PPPL PRINCETON PLASMA PHYSICS LABORATORY

PRINCETON UNIVERSITY

On the Road to Code Portability

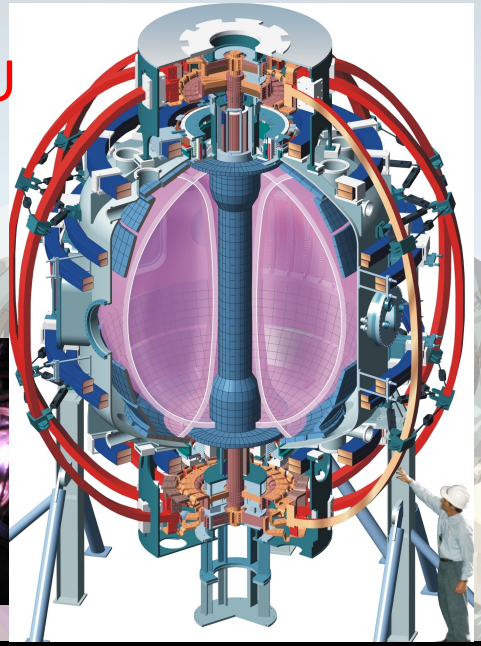
Stéphane Ethier

Princeton Plasma Physics Laboratory

OpenACC Summit 2021

September 14, 2021

NSTX-U

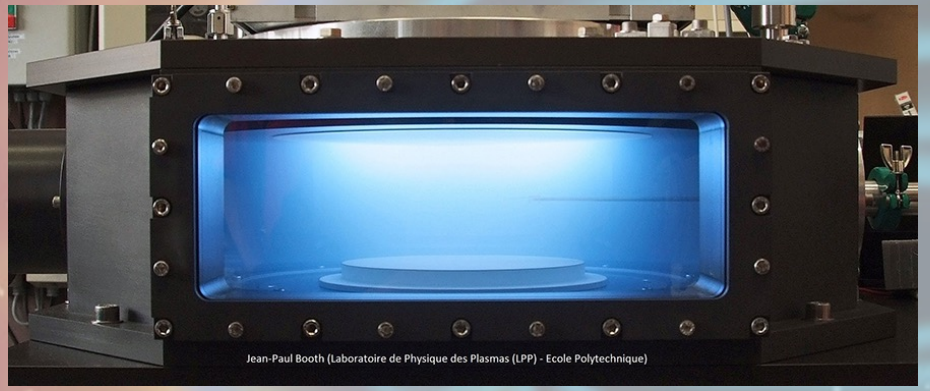


What we do at PPPL: *Magnetic Confinement Fusion*

- This is our main mission (https://www.pppl.gov/about/pppl-glance)
- Deuterium plasma heated to 100 Million degrees!
- Device wall must be kept at a few 100s degrees
- Strong magnetic field confines the hot plasma
- Large gradients of temperature and density lead to interesting physics that can be studied with:
  - Magnetohydrodynamics codes, for fluid-like plasma and magnetic field evolution
  - Kinetic codes, Particle-in-Cell (PIC) or mesh-based, for velocity-dependent physics, such as turbulence and wave-particle resonances
  - Specific to fusion community: **Gyrokinetics!**

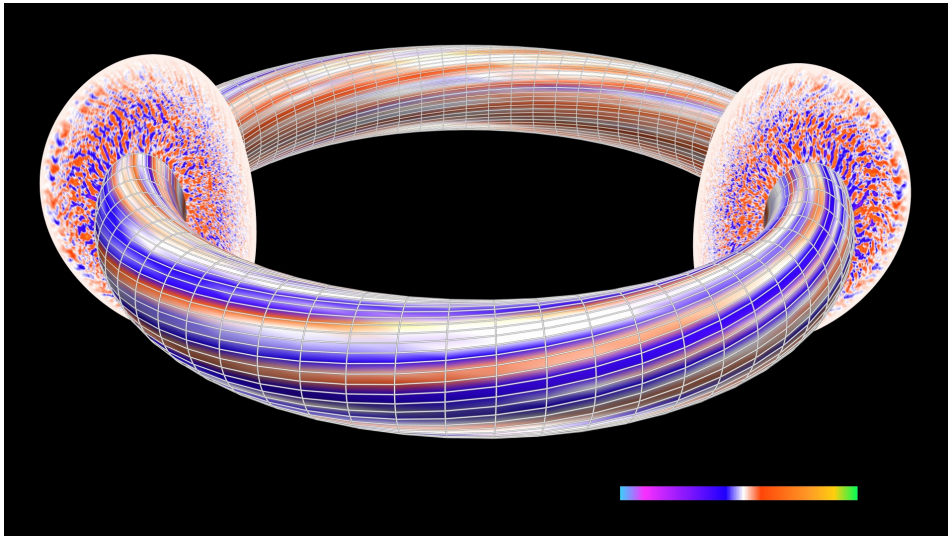
Plasma Science and Technology

- We study pretty much everything related to plasmas
  - Space plasmas
  - heliophysics
  - Plasma propulsion
  - **Industrial plasmas**
- Plasma processes in microchips fabrication is a hot topic!
- There is a strong need for 6D PIC codes that can simulate these processes with all the chemistry that is involved



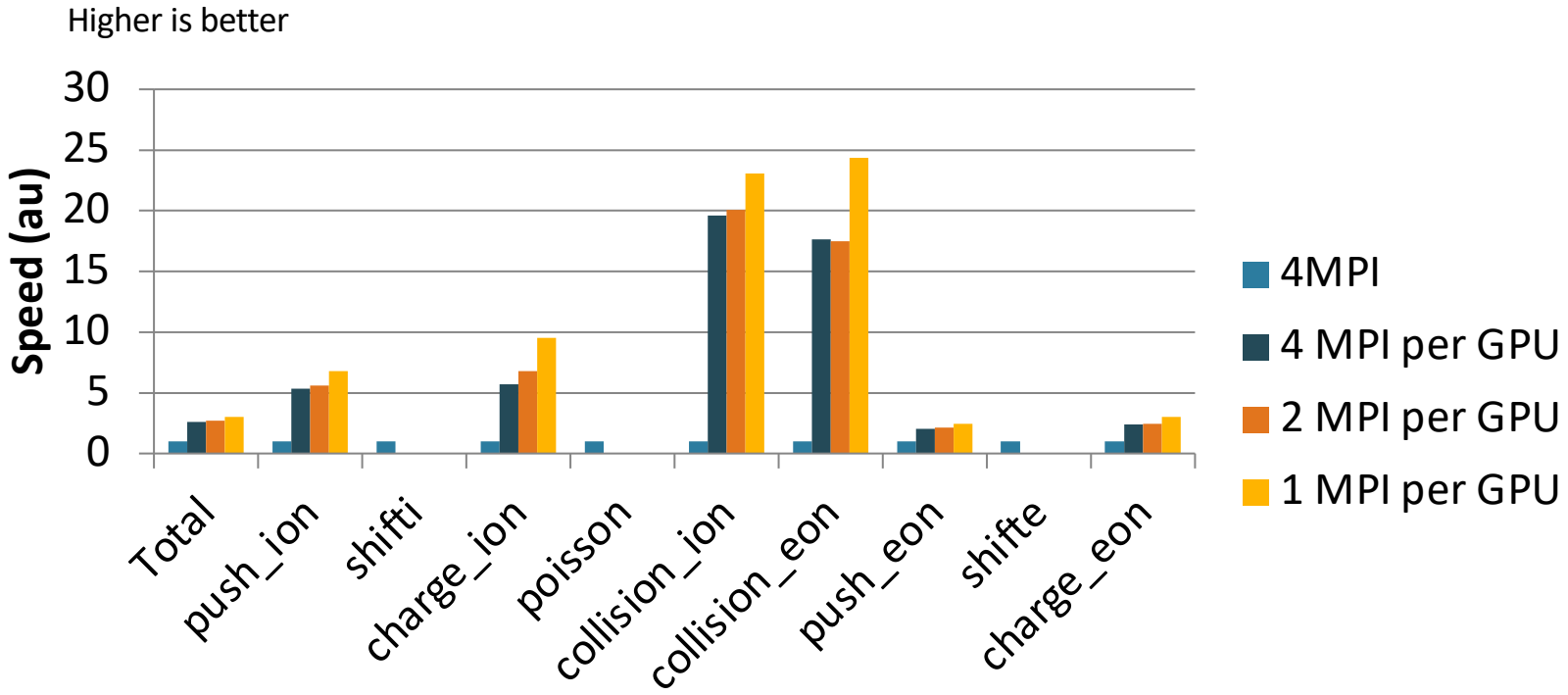
The Gyrokinetic Tokamak Simulation code (GTS)

- Ported to GPU using OpenACC during the June 2019 Princeton Hackathon
- 5D gyrokinetic PIC code
- To study microturbulence in the “core” plasma of tokamaks
- ~10,000 lines of FORTRAN + some C
- MPI + OpenMP parallelism
- All particle routines now running on GPU (most time-consuming parts)
- Charge accumulation (memory scatter operation) on the grid requires fast atomic updates on GPU



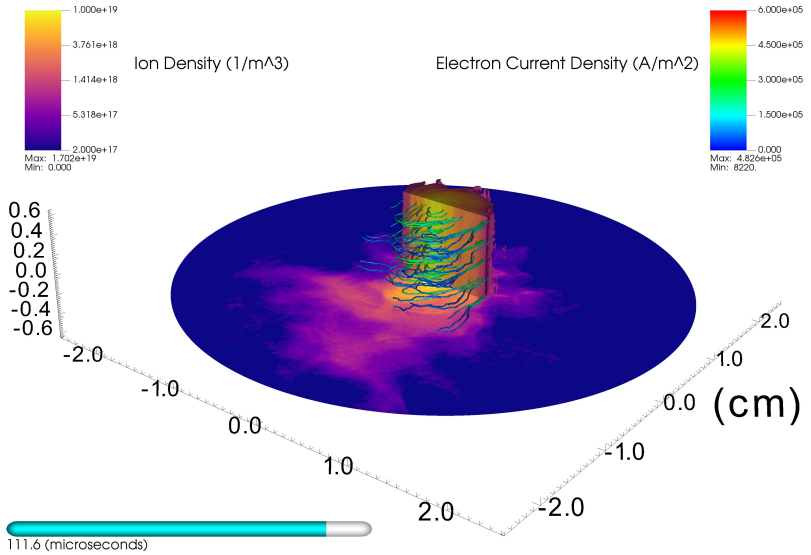
Weixing Wang (P.I.) Stéphane Ethier, Chenhao Ma, Min-Gu Yoo, Ed Startsev, Reuben Budiardja (mentor, ORNL) and inputs from Mathew Colgrove (Nvidia)

GTS Speed up on P9+V100 (Traverse)



Low Temperature Plasma PIC (LTP-PIC)

- Ported to GPU with OpenACC during the June 2020 Princeton Hackathon
- Full 6D PIC code to study low temperature plasmas for industrial applications (including propulsion)
- Written in C with MPI+OpenMP hybrid parallelism
- Also highly dependent on fast atomic operations on GPU
- Ported *Mersenne Twister* pseudo-random number generator to GPU
- Studied GPU performance of *Hypre* solver



Team Members: Andrew Tasman Powis (Princeton U./PPPL), Johan Carlsson (Radiasoft LLC), Alex Khanales (PPPL), Arjun Agarwal (PPPL summer student)

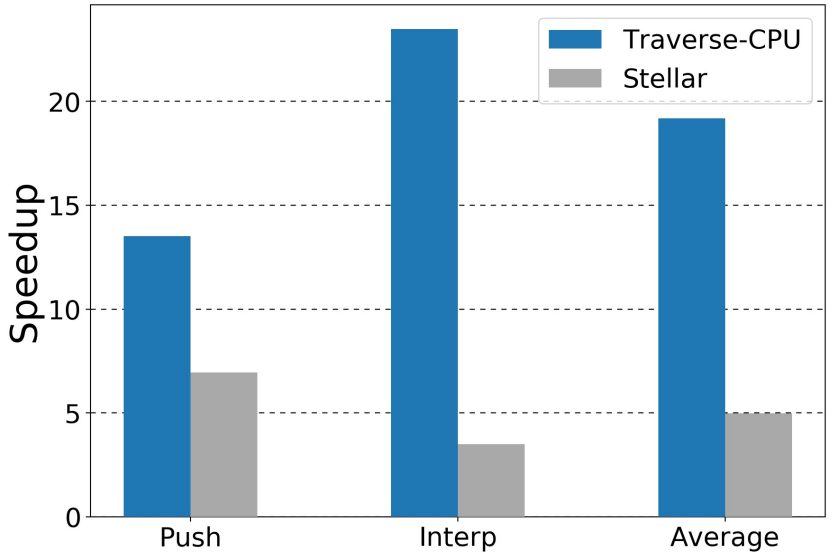
Mentors: Stéphane Ethier (PPPL), Mathew Colgrove (NVIDIA), Mozhgan Chimeh (NVIDIA)

LTP-PIC Particle Routine Acceleration with GPUs

Node to node comparison

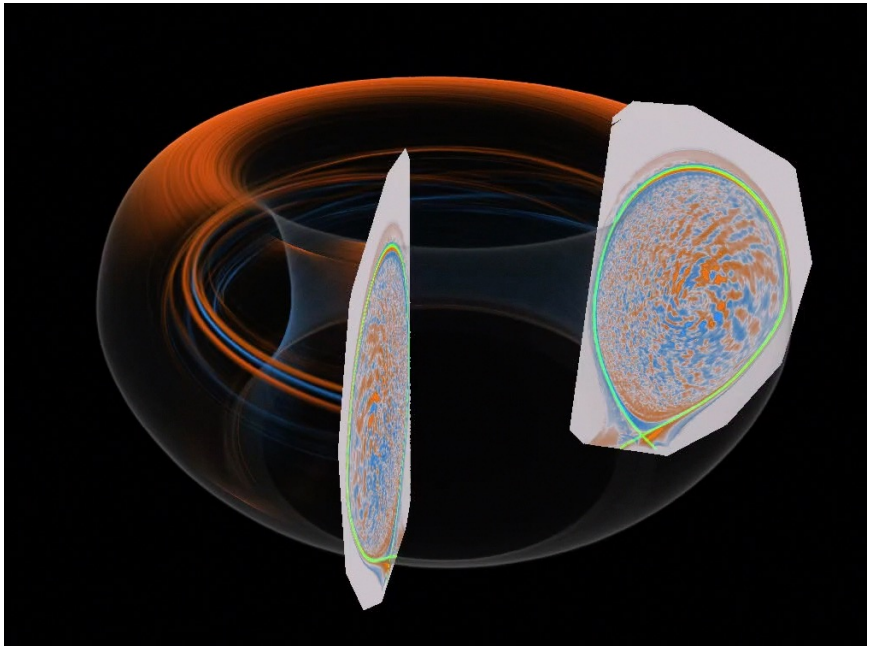
Relative speedup of particle routines with 4x NVIDIA V100 GPUs on Traverse against:

- Traverse CPU: 32 cores/node IBM POWER9s
- Stellar: 96 cores/node Intel Cascade Lakes (Xeon 6248R)



Our flagship exascale-worthy code XGC

- 5D, full-device gyrokinetic PIC code to study **edge** physics in tokamaks
- The simulation domain can include the magnetic separatrix, magnetic axis and the biased material wall
- Requires 10 to 100X more particles than delta-f “core” plasma codes
- Originally written in Fortran 90
- Uses a multi-level MPI + OpenMP hybrid parallel algorithm on CPU
- Always the first PPPL code ported to new HPC hardware



C.S. Chang (PI), S. Ku, R. Hager, A. Scheinberg, R.M. Churchill, S. Ethier, B. Sturdevant, A. Mollen, A. Sharma, S. Klasky, J. Choi, E. D’Azevedo, S. Sreepathi, M. Adams, M. Shephard

XGC during the *"Titan"* years

- Original GPU port of XGC on Titan used CUDA Fortran (particle routines) and OpenACC (collisions)
- Implemented by then postdoc Stephen Abbott at OLCF, Eisung Yoon at RPI (collisions), Ed D'Azevedo, Pat Worley (ORNL), Mark Adams (LBNL), S-H Ku, R. Hager, S. Ethier, J. Lang (PPPL)
- On each Titan node: 1 MPI task, 16 OpenMP threads, 1 K20X, 6GB GDDR5
- Concurrently pushed electrons (for 40-60 subcycling steps) on GPU and ions on CPU
- In the collision routine each OpenMP thread would launch a GPU kernel on a different stream using OpenACC *"async(stream)"*

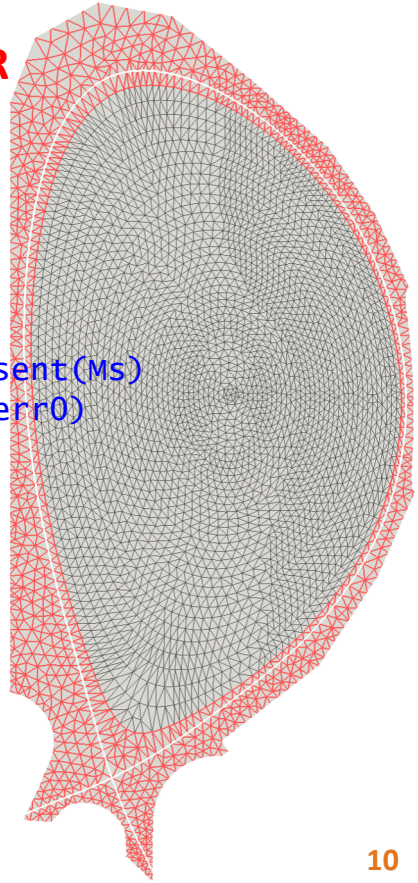
Mixing OpenMP on CPU and OpenACC

```
nthreads = omp_get_max_threads()
!$omp parallel do private(ith,node,…) num_threads(nthreads)
  do ith=1,num_mesh_pts
     call f_collision_single_sp_body(ith, grid, st, df, …)
  enddo
  ...

ithread = omp_get_thread_num(); istream = ithread + 1
!$acc enter data create(Ms) async(istream)
!$acc  kernels async(istream) &
!$acc& pcopyin(mesh_Nvr1,mesh_Nvz1,cs_mesh_r_half,tmp_vol,mesh_dz) present(Ms)
!$acc loop independent  collapse(2) gang private(k_eff, EK, EE, vpic_ierr0)
    do index_J=1, mesh_Nvr1
        do index_dz=0, mesh_Nvz1-1
!$acc loop independent vector
            do index_jp=1, mesh_Nvr1
                r=cs_mesh_r_half(index_J)
                ...
            enddo  ! index_jp
        enddo  !index_J_dz
    enddo
!$acc end kernels
!$acc wait(istream)
```

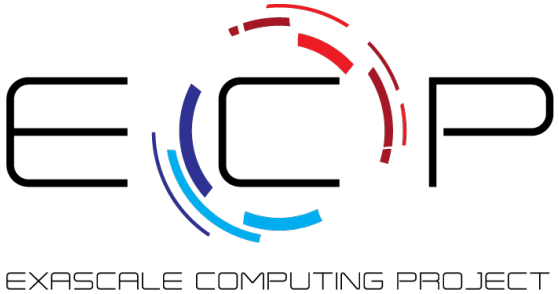
LOOP OVER MESH POINTS

LOTS OF WORK!! Loops over velocity grid



Upcoming exascale systems forced us to rethink our approach

- XGC is the main code in the **ECP-WDMApp** project
- It needs to run on the exascale systems on Day 1
- CUDA Fortran and OpenACC *"not"* available
- Decision: Let somebody else figure out the details
  - We opted for **Kokkos** as the interface to the various hardware
  - Kokkos has many different GPU backends: CUDA, HIP, SYCL, OpenMPTarget, …
  - Using **Cabana** library from ECP-CoPA project (https://github.com/ECP-copa/Cabana)
  - We are rewriting the code in C++ to avoid complex Fortran interfaces (carried out by Aaron Scheinberg, *Jubilee Development*)



Research supported by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy's Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation's exascale computing imperative.

For our OpenACC FORTRAN and C codes OpenMP target offload seems to be the most portable solution

- Would love to keep using OpenACC!!!
- Compiler developers seem to be putting most of their efforts on OpenMP target offload though
- Implementations are not stable yet and good performance is hard to achieve
  - **GEM** Fortran code (also part of ECP-WDMApp) moving to OpenMP
- On top of it, use of the new *"descriptive"* model is being encouraged (mainly by Nvidia and NERSC right now…)
  - Similar to choice between `!$acc kernels` and `!$acc parallel`

Let's hope that the other vendors/compiler developers will follow suit!

PRESCRIPTIVE

```
#ifdef TARGET_GPU
#pragma omp target teams distribute
reduction(max:error)
#else
#pragma omp parallel for reduction(max:error)
#endif
for( int j = 1; j < n-1; j++) {
#ifdef TARGET_GPU
#pragma omp parallel for reduction(max:error)
#endif
 for( int i = 1; i < m-1; i++ ) {
    Anew[j][i] = 0.25f * ( A[j][i+1] +
A[j][i-1] + A[j-1][i] + A[j+1][i]);
    error = fmaxf( error, fabsf(Anew[j][i]-
A[j][i]));
 }
}
```

DESCRIPTIVE (more freedom to the compiler)

```
#pragma omp target teams loop
reduction(max:error)
for( int j = 1; j < n-1; j++) {
#pragma omp loop reduction(max:error)
  for( int i = 1; i < m-1; i++ ) {
     Anew[j][i] = 0.25f * ( A[j][i+1] +
A[j][i-1] + A[j-1][i] + A[j+1][i]);
     error = fmaxf( error, fabsf(Anew[j][i]-
A[j][i]));
  }
}
```

Can make a big difference in performance!

https://docs.nvidia.com/hpc-sdk/compilers/hpc-compilers-user-guide/index.html#openmp-loop

Last words

- Have we achieved code portability?
  - Not quite but we're trying…
  - New hardware always brings new challenges to implementers/compiler developers
- For both OpenACC and OpenMP, ***specification*** does not equal ***implementation***!
- "**omp target teams loop**" seems like a good way to give flexibility to the compiler developers to implement to best performing version of OpenMP offload directives
- I think it's time for the languages themselves to be serious about supporting parallelism and offloading to GPU. Implementation is lagging though.
  - Fortran2008 "**DO CONCURRENT**"
  - DPC++ → C++20xx?
- Kokkos is a good alternative for C++ code but OpenACC and OpenMP still great for FORTRAN and C

Thank you!

Work supported by the U.S. Department of Energy, Office of Science, Office of Fusion Energy Sciences, and has been authored by Princeton University under Contract Number DE-AC02-09CH11466 with the U.S. Department of Energy.