Performance Evaluation of Adaptive Caching Schemes

Jie Tao and Wolfgang Karl

Institut für Technische Informatik
Universität Karlsruhe (TH)
76128 Karlsruhe, Germany
E-mail: {tao,karl}@ira.uka.de

Abstract:
As the gap between processor and main memory speed grows, cache locality becomes increasingly important. However, data caches are often not used efficiently resulting in large number of cache misses. This scenario is more critical on current and future chip-multiprocessors because these architectures often show a higher cache miss ratio than uni-processor systems.

In order to improve cache efficiency, various adaptive caching schemes, which make the conventional cache techniques more flexible and adaptive to data access behavior, have been proposed. These schemes, on the other hand, also introduce overheads. A question arises: do these techniques really improve performance?

To answer this question, we have studied various adaptive caching mechanisms, like adaptive prefetching, effective cache line replacement, and reconfigurable cache organization. This study is based on a full cache simulator that allows to configure the caches with a variety of parameters and the modeling of adaptive caching techniques. Experimental results based on standard benchmarks show that only when these techniques are finely tuned, applications benefit.

1 Introduction

Memory access latencies continue to grow, making the disparity between processor and memory speed increasingly widening. Cache performance becomes hence especially important for fully exploring the computational properties of modern single and multiprocessor systems. However, data caches are often not used efficiently, resulting in large number of cache misses. This problem is even more severe with parallel programming on shared memory systems. According to research reports, the cache miss ratio can be up to four folds on chip-multiprocessor systems than on uni-processors.

A reason for cache misses lies in the cache design of general purpose, which can not match the various requirements of different application domains. In this case, adaptive techniques could help to improve cache efficiency: for example, a reconfigurable cache that can be configured to different structures, an adaptive prefetching approach that can dynamically decide the prefetching size, and an efficient compression scheme that only compresses the
cache lines when a high miss ratio is observed. However, all these mechanisms introduce overheads. Whether applications really benefit becomes an open question.

Several research work has targeted on this question, but is restricted to specific techniques and applications. The work presented in this paper aims at carrying out a comprehensive, statistical study and giving a general conclusion. It covers a set of adaptive policies, including those proposed and novel ones. It uses a range of possible system and application configurations, in order to detect how well an adaptive scheme performs in different cases and whether performance gains can be achieved. The goal is to give researchers a hint about how an adaptive technique will perform on the target architecture.

This study is based on an existing cache simulator originally developed for evaluating the memory system on shared memory architectures. We have extended this simulator for implementing adaptive caching mechanisms, such as adaptive prefetching and remapping-based replacement. We evaluated these mechanisms with a variety of applications and a range of possible configuration parameters. Experimental results show that in most cases applications benefit from the adaptivity, if the developed adaptive strategy is well tuned.

The remainder of this paper is organized as follows. Section 2 briefly describes several proposed and novel adaptive caching techniques with some related work included. This is followed by an overview of the cache simulator and the implementation details of the studied techniques in Section 3. In Section 4, experimental results are presented and discussed. The paper concludes with a short summary and some future directions in Section 5.

2 Adaptive Caching Techniques and Related Work

Cache is actually a fast buffer between processor and the main memory. Traditionally, caches deploy a set of common techniques to deal with memory issues and to process data accesses. However, in order to improve the efficiency of caches, researchers have proposed novel policies that make the conventional techniques adaptive.

Prefetching is a common used technique for achieving spatial locality. However, often the prefetched data blocks can not be fully used causing a loss of memory bandwidth. Johnson et al. [JMH97] present a hardware scheme which dynamically adjust the amount of data fetched at a cache miss. This approach uses a Spatial Locality Detection Table to track the accesses to multiple adjacent cache blocks, facilitating detection of spatial locality across those blocks while they are cached. Simulation results show speedups of up to 26% for 200-cycle memory latencies. The improvement is due to the reduction of cache misses by utilizing small fetch sizes when spatial locality is absent, but large fetch sizes when spatial locality exists. Oliver et al. [OT00] also explore dynamic and adaptive cache prefetching. They use a rule-based decision system to dynamically choose a prefetch policy from three candidates: SSB (sequential stream buffer), SWA (sequential word access), and SLA (sequential line access). The decision is based on an analysis of process execution characteristics. Experiments in terms of instruction caches show that this new policy performs better than SSB.
Compression is a technique for improving cache performance by increasing effective cache capacity and eliminating misses. However, decompressing cache lines also increases cache access latency, potentially degrading performance. Alamelden et al. [AW04] developed an adaptive cache compression scheme that can dynamically adapt to the costs and benefits of cache compression. This scheme uses the LRU replacement state of caches to determine whether compression eliminates a miss or incurs an unnecessary decompression overhead. Based on this outcome, the adaptive policy updates a global saturating counter which predicts whether to allocate a cache line in compressed or uncompressed form. Evaluations using full-system simulation show that the adaptive policy achieves comparable benefits from compression, while never degrading performance by more than 0.4%.

Reconfigurable cache is another adaptive mechanism. Traditionally, a cache design uses fixed cache size and organization, e.g. associativity, making caches only efficient for specific application domains. Reconfigurable cache, on the other hand, is a grand challenge for matching the computational requirements of various applications. Smart Memories [MPJ+00], for example, is such a project with the goal of designing reconfigurable architecture to match different applications with different access patterns. In terms of caches this design deploys a flexible interconnection that allows to change the size and characteristics of caches and to implement other memory structures. Ranganathan et al. [RAJ00] also propose a configurable cache design, but focus only on cache capacity. This design enables the cache SRAM array to be dynamically divided into multiple partitions that can be assigned to different processor activities, e.g. used as hardware look-up tables for techniques such as instruction reuse and hardware prefetching, or as storage area for prefetched information, or as compiler-controlled memory. These activities can benefit applications that would otherwise not use the storage allocated to large conventional caches. Simulation using an analytical model and media processing benchmarks show improvements in IPC ranging from 1.04X to 1.20X across the tested applications.

Overall, different adaptive approaches have been proposed and even evaluated in the last years in order to improve cache utility. Among these techniques, prefetching and reconfigurable cache are specially important, since they directly address two of the three kind of cache misses: cold miss and capacity miss. According to our simulation results, however, usually more than 70% cache misses are conflict misses caused by cache line replacement. It is necessary to investigate adaptive approaches for this issue.

Therefore, this work focuses on the following cache strategies:

- **Prefetching**: It has been investigated in [JMH97], but the study was restricted to specific applications. We use standard benchmarks in order to give a more general conclusion.

- **Configuration**: The Smart Memories [MPJ+00] has proposed a hardware design with reconfigurable cache size and associativity, but no research results about the impact of this design on changing access pattern of applications have been reported. We address this issue in order to exhibit how a configurable cache with changing cache organization performs over different applications with different access pattern and working set sizes.
• **Replacement:** Upon an earlier study we found that cache line replacement is the most critical reason for cache misses. This is caused by the fact that frequently accessed data often has to be replaced due to interferences, while other less used blocks resident in the cache. We propose an adaptive replacement policy that remaps the missing data to a location holding a less utilized block, in case that the block, which is actually to be replaced, is frequently used. In this way the required data can be held in the cache and the rarely needed blocks are removed. An implementation of this policy will be described in the following section.

### 3 Simulation Platform

We establish these adaptive policies in an existing cache simulator which has been originally developed for studying the interconnection traffic on NUMA (Non-Uniform Memory Access) systems and for evaluating memory systems with a focus on data locality.

This cache simulator [TSK03] models the complete cache hierarchy with arbitrary levels. Each cache can be organized as either write-through or write-back, and configured to a wide variety of characteristics: directly-mapped, set-associative, or full-associative. Cache size, cache line size, and access latency etc. can also be specified via command line parameters. In addition, the cache simulator models a set of cache coherence protocols. This includes the hardware-based MESI protocol and several relaxed consistency schemes often used in shared memory systems with distributed memory structure. In the end of the simulation, the cache simulator reports statistics on cache behavior, e.g. cache hit ratio and number of cache misses. Additionally, its front-end, which builds the basic simulation infrastructure and generates memory accesses, provides elapsed simulation time and simulated processor cycles. This allows to examine the impact of various optimization schemes on the performance.

In order to study the novel and existing adaptive cache techniques described in the previous section, we have extended the cache simulator with corresponding implementation for adaptive prefetching and replacement. The property of reconfiguration has been provided by the cache simulator.

We implemented a flexible prefetching policy. The prefetching is performed at cache misses. By each cache miss a number of data blocks are loaded to the cache, together with the missing block. For adaptation, users can specify which and how many blocks should be prefetched. The prefetch delay is computed as the transfer time of memory blocks from some low level of the memory hierarchy to the related cache. This delay can be larger in real cases due to bus competitions; hence we provide a command line parameter for specifying different values.

For the adaptive replacement policy, we implemented two algorithms for remapping frequently used data in case of conflicts. The first algorithm first seeks the replacement candidate within the corresponding cache set based on the conventional LRU policy. Then the number of accesses to this set is compared with that to other sets in order to find the fewest used cache set. If a threshold is exceeded, e.g. the former is twofold as the latter, it is
<table>
<thead>
<tr>
<th>Description</th>
<th>Working set size</th>
<th>Benchmark suit</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFT</td>
<td>2**12 data points</td>
<td>SPLASH</td>
</tr>
<tr>
<td>LU</td>
<td>256 × 256 matrix</td>
<td>SPLASH</td>
</tr>
<tr>
<td>WATER</td>
<td>64 molecules</td>
<td>SPLASH</td>
</tr>
<tr>
<td>CG</td>
<td>100</td>
<td>NAS serial</td>
</tr>
<tr>
<td>SP</td>
<td>12×12×12</td>
<td>NAS serial</td>
</tr>
<tr>
<td>mgrid</td>
<td>32×32×32</td>
<td>SPEC 2000</td>
</tr>
<tr>
<td>swim</td>
<td>64×64</td>
<td>SPEC 2000</td>
</tr>
</tbody>
</table>

Tabelle 1: Applications and their working set size.

decided to keep the candidate in cache and the least used block within the fewest accessed cache set is replaced. For implementation of this algorithm, we define a set-description table to store needed information. This table would be implemented in hardware on real systems. For each set in the table, two fields are used: one records the number of references to it and the other gives the information whether a block mapped to this set is remapped. In case of remapping, where a cache block not found in the mapping set but in another one, the access time is doubled. In addition, the overheads for maintaining the set-description table are taken into account.

The second algorithm is an extension of the LRU policy. In case of interferences, the replacement candidate is searched in the whole cache, rather than within a single set as the conventional LRU does. For this, again a set-description table is maintained. Unlike the one used for the first algorithm, only remapping information is stored. Other needed information is available from the LRU implementation.

4 Experimental Results

Based on this simulation framework, we could study the proposed adaptive policies. For this study, we use a set of applications from standard benchmark suites, including SPLASH [WOT+95], NAS serial [ea94], and SPEC 2000 [Hen00]. A short description of these applications is shown in table 1.

4.1 Cache Configuration

We first examine the performance of various cache configurations. As it is a fact that cache hit ratio increases with cache size, we study only the impact of different cache set size and block size.

Figure 1 shows the variation of cache miss number with set size that has been chosen as 1, 2, 4, and 8 representing directly-mapped and traditional set associative caches. In principle, number of cache misses shall decrease as the set size increases, because more
mapping possibilities exist. As shown in the left diagram of Figure 1, most applications hold this feature. However, WATER presents different behavior, where 8-way cache shows more misses than 4-way cache. The access behavior of L2 cache is more surprising. From the right diagram of Figure 1, it can be observed that only FFT shows the traditional feature with reduced miss number for higher associativity. Other applications behave differently, where LU, SP, and swim scale up to 4-way cache, WATER, CG, and mgrid up to 2-way cache, and CG even with continuing increase.

![Graphs showing cache misses with various associativity](image)

**Abbildung 1:** Cache misses with various associativity (left: L1, right: L2).

Similar pattern can be seen with the changes of block size. As illustrated in Figure 2 which shows L1 misses with different block sizes, for all applications more cache misses are caused with a block size of 128 bytes. For FFT and LU, the best performance is achieved with a block size of 64 bytes, while for others a block size of 32 bytes outperforms. This different behavior is caused by the different access pattern of applications. Actually, for caches of same size and set size, larger block size indicates less blocks. In this case, only applications with higher spatial locality benefit from larger blocks. For applications with less spatial locality, usually data within the same block cannot be reused. On the other hand, while less blocks are available, more conflict exists, resulting in more cache misses.

Overall, both diagrams indicate that it depends on applications' access pattern, whether a chosen set or block size behaves well. This also means fine tuning with cache configuration is needed, since this kind of tuning can lead to a significant performance gain. For the FFT code in Figure 2, for example, only 62% cache misses are caused when using a cache block of 64 bytes rather than a block of 16 bytes.

### 4.2 Cache Prefetching

Prefetching is a strategy capable of reducing cold misses, or so-called first reference misses. However, prefetching also introduces additional access latency if the prefetched data
block can not be reused. In this subsection, we examine the performance of caches with prefetching enabled.

We compare cache misses in three cases: caches without prefetching (N-Prefetching), caches with traditional prefetching (the following block prefetched), and caches with adaptive prefetching (A-Prefetching) where prefetched data block and its size can be specified. These alternatives can be given using command line parameters of the simulator. The experimental results are depicted in Figure 3, with the left diagram showing cold misses and the right the total misses of the L1 cache.

Abbildung 3: L1 cold misses (left) and total misses with different prefetching policies.

As we use the default working set size for SPLASH applications, which is larger than the size of simulated L1 cache, it can be seen that FFT, LU, and WATER have the same
number of cold misses for caches without prefetching. Hence, we tested another two small applications, MATRIX (matrix multiplication) and SOR (Successive Over Relaxation). For adapting, some applications, in which data is accessed with a stride, e.g. MATRIX where the second matrix is used in columns, we prefetch the block in this stride. For others two following data blocks are prefetched.

Examine Figure 3, it can be seen that the number of cold misses decreases with prefetching and adaptive prefetching as expected. This means prefetching is needed for singly reducing cold misses. Observing the right diagram of Figure 3 which shows the total misses of L1 cache, however, different behavior can be seen. For FFT and LU, the number of total misses reduces with prefetching and adaptive prefetching. For WATER, MATRIX, and SOR, nevertheless, more cache misses are caused with prefetching or adaptive prefetching. This can be explained by the fact that using prefetching cache space is occupied by prefetched, but not or scarcely reused data, causing other reused data mapped on top of each other. Due to conflict, more misses are performed. This result indicates that prefetching is not commonly needed. Hence, an adaptive policy, which can decide whether to switch on or to regulate the prefetching, is essential for achieving efficient cache behavior.

### 4.3 Remapping

As described in the previous Section, we have implemented two algorithms for cache line remapping in case of interferences: the so-called “fewest reference” that uses the number of memory references as its decision base and extended LRU that applies the LRU in the whole cache. We compare these replacement policies with the conventional LRU scheme.

Abbildung 4: Total cache misses (left) and simulated execution time (right) with adaptive replacement.

Figure 4 depicts the experimental results. The left diagram shows the total number of cache misses with different replacement policies, while the right diagram shows the simulated
execution time in million CPU cycles. It can be observed that both adaptive schemes generally perform better than the traditional LRU (T-LRU), with “fewest reference” (FR) slightly outperforming “extended LRU (E-LRU)”. MATRIX and SOR, however, are exclusions. For MATRIX, E-LRU introduces the best performance with 16% less misses and 10.3% faster in execution than the traditional LRU approach. Nevertheless, for SOR both adaptive schemes show worse performance, especially the E-LRU scheme. This is probably caused by SOR’s working characteristics. SOR is used to iteratively solve partial differential equations. Its main working set is a large dense matrix. It begins by initializing the boundary values and then computes the values inside the matrix with given boundary conditions. The calculation terminates after a certain number of iterations. This means all elements in the matrix are needed, earlier or later, during each iteration. Hence, the E-LRU scheme, which invalidates the data earlier used, but to be reused in the next iteration, causes more frequent replacement and thereby more misses and overhead.

5 Conclusions

In this paper, we investigate the performance of different adaptive caching schemes, including novel ones with respect to cache line replacement and existing approaches with respect to prefetching and cache configuration. Experimental results depict that with fine tuning these adaptations can introduce significant performance gains.

This work will be followed by further study of other adaptive strategies, especially those additionally designed for multiprocessor systems like invalidation policies. In addition, more realistic applications will be investigated.

Literatur


GI-Edition Lecture Notes in Informatics


P-3 Ana M. Moreno, Reind P. van de Riet (Hrsg.): Applications of Natural Language to Information Systems, NLDB’2001.


P-5 Andy Schürr (Hg.): OMER - Object-Ori- ented Modeling of Embedded Real-Time Systems.


P-7 Andy Evans, Robert France, Ana Moreira, Bernhard Rumpe (Hrsg.): Practical UML-Based Rigorous Development Methods - Countering or Integrating the extremists, pUML’2001.


P-10 Mirjam Minor, Steffen Staab (Hrsg.): 1st German Workshop on Experience Management: Sharing Experiences about the Sharing Experience.


P-12 Martin Glinz, Günther Müller-Luschnat (Hrsg.): Modellierung 2002.

P-13 Jan von Knop, Peter Schirmbacher and Viljan Mahnič (Hrsg.): The Changing Universities – The Role of Technology.


P-15 Hans-Bernd Bludau, Andreas Koop (Hrsg.): Mobile Computing in Medicine.


P-21 Jörg Desel, Mathias Weske (Hrsg.): Pro- mise 2002: Prozessorientierte Methoden und Werkzeuge für die Entwicklung von Informationssystemen.


P-23 Thorsten Spitta, Jens Borchers, Harry M. Sneed (Hrsg.): Software Management 2002 - Fortschritt durch Beständigkeit

P-24 Rainer Eckstein, Robert Tolksdorf (Hrsg.): XMIDX 2003 – XML-Technologien für Middleware – Middleware für XML-Anwendungen

P-25 Key Poustchi, Klaus Turowski (Hrsg.): Mobile Commerce – Anwendungen und Perspektiven – 3. Workshop Mobile Commerce, Universität Augsburg, 04.02.2003

361
P-26 Gerhard Weikum, Harald Schöning, Erhard Rahm (Hrsg.): BTW 2003: Datenbanksysteme für Business, Technologie und Web

P-27 Michael Kroll, Hans-Gerd Lipinski, Kay Melzer (Hrsg.): Mobiles Computing in der Medizin

P-28 Ulrich Reimer, Andreas Aabecker, Steffen Staab, Gerd Stumme (Hrsg.): WM 2003: Professionelles Wissensmanagement - Erfahrungen und Visionen

P-29 Antje Düsterhöft, Bernhard Thalheim (Eds.): NLDB’2003: Natural Language Processing and Information Systems

P-30 Mikhail Godlevsky, Stephen Liddle, Heinrich C. Mayr (Eds.): Information Systems Technology and its Applications

P-31 Arslan Brömme, Christoph Busch (Eds.): BIOSIG 2003: Biometric and Electronic Signatures

P-32 Peter Hubwieser (Hrsg.): Informatische Fachkonzepte im Unterricht – INFOS 2003

P-33 Andreas Geyer-Schulz, Alfred Taudes (Hrsg.): Informationswirtschaft: Ein Sektor mit Zukunft

P-34 Klaus Dittrich, Wolfgang König, Andreas Oberweis, Kai Rannenberg, Wolfgang Wahler (Hrsg.): Informatik 2003 – Innovative Informatikanwendungen (Band 1)

P-35 Klaus Dittrich, Wolfgang König, Andreas Oberweis, Kai Rannenberg, Wolfgang Wahler (Hrsg.): Informatik 2003 – Innovative Informatikanwendungen (Band 2)

P-36 Rüdiger Grimm, Hubert B. Keller, Kai Rannenberg (Hrsg.): Informatik 2003 – Mit Sicherheit Informatik

P-37 Arndt Bode, Jörg Desel, Sabine Rathmayer, Martin Wessner (Hrsg.): DeLFI 2003: e-Learning Fachtagung Informatik

P-38 E.J. Sinz, M. Plaha, P. Neckel (Hrsg.): Modellierung betrieblicher Informationsysteme – MobIS 2003

P-39 Jens Nedon, Sandra Frings, Oliver Göbel (Hrsg.): IT-Incident Management & IT-Forensics – IMF 2003

P-40 Michael Rebstock (Hrsg.): Modellierung betrieblicher Informationssysteme – MobIS 2004

P-41 Uwe Brinkschulte, Jürgen Becker, Dietmar Fey, Karl-Erwin Großpietsch, Christian Hochberger, Erik Maehle, Thomas Runkler (Eds.): ARCS 2004 – Organic and Pervasive Computing

P-42 Key Poustchi, Klaus Turowski (Hrsg.): Mobile Economy – Transaktionen und Prozesse, Anwendungen und Dienste

P-43 Birgitta König-Ries, Michael Klein, Philipp Obreiter (Hrsg.): Persistence, Scalability, Transactions – Database Mechanisms for Mobile Applications

P-44 Jan von Knop, Wilhelm Haverkamp, Eike Jessen (Hrsg.): Security, E-Learning, E-Services

P-45 Bernhard Rumpe, Wolfgang Hesse (Hrsg.): Modellierung 2004

P-46 Ulrich Flegel, Michael Meier (Hrsg.): Detection of Intrusions of Malware & Vulnerability Assessment

P-47 Alexander Prosser, Robert Krimmer (Hrsg.): Electronic Voting in Europe – Technology, Law, Politics and Society

P-48 Anatoly Doroshenko, Terry Halpin, Stephen W. Liddle, Heinrich C. Mayr (Hrsg.): Information Systems Technology and its Applications

P-49 G. Schiefer, P. Wagner, M. Morgenstern, U. Rickert (Hrsg.): Integration und Datensicherheit – Anforderungen, Konflikte und Perspektiven

P-50 Peter Dadam, Manfred Reichert (Hrsg.): INFORMATIK 2004 – Informatik verbindet (Band 1) Beiträge der 34. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 20.-24. September 2004 in Ulm

P-51 Peter Dadam, Manfred Reichert (Hrsg.): INFORMATIK 2004 – Informatik verbindet (Band 2) Beiträge der 34. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 20.-24. September 2004 in Ulm

P-52 Gregor Engels, Silke Seehusen (Hrsg.): DeLFI 2004 – Tagungsband der 2. e-Learning Fachtagung Informatik

362
P-53 Robert Giegerich, Jens Stoye (Hrsg.): German Conference on Bioinformatics – GCB 2004

P-54 Jens Borchers, Ralf Kneuper (Hrsg.): Softwaremanagement 2004 – Outsourcing und Integration

P-55 Jan von Knop, Wilhelm Haverkamp, Eike Jessen (Hrsg.): E-Science und Grid Ad-hoc-Netze Medienintegration

P-56 Fernand Feltz, Andreas Oberweis, Benoit Ojacques (Hrsg.): EMISA 2004 - Informationssysteme im E-Business und E-Government

P-57 Klaus Turowski (Hrsg.): Architekturen, Komponenten, Anwendungen

P-58 Sami Beydada, Volker Gruhn, Johannes Mayer, Ralf Reussner, Franz Schweiggert (Hrsg.): Testing of Component-Based Systems and Software Quality

P-59 J. Felix Hampe, Franz Lehner, Key Poustchti, Kai Ranneberg, Klaus Turowski (Hrsg.): Mobile Business – Processes, Platforms, Payments

P-60 Steffen Friedrich (Hrsg.): Unterrichtskonzepte für informatische Bildung

P-61 Paul Müller, Reinhard Gotzhein, Jens B. Schmitt (Hrsg.): Kommunikation in verteilten Systemen


P-63 Roland Kaschek, Heinrich C. Mayr, Stephen Liddle (Hrsg.): Information Systems – Technology and ist Applications

P-64 Peter Liggesmeyer, Klaus Pohl, Michael Goedicke (Hrsg.): Software Engineering 2005

P-65 Gottfried Vossen, Frank Leymann, Peter Lockemann, Wolfried Stucky (Hrsg.): Datenbanksysteme in Business, Technologie und Web

P-66 Jörg M. Haake, Ulrike Lucke, Djamshid Tavangarian (Hrsg.): Del.FL 2005: 3. deutsche e-Learning Fachtagung Informatik

P-67 Armin B. Cremers, Rainer Manthey, Peter Martini, Volker Steinhaage (Hrsg.): INFORMATIK 2005 – Informatik LIVE (Band 1)

P-68 Armin B. Cremers, Rainer Manthey, Peter Martini, Volker Steinhaage (Hrsg.): INFORMATIK 2005 – Informatik LIVE (Band 2)

P-69 Robert Hirschfeld, Ryszard Kowalczyk, Andreas Polze, Matthias Weske (Hrsg.): NODe 2005, GSEM 2005

P-70 Klaus Turowski, Johannes-Maria Zaha (Hrsg.): Component-oriented Enterprise Application (COAE 2005)

P-71 Andrew Torda, Stefan Kurz, Matthias Rarey (Hrsg.): German Conference on Bioinformatics 2005

P-72 Klaus P. Jantke, Klaus-Peter Fähnrich, Wolfgang S. Wittig (Hrsg.): Marktplatz Internet: Von e-Learning bis e-Payment

P-73 Jan von Knop, Wilhelm Haverkamp, Eike Jessen (Hrsg.): “Heute schon das Morgen sehen“

P-74 Christopher Wolf, Stefan Lucks, Po-Wah Yau (Hrsg.): WEWoRC 2005 – Western European Workshop on Research in Cryptology

P-75 Jörg Desel, Ulrich Frank (Hrsg.): Enterprise Modelling and Information Systems Architecture

P-76 Thomas Kirste, Birgitta König-Riess, Key Poustchti, Klaus Turowski (Hrsg.): Mobile Informationssysteme – Potentiale, Hindernisse, Einsatz

P-77 Jana Dittmann (Hrsg.): SICHERHEIT 2006

P-78 K.-O. Wenkel, P. Wagner, M. Morgenstern, K. Luzi, P. Eissermann (Hrsg.): Land- und Ernährungswirtschaft im Wandel – Aufgaben und herausforderungen für die Agrar- und Umweltinformatik

P-79 Bettina Biel, Matthias Book, Volker Gruhn (Hrsg.): Softwareengineering 2006

363
P-80 Mareike Schoop, Christian Huemer, Michael Rebstock, Martin Bichler (Hrsg.): Service-Oriented Electronic Commerce – Proceedings zur Konferenz im Rahmen der Multikonferenz Wirtschaftsinformatik 2006


The Titles can be purchased at:

Köllen Druck + Verlag GmbH
Ernst-Robert-Curtius-Str. 14
53117 Bonn
Fax: +49 (0)228/9898222
E-Mail: druckverlag@koellen.de