— Additional material —
Interactive Visualization and On-Demand Processing of Large Volume Data: A Fully GPU-Based Out-Of-Core Approach

Jonathan Sarton, Nicolas Courilleau, Florent Duguet, Yannick Remion and Laurent Lucas

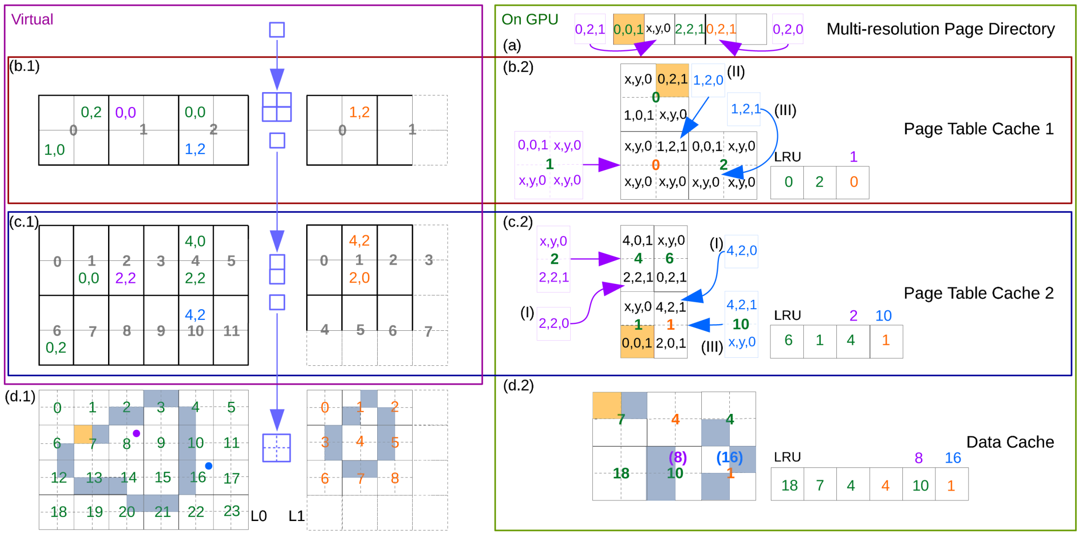

Fig. 1. **Hierarchy updates.** This example shows the virtualization of a *volume* of size $12 \times 8 \times 1$ voxels (d.1) with bricks of size $2 \times 2 \times 1$ using an MRPD and 2 levels of PT caches. Dereferencing of cache hierarchy is done as follows: One entry of the MRPD addresses $2 \times 2 \times 1$ PT1 cache entries while one entry of PT1 cache addresses $1 \times 2 \times 1$ PT2 cache entries and one entry of the PT2 cache addresses a brick of $2 \times 2 \times 1$ voxels. The MRPD and the 3 caches are stored on the GPU with one LRU for each cache. To get access to the voxel [2, 2, 0] of L0 we follow the yellow path as explain in section 5.4 of this paper. The entries stored in the MRPD and the PT caches are [X, Y, Z, F] with [X, Y, Z] the coordinate of the beginning of the entry in the next cache and F the flag of this entry. The right side of this figure illustrates the out-of-core hierarchical vision of our addressing system on GPU – from the MRPD to the data cache – when the left side of that same figure represents the data vision of the two levels of detail (d.1) physically hosted on a storage device and its virtualized representation (b.1/c.1).

Abstract—We present in this additional document a direct application of our system. We deliberately chose to present a complex case although 2D in fact, coded on 2 levels of caches in addition to the MRPD and data cache. The purpose of this description is to show step by step how our approach practically answers the question: how to update the page table hierarchy when new bricks are inserted?

Updating the Page Table Hierarchy ... in Action

The update of the hierarchy – with the addition of a brick – follows 3 sequential steps: *i)* removing the references of the bricks that will be deleted from the GPU data cache (if the cache is already full), *ii)* writing the new bricks in the GPU data cache and updating its LRU and finally *iii)* referencing these new bricks in the hierarchy.

Based on the example given in the figure 1, access to the purple and blue pixels of coordinates [5, 2, 0] and [9, 4, 0] (Fig. 1-d.1) raise a brick request respectively for the bricks number 8 and 16 of the level L0, noted L0.8 and L0.16 mutually. Considering these 2 bricks to add in the cache, we first select the last 2 bricks in its LRU (L1.1 and L0.10, Fig. 1-d.2) and we remove the link to these bricks from the PT2 cache by switching their respective flags to *unmapped* (Fig. 1-c.2.I). Once this step is performed, the old bricks are isolated and consequently are not reachable anymore from the MRPD. They are then replaced by the new bricks: the brick L1.1 is replaced by the brick L0.16 and the L0.10 by the L0.8 (Fig. 1-d.2).

Finally, the new bricks need to be referenced in the hierarchy. This step is performed in 3 sub-steps. We first need to know to which point the new bricks are referenced. Within the MRPD, an entry (Fig. 1-b.2, coordinates [0, 2, 0]) to the brick L0.16 exists in the PT1 cache. However, in the PT1 cache (Fig. 1-b.2, coordinates [2, 3, 0]) the flag for the entry to the brick L0.16 is set to 0, it means for all caches from this cache to the data cache, no PT entries exist to reference this brick. In a second step, for these caches we take the last PT entry in the LRUs (that has not been added during the current update) and remove its reference in its parent. In our case the last entry in PT2 cache is the entry L1.1 and its reference is removed from the PT1 cache (Fig. 1-b.2.II) by setting its flag to *unmapped*. This is done in order to dereference the last entry of the cache and being able to replace it with the new entry. Finally for the same caches we add new PT entries to reference the new brick L0.16 (Fig. 1-c.2.III) and reference it in its parent (Fig. 1-b.2.III). When done, these 3 final steps are performed once again for the brick L0.8. In general case, these steps are performed sequentially one brick after another.

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx