Instruction Tracing and dynamic codegen analysis to identify unique llvm performance issues.
Biplob Mishra

INTRODUCTION

Performance analysis of the machine code generated by a compiler can be carried out in different ways and can also be based on application in question. Common methods use some form of profiling on a running program which generally provides the statistical information about certain data and events. While this method does give important insights to a performance problem, some of the issues are more clearly understood when the compiled applications is actually run and the dynamic instructions of hot code execution paths are traced and analyzed in a small execution window. Trace records contain instructions and data, memory addresses and other information which provide complete visibility into the workings of an application.

TRACE OVERVIEW

- Trace is the dynamic sequence of executed instructions as it occurred when the program was executed.
- Trace provides complete history of what happens in a processor when the an application runs.
- Multiple events as occurred on instruction/code block execution can also be superimposed on the dynamic Execution sequence.
- On-right we have “a perf record” Profile of the same which contains cycle-event imposed on a static profile.
- The static profile though used does not give complete detail of running program.

WHERE CAN TRACES PLAY A ROLE

- Branch analysis
- Stack analysis
- Loop analysis/code placements
- Memory layout analysis

BEYOND

Enable cycle and other event analysis at lower code levels, particularly the hot loops or code sections within a function. So the above quicksort example cycle impact can be evaluated per function call level. The traces can further be used to evaluate core pipeline visualizations and determine hardware issues which cannot be explained by just disassembly analysis.