Porting and Optimisation of UM on ARCHER

Karthi Sivalingam, NCAS-CMS

HPC Workshop ECMWF
Acknowledgements

> NCAS-CMS
  > Bryan Lawrence
  > Jeffrey Cole
  > Rosalyn Hatcher
  > Andrew Heaps
  > David Hassell
  > Grenville Lister
  > William McGinty
  > Simon Wilson
  > Annette Osprey

> Met Office
  > Mick Carter
  > Jean-Christophe Rioual
  > Matthew Hambley
  > Jamie Kettleborough

> CRAY & EPCC
Summary

- Porting UM to ARCHER (XC30)
- Initial Performance on ARCHER
- Performance analysis
- Tuning MPI and IO
- Optimising thread performance
Porting from Cray XE6 to XC30 is simple if CRAY compiler is used
## HECTOR and ARCHER

<table>
<thead>
<tr>
<th></th>
<th>AMD Opteron</th>
<th>Intel Xeon</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Clock speed</strong></td>
<td>2.3</td>
<td>2.7</td>
</tr>
<tr>
<td><strong>Cores per node</strong></td>
<td>32</td>
<td>24</td>
</tr>
<tr>
<td><strong>Threads per core</strong></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td><strong>Vector</strong></td>
<td>sse</td>
<td>avx</td>
</tr>
<tr>
<td><strong>Bits wide</strong></td>
<td>128</td>
<td>256</td>
</tr>
<tr>
<td><strong>L2 cache</strong></td>
<td>512 KB</td>
<td>256 KB</td>
</tr>
<tr>
<td><strong>L3 cache</strong></td>
<td>6 MB</td>
<td>30 MB</td>
</tr>
<tr>
<td><strong>Peak DP flops per node</strong></td>
<td>294 GFlops</td>
<td>518 GFlops</td>
</tr>
</tbody>
</table>

Memory bandwidth, MPI bandwidth and IO bandwidth?
ARCHE... 3008 compute nodes
- Dual socket, 12 cores - forming 2 NUMA nodes
- 64 (or 128) GB memory per node
- Cray Aries network
- Cray XC30 Dragonfly Topology
- Copper and Optical cabling
- Lustre parallel file system
All the jobs are standard jobs ported from MetOffice to ARCHER. All the jobs use Cray compiler and strict bit comparability is enforced.

<table>
<thead>
<tr>
<th>Job name</th>
<th>Columns</th>
<th>Rows</th>
<th>Land points</th>
<th>Vertical levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>N96</td>
<td>192</td>
<td>144</td>
<td>11271</td>
<td>85</td>
</tr>
<tr>
<td>N216</td>
<td>432</td>
<td>324</td>
<td>52614</td>
<td>85</td>
</tr>
<tr>
<td>N512</td>
<td>1024</td>
<td>768</td>
<td>280592</td>
<td>85</td>
</tr>
</tbody>
</table>
Initial Performance Comparison

Above jobs in HECTOR and ARCHER use similar MPI decomposition and same number of threads.

Wallclock time refers to the time taken to complete simulation of 1 model day.
N96 job on ARCHER

All the above runs uses 12 MPI PEs and 2 threads per node. Cost per MAU of ARCHER is £1640
All the above runs uses 12 MPI PEs and 2 threads per node. Cost per MAU of ARCHER is £1640.
N512 job on ARCHER

All the above runs uses 12 MPI PEs and 2 threads per node. Cost per MAU of ARCHER is £1640.

When number of PEs in EW or NS goes more that 120 or 60 respectively, the model does not bit compare.
What to do?

- N512 job does not scale beyond 250 nodes.
- N216 job does not scale beyond 96 nodes.
- Is 12 MPI PEs per node ideal? Is 2 threads per PE ideal? Can we use hardware threads?
- Need to analyse the performance.
- CRAY performance analysis tools readily available.
  - CrayPAT
  - Apprentice
MPI Bandwidth

- Message passing in UM is done only by thread 0.
- Message passing overhead increases by 14% as the number of nodes is increased from 72 to 192.
- Scaling of MPI bandwidth improved by using
  - MPI Asynchronous progress
  - Use unassigned CPU for MPI
  - MPI rank placement (Grid order)
PE decomposition 24 x 36

Node 0

Node 4

Node 10

Node 6

Node 0

Node 2

Node 3

Node 4

12 PEs per node

SMP - MPI Rank Order

GRID - MPI Rank Order

Nearest neighbour communications

12 PEs per node

111 112 113

75 76 77

39 40 41

3 4 5

144 145 146

180 181 182

216 217 218

252 253 254

255 256 257

219 220 221

183 184 185

147 148 149
GRID vs SMP MPI rank order

Almost perfect scaling

% Speedup

No. of model years per day

Number of nodes

All the above results are obtained using 12 PEs per node and two OPENMP threads. IO is not included in the measurement as the IO resources are shared and accurate measurement is difficult.
Load imbalance

Can expect poor thread scaling
OPENMP directives

- 60% to 75% thread imbalance in UM jobs.
- Coverage of OPENMP regions is less than 50%.
- Cray Reveal - integrated performance analysis and code optimisation tool.
  - Provides loop analysis and scoping of serial loops.
  - Suggests OPENMP directive that can be inserted to a loop.
  - Can attach the performance data collected during execution to identify profile of loops.
  - Requires knowledge of OPENMP to resolve conflicts and issues.
  - Works only with Cray compiling environment.
  - Does not support tasks, barrier, critical or atomic regions.

For more details - refer Cray documentation (not much)
CRAY Reveal

Source - .../uml/xjle/umatemos/psrc/UM/control/top_level/microphys_ctl.f90

847 ! Dry level T values (only required for diage)
848
849 DO k = qdims%k_end + 1, tdims%k_end
     
850   DO j = qdims%j_start, qdims%j_end
851     DO i = qdims%i_start, qdims%i_end
852       t_work(i,j,k) = t_n(i,j,k)
853     END DO
854   END DO
855   END DO
856   END DO
857
858
859
860
861
862
863
864
865
866
867
868
869

A loop starting at line 849 was not vectorized because a better candidate was found at line 853.
- The loop is flat.
- The loop is flat.

/home2/m02/n02/karthik/umatemos/tmp/xjle.pl loaded
CRAY Reveal

Source - ... ee/um/xjle/umatmos/psrc/UM/control/top_level/microphys_ctl_90

Scope Loops Scoping Results

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Scope</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>Scalar</td>
<td>Private</td>
<td></td>
</tr>
<tr>
<td>j</td>
<td>Scalar</td>
<td>Private</td>
<td></td>
</tr>
<tr>
<td>k</td>
<td>Scalar</td>
<td>Private</td>
<td></td>
</tr>
<tr>
<td>t_work</td>
<td>Array</td>
<td>Shared</td>
<td></td>
</tr>
<tr>
<td>t_n</td>
<td>Array</td>
<td>Shared</td>
<td></td>
</tr>
<tr>
<td>QDIMS</td>
<td>Scalar</td>
<td>Shared</td>
<td>FAIL No scoping information available</td>
</tr>
<tr>
<td>TDIMS</td>
<td>Scalar</td>
<td>Shared</td>
<td>FAIL No scoping information available</td>
</tr>
</tbody>
</table>

First/Last Private

Enable First/Last Private

Find Name: 

Reduction

None

Insert Directive Show Directive Close

Directive inserted by Cray Reveal. May be incomplete.

$OMP parallel do default(none)

$OMP private (i)k

$OMP shared (t_n_work,QDIMS,TDIMS)
NB. Broken into two separate loops to make vectorisable.
CRAY Reveal

Source - ...
jlek/umatmos/ppsrc/UM/atmosphere/COSP/actsim/ailar_simulator.f90

- CILV 301
  kp_part(:,i) = 0.
  endwhere

- CIVf 306
  qpart(:,INDEX_LSLIQ) = q_lsiq(:,);
  oct08

- ILS 313
  where { rad_part(:,i).gt.0.0 }

- IL 314
  alpha_part(:,i) = 3.0/4.0 * Qscat &
  rhoair(:,i) * qpart(:,i) &
  / (rhopart(i) * rad_part(:,i))

- CILV 318
  alpha_part(:,i) = 0.
  endwhere

Info - Line 271
- A loop starting at line 271 was not vectorized because a recurrence was found on "zheight" at line 272.
- The loop is flat.
- The loop is flat.
2389 serial loops are parallelised using OPENMP directives as suggested by CRAY Reveal.

Fortran array notation expressions, for all and where statements are parallelised using

```fortran
!$OMP PARALLEL WORKSHARE
!$OMP END PARALLEL WORKSHARE
```

Bit reproducibility is tested for correctness of OPENMP directives.

default(none) option is used for all the newly added directives.
Performance speedup

- UM 8.6
- UM 8.6 + Cray Reveal
- % speedup

N96

N512

Time in seconds / model day

OPENMP threads

% speedups

500 675 850 1025 1200

0 5 10 15 20

2 4 6

OPENMP threads
Conclusions

- ARCHER (Cray XC30) has good MPI bandwidth and supports hardware threads
- CrayPAT tools used to analyse the performance
- MPI tuning result in 5 to 12% speedup
- UM has poor thread load balance.
- CRAY Reveal tool used to parallelise serial loops
- New OPENMP directives improve performance by 5% up to 20% when using 2 to 6 threads.
THANK YOU

- ARCHER (Cray XC30) has good MPI bandwidth and supports hardware threads
- CrayPAT tools used to analyse the performance
- MPI tuning result in 5 to 12% speedup
- UM has poor thread load balance.
- CRAY Reveal tool used to parallelise serial loops
- New OPENMP directives improve performance by 5% up to 20% when using 2 to 6 threads.