Geosci. Model Dev. Discuss., 6, C2056–C2061, 2013
www.geosci-model-dev-discuss.net/6/C2056/2013/

Geoscientific Model Development Discussions
Open Access

Interactive comment on "A distributed computing approach to improve the performance of the Parallel Ocean Program (v2.1)" *by* B. van Werkhoven et al.

B. van Werkhoven et al.

ben@cs.vu.nl

Received and published: 9 December 2013

MS-No.: gmdd-6-4705-2013
Version: First Revision
Title: A distributed computing approach to improve the performance of the Parallel Ocean Program (v2.1)
Author(s): Van Werkhoven et al.

Point by point reply to reviewer #1

9 December 2013

We thank the reviewer for his/her careful reading and for the useful comments on the manuscript.

1. *In this paper two methods for improving the performance of the Parallel Ocean Program are presented and evaluated. The first method consists of the application of a hierarchical partitioning scheme to improve the domain decomposition of the model. The second method uses GPUs to improve the performance of some parts of the model. In addition to tests on a single cluster, configuration are run, for which the model processes of a run are distributed across computing nodes of two different clusters, which are connected via a relatively slow network connection. Due to the hierarchical nature of the partitioning scheme, it allows the reduction of communication between different sets of processes. This is especially beneficial in this kind of configuration. The paper describes well structured and in high detail the work that has been done. Improving model performance is always an important topic. Doing this by changing the load balancing scheme and by using accelerator hardware are common approaches[1]. Therefore, the results of the paper present no novelty to the general modelling community. However, the paper shows that these approaches can be successfully applied to the Parallel Ocean Program. The description of how the accelerator hardware was utilised is well written and can be a good guideline for similar attempts.*

   [1]: J. Michalakes and M. Vachharajani, "GPU acceleration of numerical weather prediction." in IPDPS. IEEE, 2008, pp. 1-7.

   We thank the reviewer for the positive assessment. No changes in the text.

2. *Some reference to or comparison with similar work done in the area of load balancing and GPGPU in climate modelling would be nice.*

This will be added in the revised introduction of the paper.

3. *Regarding the hierarchical partitioning scheme, see comment by Ilja Honkonen*

   This remark is addressed in our response to Dr. Honkonen's comment.

4. *Regarding Fig. 8a. Could you add the measurements for the CPU?*

   Figure 8 is included to evaluate the performance of the different GPU implementations. Including the CPU execution times would compare the run time of the individual functions on the CPU with those on the GPU. We did not intend to make such CPU versus GPU comparisons on the level of individual functions, because these tend to be unfair in general. Without a complicated modification, the function on the CPU would only use one CPU core, while the GPU function is able to use the entire chip. No changes in the text.

5. *Do these measurements include the time spend in functions called by these functions?*

   Yes, the measurements in Figure 8 include time spent in subfunctions. That is because buoydiff and ddmix only call state as a subfunction. In these specific cases the calls to state are integrated in the code of buoydiff and ddmix in order to optimize the data access patterns. In the revised manuscript we will include a short explanation to clarify this point.

6. *Could you add measurements for the original POP version?*

   The CPU-only results shown in Figure 9 are obtained with the original POP code but with an improved load balancing strategy. Figure 7 shows the performance of the CPU-only POP with the different load balancing strategies. The results in this figure show, as explained in Section 3.3, that space filling curve was the best already implemented strategy. Our hierarchical strategy, however, performs better especially in a wide-area setting. For Figure 9, we therefore take the original POP version with the hierarchical load balancing as the best representative of the original CPU-only POP to compare it to the altered GPU-enabled POP. Although we could add more bars to Figure 9 (each representing a CPU-only POP using a different load balancing strategy), we think this would not add any information that can not already be found in Figure 7. In the revised manuscript this issue will be clarified.

7. *Could you document how many sample measurements you did for your performance results.*

   For the GPU measurements the results show the average of five runs. This is may seem very low, but the performance on GPUs is very stable so the variance between different runs is minimal. In the revised version of the manuscript we will include descriptions of how many measurements were performed when discussing the results.

8. *A figure showing the differences for your three versions (Explicit, Implicit and Stream) might help understanding the methods.*

In the revised version such a schematic will be added.

9. *For someone interested in porting other models to CUDA it might be interesting to hear about the effort required for the different steps.*

   The code for the different GPU implementations of each function is very different. As such, it can be a large effort if programmers need to implement each alternative for each kernel. In the revised version of the manuscript we will add a description of the differences between the various implementations discussed in the paper.

10. *Comparing the performance of the model for the CPU only-run and the CPU+GPU-run is not really fair. One could for example take the hardware costs and/or power consumption into account when interpreting the performance results.*

    The goal of the evaluation in Section 5.2 is to assess whether the two presented approaches are able to work in concert and efficiently utilize more than one GPU cluster. The current comparison is included to evaluate whether the addition of a GPU is at all beneficial for performance. This is certainly not trivial, especially considering that large amounts of data have to be moved back and forth between the different memories over a relatively slow PCI Express link, only a small fraction of the code currently runs on the GPU, and the fact that the GPU is shared between the various CPU cores. When additional elements such as hardware costs and/or power consumption are taken into account it is difficult to make a fair comparison and we consider this to be outside the scope of this paper. However, we will include a short paragraph on this issue in the revised discussion,

based on the following considerations.

Hardware costs depend on much more than just what performance the chip could yield. With each release of a new architecture there is a variety of specific models to choose from, the purchase costs of the different models vary greatly within each architecture. Some architectures are designed for energy efficiency, some for compute performance. This is true for both CPUs and GPUs. To be able to make a fair comparison one would need to take the top performing models for both CPU and GPU from architectures release around the same time. And even then there are many other factors that play part in market prices. For example, recent competitor's releases, commodity prices, material costs, euro/dollar exchange rates, etc.

Regarding power consumption, only a small portion of the Parallel Ocean Program (corresponding to roughly 20% of the execution time) is currently executed on the GPU. If the GPU were as fast as the CPU it would be utilized for only 20% of the total run time of the application. If power consumption is the chosen metric for evaluation it is known beforehand that the GPU will be idle for at least 80% of the time. This is not exactly a fair comparison either. While we agree that it is an interesting question to see if the Parallel Ocean Program would run more efficiently in terms of power on either CPUs, GPUs, or a combination thereof, this comparison can only be made in a truly fair way when the entire program has been ported to the GPU.