, you can change the build script line `POT3D_CUSPARSE=1`to `POT3D_CUSPARSE=0`.
5. **Submission and Presentation:**
   - Submit all your build scripts, run scripts, inputs, and output text files (pot3d.dat, pot3d.out, timing.out, etc.)

Cluster Usage

- PSC Bridges 2

  https://hpcadvisorycouncil.atlassian.net/wiki/spaces/HPCWORKS/pages/2977562625/ISC23+SCC+Getting+Started+with+Bridges-2+Cluster

- FAU Fritz

  https://hpcadvisorycouncil.atlassian.net/wiki/spaces/HPCWORKS/pages/2977628161/ISC23+SCC+Getting+Started+with+FAU+Clusters

NHR FAU

Tuning Options

- Each MPI rank's local block of grid points computed by
  - 1 CPU thread (MPI-only)
  - 1 GPU (multi-GPUs)
    `-stdpar=gpu -acc=gpu -gpu=ccall,nomanaged`
  - Many CPU threads (hybrid-CPU)
    `-stdpar=multicore -acc=multicore -mp`
    Uses: `OMP_NUM_THREADS & ACC_NUM_CORES`
- Typically on CPUs, running with MPI-only is best
- For very large numbers of CPUs, running in hybrid-CPU mode can be faster. Correctly setting up the core/thread/node/socket affinity (both in the job launch script and in the environment variables) can be difficult
- Can experiment with the number of MPI ranks per node/numa/socket, and their affinity. For AMD EPYC CPUs (which have 4 NUMA domains), it is often better to use less than the maximum cores per socket
- Best performance for on-sight ISC23 cluster will be on multiple NVIDIA GPUs. Here, the maximum memory bandwidth of the GPUs is critical to performance (more=better!)

Q & A