Geoscientific Model Development Discussions

Geosci. Model Dev. Discuss., referee comment RC3
https://doi.org/10.5194/gmd-2021-183-RC3, 2021

Comment on gmd-2021-183

Anonymous Referee #3

Referee comment on "Parallel gridded simulation framework for DSSAT-CSM (version 4.7.5.21) using MPI and NetCDF" by Phillip D. Alderman, Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2021-183-RC3, 2021

General Comments

This manuscript presents an approach to gridded crop modeling by incorporating two major capabilities into the DSSAT framework itself: the use of netCDF files to hold large lists of inputs and outputs, and the use of MPI to facilitate parallelization. The availability of the code and its incorporation into the larger DSSAT development process is another valuable aspect of this work.

As the saying goes, "there is more than one way to skin a cat" and this work demonstrates an effective way to do what it does. For example, other major gridded systems (like pSIMS or the Mink/IMPACT systems), in addition to moving climate/weather data around, actively adjust fertilizer or irrigation applications based on the phenology observed at each location, but still have parts passing through cumbersome text files. So, there will always be the question of the appropriate boundaries for what is inside the system proper and what should be done in pre-/post-processing steps (and what formats are most convenient for file storage). And for any system of this sort, the most important speedup is if it helps the researcher structure their work in a way that makes it easy to think about and identify, correct, and hopefully avoid errors.

My only major negative criticism regards the extent of the speedup and whether it is game changing or not: my impression is that there is a moderately small improvement in speed that makes gridded crop modeling slightly more convenient. However, as pointed out in the previous paragraph, the major gain is the move from having no system at all to having one that works well for the researcher who needs to use it.

Specific Comments

For pure speedup (as opposed to ease of use), the issue being addressed here is primarily about passing information between the various bits of DSSAT using memory rather than the traditional clunky-but-effective use of text files (and their implicit rounding and precision adjustments). I do not doubt that there is an improvement, it is merely the size of the gains that I am suspicious of.

For example, we should be able to obtain at least part of the potential speedup simply by putting the working directory of the DSSAT instance on a RAM-disk instead of a traditional hard-drive. And in this day and age, that may already be the case for the general storage on the computers used. From my experience, there is sometimes a little improvement to be had by using a RAM-disk, but not a whole lot and sometimes the performance is identical.

The second reason I am suspicious of the speedup comparison with the pure text file approach is that it is mediated by R. Back in the early days of this sort of work, I developed a prototype gridded system cobbled together using shell scripts, grep, sed, and so on. As you can imagine, it was abysmally slow. Porting all the non-DSSAT portions to a "real" programming language resulted in a massive improvement because I was starting from such a low base. If R is slow to prepare the inputs, the comparison may be detecting R's attributes rather than DSSAT's.

The simple way to assess the speedup is to put timers around different parts of DSSAT proper. Then, it would be possible to directly measure how much time is spent reading inputs, writing and reading the intermediate files, running the actual "model", writing the outputs, and so on.

Are there any advantages from using MPI versus scripts to distribute a myriad of small pieces across many computers or even dumping a large number of small jobs onto a queuing system and letting it farm them out? Depending on how much work needs to be done overall and the intra-run communication strategy, the overhead of setting up the working directories and any reporting between the parent and child processes can be quite substantial. So, does MPI offer an advantage here over a more custom approach to minimizing such overhead?

Perhaps, it should be noted for any unwary readers, that any parallelization scheme is necessarily tied to the conceptual structure of the problem to be solved, the physical structure of the computer to be used, and (to a lesser extent) its operating system. Hence, whenever you embark on parallelization, you retreat from portability (as alluded to on page 10, line 10). At this level, the important thing is that the program gets the job done for what needs to be done. Portability would be nice, but no matter what, it will still take a fair amount of effort to adapt it to a random collection of computers and teach someone else how to use it.