

Response to Review from Glenn Hammond

We would like to thank for the very detailed and constructive comments. Based on this review, we overworked the whole manuscript and included the following new contents:

- a new string-based coupling to reduce the computational overhead of the interface
- a more complex example (uranium leaching) to test the parallel performance of our approach
- the according new (updated) results, figures (1, 2, 4, 5, 6, 8, 9, 10, 11, 12, 13) and tables (1, 4, 5)
- supplementary material: PHREEQC script for the Engesgaard benchmark
- supplementary material: model description and simulation results of the new example

We reply to all comments individually below and will also address them where possible in the revised version (marked-up version, line numbers mentioned below refer to this version). Reviewer comments are reproduced in **bold**, with our explanations in *italic*.

1	<p>This manuscript investigates the coupling of OpenGeoSys with IPhreeqc through a file-based data transfer with the use of MPI processor groups to assign additional processes to speed up the geochemical calculation. I applaud all efforts that further the application of high performance computing to subsurface simulation. The manuscript is somewhat novel.</p>
	<p><i>Thank you very much for these encouraging words and short summary of our manuscript.</i></p>
2	<p>However, the degree to which performance is improved by the addition of geochemical processes (i.e. the scalability) is questionable and the file based data transfer is clearly not scalable on large supercomputers. Please see my comments below regarding deficiencies in the current draft of this manuscript.</p>
3	<p><i>Indeed, file based approaches scale poorly and it has been only used for the first proof of concept. In the meantime, we can present the results for the more efficient string-based coupling and we already included them into the revised manuscript. The according figures and descriptions have been updated as well. We address these changes more detailed below (see comment reply 6, 12).</i></p> <p>The abstract states, “The open source scientific software packages OpenGeoSys and IPhreeqc have been coupled, to combine their individual strengths and features to simulate thermohydro-mechanical-chemical coupled processes in porous”. Please elaborate on what makes IPhreeqc better than the existing OGS chemistry capability. This may be obvious to the authors, but not the reader.</p> <p><i>The main advantage of this combination is that we can make use of the wide range of geochemical capabilities and customizable database of PHREEQC which helps us to setup a variety of coupled THMCreact simulations faster on HPC systems.</i></p> <p><i>The parallelization scheme makes this approach especially efficient if we face more complex geochemical systems. We included these points and rewrote the abstract to better explain these advantages to the readers as follows:</i></p> <p><i>“The open source scientific software packages OpenGeoSys and IPhreeqc have been coupled to setup and simulate thermo-hydro-mechanical-chemical coupled processes with simultaneous consideration of aqueous geochemical reactions faster and more straightforwardly on high performance computers. In combination with the elaborated and extendable chemical data base of IPhreeqc, it will be possible to setup a wide range of multi-physics problems with almost any</i></p>

	<p><i>chemical reaction that is known to influence water quality in porous and fractured media. A flexible parallelization scheme using MPI (Message Passing Interface) grouping techniques has been implemented, which allows an optimized allocation of computer resources for the node-wise calculation of chemical reactions on the one hand, and the underlying processes such as for groundwater flow or solute transport on the other hand. This technical paper presents the implementation, verification, and parallelization scheme of the coupling interface, and discusses its performance and precision."</i></p>
4	<p>Line 2373:15 states that this manuscript evaluates a new parallelization scheme to provide "detailed information" for modelers and developers. My observation is that this manuscript often lacks detail. The manuscript could be greatly improved by providing a more detailed description of the implementation along with how the developers dealt with various issues that arose during implementation (so that others attempting to implement the same approach can leverage the authors' knowledge on this topic). This would make the manuscript much more impactful.</p> <p><i>We are providing more details regarding the technical implementations as well as description of benchmarks etc. in the revised manuscript version, in order to facilitate the interested readers to implement this method and reproduce the results we presented (see page 7 - 13). We are also addressing following issues which came up during the implementation of the current scheme.e.g.:</i></p> <ul style="list-style-type: none"> <i>• How to integrate updated IPhreeqc releases (see revision, page: 7:25)?</i> <i>• How to make it possible to run a block of code only for the ranks of a relevant MPI group? (see revision, page 12:27; Figure 6)</i>
5	<p>2372:13 "An elaborated code concept and development can help to reduce the time needed for solution procedures and data communication." Do you mean "a well designed and efficient parallel implementation can help to reduce::"?</p> <p><i>Yes, that is clearer. Thanks! We changed the sentence in the revision as "a well-designed code concept and efficient parallel implementation can help..." (see revision, page 4: 15)</i></p>
6	<p>2372:14 "Consequently in terms of coupled reactive transport modeling, process simulation and interaction should be closely tied to enable shared data structures and reduce data exchange procedures." Do you mean a well-designed interface should be developed that maximizes sharing of data structures in order to avoid duplication of data and/or computationally expensive mapping of data. This approach is expensive in at least two ways: (1) duplication of data structures and thus more memory use and (2) transfer of data between the two data structures (perhaps not that much of an issue if not file-based).</p> <p><i>Yes, that's what we mean and it is indeed much less of an issue for a string based coupling interface, but still it is.</i></p> <p><i>A file-based data transfer is extremely time-consuming of course, especially in a clusters infrastructure in which file reading and writing is realized through the general parallel file system (GPFS). After we found our parallelization concept very promising, we started immediately to implement a string-based coupling between OGS and IPhreeqc in order to avoid this bottleneck. The gain in terms of performance is discussed more detailed below (Response 12, in-memory-coupling). The manuscript has been overworked accordingly (see Sect. 2.3, 3.2, 4.1, 4.2; Figure 1, 5, 6, 8, 9, 10).</i></p>
7	<p>2372:17 – I believe that the "I" in IPhreeqc stands for "interface". This should be clearly stated as in many code names "I" stands for inverse. I would immediately assume that IPhreeqc is used for calibration and sensitivity analysis.</p> <p><i>Yes, the "I" in IPhreeqc stands for "interface". We clearly stated this in the revised version (page</i></p>

	4:21). Thank you for the careful reading.
8	<p>2373:2 – “If a DDC approach, e.g. for flow and transport, is applied for the attached reactions system as well, then choosing the most suitable number of compute cores will lead always to a certain trade-off.” Any choice of compute cores will lead to a “certain trade-off”. Is the point that using the same number of compute nodes for reaction as was used for flow and transport does not always lead to the most optimal performance? This could be clarified.</p> <p><i>Yes, this is the point we would like to make and it is also one important motivation of our performance tests. We clarified this in the revised manuscript, and to be clearer to the readers, we added the following explanations (page 5:6):</i></p> <p><i>“In the operator splitting approach, the chemical reaction system is solved on each finite element node individually, so that no node-wise communication is necessary. However, flow and mass transport is bound to DDC, so that additional communication is needed to exchange the results along shared subdomain boundaries. Therefore a speedup for flow/transport is not given anymore, if communication and serial fractions are more time-consuming than the parallel fractions. As a consequence, an efficient number of compute cores assigned to the transport problem may already reach a maximum limit; while a further speedup for chemical system can still be present with the adding of more compute cores.”</i></p>
9	<p>2373:9 - “Global processes will be paralleled based on DDC method”. paralleled -> parallelized?</p> <p><i>Thank you for improving the language. Yes, it should be “parallelized” (revision, page 5:22)</i></p>
10	<p>2374:3 – A tradeoff exists between implementing biogeochemistry natively in a code versus coupling to a third-party library. On the one hand, native implementation could be much faster (with respect to run times) and provide more flexibility (e.g. it is generally easier to customize one’s own code). On the other hand, duplication of effort makes leveraging a third-party library more appealing. A comparison of OGS using native biogeochemistry vs. Phreeqc biogeochemistry on the same exact problem would better inform the reader regarding the computation overhead of coupling to a third-party library. I understand the Phreeqc brings a more extensive suite of biogeochemistry to the table for the scientist, but if a scientist is employing reactions that are already natively available in OGS, is it worth the effort to use the coupling to Phreeqc as the execution is likely more complicated and computationally expensive (to some degree)? A comparison between native OGS chemistry and Phreeqc would greatly improve the impact of this manuscript.</p> <p><i>In response to this comment, we simulated the van Breukelen benchmark by using the KinReact module of OGS (“native chemistry”), and compared its runtime with OGS#Phreeqc and standalone PHREEQC. The KinReact module is faster than the coupling (1.4s vs 32.7s) of course but does not have the wide range of geochemical capabilities as PHREEQC does (e.g. surface complexation, mineral nucleation etc.).</i></p> <p><i>See revised manuscript (page 11:11) and table 5.</i></p>
11	<p>2375:10 – “The source code of PHREEQC however is not changed: : :”. Do you mean that the IPhreeqc interface does not change while the Phreeqc source code can be refactored/updated. A well-designed interface allows one to modify code with minimal impact to the interface.</p> <p><i>Yes, we do not need to change the interface. For clarification we rephrased this part in the revised manuscript (page 7:25):</i></p> <p><i>“The interface itself is version independent and can stay unchanged after updates. For example, the integration of a new IPhreeqc release into the combined code can be realized simply by updating the IPhreeqc source code. Updates which will include/exclude IPhreeqc files only need a reconfigured list in the CMake file. This allows users to benefit continuously from code developments of both sides.”</i></p>

12	<p>2375:16 – “In the first development step, : : ”. Okay, the file-based approach is a first step and the “string-based data exchange” is the next step. A more optimal approach would be in-memory coupling where IPhreeqc is called directly from OGS with no startup, initialization, etc. Just the raw data being passed in to a previously initialized IPhreeqc instance and a time step calculated with results passed back. Do you intend to take this implementation to that level of sophistication?</p> <p><i>We are grateful to the reviewer raising this issue. Yes, we were and are still taking the “in-memory” coupling into consideration, which means to access each other’s code internal data structures directly. Technically, this is possible. Tightly coupled, the interface would almost disappear. Based on our performance analysis on a Linux cluster with distributed memory and a network file system we found so far e.g. following proportional distributions for the presented 3D example (Figure 10):</i></p> <ul style="list-style-type: none"> - <i>The time spent for the interface takes e.g. around 30% (80 cores, 20 DDCs) of the total time with file-based approach. This portion increases significantly with increasing number of cores (Figure 10 c,d)).</i> - <i>The time for data exchange has been reduced by using the string-based approach, e.g. to around 10% for 80 cores and 20 DDCs (in Fig 10 c,d). Moreover, the overall time occupied by the interface increases much less with increasing number of cores.</i> <p><i>Therefore, for the presented examples there is only little evidence that the speedup and efficiency are tremendously improved by an “in-memory” coupling. Nevertheless, to have such a highly efficient interface would be important in order to make this approach scalable for large number of compute cores (see comment reply 29). This will be part of the planed studies on a massive parallel environment.</i></p> <p><i>The only concern for an “in-memory” coupling is that the OGS and IPhreeqc source code would be strong-coupled and this would result in some strong code dependencies as well. We think this is feasible and sustainably maintainable if both open-source communities can develop a common idea or even a standard for the shared data structures.</i></p> <p><i>We stated these points in the outlook part of the revised manuscript (page 22:10).</i></p>
13	<p>2375:19 - “: : will be passed to IPhreeqc to initialize the geochemical system” . A dear description of all the tasks that IPhreeqc must perform for each chemistry calculation would greatly improve the manuscript.</p> <p>Biogeochemical codes have many setup steps including potentially memory allocation, solver setup, the reading of the reaction network and databases containing parameters (log Ks, rate constants, stoichiometries, a basis [secondary species, minerals, surface complexes, etc.all defined with respect to a set of primary species], speciation to an initial condition, etc. The steps that IPhreeqc takes to perform this setup for each instance of IPhreeqc should be described in a few sentences to inform the reader of the overhead involved with setting up IPhreeqc over and over for each geochemistry calculation.</p> <p>Or are these steps performed once during initialization of OGS#IPhreeqc and the geochemistry calls are solely updates to the stored system? If the geochemical system is stored instead of being re-initialized for each geochemical calculation, what must be stored by OGS vs. IPhreeqc? This level of detail will better inform the reader of the overhead generated from coupling OGS with IPhreeqc and greatly improve the manuscript.</p> <p><i>As mentioned from the reviewer at first: during each chemistry calculation, IPhreeqc has to complete the following tasks:</i></p> <ul style="list-style-type: none"> <i>i) create an instance;</i> <i>ii) load a thermodynamic database (e.g. reaction networks, species, phases, log ks);</i> <i>iii) read and run the input (process updated data such as lists of elements, phases, and species;</i>

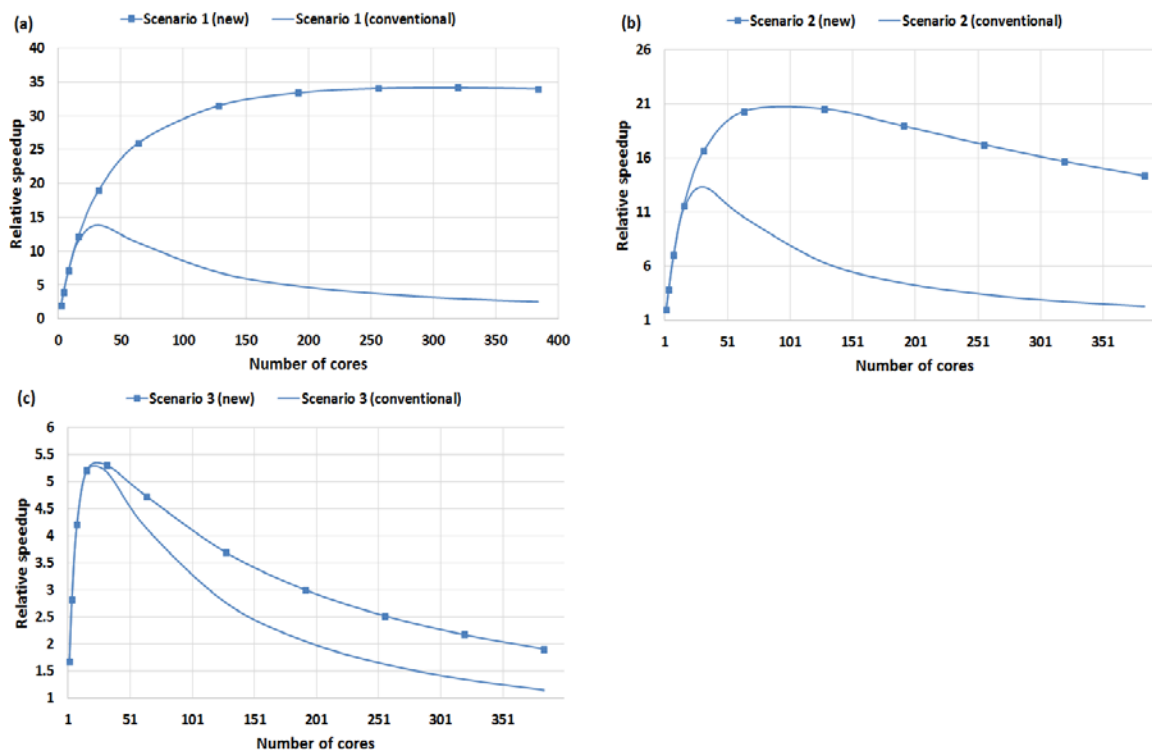
	<p>perform different calculations such as aqueous speciation, ion exchange, etc.); iv) retrieve results from calculations; v) release instance from memory.</p> <p>We described that more clearly but briefly in the revised manuscript (see page 9:3), since very detailed information can be found in Charlton et al. (2011) or Parkhurst and Appelo (2013). Furthermore we would like to state, that the involved overheads (steps other than iii and iv) are small compared to the total simulation time, especially for large problems. Actually, we already included them into the IPhreeqc time for all the test examples we presented. These overheads account for 3.8% of the total simulation time of the Engesgaard benchmark; 2.3% of that of the van Breukelen benchmark; 0.4% of the 3D parallelization example (with 20 cores and 20 DDCs). This information was added to the revised manuscript (page: 9:11, 10:17, 11:13). Nevertheless, it is attractive to minimize these overheads by performing steps i and ii only once during the initialization and step v only once at the end of the entire simulation. We stated this in the outlook part of the revised manuscript (page: 22:14).</p>
14	<p>2375:24 – “ : :an input file for IPhreeqc will be prepared.” Again, does each call from OGS to IPhreeqc entail a complete IPhreeqc run (initialization, execution, finalization) or does IPhreeqc initialize and sit idle waiting for the input file to appear and execute a single time step, but without terminating at the end. I understand that the IPhreeqc cores sit idle waiting for the next IPhreeqc step, but the question is whether the entire IPhreeqc code is re-initialized over and over for each geochemistry step. If this is stated elsewhere in the manuscript, please point me to the page:line.</p> <p>Please, see comment response 13. To avoid misunderstanding, we rewrote the whole paragraph (revision, page 8: 20, 9:11).</p>
15	<p>2376:11 – “ The latter two benchmarks will be shortly introduced here” . If either of these benchmarks have been altered in any way (i.e. if they differ from the original cited papers), a full description of the benchmark should be included in this manuscript to ensure reproducibility of results, or to allow for others to compare the performance of their simulator to OGS#IPhreeqc. If the cited references provide adequate detail the exact problem executed by OGS#IPhreeqc, this is not necessary.</p> <p>We provided a more detailed description and supplementary (PHREEQC input file) for the first benchmark (see page 10; uploaded supplementary material: Part 1). Concerning the second benchmark, the adequate details are already given in van Breukelen et al. (2005).</p>
16	<p>2376:20 – A comparison of the results between OGS#IPhreeqc and OGS-Chemapp is provided. Please discuss a comparison of the computational performance. Can you provide a detailed discussion of the breakdown of computation (e.g. % time spent in transport vs. chemistry, overall run time)? Even if this is a serial run, the breakdown helps the reader better understand the cost of coupling the codes (i.e. overhead resulting from OGS#IPhreeqc integration).</p> <p>We can provide a computational performance comparison between OGS#IPhreeqc and OGS-ChemApp for serial simulations. Providing numbers for HPC could be a bit demanding, as ChemApp requires installation and licenses on the cluster, and this seems to be somehow problematic on the machine here in Leipzig. Furthermore, we compared this benchmark with standalone PHREEQC simulation (batch version). We changed the time stepping of the according models simulated by OGS#IPhreeqc and OGS-ChemApp, in order to apply the same temporal and spatial discretization for all the three codes while fulfilling the Courant condition required by PHREEQC. The updated results are given and discussed in the revised manuscript (see page 10:11, table 4, figure 2).</p>
17	<p>2377:9 – Again, if the conceptual model executed in the second scenario differs from van</p>

	<p>Breukenlen et al. (2005), please explain for reproducibility purposes. It may be better to include a full description of the flow, transport and biogeochemistry conceptual models. Databases can be cited (or provided) to minimize the data reporting requirement.</p> <p><i>We are using the original benchmark which is fully described in van Breukenlen et al. (2005), and there are no modifications regarding the model setups.</i></p>
18	<p>2377:10 – How does the OGS#IPhreeqc computational performance compare to PHREEQC?</p> <p><i>For the first benchmark, the runtimes of OGS#IPhreeqc and PHREEQC are 7.861 s and 5.74 s. For the second benchmark, the runtimes are 32.671 s and 14.196 s, respectively. We gave an overview of the computation time for OGS#IPhreeqc in the revised manuscript in table 4, 5 and at page 10:15 and 11:11.</i></p>
19	<p>2377:18 – “All cores take part in solving the geochemical reaction system, while a subset of cores is used to solve the DDC related processes.” Is it safe to assume that all reported speedups and efficiencies factor in the idle time resulting from geochemistry cores sitting idle while flow and transport are calculated?</p> <p><i>The reported performance factors result from measuring:</i></p> <ul style="list-style-type: none"> • <i>the required time to solve flow and transport (i.e. DDC cores running while geochemistry cores sitting idle)</i> • <i>the required time to solve geochemistry (i.e. making use of all cores and no cores are sitting idle at all)</i> • <i>the required time for the interface (uses all cores as well)</i> <p><i>Does this answer the question?</i></p>
20	<p>2378:4 – This paragraph leads me to believe that each geochemistry process initializes IPhreeqc and waits for parameters to be passed to iPhreeqc at geochemical every time step. In other words, IPhreeqc is not restarted or reinitialized at every time step (no memory allocation, basis swapping, database reading, etc.). Is this true?</p> <p><i>IPhreeqc is restarted and reinitialized at each time step. Please see comment reply 13. To be clearer here, we added “(including all the IPhreeqc tasks described in Sect. 2.3)” after the sentence “which will invoke the calls to IPhreeqc for compute cores in MPI_Group2” (revision, page 13:14).</i></p>
21	<p>2378:18 – But does Ballarini et al. (2014) use OGS#IPhreeqc? If not, the parallelization schemes for reactive transport may be similar but the algorithms differ. Please clarify.</p> <p><i>No, Ballarini et al. (2014) used a loose coupling in which OGS calls simply the PHREEQC executable (page 2375: 6). We removed the sentence.</i></p>
22	<p>2379:1 – This paragraph leads me to believe that solely state variables (concentrations, molar volumes, etc.) are passed between OGS and IPhreeqc during the simulation and no initialization/memory allocation occurs after startup. Please confirm.</p> <p><i>Memory re-allocation for IPhreeqc is needed for each time step. Here, we modified the sentence “each compute core will execute IPhreeqc by using a specific input file” as “each compute core will perform a complete call to IPhreeqc by using a specific input string” (revision, page 14:9).</i></p>
23	<p>2379:10 – My experience is that Linux boxes with large core counts provide marginal performance at best for sparse nonlinear systems of equations solved implicit in time. The breakdown in performance is due to non-optimal communication/memory access through the system BUS and memory hierarchies. In general, I get a maximum speedup of around 8x on these machines no matter how many processes I employ. In fact, as the number of processes increases, performance can degrade.</p> <p><i>Yes, we agree that the performance for the calculation of finite-element-method (FEM) related</i></p>

	<p><i>processes (flow and transport) can be limited in such a hardware environment. However, chemistry is solved locally on each node within the operator splitting approach. So no communication among border nodes of different subdomains is necessary. Better speedups for chemistry were observed for the test examples presented in our study.</i></p>
24	<p>2380:14 – Please comment (or point me to a reference) on how OGS conserves local mass balance when employing a finite element discretization. I know that other FE codes such as FEHM employ a control volume approach to conserve mass. Otherwise, wouldn't the lack of local mass conservation result in instabilities in the geochemistry side of the framework (e.g. potential negative concentrations)?</p> <p><i>The standard finite element method is locally not fully mass conservative, of course. We are aware that this e.g. can lead to negative concentrations especially at sharp reaction fronts. Within OGS, we have implemented a linearized algebraic flux corrected transport (FCT) scheme and can apply it for mass transport. This FCT scheme can ensure sufficient local mass balance accuracy to a certain extend but is computational more expensive. We make no use of this scheme in the presented tests and examples. However, details regarding this implementation can be found in Kosakowski and Watanabe (2013). We could include some more detailed information to this manuscript of course, but the presented benchmark studies show sufficient accuracy and we think that a discussion about the pro and cons of FEM for reactive transport modelling is out of scope of this paper.</i></p>
25	<p>2380:16 – As the authors likely understand, the 1D and 2D (but really still 1D) problems are chemically dominant due to the simple transport (and flow?) being calculated in the problem. This could be stated.</p> <p><i>Thank you, we stated this at page 16:3. In the revised manuscript we provided a new example which involves more complex flow and transport simulations (see Sect. 4.3 or comment reply 32).</i></p>
26	<p>General note regarding Figures 8b and 9b. Relative speedup should be plotted relative to the lowest process count. The term “relative” refers to a speedup relative to the minimum number of processes available. Such plots should include an ideal speedup that proceeds from 0 (or 1 in the case of a log-log plot) to the maximum multiple of the number of processes. In other words, suppose the minimum number of processes were 4 and maximum 16. The ideal line should start at 0,0 and proceed through 4,4 (the start of the data) through 16,16. The attached image (Gwo et al., 2001 Figure 5.jpg) from:</p> <p>Gwo, J. P., E. F. D' Azevedo, H. Frenzel, M. Mayes, G. T. Yeh, P. M. Jardine, K. M. Salvage, and F. M. Hoffman (2001), HBGC123D: A high-performance computer model of coupled hydrogeological and biogeochemical processes, Comput. Geosci., 27(10), 1231 – 1242, doi 10.1016/s0098 – 3004(01)00027-9.</p> <p>illustrates the proper way to plot both speedup and the breakdown of various functionality within the total run time. Otherwise presented, it is very difficult for the reader to determine the degree to which a code is scalable. The author can state that a code is scalable, but the reader needs proof in the plotted data through comparison with the ideal speedup line. Notice the “linear speedup” line (or ideal speedup) in Gwo et al. So, take for instance Figure 8b. The ideal line should run from 0,0 to the maximum number of processes employed (20,20). The data will start at 4,4 and run through 20,20. Each line (iPhreeqc, other, total) should start at 4,4, though the ending points will differ depending on the scalability of each category. Figure 9b's ideal line will go from 20,20 to 80,80 and the data will all originate at 20,20. (As an alternative, one could set the relative speedup value to 1 for the lowest process count and label the axis accordingly. In that case, Figure 8b would run from 4,1 to 20,5 [ideal 5x speedup] and Figure 9b from 20,1 to 80,4 [ideal 4x speedup]. These plots should be fixed for this manuscript to be accepted.)</p>

	<i>Thank you very much for the detailed suggestions to improve the quality of our figures. We modified the figures according to your comments.</i>
27	<p>2381:2 – Please add a line indicating ideal speedup to Figure 8b (see comment on speedup Figures above). Within the context of the current figure, the line should be straight and run from 0,0 to 20,20 (or 4,1 to 20,5). This will help the reader compare the actual performance to ideal.</p> <p><i>Thank you, we really like this idea and we included ideal-speedup-lines. (see Fig. 8b and 9b)</i></p>
28	<p>2381:14 – Please plot Figure 8 c and d with log-log scale. In doing so, please add a single “ideal” curve which should be linear. The initial time value does not matter; it is the slope that the reader needs for comparison purposes. Again, this enables the reader to better judge performance themselves comparing the actual performance to ideal performance.</p> <p><i>We plotted Figure 8c and d regarding to wall-clock time with log-log scale and an “ideal” slope.</i></p>
29	<p>2381:14 – The challenge with this hybrid approach is that the domain decomposition flow/transport side of the code will be the bottleneck. With reaction taking 90% of the simulation time for DDC=4, the maximum speedup that one could ever get for DDC=4 (relative to DDC=4) is 10x with an infinite number of reaction processes. That is based on Amdahl’ s law and assuming DDC is the serial fraction. The question is how well this approach scales to large #s of processes.</p> <p><i>Our approach has the potential to scale to large number of processes and bring more benefit than the conventional approach for chemical dominated problems.</i></p> <p><i>If the serial fraction accounts for 10% of the total simulation time, then it is no doubt that the maximum speedup one can get is 10x, regardless of the approach and number of cores employed. In reality, we may get even less speedup, if we take the communication overhead into account. Here, we added another term to the classic Amdahl’s law, which represents the communication overhead (see equation (1)).</i></p> $S(n) = \frac{T_p(1) + T_s}{\frac{T_p(1)}{n} + T_s + T_c * (n - 1)} \quad (1)$ <p><i>where S(n) is the relative speedup compared to the serial simulation when n compute cores are employed. T_p(1) and T_s are the time spent for parallel fraction and sequential fraction, respectively. T_c is a factor for communication overhead (assuming there is a linear relationship between overhead and employed compute cores).</i></p> <p><i>The time spent for communication will lead to the degradation of the parallel performance, if it becomes dominant. For a coupled reactive transport code, communication overhead can relate to two parts i.e. DDC and interface. With a conventional approach (n = DDC), we have to stand with the growth of both parts with the increase of compute cores. In our new approach, we can fix the time consumption for the DDC (including its communication overhead) at its optimal, thus avoid its further increase.</i></p> <p><i>In order to estimate the potential benefit we can get from the new approach, we would like to present three scenarios here. In the first scenario, calculation of chemistry is dominant and the communication overhead introduced by the interface is much smaller than that of the DDC (“best case”). In the second scenario, chemistry is still dominant for the simulation. The communication time required by the interface increases, but it is still smaller than that of the DDC. In the third scenario, calculation of flow/transport becomes as dominant as chemistry and the communication overhead for the interface is now similar with that of the DDC (“worst case”). For all the three scenarios, we applied both conventional approach and the new approach.</i></p>

The results are shown in figures below, which demonstrate that the proportion of chemistry calculation and the overhead of the interface are two important factors influencing the benefit and scalability of the new method. If a problem is dominated by flow/transport and the interface is time consuming (scenario 3, Figure c), then the benefit we can get from the new method is rather limited. However, if a problem is dominated by reaction and the time consumption for the interface is considerably small (e.g. through “in-memory” coupling) (scenario 1, Figure a), then the new method has the potential to be scalable for large number of compute cores.



Section 4.2 – Again please modify the plots as follows:

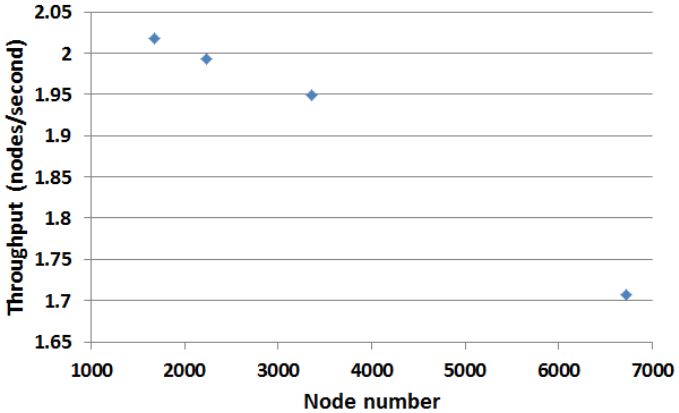
Figure 9a – Can you add an “ideal” plane that will illustrate deviation from ideal performance (I believe that one can calculate this algebraically for every DDC). This may make the figure unreadable and too complicated to decipher: : just a thought. Figure 9b – Add an “ideal” line, e.g. running from 0,0 to 80,80 (or 20,1 to 80,4) Figures 10 a-d – Replot on a log-log scale and add an line representing “ideal” slope. Also, rescale the y-axis (wall clock time) in each plot to better display results for comparison purpose. For instance Figure 10d could be rescaled to have a maximum wall clock time of 1000 seconds. It is difficult to read otherwise.

30

The “ideal plane” is again a very interesting idea. However, we think the diagram would be overloaded. Therefore we changed the figures in the following way:

- We added an “ideal” line to Figure 9b as the reviewer suggested.
- For Figure 10 (a-d) we applied log-log scale, rescaled them and included the “ideal” slopes. Additionally, we compared the results between string- and file-based approaches.
- In order to make a better comparison, we present the time spent for interface in Fig 10c.
- In Fig10d, we illustrate the total time by using both approaches.

The according discussion can be found in the revised manuscript at page 18:12.

31	<p>2382:8 – With the ideal line included in Figure 9b, can you explain the superlinear speedup in the IPhreeqc performance? Maybe I am misunderstanding something, but the degree to which that performance is superlinear is beyond cache effects, etc. A quick eyeball estimate shows ~4.75x speedup on 4x as many processes (20 vs 80). Can you explain the extra .75x speedup?</p> <p><i>In the current scheme, a compute core has to calculate chemistry for fewer nodes, when more cores are available. We think the increase of throughput with the decrease of nodes number for a single compute core can explain the super-linear speedup observed.</i></p> <p><i>In the figure shown below the throughput is plotted against the node number elaborated by one compute core. With the given conditions (hardware architecture, code structure, etc.), the throughput decreases as one core has to handle more nodes. In other words, a compute core can calculate the chemistry for one node faster, when it has to process fewer nodes. However, to understand this behavior in more details, we have to dig more into the IPhreeqc code.</i></p>  <p><i>Additionally, we do not think the extra speedup is beyond the cache effects. Depending on the situation, a speedup up to 10x or even higher can be achieved. For example, based on the documentation below (last access: Jun 27, 2015), the following latencies are given for different cache levels of Core i7 Xeon 5500 Series:</i></p> <ul style="list-style-type: none"> <i>L1 CACHE hit, ~4 cycles</i> <i>L2 CACHE hit, ~10 cycles</i> <i>L3 CACHE hit, line unshared ~40 cycles</i> <i>L3 CACHE hit, shared line in another core ~65 cycles</i> <i>L3 CACHE hit, modified in another core ~75 cycles</i> <p>https://software.intel.com/sites/products/collateral/hpc/vtune/performance_analysis_guide.pdf</p>
32	<p>General comment: Your test problems are chemically dominant. The flow and solute transport is essentially 1D. In more realistic modeling scenarios, one would expect flow and transport to become somewhat more dominant. Since the addition of cores to MPI Group 2 for chemistry does not benefit this 3D test problem much, even with the quasi-1D flow and transport, can you devise a realistic problem scenario where the hybrid DDC approach would run scale better with greater than the number of DDC cores? In other words, other than in the first, very simple 2D problem, I don't see the advantage of allocating processes to MPI Group 2, when they will sit around idle during flow and transport.</p> <p><i>Adding more cores solely for chemistry will not bring more advantages if a problem is dominated by flow/transport. In this case, using the conventional parallelization scheme (which is also possible in the current scheme through solely creating the first MPI group) will provide the best</i></p>

	<p><i>performance. We already mentioned this point in 2383: 24. Nevertheless, we rewrote that part to emphasize this to the reader (revision, page 21:21).</i></p> <p><i>As already mentioned in our answers to comment 8, calculation of chemical and non-chemical processes can have different features regarding to their parallel performance. Our new approach has more benefits than the conventional one, if severe performance degradation happens for flow and mass transport. In this case, fixing the number of DDCs at its optimal while adding more cores for chemistry would still lead to an overall better performance.</i></p> <p><i>To demonstrate this point, we added a new test example in the revision (sect 4.3), which involves the simulation of unsaturated-saturated flow. In this example, allocating compute cores to MPI_Group2 leads to better speedups than the conventional DDC approach (see Fig. 13b).</i></p> <p><i>In this example the benefit coming from chemistry side is somehow limited, since the minimum time for flow/transport takes already more than 28% of the total time. It means that the maximum speedup we can further achieve is less than 3.6 based on the fundamental point given by the reviewer in comment 29.</i></p> <p><i>However, if a problem is chemically dominant and the coupling interface is highly efficient (see scenario 1 in comment reply 29), then our approach should be scale better than the conventional one.</i></p>
33	<p>Final comments: It is not clear to me that the new methods presented in this manuscript will have a significant (beneficial) impact on subsurface simulation techniques. There are several deficiencies in the approach taken to link the two codes.</p> <p>First, limiting the cores in the second process group to solely chemistry calculations results in these cores sitting idle during flow and transport. For small, chemically dominant problems this approach may not hamper performance much, but for large scale massively-parallel simulation, this approach is not scalable as the processes should be added to the DDC side of the problem. I understand that the algorithm provides the flexibility to add processes to either side, but in reality addition to the conventional DDC side is likely the best alternative for most realistic problems. If adding cores to the DDC portion of the problem does not result in speedup, it is likely that the problem size per process (i.e. number of degrees of freedom per core) is already too small for efficient parallel computing. Speeding up the chemistry calculation will provide limited benefit in that case since the DDC portion is the bottleneck.</p> <p><i>For large scaled problems, in which flow and mass transport calculation is most time consuming, the conventional approach would be the best choice, which was also shown in Sect. 4.2 in our manuscript. We rewrote the conclusion part in order to present this point more clearly (21:18). We think that our new method can provide more benefits than the conventional approach for geochemically dominated problems with limited (or poor) parallel scalability for flow and mass transport (see response to comment 32). As already discussed in reply 29, the benefits we can get from this new approach will depend mainly on the proportion of chemical reaction and the communication overhead of the interface. The more chemical reactions dominate and the more efficient the interface, the better speedup we can obtain with the available computational resources. We are planning to setup more and even theoretical examples to test and evaluate the limitation of such approaches on large HPC machines in the near future.</i></p>
34	<p>Second, the use of file IO to transfer data between codes is obviously much slower than “in memory” data transfer and will not scale to large process counts on large problems (as the manuscript demonstrates); so why publish that approach in the first place?</p> <p>If in memory data transfer (either through the “string-based approach” mentioned in the</p>

	<p>manuscript or through double precision arrays) is the ultimate objective, why not implement the better approach and demonstrate that it scales.</p> <p><i>Our initial motivation of this work was to develop a concept, which can realize the flexible allocation of compute resources for geochemical and non-geochemical processes. That is why we focused on proving our concept and chose the file-based coupling as our first development step (technically it's easier to be implemented).</i></p> <p><i>In the meantime, we have implemented the string-based approach and discussed its advantages in Sect 4.2 (revision) as well as comment reply 12.</i></p> <p><i>There is still much room to improve our interface. We fully agree an "in-memory" coupling between OGS and IPhreeqc would be the most efficient implementation, which would also be important to ensure the scalability of this approach for large number of compute cores (see comment reply 28). This will be part of our future studies.</i></p>
35	<p>For these reasons, I would claim that the approach is novel, but certainly not revolutionary as implemented at this point in time. I believe that by addressing the questions/comments presented above, the manuscript will be greatly improved.</p> <p><i>We would like to thank the reviewer again for his very helpful review and very constructive suggestions. We believe these comments helped to improve our manuscript significantly. We agree that the presented work might be just a small step towards improving the performance of reactive transport modeling (RTM) – but we think that it shows the benefit of the novel concept and contributes indirectly to evaluate the range of application of such couplings in general.</i></p>