



FIRMWARE-BASED PLATFORM RELIABILITY

**Research &
Development
at Intel**

Version 1.0
10/30/04
Padma Apparao
Greg Averill
CTG/STL/MCL

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

INTEL PRODUCTS ARE NOT INTENDED FOR USE IN MEDICAL, LIFE SAVING, LIFE SUSTAINING APPLICATIONS. INTEL MAY MAKE CHANGES TO SPECIFICATIONS AND PRODUCT DESCRIPTIONS AT ANY TIME, WITHOUT NOTICE.

Copyright © Intel Corporation 2004

* Other names and brands may be claimed as the property of others.

Introduction and Motivation

Reliability is becoming a concern in almost all aspects of computing. Today, high-end computing tasks are handled with clusters, grids and large symmetric multi-processor (SMP) systems. Critical applications, such as military, bio-medical engineering and scientific computing, running on these high performance systems demand high availability and reliability. In the future, high performance computing capabilities will move down to mainstream platforms. As the complexity of the underlying hardware increases, it is only a matter of time until reliability becomes a comparable concern even for mainstream systems. As users come to rely on mainstream systems for critical applications such as multi-modal recognition, real-time simulation or emergency response, increasing levels of reliability are required from the underlying hardware. Driven by this need, current high-end reliability features will migrate into mainstream platforms.

Improving the reliability of these mainstream systems (be they standalone or part of a large cluster/grid solution) is essential before reliability becomes a critical issue. We propose to architect new extensions to the firmware and hardware and to incorporate improved reliability mechanisms to make the Intel platform and components more reliable and increase overall system availability.

In the rest of the paper, we will first outline the trends that will make mainstream platform reliability a critical issue sooner rather than later. We will then propose a new taxonomy of system errors and show how we plan to address them in our research. Finally, we will outline several major research objectives and offer a detailed description of a prototype Fault Prediction Agent.

Why is Reliability an Issue?

There are a number of technology trends in the areas of semiconductor physics technology and high performance computing that make reliability an increasing concern.

FET densities – As silicon process continues to follow Moore’s Law, continually increasing FET densities and the reduced amount of charge held in ever-smaller storage nodes will make these nodes exponentially more vulnerable to corruption from strikes by energetic particles such as cosmic rays or alpha particles.

Narrower Burn-in Range – Burn-in is an important production test technique for screening marginal parts and reducing infant mortality. Burn-in over a wide range of temperatures is increasingly difficult or uneconomical due to increasing leakage currents.

Lower Operating Voltages – Lower voltages necessitated by thinner gate oxides, the drive for increased frequency, and increasing device variability due to less dimensional control of ever smaller devices all contribute to margin and stability issues in storage nodes.

Circuit susceptibility to transient error mechanisms is increasing with each process generation. Some are increasing at an exponential rate. Incorporating techniques into the firmware or hardware can compensate for some of the inherent unreliability of the hardware. Though the combinational logic is less susceptible to soft errors than memory cells, there will come a point at which error rates in unprotected logic will be unacceptable. The prediction is that by year 2011, the error rates in combinational logic will reach the levels at which we had to have protection in memory.¹

High Performance Computing Needs

An emerging trend in the computing industry is to move away from large servers and monolithic symmetric multiprocessors to clustered solutions. Clusters are specifically targeted at using low cost servers that can perform, scale well, and are reliable.

¹ P. Shivakumar, M. Kistler. “Modeling the effect of Technology Trends on the Soft Error Rate of Combinational Logic.” Proceedings of the International Conference on Dependable Systems and Networks, 2002.

Firmware-Based Platform Reliability

October, 2004

Table 1 shows the revenue loss per hour of system downtime in various business segments. The figures show that an hour of downtime per year is extremely expensive, regardless of industry segment. Clearly reliability is a major concern that needs to be addressed.

Industry Sector	Loss Revenue per Hour
Energy	\$2.8 million
Telecommunications	\$2.0 million
Manufacturing	\$1.6 million
Financial Institutions	\$1.4 million
Information technology	\$1.3 million
Insurance	\$1.2 million
Retail	\$1.1 million
Pharmaceuticals	\$1.0 million
Banking	\$996,000

Table 1: Revenue Loss Per Hour by Business Segment²

Availability	Yearly Downtime
95	438 hours
99.0	88 hours
99.5	44 hours
99.9	8.8 hours
99.95	4 hours
99.99	53 minutes
99.9995	5.3 minutes
99.9999	32 seconds
99.99999	3.2 seconds

Table 2: Percentage Availability and Downtime³

Availability is another aspect of fault tolerance that is of concern, as it directly translates to system downtime. Current systems are being designed for minimal downtime of 5 minutes or less per year. HP has a program known as “5 nines,” aiming for 99.999% availability, which translates to only 5 minutes downtime per year. (See Table 2.)

These reliability and availability standards cannot be met just by using reliable parts and incorporating ECC checks. We need to eliminate errors altogether and incorporate redundancy checks where errors cannot be eliminated.

Classification of Errors

The classification and definition of errors in the literature aren't particularly consistent. Terms are often overloaded or are used interchangeably with other terms. Also the classification appears to be based on the symptoms exhibited by the errors rather than the cause. There is also some attempt to classify errors based on their duration, and this combined with the other factors, has given rise to a confusing taxonomy of errors. In this document, we attempt to give a clear, meaningful definition to the various

² Source: “IT Performance Engineering & Measurement Strategies: Quantifying Performance Loss.” Meta Group, October 2000.

³ Source: “Business Critical Computing” Computer Business Review January, 1999.

kinds of errors experienced by systems today and to define taxonomy based on the cause of the errors.

Error classes can be divided into **hard** and **transient** errors. Transient errors are further divided into **event** errors and **margin** errors. (**Soft** errors, a term, which has become practically synonymous with radiation-induced errors, are a form of event errors.)

Hard Errors

Hard errors occur when a hardware circuit or component fails to function correctly. A hard failure is reproducible with the same functional stimulus and operating environment (e.g., frequency, voltage, temperature, I/O loading, etc.) under which the failure occurred. A test can be created in which the component will consistently fail. Hard errors may have always been present in the component but have remained undetected due to an insufficient production test. Such an error is known as a test escape.

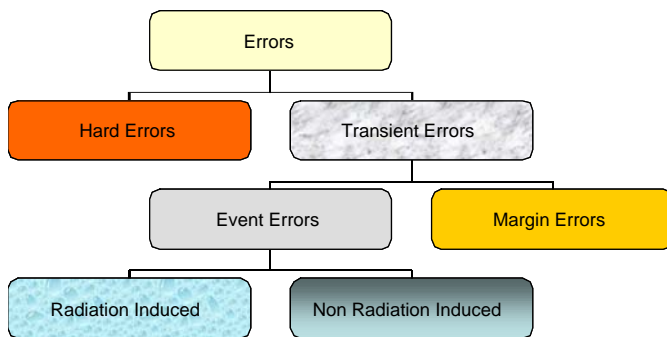


Figure 1: Classification of Errors

Hard errors may also result from damage to a part due to electrical events (e.g., static electricity or a power surge), mechanical events (e.g., shock, vibration, thermal expansion or contraction), or chemical processes (e.g., oxidation). Other causes would include degradation of a component due to processes such as the drift of silicon devices outside of their designed operating parameters or metal electro-migration.

Transient Errors

First, we define the different types of transient errors that occur and then we propose a new metric for measuring transient errors.

Types of Transient Errors

Event Errors

Event errors occur when a hardware circuit or component fails to function correctly, but the failure may not be reproducible with the same functional stimulus and the same nominal operating environment under which the failure first occurred. The error is induced by an event such as electrical noise, a power supply glitch, or an energetic particle.

Soft Errors

Errors induced by ionizing, energetic particles such as cosmic rays or alpha particles are a special class of transient event errors usually referred to as soft errors.

Margin Errors

It is not economically feasible, and it is often not possible, to exhaustively test complex VLSI components. There are some marginal circuits that pass production tests and usually function correctly when the component is within its normal operating parameters but which will malfunction under extraordinary circumstances. If a reliable test could be constructed, or if the component failed consistently, we would treat this class of error as a hard error. If not, then it is useful to treat this class of error as transient.

There are other circuits that by design have a statistically small probability of malfunctioning (e.g., moving data between two asynchronous clock domains). These we also treat as transient marginal errors.

Transient Error Detection and Correction

Transient errors may be detected or undetected. They may be corrected or not. Errors that are uncorrected are of special interest in that they directly compromise reliability. Errors that are undetected and uncorrected are known as Silent Data Corruption

Firmware-Based Platform Reliability

October, 2004

(SDC). Errors that are detected but not corrected are known as Detected Uncorrected Errors (DUE).

Error Rate Measurements

The rate at which radiation-induced errors occur is usually known as the Soft Error Rate (SER). We will call the rate at which the aggregate of transient errors occurs as the Transient Error Rate (TER).

Research Objectives

Our proposal for Autonomic Platform Reliability is a step for incorporating intelligence in the platform firmware to provide increased component reliability and improved availability of Intel platforms. We can broadly categorize our research proposal goals in several dimensions.

Fault Prediction

Error detection alone is not sufficient for increasing reliability because not all errors can be corrected. A trend in the industry now is to avoid failures if possible. We propose to incorporate heuristics in the platform firmware whereby we can predict or mitigate the hard failures of components based on the history of the soft and intermittent errors of the components and on environmental system information such as temperature, voltage, fan speeds, etc. We will discuss this further in a later section when detailing heuristics algorithms.

Autonomic Repair

In this category we will look at ways of repairing faults in a manner transparent to the operating system. The idea is that some error conditions can be recovered by the firmware. In today's systems, errors occurring in the hardware are logged to the operating system, which does little with the error information other than posting the occurrence of the event in the system log. Autonomic repair would require the development of standard error handling interfaces in the platform for communicating the results of the fault analysis to the operating system, service processor and the hardware. Another advantage of putting autonomics in the platform is that the system can watch for signs of impending trouble and call for

preemptive service such as the swapping in of spare components.

High-end RAS Capability Migration

The large mainframe systems of today have several top-of-the-line RAS characteristics included in their hardware to provide high reliability. For example IBM has exploited most of the disk RAS techniques into their memory architecture, and HP has a mechanism called RAIM (redundant array of inexpensive memory) for their memory technology. These features not only increase the amount of hardware but also add more cost to the system. The return on investment for these features may be beneficial for high-end systems, but for low-end commodity systems a trade-off must be made balancing the RAS characteristics and expense involved with the potential risk or gain.

New Paradigm RAS Research

Several new paradigms are being exploited in the computing space, such as multiple cores on a die, tiny cores and link interconnects. As the number of cores increases and their size decreases, a more focused effort is required to look at reliability of the individual cores and how multiple cores on a die could be exploited to increase system availability and reliability. These new microarchitectures will necessitate new interconnect technologies, and one could enhance these to improve system reliability and availability.

Platform-Directed Provisioning

The ultimate goal of this research is to assist in dynamic job reconfiguration based on feedback provided by the platform. Dynamic provisioning or job reconfiguration is the deployment of jobs or migration of existing jobs in a cluster or a grid for purposes such as load balancing, meeting SLAs (Service Level Agreements) or underlying hardware reliability. The focus of this research is to enable job reconfiguration based on platform feedback. For example, in the event of notification by the platform of inevitable failure of a node within a certain timeframe, the applications can choose to run in degraded mode until graceful shutdown is possible,

or the cluster manager can decide whether to migrate the applications to healthier nodes or to run the applications in degraded mode without repair or migration.

Platform feedback is also essential to drive checkpoint/restart application architectures, where the application can use knowledge of the node reliability to adjust checkpoint intervals so as to optimize performance.

Our research also includes investigation into mechanisms for translating prediction results into platform notifications to the OS/cluster manager of potential node failures.

Fault Prediction Agent (FPA)

The Fault Prediction Agent prototype is being developed as a proof-of-concept for APR.

The FPA is comprised of three main parts: the Fault Detector, the Fault Predictor and a Fault Tolerant Platform Manager. This architecture of the FPA [Figure 2] is very flexible. The various components of the FPA can reside on different machines. For example, the Fault Detector could be resident on the machine experiencing the errors, with the Fault Predictor and the Fault Tolerant Platform Manager resident on other machines. The error database could also be located on an independent machine. The flexibility does not end here. All of these components can reside either within the application layer (user space) or the OS or within the firmware, and they do not all have to be collocated in the same layer.

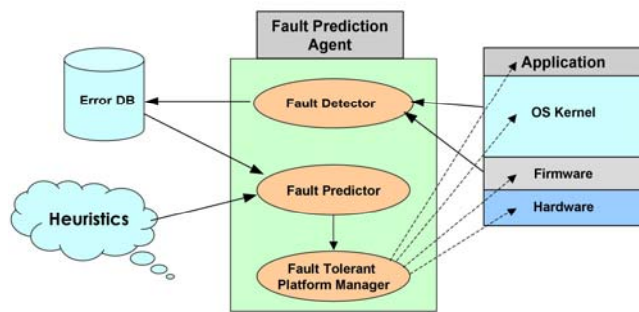


Figure 2: Fault Prediction Agent

The Fault Detector

The Fault Detector is an error monitor that extracts error information from the hardware/firmware and records the error in a database. In the field, errors occur due to radiation or due to marginal design that makes the components more vulnerable to power surges, cosmic rays, etc. In our prototype, we have used an In-Target Probe (ITP) to inject errors into the system components. Another option is to get marginal parts from manufacturing that would generate faults in the presence of certain external conditions. When an error occurs and is corrected by the hardware, the error is recorded in the firmware. The Fault Detector makes a call to the firmware to read the error record and buffers, ten error records at a time, before putting them in a database. We also have developed a web interface that displays the errors as they are updated in the database.

The Fault Predictor

The Fault Predictor analyzes the error records in the database and applies some heuristics to predict the reliability of the components based on error history and correlation of the component's errors with other conditions within the platform. For example, a simple heuristic could be to mark a component unreliable if it experiences n number of errors within some time interval, t . Other more intelligent heuristics could correlate errors from various components or factors and predict failures. For example, if temperature exceeds a certain threshold and the voltage exceeds a certain threshold then component x is most likely to fail. Another example might be that the Predictor looks at the failures of several components and predicts that if component a has experienced several errors and so has component b , then component d is likely to be affected and fail hard. These heuristics will require that the Predictor have enough information in the error records to actually predict faulty components or the unreliability of the components.

The Fault Tolerant Platform Manager

The Fault Tolerant Platform Manager makes decisions based on the Fault Predictor's analysis and communicates the decision to the OS, firmware or

Firmware-Based Platform Reliability

October, 2004

hardware to either de-allocate the component or bring down the entire platform. If the applications provide interfaces, then the FTPM can take its decision directly to the application, giving warning about faulty components and enabling application migration or graceful shutdown of applications. One of the decisions the FTPM needs to make is to check whether the unreliable component can work in a degraded mode until repair/replacement, or if it is possible to reconfigure the hardware transparent to the OS until the faulty component can be replaced. If the above two options are feasible, it gives the OS time to do process migration or let the applications run on slightly underperforming hardware until graceful system shutdown is possible. Operating system API support is needed in order to react to the agent's warnings.

Value of Firmware-based Platform Reliability

With high performance computing clusters being built from thousands of commodity systems, and with traditionally high performance computing workloads being targeted at commodity platforms, reliability is not to be taken for granted. Developing firmware-based fault prediction and autonomic reliability features will help ensure the usability and availability of future platforms for critical applications and decrease the probability of data corruption in the event of hard component failures. These features will bring added value to mainstream platforms, lower system management costs, and help to overcome the reliability challenges introduced by system architecture and semiconductor physics trends.

References

M.Abramovici, M. A. Breuer, A D Friedman. Digital System Testing & Testable Design. Computer Press, 1990.

T. Lin, D. Siewiorek. "Error Log Analysis: Statistical Trends and Heuristics Modeling Analysis." IEEE Transactions on Computers, Vol. 39, 1990.

D. Siewiorek, R. Swarz. Reliable Computer Systems: Design and Evaluation. Digital Press, 1992.

P. Shivakumar, M. Kistler. "Modeling the effect of Technology Trends on the Soft Error Rate of Combinational Logic." Proceedings of the International Conference on Dependable Systems and Networks, 2002.

[Web Services Extend High-Performance Computing Grid Capabilities](#) by Matt Gillespie

[A Framework for Platform-Based Dynamic Resource Provisioning](#) by Dean Yao, Tisson Mathew, Mazin Yousif, Sharad Garg