TEJAS VISUALIZER

Visualization tool for Tejas output files

Introduction

Tejas is an open-source, Java-based multicore architectural simulator built by Srishti research group, IIT Delhi. It includes support for Java applications simulation, multicore simulation, cache coherence, NUCA protocols, and power simulation.

Tejas Visualizer is a Java-based desktop application, which provides a GUI for the results generated by Tejas. The visualizer provides a neat interface for the display and analysis of various parameters for any simulation done on Tejas. The visualizer can show and compare features of a single or even multiple benchmarks at a time.

Prior Work

There exists a couple of applications which support some visualization, *Sniper*, *weka* are example of such tools. But they do not cater to the needs of Tejas. Sniper is web-based, while weka is quite generalized. Most importantly, none of them have capabilities to compare and analyze multiple features simultaneously or compare multiple benchmarks.

Innovations

Tejas visualizer provides following improvements over existing applications:

1. Java-based: Seamlessly integrates with Tejas.
2. Can compare multiple features for a single benchmark graphically.
3. Can compare features of multiple benchmarks graphically.
4. Calculates and displays correlation vector of a feature on demand.
5. Calculates and displays correlation vector for rate of change of a feature with other features.

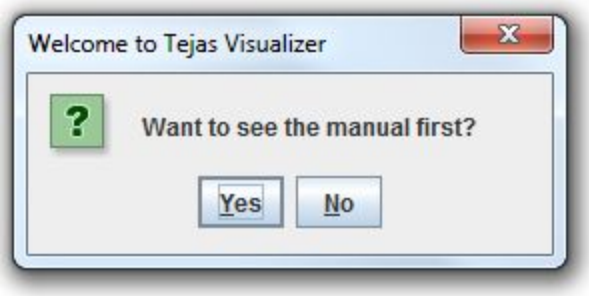
6. Ability to export all the calculated data to an external file.
7. User can dynamically provide the epochs value.

Work Done

- Studied the architecture, source code and data flow of Tejas.
- Modified the Tejas source code to generate a new dump file containing values of various parameters during simulation(*End of report).
- Designed, and implemented the GUI for Visualizer.
- Added components for calculation of correlation, average, minimum value and maximum value for single and multiple features.
- Designed a module which displays and saves graphs of various features(single and multiple benchmarks) for comparison.
- Implemented an export option that can export the displayed details to a text file.

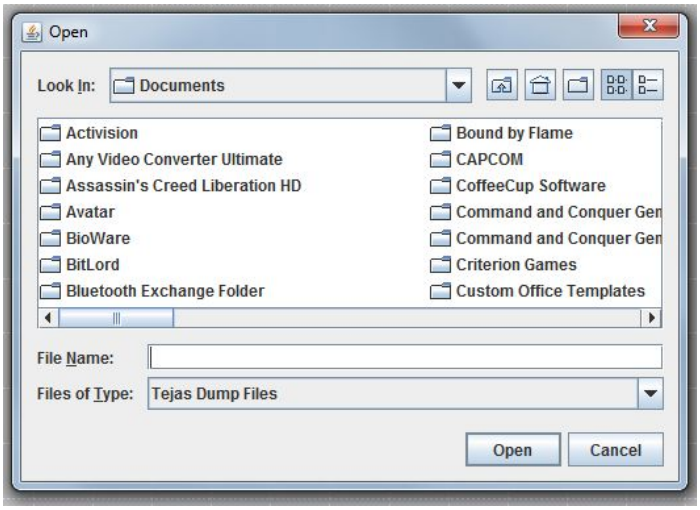
Working of Tejas Visualizer

1. When you launch the application, it asks the user whether he/she wants to see the manual first or not.

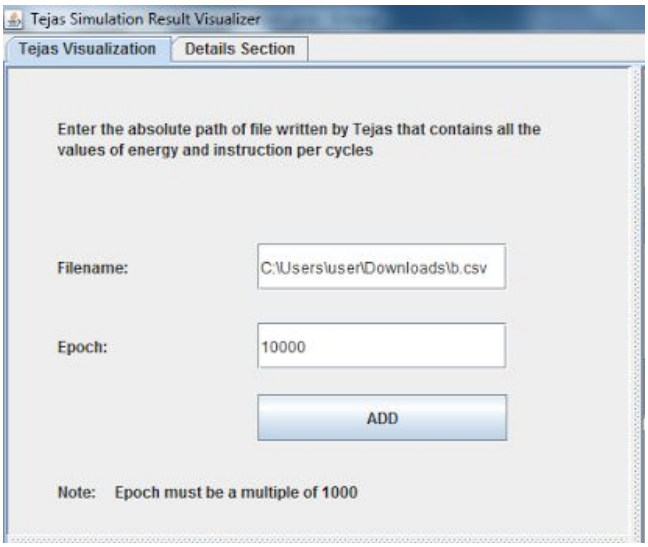


2. If a user choose to open the manual then an extra tab for manual is added to the visualizer.
3. If a user chooses to open the application without manual, following sequence of events will happen.

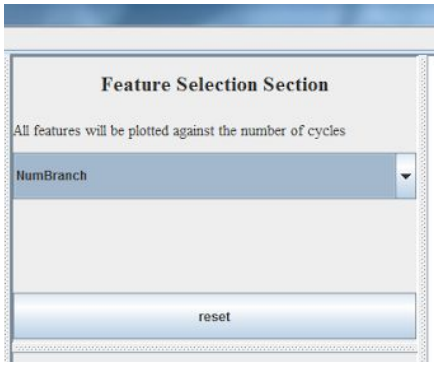
a. When user clicks on filename textfield, an open dialog box pops-up and asks user to select a Tejas output file to load into the visualizer.



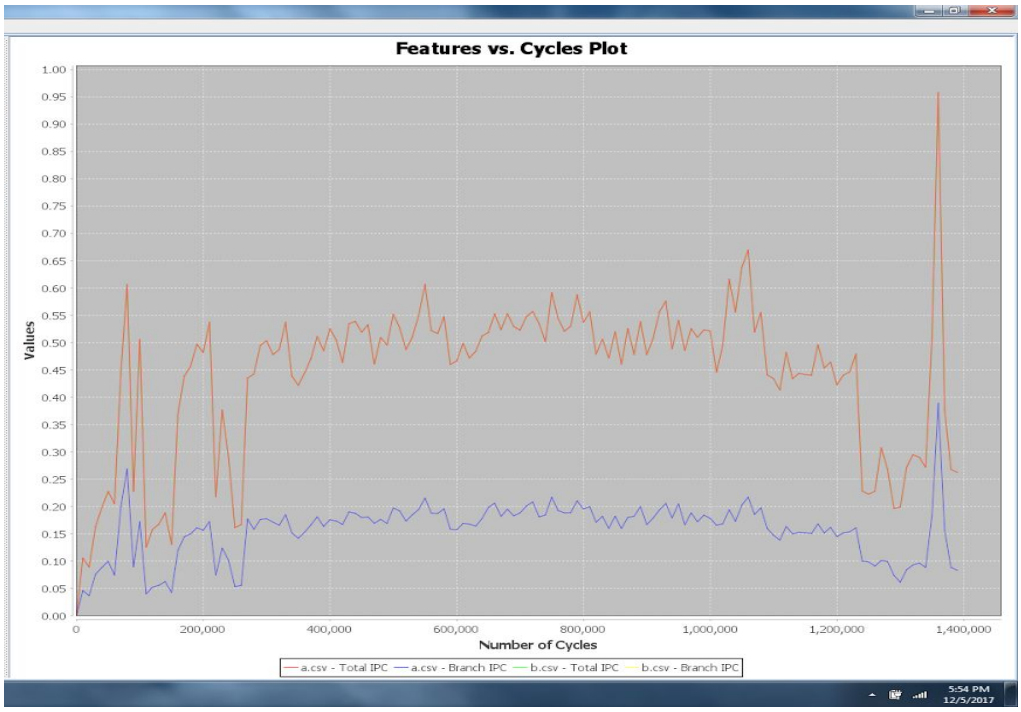
b. Then user can enter the epochs(for calculating rate of change of features) and click on add button to add the file under consideration to visualizer.



c. Then user can select the features he wants to analyze from the drop-down box

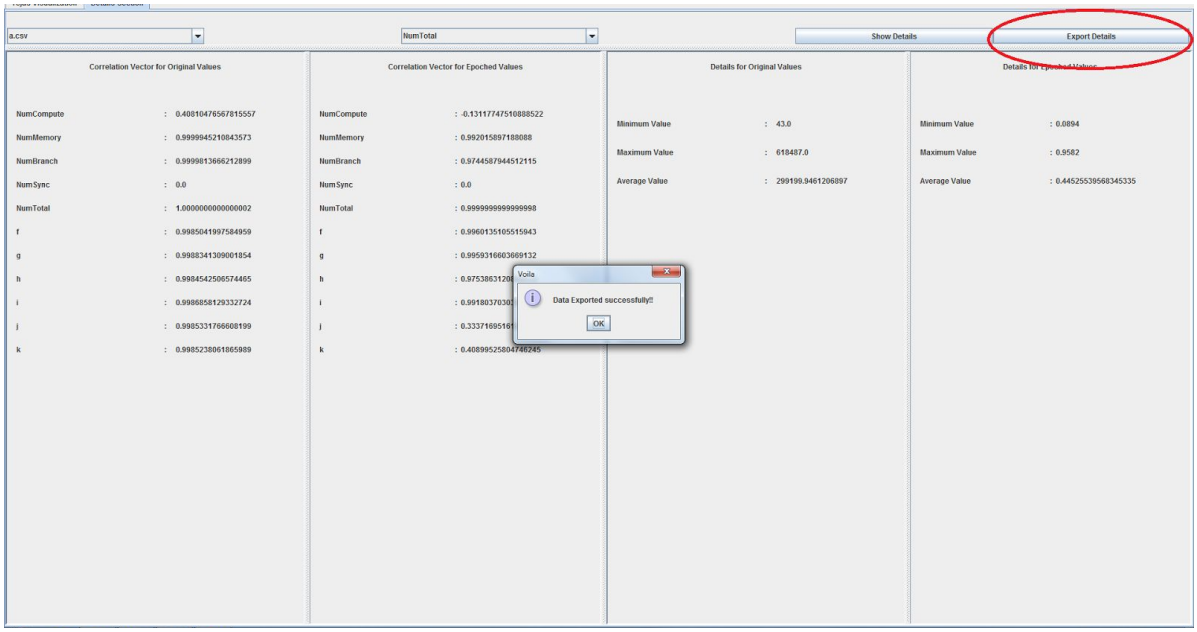


and the graph for the selected features will be plotted.



4. A user can switch the tab to the details section to see the correlation vector of a feature as well as other details that may be useful to a user.

5. User can also export all the data he is viewing currently, to an output file.



6. A user can also save the graph currently in view to a desired location.

Future Work

We can discuss with srishti research group, IIT Delhi regarding what other features they would like to visualize from Tejas and work on it.

List of files modified in Tejas Source Code

FILE NAME	MODIFICATIONS
DecodeLogic.java	Added a function to calculate energy spent in decode logic.
ExecutionLogic.java	Added a function to calculate energy spent in execution logic.
InstrutionWindow.java	Added a function to calculate energy spent in instruction window.
RegisterFile.java	Added a function to calculate energy spent in register file.
RenameLogic.java	Added a function to calculate energy spent in rename logic.

RenameTable.java	Added a function to calculate energy spent in rename table.
ReorderBuffer.java	Added a function to calculate energy spent in reorder buffer.
OutOfOrderExecutionEngine.java	Added 11 functions that receives the energy from different classes.
MultiIsseInorderExecutionEngine.java	Added 11 functions that receives the energy from different classes.
ExecUnitIn_MII.java	Added a function to calculate energy spent in execution unit.
WriteBackUnitIn_MII.java	Added a function to calculate energy spent in write back unit and a function to set the number of instructions for their types.
DecodeUnit_MII.java	Added a function to calculate energy spent in decode unit.
ExecutionEngine.java	Added abstract methods to get energy from classes.
TLB.java	Added a function to calculate energy spent in TLB.
LSQ.java	Added a function to calculate energy spent in LSQ.
Cache.java	Added a function to calculate energy spent in Cache.
ArchitecturalComponent.java	Added a function to get the number of instructions executed in every category(Memory, branch, synchronization, compute). Added functions to get energy from iCache, iTLB, dCache, dTLB., and some other internal components.
OperandType.java	Changed the class(gave integer representation to operands, created enumeration).
Core.java	Added functions to get energy and count of instructions executed from internal components.
RunnableThread.java	Added imports and other necessary stuff to get the charts generated during runtime in Tejas. Also added the file handling and energy calculation code to generate dump file from Tejas during run time. Created functions to interact with the JFreeChart charting library used for graphs.