Comments from Reviewer A

1. Remove numbering from Introduction and Conclusion sections.
2. Remove full stops from the end of the figures and tables captions.
3. Remove italic in the text of Acknowledgement section.
4. Add DOI for each item (where applicable) in the Reference. Use CrossRef service (https://search.crossref.org/) to find DOI by article's title.
5. Apply corrections according to the uploaded proof-sheet.

Answer to Reviewer A

1. I removed the numbering from Introduction and Conclusion sections.
2. I removed the full stops from he end of the figures and tables captions. The explanations of the figures and tables are merged into the manuscript shown by blue.
3. According to superfri formatting guideline, I removed the section title Acknowledgement. Italic is specified format of Acknowledgement.
4. I added DOI to each item as much as possible.
5. I arranged the appearance of the manuscript according to the proof-sheet

Comments from Reviewer C

1. I just want to know general comments of the authors on the importance of the steering, or real-time, visualization in the HPC simulations in practice. (Please do not get me wrong. I do not intend to ask a mean question. I just want to know the thoughts of the authors on these matters on the real-time visualization.)
   When one submits a (batch) job on a supercomputer system, one usually has to wait for hours or tens of hours or even days, until it starts to run. Of course, the job scheduler tells you when it will start, but, as you know, the job sometimes starts much faster than the predicted schedule. It may start at midnight. And after it starts running, it may take more than 12 hours to complete the job if you are doing a long simulation. Under this environment, do you believe that the steering (or real-time) visualization is a good way to analyze the simulation? Do you wait for the start of the job in front of the client PC and perform the real-time visualization for 12 hours during the job is running?
2. (1) typo: Section 4: "If one call Sampler less frequently,"
   ==> If one calls…
3. (2) Ref. 17: The following paper would be more appropriate: Sakamoto N., Koyamada K., KVS: A simple and effective framework for scientific visualization, Journal of Advanced Simulation in Science and Engineering 2:76–95, 2015.

Answer to Reviewer C

1. This paper describes about the performance of the interactive in-situ system. Actually, In-situ PBVR has a function of time history scan, which is not described

because it is out of the scope of this paper. Even if the simulation code coupled with In-Situ PBVR's Sampler is executed as a batch job without invoking the Daemon and Viewer, the particle data is generated using the existing visualization parameter file and is stored in the storage as in normal in-situ visualization tools. When Daemon is started up during or after the batch processing, Daemon collects the particle data on the storage, and the user can search the history on the Viewer. What is important here is that one can modify the visualization parameters during or in between batch jobs without re-compiling the load module of the simulation code. We believe that this flexibility is of critical importance for In-Situ visualization.

2. I modified the manuscript according to reviewer comments.
3. I modified the reference according to reviewer comments.

Comments from Reviewer D

1. The three issues of conventional volume rendering algorithms mentioned in the introduction can all be solved by PBVR only. IMHO, the special in-situ features of "In-Situ PBVR" should be addressed here, too.
2. The following articles need to be mentioned for reference to PBVR in 'Related Works' (or in section 'In-situ PBVR'):1) Sakamoto, Naohisa, et al. "Particle-based volume rendering."Visualization, 2007. APVIS'07. 2007 6th International Asia-Pacific Symposium on. IEEE, 2007. 2) Sakamoto, Noahisa, et al. "Improvement of particle-based volume rendering for visualizing irregular volume data sets." Computers & Graphics 34.1(2010): 34-42. (especially as T. Kawamura are even co-author of this paper)
3.
    A. Why is the maximum particle data size only several hundreds MB? From my understanding it can be any size.
    B. Please comment why Daemon communicates via file-system with the Sampler. IMHO, this is not state-of-the-art. But there might be good reasons in this special case.
4.
    A. My strongest concern is, that the title of the paper starts with "Performance Portability [...]". A performance portable implementation of an application or algorithm is one that will achieve high performance across a variety of target systems. But only results of a single system are presented.
    B. In addition the paper shows a single setup which hardly scales: with 64 times more nodes the Solver is 3.9 times faster and the Sampler is 9.5 times faster. At least the paper should have a much detailed answer on the reasons for this. The setup is far to small for strong scaling up to 1,536 compute nodes.
    C. IMHO, "Performance Portability" must be replaced by “Performance Evaluation”/"Performance".
5.

A. The In-Situ PBVR communicated over the file-system and has a major bottleneck because of that. IMHO, it is not correct to conclude, that the framework is designed to avoid the performance bottlenecks on many-core platforms.
B. I also do not see from the results, that Sampler shows excellent strong scaling up to ~100k. IMHO, I would expect much more that x9.5 to conclude this.

Answer to Reviewer D

I would like to thank the referee for the useful comments, which improve the manuscript.

1. As you pointed out, the small memory size and the view independency or rendering primitives can be resolved via the characteristics of the original PBVR. I added the description about the characteristics of PBVR in the introduction (shown by red). However, we believe that strong scaling is achieved by our parallel implementation.
2. I added these references in the introduction (shown by red).
3.
    1. As the referee pointed out, the particle data size can be any size, which can be controlled by sub-pixel level, image resolution, and transfer function. However, for 1,024 x 1,204 pixels, that is our assumption in the argument, fine images can be obtained with several hundreds MB particles. Since the image quality does not change so much above this size, we described it as “maximum”.
    2. External connections to computing nodes such as direct port forwarding are not allowed under most massive parallel environments including Oakforest-PACS. File-based communication is one of general purpose solutions, which can be available at user level. The system detail is described in the section 2.3 “Daemon” and Algorithm1. Since we could not find the file-based solution in former in-situ works, we believe that it has novelty.
4.
    1. As the referee pointed out, so far, we tested In-Situ PBVR only on two platforms, ICEX and Oakforest-PACS. So, we changed title “Portability” to “Evaluation” and modified relevant expressions. (Just to be sure, the performance of In-Situ PBVR has evaluated on Haswell architecture in our previous paper described at the beginning of the section 3).
    2. The reasons for the degradation in acceleration ratio is the followings. The performance of the Solver is limited by the reduction of the problem size per node, and the Sampler has a bottleneck of the file I/O as described in the section 3.
    3. We have answered in 4.1.
5.
    1. According to your advice, we changed the corresponding part in the section “Conclusion” shown by red. (designed to avoid the performance bottlenecks -> designed to bring out the parallel performance)
    2. According to your advice, we changed the corresponding part in the section “Conclusion” shown by red. (The experimental result shows that Sampler shows

excellent strong scaling up to ->  The result of strong scaling test shows that Sampler performance scaled up to)