FOUNDATIONS OF DEPENDABLE COMPUTING
System Implementation

edited by

Gary M. Koob
Clifford G. Lau

KLUWER ACADEMIC PUBLISHERS

FOUNDATIONS OF DEPENDABLE COMPUTING

System Implementation

edited by

Gary M. Koob
Clifford G. Lau
Office of Naval Research

KLUWER ACADEMIC PUBLISHERS
Boston / Dordrecht / London

Distributors for North America:
Kluwer Academic Publishers
101 Philip Drive
Assinippi Park
Norwell, Massachusetts 02061 USA

Distributors for all other countries:
Kluwer Academic Publishers Group
Distribution Centre
Post Office Box 322
3300 AH Dordrecht, THE NETHERLANDS

Library of Congress Cataloging-in-Publication Data
Foundations of dependable computing. System implementation / edited by Gary M. Koob, Clifford G. Lau.
p. cm. -- (The Kluwer international series in engineering and computer science ; 0285)
Includes bibliographical references and index.
ISBN 0-7923-9486-0
1. Electronic digital computers--Reliability. 2. Real-time data processing. 3. Fault-tolerant computing. 4. Systems engineering. I. Koob, Gary M., 1958- . II. Lau, Clifford. III. Series: Kluwer international series in engineering and computer science ; SECS 0285.
QA76.5.F624 1994
004.2'2--dc20 94-29138
CIP

Copyright © 1994 by Kluwer Academic Publishers

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, mechanical, photo-copying, recording, or otherwise, without the prior written permission of the publisher, Kluwer Academic Publishers, 101 Philip Drive, Assinippi Park, Norwell, Massachusetts 02061

Printed on acid-free paper.

Printed in the United States of America

FOUNDATIONS OF DEPENDABLE COMPUTING

System Implementation

THE KLUWER INTERNATIONAL SERIES IN ENGINEERING AND COMPUTER SCIENCE

OFFICE OF NAVAL RESEARCH
Advanced Book Series

Consulting Editor

André M. van Tilborg

Other titles in the series:

FOUNDATIONS OF DEPENDABLE COMPUTING: Models and Frameworks for Dependable Systems, edited by Gary M. Koob and Clifford G. Lau
ISBN: 0-7923-9484-4

FOUNDATIONS OF DEPENDABLE COMPUTING: Paradigms for Dependable Applications, edited by Gary M. Koob and Clifford G. Lau
ISBN: 0-7923-9485-2

PARALLEL ALGORITHM DERIVATION AND PROGRAM TRANSFORMATION, edited by Robert Paige, John Reif and Ralph Wachter
ISBN: 0-7923-9362-7

FOUNDATIONS OF KNOWLEDGE ACQUISITION: Cognitive Models of Complex Learning, edited by Susan Chipman and Alan L. Meyrowitz
ISBN: 0-7923-9277-9

FOUNDATIONS OF KNOWLEDGE ACQUISITION: Machine Learning, edited by Alan L. Meyrowitz and Susan Chipman
ISBN: 0-7923-9278-7

FOUNDATIONS OF REAL-TIME COMPUTING: Formal Specifications and Methods, edited by André M. van Tilborg and Gary M. Koob
ISBN: 0-7923-9167-5

FOUNDATIONS OF REAL-TIME COMPUTING: Scheduling and Resource Management, edited by André M. van Tilborg and Gary M. Koob
ISBN: 0-7923-9166-7

此为试读,需要完整PDF请访问: www.ertongbook.com

PREFACE

Dependability has long been a central concern in the design of space-based and military systems, where survivability for the prescribed mission duration is an essential requirement, and is becoming an increasingly important attribute of government and commercial systems where reduced availability may have severe financial consequences or even lead to loss of life. Historically, research in the field of dependable computing has focused on the theory and techniques for preventing hardware and environmentally induced faults through increasing the intrinsic reliability of components and systems (fault avoidance), or surviving such faults through massive redundancy at the hardware level (fault tolerance).

Recent advances in hardware, software, and measurement technology coupled with new insights into the nature, scope, and fundamental principles of dependable computing, however, contributed to the creation of a challenging new research agenda in the late eighties aimed at dramatically increasing the power, effectiveness, and efficiency of approaches to ensuring dependability in critical systems

At the core of this new agenda was a paradigm shift spurred by the recognition that dependability is fundamentally an attribute of applications and services—not platforms. Research should therefore focus on (1) developing a scientific understanding of the manifestations of faults at the application level in terms of their ultimate impact on the correctness and survivability of the application; (2) innovative, application-sensitive approaches to detecting and mitigating this impact; and (3) hierarchical system support for these new approaches.

Such a paradigm shift necessarily entailed a concomitant shift in emphasis away from inefficient, inflexible, hardware-based approaches toward higher level, more efficient and flexible software-based solutions. Consequently, the role of hardware-based mechanisms was redefined to that of providing and implementing the abstractions required to support the higher level software-based mechanisms in an integrated, hierarchical approach to ultradependable system design. This shift was furthermore compatible with an expanded view of "dependability," which had evolved to mean "the ability of the system to deliver the specified (or expected) service." Such a definition encompasses not only survival of traditional single hardware faults and environmental disturbances but more complex and less-well understood phenomena, as well: Byzantine faults, correlated errors, timing faults, software design and process interaction errors, and—most significantly—the unique issues encountered in real-

time systems in which faults and transient overload conditions must be detected and handled under hard deadline and resource constraints.

As sources of service disruption multiplied and focus shifted to their ultimate effects, traditional frameworks for reasoning about dependability had to be rethought. The classical fault/error/failure model, in which underlying anomalies *(faults)* give rise to incorrect values *(errors)*, which may ultimately cause incorrect behavior at the output *(failures)*, required extension to capture timing and performance issues. Graceful degradation, a long standing principle codifying performance/dependability trade-offs must be more carefully applied in real-time systems, where individual task requirements supercede general throughput optimization in any assessment. Indeed, *embedded* real-time systems—often characterized by interaction with physical sensors and actuators—may possess an inherent ability to tolerate brief periods of incorrect interaction, either in the values exchanged or the timing of those exchanges. Thus, a technical failure of the embedded *computer* does not necessarily imply a *system* failure. The challenge of capturing and modeling dependability for such potentially complex requirements is matched by the challenge of successfully exploiting them to devise more intelligent and efficient—as well as more complete—dependability mechanisms.

The evolution to a hierarchical, software-dominated approach would not have been possible without several enabling advances in hardware and software technology over the past decade:

(1) Advances in VLSI technology and RISC architectures have produced components with more chip real estate available for incorporation of efficient concurrent error detection mechanisms and more on-chip resources permitting software management of fine-grain redundancy;

(2) The emergence of practical parallel and distributed computing platforms possessing inherent coarse-grain redundancy of processing and communications resources—also amenable to efficient software-based management by either the system or the application;

(3) Advances in algorithms and languages for parallel and distributed computing leading to new insights in and paradigms for problem decomposition, module encapsulation, and module interaction, potentially exploitable in refining redundancy requirements and isolating faults;

(4) Advances in distributed operating systems allowing more efficient interprocess communication and more intelligent resource management;

(5) Advances in compiler technology that permit efficient, automatic instrumentation or restructuring of application code, program decomposition, and coarse and fine-grain resource management; and

(6) The emergence of fault-injection technology for conducting controlled experiments to determine the system and application-level manifestations of faults and evaluating the effectiveness or performance of fault-tolerance methods.

In response to this challenging, new vision for dependable computing research, the advent of the technological opportunities for realizing it, and its potential for addressing critical dependability needs of Naval, Defense, and commercial systems, the Office of Naval Research launched a five-year basic research initiative in 1990 in *Ultradependable Multicomputers and Electronic Systems* to accelerate and integrate progress in this important discipline. The objective of the initiative is to establish the fundamental principles as well as practical approaches for efficiently incorporating dependability into critical applications running on modern platforms. More specifically, the initiative sought increased effectiveness and efficiency through (1) Intelligent exploitation of the inherent redundancy available in modern parallel and distributed computers and VLSI components; (2) More precise characterization of the sources and manifestations of errors; (3) Exploitation of application semantics at all levels—code, task, algorithm, and domain—to allow optimization of fault-tolerance mechanisms to both application requirements and resource limitations; (4) Hierarchical, integrated software/hardware approaches; and (5) Development of scientific methods for evaluating and comparing candidate approaches.

Implementation of this broad mandate as a coherent research program necessitated focusing on a small cross-section of promising application-sensitive paradigms (including language, algorithm, and coordination-based approaches), their required hardware, compiler, and system support, and a few selected modeling and evaluation projects. In scope, the initiative emphasizes dependability primarily with respect to an expanded class of hardware and environment (both physical and operational) faults. Many of the efforts furthermore explicitly address issues of dependability unique to the domain of embedded real-time systems.

The success of the initiative and the significance of the research is demonstrated by the ongoing associations that many of our principal investigators have forged with a variety of military, Government, and commercial projects whose critical needs are leading to the rapid assimilation of concepts, approaches, and expertise arising from this initiative. Activities influenced to date include the FAA's Advanced Automation System for air traffic control, the Navy's AX project and Next Generation Computing Resources standards program, the Air Force's Center for Dependable Systems, the OSF/1 project, the space station Freedom, the Strategic

Defense Initiative, and research projects at GE, DEC, Tandem, the Naval Surface Warfare Center, and MITRE Corporation.

This book series is a compendium of papers summarizing the major results and accomplishments attained under the auspices of the ONR initiative in its first three years. Rather than providing a comprehensive text on dependable computing, the series is intended to capture the breadth, depth, and impact of recent advances in the field, as reflected through the specific research efforts represented, in the context of the vision articulated here. Each chapter does, however, incorporate appropriate background material and references. In view of the increasing importance and pervasiveness of real-time concerns in critical systems that impact our daily lives—ranging from multimedia communications to manufacturing to medical instrumentation—the real-time material is woven throughout the series rather than isolated in a single section or volume.

The series is partitioned into three volumes, corresponding to the three principal avenues of research identified at the beginning of this preface. While many of the chapters actually address issues at multiple levels, reflecting the comprehensive nature of the associated research project, they have been organized into these volumes on the basis of the primary conceptual contribution of the work. Agha and Sturman, for example, describe a framework (reflective architectures), a paradigm (replicated actors), and a prototype implementation (the Screed language and Broadway runtime system). But because the salient attribute of this work is the use of reflection to dynamically adapt an application to its environment, it is included in the *Frameworks* volume.

Volume I, *Models and Frameworks for Dependable Systems*, presents two comprehensive frameworks for reasoning about system dependability, thereby establishing a context for understanding the roles played by specific approaches presented throughout the series. This volume then explores the range of models and analysis methods necessary to design, validate, and analyze dependable systems.

Volume II, *Paradigms for Dependable Applications,* presents a variety of specific approaches to achieving dependability at the application level. Driven by the higher level fault models of Volume I and buiilt on the lower level abstractions implemented in Volume III, these approaches demonstrate how dependability may be tuned to the requirements of an application, the fault environment, and the characteristics of the target platform. Three classes of paradigms are considered: protocol-based paradigms for distributed applications, algorithm-based paradigms for parallel applications, and approaches to exploiting application semantics in embedded real-time control systems.

Volume III, *System Implementation,* explores the system infrastructure needed to support the various paradigms of Volume II. Approaches to implementing

suppport mechanisms and to incorporating additional appropriate levels of fault detection and fault tolerance at the processor, network, and operating system level are presented. A primary concern at these levels is balancing cost and performance against coverage and overall dependability. As these chapters demonstrate, low overhead, practical solutions are attainable and not necessarily incompatible with performance considerations. The section on innovative compiler support, in particular, demonstrates how the benefits of application specificity may be obtained while reducing hardware cost and run-time overhead.

This third volume in the series completes the picture established in the first two volumes by presenting detailed descriptions of techniques for implementing dependability infrastructure of the system: the operating system, run-time environment, communications, and processor levels.

Section 1 presents design approaches for implementing concurrent error detection in processors and other hardware components. Rennels and Kim apply the principles of self-checking, self-exercising design at the processor level. Rao, et al, use an extension of the well-known Berger code as the mathematical foundation for the construction of self-checking ALUs. These components provide the fundamental building blocks of dependable systems and the abstractions required by higher level software implemented protocols.

The field of fault-tolerant computing was once dominated by concerns over the dependability of the processor. In modern parallel and distributed systems, the reliability of the network is just as critical. Communications dependability—like processor dependability—is more appropriately addressed at the lower layers of the system to minimize impact on performance and to simplify higher-level protocols and algorithms. In the first chapter of Section 2, Bolding and Snyder describe how the inherent attributes of chaotic routing—an approach proposed for its performance advantages in parallel systems—may be adapted to support dependability as well. The use of non-determinism in chaotic routing is the key to realizing its goal of high throughput, but is inappropriate for real-time systems where low latency and predictability are the primary concerns. Shin and Zheng offer the concept of redundant real-time channels as a solution to the problem of dependable end-to-end communications in distributed systems under hard real-time constraints.

Compiler technology has emerged within the past decade as a dominant factor in effectively mapping application demands onto available resources to achieve high utilization. The chapters in Section 3 explore the intersection of compiler optimizations for high performance and compiler transformations to support dependability goals. Fuchs, et al, describe the similarities of compiler transformations intended to support instruction-level recovery from transient errors and the state management requirements encountered when speculative execution is employed in super-scalar architectures. Banerjee, et al, adapt parallelizing compiler technology to the automa-

tion of algorithm-based fault tolerance. The approach considers not only transformations for generating checks, but also partitioning, mapping, and granularity adjustment strategies that balance performance and dependability requirements.

Operating systems support is central to the design of modern dependable systems. Support is required for service abstractions, communication, checkpointing and recovery, and resource/redundancy management. All of these are considered in Section 4. Russinovich, et al, describe how various error detection and recovery mechanisms may be efficiently integrated into an operating system built on the popular Mach microkernel in a manner transparent to the application itself but customizable to its requirements. Protocol-based paradigms for distributed systems all seem to rely on a number of fundamental protocols such as atomic multicast and group membership. Schlichting, et al, have organized these operations into a comprehensive communications "substrate". By exploiting the interdependencies among them, an efficient implementation has been constructed. The problem of resource management in a real-time distributed system becomes even more complex when deadline-constrained redundancy and fault management must be supported. Thuel and Strosnider present a framework for managing redundancy, exceptions, and recovery under hard real-time constraints.

Gary M. Koob
Mathematical, Computer and Information Sciences Division
Office of Naval Research

Clifford G. Lau
Electronics Division
Office of Naval Research

此为试读,需要完整PDF请访问: www.ertongbook.com

ACKNOWLEDGEMENTS

The editors regret that, due to circumstances beyond their control, two planned contributions to this series could not be included in the final publications: "Compiler Generated Self-Monitoring Programs for Concurrent Detection of Run-Time Errors," by J.P. Shen and "The Hybrid Fault Effects Model for Dependable Systems," by C.J. Walter, M.M. Hugue, and N. Suri. Both represent significant, innovative contributions to the theory and practice of dependable computing and their omission diminishes the overall quality and completeness of these volumes.

The editors would also like to gratefully acknowledge the invaluable contributions of the following individuals to the success of the Office of Naval Research initiative in *Ultradependable Multicomputers and Electronic Systems* and this book series: Joe Chiara, George Gilley, Walt Heimerdinger, Robert Holland, Michelle Hugue, Miroslaw Malek, Tim Monaghan, Richard Scalzo, Jim Smith, André van Tilborg, and Chuck Weinstock.

CONTENTS

SECTION 1

DEPENDABLE COMPONENTS

SECTION 1.1

Self-Checking and Self-Exercising Design for Hierarchic Long-Life Fault-Tolerant Systems

David Rennels and Hyeongil Kim[1]

Abstract

This research deals with fault-tolerant computers capable of operating for extended periods without external maintenance. Conventional fault-tolerance techniques such as majority voting are unsuitable for these applications, because performance is too low, power consumption is too high and an excessive number of spares must be included to keep all of the replicated systems working over an extended life. The preferred design approach is to operate as many different computations as possible on single computers, thus maximizing the amount of processing available from limited hardware resources. Fault-tolerance is implemented in a hierarchic fashion. Fault recovery is either done locally within an afflicted computer or, if that is unsuccessful, by the other working computers when one fails. Concurrent error detection is required in the computers making up these systems since errors must be quickly detected and isolated to allow recovery to begin.

This chapter discusses ways of implementing concurrent error detection (i.e., self-checking) and in addition providing self-exercising capabilities that can rapidly expose dormant faults and latent errors. The fundamentals of self-checking design are presented along with an example -- the design of a self-checking self-exercising memory system. A new methodology for implementing self-checking in asynchronous subsystems is discussed along with error simulation results to examine its effectiveness.

1.1.1 Introduction

There is a class of multicomputer applications that require long unmaintained operation in the range of a decade or more. The obvious example is remote sensing, e.g., space satellites. Important new applications of this type are expected for computers that are embedded in long-life host systems where maintenance is expensive or incon-

---

1. Computer Science Department, University of California at Los Angeles. This work was supported by the Office of Naval Research, grant N00014-91-J-1009.

此为试读,需要完整PDF请访问: www.ertongbook.com