WRF Quilting and Decomposition Notes

Eric Kemp
2 March 2015
Quilting Preliminaries

• Two types of MPI tasks: compute (client) and I/O (server)
• Compute tasks:
  – Total number = nproc_x*nproc_y (number of processors along x and y axes for decomposition)
  – First number is zero
• I/O tasks:
  – Total number = nio_groups*nio_tasks_per_group
  – Nio_tasks_per_group cannot exceed nproc_y
  – First I/O task number immediately follows last compute task number
• Code will attempt to match each I/O task with compute tasks in east-west rows
  – Ideally nproc_y should be exact multiple of nio_tasks_per_group

Note: Only output is currently supported by I/O quilting, not input.
Sample Task Layouts

Example compute task layout

\[
\begin{array}{cccc}
8 & 9 & 10 & 11 \\
4 & 5 & 6 & 7 \\
0 & 1 & 2 & 3 \\
\end{array}
\]

\(nproc_x = 4, nproc_y = 3\)

If \(nio\_groups=2\) and \(nio\_groups\_per\_task=3\), then:

\[
\begin{array}{cccccccc}
14 & 8 & 9 & 10 & 11 & 17 \\
13 & 4 & 5 & 6 & 7 & 16 \\
12 & 0 & 1 & 2 & 3 & 15 \\
\end{array}
\]

I/O Group 1

I/O Group 2

- Each compute node in a row (e.g., tasks 0-3) will send data to a I/O server (e.g., 12 or 15) at output time.
- One I/O group is selected on-the-fly at output time to handle output to a particular file.
- Within an I/O group, the servers will forward data to designated “root” server which will perform the actual write.

Note: Above is based on running the code. The example in the source code comments does not agree!
Quilting Performance

- An I/O group can only handle one file write at a time
  - If too few I/O groups available, WRF will stall

<table>
<thead>
<tr>
<th></th>
<th>No Quilting</th>
<th>1 Groups</th>
<th>2 Groups</th>
<th>3 Groups</th>
</tr>
</thead>
<tbody>
<tr>
<td>wrfout</td>
<td>20.08 s</td>
<td>1.18 s</td>
<td>1.05 s</td>
<td>1.20 s</td>
</tr>
<tr>
<td>wrfdiag</td>
<td>0.64 s</td>
<td>63.88 s</td>
<td>0.50 s</td>
<td>0.62 s</td>
</tr>
<tr>
<td>wrf2dout</td>
<td>1.48 s</td>
<td>1.49 s</td>
<td>19.92 s</td>
<td>0.68 s</td>
</tr>
<tr>
<td>wrfpress</td>
<td>0.56 s</td>
<td>6.67 s</td>
<td>0.40 s</td>
<td>0.74 s</td>
</tr>
</tbody>
</table>

Sample output times for A24 grid with nproc_x=53, nproc_y=12 compute tasks, varying I/O groups with nio_tasks_per_group=12

Note: Reported output times are what compute tasks “see” when communicating and waiting – does not necessarily reflect time spent writing by a I/O group
Quilting Recommendations & Comments

• Set nio_groups = 5 (for wrfout, wrf2dout, wrfpress, wrfdiagnostics, wrfrst files)
• Try setting nio_tasks_per_groups so it can exactly divide nproc_y
  – Otherwise some I/O server(s) will have more compute tasks to handle than others
  – Don’t know how performance scales with tasks per group
• Race condition: Multiple groups can’t write to same file at time
• Simple test runs still showed some strange stalls during computational steps (not output times)
Decomposition Notes

• Found several third-party discussions of running WRF on HPC systems (Cray, STFC, Lenovo, Barcelona Supercomputing Center)

• All recommend setting nproc_x < nproc_y, but “sweet spot” is problem and hardware dependent

• Arguments (from Cray):
  – Give inner loops larger lengths for better SSE vector and register reuse
  – Shorten outer loop lengths for better cache use and register reuse
  – Get more favorable halo exchange communications pattern

• Suggest testing this!

• Cray also points out a “reorder_mesh” namelist option to try keeping adjacent compute tasks on the same node, but this doesn’t work with quilting.
WRF HPC References

- [http://weather.arsc.edu/Events/AWS10/Presentations/Johnsen.pdf](http://weather.arsc.edu/Events/AWS10/Presentations/Johnsen.pdf)
- [https://cug.org/5-publications/proceedings_attendee_lists/CUG10CD/pages/1-program/final_program/CUG10_Proceedings/pages/authors/01-5Monday/3C-Porter-paper.pdf](https://cug.org/5-publications/proceedings_attendee_lists/CUG10CD/pages/1-program/final_program/CUG10_Proceedings/pages/authors/01-5Monday/3C-Porter-paper.pdf)
- [http://www.ecmwf.int/sites/default/files/HPC-WS-Christidis.pdf](http://www.ecmwf.int/sites/default/files/HPC-WS-Christidis.pdf)