Resumo

Aluno: Tiago José de Carvalho

RA: 087343

Disciplina: MO401 (Arquitetura de Computadores I)

27 de abril de 2009

1 Resumo

[1] começa com uma introdução falando a respeito do z-buffer e suas limitações, explorando a seguir os desafios para a criação de um software, algoritmos e uma nova arquitetura que permitam efeitos visuais mais robustos e apresentando um sistema completo (que engloba todas essas características) chamado COPERNICUS, o qual possui um software (de nome RAZOR) altamente paralelo.

O RAZOR é o sistema proposto para traçado de raios e uma de suas características é suportar cenas dinâmicas com efeitos de alta qualidade. Ao contrário da abordagem tradicional ele recalcula a "cena" a cada frame, é paralelizado utilizando multi-thread e apesar de ser desenvolvido para arquiteturas futuras roda nos hardwares atuais sendo robusto o suficiente para extrair cenas de um jogo comercial.

No que diz respeito ao paralelismo o RAZOR é baseado em otimizações físicas que utilizam redundância para obter um melhor trabalho paralelo. Um de seus pontos chave é explorar a coerência nos raios primários e secundários. Como um resultado, redundantemente constrói a "KD-Tree" para todas as threads.

O código faz intenso uso da memória com apenas uma pequena fração de "branches". Porém a maioria desses branches possuem dependência de dados e não podem ser eficientemente suportados apenas com previsão.

A maioria do tempo de computação é gasto com três funções principais, as quais pegam o pacote de raios e cruzam com a estrutura de dados "KD-Tree".

Algo interessante a se observar é que melhoramentos de aplicações específicas podem melhorar a performance de maneira significativa para o RAZOR.

No que diz respeito a arquitetura do COPERNICUS é proposta uma organização de 2-níveis multicore com um conjunto de cores e uma cache L2 compartilhada formando um bloco. O chip completo é formado de múltiplos blocos e um controlador de I/O.

Cada bloco do core tem uma programação parecida com a de um core normal. Ao contrário da GPU, é esperado que a cache L2 apresente um bom comportamento e utilize entre 2 a 8 threads por core para esconder a latência de memória.

Cada bloco inclui uma cache L2 compartilhada que é conectada por um barramento cruzado aos diferentes cores. Os diferentes blocos são conectados através de uma malha de comunicação que eventualmente envia as requisições a um controlador de memória integrado. O sistema de cachel L1-L2 é desenvolvido para simplificar e segue um protocolo de coerência de cache simples. A cache L1 é internamente escrita e L2 mantém uma cópia das tags de L1.

Por fim são apresentados os resultados do COPERNICUS utilizando cinco benchmarks e um modelo de simulação completo, além de modelos analíticos para estimar a performance.

Referências

[1] Venkatraman Govindaraju, Peter Djeu, Karthikeyan Sankaralingam, Mary Vernon, and William R. Mark. Toward a multicore architecture for real-time ray-tracing. In *MICRO '08: Proceedings of the 2008 41st IEEE/ACM International Symposium on Microarchitecture*, pages 176–187, Washington, DC, USA, 2008. IEEE Computer Society.