Creating the Chestnut Parallel Programming Language

Andrew Stromme, advised by **Tia Newhall**

Swarthmore, Pennsylvania
astromm1@swarthmore.edu, newhall@cs.swarthmore.edu

Abstract

Chestnut is a new language for parallel programming that was designed and implemented this summer. It aims to capture the power of graphics cards for performing general purpose processing. Existing languages and APIs for graphics card computation such as CUDA C and OpenCL expose enormous raw computing power but are difficult to learn and even more difficult to master. These languages require knowledge of how graphics cards work and use a low level syntax. Chestnut removes this complexity by focusing on a parallel context model that does not require knowledge of the underlying hardware. Chestnut borrows from the Mint[1] assumptions regarding locality and can therefore be applied to the same sorts of problems as Mint can (such as heatflow simulation). Mint focuses on the stencil model where computations are dependent on physically-local data.

Visualizations are natively supported and are designed to be fast. Programmers use the normal Chestnut model to specify how to create each pixel in a display window with full access to their existing data. Also included is the Chestnut Designer, a graphical environment which aims to provide an even simpler introduction to the language and its concepts. Employing a drag and drop interface with color coding and shape coding for inputs and outputs, the designer is a good way to model problems visually. This designer pulls concepts from the Scratch[3] graphical language and environment.

This summer a prototype of the Chestnut compiler was implemented in Python; this compiler transforms Chestnut code into CUDA C. I worked with my advisor Tia Newhall from the Swarthmore CS Department and will continue my research this fall for my senior thesis. Initial performance comparisons are very positive with Chestnut programs performing orders of magnitude faster than their sequential counterparts while only trailing behind hand-tuned CUDA versions by reasonable amounts.

Literature cited

[1] Unat, D., Cai, X., & Baden, S. (2011). Mint: Realizing CUDA performance in 3D stencil methods with annotated C. *Proceedings of the ACM International Conference on Supercomputing ICS*.

[3] Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., Kafai, Y. (2009). *Scratch: Programming for All*. Communications of the ACM, November 2009.