Driving The Gooda Visualizer
point browser at gooda-visualizer/index.html

Modern superscalar, e-out-of-order microprocessors dominate large scale server computing. Monitoring their activity, during program execution, has become complicated due to the complexity of the microarchitectures and their IO interactions. Recent processors have thousands of performance monitoring events. These are required to actually provide coverage for all of the complex interactions and performance issues that can occur. Knowing which data to collect and how to interpret the results has become an unreasonable burden for code developers whose tasks are already hard enough. It becomes the task of the analysis tool developer to bridge this gap.

To address this issue, a generic decomposition of how a microprocessor is using the consumed cycles allows code developers to quickly understand which of the myriad of microarchitecture components they are bottling, without requiring a detailed knowledge of the microarchitecture. When this approach is intrinsically integrated into a performance data analysis tool, it enables software developers to take advantage of the microarchitectural methodology that has only been available to experts. The Generic Optimization Data Analyzer (GOODA) project integrates this expertise into a profiling tool in order to lower the required expertise of the user and, being designed from the ground up with large-scale object-oriented applications in mind, it will be particularly useful for large mpi codes.
Open a report (click on Samples) Metrics in **GREEN** are in cycles
expand the process to show modules
Click on the Magnifying glass to expand the metric
expand the Load_latency metric to display its components
Expand the sub-component: Dtlb_latency
Double click on Function name to go to source view
Three pane source view
Collapse the basic blocks
Sort by cycles
Move all 3 panes in unison
Expand 1 basic block
The Call Count Graph
Expand to see immediate sources and targets
Shrink the graph