# A FLEXIBLE CLUSTERED APPROACH to HIGH AVAILABLITY

Gary Hughes-Fenchel
Lucent Technologies

## PRACTICAL EXPERIENCE ABSTRACT

The Reliable Clustered Computing project created a system which enables applications to improve the reliability of off the shelf computers from a typical 99% (about 90 hours of downtime per year) to 99.99% (under one hour of downtime per year) in a cost-effective manner. The chief constraints were the need to achieve high reliability while minimizing cost and maintaining vendor independence. This was realized by creating a vendor independent clustered configuration comprised of two or more computers capable of recovering from hardware or software errors by restarting one or more processes on the current machine or by failing over one or more processes to another machine. Only two inexpensive custom hardware components were required for this solution: a WatchDog, to monitor component status, and a PowerDog, to control electrical power to processing elements (and optional peripherals).The bulk of the functionality was provided by software.

## INTRODUCTION

The telecommunications switching industry has a historical legacy of extremely high computer availability. In the past, this was achieved by the creation and maintenance of specialized hardware and software which, although quite reliable, was very expensive to develop and maintain. Performance on these machines generally lagged behind the commercial computer industry. In the current world-wide telecommunications business climate, the price of equipment has become a critical factor in sales. A need was perceived to leverage (inexpensive) computer commodity hardware riding the price/performance curve and incorporate it into various telecommunication systems without significantly compromising service availability. The need for low pricing precluded using commercially available fault tolerant computers such as those made by Tandem® or Stratus®. Since the project was to provide a platform for a wide variety of applications, it was necessary to provide an extremely high degree of flexibility in terms of interfacing with the system and to make the platform vendor independent.

The Reliable Clustered Computing ("RCC") project was able to devise a method which offered applications the ability to improve availability of commercial non-fault tolerant computer systems two orders of magnitude, from a services provided standpoint of about 90 hours of downtime per year to under 1 hour per year. (This estimate of downtime includes only the time computing services are not available to processes and assumes one active and one spare computer. Time spent by an application in re-initializing varies widely from application to application and is not included). The software is currently supported on three different hardware platforms (SUN SPARC®, INTEL PENTIUM® based machines, and several HP® machines).

## CONCEPTS

- Cluster - a grouping of one or more loosely coupled physical machines able to support the same set of virtual machines. Any machine in a cluster can replace any other machine in that same cluster.
- Cluster element - one machine in a cluster.
- Virtual machine - an application program with all the following properties:
    1. It has an IP address associated with it
    2. It has a set of resources associated with it
    3. In a multi-computer cluster, it can be moved between physical machines
    4. It consists of one or more processes



0731-3071/97 $10.00 © 1997 IEEE

- Resource - any hardware, process, or collection of processes monitored as an entity by RCC and required by one or more virtual machines.
- Heartbeat - a periodic message sent from one process to another process to verify the sanity of the (first) process, or a periodic message sent from a piece of monitored hardware to the WatchDog to verify sanity of the monitored hardware. Faults are generally detected by missing heartbeats, and heartbeats are usually expected every few seconds. Faults are detected more slowly than would be the case in specialized hardware, where problem detection time is commonly measured in machine cycles. To the extent that applications are monitored via heartbeats, this requires intrusion into source code so the application can send heartbeats.Heartbeats periods are tunable.
- PowerDog - a customized hardware device which can turn electrical power on or off upon command.
- WatchDog - a customized hardware device responsible for maintaining state information about physical machines in the RCC system

## OVERVIEW

The RCC system is a state-based system. Each piece of monitored hardware is assigned one of the following states, which are listed from most ready to do work (highest state) to least ready to do work (lowest state):

- Lead-Active: (Computers only): The processor is doing useful work, and is responsible for any resource which could be connected to more than one machine.
- Active: A resource doing useful work
- Standby: A resource ready to do useful work
- Init: A resource performing tasks necessary to get to the standby state
- Offline: A resource ready to transition to the Init state when instructed to do so.
- Unavailable: A resource in none of the above states. It may have power removed.

RCC supports a wide range of architectures by modeling an "N +K" system, where N is the number of active nodes, and K is the number of spare nodes. Spare nodes complete as much initialization as possible so that when an active node fails the spare node can pick up it's work without intervention and fairly quickly. Duplication of critical hardware is recommended but not required, so that the system can survive any single fault. Full support is provided to allow duplicated ethernet ports on a single machine to be managed by the RCC software so that the application does not need to know when one port fails.

By relying on this basic model, RCC can be used with a single computer (N=1, K=0), with an active-standby two computer system (N=1, K=1), or other arrangement. A wide variety of network architectures can be supported. By making peripherals shareable (i.e. switchable) redundancy and cost can be decreased while simultaneously decreasing recovery time without loss of availability. RCC can intelligently manage that sharing.

The goal of RCC was to provide high availability at a low cost. This precluded the design of large amounts of specialized hardware. Only two pieces of specialized fairly simple hardware are needed by RCC: a WatchDog and a PowerDog. RCC is primarily a software effort, and most of the complexity of the system lies within its software (which is written in ANSI-C and ANSI-C++). All core RCC software must be running on every computer in the Init state or a higher state.

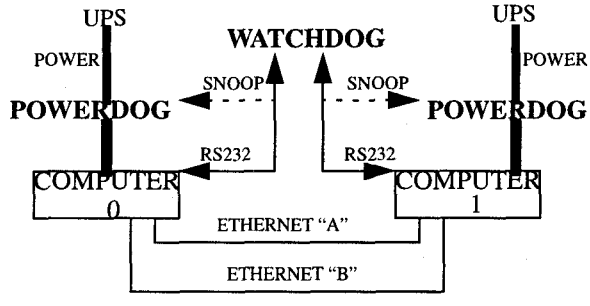


**Figure 1: RCC in it's most basic reliable form**

## THE WATCHDOG

The WatchDog itself consists of a microprocessor, some memory (both RAM and ROM), and at least 20 RS232 standard serial ports. Four of these ports have functions which cannot be altered: ports 0 and 1 are dedicated to an emergency action interface, and ports 2 and 3 are dedicated to cluster elements (computers). Usage of the remaining ports is determined by the application. WatchDog hardware is designed to fail passively, so that in the event of WatchDog failure the system is able to continue as long as the remainder of the system is stable. This prevents the WatchDog (which is not redundant) from becoming a single point of failure. The design is simple to assure a low failures-in-time (FIT) rate with an anticipated mean time between failures (MTBF) of about 80 years.

In it's most basic highly available form (see figure one), RCC consists of two computers (one Active, one Standby) connected by two separate Ethernets and a single WatchDog, plus two PowerDogs (one for each computer). The WatchDog is responsible for monitoring basic hardware sanity for multiple pieces of intelligent hardware (such as computers) by listening for heartbeats, and forcing

a recovery if any monitored hardware appears to be insane. Each individual WatchDog port except port 0,1,2,and 3 can be configured to operate in one of three modes:

1. Active ("bite"): The WatchDog actively monitors the cluster and automatically takes recovery actions when appropriate
2. Report ("bark"): The WatchDog actively monitors the cluster and reports state changes but does NOT automatically take recovery actions. The application remains in control of configuration management and recovery actions based on information reported by the WatchDog.
3. Passive ("obey"): The WatchDog does NOT monitor the operation of the nodes in the cluster; it will accept commands to take action, however. It is assumed that the monitoring and logical determination of the appropriate recovery action to take are covered by application provided software.

Recovery is accomplished either by the WatchDog sending an appropriate message to trigger a recovery sequence, or (in a more drastic fashion) sending a message to the PowerDog to remove electrical power from the offending unit.

In the event a WatchDog has to be reinitialized while RCC is stable and running (for example, when replacing a failed WatchDog) the WatchDog is able to use information in RCC data files as a checkpoint and be gracefully reintegrated into the system to resume its monitoring functions.

## THE POWERDOG

The PowerDog is capable of removing or applying the electrical power supply to a single device. It "snoops" on the WatchDog RS232 line (i.e. it monitors all traffic over the line) and responds to messages designated for it by powering up or down the electrical line it controls. It has very little intelligence and a low FIT rate.

## THE ETHERNET NETWORK

The Ethernet network supports a TCP/IP protocol connecting the computers to each other. To avoid a single point of failure, it is necessary to have at least two networks. Nearly all inter-computer communication is transmitted over the Ethernet with the remainder going over the WatchDog ports. The application may also use this Ethernet so long as sufficient bandwidth exists, or it can have it's own network.

## HARDWARE SWITCHES

Although not a necessary piece of RCC, some applications have chosen to use hardware switches to allow hardware to electrically attach to any one of multiple machines. A SCSI switch, for example, could be used to allow a single reliable group of disks to be used rather than requiring a separate group of disks for each computer and preempts the need for maintaining a wide band network to keep groups of disks on multiple machines identical. In the event the active computer fails the disks are electrically disconnected from the failing computer and attached to the spare which will replace the failing computer. The procedure can be managed by RCC.

## SYSTEM INTEGRITY SOFTWARE

System integrity assures that applications and cluster elements are sane. It is dealt with at three levels. Individual executables can be inserted into RCC data files. When RCC initializes, they will be treated as an RCC process and monitored for existence. Any process which does not exist will be restarted. If the restart count exceeds a specified threshold then RCC will re-initialize itself on that machine. This generally means that another machine, currently in Standby state, will be promoted to Active.

The second level of system integrity assurance is the virtual machine level. Virtual machines are expected to register and send heartbeats to RCC. If a virtual machine fails to send heartbeats it is torn down. RCC will then recover from the loss of the virtual machine by restarting that virtual machine somewhere in the cluster.

The third level of system integrity is the cluster element level. When the WatchDog is initialized it is informed of what is attached to each of it's ports, and which of the devices attached to it's ports are capable of supplying heartbeats. Each heartbeat includes information about when to expect the next heartbeat. If an expected heartbeat is missed, the WatchDog will take recovery action following an escalation strategy. This action could be reinitializing the cluster element, power cycling the cluster element, or even removing power from the cluster element.

## RESOURCE MONITORS

Cluster resources other than computers are frequently not intelligent enough to provide heartbeats. For this reason a resource monitor, which is itself a virtual machine, can be used to provide system integrity. The resource monitor performs what ever tasks are necessary to assure that the resource for which it is responsible is available, and recovers from problems. Resource monitors for disks and for the network are supplied by RCC. Applications are free to use these or to write their own monitors. The ability to

add resource monitors provides RCC with flexibility to deal with a wide variety of resource types.

## INITIALIZATION AND FAULT RECOVERY SCENERIOS

Typically, each computer is set up to invoke RCC as one of the later steps in a reboot. Once the heart of RCC is up it reads RCC data files to determine what processes need to be brought up. For each virtual machine RCC may be instructed to run specific code (known as a "prescript") prior to invocation. (It may also be instructed to run code after tearing the process down, known as a "postscript").

As the virtual machines come up they may register with RCC as appropriate and begin their tasks, reporting status to RCC as they become aware of it. Eventually, a cluster element (a computer) will report to the WatchDog that it has completed it's initializations and has changed it's state to standby. The WatchDog then decides which (if any) cluster elements ought to be promoted to a higher state (such as active, or lead-active). In the simplest case (two computers forming a single Active/Standby cluster) one element will be promoted to lead active, the other will remain standby. In the event the active element gets demoted (e.g. via a reboot) the standby element will quickly get promoted. This provides for quick recovery from disasters.

If a virtual machine fails, RCC determines on which computer (if any) it should be restarted. Since only an active computer runs virtual machines, in a two computer cluster with an active/standby configuration the choice is trivial. However, it is possible to have multiple computers in a cluster, from one (where there is no redundancy and hence much less availability) to N. The system is primarily limited by the number of available WatchDog ports.

Software failure is generally detected by loss of heartbeats to RCC software or the lack of existence of a particular process. Hardware resource failure is detected by it's resource monitor or by loss of heartbeat to the WatchDog. RCC uses different escalation recovery mechanisms depending on what is being recovered. For software, the mechanism used (and any needed parameters) are specified in RCC data files. Strategies include restarting the software on the current or another machine, reinitializing a virtual machine, ignoring the problem, reinitializing the physical machine, power cycling the physical machine, or powering off the physical machine. For hardware, the escalation mechanism used depends on the resource monitor and could include running specialized recovery software, power cycling, or powering down. Data for the resource monitor may also be stored in RCC data files.

## APPLICATIONS

The precise amount of additional availability offered by RCC is dependent on how the application uses the services RCC offers and what sort of physical configuration is chosen. At the very lowest end, an RCC system could consist of a single computer with no WatchDog and no PowerDog with the application source almost completely unchanged, and a single entry in one RCC data file. Under these conditions, RCC only provides restart capability. Adding a WatchDog, a PowerDog, a second computer (for failover capability), and two Ethernets would significantly improve availability by guaranteeing a computer was (nearly) always available to run the application.

Applications can choose how tightly to be coupled with RCC. Applications can query the status of any resource RCC controls, demand certain state changes, or affect RCC behavior on remote machines. Fault detection and recovery may be had either through RCC's services, or an application may provide it on it's own, or in combination with RCC. Applications can be coupled with RCC at the virtual machine level as one or more virtual machines, or at a process level. It is the ability to directly interact with application components that makes RCC significantly more flexible than other clustering approaches to improved availability.

The RCC scheme has been demonstrated to be extremely robust in telecommunication applications which have tightly coupled themselves to RCC. This has allowed, for example, one application to replace relatively expensive commercially available hardware duplexed fault tolerant computers with pairs of inexpensive PENTIUM® based computers. The application's own process monitor - which initialized, monitored and controlled the entire application as it existed on the duplexed computer - was presented to the RCC system as a single virtual machine. This made the transition from the expensive duplexed hardware to RCC possible without significantly restructuring of their software. The ability of RCC to control power to individual pieces of application hardware provided more control over the application environment than the more expensive duplexed fault tolerant computer.

## COMPETITIVE ANALYSIS

There are numerous fault tolerant offerings available currently in the marketplace. Generally speaking, they fall into two camps: those which are primarily hardware based, and those which are primarily software based. Hardware based systems tend to be more robust, with much quicker recovery time from faults. However, hardware based

systems also tend to be significantly more expensive. Since RCC is a (primarily) software based system, only other software systems will be considered.

There are two issues critical to any fault tolerant system: how to detect errors, and how to recover from errors once detected. Error detection in RCC is performed primarily by monitoring heartbeats. It is assumed that processes which run amok will fail to heartbeat. Of course, this is not necessarily the case, and RCC is defenseless against a rogue process which nevertheless retains enough sanity to heartbeat. Other error detection schemes do exist, such as checkpointing and voting (which requires multiple machines running essentially the same process), but they tend to be more difficult to manage, and/or more intrusive into application code. These schemes are generally more expensive to implement and have less flexibility than RCC.

The specifics of error recovery vary from system to system, according to it's architecture. However, most or all systems use an escalation strategy where possible. RCC attempts to keep recover efforts as localized as possible. In addition to offering "standard" resource monitoring (e.g. monitoring of faults on disks, tape drives, etcetera) found on many other commercially available systems RCC also supports monitoring of "non-standard" resources. This includes RCC supplied resources monitors for SCSI switches and RS232 switches, as well as full support for application provided resource monitors. This suggests that applications can use RCC to control any widget capable of being monitored via software. The RCC PowerDog offers power cycling as part of the recovery strategy for any hardware an application wishes to use.

Most (but not all) fault tolerant middleware available on the market is targeted to a specific vendor's hardware. It is not clear if this is largely a technical decision, or a marketing decision. RCC was designed specifically to avoid this dependency. Although it is currently available only under UNIX, plans exist to make it available under WINDOWS at a future date. It is currently available on three significantly different computer architectures (SUN SPARC®, HP series 800, and INTEL PENTIUM® based machines).

RCC is lower cost with better scaling than commercially available packages. Applications can define exactly how much redundancy they want. For example, multiple machines can "share" a single spare. It is even possible for a system to "share" excess capacity with zero spares. In such a system, if a machine generated a fault (i.e. went down for some reason) the critical processes running on that machine would begin running on other machines.

In several areas not related to it's fault tolerance abilities RCC lags behind most other offerings on the market. RCC's flexibility comes primarily from it's reliance on small databases for information, rather than hardcoding that information. Installing these databases requires a high level of skill. When an error occurs it is sometimes difficult to understand the initial cause: RCC's flexibility in configuration means that root causes for errors may be difficult to trace. These are shortcomings which will continue to be viewed as tolerable only for as long as RCC remains available exclusively to internal Lucent customers.

Failover time is currently indeterminate since initialization of application processes occurs at failover time and may be minimal or substantial, depending on the application. However, provision is being made for a "warm" state for an application, where the application will be initialized and made ready to run on a standby processor. The mathematically inclined will deduce (correctly) from this that numeric claims about reliability are also indeterminate. The reliability estimate given earlier in this paper assumes a very short application initialization interval and no application-induced recovery events.

By design, RCC recovers from faults, it does not correct them. A latent bug in application code which causes a fault is unlikely to re-occur after a failover since the application will be re-initialized. This is not the case in systems where identical application code is executing on two or more processors. This vulnerability to application bugs is one drawback to systems which create very fast recovery by running applications in lockstep or checkpointing. Such methods work well for hardware induced faults, but not for software induced faults.