
# STUDY AND SIMULATION OF A DISTRIBUTED REAL-TIME FAULT-TOLERANCE WEB MONITORING SYSTEM

Albert M. K. Cheng, Shaohong Fang
Department of Computer Science
University of Houston
Houston, TX, 77204, USA
http://www.cs.uh.edu

Technical Report Number 001

August 8, 2005

Keywords: Fault-Tolerance, Real-Time, Distributed System, Web Monitoring

## Abstract

The goal of this project is to study and simulate 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 due to a fault, the scheduled backup task is run and all tasks are assured to complete. 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 use the existing the Ganglia network and the 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 only for simulation purposes. It is used to show the results of the simulation graphically, but it can't accurately store the status data of the tasks. We will use the MySQL to store the status of the real time tasks instead of the RRDTool in further research and to yield more accurate results within the time constraints.

# STUDY AND SIMULATION OF DISTRIBUTED REAL-TIME FAULT-TOLERANCE WEB MONITORING SYSTEM

Albert M. K. Cheng, Shaohong Fang

**Abstract**

The goal of this project is to study and simulate 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 due to a fault, the scheduled backup task is run and all tasks are assured to complete. 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 use the existing the Ganglia network and the 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 only for simulation purposes. It is used to show the results of the simulation graphically, but it can't accurately store the status data of the tasks. We will use the MySQL to store the status of the real time tasks instead of the RRDTool in further research and to yield more accurate results within the time constraints.

**Index Terms**

Fault-Tolerance, Real-Time, Distributed System, Web Monitoring

## I. INTRODUCTION

Technological development has placed computer science in a central position in modern human life; humans are increasingly dependent on computing systems. Unfortunately system failure always happens in the real world. This project focuses on the simulation of a distributed real-time fault-tolerance web monitoring system. It is involved in the scheduling of the 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 due to 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 the values 1 and value 2 to distinguish the primary task and the backup task from the two different computers. We use the web to monitor the fault-tolerant behavior in the distributed system. Users always see fresh and consistent data for the primary task (value 1) or the backup task (value 2). A real-time system is a system in which data on the web should be updated constantly and tasks must be 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 replaced by a backup task permanently. 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 for the case where the primary task fails due to a processor failure. We can use another way to monitor the backup processor through the backup website, and guarantee all the tasks complete. 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 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 of the following: (1) real-time response, (2) distributed system, (3) fault-tolerance, and (4) web monitoring. To achieve our goals quickly, we choose to use the existing Ganglia network and the RRDTool technology.

## II. RELATED WORKS

### *A. Ganglia System for Distributed Real-Time Fault Tolerance Systems*

There exists a lot of network software targeting distribution systems. A typical well-performing one 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. The 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. Due to 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.

### *B. RRDTool for the Simulation and Visualization*

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, due to its efficiency of using hardware space and quickly generating graphical representations.

## III. SYSTEM ASSUMPTIONS

1) The Linux Fedora Core 3 platform and the Ganglia system are installed into the computers, both the primary and backup computers, which are connected through the Internet.
2) Tasks are aperiodic.
3) Each task has two versions, namely primary copy and 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 till 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 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 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.

## IV. FAULT INJECTION

The fault injection program provides the test cases for our system software. These test cases are used for evaluate, the performance or behavior of the newly designed software with the desired targets. Input faults to the system are in the form of PHP script which implements Java programs, Java programs 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 replicas task in the distributed computing node. It also switches back to the original tasks if the original tasks have been repaired.

## V. SYSTEM IMPLEMENTATION

### *A. Implementation Using Linux*

Our monitor 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 "sh" command, which is not allowed by the htppd deamon under Selinux with the default policy.

### *B. System Architecture*

Our system architecture is shown in Figure 1. The green components indicate additional components, Web components have been modified for this project. The light blue and gray components represent the Ganglia system [8]. The Gmetric command has to be wrapped into a shell program in order to execute in Java.

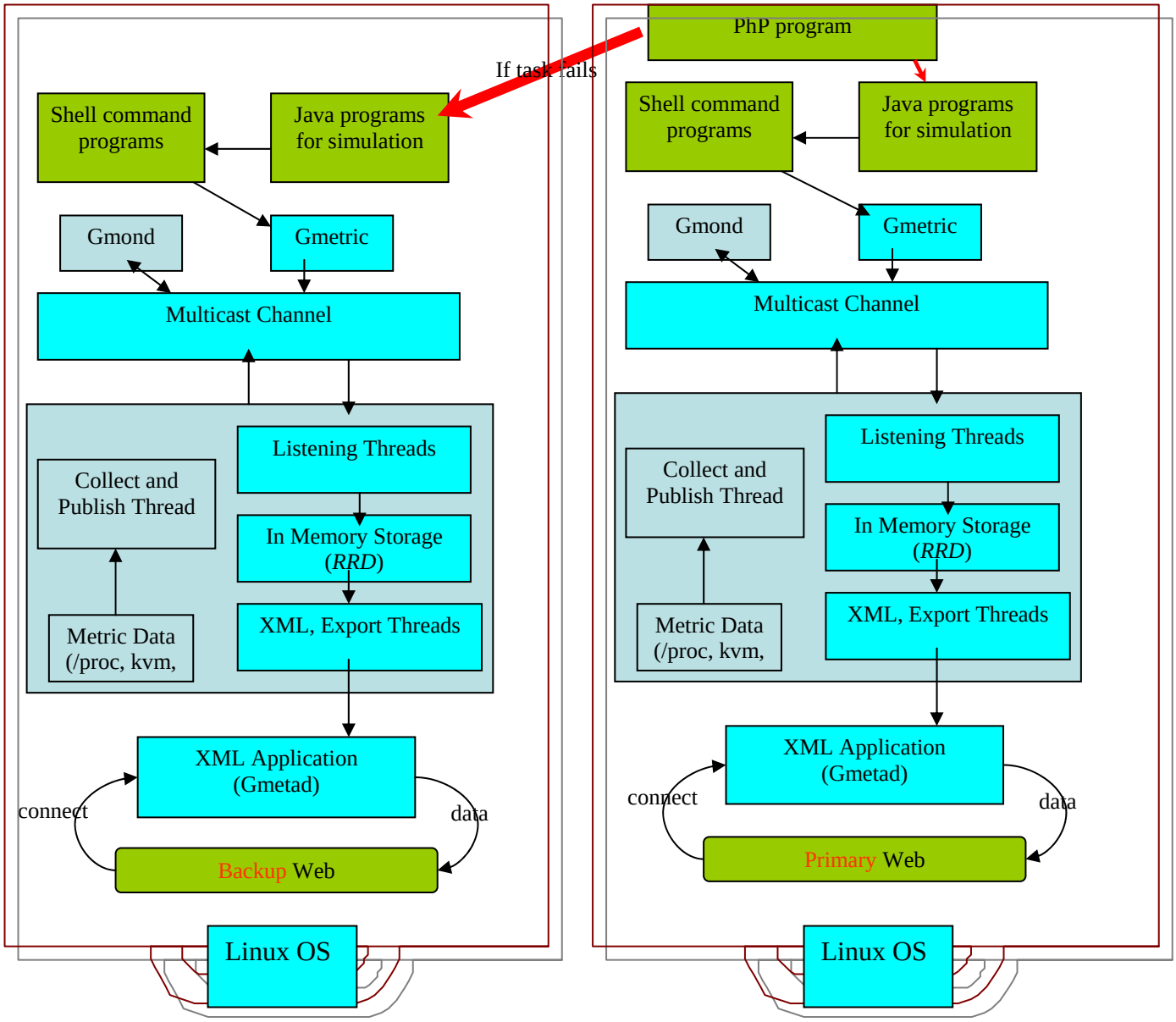


Fig. 1: The green components indicate additional components, Web components have been modified for this project. The light blue and gray components represent the Ganglia system.

## VI. SIMULATION

Our goals are: (1) to monitor the fault tolerance in the distributed system, (2) to monitor the backup copies of all the tasks, (3) to monitor the primary tasks and their failure and then schedule the backup tasks on the backup computer in the distributed system, and (4) to visualize the system's behavior graphically. Simulation results are shown in the Appendix.

## VII. FACTS AND ANALYZATION

Our simulation results show some unexpected results. We next describe several very interesting major functions and drawbacks of the RRDTool.

### *A. Data Acquisition*

We found that the interval between the primary task and the replaced backup tasks in our simulation is different because when monitoring the states of the real-time tasks, it is convenient to have the tasks status data available at a constant interval. Unfortunately it is not always to fetch data at exactly the time one wants to. Refer to Figure 3 as an example. Therefore, the RRDTtool allows updating the log file at any time. It will automatically interpolate the value of the data-source at the latest official time-slot and write this value to the log. Refer to Figure 2. The value supplied is stored and is also taken into account when interpolating the next log entry.

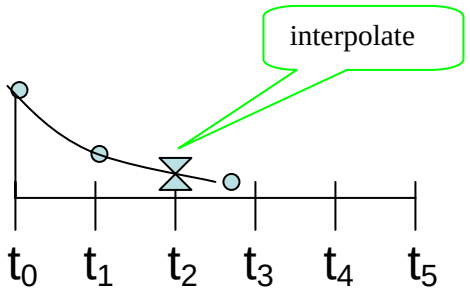


Fig 2: Interpolation of random time data into fixed time steps; Ganglia pull the data at fix interval.

### *B. Consolidation*

The results show that it is not the same data used as input to the database, between the interval of the primary task and the replaced backup tasks in our simulation. Refer to Figure 3 as an example. This is due to the fact that logging data with a fixed interval is not suitable to monitor the development of the data over a long period because of the limited hardware space and the consideration of increasing time of a large data set. RRDtool offers a solution to this problem through its data consolidation feature. When setting up a Round Robin Database (RRD), one can define at which interval 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.

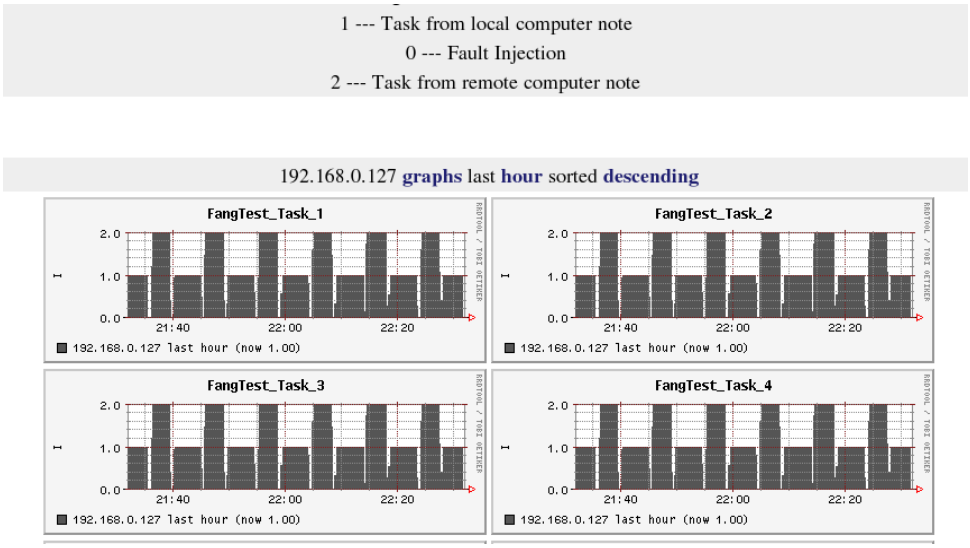


Fig 3: This is our simulation result. The result shows that the space interval between the primary task and the replaced backup task is different. The result also shows that the data at interval is not the same data used as input to the database.

## *C. Round Robin Archives*

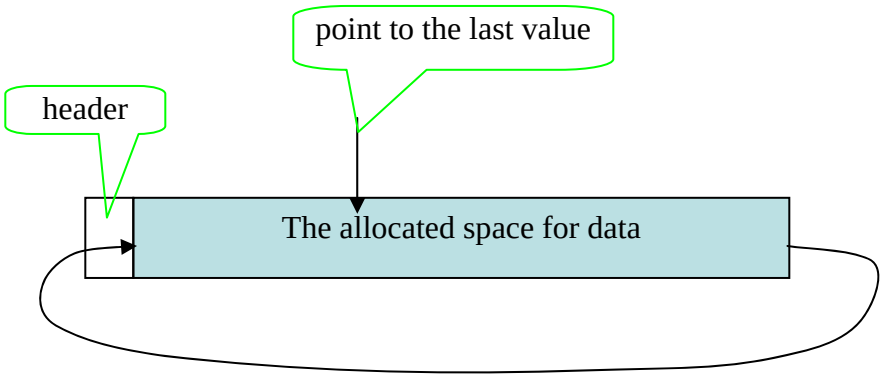


Fig 4: Space allocated for RRD consolidated data. The header holds the pointer information which points to the last value.

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. For approximately one day later, the data may become less accurate because of the consolidation stated above, which this is a negative side of the RRD.

## VIII. CONCLUSION AND FUTURE WORK

The Ganglia system and the RRDTool provide information that can help to better understand the fault-tolerance of a distributed system, and they provide real-time monitoring which make it possible to monitor and respond to potential faults and to schedule the backup tasks. The simulation results give us the further direction to implement the distributed real-time fault-tolerance system and to use web to monitor real tasks. We will use the MySQL to store the real status of real-time tasks instead of the RRDTool to yield more accurate results within the time constraints.

## IX. APPENDIX: SIMULATION RESULTS

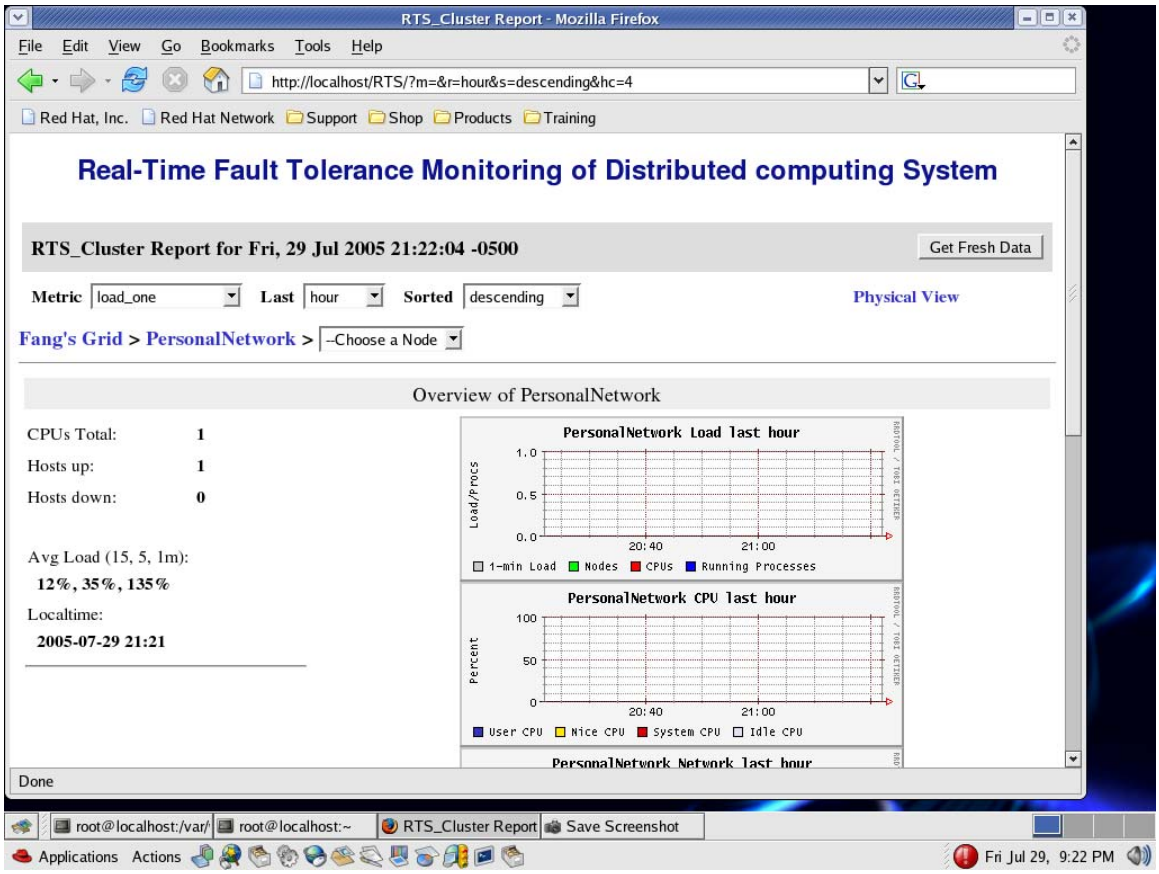


Fig. 5: Web monitor screen showing the major functionality in the remote backup computer.

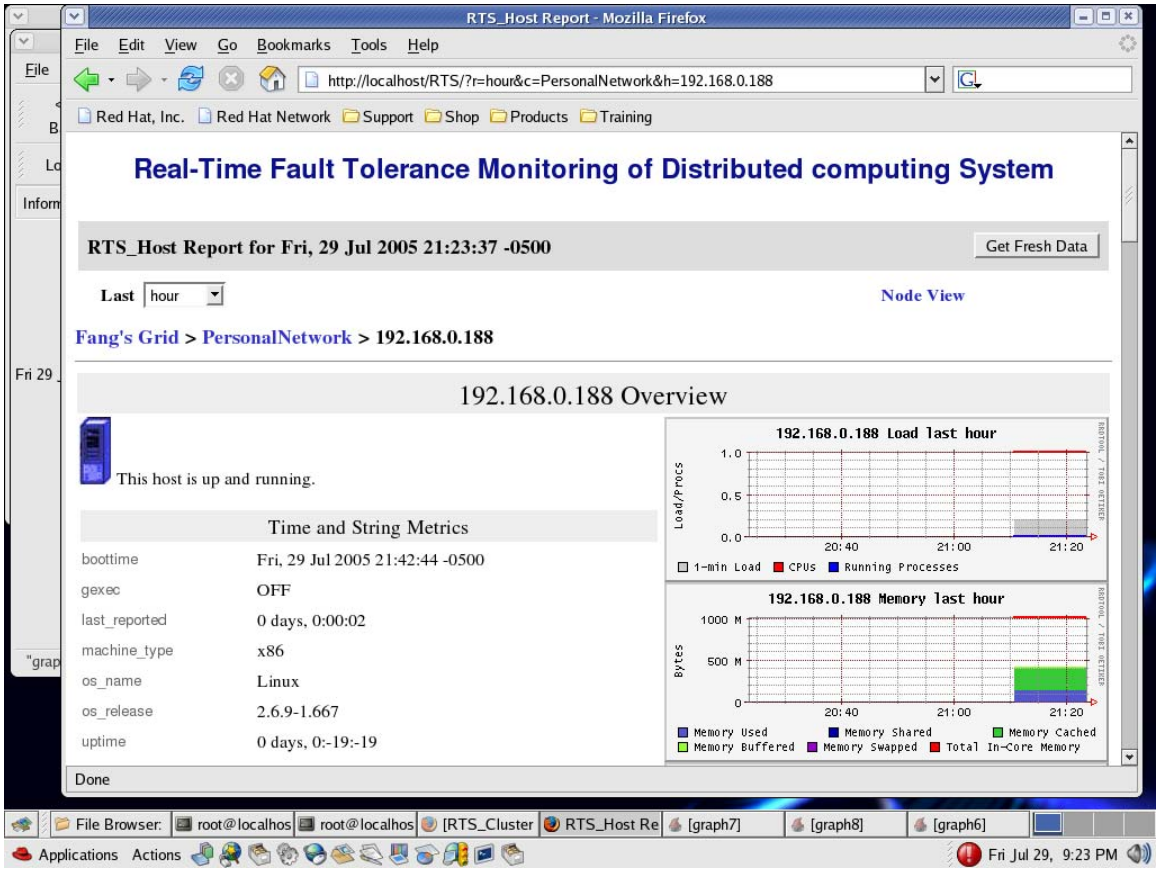


Fig. 6: Web monitor screen in the remote backup computer: clicking down list of choose a node, one can get the computer's IP address, its information, and tasks running on it.

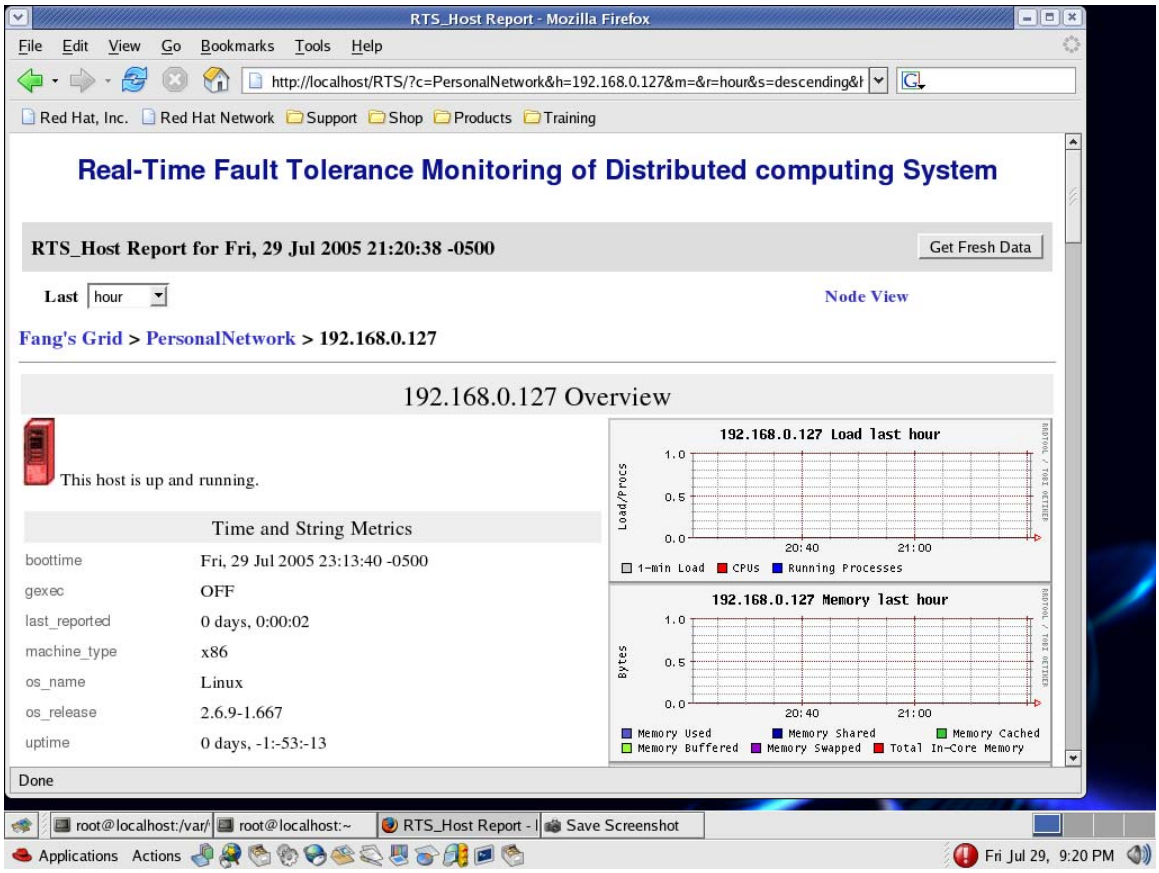


Fig. 7: Web monitor screen in the primary computer, showing the IP address and its information and the tasks running on it.

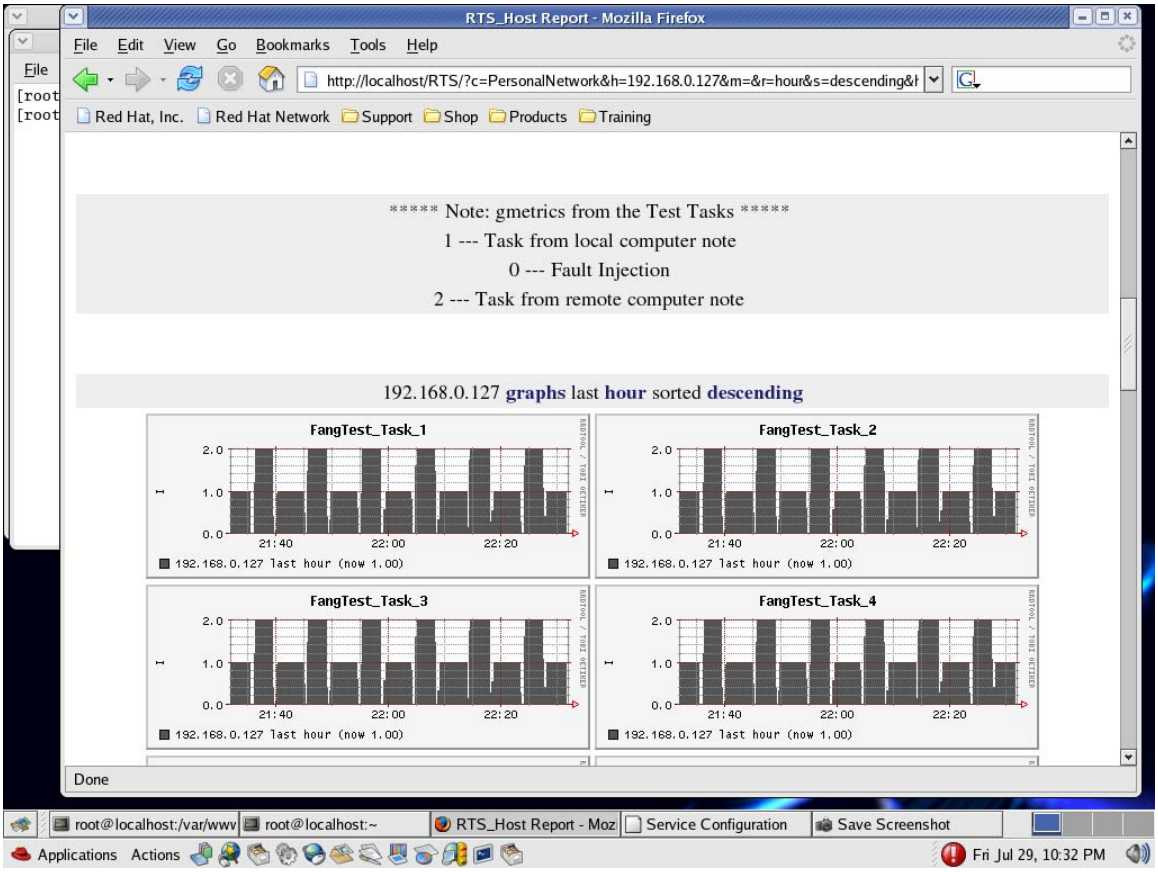


Fig. 8: Web monitor screen in the primary system. The intervals between the primary task and the replaced backup tasks in our simulation result are different. The data closed to the interval are not the same data we used to input to the database between the intervals of the primary task and the replaced backup tasks in our simulation.

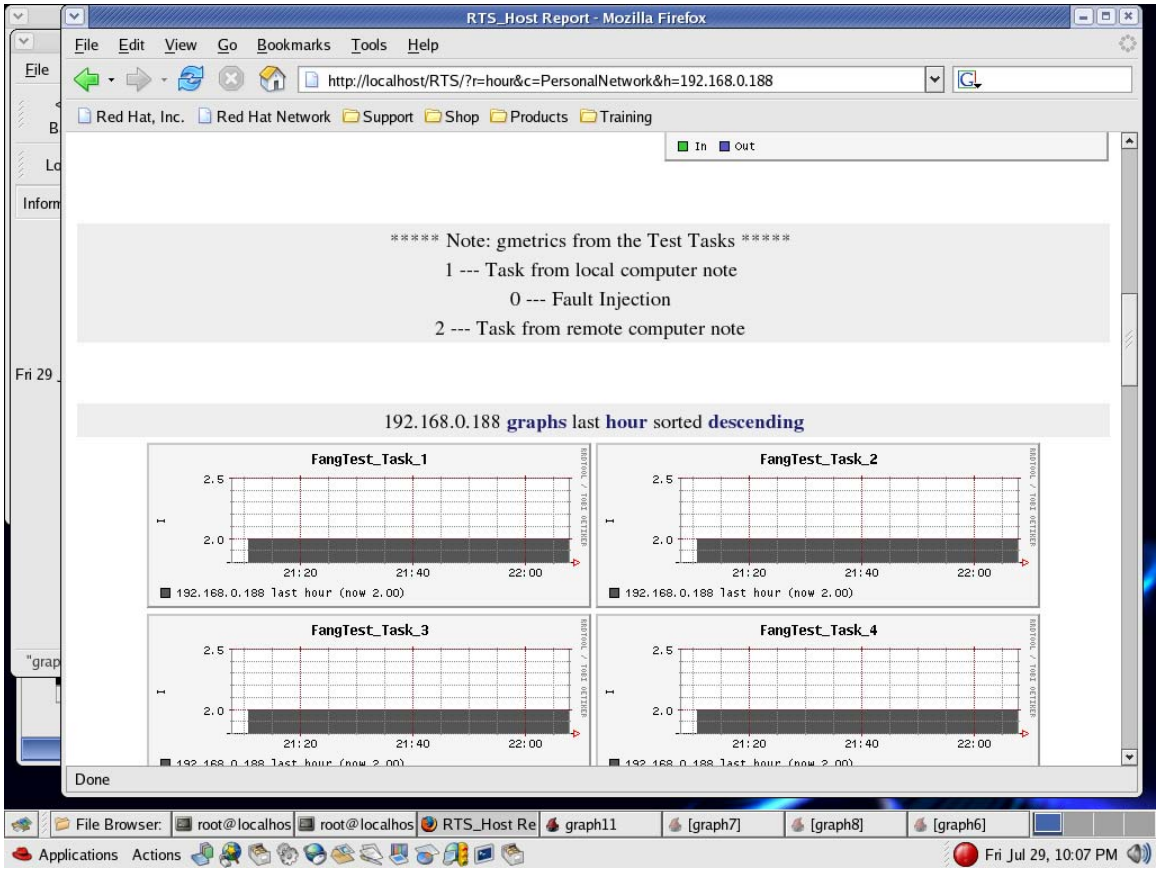


Fig. 9: Web monitor screen in the remote backup system showing the backup tasks.

## REFERENCES


[1] S. Balaji, Lawrence Jenkins, L.M.Patnaik, and P.S.Goel. Workload Redistribution for Fault Tolerance in a Hard Real-Time Distributed Computing System, In IEEE Fault Tolerance Computing Symposium (FTCS-19), pages 366-373, 1989.

[2] P.M.Melliar-Smith, L.E Moser, Progress in real-time fault tolerance; Reliable Distributed Systems, 2004. Proceedings of the 23rd IEEE International Symposium on
18-20 Oct. 2004 Page(s):109 - 111

[3] Indranil Gupta, G. Manimaran, and C. Siva Ram Murthy, "Primary-Backup based fault-tolerant dynamic scheduling of object-based tasks in multiprocessor real-time systems," Chapter 20 in Dependable Network Computing, D.R. Avresky (editor), Kluwer Academic Publishers, MA, USA.

[4] Indranil Gupta, G. Manimaran, C. Siva Ram Murthy, "Fault-Tolerant Dynamic Scheduling of Object-Based Tasks in Multiprocessor Real-Time Systems," In Annual IEEE Workshop on Fault-Tolerant Parallel and Distributed Systems, pp.83-107, Puerto Rico, April 16, 1999.

[5] G. 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, vol.9, no.11, pp.1137-1152, Nov. 1998.

[6] M.L Massie., B.N. Chun and D.E. Culler, « The ganglia distributed monitoring system: design, implementation, and experience », Paralle Computing 30 (2004) 817-840, pending publication,

[7] http://ganglia.sourceforge.net

[8] http://www.cs.berkeley.edu/~massie/papers/science.pdf