

# A Dependability Architecture Framework for Remote Exploration & Experimentation Computers

Jaynarayan H. Lala  
Draper Laboratory  
lala@draper.com

J. Terry Sims  
Draper Laboratory  
tsims@draper.com

## 1. Problem Statement

The NASA Remote Exploration and Experimentation (REE) Project, being managed by the Jet Propulsion Laboratory, has the vision of bringing commercial supercomputing technology into space, in a form which meets the demanding environmental requirements, to enable a new class of science investigation and discovery [1]. A principal REE goal is to demonstrate a high throughput to power ratio of about 300 MOPS/watt. Dependability goals include 99% reliability over 5 years, 99% availability, and a worst case outage of no more than 10 min/day. Additionally, system must be scalable from 1 to 50 nodes, maximize use of COTS hardware and system software, and be easily adaptable to soft real-time applications of various NASA science missions. In response to these challenging requirements, Draper Laboratory, in cooperation with JPL, developed a dependability design approach and synthesized a REE computer system architecture. (This work was performed under contract 961564 from JPL.) This abstract provides a brief architecture overview and a summary of its salient features.

## 2. Architecture Overview

The REE SIFT (Software Implemented Fault Tolerance) architecture provides the general software structure in which to implement the various fault tolerance techniques chosen for a specific instantiation of the REE computer concept. This architecture, illustrated in Figure 1, recognizes four major compositional elements: REE system management tasks, application software components, REE hardware elements, and SIFT software components. The SIFT software components are positioned to monitor

and control the application software and the hardware resources as required to meet the overall computer system's goals and policies established by the higher level REE system management tasks.

The REE system management architectural component has three principal responsibilities. First, it is responsible for establishing and maintaining the current system goals (i.e., What is the REE computer system supposed to be doing?) and policies (e.g., What are the dependability requirements for the various system goals? What are the current power conservation requirements for the system?). Second, it is responsible for making decisions regarding the optimal use of the available hardware resources and the application programs to meet the system's goals. And third, it is responsible for determining the "available for use" status of the various hardware components in the system based upon initial system configuration, power control decisions, and fault detection/isolation events.

The REE architecture is designed to support application software components that are active fault tolerance participants in terms of providing application-specific fault detection and recovery capabilities. The architecture is predicated on the ability to provide adequate application execution dependability with simplex application-specific techniques (e.g., Algorithm-Based Fault Tolerance techniques, value and time domain fault detection techniques, and checkpoint/rollback

mechanisms) without resorting to the complexity and resource costs associated with redundant execution of application software.

The Application-Specific FT Management SIFT component shown on the left in Figure 1, is responsible for monitoring the execution of the application software, for determining when

the application software execution has been disrupted by hardware faults, and for executing recovery responses. A

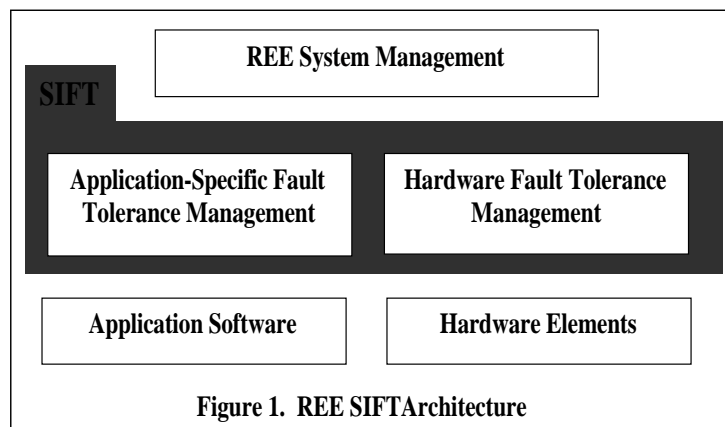


Figure 1. REE SIFT Architecture

single application will most likely be distributed across multiple nodes to achieve the parallel processing performance expected of the REE system. Thus, the SIFT management capability will involve fault detection, isolation, and response activities both at the level of individual processing nodes (e.g., making local corrective responses when possible, for communicating fault status, and responding to received control commands which affect the execution control of the node-level application software elements), and at the level of a complete distributed application (controlling and monitoring the execution of the various node-level managers, supporting “global” fault detection, isolation, and response functionality involving coordination among an ensemble of individual application programs running on multiple nodes).

The hardware architectural component will consist of multiple separate processing nodes and hardware components to support a communication interconnect between the processing nodes. The architecture will support fault tolerant techniques based upon fault detection and recovery mechanisms available in these hardware resources (e.g., error detection and correction memory devices, microprocessor exception detection mechanisms, communication error detection mechanisms).

The “hardware-centric” SIFT components represented on the right hand side of Figure 1 support the REE System Manager’s objective of continually determining the “available for use” status of the various hardware components in the system. The SIFT hardware fault detection and isolation capabilities produce fault status information used in the determination of this availability status. These SIFT management components are also responsible for attempting to recover hardware resources in response to transient faults. As with the “application-centric” SIFT components, there is a hierarchical organization to the “hardware-centric” components. A Global manager is responsible for integrating fault status from all of the individual processing node and network element managers to determine a final fault status and for coordinating hardware element recovery attempts.

Furthermore, these SIFT components provide the software functionality to create and maintain a highly reliable, Triple Modular Redundant (TMR) “Kernel” operating environment for the execution of the REE System Management functions and the more system-critical SIFT management components themselves. The TMR capabilities provide a fault masking, high fault coverage operating environment for these “Kernel” functions by making use of a small subset of the multitude of processing nodes in the REE computer system and the REE’s high performance interconnect. To tolerate permanent hardware faults, Dynamic Hardware Redundancy concept is employed. When a Kernel node suffers a permanent failure, a new node is designated to

take its place. A replica of the system tasks is created on the new node and the task states are made congruent with the other two operational replicas. This process of replacing a failed node with another node is performed without restarting the system tasks.

The traditional approach to a fault-masking execution of the Kernel tasks would require dedicating at least three nodes to these functions. That would result in a 200% overhead in throughput and power. Instead, a novel concept, called the Dynamic Task Redundancy (DTR), allows us to keep the overhead rather small while still providing a very high degree of dependability.

The DTR technique decouples the application software redundancy from the underlying hardware redundancy. A node that is a member of the Kernel Triple Modular Redundant (TMR) set, can execute some applications that are simplex while also being time-shared to run a replica of the Kernel software. The three Kernel nodes would nominally be executing different node application programs and occasionally, on a periodic basis, would be synchronized to execute redundant replicas of system tasks.

### 3. Salient Features

The proposed dependability architecture combines proven fault tolerance techniques with many innovative ideas to meet the unique REE requirements. A reliable Kernel that masks SEU effects performs the critical resource management functions. Fault tolerance mechanisms are spread hierarchically at different levels of hardware, software and applications. This ensemble can be tailored for different missions and for mission phases. Dynamic task redundancy provides low overhead implementation of high dependability for critical functions without forcing an equal overhead on other functions. A node may execute a set of applications, each of which has a different redundancy level. The REE dependability architecture does not require specialized fault tolerance features in hardware platform or the operating system beyond what is generally available in commercial high performance products. Other salient features include a software-implemented fault tolerant middleware, transparent fault masking execution of critical functions, and theoretically proven concepts and ease of validation of fault tolerance mechanisms. Although designed for soft real-time science applications, the architecture is also easily extensible to spacecraft control and telescope adaptive optics control.

#### Reference

- [1] Rohr, John, “Software-Implemented Fault Tolerance for Supercomputing in Space,” Fast Abstract, 1998 FTCS, Munich, Germany.