

ZippyGPU: Programming Toolkit for General-Purpose Computation on GPU Clusters

Zhe Fan, Feng Qiu, Arie Kaufman *

Computer Science Department and Center for Visual Computing (CVC)
Stony Brook University, Stony Brook, NY 11794-4400

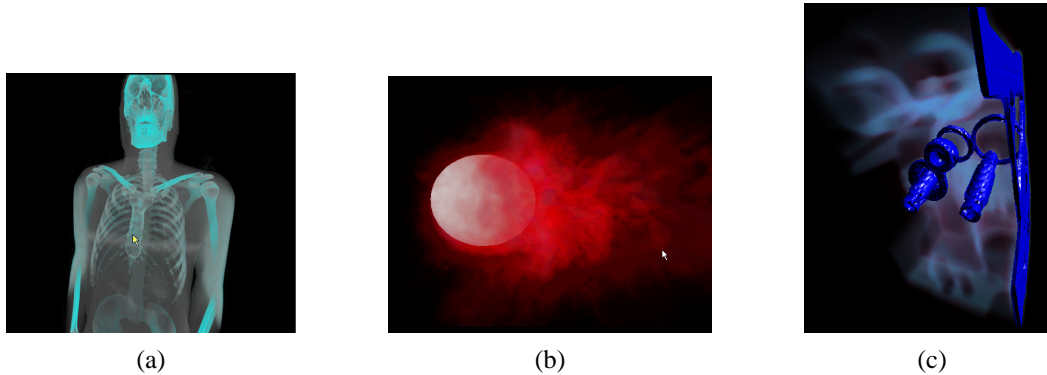


Figure 1: Three experimental applications running on our GPU cluster, developed using ZippyGPU: (a) parallel volume rendering of a large volume dataset (Visible Human on 8 GPUs); (b) real-time flow simulation using the LBM and visualization (gas flow around a sphere on 16 GPUs); and (c) real-time Marching Cubes iso-surface extraction and rendering (industrial CT of an engine part on 4 GPUs).

A GPU cluster is a distributed-memory architecture. Programming it requires substantial experience in parallel programming and GPU programming. Especially, the programmer needs to deal with the network communication and CPU-GPU data transfers explicitly. To implement complex applications running efficiently on GPU clusters can be tedious and difficult for programmers. In this work, we present ZippyGPU, a toolkit that facilitates programming of general-purpose computation on GPU clusters.

ZippyGPU is an object-oriented library sitting above the graphics APIs, OpenGL and Cg, and the communication API, MPI. At the low level, it encapsulates and manages the objects and functionalities provided by the above APIs, which are often used in GPU-based general-purpose computations.

Following the concept of the Global Arrays (GA) Toolkit [1], ZippyGPU combines shared memory and distributed memory programming models to make the inter-GPU data communication easy-to-code and efficient-to-execute. Its high-level data structure, the global array, is an N-dimensional array (N ranges from 1 to 7) distributed on multiple GPUs. This data structure provides an abstraction of virtually shared address space, so that the access of remote data can be easily accomplished using shared-memory-style library functions of ZippyGPU, such as copy and user-customized composite (see Figure 2). Meanwhile, the locality information of data is explicitly managed by ZippyGPU and is accessible to programmers. Therefore, the Non-Uniform Memory Access (NUMA) nature of the GPU cluster, that is the local GPU memory is faster to access than remote GPU memories, can be recognized and utilized for efficient communication. For example, utilizing the data locality information, the programmers are able to implement the communication in their applications as coarse grained as possible in order to maximize the communication performance.

Applications of ZippyGPU include computation, visualization, and systems combining them. Figure 1 shows three examples that we have developed on our GPU cluster. In the first application, a parallel volume rendering system, ZippyGPU allows us to treat all

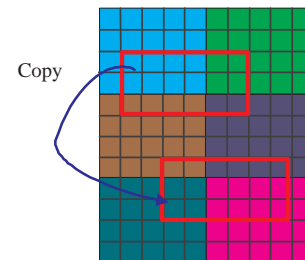


Figure 2: An example of 2D global array on a GPU cluster. Different colors represent different sub-blocks that reside in the memories of different GPUs. Shared-memory style functions, e.g., copy or user customized composite, can be used to easily operate on the data in the virtually shared address space.

partial images as a whole large image and simply implement the parallel image compositing using the shared-space operations. In the second example, we rewrote our previous hand-written GPU-cluster based implementation of the lattice-Boltzman model (LBM) flow simulation. Achieving the same performance, our new implementation takes only 100 lines of C++ code for the inter-GPU communication while a previous implementation requires 2800 lines of C++ code for the same task. The last example is a Marching Cubes iso-surface extraction and rendering system running on multiple GPUs.

References

- [1] J. Nieplocha, R. J. Harrison, and R. J. Littlefield. Global Arrays: A nonuniform memory access programming model for high-performance computers. *The Journal of Supercomputing*, 1996.

* {fzhe, qfeng, ari}@cs.sunysb.edu