**Senior Project – Computer Science – 2013**

# *Visualization of Parallel Programming*

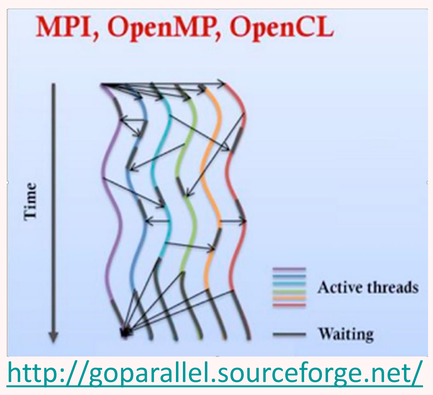
Andrew Schwartz

Advisor – Prof. Valerie Barr

Objective: Create a tool to aid in teaching best practices of parallel programming through the use of profiling and visualization of simulated standard message passing function calls in a user-defined network topology.

## PROBLEM

Conventional teaching methods for introductory courses in parallel programming insufficiently explain the data flow of a network during standard parallel programming design patterns. This can lead to poor design decisions when developing an application. Students are taught with static diagrams when the execution of a program is inherently dynamic. In order to fully grasp the underlying methods behind message passing in a network, it is necessary to see an active visualization of the process.




http://goparallel.sourceforge.net/

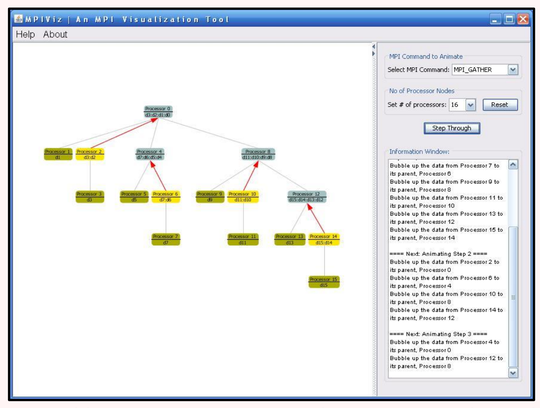
This static diagram illustrates the difficulty in conceptualizing data flow in a parallel program.

## BACKGROUND

Parallel programming visualization as a teaching aid has been attempted before by Virginia Tech, but lacks important profiling features. Parallel program profilers report too much information for use as a teaching aid, and are generally meant for developers optimizing their code, not students trying to learn the paradigms.

MPIViz (2008)

- Developed at Virginia Tech
- Options very limited
  - Number of nodes
  - Function
- Information very limited
  - Log and visual cues
  - No numerical data



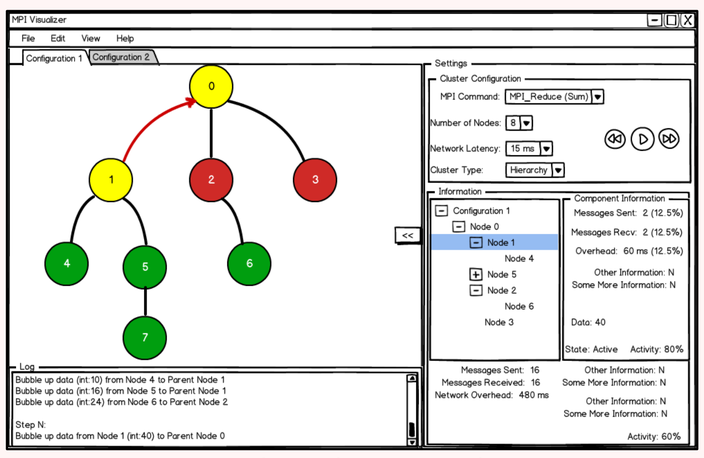

http://research.cs.vt.edu/AVresearch/MPI/

## DESIGN GOALS

The minimalistic design of MPIViz has good usability, but lacks features that could prove beneficial to understanding the effects of different topologies on different function calls.

The goal is to retain the positive attributes (usability, visual cues) of MPIViz, while also intuitively incorporating more information.

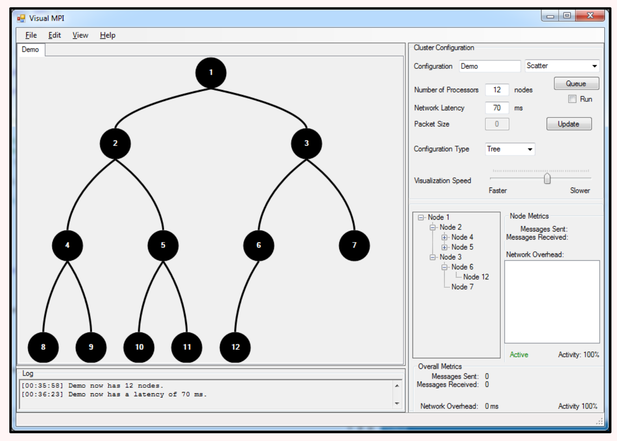
- Retain
  - Colorful nodes
  - Connection highlighting
  - Log
- Add support for:
  - Multiple configurations
  - Node analysis
  - Different topologies




Created in Balsamiq Mockups

## IMPLEMENTATION

The program uses OO programming paradigms to represent a model of each configuration as an object. Configurations then encapsulate a set of nodes which carry both state information and profiling statistics. The custom Tab View's paint method is overwritten to display the current configuration's state given its model. The simulation emits events that modify the state of a model. Thus, through a specially crafted series of events determined by the topology and function, we can simulate each step of the algorithm and update the view accordingly.




WinForms UI

- WYSIWYG
- Quickly iterate changes
- Spend more time on code rather than UI

Custom Tab View

- Overwrite paint
- Becomes canvas for visualization

For each topology, a specific set of events needs to be crafted to accurately simulate the function on the given topology. Topologies were restricted to rings and trees for this reason. An algorithm to construct the proper set of events for each function dynamically creates the queue at runtime for the user-defined topology. The bookkeeping for profiling information takes place in the event queue handler which adjusts each node's state depending on the type of event emitted.

C#.NET

- Visual Studio IDE
- WinForms Compatible
- OO
- Cross-platform with Mono (UNIX/Windows)
- Java-like familiar syntax
  - But don't have to use Swing/other UI

```
private void eventTimer_Tick(object sender, EventArgs e)
{
    ConfigEvent configEvent = null;
    do
    {
        if (eventQueue.Count == 0)
        {
            break;
        }
        else
        {
            configEvent = (ConfigEvent)eventQueue.Dequeue();
            currentConfig = configEvent.run();
            eventQueueBox.Items.RemoveAt(0);
            UpdateFormFromConfiguration(currentConfig);
        }

    } while (configEvent is StateEvent);

}
```

Mainform.cs 407-425

## CONCLUSIONS

- Core design challenges of the project were met
- Some quality of life/usability functionality still left to be completed
- Extendable project design, just need to come up with algorithms to construct new event sets and it can be added
- Port to UNIX while simple through Mono migration, hasn't been tested.
- Initial results seem promising but still need to test the efficacy in the classroom.