ICOOOLPS'2006
ECOOP Workshop on Implementation, Compilation, Optimization of Object-Oriented Languages, Programs and Systems

Roland Ducournau[1], Etienne Gagnon[2], Chandra Krintz[3], Philippe Mulet[4], Jan Vitek[5], and Olivier Zendra[6]

[1] LIRMM, France
[2] UQAM, Canada
[3] UCSB, USA
[4] IBM, France
[5] Purdue University, USA
[6] INRIA-LORIA, France

Abstract. ICOOOLPS'2006 was the first edition of ECOOP-ICOOOLPS workshop. It intended to bring researchers and practitioners both from academia and industry together, with a spirit of openness, to try and identify and begin to address the numerous and very varied issues of optimization. This succeeded, as can be seen from the papers, the attendance and the liveliness of the discussions that took place during and after the workshop, not to mention a few new cooperations or postdoctoral contracts. The 22 talented people from different groups who participated were unanimous to appreciate this first edition and recommend that ICOOOLPS be continued next year. A community is thus beginning to form, and should be reinforced by a second edition next year, with all the improvements this first edition made emerge.

1 Objectives and call for papers

Object-oriented languages are pervasive and play a significant role in computer science and engineering life and sometime appear as ubiquitous and completely mature. However, despite a large number of works, there is still a clear need for solutions for efficient implementation and compilation of OO languages in various application domains ranging from embedded and real-time systems to desktop systems.

The ICOOOLPS workshop thus aims to address this crucial issue of optimization in OO languages, programs and systems. It intends to do so by bringing together researchers and practitioners working in the field of object-oriented languages implementation and optimization. Its main goals are identifying fundamental bases and key current issues pertaining to the efficient implementation, compilation and optimization of OO languages, and outlining future challenges and research directions.

Topics of interest for ICOOOLPS include but are not limited to:

- implementation of fundamental OOL features:
  - inheritance (object layout, late binding, subtype test...)
  - genericity (parametric types)
  - memory management
- runtime systems:
  - compilers
  - linkers
  - virtual machines
- optimizations:
  - static and dynamic analyses
  - adaptive virtual machines
- resource constraints:
  - real-time systems
  - embedded systems (space, low power)...
- relevant choices and tradeoffs:
  - constant time vs. non-constant time mechanisms
  - separate compilation vs. global compilation
  - dynamic loading vs. global linking
  - dynamic checking vs. proof-carrying code

This workshop tries to identify fundamental bases and key current issues pertaining to the efficient implementation and compilation of OO languages, in order to spread them further amongst the various computing systems. It is also intended to extend this synthesis to encompass future challenges and research directions in the field of OO languages implementation and optimization.

Finally, this workshop is intended to become a recurrent one. Thus, the organization (most relevant format and hottest topics) of this workshop future occurrences will be adapted by the organizers and attendees according to the main outcome of this workshop discussions.

In order to have a solid basis on which the discussions could be based and to keep them focused, each prospective participant was required to submit either a short paper describing ongoing work or a position paper describing an open issue, likely solutions, drawbacks of current solutions or alternative solutions to well known problems. Papers had to be written in English and their final version could not exceed 8 pages in LNCS style.

2 Organizers

Olivier ZENDRA (chair), INRIA-LORIA, Nancy, France.

Email:	`olivier.zendra@loria.fr`
Web:	`http://wwW.loria.fr/~zendra`
Address:	INRIA / LORIA 615 Rue du Jardin Botanique BP 101 54602 Villers-Lès-Nancy Cedex, FRANCE

Olivier Zendra is a full-time permanent computer science researcher at INRIA / LORIA, in Nancy, France. His research topics cover compilation, optimization and automatic memory management. He worked on the compilation and optimization of object-oriented languages and was one of the two people who created and implemented SmartEiffel, The GNU Eiffel Compiler (at the time SmallEiffel). His current research application domains are compilation, memory management and embedded systems, with a specific focus on low energy.

Roland DUCOURNAU (co-chair), LIRMM, Montpellier, France.
Email: ducour@lirmm.fr
Web: http://www.lirmm.fr/~ducour
Address: LIRMM,
161, rue Ada
34392 Montpellier Cedex 5, FRANCE

Roland Ducournau is Professor of Computer Science at the University of Montpellier. In the late 80s, while with Sema Group, he designed and developed the YAFOOL language, based on frames and prototypes and dedicated to knowledge based systems. His research topics focuses on class specialization and inheritance, especially multiple inheritance. His recent works are dedicated to implementation of OO languages.

Etienne GAGNON, UQAM, Montréal, Québec, Canada.
Email: egagnon@sablevm.org
Web: http://www.info2.uqam.ca/~egagnon
Address: Département d'informatique
UQAM
Case postale 8888, succursale Centre-ville
Montréal (Québec) Canada / H3C 3P8

Etienne Gagnon is a Professor of Computer Science at Université du Québec à Montréal (UQAM) since 2001. Etienne has developed the SableVM portable research virtual machine for Java, and the SableCC compiler framework generator. His research topics include language design, memory management, synchronization, verification, portability, and efficient interpretation techniques in virtual machines.

Chandra KRINTZ, UC Santa Barbara, CA, USA.
Email: ckrintz@cs.ucsb.edu
Web: http://www.cs.ucsb.edu/~ckrintz
Address: University of California
Engineering I, Rm. 1121
Department of Computer Science
Santa Barbara, CA 93106-5110, USA

Chandra Krintz is an Assistant Professor at the University of California, Santa Barbara (UCSB); she joined the UCSB faculty in 2001. Chandra's research interests include automatic and adaptive compiler and virtual runtime techniques for object-oriented languages that improve performance and increase battery life. In particular, her work focuses on exploiting repeating patterns in the time-varying behavior of underlying resources, applications, and workloads to guide dynamic optimization and specialization of program and system components.

Philippe MULET, IBM, Saint-Nazaire, France.
Email: philippe_mulet@fr.ibm.com
Address: IBM France - Paris Laboratory
69, rue de la Vecquerie
44600 Saint-Nazaire, France

Philippe Mulet is the lead for the Java Development Tooling (JDT) Eclipse subproject, working at IBM since 1996; he is currently located in Saint-Nazaire (France). In late 1990s, Philippe was responsible for the compiler and codeassist tools in IBM Java Integrated Development Environments (IDEs): VisualAge for Java standard and micro editions. Philippe then became in charge of the Java infrastructure for the Eclipse platform, and more recently of the entire Java tooling for Eclipse. Philippe is a member of the Eclipse Project PMC. Philippe is also a member of the expert group on compiler API (JSR199), representing IBM. His main interests are in compilation, performance, scalability and meta-level architectures.

Jan VITEK, Purdue Univ., West Lafayette, IN, USA.
Email: jv@cs.purdue.edu
Web: http://www.cs.purdue.edu/homes/jv
Address: Dept. of Computer Sciences
Purdue University
West Lafayette, IN 47907, USA

Jan Vitek is an Associate Professor in Computer Science at Purdue University. He leads the Secure Software Systems lab. He obtained his PhD from the University of Geneva in 1999, and a MSc from the University of Victoria in 1995. Prof. Vitek research interests include programming language, virtual machines, mobile code, software engineering and information security.

3 Participants

ICOOOLPS attendance was limited to 30 people. In addition, as mentioned in the call for paper, only people who were giving a talk were initially allowed to attend ICOOOLPS. However, since on-site there were a lot of other people interested in the workshop, the rules were relaxed to match the demand.

Finally, 22 people from 8 countries attended this first edition of ICOOOLPS, filling the allocated room, as detailed in the following table:

First name	Name	Affiliation	Country	Email
Daniel	Benquides	EMN - Nantes	France	lbenquid@emn.fr
Rhodes	Brown	Univ. of Victoria	Canada	rhodesb@cs.uvic.ca
Roland	Ducournau	Univ. of Montpellier	France	ducour@lirmm.fr
Andres	Fortier	LIFIA (UNLP)	Argentina	andres@lifia.info.unlp.edu.ar
Etienne	Gagnon	UQAM	Canada	egagnon@sablevm.org
Olivier	Gruber	IBM Research	France	ogruber@us.ibm.com
Elisa	Gonzales Bax	VUB	Belgium	egonzale@vub.ac.be
Teresa	Higuera	UCM	Spain	mthiguer@dacya.ucm.es
Yann	Hodique	USTL Lille 1	France	hodique@lifl.fr
Richard	Jones	Univ. of Kent	UK	R.E.Jones@kent.ac.uk
Susanne	Jucknath	TU Berlin	Germany	susannej@cs.tu-berlin.de
Eric	Jul	DIKU	Denmark	eric@diku.dk
Chandra	Krintz	UC Santa Barbara	USA	ckrintz@cs.ucsb.edu
Paul	McGregor	Goldman Sachs	USA	paul.regtech.mcgregor@gs.com
Philippe	Mulet	IBM Rational Software	France	philippe_mulet@fr.ibm.com
Marco	Pistoia	IBM Watson Research	USA	pistoia@us.ibm.com
Jean	Privat	Univ. of Montpellier	France	privat@lirmm.fr
Guillaume	Salagnac	Verimag lab.	France	Guillaume.Salagnac@imag.fr
Christophe	Rippert	Verimag lab.	France	Christophe.Rippert@imag.fr
Jan	Vitek	Purdue Univ.	USA	v@cs.purdue.edu
Hiroshi	Yamauchi	Purdue Univ.	USA	yamauchi@cs.purdue.edu
Olivier	Zendra	INRIA-LORIA	France	Olivier.Zendra@loria.fr

4 Contributions

All the papers and presentations are available from the ICOOOLPS web site at `http://icooolps.loria.fr`.

4.1 Real-time and embedded systems

This session clustered papers and questions related to real-time and/or embedded systems.

In "Java for Hard Real-Time", Jan Vitek presented the numerous challenges caused by trying to put Java, a high-level, expressive object-oriented language, in systems that require hard real-time guarantees (such as avionics). He detailed OVM (Open Virtual Machine), developed at Purdue Univ.

In "Can small and open embedded systems benefit from escape analysis ?" Gilles Grimaud, Yann Hodique and Isabelle Simplot-Rey explained how a commonly known technique, escape analysis, can be used in small constrained embedded systems to improve time through a better memory management, at low cost.

In "Memory and compiler optimizations for low-power in embedded systems" Olivier Zendra aimed at raising awareness about low-power and low-energy issues in embedded systems among the object-oriented and languages communities. He showed how mostly known time- or size-optimization techniques can be and should observed from a different point of view, namely energy. He surveyed a number of solutions and outlined remaining challenges.

Based on the papers, presentations and discussions in this session, several trends clearly show.

First, the ever increasing importance of embedded systems, whether they are real-time of not, in software research.

Second, it could be argued (and has in the past) that, in such highly constrained systems, the powerful features and expressiveness of object-oriented languages and their compiler are too expensive to be relied on. However, a trend can be seen in research that tries to bring these features to smaller and smaller systems, trying to bridge a gap. Hence, "object-oriented" and "embedded" are no longer opposite terms, but on the contrary form together a very active and promising research area.

Finally, new challenges (power, energy...) emerge, that require either the proper integration of known techniques, or the development of new ones. As such, being able to take into account low-level (hardware) features at high level (OO, JVM...) appear quite challenging but offer a high potential.

It is however of course always very challenging to both be able to increase the level of abstraction and at the same time get a finer, lower-level understanding of the application.

4.2 Memory management

This session grouped papers whose main topic was memory management.

"Efficient Region-Based Memory Management for Resource-limited Real-Time Embedded Systems", by Chaker Nakhli, Christopher Rippert, Guillaume Salagnac and Sergio Yovine, presents a static algorithm to make dynamic region-based memory allocations for Java applications that have real-time constraints. M. Teresa Higuera-Toledano addresses close issues, aiming at "Improving the Scoped Memory-Region GC of Real-Time Java".

This confirms the growing importance of real-time for object-oriented languages in general, and more specifically Java, with the RTSJ (Real-Time Specification for Java). This is additional evidence for the trend we mentioned in section 4.1 towards bringing high expressiveness, easy to use languages in smaller and/or more constrained systems

Richard Jones and Chris Ryder argued, in "Garbage Collection Should be Lifetime Aware", that the efficiency of garbage collectors can be improved by making them more aware of actual objects lifetimes in the mutator. Indeed, even current generational garbage collectors generally observe the mutator with a rather coarse view, and do not provide enough flexibility when clustering objects

according to their expected lifetimes. This is an area where the potential gain in performance is quite considerable.

This presentation and the following discussions where quite refreshing and confirm that even in a rather technical and well explored domain, new ideas can emerge that have both high potential and are relatively easy to grasp, especially when explained in a metaphorical way.

Finally, in "Enabling Efficient and Adaptive Specialization of Object-Oriented, Garbage Collected Programs", Chandra Krintz defended code optimizations (specialization) which are aggressively and speculatively performed and can be, if the need arises, invalidated on the fly, through OSR (On Stack Replacement).

Here again, we can spot the trend that was mentioned during the discussion for session 4.1 and tends to bridge the gap between hardware and software. Indeed, the presented technique bear some similarities with what processors do in hardware, with speculative execution and invalidation.

All the above mentioned papers and discussions make it clear that memory management is an area where a lot of progress can be made, be it in small or large strides. Memory management is furthermore an area which has an important impact over program speed. In addition to speed, memory management can also affect very significantly energy usage, as discussed during session 4.1. Memory-targeted optimizations should thus always be taken into account when trying to reach higher performance.

4.3 Optimization

This session was devoted to papers and questions related known or very specific optimizations.

In "OO specific redundancy elimination techniques", Rhodes Brown and Nigel Horspool advocated a holistic view of optimization where not only one but in fact the whole set of program properties are taken into account together, without forgetting their mutual interactions. They presented how annotations could be used, in conjunction with static and dynamic invariants, to improve program performance.

This echoes a relatively novel trend in object-oriented program optimization, that tries to analysis not only one specific optimization, but optimization composition or sequences.

"Performing loop optimizations offline", by Hiroshi Yamauchi, shows how the overhead of some loop optimizations can be removed from execution time by performing them offline. This is especially important in the context of system that require a high level of responsiveness.

This shows how even very specific and potentially common optimizations can be reconsidered in the light of new constraints, for example those of the real-time systems that were already mentioned in session 4.1.

Here again, as in session 4.1, we see that not optimization work tends to evolve. First, they more and more focus not only on one criterion which is often

program speed, but also integrate other criteria, such as responsiveness, that correspond more to new current computing systems with tight real-time and/or space constraints. Second, larger sets of optimizations tend to be considered together, as optimization sequences or compositions, to better encompass the complexity inherent to real life systems and the various interactions that can take place when optimizing.

4.4 Abstraction and frameworks

This last session aimed at regrouping papers and talks about higher level or broader points of view for optimization.

In "Efficient Separate Compilation of OO languages" Jean Privat, Floréal Morandat and Roland Ducournau present a scheme to reconcile separate and global compilation, hence global optimization. They detail their practical and implemented solution in the context of an object-oriented language.

This is truly another example of research work trying to successfully bridge a gap: the gap between separate compilation, which commonly used in industry, and global compilation, that brings the best optimization results.

"Java Framework for Runtime Modules" by Olivier Gruber and Richard Hall is a paper that takes a broad view of optimization. It proposes a framework to more easily build modules and reuse components and that could be integrated in the Java Runtime Environment.

By bringing this discussion to the workshop, the authors clearly enlarged to scope of the discussions and tried to connect the optimization and software engineering communities. This kind of openness is quite useful in a workshop so as to foster slightly unusual cooperation and work.

Finally, "The Importance of Abstraction in Efficient OO implementations" by Eric Jul, made a case for clearly and strictly separating the abstraction (language level) and the concrete (implementation) level. This indeed gives more freedom to the implementer, hence more possibilities for optimizations, while the language user does not have to worry about low-level details but only about the semantics of the program.

Here, we see that bridging the gap between what is expressed and what is implemented is important, but should not be left to the developer. That's the compiler's job, or rather the compiler implementers' job. Eric's position is thus quite important to remind us not to pollute the high level with too many low-level details. Of course, one question that remains open is how to properly abstract things, especially low-level, possible hardware, details.

This session was interesting in that it made the workshop participant not forget a high-level, software engineering oriented point of view and the related issues. Indeed, there is always a risk that, being focused on one specific optimization, the researcher forgets the larger picture. Considering issues at a high level, with abstraction, may avoids getting swamped in details. Reuse of optimizations, like reuse of modules, is a requirement to evolve from software optimizations as

a craftsmanship to software optimizations as an industrial process. Of course, quite some work remains to be done before we're there, but it a goal worth aiming at.

5 Closing debates

The presentation sessions finished later than scheduled. As a consequence, the discussion time that was planned at the end of the workshop was shorter than initially expected. This may have somehow limited the discussions.

This is one of the points that shall be improved in future occurrences of ICOOOLPS (see section 6).

The discussion session was very spontaneous, with attendees being encouraged to bring their favorite topic, main itch, etc. From their summary emerge two main treads.

5.1 "Written down in code vs. inferred"

"The user knows" what is intended and what is going on in an application. Thus, it seems to make sense to have the developer *annotate the code with meta information*, that can then be used by the compiler to optimize.

However the code — especially for libraries — can be reused in a different context. Would the annotations remain valid ? This seems to call for *context-dependent annotations.*

But what is "the context" when you write 10% and reuse 90% ?

"Annotation-guided optimization" looks quite appealing. However, the analyses done by the compiler have to be performed anyway, whether annotations are present or not. What should the compiler do if annotations appear to be *contradictory* with what it infers ?

Relying on developer annotations puts a burden on her/him. But we all know there are good developers and not-so-good ones, with a majority in the second category, so is it *realistic* ?

There are similarities between the user-software interface and the hardware-software interface: interactions are needed, as well as information passing (both ways). For example, feedback to the user is very useful, so that s/he can improve her/his coding.

The developer knows the application, but should not have to worry about the underlying OS, hardware, etc. Annotations thus should make it possible to express *what the developer wants, not how to do it.*

A lot of interest was expressed in this long debate with many attendees involved.

Annotations by the developer seem appealing but their nature is an issue. A lot depends on the developer level, so how far can annotations be trusted ?

This discussion thread certainly is worth digging deeper into during the next edition of ICOOOLPS.

5.2 "Do threads make sense ?"

Isn't the threading model fundamentally flawed, that is inappropriate/problematic for object-oriented design and implementation ? Indeed, threads in Java are build on top of an existing, independent model. They thus seem poorly integrated.

See Hans-J. Boehm, "Threads Cannot Be Implemented as a Library" - PLDI 2006 and Edward A. Lee, "The Problem with Threads" - IEEE Computer, May 2006.

This topic, however, did not spark much debate, maybe because of lack of time.

6 Conclusion and perspectives

This first edition of ICOOOLPS was able to reach one its goals: bringing together people from various horizons, in an open way, so as to foster new, original and fruitful discussions and exchanges pertaining to optimizations. The presence of people from 8 different countries, from academia and industry, researcher as well as practioners, is in itself a success. The fact that more people that expected showed up is another.

Thanks to the skills of the speakers and active participation of the attendants, the discussions were lively, open-minded and allowed good exchanges. Identifying the mains challenges for optimization is not that easy though. Indeed, as emerged more clearly during ICOOOLPS, optimizations for object-oriented languages come in variety of contexts with very different constraints (embedded, real-time, dynamic, legacy...). The optimizations criteria considered, thus the goal, also tend to differ a lot: speed, size, memory footprint, more recently energy... In addition, all these have to be tackled keeping in my higher-level, software engineering-oriented issues, such as modularity, composability, reusability, ease of use...

Some trends can however be sketched. Optimizations tend to encompass more and more target criteria (multi-criteria rather than single criterion), such as energy and speed, or memory footprint and responsiveness. Multiple optimizations tend to be evaluated in conjunction, as sequences of optimizations, rather than in an isolated way. Separating semantics and implementation is crucial, for expressiveness, ease of use and the possibility to perform optimizations at compile level. However, it appears at the same time necessary to be able to better take into account the actual execution of a program when optimizing, that is better take into account the behavior of the software and the hardware as well.

Large challenges thus remain, and should be addressed by the community. That's what ICOOOLPS intend to do in its next editions.

Indeed, the perspectives for the ECOOP-ICOOOLPS workshop appear quite bright. One of the questions was whether this workshop should be pursued in

the next years, and with which periodicity. The answer was unanimously positive: attendees are in favor of continuing the workshop next year with a yearly periodicity.

Overall satisfaction is thus quite high for this very first edition of the workshop.

A few ways to improve ICOOOLPS emerged during the workshop and should be taken into account in 2007:

- Presentations should be significantly shorter, to save time for longer discussions. The later should take place during the sessions, for example one session comprising 3 talks lasting 5 to 10 minutes each, plus a 30 to 60 minutes discussion.
- More time could also be allotted for discussions at the very end of the workshop.
- Session report drafts should be written during a session (papers and talks) and maybe briefly discussed at the end of each session (not after the workshop).
- Attendees could be given the possibility to submit (written) questions to paper presenters before the workshop itself. This would give a starting base for discussions, or at the very least the question "session" at the end of each talk.
- The workshop could be open to anyone, not only authors/speakers. This year indeed, although no call for participation had been issued after the call for paper was closed, because the workshop was for presenters only, many more people asked to be admitted in. Since the aim of an ECOOP workshop is to foster discussions and exchanges, refusing interested people would have been a bad idea. Having everyone (not only authors) present themselves and their work in a few minutes would be an added value.
- A larger room is necessary. 15 attendants were expected but 22 came, so the room was very crowded, which made it difficult for some attendants to properly see the presentation slides.

7 Related work

In order to provide a fixed point for ICOOOLPS related matters, the web site for the workshop is maintained at `http://icooolps.loria.fr`. All the papers and presentations done for ICOOLPS'2006 are freely available there.

References

1. Mohammed Javed Absar and Francky Catthoor. Compiler-based approach for exploiting scratch-pad in presence of irregular array access. In *DATE*, pages 1162–1167, 2005.
2. Wolfram Amme, Niall Dalton, Michael Franz, and Jeffery von Ronne. Safetsa: A type safe and referentially secure mobile-code representation based on static single assignment form. In *PLDI*, pages 137–147, 2001.

3. R. Athavale, Narayanan Vijaykrishnan, Mahmut T. Kandemir, and Mary Jane Irwin. Influence of array allocation mechanisms on memory system energy. In *IPDPS*, page 3, 2001.
4. Oren Avissar, Rajeev Barua, and Dave Stewart. An optimal memory allocation scheme for scratch-pad-based embedded systems. *Transaction. on Embedded Computing Systems.*, 1(1):6–26, 2002.
5. David F. Bacon, Perry Cheng, and V. T. Rajan. A real-time garbage collector with low overhead and consistent utilization. In *POPL*, pages 285–298, 2003.
6. Rajeshwari Banakar, Stefan Steinke, Bo-Sik Lee, M. Balakrishnan, and Peter Marwedel. Scratchpad memory: design alternative for cache on-chip memory in embedded systems. In *10th international symposium on Hardware/software codesign (CODES'02)*, pages 73–78, New York, NY, USA, 2002. ACM Press.
7. Matthew Q. Beers, Christian Stork, and Michael Franz. Efficiently verifiable escape analysis. In *ECOOP*, pages 75–95, 2004.
8. Stephen Blackburn, Richard Jones, Kathryn S. McKinley, and J. Eliot B. Moss. Beltway: Getting around garbage collection gridlock. In *PLDI*, pages 153–164, 2002.
9. Stephen M. Blackburn, Perry Cheng, and Kathryn S. McKinley. Oil and water? high performance garbage collection in java with mmtk. In *ICSE*, pages 137–146, 2004.
10. Bruno Blanchet. Escape analysis for object-oriented languages: Application to java. In *OOPSLA*, pages 20–34, 1999.
11. Bruno Blanchet. Escape analysis for java$^{\text{tm}}$: Theory and practice. *ACM Trans. Program. Lang. Syst.*, 25(6):713–775, 2003.
12. Gregory Bollella and James Gosling. The real-time specification for java. *IEEE Computer*, 33(6):47–54, 2000.
13. Sigmund Cherem and Radu Rugina. Region analysis and transformation for java programs. In *ISMM*, pages 85–96, 2004.
14. Darren D. Cofer and Murali Rangarajan. Formal modeling and analysis of advanced scheduling features in an avionics rtos. In *EMSOFT*, pages 138–152, 2002.
15. Dominique Colnet, Philippe Coucaud, and Olivier Zendra. Compiler support to customize the mark and sweep algorithm. In *ISMM*, pages 154–165, 1998.
16. V. Delaluz, M. Kandemir, N. Vijaykrishnan, M. J. Irwin, A. Sivasubramaniam, and I. Kolcu. Compiler-directed array interleaving for reducing energy in multi-bank memories. In *2002 conference on Asia South Pacific design automation/VLSI Design (ASP-DAC'02)*, page 288, Washington, DC, USA, 2002. IEEE Computer Society.
17. Morgan Deters and Ron Cytron. Automated discovery of scoped memory regions for real-time java. In *MSP/ISMM*, pages 132–142, 2002.
18. David Detlefs. A hard look at hard real-time garbage collection. In *ISORC*, pages 23–32, 2004.
19. Angel Dominguez, Sumesh Udayakumaran, and Rajeev Barua. Heap data allocation to scratch-pad memory in embedded systems. *Journal of Embedded Computing (JEC)*, 1(4), 2005.
20. Matthew B. Dwyer, John Hatcliff, Robby, and Venkatesh Prasad Ranganath. Exploiting object escape and locking information in partial-order reductions for concurrent object-oriented programs. *Formal Methods in System Design*, 25(2-3):199–240, 2004.
21. Robert P. Fitzgerald and David Tarditi. The case for profile-directed selection of garbage collectors. In *ISMM*, pages 111–120, 2000.

22. Etienne M. Gagnon and Laurie J. Hendren. Sablevm: A research framework for the efficient execution of java bytecode. In *Java Virtual Machine Research and Technology Symposium*, pages 27–40, 2001.
23. Robert Graybill and Rami Melhem. *Power aware computing.* Kluwer Academic Publishers, Norwell, MA, USA, 2002.
24. David Grove and Craig Chambers. A framework for call graph construction algorithms. *ACM Trans. Program. Lang. Syst.*, 23(6):685–746, 2001.
25. Richard S. Hall. A policy-driven class loader to support deployment in extensible frameworks. In *Component Deployment*, pages 81–96, 2004.
26. Timothy L. Harris. Dynamic adaptive pre-tenuring. In *ISMM*, pages 127–136, 2000.
27. M. Teresa Higuera-Toledano, Valérie Issarny, Michel Banâtre, Gilbert Cabillic, Jean-Philippe Lesot, and Frédéric Parain. Region-based memory management for real-time java. In *ISORC*, pages 387–394, 2001.
28. Martin Hirzel, Amer Diwan, and Matthew Hertz. Connectivity-based garbage collection. In *OOPSLA*, pages 359–373, 2003.
29. Martin Hirzel, Johannes Henkel, Amer Diwan, and Michael Hind. Understanding the connectivity of heap objects. In *MSP/ISMM*, pages 143–156, 2002.
30. J. Hom and U. Kremer. Energy management of virtual memory on diskless devices. In *Workshop on Compilers and Operating Systems for Low Power (COLP'01)*, Barcelone, Espagne, September 2001.
31. Jerry Hom and Ulrich Kremer. Inter-program optimizations for conserving disk energy. In *2005 international symposium on Low power electronics and design (ISLPED'05)*, pages 335–338, New York, NY, USA, 2005. ACM Press.
32. ITRS. International technology roadmap for semiconductors, 2005. http://public.itrs.net.
33. Richard Jones and Rafael Lins. *Garbage Collection: Algorithms for Automatic Dynamic Memory Management.* Wiley, 1996.
34. M. Kandemir, N. Vijaykrishnan, M.J. Irwin, W. Ye, and I. Demirkiran. Register relabeling: A post compilation technique for energy reduction. In *Workshop on Compilers and Operating Systems for Low Power (COLP'00)*, Philadelphie, PA, USA, October 2000.
35. Chandra Krintz and Brad Calder. Using annotation to reduce dynamic optimization time. In *PLDI*, pages 156–167, 2001.
36. M. Lee, V. Tiwari, S. Malik, and M. Fujita. Power analysis and minimization techniques for embedded dsp software. *IEEE Transactions on Very Large Scale Integration*, 5, March 1997.
37. Pierre-Etienne Moreau and Olivier Zendra. Gc^2: a generational conservative garbage collector for the atterm library. *J. Log. Algebr. Program.*, 59(1-2):5–34, 2004.
38. Steven S. Muchnick. *Advanced compiler design and implementation.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.
39. Priya Nagpurkar, Chandra Krintz, Michael Hind, Peter F. Sweeney, and V. T. Rajan. Online phase detection algorithms. In *CGO*, pages 111–123, 2006.
40. George C. Necula. Proof-carrying code. In *POPL*, pages 106–119, 1997.
41. Nathaniel Nystrom, Michael R. Clarkson, and Andrew C. Myers. Polyglot: An extensible compiler framework for java. In *CC*, pages 138–152, 2003.
42. Krzysztof Palacz and Jan Vitek. Java subtype tests in real-time. In *ECOOP*, pages 378–404, 2003.

43. Filip Pizlo, J. M. Fox, David Holmes, and Jan Vitek. Real-time java scoped memory: Design patterns and semantics. In *ISORC*, pages 101–110, 2004.
44. Francesco Poletti, Paul Marchal, David Atienza, Luca Benini, Francky Catthoor, and Jose Manuel Mendias. An integrated hardware/software approach for run-time scratchpad management. In *DAC*, pages 238–243, 2004.
45. Jean Privat and Roland Ducournau. Link-time static analysis for efficient separate compilation of object-oriented languages. In *PASTE*, pages 20–27, 2005.
46. Rajiv A. Ravindran, Robert M. Senger, Eric D. Marsman, Ganesh S. Dasika, Matthew R. Guthaus, Scott A. Mahlke, and Richard B. Brown. Partitioning variables across register windows to reduce spill code in a low-power processor. *IEEE Transaction on Computers*, 54(8):998–1012, 2005.
47. Fridtjof Siebert. Hard real-time garbage-collection in the jamaica virtual machine. In *RTCSA*, pages 96–102, 1999.
48. Sunil Soman, Chandra Krintz, and David F. Bacon. Dynamic selection of application-specific garbage collectors. In *ISMM*, pages 49–60, 2004.
49. Sriraman Tallam and Rajiv Gupta. Bitwidth aware global register allocation. In *POPL*, pages 85–96, 2003.
50. Mads Tofte and Jean-Pierre Talpin. Region-based memory management. *Inf. Comput.*, 132(2):109–176, 1997.
51. John Whaley and Martin C. Rinard. Compositional pointer and escape analysis for java programs. In *OOPSLA*, pages 187–206, 1999.
52. Seungdo Woo, Jungroin Yoon, and Jihong Kim. Low-power instruction encoding techniques. In *SOC Design Conference*, 2001.
53. Fen Xie, Margaret Martonosi, and Sharad Malik. Intraprogram dynamic voltage scaling: Bounding opportunities with analytic modeling. *ACM Transactions on Architure and Code Optimization (TACO)*, 1(3):323–367, 2004.
54. Olivier Zendra and Karel Driesen. Stress-testing control structures for dynamic dispatch in java. In *Java Virtual Machine Research and Technology Symposium*, pages 105–118, 2002.
55. Youtao Zhang and Rajiv Gupta. Data compression transformations for dynamically allocated data structures. In *11th International Conference on Compiler Construction (CC'02), Lecture Notes in Computer Science*, volume 2304, pages 14–28, London, UK, 2002. Springer-Verlag.
56. Xiaotong Zhuang, ChokSheak Lau, and Santosh Pande. Storage assignment optimizations through variable coalescence for embedded processors. In *LCTES '03: 2003 ACM SIGPLAN conference on Language, Compiler, and Tool for Embedded Systems*, pages 220–231, New York, NY, USA, 2003. ACM Press.