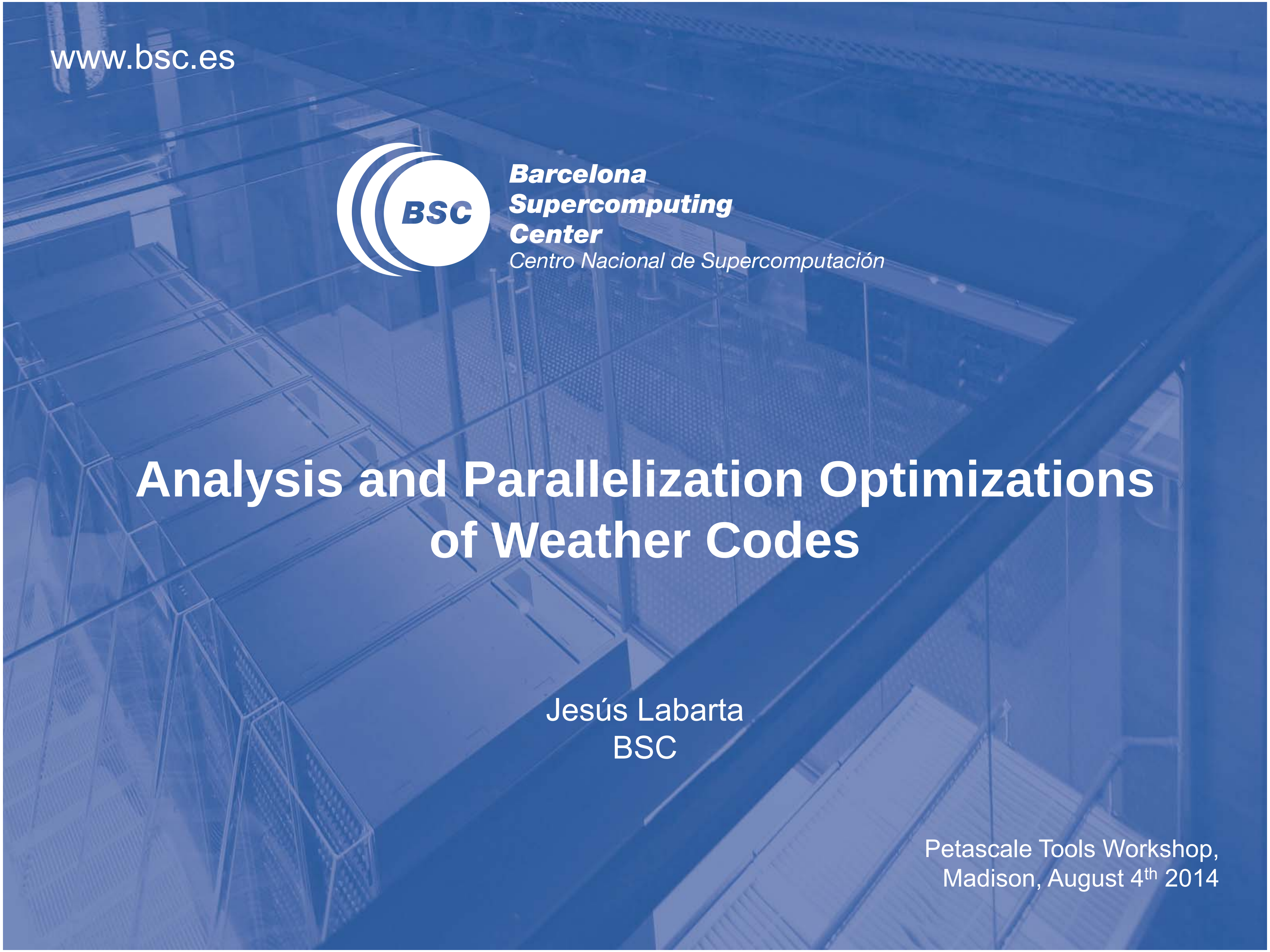


www.bsc.es

BSC

Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

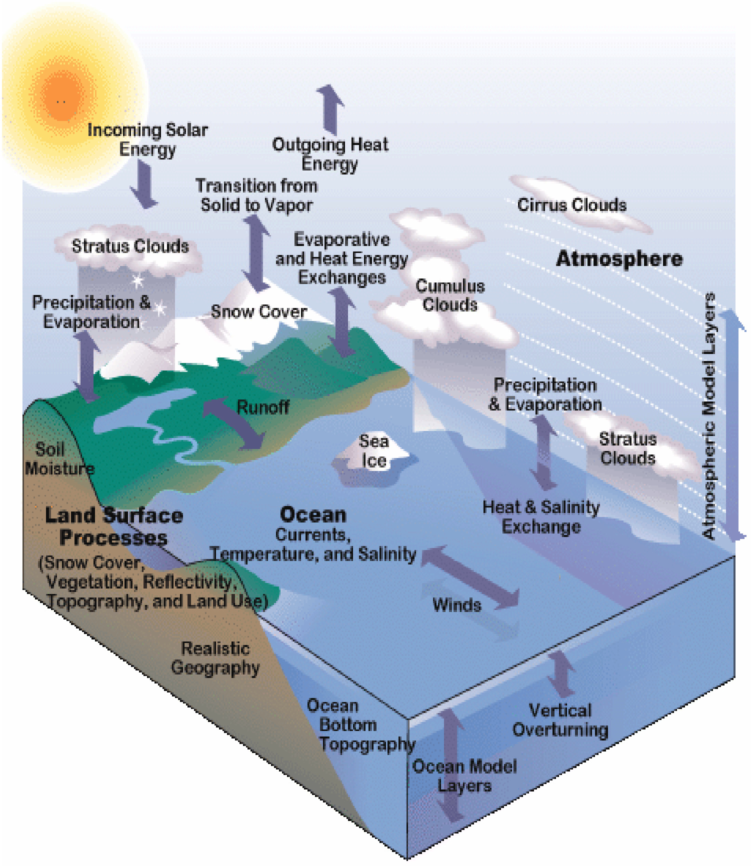
Analysis and Parallelization Optimizations of Weather Codes

Jesús Labarta
BSC

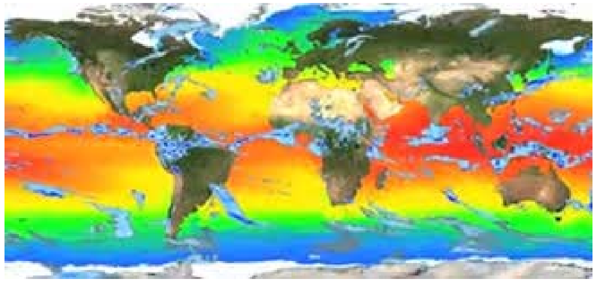
Petascale Tools Workshop,
Madison, August 4th 2014

Earth and Climate

- A complex system
  - Multicomponent
  - Dynamic
- High impact
  - Societal, economic
- Need to
  - Understand and predict
  - Accuracy ↑ uncertainty ↓
  - Compute capacity → exascale
- Complex codes
  - Not toys
  - Not easy bottleneck



Exposed to several weather/climate related codes



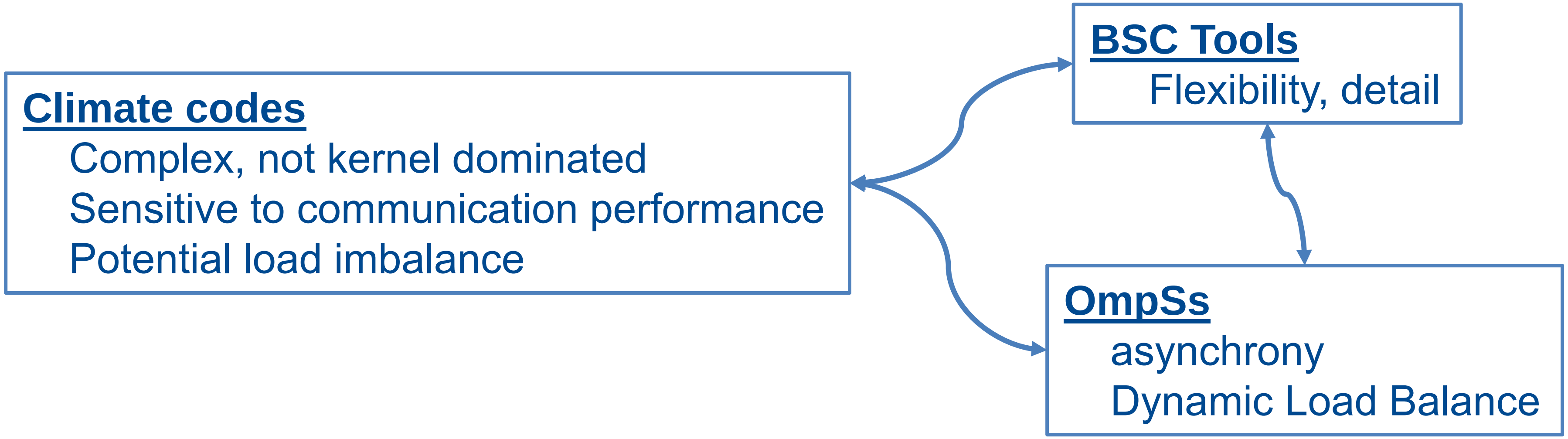
- CESM
  - Cooperation with Rich Loft/John Dennis (NCAR)
  - Full scale code
  - G8 ECS project

- CGPOP
  - Ocean model Kernel
  - G8 ECS Project
- NMMB
  - Cooperation with Oriol Jorba, Georgios Markomanolis (BSC)
  - Full scale code
  - Developing chemical and transport modules on top of NMMB by NCEP

- IFS_KERNEL
  - Kernel by George Mozdzynski (ECMRWF)
  - … mimicking some aspects of the IFS weather forecast code …
  - … to investigate issues and potential of hybrid task based models
  - Some very important restrictions
    - Just 1D decomposition vs 2D in production code
      - More load imbalance than the real code
    - No real physics code
    - No real FFT …

Our interest

- Learn about the three components and their interaction …



Climate codes

Complex, not kernel dominated

Sensitive to communication performance

Potential load imbalance

BSC Tools

Flexibility, detail

OmpSs

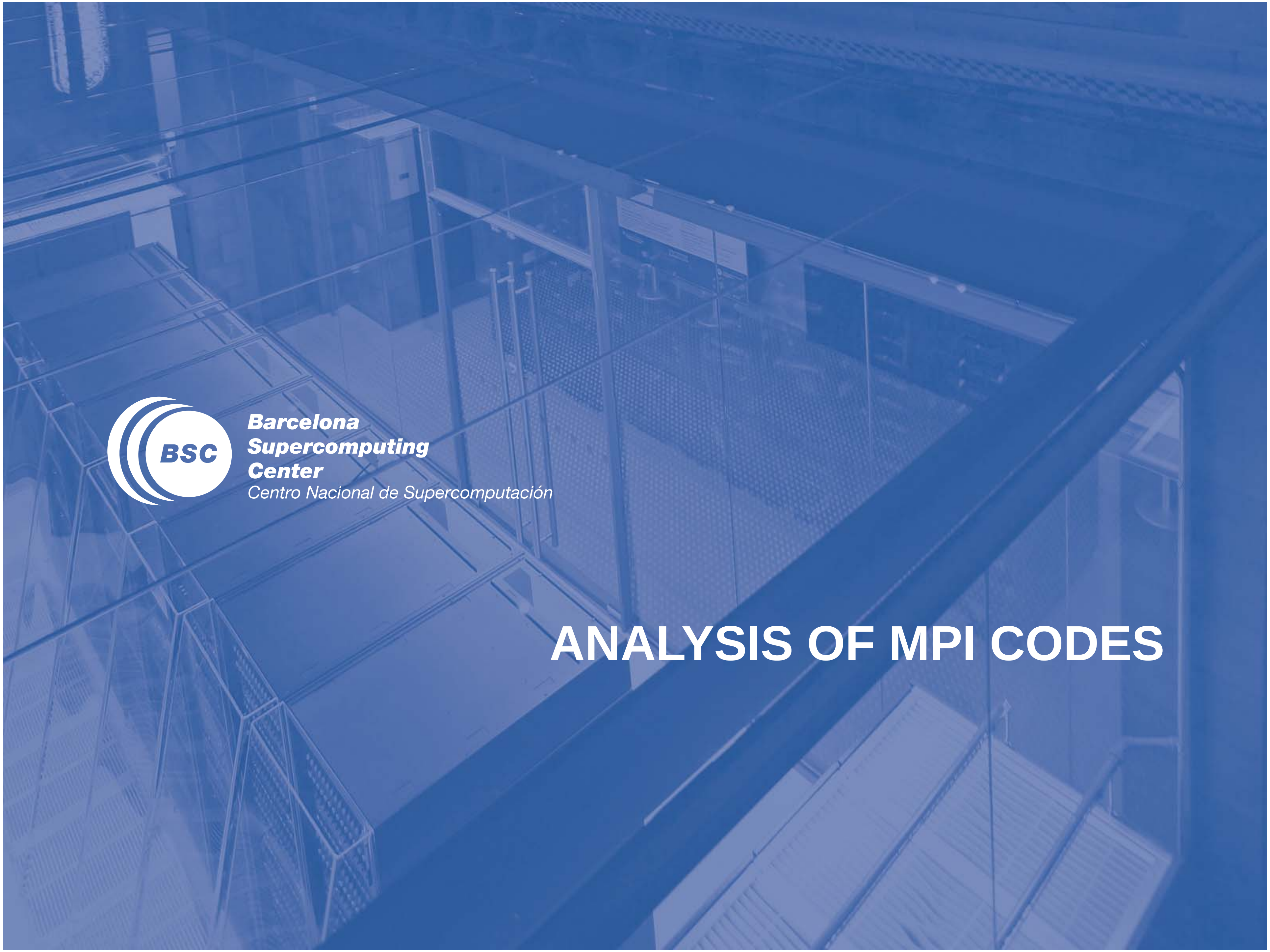
asynchrony

Dynamic Load Balance

- … identify programming model codesign issues/opportunities …
- … report experiences and ongoing work

Index

- Original MPI weather codes
  - Basic analysis
  - Scalability
- OmpSs instrumentation
- Programming patterns
- Dynamic Load Balance

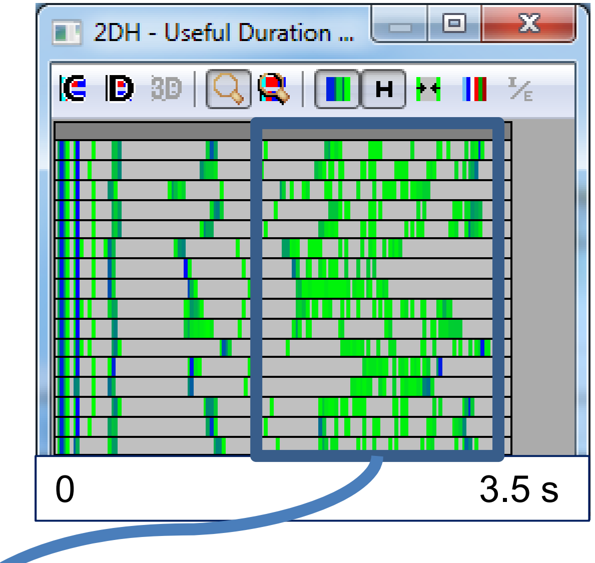


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

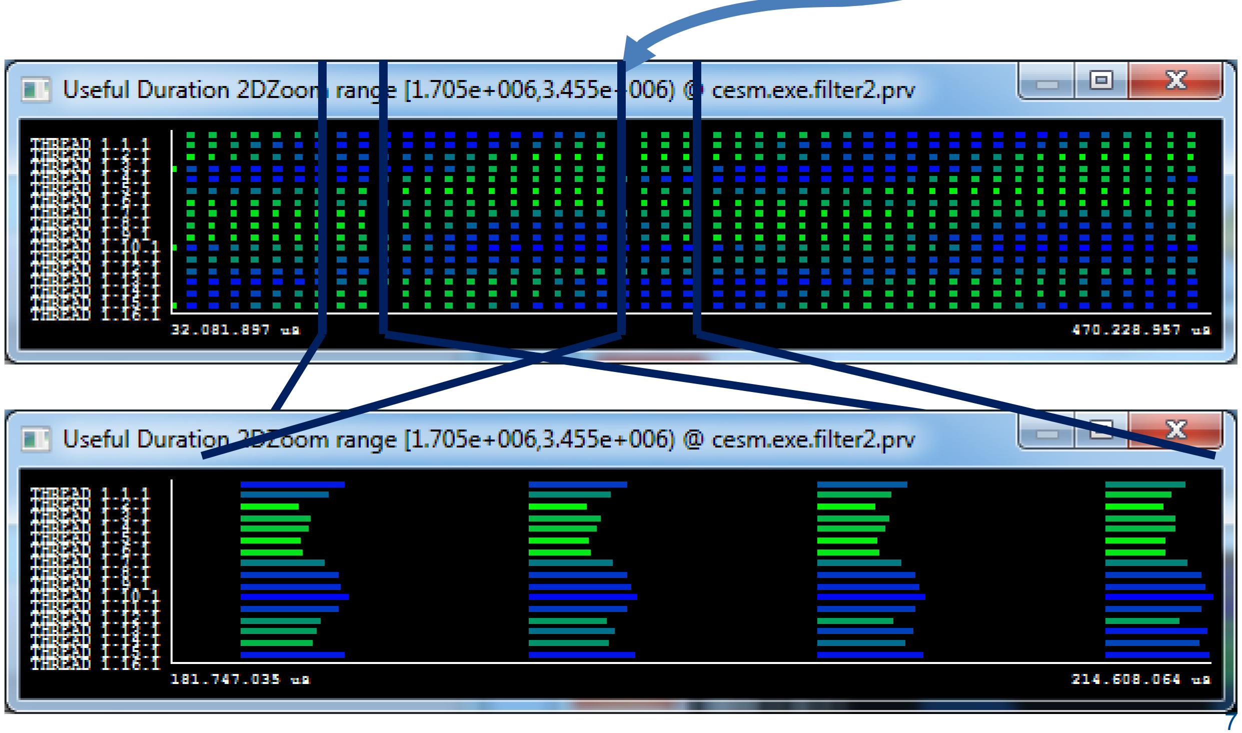
ANALYSIS OF MPI CODES

A “different” view point

- Look at structure …
  - Of **behavior**, not syntax
  - Differentiated or repetitive **patterns** in **time** and **space**
  - Focus on **computation regions** (Burst)



- CESM
  - Micro load imbalance
  - Due to Physics


Useful Duration 2DZoom range [1.705e+006,3.455e+006) @ cesm.exe.filter2.prv
THREAD 1.1.1
THREAD 1.16.1
32.081.897 us
470.228.957 us
Useful Duration 2DZoom range [1.705e+006,3.455e+006) @ cesm.exe.filter2.prv
THREAD 1.1.1
THREAD 1.16.1
181.747.035 us
214.608.064 us

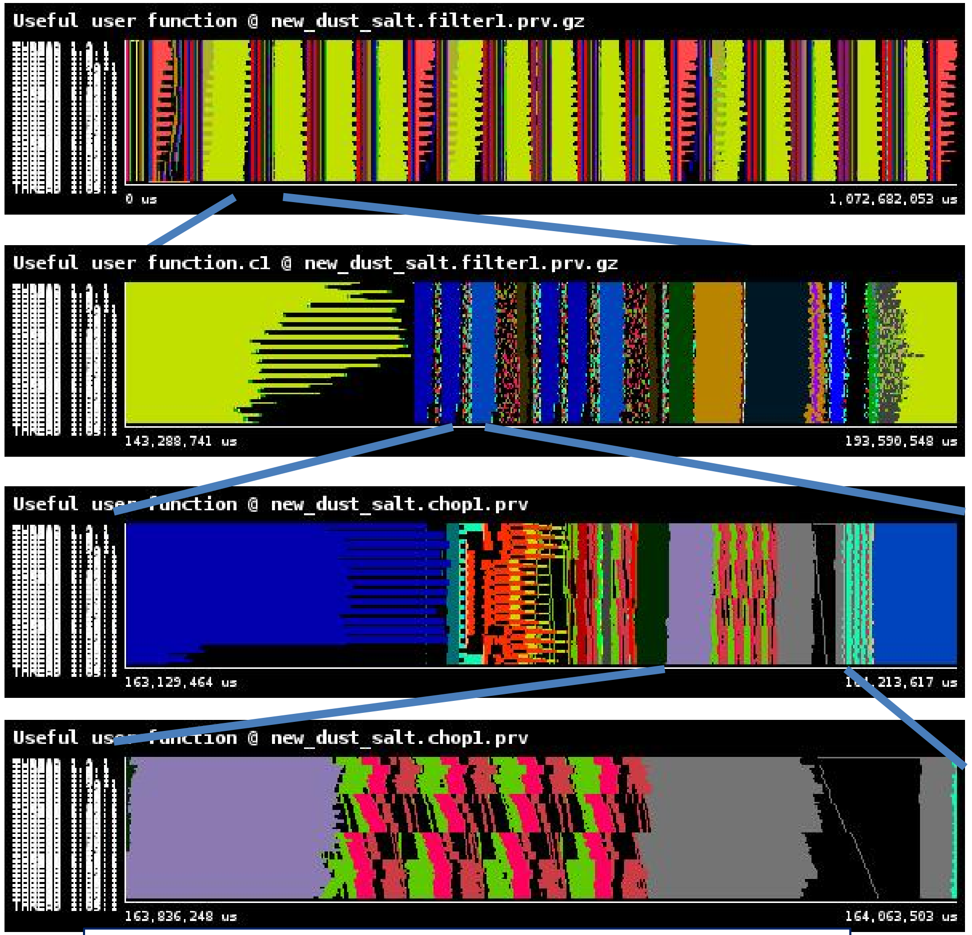
A “different” view point

- … and fundamental metrics

$$\eta_{\|} = LB * Ser * Trf$$

LB	Ser	Trf	Eff
0.83	0.97		0.80
0.87	0.90		0.78
0.88	0.97	0.84	0.73
0.88	0.96	0.75	0.61

Useful user function @ NMMB

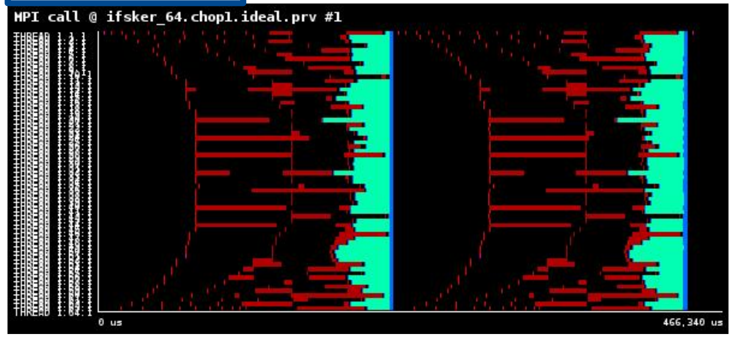

Useful user function @ new_dust_salt.filter1.prv.gz
0 us
1,072,682,053 us
Useful user function.c1 @ new_dust_salt.filter1.prv.gz
143,288,741 us
193,590,548 us
Useful user function @ new_dust_salt.chop1.prv
163,129,464 us
164,213,617 us
Useful user function @ new_dust_salt.chop1.prv
163,836,248 us
164,063,503 us

adv2 (gather–fft-scatter)* mono

M. Casas et al, “Automatic analysis of speedup of MPI applications”. ICS 2008.

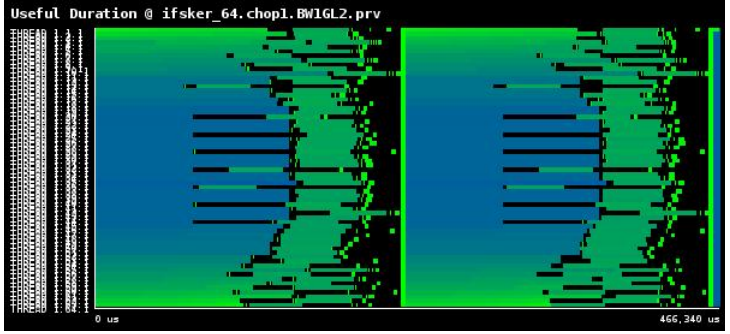
IFS_KERNEL structure and efficiency

MPI calls



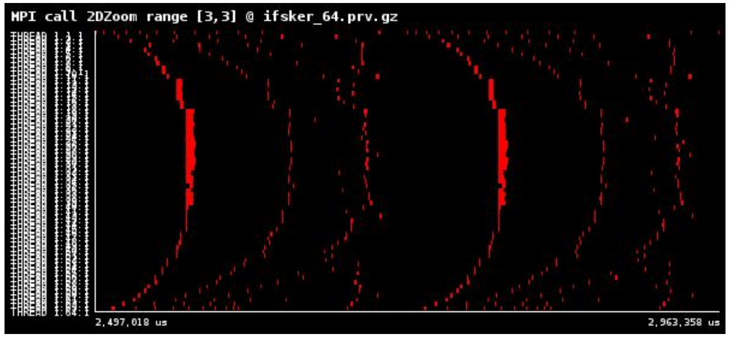
Useful = 0.73; MPI = 0.28

Useful duration

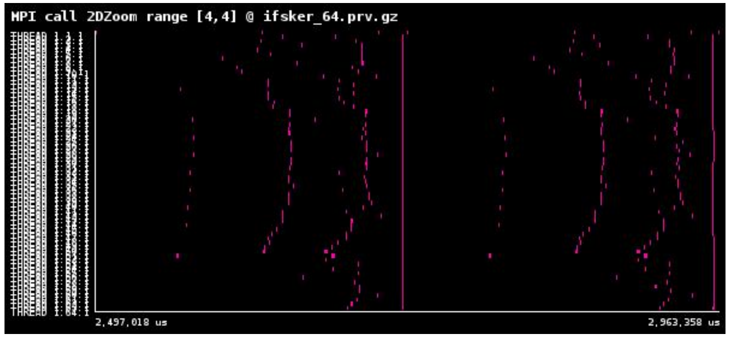


Eff = 0.73; LB = 0.79; Ser = 0.98; Trf = 0.94

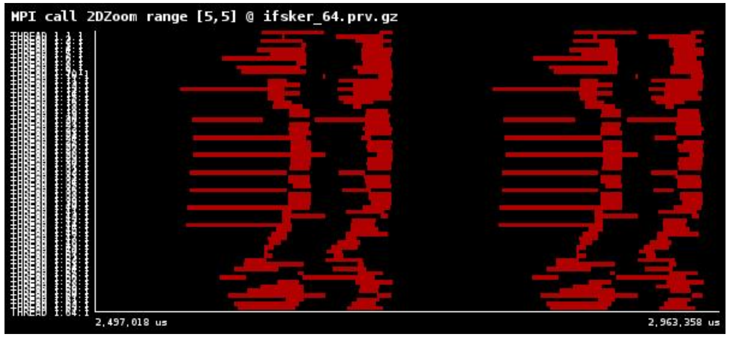
Isends



Irecvs



waits

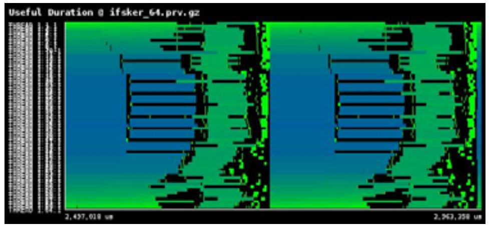


Sensitivity to network bandwidth

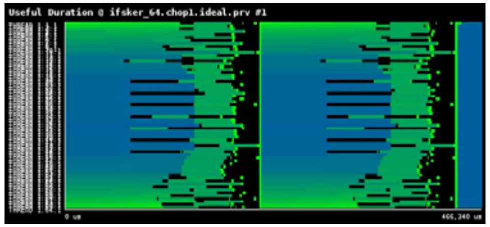
- Dimemas simulations

- Starts to be sensitive to bandwidth at below 500MB/s

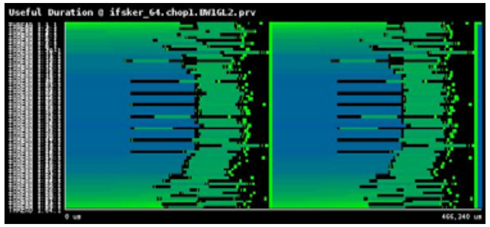
Real



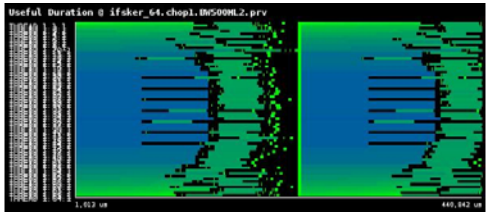
Ideal



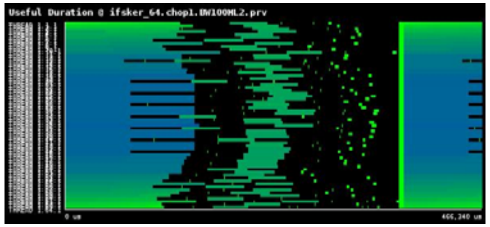
1 GB/s



500 MB/s



100 MB/s



Scalability

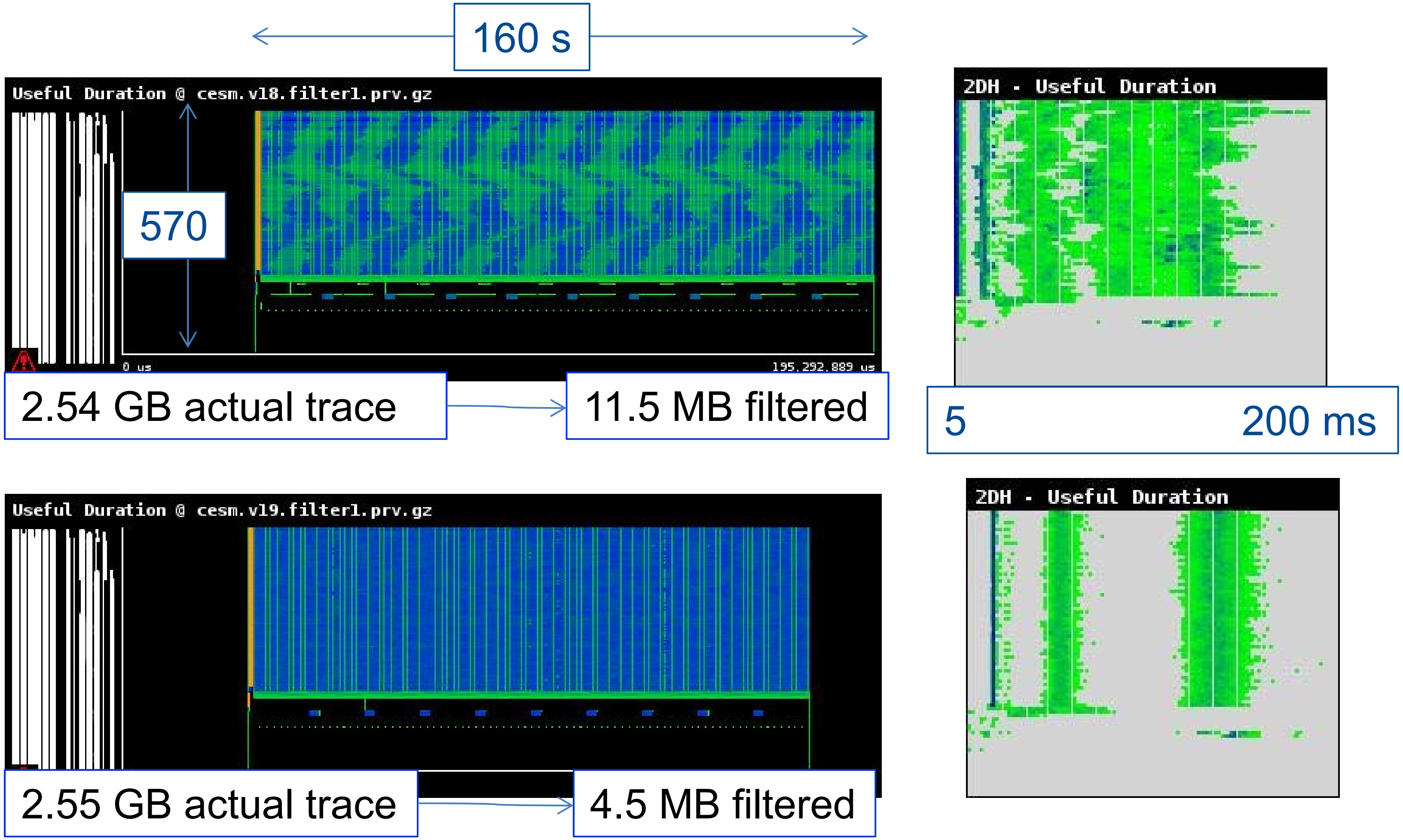
- Size
  - Handle decent time intervals and core counts
  - Instrumentation tracing modes …
    - Full
    - Burst
      - Precise characterization of long computation bursts
      - Summarized stats for sequences of short computation bursts
  - … + sampling
  - Paraver trace manipulation utilities
    - Filter and cutter
  - Paramedir: non GUI version of paraver (installed at tracing platform)
  - Practice:
    - Large trace never leaves tracing platform.
    - Paraver analysis on laptop

- Dynamic range
  - Handle/visualize events of very different duration

Trace manipulation utilities (filter)

- Understand Grid Distribution load balance impact @ CESM

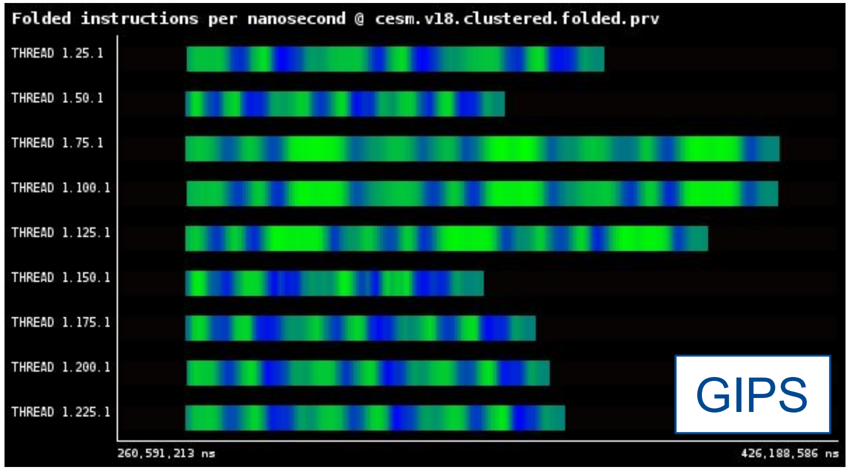
ATM:	384
LND:	16
ICE:	32
OCN:	10
CPL:	128

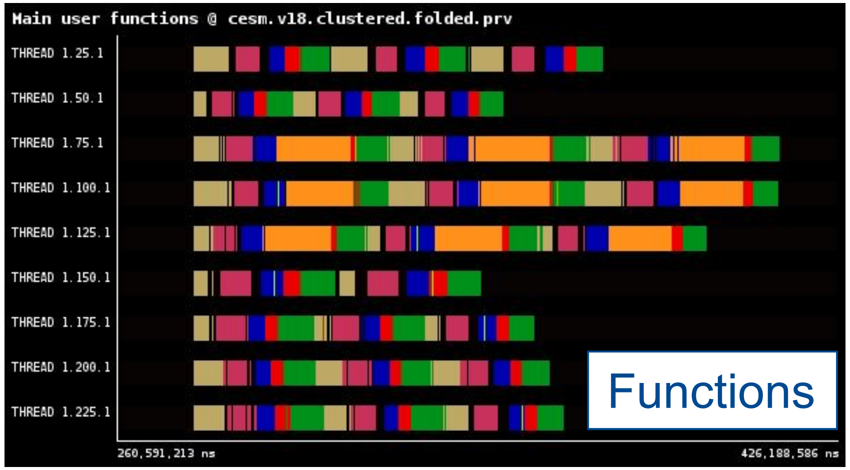


Instantaneous metrics at “no” cost

- Folding: Obtaining detailed information with minimal overhead
  - Instantaneous hardware counter metrics
  - Source behavioral structure: Structured time evolution of call stack
- Applicable to traces of large runs
  - Scripting support …
  - Orchestrating workflow of analytics algorithms based on clustering and folding functionalities …
  - … Integrated in Paraver GUI
  - More analytics being integrated

Subset of CESM @570

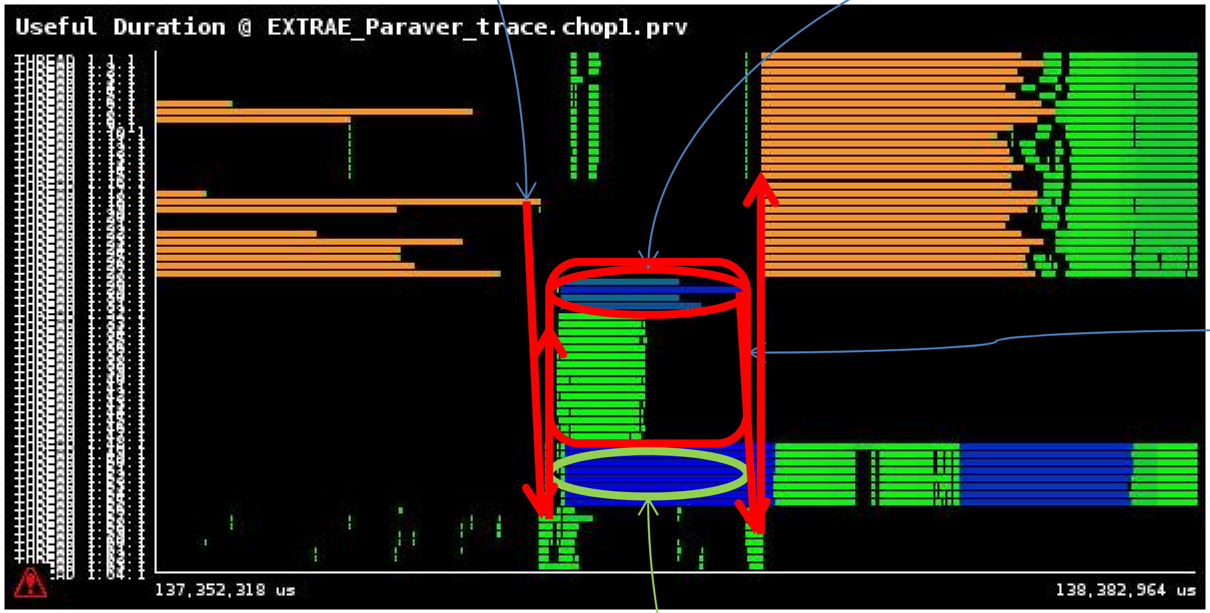

Folded instructions per nanosecond @ cesm.v18.clustered.folded.prv


Main user functions @ cesm.v18.clustered.folded.prv

🟨 Convect_shallow_tend	🟦 aer_rad_props_sw	🟧 rrtmg_sg
🟥 Microp_driver_tend	🟥 aer_rads_prop_lw	🟩 rad_rrtmg_lw

Paraver trace manipulation utilities (cut)

- To focus on detailed towards insight



Critical path

Imbalance within CLM

Imbalance between CLM and CICE

Longer computation in POP but not in critical path (does not communicate with Coupler at this point)

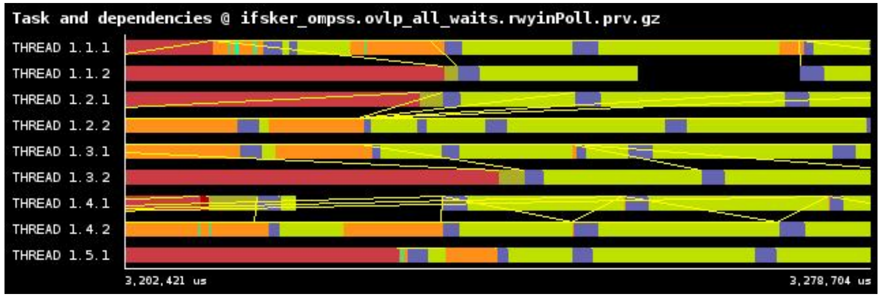
Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

OMPSS INSTRUMENTATION

OmpSs instrumentation

- Instrumented runtime … (leveraged flexible paraver format)
  - Tasks, dependences
  - Runtime internals: task creation, number, NANOS/DLB API, allocated cores,…

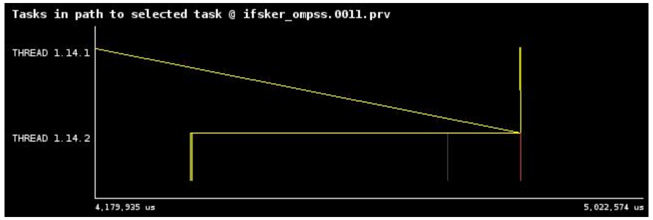
- Useful views
  - Tasks
  - Tasks and deps
  - Task not doing MPI
  - Task number
  - Creating/submitting
  - Waits
  - Critical



Task and dependencies @ ifsker_ompss.ovlp_all_waits.rwyinPoll.prv.gz

- Useful Paraver Features
  - Handle high dynamic range in task sizes: finding needles in haystacks
  - Complex derived views (i.e. Tasks not doing MPI)
  - Scripts to track dependencies
  - Big pixels, non linear rendering,…

- Potential input for OMPT

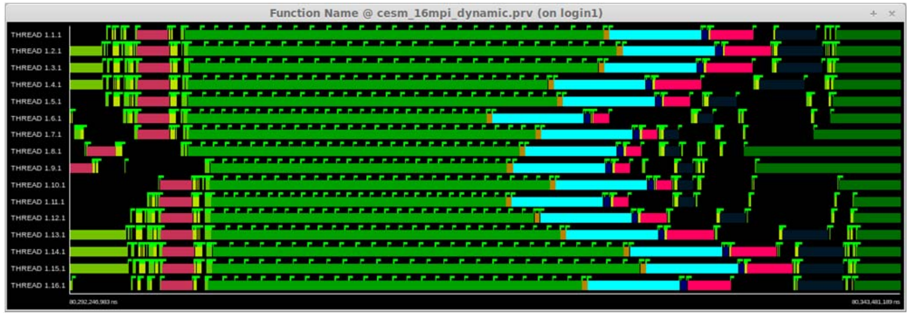


Tasks in path to selected task @ ifsker_ompss.0011.prv

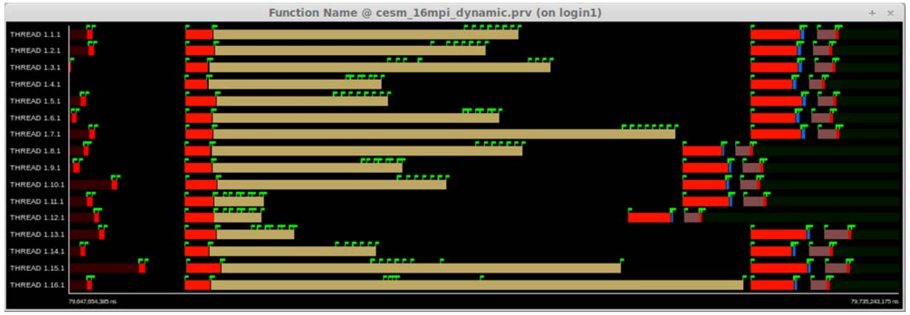
Programming model instrumentation

- Eases instrumentation
  - Original worksharing OpenMP pragmas (+ schedule dynamic)
  - MPI+OmpSs OMP_NUM_THREADS=1

- Work sharing loops @CESM
  - Micro load balance @ MPI level
  - Different internal structure
  - Impact on how to address it



~ uniform iteration cost



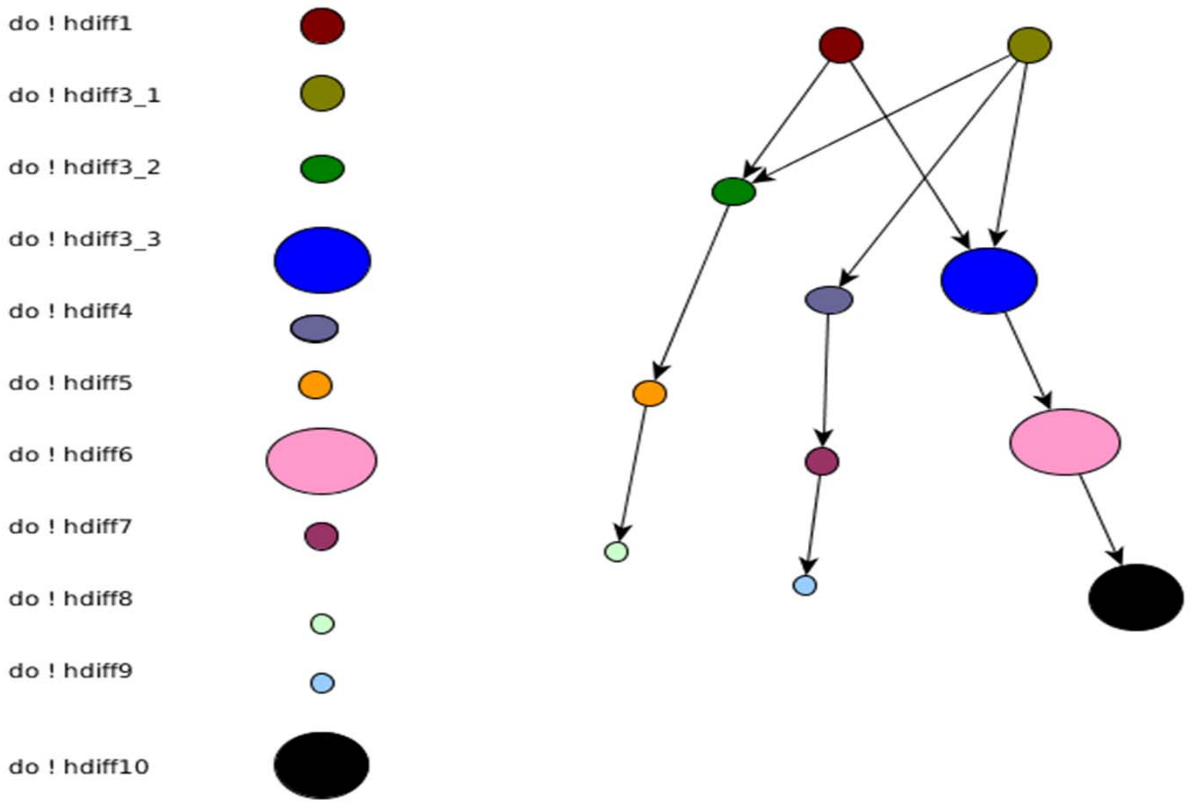
Non uniform iteration cost

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Programming model instrumentation

- Eases instrumentation
  - Task have structural semantics
  - !$OMP TASK **LABEL(XXX)** **DEFAULT(SHARED)** **IF(.FALSE.)**

Sequence of loops @ NMMB



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

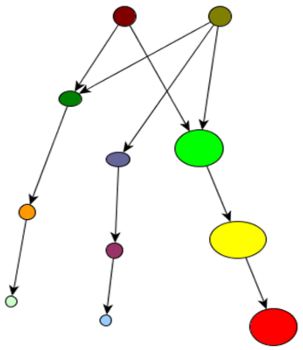
PROGRAMMING PATTERNS/PRACTICES

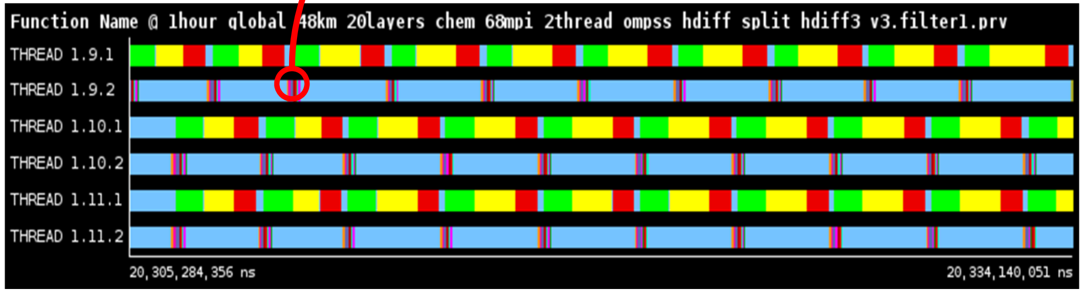
To overlap: what and how

- Computation - Communication?
- Computation - Computation?

- Syntactically simple?
  - Manually refactor code with quite unpredictable effects
    - Not very productive
  - OmpSs (OpenMP4.0):
    - Specify ordering constraints as IN/OUT pragmas
      - Productive
    - Interprocedural reorderings
      - High flexibility

Towards a top down parallelization



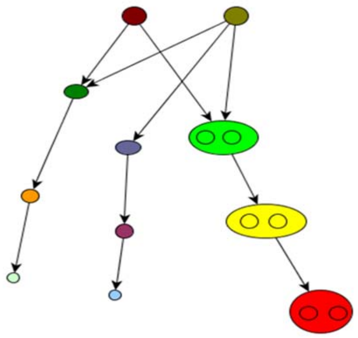


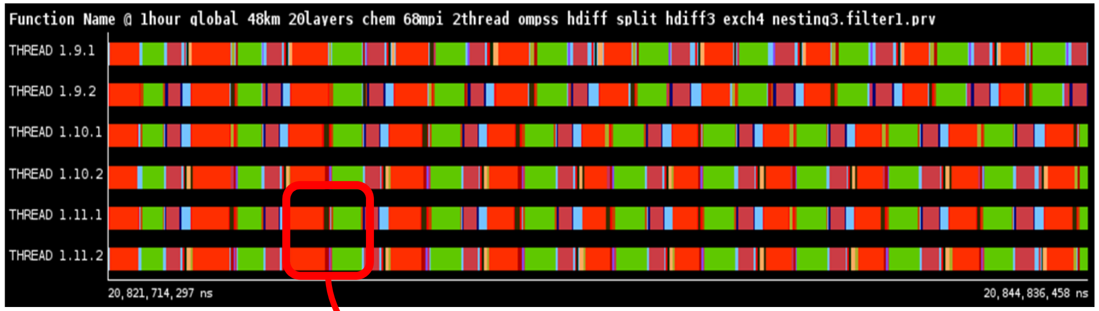
Function Name @ 1hour global 48km 20layers chem 68mpi 2thread ompss hdiff split hdiff3 v3.filter1.prv

THREAD 1.9.1
THREAD 1.9.2
THREAD 1.10.1
THREAD 1.10.2
THREAD 1.11.1
THREAD 1.11.2

20,305,284,356 ns
20,334,140,051 ns

Small tasks can be put outside of the critical path





Function Name @ 1hour global 48km 20layers chem 68mpi 2thread ompss hdiff split hdiff3 exch4 nesting3.filter1.prv

THREAD 1.9.1
THREAD 1.9.2
THREAD 1.10.1
THREAD 1.10.2
THREAD 1.11.1
THREAD 1.11.2

20,821,714,297 ns
20,844,836,458 ns

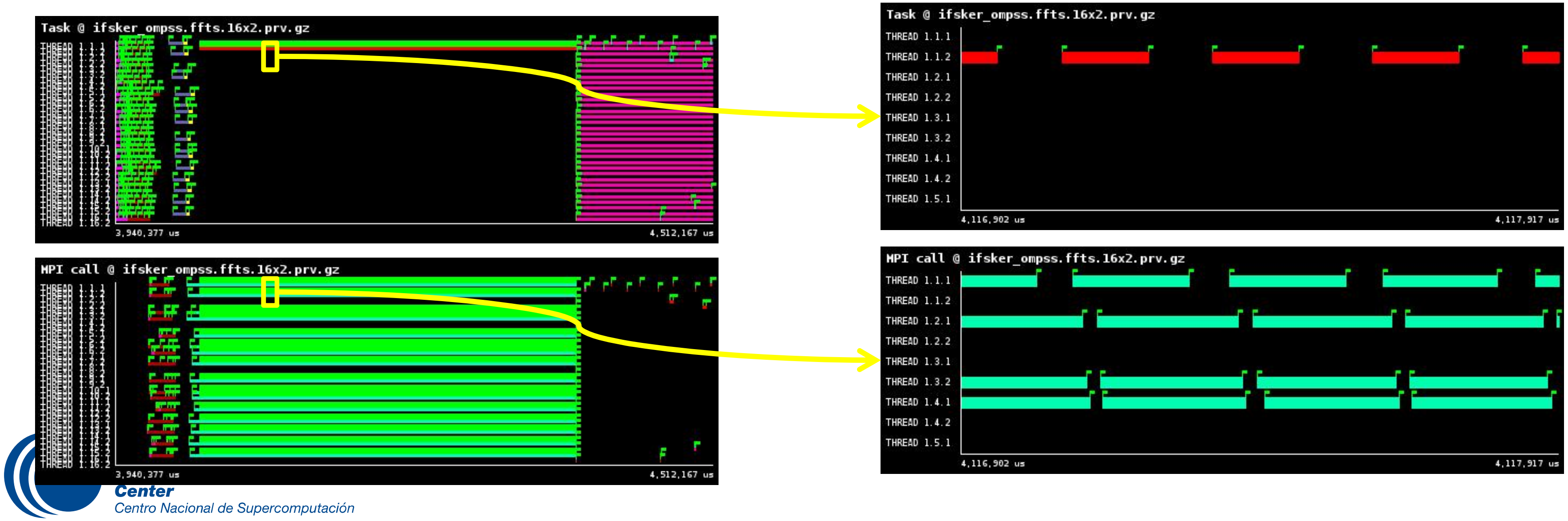
Big task can be workshared (nested) (30% gain)

- All levels contribute
- Address granularity issues of single level parallelization

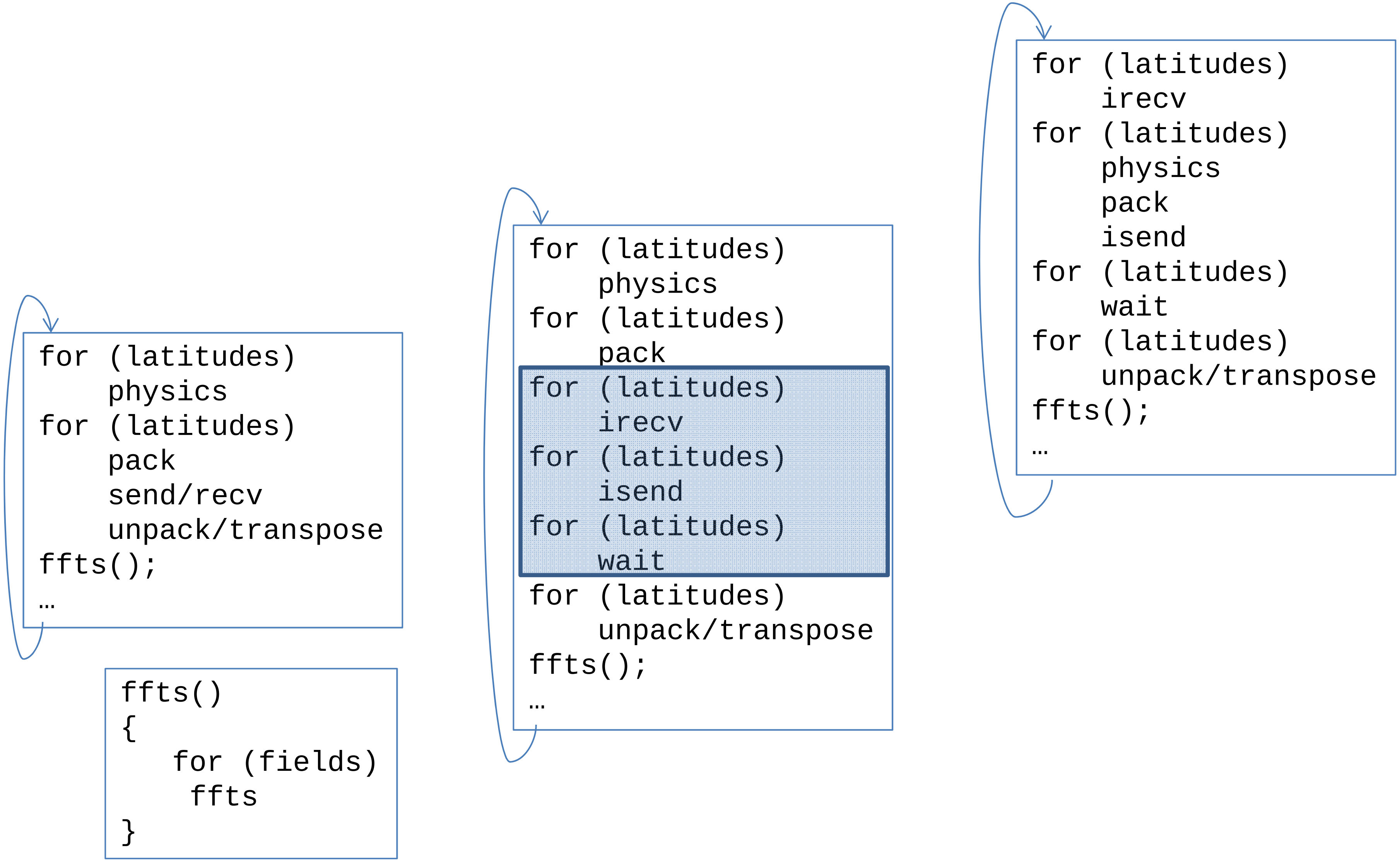
“Background” computation and I/O overlap

- Communication - computation or I/O sequences
- Instrumentation quantifies relevance
  - Pattern often generates MPI imbalance
- Spawning tasks achieves “background” execution
  - FIRSTPRIVATE does useful memory management

```
do jv=1,nvars2d
   ifld=ifld+1
   do j=1,ngptot
      znorms(j)=zgp(ifld,j)
   enddo
   call mpi_gatherv(znorms(:),ngptot,MPI_REAL8,znormsg(:),…)
   if( myproc==1 )then
!$OMP TASK PRIVATE (zmin, zmax, zave) INOUT(ZDUM) &
!$OMP&      FIRSTPRIVATE(ngptotg, nstep, jv, znormsg) &
!$OMP&      DEFAULT(NONE) LABEL(MIN_MAX)
      zmin=minval(znormsg(:))
      zmax=maxval(znormsg(:))
      zave=sum(znormsg(:))/real(ngptotg)
      write(*,…) nstep,jv,zmin,zmax,zave
!$OMP END TASK
   endif
 enddo
```



To overlap: what and how



```
for (latitudes)
    physics
for (latitudes)
    pack
    send/recv
    unpack/transpose
ffts();
…
```

```
ffts()
{
   for (fields)
    ffts
}
```

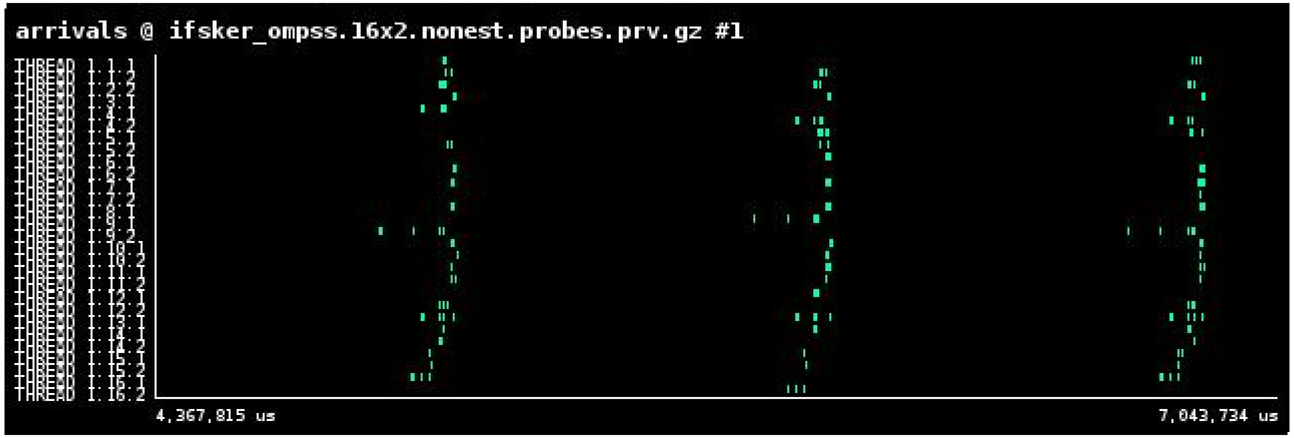
```
for (latitudes)
    physics
for (latitudes)
    pack
for (latitudes)
    irecv
for (latitudes)
    isend
for (latitudes)
    wait
for (latitudes)
    unpack/transpose
ffts();
…
```

```
for (latitudes)
    irecv
for (latitudes)
    physics
    pack
    isend
for (latitudes)
    wait
for (latitudes)
    unpack/transpose
ffts();
…
```

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Communication schedule issues

- User specified order of waits vs. order of arrivals?
- How to visualize? Quantify?
  - Used polling and fake MSG_READY task (print msg)
    - 0.0177% of time
    - Count is important
      - Within 640 waits 575 times other msgs are ready
    - Position IS important !!!
      - When do messages arrive. Worthwhile to reschedule? Repetitive?
      - → scheduling issue → programming model/runtime (co)design
      - Need to find needles in haystacks



tasks

waits

Arrived while waiting for other

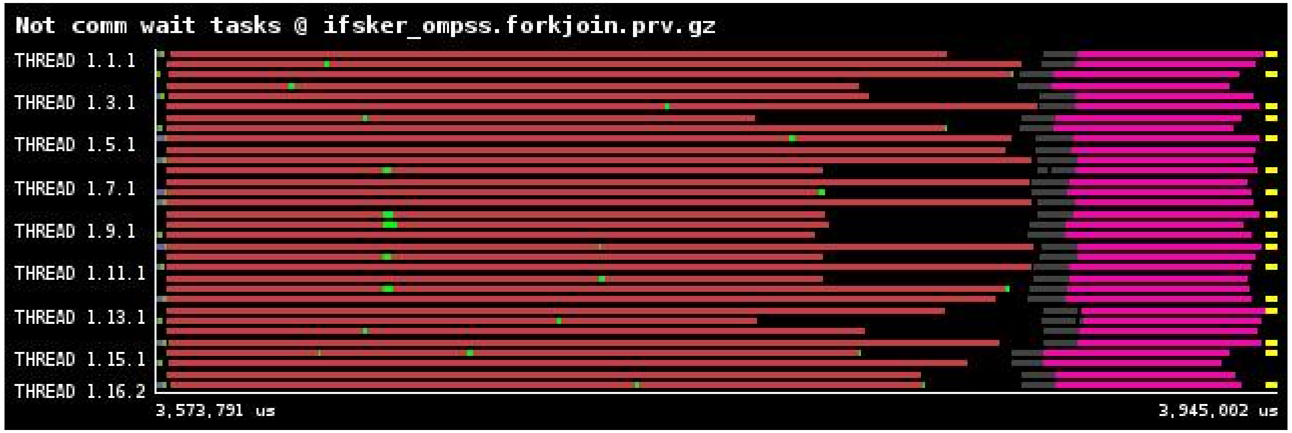
Communication schedule issues

- How to address?
  - Application level
    - Change issue order of calls. Need detailed knowledge of communication pattern, machine characteristics, runtime behavior,
    - … might not be feasible
  - Application – task runtime codesign
    - Out of order/concurrent execution of communication tasks
      - Potential deadlock. Impose some order that does ensure no deadlock
      - Critical or MPI_THREAD_MULTIPLE
    - Similar scheduling issues → codesign choices
      - Polling + Nanos_yield + multiple concurrent wait tasks
      - …
  - Runtime level
    - Codesign MPI and task runtimes

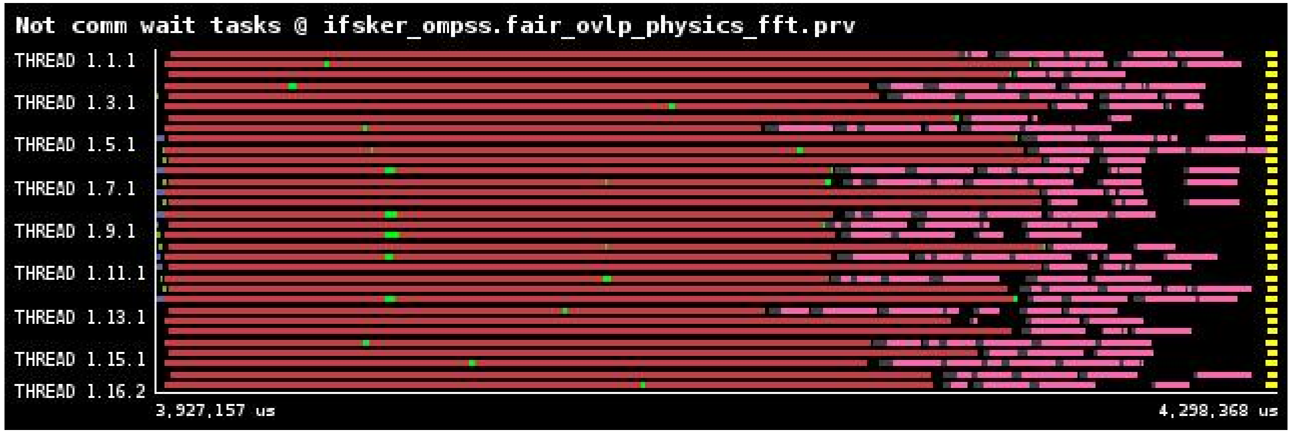
To overlap: what and how

tasks (excluding communication tasks)

Sequential



Out of order execution



Communication schedule issues

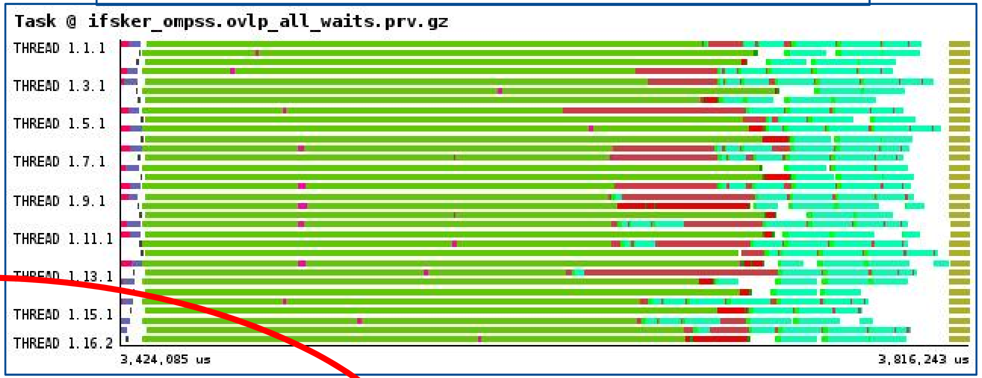
- How to address?

  - Application – task runtime codesign
    - Out of order/concurrent execution of communication tasks
      - Potential deadlock. Impose some order that does ensure no deadlock
      - Critical or MPI_THREAD_MULTIPLE
    - Similar scheduling issues → codesign choices
      - Polling + Nanos_yield + multiple concurrent wait tasks
      - …

Scheduling issues

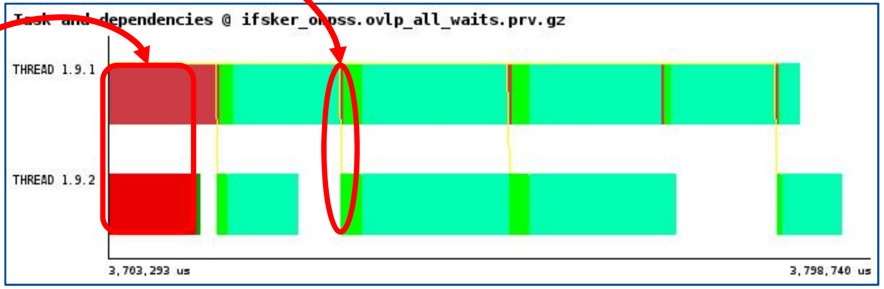
- Between MPI and computation

Overlap waits for recvs and sends



Wait for reception vs fft computation

Simultaneous wait for two MPI requests (progression engine issue)

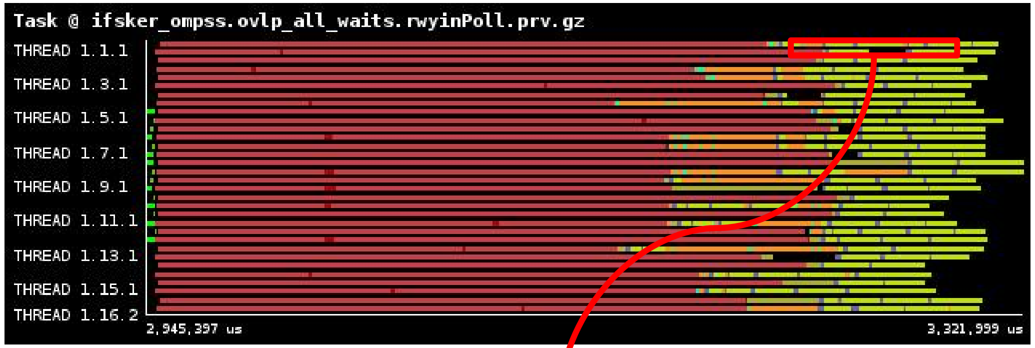


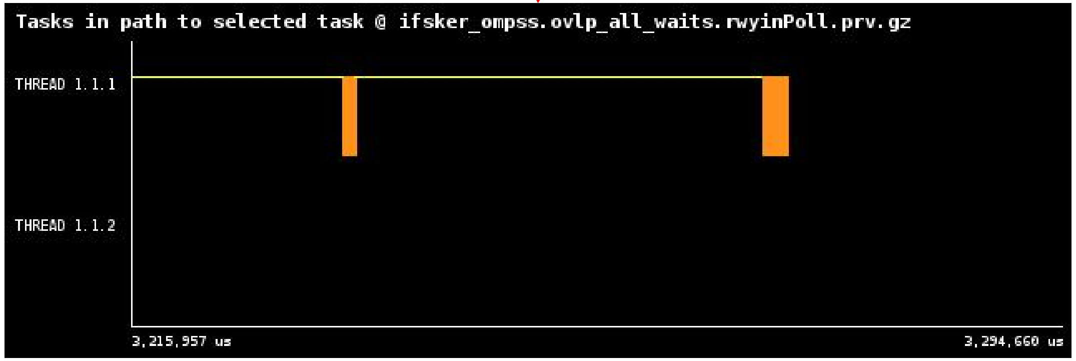
- Need for codesign of MPI and OmpSs runtimes
- Need to see details and gain insight

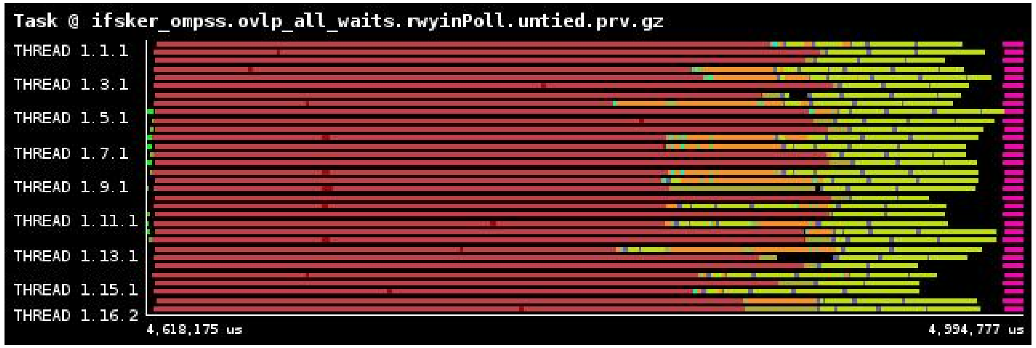
Scheduling issues

- Issues can be very varied
  - Communication task yields
  - Default untied tasks

- Solutions too
  - Declare communication task untied




Tasks in path to selected task @ ifsker_ompss.ovlp_all_waits.rwyinPoll.prv.gz
THREAD 1.1.1
THREAD 1.1.2
3,215,957 us
3,294,660 us



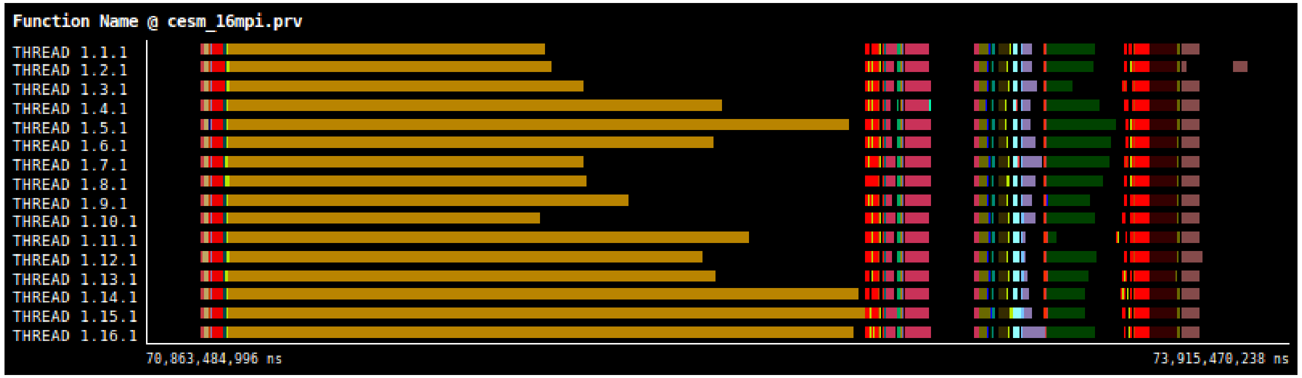
Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

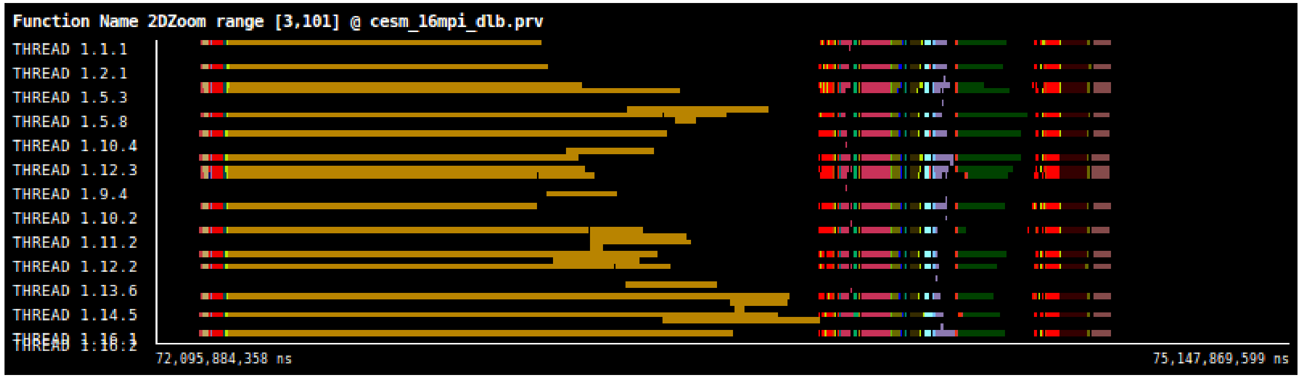
DLB

CESM and DLB

- Place DLB API calls after the most unbalanced for loops
  - DLB_Lend / DLB_Retrieve

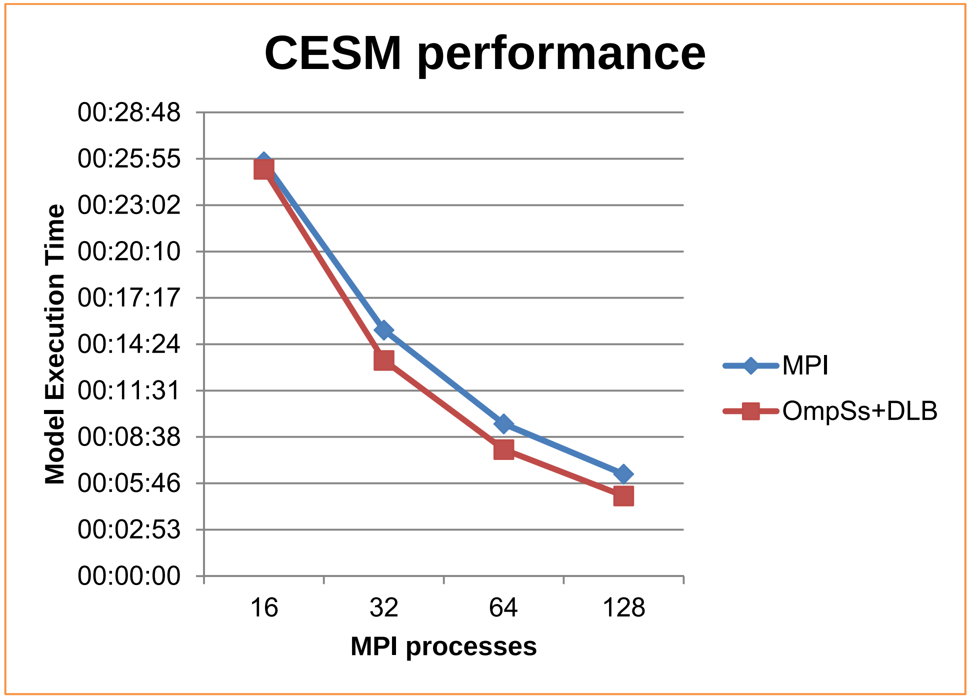
- Same scale:

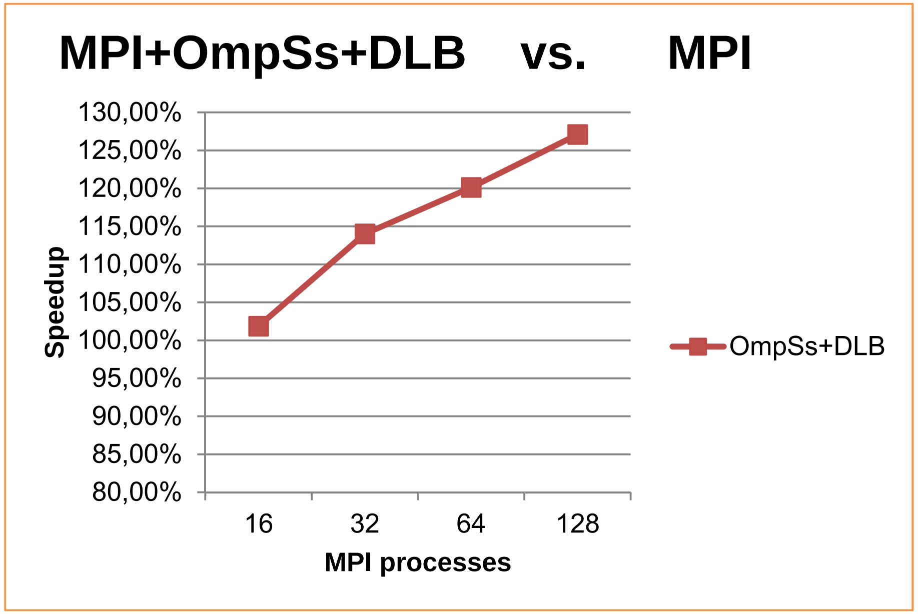




CESM performance results

- DLB total improvement is proportional to application load unbalance
- But the performance depends on the malleability of the second level of parallelism



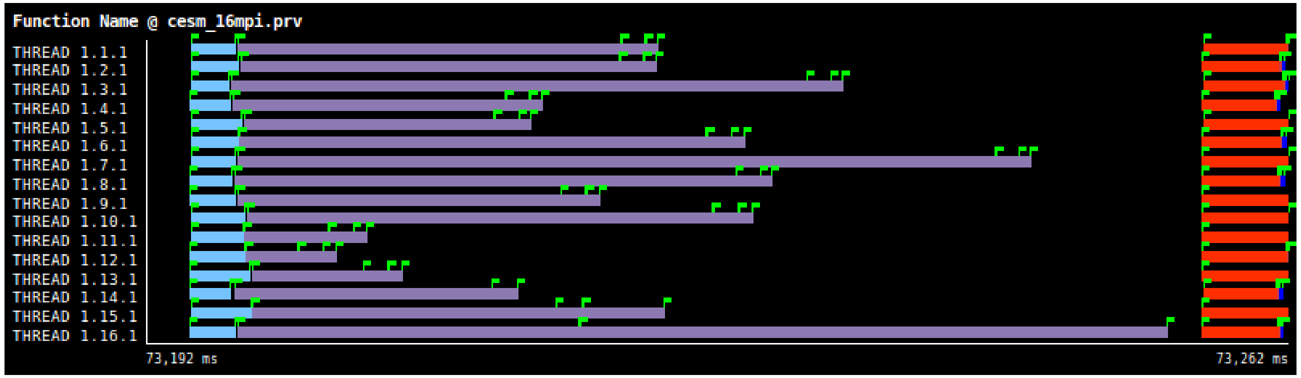


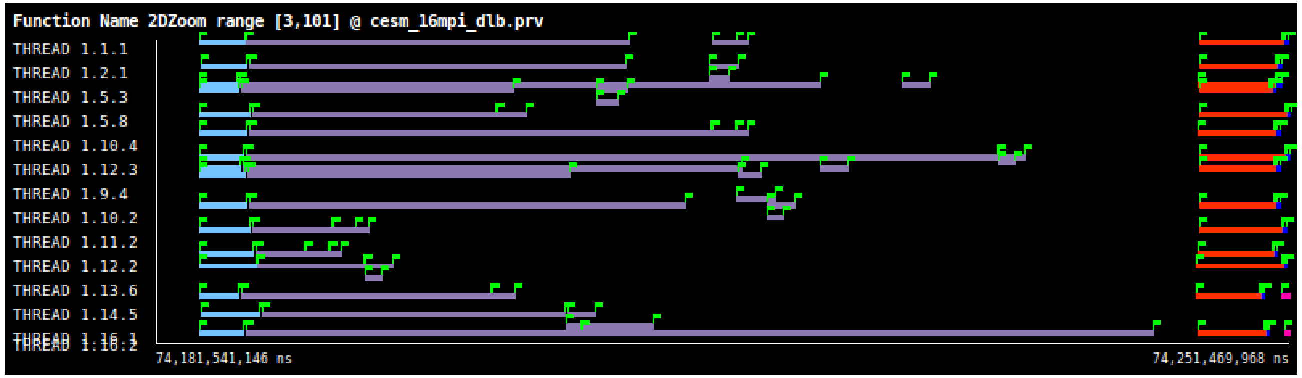
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

CESM and DLB

- Dynamic Load Balance needs malleability!
  - Uneven or serialized tasks prevent the efficient load balance

- Same scale:





Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

CONCLUSION

Conclusion

- Tools needed for informed incremental parallelization and real insight into behavior

- Task based models:
  - Easy to introduce significant changes in restructuring of code execution
  - Good and a risk
    - Scheduling: a very non linear behavior → Intricate relationship between components and their interactions
    - A good transformation may be hidden by another behavior. Moving bottlenecks
    - Need detailed tools to properly identify and detect new unexpected behaviors, bottlenecks,…

- Production Climate code
  - A challenge … affordable

- Potential/Need to co-design
  - applications ↔ tools ↔ programming models
  - Between programming model runtimes (MPI↔OmpSs)

Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

THANKS