A PGAS implementation by co-design of the ECMWF Integrated Forecasting System

George Mozdzynski , ECMWF

Acknowledgements

Mats Hamrud	ECMWF
Nils Wedi	ECMWF
Jens Doleschal	Technische Universität Dresden
Harvey Richardson	Cray Research UK

And my other partners in the CRESTA Project

The CRESTA project has received funding from the EU Seventh Framework Programme (ICT-2011.9.13)

What is CRESTA - see http://cresta-project.eu/

- Collaborative Research into Exascale Systemware, Tools and Applications
- EU funded project, 3 years (year 1 just completed), ~ 50 scientists
- Six co-design vehicles (aka applications)
  - ELMFIRE (CSC, ABO,UEDIN) - fusion plasma
  - GROMACS (KTH) - molecular dynamics
  - HEMELB (UCL) - biomedical
  - IFS (ECMWF) - weather
  - NEK5000 (KTH) & OPENFOAM (USTUTT, UEDIN) - comp. fluid dynamics
- Two tool suppliers
  - ALLINEA (ddt : debugger) & TUD (vampir : performance analysis)
- Technology and system supplier – CRAY UK
- Many Others (mostly universities)
  - ABO, CRSA, CSC, DLR, JYU, KTH, UCL, UEDIN-EPCC, USTUTT-HRLS

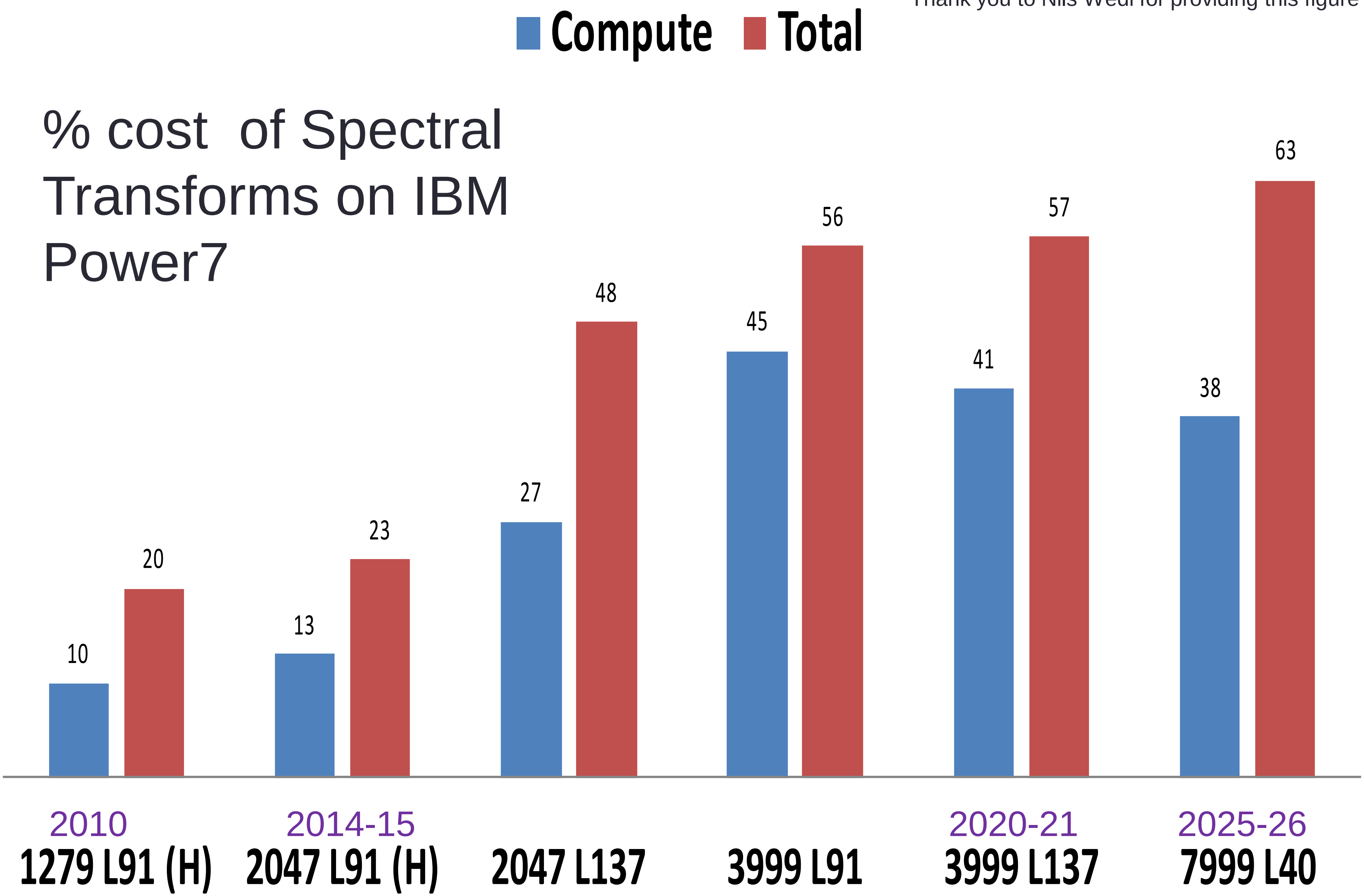
IFS model: current and planned model resolutions

IFS model resolution	Envisaged Operational Implementation	Grid point spacing (km)	Time-step (seconds)	Estimated number of cores[1]
T1279 H[2]	2010 (L91) 2012 (L137)	16	600	1100 1600
T2047 H	2014-2015	10	450	6K
T3999 NH[3]	2020-2021	5	240	80K
T7999 NH	2025-2026	2.5	30-120	1-4M

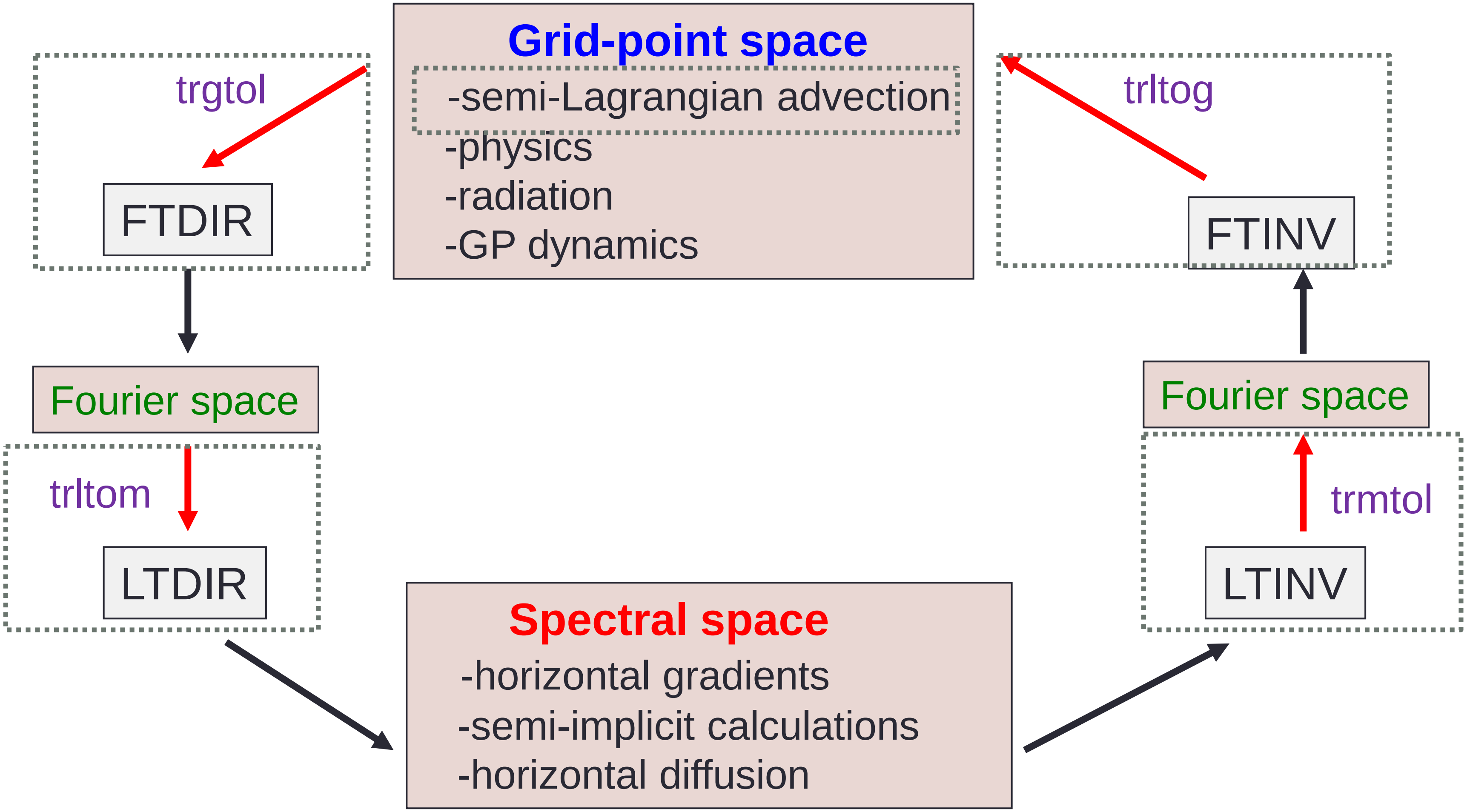
1 - a gross estimate for the number of ‘Power7’ equivalent cores needed to achieve a 10 day model forecast in under 1 hour (~240 FD/D), system size would normally be 10 times this number.
2 – Hydrostatic Dynamics
3 – Non-Hydrostatic Dynamics

Thank you to Nils Wedi for providing this figure

% cost of Spectral Transforms on IBM Power7



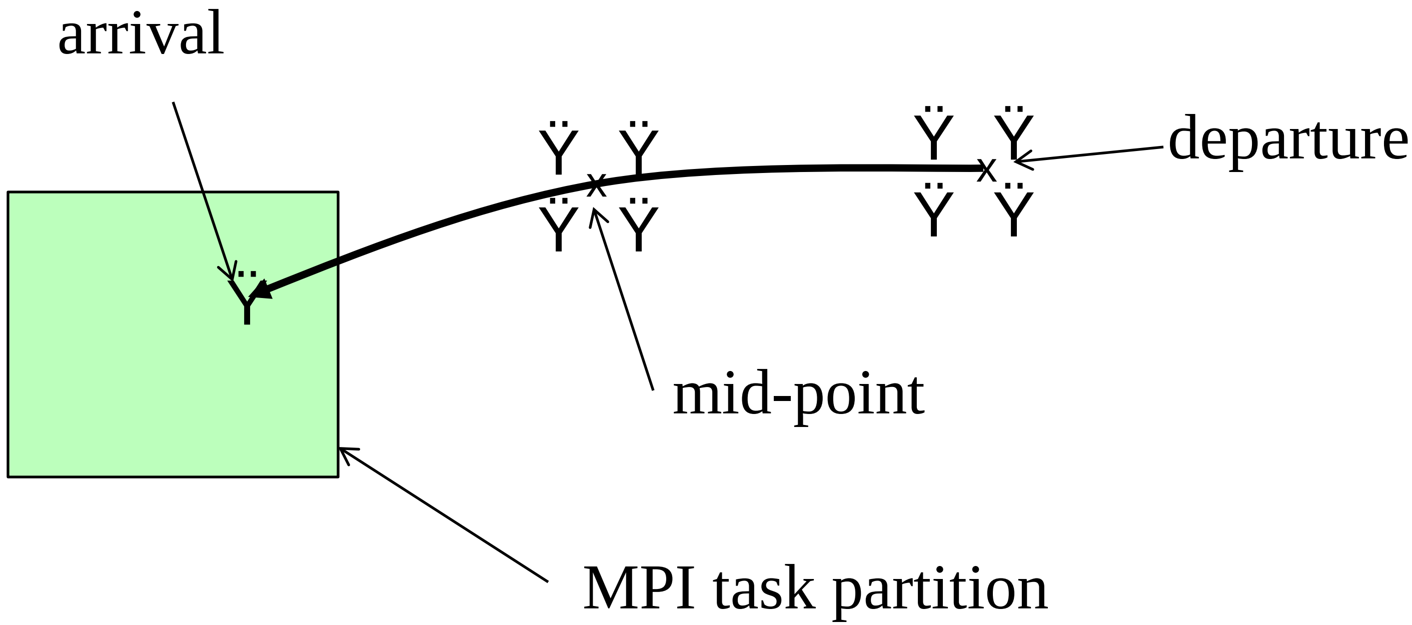
Planned IFS optimisations for [Tera,Peta,Exa]scale



CREST

Semi-Lagrangian Transport

- **Computation of a trajectory from each grid-point backwards in time, and**
- **Interpolation of various quantities at the departure and at the mid-point of the trajectory**

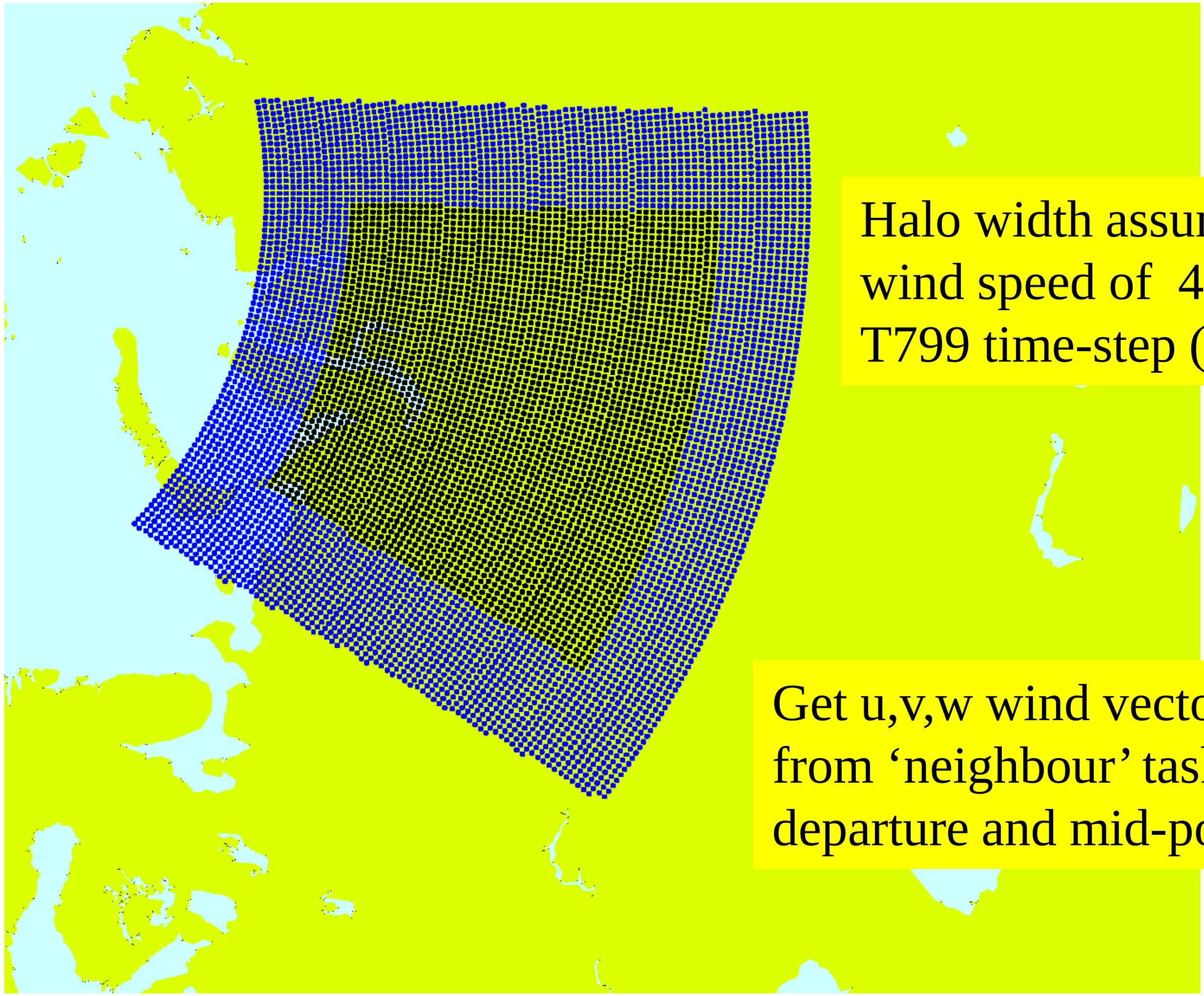


Semi-Lagrangian Transport: T799 model, 256 tasks

Task 11 encountered the highest wind speed of 120 m/s (268 mph) during a 10 day forecast starting 15 Oct 2004

ECMWF

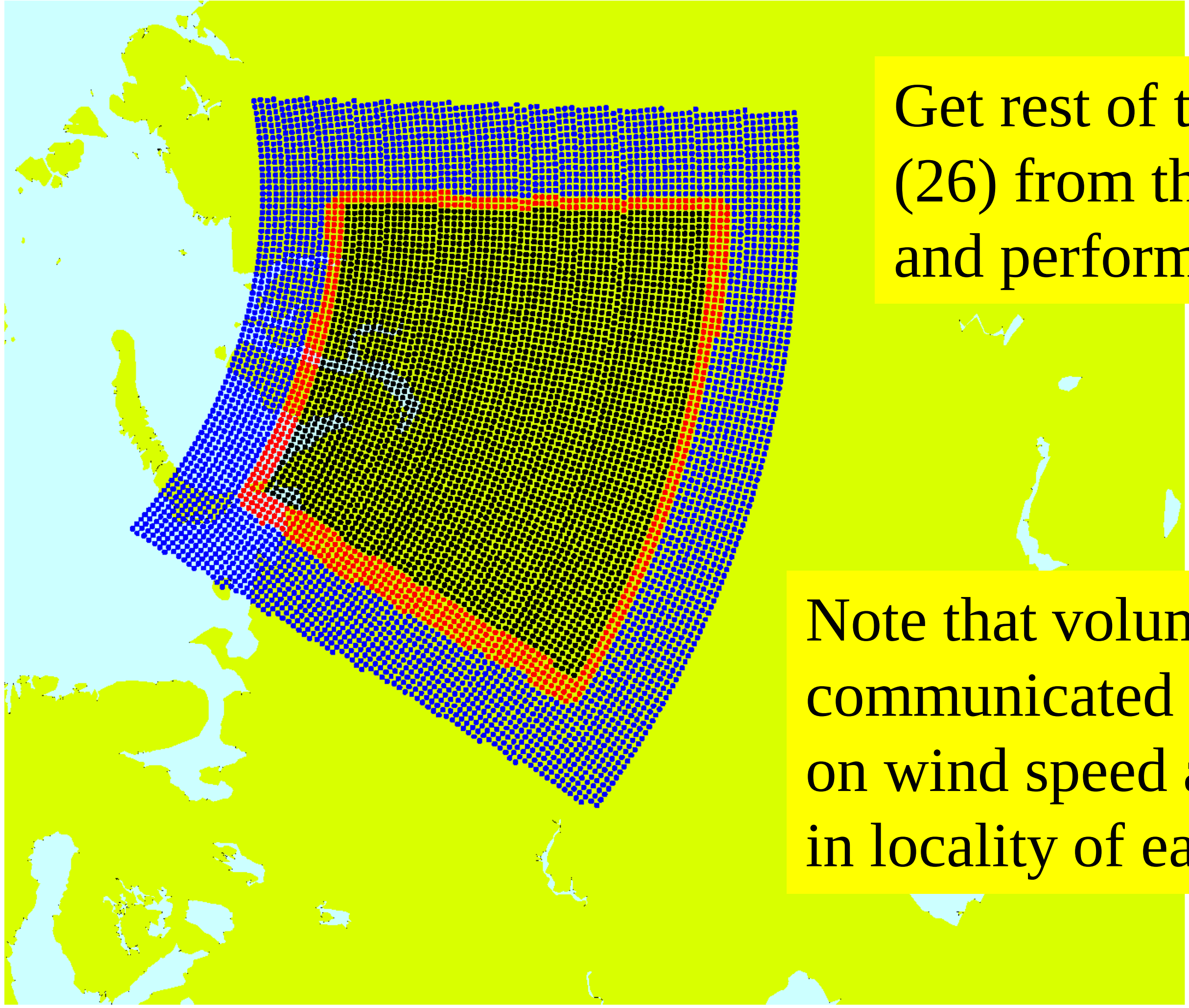
blue: halo area



Halo width assumes a maximum wind speed of 400 m/s x 720 s T799 time-step (288 km)

Get u,v,w wind vector variables (3) from 'neighbour' tasks to determine departure and mid-point of trajectory

red: halo points actually used



Get rest of the variables (26) from the red halo area and perform interpolations

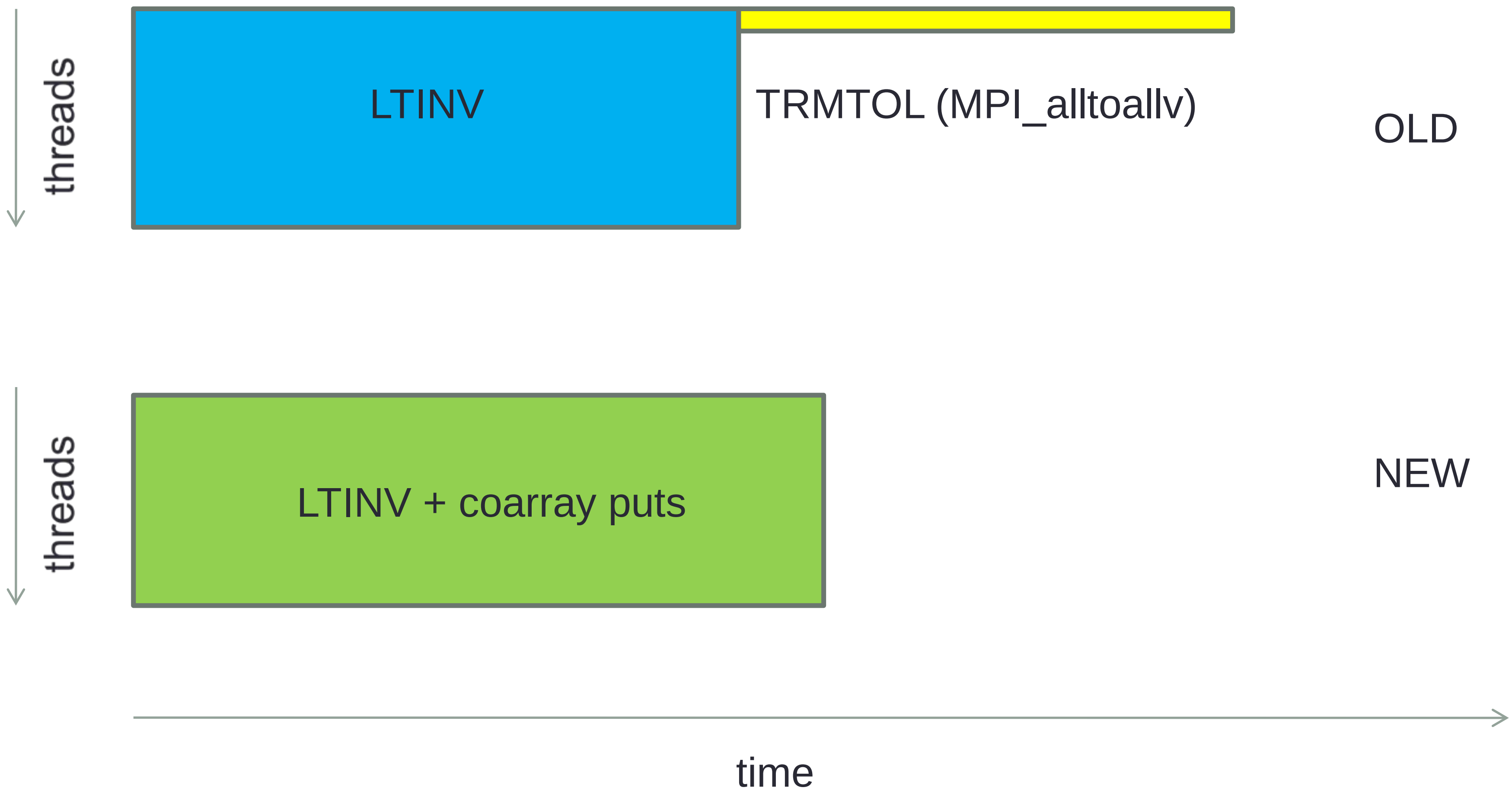
Note that volume of halo data communicated is dependent on wind speed and direction in locality of each task

IFS Optimisations for ExaScale & Co-design

- All currently planned IFS optimisations in the CRESTA project
  - Involve use of Fortran2008 coarrays (CAF)
  - Used within context of OpenMP parallel regions
- Overlap Legendre transforms with associated transpositions
- Overlap Fourier transforms with associated transpositions
- Rework semi-Lagrangian communications
  - To substantially reduce communicated halo data
  - To overlap halo communications with SL interpolations
- CAF co-design team
  - caf-co-design@cresta-project.eu
  - ECMWF – optimise IFS as described above
  - CRAY – optimize DMAPP to be thread safe
  - TUD – visualize CAF operations in IFS with vampir
  - ALLINEA – debug IFS at scale with ddt (MPI/OMP/CAF)

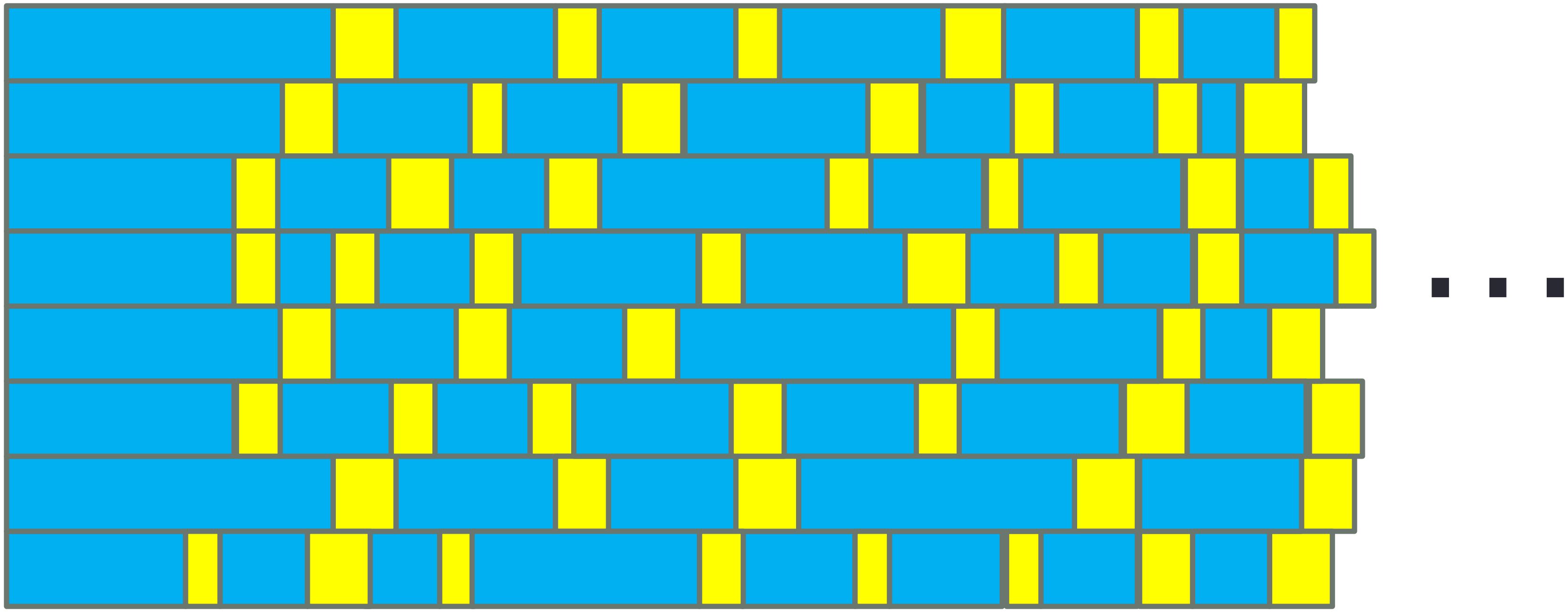
george.mozdzynski@ecmwf.int
mats.hamrud@ecmwf.int
harveyr@cray.com
michs@kth.se
tobias.hilbrich@tu-dresden.de
kostas@ihs.uni-stuttgart.de
m.bull@epcc.ed.ac.uk
jens.doleschal@tu-dresden.de
xaguilar@pdc.kth.se
david@allinea.com
jeremy@epcc.ed.ac.uk

Overlap Legendre transforms with associated transpositions



CREST

Overlap Legendre transforms with associated transpositions/3 (LTINV + coarray puts)



Expectation is that compute (LTINV-blue) and communication (coarray puts-yellow) overlap in time. We should be able to see this in the future with an extension to vampir being developed in CRESTA

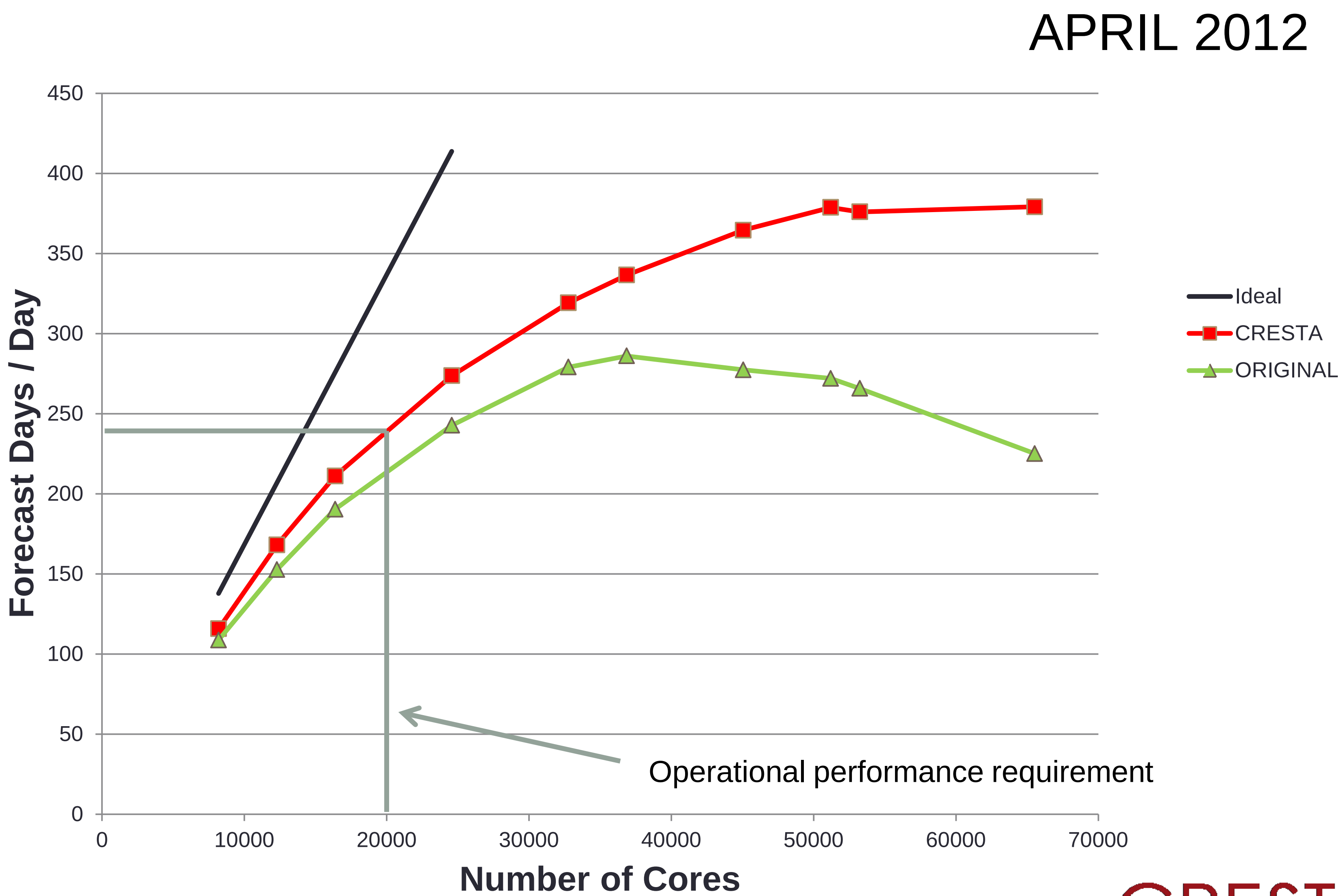
Semi-Lagrangian – coarray implementation

red: only the halo points that are used are communicated

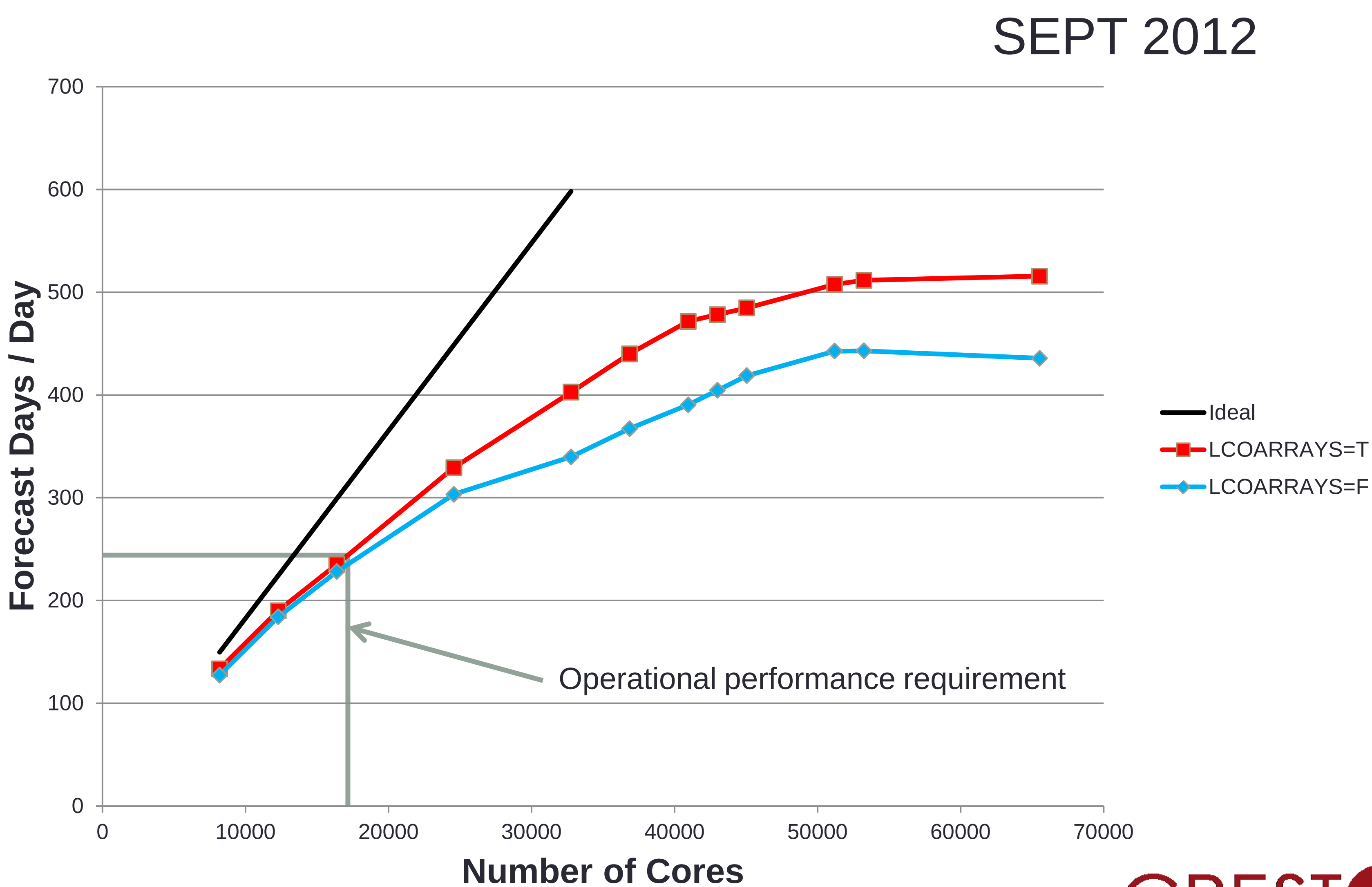
Note no more blue area (max wind halo) and associated overhead.

Also, halo coarray transfers take place in same OpenMP loop as the interpolations.

T2047L137 model performance on HECToR (CRAY XE6) RAPS12 IFS (CY37R3), cce=7.4.4

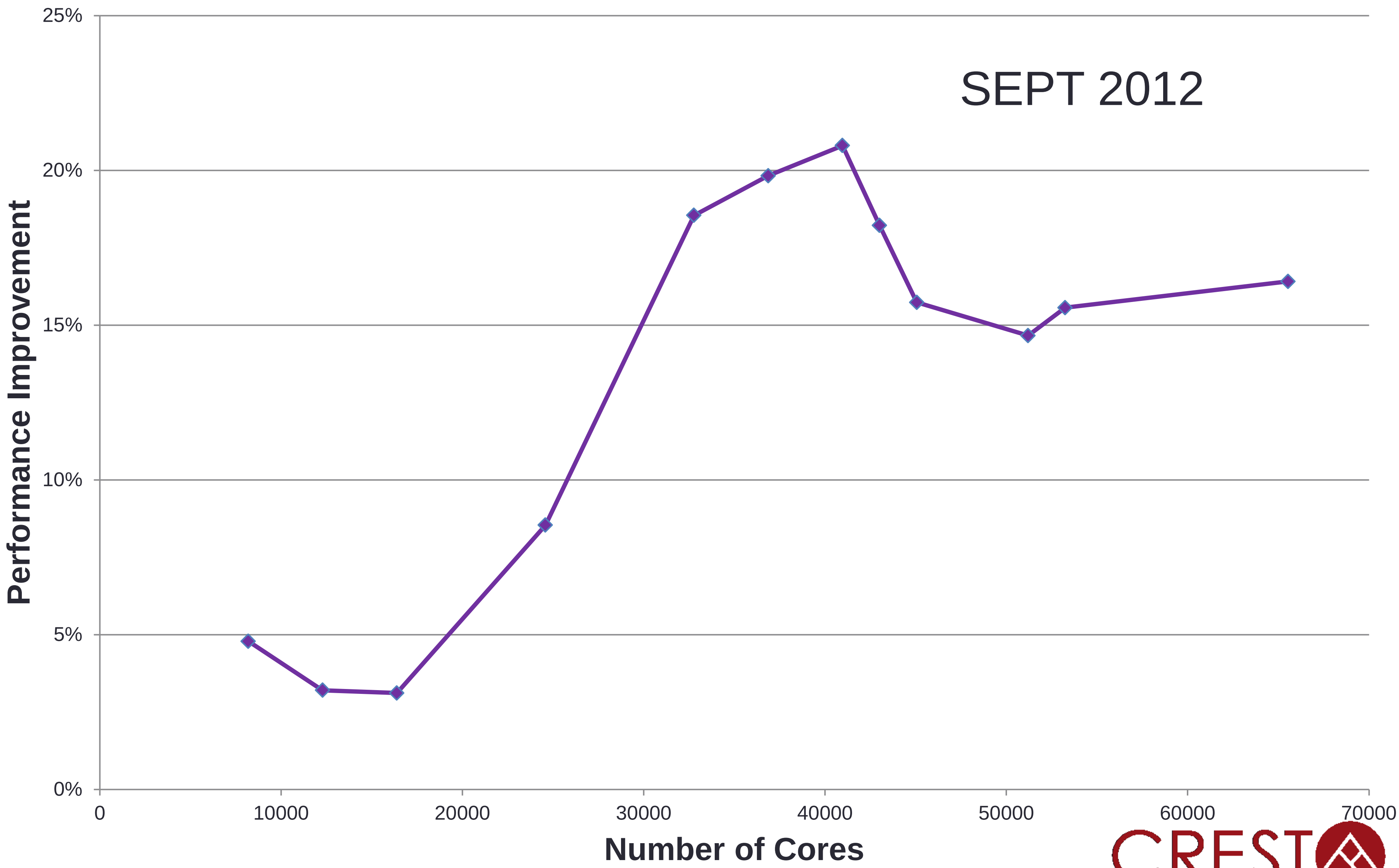


T2047L137 model performance on HECToR (CRAY XE6)
RAPS12 IFS (CY37R3), cce=8.0.6 -hflex_mp=intolerant

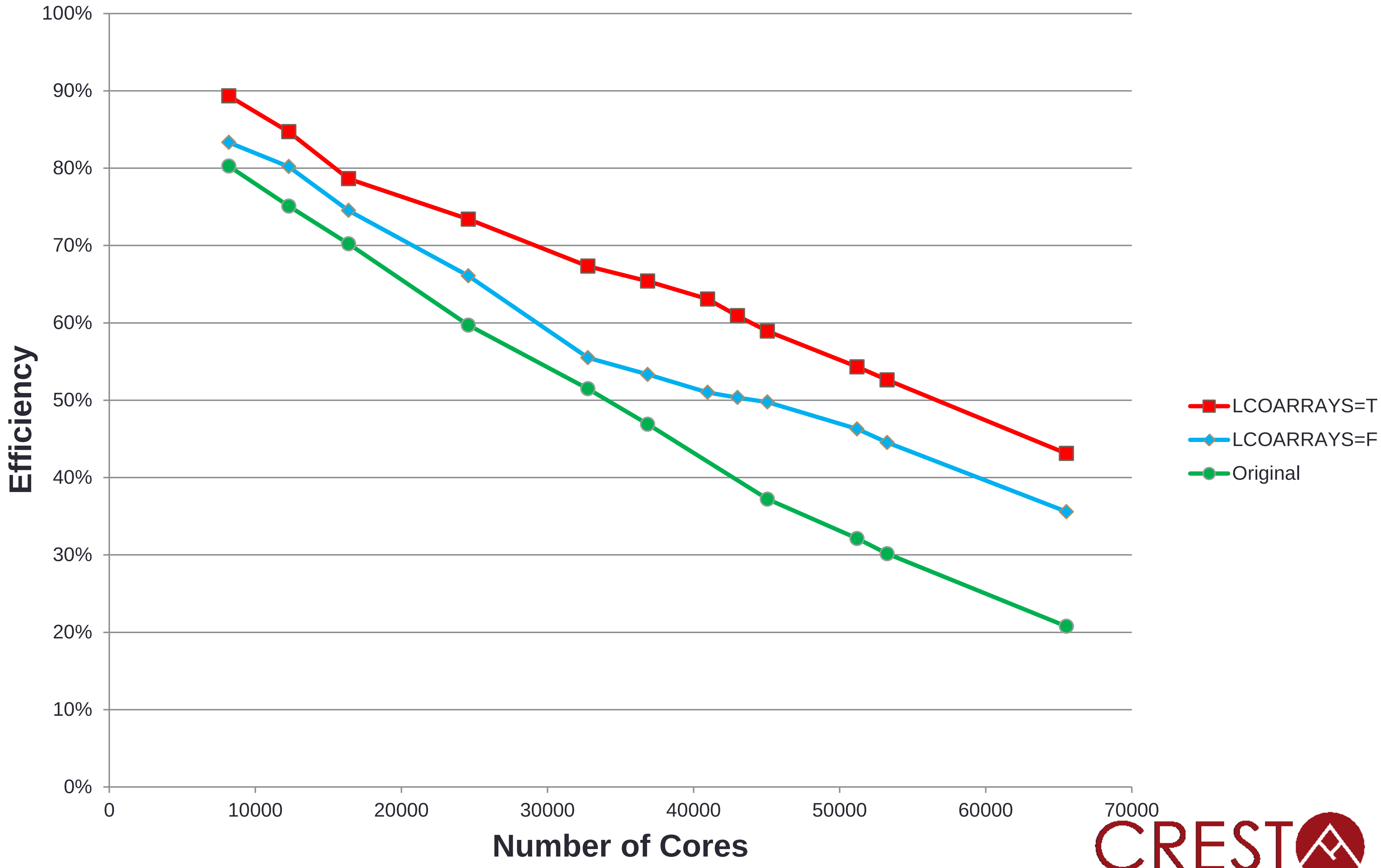


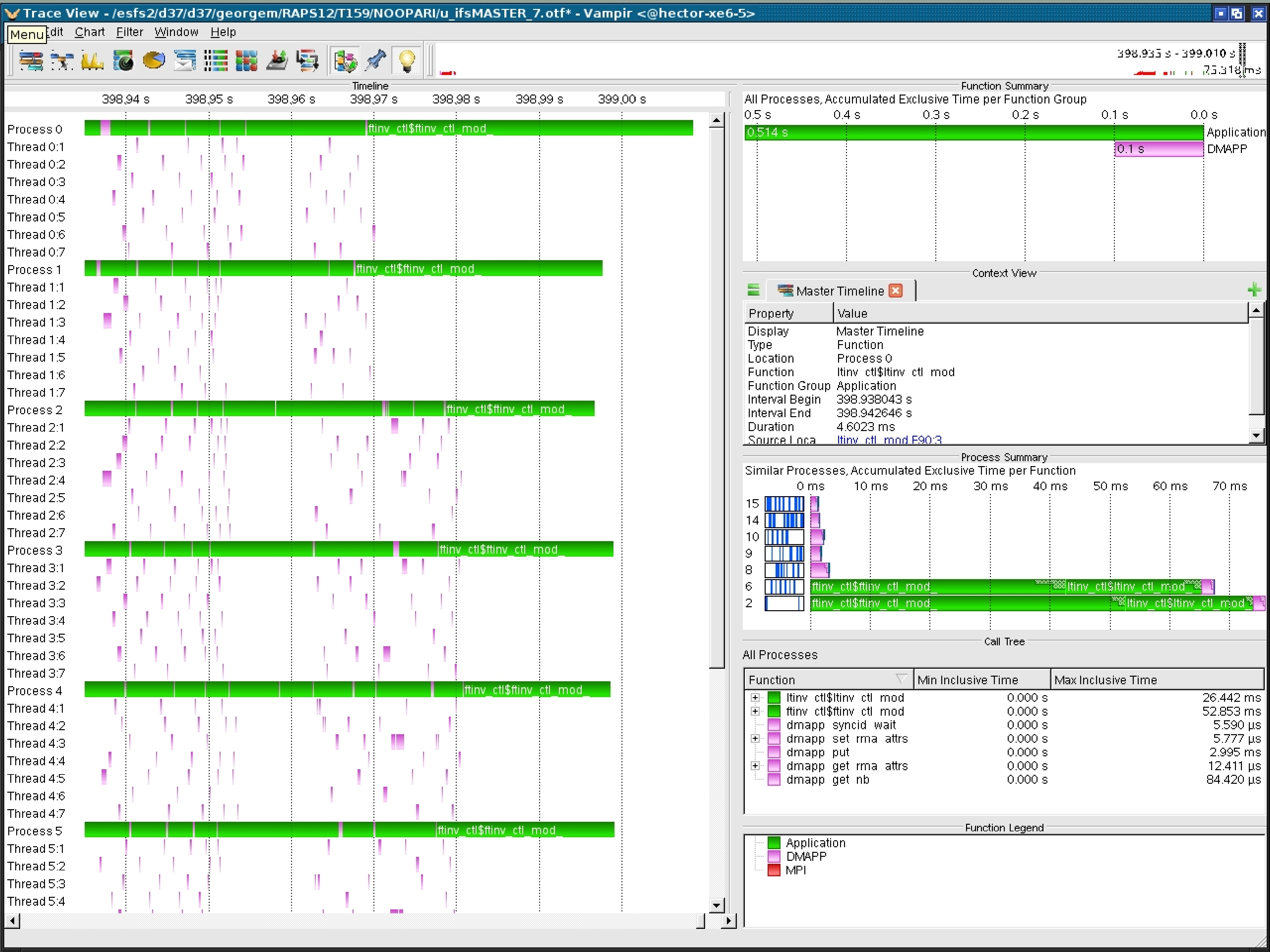
CREST

T2047L137 IFS model performance improvment by using Fortran2008 coarrays on HECToR (CRAY XE6)



T2047L137 model Efficiency on HECToR (CRAY XE6) RAPS12 IFS (CY37R3)




Trace View - /esfs2/d37/d37/georgem/RAPS12/T159/NOOPARI/u_ifsMASTER_7.otf* - Vampir <@hector-xe6-5>
Menu Edit Chart Filter Window Help
398.935 s - 399.010 s
75.318 ms
Timeline
398.94 s 398.95 s 398.96 s 398.97 s 398.98 s 398.99 s 399.00 s
Process 0
Thread 0:1
Thread 0:2
Thread 0:3
Thread 0:4
Thread 0:5
Thread 0:6
Thread 0:7
Process 1
Thread 1:1
Thread 1:2
Thread 1:3
Thread 1:4
Thread 1:5
Thread 1:6
Thread 1:7
Process 2
Thread 2:1
Thread 2:2
Thread 2:3
Thread 2:4
Thread 2:5
Thread 2:6
Thread 2:7
Process 3
Thread 3:1
Thread 3:2
Thread 3:3
Thread 3:4
Thread 3:5
Thread 3:6
Thread 3:7
Process 4
Thread 4:1
Thread 4:2
Thread 4:3
Thread 4:4
Thread 4:5
Thread 4:6
Thread 4:7
Process 5
Thread 5:1
Thread 5:2
Thread 5:3
Thread 5:4
ftinv_ctl$ftinv_ctl_mod_
Function Summary
All Processes, Accumulated Exclusive Time per Function Group
0.5 s 0.4 s 0.3 s 0.2 s 0.1 s 0.0 s
0.514 s Application
0.1 s DMAPP
Context View
Master Timeline
Property Value
Display Master Timeline
Type Function
Location Process 0
Function ltinv_ctl$ltinv_ctl_mod
Function Group Application
Interval Begin 398.938043 s
Interval End 398.942646 s
Duration 4.6023 ms
Source Loca ltinv_ctl_mod.F90:3
Process Summary
Similar Processes, Accumulated Exclusive Time per Function
0 ms 10 ms 20 ms 30 ms 40 ms 50 ms 60 ms 70 ms
15
14
10
9
8
6
2
ftinv_ctl$ftinv_ctl_mod_
ltinv_ctl$ltinv_ctl_mod_
Call Tree
All Processes
Function | Min Inclusive Time | Max Inclusive Time
ltinv_ctl$ltinv_ctl_mod 0.000 s 26.442 ms
ftinv_ctl$ftinv_ctl_mod 0.000 s 52.853 ms
dmapp_syncid_wait 0.000 s 5.590 µs
dmapp_set_rma_attrs 0.000 s 5.777 µs
dmapp_put 0.000 s 2.995 ms
dmapp_get_rma_attrs 0.000 s 12.411 µs
dmapp_get_nb 0.000 s 84.420 µs
Function Legend
Application
DMAPP
MPI

Schedule for IFS optimisations in CRESTA

When	Activity
4Q2011-1Q2012	Coarray Kernel ✔
1Q2012	IFS CY37R3 port to HECToR ✔ Run T2047 model at scale and analyze performance ✔
2Q2012	Scalability improvements arising from T2047 analyses (“low hanging fruit”) ✔ Overlap Legendre transform computations with associated TRMTOL & TRLTOM transpositions ✔
3Q2012	Semi-Lagrangian optimisation ✔ Overlap TRGTOL & TRLTOG transpositions with associated Fourier transforms ✔
4Q2012	RAPS13 IFS CY38R2 port to HECToR (contains Fast Legendre Transform) T3999 model runs on HECToR Test with IBM F2008 ‘coarray technology preview’ compiler on Power7 at ECMWF
1Q2013	Use coarrays to optimise TRMTOS/TRSTOM transpositions Initial use of GPUs for IFS (targeting LTDIR/LTDIR dgemm’s)
2013-2014	Other IFS scalability optimisations (transpose SL data, physics load balancing, +++) Development & testing of a future solver for IFS (Plan B) Following closely developments in GungHO! project (MetOffice, NERC, STFC) GungHO=Globally Uniform Next Generation Highly Optimised

Thank you for your attention

QUESTIONS?

IFS model coarray developments

Compile with –DCOARRAYS

for compilers that support Fortran2008 coarray syntax

Run with,

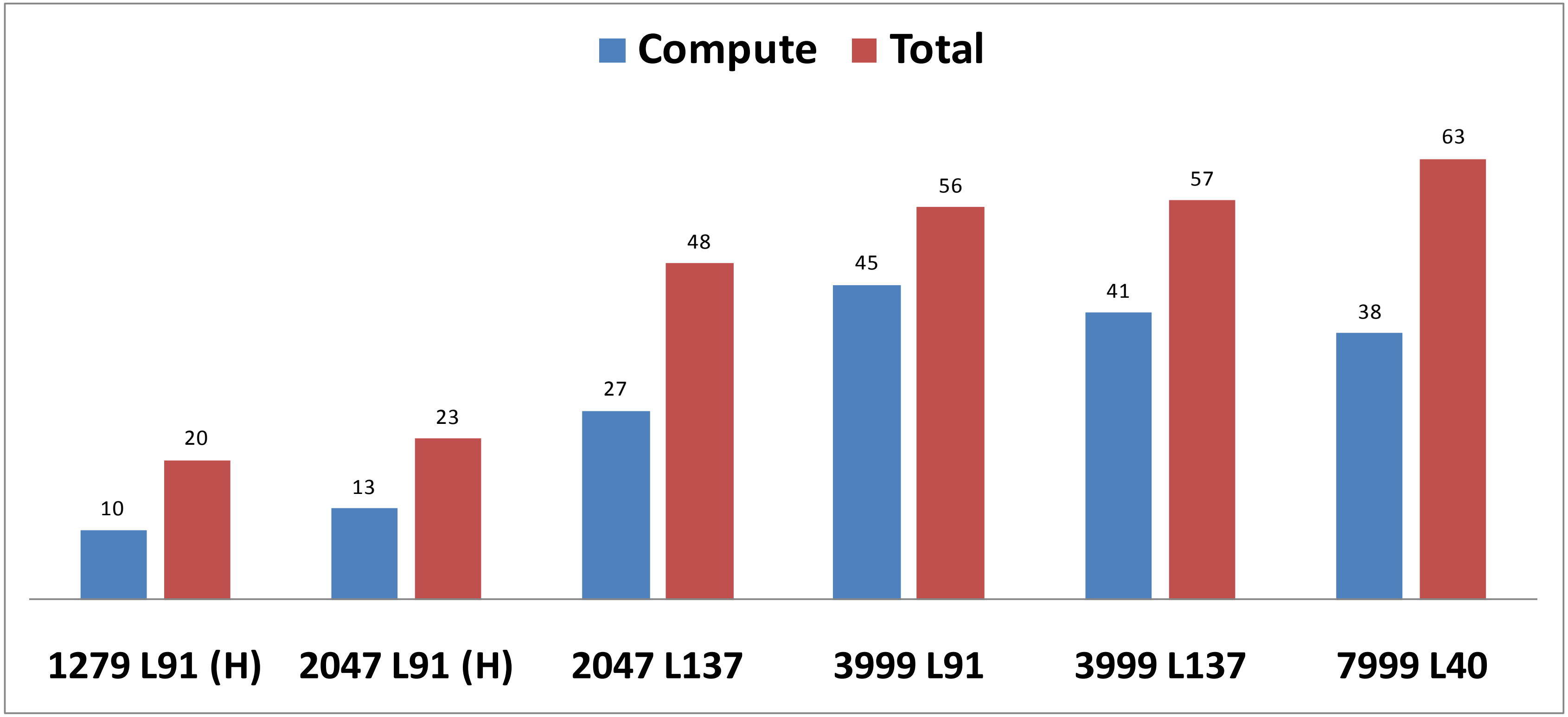
&NAMPAR1
LCOARRAYS=true, to use coarray optimizations

&NAMPAR1
LCOARRAYS=false, to use original MPI implementation

CRESTA

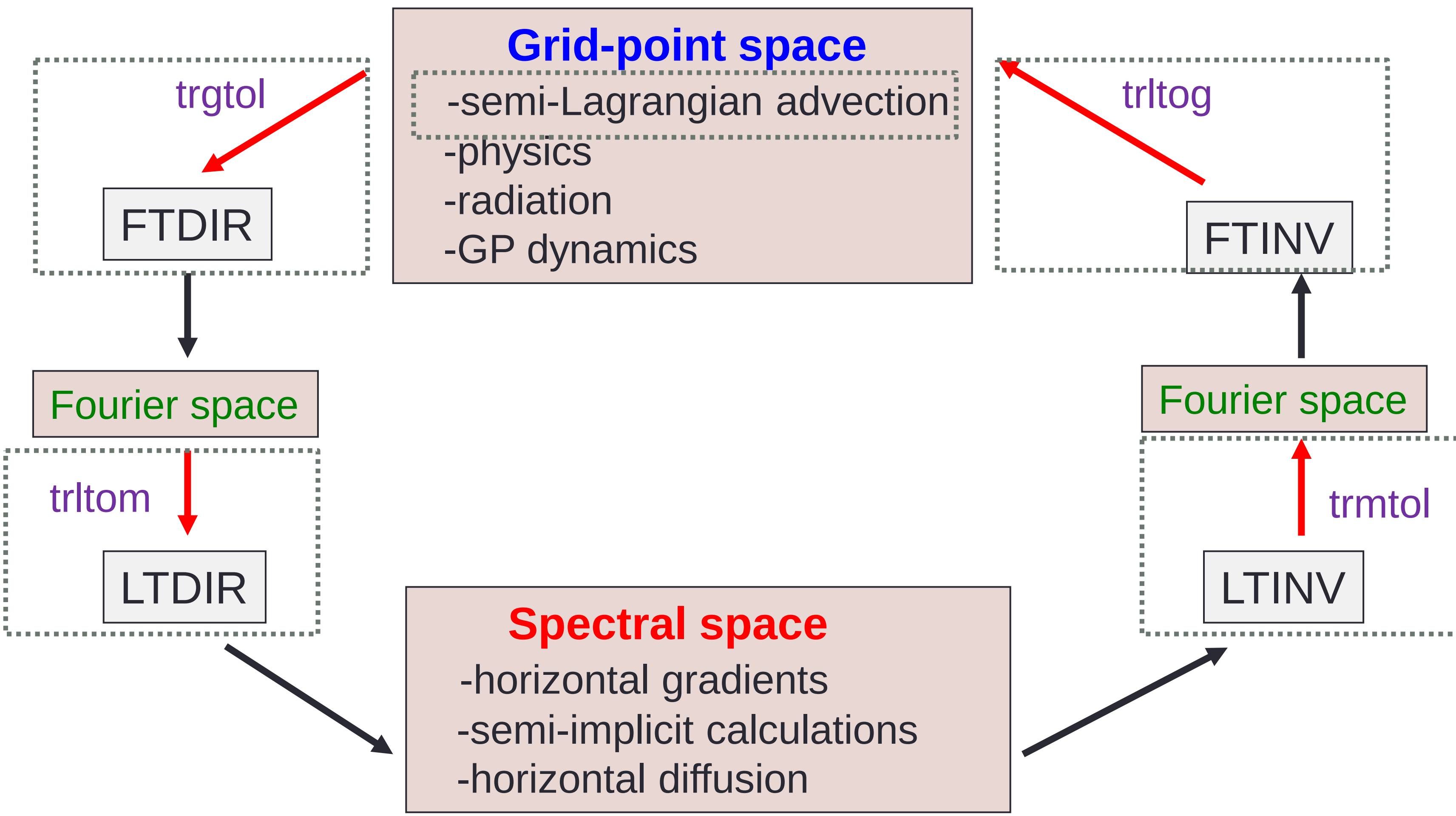
Motivation – T2047 and T3999 costs on IBM Power7 (percentage of wall clock time)

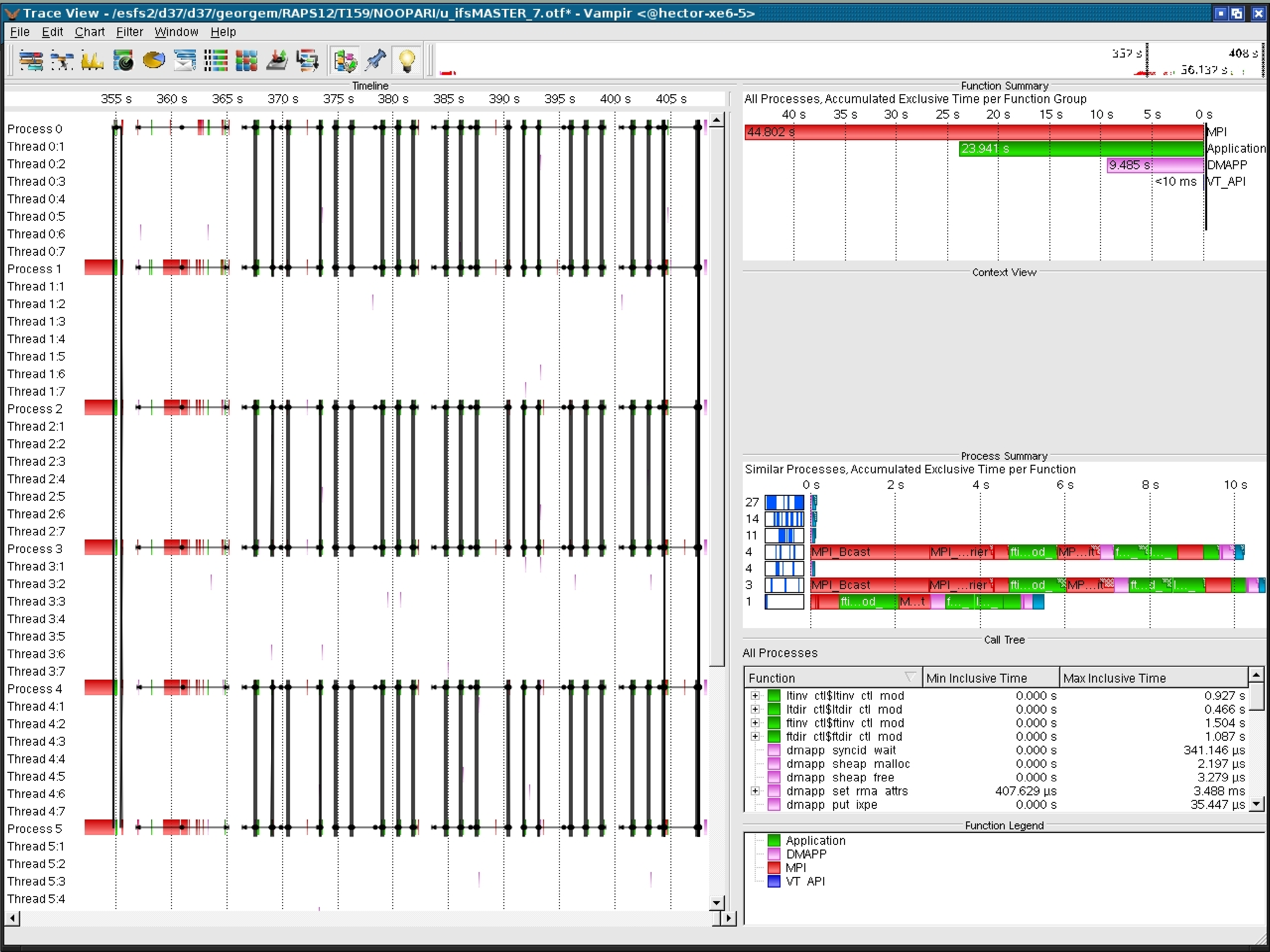
		T2047(%) 2014-15	T3999(%) 2020-21
LTINV_CTL	- INVERSE LEGENDRE TRANSFORM	3.30	8.40
LTINV_CTL	- M TO L TRANSPOSITION	5.37	5.24
LTDIR_CTL	- DIRECT LEGENDRE TRANSFORM	3.56	5.30
LTDIR_CTL	- L TO M TRANSPOSITION	2.84	3.14
FTDIR_CTL	- DIRECT FOURIER TRANSFORM	0.20	1.07
FTDIR_CTL	- G TO L TRANSPOSITION	2.85	2.21
FTINV_CTL	- INVERSE FOURIER TRANSFORM	0.72	3.76
FTINV_CTL	- L TO G TRANSPOSITION	4.47	7.36
	SUM(%)	23.4	36.5
		L137/LT 4224Tx8t 528 Nodes 470 FD/D	L91/FLT 1024Tx16t 256 Nodes 28 FD/D



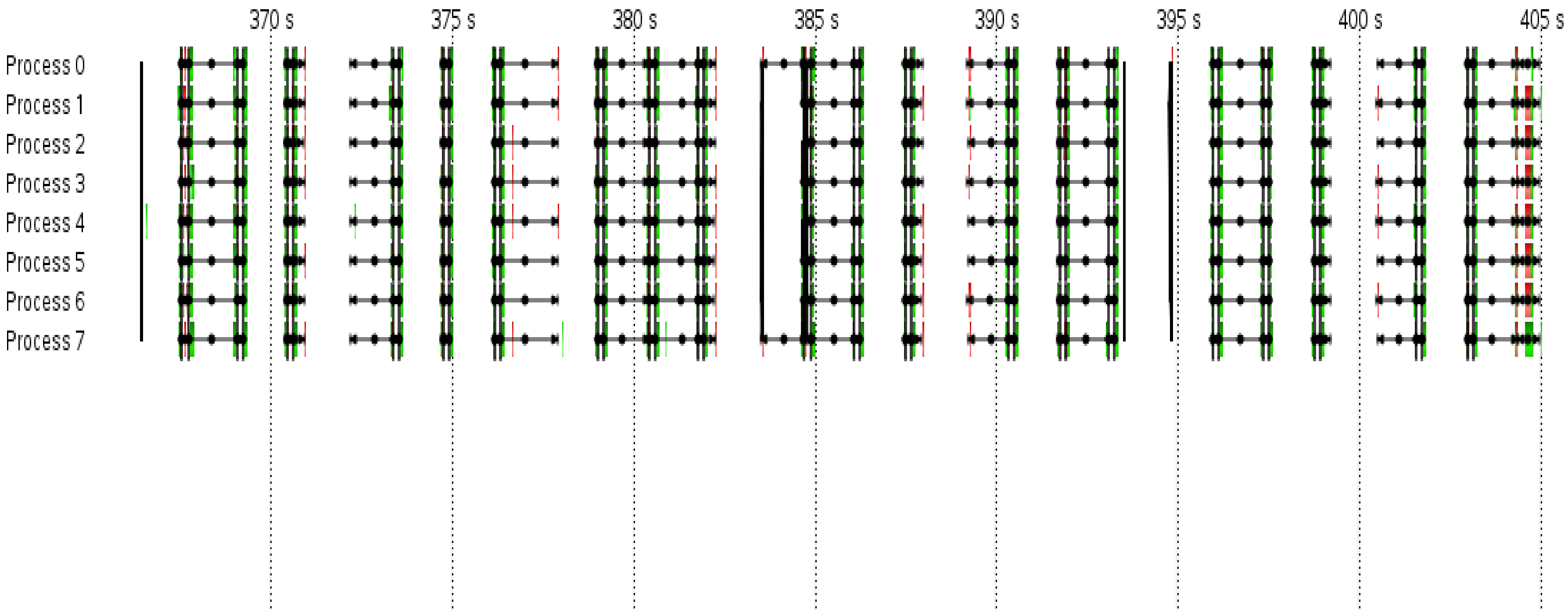
Relative ***computational*** *cost* of the spherical harmonics transforms plus the spectral computations (solving the Helmholtz equation) as a percentage of the overall model cost for various configurations. Red bars indicate the total cost ***including*** the global communications involved. Percentages have been derived considering all gridpoint dynamics and physics computations but without considering IO, synchronization costs (barriers), and any other ancillary costs. All runs are non-hydrostatic unless indicated with (H). All runs further show that the communications cost is less than or equal to the compute cost on the IBM Power7 and have good potential for "hiding" this overhead. However, communication cost is likely to increase with the number of cores.

Planned IFS optimisations for [Tera,Peta,Exa]scale

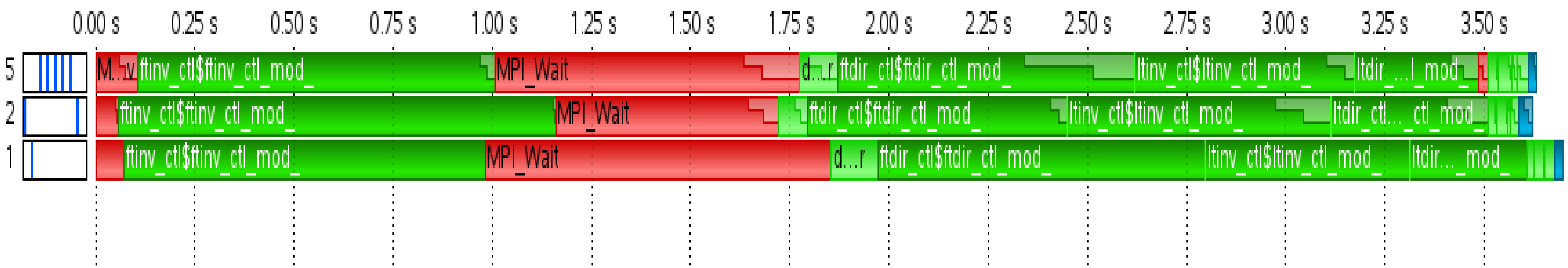


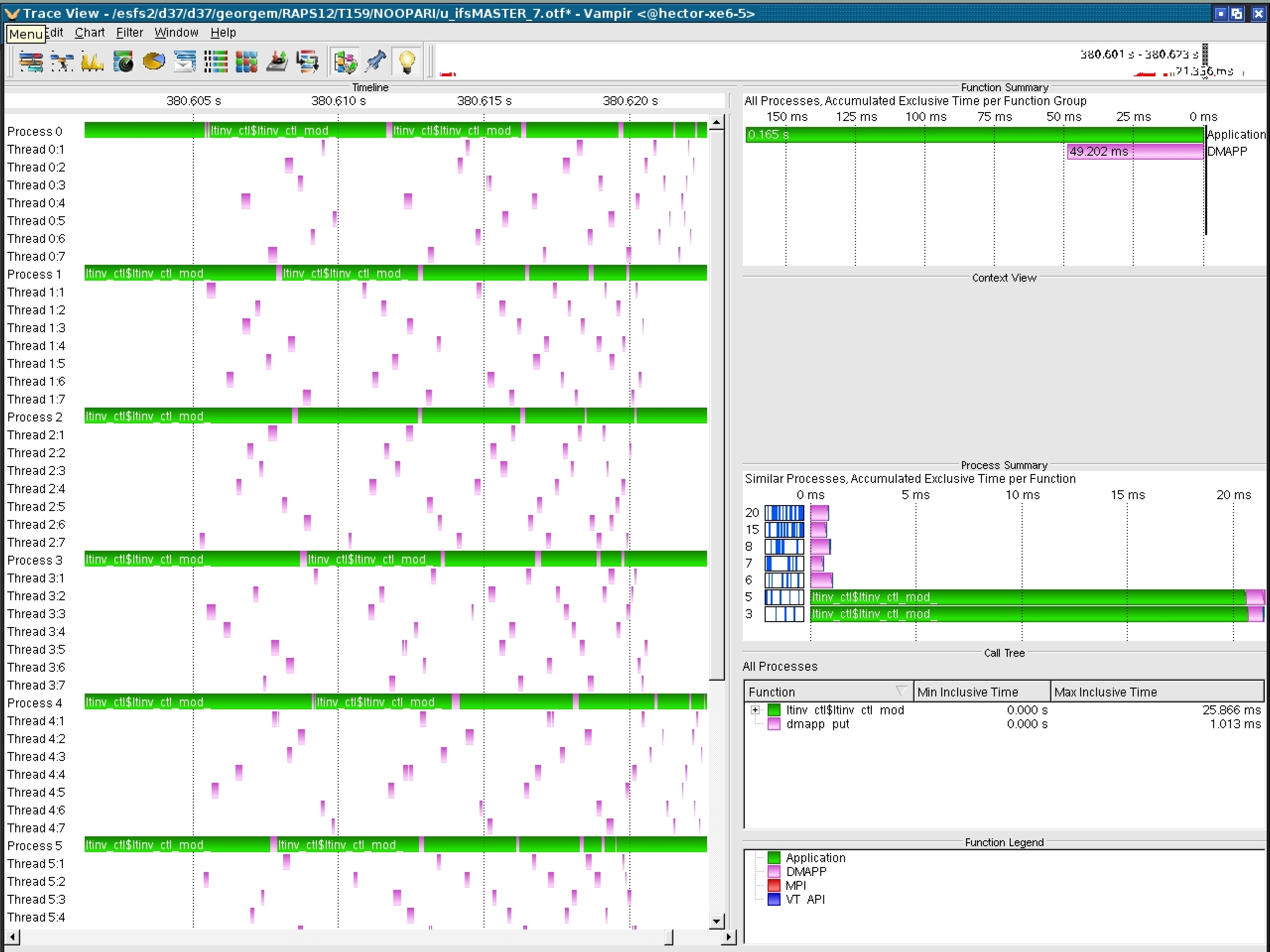


Trace View - /esfs2/d37/d37/georgem/RAPS12/T159/NOOPARI/u_ifsMASTER_7.otf* - Vampir <@hector-xe6-5>
File Edit Chart Filter Window Help
Timeline
355 s 360 s 365 s 370 s 375 s 380 s 385 s 390 s 395 s 400 s 405 s
Process 0
Thread 0:1
Thread 0:2
Thread 0:3
Thread 0:4
Thread 0:5
Thread 0:6
Thread 0:7
Process 1
Thread 1:1
Thread 1:2
Thread 1:3
Thread 1:4
Thread 1:5
Thread 1:6
Thread 1:7
Process 2
Thread 2:1
Thread 2:2
Thread 2:3
Thread 2:4
Thread 2:5
Thread 2:6
Thread 2:7
Process 3
Thread 3:1
Thread 3:2
Thread 3:3
Thread 3:4
Thread 3:5
Thread 3:6
Thread 3:7
Process 4
Thread 4:1
Thread 4:2
Thread 4:3
Thread 4:4
Thread 4:5
Thread 4:6
Thread 4:7
Process 5
Thread 5:1
Thread 5:2
Thread 5:3
Thread 5:4
Function Summary
All Processes, Accumulated Exclusive Time per Function Group
40 s 35 s 30 s 25 s 20 s 15 s 10 s 5 s 0 s
44.802 s MPI
23.941 s Application
9.485 s DMAPP
<10 ms VT_API
Context View
Process Summary
Similar Processes, Accumulated Exclusive Time per Function
0 s 2 s 4 s 6 s 8 s 10 s
27
14
11
4 MPI_Bcast MPI_...rier
4
3 MPI_Bcast MPI_...rier
1
Call Tree
All Processes
Function | Min Inclusive Time | Max Inclusive Time
ltinv ctl$ltinv ctl mod 0.000 s 0.927 s
ltdir ctl$ltdir ctl mod 0.000 s 0.466 s
ftinv ctl$ftinv ctl mod 0.000 s 1.504 s
ftdir ctl$ftdir ctl mod 0.000 s 1.087 s
dmapp syncid wait 0.000 s 341.146 µs
dmapp sheap malloc 0.000 s 2.197 µs
dmapp sheap free 0.000 s 3.279 µs
dmapp set rma attrs 407.629 µs 3.488 ms
dmapp put ixpe 0.000 s 35.447 µs
Function Legend
Application
DMAPP
MPI
VT API

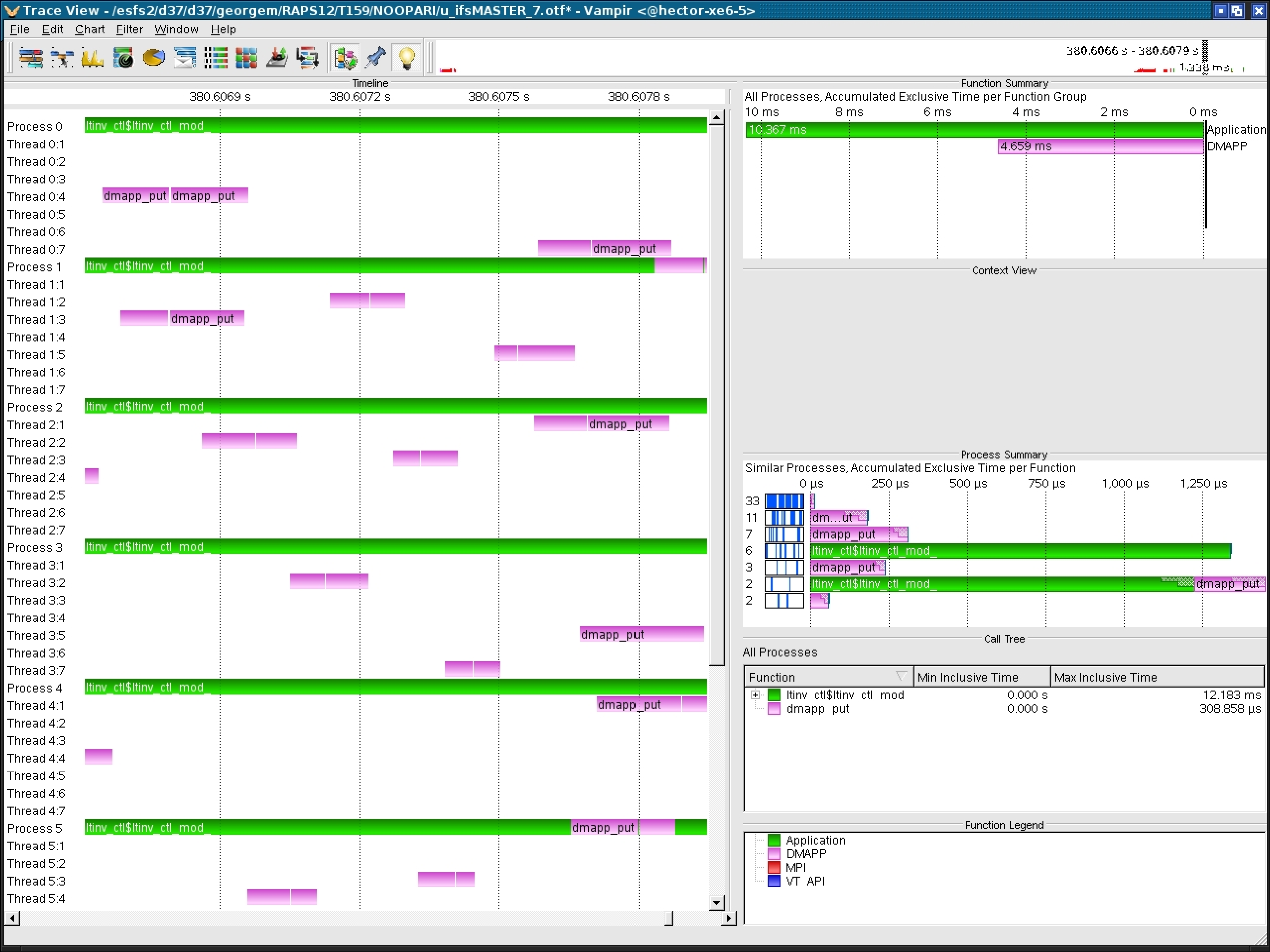


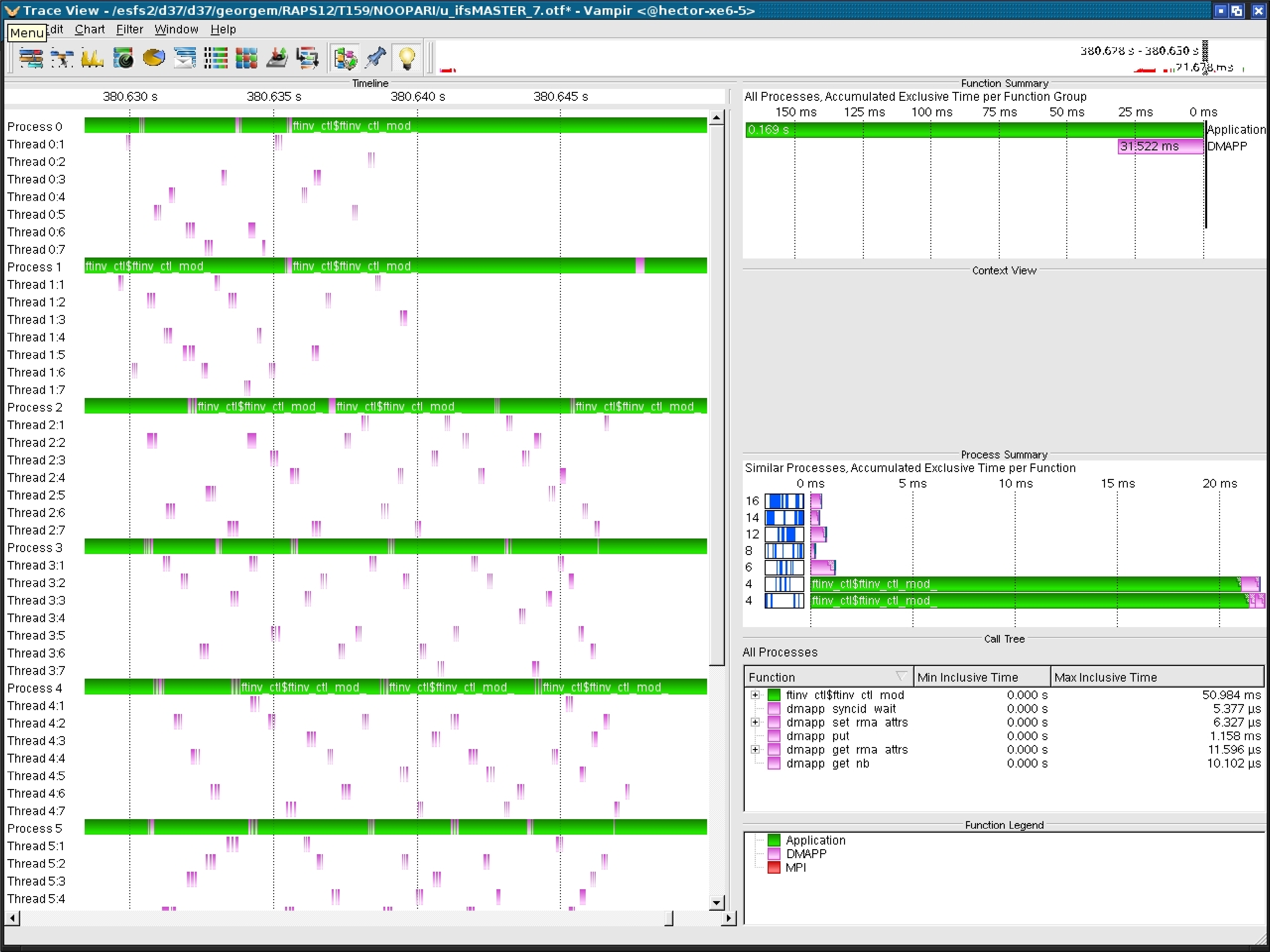
Similar Processes, Accumulated Exclusive Time per Function

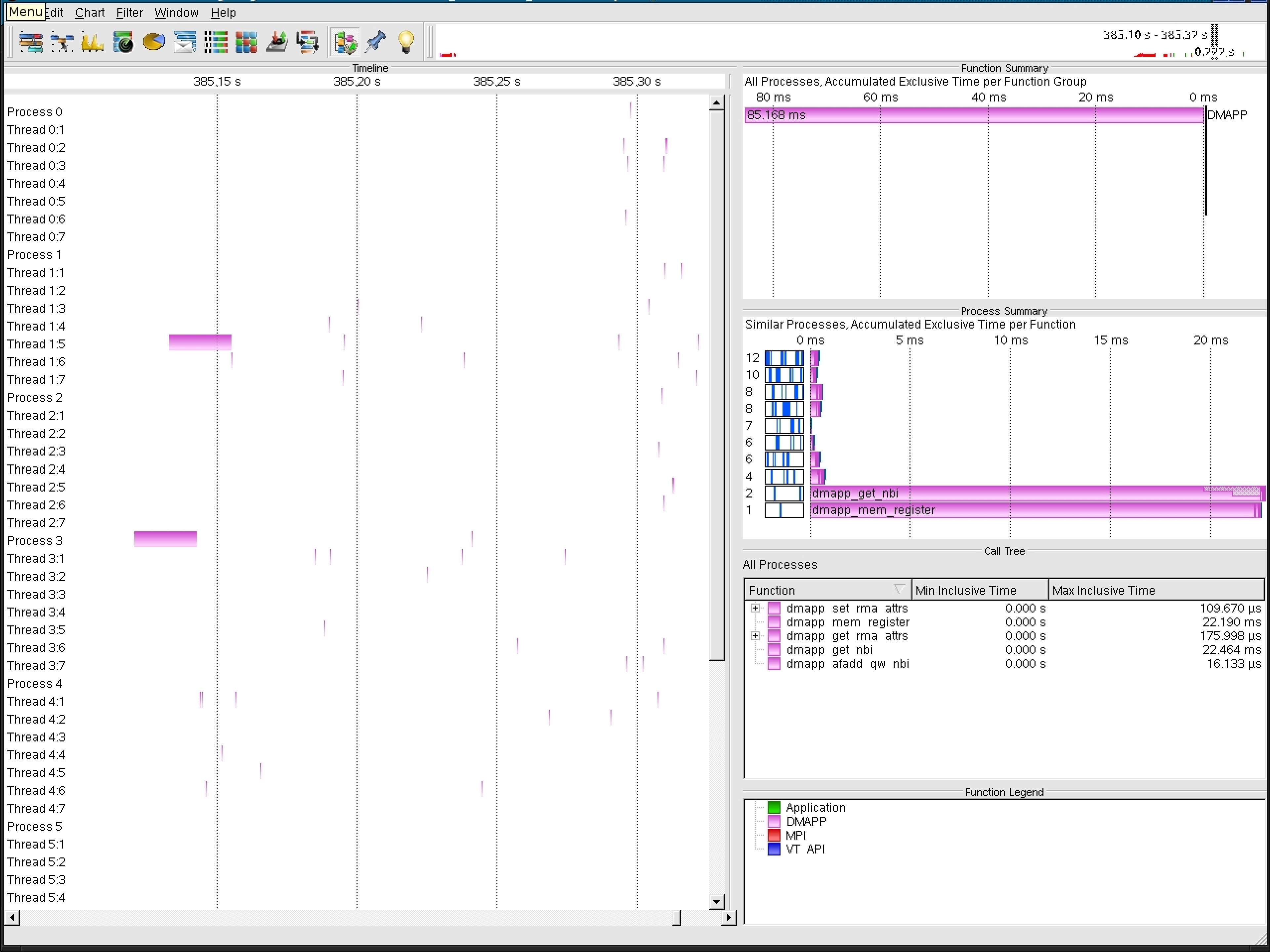




Trace View - /esfs2/d37/d37/georgem/RAPS12/T159/NOOPARI/u_ifsMASTER_7.otf* - Vampir <@hector-xe6-5>
Menu Edit Chart Filter Window Help
380.601 s - 380.673 s
71.336 ms
Timeline
380.605 s
380.610 s
380.615 s
380.620 s
Process 0
Thread 0:1
Thread 0:2
Thread 0:3
Thread 0:4
Thread 0:5
Thread 0:6
Thread 0:7
Process 1
Thread 1:1
Thread 1:2
Thread 1:3
Thread 1:4
Thread 1:5
Thread 1:6
Thread 1:7
Process 2
Thread 2:1
Thread 2:2
Thread 2:3
Thread 2:4
Thread 2:5
Thread 2:6
Thread 2:7
Process 3
Thread 3:1
Thread 3:2
Thread 3:3
Thread 3:4
Thread 3:5
Thread 3:6
Thread 3:7
Process 4
Thread 4:1
Thread 4:2
Thread 4:3
Thread 4:4
Thread 4:5
Thread 4:6
Thread 4:7
Process 5
Thread 5:1
Thread 5:2
Thread 5:3
Thread 5:4
ltinv_ctl$ltinv_ctl_mod_
Function Summary
All Processes, Accumulated Exclusive Time per Function Group
150 ms
125 ms
100 ms
75 ms
50 ms
25 ms
0 ms
0.165 s
Application
49.202 ms
DMAPP
Context View
Process Summary
Similar Processes, Accumulated Exclusive Time per Function
0 ms
5 ms
10 ms
15 ms
20 ms
20
15
8
7
6
5
3
ltinv_ctl$ltinv_ctl_mod_
Call Tree
All Processes
Function
Min Inclusive Time
Max Inclusive Time
ltinv_ctl$ltinv_ctl_mod_
0.000 s
25.866 ms
dmapp_put
0.000 s
1.013 ms
Function Legend
Application
DMAPP
MPI
VT_API


Trace View - /esfs2/d37/d37/georgem/RAPS12/T159/NOOPARI/u_ifsMASTER_7.otf* - Vampir <@hector-xe6-5>
File Edit Chart Filter Window Help
380.6066 s - 380.6079 s
1.338 ms
Timeline
380.6069 s
380.6072 s
380.6075 s
380.6078 s
Process 0
Thread 0:1
Thread 0:2
Thread 0:3
Thread 0:4
Thread 0:5
Thread 0:6
Thread 0:7
Process 1
Thread 1:1
Thread 1:2
Thread 1:3
Thread 1:4
Thread 1:5
Thread 1:6
Thread 1:7
Process 2
Thread 2:1
Thread 2:2
Thread 2:3
Thread 2:4
Thread 2:5
Thread 2:6
Thread 2:7
Process 3
Thread 3:1
Thread 3:2
Thread 3:3
Thread 3:4
Thread 3:5
Thread 3:6
Thread 3:7
Process 4
Thread 4:1
Thread 4:2
Thread 4:3
Thread 4:4
Thread 4:5
Thread 4:6
Thread 4:7
Process 5
Thread 5:1
Thread 5:2
Thread 5:3
Thread 5:4
ltinv_ctl$ltinv_ctl_mod_
dmapp_put
Function Summary
All Processes, Accumulated Exclusive Time per Function Group
10 ms
8 ms
6 ms
4 ms
2 ms
0 ms
10.367 ms
Application
4.659 ms
DMAPP
Context View
Process Summary
Similar Processes, Accumulated Exclusive Time per Function
0 µs
250 µs
500 µs
750 µs
1,000 µs
1,250 µs
33
11
7
6
3
2
2
dm...ut
dmapp_put
ltinv_ctl$ltinv_ctl_mod_
Call Tree
All Processes
Function
Min Inclusive Time
Max Inclusive Time
ltinv_ctl$ltinv_ctl_mod 0.000 s 12.183 ms
dmapp_put 0.000 s 308.858 µs
Function Legend
Application
DMAPP
MPI
VT_API


Trace View - /esfs2/d37/d37/georgem/RAPS12/T159/NOOPARI/u_ifsMASTER_7.otf* - Vampir <@hector-xe6-5>
Menu
Edit
Chart
Filter
Window
Help
380.678 s - 380.630 s
71.678 ms
Timeline
380.630 s
380.635 s
380.640 s
380.645 s
Process 0
Thread 0:1
Thread 0:2
Thread 0:3
Thread 0:4
Thread 0:5
Thread 0:6
Thread 0:7
Process 1
Thread 1:1
Thread 1:2
Thread 1:3
Thread 1:4
Thread 1:5
Thread 1:6
Thread 1:7
Process 2
Thread 2:1
Thread 2:2
Thread 2:3
Thread 2:4
Thread 2:5
Thread 2:6
Thread 2:7
Process 3
Thread 3:1
Thread 3:2
Thread 3:3
Thread 3:4
Thread 3:5
Thread 3:6
Thread 3:7
Process 4
Thread 4:1
Thread 4:2
Thread 4:3
Thread 4:4
Thread 4:5
Thread 4:6
Thread 4:7
Process 5
Thread 5:1
Thread 5:2
Thread 5:3
Thread 5:4
ftinv_ctl$ftinv_ctl_mod
Function Summary
All Processes, Accumulated Exclusive Time per Function Group
150 ms
125 ms
100 ms
75 ms
50 ms
25 ms
0 ms
0.169 s
Application
31.522 ms
DMAPP
Context View
Process Summary
Similar Processes, Accumulated Exclusive Time per Function
0 ms
5 ms
10 ms
15 ms
20 ms
16
14
12
8
6
4
4
Call Tree
All Processes
Function
Min Inclusive Time
Max Inclusive Time
ftinv_ctl$ftinv_ctl_mod 0.000 s 50.984 ms
dmapp_syncid_wait 0.000 s 5.377 µs
dmapp_set_rma_attrs 0.000 s 6.327 µs
dmapp_put 0.000 s 1.158 ms
dmapp_get_rma_attrs 0.000 s 11.596 µs
dmapp_get_nb 0.000 s 10.102 µs
Function Legend
Application
DMAPP
MPI


Menu Edit Chart Filter Window Help
385.10 s - 385.37 s
0.272 s
Timeline
385.15 s
385.20 s
385.25 s
385.30 s
Process 0
Thread 0:1
Thread 0:2
Thread 0:3
Thread 0:4
Thread 0:5
Thread 0:6
Thread 0:7
Process 1
Thread 1:1
Thread 1:2
Thread 1:3
Thread 1:4
Thread 1:5
Thread 1:6
Thread 1:7
Process 2
Thread 2:1
Thread 2:2
Thread 2:3
Thread 2:4
Thread 2:5
Thread 2:6
Thread 2:7
Process 3
Thread 3:1
Thread 3:2
Thread 3:3
Thread 3:4
Thread 3:5
Thread 3:6
Thread 3:7
Process 4
Thread 4:1
Thread 4:2
Thread 4:3
Thread 4:4
Thread 4:5
Thread 4:6
Thread 4:7
Process 5
Thread 5:1
Thread 5:2
Thread 5:3
Thread 5:4
Function Summary
All Processes, Accumulated Exclusive Time per Function Group
80 ms
60 ms
40 ms
20 ms
0 ms
85.168 ms
DMAPP
Process Summary
Similar Processes, Accumulated Exclusive Time per Function
0 ms
5 ms
10 ms
15 ms
20 ms
12
10
8
8
7
6
6
4
2
1
dmapp_get_nbi
dmapp_mem_register
Call Tree
All Processes
Function
Min Inclusive Time
Max Inclusive Time
dmapp_set_rma_attrs 0.000 s 109.670 µs
dmapp_mem_register 0.000 s 22.190 ms
dmapp_get_rma_attrs 0.000 s 175.998 µs
dmapp_get_nbi 0.000 s 22.464 ms
dmapp_afadd_qw_nbi 0.000 s 16.133 µs
Function Legend
Application
DMAPP
MPI
VT_API

Other Fortran 2008 compilers

- License finally agreed with IBM
  - ECMWF will install xlf v14 compiler on Power7
  - Only took 1 year from first inquiry (pre-CRESTA)
  - Subject to non-disclosure
  - Am sure we will be granted permission to present and publish results if they are good
  - Plan is first to test IFS RAPS12 with this compiler
- Promoting need for Fortran 2008 to vendors is important
- Intel ?
- Fujitsu ?
- gfortran ?
- PGI ?

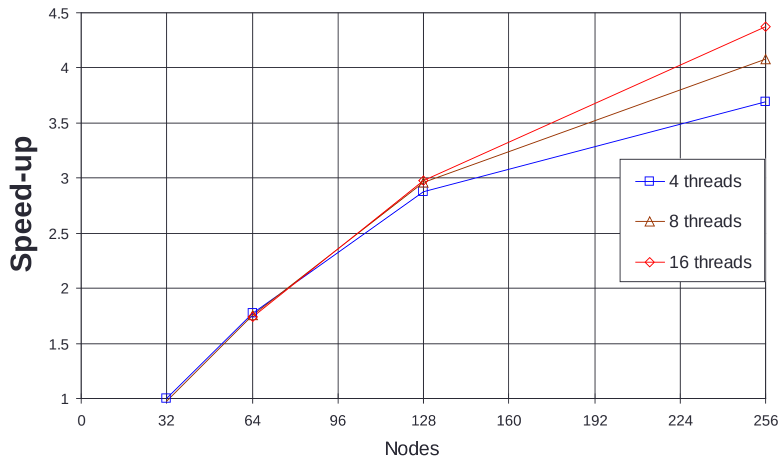
Using HECToR

- Moved IFS from cce=8.0.3 to cce=8.0.6
  - To pick up fix to random hangs at start of job
    - Job would run to cp time limit without executing a single application statement
    - Refunded lost KAu's
  - 8.0.6 also fixed a couple of random coarray runtime failures
  - Thanks to CRAY for providing a good compiler release
- Multiple aprun's in high core count jobs (10K to 64K cores)
  - To improve overall system resource utilization
  - Small, medium and large batched jobs
  - Some waste due to unused cores in each job
  - Promise of refund (more KAus) at some time in future

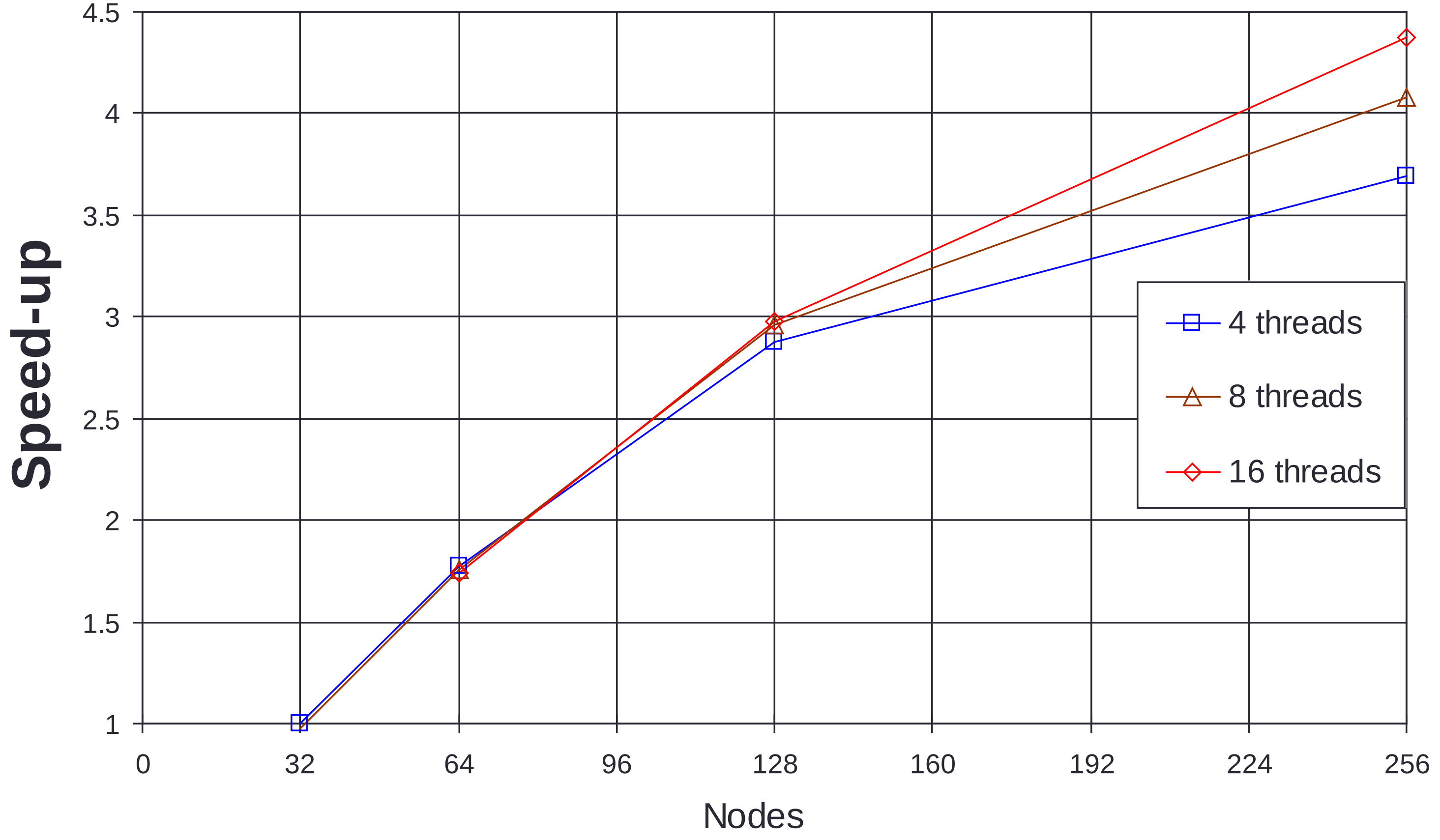
Hybrid runtime support - IFS

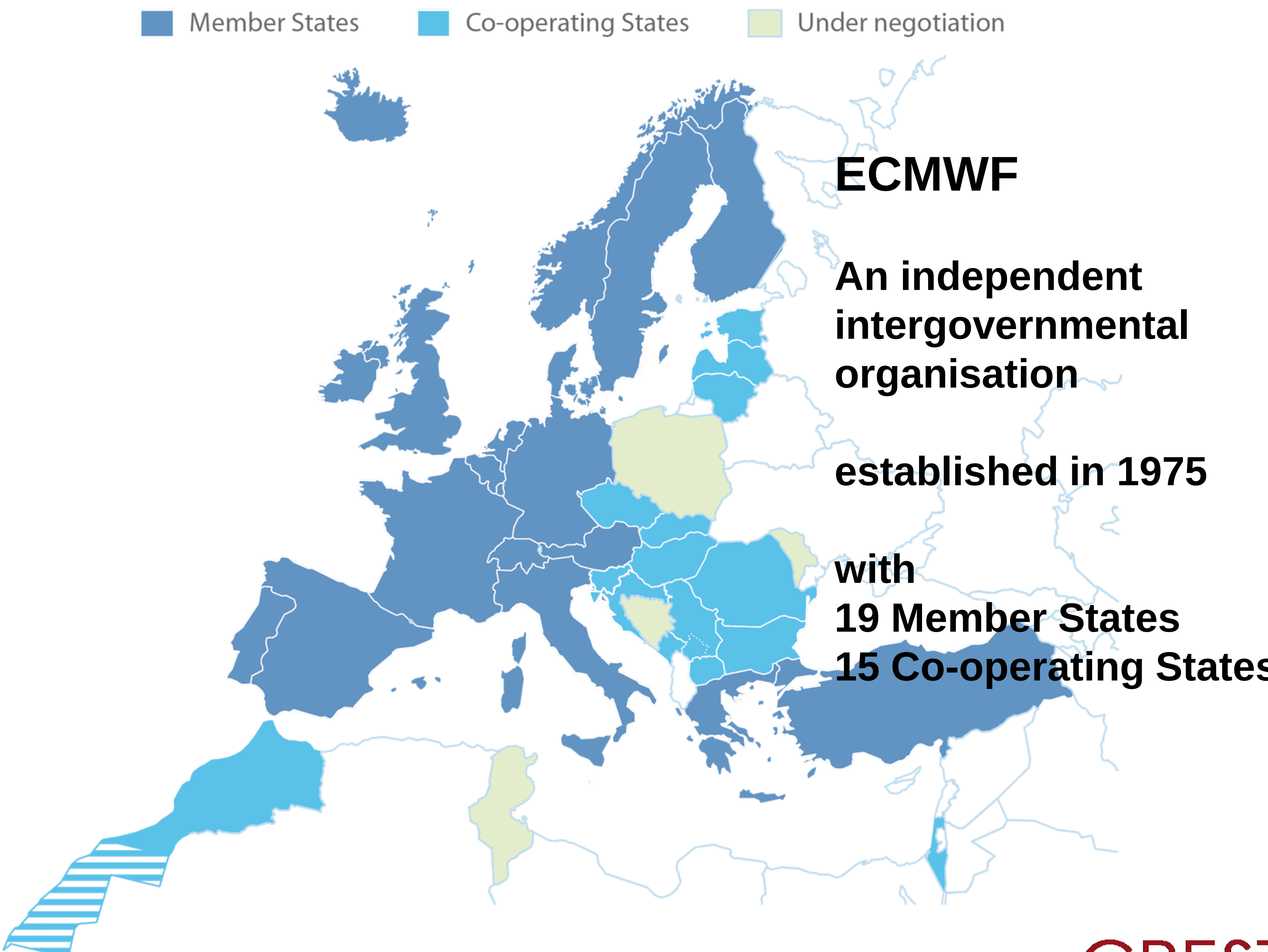
- Initial IFS MPI implementation 1994-1996
- Hybrid MPI/OpenMP implementation ~1999
  - OpenMP implementation at highest level
  - Single parallel regions for each of physics, radiation scheme, dynamics, Legendre transforms, Fourier transforms and Fourier space computations
  - Schedule dynamic used in most parallel regions
- Hybrid implementation benefits
  - About 20 percent performance improvement at scale
  - Huge memory savings , memory use reduces linearly with number of OpenMP threads
- Next evolutionary step: use of Fortran 2008 coarrays to
  - Overlap computation with communication in transpositions
    - Fourier space <-> Spectral space comms, overlapped with Legendre transforms
    - grid point space <-> Fourier space comms, overlapped with FFTs and Fourier space computations
  - Reduce total halo communication in semi-Lagrangian scheme
  - Dominant coarray communications in OpenMP parallel regions

OpenMP for IFS T1279L91 model on IBM Power6 (~2009)



OpenMP for IFS T1279L91 model on IBM Power6 (~2009)





ECMWF

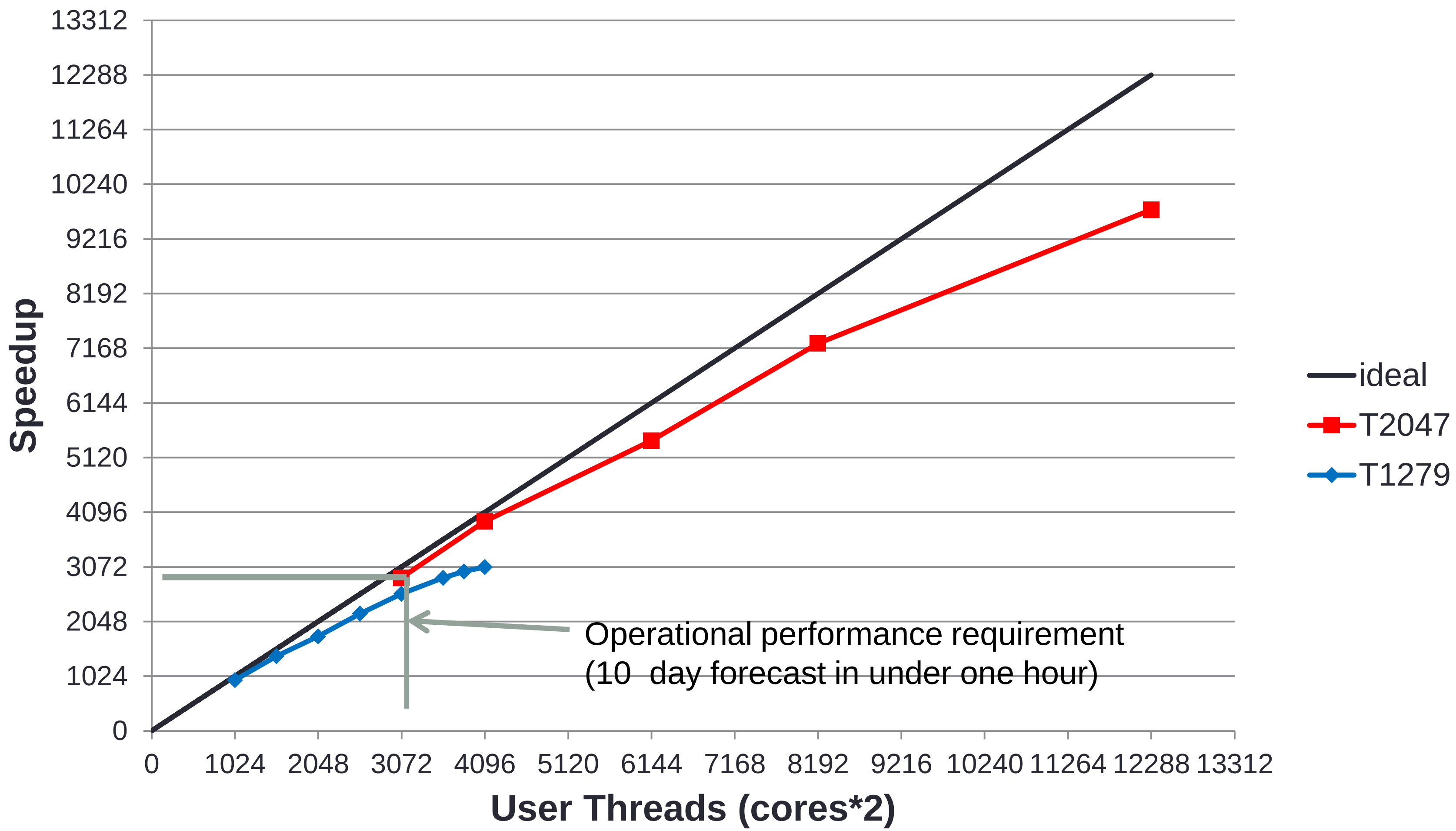
An independent intergovernmental organisation

established in 1975

with
19 Member States
15 Co-operating States

CREST

IFS model speedup on IBM Power6 (~2010)



ECMWF

Medium-range forecasts to 15 days ahead, including early warnings of severe weather.

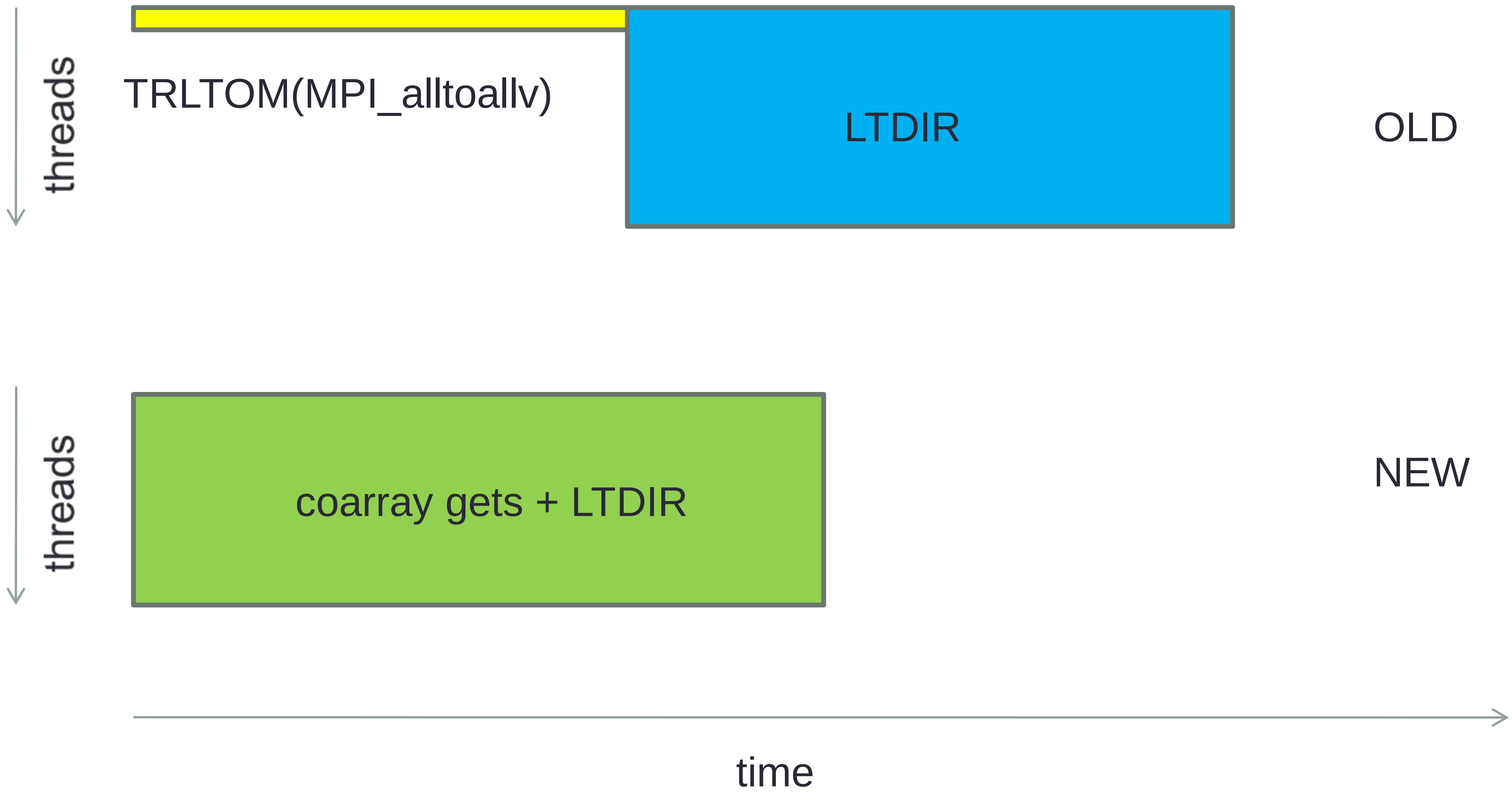
National Meteorological Services

Early warnings interpreted and tailored to specific user needs (including impacts); more detailed short-range (1–2 day) warnings.

National civil protection agencies

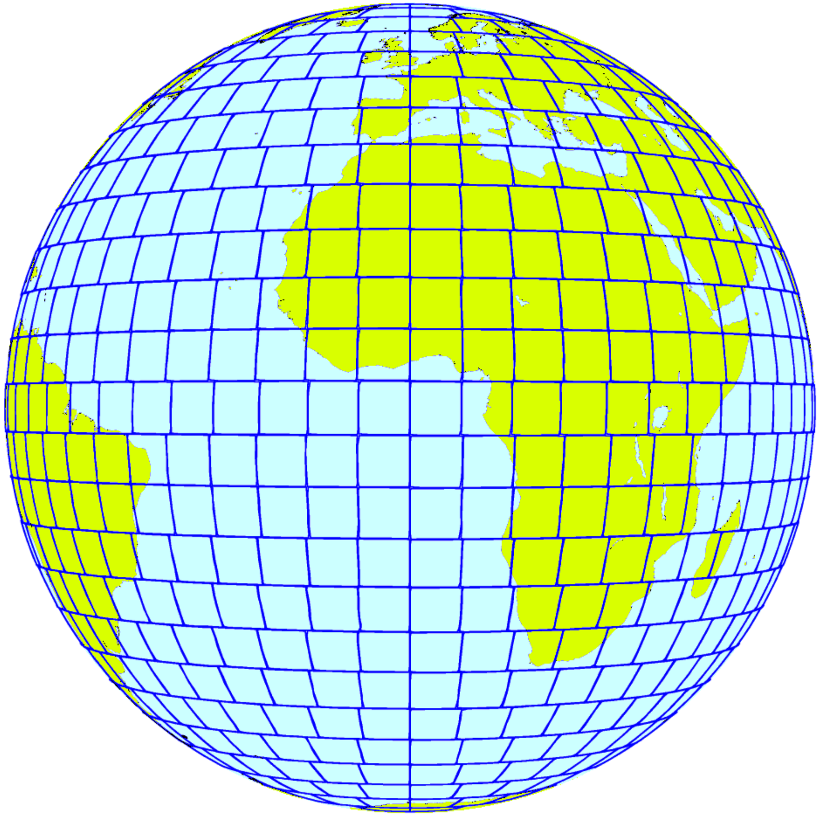
Public

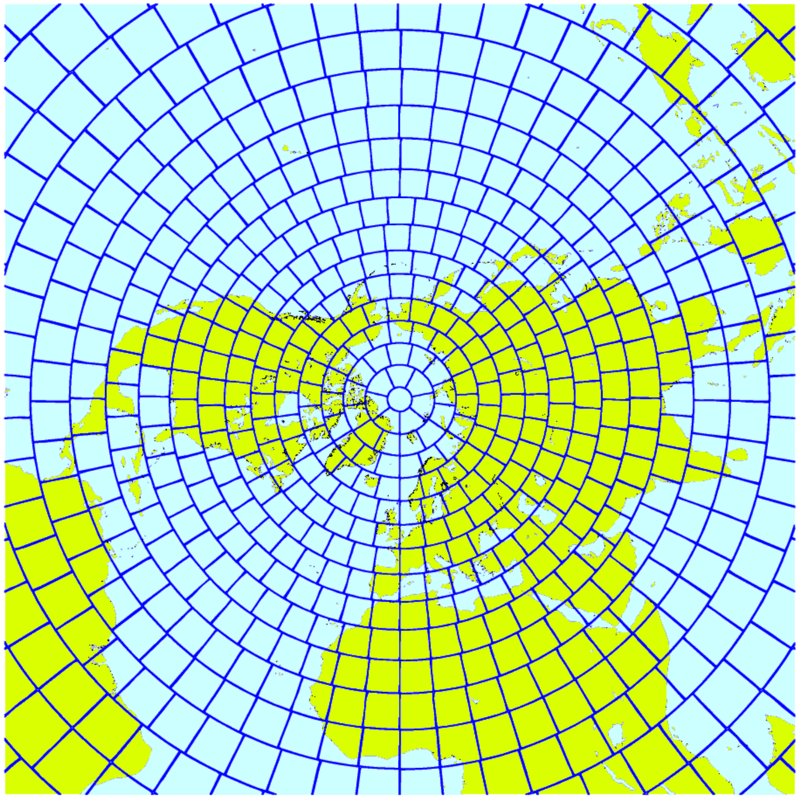
Overlap Legendre transforms with associated transpositions/2



IFS grid point space: “EQ_REGIONS” partitioning for 1024 MPI tasks

Each MPI task has an equal number of grid points





LTINV recoding

COMPUTE
COMMUNICATION

ORIGINAL code

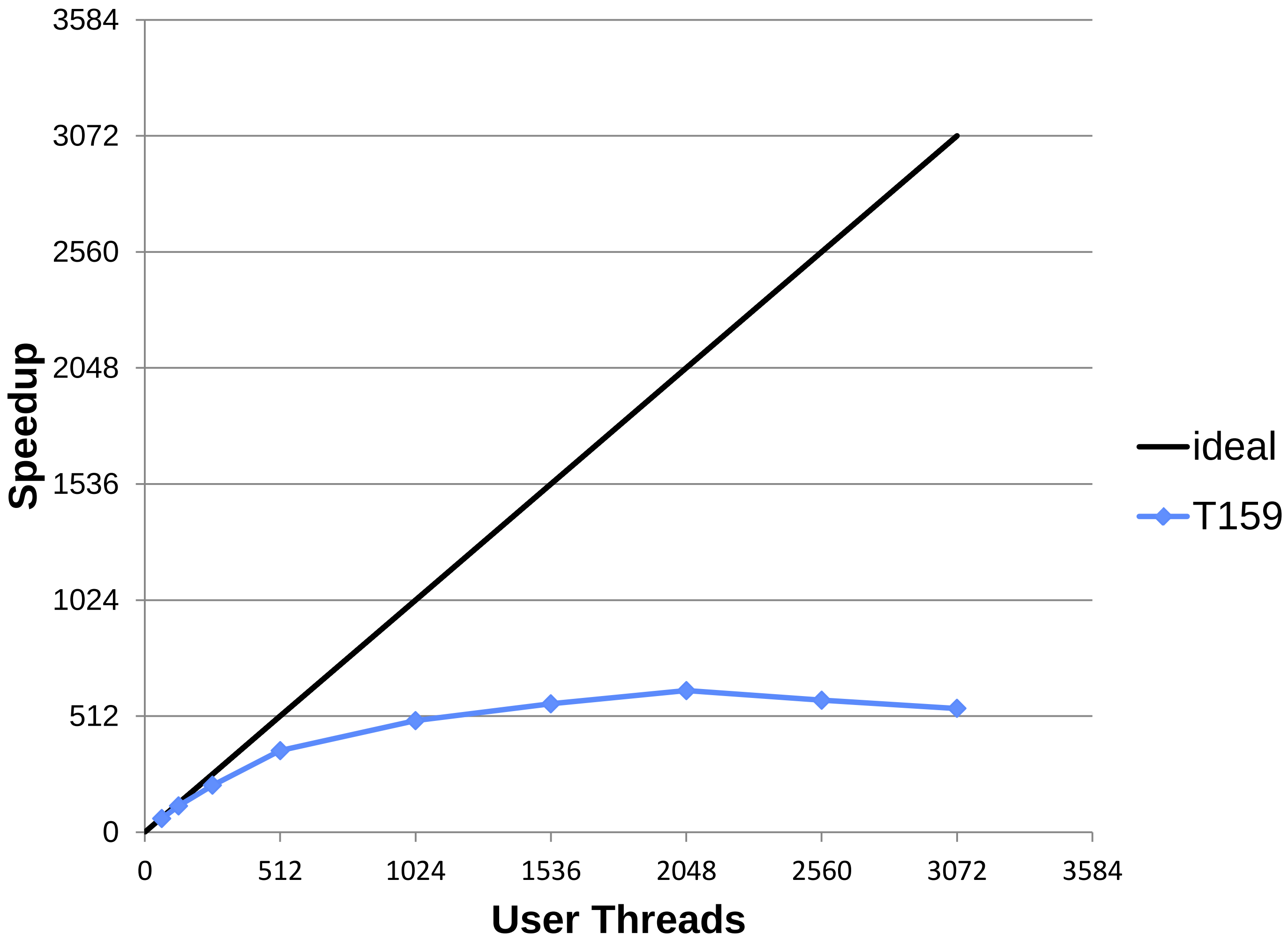
```
!$OMP PARALLEL DO SCHEDULE(DYNAMIC,1) PRIVATE(JM,IM)
DO JM=1,D%NUMP
  IM = D%MYMS(JM)
  CALL LTINV(IM,JM,KF_OUT_LT,KF_UV,KF_SCALARS,KF_SCDERS,ILEI2,IDIM1,&
   & PSPVOR,PSPDIV,PSPSCALAR ,&
   & PSPSC3A,PSPSC3B,PSPSC2 , &
   & KFLDPTRUV,KFLDPTRSC,FSPGL_PROC)
ENDDO
!$OMP END PARALLEL DO
DO J=1,NPRTRW
  ILENS(J) = D%NLTSFTB(J)*IFIELD
  IOFFS(J) = D%NSTAGT0B(J)*IFIELD
  ILENR(J) = D%NLTSGTB(J)*IFIELD
  IOFFR(J) = D%NSTAGT0B(D%MSTABF(J))*IFIELD
ENDDO
CALL MPL_ALLTOALLV(PSENDBUF=FOUBUF_IN,KSENDCOUNTS=ILENS,&
 & PRECVBUF=FOUBUF,KRECVCOUNTS=ILENR,&
 & KSENDDISPL=IOFFS,KRECVDISPL=IOFFR,&
 & KCOMM=MPL_ALL_MS_COMM,CDSTRING='TRMTOL:')
```

NEW code

```
!$OMP PARALLEL DO SCHEDULE(DYNAMIC,1) PRIVATE(JM,IM,JW,IPE,ILEN,ILENS,IOFFS,IOFFR)
DO JM=1,D%NUMP
  IM = D%MYMS(JM)
  CALL LTINV(IM,JM,KF_OUT_LT,KF_UV,KF_SCALARS,KF_SCDERS,ILEI2,IDIM1,&
    & PSPVOR,PSPDIV,PSPSCALAR ,&
    & PSPSC3A,PSPSC3B,PSPSC2 , &
    & KFLDPTRUV,KFLDPTRSC,FSPGL_PROC)
  DO JW=1,NPRTRW
    CALL SET2PE(IPE,0,0,JW,MYSETV)
    ILEN = D%NLEN_M(JW,1,JM)*IFIELD
    IF( ILEN > 0 )THEN
      IOFFS = (D%NSTAGT0B(JW)+D%NOFF_M(JW,1,JM))*IFIELD
      IOFFR = (D%NSTAGT0BW(JW,MYSETW)+D%NOFF_M(JW,1,JM))*IFIELD
      FOUBUF_C(IOFFR+1:IOFFR+ILEN)[IPE]=FOUBUF_IN(IOFFS+1:IOFFS+ILEN)
    ENDIF
    ILENS = D%NLEN_M(JW,2,JM)*IFIELD
    IF( ILENS > 0 )THEN
      IOFFS = (D%NSTAGT0B(JW)+D%NOFF_M(JW,2,JM))*IFIELD
      IOFFR = (D%NSTAGT0BW(JW,MYSETW)+D%NOFF_M(JW,2,JM))*IFIELD
      FOUBUF_C(IOFFR+1:IOFFR+ILENS)[IPE]=FOUBUF_IN(IOFFS+1:IOFFS+ILENS)
    ENDIF
  ENDDO
ENDDO
!$OMP END PARALLEL DO
SYNC IMAGES(D%NMYSETW)
FOUBUF(1:IBLEN)=FOUBUF_C(1:IBLEN)[MYPROC]
```

CREST

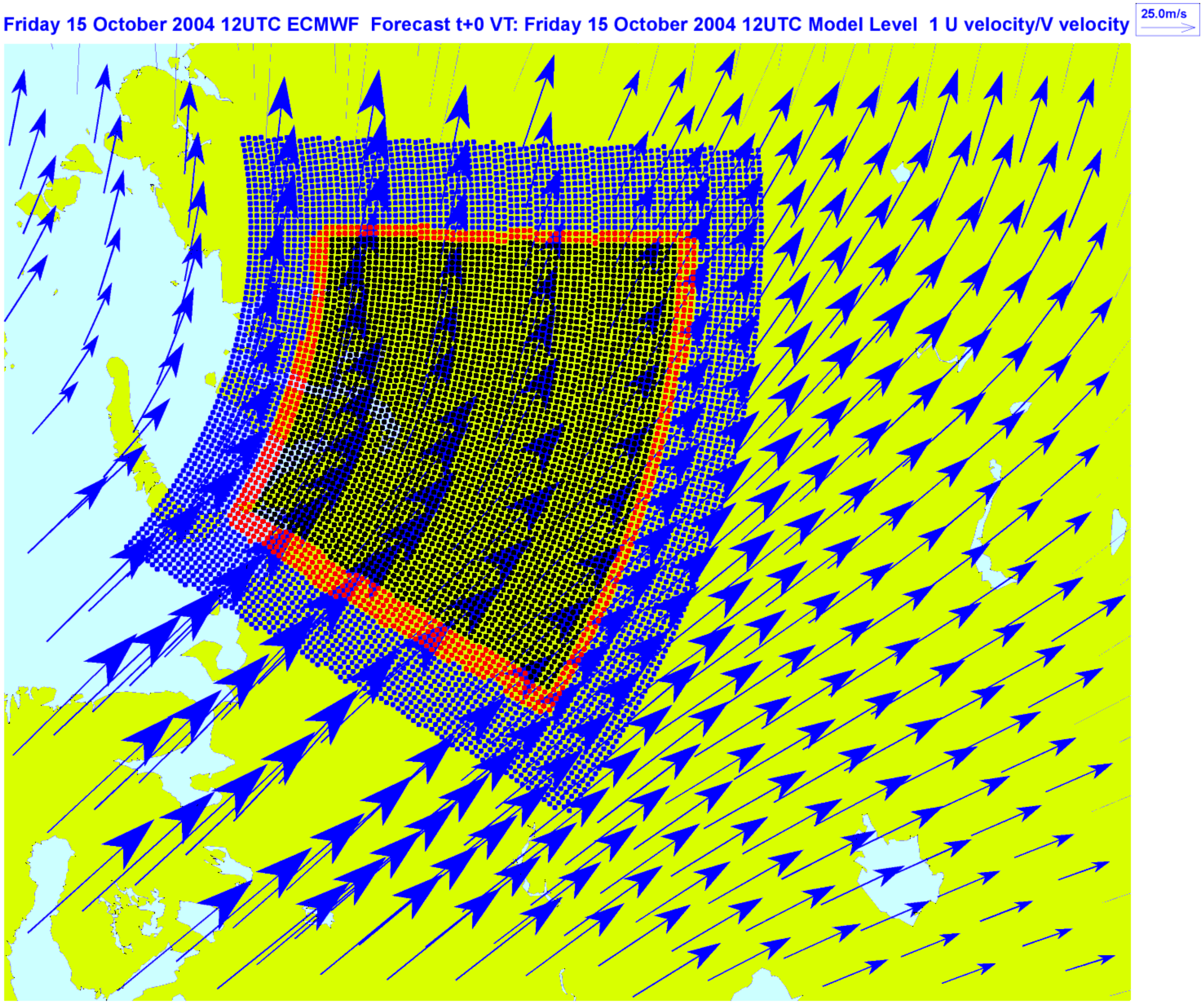
T159 model scaling: small model with ‘large’ number of user threads (4 threads per task)



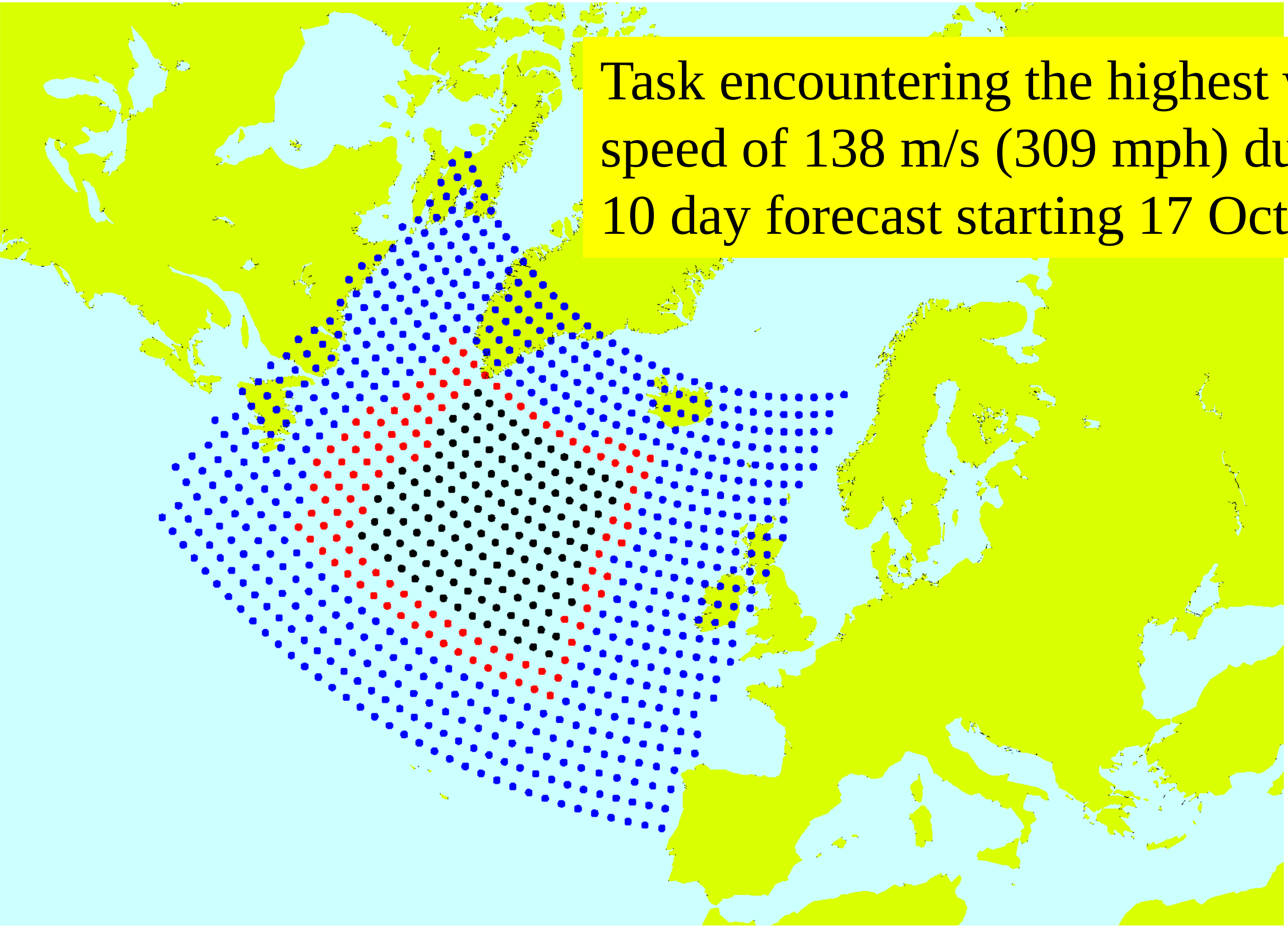
IFS Semi-Lagrangian Comms

- **SL comms scaling limited by**
  - **constant width halo for u,v,w (400 m/s x time step)**
  - **Halo volume communicated, which is a function of wind speed and direction in locality of each task**
- **'Halo-lite' approach tested (2010)**
  - **Only get (using MPI) grid columns from neighbouring tasks that your task needs, i.e. only the red points**
  - **Requires more MPI communication steps (e.g. mid-point, departure point)**
  - **No faster than original approach due to overheads of above**
- **CRESTA optimisation using F2008 coarrays (2012)**
  - **Only get grid columns from neighbouring tasks that your task needs, i.e. only the red points**
  - **Do the above in the context of an OpenMP parallel region; overlapping interpolations for determining the departure point & mid-point and interpolations at these points**

wind plot

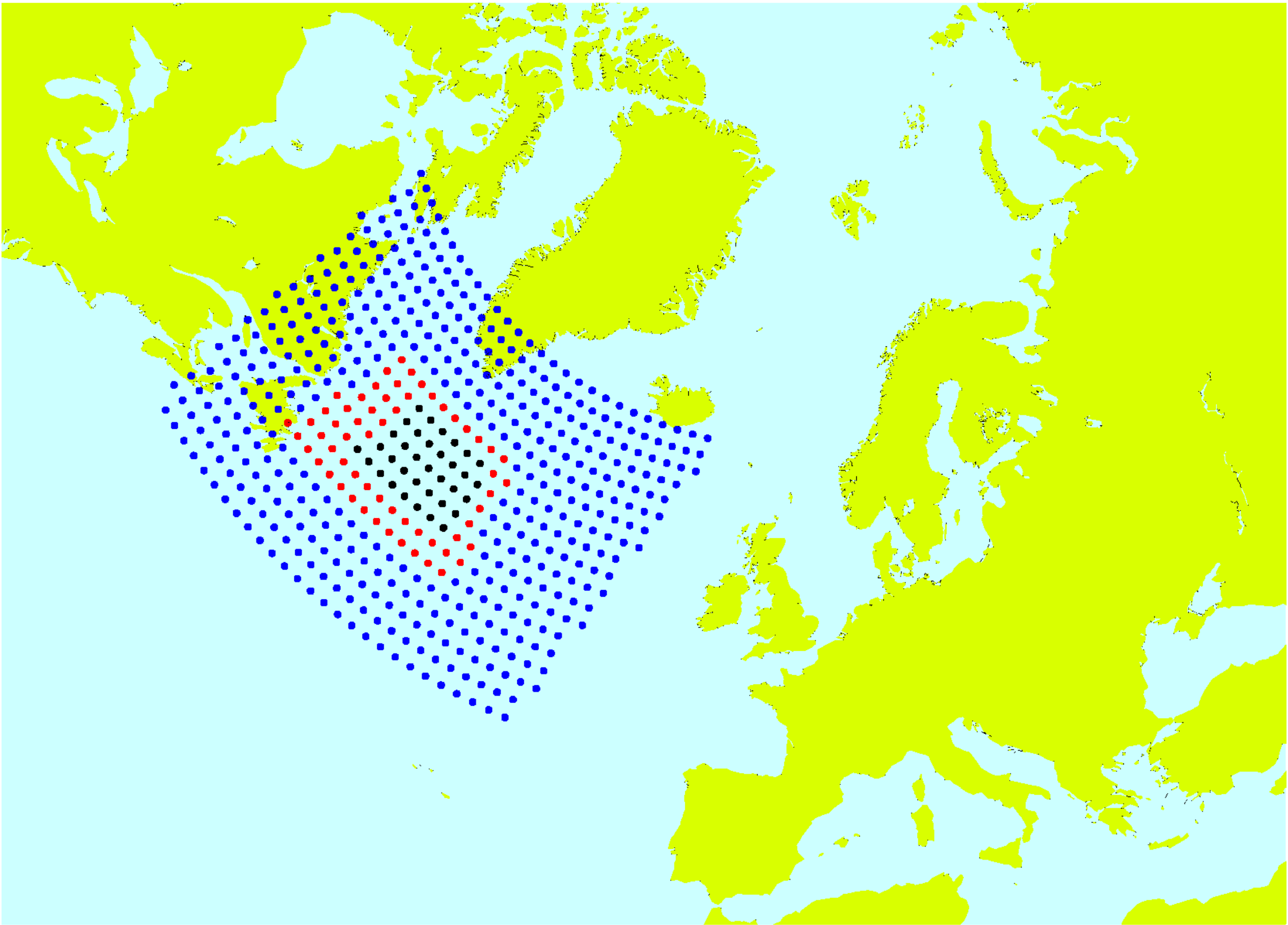

Friday 15 October 2004 12UTC ECMWF Forecast t+0 VT: Friday 15 October 2004 12UTC Model Level 1 U velocity/V velocity
25.0m/s

T159 model task 37 of 256 tasks

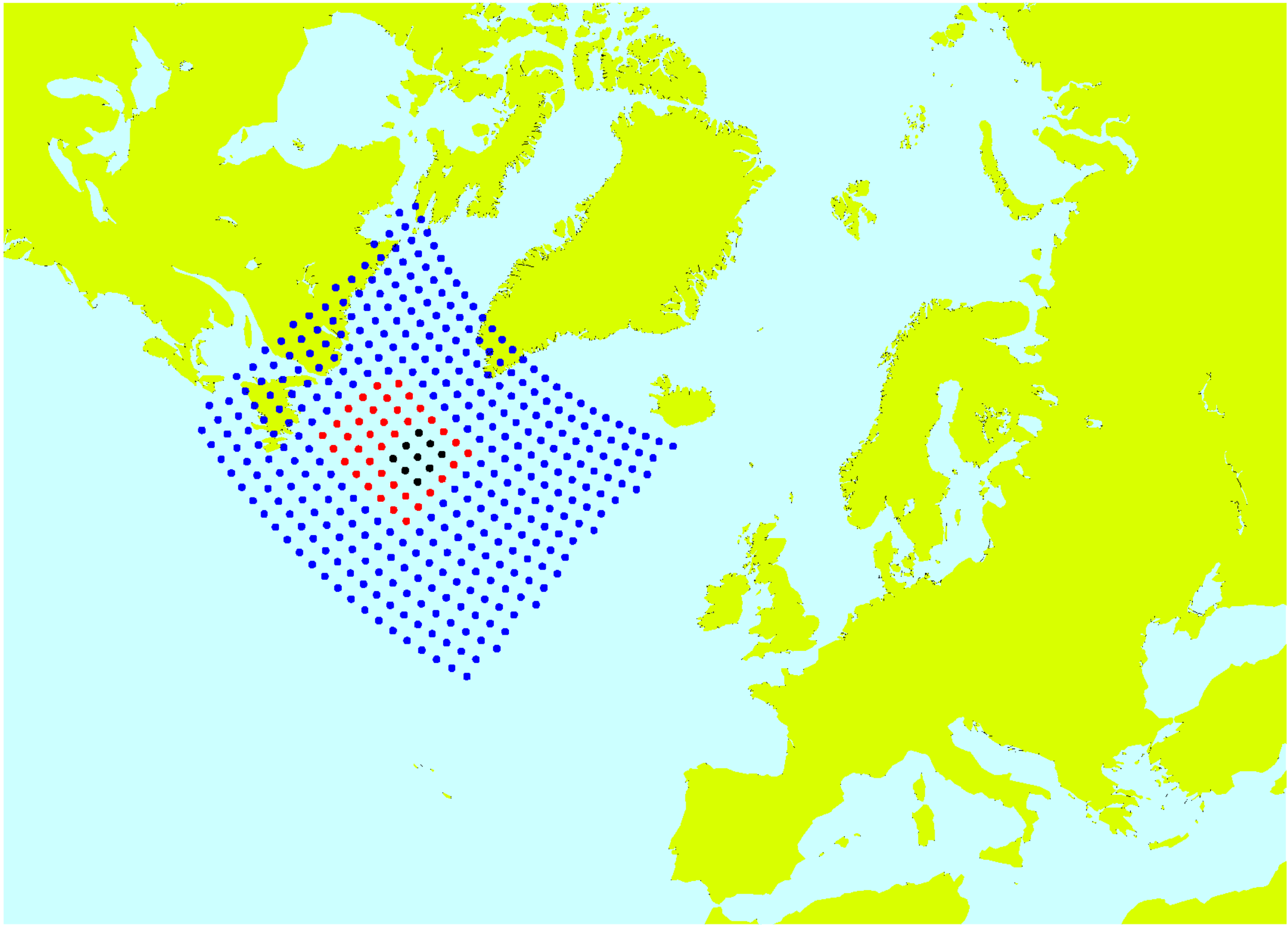


Task encountering the highest wind speed of 138 m/s (309 mph) during a 10 day forecast starting 17 Oct 2010

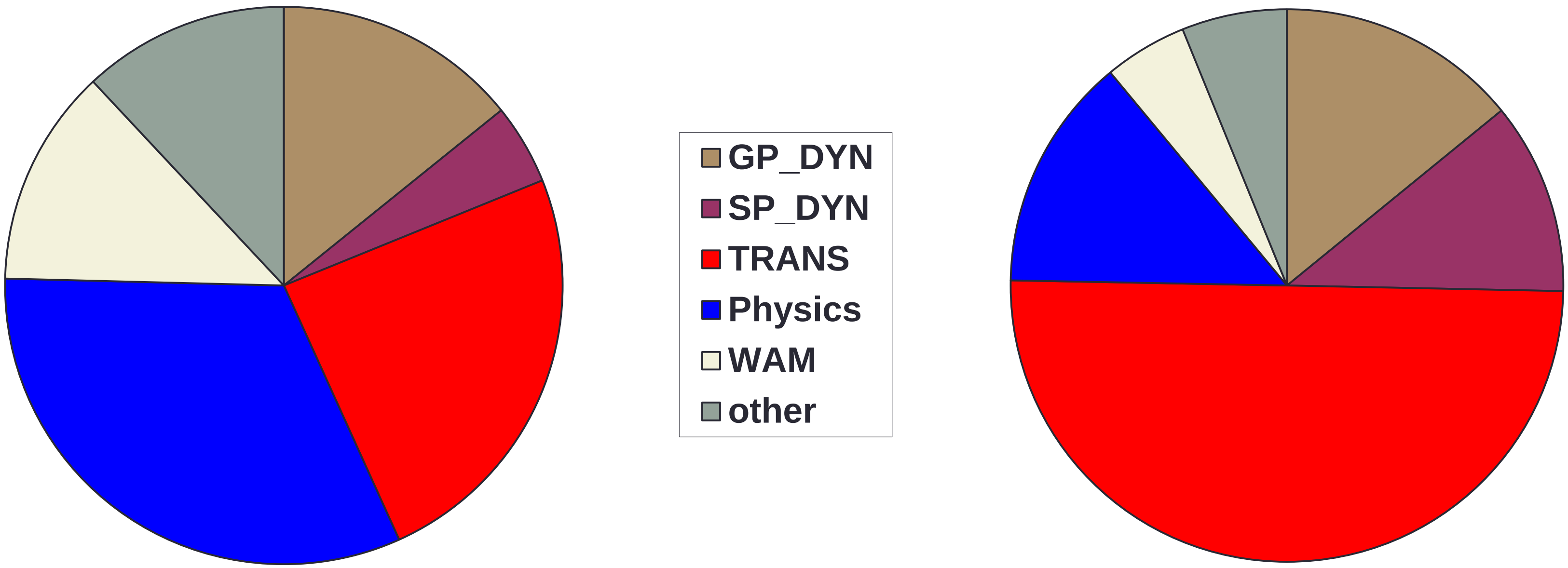
T159 model task 128 of 1024 tasks



T159 model task 462 of 4096 tasks



Computational Cost at T2047 and T3999



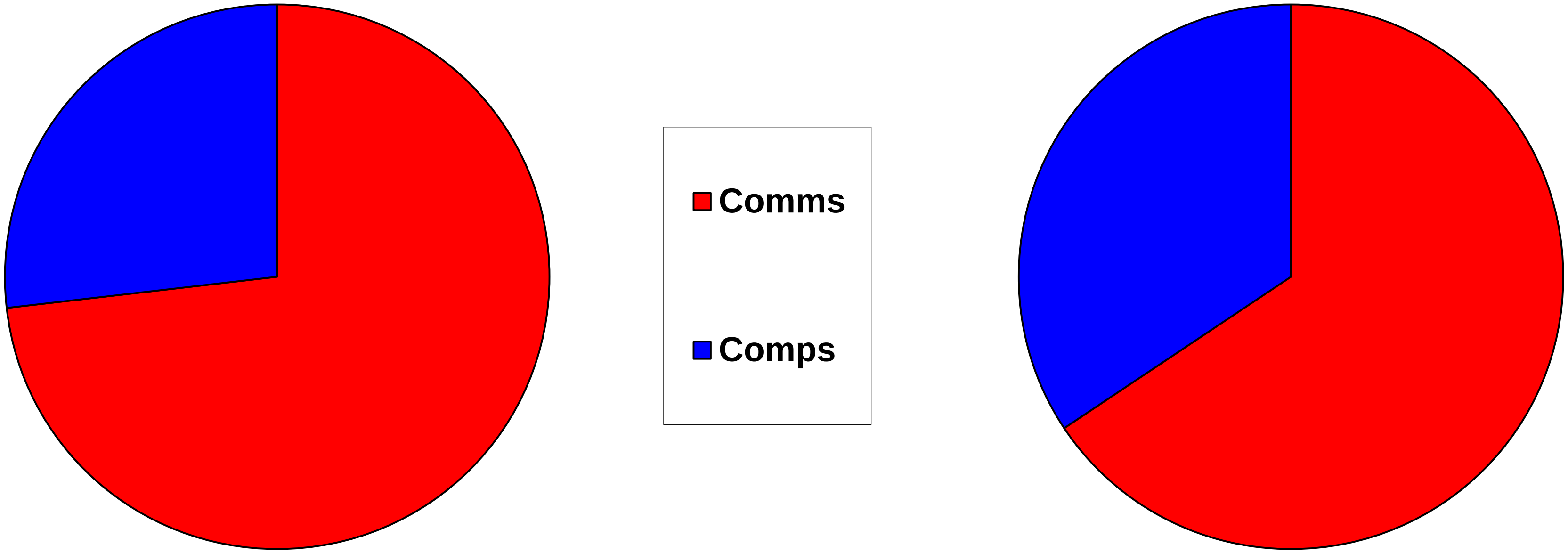
Hydrostatic T_L2047

Tstep=450s, 5.8s/Tstep
With 256x16 ibm_power6

Non-Hydrostatic T_L3999

Tstep=240s, 13.6s/Tstep
With 512x16 ibm_power6

Breakdown of TRANS cost: Computations vs. Communications



H T_L2047 ~2015

Data sent/received: 117.8GB/s

NH T_L3999 ~2020

Data sent/received: 289.6GB/s

IFS Introduction – A history

- **Resolution increases of the deterministic 10-day medium-range Integrated Forecast System (IFS) over ~25 years at ECMWF:**
  - **1987: T 106 (~125km)**
  - **1991: T 213 (~63km)**
  - **1998: T_L319 (~63km)**
  - **2000: T_L511 (~39km)**
  - **2006: T_L799 (~25km)**
  - **2010: T_L1279 (~16km)**

Introduction – A history

- **Resolution increases of the deterministic 10-day medium-range Integrated Forecast System (IFS) over ~25 years at ECMWF:**
  - **1987: T 106 (~125km)**
  - **1991: T 213 (~63km)**
  - **1998: T_L319 (~63km)**
  - **2000: T_L511 (~39km)**
  - **2006: T_L799 (~25km)**
  - **2010: T_L1279 (~16km)**
  - **2015?: T_L2047 (~10km)**
  - **2020-???: (~1-10km) Non-hydrostatic, cloud-permitting, substantially different cloud-microphysics and turbulence parametrization, substantially different dynamics-physics interaction ?**

The Gaussian grid

About 30% reduction in number of points

Full grid

Reduced grid

Reduction in the number of Fourier points at high latitudes is possible because associated Legendre functions are very small near the poles for large *m*.

(Adaptive) Mesh Refinement

- **The IFS model is inherently based on a fixed structured mesh due to the link between the spectral representation and the position of the grid-points (zero's of the ordinary Legendre polynomials), which makes selective mesh refinement (adaptive or not) difficult to achieve.**
- **"AMR" possibilities: coexisting global multigrids, physics/ dynamics on different grids, wavelet-collocation methods, …: Costly investment both in RD and computational cost**
- **Hence it is of strategic importance to understand the added-value of adaptive or static mesh refinement for multiscale global NWP and climate prediction !**

Nonhydrostatic IFS (NH-IFS)

Bubnovă et al. (1995); Bénard et al. (2004), Bénard et al. (2005), Bénard et al. (2010), Wedi et al. (2009), Yessad and Wedi (2011)

- **Arpégé/ALADIN/Arome/HIRLAM/ECMWF nonhydrostatic dynamical core, which was developed by Météo-France and their ALADIN partners and later incorporated into the ECMWF model and also adopted by HIRLAM.**

Numerical solution

- **Two-time-level, semi-implicit, semi-Lagrangian.**
- **Semi-implicit procedure with two reference states, with respect to gravity and acoustic waves, respectively.**
- **The resulting Helmholtz equation can be solved (subject to some constraints on the vertical discretization) with a *direct spectral method*, *that is, a mathematical separation of the horizontal and vertical part of the linear problem in spectral space, with the remainder representing at most a pentadiagonal problem of dimension NLEV*^{2}*. Non-linear residuals are treated explicitly (or iteratively implicitly)!***

(Robert, 1972; Bénard et al 2004,2005,2010)

The spectral transform method

Eliasen et. al (1970), Orszaag (1970)

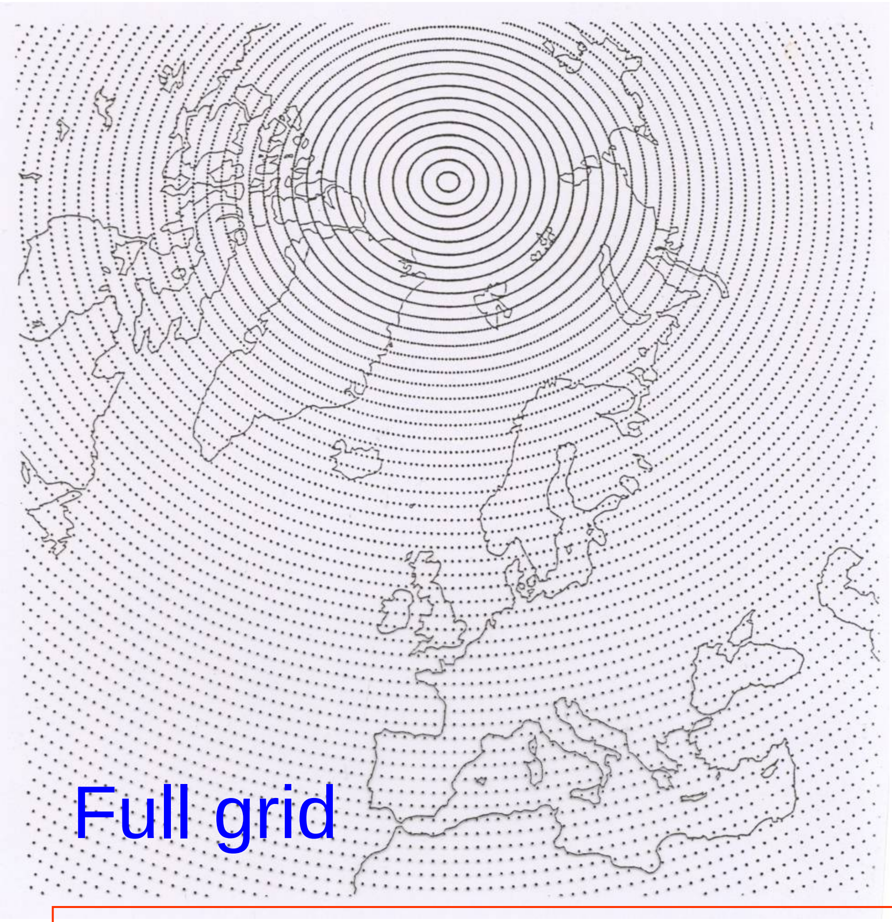
Applied at ECMWF for the last 30 years …

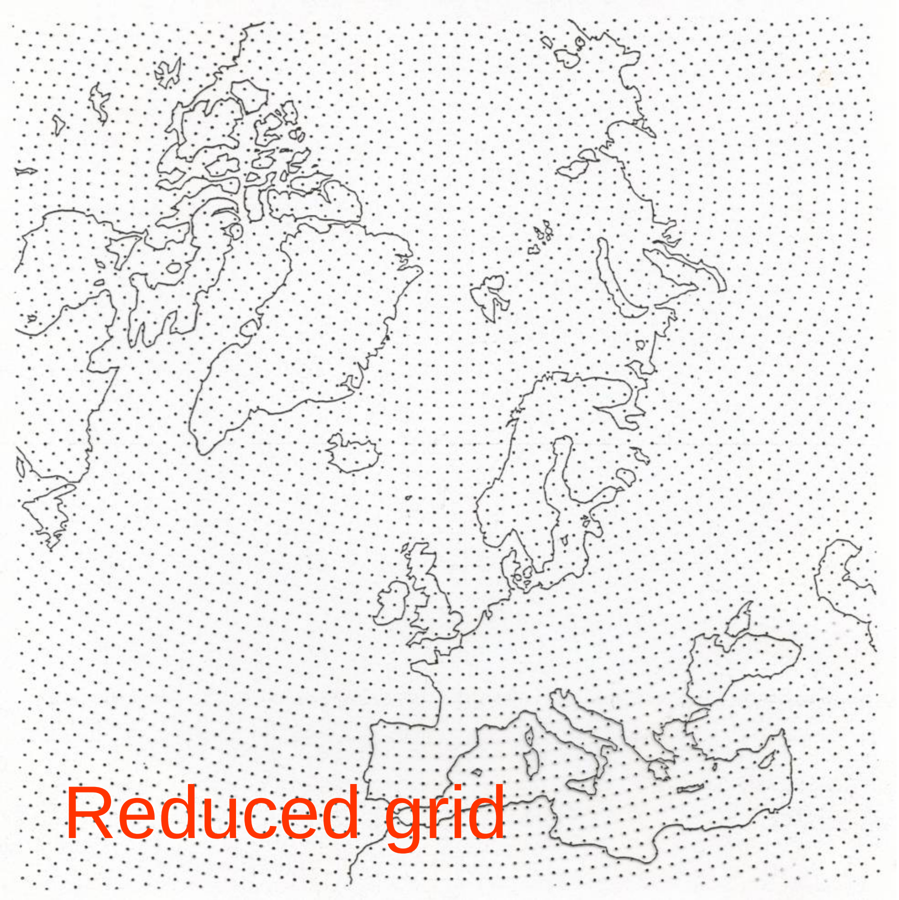
Spectral semi-Lagrangian semi-implicit (compressible) a viable option ?

-Computational efficiency on future MPP architectures ?
-Accuracy at cloud-resolving scales ?
-Suitability for the likely mixture of medium and high resolution ensembles and ultra-high resolution forecasts ?

The Gaussian grid

About 30% reduction in number of points





Reduction in the number of Fourier points at high latitudes is possible because associated Legendre polynomials are very small near the poles for large *m*.

Note: number of points nearly equivalent to quasi-uniform icosahedral grid cells of the ICON model.

Cost of the spectral transform method

- FFT can be computed as C*N*log(N) where C is a small positive number and N is the cut-off wave number in the triangular truncation.
- Ordinary Legendre transform is $O(N^2)$ but can be combined with the fields/levels such that the arising matrix-matrix multiplies make use of the highly optimized BLAS routine DGEMM.
- But overall cost is $O(N^3)$ for both memory and CPU time requirements.

⇨ Desire to use a fast Legendre transform where the cost is proportional to C*N*log(N) with C << N

and thus overall cost N^2*log(N)

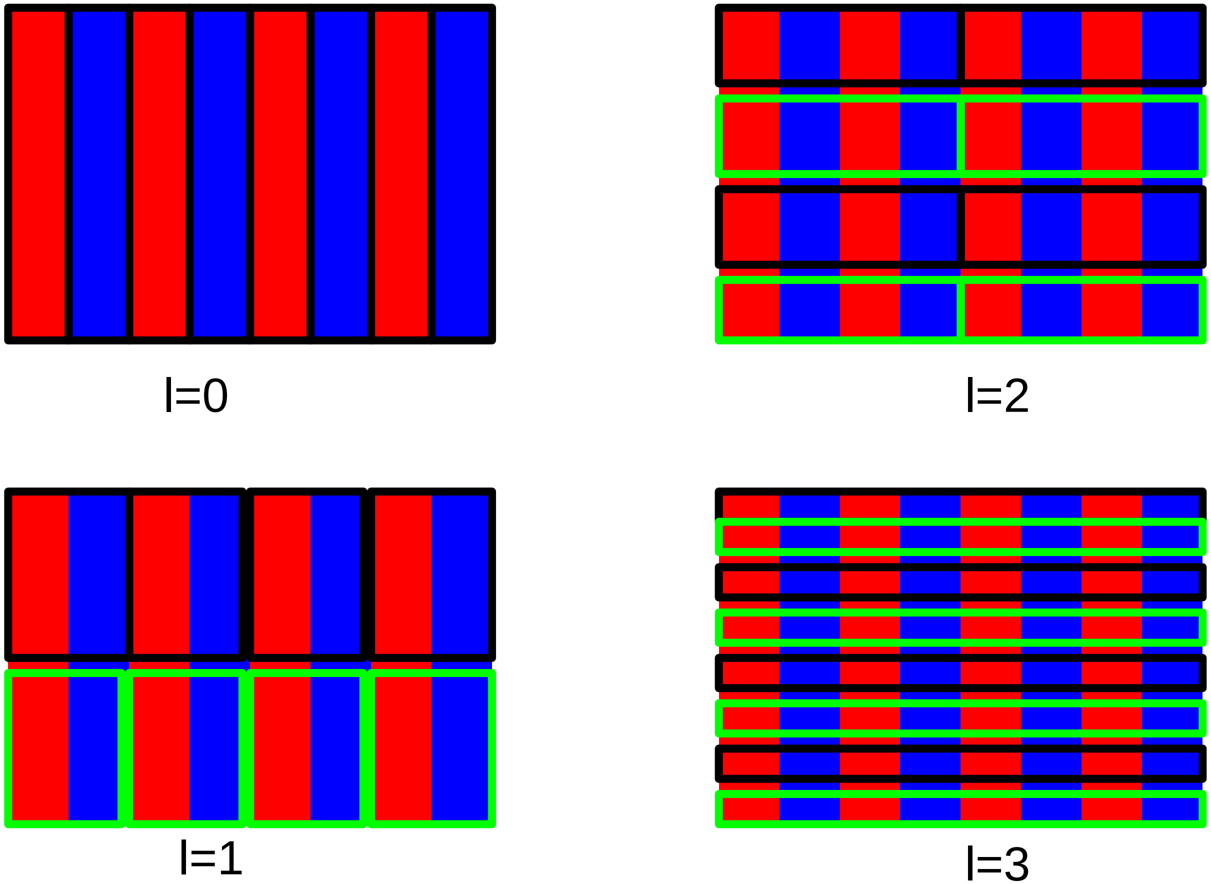
Fast Legendre Transform (FLT)

- **The algorithm proposed in *(Tygert, 2008,2010)* suitably fits into the IFS transform library by simply replacing the matrix-matrix multiply DGEMM call with a BUTTERFLY_MATRIX_MULT call plus slightly more expensive pre-computations.**
- **(1) Instead of the recursive *Cuppen divide-and-conquer algorithm (Tygert, 2008)* we use the so called *butterfly algorithm (O'Neil et al, 2009; Tygert, 2010)* based on a matrix compression technique via rank reduction with a specified accuracy to accelerate the arising *matrix-vector/matrix multiplies (sub-problems still use DGEMM).***
- **(2) We apply the matrix compression directly on the matrix of the associated polynomials, which reduces the required precomputa-tions and eliminates the need to apply *FMM (fast multipole method)* accelerated interpolations. Notably, the latter were an essential part of the proposed FLT in *Suda and Takami (2001).***

The butterfly compression

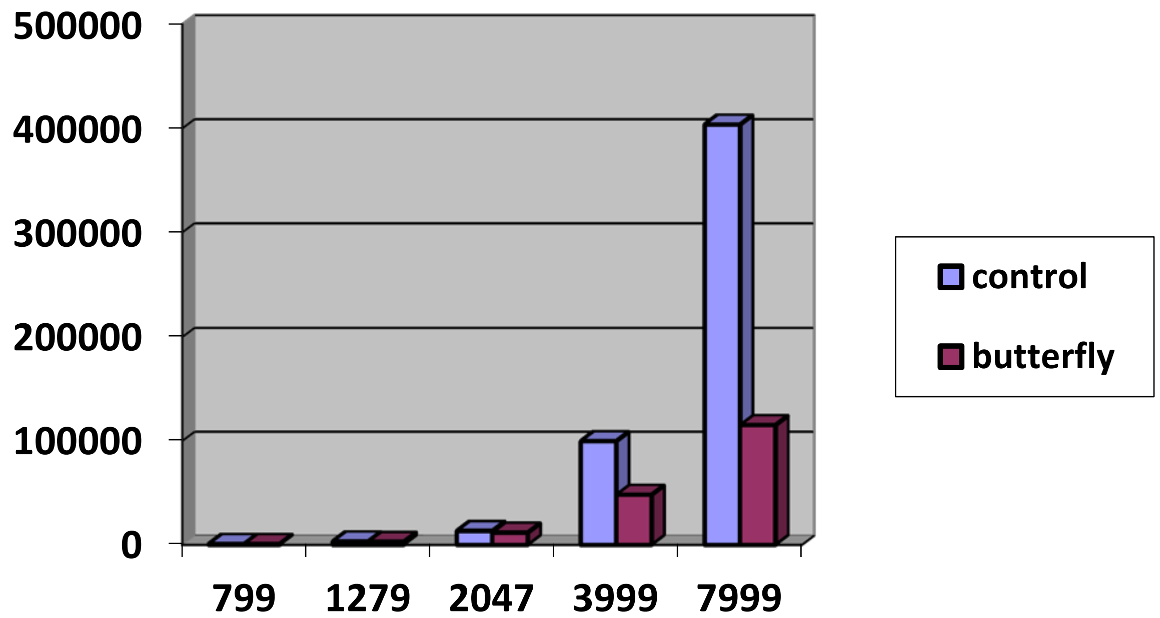
(O'Neil, Woolfe, Rokhlin, 2009; Tygert 2010)

With each level l, double the columns and half the rows



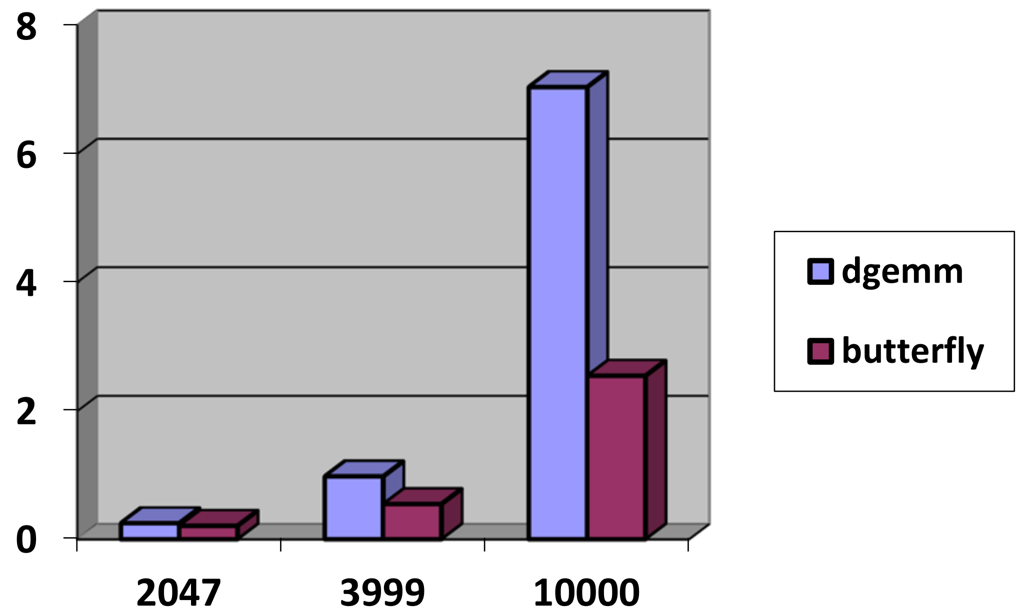
Floating point operations per time-step in Gflop

Inverse transform of single field/level



Wallclock time in seconds

inverse transform of 10 fields, offline test environment

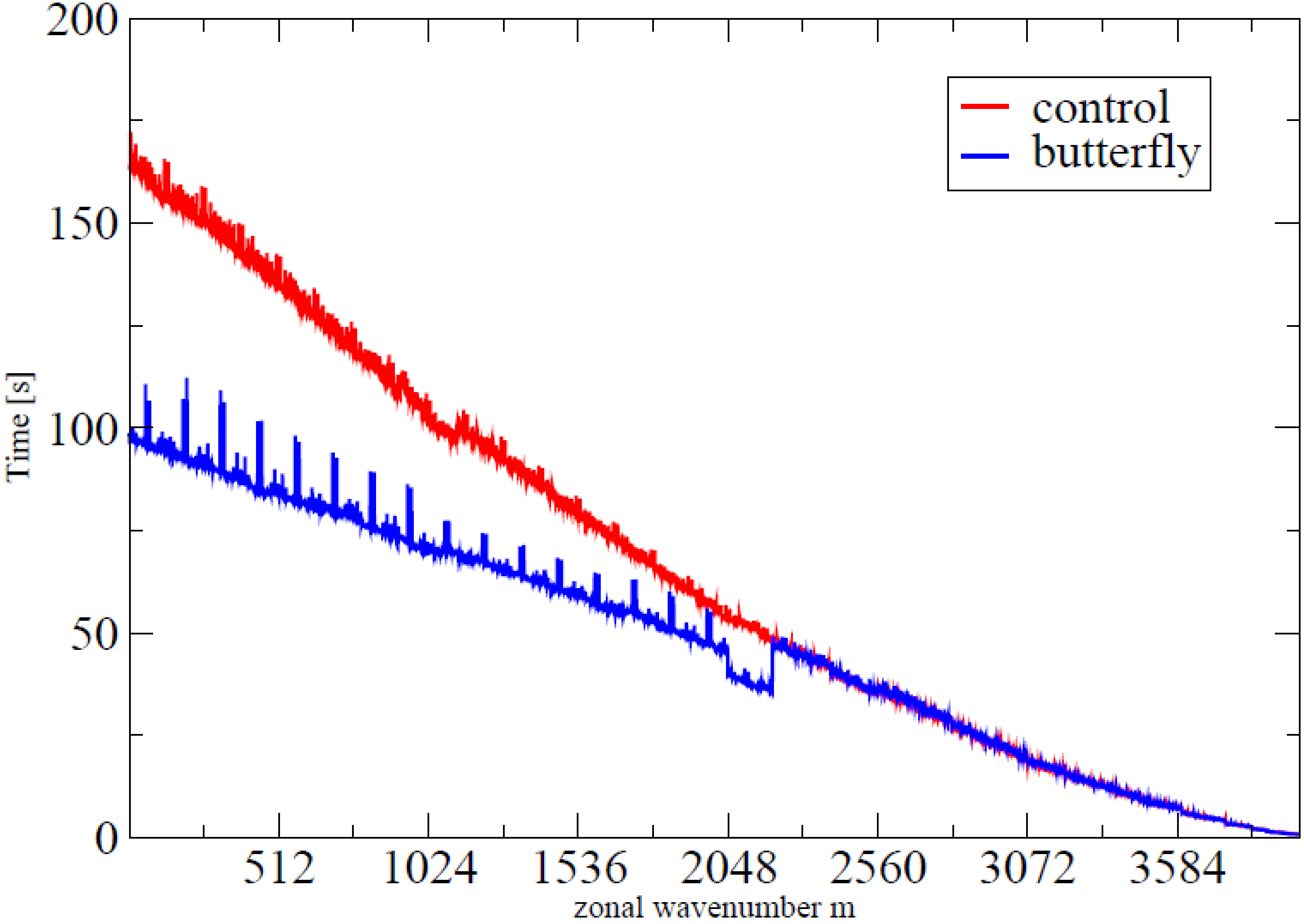


ECMWF

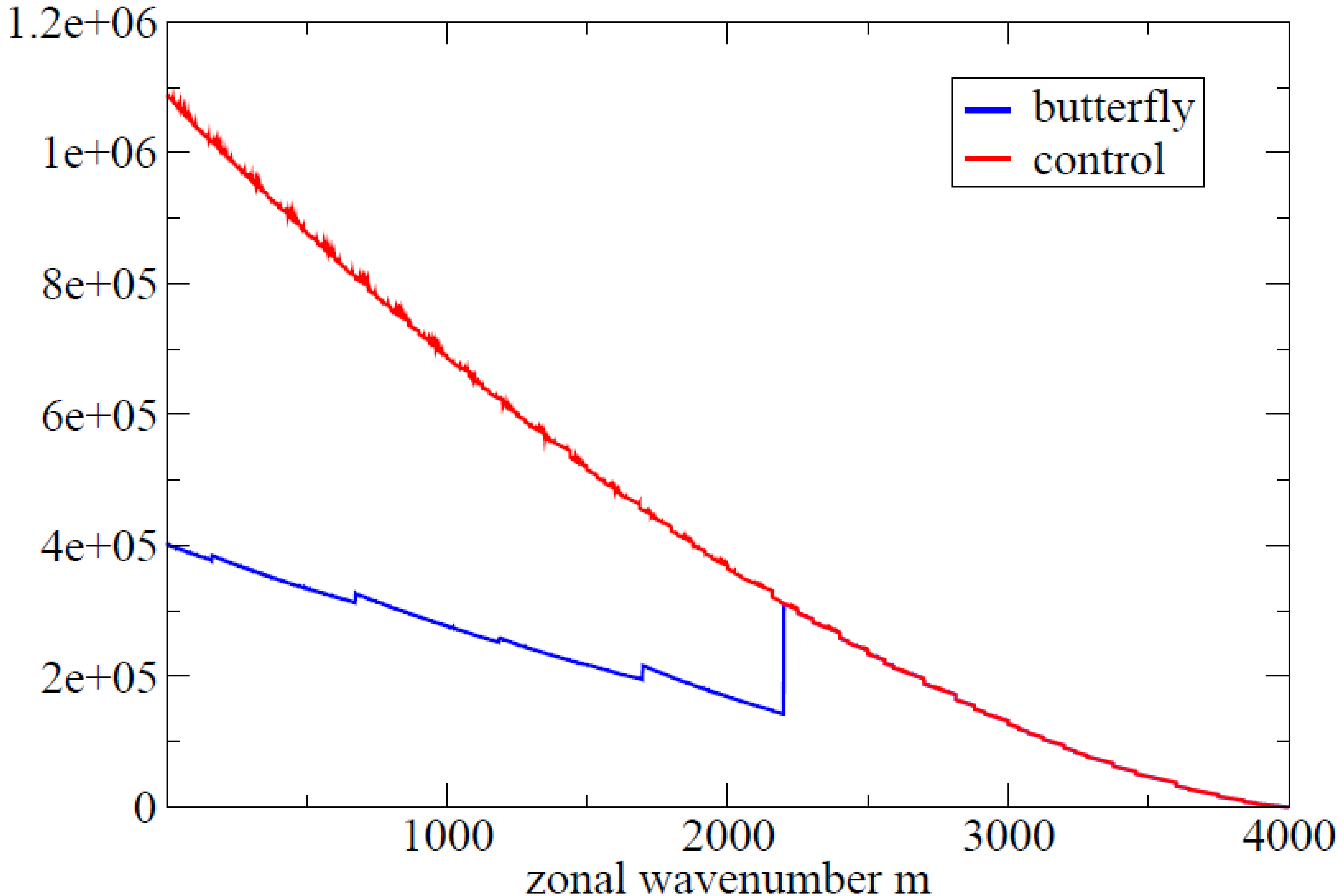
Selected projects to prepare for exascale computing in Meteorology (NWP)

- ***ICOMEX*** **–** ***ICO*****sahedral grid Models for** ***EX*****ascale Earth system simulations (2011-2014)**
- **Gung-Ho – Development of the Next Generation Dynamical Core for the UK MetOffice (2 phases, 2011-2013, 2013-2016)**
- **CRESTA –** ***C*****ollaborative** ***R*****esearch into** ***E*****xascale** ***S*****ystemware,** ***T*****ools &** ***A*****pplications (2011-2014)**

T3999 6h forecast - inverse transforms: CPU time vs. wave number



T3999 6h forecast - inverse transforms: Floating point operations vs. wave



ECMWF

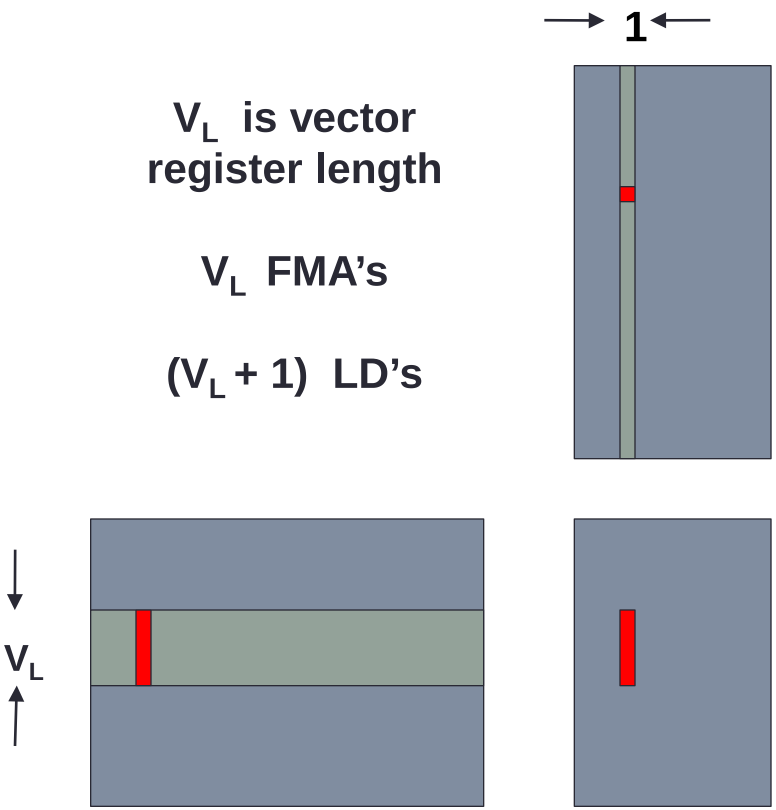
Why is Matrix Multiply (DGEMM) so efficient?

VECTOR

V_L is vector register length

V_L FMA's

(V_L + 1) LD's



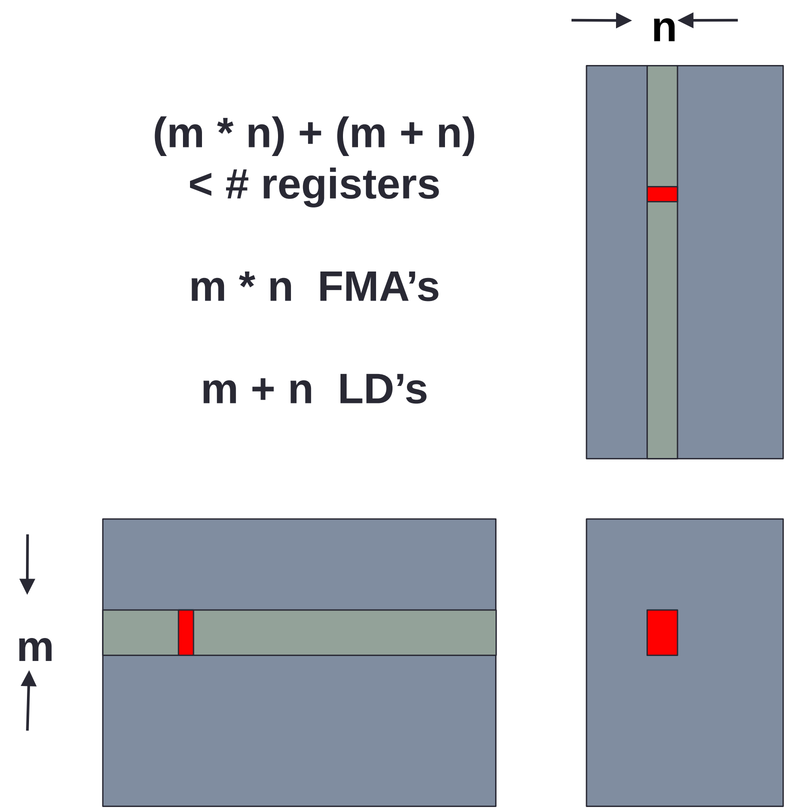
FMA's ~= LD's

SCALAR / CACHE

(m * n) + (m + n) < # registers

m * n FMA's

m + n LD's



FMA's >> LD's