THE OFFICE OF NAVAL RESEARCH INITIATIVE ON
ULTRADEPENDABLE MULTICOMPUTERS AND ELECTRONIC SYSTEMS

Gary M. Koob

Computer Science Division
Office of Naval Research
Arlington VA 22217-5000

ABSTRACT

The Office of Naval Research initiative in Ultradependable Multicomputers and Electronic Systems is focused on the development of novel techniques for achieving high dependability in parallel and distributed systems. The objective of the program is to achieve greater effectiveness and efficiency (than traditional massive redundancy) through: increased understanding of physical failure modes and their effects in real systems; increased sensitivity to the structure, semantics, and requirements of real applications; exploitation of the inherent redundancy in parallel systems; and examination of the system- and environment-oriented view naturally shifts much of the burden of ensuring dependable computing onto the software—chiefly the operating system but increasingly the programmer, the language, and the run-time system, as attention has now been refocused on the effects of failures on the application rather than on the hardware.

Of course, some hardware redundancy will always be required but the research questions now are how much and what form that redundancy must take and how to effectively manage it given the expected failure characteristics of the system and their potential effects on the application.

It was this exciting new direction in dependable computing research that inspired the Office of Naval Research to launch a five-year initiative specifically targeted at parallel and distributed platforms and applications. The initiative, jointly managed by the Computer Science and Electronics Divisions in coordination with the Applied Research and Technology Directorate, was predicated on the following tenets:

1. Introduction

Historically, fault tolerance has been a concern primarily in systems where failure could lead to catastrophic consequences (including possible loss of life) or where timely repairs were infeasible or impossible. Such systems include avionics and space-borne systems and transaction-oriented systems where high availability is essential. Unlike communication theory, where reliability problems caused by physical phenomena have traditionally been addressed by incorporating redundancy into the data (i.e., through encoding), the reliability of computer systems has traditionally been considered a hardware design issue and has been attacked through hardware redundancy (i.e., duplication and triplication).

Such hardware-based approaches suffer from lack of efficiency and the inability to detect certain types of faults. Research within the past decade has attempted to address this efficiency problem on two fronts: first, fault detection at the processor level can be done with modest amounts of redundancy (short of duplication) using coding techniques based on restricted fault models; second, the advent of parallel and distributed systems has fostered research into software-based methods using techniques that are based on ignorance of the detailed failure modes and achieve their reliability at the expense of efficiency. Can better understanding of failure modes and error propagation lead to more detailed models and more efficient methods?

2. Efficient solutions require scientific understanding of how system components fail and how errors are manifested at the system and application level. Traditional massive redundancy (duplication and triplication) techniques are based on ignorance of the detailed failure modes and achieve their reliability at the expense of efficiency. Can better understanding of failure modes and error propagation lead to more detailed models and more efficient methods?

3. Effective solutions must take account of the effect of errors on the application program. Software techniques including language primitives and algorithm transformations offer flexible, adaptive, low overhead approaches to tuning the type of redundancy and recovery mechanisms employed to the needs of the application.

4. Software solutions are necessarily based on models, or abstractions, of how the hardware behaves in the presence of failures. Actual hardware behavior may be too complex for software to deal with efficiently. Therefore, the limitations of software-based methods must be identified and a clean interface to hardware established through the use of appropriate abstractions. The

U.S. Government work not protected by U.S. copyright.
research issues then become (1) defining the “right” abstractions, (2) implementing them, and (3) understanding how to use them. Examples of useful abstractions are fail-stop processors, reliable data storage, and reliable interprocessor communications.

(4) The principal Navy interest in dependable computing and a principal historical motivation for much fault-tolerance research is real-time computing. Yet the two research disciplines have almost always been treated as mutually exclusive. A major goal of the Ultradependable Multicomputing initiative is to lay the foundation for a better understanding of the unique requirements and constraints that a real-time environment imposes on dependability approaches.

The initiative therefore focuses on four interrelated primary topics: (1) measurement and modeling; (2) software-based, application-sensitive techniques; (3) hardware abstractions and their implementation; and (4) dependable real-time systems. In the remainder of the paper, some of the approaches being funded through the initiative and their role in the program as a whole will be surveyed. Although this is a basic research program and not a development project, an attempt has been made to construct an integrated, coherent initiative in order to provide the perspective necessary to develop an understanding of the full range of the problem and the tradeoffs involved.

2. Measurement and modeling

All techniques for detecting and handling component failures in a system must be based on a model of the expected failure modes and consequent system behavior. Important characteristics include fault types and statistics, relevant physical details that affect independence assumptions and containment strategies, and knowledge of the workload. Previous models have been oversimplified for mathematical tractability and have rarely been validated with measurements from real systems. An important goal of the initiative is to help re-establish the scientific foundations of fault-tolerant computing through the creation of new, more detailed and realistic models derived from empirical and experimental studies.

Aside from the obvious need for model calibration, there are numerous questions that have arisen in recent years that can only be answered through systematic studies of data from real systems or controlled experiments. The so-called Byzantine generals problem was the focus of much research in agreement protocols for distributed systems in the past decade, yet there is still a debate over whether such malicious faults ever actually occur in real systems. Fault latency—the interval between the occurrence of a fault and its manifestation as an error—is becoming an increasingly important topic of concern since fault interactions are unpredictable and timely fault detection is crucial to the effectiveness of most schemes. The latency problem directly impacts the coverage issue, but there are other related issues, as well. Old assumptions regarding fault correlation are being challenged, partially due to the emergence of transient and intermittent faults as the dominant types in modern systems. Finally, there is a growing body of evidence that the system workload affects both the fault characteristics themselves and the ability of the system to detect errors.

ONR is currently supporting research efforts in modeling and validation at the University of Illinois, Carnegie Mellon University, and Allied-Signal Aerospace Technology Corporation. All involve experimental work, although the Allied project [Walt90] is initially aimed at refinement of theoretical models. A fourth project [Lin90b] at the University of California at San Diego is studying the fundamental failure modes of opto-electronic systems.

Experimental work in dependability typically takes one of three forms: (1) analysis of event logs from production systems, (2) hardware/software monitoring of production systems, (3) controlled fault-injection experiments.

Event log analysis [Lin90a] has the advantage of providing a rich history documenting many machine-years of operation potentially allowing significant, general conclusions to be drawn. Unfortunately this type of “archaeology” is of limited value due to various gaps in the recorded information:

- Event logs typically include non-error related data that must be filtered out.
- The log is inherently incomplete due to poor coverage.
- Statistical knowledge is missing and difficult to estimate due to the rarity of events.
- The record lacks vital contextual information since the system state and workload are unknown at the time of each event.

In spite of these shortcomings, error log analysis has been successfully used at Illinois and CMU to identify error sources, predict system failure, and establish bounds on model parameters. Clustering techniques have been used to distinguish intermittent faults from transient ones and heuristics have been devised to detect patterns of events leading to system failure.

The low observability afforded by error log analysis can be partially ameliorated through custom hardware and software instrumentation of production systems. CMU currently has plans to monitor a campus-wide network of workstations while Illinois is instrumenting several parallel machines running scientific workloads. Although this approach permits the collection of much more detailed information [Chl97], the rarity of events dictates a long term commitment to an experiment whose results may still lack statistical significance.

One way of accelerating the process is through controlled fault injection experiments, being conducted both at CMU and Illinois. Such experiments offer repeatability and the prospect of measuring otherwise elusive quantities, such as fault latency distributions. All experimental work, however, must take place in the context of a model that provides a framework for conducting the experiment and interpreting the results. The principal issue in fault injection, then, is the design of the experiment, including fault types, locations, and distributions and the establishment of controls such as workload. The validity of a set of experiments will depend on how well it reflects the behavior of a real system in a typical environment. Obviously, some preliminary modeling is necessary to guide the experimentation.

All three of these methods have advantages and disadvantages; none is perfect and each is necessary to provide a different piece of the puzzle. The next step after data collection and analysis is to generalize the results into a model that can be used to design strategies for detection, isolation, and recovery.
3. Software-based approaches

The experimental work outlined in the previous section is aimed at establishing more realistic models as the basis for more effective dependable computing techniques. Efficient techniques, however, will also require more attention to the behavior and needs of the applications. The second major thrust of the initiative, then, is focused on the development of application-sensitive, software-based techniques for the construction of dependable programs.

Software-based approaches probably date to the system diagnosis theory developed in the ’60s [Prep67], in which faulty processors in a distributed system are identified through rounds of mutual testing and exchange of test results. This work was the direct precursor of the interactive consistency protocols developed in the ’80s to deal with Byzantine faults [Lamp82]. The SIFT project at the Jet Propulsion Lab in the late ’70s [Wens78] was another step toward changing the focus from hardware solutions to software. Yet SIFT was merely evolutionary in that it employed a dynamic, software implementation of TMR. The goal of ONR’s initiative is to investigate revolutionary software methods that employ sophisticated language and algorithm-based techniques.

Granularity and application differences between distributed and parallel computing platforms dictate different approaches. In the area of distributed computing, we focus on the design and implementation of system level abstractions, primarily the service abstraction, using language constructs. The finer granularity and tighter coupling typical of parallel programming suggests that even greater efficiency can be obtained through algorithm-based techniques. In this section, we first describe the language and abstraction based approaches for distributed systems. Algorithm-based techniques appropriate for parallel systems are then discussed and we conclude with a brief description of automatic, compiler-based transformations of programs for error detection and recovery.

3.1. Languages and abstractions

Much research in the area of distributed systems has concentrated on the service paradigm whereby processes request services such as a clock value from the system. Reliability has always been an issue in distributed systems and research on reliable implementation of services has focused on replication strategies and protocols for communication, agreement, and message ordering. We seek to investigate new implementation strategies capable of exploiting application semantics to a greater degree.

Work at Cornell [Schn90], for example, is studying semantic-based data distribution schemes that are as robust as replication but potentially more efficient. Such schemes evaluate a distributed representation not only in terms of its resilience to node failures but also in terms of the complexity of implementing the associated operations of interest and recovering from failure.

Research at the University of Arizona [Schl90] is examining various protocols for fault-tolerant distributed computing, including monitoring, membership, checkpointing, and recovery. The goal is to decompose these protocols into more primitive micro-protocols in order to isolate fundamental, independent constituent activities. These micro-protocols can then be used as building blocks for the construction of more efficient, application-specific protocols. For example, the recovery protocol can be decomposed into state recovery and membership view updating constituents. The latter micro-protocol is common to other protocols as well, suggesting that it may be more fundamental.

These protocols are too high level to be efficient for systems software. Lower level language constructs permitting precise and rapid control over error handling and recovery are also being investigated at the University of Arizona [Schl89]. Specifically, an event-based programming paradigm which allows explicit program-controlled response to asynchronous events, such as process failure or recovery is being developed.

Finally, the Actor model of concurrent computation [Agha90] is being studied in the context of dependability at the University of Illinois. Actors provide object-like encapsulation of data and localization of effects that can potentially form the basis of a robust programming paradigm.

3.2. Algorithm-based techniques

Unlike distributed systems where the “recoverable unit” is typically a process or service, parallel systems are characterized by finer granularity, and consequently more frequent and complex interaction of the processes. Techniques appropriate for small systems are also inefficient or impractical for massively parallel systems composed of hundreds or thousands of processors. Large scale parallel systems, however, are characterized by inherent redundancy in processing power and connectivity. This allows great flexibility in mapping applications onto the parallel hardware and graceful degradation in the event of node or link failures. Parallel applications also rarely achieve high efficiency due to the communication requirements. Even high performance systolic algorithms (characterized by highly regular, synchronous communication patterns) achieve only 50% utilization. This idle time inherent in many applications also represents a kind of temporal redundancy that can be exploited for dependability purposes.

One method of exploiting this redundancy is through algorithm "encoding" analogous to the data encoding employed in communications systems. Just as check bits (e.g., parity) can be appended to information bits to provide a low overhead mechanism for detecting or correcting errors, redundant computations can be incorporated into algorithms to check the validity of the target results. These checks, derived using the structure and semantics of the target computation, can be combined with knowledge of the architecture and mapping employed to detect and/or correct errors and diagnose node failures.

A simple example of the technique is to append an extra row and an extra column onto each data array in matrix computations. The redundant elements maintain checksums over their associated columns or rows which are invariant with respect to common matrix operations and thus allow easy detection of errors. This algorithm-based fault-tolerance...
work, first codified in [Bane84] and currently being conducted at the University of Illinois, is highly application specific and appears most appropriate for numerical and scientific computations. Some general principles have been discovered, however, and work is in progress at Princeton [Vinn90] to make the technique more automatable—perhaps through compiler generated checks.

A different—and potentially more general—technique is being investigated at Brown [Kane89] for use in shared memory architectures. This work views a process as a multi-threaded task augmented with a number of “bookkeeping” data structures. The parallel threads continuously monitor each others’ progress through these shared data structures as they cooperate on the task, with healthy threads automatically assuming the incompleted work from failed ones. This approach to robust algorithm design (in which progress occurs in spite of failures without explicit detection and recovery) has the advantage that performance is directly related to the number of failed threads; in the absence of any failures, the only overhead is the additional space for the bookkeeping information, the time to maintain it, and the time to verify successful completion of the task.

3.3. Compiler-based techniques

In addition to the work at Princeton on automatic synthesis of algorithm-based fault-tolerance, researchers at Illinois and CMU are investigating the use of compiler generated operations for error checking and recovery assistance.

The work at CMU extends previous work on instruction signature monitoring [Wik88], a technique for detecting control flow errors, by implementing the monitoring in software with compiler-generated collection and checking instructions. The work at Illinois supports rapid recovery from transient and intermittent failures through a software (as opposed to hardware) implemented retry mechanism capable of multiple instruction reissue. The mechanism relies on removal of data antidependencies to guarantee a consistent state and complete-based memory management for fast checkpointing and recovery [Li90].

4. Hardware-based approaches

Attaining a better understanding of the proper division of labor between hardware and software in dependable computing is one of the goals of this initiative. Many of the software-based techniques being supported under the initiative, for example, rely on abstractions or models of how hardware behaves under failure. The ultimate efficacy of the novel software-oriented approaches will be decided by whether these abstractions are realistic and implementable by the hardware designers, as well as whether the abstractions offer the “right” interface for the development of software techniques.

One useful abstraction is that of the fail-stop processor [Sch83]: a node whose only failure mode is a detectable crash with no corruption of external data. Implementation can be accomplished through duplication and comparison or voting approaches but in large scale systems the cost of such a strategy might be prohibitive (not to mention the decrease in raw reliability due to the more than doubling of the component count.) ONR is therefore supporting several projects to develop techniques for the design of low redundancy, self-checking microprocessors based on internal data encoding methods. These efforts are being pursued at the University of Texas at Austin, UCLA (Chau88), and the University of Massachusetts [Prad89].

Reliable communications is another useful abstraction that frees the programmer to concentrate on higher level issues of process failure and recovery. The fault-tolerance of the Chaotic Router, originally developed for high performance, deadlock free communications at the University of Washington is now being investigated [Bold90]. The router uses randomization to route messages around congested or failed nodes providing an inherent robustness that non-adaptive algorithms lack. Fault-tolerant real-time communications through dynamically established redundant circuits is also relevant to the real-time component of the initiative. Finally reliable data distribution and storage in distributed systems through a sophisticated encoding scheme called the Information Dispersal Algorithm [Rabi89] is being studied at Harvard. The scheme offers greater efficiency than replication approaches.

5. Dependable real-time computing

ONR is committed to the support of research aimed at establishing a solid scientific foundation for the design and analysis of real-time systems. Considerable progress has been made through our Foundations of Real-time Computing initiative in the areas of scheduling theory and formal methods for the specification and verification of real-time systems. It is a long-term goal of ONR to address the critical research issues that affect the integration of dependability into real-time systems. Several meetings have been held within the past two years to help define these issues and set a research agenda. Although many issues remain unclear, a part of the Ultra-dependable Multicomputers initiative is devoted to some preliminary efforts in this area.

The objectives of the principal effort (at CMU [Stro90]) are first to characterize real-time workloads and then to understand how dependability functions such as failure detection and recovery can be incorporated into schedulers and resource management algorithms to minimize the disruption of service and still satisfy timing constraints in the event of failures.

Other projects are investigating the formal specification of dependability requirements for real-time systems (University of Texas [Male90] and fault-tolerant control systems (University of Michigan [Shin90]).

6. Summary

The ONR initiative in Ultra-dependable Multicomputers and Electronic Systems is supporting innovative basic research aimed at establishing a more scientific foundation for the design and evaluation of dependable systems, in general, and real-time systems, in particular. Work is progressing on two primary fronts: experimental and modeling research to attain a better understanding of the causes and effects of physical failures in real systems with realistic workloads and software-based techniques that offer higher efficiency and flexibility by exploiting the inherent redundancy in parallel and distributed systems. Other components of the program address dependability problems best handled through hardware solutions and unique dependability issues in real-time systems that provide opportunities for innovative
solutions. While the latter area holds the most promise for avionics systems, the entire initiative will produce results, techniques, and insights pertinent to the design of all modern fault-tolerant multicomputer systems.

References


