

**Whitepaper:
Building Highly Available
Application Servers: The Network
Infrastructure**

Table of Contents

Achieving High Quality of Service with Carrier-Grade Availability3
Example: Support for New NGN Service4
Application Environment.....4
Controller Data Management: Carrier-Grade Availability6
Blade Data Management: Data Distribution8
Scalability9
Summary9

Achieving High Quality of Services with Carrier-Grade Availability

The increasing pace of life and business is providing the opportunity to deliver value-add products and services to an increasingly mobile and connected customer base across the IP network. Experience with earlier online applications has shown that customers have a marked aversion to downtime and delay. This has driven the need for what has come to be called carrier-grade availability. This is often expressed as 99.999% up-time, the well-known five-nines, which represents a maximum of about 5 minutes downtime per year.

In fact, a continuous 5 minutes of downtime would be catastrophic for many applications. In many cases the goal is continuity of service, as perceived by the user. To assist in this drive for uptime, hardware and OS vendors continue to improve the mean time between failure (MTBF) of their products. Hardware redundancy can also dramatically reduce downtime due to hardware failure. With this problem largely solved, the current focus is on enabling software to fail over in the same way, with imperceptible delays, and with full preservation of operational state. By surviving hardware faults, highly available software reduces the mean time to repair (MTTR) for the service or application as a whole.

This paper explains how to use Solid products to create:

- a robust and reliable data management platform
 - that enables developers to create and deploy scalable carrier-grade applications
 - is built on the blade-based architecture employed extensively in the network infrastructure
 - that provides continuous service availability

High availability can be achieved without using Solid as the reliable data manager, but it tends to be limited in scope and application-specific. Developers can also create their own data management platform as part of the architecture, and many development groups are doing just that. The highest cost of the Do-It-Yourself approach may be the lost opportunity that is caused by development delays, as customer-focused teams are forced to deal with system middleware issues. Developers concerned with time to market take advantage of pre-built high-value components wherever they find them. Reliable data management is a facility that enterprise application developers have taken for granted for several decades. The same functionality is now available in a form suited to the unique demands of the network infrastructure. Taking advantage of this platform can result in reduced development costs, faster time to market, lower total cost of ownership for the application, and more robust applications.

Solid provides an embeddable data manager that can be configured to provide carrier-grade availability. It also offers asynchronous communication to tie together the loosely connected parts of the application. It supports diskless nodes, and provides tunable in-memory data management that enables designers to finely tune price/performance.

Example: Support for New NGN Service

As an example, we will describe a service deployed on a blade server. We could use as example any application being deployed to support next generation services, such as GPRS Support Nodes (xGSNs), instant voice messaging, voice over Ip (VoIP), etc. The service is provided by a series of blades in a chassis. The chassis is controlled by Controller Cards which are, in turn, under the control of an Element Management System. Controller cards manage application data like buddy lists, presence information, etc.

We build this example up in stages. Not all applications will benefit from all of the options we describe here, but at the core is a recipe for building a highly scalable, highly available system on multi-tier, multiply redundant hardware and a carrier-grade OS.

Application Environment

No IP-based service exists in a vacuum, and of course our new service must communicate and cooperate with other systems such as billing, etc. To simplify matters, this paper examines just the underpinnings of the server on which the new application resides, as shown in Figure 1.



Figure 1. Application Server

Even though the diagram makes the Application Server look like a single entity, in fact it has considerable internal structure. Physically, it is composed of a chassis into which are plugged at least one controller board and at least one blade. The chassis provides such services as redundant power, cooling, and Ethernet connections, as well as a management interface that can be used by an Element Management System to control the hardware. The Controller

manages application provisioning, configuration, error handling, etc. for the blade cards. Because this task is so critical to the operation of the system, there is usually a redundant controller card, which can take over if the primary card fails or is taken down for maintenance. Often the tasks that the blades are performing are also mission-critical, and blades may also be deployed in redundant pairs.

The result is that the application runs in multiple tiers of the architecture simultaneously, with the requirement of transparent fail over at any level of the architecture with effectively no loss of service or information. These are daunting requirements. The application environment is shown in Figure 2.

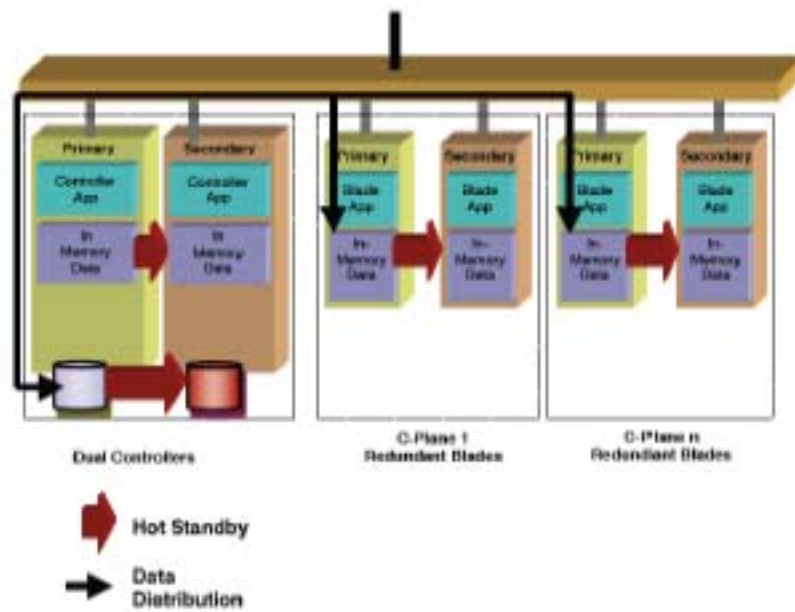


Figure 2. Application Server Architecture

There is a pair of redundant controller cards to manage the chassis. Software at the primary controller ensures that all of its data, even in-memory data, is faithfully instantiated in the secondary. This means that in the event of a failure of the primary controller, the secondary can take over from the last completed transaction, automatically rolling back any actions that were unfinished. Solid maintains the system state through failover, so that applications see no interruption in service availability. Applications will receive an error condition that will force them to reissue in-flight transactions to the secondary. Since they already have that connection open, once they are informed of the fault, applications can failover in milliseconds, depending on the high availability environment.

The blades in this particular application have no local disk, but obtain their initial state from the controller at boot time. Nevertheless, each pair of blades maintains a redundant copy of the in-memory-data, so that the secondary blade can take over with no loss of state or information in case the primary fails.

The combination of redundant controllers and redundant blades ensures not just a high overall uptime, but extremely short mean time to repair (MTTR), resulting in continuity of service.

Controller Data Management: Carrier-Grade Availability

At the core of the reliability of this system is the ability to maintain redundant copies of on-disk and in-memory data. Solid provides this functionality with its CarrierGrade Option. Under this option, a pair of Solid databases are kept in complete synchrony at all times. The task running at the controller, for example, writes to the primary database. Solid ensures that the data is also written to the secondary database at the same time. This secondary will normally run on the redundant controller card. It is a live database that can be used as a read resource.

A watch-dog application, either stand-alone or integrated with the high availability platform, monitors the health of the two servers. On failure of the primary node, all completed transactions are already on the secondary. The watchdog tells the secondary to take over the primary role. The secondary undoes any partial transactions that were in progress when the primary node failed, and then becomes available for activity. Applications normally maintain an open connection to both copies of the database, and, re-issue their transaction to the new primary, and then continue uninterrupted. When the primary node is repaired or replaced, it will automatically re-establish itself as a secondary node and resynchronize itself.

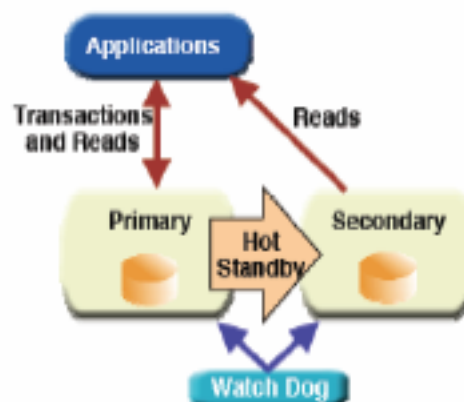


Figure 3. Solid CarrierGrade Option



Figure 4. Solid BoostEngine On-Disk and In- Memory Data Management

Solid BoostEngine™ allows an application designer to store and retrieve data in-memory, as well as on disk. This is done to obtain the significant speed advantage that memory has over disk. However, memory also has a severe cost disadvantage, so designers want to locate little-used data on disk for economy. Solid allows data to be located, tables by table, either in memory or on disc. Solid is the only data manager that gives the designer this fine-grained control. Solid guarantees recovery for in-memory data, and provides carrier grade availability facilities for both on-disk and in-memory data. Thus, even high-speed activities that take place in main memory can be protected with Solid CarrierGrade Option. In addition, the blades can use this feature to protect their in-memory data by having Solid copy it to a standby blade.

Because Solid is a full-featured relational data manager, it guarantees data integrity and consistency even in shared environments with carrier grade availability protection. The CarrierGrade Option guarantees that every completed transaction will be reliably saved across the pair of databases, and that every incomplete transaction caused by node or network failure will be completely undone and data returned to its prior state. Thus applications can remain unconcerned with controller card failure – for them it is one of many error conditions for which there is an automatic response. Building on the reliable redundant hardware foundation, Solid delivers easily managed application availability.

Solid CarrierGrade technology has been certified on MontaVista Linux Carrier Grade Edition, Sun Netra HA Suite, and the HP Service Guard architecture. Solid is an Early Access Partner in the Intel Modular Communications Architecture initiative.

Blade Data Management: Data Distribution

Within the chassis, data has to move between the controller card and the blades that are doing the application work. One way to do this is to locate a client application on the blade, accessing the server on the redundant controller cards for its data, using standard ODBC/JDBC connectivity. Another alternative is to locate a database on each of the blades themselves.

This is not a case for Solid GarrierGrade configuration, because the blades and the controller perform very different functions. Rather, the controller has to ensure that control data and application data like subscriber buddy lists gets to the appropriate blade, and it has to receive information from the blade so that it can control the entire chassis, and report back to its Element Management System. Solid provides a simple means to achieve this: Solid SmartFlow™ Option provides the appropriate data distribution.

To use Solid SmartFlow Option, the controller publishes views into its data. There is a view appropriate for each of the blades. Each primary blade subscribes to its publication, and gets a copy of the data from the controller in the form of a replica database. The blade can ask for a refresh of the replica data at any time. The controller can also signal to a blade that there is new information for it to download. In this way, blades can keep their data as fresh as necessary without having to continually poll the controller for new data.

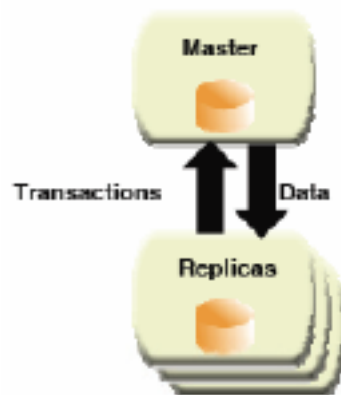


Figure 5. Solid SmartFlow Option

In this particular example, the blades are diskless. Solid supports this kind of architecture by allowing a remote boot the database from the controller card. However, Solid also supports disk-equipped blades as well. In that case, an application designer can locate tables in memory or on disk, as was described in the section above that described the controller card architecture.

Solid hot standby technology can be used to protect the blades as well, whether or not they are equipped with persistent storage.

Scalability

One of the challenges with next generation services is the speed of uptake. At initial deployment, applications must make the difficult transition from a prototype on a windows or Unix PC to a scalable blade-based architecture. When the application is in service, user populations can grow rapidly and the application has to be able to scale to meet the demand. Solid data management running on a blade server solves both of these problems.

The first step is to recognize that the target environment for the application is very different from central server. The application must be made ready for high availability by porting it to a carrier-grade operating system running on redundant hardware. The application has to recognize signals coming from the platform high-availability services that force shutdown, failover, etc. It needs to be aware of how to obtain and refresh its data and be programmed to do this based on timers or events.

Once this is accomplished, most of the work is done. It may be necessary to set up a partitioning scheme in the controller that is capable of receiving instructions to rescale the application over a different number of nodes (usually a larger number). The controller master database should publish its data based on parameterized views, so that the publications automatically recompute themselves when the number of nodes changes.

Subsequently, each additional node adds its own intelligence and computing power, so the architecture is free of single points of failure.

Summary

This paper has shown how to use Solid to provide a reliable underpinning for next generation network applications. Solid is designed to take advantage of modern blade-based hardware and hardened operating systems. It ensures data reliability and application availability within any given application tier by providing easy-to-use carrier grade high availability technology. Solid's data distribution technology can be used to automate data movement between tiers to support scalable distributed applications. With its ease of management that supports lights out operation, Solid can dramatically reduce time to market and total cost of ownership for next generation network service applications.



© Copyright IBM Corporation 2008

IBM Corporation
Software Group
Route 100
Somers, New York 10589
USA

Produced in the USA

02-08

All Rights Reserved.

IBM, the IBM logo, solid Information Technology, the solid logo, solidDB, EmbeddedEngine, BoostEngine, CarrierGrade Option, and SmartFlow Option are either registered trademarks or trademarks of IBM Corporation in the United States, other countries, or both. Other company or product names are registered trademarks or trademarks of their respective companies.

References in this publication to IBM products or services do not imply that IBM intends to make them available in all countries in which IBM operates..