

The Cache Visualisation Tool

Eric van der Deijl Gerco Kanbier

Januari 1995

Contents

1	Introduction	5
2	Cache Theory	7
2.1	Introduction to caches	7
2.2	Set Associativity	8
2.3	Cache line identification	8
2.4	Replacement policies	8
2.5	Write Policies	9
2.6	Problems with caches	10
3	CVT Description	11
3.1	The global structure	11
3.2	The Screen	12
3.2.1	The Cache Area	12
3.2.2	The Program Area	12
3.2.3	The Statistics Area	13
3.3	Controlling the CVT	15
3.3.1	Button "> " <i>One Step</i>	15
3.3.2	Button ">" <i>Run</i>	16
3.3.3	Button ">>" <i>Fast Forward</i>	16
3.3.4	Button "<<" <i>Rewind</i>	16
3.3.5	Button " " <i>Pause</i>	17
3.3.6	Button "□" <i>Abort</i>	17
3.3.7	Speed	17
3.4	<i>Menu-Option</i> Tool	18
3.4.1	<i>Sub-Option</i> About Tool	18
3.4.2	<i>Sub-Option</i> Set Fast Forward	18
3.4.3	<i>Sub-Option</i> Set Rewind	18
3.4.4	<i>Sub-Option</i> Static Parameters	19
3.4.5	<i>Sub-Option</i> Quit	20
3.5	<i>Menu-Option</i> Program	20
3.5.1	<i>Sub-Option</i> Load	20
3.5.2	<i>Sub-Option</i> Colors	20
3.6	<i>Menu-Option</i> Trace	23
3.6.1	<i>Sub-Option</i> Load	23
3.6.2	<i>Sub-Option</i> Colors	24
3.7	<i>Menu-Option</i> Breakpoints	25
3.7.1	<i>Sub-Option</i> Add Cache Breakpoint	25
3.7.2	<i>Sub-Option</i> Timer Breakpoint	26
3.7.3	<i>Sub-Option</i> Add Loopvalue Breakpoint	26
3.7.4	<i>Sub-Option</i> Add Statement Breakpoint	27
3.7.5	<i>Sub-Option</i> Add PC Breakpoint	27

3.7.6	<i>Sub-Option</i> Show List of Breakpoints	28
3.8	<i>Menu-Option</i> Parameters	29
3.8.1	Architecture	29
3.8.2	Write policy	30
3.8.3	Allocate policy	30
3.8.4	Replacement policy	30
3.9	<i>Menu-Option</i> Statistics	31
3.9.1	Cache Statistics	31
3.9.2	Array statistics	31
3.9.3	Trace miss, reuse, references and top-16	32
3.10	<i>Menu-Option</i> Others	32
3.10.1	Refresh screen	32
3.10.2	Grid Mode	32
3.10.3	Swap Page	32
3.10.4	<i>Sub-Option</i> Messages	33
3.10.5	<i>Sub-Option</i> Extra Info	34
3.11	Making the Input	35
3.11.1	Program	35
3.11.2	Traces	37
3.12	Further Tuning	38
3.12.1	The Simulator	38
3.12.2	Using and changing CVT's data structures and variables	40
4	Using the CVT for Software Optimizations	42
4.1	Introduction	42
4.2	Cache Interferences	43
4.3	Blocking	46
4.4	Nonsingular Loop Transformations	48
4.4.1	Some theory	48
4.4.2	Unimodular transformations of double loops	49
4.4.3	Optimizing data locality through unimodular loop transformations	50
4.4.4	Nonsingular loop transformations	52
4.5	Software Prefetching	53
4.6	Sparse codes	55
5	Using the CVT for Hardware Optimizations	58
5.1	Introduction	58
5.2	Terminology	58
5.3	Memory Hierarchy Evaluation	58
5.3.1	Cache Parameters	59
5.4	Caches	59
5.5	Replacement policy	60
5.6	Write policy	61
5.7	Opportunities of the CVT	61
5.7.1	How can we do research?	61
5.7.2	Victim cache	62
5.8	Test results	64
5.8.1	Cache size and Set associativity	64
5.8.2	Cacheline size and Set Associativity	66
6	Conclusions	68

A	CVT Programs	70
A.1	CVT Code for FLO52	70
A.2	CVT code for Blocked Matrix x Matrix	70
A.3	CVT code for SOR	71
B	Trace Makers	72
B.1	Making a trace for software prefetched matrix matrix multiply	72
B.2	Making a trace for Sparse Matrix Vector multiply	72

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 tremendous performance slow down on high performance super-computers. Other previous simulators could only unveil bad performance by indicators 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 scientist 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 original 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 multiple times. This opportunity of reuse must be exploited by the cache in order to 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 mem-

ory 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 therefore be avoided!

Numerical codes are typical examples where nested-loops and arrays are often used and thus can severely suffer from cross- and self-interferences. 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 developing new soft- and/or hardware solutions by visualizing the cache behaviour crystal 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 possibility 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. Chapter three is written as a user-manual. Every possibility in the CVT is thoroughly discussed and an additional picture will clarify 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 optimizations and compare several hardware hierarchies. Finally chapter six will give our conclusions about this subject.