7
Conclusion and Future Work

In this work, we propose a strategy for GPU-based parallel fracture, microbranching, and fragmentation simulations based on the extrinsic cohesive zone model. Such parallel simulations impose two main challenges. First, we need to be able to handle mesh modifications because cohesive elements, bulk elements, and nodes are inserted (and removed) adaptively along the simulation. Second, we need to efficiently perform intensive numerical computation, with numerous memory accesses, in parallel. Using a simple topological data structure, shared memory, kernel splitting, texture fetch, and minimizing global memory accesses, we could effectively map and optimize the CPU implementation of a fragmentation simulation to a GPU environment, taking advantage of CUDA benefits. Mesh coloring proved to be an effective means to avoid race conditions, but the gather strategy also took advantage of the absence of the coloring technique in the adaptive simulation.

Investigation of adaptive refinement and coarsening schemes on the structured 4k mesh for dynamic fracture simulation on the massively parallel GPU architecture reveals insight into intricacies of the numerical simulation. First, a specialized data structure and new approach to performing finite element calculations in parallel was detailed. The race condition and expensive graph coloring algorithms are avoided by performing finite element calculations on a nodal basis. Nodal quantities are gathered by launching threads per nodes and accumulating element contributions rather than by launching threads per elements. Using the assumption that areas near crack tips need to be the most refined and a strain criteria to determine where elements can be coarsened, we adaptively change the mesh resolution during the simulation. We detail the parallel algorithms to systematically change the topology of the mesh.

The variations that normally occur during floating point operations are not usually apparent in serial or even parallel fracture simulations on structured meshes. This is because the order in which operations are performed, and thus the accumulation of variation, is usually the same from one simulation to the next. However, in the present implementation, new elements and nodes are inserted in a non-deterministic order, meaning that the quantities added to the node are not done so in the same order from one simulation to another due to the chosen adjacent element to the node. While not incorrect, the result is that fracture patterns look quite different from one simulation to the next. To demonstrate the validity of the approach given the very different appearance of the fracture pattern, we quantified the crack patterns through several parameters and showed that those that are physically based agree well between simulations. Interestingly, the parallel approach adds some randomness into the finite element simulation on the structured mesh in a similar way as a would be expected from a random mesh.

Thanks to a data structure and adaptive mesh modification scheme developed specially for the GPU architecture, we are able to represent much larger finite element meshes than without adaptivity. With the large scale simulation of the microbranching problem, we are able to make more direct comparisons to the original experiment and find excellent agreement with those results.

Experiments with distributed 3-dimensional simulation indicate that the main bottleneck is concentrated in the sending and receiving of messages via network. Although we greatly reduced this bottleneck by duplicating ghost nodes, the size of the message is still large (proportional to the number of *ghost nodes* in each partition) and the GPU parallelization over CPU implementation decreased analysis code bottleneck and the node synchronization dominated the simulation time. Another advantage of duplicating ghost nodes is saving additional *kernel* processing and memory consumption.

Although nodal synchronization dominated the simulation time, the performance of the large-scale models were significant compared to the large-scale simulations on the CPU performed by Espinha et al. [14], considering the size of the problem, problem type, number of time steps, and time step size. Our reduced large-scale mesh ran in 15 minutes, while the increased size large-scale mesh ran in approximately 82 minutes.

Finally, while METIS does its best to balance the number of bulk elements during mesh partitioning, we noticed an uneven distribution of specific tasks among partitions. The main reason is due to the main crack location, which leads to different number of cohesive elements between partitions. Cohesive insertion and force calulation *kernels* process a significant different number of threads depending on the main crack location. It is worth remembering that cohesive forces *kernel* is one of the most costly *kernels* of the simulations, with numerous arithmetic operations, even with the strategy of splitting it into different *kernels* and at the same time avoiding additional global memory consumption.

Regarding the Physics-based animation framework, our physical technique to switch between rigid body and fracture modes, i.e., when to treat the mesh as rigid body or as a breaking object by inserting cohesive elements during collision, is to detect when we do not insert cohesive elements during a certain number of steps. We leave as future work a way to automatically detect when to switch back to rigid body mode. A technique inspired by engineering analysis could be a strain-based criteria much like when coarsening the mesh in the adaptive simulation. For example, when the strain energy is high, we switch to fracture mode. When the strain energy is low enough, fracture mode is switched to rigid body dynamics mode.

A natural extension of this work to the adaptive simulation would include all three dimensions. However, for a single GPU, the problem size would be quite limited, which is not well suited for three-dimensional finite element applications. The node traversal (gather strategy) would have to be used while coloring would have to be eliminated from the mesh. This would decrease performance since the gather strategy for 3D meshes is much more costly. Therefore, current development focus on distributed computing where different parts of the model would be simulated on different GPUs.

The implementation of fracture simulations with polygonal finite elements is considered for future work. It would be a great challenge to extend our data structure, especially regarding the adjacency since each bulk element contains a varied number of facets.