Exploration of Compiler Optimizations

Sumanth M Raviteja G

April 15, 2018

Abstract

In this document, we report about the tools and languages used in our term paper which is ”Exploration of Compiler Optimizations” in detailed view. We even try to state the concepts that are helpful in realizing the problem statement.

1 Tools and Languages

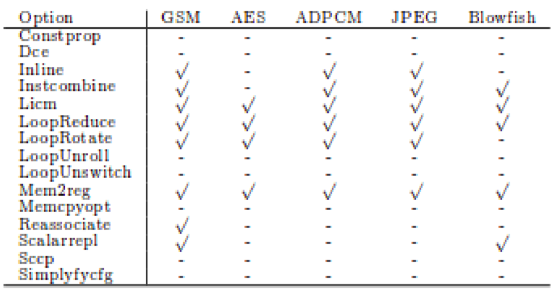
1.1 Design Space Exploration

This is the method we declared in our term paper to select the optimizations of compiler. This methods mainly explore the parameters of compiler options so that it explore the design space automatically and can even analyze the architecture of compiler.

We have been visualized by a tool called Multi-Objective System Tuner(MOST), a simulator namely LLVM and also a wrapper. They can explore, analyze and also optimize the compiler.

1.1.1 Kruskal-Wallis Test

In this test, we need to consider many compiler options like ConstProp, LoopReduce, LoopUnroll, LoopUnswitch etc..
We have analyzed the table which represents the estimation of utilizing specific



Option	GSM	AES	ADPCM	JPEG	Blowfish
Constprop	-	-	-	-	-
Dce	-	-	-	-	-
Inline	✓	-	✓	✓	-
Instcombine	✓	-	✓	✓	✓
Licm	✓	✓	✓	✓	✓
LoopReduce	✓	✓	✓	✓	✓
LoopRotate	✓	✓	✓	✓	-
LoopUnroll	-	-	-	-	-
LoopUnswitch	-	-	-	-	-
Mem2reg	✓	✓	✓	✓	✓
Memcpyopt	-	-	-	-	-
Reassociate	✓	-	-	-	-
Scalarrepl	✓	-	-	-	✓
Sccp	-	-	-	-	-
Simplyfycfg	-	-	-	-	-

Table 1: Kruskal-Wallis Test on Performance

Figure 1: Kruskal Wallis Test Table performed on Different Architectures

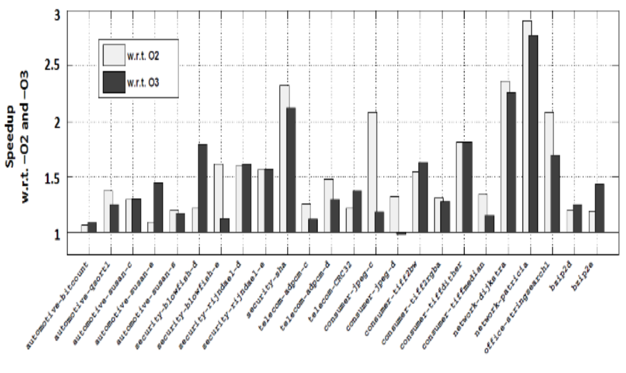


Figure 2: Improvment in performance of BN w.r.t -O2 and -O3

compiler options on selected applications with the Kruskal-Wallis test. We already observed that not all compiler options which have been utilized, plays a part in regulating the performance parameters on selected application.

2 Machine Learning Models

2.1 Bayesian Networks

In regards with DSE approach for finding the best compiler optimizations to tackle the problem of selecting the best compiler passes which leads to machine learning, COBAYN: Compiler autotuning framework using Bayesian Networks. An autotuning methodology based on machine learning to speed up the performance of application and to reduce the cost of the compiler optimization phases. We have gone through some benchmark suites namely Polybench, Cbench.

Finally, the proposed method demonstrates 4 times and 3 times speedup, respectively on cBench and Polybench, in terms of exploration efficiency

We have been demonstrated an application performance speedup of up to 4.6 times on Polybench and 3.1 times on Cbench with respect to standard optimization levels like -O2 and -O3.
In figure2 , one can find the graph used for evaluating performance.

3 Conclusions

We conclude that compiler optimizations mainly depend on compiler options or parameters. If the selected passes and order has given the best computation time and low cost, then that is considered as Compiler Optimization with efficient accuracy. In future, One need to work mainly on time and cost reduction.