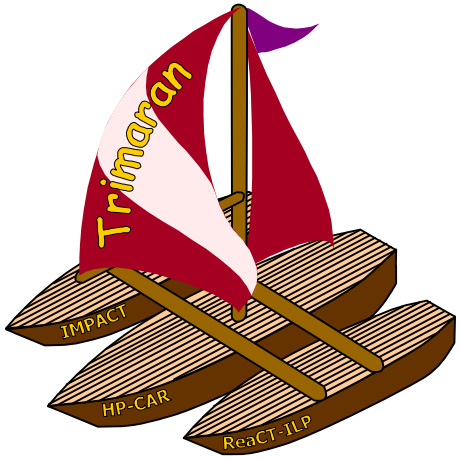
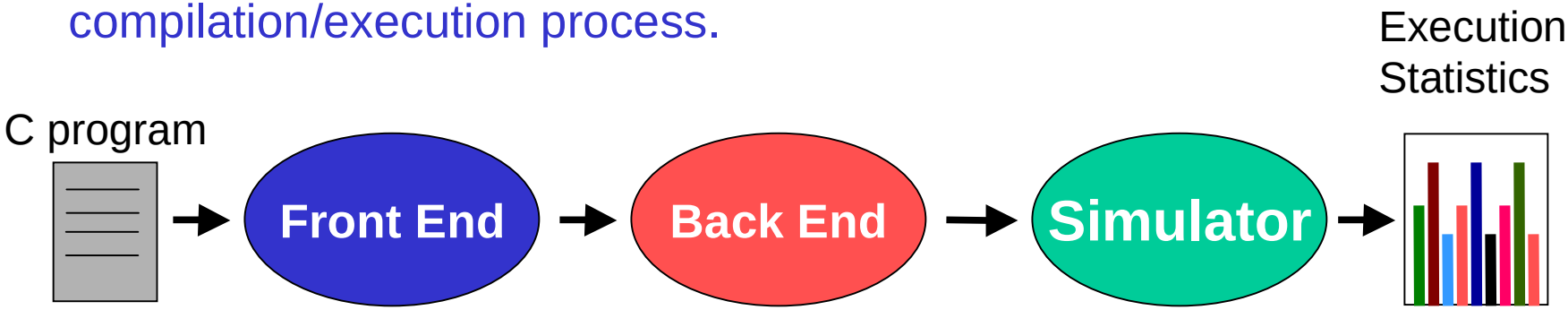



# The HPL-PD Simulator and Performance Monitoring Environment

# User View of the Simulator

- To the user, the simulator is simply another phase of the compilation/execution process.




- Transparent to the user, Makefiles guide the
  - Configuration of the simulator using MDES
  - Generation of “executable” code from the Rebel output of the back end.
  - Creation of interface for “foreign calls”
    - to C routines provided by the user or as part of a standard library.
- A GUI is provided to extract and analyze the execution results of the simulator.

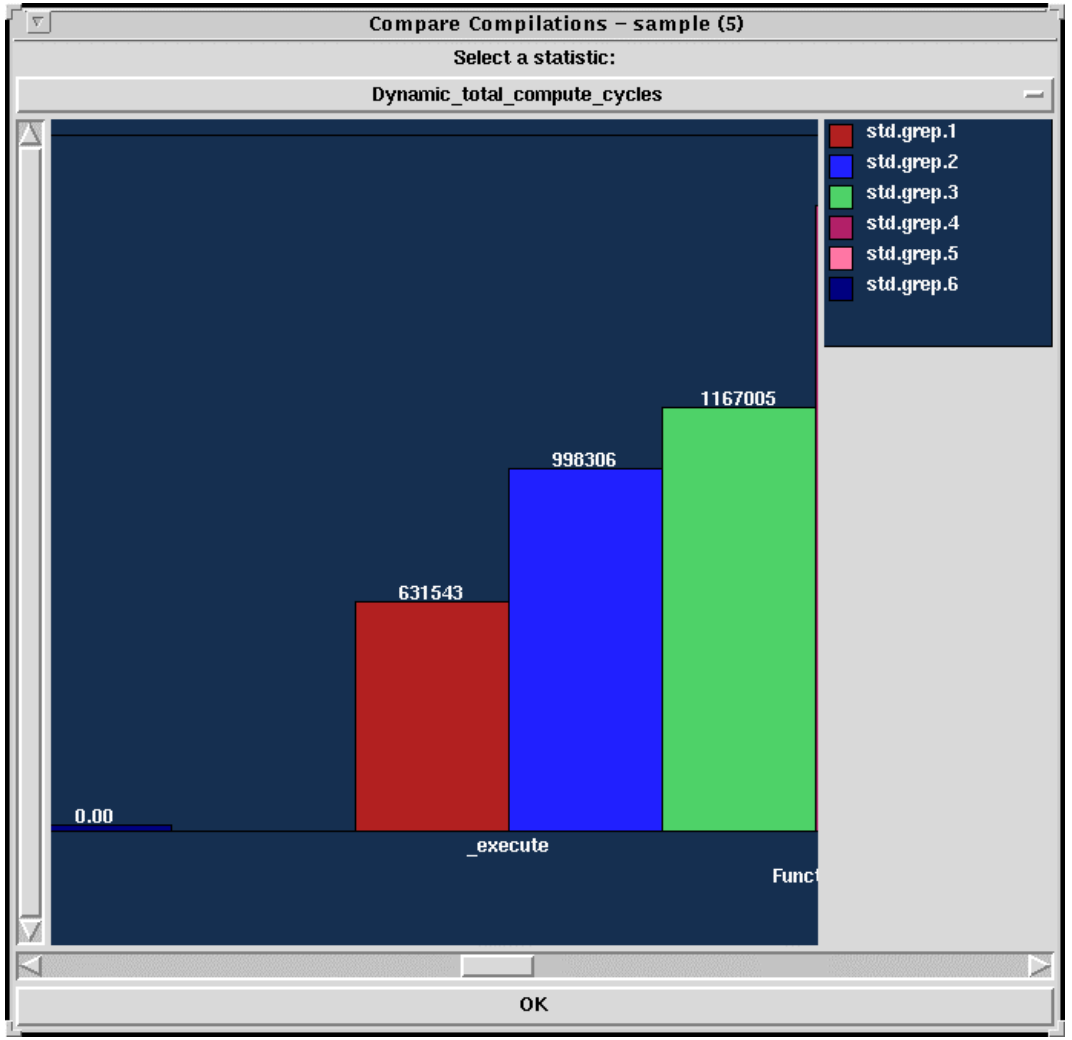
# Execution Results

- During execution, the simulator produces raw data, namely a trace specifying
  - Control flow execution
    - gives the order of control-block execution
  - Memory addresses referenced
  - Guarded predicate values
    - whether an operation within a HPL-PD instruction was disabled by predication.
- A trace-driven profiler tool is run after execution.
  - Reads the trace, and Rebel file(s), and extracts the desired information.
  - Emits a detailed statistics / profile information file.

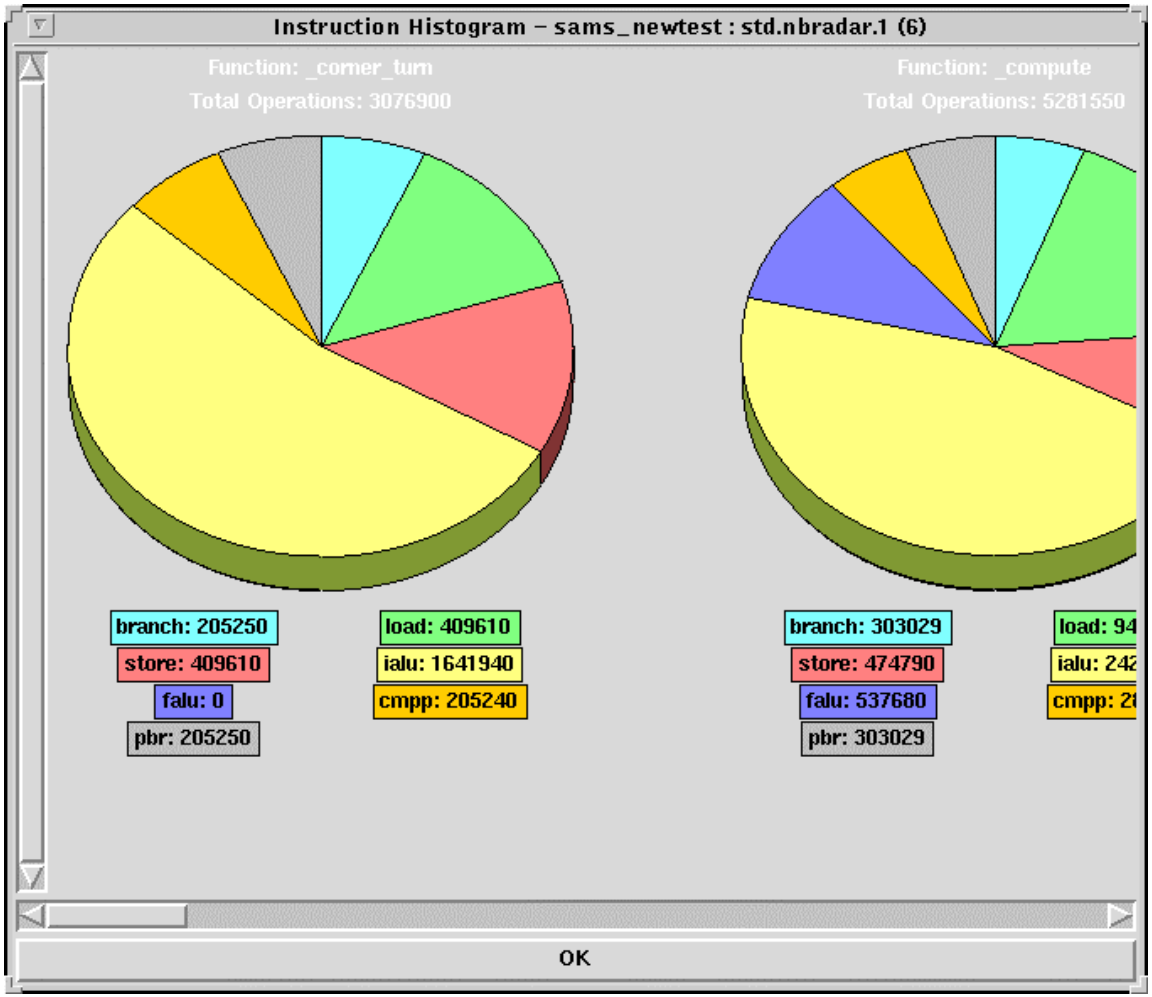
# Statistics

- List of items generated by the Trace-Driven Profiler
  - IPC (number of HPL-PD operations / clock cycle).
  - Memory address usage frequencies.
  - Control block visit frequencies.
  - Resource utilization.
    - Register Usage frequencies.
    - Functional Unit utilization.
    - Memory(Stack / Heap) utilization.
  - Effectiveness of guarded predicates.
  - Register allocation overhead.

# Viewing execution statistics using the GUI



# Viewing execution statistics using the GUI



# HPL-PD and Native Code Interaction

- Perhaps the most interesting aspect of the simulator is the ability to combine
  - HPL-PD code (as generated by Trimaran) with
  - Native machine code (generated from C by a native code compiler such as GCC)

  within a single simulation.
- The native code may come from a C library or be compiled from user-supplied C code.
- Run-time execution statistics are generated for the HPL-PD code, while it is executing.
  - No statistics are generated for native code.

# HPL-PD and Native Code (cont)

Why would you want to mix code compiled for HPL-PD by Trimaran with ordinary compiled C code?

- To utilize C libraries, without having to recompile them with Trimaran every time the machine configuration changes.
  - Generally not interested in run-time statistics about printf, etc.
- In a large program, you might be interested in obtaining run-time statistics (branch frequencies, etc) about a small part of the program.
  - Most of the program can be compiled using GCC. Only the parts whose HPL-PD execution behavior is of interest need be compiled by Trimaran.
  - Simulated execution runs much slower than native code, but you don’t pay the simulation overhead on most of the program.

# Summary

- Trimaran provides detailed execution statistics
  - Viewed graphically
  - Fed back into the compiler
- The simulator is integrated seamlessly into the rest of the system.
  - Controlled via the GUI
- Simulation overhead is paid only on those portions of the program that are being instrumented.