Geosci. Model Dev. Discuss., 4, C110–C112, 2011
www.geosci-model-dev-discuss.net/4/C110/2011/

**Geoscientific
Model Development
Discussions**

Interactive comment on "Influence of the compiler on multi-CPU performance of WRFv3" *by* T. Langkamp

T. Langkamp

thomas.langkamp@uni-hamburg.de

Received and published: 5 April 2011

Dear Julian, first of all, thank you for starting the interactive discussion and also for the constructive comments (coming from an DKRZ expert on HPC benchmarking).

In your first point you propose to measure the time in the program only for the computation (without communication). That would be a nice enhancement, because it would reduce the overall benchmarking effort in terms of needed repetitions. Instead of 45 time steps and finding the minimum computation time, just the computation of a hand full of time steps would have been sufficient. However, I think the main result of 26 % performance gain for the Intel compiler would have been almost the same, because this was recorded for 8 cores, equal to one node, which is not affected by the latencies of node-interconnects, but by the node-internal interconnects, which usually are fast

enough not to slow down the computation significantly. Is this correct? Nevertheless, I would like to use a program that measures the computation time only and will look for one.

Your second point was to use VampirTrace also to prove my other assumptions. As I learn from the Open MPI web page (http://www.open-mpi.org/faq/?category=vampirtrace) VampirTrace already is integrated in Open MPI. I wonder if the administrators of Tornado know about this, because they never told me. I would like to use VampirTrace for future evaluations, because to re-run the evaluations done for this paper would mean to recompile all 12 WRF installations again, which is very time-consuming. (Note: The web page above also mentions that VampirTrace itself produces an overhead of 0.x% - 5%.)

Your third point was to measure FLOP/s and other hardware metrics. However, if I understand your explanations on FLOP/s correctly, this would be useful to understand compiler inefïňĄciencies on specific hardware and also to compare different hardware / HPCs. The latter is not as interesting, as I argued in Sect. 1.1: The research goal was to compare different software as the compiler, because hardware-related benchmarks are already available in huge numbers on John Michalakes WRF benchmark page www.mmm.ucar.edu/wrf/WG2/benchv3. Thus this leaves measuring FLOP/s to generate useful results for compiler programmers, which always like to optimize their compilers. I will look into it for future evaluations.

Your fourth point was to try out the proïňĄle guided optimization for GCC. I definitely will have a look into it. After a first glance on this topic, it doesn't seem difficult to implement it. I only wonder if it works with such a complex benchmark as WRF is. I am also not familiar with the term "C102" you mentioned?

Your fifth point was to integrate an independent metric (like FLOP/s?) within WRF in the long run. As I am not a programmer I would like to collaborate with you on this matter. It would be of great benefit to get rid of the grid size dependency.

Thank you so far for your detailed comments. I have learned a lot! Yours sincerely, Thomas

---

Interactive comment on Geosci. Model Dev. Discuss., 4, 547, 2011.