Visualizing Data Locality

Seminar Thesis

Til Mohr

Chair for High Performance Computing, IT Center,
RWTH Aachen, Seffenter Weg 23,
52074 Aachen, Germany
Supervisor: Isa Thäringen, M.Sc.

This is the abstract. It is a short summary of the thesis contents (100 to 150 words).

Keywords: data locality, data movement, data access, optimization, visualization

1 Introduction

- Increasing Processor-Memory Speed Gap (2.1) and increasing requirements (2.2) result in a large increase in the affect of data movement costs and their resulting bottlenecks. While breakthroughs in hardware research can improve this issue, software engineers can also try to mitigate these issues by improving data locality (2.4) through software.
- Increasing complexity of programs makes it difficult to create a mental model of the data movement of a program. This makes it challenging for experts and impossible for domain researchers to optimize such a program.

2 Background

2.1 Processor-Memory Speed Gap

Present the processor-memory speed gap. Explanation, why it is a problem will

2.2 Computation and Memory Requirements

Present that requirements are increasing rapidly based on examples like ChatGPT, etc. -> There is always a need for more memory and more computation power

2.3 Data Transfer Costs and Bottlenecks

Explain what common data transfer costs are (move from memory to caches and in between caches) and briefly explain bottlenecks (e.g. Cache misses, memory bandwidth, etc.)

2.4 Data Locality

Explain/Define the term data locality

3 Methods

The goal is to make it easily accessible and possible for everyone (experts and domain researchers) to understand a programs' data movements and fix issues. To do this, data has to be gathered automatically (3.1,3.2,3.3) and then visualized in an understandable manner (3.4).

In this section, we will discuss the different approaches to gathering data for visualizing data movements in a program.

3.1 Dynamic Analysis

Run program and gather data while it is running, using Hardware Counters, Profiling, or Tracing.

Advantages:

- No need for parameterization → already is compiled for specific hardware
- Can be used in combination with actual data → even more accurate information

Disadvantages:

- Running a whole program is expensive (time and cost)

- Difficult to isolate and analyze specific parts of the program
- Very coarse time granularity → can not measure very short time intervals (hardware counters are not precise enough in aspects of being updated / read)

3.2 Static Analysis

Analyze the program statically using a compiler

Advantages:

- Can be used to analyze specific parts of the program
- Fast and cheap → No need to run program

Disadvantages:

- Needs to be parameterized for specific hardware (and often not accurate enough → might miss some details)
- Can not be used in combination with actual data

3.3 Simulation

Simulate the program on a simulator

Advantages: **TODO**

- Can be used to analyze specific parts of the program
- In between Static and Dynamic Analysis in terms of precision and speed and cost

Disadvantages: **TODO**

3.4 Visualization Techniques

Very brief overview of different visualizations (Colored Graphs, Heatmaps, etc.)

4 Selected Works

Take ~3 papers, briefly present how they work (which data gathering technique and what visualization), and what results they have shown.

4.1 MemAxes: Visualization and Analytics for Characterizing Complex Memory Performance Behaviors

4.2 Abstract Visualization of Runtime Memory Behavior

4.3 Boosting Performance Optimization with Interactive Data Movement Visualization

5 Conclusions

Conclusion

Future Work:

- Data Gathering and Visualizations can always be improved
- But we can use the gathered data to automatically optimize programs (reference to paper) → Even less work for the programmer (who might be just a domain researcher)
- Deep Learning is also being experimented with to automatically optimize programs at compile time (reference to paper)