3rd Annual Meeting of EUROfusion HPC ACHs

Status of BIT1 Porting to GPU and Progress on X-tork

Xavier Sáez

Eduardo Cabrera

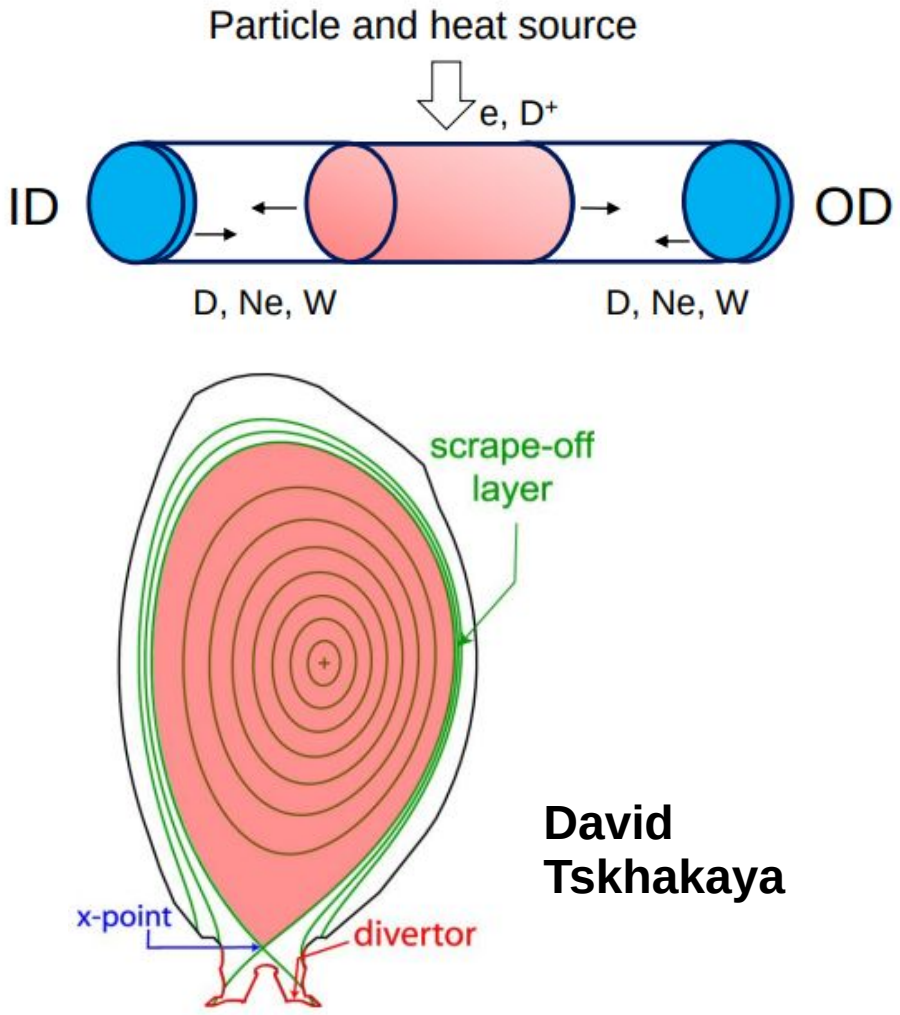
ACH@CIEMAT-BSC

This work has been carried out within the framework of the EUROfusion Consortium, funded by the European Union via the Euratom Research and Training Programme (Grant Agreement No 101052200 — EUROfusion). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the European Commission can be held responsible for them.

BIT1

Xavier Sáez

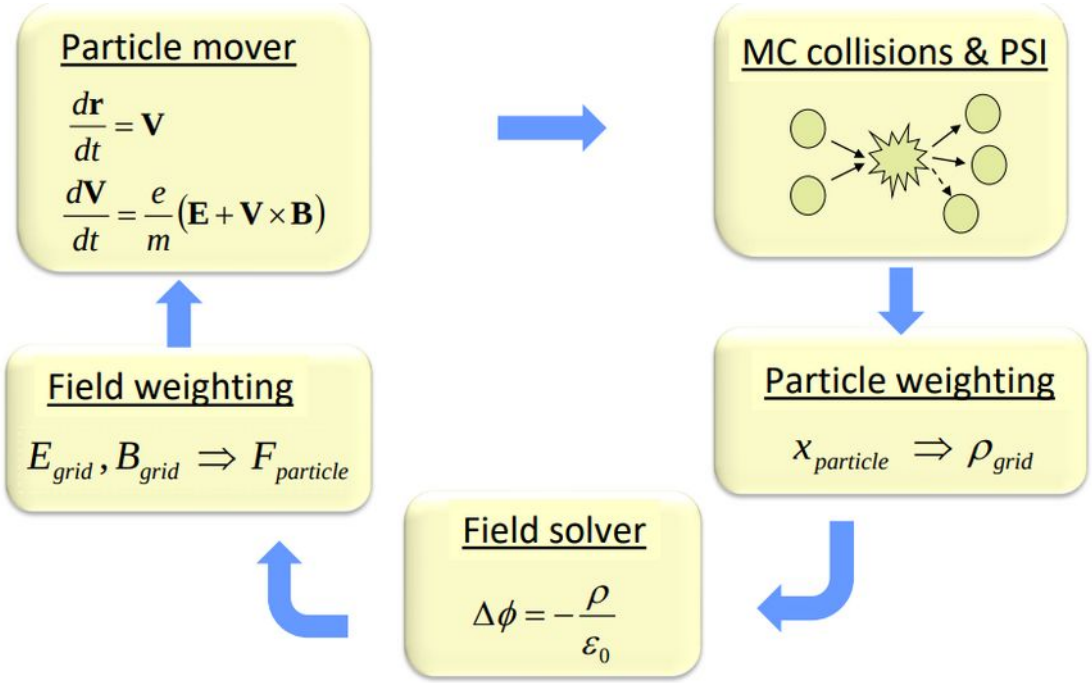
BIT1: Introduction



David Tskhakaya

- BIT1 is an electrostatic 1D3V PIC + 2D3V direct Monte Carlo code for plasma simulations.
- **Input:** Particle, heat sources, geometry and divertor material
- **Simulated particle species:** El, D (H, T), Li, He, Be, O, …
- Highly accurate → $5x10^6$ cells →capability to simulate whole SOL
- **Language:** C
- **Libraries**: No external libs
- **Parallelization**: MPI using domain decomposition

BIT1: Algorithm



- BIT1 is a **Particle-in-cell (PIC) code**

BIT1: Project Goal

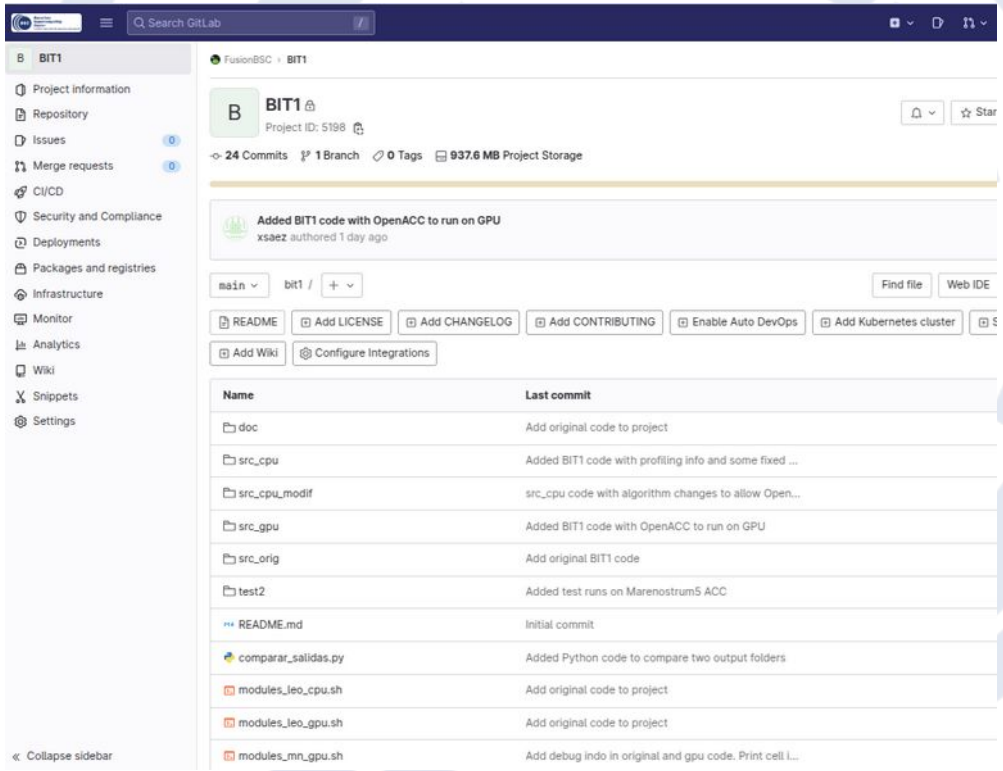
- **Main objective:** port BIT1 to GPUs using OpenACC.
  - Ensure maintainability and readability of the codebase.
  - Achieve acceleration by offloading costly routines to GPUs.

- **Sub-objectives:**
  - Introduce OpenACC directives minimally invasively.
  - Preserve usability for non-GPU experts.
  - Measure performance vs MPI-only CPU version.
  - Document issues and propose next steps.

BIT1: Execution Environment

- **Platforms used:**
  - Leonardo (A100 GPUs, HDR IB)
  - MareNostrum 5 (H100 GPUs, NDR IB)
  - Pitagora (incomplete GPU ecosystem)
- **Test case: DCSM_0_e21_c8.inp**
  - 1D bounded plasma with electrons, D^+ ions, and neutrals.
  - Domain: 0.1 m, 100,000 cells.
  - Time step: 4e-14 s, 1,000 iterations.
  - Each job used 4 MPI ranks (one per GPU), and Each MPI task is explicitly bound to a dedicated GPU.
- **Profiling tools:**
  - NVTX for instrumentation.
  - Nsight Systems for GPU/CPU trace analysis.

- **GitLab Repository: FusionBSC/BIT1**
  - All codes, tests, results and traces are available.



BIT1: Preliminary Work

- **Environment setup:**
  - Creation of module-loading scripts per platform.
- **Cleanup of unused variables to remove warnings.**

```
picc  -c -O2 -noacc  bincc.c
"bincc.c",   line   13:   warning:   variable   "iisp"   was   declared   but   never   referenced
[declared_but_not_referenced]
   int        isp, iisp, i, s, p1, p2, nnp, k, sid, sod;
                     ^
```

- **Fixing segmentation fault caused by long file paths.**

```
Can't find input file /gpfs/projects/bsc21/bsc021331/BIT1/bit1-feature-GPU-OpenACC/tests/../BIT1_c8/at@N
Can't find input file /gpfs/projects/bsc21/bsc021331/BIT1/bit1-feature-GPU-OpenACC/tests/../BIT1_c8/at@@eH
```

- **Preparation for GPU port:**
  - Ensuring build consistency across supercomputers.

BIT1: CPU Version

- **Reference baseline performance:**
  - **Original BIT1 CPU:** runtime ~244.5 s.
  - **Instrumented CPU version:** runtime ~249 s (minimal overhead).
    step time ~260 ms
- **Modifications for GPU preparation:**
  - Inlining routines for GPU compatibility.

```
void chist_N()
{
 int     isp;

 for (isp=0; isp<nsp; isp++)
 {
   np_hist[isp][hist_hi2] = fnp(isp);

 }    /* end of for (isp=0...*/
}  /* END of chist_N() */
```

```
void chist_N()
{
 int     isp;
 int s, j;

 for (isp=0; isp<nsp; isp++)
 {
   //   np_hist[isp][hist_hi2] = fnp(isp);
   int j= 0;
   for (s=0; s<nc; s++)
     j += np[isp][s];
   np_hist[isp][hist_hi2] = j;
 }    /* end of for (isp=0...*/
}  /* END of chist_N() */
```

  - Replacing complex array accesses with local vars.

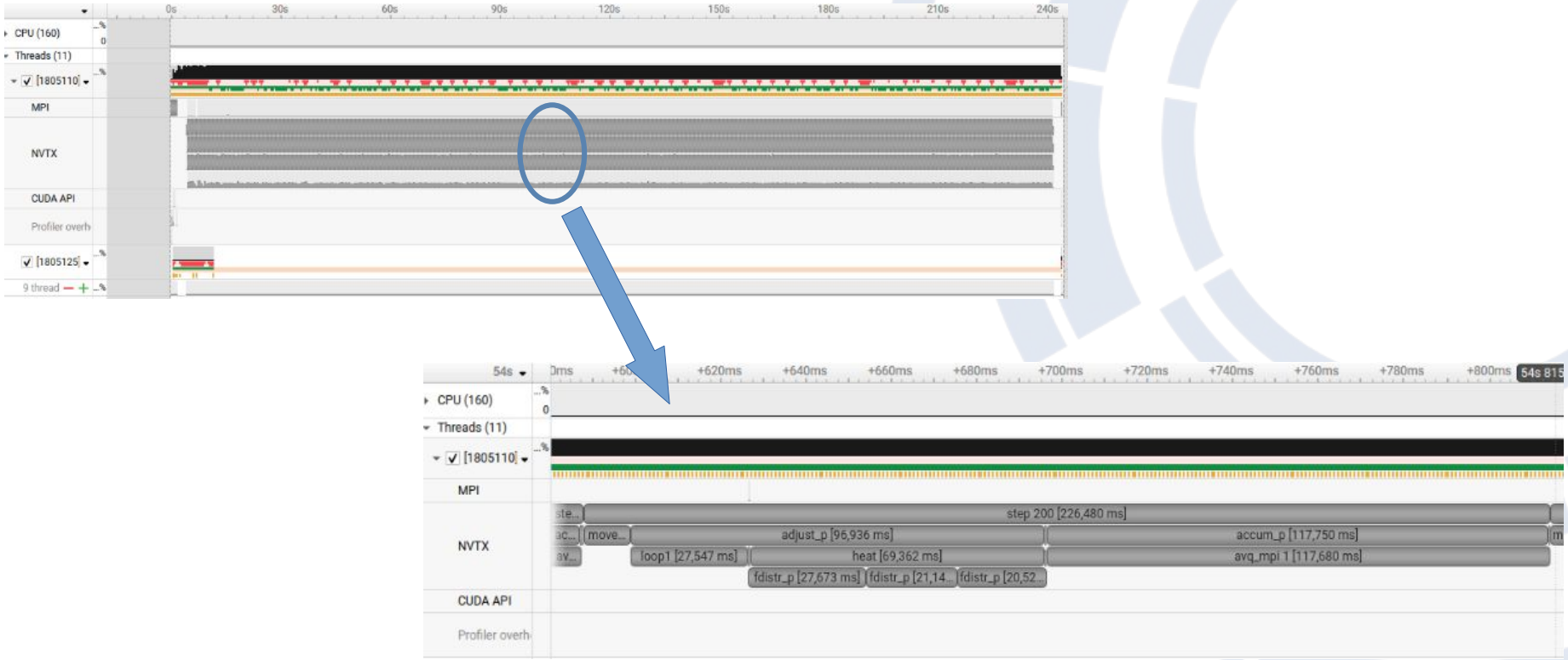
```
if (np[iisp][j])
{
  Tx[iisp][j] *= Tscale[iisp]/np[iisp][j];
  T_dif[iisp][j] *= Escale[iisp]/np[iisp][j];
}
```

```
 np_tmp = np[iisp][j];
 if (np_tmp)
{
  Tx[iisp][j] *= Tscale[iisp]/np_tmp;
  T_dif[iisp][j] *= Escale[iisp]/np_tmp;
}
```

  - Splitting large loops for better parallel analysis.

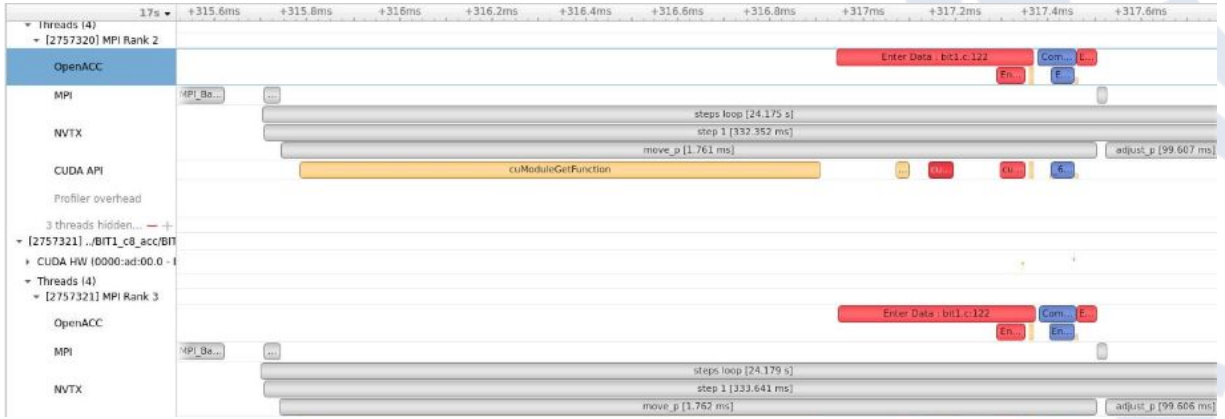
BIT1: CPU Version (2)

- **Trace:**
  - Execution flow of main PIC loop.
  - Time spent in particle mover and field solver.
  - Profiling used to identify GPU porting targets.



BIT1: First GPU Approach

- **Initial GPU-enabled routines:**
  - move0 (particle mover).
  - arrj (particle-to-cell assignment).
- **Results:**
  - move0 accelerated from 43.6s → 2.16s (20×).
  - Overall runtime worsened (0.87× speedup).
- **Reason: excessive CPU–GPU transfers.**
  - Led to strategy of porting all routines in test case.

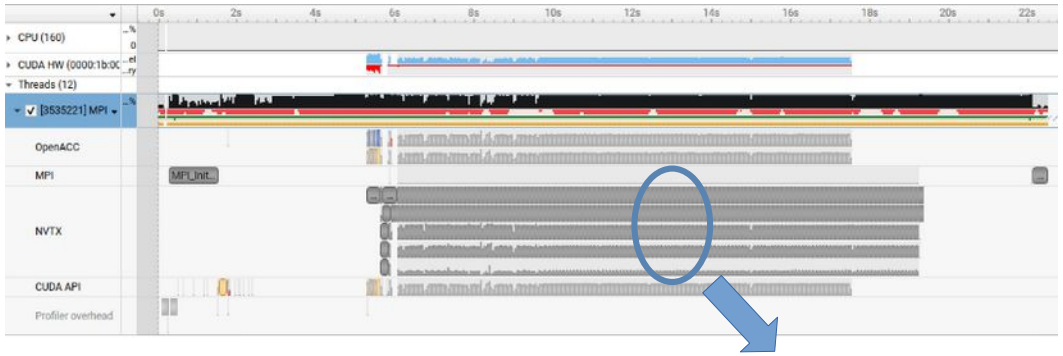


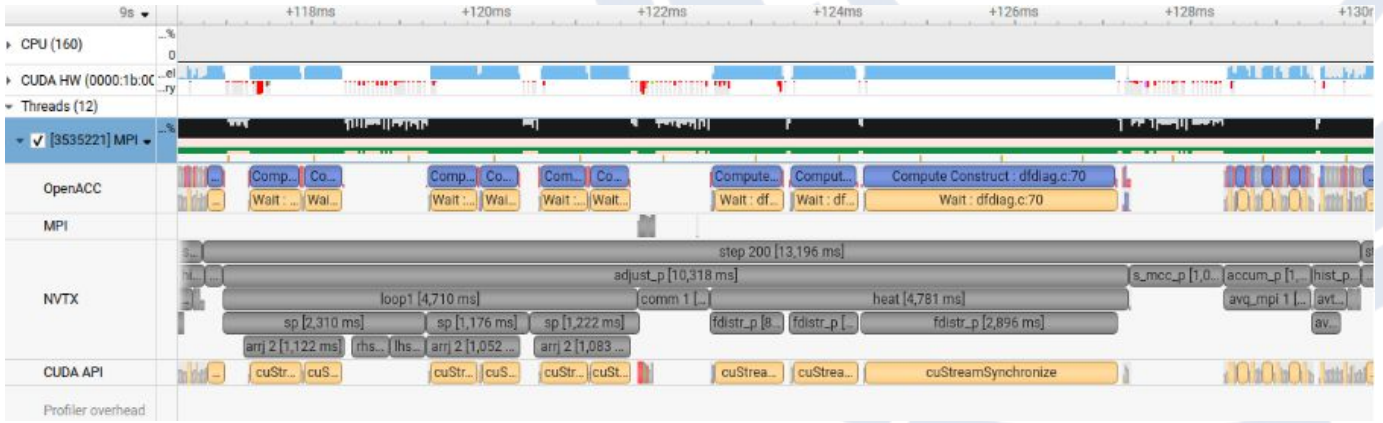
BIT1: GPU Version

- **GPU-enabled routines:**
  - 16 routines across 10 source files.
- **Substantial redesign of arrj for parallel execution.**
- **Performance:**
  - Total runtime improved from 249 s to 35.8 s (~7× faster).
  - Step time improved from 260 ms to 13 ms (~20× faster).
- **Most runtime now spent on GPU kernels.**

BIT1: GPU Version (2)

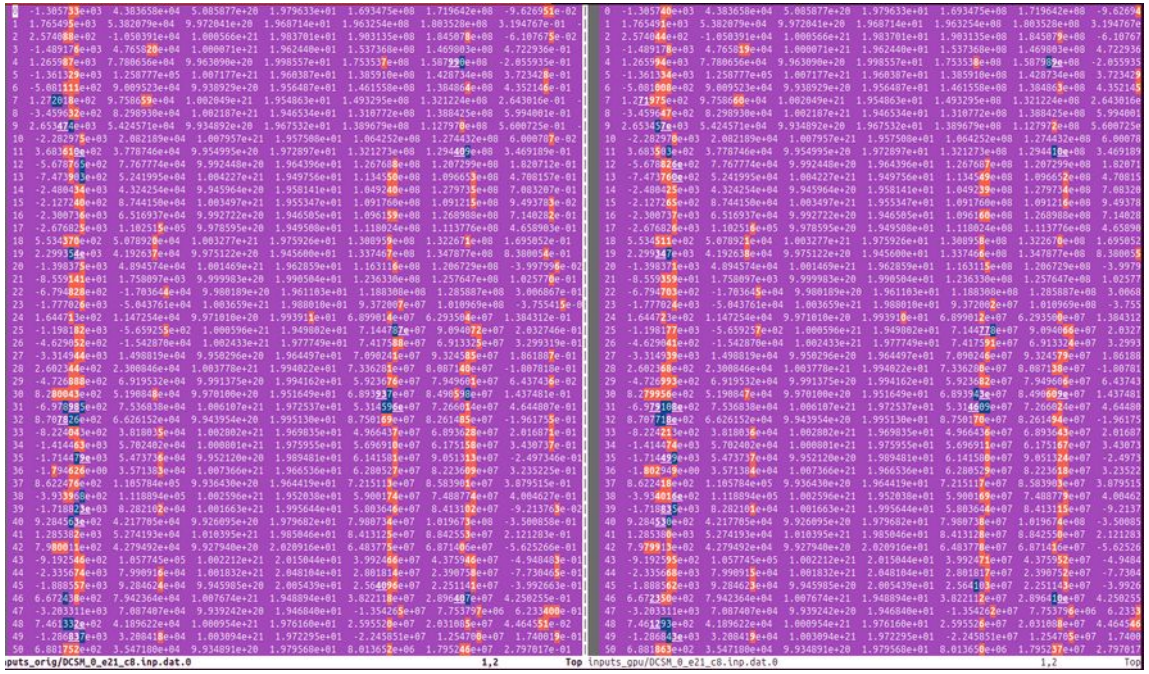
- **Trace:**
  - Dominance of GPU kernels over CPU execution.
  - Greatly reduced host–device transfers.
  - Balanced overlap of MPI communication.

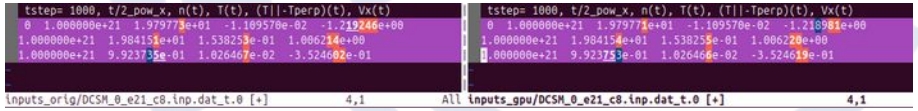




BIT1: Floating-Point Differences

- **GPU results differ slightly from CPU outputs:**
  - Due to different operation ordering in parallel execution.
  - Floating-point arithmetic differences between CPU/GPU.
- **All outputs provided for validation in the repository.**

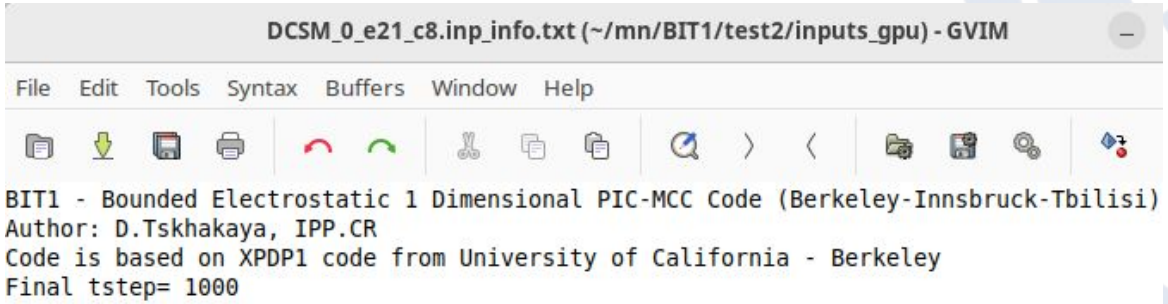




BIT1: Segmentation Fault After Completion

- **Observed crash after simulation ends:**
  - Occurs after all files are written.
  - Likely related to memory deallocation.
- **Does not affect correctness.**
  - Requires future debugging for clean execution.

```
=== Execute gpu version ===
time mpirun -n 4 ./wrap_nsys.sh ../src_gpu/BIT1 inputs_gpu/DCSM_0_e21_c8.inp
Max. number of particles per cell per species: 0  2000 1  2000 2  2000
[as03r5b31:3535221:0:3535221] Caught signal 11 (Segmentation fault: address not mapped to object at address 0x42058058)
Generating '/scratch/tmp/31995526/nsys-report-9ded.qdstrm'
```



BIT1: Conclusion

- **The main objective of porting BIT1 to GPUs using OpenACC was achieved,** with a maintainable and readable codebase preserved and GPU execution successfully demonstrated on a test case.

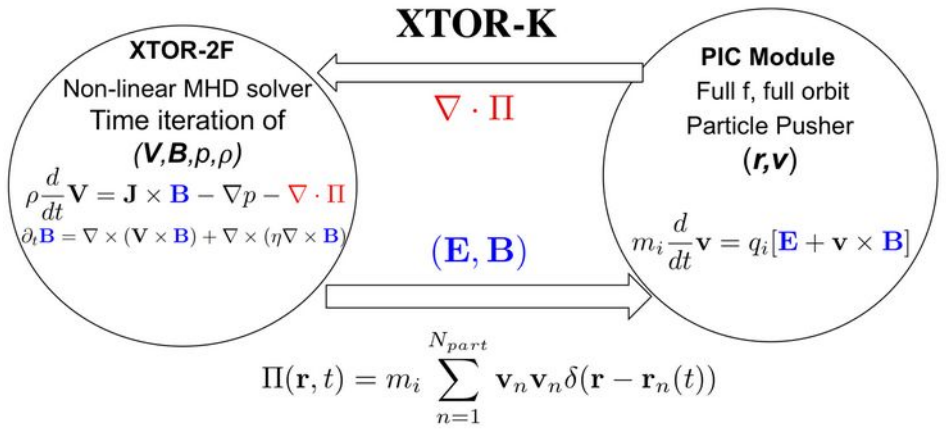
- **Next steps:**
  - **Fix finalization segmentation fault.**
  - **Extend GPU port with OpenACC:**
    - Continue adding OpenACC directives to the remaining code paths → decide new test cases.
    - Improve host–device data management:
      - Use manual control of data regions instead auto-managed memory.
      - Overlap communication and computation where possible.
  - **Perform a deep GPU redesign**

XTOR-K

Eduardo Cabrera

XTOR-K: Introduction

- XTOR-K is a fully implicit, HPC-oriented hybrid fluid–kinetic model that incorporates a Particle-in-Cell (PIC) Boris–Buneman particle pusher applied to one or more populations of kinetic ions. This enables the simulation of couplings between the Alfvénic spectrum and ion kinetic effects.
- It couples in a self-consistent manner the time advance of a complete set of bi-fluid equations in full tokamak geometry (thermalized plasma electron and ions background) with a 6D PIC method accurately integrating the trajectories of selected ion populations, making accessible new families of instabilities which are not allowed in a simplified framework such as magnetohydrodynamics.

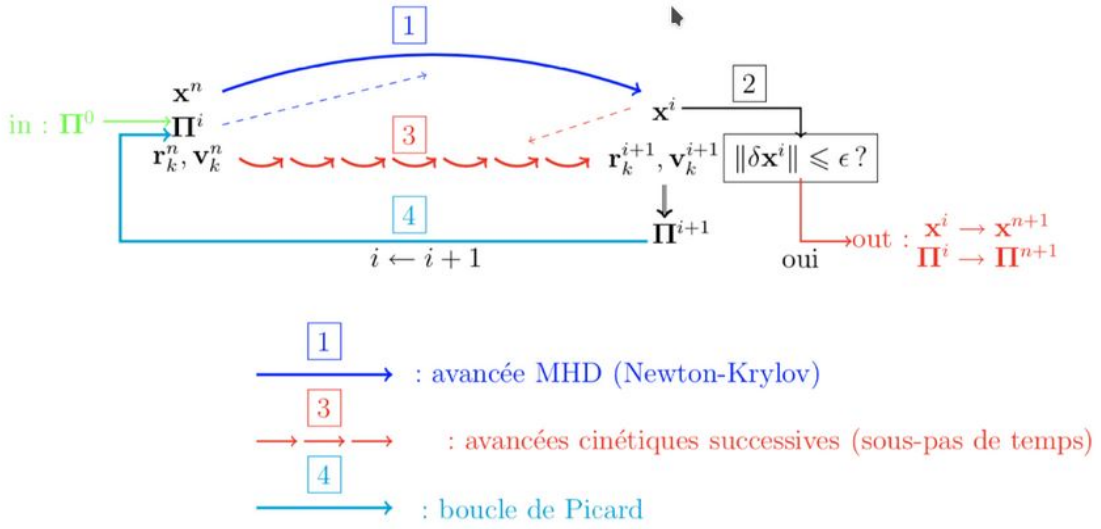


Hinrich Lütjens

- It is implemented in Fortran2008, parallelised with MPI and OpenMP already.

XTOR-K: Project Goal

- **Motivation:**
  - The kinetic component dominates the overall computational cost.
  - It relies on a PIC time-advance scheme.
- **Goal: Porting the kinetic components to GPU**
  - Reduce time spent in kinetic routines → Improve overall scaling.
  - Significantly increase the number of markers per simulation → Reduce the numerical noise inherent to the PIC method.



XTOR-K: Profiling & Bottlenecks

- **Profiling tool:**
  - Application Performance Snapshot (APS) from Intel VTune.
  - Runs performed on MareNostrum 5 (H100 GPUs, NDR InfiniBand)
- **Profiling results:**
  - Sligntly low vectorization → addressed using appropriate compiler flags
  - Main hotspots identified:
    - interpolate_eb_fields (most time-consuming routine).
    - moveRZ (calls interpolate_eb_fields multiple times).
    - OpenMP barrier synchronization overhead.
- **Conclusion:**
  - GPU porting targets these bottlenecks.

XTOR-K: CPU code

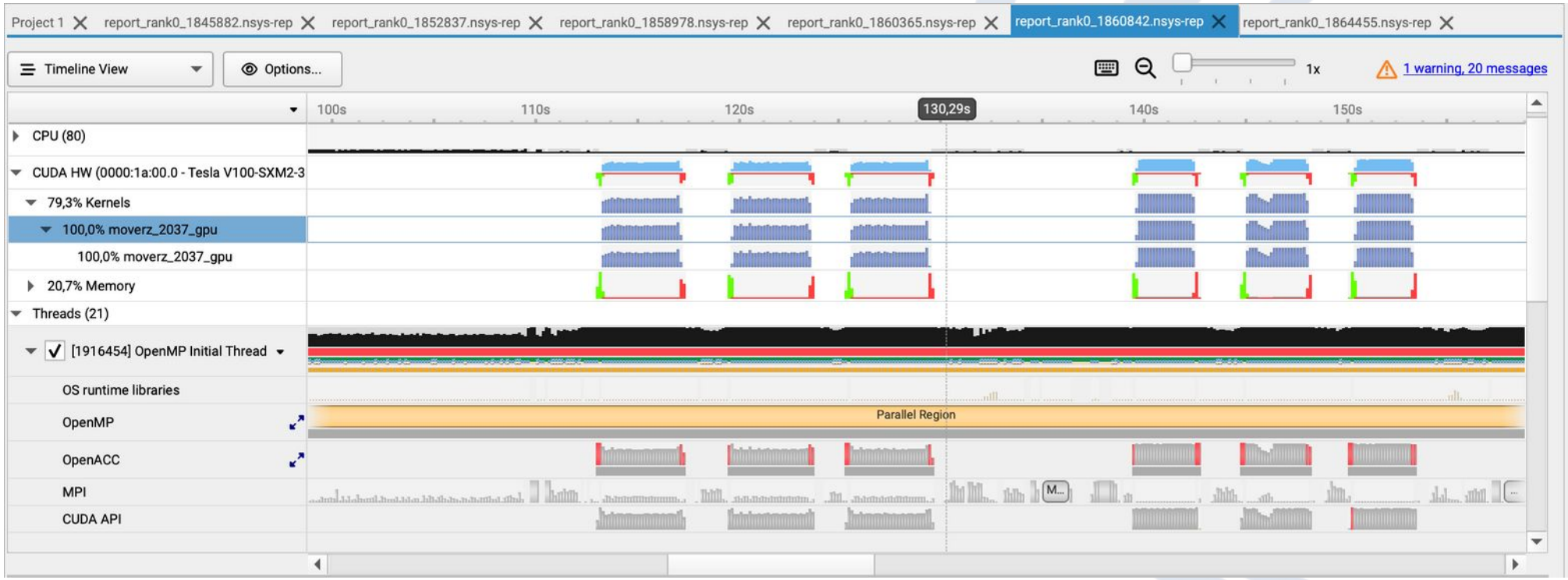
- **Ensuring a stable bug-free CPU version**
- **Identified and fixed issues when compiling with newer Intel compiler version**
  - PETSc type mismatches (e.g. ierr must be PetscErrorCode, not PetscInt).
  - Incorrect use of REAL for array indices → Replaced with proper INTEGER indices.
  - Division-by-zero errors when accessing ghost-point indices at domain boundaries.
  - Makefile compilation order issues → Reordered modules so dependencies are compiled first.
  - Mismatch between routine definitions and their implementations (argument list).
- **Outcome**
  - Stable and validated hybrid MPI/OpenMP CPU version ready.

XTOR-K: GPU Porting

- **Initial Approach:**
  - Tested OpenMP offloading option → more than 3x slower than CPU.
- **Final Approach**
  - Adoption of OpenACC.
  - Porting of 6 kinetic subroutines: particlepositioninfo, interpolate_eb_fields, particleshaperz, depositmoments, inirun and **moveRZ**.
  - **Issue:** Loops often too small (≤64 iterations) for efficient GPU parallelism.
  - **Revised Strategy:**
    - Parallelize only the outermost loop in moveRZ (# particles).
    - Execute the routines it calls serially on the GPU.
    - Transfer all required data (CPU -> GPU) prior to executing moveRZ routine.
- **Work completed:**
  - Full moveRZ routine successfully ported to GPU.
  - Additional routines (fullebeparmappingrz and fullparticleadvance) from non-kinetic part were ported.
  - Data transfer between CPU ans CPU streamlined.
  - Redundant MPI barriers calls removed.

XTOR-K: Performance Results

- **The GPU version outperforms the CPU version.**
- **Achieved simulations with up to 100 million particles per GPU → Not feasible on CPU nodes.**
- **GPU porting now enables large-scale simulations that were not feasible on CPU-only systems.**



XTOR-K: Future work

- **Ongoing Work:**
  - Porting projectedmomentsmappingrtheta subroutine using cuFFT and cuBLAS.
  - **Issues:** ~30 missed deallocations detected with compute-sanitizer tool.
  - **Critical issue:**
    - Argument-passing inconsistency: 3D array passed as 4D.
    - Fortran requires exact matching of array dimensions.
    - This inconsistency currently blocks completition of the GPU port.
- **Next steps:**
  - Clarification has been requested from developers regarding whether this array mismatch is intentional to evaluate alternatives.
  - Complete GPU port of the remaining routines.
  - Further optimization needed for data transfers.

Thanks for your attention !