Distributed Graphical Visualizations on a Bramble of Raspberry Pis

James Felton, Adam Al-Ibrahim, Chris Johnson, and Peter Bui
Department of Computer Science
University of Wisconsin - Eau Claire
Eau Claire, WI 54702
{feltonj, alibraay, johnch, buipj}@uwec.edu

Abstract

Raspberry Pis are inexpensive but fully capable credit card sized computer boards designed to encourage students to experiment with computing. By clustering multiple Raspberry Pis into a *bramble*, we have constructed a cheap and powerful distributed system. For our demonstration, we will present the use of our Raspberry Pi cluster in two distributed graphical visualization applications. Specifically, we will present one cluster of Raspberry Pis driving a video display wall consisting of multiple monitors and a second cluster executing a multi-player network game.

Our approach to building an inexpensive distributed system on a budget was to cluster multiple Raspberry Pis into a bramble. Although the CPU on these devices is relatively weak compared to those found in conventional computers, the GPU on the Raspberry Pis is quite powerful and supports OpenGL ES 2.0, which means that applications running on these devices can utilize shaders and other advanced computer graphics techniques.

We take advantage of having multiple nodes with significant graphical processing power by networking the individual Raspberry Pis into a cluster. Moreover, we attach monitors to each separate node so that each node is responsible for a portion of the screen space. With such a system configuration, we implemented a video display wall where portions of each video frame are rendered by separate nodes while maintaining synchronization. Additionally, we also programmed a networked multiplayer game where each board is responsible for a segmented portion of the game space.

For our video display wall, we utilize OpenCV to decompose a video file into a stream of frames that are then rendered by each individual Raspberry Pi directly to its video card's framebuffer using OpenGL ES. To synchronize the video playback, we rely on MPI's barrier mechanism. Our video game, Four Lords, has a similar implementation. Each board renders its portion of the game space using OpenGL ES and maintains synchronization using MPI.