A Fault-Tolerant Supercomputer to Enable a New Class of Scientific Investigation and Discovery in Space
DRAFT

Daniel S. Katz
Jet Propulsion Laboratory
Daniel.S.Katz@jpl.nasa.gov

Thomas I. McVittie
Jet Propulsion Laboratory
Thomas.I.Mcvittie@jpl.nasa.gov

Alfred G. Silliman, Jr.
Jet Propulsion Laboratoy
Alfred.G.Silliman@jpl.nasa.gov

1. Introduction

The goal of the Remote Exploration and Experimentation (REE) Project [1] is to move supercomputing into space in a cost effective manner and to allow the use of inexpensive, state of the art, commercial-off-the-shelf (COTS) components and subsystems in these space-based supercomputers. The motivation for the project is the lack of bandwidth and long round trip communication delays which severely constrain current space science missions. By moving the processing capability from the ground to the spacecraft, REE will enable a new class of scientific investigation and discovery in space.

Unlike typical radiation-hardened space-based systems, the use of COTS hardware will require the REE system to withstand relatively high rates of single event upset (SEU) induced errors. Depending on mission environments and component technologies, an REE system will be required to withstand average fault rates of between 1 and 100 SEU-induced soft errors per CPU-MB-day with occasional peaks of up to 1000 soft errors per CPU-MB-day[2]. Unlike traditional fault tolerant computer systems, however, the REE computer need not provide 100% reliability, but is instead, as with many sampled data or convergent-computation systems, allowed to occasionally fail in a computation. Periodic resets to flush latent errors, and other techniques which provide less than 100% availability, are also permissible. Further, the REE computer need not support hard real time or mission critical computation, as these tasks can be off-loaded to the spacecraft control computer. The flexibility afforded by the above requirements allows the system to be optimized for high-performance, low-power, supercomputing rather than for "hard" fault tolerance.

2. System description

The REE system is composed of three major components; commercial hardware, application programs, and a middleware layer of software-implemented fault-tolerance (SIFT). As new and more capable hardware becomes available commercially it will be incorporated into REE. Applications will vary from mission to mission depending on the science nature of the individual mission. The SIFT layer will utilize a combination of fault tolerant techniques to achieve the reliability and availability requirements and may be configurable, depending on the needs of individual users. This layer is intended to be portable as hardware and applications change.

Hardware

The REE computer architecture is a Beowulf-type[1] [3] parallel processing supercomputer comprising a multiplicity of processing nodes interconnected by a high speed, multiply redundant communication fabric. In the current instantiation of the system, dual Power PC 750 based computational nodes containing 128MB of main memory and dual redundant Myrinet [4] interfaces are interconnected via a redundant Myrinet fabric. The node level operating system is Lynx [5] Operating System (OS), to which multiple versions of MPI [6] have been ported. The system is to be extensible to at least 50 nodes with a power:performance of better than 300 MOPS/Watt.

Applications

The applications are written so that they may be automatically configured to execute on up to 50 processors with the system being informed, by the application, of the optimal number of processors for maximum throughput and the system assigning the number of processors available based on system status and operational constraints such as available power, spares availability and mission phase. There are currently 5 science teams writing applications for potential future NASA missions which may incorporate the REE computer. To aid application developers, a library of fault-detection-enabled scientific subroutines for linear algebra and Fast-Fourier Transform (FFT) routines has been developed. Work is ongoing to determine the utility of an error-correction-enabled library. In addition,

[1] Beowulf-class computers were originally defined as parallel clusters of commodity hardware and open-source operating systems and tools. This definition has grown to include most clusters composed of personal computer central processing units (CPUs) and commodity operating systems and tools.

continued analysis of application fault tolerance requirements and determination of the applications' native error tolerance is ongoing, as is the development of a generalized taxonomy of scientific software structure and the applicability (and overhead costs) of various software-implemented fault-tolerance (SIFT) mechanisms to these constructs.

SIFT Layer

The REE project requires that dependability be provided through software. Extensive error detection and recovery services are provided to the target applications through a variety of mechanisms including a global resource manager, an applications manager, and ABFT-enabled scientific subroutine libraries.

The global resource manager performs the executive functions of the supercomputer including control and scheduling of applications. It must always be available to communicate and coordinate REE activities with the spacecraft control computer. The global resource manager uses the Configurable Fault Model designed and implemented by WW Technology Group and General Dynamics. It uses synchronization, data replication and validation techniques to supply a high dependability cluster.

REE currently uses the Chameleon application manager written by Prof. Ravi Iyer et. al at the University of Illinois [7] [8]. Once the application starts successfully, Chameleon ensures that it continues running until it has completed. The application has been slightly modified to make heartbeat calls to the Chameleon ARMORs, and Chameleon is aware of how often these heartbeats should occur. If one fails to occur within the response window, Chameleon assumes that the application has hung, or is stuck in a loop, and restarts it.

The second level of fault-protection is inside the application, though the application code itself is not modified. Instead, an ABFT [10] version of the library routine is used. The ABFT versions have the same calling sequences as the basic routines, but they check to see if the routine was completed successfully before returning. If the routine was not successful, they retry once. If this retry is also unsuccessful, the ABFT version calls exit, which essentially promotes the problem to Chameleon to deal with by restarting the application.

4. Conclusions

An initial REE system has been assembled at JPL. It consists of commercial Power PC processors running in a VME chassis, a combination of several fault tolerance techniques merged into the SIFT layer, and a Mars Rover geology application program [9]. Using a variety of fault injectors, this system is being characterized to understand its performance capabilities. Early results indicate that the SIFT can be effective in correcting SEU errors. Over the next 18 months, the REE project will continue the development of SIFT approaches for space-based parallel COTS supercomputing. The Project will culminate in the development of a flight-capable prototype hardware/software system during the 2003-2004 time frame.

5. Acknowledgements

The work described in this publication was carried out at the Jet Propulsion Laboratory (JPL), California Institute of Technology under a contract with the National Aeronautics and Space Administration (NASA).

6. References

[1] R. Ferraro, "NASA Remote Exploration and Experimentation Project," http://www-ree.jpl.nasa.gov.

[2] R. Ferraro, R. R. Some, J. Beahan, A. Johnston, and D S. Katz, "Detailed Radiation Fault Modeling of the REE First Generation Testbed Architecture," Proceedings of *2000 IEEE Aerospace Conference.*

[3] T. Sterling, J. Salmon, D. Becker, D. Savarese, *How to Build a Beowulf*, The MIT Press, 1999.

[4] Myrinet is a class of products of Myricom, Inc. (http://www.myricom.com/).

[5] Lynx OS is a product of Lynx Real Time Systems, Inc. (http://www.lynx.com/)

[6] M. Snir, S. W. Otto, S. Huss-Lederman, D. W. Walker, J. Dongarra, *MPI: The Complete Reference*, The MIT Press, 1996.

[7] S. Bagchi, B. Srinivasan, K. Whisnant, Z. Kalbarczyk, R. Iyer, "Hierarchical Error Detection in a Software Implemented Fault Tolerance (SIFT) Environment," *IEEE Transactions on Knowledge and Data Engineering*, March 2000.

[8] Z. Kalbarczyk, S. Bagchi, K. Whisnant, R. Iyer, "Chameleon: A Software Infrastructure for Adaptive Fault Tolerance," *IEEE Transactions on Parallel and Distributed Computing*, June 1999.

[9] R. Castaño, T. Mann and E. Mjolsness, "Texture Analysis for Mars Rover Images," *Applications of Digital Image Processing XXII*, Proc. of SPIE, Vol. 3808, Denver, July, 1999.

[10] M. Turmon, R. Granat, "Algorithm-Based Fault Tolerance for Spaceborne Computing: Basis and Implementations," Proceedings of *2000 IEEE Aerospace Conference.*

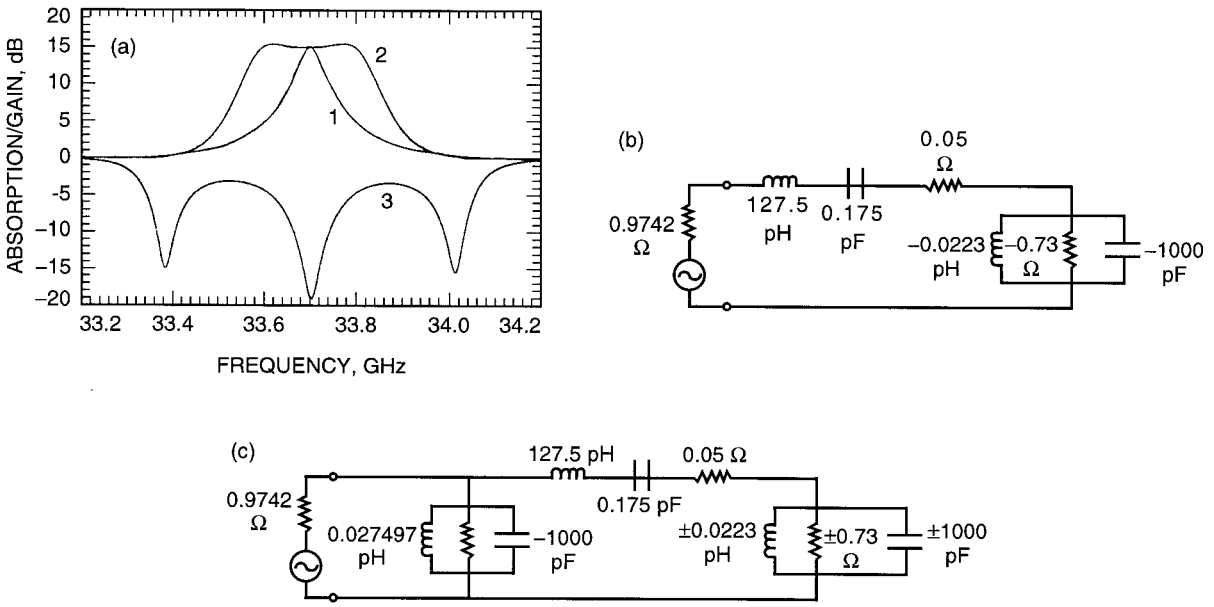

Fig. 2. Calculated gain curves (using MMICAD) of lumped-element circuits representing (a) a cavity containing a spin system (curve 1), that same cavity spin system broadbanded with an additional cavity and the spin sytem inverted (curve 2), and the spin system absorbing (curve 3), (b) the equivalent circuit for [curve x], and (c) the equivalent circuit for [curve y].