# Tiling development progress (September 2020)

## Summary

Tiling has been implemented for most of the code in the “active tracers” part of the timestepping subroutine (between *trc_stp* and *tra_atf*). In routines that could not be completely tiled, some workarounds are necessary to preserve the results. Some of these workarounds function by temporarily disabling the tiling so that work is instead done on the full domain.

timing.output results from a GYRE benchmark show a reduction in cost of 35-70% for *tra_ldf*, 35-50% for *tra_adv*, ~65% for *tra_zdf* and ~35% for *tra_sbc*. For the parameters *jpni*, *jpnj* = (6, 3) and *jpi*, *jpj* = (42, 122), the optimal tile size was found to be i, j = (40, 7). Tiling in i reduces the performance improvements.

The next course of action is to determine whether this code can be prepared for the November merge. The priority would be to remove workarounds where possible, as some of this code does not benefit from the tiling. The code must also be tested with *nn_hls* = 2 and in SETTE.

## Branch and wiki

http://forge.ipsl.jussieu.fr/nemo/browser/NEMO/branches/2020/dev_r13383_HPC-02_Daley_Tiling

https://forge.ipsl.jussieu.fr/nemo/wiki/2020WP/HPC-02_Daley_Tiling

## Code changes

The code changes can be split into:

1. New code that implements the tiling functionality
2. Code that is tiled
3. Code that is tiled, but contains workarounds that **would** be removed for the merge
4. Code that is tiled, but contains workarounds that **would not** be removed for the merge
5. Code that is not tiled and requires workarounds
6. Code that can’t be tiled without changing results

Changes #1 and #2 are described on the Wiki page.

Changes #3 - #6 are some of those described in a separate document on the issues encountered during the development. These need to be considered before the merge and are highlighted here.

### Workarounds that would be removed for the merge

- *lbc_lnk* calls

  Code containing *lbc_lnk* calls requires that all tiles be processed prior to the call.

  The workaround is to disable the tiling when calling these routines (but note that the tiling changes are still implemented).

  This is a particular problem for the tracer advection code, as tiling can only be used with second order centred advection. The workaround in this case is a small block of code at the start of *tra_adv* that disables the tiling so that the full domain is worked on, rather than the tile. A similar workaround is implemented in *tra_ldf*.

  It is hoped that all *lbc_lnk* workarounds would be removed before the merge. The *lbc_lnk* calls used to set halo points for *iom_put* can already be removed as these are no longer output by the diagnostics, so the workarounds can also be removed. Other *lbc_lnk* calls will remain in place while one-point haloes are still supported. The workarounds associated with these calls can be removed by requiring that tiling be used with multi-point haloes (i.e. *nn_hls* > 1).

Workarounds that would not be removed for the merge

- *iom_put* calls

  This is a similar issue to the *lbc_lnk* calls, in that all tiles must be processed prior to the call.

  The workaround is to save the result of each tile in a working array of size (*jpi*, *jpj*), then send this to XIOS. This allows the rest of the code in the routine to be tiled at the cost of increasing memory usage.

  This workaround can be removed when XIOS is able to receive data separately from each tile. It is not expected that this will be available in time for the merge.

Workarounds for untiled code

- *trd_tra* calls

  *trd_tra* has not yet been tiled and also contains *iom_put* calls. Since these calls appear in tiled TRA code, a workaround is required to avoid spurious values in the trends diagnostics.

  This workaround is the same as for *iom_put* calls; the result of each tile is saved in a working array of size (*jpi*, *jpj*), which is passed to the *trd_tra* call.

  This is implemented in every routine with a *trd_tra* call and the changes are therefore both numerous and repetitive. They can be removed after *trd_tra* is tiled and the requisite XIOS functionality is available, neither of which are expected in time for the merge.

Tiled code that changes results

- *prt_ctl* calls

  As described in this document, the use of tiling currently changes the results of *prtctl*. However, the results of other diagnostics including *stpctl* (run.stat) are unchanged by the tiling.

  This seems nontrivial to resolve. Provided that the tiling does not otherwise affect reproducibility (i.e. a trunk and development branch should bit compare if the same tiling is used in both), would this be considered acceptable for the merge?

**Performance results**

The following refers to the results in this document.

Tables 1-3 show timing.output results for a 180-day benchmark GYRE simulation with parameters *jpni*, *jpnj* = (6, 3) and *jpi*, *jpj* = (42, 122). Elapsed times relative to the trunk (“REF”) are shown for several tile decompositions, with the “0x0” decomposition indicating that tiling is not used. Furthermore, *iom_put* timings have been implemented to account for the time spent waiting for XIOS.

Table 1 shows results for the standard GYRE configuration, except using the second order centred advection scheme instead of the FCT scheme. This was chosen because tiling is currently disabled for the FCT scheme due to *lbc_lnk* calls.

The tiling reduces the cost of *tra_ldf* by up to ~30%, *tra_adv* by ~35%, *tra_zdf* by ~65% and *tra_sbc* by ~35%. The optimal tile size for the first three, most expensive sections is i, j = (40, 7). Tiling in the i dimension results in reduced performance, consistent with preliminary tests by Maff Glover (Met Office).

The cost of *dia_ptr* is reduced by up to ~20%, with only a weak dependence on tile size. This is attributed to the restructuring of *dia_ptr* to accommodate tiling, which has reduced the number of *mpp_sum* communications per timestep from 90 to 13.

The cost of *dia_wri* is reduced by ~40%, regardless of tile size. It is not clear why this is the case, as this code has not been tiled at all. Similar results are shown for sections that take less than a second to run, but these are not likely to be accurate.

Table 2 shows timing results for each combination of the *tra_ldf* namelist options and only for the *tra_ldf* section. The cost of *tra_ldf* is reduced by 35-70%, depending on the choice of scheme. As tiling is disabled for the bilaplacian operator ("blp_" jobs) due to *lbc_lnk* calls, none of these simulations show any performance improvements from the tiling. Across all schemes, the optimal tile size is i, j = (40, 7).

Table 3 shows the same results as table 2, except for the *tra_adv* namelist options and section. The second order centred advection ("cen_h2" jobs) is the only scheme that benefits from the tiling, a reduction in cost of 35-50%, as tiling is disabled for all other schemes due to *lbc_lnk* calls. Again, the optimal tile size is i, j = (40, 7).