Use of general purpose graphical processing units with MODFLOW-2005 (Ground Water Manuscript GW20120518-0099)

Authors: Hughes, J.D., and White, J.T

September 3, 2012

Internal reviews were performed by (1) Dr. Yong Liu a senior research scientist at the National Center for Supercomputing Applications (NCSA) University of Illinois at Urbana-Champaign, (2) Randy Hanson a research hydrologist in the California Water Science Center, and (3) Vevek Bedekar a professional engineer with S.S. Papadopulos & Associates, Inc. Randy Hanson provided a few comments in a review memorandum, which are included and addressed below, and as inline comments in the manuscript pdf; inline comments are addressed in the document titled MODFLOW-GPU_03.jdhResponse.pdf. Internal review comments were helpful and have been addressed in the revised manuscript. Review comments are included (*in italics*) prior to our responses. Revised material in the manuscript are indicated with cyan text if it has not been addressed previously in the responses to journal reviewer comments (which are indicated with blue text).

Yong Liu Comments:

1. *One thing that I would suggest to add is to say a few words about any possible future research (e.g., GPU cluster, Cloud-based GPU resources such as Amazon GPU cluster) and practical usage of GPU solvers for MODFLOW 2005 for real-world use cases (an example real-world use case that would benefit from using this research findings and the GPU solver would be excellent!.*

   We have added a brief discussion of current research being conducted by others of improved parallelization of incomplete factorization preconditioners and there is a discussion of possible future enhancements of the UPCG solver in the last pragraph of the **Conclusions** section.

   We have not added a practical example because we believe the test problems presented represent a synthetic but realistic system and is a good control problem for evaluating the effectiveness of the UPCG solver – and this is the ultimate goal of this paper.

2. *It would be also nice if the authors can indicate where and how to obtain the source code (will it be in public domain just like the MODFLOW source code?).*

   We have included a new section titled **How to Obtain the Software** that includes this information.

3. *It would be nice if you could also provide the python code so that I or someone else can reproduce some of your experiments.*

This will be included in the github repository that is described in the **How to Obtain the Software** section.

Randy Hanson Comments:

1. *In particular the preconditioning seems to be an issue and one is left wondering if the code could be written better or is it simply a hardware issue given the structure of the code. I have made several inline comments that may give more details as to where these issues might require minor additional elaboration or insight based on your experience. Would a graph showing the change in performance while varying the inner and outer iterations for one of the models provide any additional insight on how to best set the combination of these two, or is this too problem dependent?*

   Undoubtedly, there is room for improving the efficiency of the UPCG solver code (and probably any experimental code) but we believe the issues related to parallelization of the Jacobi, GLSPOLY, and MILU0 preconditioners would still apply even if the code were more efficient. However having said that, we believe there is some problem dependence to the optimal number of Picard and inner iterations for a given problem. We set up the test problem so that we maximized the amount of time spent in the linear solver to maximize the conjugates evaluated and allowed a large number of Picard iterations to ensure convergence was achieved for most simulations.

   See responses to inline comments in MODFLOW-GPU_03.jdhResponse.pdf.

2. *The reader should also have a clear idea of how he could implement these choices and which choice is best for the problem at hand to some degree (although this may be outside the scope of this paper).*

   We have included our best attempt at guidance based on our simulations for effectively using the various preconditioners and/or hardware options in the **Discussion** section.

3. *The reader may also wonder if additional compilation features such as provided by Intel compiler for memory alignment, vectorization, use of coarrays, use of math-kernal library vectorized functions, or chip specific architectures could provide additional performance enhancements.*

   We haven't seen significant improvements with chip specific compiler switches, MKL libraries (beyond improvements related to OpenMP), data alignment, or vectorization (the UPCG solve is a vector solver that uses CRS indirect indexing). Co-arrays may be beneficial for cluster applications but probably not as beneficial for multi–core CPUs because of the limited number of available threads. A cluster solution is something that we are considering but would be less accessible to many readers of Ground Water. We suspect that parallel CPU and GPGPU solutions would be more accessible to readers.

4. *Inline comments in pdf MODFLOW-GPU_03.pdf.*

   See responses in MODFLOW-GPU_03.jdhResponse.pdf.

Vevek Bedekar Comments:

1. *Specify whether this work is first of its kind in applying GPUs in the field of groundwater modeling or specifically in the application of MODFLOW. Otherwise, cite other works that apply GPUs to solve groundwater simulations. The following two search results were found as a result of a quick Google search.*

   Ji, Xiaohui and Cheng, Tangpei and Wang, Qun, 2010, A simulation of large-scale groundwater flow on CUDA-enabled GPUs: Proceedings of the 2010 ACM Symposium on Applied Computing, 2402–2403. doi: 10.1145/1774088.1774588

   Ji, Xiaohui and Cheng, Tangpei and Wang, Qun, 2010, CUDA-based solver for large-scale groundwater flow simulation: Engineering with Computers, 28(1), 13–19. doi: 10.1007/s00366-011-0213-2

   The second reference (Ji *et al.*, 2012) is a reference in the manuscript but this is difficult to determine from the link provided. The first reference (Ji *et al.*, 2010) was not included in the original manuscript since it is a conference proceeding and may be difficult for some readers to locate. It has been added to ensure Ji *et al.*, 2010 is credited as being the first application of solving MODFLOW using GPGPUs.

2. *Define the term speedup in the paper. Speedup is used in many different ways: solver speedup, speedup of parallelized loops, or speedup of total simulation time. Since simulation efficiency is at the core of this paper, it will be useful to mention what the calculated speedups are based on.*

   The definition of speedup has been clarified to indicate that it is based solely on the time spent within solver routines.

3. *Are there any drawbacks to using GPUs and how these compare to using CPUs: limitations on the size of the problem; a general cost comparison (twice/thrice) etc.*

   Information on the RAM limitations of the specific GPGPUs is given in the **Test Cases** and **Conclusions** sections. A summary under which use of parallel GPGPU and CPU approaches should be beneficial has been added to the **Discussion** section.

4. *Page 9 mentions that two different machines were used (2.4 GHz and 3 GHz) for comparing the CPU and GPU simulations. Were comparisons made between the two machines by running exactly same simulation runs on the two CPUs (without multi-threading and by using 2 cores) to benchmark whether there are inherent differences in the performance of the two machines? Besides the processor speed other factors like bus speed, motherboard, hard drives, etc. could be impacting the results.*

   Serial and parallel runs were made on both machines so that the correct speedup could be calculated on each machine. Additional text has been added to clarify the approach used for model simulations on both machines.

5. *Did the authors study if a relationship existed between the total number of inner iterations and the speedups in the test simulations? If such a comparison was made, it may*

be interesting to share that with the readers. That would be an indication to the users as to which problems may reap the benefits of GPU more than others.

Yes there is a relationship between inner iterations and speedup. The runtime information shown in **figure 4** tells part of the story but marginal to poor speedups for the Jacobi preconditioner is related to the number of iterations required for convergence.

A new table has been added (**table 1** in the revised manuscript) that summarizes the number of Picard and inner iterations used by each solver and preconditioner evaluated. Additional text that discusses the relationship between the strength of the preconditioner and the observed speedup has been added to the **Discussion** section. We believe this will provide readers with some guidance on when to try parallel GPGPU or CPU approaches to a given problem.

6. *Was the total number of inner and outer iterations compared between the CPU and GPU simulations? The number should come out to be very similar in all simulations since the only thing GPU is doing is to parallelize the same computations that CPU is doing. This comparison would provide more confidence in the results that no other processes/algorithms besides GPU and its parallelization are responsible for the speedups.*

   A new table has been added (**table 1** in the revised manuscript) that summarizes the number of Picard and inner iterations used by each solver and preconditioner evaluated. Additional text has been added to the **Results** section to summarize the data included on the new table.

7. *There are more robust and faster solvers available besides PCG2 (XMD of MODFLOWNWT; PCG5 of MODFLOWSURFACT). It would be interesting to see how the GPU speedup (of 8 when compared to PCG2), compares with faster algorithms used in other solvers like XMD and PCG5.*

   There are certainly other solvers available that have been incorporated into MODFLOW or MODFLOW-variants by others but we have intentionally limited our discussion to standard solvers available for MODFLOW-2005. This study was meant to be an evaluation of the applicability of GPGPU parallelization to a PCG solver not all solvers. The findings of this study should be applicable to other solvers and could be implemented in a similar fashion. It is our expectation that speedups for the solvers mentioned would be similar to the UPCG solver with the MILU0 preconditioner because of the difficulty of parallelizing an incomplete factorization.

   It is our assessment that it is the parallel GLSPOLY preconditioner that makes use of the GPGPU useful for MODFLOW at this point. Furthermore, we expect that future research on ways to parallelize incomplete factorization preconditioners will greatly improve the speedup for parallel solvers on CPUs and GPGPUs.

8. *The LMG2 (SAMG) solver is a parallel solver that claims speedups of up to 100 times when compared to the PCG solver (*`http://www.scai.fraunhofer.de/fileadmin/download/samg/paper/modflow11.pdf`*;* `http://www.scai.fraunhofer.de/fileadmin/download/samg/paper/Modflow_Paper.pdf`*). The authors may wish to comment about the combination of the GPU technology and the*

SAMG solver.

The authors have considered parallelization of a multi-grid solver but have not performed any evaluation of the effectiveness of parallelization of a multi-grid solver relative to a traditional Krylov solver. Parallel multi-grid solvers are reported to be more scalable than traditional Krylov solvers. To keep the paper on topic - comparison of parallel strategies on the UPCG solver with speedups based on the PCG solver - we have elected not to add a discussion of parallelization of the LMG2 solver on GPGPUs.

9. *How does PCG and UPCG sequential simulation compare?*

   PCG and UPCG are competitive. For some problems PCG is slightly faster for other problem UPCG is slightly faster. Iteration information is summarized on **table 1** in the revised manuscript. In general, the performance of PCG suffers with larger problems and is summarized in the **Results** section.

10. *Mention that the groundwater simulation results are the same irrespective of the solver/processor used.*

    This has been added to the end of the **Problem Description** subsection.

11. *Line 35: replace SOR with SSOR.*

    Corrected

12. *Line 40: also refer to other more efficient solvers, including the XMD, LMG2, and PCG5 solvers.*

    There are certainly other solvers available that have been incorporated into MODFLOW or MODFLOW-variants by others but we have intentionally limited our discussion to standard solvers available for MODFLOW-2005. XMD has been incorporated into MODFLOW-NWT but it's real strength is its ability to solve non-symmetric matrices using more sophisticated level-fill and drop tolerance approaches. Since the test problems we are evaluating are symmetric it is questionable whether these approaches would add convergence improvements that outweigh the additional numerical overhead of these approaches. In my experience XMD is less efficient in simulating standard MODFLOW problems than the PCG and UPCG solvers.

13. *Line 44: it is important to note that speedup in this paper is based on "solver computing time" as defined in the paper. Upon testing, I had found that the loops parallelized took 25–30% of the total simulation time for the problem I was testing. Effectively, 5 times speedup (20%) of 25–30% comes out to 5–6% saving in the total simulation time.*

    This clarification has been added.

14. *Line 92: a appears twice.*

    Corrected – moved to **box 2**.

15. *Lines 161 and 163: is there a citation for CUBLAS and BLAS? If these are acronyms, what do they stand for?*

The acronyms has been defined in the revised text. References for CUBLAS and BLAS have been added.

16. *Line 162: Saad, 2003 is misspelled.*

    Corrected

17. *Line 173: what is GEMV? Is this an acronym?*

    It is an abbreviation for general matrix-vector (GEMV) product. This has been added to the manuscript.

18. *Lines 193 and 196: Xeon is misspelled.*

    Corrected

19. *Lines 251–252: is this treatment of RCLOSE applied to PCG solver as well? If not, then the comparison between PCG and UPCG is unfair since the two solvers would converge to different RCLOSE values.*

    Yes. This has been added.

20. *Line 317: if MIC is more efficient, was parallelization of the MIC preconditioner attempted? Please make a statement as to why not.*

    We did not intend to say anything more than the standard PCG solver is generally fairly efficient rather than something specific about the efficiency of the MIC preconditioner relative to the Jacobi, GLSPOLY, or MILU0 preconditioners available with the UPCG solver. This sentence has been revised to clarify this point.

21. *Table 3: several values in this table are less than 1, indicating that UPCG is slower than PCG. What is the possible reason of this slowdown?*

    Additional clarification has been provided for PCG(MIC)/UPCG-OpenMP speedups less than 1 in the **Discussion** section.