A Split Node Cache Scheme for Fast Ray Tracing

Jae-ho Nah* Department of Computer Science, Yonsei University

Jin-suk Heo† School of Electrical and Electronic Engineering, Yonsei University

Woo-chan Park‡ Department of Computer Engineering, Sejong University

Tack-don Han§ Department of Computer Science, Yonsei University

ABSTRACT

We propose a node cache scheme for efficient ray tracing hardware. The scheme uses an aspect that traversing high-level nodes have more locality. In this method, a node cache is split into a high-level node cache and a low-level node cache. The data of the high-level nodes is retained during one frame. In addition, this scheme has a hybrid tree layout. That is, the high-level nodes are represented to the breadth-first layout for a division of nodes, and the low-level nodes are represented to the depth-first layout for an effective use of locality. Simulation results show the reduction in cache miss ratio up to around three percent.

KEYWORDS: Ray tracing, cache scheme

INDEX TERMS: I.3.1 [Hardware Architecture]: Graphics processors

1 MOTIVATION

Nowadays, there are various approaches for real-time ray tracing. A dedicated hardware for ray tracing (ray tracing hardware, RT H/W) is one of them, and an efficient memory management is an important factor in the performance of the RT H/W. Therefore, the RT H/W has caches between processing units and memory. Among the caches, a node cache is used when a traversal unit reads tree nodes. According to Woop [1], a hit ratio of the node cache in the RT H/W can decline by up to 31.2 percent when the RT H/W renders complex scenes. Thus, an effective cache scheme is needed to increase the cache efficiency.

2 PROPOSED CACHE SCHEME

Our cache scheme is made up of two concepts, they are a split node cache and a hybrid tree layout. (Figure 1)

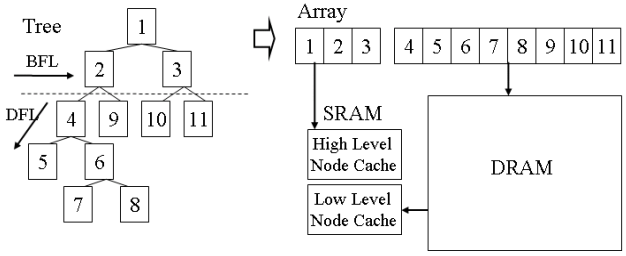

Figure 1. Proposed cache scheme.

Split node cache In the scheme, a node cache is split into two caches. One is a high-level node cache and the other is a low-level node cache. We apply different policies to these two node caches. Data of high-level nodes is retained during one frame, so cache misses don't occur. In contrast, the low-level node cache uses a common cache policy.

The reason of this splitting is an increase in cache locality. The space of parent node includes the space of child node, so the traversal processing in ray tracing uses a depth-first search. That is, access to the high-level node has more locality than access to the low-level node. Thus, we use different policies for the high-level node.

Hybrid tree layout In this section, we explain the tree layout for the split node cache. Typical tree layouts are the breadth-first layout (BFL) and the depth-first layout (DFL). The BFL fits the split node cache. A mapping between index of nodes and memory address of the nodes is easier than the DFL because we can distinguish levels of the nodes easily using the index of nodes. However, parent nodes and their child nodes aren't placed sequentially, so a locality can decrease in the traversal. Therefore, we propose the hybrid layout, high-level nodes are represented by the BFL, and low-level nodes are represented by the DFL. The hybrid layout doesn't have the locality problem, so it has advantages over both the BFL and the DFL in the split node cache.

We use a deque to represent the hybrid tree layout because the layout needs both breadth-first search and depth-first search. In addition, we modify traversal algorithm in ray shooting because an index format of child nodes is changed.

3 EXPERIMENTAL SIMULATION RESULTS

We implemented a Whitted-style ray tracer to verify our scheme. An acceleration structure of the ray tracer is the kd-tree. The ray tracer makes a statistical data of memory accesses, and the data is used for input of Dinero IV cache simulator.

Our test environment is as follows. The resolution is 800x600. The max depth of ray tracing is 10. The parameters of the node cache are eight-byte block and direct-mapped, and we assume that the high-level node cache and the low-level node cache have same size.

Table 1 shows the test results. In most cases, our scheme leads to reduced cache miss ratio.

Scene / Cache Size	**BART Robot** (7K triangles)		**BART Kitchen** (11K triangles)		**Fairy Forest** (17K triangles)	
	Without the split scheme	With the split scheme	Without the split scheme	With the split scheme	Without the split scheme	With the split scheme
4KB	12.42%	13.29%	13.63%	10.70%	11.33%	10.34%
8KB	5.52%	4.47%	7.12%	5.00%	7.01%	5.87%
16KB	1.82%	1.73%	4.44%	2.66%	4.29%	3.97%

Table 1. Experimental simulation results - cache miss ratio.

4 CONCLUSION

We propose the node cache scheme for fast ray tracing. We will release a new RT H/W using the split cache scheme in near future.

REFERENCES

[1] Sven Woop. Realtime Ray Tracing and Interactive Global Illumination. PhD thesis, Saarland University, 2007

E-mail: *jhnah@msl.yonsei.ac.kr, †hurjs0406@hotmail.com, ‡pwchan@sejong.ac.kr, §hantack@msl.yonsei.ac.kr