Cover letter

Dear Editor,

Thank you very much for the reviews of our paper (manuscript:ID IJGIS-2013-0120 entitled "A GRASS GIS parallel module for radio-propagation predictions") that we submitted as a publication candidate to IJGIS. The review comments have been extremely useful to us and we are very grateful for the feedback given by the reviewers as it helped us produce a more solid and complete manuscript.

We acknowledge english is not our native languages, for this reason we had the text reviewed by ...

Please find below the answers and detailed information about the changes to the manuscript that we made in response to each of the reviewers' comments.

Answers to reviewers' comments

Reviewer 1

1. In the last sentence of the abstract, "which sizes are . . . " should be "whose sizes. . . ".

No problem ...

2. Evaluating the existing researches on utilizing HPC to solve Geo-spatial problems, the authors said, ". . . most of these works are application-specific, and do not introduce general principles for jointing GIS and HPC." I quite disagree to such point. From this manuscript, I can found:

2.1 The topic of this research is also application-specific, and the parallelism for such radio-propagation prediction module is not complicated, namely, the workers just do some pure computations, has nothing with the GRASS environment. The process is that the workers do the step of calculating each transmitter path-loss respectively and mutually exclusive, and then write the intermediate results to the external DB. Thus, the parallel module implementation can fully use the characteristic according to the transmitter assignment independently.

The reviewer claims that the radio-propagation prediction module is not complicated. We will argue this is not the case and provide the computational-complexity analysis of the radio-propagation prediction algorithm as evidence to support our assertion. Moreover, that the parallel implementation abstracts GRASS from its computation is a feature of our implementation that to overcome the limitations of GRASS in regard of concurrent access to spatial data. Additionally, writing the intermediate results to an external DB is also a feature of the presented approach, that considerably improves the scalability of the presented implementation when several concurrent processes are involved. On the other hand, spatial-data independence is a property of the problem itself, which we exploit by applying block-data distribution, in as similar way as in [ref!*Optimal tilt and orientation maps: a multi-algorithm approach for heterogeneous multicore-GPU systems*, *High-performance three-horizon composition*

algorithm for large-scale terrains].

2.2 Such methodology to make the GRASS GIS module parallel has appeared in the paper titled "Explorations of the implementation of a parallel IDW interpolation algorithm in a Linux cluster-based parallel GIS" (published on C&G). In that research, the slave nodes also just do their own pure computations assigned from the master node. Therefore, the author's discussion on that research is also problematic.

This work also abstracts the GRASS data types into its own struct and MPI data types, thus not needing GRASS in the worker nodes. Data is evenly divided by row among the workers, with each one receiving an exclusive column extent on which to work. The test cluster contains heterogenous hardware configurations. About this point, we argue that evenly distributing the data among the worker does not maximize the parallelization gain, since the slowest node in the cluster is an upper bound of the achieved speed-up and efficiency (i.e. speed-up/num. of parallel processes) . The authors also note that other GRASS modules can be adapted to parallel versions using the presented procedure. Regarding the data sets during the simulations, the largest one contains 3,265,110 points, which is almost 20 times smaller than the data sets used by PRATO. The authors note that the data-set size is bounded by the amount of memory on each of the nodes, since they allocate memory for the whole map as part of the set-up stage, before starting the calculation. It follows that memory consumption is an issue with this approach. Indeed, their *Point* struct contains four double variables for each pixel of the loaded map. Consequently, the amount of required memory is a priori 4 times bigger than the loaded data set. In our case, only one double variable per pixel is used, with only one 2D geographical offset per loaded matrix. It is not clear what *npoints* is. Their conclusion is that the data-set size should be large enough for the communication overhead to be hidden by the calculation time, so that parallelization pays off.

3. In paragraph 4 "... radio-coverage prediction is a computationally-intensive and time-consuming task, hence the importance of using fast and...", you'd better provide the quantitative data.

To demonstrate that solving the radio-coverage prediction is a computational intensive problem, we have added a section discussing the computational complexity of the problem.

4. In Figure 4, it is better to present the raster map involved more transmitters;

We have added a raster map that includes X transmiters, this is Y times more than the original manuscript.

5. In Section 3.3, to the parallel module design and implementation, I have some questions:

5.1 In order to avoid the workers' concurrent writing operations, the research adopt the approach of making the workers' partial coverage evaluation results writing into the external database. After that, the master will reread the intermediate results and then give the aggregation output. In this procedure, it is clever to avoid the bottleneck to adopt this way; however, the efficiency of the module will be affected by the repeat and needless outputting and inputting;

Writing the partial coverage evaluation results to an external DB avoids overloading the master process, thus making better use of the available resources. This point is clarified with an extra set of experiments, comparing the advantanges of using/not using the external DB for saving intermediate results as well as the final aggregation of the radio-coverage prediction. Ideally, the time spent during each algorithm step should be discriminated in order to find out if the time contribution using the DB is significant. Moreover, we argue that in a traditional master-worker approach, the master process suffers execution overload, when several requests of the worker processes. It follows that the efficiency degradation of the parallel implementation is directly proportional to the capacity of the master process to cope with multiple worker request. By using an external DB, we avoid such situation. Specifically, we have clarified the running time of different problem instances using a traditional master-worker approach and a master-worker-DB alternative.

5.2 Meanwhile, the adopted task distribution method may obtain low overhead for the heterogeneous cluster system, i.e., different computing tasks will assigned to different workers according with their hardware configurations. However, to the homogeneous cluster, as each node has almost the same hardware, to adopt such approach, the communication between the master and the workers will be intensive, which will also affect the efficiency of the parallel module. Under this circumstance, I think it's better to make all the transmitters' coverage evaluation calculation distribute to all node evenly.

About this point, we argue that evenly distributing the data among the worker processes has several drawbacks:

(a) when deployed over heterogenous hardware, this approach does not maximize the parallelization gain, since the slowest node in the cluster is an upper bound of the achieved speed-up and efficiency (i.e. speed-up/num. of parallel processes) ;
(b) in practice, it is rare to find a high number of dedicated computing nodes featuring the exact same configuration, i.e. not every organization has a computer cluster; consequently working on methods and techniques supporting heterogenous hardware configurations, exploitable over a typical LAN, is, from the practical point of view, a far more useful approach in order to fully exploit available computing resources;
(c) the overhead representing the partial-data distribution is neglectable when the number of processing worker are above a given threshold.

Moreover, it is often the case for university departments and small companies to have small computer clusters that grow organically, as every year new computers with different memory and CPU configurations are added.

6. Please cite Neteler and Mitasova 2008 (3rd edition), the second edition is obsolete.

Changed citation to the 3rd edition.

7. Some trivial problems relate to spelling, formatting etc. e.g., "geosciences" in the references.

Huang, F., et al., 2011., changed journal name from "Computers & geosciences" to "Computers & Geosciences".

Reviewer 2

1. The calculation of radio-coverage prediction is not well described. The description of LOS and NLOS is separated from other parts of the coverage prediction, which is confusing. I had to constantly go back and forth to put all information together in order to understand the process. I'd suggest the author put them into one section. Also, some figures would be helpful to explain the process. For example, how to determine if a receiver is within LOS? how is antenna influence calculated?

No problem ...

2. The parallel algorithm presented in this paper uses some techniques that have been commonly used in general-purpose parallel computing, e.g., master-worker configuration, task-farm dynamic load-balancing. Were there any innovative techniques created for this particular analysis? Or, were there any innovative techniques developed for spatial analysis in general? If not, the lack of innovation will greatly degrade the scientific significance of this work.

We argue that saving the intermediate calculation results in an external DB from an independent thread of the worker process, in a parallel and asynchronous way, is a novel approach that improves the efficiency of the parallel implementation when several processes are involved. Also, using the same DB to aggregate the intermediate results before dispatching them to the master process for creating the final output is also an innovative technique. To the best of our knowledge, none of these techniques have yet been used in other GIS-related work.

3. All experiments in this paper were conducted using man-made transmitter data. Is it because the real-world data is not available, or because the real-world data is not big enough for this parallel program to demonstrate the speed-up? I'd like to see some experiment results using real data to show the practical value of this parallel algorithm.

Real data that is used in practice by radio-propagation engineers at telecom companies is generaly not readily available. This data is special as it contains data of transmitters configuration, antena placement, and relevant terrain parameters for the application. However, Telekom Slovenije, d.d., has given us access to a real data set of more than 2000 transmitters. Nevertheless, this is only a small to medium set so in order to test PRATO for larger networks we haver resorted to synthetically create them at ...

In Telekom Slovenije, d.d., we have real data of more than 2,000 transmitters at our disposal. Since the experimental data sets contain many times this number, we have to synthentically create them by randomly replicating and distributing the real transmitters. Notice that the number of deployed transmitters is directly influenced by two factors. First, the population density over a geographical area, and second, the total area of the country. In this regard, the relatively small population of Slovenia (around 2,000,000 people), and its small surface are the reason behind the real-data volume we have available. However, in other countries and regions, e.g. Great London as noted in the paper, transmitter count greater than 10,000 are very common. For a transmitter count of

different mobile network technologies, we refer the reader to ...

4. Section 5 "related work" should be put in the introduction section.

No problem ...

Reviewer 3

1. The contribution of this paper is not novel since parallelizing existent sequential modules using the data-parallel model in general and task-farming in special has already been tackled by several previous works. Besides, there exist several high-level tools that allow implementing serial modules with a specific patterns or skeletons. I would recommend including a brief review to such tools and explaining the reason why the authors implemented this module manually.

The authors' experience on the area of HPC allowed us to implement the application using MPI and POSIX threads directly. This includes several in-house developed tools and libraries which accelerated the development of the presented implementation significantly. Consequently, for the authors to learn a new framework would have meant a bigger time investment. Moreover, the fine-grained control needed for the multithreaded version of the workers is not available in all parallelization frameworks. However, as the reviewer notes, there are some parallelization frameworks already available, e.g.:

- PyMW: the authors acknowledge that PyMW is not well suited for computations with frequent communication and sharing of data between workers.
- ROme OpTimistic Simulator (ROOT-Sim): no user's documentation available. Available at http://www.dis.uniroma1.it/~quaglia/software/ROOT-Sim/
- HTCondor MW: a tool for making a master-worker style application that works in the distributed, opportunistic environment of HTCondor. No cons so far ...
- Charm + + is a good candidate framework: http://en.wikipedia.org/wiki/Charm% 2B% 2B * If the dynamic component of the application is the limit of scalability using MPI, using a Charm ++ framework makes sense. In particular, the use of charm + + (or AMPI) to access features such as work / comm overlapping and fault tolerance is now proposed as future work.

2. In the third paragraph in introduction, page 2, the authors cited few related works that use HPC but didn't specify which parallel model, data partition and parallel architectures were used in these works. As the final objective of this manuscript is to find general principals to join GIS and HPC it would be very interesting to provide a complete review of related HPC-GIS works.

We have described the basic parallel principles used by each cited work, in order to briefly review the state-of-the-art techniques in the area of parallel techniques for GIS problem solving. In this context, we have also discussed the advantages and drawbacks of the cited techniques, and how writing the results to an external DB corresponds to an efficiency gain for spatial parallelization.

3. For example, "High-performance three-horizon composition algorithm for large-scale terrains. International Journal of Geographical Information Science 25(4): 541-555 (2011) by Tabik et al." partitions the full resolution DEM by block among cores/processors and assigns a multiple halo of lower resolution to the corresponding processor to eliminate the negative edge effect. It uses work sharing as parallel model for both message-passing and shared memory platforms.

This work uses parallelization within each worker process with OpenMP (not POSIX threads as we do). Moreover, there is one worker process per worker node, and the complete exploitation of the computing resources of the worker node is achieved with OpenMP. The tested data sets are two orders of magnitude bigger than ours, whereas the experimental environment in only one host, consequently not taking into account the communication problems/bottlenecks arising in larger scale HPC. As well as in our work, there is no data dependency among the calculated spatial blocks. Consequently, the DEM may be divided in separate blocks to be independently calculated by each worker process. The authors present an improved algorithm, based on the Stewart's one, that achieve the same accuracy with less arithmetic operations and less memory usage. The presented algorithm can also be used to accelerate other applications like visibility. Similarly as in our paper, tasks are dynamically assigned to idle nodes using a task-farming paradigm over MPI.

4. "Optimal tilt and orientation maps: a multi-algorithm approach for heterogeneous multicore-GPU systems. The Journal of Supercomputing. http://link.springer.com/article/10.10 013-0891-1#page-2 " uses two different algorithms to compute the maximum solar energy maps on hybrid GPU-multicore systems; the first algorithm is the most efficient on GPUs and the second method is the fastest on multicore processors. This work also uses work sharing as parallel strategy and partition the DEM by block among GPUs and CPUs.

As well as in our work, there is no data dependency among the calculated spatial blocks. Consequently, the DEM may be divided in separate blocks to be independently calculated by each worker process. The experimental evaluation is done over multiple cores of one CPU and a GPU, consequently not taking into account the communication problems/bottlenecks arising in larger scale HPC.

Algorithm time complexity

The complexity of the radio-coverage prediction algorithm is depicted in Algorithm 1. The other algorithm listings are given for reference.

To do's

- Look for an alternative International Journal about parallel computing or GIS (non-SCI) that responds as fast as possible. Japanese collegues agreed to cofinance the publication fee.
- Experimental comparison of master-worker and master-worker-DB combo.

Algorithm 1 Pseudo-code of the coverage-prediction algorithm.

$DEM \leftarrow$ Digital Elevation Model (DEM) of the whole area. ▷ $O(M)$
$Clutter \leftarrow$ signal Losses due to land usage of the whole area. ▷ $O(M)$
$T \leftarrow$ transmitter configuration data. ▷ $O(n)$
for all $t \in T$ **do** ▷ $O(n \cdot m^2)$
  $DEM_t \leftarrow$ DEM area within transmission radius of t ▷ $O(m)$
  $Clut_t \leftarrow$ Clutter area within transmission radius t ▷ $O(m)$
  $LoS_t \leftarrow$ LineOfSight (DEM_t) ▷ $O(m^2)$
  $PL_t \leftarrow$ PathLoss $(DEM_t, Clut_t, LoS_t)$ ▷ $O(m^2)$
  $Diag_t \leftarrow$ Antenna diagram of t ▷ $O(1)$
  $PL_t \leftarrow$ AntennaInfluence $(Diag_t, PL_t)$ ▷ $O(m)$
end for
for all $t \in T$ **do** ▷ $O(n \cdot m)$
  $CoveragePrediction \leftarrow$ PathLossAggregation (t, PL_t) ▷ $O(m)$
end for
return $CoveragePrediction$

- Calculate the computational complexity of the radio-coverage prediction algorithm in order to demonstrate the high time-complexity needed to calculate radio predictions for multiple transmitters.
- Change the objective of the paper from general principles to a more educational/didactical approach.
  - data decoupling from GRASS before parallelizing;
  - comparison of master-worker and master-worker-DB combo;
  - work pool (i.e. task farming), add a reference to work pool;
  - experimental tests comprising a higher number of computing nodes and more worker processes;
  - change the relative-time plots with the overhead introduces by our message-passing technique, that enables the use of heterogenous clusters.