Incorporating Profile Information During Compilation

Profiling in the simulator

- The simulator has two major profiling components.
  - The “lightweight” profiler
    - generates summary information on the fly
  - The trace-driven profiler
    - reads execution trace to generate profile information
- Right now, the lightweight profiler generates profile information to be used for profile-driven compilation
  - In the system release, the trace-driven profiler’s output should be available for profiling.

The lightweight Profiler

- The control flow behavior of the program is being computed while the program is executing on the simulator.
  - The number of visits to each basic block or hyperblock is recorded.

Visit Counts

- There is a tool that feeds the visit counts back to the program graph in its IR form.
  - – Visit counts are used to update the weight field of each compound region and the various edges.
  - – Generally, we run an unoptimized program through the simulator
    - i.e. with all the Elcor optimizations turned off

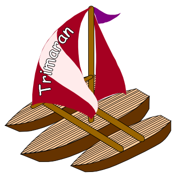
    to generate the profile information, and then optimize it using these weights.

The Weight Fields

- The weight fields are used by the region formation module.
  - the module forms larger control blocks, such as superblocks and hyperblocks, based on execution frequency
- We would also like to run unallocated and unscheduled code through the simulator
  - work in progress, requires some modification to the current simulator.

Lightweight Program Statistics

- Visit counts can also be used to collect dynamic execution statistics
    - for inspection by user
  - Cycle count, IPC, resource utilization, and dynamic operation histogram.
  - The information is derived by multiplying this frequency count by the static information of the control flow graph.
    - Thus it yields approximate results for if-converted (I.e. predicated) code.



The Trace-Driven Profiler

- This will be supported in subsequent releases
  - Will provide a larger range of information useful to the compiler.
- Control Flow Information
  - More precise control flow information is generated than from the lightweight profiler
    - e.g. predicated code execution

The Trace-Driven Profiler (cont)

- Memory & Cache behavior
  - Instruction & Data Cache
  - Memory addresses are output whenever the simulator executes a load or a store operation.
  - This information, coupled with the register value information, can be used for memory disambiguation in the compiler.

Performance

- The lightweight profiler was indeed confirmed to be lightweight, since it introduces a slow down factor of only 1.15x to 1.2x of the un-traced simulated code.
- On the other hand, the current implementation of tracer causes a very significant slowdown.
  - benchmarking is underway.