Accessing CUDA features in the OpenGL rendering pipeline: A case study using N-Body simulation

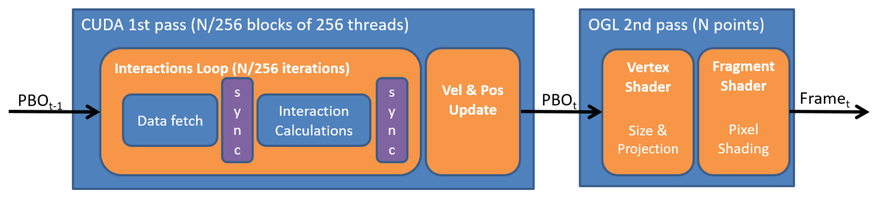
PAPER ID 30

M. S. Camillo, S.-T. Wu
Computer Engineering and Industrial Automation Dept., School of Electrical and Computer Engineering, Unicamp

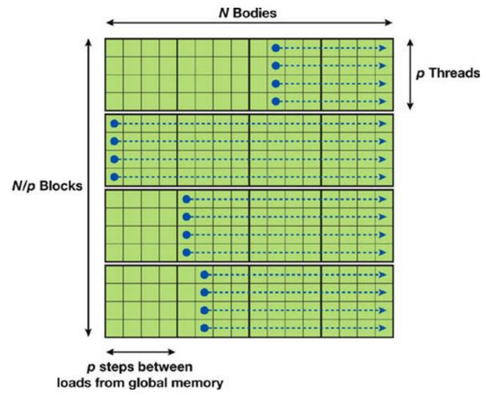
Introduction: Since their introduction to the consumer digital graphics market in the 1990s the GPUs (Graphics Processing Units) have evolved from specialized graphics processors to general purpose computing ones.
In general, for graphics rendering applications it is recommended to access 3D graphics specialized hardware through the open standard OpenGL API, and for general purpose programming the CUDA or the OpenCL APIs have been the preferred ones. The former abstracts the GPU architecture as a rendering pipeline, while the latter exposes more directly the underlying hardware features.
Several studies however show that using the OpenGL API even for compute heavy problems yield better performance [1,2,3,4].

Question: What is the potential of a series of new graphics features available in OpenGL 4.5 in closing the performance gaps between an OpenGL-based rendering pipeline and an optimized CUDA-based implementation?

Comparative Analysis:
Basis of comparison: A well known and optimized CUDA implementation of an N-Body simulation available in the Nvidia CUDA SDK (Figures (a) and (b))

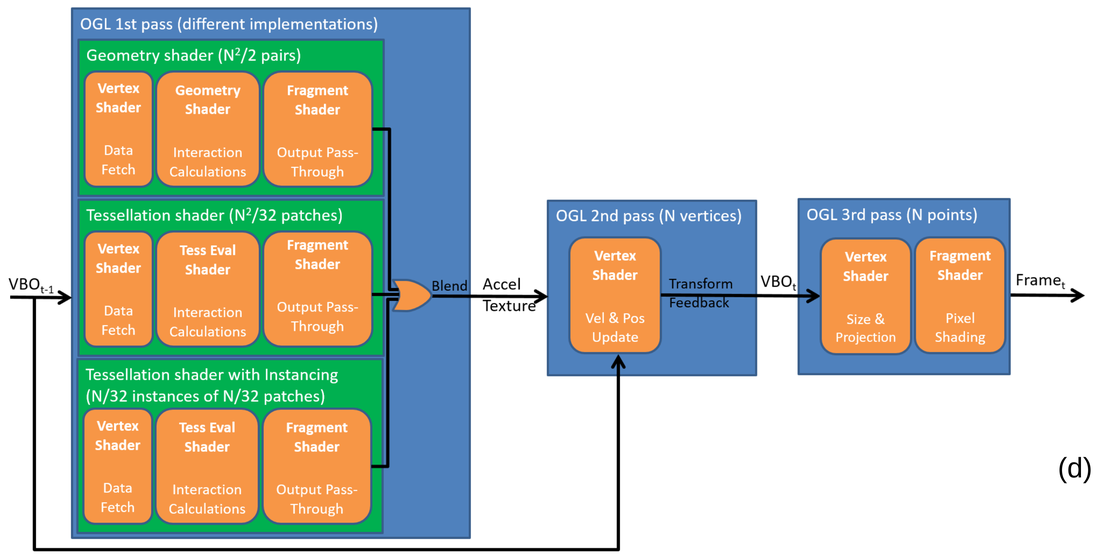


(a) Data flow for CUDA implementation

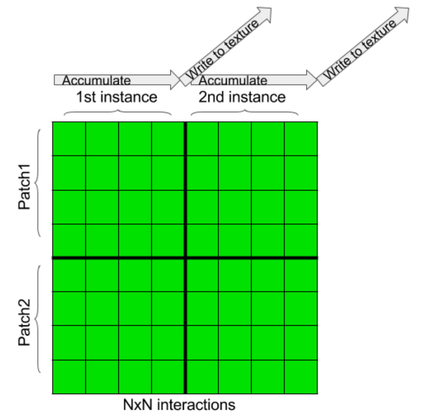


(b) Execution pattern in CUDA

Compared solutions: Three different OpenGL implementations (Figure (c)) to approach the execution pattern in CUDA:
•Geometry shader: streaming pairs of bodies to try to maximize parallelism
•Tessellation shader: access to shared memory as the tessellation shader pre-loads its patch data into it.
•Tesselation Instancing: Organize memory access and execution flow. Most similar pattern to CUDA implementation (Figure (d))

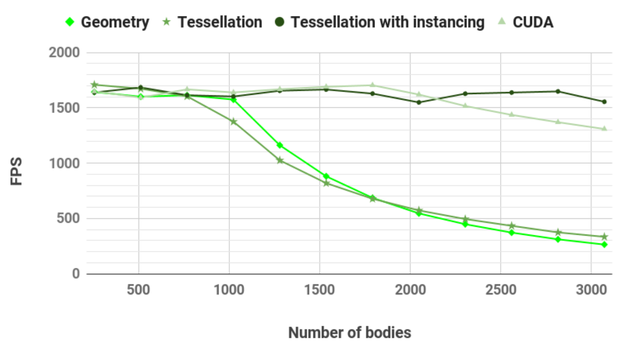


(c) Data flow for OpenGL implementations

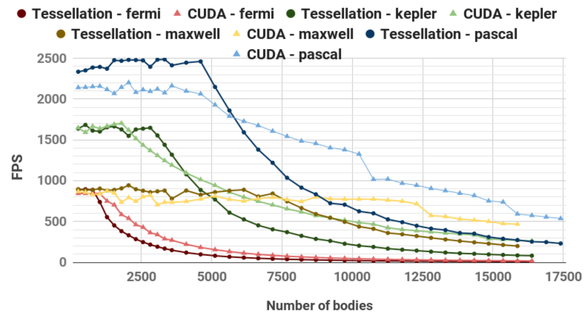


(d) Execution pattern in tessellation with instancing

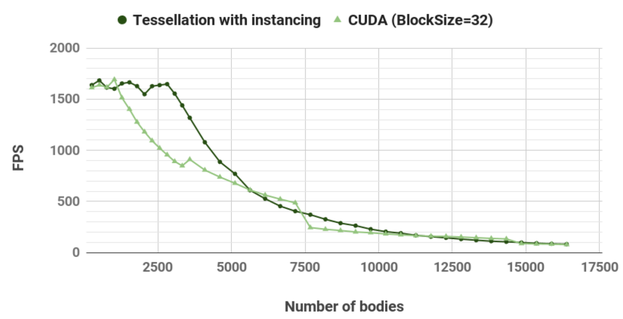
CUDA x OpenGL: We found that the geometry and the tessellation features do quickly degrade even with small numbers of bodies, while CUDA and tessellation with instancing maintain a good performance even when the number of bodies grows.
Tessellation with instancing performance drops for large number of bodies compared to CUDA. This occurs because in the former we have 32 bodies per patch and in the latter we have 256 bodies per block. When comparing both implementations with the same number of bodies per processing group we obtained quite similar curves. The limitation of the number of vertices in each patch along the OpenGL rendering pipeline impedes it achieving the same performance as CUDA. Empirically, we come to the following conjecture: use of geometry shader may improve the execution parallelism; tesselation shader provide access to shared memory; and instanced rendering may optimize memory access pattern.



(e) Performance comparison of all 4 algorithms



(f) Performance comparison on latest Nvidia GPU architectures



(g) Performance comparison CUDA and Tessellation with instancing with blocksize=32

Conclusion: Memory accesses are the major performace bottleneck. OpenGL features improving these accesses lead to better performance. Applying our findings in the implementation of a memory-intense, but highly parallelizable, N-body simulation, we got an OpenGL-based implementation that rivals a well optimized CUDA-based version. We believe that these findings are useful for devising new performance optimization strategies for the applications developed with the OpenGL API. Nevertheless, to fully benefit from the hardware resources we should deepen our understanding of some unexpected behavior patterns by profiling our shaders with an appropriate tool such as Nvidia Nsight.

References: [1] M. Fratarcangeli, in Game Engine Gems 2, 2011, ch. 22, pp. 365–378;
[2] T. I. Vassilev, in CompSysTech'11. NewYork,NY, USA: ACM,2011,pp.204–209;
[3] J. Hunz, Available: https://kola.opus.hbznrw.de/files/786/JochenHunzBachelorThesis.pdf ;
[4] F. Sans and R. Carmona, in XLII CLEI, Oct 2016, pp. 1–11;.