Revision Plan of the Paper

Boosting GPU virtualization performance with relaxed shadow page table

Yaozu Dong, Mochi Xue, Xiao Zheng, Jiajun Wang,

Zhengwei Qi, Haibing Guan

We gratefully acknowledge the constructive comments and advice from the committee and reviewers. We would follow the comments and revise the paper by fixing the problems, and rewriting the part of the draft. Also, we would focus on the English writing and the overall organization of our paper.

In the following section, we first present the overview of our revision plan, and then give out the responses and action plan to the committee and reviewers' comments.

1. Overview of Revision Plan

In this section, we would give out the overview of our revision plan.

1) Re-organize the paper and improve the English writing

We would first fix the typos and grammar errors (c.f., Response A5#, D4#) and rewrite sections of our paper to improve the English writing(c.f., Response B5#, B6#, D2#). Also, as part of the structure of our paper, we would be re-organizing to improve the readability. After our revision, we will use the services of a paid technical English editor. Finally, we would invite our colleague, who is a native English speaker to proofread our paper. (c.f., Response P1#, B1#)

2) Explain the terminologies in detail

We will add detailed and clear explanation of some terminologies in the paper, especially within the abstract, while eliminating the usage of new terminology that appears only once or twice. Some new content would be added to present the GPU programming model and commands, while removing some non-relevant description in Section 2. (c.f., Response A1#, B4#, B7#, C4#, D3#)

3) Clarify the inaccurate statement about GPGPU benchmark

We would revise our statement "There are no suitable tools to measure the performance of GPGPU applications" to be more specific. (c.f., Response A4#, B3#, C2#)

Reviewers pointed out that there are two popular benchmarks Rodinia and Parboil for GPGPU, unfortunately these two benchmarks are currently not available for Intel's GPU.

Rodinia is a benchmark suite for heterogeneous computing. It helps architects study emerging platforms such as GPUs (Graphics Processing Units). Rodinia includes applications and kernels that target multi-core CPU and GPU platforms. However, it is not available for Intel's GPU. The Parboil is a set of throughput computing applications useful for studying the performance of throughput computing architecture and compilers. It collects benchmarks throughout computing application researchers in many different scientific and commercial fields including image processing, bio-molecular simulation, fluid dynamics, and astronomy. Notably, the applications of Parboil cannot run on Intel's GPU currently.

4) Renew some terminologies

Inspired by reviewer B, we would like to use the term "relaxed shadow page table shadowing mechanism" to replace the term "asynchronous page table shadowing mechanism", and replace "synchronous page table shadowing" with "strict page table shadowing". Therefore, we will change our title from "boost up GPU virtualization with asynchronous shadow page table" into "boosting up GPU

virtualization with relaxed shadow page table."

5) Stress the generalization of our work

We would add some discussion about the universality of our work in Section 3.1. If the modification of page table does not require to take effective immediately on the hardware side (e.g., in GPU programming model we mentioned in Section 2.2), our relaxed page table shadowing mechanism may be applied. The benchmark GMedia presented is barely a wrapper which directly invokes the media functions of Intel's MSDK (Media Software Development Kit) containing all the fundamental functions of media processing. The test cases in GMedia should be effective in a real environment with workloads developed based on Intel's MSDK. Our work improves the performance of GMedia, and we believe the optimization can potentially be applied to all the applications which uses Intel's MSDK. (c.f., Response C3#, D1#)

6) Redraw some figure and adjust the size of figure font

Some unclear figure (especially Figure 6, Figure 7) would be redrawn along with the adjustments of the figure font. (c.f., Response A2#, A3#, B2#, C1#)

7) Compare gVirt with trace technique mentioned in the paper

https://www.vmware.com/pdf/asplos235_adams.pdf

We would compare our gVirt with the tracing technique mentioned in the paper above in Section 6.

The VMM uses traces to prevent its shadow PTEs from becoming incoherent with the guest PTEs, i.e. updating the shadow page table to make it coherent with guest page table. Typically, VM trace mentioned in the paper uses write-protection mechanism, which, however, can be the source of overhead. This technique is similar to current gVirt's strict page table shadowing mechanism. The strict page table shadowing mechanism, which frequently traps and emulates the page faults of the shadow page table, causes overhead. gHyvi removes the write protection from the shadow page table to eliminate the overhead caused by excessive trap-and-emulation, taking advantage of the GPU programming model. (c.f., Response P2#, A6#)

2. Responses and Action Plan to Each Comment

==========

Comment

Paper #115: Boost GPU virtualization performance with asynchronous shadow page table

---

Summary of the program committee discussion:

The committee decided that the contribution of this paper is worthy of publication, but that, in its current state, the paper cannot be accepted. The authors are given a chance to resubmit. To be accepted, the authors are requested to address the issues raised in the reviews and to put special emphasis on rewiring the paper for clarity. The presentation needs to make sense and explain the ideas one after the other in a logical, chronological order. The grammar must be correct.

Comment P1#

The committee highly recommends that (1) the authors use the services of a paid technical English editor, and (2) ask for feedback regarding the writing from colleagues---who are native English speakers---capable of understanding the paper.

Response P1#

Thanks for this great suggestion. After our revision, we would use the services of a paid technical English editor. Also, we will invite our colleague who is a native English speaker to review the revised paper.

Comment P2#

Lastly, the committee requires that the authors will contrast their work with https://www.vmware.com/pdf/asplos235_adams.pdf which says about shadow pages that: "The VMM uses traces to prevent its shadow PTEs from becoming incoherent with the guest PTEs. The resulting trace faults can themselves be a source of overhead, and other coherency mechanisms are possible. At the other extreme, avoiding all use of traces causes either a large number of hidden faults or an expensive context switch to prevalidate shadow page tables for the new context."

Response P2#

This is a very good suggestion. We would contrast our gVirt with the tracing technique mentioned in the above paper in Section 6.

The trace technique mentioned in the paper is similar to current gVirt's synchronous page table shadowing mechanism. We would compare these two techniques in our revision plan 7#.

Reviewer 1#
USENIX ATC '15 Review #115A

Paper #115: Boost GPU virtualization performance with asynchronous shadow page table

---

Overall merit: 3. Weak accept
Reviewer expertise: 3. Knowledgeable

===== Paper summary =====

The paper presents gHyvi, an enhancement to Intel's gVirt GPU virtualization for Xen. gHyvi eliminates frequent VM exits due to GPU page table manipulation in the guest, and improves performance of certain workloads by up to 13x.

===== Strengths =====

The key methodology is clever, and the measurements presented are detailed and convincing. Not only does gHyvi solve the particular problem described in the paper, it doesn't regress existing benchmarks.

===== Weaknesses =====

The writing is very confusing and hard to follow, and the whole paper needs to be re-organized. It's not entirely clear how widely applicable the work is. It seems to be purely focused on media transcoding, but no studies or numbers are given to show usage in real world environments.

===== Detailed Comments =====

Although the focus of the paper seems narrow, I think that with some work this could be an interesting paper to a wider audience. I believe that the study of a particular class of VM exits, and a clever solution to a performance problem would appeal to many people. However, the paper as it stands is disorganized, and full of confusing and unclear prose.

More Detailed Comments:

Comment A1#
A more clear/concise definition of "Massive Update Issue" should be in Section 1.

Response A1#
Thanks for your advice. We would give a concise definition of “Massive Update Issue” in the Section 1.

The Massive Update Issue is an issue caused by the VM’s frequent modifications on the guest page table. In order to guarantee the coherence between guest page table and shadow page table, gVirt has to massively update the shadow page table and it results in a lot of overhead.
We would describe this issue in detail in Section 5.2 by profiling the test cases with the Massive Update Issue. To improve the readability, we would add some instructions in Section 1.

Comment A2#
There is no need to quote fps and percentages for every single value from every bar of every bar chart. Please summarize - we can see the bars and their relative relationships for ourselves.

Response A2#
Thanks for your suggestion. We would remove the detailed quotation and summarize the value of fps and percentage.

Comment A3#
In Figure 11, you claim that static partial 200 achieves the best performance, but that is not clear at all from the graph.

Response A3#
Thanks for the comment. We would redraw the Figure 11 and revise the description to make the presentation clearer.

Indeed, it is not clear from the graph that static partial 200 achieves the best performance. Additionally, the statement in the paper is not accurate, what we want to present is that in cases with the Massive Update Issue, static 200 achieves best performance.

Comment A4#
In Figures 13 and 14 you claim "no negative effects" of gHyvi, but there are 4 different tests where gHyvi performs slightly worse than gVirt. This needs to be explained.

Response A4#
Thanks for pointing out this inaccurate claim. We would revise this statement to make it accurate and detailed.

The statement “no negative effects” was too absolute, and the experiments demonstrated that for 2D and 3D performance gHyvi achieves comparable performance with gVirt. In 2D and 3D benchmarks, there were a total of 13 test cases, and in 4 cases gHyvi performs slightly worse than gVirt. We believe there to be noise within the data, and the results are acceptable.

Comment A5#
"Dorm0" -> Dom0
"VPU" -> VCPU

Response A5#
Thanks for pointing out these typos. We would proofread the whole paper in order to rectify this.

Comment A6#
This paper really needs to differentiate itself explicitly from: Keith Adams and Ole Agesen. A comparison of software and hardware techniques for x86 virtualization. https://www.vmware.com/pdf/asplos235_adams.pdf.

Response A6#
This is a very good comment. We would contrast our gVirt with the tracing technique mentioned in the above paper.

The VMM uses traces to prevent its shadow PTEs from becoming incoherent with the guest PTEs. This technique is similar to gVirt's synchronous page table shadowing mechanism. We have compared these two techniques in our revision plan 7#.

==========
======
USENIX ATC '15 Review #115B

Paper #115: Boost GPU virtualization performance with asynchronous shadow page table
---

Overall merit: 4. Accept
Reviewer expertise: 3. Knowledgeable

===== Paper summary =====

The authors build on and extent previous work called gVirt [30], which virtualizes the GPU using shadow page tables of GPU address spaces. They observe that newly created PTEs are to be used by the GPU only when the next GPU command is issued, and that that command will trigger an exit. They thus suggest a more relaxed page table shadowing mechanism: rather than trapping every change to the guests GPU page table to update the shadow page table eagerly, they only trap the first access to each page table and mark this table as being modified; later, when a GPU command is finally issued, they lazily go over all modified guest tables and update the shadow page hierarchy accordingly, before issuing the command.
The authors find that the reconstruct mechanism is more performant when a large number of entries in a page are modified between command issues but less performant when there are few modification to a page.
They thus suggest various policies to decide which approach to use on a per page table basis. They evaluate the various polices and pick the one that they consider the best. They demonstrate a significant performance improvement for a multithreaded media transcoding benchmark they come up with. They further show no negative

performance impact on standard 2D and 3D graphics benchmark.

===== Strengths =====
The proposed optimization makes sense and is supported by a workload analysis that convincingly highlights why is works.

===== Weaknesses =====

The English writing and presentation of the paper should be much improved. The figure fonts are much too small and illegible. Claim regarding lack of GPGPU benchmarks is not true.

===== Detailed Comments =====

Comment B1#
While the idea is small, it makes sense and is supported by a convincing performance analysis. The paper, however, is full of grammar mistakes and would greatly benefit from a professional English editor---I warmly recommend that you use one. (Example: "and policy with more pages reconstruct pages not been accessed".)

Response B1#
Thanks for the suggestion. After our revision, we would definitely use the service of professional technical editor.

Comment B2#
The figure fonts are unacceptably small; with such figure, the paper cannot be accepted for publication. Rule of thumb: the size of the text in the figure should be similar to the size of the text in the caption of the figure. As it is, I had to sometimes enlarge the figs by 300% to be able to read them.

Response B2#
Thanks for this kind suggestion. We would redraw some figures, and of course, the font size would be adjusted according to the rule.

Comment B3#
You say that there is no commonly used benchmark for GPGPU. ("There are no suitable tools to measure the performance on GPGPU applications.") This is not true. The Rodinia benchmark is cited >600 times:

Rodinia: A benchmark suite for heterogeneous computing. Che, Shuai; Boyer, Michael; Meng, Jiayuan; Tarjan, David; Sheaffer, Jeremy W.; Lee, Sang-Ha; Skadron, Kevin. IISWC 2009. http://dx.doi.org/10.1109/IISWC.2009.5306797

And there's also the Parboil benchmark that is cited >100 times:

Parboil: A Revised Benchmark Suite for Scientific and Commercial Throughput Computing
http://impact.crhc.illinois.edu/shared/docs/impact-12-01.parboil.pdf

Response B3#

Thanks for this good and detailed comment. Our apologies that our incorrect statement raised so much confusion and misunderstanding. We would definitely make our statement more accurate and add more content about the GPGPU benchmarks.

The statement “There are no suitable tools to measure the performance on GPGPU applications” is inaccurate. There are some tools implemented to measure the performance of GPGPU applications, such as Rodinia and Parboil.

Unfortunately, the two benchmarks listed above are currently not available for Intel's GPU. And we have discussed about these two benchmarks in our revision plan 3#.

Comment B4#

The abstract should be understandable by readers. But there's no way anyone will understand what you mean by "coarse-grain model", which is the enigmatic term you use to describe what you did. I Suggest to drop this term and instead explain what you mean by it.

Response B4#

Thanks for your comment. The “coarse-grain model” in the abstract would be dropped. Here the “coarse-grain model” means that gVirt updates the shadow page table as long as the guest page is modified.

Indeed, the abstract should be understandable by readers, and enigmatic terms should be replaced with understandable words.

Comment B5#

Grandiose and dramatic language especially when unsubstantiated. Statements like "GPU clouds are doesn't help your paper. On the contrary; it weakens the paper,now in EXCESSIVE need for full GPU virtualization" or "GREAT demands of hosting rich GPU applications make an URGENT call for full GPU virtualization" are annoying, if not laughable. They achieve exactly the opposite of your intent. Simply remove them.

Response B5#

Thanks for pointing this out. We are so sorry that we used these annoying words and we will definitely remove these words.

Comment B6#
Specifying a long list of (percent) numbers in the body of the text is unreadable and unacceptable ("For cases without issue, it achieves 69.46%, 93.93%, 95.45%, 97.02%, 94.37%, 101.02%, 87.76% and 89.65% of native...".) Arrange such lists in proper tables.

Response B6#
Thanks for this suggestion. We would re-organize the statics from the figures.

Comment B7#
"Shared shadow global page table is implemented for all VMs to achieve resource partition and address space ballooning." -- I don't get the statement regarding balloon. Please explain.

Response B7#
Sure. We would add a detailed explanation of ballooning of the gVirt in Section 2.2.

Here the ballooning means the technique gVirt isolates the address spaces of different VMs in shared shadow global page table. The shared global page table is accessible for every VM. However, only the part of the shared global page table can be accessed for each VM to guarantee the isolation, and gVirt uses the ballooning technique to hide the rest part of share shadow global page table from the VMs.

==========
======
USENIX ATC '15 Review #115C

Paper #115: Boost GPU virtualization performance with asynchronous shadow page table
---

Overall merit: 2. Weak reject
Reviewer expertise: 2. Some familiarity

===== Paper summary =====

This paper observes the massive update issue for GPU virtualization using an in-house media transcoding benchmark, GMedia. To improve the performance of virtualization, the authors propose a hybrid page table shadowing mechanism, which combines the synchronous and asynchronous page table shadowing scheme to reduce the overheads of trap-and-emulate. Results show that the proposed mechanism achieves up to 13x improvement over the baseline solution.

===== Strengths =====

This paper points out that frequent page table updates could be an issue for GPU virtualization.

===== Weaknesses =====

The paper is poorly written. It is very difficult to understand the proposed mechanism. The proposed mechanism is only evaluated on an in-house media transcoding benchmark.

===== Detailed Comments =====

Comment C1#
1. I can't understand Section 3.2. What do the legends of x-axis mean? What are Dom0? What is the number of channel? Which part of Figure 2 tells us the performance degradation of gVirt is actually caused by frequent page faults?

Response C1#
Thanks for your comment. We would redraw Figure 2 in Section 3.2 to make it clear. Additionally, we will explain the terms in Section 3.2 in detail.

The legends of x-axis are the brief description of our test cases; "1-th 480p" means one thread 480p media transcoding; "5-th 720p" means 720p media transcoding run in five threads.

Dom0 is the initial domain started by the Xen hypervisor on boot. Dom0 is an abbreviation of "Domain 0" (sometimes written as "domain zero" or the "host domain"). Dom0 is a privileged domain that starts first and manages the DomU unprivileged domains.

"Channel" is actually the term referred to thread, and later we would replace this word with "thread".

Figure 2 tells us that the performance degradation happens when the threads reaches, and surpasses, a certain threshold. In Section 5.2, the profiling of cases with performance degradation tells us that the degradation is caused by frequent page faults.

Comment C2#
2. In Section 3.1, the authors claim that there are no suitable tools to measure the performance of GPGPU applications because of GPU's lack of DRAM ECC. However, there are tons of general purpose applications that run on GPUs, such as physical simulation, financial computing, or bioinformatics. Furthermore, there are

benchmark suites that cover a wide-range of GPGPU applications, such as Rodinia or Parboil. This section needs to be revised.

Response C2#

Thanks for your detailed comment. Our apologies that our presentation caused so many confusions and we have revised this section, and removed the inaccurate statement.

Actually, what we intended to say that there is no GPGPU benchmark that runs on Intel's GPUs. Intel graphics chips are not widely used in the GPGPU environment, most applications and benchmarks do not run on Intel graphics chips currently. Rodinia and Parboil do not supported on Intel GPU either.

Comment C3#

3. The authors need to show Massive Update is a common issue for GPGPU workloads.

Response C3#

Thanks for your comment. We are sorry again for our poor presentation. We would add a detailed explanation of the Massive Update Issue in Section 3.2.

Our intentions were to present is the Massive Update is a common issue for media processing. As we said, we found this issue while running our benchmark GMedia. The benchmark GMedia we present is barely a wrapper which directly invokes the media functions of Intel's MSDK (Media Software Development Kit) which contains all the fundamental functions of media processing. The test cases in GMedia should be able to stand for the real environment workloads developed based on Intel's MSDK. So, the Massive Update Issue is a common issue within Media processing.

Comment C4#

4. I can't understand the explanation the authors provide why asynchronous page table update could work. What are CMDs? What do you mean "a real GPU may fetch the CMDs and cache the page table internally"? If this is particular to gVirt, the authors should explain gVirt in details here.

Response C4#

Thanks for your comment. We would try to avoid using abbreviated terms and rewrite the Section 4.2 to make the description of our asynchronous page table shadowing mechanism more clear and understandable. Additional content about Intel's gVirt and GPU programming model would be added in our revised paper.

Here CMD is short for command, and here the commands especially mean the GPU commands. "A real GPU may fetch CMDs and cache the page table internally", this is how Intel's GPU works to update the modifications of the page table. This is not

particular to gVirt, instead, this is the general GPU programming model.

==========
======
USENIX ATC '15 Review #115D

Paper #115: Boost GPU virtualization performance with asynchronous shadow page table
---

Overall merit: 2. Weak reject
Reviewer expertise: 2. Some familiarity

===== Paper summary =====

The paper presents a hybrid shadow page table technique for GPGPU virtualization. Authors argue that synchronous mode is not performant in the case of massive update. On the other hand asynchronous mode is not good for workloads with infrequent updates. Authors build a hybrid system that adaptively switches from one mode to the other.

===== Strengths =====

- Hybrid scheme for page table manipulation

===== Weaknesses =====

- Unclear, if this extends to more workloads
- Hard to understand evaluation

===== Detailed Comments =====

The paper clearly identifies a problem in the massive update use-case and solves the challenge by introducing a hybrid approach for shadow page table management. I like the clear explanation of pros and cons of synchronous and asynchronous modes.

Comment D1#
However, the evaluation mainly focuses on the massive update use-case with their customized benchmark. The benefits seem to be good, but it's unclear to me if this extends in general. The aspect of adaptation is not discussed clearly. How do you define "frequently updating the page". Would be good to get a more formal answer on this.

Response D1#

Yes, our optimization applies to all types of workloads. We would add discussion whether our optimization is general in the revised paper. A formal definition about frequency would be added, as well.

It significantly improves the media performance while shows slight negative effects on 2D/3D performance. The benchmark GMedia we present directly invokes the media functions of Intel's MSDK (Media Software Development Kit) which contains all the fundamental functions of media processing. The test cases in GMedia should be able to withstand the real environment workloads developed based on Intel's MSDK.

For cases without the Massive Update Issue, the amount of updates on the shadow table is less than 4000 per case, and the frequency is less than 100 times per second. However, when the Massive Update Issue occurs, the amount of updates increases, and the frequency of update could be up to 60k per second. We believe that the frequency over 10k per second is frequent.

Comment D2#

The evaluation is difficult to read with many percentage numbers given for each case. I suggest the authors to not repeat what is already described in the figures (may be make the figures larger to make the percentage improvements clearer or draw another figure showing improvements).

Response D2#

Thanks for the constructive suggestions. We would remove most of the percentage numbers or move them into tables to make the performance compare clearer to the readers.

Comment D3#

The paper has some GPU/media jargon that was difficult to follow. The x-axis in Figure 2 (and many later figures) is not clearly explained. What does it mean to run multiple threads on a GPU? Are all these threads executing GPU instructions or not? The evaluation talks about various types of workloads in a benchmark, but the characteristics of those workloads are not clearly explained. Especially, for a non-expert, it's difficult to see what these workloads represent and whether they are general enough or not.

Response D3#

Indeed, the use of too much terminology makes the paper difficult to follow, especially for readers who are not familiar with the field. We would revamp most of our figures to make the presentation clear, and we would add a detailed explanation of the figure. Section 3.1 would be revised to add a more detailed description about our benchmark.

When transcoding the media, we can control how many threads to use in order to increase the bandwidth. All these GPU threads are executing GPU instructions.
As for the workloads, we would summarize their characteristics, stating that the workloads frequently allocate graphic memory, which leads to the massive update of the shadow page table.

Comment D4#
Minor typos
---
1. Section 1. "... which achieves which achieves ..." => "which achieves"
2. Section 5.1 "Dorm0" => "Dom0"

Response D4#
Thanks for pointing out the typos. We would proofread and rectify the typos.