Computer Architectures and Aspects of NWP models

Deborah Salmond

ECMWF
Shinfield Park, Reading, UK

Abstract: This paper describes the supercomputers used in operational NWP together with the computer aspects of NWP models. The IFS from ECMWF and the UM from the UK Met Office are used to illustrate these aspects being typical examples of a spectral model and a grid-point model. The use of message passing and OpenMP to obtain highly scalable parallelised NWP models is shown. Finally code optimization is briefly described.

1. Introduction

This paper describes aspects of high-performance computer architectures that are relevant to Numerical Weather Prediction (NWP) model performance.

Over the years high-performance computers have been developed based on either scalar or vector processors[1]. Each has its own particular issues that need to be considered when writing optimized code. For vector processors it is most important to vectorise as much as possible of the code and this can give performance improvements of the order of ten times. Scalar processors have an intermediate cache between the processors and the memory and it is important to consider locality of data and data access and reuse.

High performance computers now have a large number of processors and the NWP model code has to be parallelised in a scalable way to take advantage of this. Initially computers with multiple processors were made with all processors connected with equal access to a single shared memory, for example the CRAY C90. This was followed by systems with multiple processors each with its own memory and connected together by a high speed interconnect. For these distributed memory systems all communications between processors had to be done by message passing, examples of these are the Fujitsu VPP5000 and the CRAY T3E. More recently there is a combination of both memory architectures where a system consists of a cluster of nodes that are connected by a high performance switch, where each node is a group of processors connected to a shared memory. Examples of this are the IBM p690, the NEC SX-6, and the CRAY X1. Parallelisation on these shared memory clusters can be a combination of message passing and shared memory parallelism.

Over the years NWP models have been adapted to suit the available computers. The UK Met Office model was initially coded in assembler in the 1970s on an IBM 360/195 with inner loops over vertical levels. After this the loops were reordered to suit the long vector requirements of the CDC Cyber 205, putting the vertical levels loop on the outside and merging the inner horizontal loops. More recently the Unified Model (UM) has been coded with the horizontal loops separated to improve the readability of the code and make it look more like the finite difference equations.

The ECMWF Integrated Forecasting System (IFS) has been coded with the inner loops over the horizontal dimensions, but in groups of NPROMA points. This was done to give a long vector capability with latitude lines merged. However, the 'NPROMA' coding style can also be used to give short loops, giving a smaller working set of data suitable for a scalar architecture with cache.

In this paper some of the computer related aspects of the IFS from ECMWF are compared with the UM from the UK Met Office.

The IFS is a hydrostatic spectral model, currently run at about 40 km horizontal resolution to 10 days on 256 processors of a scalar computer (IBM p690) in 1 hour at 88 Gflops. The resolution is T511 L60, which is a spectral truncation of 511 and 60 vertical levels. The total number of floating point operations for the run is 315 Tflop.

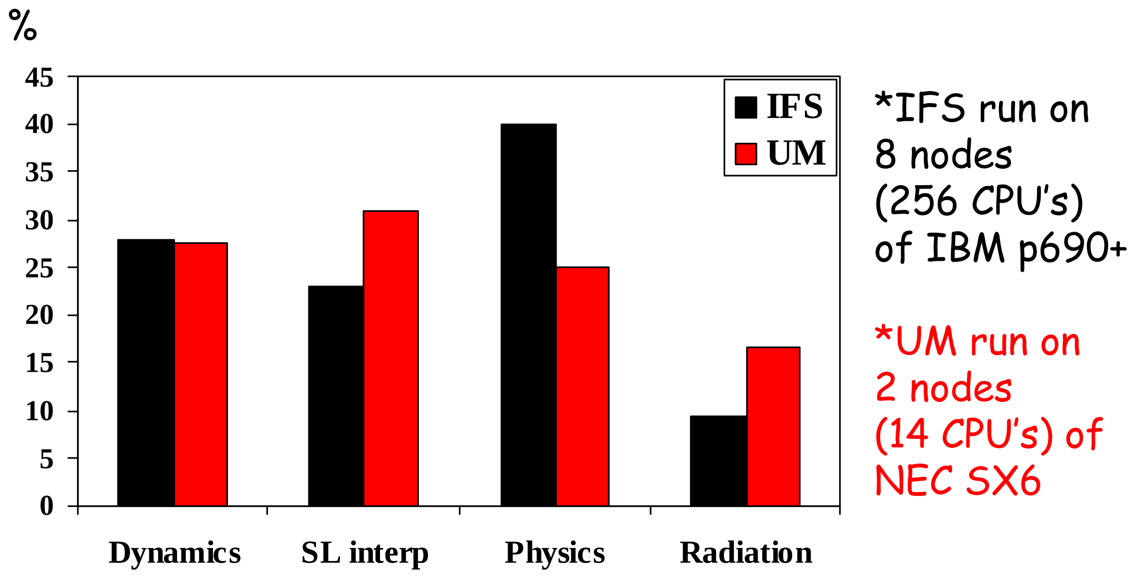


Figure 1 Profiles of IFS and UM

The UM is a non-hydrostatic grid-point model run at about 60 km horizontal resolution to 7 days on 14 processors of a vector computer (NEC SX-6) in 37 minutes at 16 Gflops. The resolution is 432 x 325 in latitude and longitude and 38 vertical levels. The total number of floating point operations for the run is 35 Tflop. Figure 1 shows a comparison of the CPU profiles of IFS and UM. (Note that in this profile the IFS 'Dynamics' includes the spectral transforms and associated communications and the UM 'Dynamics' includes the 3-D solver.)

2. Supercomputers for NWP

Figure 2 shows some examples of the high-performance computers used for operational weather prediction.

ECMWF has 2 IBM p690 clusters known as hpca and hpcb. These are configured with 120 nodes, each node having 8 IBM Power4 processors with a peak performance of 5.2 Gflops per processor. During the second half of 2004 ECMWF are replacing hpca and hpcb with 2 IBM p690+ clusters to be known as hpcc and hpcd. There are configured with 68 nodes, each node having 32 IBM Power4+ processors, with a peak performance of 7.6 Gflops per processor. The 'peak performance' in flops is the theoretical maximum number of floating-point operations per second that can be achieved on the system.

The UK Met Office has 2 NEC SX-8 clusters. These are configured with 15 nodes, each node having 8 processors with a peak performance of 8 Gflops per processor.

Météo France has a Fujitsu VPP5000 split into two systems with a total of 124 processors with a peak of 9.6 Gflops per processor.

INM in Spain have a CRAY X1 system configured with application 15 nodes. Each node comprises 4 MSPs connected to a shared memory. Each MSP is made up of 4 SSPs (with peak of 3.2 Gflops per SSP). The CRAY X1 can be run in 'SSP mode' where each SSP is considered as a separate processor or in 'MSP mode' where the compiler distributes the work between SSPs.

			Number of procs	Peak Gflops per proc	Number of procs per node
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	1
INM	CRAY X1	vector	60 MSPs 240 SSPs	12.8 3.2	4
DWD	IBM NH-2	scalar	1920	1.5	16

Figure 2 Examples of supercomputers used for operational NWP

DWD in Germany have an IBM Nighthawk-2 system with 120 nodes, each node having 16 Power3 processors with a peak of 1.5 Gflops per processor.

In Figure 3 the previous supercomputers at these sites are listed. In the case of ECMWF and the UK Met Office both sites have recently switched between vector and scalar processors: ECMWF from vector to scalar and UK Met Office from scalar to vector. This shows the importance of having and retaining an NWP model and analysis code, which performs well on both vector and scalar processors. The style of programming suitable for the machine on which the code is first parallelised often remains with the code on subsequent systems.

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

Figure 3 Previous computers used for operational NWP

Figure 4 shows the history of computers used for operational NWP at the UK Met Office in terms of peak performance. The first computer, LEO made by J. Lyons & Co. was used in 1954 to run a grid-point model with a 14x10 grid that took 4 hours to run a 24-hour forecast[2].

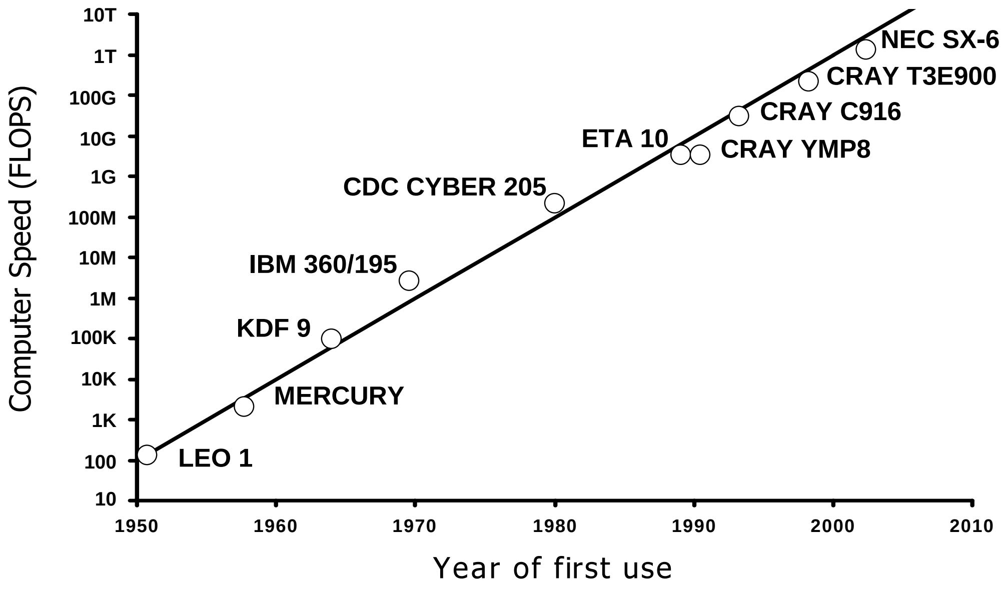


Figure 4 Supercomputers used for operational NWP by the UK Met Office in terms of Peak performance

Another measure of computer performance is the 'sustained performance' or the performance that is achieved by the NWP model. Figure 5 shows the sustained performance of the IFS code from ECMWF on the most recent supercomputers. The Fujitsu VPP5000 installed in 1999, the IBM p690 installed in 2002, and the IBM p690+ installed in 2004.

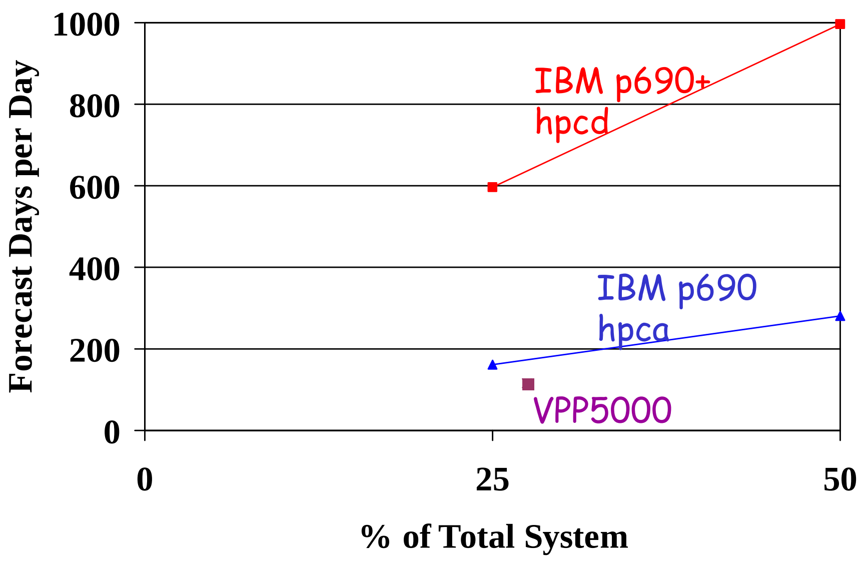


Figure 5 Recent supercomputers used for operational NWP by ECMWF in terms of sustained performance on IFS.

The IFS and several other codes are part of a RAPS (Real Applications for Parallel Systems) benchmark suite. The runs of IFS in Figure 5 are from the RAPS-6 benchmark that was used in the procurement of the IBM systems in 2001. The resolution was T799 L90.

3. IFS Overview

The IFS is a spectral model with semi-Lagrangian advection. It has been parallelised for distributed memory using message passing (MPI) and for shared memory using parallelisation directives (OpenMP). The physics and dynamics are computed in blocks of NPROMA length. The model has been coded so that bit reproducible results are obtained with different values of NPROMA and different numbers of MPI tasks and OpenMP threads.

The performance of the IFS model is shown in Figure 6. Here runs of IFS in 1998 on the CRAY T3E and in 2004 on the IBM p690+ are compared. The CRAY T3E run was at the operational resolution of the time - T213 L31 and reached nearly 100 Gflops on 1024 processors. The IBM p690+ is run at the next planned operational resolution T799 L91 and just reaches 1 Tflops on 2048 processors.

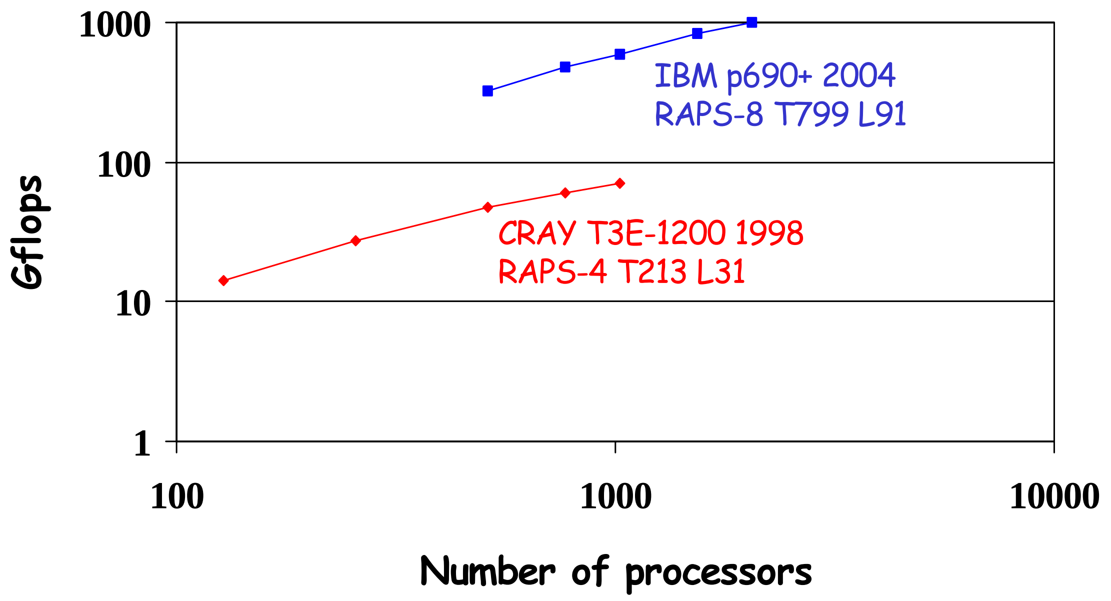


Figure 6 Performance of IFS forecast 1998 - 2004

The NPROMA blocking in the dynamics and physics is possible when there are no dependencies in the horizontal. NPROMA is chosen to be long (>1024) to give long vector lengths for vector processors and short (~24) to reduce the working size of the data in the cache for scalar processors. Small NPROMA also gives memory savings as all work arrays are dimensioned on NPROMA. The NPROMA blocking loops also give a convenient high-level position for the OpenMP parallelisation directives. The structure of the code is shown in Figure 7 for the blocking loop for the physics routines. In Figure 8 performance at different values of NPROMA is shown for vector and scalar processors.

```
!$ DO PARALLEL
DO Iblocks = 1,Nblocks
  CALL EC_PHYS
ENDDO
!$ END DO PARALLEL
```

```
Subroutine EC_PHYS
! call Physics subtoutines
! with many loops of type
  DO Jlev=1,Nlev
        DO I-1,NPROMA
```

Figure 7 NPROMA loop structure with OpenMP directive.

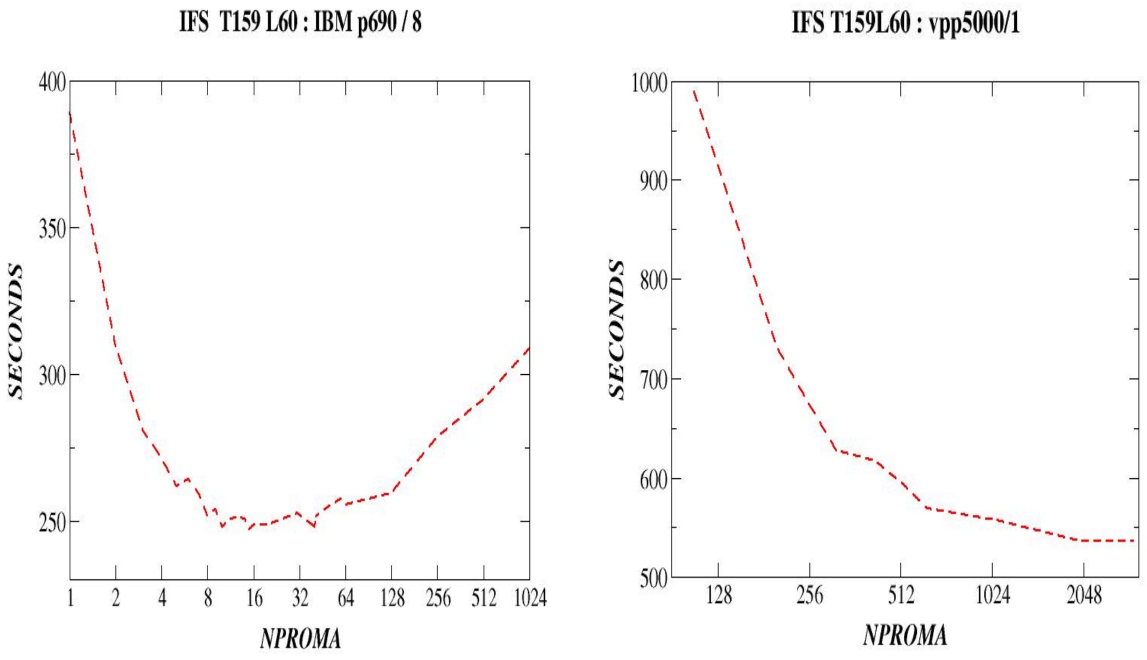


Figure 8 Performance of IFS with different values of NPROMA on scalar and vector processors.

The scalability of the IFS has been much improved between the IBM p690 with the 'colony switch' and the IBM p690+ with the new 'federation switch'. This can be seen in Figure 9 where speed-ups are plotted for a T511 L60 forecast. The 'federation switch' is configured to have a bandwidth of 8 Gbytes/s for 32 CPUs compared with 350 Mbytes/s for 8 CPUs for the 'colony switch'. Figure 10 shows the breakdown of communications and parallel time for these runs. Note that to get accurate measures of communications time, barriers must be inserted before each communications section - the time spent in barriers is due to load imbalance.

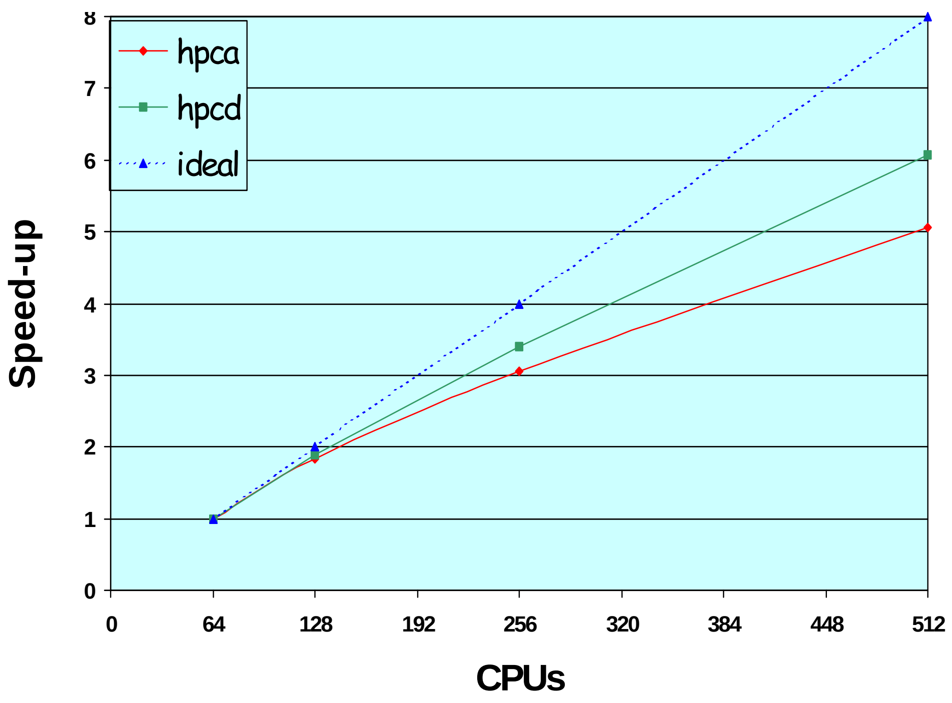


Figure 9 Scalability of T511 L60 1 day forecast on hpca and hpcd run with 4 OpenMP threads

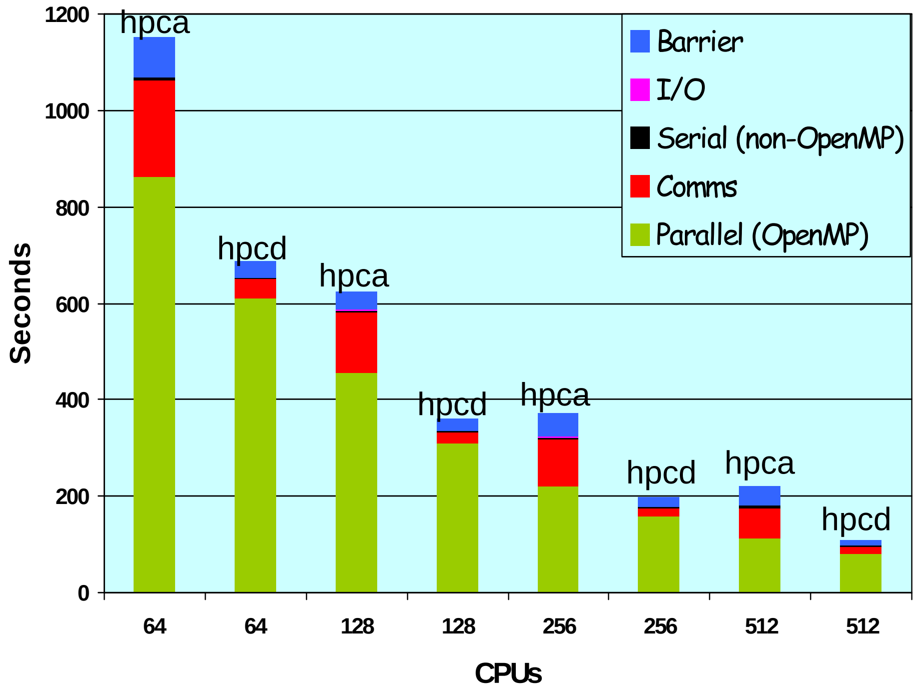
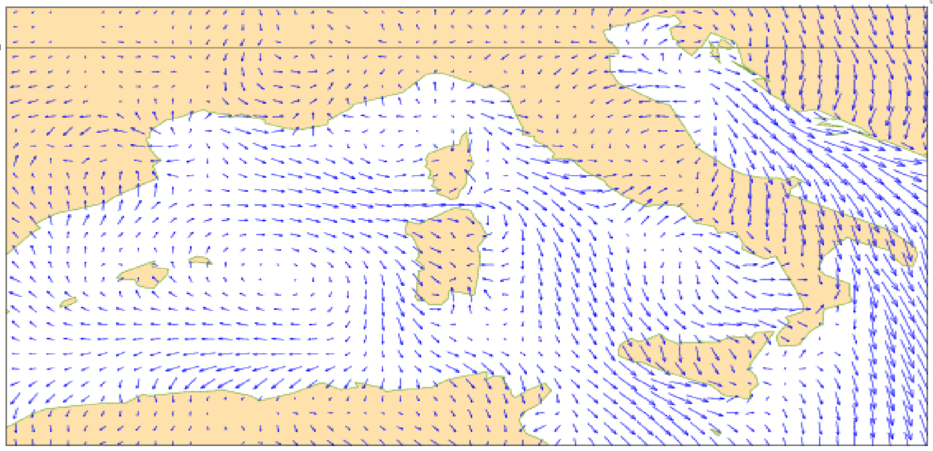


Figure 10 Scalability of T511 L60 1 day forecast on hpca and hpcd.

The next planned resolution for the IFS 10 day high resolution forecast is T799 L91. Figure 11 shows the improvement in meteorological detail of 10 metre winds for a section of the Mediterranean Sea.

14 July 2003 12 UTC ECMWF Forecast T+120 : 19 July 2003 12 UTC Surface 10m winds

T511



T799

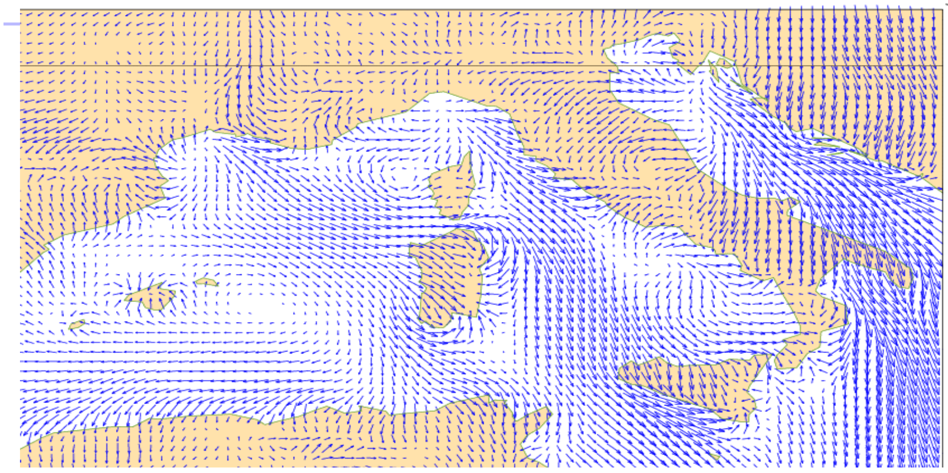


Figure 11 10 metre winds for T511 L60 compared with T799 L91

Figure 12 shows the high level profile for T511 L60 compared with T799 L91. Note that the Legendre and Fourier transforms increase slightly more than the other sections of the model. The extra cost per timestep is 3.5 times and the timestep is reduced from 900 seconds to 720 seconds.

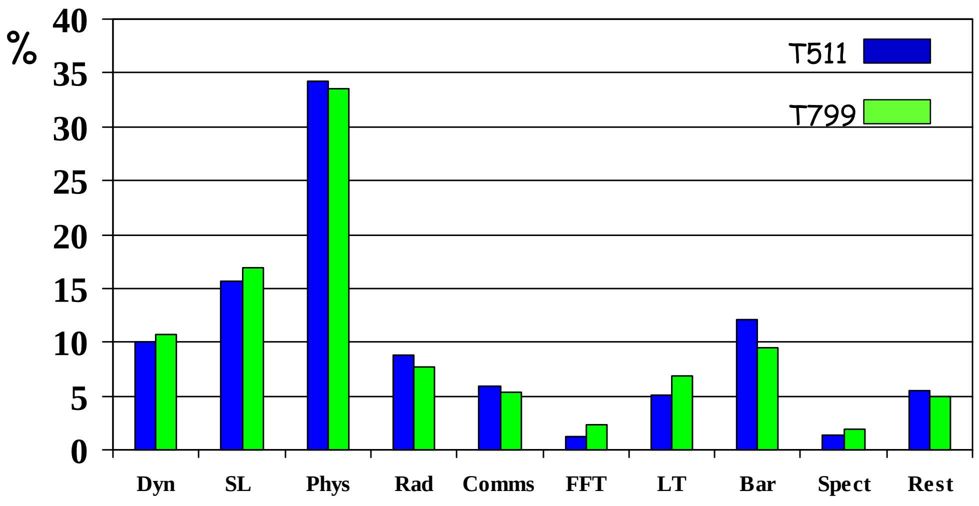


Figure 12 Profile of T511 L60 compared with T799 L91 run on hpcd with 64 MPI tasks and 4 OpenMP threads.

Figure 13 shows the runtimes for T799 L91 run on 256 to 2048 processors, in terms of total time for the run in seconds and Forecast Days/Day (FCD/D). The percentage of peak performance and the percentage of total time spent in communications are given. The total number of floating-point operations for this run is 1630 Tflop.

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%

Figure 13 RAPS-8 T799 L91 10 day forecast run on hpcd with percentage of peak.

An ensemble of 51 lower resolution 10 day forecasts is run operationally at ECMWF. This is planned to be at T399 L62 resolution. Figure 14 shows the timings and percentage of peak for these runs. Also shown for comparison is T399 L62 run on the CRAY X1 at INM. Note that the percentage of peak on the vector machine is higher but the runtime on the same number of CPUs is exactly the same. The total number of floating-point operations for 51 copies of T399 L62 is 4400 Tflop. This is significantly more than the number of operations for the high-resolution 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%

Figure 14 RAPS-8 T399 L62 10 day 'ensemble forecast' run on hpcd with percentage of peak.

4. UM Overview

The UM is a grid-point model with semi-Lagrangian advection. The most time consuming part of the UM is the 3-D Helmholtz solver. The parallelisation is for distributed memory only. This was done with a 2-D decomposition on the CRAY T3E but on the NEC SX-6 a 1-D decomposition is more appropriate to maintain the vector length. However, a 1-D decomposition can only be used for up to 54 processors. The code has many barriers partly because of the more frequent boundary exchanges necessary in a grid-point model and partly because it was originally developed on the CRAY T3E using low latency CRAY-shmem message passing.

The communications are wide-halo boundary swaps with halo of width 5 in North/South direction and 4 in East/West direction. Bit reproducibility is maintained for different numbers of processors for the climate version of the model, but has been relaxed for the forecast version to give better efficiency on the NEC. The grid is a regular latitude/longitude grid with special treatment at the pole - which introduces some load-imbalance.

The UM is run operationally on the NEC SX-6 with 432 x 325 x 38 grid-points to 7 days. This takes 37 minutes on 2 nodes, using only 14 out of 16 processors. The remaining 2 processors remain idle to handle operating-system interrupts. If all 16 processors are used the UM will run slower than on 14 processors. Figure 15 shows the profiles obtained by changing the decomposition from a 1x16 (which has vector length

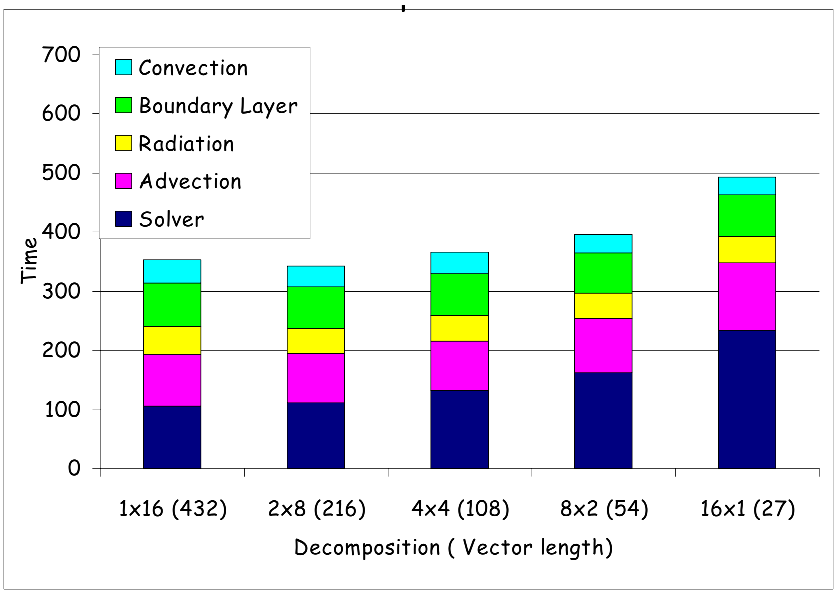


Figure 15 Profiles for the UM on 16 processors showing the effect of changing from 1x N to N x 1 decomposition

of 432) to 16x1 (which has vector length of 27)[3]. The main increase in cost comes from the 3-D solver as the vector length decreases and the complexity of the global communications increases.

The UM also has an NPROMA type structure which is called ‘segmenting’ in the convection and the radiation.

The UM uses a regular lat/lon grid while the IFS uses a reduced grid to give a more homogeneous coverage of the globe. Figures 16 and 17 show the difference in numbers of horizontal grid-points for UM 432 x 325, IFS T511 and IFS T799.

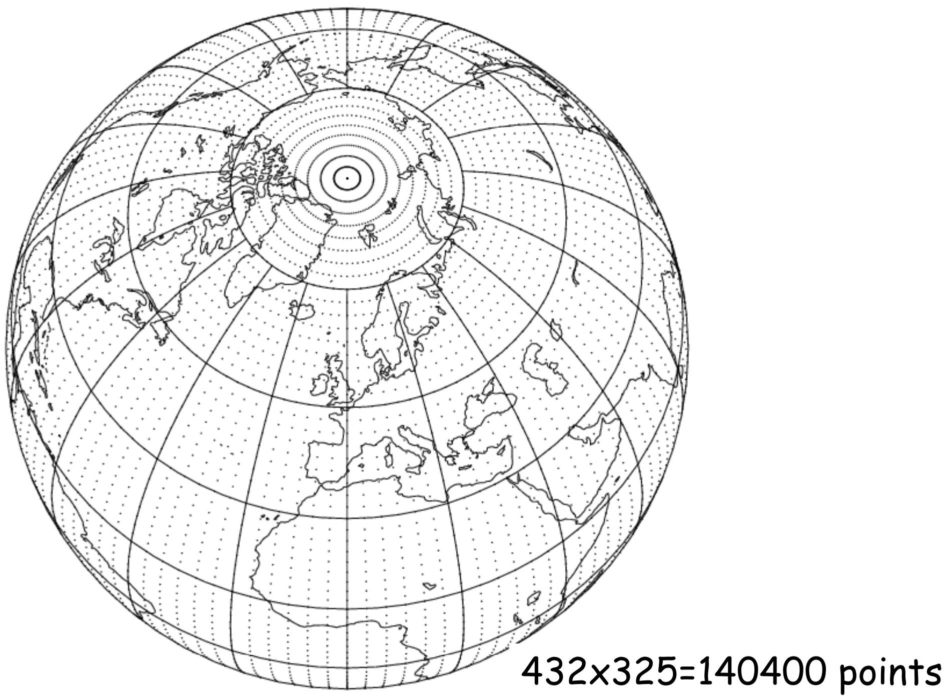


Figure 16 Lat/Lon grid for UM.

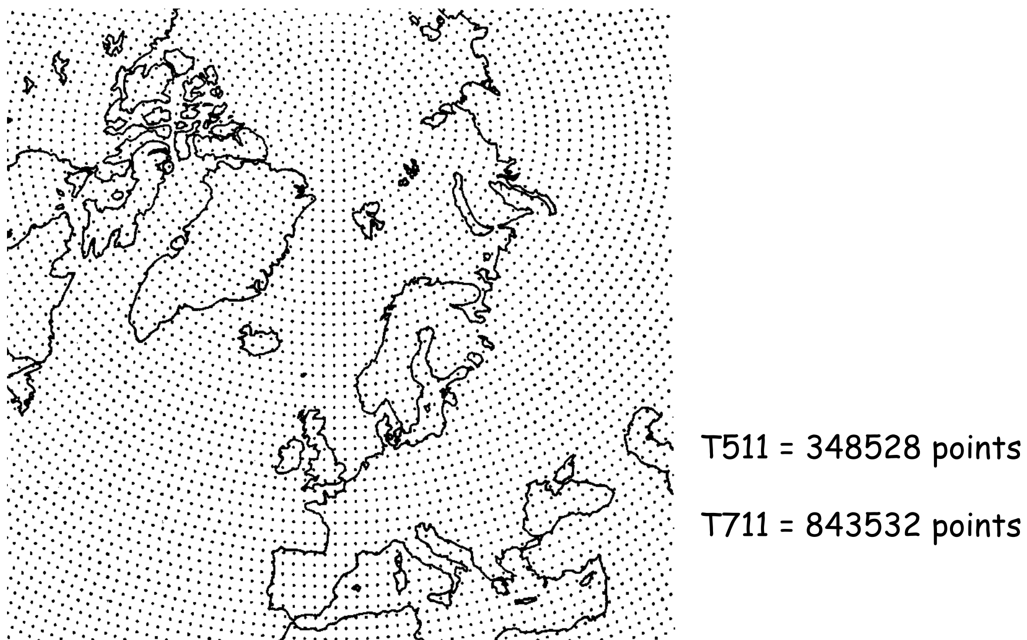


Figure 17 Reduced grid for IFS

5. Message Passing

Parallelisation for distributed memory involves message passing between different tasks. MPI is used for portability. The message passing for a grid-point model such as the UM is for halo exchange that typically has short message lengths. A low latency interconnect will be important for good performance for grid-point models. Message passing for a spectral model such as the IFS is mainly in the transpositions for the spectral transforms, which typically have relatively long messages. A high bandwidth interconnect is important for good performance for spectral models.

An interesting comparison between message passing in spectral and grid-point models comes by comparing the number of barriers per timestep and the number of useful floating-point operations per byte sent by message passing. For the HIRLAM grid-point model benchmark run on the CRAY X1 at INM there were 6440 barriers per timestep and 20 flops/byte. For the IFS benchmark run on the IBM p690+ at ECMWF there are 8 barriers per timestep and 60 flops/byte.

For global and group message gathering and scattering it is often best to use the MPI global and group calls.

The decomposition of the IFS between PEs (Processing Elements) is in 4 states with 3 transpositions between them:

G - Grid-point space

The decomposition is 2-D in the horizontal with all points in a vertical column on the same PE.
The decomposition can be modified such that there is almost an equal number of grid-points on each PE.

The decomposition is in North/South (NS) and East/West (EW)

L - Fourier transform

The data is transposed so that all points in the EW are on the same PE before taking the Fourier Transform.
The decomposition is in vertical and NS

M - Legendre transform

The data is transposed so that all points in the NS are on the same PE before taking the Legendre Transform
The decomposition is in the vertical and the Fourier variable L

S - Semi-Implicit

The data is transposed so that all points in a vertical column are on the same PE.
The decomposition is in the Legendre and Fourier variables M and L

The communications needed for the semi-Lagrangian interpolation are a wide-halo swap between neighboring PEs. This can be expensive and will not be scalable. To improve this, 'On Demand' schemes have been developed so that only the points needed are communicated. This is shown schematically in Figure 18, where once the departure points are known only the red points in the halo need to be communicated.

The IFS semi-Lagrangian communications scheme is in 3 parts

- A full halo is sent for U,V and W so that the departure points can be calculated (SL1)
- From the departure points the halo points needed for the interpolation are determined and a list sent to the surrounding PE's
- Surrounding PEs send the requested points (SL2)

The UM scheme is different in that the surrounding PEs actually do the interpolation and send back interpolated values. Also the UM scheme is in EW direction only whereas the IFS scheme is in both directions.

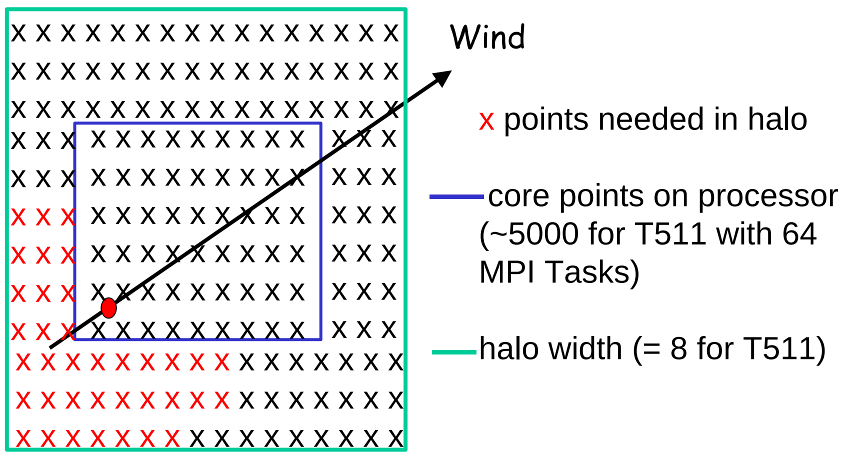

Wind
x points needed in halo
core points on processor (~5000 for T511 with 64 MPI Tasks)
halo width (= 8 for T511)

Figure 18 Semi-Lagrangian 'On Demand' scheme

Figure 19 shows the message lengths and number of messages for an IFS T511 L60 10 day forecast run with 32 MPI tasks. The longest messages are 1 and 2 Mbytes for the 'LtoM' and 'MtoL' transpositions.

For clusters of shared memory nodes the message passing can be either between PEs on the same node or between PEs on different nodes. The message passing between PEs on the same node uses the shared memory and is good for nearest neighbour communications. The message passing between PEs on different nodes uses the interconnect and is best for long messages. In the case of IFS, the LtoM and MtoL communications are mostly 'off node' and the LtoG, GtoL and the semi-Lagrangian communications SL1 and SL2 are mostly 'on node' communications.

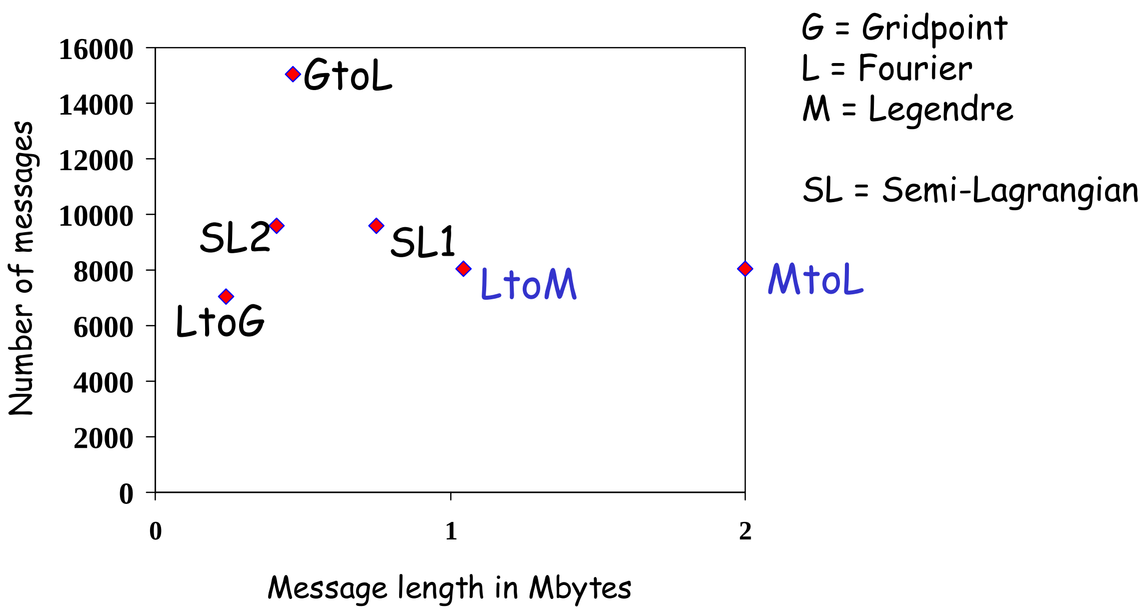
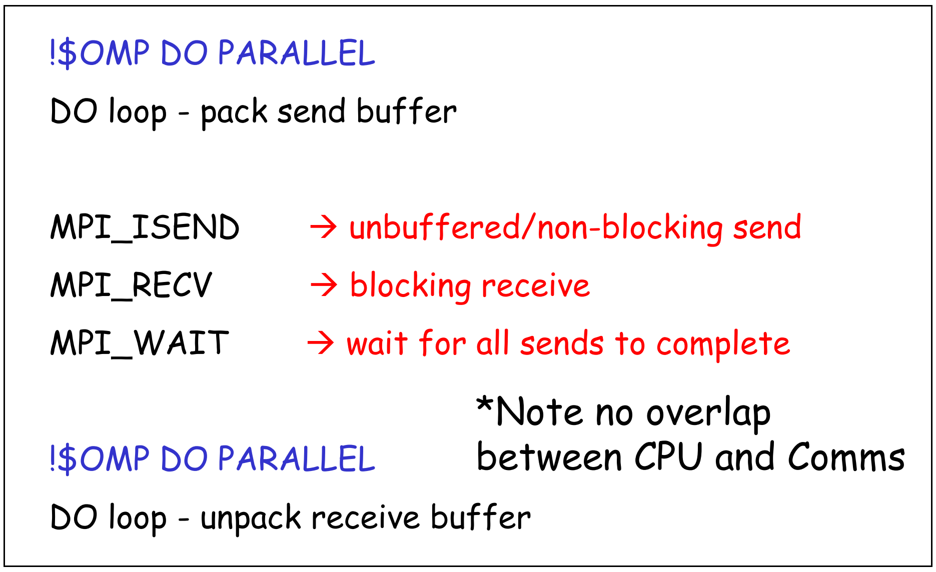


Figure 19 Message lengths and number of messages for IFS T511 10 day forecast

The message passing in the IFS has been written using un-buffered, non-blocking MPI_SENDs. Also it was found that best performance was obtained when there is no overlap between message passing and CPU activity such as buffer packing. An example of the IFS transposition code structure is shown in Figure 20. The messages are packed and unpacked into buffers in loops parallelised with OpenMP. The non-overlapping paradigm has been found to be best on all computers we have tested.



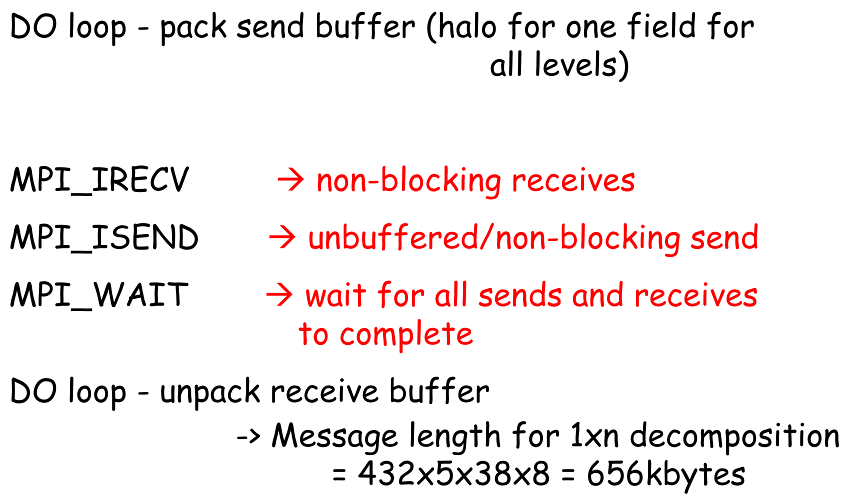
```
!$OMP DO PARALLEL
DO loop - pack send buffer

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

                         *Note no overlap
!$OMP DO PARALLEL        between CPU and Comms
DO loop - unpack receive buffer
```

Figure 20   Transposition routine from IFS

The optimized boundary swap routine from the UM is coded similarly with no overlap between communications and CPU. The difference being that non-blocking MPI_IRECVs are posted first followed by non-blocking MPI_ISENDs.



```
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
```

Figure 21   Boundary swap routine from UM

6. OpenMP parallelisation

OpenMP parallelisation can be used for systems with shared memory nodes. The OpenMP parallelisation may be used in combination with an MPI distributed memory parallelisation as in IFS. Where possible the OpenMP parallelisation should be at the highest possible level. For example in IFS the Nblocks loop in Figure 7 is parallelized with the high-level physics calling routine inside. In between some lower level loops are parallelised with OpenMP - for example the buffer packing in the transposition routines. Note OpenMP parallelisation at low level does not scale so well as the high level because all threads access memory at the same time.

The advantage of using OpenMP in addition to MPI is that there are less MPI tasks and therefore less MPI massage passing for a fixed number of processors. The disadvantage of introducing OpenMP is that it can give code maintenance problems and bugs can be difficult to find. All low level code in a parallel region must be thread safe. For example different threads must not update the same array in a module. Many runs can be done before a bug is found, as the problem will be timing dependent. Also to have an efficient OpenMP implementation, almost the whole code needs to be parallelised - except for the MPI communications. But as the OpenMP threads are idle during the MPI communications, processors are available to carry out operating system functions.

For IFS, using mixed MPI and OpenMP is better than MPI alone for the same number of processors. This is particularly so for larger numbers of processors as can be seen in Figure 22. The memory saving from running with OpenMP threads is especially important for IFS because MPI tasks with multiple threads use very little more memory than a single thread.

There are three different possible paradigms for mixed MPI and OpenMP parallelisations:

- 'Master only' - This is where all communications are done by the master thread and the other threads are idle during the communications. This is how IFS has been parallelised.
- 'Funneled' - This is when all communications are done by one thread but the other threads can use the CPU during communications.
- 'Multiple' - This is when several threads (maybe all) participate in communications.

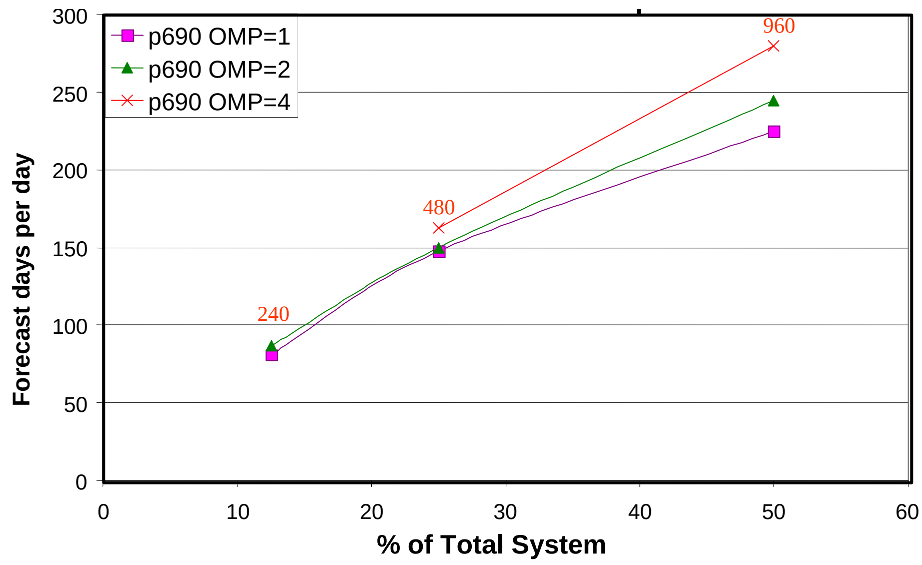


Figure 22 RAPS-6 benchmark - IFS T799 L90 - run on different numbers of OpenMP threads on hpca

7. Optimization

Today's Fortran compilers do most of the basic code tuning. This depends on the optimization level that has been chosen. At ECMWF we use '-O3 -qstrict' flags on the IBM Fortran compiler. The -qstrict flag ensures that operations are performed in a strictly defined order. With higher optimization levels we found that we did not get bit-reproducible results when NPROMA was changed. The -qstrict flag does however stop certain optimizations like replacing several divides by the same quantity, by one reciprocal and corresponding multiplies.

The main optimization activities done for IFS on the IBM p690 system have been as follows:

- Add timings (at subroutine level and also for specific sections of the code using Dr.Hook[4])
- MPI communications (remove buffered MPI_SENDS and remove overlap of communications and CPU)
- Add more OpenMP parallel regions
- 'By hand' optimization of divides - necessary because of -qstrict
- Use IBM 'vector' functions
- Remove copies and zeroing of arrays where possible
- Optimize data access for cache - by grouping different uses of the same array together.
- Remove allocation of arrays in low-level routines.
- Basic Fortran optimizations, e.g. ensuring inner loops are on first array index.

The IBM has scalar processors, but as we have worked on optimization we have tried to keep a code that performs well on scalar and vector processors. This has meant that in a few places we have had to put in alternate code for vector and scalar versions.

To investigate the effects of various optimizations we have in the IFS code over the years, a run was done with some of the main improvements removed and then each was replaced one by one. The result is shown in Figure 23.

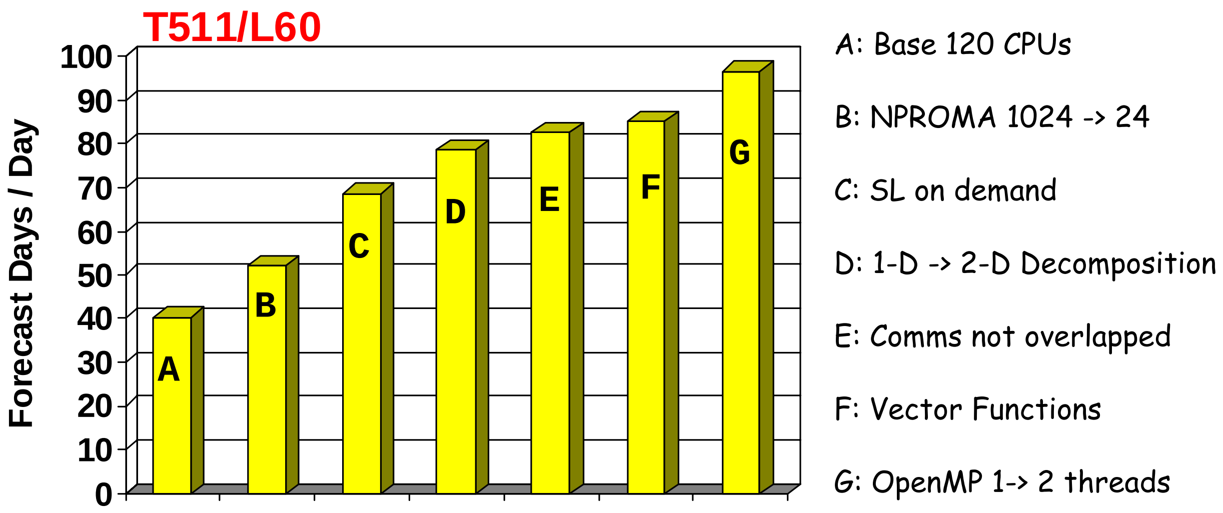


Figure 23 Effects of various optimizations on IFS performance

Acknowledgements

I am very grateful for assistance from Bob Carruthers from Cray, John Hague from IBM, Alan Dickinson and Paul Selwood from the Met Office and many colleagues from ECMWF.

References

Burton, P. and B. Carruthers: Porting the UK Met Office’s Unified Model to the CRAY X1. CUG proceedings 2004.

Dent, D.: The influence of Computer Architectures on Algorithms. ECMWF: Recent developments in numerical methods for atmospheric modeling. 7-11 September 1998.

Persson, A.:: Early Operational Numerical Weather Prediction outside the USA. 17 May 2004

Saarinen, S.: ODB user guide. ECMWF 2004