Geosci. Model Dev. Discuss., 5, C626–C628, 2012
www.geosci-model-dev-discuss.net/5/C626/2012/

Geoscientific
Model Development
Discussions

Interactive comment on "Porting marine ecosystem model spin-up using transport matrices to GPUs" *by* E. Siewertsen et al.

E. Siewertsen et al.

ts@informatik.uni-kiel.de

Received and published: 3 September 2012

Thank you very much for your comments. We give some answers here, referring to your numbering:

1. In fact, what was meant here was: "Matrix-vector multiplications are predestined for parallelization." This is also true for sparse matrices. And thus we expect a performance gain when porting from a single processor to a GPU. At the end of the paper (end of section 5.2 and Fig. 6), we also compare (the overall performance of the spin-up, not just the matrix-vector-multiplication) to a parallel version on CPU clusters. -> Suggestion for the final version: Change this sentence.

2a. By now, we only have computed the means, we will check if the data are still

available and maybe will be able to provide the standard deviations for the final version of the paper.

2b. Yes, the CPU computations were done on one core, not parallel. -> Suggestion for the final version: This fact will be explicitly noted in the text.

2c. (i) We explain the high performance gain for the MatCopy, MatScale, MatAXPY routines by the fact that all input matrices for these routines have the same sparsity pattern, and thus these three routines - since the matrices are stored in CSR format - are just operations element-wise on vectors, which is ideal for parallelization and thus for the GPU. The memory bandwidth on the GPU is much higher than the on the CPU. The data on peak performance and memory bandwidth utilization have not been generated yet, we will try to provide them for a final version. (ii) The source-minus-sink terms are mainly local (i.e., point-wise) operations. The code is now available on https://github.com/jpicau/model. By now, we have no explanation why the speed-up for the coupled model is higher than for the simple I-Cs model. -> Suggestion for the final version: The link to the code will be added. We do not think it make sense to include the code in the paper, still it is too long. We can add a short description, e.g., that it basically consists of two nested for-loops over the vertical grid points (for the N_DOP model).

3. The number 20 as factor of the performance gain was based on experience from other computations (in a total different field) on GPUs, and there it was the lower limit. It is also motivated by the results in [N. Bell, M. Garland: Efficient Matrix-Vector multiplication on CUDA. NVIDIA Technical Report NVR-2008-04, http://www.nvidia.com/docs/IO/66889/nvr-2008-004.pdf, Fig. 37], where acceleration factors of up to (!) 20 are showed for unstructured sparse matrices, but for a different card and different (also Xeon) processor, but still with CSR matrix format. Our matrices have "more" structure, but not a perfect one since the depth of the water layers differ depending on the horizontal coordinates. -> We will include this reference in a final version and give an explanation for the words "as expected".

p2180-2184: -> For a final version the typos will be corrected.

p2192-2195: -> We agree that some of the details of PETSc, Thrust, and Cusp can be shortened or omitted. We will do so in the final version.
