Available online at www.sciencedirect.com
SCIENCE DIRECT®

Future Generation Computer Systems 21 (2005) 1298–1311

FGCS
FUTURE GENERATION COMPUTER SYSTEMS
www.elsevier.com/locate/fgcs

Monitoring cache behavior on parallel SMP architectures and related programming tools

Thomas Brandes[a,*], Helmut Schwamborn[a], Michael Gerndt[b], Jürgen Jeitner[b], Edmond Kereku[b], Martin Schulz[b], Holger Brunst[c], Wolfgang Nagel[c], Reinhard Neumann[c], Ralph Müller-Pfefferkorn[c], Bernd Trenkler[c], Wolfgang Karl[d], Jie Tao[d], Hans-Christian Hoppe[e]

[a] *Institut für Algorithmen und Wissenschaftliches Rechnen (SCAI), Fraunhofer Gesellschaft (FhG), Schloss Birlinghoven, D-53754 St. Augustin, Germany*
[b] *Lehrstuhl für Rechnertechnik und Rechnerorganisation (LRR), Technische Universität München, Boltzmannstr. 3, D-85748 Garching, Germany*
[c] *Zentrum für Hochleistungsrechnen (ZHR), Technische Universität Dresden, Zellescher Weg 12, D-01062 Dresden, Germany*
[d] *Institut für Rechnerentwurf und Fehlertoleranz, Universität Karlsruhe (TH), Kaiserstraße 12, D-76128 Karlsruhe, Germany*
[e] *Intel GmbH, Software and Solutions Group, Hermühlheimer Str. 8a, D-50321 Brühl, Germany*

Available online 30 December 2004

Abstract

This paper describes the ideas and developments of the project EP-CACHE. Within this project new methods and tools are developed to improve the analysis and the optimization of programs for cache architectures, especially for SMP clusters. The tool set comprises the semi-automatic instrumentation of user programs, the monitoring of the cache behavior, the visualization of the measured data, and optimization techniques for improving the user program for better cache usage.

As current hardware performance counters do not give sufficient user relevant information, new hardware monitors are designed that provide more detailed information about the cache utilization related to the data structures and code blocks in the user program. The expense of the hardware and software realization will be assessed to minimize the risk of a real implementation of the investigated monitors. The usefulness of the hardware monitors is evaluated by a cache simulator.

Keywords: Hardware cache monitoring; Performance analysis; Cache optimizations; Parallel programming tools; SMP cluster

* Corresponding author.
E-mail addresses: brandes@scai.fhg.de (T. Brandes), gerndt@in.tum.de (M. Gerndt), nagel@zhr.tu-dresden.de (W. Nagel), karl@ira.uka.de (W. Karl), hans-christian.hoppe@intel.com (H.-C. Hoppe).

0167-739X/$ – see front matter © 2004 Published by Elsevier B.V.
doi:10.1016/j.future.2004.09.005

1. Introduction

The gap between CPU performance (increasing approximately by a factor of 1.5 per year) and memory performance (factor of 1.07 per year) has increased dramatically and might continue to increase during the next years. Caches placed between memory and processor provide the possibility to access data much faster by duplicating parts of main memory in smaller and faster memory. But the advantages of the cache are only given if temporal and spatial data locality is exploited in the user program. Re-using data in the cache provides temporal locality and can speed up a program by a factor of 6 to 10 on many computers.

Straight-forward written programs not taking the cache hierarchy into account achieve only a small fraction of the theoretical peak performance. Tuning the program for better cache utilization has become an expensive part of the software development cycle. Furthermore, the modular and extendible design of software conflicts directly with the locality needed for the exploitation of the cache.

Identification and understanding of bottlenecks in a program due to cache problems is one of the most critical issues. General information about cache misses is not very useful as they give the user only the information that something goes wrong but not where and why.

The project EP-CACHE [5] (funded by the German Federal Ministry of Education and Research, BMBF) is intended to overcome this problem. By exploiting hardware monitors and related monitor control techniques the user can gain more useful information about the cache behavior in his program. Related tools for monitor controlling, performance visualization and optimization provide new and advanced possibilities for the identification of memory-cache problems and their elimination. The tools address the analysis and the optimization of programs for cache architectures, especially for SMP clusters.

The rest of the paper is organized as follows. Section 2 describes our approach of hardware supported cache monitoring. The performance analysis outlined in Section 3 uses the VAMPIR tool for visualization of the monitored data. In Section 4 we outline our tools for optimizing the source program for better cache usage. We present first evaluation results in Section 5 and we give a summary in Section 6.

2. Monitoring cache behavior

Any technique implementing an efficient cache optimization strategy will depend on an accurate observation of the memory behavior in the target system. Only this will provide the necessary basis for the determination of existing bottlenecks.

2.1. *A new approach for hardware cache monitoring*

Cache monitoring is available in most microprocessor architectures, but the current approaches are very restricted in their capabilities. They only allow the monitoring of a few (typically 4–8) global event types and do not provide any means to associate the observed behavior with the related addresses causing the events. This severely restricts their applicability and hence also the potential optimization techniques that can be derived from their observation.

Within this project a new cache monitoring architecture has been designed, which overcomes these limitations and delivers a detailed overview of the complete memory access behavior of an application in relation to its virtual address space. The core of this approach is an associative counter array capable of recording all events snooped from an arbitrary bus together with their address information. This work is based on previous research in the area of hardware DSM monitoring [9], which has been followed all the way to a corresponding hardware prototype, showing the feasibility of the discussed approach.

These counters can be used to record specific events for a specified data and/or instruction address range. This enables, for example, detailed measurements of the number of cache misses for a specific data structure in a specific code region. The event type and the address ranges can be configured.

Configuration of the counters can also be controlled by the hardware monitor itself. The address space to be monitored is partitioned into blocks according to a given granularity. If an access to an address currently not covered by a counter leads to a miss, a free counter is identified or a counter is freed by purging its content to a monitor buffer. This mode can be used to generate access histograms with a given granularity for the specified address range.