The Cache Visualisation Tool

Eric van der Deijl Gerco Kanbier

Januari 1995

Contents

Preface

This master-thesis is written after we both have passed the seminar about code generation, where we read a lot of papers you will find in the bibliography. This seminar was combined with a project-study and lead us finally to a poster-presentation- at SuperComputing 1993 (Portland, Oregon, USA), where we find a lot of interest in our project. After this unique experience, we got the permission to improve the CVT and graduate on this project. After a year of programming, we still feel there can be many things improved, but this masters- thesis must be sufficient for further research done by other scientists.

We'll contribute this master-thesis to:
Elana D. Granston, who taught us the basic knowledge of caches in her seminar and project-study.
Olivier Temam, who designed the original plan for the CVT and is the brain behind all the cache design decisions.
Harry Wijshoff, who supervised all our courses and encouraged our enthusiasm in this project.

```
A picture shows me at a glance what it
takes dozens of pages of a book to
expound.

     Ivan S. Turgenev,
     Fathers and Sons, 1862
```

```
Scientific data are often complex,
and relationships among the
ingredients of an experiment can
be difficult to visualize.
Graphics provide a superb
tool for presenting scientific
information in a way that can
easily be grasped.

     F.S. Hill Jr., 1990
```

Chapter 1

Introduction

This tool is a cache simulator especially developed in order to gain insight into unpredictable cache phenomena which cause a trementous performance slow down on high performance super- computers. Other previous simulators could only unveil bad performance by indicaters like performance, miss- and hit- ratio. This simulator can not only show global figures about the performance of a program, but also visualize the bottle- neck and thus deal with the roots of this problem. The scien- tist is now able to deal with complex reference patterns which are hard to understand without visualization. With this tool, research can be done to all kind of programs on all kind of cache hierarchies.

The CVT supports two kind of software; traces and programs. Traces are made by the user; a suspicious code can be translated into a trace-file where program counter, base address and a read or a write are stored. Simulation gives an overall view of performance of this particular code. Bottle-necks are visualized by the statistics, where program counters with a high miss-ratio indicate a bad performance. Often, when you look in the original program of this trace, this program counter is often used in nested DO-loops. These structures are highly sensitive to interferences and therefor need a closer look. The user can translate a suspicious nested DO-loop from the orignal program to the CVT-program format. These programs compiled by the CVT contain only nested DO-loops. These loops can do a lot of iterations and some data might be used multip- le times. This opportunity of reuse must be exploited by the cache in order the improve the performance. Though, instead of reuse the data, it can also be bumped out of cache before it is reused. Now we're forced to get the data from memory instead of cache, which is a high price we have to pay because we don't exploit the cache which is developed to improve the performance. This miss-penalty is high because of the enormous development in cpu-speed and relatively slow memory. Misses caused by cross-, self- and capacity-interferences must there- for be avoided!

Numerical codes are typical examples where nested-loops and arrays are often used and thus can severely suffer from cross- and self-intereferences. Because of these phenomena, the potential capacity of some supercomputers lacks with the final performance, which is crucial for some programs. This tool can be used to learn about the basic cache behaviour as well for scientific research to unpredictable cache phenomena in all kind of different hard- and software environments. Next to the software support, the CVT supports different hardware environments. New developments can be tested on this tool. The only drawback in this tool is that the visualization of only one level in a hierarchy is possible at a time. The user will need to slightly change the simulator and do the simulation for another level in this hierarchy. But in fact the user can simulate any architecture.

This CVT is a complete tool and can be used for developping new soft- and/or hardware solutions by visualizing the cache behaviour cristal clear and makes the cache behaviour more predictable and understandable. This is an improvement to previous simulators because only global figures implied a worse performance, where there was no understanding about the cause of the performance slowdown. Now there is a possiblity to see the problems we'll deal with and solution can be thought of (which can also be tested, of course)

The next section is a theoretical chapter

about caches, where locality is discussed as well as the problems arising in cache followed by cache policies and set-associativity caches. Chap- ter three is written as a user-manual. Every possibility in the CVT is thoroughly discussed and an additional picture will clearify the text. Chapter four discusses the known software techniques to improve the performance and shows the user how to use the tool in order to detect cache phenomena. Chapter five will test a few known hardware optimazations and compare several hardware hierarchies. Finally chapter six will give our conclusions about this subject.