

Hybrid Parallelization in SOLPS-ITER

Gaurav Saxena (HLST)
David Vicente Dorca (Head, User Support)

[Barcelona Supercomputing Center, Spain]

EUROfusion Meeting | 25th Nov. 2025



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.

Presentation Code

RGB Presentation Color Code



Bad / Alarm



Good / Improvement



Keyword / Code



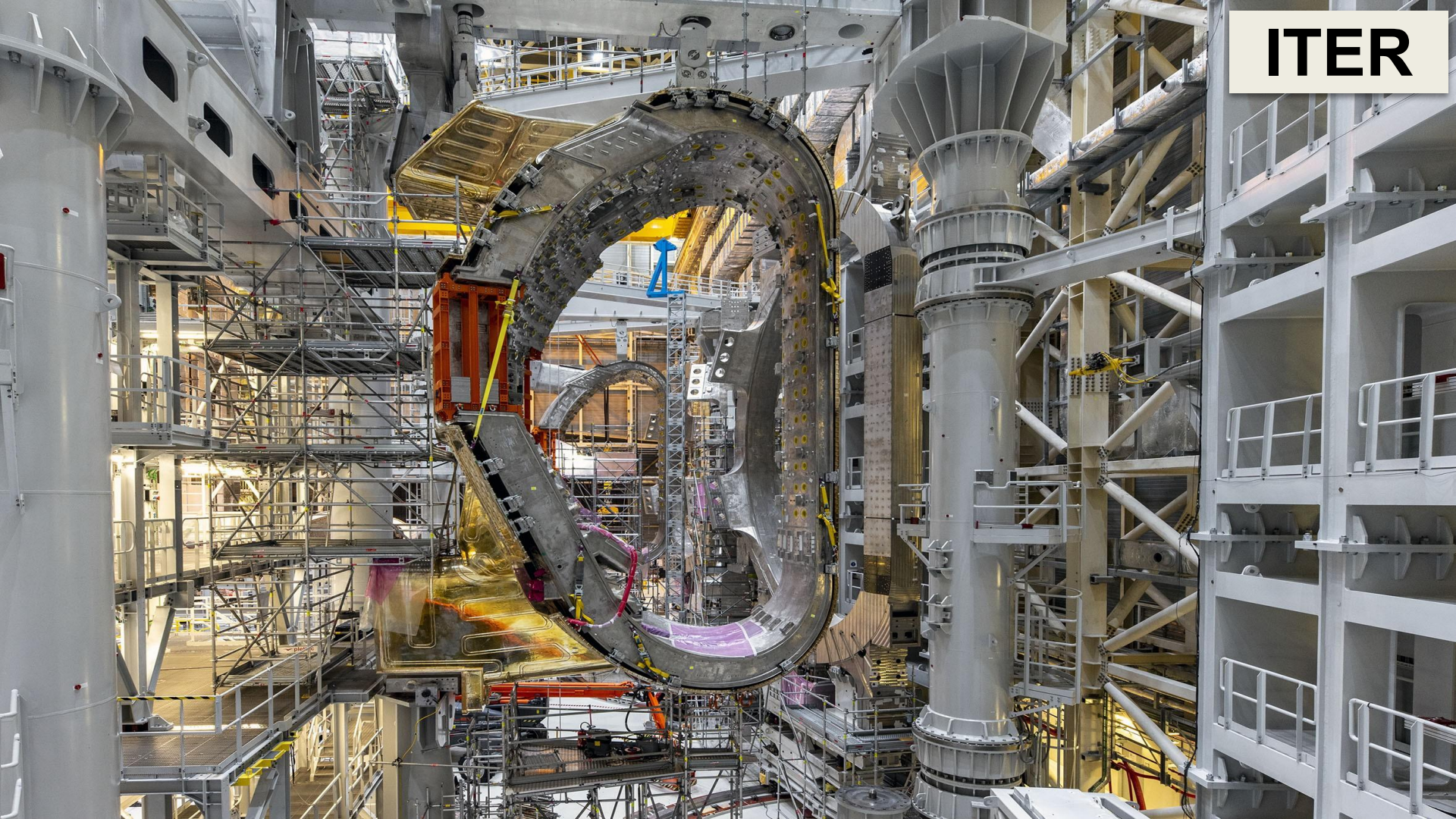
File name (can ignore)

What is SOLPS-ITER ?

SOLPS- ITER

- **Scrape-Off Layer Plasma Simulation** (boundary plasma)
- **ITER** = International Thermonuclear Experimental Reactor
- Monte Carlo code **Eirene** (**MPI** parallelized + now **OpenMP**)
- **B2.5** Plasma Fluid solver (**OpenMP** parallelized, No **MPI**)
- Fortran 77 (fixed form), Fortran 90
- **Our Focus** - Coupled (**B2.5+Eirene**) Hybrid (**MPI + OpenMP**)
runs

ITER



(Very) Brief Recap (on MN4/MN5 in 2023 & 2024)

- Worked in optimizing B2.5
- Re-wrote `get_num_threads()` function (5.6 - 17% time reduction).
- Added some compiler options & correct affinity: ($\approx$ 36% time reduction)
- Removed unneeded critical sections (4.45x Speed-up Vs 1.23x original)
- Vectorization in serial/parallel functions e.g., function `fka()`
- Identified multiple bottlenecks `b2sifr_`, `b2tfnb`, `ma30bd`, `b2txcy`, `b2stbr_phys` and `b2sihs` ...
- Migration to MN4 $\rightarrow$ MN5
- Assistance for porting code to *Red-Hat Linux* nodes at ITER.
- Assistance for removing extremely critical bugs when using `gfortran`.
- Identifying bugs in Intel *ifx* compiler (NOTE: use only classic *ifort* compiler in 2025)

Hybrid Parallelization in SOLPS-ITER

MPI + OpenMP Eirene

- Eirene originally parallelized using MPI
- Later OpenMP parallelization added to Eirene
- Mail from Xavier Bonnin on 4th February 2025 informed us that

MPI + OpenMP Eirene version non-functional

- Identified as [most urgent requirement of 2025](#) (by developers).

MPI + OpenMP Eirene ... example & building

- Example chosen for experiments: AUG_16151_D+C+He (specifically 16151_1.6MW_2.0e19_D=0.4_chi=1.6_pump=0.90)
- Branch: [release/3.1.1](#)
- Make target: [make all_nox_openmp_mpi](#)
- Repaired CMakeCacheFiles.txt reference to [mpif90](#) as using [mpiifort](#)
 - export [MPI_FC=mpiifort](#) ✓
 - export [FC=mpiifort](#) 🤔
- ['b2mndr_eirene'](#) to ['1'](#) (master switch according to the manual) in [b2mn.dat](#)

MPI + OpenMP Eirene - OpenMP switch

- Problems in enabling OpenMP support in Eirene - needed to switch ON option in CMakeLists.txt (see below)

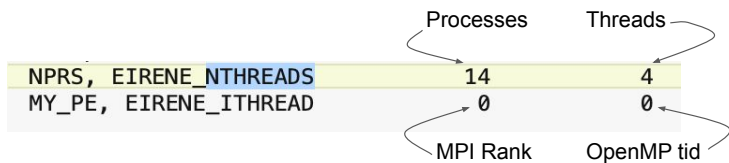
```
# Add option to enable OpenMP with the parallel region defined in eirmod_mcarlo.  
option(OPENMP "Build for use with internal parallel region" ON) # Gaurav Saxena making it ON, it was OFF  
if (OPENMP)  
  cmake_minimum_required(VERSION 3.1 FATAL_ERROR)  
  message(STATUS "-- Adding USE_OPENMP flag...")  
  add_definitions(-DUSE_OPENMP)  
  find_package(OpenMP REQUIRED)  
  if ("${CMAKE_Fortran_COMPILER_ID}" STREQUAL "IntelLLVM")  
    message(FATAL_ERROR "-- OpenMP is unsafe with ifx compiler! Do not use. --")  
  endif()  
endif()
```

ifx warning !

MUTUALLY EXCLUSIVE

```
# Add option to enable OpenMP with the parallel region defined in the calling program.  
option(EXT_OPENMP "Build for use with external parallel region" OFF) # Gaurav Saxena switch OFF if above is ON  
if (EXT_OPENMP)  
  cmake_minimum_required(VERSION 3.1 FATAL_ERROR)  
  message(STATUS "-- Adding USE_EXT_OPENMP flag...")  
  add_definitions(-DUSE_EXT_OPENMP)  
  find_package(OpenMP REQUIRED)  
endif()
```

MPI + OpenMP Eirene: confirmation & error



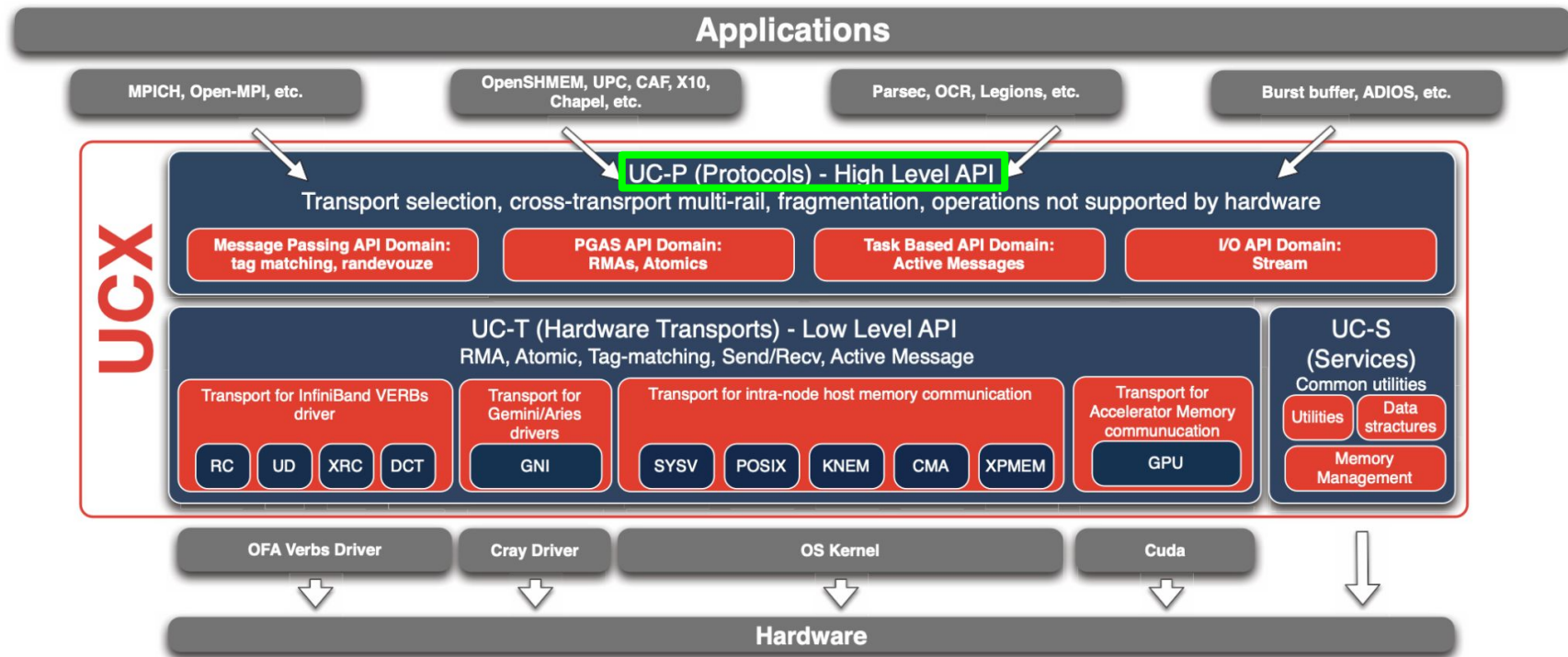
```
All PEs Calculate All Strata
Creating communicator within stratum
Creating group from ranks    0  1  2  3  4  5  6  7  8  9 10 11 12
Creating communicator within stratum
Creating group from ranks    0  1  2  3  4  5  6  7  8  9 10 11 12
Creating communicator within stratum
Creating group from ranks    0  1  2  3  4  5  6  7  8  9 10 11 12
Creating communicator within stratum
Creating group from ranks    0
Creating communicator for stratum leaders
Creating group from ranks    0
[gs08r1b29:2330033:0:2330156] ucp_worker.c:2888 Assertion `worker->inprogress++ == 0' failed
[gs08r1b29:2330025:0:2330143] ucp_worker.c:2888 Assertion `worker->inprogress++ == 0' failed
[gs08r1b29:2330033:1:2330155] ucp_worker.c:2888 Assertion `worker->inprogress++ == 0' failed
[gs08r1b29:2330025:1:2330144] ucp_worker.c:2888 Assertion `worker->inprogress++ == 0' failed
[gs08r1b29:2330032:0:2330142] ucp_worker.c:2888 Assertion `worker->inprogress++ == 0' failed
[gs08r1b29:2330032:1:2330140] ucp_worker.c:2888 Assertion `worker->inprogress++ == 0' failed
[gs08r1b29:2330035:0:2330035] ucp_worker.c:2888 Assertion `worker->inprogress++ == 0' failed
```

- Successfully spawns MPI processes and OpenMP threads
- Execution **crashes** with UCP (Part of UCX - Unified Communications X) error above
- **\$ module load ucx** - on MN5 does not help

What is UCP ?

- UCX - Unified Communications X
- UCX is a Networking stack = Implementation of APIs + Protocols
- UCX APIs satisfy networking needs of Programming models like MPI
- UCX is highly performant (close to device driver performance)
- UCX = UC-P(Protocol) + UC-T (Transport) + UC-S (Services)
- UCP
 - Implements higher level protocols used by MPI, PGAS etc.
 - UCP uses the services provided by UCT (typical of layered architecture)
 - Library initialization
 - Transport selection
 - Message fragmentation
 - Multi-rail communication (each process can use more than 1 active port on host)

UCX Networking Stack



Problem Diagnosis

- Diagnosis: Problem was fundamental and software stack related
- No segmentation fault, buffer overflow, NaN
- Problem was with MPI as developers did not touch UCX
- MPI thread support level in SOLPS-ITER: **MPI_THREAD_FUNNELED** (specified in eirmod_mpi.F90)

```
1291
1292  subroutine eirene_mpi_init(ier)
1293    ! MPI initialization routine used by either
1294    ! eirene_main (for standalone) or
1295    ! eirene_eirene (coupled) subroutines
1296    integer, intent(out) :: ier
1297    #ifdef USE_OPENMP
1298      call mpi_init_thread(mpi_thread_funneled, mpi_thread_provided, ier)
1299    #else
1300      call mpi_init(ier)
1301    #endif
1302  end subroutine eirene_mpi_init
1303
1304 end module eirmod_mpi
```

Thread Support Levels & specification violation

- Whenever MPI is mixed with OpenMP, users need to specify “How” MPI will communicate.
- This “how” give rise to Thread Safety levels - 4 in number
 - `MPI_THREAD_SINGLE` - no OpenMP, only master thread, same as MPI process-only.
 - `MPI_THREAD_FUNNELED` - *comm. within OpenMP region from Master thread only*
 - `MPI_THREAD_SERIALIZED` - any thread can communicate but not at the same time
 - `MPI_THREAD_MULTIPLE` - any thread any time.
- ⇒ All MPI calls within OpenMP region must be enclosed in `$OMP MASTER` region.
- **Violation:** `MPI_ALLGATHER`, `MPI_ABORT` and `MPI_FINALIZE` in `EIRENE_EXIT_OWN()` and `EIRENE_CHECK_EXIT()` in `exit_own.F` & others.

Code correction - MPI + OpenMP

26	26	<code>#ifdef USE_MPI</code>	1676	<code>call ioflush_usr</code>
27	+	<code>#if (defined(USE_OPENMP) defined(USE_EXT_OPENMP))</code>	1677	<code>#if(defined(USE_OPENMP) && !defined(USE_EXT_OPENMP))</code>
28	+	<code>!\$OMP MASTER</code>	1678	<code>CALL EIRENE_CHECK_EXIT</code>
29	+	<code>#endif</code>	1679	<code>WRITE(iunout,*)'Line 1679, eirmod_mcarlo.F'</code>
27	30	<code>CALL MPI_ALLGATHER (LXOWN, 1, MPI_LOGICAL,</code>	1680	<code>!\$OMP MASTER</code>
28	31	<code>LEXIT, 1, MPI_LOGICAL, MPI_COMM_WORLD, IER)</code>	1681	<code>CALL MPI_BARRIER(MPI_COMM_WORLD,ier)</code>
32	+	<code>#if (defined(USE_OPENMP) defined(USE_EXT_OPENMP))</code>	1682	<code>!\$OMP END MASTER</code>
33	+	<code>!\$OMP END MASTER</code>	1683	<code>!\$OMP BARRIER</code>
34	+	<code>#endif</code>	1684	<code>#endif</code>

- The example above shows `MPI_ALLGATHER()` [in `exit_own.F`] and `MPI_Barrier()` [in `eirmod_mcarlo.F`] being placed in `!$OMP MASTER`.
- Further, `!$OMP ATOMIC WRITE` from `LGEXIT` (boolean threadprivate) variable can/will be removed (not shown, if not removed no harm only performance penalty).
- Result: `UCX / UCP error resolved` but `program now hangs`.

Stratums, communicators, groups, MPI Ranks

- Particles on each stratum solved by multiple processes
- These multiple processes associated with the stratum form a “communicator”. (See (A))
- Communicator is an MPI concept - just means ranked processes that can communicate.
- Two types of cases:
 - Presence of a Time-census (Last Stratum) stratum : **Problematic case**
 - Absence of Time-census stratum : **Program executes successfully**
- Time-census stratum turned-off by:
 - Setting **NPRNLI** variable to **0** in the **block 13** of the Eirene input file (input.dat) and (See (B))
 - removing the ‘**T**’ element in the **b2.neutrals.parameters** file (See (C))

```

Initializing APCAS workload distribution
All PEs Calculate All Strata
Creating communicator within stratum      6
Creating group from ranks    0  1  2  3
Creating communicator within stratum      7
Creating group from ranks    0  1  2  3
Creating communicator within stratum      8
Creating group from ranks    0  1  2  3
Creating communicator within stratum      9
Creating group from ranks      0
Creating communicator for stratum leaders
Creating group from ranks      0
    
```

(A)

```

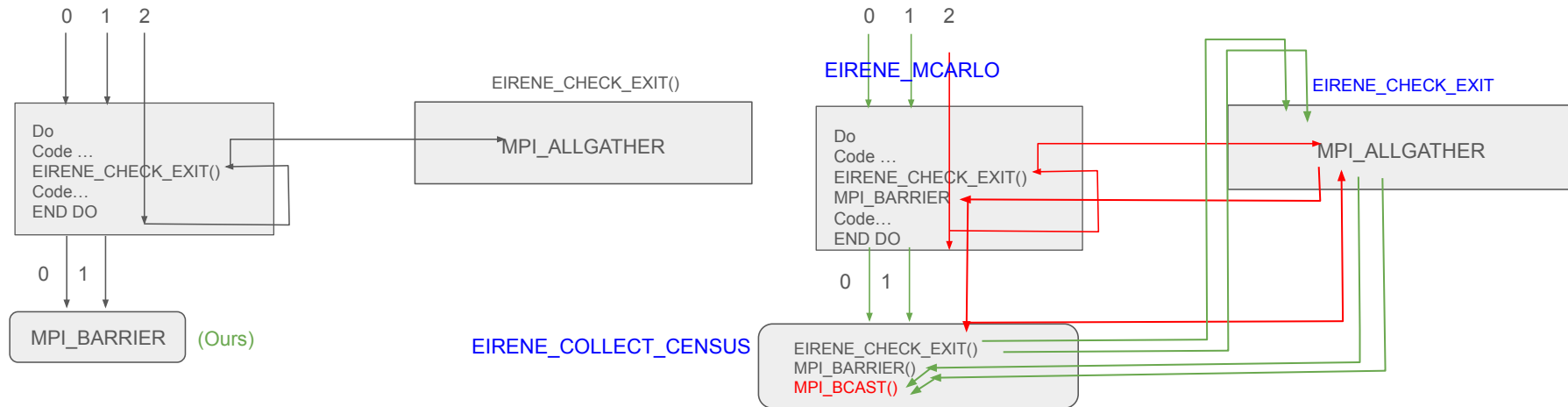
1418  *** 13. Data for nonlinear and time dep. option
1419      50000
1420          0      1
1421      1.00000E+00 0.00000E+00
1422      ** 13a. Data for snapshot tallies
1423          0
    
```

(B)

1	&NEUTRALS
2	NSTRAI=8,
3	CRCSTRA= 'W','E','S','S','N','V','V','V','T',
4	RCPOS= -1, 96, -1, -1, 36, 0, 0, 0, 0,
5	RCSTART= 0, 0, 0, 72, 0, 0, 0, 0, 0,
6	RCEND= 35, 35, 23, 95, 95, 0, 0, 0, 0,

(C)

Diverging behaviour of MPI processes



(A) Problem: Diagnosis

(B) Actual Problem:

- (a) Rank 2 blocked in MPI_ALLGATHER()
- (b) Rank 0 & 1 blocked in MPI_BCAST()

Visualization: MPI process handling time-census stays 1-step behind when there are no time-census particles on other processes.

Occam's Razor - Simplest solution

- Removed several MPI_BARRIER calls from eirmod_mcarlo.F
- As reported by Xavier Bonnin
 - NPRLL = 0 or 2 (“embarrassingly parallel” or APCAS strategy) ✓
 - NPRLL = 1 (the ORIGINAL load balancing strategy), hangs ✗
 - NPRLL = 3 (the BALANCED load balancing strategy), fails PMPI error ✗
 - NPRLL = 4 (the IMPROVED load balancing strategy), hangs ✗
- Final nail in eirmod_calstr_buffered.F90

```
105     if (n > next_check(istra)) then
106     !$OMP MASTER
107         call MPI_Testall(N_BUFFERS, calstr_request, flag, MPI_STATUSES_IGNORE, ierr)
108     !$OMP END MASTER
109         next_check(istra) = next_check(istra) + &
110             max(1, nparts_loc(istra)/n_progress_check)
```



<https://git.iter.org/projects/BND/repos/solps-iter/commits> master branch

	Bonnin Xavier	f193e95af28	Pointer to Eirene submodule with final bug fix for hybrid OpenMP+MPI parallelization
	Bonnin Xavier	38beb7a9f6e	Pointer to B2.5 submodule with modified argument lists for IMAS v4
	Bonnin Xavier	219fa07ea0b	Bug fix to allow hybrid OpenMP+MPI Eirene parallelization for NPRL=1 (provided by BSC ACH)
	Bonnin Xavier	80670e898e1	- Pointer to B2.5 submodule with bug fixes for midplane grid subset definitions when imposed by jxi/jxa switches - Advancing to IDStools/
	Bonnin Xavier	82ef450d64b	Pointer to B2.5 submodule with bug fixes for limiter cases and IMAS data entry write-up
	Bonnin Xavier	0c944aab0a5	Pointer to B2.5 submodule with added IMAS major version argument to create_db_entry call
	Bonnin Xavier	186697b37c9	Pointer to B2.5 submodule with modified guard cell coverage test for unconnected cells
	Bonnin Xavier	a5ec99a0541	Bug fix for OpenMP runs on Gateway
	Bonnin Xavier	7b3f5258358	Pointer to Eirene submodule with partial bug fix for hybrid OpenMP+MPI parallelization + documentation improvements

Performance Optimization

1 MPI Process, 1 OpenMP Thread Execution

AUG_16151_D+C+He, 'b2mndr_ntim' '50', Elapsed Time = 736.46 sec

Function Name	CPU Time (sec)	Source File
ma30ad() [parallel]	172.31	ma28.f
b2tfnb() [parallel]	71.16	b2tfnb.f
eirene_mc() [serial]	47.08	b2mod_eirene.f
eirene_update() [parallel]	33.13	update.f
eirene_fpath() [parallel]	28.72	fpath.f
__intel_skx_avx512_memset()	18.96	[Unknown]
intcor() [parallel]	18.24	intcor.f
eirene_if3cop() [serial]	15.03	eirmod_infcop.F

OpenMP regions in Eirene

- 6 OpenMP Parallel regions in Eirene
 - ./main-routines/eirmod_eirene.F: !\$OMP PARALLEL FIRSTPRIVATE(OP,INIT_OPEN)
 - ./main-routines/eirmod_mcarlo.F: !\$OMP PARALLEL DEFAULT(SHARED)
 - ./modules/eirmod_openmp.F: !\$OMP PARALLEL DEFAULT(SHARED)
 - ./modules/eirmod_openmp.F: !\$OMP PARALLEL DEFAULT(SHARED)
 - ./particle-tracing/eirmod_samvol.F: !\$OMP PARALLEL DEFAULT(none)
 - ./particle-tracing/eirmod_samvol.F: !\$OMP PARALLEL DEFAULT(none)
- Only one appears in hotspots: (H1)
eirmod_mcarlo_mp_eirene_mcarlo_\$omp\$parallel@626 (4.3 seconds) in eirmod_mcarlo.F
- Percentage of total time = $(4.3/736.46) \times 100 = 0.5\%$
- (H1) Highly load imbalanced, 95% of work done by Master thread
- **Conclusion:** OpenMP regions in Eirene need not be optimized further.

Scaling with 1 MPI Process, multiple OpenMP threads

Coupled run, Example: AUG_16151_D+C+He, 'b2mndr_ntim' '50'

Threads	Total Time	Speed-up (S) = T_1/T_P	Efficiency = S/P
1	724 sec	1	100%
2	503 sec	1.43	71.5%
4	370 sec	1.95	48.75%
8	335 sec	2.16	27%
16	304 sec	2.38	14.88%

(A) `ma30ad()` & (B) `b2tfnb()` Hotspots of B2.5

- 1 MPI process & 1 OpenMP thread (A) `ma30ad()` $\Rightarrow$ biggest hotspot (172 sec)
- 1 MPI & 2 OpenMP threads, for (A) `ma30ad()`
 - Thread#0 takes 46.61 sec,
 - Thread#1 takes 42.59 sec,
 - Total = $89.2 \approx 172/2$, where are the remaining 82.8 seconds ?
- **Answer:** One additional function named (A) `ma30adcopy()` executed when threads > 1.
 - Thread#0 executes (A) `ma30adcopy()` in 47.38 sec and
 - Thread#1 in 45.25,
 - Total $46.61+42.59+47.38+45.25 = 181.83$ (close to 172 with || overhead).
- 1 MPI + 4 OpenMP threads, (A) `ma30ad()` takes 87.87 total and (A) `ma30adcopy()` takes 93.66, total = 181.53 (close to 172 with || overhead)
- (B) `b2tfnb()` $\rightarrow$ 71.16 (1 thread) $\rightarrow$ $32.84+34.82=67.66$ (2 threads) $\rightarrow$ $13.58+18.66+18.01+17.09 = 67.34$ (4 threads)

eirene_mc() Hotspot of Eirene-B2.5 ...1

- **Serial** subroutine in file `b2mod_eirene.F`
- Approx. 4000 lines, time-intensive loop approx. 1000 lines.
- At 8 threads the biggest hotspot (`ma30ad()`, `b2tfnb()` are parallel)
- Takes ≈ 45 seconds at 1,2,4,8 threads
[`'b2mndr_ntim' '50'`]
- Two clear optimization points:
 - Loop invariant computations
 - Serial but vectorized
 - *copy* of 4 dimensional arrays.
 - *initialization* of 4 & 5 dim arrays

```
2983 | | do istrai=1,nstrat !{
2984 | * by ns
2985 | | recyc_hlp=0.0_R8      Loop invariant w.r.t. is, ix, iy
2986 | | gs_ti = tflux(istrai)
2987 | | gs_fsi = flux_scale(istrai)
2988 | | gs_inv_dbihend = 1.0_R8/dble(ih_end)
2989 | | gs_prod_3 = gs_ti * gs_fsi * gs_inv_dbihend
2990 | | do is=0,ns-1 !{
2991 | | | j=b2eater(is)
2992 | | | gs_inv_sqrt_amppme = 1.0_R8/sqrt(am(is)*mp/me)
2993 | | | if(use_recyceir.gt.0) then
2994 | | | | recyctmp=recyceir(istrai)
2995 | | | else
2996 | | | | recyctmp=recyc(is,istrai)
2997 | | | endif      Loop invariant w.r.t. ix, iy
2998 | | | gs_prod_4 = gs_prod_3 * recyctmp
2999 | | | recyc_hlp=max(recyc_hlp,recyctmp)
3000 | | | if(.not.is_neutral(is)) then !{
3001 | | | | ifl=fluids_list(is)
3002 | | | | do iy=-1,ny !{
3003 | | | | | do ix=-1,nx !{
3004 | | | | | | if(leftix(ix,iy).ne.-2 .and.
3005 | | | | | | & rightix(ix,iy).ne.nx+1 .and.
3006 | | | | | | & bottomiy(ix,iy).ne.-2 .and.
3007 | | | | | | & topiy(ix,iy).ne.ny+1 .and.
3008 | | | | | | & region(ix,iy,0).ne.0) then !{
```

eirene_mc() Hotspot of Eirene-B2.5 ...2

- Optimization 1
 - Quantities calculated once outside loops, used as simple variables
 - Products combined
 - Decreases register pressure
- Optimization 2
 - Use OpenMP to parallelize copying / initialization
 - Used collapse clause to flatten iteration space
- Total decrease in eirene_mc() time at 8 threads
= $(45.69 - 35.15)/45.69 \times 100 = 23.1\%$

```
2600  #if defined(_OPENMP)
2601  !$OMP DO COLLAPSE(4)
2602  #endif
2603      do gs_i4=1,gs_smo_d(1,4)
2604          do gs_i3=1,gs_smo_d(1,3)
2605              do gs_i2=0,gs_smo_d(1,2)
2606                  do gs_i1=0,gs_smo_d(1,1)
2607                      smo_av_mapl(gs_i1,gs_i2,gs_i3,gs_i4) =
2608                      &      smo_mapl(gs_i1,gs_i2,gs_i3,gs_i4)
2609                  end do
2610              end do
2611          end do
2612      end do
2613  #if defined(_OPENMP)
2614  !$OMP END DO
2615  #endif
```

Time reduction for the SOLPS-ITER Application

- Due to the previous changes ([eirene_mc\(\)](#)), the project Developers (Xavier Bonnin) reported the following data points:
 - [D+N](#) case, 10 species, 100 coupled iterations : 173m 45s → 153m 48s = **11.5%** time reduction
 - ITER production case, [D+He+Ne](#), 16 species, 5000 coupled iterations, 36 cores, MPI parallelization, SDCC cluster: 22h 54m → 20h 41m = **9.67%** time reduction

Coupled Pure OpenMP Vs Coupled MPI+OpenMP

B2.5 (Threads)	Eirene (Threads)	Time	B2.5 (Threads)	Eirene (Processes)	Time
1	1	781.42	1	1	674.47
2	2	645.35	2	2	569.23
4	4	507.96	4	4	458.99
8	8	466.42	8	8	417.73

On-going work - Functions with high time variation

Both have 1 B2.5 thread

(A) Eirene 1 MPI process

(no OpenMP thread)

(B) Eirene 1 MPI process

(with 1 OpenMP thread)

		(A)	(B)
1	<u>eirene_collide</u>	0.959s	10.439s
2	<u>eirene_fpath</u>	20.126s	30.564s
3	<u>eirene_update</u>	18.578s	40.502s
4	<u>eirene_switch_partinfo</u>	2.611s	10.071s
5	<u>eirene_timer_triangle_mesh</u>	6.202s	11.929s
6	<u>eirene_time_to_standard_surface</u>	1.070s	4.578s
7	<u>eirene_uptbkg</u>	1.140s	5.362s
8	<u>eirene_uptcop</u>	1.189s	5.743s
9	<u>eirene_veloel</u>	0.901s	5.038s
10	<u>h1rn</u>	1.330s	4.771s
11	<u>eirene_escape</u>	0.250s	1.210s
12	<u>eirene_folneut</u>	4.079s	6.630s
13	<u>eirene_locat1</u>	0.430s	2.040s
14	<u>eirene_reflcl</u>	0.460s	2.190s
15	<u>eirene_sputr1</u>	0.320s	1.160s
16	<u>eirene_stdcol</u>	0.060s	0.920s
17	<u>eirene_stdcol_ass</u>	0.010s	0.330s
18	<u>eirene_stdcol_x_rad</u>	0.120s	0.830s
19	<u>eirene_stdnor</u>	0.010s	0.140s
20	<u>eirene_timea1</u>	0.241s	1.380s
21	<u>eirene_timer</u>	0.450s	6.699s
22	<u>eirene_timet</u>	0.561s	2.330s
23	<u>ncio_px_sync</u>	0.010s	0.861s
24	<u>_svml_i64rem8_z0</u>	14.117s	11.969s
25	<u>_f90_reduction_final_strided</u>	0.010s	1.442s
26	<u>_f90_reduction_init_array</u>	0.020s	1.198s
27	<u>atan2</u>	0.010s	0.210s

Acknowledgement

Many thanks to [Xavier Bonnin](#) & [David Coster](#) for solving all our queries, the quick replies, ideas, testing the code promptly & most importantly for their patience and encouragement.

Thank you !
For any questions:

Gaurav Saxena

(gaurav.saxena@bsc.es)

&

David Vicente Dorca

(david.vicente@bsc.es)

