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

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

August 1, 2012

Add some introductory text about how reviewer 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 blue text.

Editor's Comments:

1. *Our layout doesn't require section numbering, so please delete those.*

   Section numbering has been eliminated in the revised manuscript.

Reviewer 1 Comments:

1. *The introduction should also introduce the speedup potential and bottlenecks of GPGPU and parallel CPU processing for the UPCG solver.*

   Response.

2. *Section 2 has too much jargon. I suggest removing what is irrelevant and explaining what is, in terms of how that helps speed up or slow down GPGPU processing compared to serial computing or multi-core processing on CPUs.*

   Response.

3. *The manuscript needs to mention how the solver is available for M2K5 for the GPGPU and multi-core CPU versions (are they DLLs, how are they linked into the main code, etc). I do not expect people to generally compile their codes, let alone within CUDA. Some direction therefore is needed on accessibility.*

   A section titled "How to Obtain the Software" has been added that provides information on how to obtain the software.

4. *Also, direction is needed on hardware that may be best suited for the GPUGPU processing will benefit this manuscript.*

The authors are not experts on the hardware that is best suited for GPGPU processing. Rather than providing guidance on appropriate hardware we feel it is more appropriate for potential users to consult with a qualified software dealer and identify the hardware that is most appropriate for their particular organization.

5. *GPGPU speedup has been compared to the PCG5 solver on a serial CPU. With more advanced solvers coming into the MODFLOW family (for example the XMD solver in MODFLOW-NWT,)it would be interesting to note the speedup of UPCG compared to even more robust higher level ILU solvers which may be even faster than PCG5 for hard problems.*

   The analyses presented in this manuscript are not meant to be an exhaustive comparison of GPGPU solutions but an evaluation of the speedup possible.

6. *Issues with GPGPU processing for different schemes are scattered throughout the manuscript. These issues should be collated into a separate section or a table. For instance: GPGPU memory requirements for larger problems; need to maximize the amount of time spent in the linear solvers; the overhead associated with GPGPU-CPU memory copy operations relative to the solution time; higher percentage of runtime spent transferring preconditioner data; why the total time spent applying the preconditioner actually increases for the MILU0 preconditioner; what is less optimized in Intel FORTRAN for multi-core processing; details on problem size limitation due to RAM available on the GPGPU; and others.*

   Response.

Reviewer 2 Comments:

1. *Overall, lots of jargon is used. Naturally, this is an emerging field, and one that many readers of Ground Water will not be familiar with, so the use of jargon will be difficult for the uninitiated reader. For example, lines 75-79 are filled with terms that will mean nothing to the average groundwater modeler - consider deleting (or moving to appendix or online material) this paragraph and other similar paragraphs that might not hold much meaning to the average reader.*

   Response.

2. *Thorough comparisons were made for different preconditioners and different parallelization strategies and the authors did a great job of making the comparisons fair. Considering the results of the various preconditioners, it appears that Jacobi preconditioning is good enough for many of the problems, which leads me to believe that the K fields are fairly smooth (and the spread of the eigenvalues are nicely clustered). I tend to think of this as the exception in rather than the rule in most practical groundwater models. Can you provide guidance for using MIC vs. POLY vs. Jacobi based on the K field? In other words, I think these results are highly dependent on the K field and only 2 fields are used. How would the results vary if you stretched the contrasts in your K field? In*

that regard, to be fair, the GMG solver should also be used as a basis for comparison and give insight about which strategy should be used for optimal performance.

Response.

3. *There seems to be a tradeoff between parallelizing on the CPU vs. the GPU depending on the sophistication of the preconditioner. What is not clear was if it Open-MP was used on PCG2. Reference is made to the work by to Dong and Li, but it appears that this was not considered. How do these results compare to that of Dong and Li? The intent here is not to say that your results are better/worse than those of Dong and Li. Simply, not everyone has access to a video card with double precision GPU capabilities, whereas multicore CPU's seem to be more common, so some guidance of which strategy might be more effective depending on the user's hardware would be helpful (ie, If I were buying a new workstation, should I focus on getting the best multicore CPU, or should I get a less powerful CPU and spend money on a high-end GPU?) I understand that the answer to this question is not straightforward, but any additional insights you could provide regarding this tradeoff would be helpful.*

Response.

4. *Line 26 - "can be *easily* parallelize"*

Response.

5. *Lines 28 - 34 - there is some time discrepancies here from 2005 to 1984, to 1988. Perhaps just say "MODFLOW" on line 28 and remove the "since 1984" and insert "numerous" before "two-and"*

Response.

6. *Line 47 "present" (plural)*

Response.

7. *Line 48 - unclear - faster at what exactly? Operations,cycles/second?*

Response.

8. *Lines 75-108 is filled with a lot of technical jargon. Consider moving some of this to an appendix (as noted above).*

Response.

9. *Line 108 - really, the parallelization you mention can be applied to many simulators, not just MODFLOW-2005. Consider mentioning wider range applicability.*

Response.

10. *Line 213-214 - unclear. Consider rephrasing*

Response.

11. *Line 351 - "developed to *use* the PCG" (one does not "solve" the algorithm)*

Response.

12. *Line376 - as indicated above, can you provide some guidance here about when one approach might be better than another depending on the system properties (size, K field). Also, GMG should be included here to so that the full suite of approaches are compared (weak but highly parallelizable to strong but not very parallelizable).*

Response.

13. *Figure 4 - I think this figure is extremely important as it shows where the speed ups occur. However, it is not clear how much time is spent in the GPU for the matrix-vector and BLAS operations because the bars seem to not show up on the scale? Can you include a value here so the reader does not assume these are zero.*

Response.

14. *In light of the speed ups coming from CUBLAS operations being parallelized over the cells in the model, I am wondering if additional speed up could be obtained by using other primitives in the CUSparse library for the triangular solves and/or the decomposition? Was this considered as well? There is some new work that tries to parallelize ILU (see: http://research.nvidia.com/publication/parallel-incomplete-lu-and-cholesky-factorization-preconditioned-iterative-methods-gpu)*

Response.

15. *As indicated above lines 75-108 are filled with jargon and should be reconsidered. Response.*

Response.

Reviewer 3 Comments:

1. *Quality of figures should be improved (all figures but in particular, figures 3 to 6). Texts (titles and legends) are too small to read. Lines are very hard to read.*

Response.

2. *Tables should be reworked. I do not think we need 3 digits after decimal point for these comparisons. Please also consider significant digits in the tables.*

Response.

3. *The authors only consider total execution time which includes time required for both Picard and linear solver iterations. Did you closely examined these iteration performance in solver comparisons? How these iteration performance changes among different preconditioning schemes?*

Response.

4. *Lines 28-32: the authors mentioned computational performance changes due to domain discretization. Does this cause by FD formulation or matrix solver scheme? Explanation is needed.*

   Response.

5. *Line 48: 448 → 448 cores.*

   Response.

6. *Line 114: is L3/L3T correct?*

   Response.

7. *Line 123: is L2/T correct?*

   Response.

8. *Line 161: What is CUBLAS?*

   Response.

9. *Line 199: Problem Description. Reasoning for choosing test cases is helpful for the readers.*

   Response.

10. *Line 236: "(10-0)/1000"is not necessary.*

    Response.