Computer Architectures & Aspects of NWP models

Deborah Salmond
ECMWF

Plan

1) Supercomputers for NWP
2) IFS from ECMWF and UM from Met Office
3) Message passing
4) OpenMP
5) Dr Hook
6) Optimisation
7) 4D-Var

Design evolution of supercomputers

-Scalar or Vector:

Cache & locality of data or memory bank conflicts

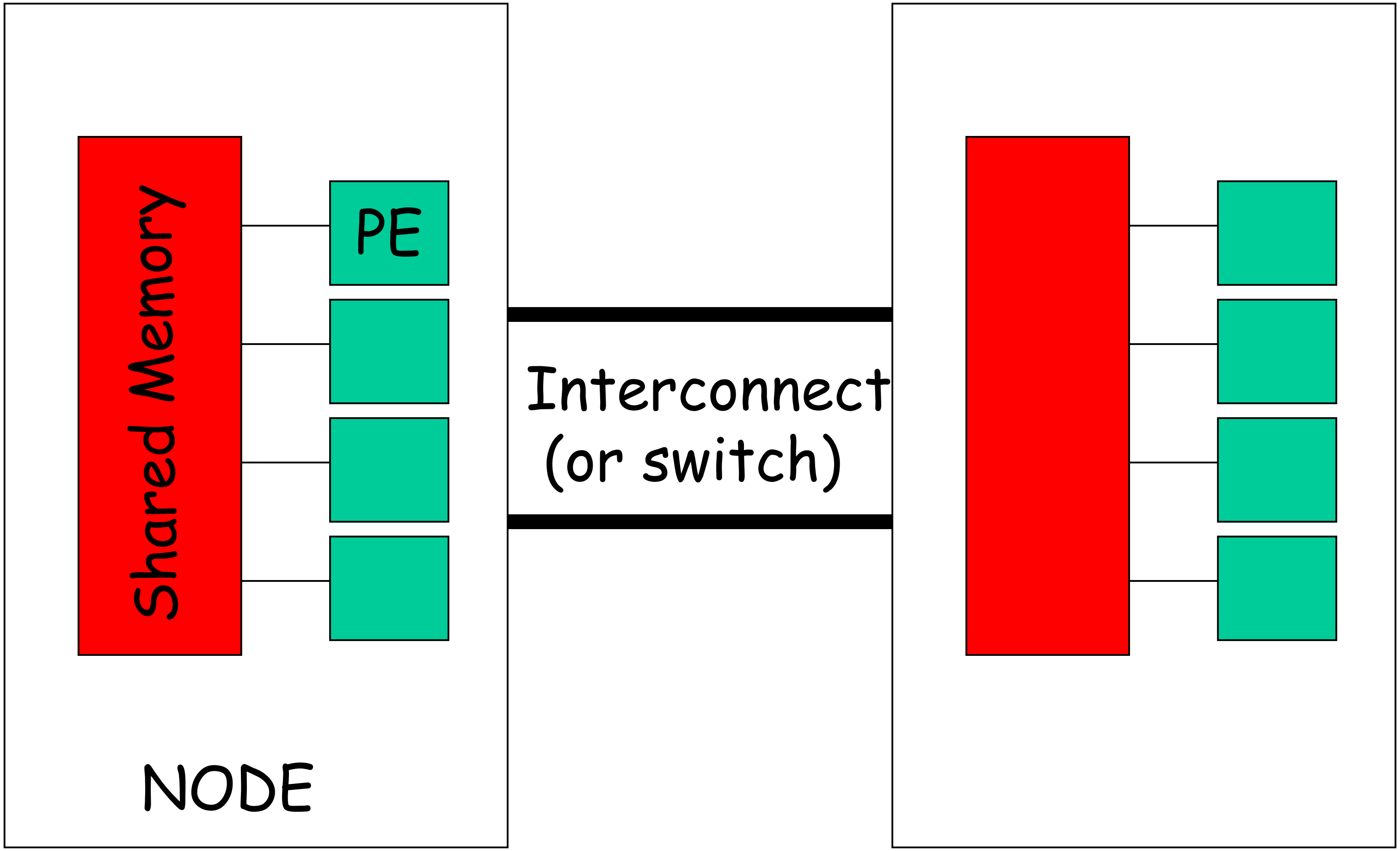
-Single processor -> Multi-processor

Shared memory -> Distributed memory -> Clusters

-Parallelisation: MPI – distributed memory – 'tasks'

OpenMP – shared memory – 'threads'

Shared Memory – Distributed Memory



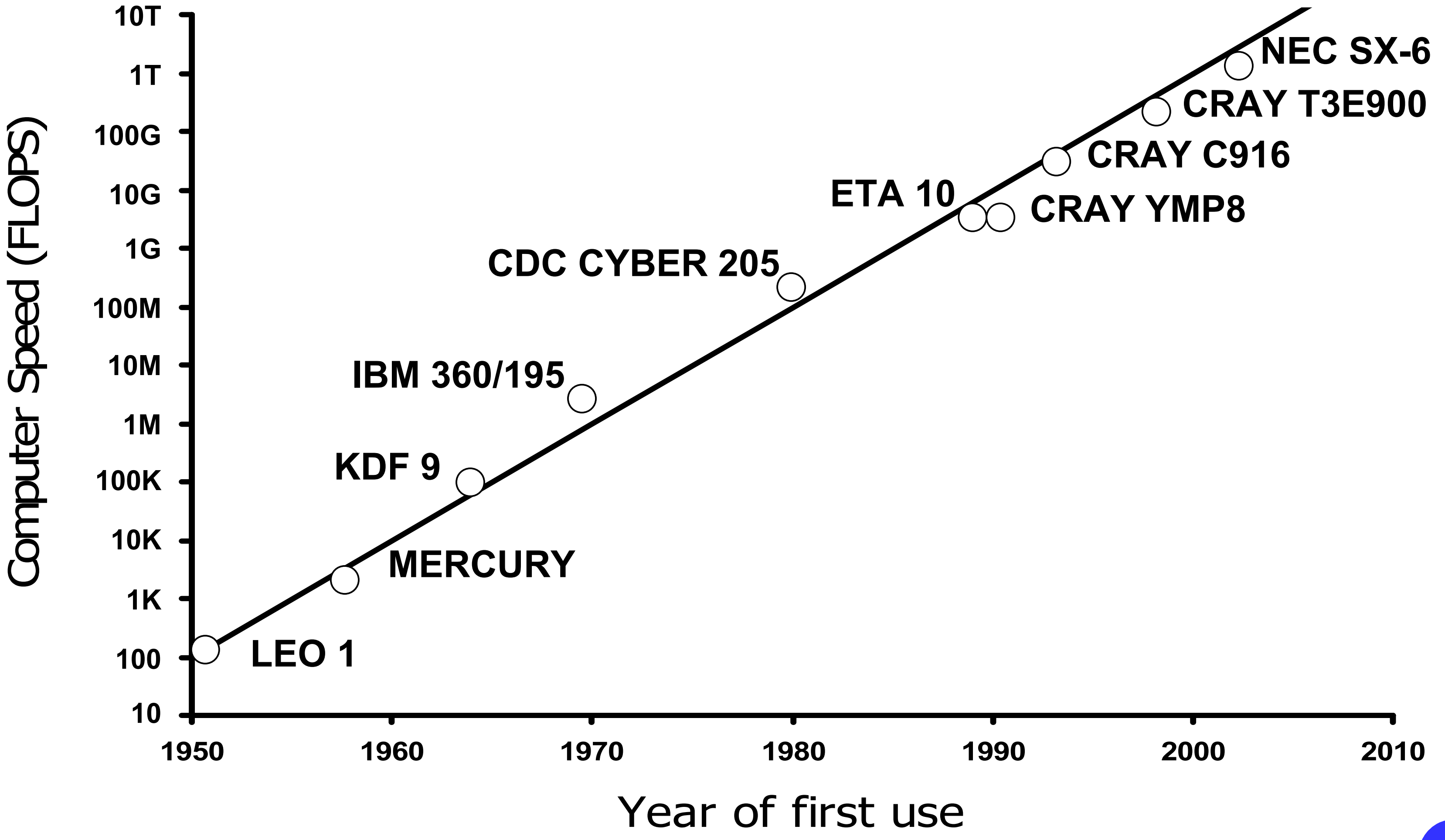
Supercomputers for NWP

			Number of procs	Peak Gflops per proc	Shared memory nodes
ECMWF	IBM p690 (hpca) IBM p690+ (hpcd)	scalar	2 x 960 2 x 2176	5.2 7.6	8 32
Met Office	NEC SX-6	vector	2 x 120	8	8
Météo France	Fujitsu VPP5000	vector	124	9.6	No
INM	CRAY X1	vector	60 MSPs 240 SSPs	12.8 3.2	4
DWD	IBM NH-2	scalar	1920	1.5	16

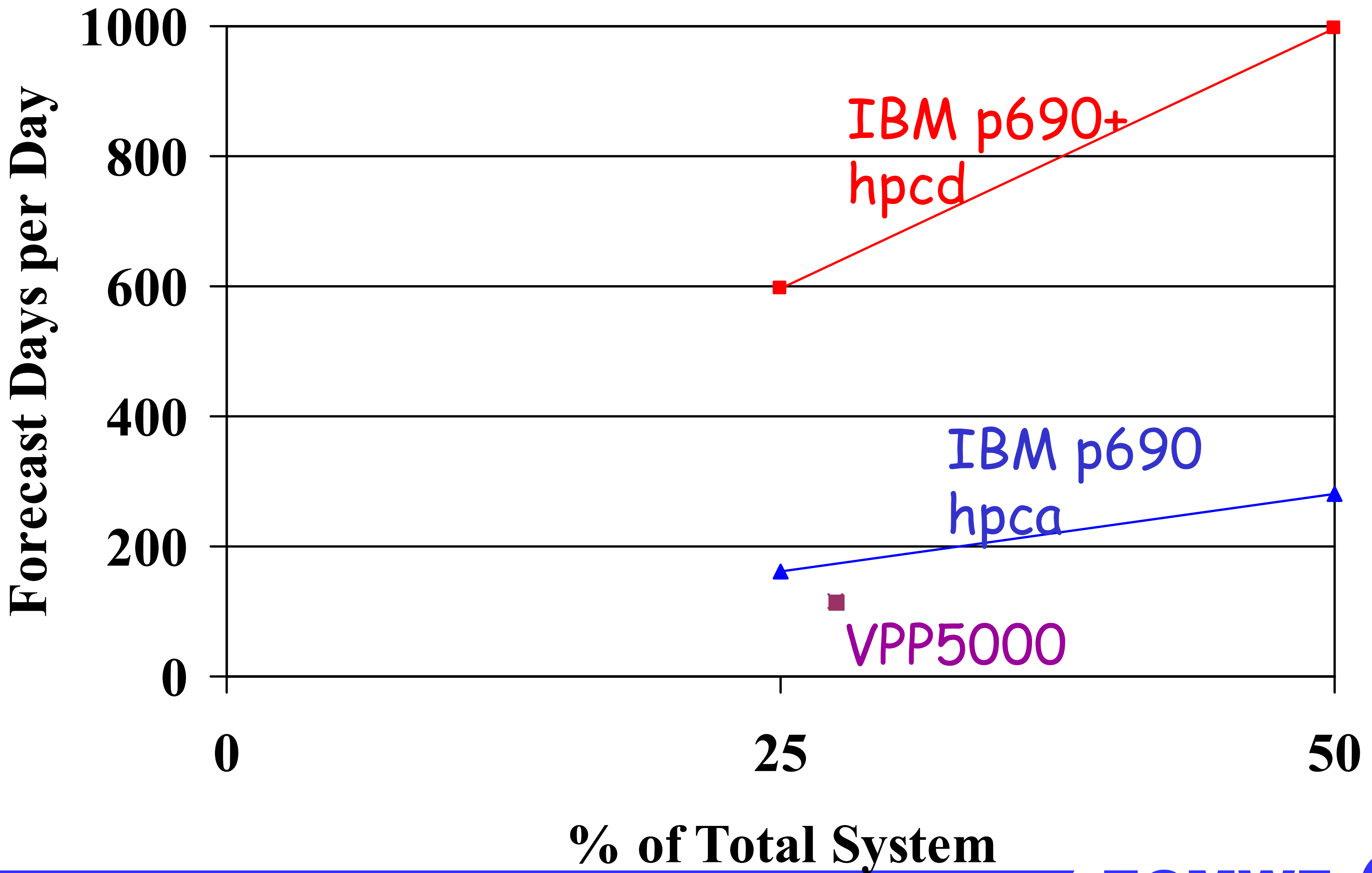
Previous -> Current Supercomputers for NWP

ECMWF	Fujitsu VPP5000 -> 2 IBM p690+	vector -> scalar
Met Office	2 CRAY T3Es -> 2 NEC SX-6s	scalar -> vector
Météo France	CRAY C90 -> Fujitsu VPP5000	vector
INM	CRAY SV1 -> CRAY X-1	vector
DWD	CRAY T3E -> IBM NH-2	scalar

Computers used by ‘The Met Office’



RAPS-6 : T799 L90 benchmark run on ECMWF supercomputers



Different adaptations of the NWP models to suit available computers

- **Met.Office model** initially coded in 1970's on IBM 360/195 with inner loop over vertical levels in assembler on single CPU

  -> Coded for CDC Cyber205 with Nlat*Nlon inner loop
  -> 'New Dynamics' coded with Nlat and Nlon loops

-**IFS** coded with inner loops over horizontal in groups of NPROMA to give long vectors - now good for cache.

  -> Parallelised using MPI and OpenMP & scalable up to O(2000) processors.

IFS from ECMWF & UM from Met Office

Compare some computer characteristics from 2 different models:

-> IFS a hydrostatic spectral model run at 40 km resolution (60 levels) to 10 days on 256 processors on a Scalar computer (IBM p690) in 1 hour at 80 Gflops (317 Tflop)

-> UM a non-hydrostatic grid-point model run at 60 km resolution (38 levels) to 7 days on 14 processors on a Vector computer (NEC SX-6) in ½ hour at 16 Gflops (35 Tflop)

IFS - Overview

Spectral Model with Semi-Lagrangian Advection

Parallelisation

- for distributed memory and shared memory

Reduced Grid

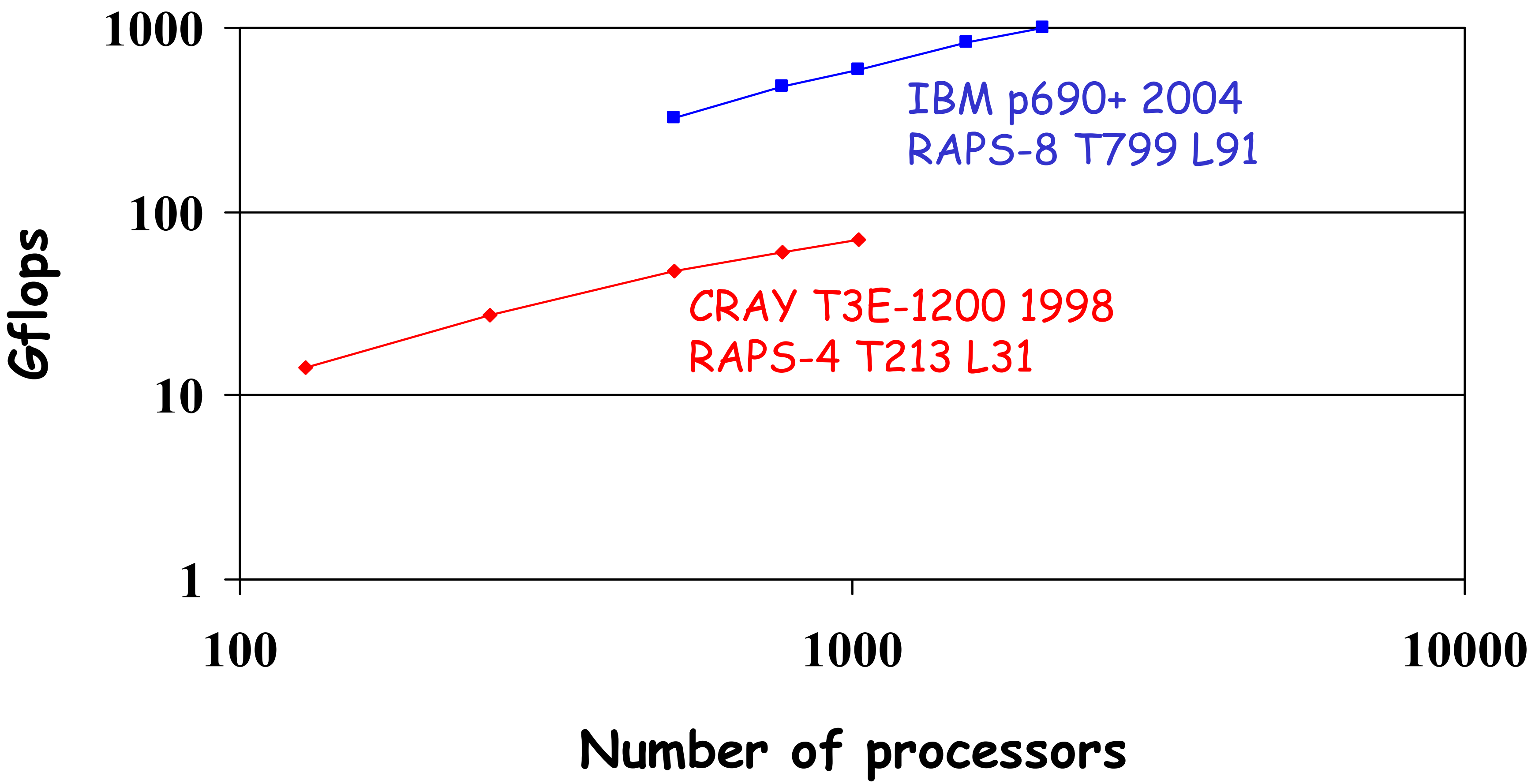
Physics and Dynamics

- computed in blocks of NPROMA

Bit reproducible with different NPROMA & no. of PEs

FORTRAN 90 + C

Performance of IFS Forecast



NPROMA

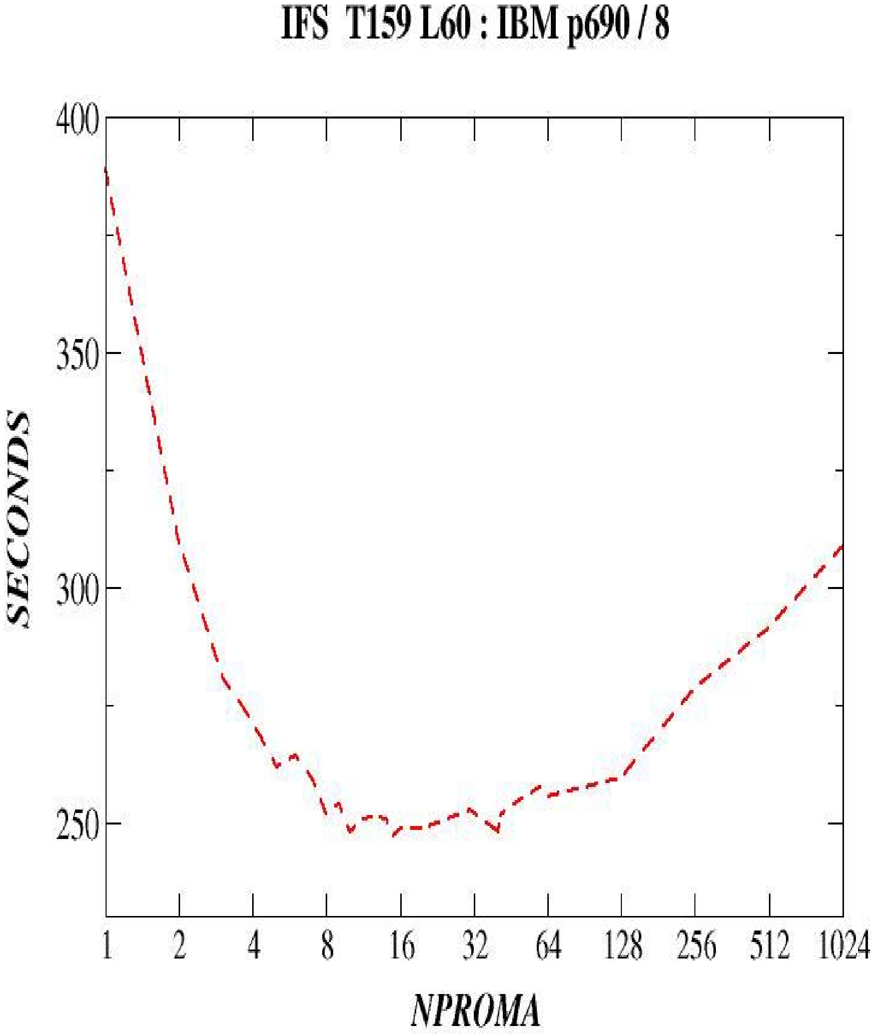
No dependency in horizontal
-> so dynamics and physics can be done in groups of
NPROMA horizontal points

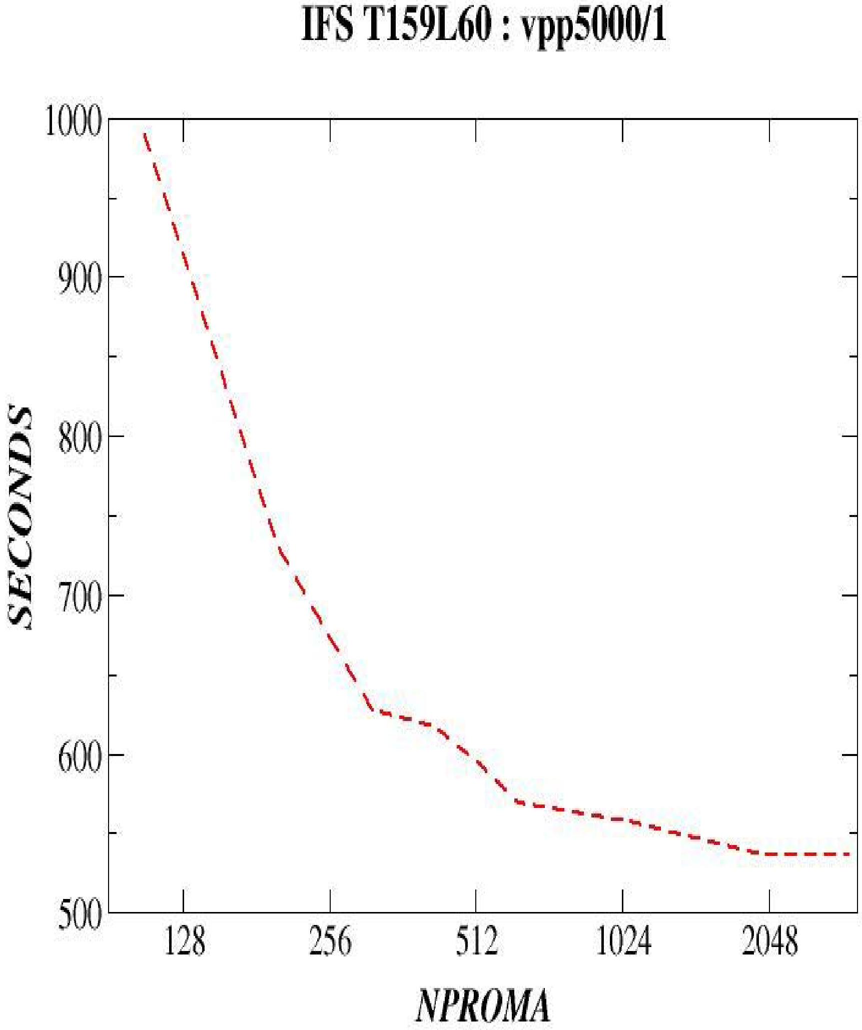
NPROMA is chosen to be
Long for Vector
Short for Scalar/cache

Memory Saving and OpenMP

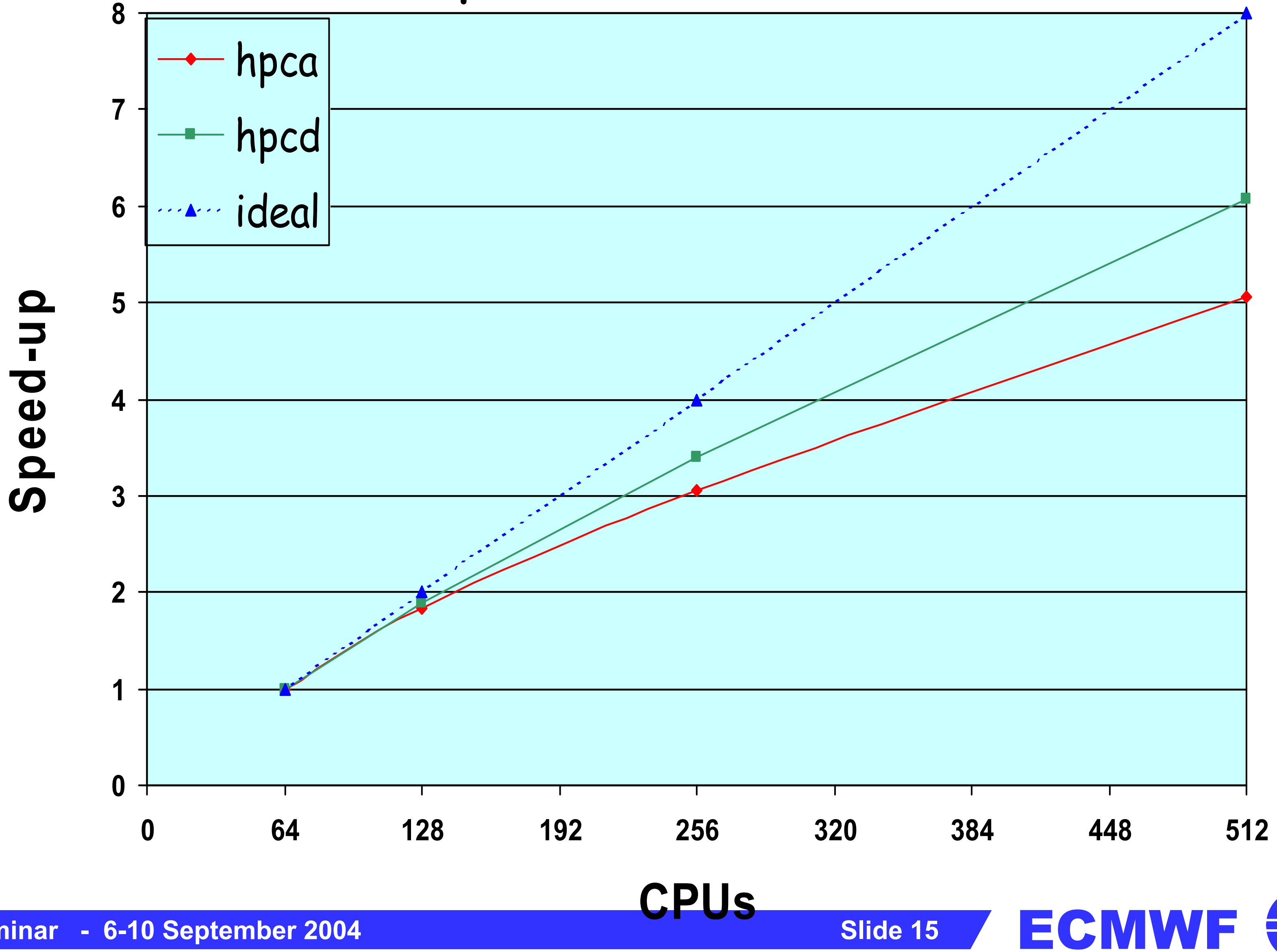
```
!$DO PARALLEL
Do IBL=1,nblocks
   CALL EC_PHYS

  Do Jlev=1,Nlev
     Do I=1,Nproma
```

NPROMA

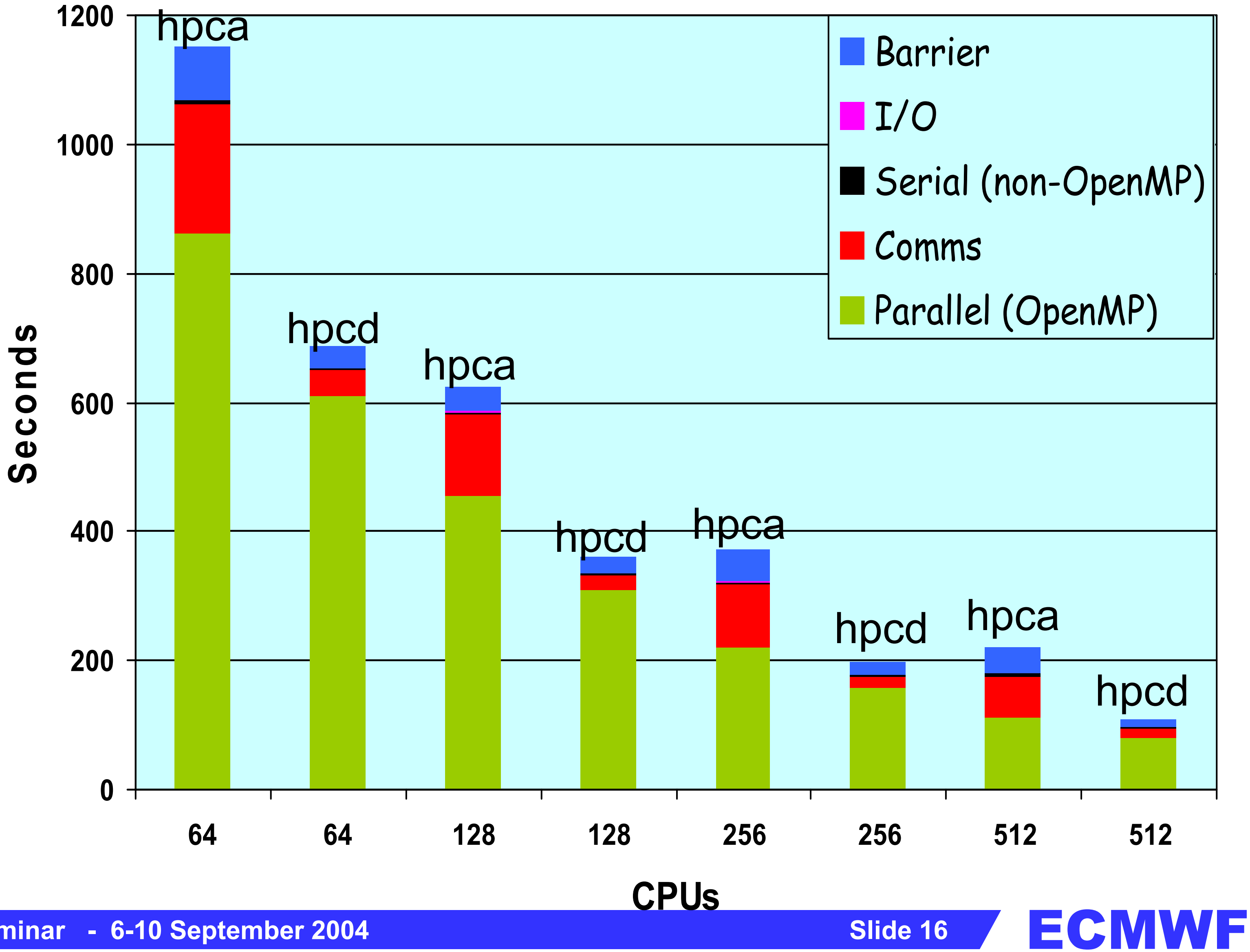

IFS T159 L60 : IBM p690 / 8
SECONDS
400
350
300
250
1
2
4
8
16
32
64
128
256
512
1024
NPROMA

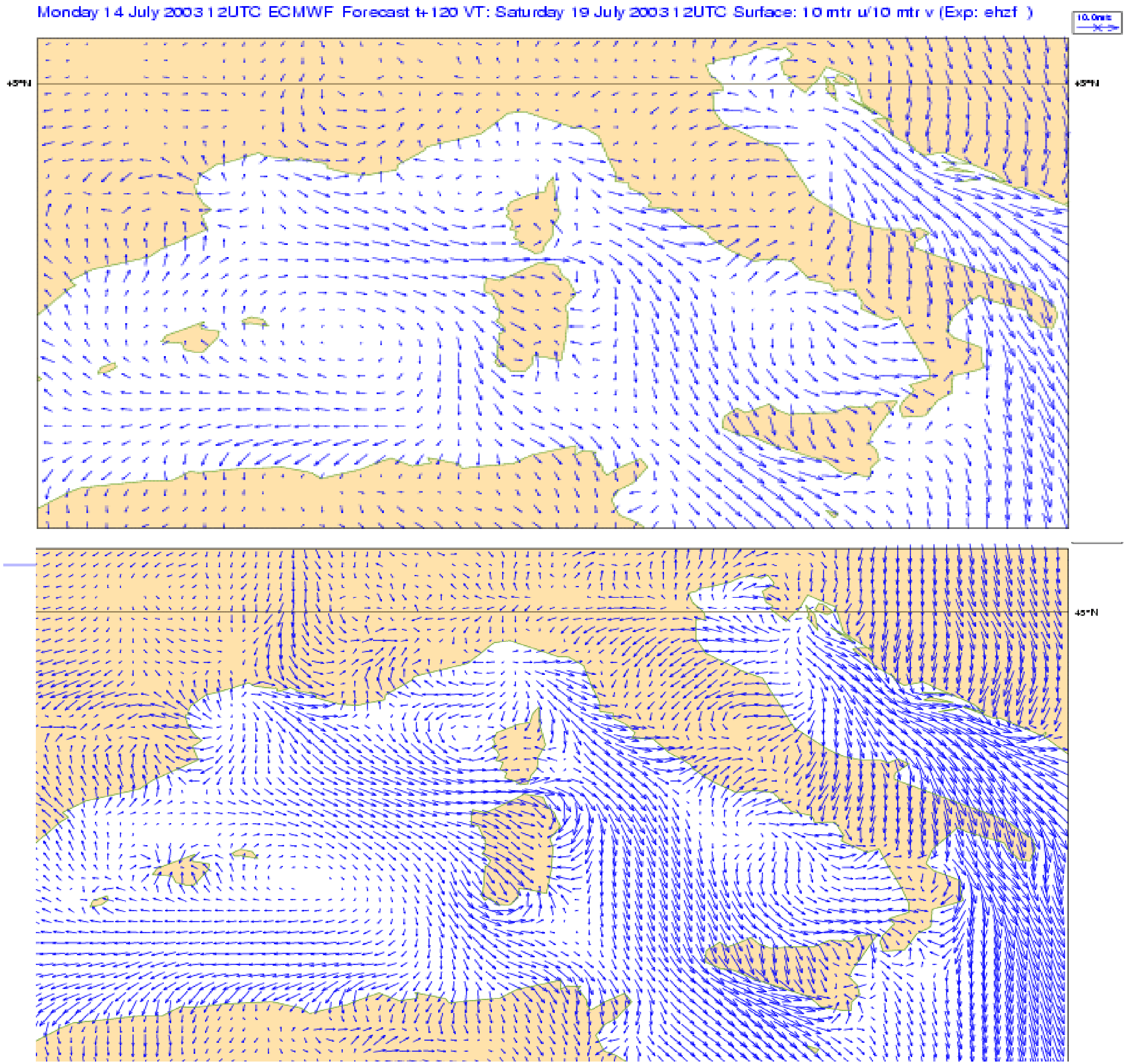

IFS T159L60 : vpp5000/1
SECONDS
1000
900
800
700
600
500
128
256
512
1024
2048
NPROMA

T511 1-day Forecast CY28R2 on hpca v hpcd
4 OpenMP threads



T511 1-day Forecast CY28R2 on hpca & hpcd
4 OpenMP threads



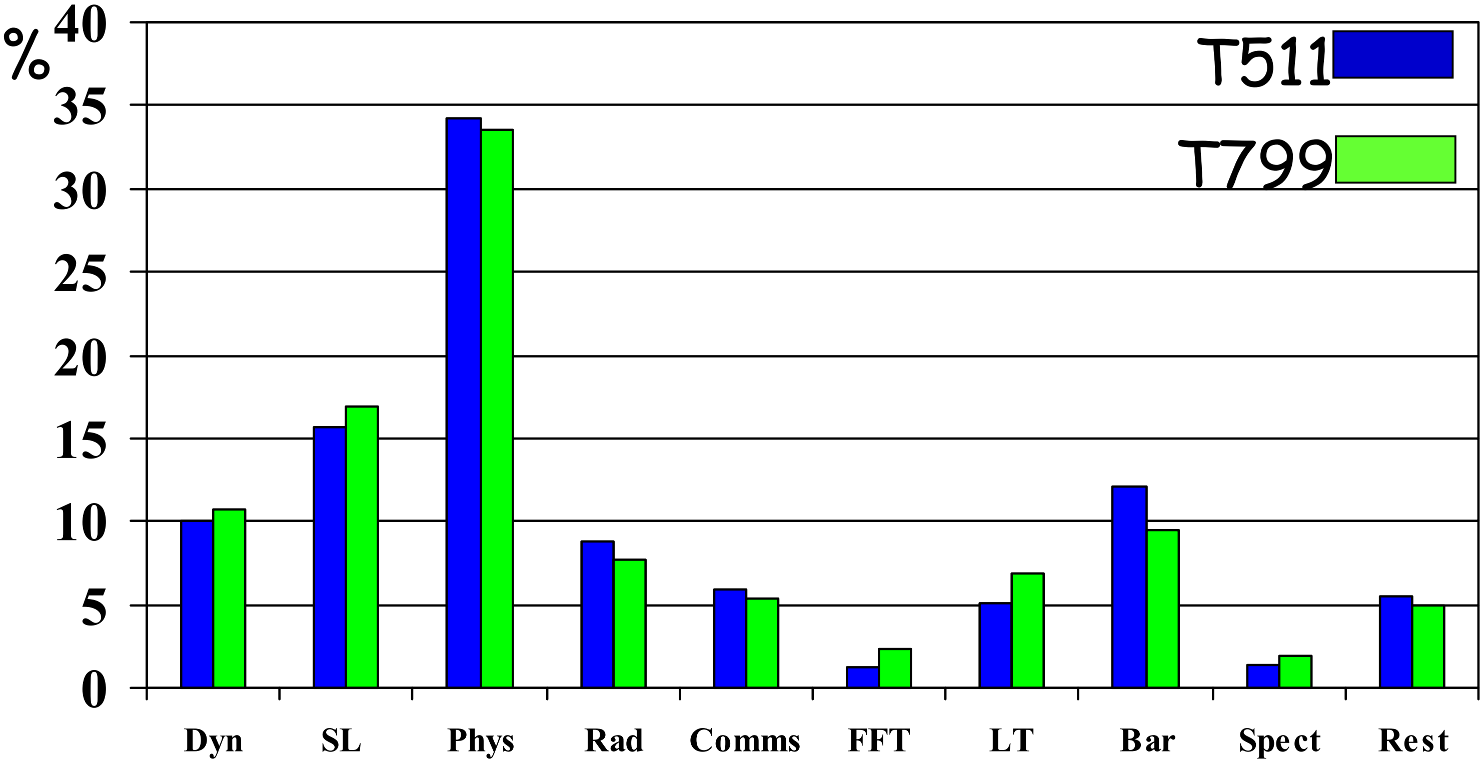

Monday 14 July 2003 12UTC ECMWF Forecast t+120 VT: Saturday 19 July 2003 12UTC Surface: 10 mtr u/10 mtr v (Exp: ehzf)
10.0m/s
45°N
45°N
45°N

T511

10m
winds

T799

T511- L60 compared with T799 - L91 run on hpcd with 64 MPI Tasks and 4 OpenMP threads



Extra cost for T799 - L91 = (900/720)*3.5 times

RAPS-8: T799 L91 10 day forecast on hpcd

MPI x OpenMP	Wall (secs)	FCD/D	Gflops	% of peak	% comms
64x4	8850	102.7	193	10.0%	5.2%
128 x 4	4410	197.8	369	9.5%	5.8%
192 x 4	3187	274.7	509	8.7%	8.3%
256 x 4	2534	346.1	644	8.2%	9.0%
384 x 4	1830	483.6	886	7.6%	10.5%
256 x 8	1523	584.4	1000	6.9%	13.2%

Total Tflop = 1630

RAPS-8: Ensemble forecast T399 L62 - 10 Day Forecast

Computer	CPUs MPI x OpenMP	Wall (secs) (FCD/D)	Gflops	% of Peak
CRAY X1 at INM	64 SSPs	2102 (418)	41	20.0%
IBM p690+ at ECMWF	64 CPUs 16 x 4	2102 (412)	41	8.4%
IBM p690+ at ECMWF	128 CPUs 32 x 4	1084 (805)	80	8.2%

Total Tflop for 51 copies as to be run for EPS at ECMWF = 4400

UM - Overview

Grid-point model with Semi-Lagrangian Advection

Non-hydrostatic -> 3D-solver

Parallelisation: for distributed memory only

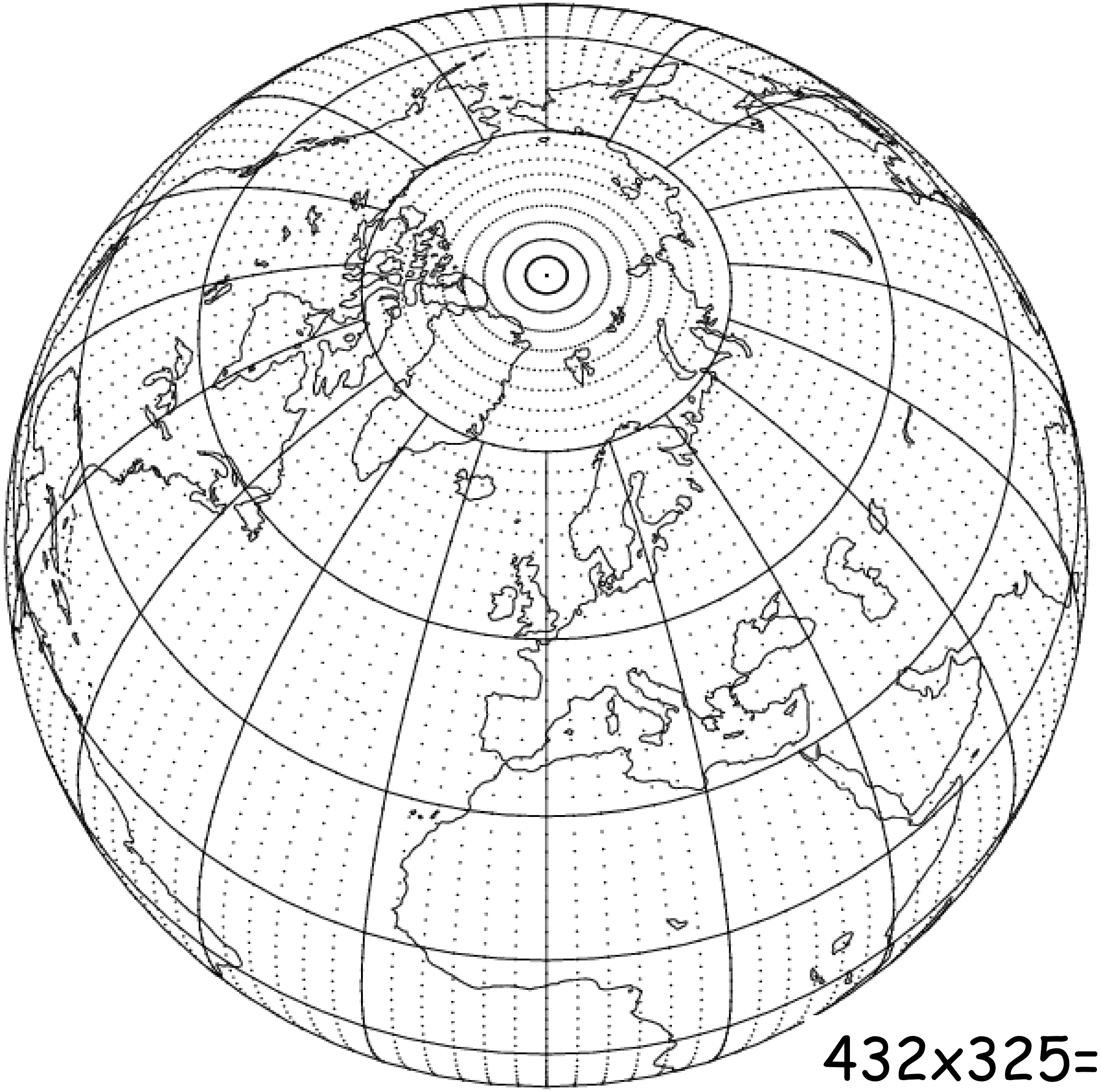
- 2D decomposition on CRAY T3E
- 1xn decomposition on NEC SX6 (limited to <54 Pes)
- many barriers

Communications – wide halo boundary swap

Bit reproducible – for climate runs
- relaxed for forecast runs

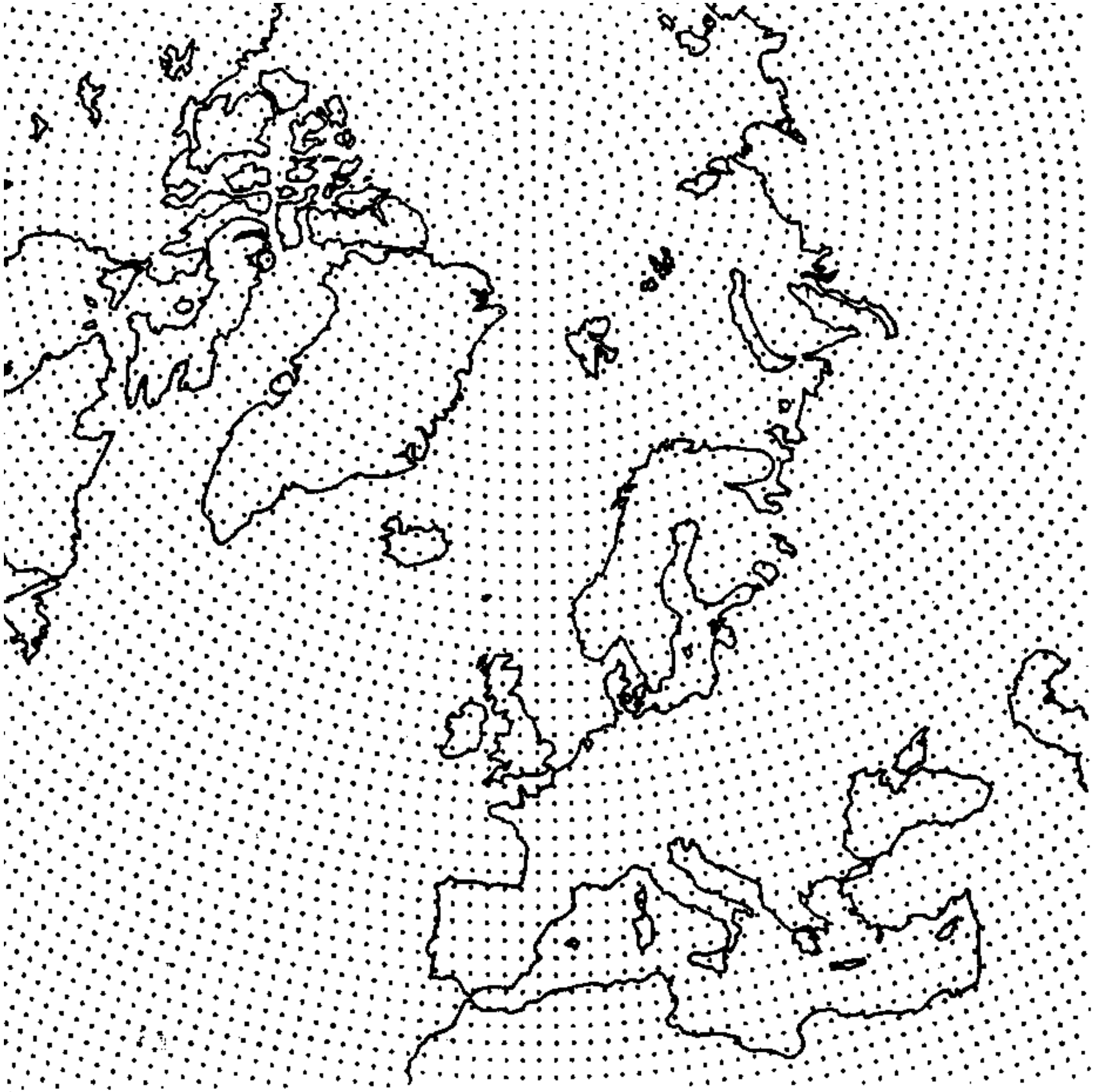
Regular lat/lon grid -> special treatment at Pole

Lat/Lon grid for UM



432x325=140400 points

Reduced Grid for IFS



Total points for T511
= 348528

Total points for T799
= 843532

Performance of UM on NEC SX-6

Resolution ~ 60km
Grid = 432 x 325 x 38

7 day forecast

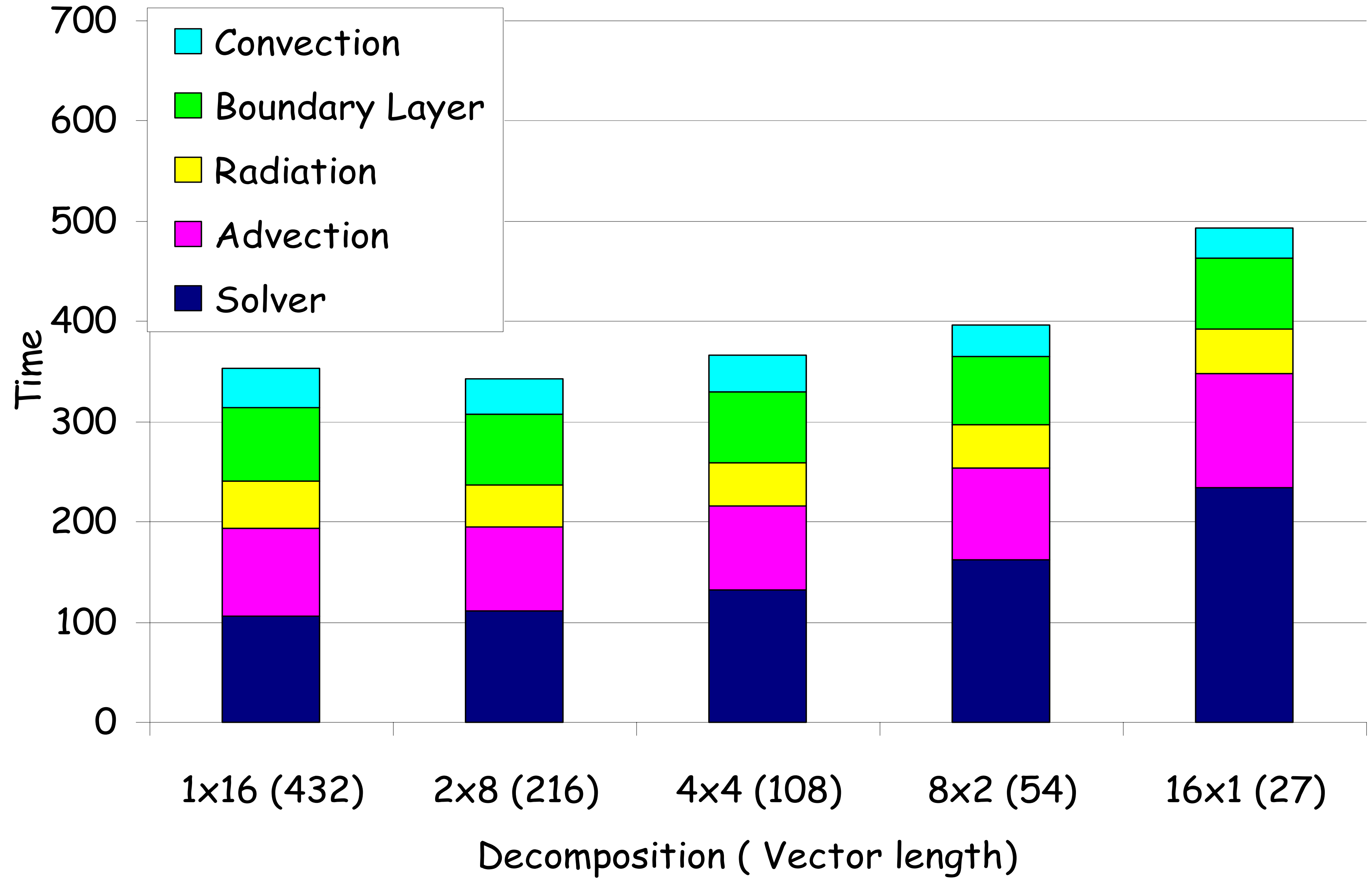
- run operationally on 2 nodes
- only 14 out of 16 CPUs used
- 1x14 decomposition
- takes 37 minutes

=> 1 NEC CPU =37 CRAY T3E procs

- runs at 1.1 Gflops per processor = 16 Gflops

Total flops = 35 Tflop

Profiles for UM on 16 processors of NEC SX-6



Message Passing

- Parallelisation for distributed memory
    - Message pass between different ‘tasks’ using MPI

        - Boundary swap for grid-point model
            --> short messages (latency of interconnect)
        - Transposition for spectral transforms
            --> long messages (bandwidth of interconnect)

    - MPI global/group communications

    - MPL and GCOM intermediate libraries

Comparison of message passing in Spectral and Grid-Point models

HIRLAM run on CRAY X1 at INM

- on average 20 flops per byte sent by MPI
- 6440 barriers per timestep

IFS run on IBM p690 at ECMWF

- on average 60 flops per byte sent by MPI
- 8 barriers per timestep

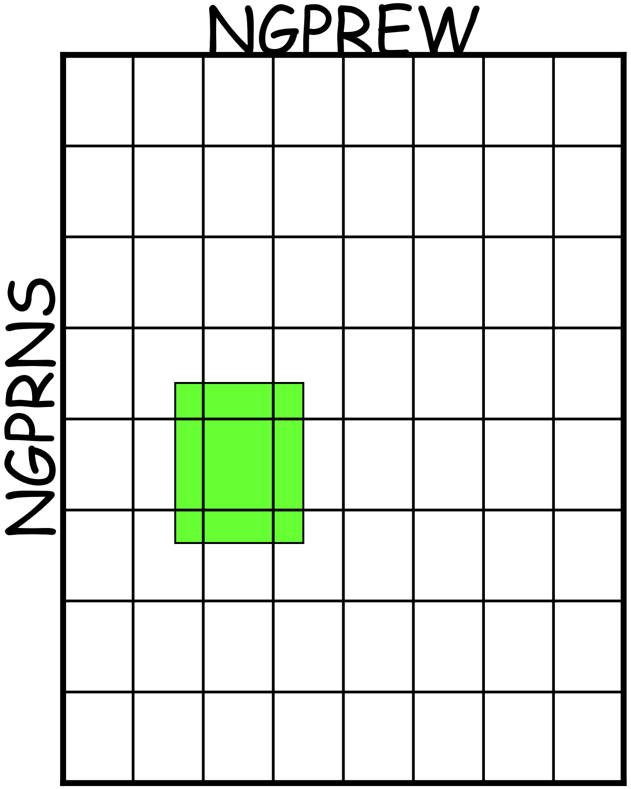
2D decomposition and Transpositions

G Grid-point space
- decompose in horizontal*
- all vertical points on same PE

L Fourier Transform
- decompose in vertical and NS
- all points in EW on same PE

M Legendre Transform
- decompose in vertical and L
- all points in NS on same PE

S Semi-Implicit
- decompose in M and L
- all points in vertical on same PE



*LSPLIT to have same number of points on each PE

Example 1: Transposition routine from IFS

!$OMP DO PARALLEL

DO loop - pack send buffer

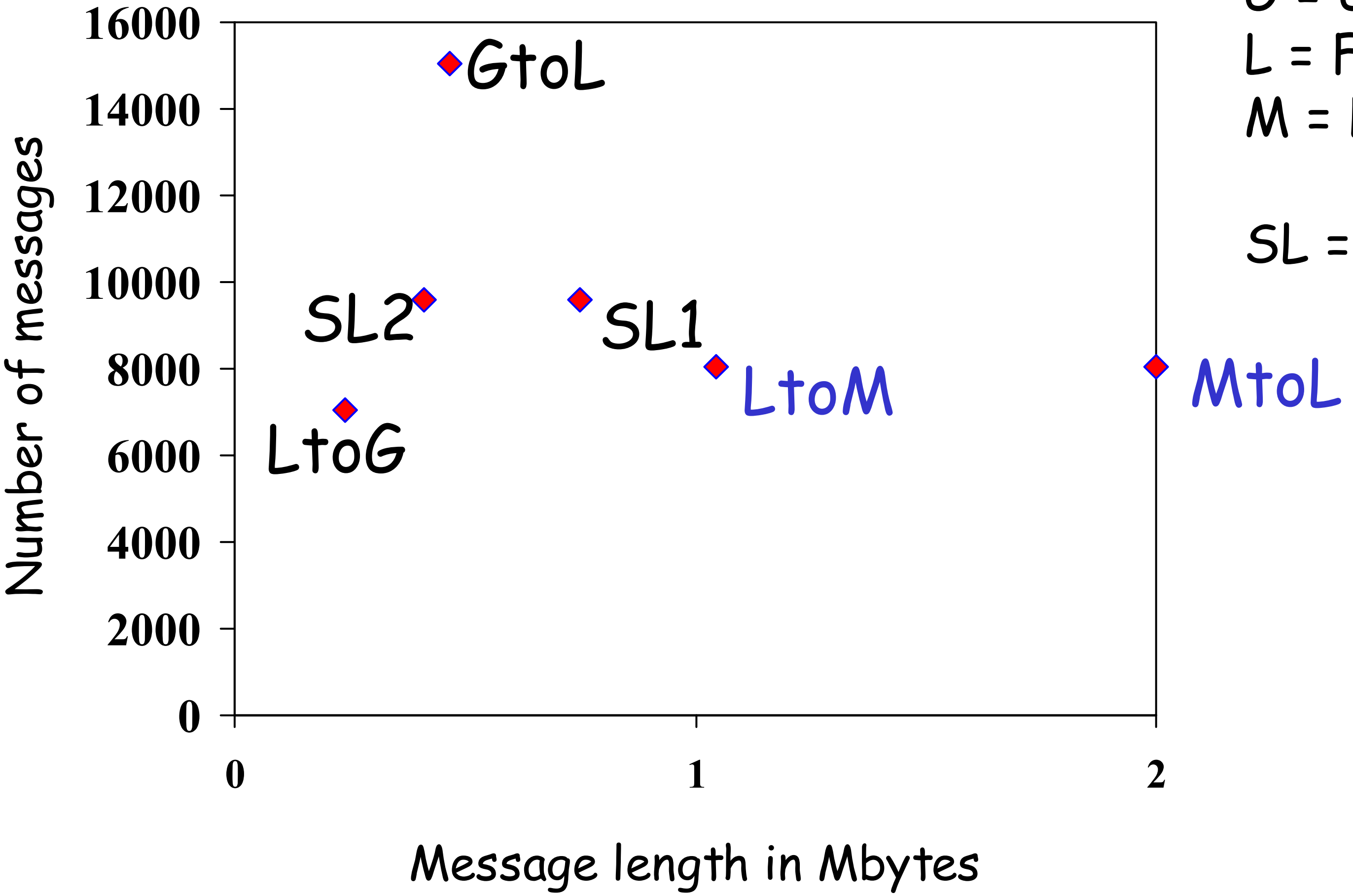
MPI_ISEND	→ unbuffered/non-blocking send
MPI_RECV	→ blocking receive
MPI_WAIT	→ wait for all sends to complete

!$OMP DO PARALLEL

DO loop - unpack receive buffer

*Note no overlap between CPU and Comms

IFS T511 10 day forecast - message lengths and number of messages



G = Gridpoint
L = Fourier
M = Legendre

SL = Semi-Lagrangian

'Off-node' or 'On-node' communcations

-MPI comms between PEs on same shared memory node

→use shared memory copies

→Good for nearest neighbour comms

-MPI comms between PEs on different nodes

→use interconnect

→OK for long messages

-MPI barrier

→Can use 2-stage barrier

→Barrier intra-node and then barrier inter-node

Example 2: Boundary Swap from UM

DO loop - pack send buffer (halo for one field for all levels)

MPI_IRECV → non-blocking receives

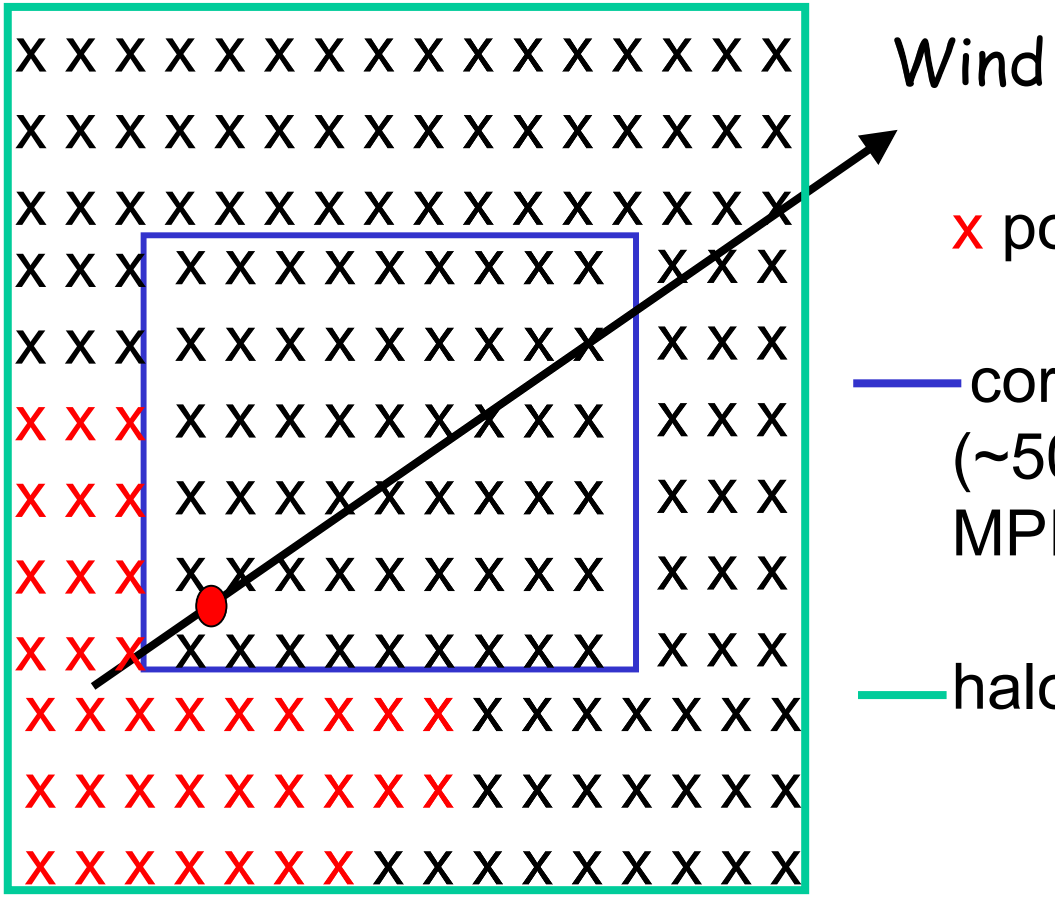
MPI_ISEND → unbuffered/non-blocking send

MPI_WAIT → wait for all sends and receives to complete

DO loop - unpack receive buffer

-> Message length for 1xn decomposition = 432x5x38x8 = 656kbytes

IFS - Semi-Lagrangian 'On Demand'



x points needed in halo

core points on processor (~5000 for T511 with 64 MPI Tasks)

halo width (= 8 for T511)

Semi-Lagrangian Communications

'On-demand' schemes

IFS – send full halo to surrounding PEs for U,V,W (SL1)
- calculate departure points
- determine halo points needed for interpolation
- send list of halo points to surrounding PE's
- surrounding PE's send points requested (SL2)
- EW and NS

UM – calculate departure points
- send list of departure points to surrounding PE's
- surrounding PE's do interpolation
- send back interpolated quantities
- E/W only

OpenMP parallelisation

- Shared memory nodes

- Inside MPI parallelisation
- High level and Loop level

```
!$OMP DO PARALLEL
do I=1,Kblks
   call ec_phys
enddo
```

- Advantages ……
  - Lower MPI overheads
  - Memory saving
  - Frees up processors for OS functions

- But ……
  - More code maintenance -> bugs can lurk unknown!
  - Need to be thread-safe
  - Whole code needs to be done (but not comms)

Different mixed OpenMP/MPI schemes

Master Only

- all communications done by Master Thread
- other threads idle during communications

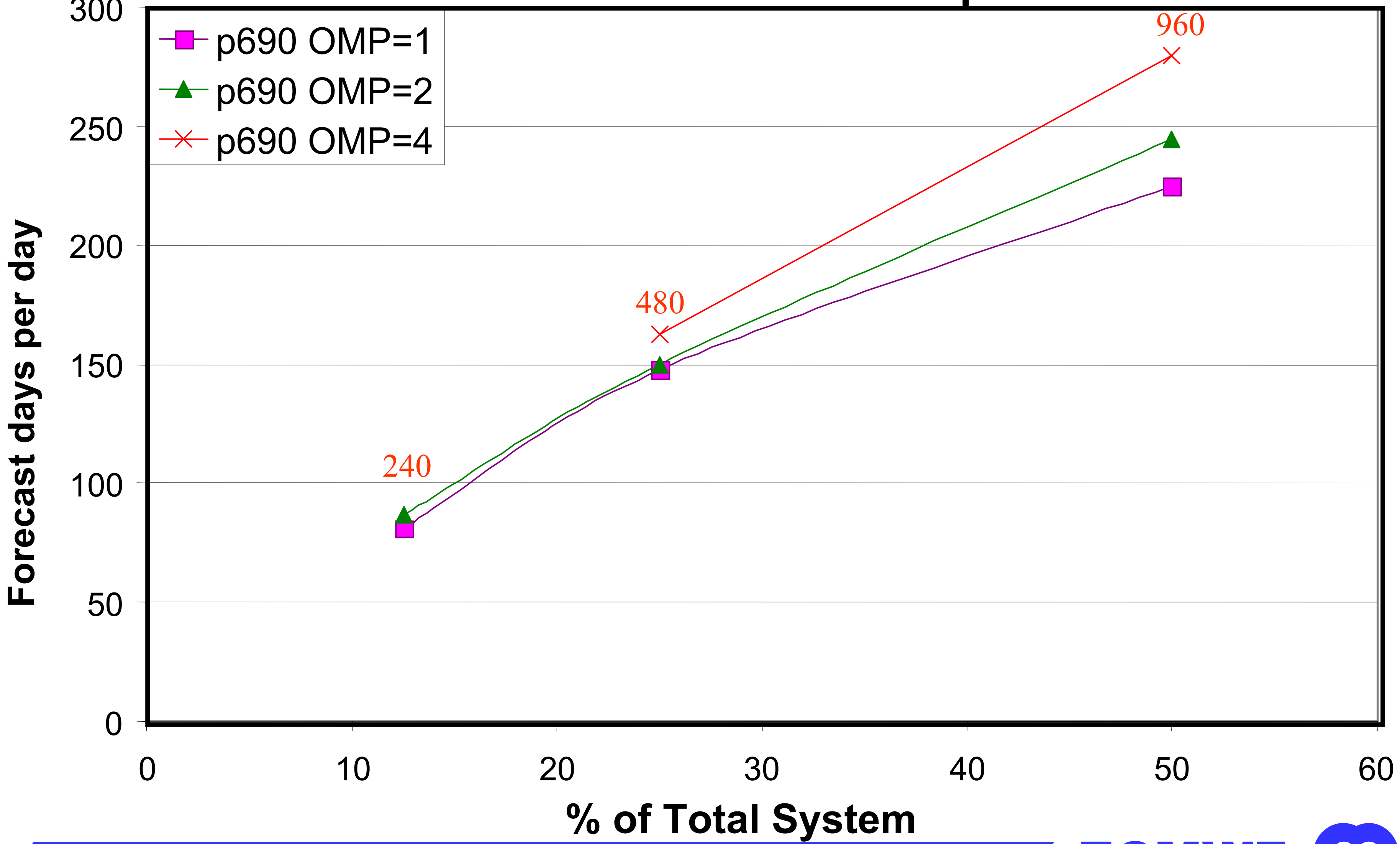
Funnelled

- all communications is done by one OpenMP thread
- other threads can use CPU during communications

Multiple

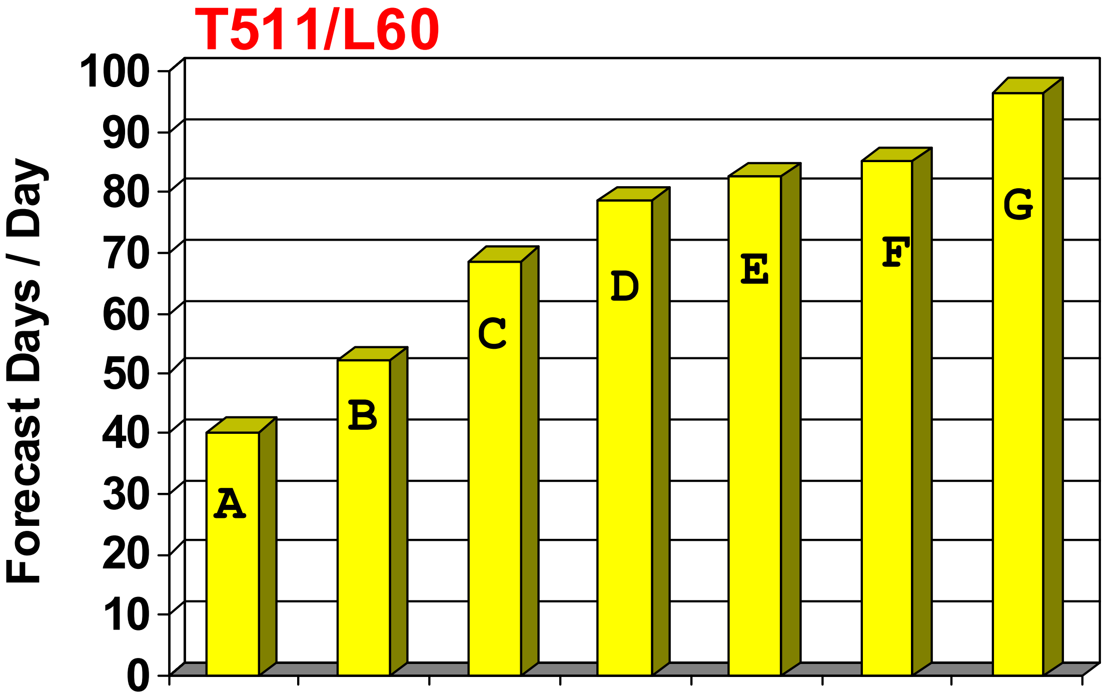
- several threads - maybe all communicate

RAPS-6 : IFS benchmark on hpca : T799 L90 run on different numbers of OpenMP threads



ECMWF

Effects of various Optimisations on IFS Performance



A: Base 128 CPUs

B: NPROMA 1024 ->24

C: SL on demand

D: 1D -> 2D Decomposition

E: Comms not overlapped

F: Vector Mass Functions

G: OpenMP 1-> 2 Threads

IFS - Different Sections

Dynamics	– NPROMA loops – ‘stride 1’ addressing & ARRAY syntax
Semi-Lagrangian	– indirect addressing - wide halo comms
Physics	- NPROMA loops - IF tests
Radiation	- ‘scalar’ loops - interpolate & run at lower resolution - called every hour
Transforms	- long messages - copy loops for buffer packing - SGEMMX for Legendre Transform

Dr. Hook for T511 forecast

```
% Time      Self        Total       # of calls MFlops   Div-%  Routine
 (self)     (sec)        (sec)

  4.77     13.334      13.352        5809       703      3.6   *CLOUDSC@3

  2.94      8.206       8.222        3486      2879      0.0   *MXMAOP@1

  2.83      7.892       8.425          51         0      0.1    TRGTOL@1

  2.30      6.428       6.479       27720       946      0.0   *LAITQM@1

  2.12      5.908       5.959       27756      1201      0.6   *LARCHE@1

  2.07      5.782       5.906       67662      2381      0.0   *VERINT@1

  2.07      5.777      13.502       11558       122      3.5   *CUASCN@4
```

UM – Different Sections

Dynamics – 2D Arrays –> sensitive to decomposition in EW
- 432 x 325 x 38
- finite differences
- Halo + Boundary swap

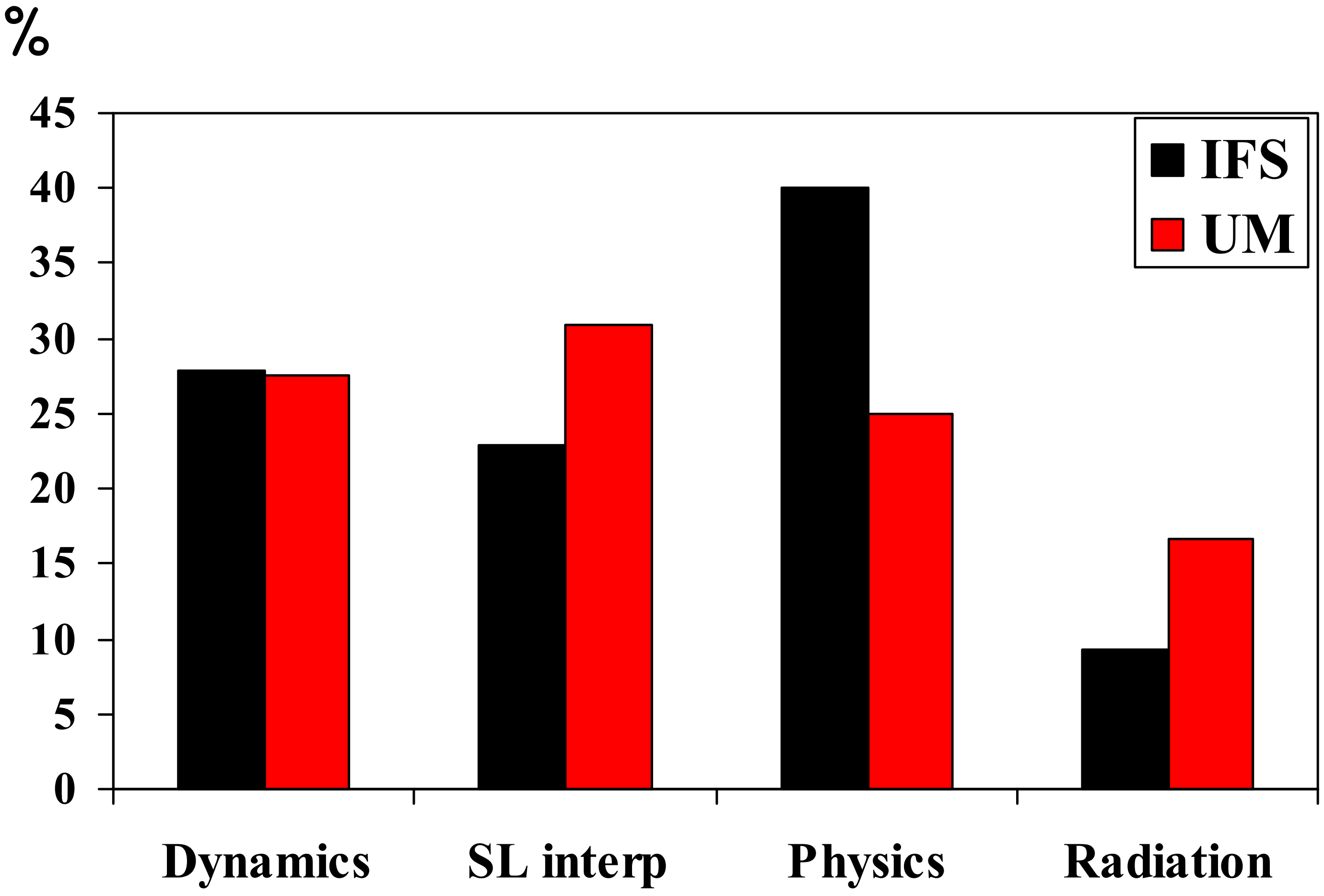
Semi-Lagrangian – on demand communications in EW only
- halo width 5 in NS and 4 in EW

Physics - compress to active points -> copy loops
- 'segments' for convection

Radiation - every other point and called every 3 hours
- SW compressed to daylight points -> load imbalance
- 'segments' for radiation

Semi-Implicit - 3D Helmholtz solver -> Global communications
- ADI preconditioner

Profiles for IFS and UM



*IFS run on 8 nodes (256 CPU's) of IBM p690+

*UM run on 2 nodes (14 CPU's) of NEC SX6

Optimisation and Debugging

-Profiles :

High level ->GSTATS (IFS) & TIMER (UM)
Subroutine level ->DrHook
->machine specific prof

-Hardware performance monitor -> Gflops

-Traceback

-> DrHook
-> machine specific - can give line number

What is Dr.Hook ?

- A Fortran & C-callable instrumentation library to
  - Trap run-time problems
  - Gather profile info per subroutine
    - Wall-clock or CPU-times or Memory used*
    - Mflop/s & MIPS -rates

- The basic feature: keep track of the calling tree
  - For every MPI-task and OpenMP-thread
  - Upon error (when caught via Unix-signals) tries to print the current active calling tree
  - System's own traceback can also be printed

- Portable with low overhead (~1%)

How to instrument a Fortran90 program with Dr.Hook?

```
  SUBROUTINE SUB
  USE YOMHOOK, ONLY : LHOOK, DR_HOOK
  IMPLICIT NONE

    REAL(8) ZHOOK_HANDLE ! Must be a local (stack) variable

!- The very first statement in the subroutine
    IF (LHOOK) CALL DR_HOOK(‘SUB’,0,ZHOOK_HANDLE)

!--- Body of the routine goes here ---

!- Just before RETURNing from the subroutine
    IF (LHOOK) CALL DR_HOOK(‘SUB’,1,ZHOOK_HANDLE)

    END SUBROUTINE SUB
```

Dr. Hook Traceback

```
0:  15:57:40 STEP  936 H= 234:00 +CPU= 41.379
 13:[myproc#14,tid#4,pid#55924]: Received signal#24 (SIGXCPU) ; Memory: 2019178K (heap)
 13:[myproc#14,tid#1,pid#55924]:  MASTER ,#1,st=1,wall=0.000s/0.000s
 13:[myproc#14,tid#1,pid#55924]:   CNT0 ,#1,st=1,wall=0.000s/0.000s
 13:[myproc#14,tid#1,pid#55924]:    CNT1 ,#1,st=1,wall=0.000s/0.000s
 13:[myproc#14,tid#1,pid#55924]:     CNT2 ,#1,st=1,wall=0.000s/0.000s
 13:[myproc#14,tid#1,pid#55924]:      CNT3 ,#1,st=1,wall=0.000s/0.000s
 13:[myproc#14,tid#1,pid#55924]:       CNT4 ,#1,st=1,wall=0.000s/0.000s
 13:[myproc#14,tid#1,pid#55924]:        STEPO ,#978,st=1,wall=10531.259s/0.000s
 13:[myproc#14,tid#1,pid#55924]:         SCAN2H ,#1018,st=1,wall=8913.967s/0.043s
 13:[myproc#14,tid#1,pid#55924]:          SCAN2MDM ,#1018,st=1,wall=8913.896s/32.036s
 13:[myproc#14,tid#1,pid#55924]:           GP_MODEL ,#938,st=1,wall=8845.641s/4.830s
 13:[myproc#14,tid#1,pid#55924]:            EC_PHYS ,#213893,st=1,wall=6144.597s/22.378s
 13:[myproc#14,tid#1,pid#55924]:             CALLPAR ,#213893,st=1,wall=5856.788s/88.130s
 13:[myproc#14,tid#1,pid#55924]:              SLTEND ,#213893,st=1,wall=662.390s/179.559s
 13:[myproc#14,tid#1,pid#55924]:               CUADJTQ ,#117188599,st=1,wall=1992.364s/477.382s

 13:  Signal received: SIGXCPU - CPU time limit exceeded
 13:
 13:  Traceback:
 13:    Location 0x0000377c
 13:    Offset 0x0000009c in procedure _event_sleep
 13:    Offset 0x00000318 in procedure sigwait
 13:    Offset 0x000006c8 in procedure pm_async_thread
 13:    Offset 0x000000a4 in procedure _pthread_body
 13:    --- End of call chain ---
```

Dr. Hook for T511 forecast - hpca

```
 #   % Time      Cumul        Self       Total  # of calls   MIPS  MFlops   Div-%    Routine@<tid>
     (self)      (sec)        (sec)       (sec)                                        [Cluster:(id,size)]

 1     7.43      35.027      35.027      40.573          49     961     273     2.9    WVCOUPLE@1 [567,1]
 2     3.67      52.349      17.322      17.367        5824    1113     546     3.6   *CLOUDSC@1 [5,4]
 3     3.65      52.349      17.204      17.287        5791    1116     548     3.6    CLOUDSC@4 [5,4]
 4     3.64      52.349      17.181      17.289        5769    1118     549     3.6    CLOUDSC@2 [5,4]
 5     3.63      52.349      17.138      17.202        5770    1117     549     3.6    CLOUDSC@3 [5,4]
 6     3.51      68.918      16.569      16.584          54     783       0    27.6    TRMTOL_COMMS@1 [525,1]
 7     2.76      81.935      13.017      18.260          51     926       1     2.8    TRGTOL@1 [520,1]
 8     2.51      93.763      11.829      11.831          54     742       0    24.8    TRLTOG_COMMS@1 [523,1]
 9     2.41     105.145      11.382      30.536       11540    1106      88     3.4   *CUASCN@3 [30,4]
10     2.40     105.145      11.336      30.436       11538    1112      88     3.4    CUASCN@2 [30,4]
11     2.39     105.145      11.274      30.394       11582    1110      88     3.4    CUASCN@4 [30,4]
12     2.39     105.145      11.267      30.072       11648    1113      86     3.4    CUASCN@1 [30,4]
13     2.36     116.296      11.150      11.185        3492    2135    2172     0.0   *MXMAOP@1 [166,4]
14     2.31     116.296      10.897      10.940        3502    2218    2259     0.0    MXMAOP@2 [166,4]
15     2.30     116.296      10.832      10.920        3474    2216    2258     0.0    MXMAOP@4 [166,4]
16     2.29     116.296      10.816      10.910        3484    2224    2266     0.0    MXMAOP@3 [166,4]
17     1.94     125.448       9.152       9.327       27785    1433     682     0.0   *LAITQM@3 [138,4]
18     1.94     125.448       9.130       9.263       27980    1434     679     0.0    LAITQM@1 [138,4]
19     1.92     125.448       9.073       9.256       27715    1432     682     0.0    LAITQM@4 [138,4]
20     1.92     125.448       9.045       9.220       27750    1440     686     0.0    LAITQM@2 [138,4]
21     1.85     134.173       8.725       8.785        5563     985     592     2.2   *SLTEND@4 [297,4]
22     1.85     134.173       8.724       8.777        5596     987     593     2.2    SLTEND@1 [297,4]
23     1.83     134.173       8.654       8.741        5541     986     593     2.2    SLTEND@2 [297,4]
24     1.83     134.173       8.621       8.658        5546     989     595     2.2    SLTEND@3 [297,4]
25     1.82     142.737       8.565       8.580          51     782       0    21.6    TRLTOM_COMMS@1 [524,1]
26     1.80     151.219       8.482      69.102          13     581      22    10.6    RADINTG@1 [207,1]
```

What the compiler does for you?

-Does most basic tuning

-Depends on optimisation level

  -ECMWF use '–O3 –qstrict' on IBM to get bit reproducible results

-Loops

  -Unrolling, Splitting, Fusion, Invert

  -Move invariants outside

  -Schedule instructions to match number of registers and pipes

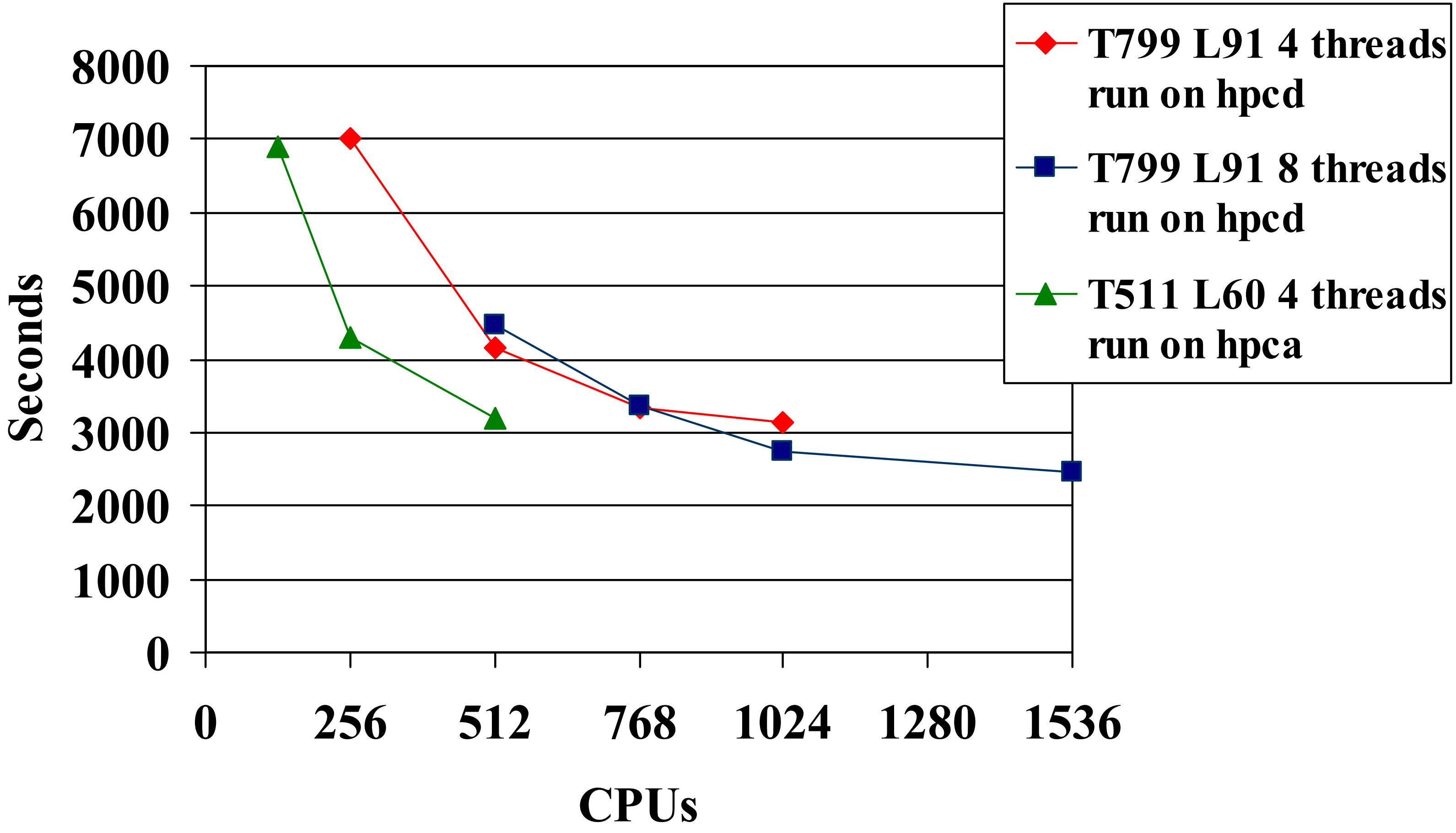
-Memory

  -Compiler cannot know whether data is in cache or in memory

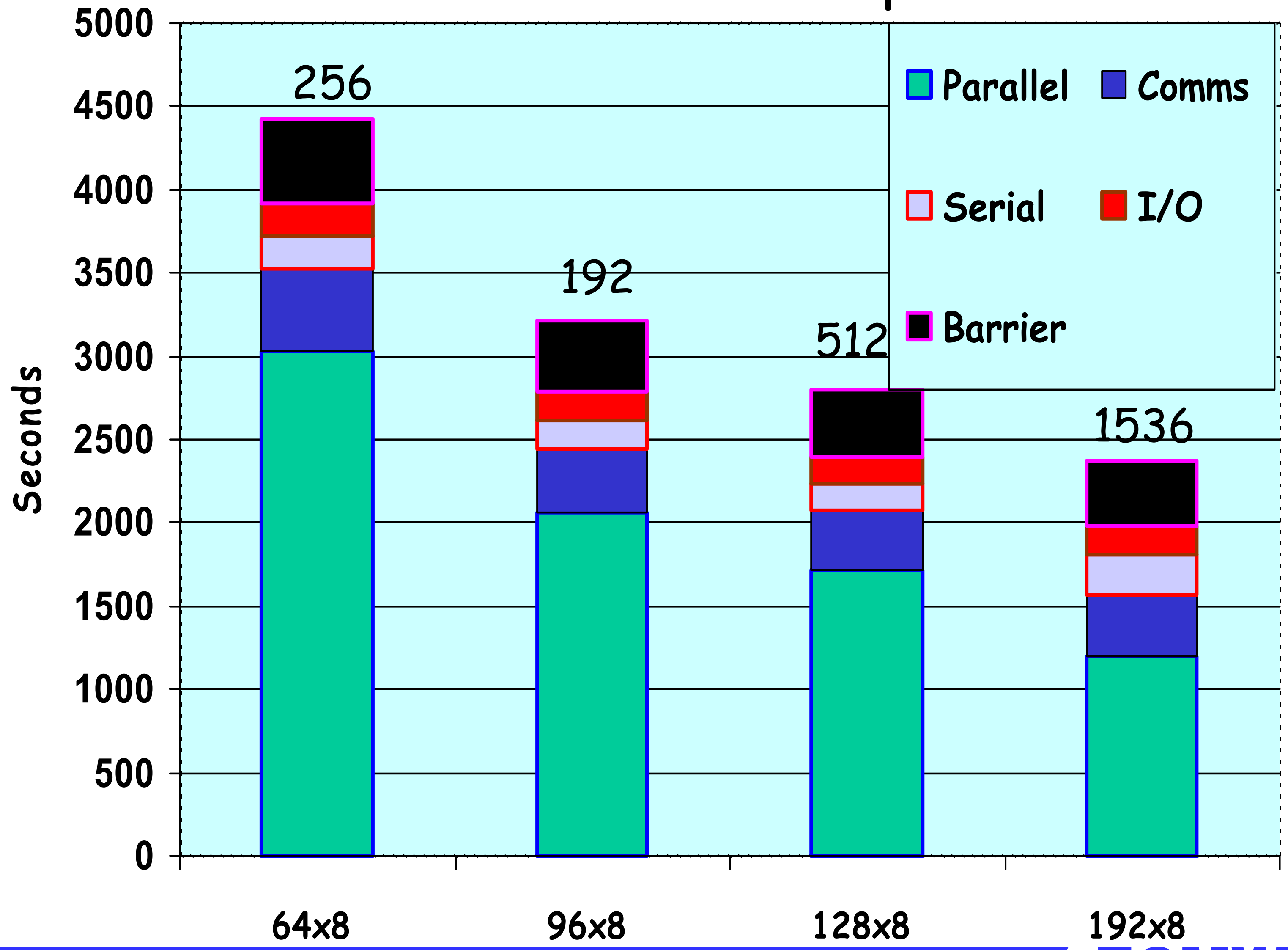
Top Optimisation activities for IFS

1. Add timings
2. Improve MPI comms (not buffered and no overlap)
3. Add more OpenMP parallel regions
4. Divides (-qstrict)
5. Use vector functions and machine specific libraries
6. Remove copies and zeroing of arrays
7. Optimise data access
8. Remove low level allocates

Scalability of IFS - 4D-Var



4D-Var -CY28R3 T799/T255 L91
Total Times on hpcd



T799 L91/T95/T255: 4D-Var run on hpcd

MPIx OpenMP	Traj0	Min0 T95	Traj1	Min1 T255	Traj2	TOTAL (% peak)
64 x 4	682	488	588	3317	872	5950 (7.9%)
128 x 4	446	361	332	1829	576	3547 (6.7%)
96 x 8	342	307	257	1343	487	2738 (5.7%)
128 x 8	301	301	235	1059	427	2359 (5.0%)

References

David Dent
 'The influence of Computer Architectures on Algorithms' - ECMWF Numerical methods seminar 1998

Paul Burton and Bob Carruthers
 'Porting the UK Met Office's Unified Model to the CRAY X-1' - CUG proceedings 2004

Thanks to

Bob Carruthers CRAY

John Hague IBM

Alan Dickinson Met Office
Paul Selwood Met Office

ECMWF Colleagues ……………………