AMULET: Adaptive Matrix-Multiplication-Like Tasks

COLUMBIA UNIVERSITY DATA SCIENCE INSTITUTE

Junyoung Kim[1], Kenneth Ross[1], Eric Sedlar[2], Lukas Stadler[2]
Department of Computer Science, Columbia University[1]
Oracle Labs[2]

Introduction

Variations of matrix multiplication are commonly used in data science and ML

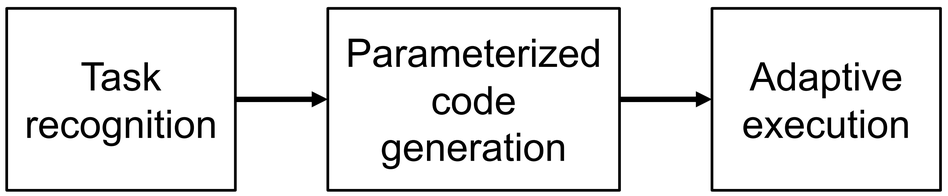
```
for(i = 0; i < M; i++)
 for(j = 0; j < N; j++)
  for(k = 0; k < K; k++)
   R[i][j] += A[i][k]*B[k][j]+
     (A[i][k]*B[k][j]>thres[j])*(A[i][k]*B[k][j]-thres[j]);
```

Figure 1. Matrix-multiplication-like task for ML that emphasizes high single products

However, performing such tasks is difficult as
- Matrix libraries only support a limited class of manually tuned computations
- Current compilers generate inefficient code for the task written as a nested loop

Amulet is a compiler framework that **generates fast code for matrix-multiplication-like tasks** tailored to its execution environment by using both compiler and database-style optimization techniques.

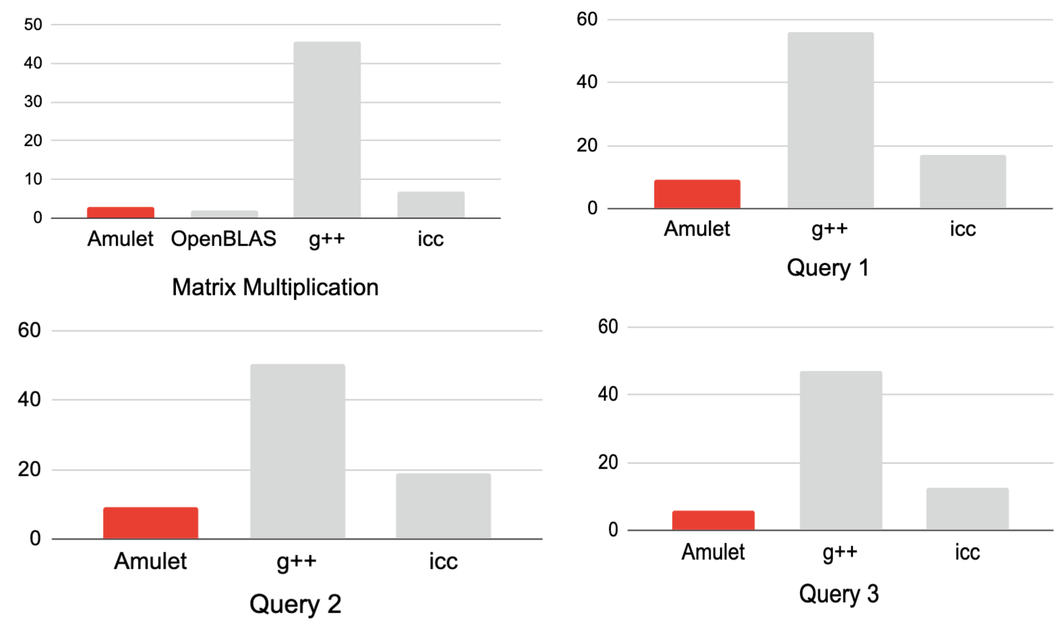
Process Overview



1) **Task recogition**
   - Amulet identifies a matrix-multiplication-like task using a set of rules.
2) **Parameterized code generation**
   - Using the method in [1], the task is transformed into a loop that partitions the task so that the cost of moving data is amortized for each level of the memory hierarchy.
   - Partition sizes are not hard-coded and are parameters Amulet can set before execution.
3) **Adaptive Execution**
   - Performant parameters for partitioning are found by testing various parameter values at runtime on a small subset of the input data.
   - The remaining data is executed using the found parameters.

Results

Settings
- Execution time (in seconds) is measured for matrix multiplication and other matrix-multiplication-like tasks (called Query 1, Query 2, Query 3)
- Baselines: OpenBLAS (manually tuned library for matrix multiplication), g++, icc compiler



Amulet acheives 1.8~2x speedup compared to the fastest compiler baseline (icc) and has decent matrix multiplication performance (45% slower) compared to libraries that are hand-written for different hardware.

Conclusions

Amulet generates code for various matrix-multiplication-like tasks that is 1.8~2x faster than code generated by existing compilers. We expect that Amulet will improve productivity for common data analysis tasks and facilitate research in the ML community by allowing scientists to write simple code that is also very efficient.

Acknowledgments

This work was supported by the National Science Foundation under grant IIS-2008295 and by a gift from Oracle.

References

[1] Kazushige Goto and Robert A. van de Geijn. 2008. Anatomy of High-PerformanceMatrix Multiplication.ACM Trans. Math. Softw.34, 3, Article 12 (may 2008),25 pages. https://doi.org/10.1145/1356052.1356053