

Building and Verifying Fault-Tolerant Autonomous Real-Time Systems for Space Applications

Albert M. K. Cheng

Abstract—NASA missions require autonomous systems that perform correctly for an extended period of time. These systems must make real-time decisions in logical sequence that meet timing requirements. These systems must anticipate faults induced by environmental change, but it is difficult to anticipate the infinite variety of situations one must encounter for the design of robotic explorers, spacecraft, and systems necessary for the management of a mission. An Automated Reasoning program seeks to establish tolerances for change or alteration and seeks computational formulas for verifying scalable fault tolerances.

AUTONOMOUS SYSTEMS AND their embedded autonomy software in many NASA missions must perform correctly for an extended period of time and make real-time decisions that meet both logical and timing requirements. Furthermore, autonomy software must tolerate implementation and environment-induced faults. Developing and verifying these systems are especially difficult because of the large (and often infinite) state space and execution sequences as well as the uncertainties in the environment in which these systems operate. One focus of NASA's Automated Reasoning thrust is to "enhance the autonomous decision-making capabilities of robotic explorers, spacecraft, and mission management systems." The objectives of this project are to address two key technology areas in the Automated Reasoning category: (1) adding scalable fault-tolerance in the decision-making autonomy software of a real-time autonomous system and (2) augmenting the capability of formal verification strategies by providing an alternative based on scalable rulebase analysis to model checking and theorem proving.

To evaluate our proposed fault-tolerance strategies under a variety of fault models, we needed a simulation platform. Therefore, we first studied a distributed real-time fault-tolerance Web monitoring system. The method of providing fault-tolerance is to schedule multiple copies of a task on different computer nodes in a distributed computing system. A fault-tolerant system automatically recovers from a specified number of failures. If the primary task cannot be completed because of a fault, the scheduled backup task is run and all tasks are assured to be completed.



Dr. Albert M. K. Cheng

We use the Web to monitor the fault-tolerant behavior of a distributed system. A Web monitoring system is a convenient way to monitor remote tasks, both primary and backup. Our simulation results show that it is possible to set up a distributed real-time fault-tolerance Web monitoring system. To achieve our goal quickly, we used existing Ganglia network and RRDTool technology. We found that we can use the Ganglia system to set up a distributed real-time fault-tolerance Web monitoring system. The RRDTool is employed only for simulation purposes. It is used to show the results of the simulation graphically, but it cannot accurately store the status data of the tasks. We use MySQL to store the status of real time tasks instead of the RRDTool in further research and to yield more accurate results within time constraints.

We focused on the simulation of a distributed real-time fault-tolerance Web monitoring system. It is involved in the scheduling of real-time tasks in a distributed computing system. The real-time system is based on multiple processors, which are distributed and connected by a local area network; one way of providing fault-tolerance is to schedule multiple copies of a task on different computer nodes in the distributed system. A fault-tolerant system automatically recovers from a specified number of failures. The two tasks, both primary and backup, are scheduled to start at the same time and execute concurrently on the two computers in the distributed system. If the primary task cannot be completed because of a fault, the scheduled backup task is run so that all tasks can be completed. To maintain the consistency of the two tasks, we implemented the two tasks in software mode. However, in our simulation, we input value 1 and value 2 to distinguish the primary task and backup task from the two different computers. We utilized the Web to monitor fault-tolerant behavior in the distributed system. Users always see fresh and consistent data for primary tasks (value 1) or backup tasks (value 2). A real-time system is a system in which data on

the Web should be maintained fresh and tasks completed within specified deadlines.

In this project, a real-time system is based on a distributed computing system through the internet. A Web monitoring system is a convenient way to monitor remote tasks, both primary and backup. We assume two approaches. One approach is the primary task failure; for example, some tasks communicate with a hardware device. If the hardware device fails, then we call it a task failure, and it is not easy to recover. In this way, the primary task must be permanently replaced by a backup task. According to our study and simulation, if a primary task fails, its duties will be assigned to a backup task in another computer node. Another approach is the case in which primary task failure is due to a processor failure. We can use another means for monitoring the backup processor through the backup Website, and guarantee the completion of all tasks. We add fault injection programs to test the fault-tolerance and robustness of the system. Our system simulates the occurrence of faults and the switching of the replica task in the distributed computing node. All activities are monitored through a Web report.

The report presented here focuses on the visualization of the real-time fault-tolerance system information on the Web. Our goals are to study and simulate the following: (1) real-time response, (2) the distributed system, (3) fault-tolerance, and (4) Web monitoring. To achieve our goals quickly, we chose to use the existing Ganglia network and RRDTool technology.

Ganglia System for Distributed Real-Time Fault Tolerance Systems

A lot of existing network software targets distribution systems. A typical well-performing system for monitoring computer nodes information is the Ganglia distributed monitoring system,^{6,7} which is an open-source system developed by a research group at UC Berkeley. Ganglia is a scalable distributed monitoring system for high-performance computing systems such as clusters and grids. It is based on a hierarchical design targeted at federations of clusters. It relies on a multicast-based listen/announce protocol to monitor the state within clusters and uses a tree of point-to-point connections amongst representative cluster nodes to federate clusters and aggregate their state.

It leverages widely used technologies. The implementation is robust and has been ported to an extensive set of operating systems and processor architectures. Because of its power, the Ganglia distributed monitoring system becomes the first selection for the Distributed Real-Time Fault-Tolerance System in this project. We have modified its programs to fit our system and set up its configuration.

RRDTool for the Simulation and Visualization

The RRDTool is for data storage and visualization.^{6,7} It uses carefully engineered data structures and algorithms to achieve very low per-node overheads and high concurrency. We use it to simulate the faults occurring in the tasks and the replacement of the backup tasks from the backup system because of its efficiency in using hardware space and quickly generating graphical representations.

Methodology

We first defined the following system model:

1. The Linux Fedora Core 3 platform and the Ganglia system are installed into the computers, both primary and backup computers, which are connected through the Internet.
2. Tasks are aperiodic.
3. Each task has two versions, a primary copy and a backup copy. If the primary fails, its backup always succeeds.
4. Tasks are non-preemptable; when a task starts execution on a processor, it runs until its completion if the task does not fail.
5. Tasks are parallelizable, which means that a task can be executed on both the primary processor and the backup processor.
6. When the fault injection program aborts one of the tasks, the fault-detection mechanism detects task faults and then the scheduler will schedule the backup task.
7. When a fault occurs, extra time is required during task execution to handle fault detection and to schedule the backup task or to switch from the backup task to the recovered primary task.

We assume that there are many tasks; each task can be run on both the primary and the backup processors in a distributed system. We also assume that the primary tasks are independent and run on the uniprocessor in a FIFO order. We concentrate only on the fault-tolerant scheduling of the non-preemptive tasks in the real-time system since faults need to be identified before scheduling the backup tasks on another computer node in the same distributed system.

Tasks scheduled on this system are guaranteed to complete themselves if a task fails at any instant of time. We address the fault-tolerant scheduling problem by using a primary/backup approach. When a task arrives into the system, two copies, a primary and a backup, are scheduled on two different processors within the task's window. The backup copy of a task executes only if the execution of the primary copy fault is detected and the backup is activated. Since we assume a dynamic system, it is possible to release resources reserved for backup copies of the tasks as soon as the primary copies finish executing. In this manner, we are able to better estimate the system's available free time when new tasks are scheduled.

The fault injection program provides test cases for our system software. These test cases are used to evaluate the performance or behavior of the newly designed software with the desired targets. Input faults to the system are in the form of a PHP script which implements Java programs, which simulate the tasks in the local computer. One program contains sequence representation of errors which abort the regularly running tasks. The fault injection is used to test the fault-tolerance and robustness of the system. It simulates the occurrence of faults and the switching of the replica's task in the distributed computing node. It also switches back to the original tasks if the original tasks have been repaired.

Our monitoring system cannot be implemented without releasing the extra security layer with LINUX (Fedora Core 3). The first reason is that Selinux is set on by default. Selinux is an extra security layer. One can turn it off in /etc/selinux/config by setting SELINUX to "disabled" and rebooting. The second reason is that our PHP programs call "exec" which invokes the

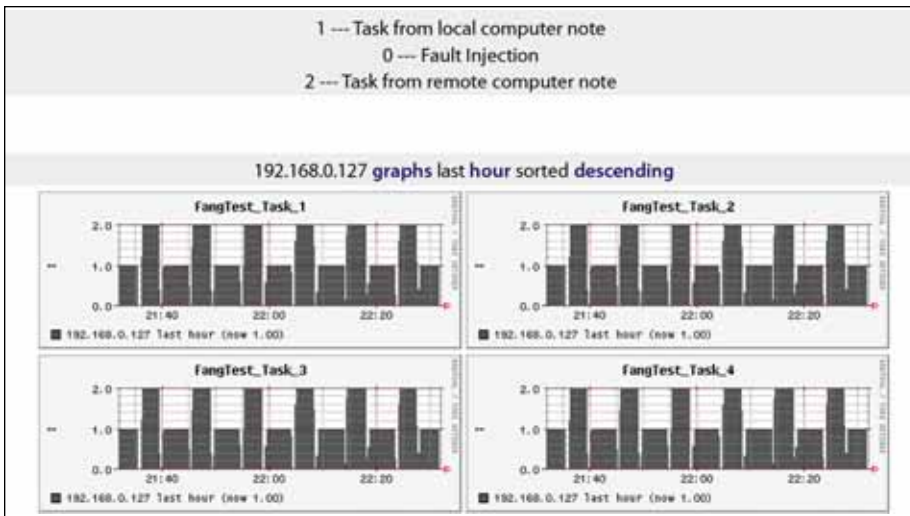


Figure 2. This is our simulation result. The result shows that the space interval between the primary task and the replaced backup task is different. The result also shows that the data at interval are not the same data used as input to the database.

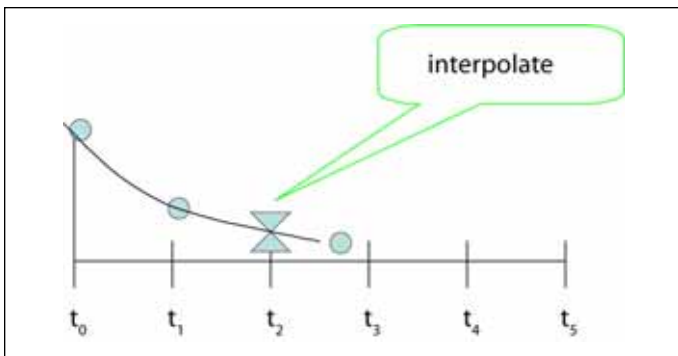


Figure 3. Interpolation of random time data into fixed time steps; Ganglia pull the data at fixed intervals.

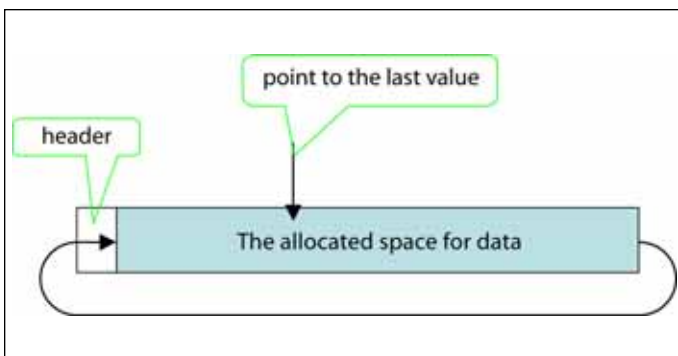


Figure 4. Space allocated for RRD consolidated data. The header holds the pointer information which points to the last value.

the interval at which this consolidation should occur and which consolidation functions (average, minimum, maximum, last) should be used to build the consolidated values. One can define any number of different consolidation setups within one RRD.

They will all be maintained on the fly when new data are loaded into the RRD.

Round Robin Archives

The Round Robin Archive (RRA) is very efficient for storing data for a certain amount of time while using a known amount of storage space, but old data are automatically eliminated over time. Data values of the same consolidation setup are stored into the Round Robin Archive (RRA). The use of RRA guarantees that the RRD does not grow over time and that old data are automatically eliminated. By using the consolidation feature, one can still keep the data for a very long time, while gradually reducing the resolution of the data along the time axis. Using different consolidation functions allows one to store exactly the type of information that is actually of interest.

Approximately one day later, the data may become less accurate because of the consolidation stated above; this is a negative side of the RRD. Refer to Fig. 4.

Discussion

The Ganglia system and the RRDTOOL provide information that can help to better understand the fault-tolerance of a distributed system, and both provide real-time monitoring which makes it possible to monitor and respond to potential faults and to schedule backup tasks. Simulation results give us further direction for implementing the distributed real-time fault-tolerance system and for using the Web to monitor real tasks. We will use MySQL to store the real status of real-time tasks instead of the RRDTOOL to yield more accurate results within time constraints.

Conclusions

Since our approach can transform subsets of the code into self-stabilizing equivalents with different code modifications depending on their syntactic/semantic forms, it is scalable to increasingly large and complex autonomy software systems. Coupled with the compositional analysis/verification strategy that identifies these syntactic/semantic code subsets (described in the second half of the proposal), our overall approach further scales to deriving and verifying highly fault-tolerant autonomy software systems. Ongoing work evaluates this strategy on the modified Ganglia simulation platform.

References

- ¹S. Balaji, L. Jenkins, L. M. Patnaik, and P. S. Goel, "Workload Redistribution for Fault Tolerance in a Hard Real-Time Distributed Computing System," *IEEE Fault Tolerance Computing Symposium* (1989): 366-73.
- ²P. M. Melliar-Smith and L. E Moser, "Progress in Real-Time Fault Tolerance; Reliable Distributed Systems," *Proc., 23rd IEEE Intl. Symposium* (2004): 109-11.
- ³I. Gupta, G. Manimaran, and C. Siva Ram Murthy, "Primary-

Backup Based Fault-Tolerant Dynamic Scheduling of Object-Based Tasks in Multiprocessor Real-Time Systems,” in *Dependable Network Computing*. Ed. D. R. Avresky. Boston: Kluwer Academic Pub., 1999.

⁴I. Gupta, G. Manimaran, C. Siva Ram Murthy, “Fault-Tolerant Dynamic Scheduling of Object-Based Tasks in Multiprocessor Real-Time Systems,” Annual IEEE Workshop on Fault-Tolerant Parallel and Distributed Systems (1999): 83-107.

⁵G. Manimaran and C. Siva Ram Murthy, “A Fault-Tolerant Dynamic Scheduling Algorithm for Real-Time Multiprocessor Systems and its Analysis,” *IEEE Trans. Parallel and Distributed Systems* 9.11 (1998): 1137-52.

⁶M. L. Massie., B. N. Chun, and D. E. Culler, “The Ganglia Distributed Monitoring System: Design, Implementation, and Experience,” *Parallel Computing* 30 (2004): 817-40. (*Pending*.)

⁷Ganglia Monitoring System, Nov. 8, 2005 <<http://ganglia.sourceforge.net>>.

⁸F. D. Sacerdoti, M. J. Katz, M. L. Massie, and D. E. Culler, “Wide Area Cluster Monitoring with Ganglia,” San Diego Supercomputing Center and Univ. of California at Berkeley, 2004 <<http://ganglia.sourceforge.net/papers/Sacerdoti03Monitoring.pdf#search='ganglia%20massie'>>

Publications

Andrei, S., W.-N. Chin, A. M. K. Cheng, and M. Lupu. “Automatic Debugging of Real-Time Systems Based on Incremental Satisfiability Counting,” *IEEE Trans. on Computers*, 2006. (*To appear*.)

Andrei, S., W.-N. Chin, A. M. K. Cheng, and M. Lupu. “Incremental Automatic Debugging of Real-Time Systems Based on Satisfiability Counting,” IEEE-CS Real-Time and Embedded Technology and Applications Symposium, San Francisco, CA, March 2005.

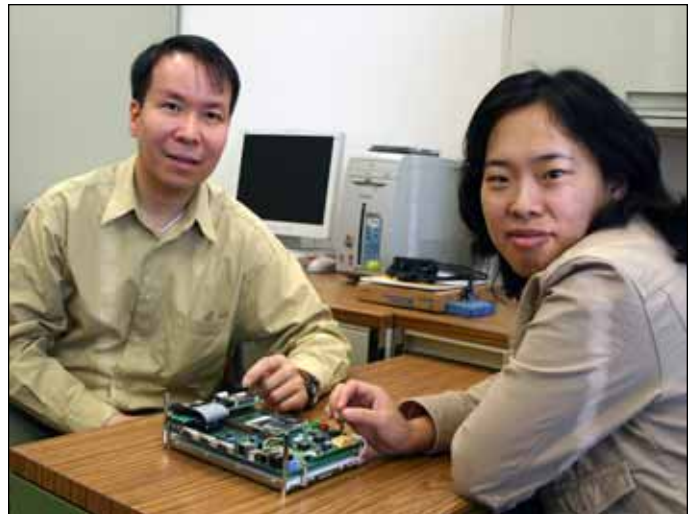
Andrei, S., W.-N. Chin, A. M. K. Cheng, and Y. Zhu. “Runtime-Coordinated Scalable Incremental Checksum Testing of Combinational Circuits based on #SAT Problem,” *Proc.*, 11th IEEE-CS Intl. Conf. on Embedded and Real-Time Computing Systems and Applications, Hong Kong, Aug. 17–19, 2005.

Cheng, A. M. K. “A Survey of Formal Verification Methods and Tools for Embedded and Real-Time Systems,” *Intl. J. Embedded Systems* 1 (2006).

Cheng, A. M. K. and F. Shang. “Priority-Driven Coding of Progressive JPEG Images for Transmission in Real-Time Applications,” *Proc.*, 11th IEEE-CS Intl. Conf. on Embedded and Real-Time Computing Systems and Applications, Hong Kong, Aug. 17–19, 2005.

Cheng, A. M. K. and S. Fang. “Study and Simulation of a Distributed Real-Time Fault-Tolerance Web Monitoring System,” *Proc.*, IEEE-CS Real-Time Systems Symposium WIP Session, Miami, FL, Dec. 5–8, 2005.

Zhang, W., A. M. K. Cheng, B. Fang, and M. Hu. “An Adaptive Multisite Scheduling Algorithm for Parallel Jobs in Computational Grid Environments,” *Proc.*, 3rd High-Performance Grid Computing Workshop, Intl. Parallel and Distributed Processing Symposium, Rhodes Island, Greece, April 29, 2006.



INTEL X-SCALE PXA255—Dr. Albert M. K. Cheng (l.) reviews with Bin Lu (r.) an embedded development platform he has researched to test embedded real-time applications. Lu earned her B.S. in electrical engineering at the Shanghai Jiao Tong University. She is currently enrolled in the doctoral program in computer science. The design of the Intel X-scale PXA255 has applications in high-end cell phones and avionics controls.

Presentations

Cheng, A. M. K. “Applying Formal Specification and Verification Methods to Real-Time and Embedded Systems,” Distinguished Lecture Presentation, University of Texas at Dallas, Sept. 2005.

— “Software Engineering for Embedded Software: How Useful Are the Newer Paradigms?” 9th Intl. Workshop on Software and Compilers for Embedded Systems, Dallas, TX, Sept. 2005. (*Invited panel speaker*.)

Cheng, A. M. K. and F. Shang. “Priority-Driven Coding of Progressive JPEG Images for Transmission in Real-Time Applications,” *Proc.*, 11th IEEE-CS Intl. Conf. on Embedded and Real-Time Computing Systems and Applications, Hong Kong, Aug. 17–19, 2005.

Funding and Proposals

“Faster and Stronger Schedulers for Real-Time Tasks on Heterogeneous Multiprocessors,” NSF, \$310,293. (*Submitted*.)

Update of ISSO 2004 Seed-Grant Project

Optimizing System Reward in Battery-Powered Spacecrafts and Rovers (ISSO 2004 Annual Report, 54-58)

Albert M. K. Cheng

Publications

Cheng, A. M. K. and C. Feng. “Predictive Thermal Management for Hard Real-Time Tasks,” *ACM Special Interest Group on Embedded Systems Review* 3.1 (2006).

(Continued on page 95.)



Courtesy NASA-Kennedy Space Center

Building and Verifying Fault-Tolerant Autonomous Real-Time Systems for Space Applications

(Continued from page 57.)

Cheng, A. M. K. and C. Feng. "Predictive Thermal Management for Hard Real-Time Tasks," *Proc.*, IEEE-CS Real-Time Systems Symposium WIP Session, Miami, FL, Dec. 5–8, 2005.

Presentations

Cheng, A. M. K. and C. Feng. "Predictive Thermal Management for Hard Real-Time Tasks," *Proc.*, IEEE-CS Real-Time Systems Symposium WIP Session, Miami, FL, Dec. 5–8, 2005.

Update of ISSO 2003 Seed-Grant Project

Timing Analysis and Scheduling of the X-38 Space Station Crew Return Vehicle and Other Space Vehicles (ISSO 2004 Annual Report, 58-59)

Albert M. K. Cheng

Publications

Andrei, S., W.-N. Chin, A. M. K. Cheng, and M. Lupu. "Automatic Debugging of Real-Time Systems Based on Incremental Satisfiability Counting," *IEEE Trans. on Computers*, 2006. *(To appear.)*

Andrei, S., W.-N. Chin, A. M. K. Cheng, and M. Lupu. "Incremental Automatic Debugging of Real-Time Systems Based on Satisfiability Counting," IEEE-CS Real-Time and Embedded Technology and Applications Symposium, San Francisco, CA, March 2005.

Andrei, S., W.-N. Chin, A. M. K. Cheng, and Y. Zhu. "Runtime-Coordinated Scalable Incremental Checksum Testing of Combinational Circuits based on #SAT Problem," *Proc.*, 11th IEEE-CS Intl. Conf. on Embedded and Real-Time Computing Systems and Applications, Hong Kong, Aug. 17–19, 2005.

Cheng, A. M. K. "A Survey of Formal Verification Methods and Tools for Embedded and Real-Time Systems," *Intl. J. Embedded Systems* 1 (2006).