IFS - past, present and future

Yannick Trémolet, Mike Fisher, Anne Fouilloux, John Hague, Tomas Wilhelmsson, George Mozdzynski, Mats Hamrud, Nils Wedi, Peter Towers, Karim Yessad, Ryad El Khatib, Philippe Marguinaud & Deborah Salmond
Agenda

• ECMWF HPC systems

• IFS Forecast and 4D-Var: performance and scalability

• Developments at ECMWF
One of ECMWF’s two IBM Power6 clusters

P6 : 24 frames = 24*12*32 = 9216 cores = 18432 threads

SMT : 2 Threads per core
Peak per core : 18.8 Gflops
IB switch : 8 links per node
2011/12 : ECMWF will install two Power7 clusters

P6 : 24 frames = 24*12*32 = 9216 cores = 18432 threads

SMT : 2 Threads per core
Peak per core : 18.8 Gflops
IB switch : 8 links per node

P7 : 8 frames = 762*32 = 24384 cores = 48768 threads

SMT : 2-4 Threads per core
Peak per core : 2 x 15 Gflops
HFI switch : 31 links per node
History of scalability of IFS forecast model

![Graph showing the history of scalability of IFS forecast model.

- CRAY T3E-1200 1998
- IBM p690+ 2004
- IBM p575+ 2006
- IBM Power 6 p575 2008

Scale: Gflops on the y-axis and Number of cores on the x-axis. Ideal scaling is indicated by a straight line.](image)
T1279 L91 Forecast runs up to whole P6 cluster

- 9% peak with 2048 threads
- 5% peak with 16384 threads
T1279 L91 Forecast runs up to whole P6 cluster

Operations
48 Nodes

Time-step = 10 min
Horizontal = 16km
Vertical = 91 levels
Horiz points = 2x10^6
Total Pflops = 6.5

9% peak
2048 threads

5% peak
16384 threads

Tflops

Nodes
4D-Var and 10-day forecast
4D-Var and 10-day forecast

Obs time window = 12hr

Forecast &
Traj_0, 1, 2 : T1279
Horiz Points = 2 \times 10^6
NPROMA = 32

Min_0 : T159
Horiz points = 36000
NPROMA = 12

Min_1 & 2 : T255
Horiz points = 89000
NPROMA = 29

Vertical = 91 levels
Scalability of T1279 Forecast and 4D-Var

- 10-day Forecast
- 4D-Var
- Ideal

Operations
48 Nodes

User Threads on IBM Power6

Speed-up

Operations
48 Nodes
### Computational efficiency of 4D-Var (12hr window) & 10-day Forecast on 48 nodes

<table>
<thead>
<tr>
<th>Step</th>
<th>Walltime (secs)</th>
<th>%peak</th>
<th>Time-step (secs)</th>
<th>Number of timesteps</th>
<th>**Number of Grid-points per MPI task</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1279 10-day Forecast</td>
<td>2771</td>
<td>8.3</td>
<td>600</td>
<td>1440</td>
<td>10000</td>
</tr>
<tr>
<td>Total 4D-Var = 4xTraj + 3xMin</td>
<td>2735</td>
<td>3.2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**All runs with 192 MPI tasks and 16 OpenMP threads**
Scalability of T1279 Forecast and 4D-Var User Threads on IBM Power6

Speed-up

1.2
1.4
1.6
1.8
2
2.2
2.4
2.6

Operations
48 Nodes

10-day Forecast
4D-Var
Ideal

Traj_1 & Traj_2
Traj_0

User Threads on IBM Power6

Operations 48 Nodes
Computational efficiency of 4D-Var (12hr window) & 10-day Forecast on 48 nodes

<table>
<thead>
<tr>
<th>Step</th>
<th>Walltime (secs)</th>
<th>%peak</th>
<th>Time-step (secs)</th>
<th>Number of timesteps</th>
<th>**Number of Grid-points per MPI task</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1279 10-day Forecast</td>
<td>2771</td>
<td>8.3</td>
<td>600</td>
<td>1440</td>
<td>10000</td>
</tr>
<tr>
<td>T1279 Traj_0 (incl. screening)</td>
<td>362</td>
<td>3.4</td>
<td>600</td>
<td>72</td>
<td>10000</td>
</tr>
<tr>
<td>Total 4D-Var = 4xTraj + 3xMin</td>
<td>2735</td>
<td>3.2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**All runs with 192 MPI tasks and 16 OpenMP threads**
Scalability of T1279 Forecast and 4D-Var

Operations
48 Nodes

Min_0
Min_1 & Min_2

10-day Forecast
4D-Var
Ideal

User Threads on IBM Power6

Speed-up

2.6
2.4
2.2
2
1.8
1.6
1.4
1.2
1
2000 3000 4000 5000 6000
Computational efficiency of 4D-Var (12hr window) & 10-day Forecast on 48 nodes

<table>
<thead>
<tr>
<th>Step</th>
<th>Walltime (secs)</th>
<th>%peak</th>
<th>Time-step (secs)</th>
<th>Number of timesteps</th>
<th>**Number of Grid-points per MPI task</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1279 10-day Forecast</td>
<td>2771</td>
<td>8.3</td>
<td>600</td>
<td>1440</td>
<td>10000</td>
</tr>
<tr>
<td>T1279 Traj_0 (incl. screening)</td>
<td>362</td>
<td>3.4</td>
<td>600</td>
<td>72</td>
<td>10000</td>
</tr>
<tr>
<td>T255 Min_2</td>
<td>389</td>
<td>3.2</td>
<td>1800</td>
<td>*1152</td>
<td>500</td>
</tr>
<tr>
<td>Total 4D-Var = 4xTraj + 3xMin</td>
<td>2735</td>
<td>3.2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*T255 Min_2 : 1152 = 24 iterations x 24 steps x 2 (TL and AD)

**All runs with 192 MPI tasks and 16 OpenMP threads
Scalability of Forecast: 48 to 96 nodes

* BARRIERS inserted for timing purposes = load imbalance + jitter
Scalability of 4D-Var: 48 to 96 nodes

* Several types of I/O including Observational Data Base
How do we make 4D-Var scale to more processors?
T1279 Forecast and 4D-Var (weak) scalability from 91 to 137 levels

<table>
<thead>
<tr>
<th></th>
<th>91 levels on 48 nodes</th>
<th>137 levels on 96 nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Wallclock time in secs</td>
<td>Wallclock time in secs</td>
</tr>
<tr>
<td>10-day forecast</td>
<td>2734</td>
<td>2491</td>
</tr>
<tr>
<td>4D-Var</td>
<td>2735</td>
<td>2780</td>
</tr>
</tbody>
</table>

1.5 x levels $\rightarrow$ 1.6 times flop count
Ensemble Data Assimilation (EDA) compared to Deterministic 4D-Var (12hr)

<table>
<thead>
<tr>
<th></th>
<th>Wallclock time in secs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>EDA</strong> ( (1+10) \times 4D-Var ) T399/T159</td>
<td>2428 on 88 nodes (8 nodes per member)</td>
</tr>
<tr>
<td><strong>Deterministic 4D-Var</strong> T1279/T255</td>
<td>2735 on 48 nodes</td>
</tr>
</tbody>
</table>

EDA \( \sim 1.6 \times \) cost of high-resolution deterministic
4D-Var and 10-day forecast

- Obs time window = 12hr
- Forecast &
  - Traj_0,1,2 : T1279
  - Horiz Points = 2x10^6
  - NPROMA = 32
- Min_0 : T159
  - Horiz points = 36000
  - NPROMA = 12
- Min_1 & 2 : T255
  - Horiz points = 89000
  - NPROMA = 29
- Vertical = 91 levels
4D-Var and 10-day forecast

Obs time window = 12hr

Forecast &
Traj_0,1,2 : T1279
Horiz Points =2x10^6
NPROMA=32

Min_0 : T159
Horiz points =36000
NPROMA=12

Min_1 & 2 : T255
Horiz points =89000
NPROMA=29

Vertical = 91 levels
Object-Oriented Prediction System – The OOPS project

• Data Assimilation algorithms manipulate a limited number of entities (objects):
  - $x$ (State), $y$ (Observation),
  - $H$ (Observation operator), $M$ (Model), $H^* \& M^*$ (Adjoint),
  - $B \& R$ (Covariance matrices), etc.

  - To enable development of new data assimilation algorithms in IFS, these objects should be easily available & re-usable

- More Scalable Data Assimilation
- Cleaner, more Modular IFS
OOPS \(\rightarrow\) More Scalable Data Assimilation

- One execution instead of many will reduce start-up - also I/O between steps will not be necessary

- New more parallel minimisation schemes
  - Saddle-point formulation

- For long-window, weak-constraint 4D-Var: Min steps for different sub-windows can run in parallel as part of same execution
OOPS → More Scalable Data Assimilation

- One execution instead of many will reduce start-up - also I/O between steps will not be necessary

- New more parallel minimisation schemes
  - Saddle-point formulation

- For long-window, weak-constraint 4D-Var: Min steps for different sub-windows can run in parallel as part of same execution

<table>
<thead>
<tr>
<th>48 nodes</th>
<th>0-6hr</th>
<th>7-12hr</th>
<th>13-18hr</th>
<th>19-24hr</th>
<th>Minimise</th>
</tr>
</thead>
<tbody>
<tr>
<td>192 nodes</td>
<td>0-6hr</td>
<td>7-12hr</td>
<td>13-18hr</td>
<td>19-24hr</td>
<td>Minimise</td>
</tr>
</tbody>
</table>

Time
IFS code cleaning:
or Global variables in Fortran90 modules /= modular code

• IFS has many globally accessible variables stored in Fortran modules - equivalent to COMMON blocks

• Target is to remove (most) global variables

• Convert references to the module variables to references of elements of derived types

• Input & output to code block - by passing derived types through argument lists

• This and other cleaning and modularisation work will be done incrementally in existing IFS code
Plan for IFS → OOPS

- Keep computational parts of the existing IFS code and reuse them in a new OOPS structure

- Use top-down and bottom-up approach:
  1. From the top: develop a new modern, flexible structure, using advanced programming and object-oriented concepts
     - OOPS C++ layer tested with toy models (Lorenz, QG)
  2. From the bottom: Progressively create self-contained, encapsulated units of code (Fortran) extracted from IFS
     - Include as part of new IFS releases
  3. Put the two together: plug encapsulated entities of the IFS into OOPS
OOPS-IFS

O-O control layer (written in C++)
replaces IFS control routines → defines Algorithm

OOPS interface methods to IFS (written in Fortran)

IFS Encapsulated code units

- State & Forecast Model
- Observation operators
- ODB in C
- Covariance Matrices
Horizontal Resolution for 10-day forecast at ECMWF

<table>
<thead>
<tr>
<th>Date</th>
<th>Truncation</th>
<th>Resolution</th>
<th>CPUs on HPC system</th>
<th>Computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1983</td>
<td>T63</td>
<td>316km</td>
<td>1</td>
<td>Cray-1</td>
</tr>
<tr>
<td>1987</td>
<td>T106</td>
<td>188km</td>
<td>4</td>
<td>Cray XMP</td>
</tr>
<tr>
<td>1991</td>
<td>T213</td>
<td>95km</td>
<td>8</td>
<td>Cray YMP</td>
</tr>
<tr>
<td>1998</td>
<td>T319</td>
<td>63km</td>
<td>116</td>
<td>VPP700</td>
</tr>
<tr>
<td>2000</td>
<td>T511</td>
<td>39km</td>
<td>100</td>
<td>VPP500</td>
</tr>
<tr>
<td>2006</td>
<td>T799</td>
<td>25km</td>
<td>4960</td>
<td>IBM P5</td>
</tr>
<tr>
<td>2010</td>
<td>T1279</td>
<td>16km</td>
<td>18000</td>
<td>IBM P6</td>
</tr>
<tr>
<td>~2015</td>
<td>T2047</td>
<td>10km</td>
<td></td>
<td></td>
</tr>
<tr>
<td>~2020</td>
<td>*T3999</td>
<td>5km</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* 5km → Non-hydrostatic: substantially different cloud-microphysics, turbulence parameterization & different dynamics-physics interaction
ALSO need 2 iterations per timestep of dynamics for stability

--

ECMWF
Computational Cost at T3999: Hydrostatic vs. Non-Hydrostatic IFS

Non-Hydrostatic T3999

Hydrostatic T3999
Fast Legendre Transform in IFS

Floating point ops per time-step (Gflop)

Walltime per time-step (secs)

* butterfly algorithm
Tygert 2010
\[ \text{cost } C \times N \times \log(N) \quad (C \ll N) \]
- not \( N \times N \)
IFS → Climate:
The Athena Project (6 months of IFS on a Cray XT4)

• Athena: Cray XT4 with 4489 nodes - 2.3 GHz quad-core AMD Opteron
  • 9.2 Gflops peak per core
  • 18000 cores

• 2 x T1279L91 (~16 km, Δt = 600s) 47-year forecasts (1960-2007 and 2070-2117)
  • 5120 Opteron cores = 1280 MPI tasks x 4 OpenMP threads
  • Walltime 50 days for 47 years

• NWP runs at ECMWF
  • 1536 Power6 cores = 384 MPI tasks x 8 OpenMP threads
  • Walltime 45 mins for 10 days
Summary

• ECMWF HPC systems
  • IBM Power6 (2x9000 cores) \(\rightarrow\) IBM Power7 (2x25000 cores)

• IFS Forecast and 4D-Var: performance and scalability
  • 10-day T1279L91 Forecast & 4D-Var T1279/T255
  • Increase vertical resolution from 91 to 137 levels
  • Operational Ensemble Data Assimilation (EDA) T399/T159

• Developments at ECMWF
  • IFS goes Object-Oriented \(\rightarrow\) OOPS
    - Long-Window 4D-Var (increase from 12hr to 24hr)
    - More Scalable minimisation \(\rightarrow\) Saddle-point formulation
  • Investigation of Ensemble Kalman Filter (EnKF)
  • Non-Hydrostatic experimentation \(\rightarrow\) T4000
  • Continuous Observation Processing at ECMWF (COPE)
Questions?