OASIS3-MCT_4.0 Timing Study with MCT 2.10.beta1

Tony Craig, Sophie Valcke
March 2018

CECI, Université de Toulouse, CNRS, CERFACS, Toulouse, France
TR-CMGC-18-38

Introduction

This document summarizes the results from a timing exercise carried out in January 2018 on the tc17b branch of OASIS3-MCT. The modifications on this branch will be included in the OASIS3-MCT_4.0 release. The motivation is to document performance differences between MCT 2.8 and MCT 2.10.beta1 available from the MCT repository in September 2017 for a high-resolution case running on a high number of cores, as one of the most challenging cases for OASIS3-MCT and MCT. MCT 2.10.beta1 includes a refactoring of the router initialization in MCT that specifically addresses initialization performance at high numbers of cores. At the same time, timing tests were carried out for a couple of other new OASIS3-MCT features and those results are also documented.

Executive Summary

The main conclusions from testing the relatively high-resolution IS-ENES2 bechmark case (3000 x 3000 grid points) on 1600 and 3600 cores per component on Météo-France Bullx "beaufix" are:

- Compared to MCT 2.8, MCT 2.10.beta1 improves by one to two orders of magnitude the performance of the initialization of the MCT routers for this test case. This is particularly true for complex decompositions and rearrangements. The routers define the rearrangement patterns for the sparse matrix multiply (see section 3), and the rearrangement associated with coupling data between the source and target processes (see section 4). This reduces the total initialization time in OASIS3-MCT from O(1-3 minutes) to O(10-20 seconds) for the high resolution case (3000x3000 grid points) on 3600 cores per component.
- The remapping cost with the new "decomp_wghtfile" method, using the remapping weights to define the mapping decomposition, is generally faster than the original "decomp_1d" method at runtime. However, the decomp_wghtfile also takes longer to initialize, as expected. The extra initialization cost is partly due to the fact that the mapping weight file has to be read twice for the decomp_wghtfile option. The decomp_wghtfile option also results in increased cost for the initialization of the mapping decomposition of the target grid on the sources tasks, the sparse matrix initialization, and for the initialization of the router between the mapping

decomposition and the target decomposition. These additional costs are all during initialization and created because of the complexity of the decomp_wghtfile decomposition compared to the 1d decomposition. However, while those increased costs can be relatively high, MCT 2.10.beta1 has mitigated the absolute cost to O(seconds). So in general, those costs will usually be small enough that for production run lengths, it will be worth spending extra time during initialization to improve the run time performance.

- The current tests suggest that the original way to read the mapping file ("orig", where the weights read by the model master task are then broadcast to all other tasks and each task then saves the weights that will be applied to its grid points) is faster than the "ceg" option (where the master task reads the weights and then identifies to which other task each weight should be sent and effectively sends only those). But both perform similar and absolute times are reasonable.

As always, the optimal OASIS3-MCT performance settings may vary. Each coupled system will perform differently depending on the coupling sequence, number of coupling fields, resolution, number of tasks, machine, and so forth, and each coupled system should evaluate the impact of the different options offered by the coupler on its overall performance.

Overview

The VHR configuration of the IS-ENES2 coupling technology benchmarks (Valcke et al. 2017) was run on Météo-France Bullx "beaufix", coupling two components of size 3000x3000 running concurrently on different MPI tasks and different cores using the intel 16.1.150 compiler and the intelmpi 5.1.2.150 MPI library. In this case, the decompositions on the two components were the same, a 2d decomposition of square blocks. The runs were done with MCT 2.8 and MCT 2.10.beta1 with either 1600 or 3600 MPI tasks per component.

For this set of tests the remapping (also known as interpolation or regridding) was always performed on the source component tasks. So at run time, a sparse matrix multiplication using the mapping weights will be performed first to transform the source coupling field from its source decomposition to the mapping decomposition of the target grid on the source tasks. Then the coupling field will be rearranged from that mapping decomposition on the source tasks to the target decomposition on the target tasks. In practice, remapping can also be done on the target tasks after coupling the data from the source to the target processes, and while the timings would be different, the overall conclusions should be similar.

For these timings, different remapping options, some of them already in OASIS3-MCT_3.0 (Craig et al. 2017) and some of them available in the next OASIS3-MCT_4.0 release, were considered. Reading the mapping weight file was done with both the

“orig” or “ceg”method. In both methods, the weights are read in chunks by the model master task. With the orig option, the weights are then broadcast to all other tasks and each task then saves the weights that will be applied to its grid points. With the ceg option, the master task reads the weights and then identifies to which other task each weight should be sent. Then a series of exchanges are carried out with the root task sending each other task just the weights needed by that other task. The orig method sends much more data but is more parallel, while the ceg method does most of the work on the master task but less data is communicated.

The mapping decomposition was tested with both the original “decomp_1d” or the new optimized “decomp_wghtfile” method that will be available in OASIS3-MCT_4.0. To perform the remapping on the source processes, OASIS3-MCT creates a “mapping decomposition” of the target grid on the source task. This mapping decomposition is somewhat arbitrary and two options have been implemented up to now, decomp_1d or decomp_wghtfile. In decomp_1d, each target grid point is assigned to a source task in a trivial one dimensional approach. With decomp_wghtfile, a more optimal mapping decomposition can be created based on the information from the mapping weight file such that a target grid point will be associated with the source task which holds the source grid points needed for the calculation of its interpolated value.

The timings done in these tests measure various initialization and runtime costs including MCT performance, the cost to read the mapping file, the runtime sparse matrix multiply, and the total run loop cost (excluding the first and last coupling period). Some additional timers were turned on to carry out these timing with barriers as needed. The main goal was to measure individual kernels of the initialization cost. Times measured represent the maximum time across all tasks for that kernel. For the 1600 task case, each case was run twice and in all cases, both component1 and component2 timers were saved. Because most of the timing is in I/O or MPI operations, significant timing variation is seen in some timers run-to-run. This can be caused by run-to-run differences in nodes allocated or contention on the file system or interconnect. In general, the best times measured between different runs are presented, but variations and their impact on the conclusions are also sometimes noted. To a large degree, the timing difference being measured between features is much larger than the run-to-run variability.

For the current tests, the mapping is done on the source side by rearranging the data from the source decomposition to the mapping decomposition and then applying the weights. In this VHR benchmark case with identical grids on both sides, the mapping file consists of all 1s with 1 weight per grid cell. This is a particularly trivial mapping, but is useful for timing. So, in these cases, data on the source side is rearranged from the source decomposition onto the mapping decomposition, the mapping weights are applied, then the data is rearranged again from the mapping decomposition to the target component 2d block decomposition. This configuration is a clean test case for timing purposes. In practice, if two components on the same grid were being coupled with an identity matrix remapping, the remapping step would be excluded completely for a production run.

Tables are presented below and the timing output headings are defined as follows

- "part_create" is the cost to initialize the mapping decomposition of the target grid on the source tasks (with either decomp_1d or decomp_wghtfile options). In the MCT jargon, this is the cost to generate an MCT global seg map (gsmap) for the mapping decomposition.
- "mapfile read" is the cost to read the mapping weights file using the orig or ceg algorithm.
- "sminit" is the MCT sparse matrix initialization cost. This cost is largely determined by the cost to compute the router between the source decomposition and the mapping decomposition to support rearrangement in the sparse matrix multiplication on the source tasks.
- "router init" is the cost associated with computing a router between the mapping decomposition of the target grid on the source tasks and the target decomposition of the target grid on the target tasks.
- "avmult" is the sparse matrix multiply cost during the run loop and includes both the rearrangement cost and the multiply-add cost.
- "run loop" is the total time spent coupling excluding the first and last coupling period.

Timing results are broken down into smaller pieces to highlight various features.

1. Initialization of the mapping decomposition (part_create)

The table below shows the times to initialize the mapping decomposition of the target grid on the source tasks via a call to MCT. In the decomp_1d case (1d), this is a trivial one-dimensional mapping with one segment per task. For decomp_wghtfile (wghtfile), the mapping decomposition will, in this case, be similar to the two-dimensional square block decomposition associated with the source model decomposition. This decomposition is more complicated to describe and initialize within the MCT gsmap. The gsmap is initialized by having each task specify the gridcells that are on that task. The gsmap then aggregates the information across tasks so every task has the complete decomposition information of the entire grid. MPI is used to aggregate the information.

tasks per component	setting	part_create time (s), MCT 2.8	part_create time (s), MCT 2.10.beta1
1600	1d, ceg	0.0282	0.0288
1600	1d, orig	0.0283	0.0541
1600	wghtfile, ceg	0.8984	0.8808
1600	wghtfile, orig	0.8833	0.8817
3600	1d, ceg	0.0719	0.0746
3600	1d, orig	0.1140	0.0751
3600	wghtfile, ceg	0.9725	0.9967
3600	wghtfile, orig	0.9963	0.9686

- The total cost to initialize the gsmap is less than 1 second in all cases.
- The time to initialize the mapping decomposition for decomp_wghtfile is 10x-40x more expensive than for decomp_1d because of the extra complexity of the decomposition. This is likely caused by the MPI cost of sending many more shorter messages compared to the decomp_1d case and handling multiple segments per task.
- The cost to initialize the mapping decomposition on 3600 tasks is 4x more expensive for decomp_1d and 10% more expensive for decomp_wghtfile compared to the cost on 1600 tasks. It is not surprising the cost increases as tasks increase. The fact that decomp_wghtfile is already so much more expensive than decomp_1d must play a role in the relative extra cost of 3600 vs 1600.
- There is no difference in the cost to initialize the gsmap between MCT 2.8 and MCT 2.10.beta1.
- There is no difference in cost between ceg and orig, as expected, as the map reading algorithm plays no role in this kernel.

2. Reading of the Mapping File (mapfile read)

The table below shows the times for reading the mapping file. As noted above, decomp_1d results in a simpler mapping decomposition compared to decomp_wghtfile. In addition, to determine the decomp_wghtfile decomposition, the weight file has to be read separately to create the mapping decomposition. Once the mapping decomposition is created, the weights are assigned to tasks for use in the sparse matrix multiply. In the reading kernel, the file is read on the root processor in chunks of 100,000 weights at a time. In the orig method, the mapping weights are broadcast to all tasks then each task stores just the weights that it uses. In ceg, the root task figures out which task owns each weight, then a series of send/recvs are setup to transfer weights to each task. The kernel involves both I/O and MPI, but the I/O cost should not fundamentally change in different runs for different cases, except due to run-to-run variability as that algorithm is fixed.

tasks per component	setting	mapfile read (s), MCT 2.8	mapfile read (s), MCT 2.10.beta1
1600	1d, ceg	1.3385	1.3844
1600	1d, orig	1.0869	1.3264
1600	wghtfile, ceg	2.1584 + 2.1282	2.1321 + 2.1929
1600	wghtfile, orig	1.2058 + 2.2667	1.1578 + 1.7497
3600	1d, ceg	2.6334	2.5936
3600	1d, orig	2.2354	1.0812
3600	wghtfile, ceg	2.2390 + 1.6841	2.1922 + 2.4315
3600	wghtfile, orig	1.1740 + 1.8700	1.1276 + 1.2980

- The total cost to read and distribute the weights is less than about 5 seconds total for all configurations tested even for the case where the file is read twice.

- There is no clear difference in the cost of a single reading for the decomp_1d vs decomp_wghtfile settings. There is quite a bit of variability in the timing results. But overall, the decomp_wghtfile is always more expensive because the mapping file has to be read twice (once before the mapping decomposition taking into account the source decomposition and once taking into account the mapping decomposition after it is defined) instead of only once for decomp_1d.
- In these cases, orig is up to 2x faster than ceg, but that will ultimately depend on the configuration. When there are lots of tasks, orig might be faster than ceg. When there are fewer tasks and lots of data, ceg should be faster. That trend might exist in the data above, although it's a bit fuzzy.
- There is no difference in timing for reading the mapping file for different MCT versions. This is expected, as MCT is not used for the mapfile read.

3. Sparse Matrix Initialization (sminit)

The sparse matrix initialization happens after the weights are read and distributed. At that point, information about the source and mapping decomposition and the weights is fed into the MCT sparse matrix initialization where the MCT routers are created that will efficiently rearrange data during the Sparse Matrix Multiply. The Sparse Matrix Initialization is therefore largely an MCT router initialization.

tasks per component	setting	sminit time (s), MCT 2.8	sminit time (s), MCT 2.10.beta1
1600	1d, ceg	0.2333	0.0509
1600	1d, orig	0.2369	0.0776
1600	wghtfile, ceg	19.3781	0.5114
1600	wghtfile, orig	19.3543	0.5454
3600	1d, ceg	0.7105	0.1160
3600	1d, orig	0.9557	0.1022
3600	wghtfile, ceg	41.0135	0.7630
3600	wghtfile, orig	41.0934	0.5840

- The cost of the sminit is less than 1 second in MCT 2.10.beta1 for these configurations.
- There is a significant improvement in the sparse matrix initialization time between MCT 2.8 and MCT 2.10.beta1 . Times were consistently reduced by 3x to 60x. The largest differences are seen with decomp_wghtfile.
- The sminit time is much higher for the decomp_wghtfile case compared to the decomp_1d case. The decomp_wghtfile mapping decomposition is much more complicated with many segments. The number of segments plays a role in the router initialization cost. In MCT 2.8, the cost is greater for the decomp_wghtfile vs decomp_1d by about 50x for 3600 tasks and 80x for 1600 tasks. In MCT 2.10.beta1 , this is reduced to about between 8x and 10x respectively. So both the relative and absolute performance has been improved for the two decompositions used in these tests.

- There is no difference between ceg and orig settings, as expected, as how weights are read plays no role in the sparse matrix initialization.

4. Initialization of the router between the mapping and the target decompositions (router init)

The router initialization in OASIS3-MCT sets up the coupling rearrangement between the mapping decomposition of the target grid on the source tasks and the target decomposition of the target grid on the target tasks. This is similar to a mapping rearrangement between two components with the same grid on two different sets of cores.

tasks per component	setting	router_init time (s), MCT 2.8	router_init time (s), MCT 2.10.beta1
1600	1d, ceg	1.8794	1.5398
1600	1d, orig	1.6447	2.0391
1600	wghtfile, ceg	61.1692	7.5321
1600	wghtfile, orig	59.9303	5.9580
3600	1d, ceg	14.0460	12.4417
3600	1d, orig	3.0845	1.4561
3600	wghtfile, ceg	124.2722	7.5897
3600	wghtfile, orig	122.7938	5.2974

- The cost of router initialization in MCT2.10.beta1 is O(10 seconds) for these cases.
- Router initialization cost is reduced by up to 24x with MCT 2.10.beta1 compared to MCT 2.8. The greatest reduction is with decomp_wghtfile on 3600 tasks.
- The ceg and orig settings should not be playing a role in timing or the router, so the inconsistency in the timing must be associated partly with run-to-run variability.
- If we consider only the orig timings, the cost with MCT 2.8 is ~40x greater for the decomp_wghtfile vs decomp_1d for 3600 tasks and ~60x for 1600 tasks with MCT 2.10.beta1. This is reduced to ~4x for 3600 tasks and ~3x for 1600 tasks.
- These results are similar to the sminit timings. Both are initializing a router, one on overlapping tasks, the other on concurrent tasks.

5. Runtime Sparse Matrix Multiply (avmult) and total coupling time (run loop)

The total coupling time (run loop) in these cases consist primarily of the Sparse Matrix Multiply (avmult) associated with mapping and the rearrangement of coupling data between the two components on two different sets of tasks. The run loop is measured in the models themselves and the first and last coupling period are excluded as these often carry extra cost related to restart files and other initial/final

operations. The Sparse Matrix Multiply is measured inside OASIS3-MCT and does not exclude the first and last steps, but the Sparse Matrix Multiply is barriered to isolate its cost. 100 loops (ping-pongs) were carried out in each run, and the total run times for the entire run is shown in the tables. This data has not been normalized to "per ping-pong". To do so, the run loop times should be divided by 98 which excludes the first and last ping-pong.

tasks per component	setting	avmult (s), MCT 2.8	avmult (s), MCT 2.10.beta1	Run loop (s), MCT 2.8	Run loop (s), MCT 2.10.beta1
1600	1d, ceg	0.0332	0.0432	0.2603	0.2651
1600	1d, orig	0.0337	0.0436	0.2855	0.2541
1600	wghtfile, ceg	0.0261	0.0254	0.2519	0.2341
1600	wghtfile, orig	0.0259	0.0264	0.2634	0.2337
3600	1d, ceg	0.3312	0.1779	0.4503	0.3547
3600	1d, orig	0.2586	0.1820	0.4138	0.3807
3600	wghtfile, ceg	0.0367	0.0273	0.1573	0.1508
3600	wghtfile, orig	0.0274	0.0286	0.1607	0.1525

- The avmult cost is much lower for the decomp_wghtfile cases compared to the decomp_1d cases because decomp_wghtfile minimizes rearrangement. This translates directly into reduced run loop times. With MCT 2.10.beta1, the avmult is faster by 2x on 1600 tasks and by 7x on 3600 tasks for the decomp_wghtfile compared to decomp_1d. The decomp_wghtfile costs more to initialize but the run time performance benefit is significant for these high resolution, high number of core cases, as expected. The avmult represents about 50% of the total run time for the decomp_1d cases, while it's about 5-10% of the total run time for the decomp_wghtfile.
- MCT 2.10.beta1 is generally a little faster than MCT 2.8 in the total runtimes.
- The avmult cost is just about the same between MCT 2.8 and MCT 2.10.beta1 . The largest difference in the avmult cost is for the decomp_1d case at 3600 tasks. The avmult cost is reduced by at least 30% in MCT 2.10.beta1 compared to MCT 2.8 and this translates directly into a reduction of the runtime of about 10-20%.
- There is no difference between ceg and orig as expected.
- With decomp_wghtfile, the runtime is still decreasing from 1600 to 3600 because of the reduced cost of the rearrange in the sparse matrix multiply. With decomp_1d, the run times on 3600 tasks are greater than 1600 tasks.
- There are still a number of minor open questions,
    - The first and last coupling period are excluded in the runtime but not in the avmult, how does that impact conclusions?
    - Why are the 3600 decomp_1d cases so much more expensive than the 1600 decomp_1d cases? This seems to be a robust result. The rearrangement in the decomp_wghtfile case should be minimal, both for mapping and coupling while the cost of rearrangement for decomp_1d is much larger in both. It could be that in the 3600 tasks

case, the interconnect scaling starts to really drop off, and the decomp_1d decomposition and extra rearrangement associated with it result in significant cost penalties. This is not observed at 1600 tasks or with the decomp_wghtfile decomp which minimizes rearrangement.

Summary

MCT2.10beta1 will provide a significant improvement in the initialization performance in OASIS3-MCT_4.0. The total initialization cost can be expected to be O(seconds) per coupling interaction for most configurations, including high resolution cases running on high core counts, even taking into account the more expensive decomp_wghtfile option. More typical moderate configurations will be significantly less expensive that what has been considered in this analysis. In addition, the decomp_wghtfile option will provide a measurable improvement in the runtime remapping performance for some configurations. Compared to the results in Valcke et al. (2017), Oasis3-MCT_4.0 has upgrades that both improve the initialization cost and the runtime cost in significant ways, and those should benefit all users.

REFERENCES

S. Valcke, G. Jonville, R. Ford, M. Hobson, A. Porter and G. Riley (2017), Report on benchmark suite for evaluation of coupling strategies, UMR 5318 CECI, CERFACS/CNRS, TR-CMGC-17-87, Toulouse, France (http://cerfacs.fr/wp-content/uploads/2017/05/GLOBC-TR-IS-ENES2_D10.3_MAI2017.pdf)

A. Craig, S. Valcke, L. Coquart, 2017: Development and performance of a new version of the OASIS coupler, OASIS3-MCT_3.0, Geosci. Model Dev., 10, 3297-3308, https://doi.org/10.5194/gmd-10-3297-2017, 2017.