

HIGH AVAILABILITY AND DISASTER RECOVERY — ORACLE 10G & 9I ADVANCED FEATURES VERSUS THE COMPETITION

*Jeffrey Bertman, DataBase Intelligence Group (DBIG)
and Ran Pan, PhD, Public Company Accounting Oversight Board (PCAOB)*

Table of Contents

→ Ctrl-Click on desired topic below to navigate directly to that section ←

OVERVIEW.....	2
BASIC ESSENTIALS.....	3
PUTTING THINGS IN PERSPECTIVE	4
SERVICE INTERRUPTION AND DATA LOSS — CAUSES AND PREVENTION	5
UNPLANNED OUTAGES	5
PLANNED OUTAGES.....	6
DATA LOSS.....	8
MORE TERMS AND METRICS (FOR BUSINESS CONTINUANCE REQUIREMENTS AND SOLUTIONS).....	10
AVAILABILITY AND RECOVERABILITY PERFORMANCE METRICS	11
DATA PROTECTION METRICS	13
INFRASTRUCTURE METRICS.....	13
MANAGEABILITY AND MAINTAINABILITY	15
ORACLE HA/DR FEATURES, STRENGTHS, AND WEAKNESSES	16
RECOVERY MANAGER (RMAN) AND FLASH RECOVERY AREA.....	18
AUTOMATIC STORAGE MANAGEMENT (ASM).....	19
FLASHBACK TECHNOLOGY	20
REAL APPLICATION CLUSTERS (RAC)	22
DATA GUARD.....	23
STREAMS AND ADVANCED REPLICATION.....	25
GRID CONTROL.....	28
BEST PRACTICES FOR BASIC INFRASTRUCTURE	29
A FEW SIMPLE RULES FOR HIGH AVAILABILITY DBAs AND SYSTEM ADMINISTRATORS	29
DISK STORAGE.....	30
NETWORK COMMUNICATIONS	32
APPLICATION SERVER AND MID-TIER SUPPORT	33
ADVANCED THIRD PARTY HA/DR SOLUTIONS.....	34
REMOTE MIRRORING EXPLAINED.....	35
REMOTE MIRRORING PRODUCTS VERSUS ORACLE	47
TRANSACTIONAL PROPAGATION EXPLAINED	47
TRANSACTIONAL PROPAGATION PRODUCTS VERSUS ORACLE	47
PROCESS CLUSTERING EXPLAINED	51
PROCESS CLUSTERING PRODUCTS VERSUS ORACLE.....	51
BASIC HA/DR SCENARIOS AND EXAMPLES.....	52
SCENARIO 1: COMPUTER FAILURE	52
SCENARIO 2: MEDIA (STORAGE) FAILURE.....	54

SCENARIO 3: HUMAN ERROR	57
SCENARIO 4: SITE FAILURE	60
SCENARIO 5: NETWORK FAILURE	62
SCENARIO 6: APPLICATION FAILURE	62
WRAPPING UP.....	62
ABOUT THE AUTHORS	63

PLEASE EXCUSE OUR DUST...

This paper is version 1.0. See upcoming releases at www.ioug.org (look for IOUG Live! 2005+) and other popular conferences. Or contact the authors directly per contact information provided in *About the Authors* section at the end of the paper.

➔ This paper contains over 60 pages (so far) of rough draft content ultimately slated for a book on high availability and disaster recovery. Most IOUG papers are about 7 to 10 pages. Due to space and time considerations for submitting version 1 of this paper for publishing in the conference Proceedings Manual/CD, there are definite rough edges. Subsequent versions will add new content (with even more details on third party tools), and also concentrate on cleaning up and reorganizing existing content. Meanwhile, we apologize for our “dust” :-).

OVERVIEW

In today’s highly competitive business and scientific communities where so much of what we know and what we do relies on computer automation, building a dependable IT architecture is critical to the success and essential well being all enterprises. Major database vendors and third party companies provide products that improve key reliability factors such as system availability, while optimizing indirect aspects of our environment such as minimizing recovery time and the amount of data loss in the event of disaster. The bottom line principal is that reliable IT infrastructure has become crucial to keeping business running smoothly. As we’ve learned from terrorist catastrophes like 9/11 and natural disasters such as hurricanes, floods, fires, and earthquakes, and even minor disasters such as computer or storage failure, IT availability is often key to keeping the overall business running — *period*.

Establishing a business continuity (BC) framework is crucial for most organizations, and a necessity for 24x7 mission critical OLTP and business critical decision support databases. This paper elaborates on the two major aspects of business continuity: High Availability (HA) and Disaster Recovery (DR). Primary focus is at the database applications level, including coverage of fringe technologies such as server, network, and disk storage. A multitude of comparison information and detailed best practices are conveyed for leveraging Oracle specific and third party offerings which sometime overlap and compete with Oracle, or may be complementary and provide added value. Oracle 9i and 10g specific topics include Real Application Clusters (RAC), Data Guard, Grid, Automatic Storage Management (ASM), Advanced Replication, Streams, Flashback functionality, and RMAN. Competing technologies come in three basic flavors: remote mirroring, transactional propagation, and process clustering. These technologies are explained in depth, and example products are discussed from a variety of perspectives. Vendors such as EMC, Quest Software, DataMirror, VERITAS, Network Appliance (NetApp), Hitachi Data Systems (HDS), and Mendocino Software are covered. Innovations are also presented, including when it is valuable to combine or complement technologies for maximum benefit. Business continuance requirements along with measurable goals and metrics are explained and quantified in meaningful terms, to help evaluate appropriate solutions for various environments. Real world planned and unplanned outage scenarios are cross referenced to specific architectures to better understand and validate suitability of different HA/DR solutions.

BASIC ESSENTIALS

The commercialization of most revolutionary inventions and discoveries breeds two basic results: progress and, ultimately, dependence. Once an invention becomes viable for mass production, we build new infrastructure that assumes its existence — and the invention that was once a remarkable achievement eventually becomes so commonplace that normal activity is impeded without it. Obvious historic examples include fire, the wheel, electricity, and discovery of the atom — all of which may be viewed as fundamental achievements. Compound achievements then combine and build on the fundamentals, e.g. the automobile and other machinated transportation, quantum physics, and computers. Just as the stove and even TV dinners are an evolution of how people use fire and electricity, information technology (IT) is a manifestation of how we use computers. And now that IT is an integral part of virtually every modern industry, living without it for even a short time would be like commuting to work without wheels, or like running a factory without gears or cogs.

The concept of fault tolerance involves taking things we depend on and adding safeguards to help ensure they remain available in working order. The safeguards may be proactive or retroactive, involving redundancy or some kind of restoration or recovery. An ancient example of redundant fault tolerance would be the perimeter walls of a city or castle fortress. The outside walls protect inhabitants from invasion, yet once broken there are additional barriers such as walls, doors, and locks. Trucks and airplanes have extra tires and engines (respectively). An automobile spare tire, jack, and wrench are examples of recovery tools used in the replacement of a defective part. In the IT world, we express the concepts of fault tolerance with terms like business continuity (BC), high availability (HA), and disaster recovery (DR). These refer to the basic means of keeping computers and related IT infrastructure running.

Putting these terms in perspective, fault tolerance and business continuity are basically synonymous, except that business continuity is a logical level concept which infers business operations as well as IT infrastructure. When automated systems fail for an extended timeframe, a degree of alternate procedures must be invoked and sustained, impacting the way people work. Fault tolerance is a more physical term representing the technical infrastructure providing protection to prevent or reduce the impact of system failure. High availability and disaster recovery are underlying practices we follow to maintain business continuity. High availability basically involves keeping the environment up with zero or minimal interruption in spite of the failure of one or more system components. HA framework often includes the implementation of redundant equipment or processes. Disaster recovery crudely equates to the automobile spare tire, jack, and wrench — the restoration of business and technical operations after a failure has caused unplanned disruption. The disruption is usually noticeable in that normal operations are interrupted for a (hopefully short) period of time. “Fast recovery” may seem to overlap with some forms of high availability, e.g. active-passive HA failovers which often involve a brief outage as control passes from one or more components to their fault tolerant “standby” counterparts. While basic DR focuses on recovery of primary resources, more advanced DR often involves restoration to highly available fallback resources. We’ll be crystallizing these concepts and the related technologies as you read on.

Perfect availability means providing 100 percent uptime — the ultimate “24x7” state (more on this later). Technically, 100 percent uptime precludes the need for disaster recovery. Since technology is not perfect, however, we need high availability and disaster recovery to complement one another. Availability is a type of reliability metric, as is the sometimes broader concept of business continuance. Emphasis on the word *sometimes* here, since business continuance for some people infers additional, tangent goals related to overall serviceability and performance. While this paper does consider performance when valuing and engineering various HA/DR solutions, focus centers on HA/DR versus the reliability and serviceability of business applications themselves.

True 24x7x365 infrastructure depicts an ideal state which, given today’s technologies, is virtually attainable — but not quite. Even if it were, the procurement, design, and implementation of HA/DR technology adds expense and is usually rationed. It has not yet become as commoditized as other aspects of IT. The good news is that high availability does not imply perfect availability, and there is exceptional value between mediocrity and perfection. High availability is typically measured in 9’s. For example, 99.999 percent uptime is known as *five 9’s*, which allows for approximately 5 minutes of downtime per year. Prorated this allows for only 25 seconds per month! *Four 9’s* is 99.99 percent, and so on. The HA Percent formula is as follows:

$$\text{HA Percent} = \text{MTBF} / (\text{MTBF} + \text{MTTR}) * 100$$

MTBF represents the Mean Time Between Failures and MTTR is the Mean Time to Recovery (how long it takes a system to become available again after a failure). There are many metrics associated with HA and DR, involving latency, data loss, performance degradation tolerance, and more. For now, just note that from a time-only perspective, the bottom of the high availability realm is usually considered to be three or two 9's, as shown in the table below. The near perfection of 6 Nines, while impractical for most environments, has at least started appearing on charts such as this over the past couple of years.

Availability Percent	Annual Downtime (Planned+Unplanned)	Monthly Downtime (Prorated)
99.9999 (6 Nines)	Approx 0.53 Seconds	0.53 Seconds
99.999 (5 Nines)	Approx 5.26 Minutes (315.36 Secs)	26 Seconds
99.99 (4 Nines)	Approx 1 Hour (53 Minutes)	4 Minutes
99.9 (3 Nines)	Approx 9 Hours (8.76 Hours)	0.73 Hours
99 (2 Nines)	Approx 88 Hours (87.6 Hours)	7.33 Hours

So we have decisions to make before determining the right HA/DR framework for our environment. As with life in general, building a successful HA/DR framework is especially dependent on setting the right priorities and making the right choices for each situation.

PUTTING THINGS IN PERSPECTIVE

IT systems with demanding HA/DR requirements are typically referred to as highly available or, perhaps more commonly, 24x7 high availability environments. They often incur steep financial penalties and/or lost opportunity costs when excessive outages and or performance problems occur. Many in-house and consulting based IT organizations offer Service Level Agreements (SLAs) which assure explicit uptime and performance levels, and specific remittance for undercompliance.

Once you have decided that your environment can (or should) be classified as highly available, there will be many requirements and constraints to analyze and technologies to evaluate before determining your most appropriate business continuity framework. That framework will generally combine one or more high availability architectures with at least one advanced disaster recovery mechanism. Fault tolerant technologies include RAID disk storage and redundant NICs as examples at the low end (we often take these fundamentals for granted), and redundant or clustered servers and databases at the high end where finding the funds and skilled talent is more challenging. It can seem quite challenging at first to devise and integrate a suitable set of requirements and features into a unified and affordable business continuance solution which addresses all business and technical requirements. But once you consider the cost of downtime in most HA environments — typically ranging from \$10,000 USD to **millions** of dollars per day or per hour — cost benefit becomes easily justified.

Industry	Operation	Approx Average Downtime Cost/Hour
Financial	Brokerage	\$6,450,000
Financial	Credit Card	\$2,600,000
Media	TV Pay-per-View	\$150,000
Retail	TV Home Shopping	\$113,000
Telecom	B2B Sales & Trouble Ticket (B2C and call processing/provisioning are generally <i>higher</i>)	\$100,000 SLA + Lost Opportunity

Industry	Operation	Approx Average Downtime Cost/Hour
Retail	Catalog Sales	\$90,000
Transportation	Airline Reservations	\$89,000
Media	Teleticket Sales	\$69,000
Transportation	Package Shipping	\$28,000
Finance	ATM Fees	\$14,000

Cost of Downtime — examples for various industries

Proven technologies become more abundant and more capable every year. The many choices can be staggering, though, so it is a good idea to bring in HA/DR specialists to assist. Even when narrowing our focus down to the database and fringe architectures, we may still end up with multiple selections. Let's start our analysis with some basic classifications. Later, we'll follow-up with more specific metrics.

Computing environments with the most demanding business continuity requirements generally involve *mission critical*, operational applications. Examples include internet/online sales, stock trading/brokerage, point of sale (e.g. credit card processing, etc), satellite communications, ATM processing, utility provisioning and support (e.g. for phone, power, water), automated tollgate operation, real time conveyor routing (e.g. for manufacturing and distribution centers), and reservation systems.

→ Business continuity framework is an **absolute necessity** for highly available *mission critical OLTP* systems.

Accounting and related operations such as payroll, billing, receivables, and basic procurement are classic examples of secondary, *business critical* applications which typically take next priority in the HA/DR chain.

→ Business continuity framework is a **prime imperative** for highly available *business critical OLTP* systems.

Business intelligence (BI) plays a key role in strategic planning, revenue and efficiency generation, and overall decision support. Only over the past several years, however, has BI been growing into a critical dependency. Cases in point include web site personalization (e.g. real time content customization based on visitor demographics, observed clickstream and spending patterns, affinities, etc) and supply chain intelligence (e.g. optimizing Just-In-Time inventory management, procurement costs, reliability management, etc).

→ Business continuity framework is a **secondary imperative** for highly available *business critical BI* systems.

SERVICE INTERRUPTION AND DATA LOSS — CAUSES AND PREVENTION

Before discussing requirements for building high availability and disaster recovery framework, we need to define and understand what constitutes a disaster as well as lesser forms of service interruption. We must also distinguish between *planned* and *unplanned* incidents. First, let's loosely define an *outage* as any event that prevents the data center from providing services needed by the designated applications for a period time. As we will see, there are full and partial outages. Applications may support a variety of real world activity. We often use the term *business* to generalize all of these pursuits, but remember there is plethora of non-business endeavors such as scientific applications used to survey and analyze geological or sub-geo materials, derive vital biological research results, or even keep satellites and astronauts in space, relaying vital information back to earth in real time.

UNPLANNED OUTAGES

Disaster and its recovery processes involve *unplanned* interruption of service. Unplanned downtime is mainly caused by computer failure, data failure, network failure, or even failure of a complete data center or site. Unplanned downtime may be strictly technical, as in a failed memory chip, or due to human error, as in erroneous calculations in a batch process that affect

thousands or millions of records, or the accidental deletion of records or even a complete table. Technical failures are physical in nature, whereas human cause is said to produce a *logical error* (or illogical, depending on your perspective :-)).

People often use the terms *outage* and *downtime* rather loosely. They usually imply loss of all service from an end user perspective, but could in fact mean some form of restricted access or partial outage. Users, for example, might remain uninterrupted but vital batch processes such as posting to General Ledger might fail because the disks supporting GL data files failed. So the CEO doesn't get his financial statements on time and an outage is declared. What has actually occurred is a reduction in *serviceability* of the application. This is an important distinction between availability and serviceability. Calculating statistics for serviceability are much more difficult than for availability, but both are indicators of the overall reliability or dependability of your environment. There are additional factors such as the accuracy of your data, calculations, etc. HA/DR metrics are covered in a different section of this paper, but application specific anomalies and such are outside our scope.

Another restricted access example leads us to the concept of planned full or partial outages. Say users can read but not write data — in a system where they are normally allowed to do both. That would constitute restricted access, although it may be referred to as an outage. If you are the culprit who directly or indirectly contributed to the problem, you might emphasize that the incident was technically a *partial outage*. From a business perspective, however, that may not be true. Some programs write to transitory work tables or even permanent audit tables simply when navigating between menu items. The inability to do so would restrict users from performing even elementary read-only tasks. Restricted write access in this case is functionally equivalent to a full outage, complete interruption of service. Despite this example, most restricted downtime is due to planned system maintenance work, e.g. table repartitioning or defragging, and bundled into the *planned downtime* category. Both full and partial outages should be tracked in your organization, and they should further be distinguished as planned or unplanned.

Table 1 presents major causes of *unplanned* downtime.

Unplanned Outage Types	Examples
Computer Failures	Server hardware (e.g. CPU, RAM, fan, NIC) Network components (e.g. cable, router, switch, hub)
Storage/Media Failures	Storage components (e.g. disk/RAID controller, HBA or NIC, disk platter/media)
Software Failures	OS, firmware, device drivers Database/Middleware/Applications
Human Errors (unintentional/malicious)	Logical data corruption (wrong batch job...) Physical data corruption (installed unsupported components) Drop wrong table or Delete wrong Data
Disaster (Local/Regional)	Natural: Fire, Earthquake, Flooding Unnatural: Power outage, fire, terrorism, accidents...

Table 1: Unplanned Outages

Since most HA/DR focus is implicitly concentrated on disasters, there are numerous tips and techniques for minimizing the occurrence of and optimizing treatment of unplanned outages. To facilitate understanding, best practices for both planned and unplanned incidents are revealed throughout this paper as various aspects of technology and framework are introduced.

PLANNED OUTAGES

As foreshadowed by example above, *planned downtime* is primarily due to scheduled data changes or system maintenance that must be performed periodically or sometimes as discovered or diagnosed. High availability environments generally require notice far in advance before allowing a full or partial planned outage from interfering with normal operations, even when scheduled for low utilization period. Typical lead times vary from 1 to 12 weeks. Exceptions are more unlikely than you

might think. One category of exceptions is security. Vulnerability to viruses worms, trojan horses, and other forms of malicious or even non-malicious intrusion is a common cause of system failure. We've all heard of and seen security related horror stories. So if your database or OS vendor sends a security alert advisory to apply a special patch, you would need to implement it the same day, right? The right answer is maybe, maybe not – *it depends*. (I try to avoid using that term, although it is a favorite of a friend of mine ;-)). See the *Best Practices* box below, as well as additional best practices presented later in this paper as various concepts materialize.

BEST PRACTICES: Dealing Proactively and Retroactively with Security Risks

Security is always a primary priority for any environment, and even more crucial in the HA/DR arena. Books on information security exist in abundance, so instead of digressing into discussion on intrusion detection, demilitarized zones (DMZs), encryption, and such, we offer a few notable tips highly relevant for HA/DR:

- Security patches require special and immediate attention. Do not automatically assume, however, that they always require breaking your planned maintenance lead time requirement (above). HA/DR framework unfortunately tends to be more susceptible than conventional environments to interoperability dependencies between various infrastructural components (topic covered in more depth elsewhere in this paper). Installing a patch of any kind, even a minor one, could soon follow with unplanned downtime which can be extremely difficult to back out. More frustrating might be the “you’re not supported” response you receive when seeking help after violating your vendor’s compatibility certification matrix (CCM). CCM basically describes the myriad of heterogeneous components a vendor officially supports for use with their product, e.g. Oracle RAC 10gR1 is certified to work only with a specific release and patch level of VERITAS’ clusterware (DBE/AC).

GOOD NEWS: If you have one or more standby environments, be inclined to expedite the patch as soon as possible since you have extra protection — unless your standby environment requires exactly the same patch level as your primary. Due to this conundrum, most HA/DR vendors have started decreasing this level of dependency.

BOTTOM LINE: While security and compatibility risks both pose availability concerns, *data and software components are more likely to be damaged by security breaches*. So in general, the adage “better to be safe than sorry” often applies. Ultimately you must exercise your best judgment, complemented by factors and tips you receive from well known industry specialists.

- Actively look for security alert postings on all relevant vendor websites, bulletin boards, and other documentation on a regular basis, regardless of what automated monitoring, safeguards, and countermeasures you have in place. Some vendors routinely send alerts by email, but do not depend exclusively on them. Be active and regular in your pursuit.
- Scan everything and do it often. Worms, viruses, and the like can work their way from development and testing environments into production and vice versa. Regardless of where people are going to connect to your network, any laptops brought in from outside should be scanned and cleaned of infections and any software/operating system security updates should be applied — before connecting to the system. Fixed workstations should also be regularly scanned at least every week and ideally each time they are booted, especially when connectivity to the Internet is allowed.

Scanning software which identifies both infections and major OS/software security patch requirements is available from a variety of commercial and government organizations, much of which is quite inexpensive and simple to install, operate, and maintain. For starters, see products from Harris Corporation, Computer Associates, and Defense Intelligence Agency (DISA). Note that products from DISA may be available only for government personnel and may require that you connect from a .mil internet domain, even for unclassified products. This is certainly not a complete list and no particular products are endorsed or promoted by this paper or the authors thereof; but the general practice of performing frequent scans cannot be overemphasized.

As HA/DR and even conventional technologies have advanced, there are less maintenance scenarios which require service interruption. The word conventional is important here. Availability constraints inherent in your basic DBMS, application servers, and other software infrastructure may render useless certain layers of HA/DR framework you have applied on top. Obvious examples are the need to shutdown the database to apply certain upgrades, or to restrict write access briefly on certain objects when performing special infrastructural changes. While adding active-active clustering in the case of Oracle may help with the upgrade scenario (Oracle 10g supports rolling upgrades, discussed later), some HA technologies merely replace one kind of restriction with another. Some DBMS's use "shared nothing" versus the "shared disk" clustering architecture used by Oracle. While there is a mix of pros and cons between the two, shared nothing architectures pose some steep challenges and restrictions on availability when performing tasks that are much more routine in shared disk. For example, adding a new server to scale-out a shared nothing database requires data to be redistributed from other live database server nodes to the new server. This usually requires some degree of read-only access restriction. Some tools help alleviate the problem by adding convenient tools to automate the migration bit by bit each night (affecting the least users) until enough data has been redistributed. Such a tool might be called a high availability add-on. Now contrast that with a shared-disk architecture which allows additional servers to be added with no access restrictions of any kind. No add-on is necessary and the availability picture is still brighter.

Table 2 lists common reasons for planned downtime.

Planned Outage Types	Examples
Hardware Upgrade	Server upgrade Network upgrade Storage upgrade
Software Upgrade/Patch Apply	OS, firmware, device drivers (OS, Middleware, Database, Applications)
Other	Backup/Restore Logical data maintenance (e.g. schema change, data transformation) General data center/site maintenance

Table 2: Planned Outages

DATA LOSS

Many disasters seem unlikely to occur, but become a frightening reality when they finally do. Consequences become more severe after recovery when you need to deal with potentially lost data. Successful HA/DR strategies must protect both the availability of your environment as well as the data you rely on. For some businesses the cost of data loss can be more detrimental than complete and extended downtime. In addition to providing highly available systems, today's enterprises also require a formal data protection plan with guaranteed zero or minimal data loss. Your disaster recovery plan must accommodate the whole picture. To help in this regard, the metrics provided in the next section address both availability and data protection.

One of the most difficult aspects of data loss is dealing with the question "Where do we pick up?". Let's call this process *transactional reconciliation*. Everyone agreed in your DR plan that minimal data loss was acceptable, and you might even have documented a specific tolerance such as 10MB or 1000 physical transactions. Most DR tools understand only physical metrics such as disk blocks or physical transactions. An organization's true recovery time (e.g. MTTR) must wait until business matters are running again, even if physical recovery has been complete. So determining data loss can be just as important as minimizing it. How do you correlate physical MB or physical transactions to meaningful business transactions which can affect many disk blocks at the same time, or generate hundreds or thousands of physical transaction updates for one business transaction such as posting an hourly batch of invoices to Accounts Receivable? (Posting just a single invoice results in the same problem.)

The most popular transactional reconciliation method to date involves manually comparing database data to the discrete data recorded on paper or some other analog data store — if indeed one exists. Since this is often performed in a hurry, it is prone to failure. People hope the errors will surface later on, e.g. during monthly account balancing or when people call to complain about missing something. In the most unfortunate event of complete site destruction, paper reconciliation is not an option. See *Best Practices* box below for more information.

BEST PRACTICES: Minimizing and Recovering from Data Loss

- Here is some information to help reduce the drudgery of transactional reconciliation after an otherwise successful recovery. This is where the “Where do we pick up?” question applies (above). Some of the tips below are obvious and well known, others are a little outside the box.
 - When data is entered from paper, place paper for completed entries in a separate stack. Ideally, stacks for different time periods should be separated. Pre-file-drawer separation should be more granular (e.g. each hour) than the final filing in a file cabinet (e.g. daily, weekly, monthly). By the time the paper is staged off to microfiche or the circular file, it certainly should not be required for reconciliation in an HA/DR environment.
 - Use offsite data mirroring when paper is not a transaction source, or even when it is — to minimize the tedium of manual reconciliation and to resolve the problem of site disaster which could destroy paperwork. Offsite data mirroring should not be confused with Remote Data Mirroring which is one kind of offsite mirroring (RDM is discussed in detail later). Offsite mirroring simply means copying changed data to both your primary database and at least one remote data store. The remote data store does not have to be a database, but is usually easier to accomplish when it is. Oracle offers free technology to duplicate database transactions from a primary to one or more standby databases (Data Guard and other replication technologies are discussed later). Third party vendors offer competing and complementary solutions as well (also covered later).
 - Use **application aware** propagation if possible. Physical data mirroring and replication technologies have recently been introducing business transaction intelligence such as the ability to bookmark in an electronic journal the fact that payroll posting has just completed, or that the next batch of 100 loan approvals was just finalized. Application aware technologies are a crucial emerging technology, and covered in more depth later in this paper.
- In order to achieve zero data loss, offsite mirroring must leverage expensive synchronous communications channels which simultaneously propagate data to the multiple locations. Synchronous, asynchronous, and semi-synchronous communication modes are discussed in detail later.
- To relieve the cost of synchronous communication, several things can be done while maintaining zero data loss:
 - Consider propagating only your redo and archive logs synchronously. Other data files can be sent using slower communication modes on different, less expensive channels.
 - If you mirror the same data to multiple secondary locations, consider propagating synchronously to one and asynchronously to the others.
 - If propagating all data asynchronously everywhere, it is even more important to monitor the environment for communications backlog. Set your monitor polling for every one to several minutes worst case. Also add a simple redundant monitoring layer to make sure the primary monitoring process is functional, at least for the monitoring components sensitive to data loss (e.g. the offsite mirroring processes).
 - Consider compressing data before it is propagated, especially if you have long data, e.g. LOB data types. This can be done within the DBMS to some degree and/or at the application level if you have the option of customizing the code.
 - Consider *Abbreviated Offsite Mirroring (AOM)*: It is faster and less expensive to send over a network the logical fact that sales order #1289 for John Doe at phone number 123-456-7890 was completed on 1/1/2010, than forcing every bit of sales order data through the network pipeline in real time speed. The abbreviated data may be small enough to send synchronously (real time as far as business is concerned) and remain in budget for network bandwidth. Even if abbreviated data is sent asynchronously, it should theoretically arrive at your standby data location before your physical data propagation stream, minimizing the extent of data loss. When disaster strikes,

you can call the people, ask for their sympathy, and hopefully regain the otherwise lost revenue data. Abbreviated data mirroring introduces other costs and should not be a sole substitute for physical mirroring technologies mentioned above. Application coding changes to accommodate AOM can be exorbitant to develop and maintain. You also need to utilize some other messaging vehicle to send the abbreviated data, e.g. messaging software which can be quite expensive and does not always support synchronous communications, or email which is certainly not synchronous and may intrude on performance of normal email unless conveyed on different servers and communication channels.

→ NOTE: The full extent of AOM costing is more difficult to calculate than other offsite mirroring strategies, unless your application vendor provides the option as a built-in or add-on feature. It is easy to calculate and determine long term pricing for synchronous versus asynchronous communications channels. Determining initial development and ongoing maintenance costs of AOM application customization is much more challenging.

- If you're going to use AOM, try to keep things simple in large applications by replicating only the most critical transactions.

MORE TERMS AND METRICS **(FOR BUSINESS CONTINUANCE REQUIREMENTS AND SOLUTIONS)**

High Availability is a well-known concept in the IT industry. It can be compared to Physics in the scientific community. Physics is said to be the mother of all sciences, and other disciplines like chemistry, biology, and arguably even mathematics stem from it (although mathematicians would say the opposite). Similarly, the practices of fault tolerance, disaster recovery, redundancy, and such all come from the main concept of keeping the system up – Availability. An environments where uptime is crucial to business (or scientific) operations requires High Availability. Of course, the word High is rather vague. In this section we will introduce some major terms and, better yet, metrics which help people translate the needs of their business environment into measurable IT semantics used to formulate a meaningful disaster recovery plan and engineer the solution architectures to support it. In this paper, the metrics improve our understanding of various HA/DR technologies, and narrow down an appropriate technical framework for supporting our business level HA/DR requirements.

Most of the metrics, as you would think, focus on dealing with a crisis situation, which we often refer to as a DR scenario. Although the majority of DR scenarios are relatively brief — a matter of seconds to hours — some catastrophic situations require days or even months to restore IT operations. There may be external dependencies (outside of IT) such as building power or even more drastic matters like quarantine period or site reconstruction. In longer term disasters, we need additional metrics to discern how long the business can operate under potentially degraded circumstances — either by dealing with reduced functionality at the primary site or by failing over to a secondary standby site which may be hundreds or thousands of miles away. All kinds of technical issues surface, such as dealing with secondary computers and related infrastructure which may not have as much horsepower as in the primary site. Network bandwidth and latency may be a concern based on the revised distances between a standby site and where most of the users work. That leads us to the relevant business concerns which are significantly more impacting in long term versus short term scenarios. For example, users might need to change work location to nearby or even distant offices. You might need to facilitate commuting or even relocation and rehiring if the failover turns out to be a more permanent “flopover” or “cutover”. Restoring normal or even diminished business operations goes hand in hand with IT infrastructure.

There are a plethora of DR related business concerns outside the scope of this paper. Some business matters, though, are so tightly bound to IT that we are obligated to address them to some degree. Of obvious relevance are certain fiscal metrics associated with the extra costs of operating a HA/DR environment versus one with more tolerance for outages and loss of data. Total Cost of Ownership (TCO) and Return on Investment (ROI) come to mind first for most people. If you happen to be in the lucky position of just beginning your HA/DR framework, then you can start gathering downtime statistics now and compare them after your improved HA/DR framework is established. Every time you improve the framework is an opportunity for additional metrics. Base level metrics such as the duration of each outage averaged over time (MTTR) and how much time passes between outages (MTBF) are projected over the actual costs of downtime to help derive higher level

cost benefit metrics. Outage costs are easy to determine if you have a well defined SLA, but you should dig in and try to validate your SLA regardless. Remember that SLAs often consider system performance in addition to downtime or restricted access metrics, so you may need to weed out irrelevant numbers. Also make sure to consider lost opportunity costs which remarkably are not considered in some SLAs. For example, some utility company SLAs include only certain fees they must pay the government or other utility firms for outage or performance problems (remember utility companies often depend on each other's infrastructure in geographic regions where their own is undercapacity). But what about all the customers who could not enroll for service through the online ordering system? This is a classic lost opportunity example. Lost opportunity for restricted access (versus full outage) can be difficult to measure, but can usually be interpolated. A typical restricted access scenario is where users could read data but not write it — for a system where they are normally allowed to do both. In many situations, not being able to write could be functionally equivalent to pure downtime if the application freezes or errors out when trying to write to a temporary work table in order to run a read-only report. So you have to count the incident as a full outage. In other cases, you might have an outage that affects only half your users. This would occur if you load balance your user community across different servers in a fully or partially replicated environment based on geographic regions or market segments, etc. To determine lost opportunity in this case, first calculate downtime for everyone and then prorate accordingly for the percent of affected users. Also remember that outages sometimes affect batch operations instead of users, but the business consequences can be just as severe if not worse. What happens if your payroll interface feed fails, for instance, and no one gets paid for the upcoming pay period? The costs for having your payroll outsourcer process out of cycle checks are easy, but what about calculating the impact on employees who depend on regular checks to pay their own bills? Worse, and basically immeasurable, is the impression conveyed to them resulting in lack of confidence and security in their employer. Since this is first and foremost a technical paper, we'll stop digging here, but keep in mind that someone in your organization either thinks or should be thinking at this level.

Major Business Continuity Metrics	
AVAILABILITY AND RECOVERABILITY PERFORMANCE METRICS	Remarks (+ Best Practices where noted)
<ul style="list-style-type: none"> ● MTTR, Basic MTTR and Full MTTR: Actual versus Tolerance <ul style="list-style-type: none"> ○ Basic MTTR: When data first becomes available in the failover environment. ○ Full MTTR: When <i>complete data restoration</i> is accomplished — all data is restored to the desired point in time or business event. ○ MTTR by itself refers implicitly to Max MTTR, although Basic MTTR is commonly implemented in its place (see comments). ○ May provide different MTTR metrics for recovery, failover, failback, and even switchover and switchback (see comments). ○ Actual MTTR metrics are the real figures experienced in your environment over time – remember the “M” in MTTR is effectively an average based on the measured amount of outage time per failure. ○ MTTR Tolerance determines how much downtime during a single outage the business can tolerate (typically before critical, possibly irreversible business consequences occur). 	<p>→ MTTR = Mean Time to Recovery.</p> <p>→ Basic MTTR is often mistakenly implemented in DR/failover scenarios. Before allowing users online in the secondary environment, it is strongly recommended that you wait until all in-transit data is applied to the database. Products like Oracle DataGuard's logical standby feature, Advanced Replication, and Oracle Streams allow users to go online before all data propagated from the primary site has caught up. Multi-Master replicated environments pose more risk in achieving Max MTTR since data sent to the failover/secondary database may not be allowed to apply until it is balanced by data in the other direction. Unfortunately, the reverse data flow is likely to fail when the primary database is inaccessible (which is likely to be the case when performing cross site failover).</p> <p>→ Switchover and switchback refer to <i>planned</i> failover/failback.</p>
<ul style="list-style-type: none"> ● MTBF: Actual versus Tolerance 	<p>MTBF = Mean Time Between Failures.</p>

<ul style="list-style-type: none"> ○ Measure of how much calendar time elapses between failures. ○ May distinguish between full versus restricted or degraded availability. ○ Actual MTTR metrics are the real figures experienced in your environment over time – remember the “M” in MTBF is effectively an average based on the amount of time between outages, usually measured from the end of one failure to the beginning of the next. 	
<ul style="list-style-type: none"> ● Required Availability Percent (RAP) <ul style="list-style-type: none"> ○ Measure of how much outage time can be tolerated over a period of time. ○ At a minimum, provide a RAP value for fully functional uptime support, noting that fully functional for a reporting database might be read-only access, whereas for OLTP it means completely read-write enabled. ○ Should also distinguish between planned versus unplanned availability outages. ○ May optionally distinguish between Full Access and a variety of degraded availability and serviceability states such as Read-Only Access and Degraded Performance (where performance is undercompliant with service level requirements, the system is still running). Detailed degraded availability examples are discussed elsewhere in this paper (multiple locations). 	<p>E.g. 99.0% (two 9's) to 99.9999% (six 9's): $RAP = MTBF / (MTBF + MTTR) * 100.$</p>
<ul style="list-style-type: none"> ● Planned Restrictions Schedule <ul style="list-style-type: none"> ○ Measure of how often and for how long planned maintenance activity can interfere with full, unencumbered availability. ○ Planned maintenance can involve a variety of availability restrictions ranging from Complete Outage (as far as business activity is concerned) to different kinds of restricted access, e.g. Read-Only Access and Degraded Performance (where maintenance activity slows the system down but things still run). 	<p>→ BEST PRACTICES — 4 SIMPLE RULES: See separate section of this paper: <i>A Few Simple Rules for the High Availability DBA.</i></p>
<ul style="list-style-type: none"> ● DR Mode Performance Tolerance (aka Role Reversal Tolerance) <ul style="list-style-type: none"> ○ Measure of how much performance degradation the business can deal with and for how long -- after failing over to a secondary DR site. ○ Units include: <ul style="list-style-type: none"> - Degradation Percent - Duration minutes or hours ○ Both full peak and mid peak degradation figures should be provided. 	<p>→ Applies to planned switchovers (not just unplanned failovers). Whenever secondary environment becomes primary or vice versa (via role reversal or flip/flop), if the new environment has less power or capacity than the original environment, there may be a time restriction on how long operations can stay in the new environment before a change is made. A change may involve adding new infrastructure to raise performance or capacity levels. Or it may mean that you must flop back to the original environment. If all equipment is functionally equivalent at primary and secondary sites, geographic distance between most users and the relocated site can still impact environment performance. If all factors are truly identical, the DR Mode Performance Tolerance is 100 percent forever.</p>
<ul style="list-style-type: none"> ● Max (and Min) Decision Time: Actual versus Tolerance <ul style="list-style-type: none"> ○ Measure of how much time occurs before deciding on the recovery solution for a crisis situation (e.g. recover in place, failover, etc). ○ Actual metrics are the real figures experienced in your environment over time, whereas Tolerance indicates how much your DR plan allows before people are forced into a default failover scenario. ○ Max and Min for Tolerance figures represent, respectively, the most amount of time that may be tolerated, and the minimum 	

<p>amount of time required to consider a solution before implementing it.</p> <ul style="list-style-type: none"> ○ May break down this metric, distinguishing between decision times for various basic scenarios, e.g. recover in place, failover, and possibly even fallback. 	
<ul style="list-style-type: none"> ● Max Technical Prep Time: Actual versus Tolerance ○ Measure of how much lead time is required to ready the environment from a technical perspective before failover can begin. ○ Actual metrics are the real figures experienced in your environment over time, whereas Tolerance indicates how much your DR plan allows before other HA/DR metrics are sacrificed, e.g. MTTR. ○ Should distinguish between Planned and Unplanned scenarios, and probably failover and fallback as well. 	

DATA PROTECTION METRICS	Remarks (+ Best Practices where noted)
--------------------------------	---

<ul style="list-style-type: none"> ● Data Loss: Actual versus Tolerance ○ Measure of how much data is lost during a crisis scenario, regardless of the kind of DR solution used (if any) to resolve the crisis. ○ Possible units include: <ul style="list-style-type: none"> - Logical and/or physical number of transactions - KB or MB of data – hopefully kept down to KB - Recovery Point Objective (RPO) is a popular term which equates to Data Loss Tolerance (see comments). ○ Actual metrics are the real figures experienced in your environment over time, whereas Tolerance indicates how much your DR plan allows for before critical and possibly irreversible business consequences occur. ○ May need to categorize by peak, mid-peak, and low utilization periods, since data at various times of the day/week/etc may have differing value to the organization. ○ Should distinguish between Planned and Unplanned scenarios, and probably failover and fallback as well. 	<p>Recovery Point Objective (RPO) and Data Loss Tolerance are basically synonymous — the maximum amount of data you can afford to lose before critical, possibly irreversible business consequences occur. The word critical can represent entirely different thresholds for different organizations, depending on many factors such as Service Level Agreement (SLA) penalties and lost opportunity. Lost sales are a classic form of lost business opportunity should your online order processing system experience unrecoverable data corruption or lose unrecoverable data in transit from primary to secondary servers during a failover.</p>
---	---

<ul style="list-style-type: none"> ● Data Latency: Actual versus Tolerance ○ Measure of lag time between primary and secondary data currency. May indicate either technical/capacity bottleneck conditions, or could be intentional (see comments). ○ Actual metrics are the real figures experienced in your environment over time, whereas Tolerance indicates how much your DR plan has allows for. ○ May need to categorize by peak, mid-peak, and low utilization periods, since data at various times of the day/week/etc may have differing value to the organization. ○ CRITICAL to distinguish latency across all replication channels, e.g. from primary to multiple secondary servers, regardless of whether the multiple targets are all one replication hop away or they are architected in a cascade (multi-hop) fashion. At a minimum, distinguishing between intra-site and inter-site latency figures. ○ Should distinguish between Planned and Unplanned scenarios, and probably failover and fallback as well. 	<p>→ Note that planned latency may be required to recover from logical corruption, depending on the DR technologies utilized in your environment (discussed later). It is possible to have zero or minimal latency between primary and one secondary server, while maintaining a larger gap, e.g. one hour, between primary and a different secondary server. This minimizes latency from inadvertently causing data loss to spike in the event of failure.</p> <p>→ Latency is a prime cause of data loss, so this metric is extremely important.</p> <p>→ Inter-site latency is often irrecoverable (versus intra-site latency which simply takes extra time to apply).</p>
--	---

INFRASTRUCTURE METRICS	Remarks (+ Best Practices where noted)
-------------------------------	---

<ul style="list-style-type: none"> • Role Reversal Tolerance 	<p>→ See <i>DR Mode Performance Tolerance (aka Role Reversal Tolerance)</i> in the <i>Availability and Performance Metrics</i> section above.</p>
<ul style="list-style-type: none"> • Secondary (DR) Server Utilization: Actual versus Required <ul style="list-style-type: none"> ○ Measure of the time windows that a secondary site is available for operational purposes, e.g. to offload reporting, testing, and even recovery data analysis in the event of a crisis. ○ Units typically include, at a minimum: <ul style="list-style-type: none"> - Amount of contiguous time per online availability window - How many online availability windows per day or week ○ CRITICAL TO DISTINGUISH BETWEEN ONLINE SUPPORT LEVELS listed below in order of increasing functionality for secondary server: <ol style="list-style-type: none"> a) Always Passive: Secondary site is never placed online until actual failover to the site occurs. b) Physical Read-Only (Restricted Reporting): Many kinds of reports can be run, but not anything that requires database writes to persistent storage. c) Full Reporting: Any kind of report can be run, including those which require writes to interim or temporary work tables. d) Unidirectional Writes: There is sufficient capacity (and enabling technology) for secondary environment(s) to support read+write testing or other activity which can write to secondary storage but not care about retaining the writes. e) Bidirectional Writes: Data can be written on both primary and one or more secondary environments, but there is no means of performing conflict detection, conflict resolution, or even periodic data resynchronization. In database terms, this means that sessions on one server should not write to the same records that are written to by sessions on replicated servers. Enforcement may be arbitrary, however (enforced only by convention, not by physical policy restriction). Bidirectional is viewed by some as Peer-to-Peer Replication, but we avoid the term here since to many it connotes Multi-Master (below). f) Full Multi-Master: Write transactions can occur on any server. Distinction between primary and secondary may not exist at a logical level. 	<p>→ Online Support Levels starting with <i>Physical Read-Only</i> (and higher) generally allow secondary servers to have <i>composition</i> differences. For example, database init parameters may be tuned for reporting versus OLTP. Depending on the technology being used, <i>Full Reporting</i> and higher levels usually allow DBMS indexes, horizontal partitions, and such to differ as well. <i>Content</i> differences might also be supported. This means, for example, that data in a primary site could be vertically partitioned or distributed across multiple secondary servers which all would be required for front line production in the event of failover. This can get ugly real quick...</p> <p>BEST PRACTICE: Use EXTREME CAUTION when varying any secondary site by composition or content (above). If you change your only secondary site and then need to failover to it, your normal production environment may not function well under the reconfigured conditions. See <i>DR Mode Performance Tolerance (aka Role Reversal Tolerance)</i> in the <i>Availability and Performance Metrics</i> section above.</p> <p>→ DR environments with <i>Always Passive</i> online support level for secondary site(s) are said to be active-passive. Compare this with higher online support levels which all constitute varying degrees of active-active. We usually avoid the term active-active, though since to many it implies one or both of the top two online support levels (<i>Bidirectional Writes</i> or <i>Full Multi-Master</i>).</p> <p>→ Oracle Data Guard's physical standby feature falls into the <i>Physical Read-Only (Restricted Reporting)</i> online support level.</p> <p>→ Oracle Data Guard's logical standby feature (aka SQL apply) falls into both the <i>Full Reporting</i> and <i>Unidirectional Writes</i> online support levels, depending on how you use it.</p> <p>→ The Online support levels known as <i>Bidirectional Writes</i> and <i>Full Multi-Master</i> are satisfied by a variety of sophisticated replication technologies which may or may not be configured for automatic conflict detection, automatic conflict resolution, and periodic data resynchronization. Examples include Oracle Streams, Advanced Replication, and third party products such as Quest SharePlex, DataMirror iReflect, and DataMirror Transformation Server.</p> <p>BEST PRACTICE: The desire to milk your failover environment for all the functionality you can should not be the exclusive driver in deciding on multi-master replication. Yes, it may be frustrating to let millions of dollars of equipment sit in zero-or-under-utilization mode, but the unnerving</p>

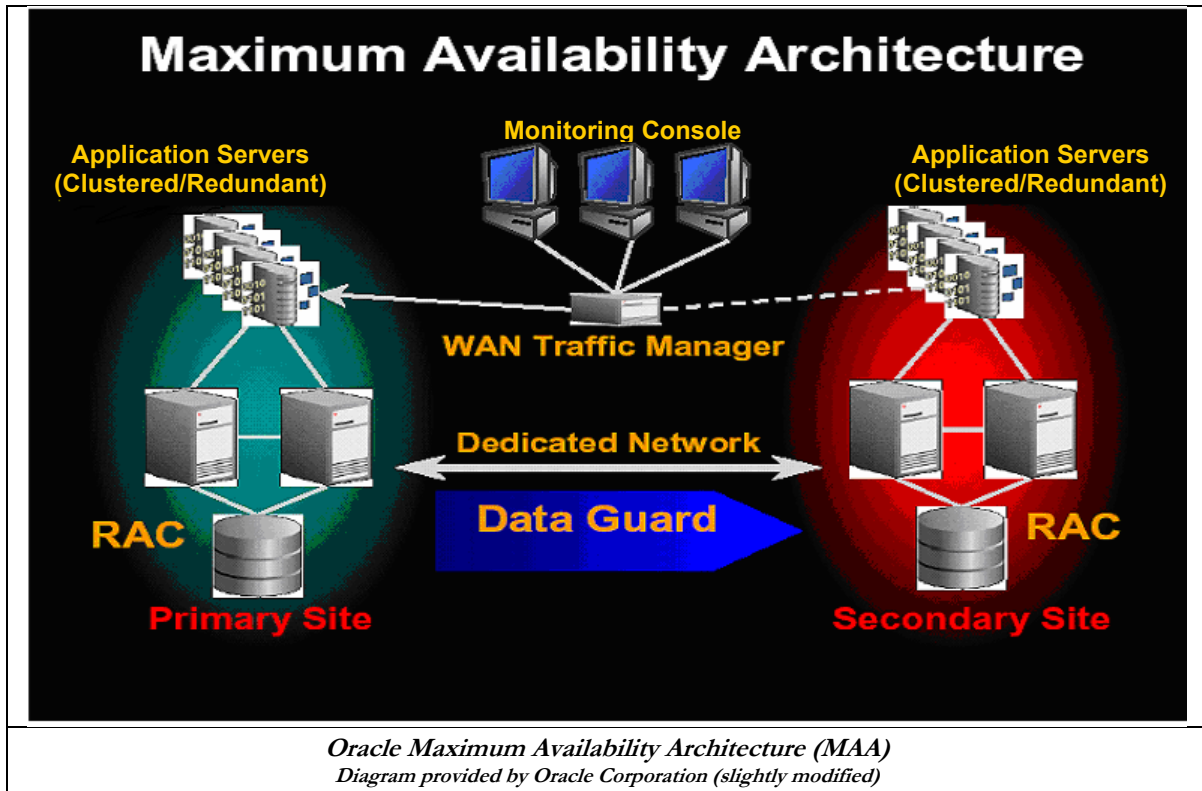
	<p>experience of walking a high wire with neither safety net nor prior experience would probably outweigh your frustration in terms of intensity. True multi-master replication (MMR) involves managing data contention in a distributed environment where some of the same logical records replicated across potentially distant servers are updated or deleted by different people at the same time. Careful planning is required to avoid, detect, and resolve the data contention — in this specific order. Jumping into MMR is like leaping onto the high wire for the first time, except we need to modify the circus act a little. If you fall from the MMR high wire, you won't break your neck on bare concrete. Instead, you'll fall into a tar pit just shallow enough to not drown yet deep enough to cover you with muck. The muck is equivalent to data corruption, and it can be just as tedious to clean up. MMR focused best practices are too numerous to list here, although you may find respite in a sister paper entitled <i>Bridging the Gap in Multi-Master Replication — Building an Advanced Replication Toolkit?</i> (same author).</p>
<ul style="list-style-type: none"> • Key Physical Characteristics <ul style="list-style-type: none"> ○ Geographic Distance Between Locations: Measurement in feet, miles, or kilometers between each replication pathway. ○ Intrinsic Infrastructural Resource Requirements: Capacity measurement of CPU, memory, disk processing requirements, etc as they pertain specifically to the support of the HA/DR framework. ○ Extrinsic Computer Resource Requirements: Same as for intrinsic, but focused on the infrastructure requirements of the application environment, independent of the HA/DR framework. 	<p>→ Geographic distance is a key factor of data latency (reviewed above). The longer the distance, the greater inclination for latency and, of course, consequential data loss should a failure occur while data is backlogged.</p> <p>→ Probably the most sensitive Intrinsic Resource Requirement is communications throughput capacity of each relevant replication path (between locations). It is typically measured in MBits or GBits per second.</p>
<ul style="list-style-type: none"> • Depth of Protection <ul style="list-style-type: none"> ○ Points of Failure guidelines. ○ Redundancy guidelines. ○ Automation, scripting, packaging guidelines versus manual procedure which is more prone to failure. ○ Application/session level resilience characteristics. (E.g. Session reconnect iterations, Transparent Application Failover success rate or delays, etc.) 	<p>→ Just as security plans depend on multiple layers of protection, HA/DR plans may specify thresholds for key mitigators such as maximum points of failure and/or minimum levels of redundancy.</p> <p>→ Application resilience guidelines may also be established, including metrics on tolerable failover timing and iteration thresholds. For example, if a session takes an average of three attempts to reconnect on failover before it is successful, this may be deemed unacceptable. An application may have too many scenarios where Transparent Application Failover provides insufficient value or untimely delays.</p>
<p>MANAGEABILITY AND MAINTAINABILITY</p>	<p>Remarks (+ Best Practices where noted)</p>
<ul style="list-style-type: none"> • Environment Complexity • Availability of Talent/Resources familiar with the HA/DR Technology • Reliability and resiliency of the HA/DR framework • Monitoring ability of the HA/DR framework • Interoperability and coexistence with other necessary infrastructure 	<p>→ This broad category of requirements is difficult to quantify or measure, but crucial to note that an easy to manage and maintain environment is less prone to failure. And when failure does occur, MTTR, RPO, and other major metrics are indirectly improved due to increased ability to diagnose and resolve problems under the stress.</p>

<ul style="list-style-type: none"> • Adaptability to changing infrastructure • 	
Total Cost of Ownership (TCO)	Remarks (+ Best Practices where noted)
<ul style="list-style-type: none"> • Pricing • Cost of skilled talent and resources • Ramp-up or Training Requirements (time and \$) • Inefficiency and Erroneous Processing Costs (goes back to service level compliance, lost opportunity, etc if HA/DR framework is ineffective) 	<p>→ For today's high-end IT systems, the initial licensing/acquisition costs are only a fraction of the total costs required to own and maintain the systems. This is especially true for DR solutions that must provide adequate protection from the outages typical in the enterprise. For example, a DR solution may offer low licensing costs, yet – if it is unable to protect the enterprise from critical outages that may in turn cause the enterprise significant costs of downtime, the total costs for owning and maintaining such a system may run into millions of dollars. Because of this, along with management costs, downtime costs must also be considered while doing any TCO analysis of a DR solution.</p>

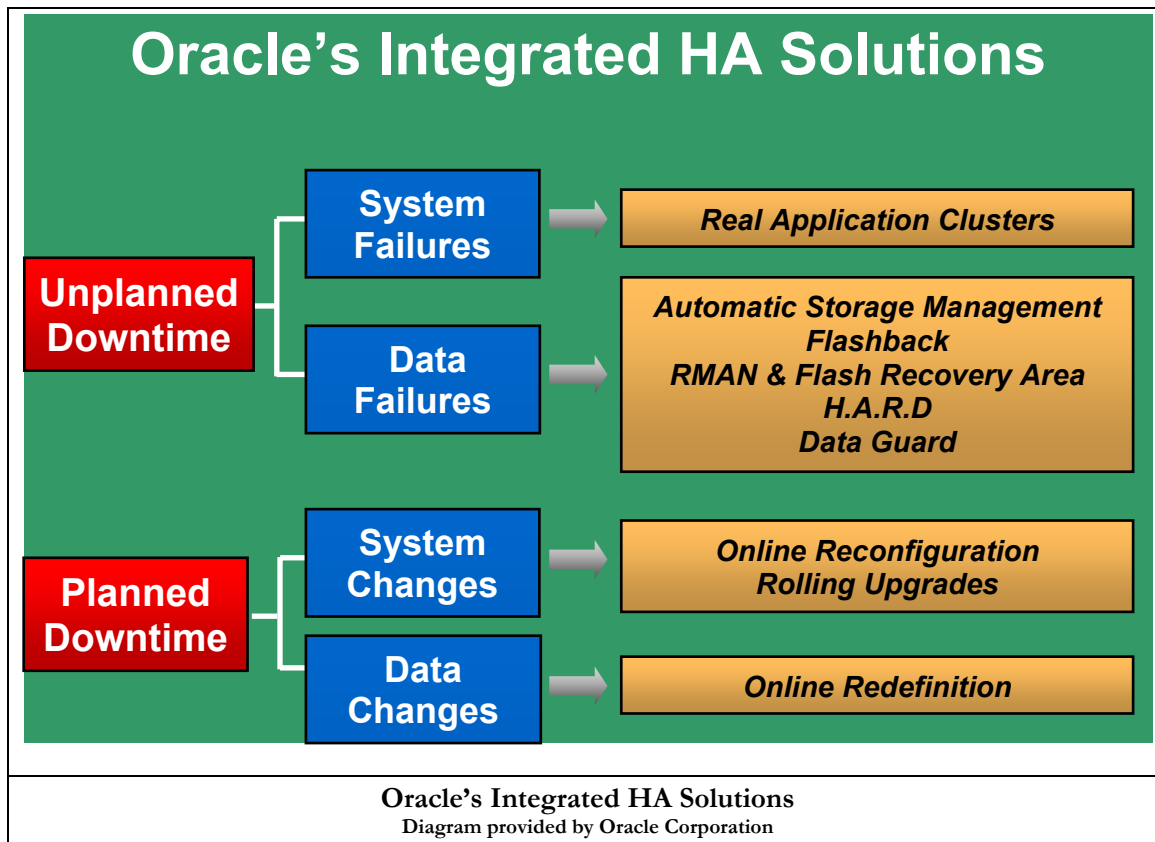
ORACLE HA/DR FEATURES, STRENGTHS, AND WEAKNESSES

Oracle has established a high availability best-practices blueprint called Maximum Availability Architecture (MAA), which aims to maximize system availability while reducing the design complexity of an optimal HA architecture. Oracle 9i and 10g HA/DR features and products are robust and efficient for handling most forms of planned and unplanned downtime. Oracle leverages several technologies to address the increasing complexity of managing and protecting ever-growing databases, and in providing optimal levels of business continuity.

Generally speaking, Oracle MMA provides three types of solutions, including *Data Guard*, *Oracle Real Application Clusters (RAC)*, and *data protection technology* including Recovery Manager and Flashback Technology. Figure 1 illustrates comprehensive maximum availability architecture; it includes primary and secondary sites where each site uses a multiple nodes of RAC. Clients are initially routed to the primary site until a failover or switchover occurs in case of site failure. The RAC server nodes on both primary site and secondary site are used to protect from operating system, database instance, and network failures where the applications can still survive and other infrastructure remains basically static. With Data Guard switchover and failover functions, the roles between primary and secondary sites can be easily reversed short or long term.



The following diagram and associated table present Oracle 9i/10g advanced features and benefits in support of MAA and general HA/DR principles. Some vital best practices are included in the table, although they are not necessarily comprehensive. Additional best practices are included in various sections of this paper.



HA/DR FEATURE	BASIC FUNCTIONALITY (and Specific HA/DR Strengths & Weaknesses)
RECOVERY MANAGER (RMAN) AND FLASH RECOVERY AREA	Oracle utility used to backup, restores, and recover database. In Oracle 10g, RMAN combined with Flash Recovery Area, automate backup to disk for more reliable backup and recovery
Strengths	<ul style="list-style-type: none"> • Optimized backup time and disk space: <ul style="list-style-type: none"> ○ Backup only used blocks. ○ Incremental and differential backups — ability to backup only changed blocks and to merge blocks to reduce number of backup pieces. • Resumeable backup/restore in case of failure — saves time and decreases MTTR. RMAN picks up where it left off (after the last successfully processed file). • <i>Online</i> block corruption detection and repair — increases availability and decreases or eliminates MTTR. Recovers only what is damaged (versus entire file) while keeping the system available for both reads and writes (to undamaged data). Minimal intrusiveness on production environment for both availability <i>and</i> performance since network traffic and redo activity is limited to the recovered blocks. • Comprehensive historical reporting — some through RMAN, complemented by Enterprise Manager and Oracle dictionary RMAN catalog and v\$ views. • Centralized, operating system independent management of backups, recovery procedures, and supporting infrastructure, e.g. backup scripts, configuration options, retention policies, etc). Supports one or many databases on heterogeneous os platforms. • Fine-granular data operations at tablespace, data file, archive log, control file, and block levels. • Fine-granular time based operations, e.g. recovery to present, specific point in time, or specific system change number (SCN). • Extensible to third party storage media managers, e.g. to access tape libraries and integrate with enterprise

HA/DR FEATURE	BASIC FUNCTIONALITY (and Specific HA/DR Strengths & Weaknesses)
	<p>backup systems, etc. Native interface options for products such as EMC NetBackup, IBM Tivoli Storage Manager (TSM).</p> <ul style="list-style-type: none"> ➔ Emerging support for interfaces with third party DR replication products such as remote mirroring technology which synchronizes periodic database quiesces (or online checkpoints) with propagation snapshots to ensure or emulate write order fidelity. The periodic “snaps” or other application aware features (e.g. awareness that payroll processing just completed or another batch of 1000 invoices was just entered) can also be synchronized with backup generation, e.g. through RMAN interface with the DR software. • BEST PRACTICE: Backup control files at least daily, along with whatever level of daily RMAN backup makes sense for your environment (e.g. full, incremental, or differential). While RMAN creates a control file backup, it should ideally be supplemented with a plain text file version using the SQL*Plus command “alter database backup controlfile to trace”. A simple textual version facilitates planning and diagnostics under duress (e.g. when a failure occurs and you need to see everything quickly in one tidy place). <hr/> <p><u>New for 10g:</u></p> <ul style="list-style-type: none"> • Integration with Oracle Flash Recovery Area yields the following benefits: <ul style="list-style-type: none"> ○ Simplified management of retention policy, space reclamation, etc. ○ Simplified and improved sustainability of backup media through intelligent disk consumption monitoring. Automatic notification can be configured via RMAN command-line or Enterprise Manager GUI to notify and help automate certain Flash Recovery Area states, e.g. low disk space and obsolescence of expired backup sets. • Increased fault tolerance, e.g. resilient to backup/restore streaming errors — RMAN continues processing on parallel/failover channel(s). • Automatic file restore/recovery fallback — if backup file is corrupted, RMAN automatically retrieves from alternate valid backups before returning an error. • Improved time and space savings: <ul style="list-style-type: none"> ○ Incremental backup aggregation drastically decreases MTTR. Incremental backups can now <i>cumulatively</i> update the baseline backup pieces (data files), so recovery eliminates extra time applying the incrementals. ○ Recovery of tablespace by cloning is now fully automated, decreasing MTTR. RMAN automatically creates clone instance (on same server as defective database), and automatically removes it after tablespace recovery. ○ Ability to control intrusiveness on production database, network, etc. RMAN backup “duration” option can prioritize backup performance versus impact on system load to comply with service level requirements which often accompany availability policies (e.g. in your organization’s SLA). ○ Better database block change tracking allows incremental backups to intelligently read and record only the deltas, versus performing full file scans. ○ Binary compression saves 50-75% backup set disk space. <i>WARNING:</i> can increase MTTR.
Weaknesses	<ul style="list-style-type: none"> • Protection is limited to individual databases — backup and recovery (although clone recovery may be directed to alternate filesystems and/or servers). For high availability environments, use RMAN in conjunction with advanced HA/DR replication oriented technologies such as Data Guard and/or third party products.
AUTOMATIC STORAGE MANAGEMENT (ASM)	<p>Integrates filesystem, disk volume management, and database file management into a single tool which automates the ongoing maintenance of potentially hundreds or thousands of database files, making them virtually transparent to the DBA (although initial configuration and periodic expansion and maintenance tasks require manual intervention). ASM is an entirely new feature set for Oracle 10g.</p>
Strengths	<ul style="list-style-type: none"> • ASM introduces the absolutely crucial and long neglected concept of Failure Groups. Most third party mass storage configurations suffer from one critical problem: there is no guarantee that the same physical disks are not shared in any way across the logically distinct volumes or filesystems conveyed to the DBA. Physical disk separation is vital to both availability and performance, although many mass storage units are powerful and fault tolerant enough to hide certain inefficiencies. The bottom line, however, is that if you place a redo log (for example) on one filesystem and its mirrored counterpart on another, there is usually

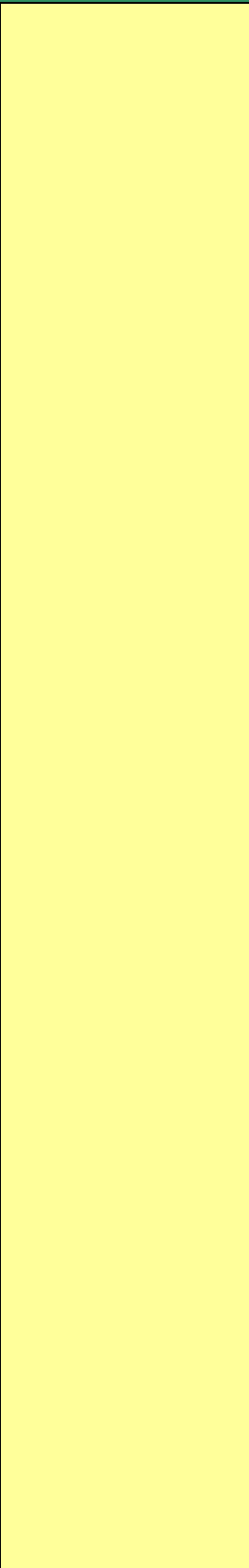
HA/DR FEATURE	BASIC FUNCTIONALITY (and Specific HA/DR Strengths & Weaknesses)
	<p>no guarantee that a single disk or array failure will not impact both files. Obtaining such assurance, if at all possible, typically takes many hours or even days of intense “discussion” between DBAs, system administrators, and disk storage administrators who must be intimately familiar with specialized jargon like physical and logical volume allocation, logical to physical volume ratios, stripe depth, stripe width, and hypersplits.</p> <p>Enter ASM’s failure group. If all the disk storage is managed entirely by Oracle down to the individual disk level, Oracle guarantees that all file extents in one failure group never share the same physical storage as those of another failure group. Failure groups are established as a subset of disks within a disk group.</p> <ul style="list-style-type: none"> • ASM storage is dependent on a single point of failure to store its configuration metadata — a separate ASM specific database instance. Unlike third party volume managers, however, the configuration information inherits the various HA/DR mechanisms available for Oracle databases. • ASM three basic redundancy configurations which can be set differently for each disk group: <ol style="list-style-type: none"> 1) External Redundancy disables ASM fault tolerance control. All redundancy is provided externally by third party vendor. 2) Normal Redundancy offers two way mirroring provided by ASM. This is the default disk group setting. 3) High Redundancy provides three way mirroring. It is currently the highest degree of fault tolerance provided by ASM. • ASM is pre-tuned for basic database awareness as far as disk geometry is concerned. Like a basic third party volume manager, it automatically distributes I/O across the available disks in a disk group, but unlike third party products, it automatically takes DBMS characteristics into consideration. For example, the smallest contiguous disk space within ASM is 1MB for most if not all platforms. This is a defacto standard chunk or stripe size for databases — ASM calls it the allocation unit (AU). Unfortunately, AU is not an overrideable setting. • ASM simplifies management of database storage. While this is not an HA/DR specific strength, ease of use can improve reliability of your environment by reducing mistakes and inconsistencies. It can also improve MTTR for various downtime situations. • ASM files can be managed as either cooked storage (filesystem level) or raw devices. It can even manage filesystem data files created outside of ASM. • While ASM hides the complexity of underlying disk storage characteristics and even file names during daily administration, that information (or metadata) can be viewed via ASM database dictionary views, and some of it can be controlled as well. In this way, the people who need to see major disk configuration details the most (e.g. DBAs and possibly system administrators) have convenient access when they need it. Moreover, the DBA can provide policy templates to help automate file naming and configuration options such as striping and mirroring, so various kinds of files automatically receive the appropriate characteristics. For example, redo and archive logs may be set for mirroring but not striping, versus other data files which would typically implement both mirroring and striping. The DBA may also (optionally) create a hierarchy of alias names to help mnemonically distinguish different kinds of ASM files from one another, since the actual file names are essentially numeric. The bottom line here is that ASM isolates administrators from unnecessary detail, while still allowing certain control and visibility under the hood. <hr/> <p><u>New for 10g:</u></p> <ul style="list-style-type: none"> • ASM is an entirely new feature set for Oracle 10g.
Weaknesses	<ul style="list-style-type: none"> • Manages only database storage. All other files such as binary executables, configuration files, alert and error log files, source code, DDL, and batch process scripts are not manageable through ASM. • For mass storage technology (e.g. EMC Symmetrix, IBM DataShark, etc.), while there is certainly reduced need to meticulously tune underlying disk geometry for performance and capacity planning, high throughput environments still require some low level configuration. Someone still has to deal with settings like logical-to-physical partition ratios, stripe width/depth, hypersplits, etc.
FLASHBACK TECHNOLOGY	<p>Oracle 10g leverage several technology features to support viewing and rewinding data back and forth in time and provide the fastest and most efficient method of recovery from potential disasters.</p>

HA/DR FEATURE	BASIC FUNCTIONALITY (and Specific HA/DR Strengths & Weaknesses)
<p>Strengths</p>	<ul style="list-style-type: none"> • Accessible from familiar interfaces (Oracle SQL*Plus and RMAN) and can be automated through batch scripting. • Flashback Query first introduced in Oracle 9i. • Provides limited protection from logical corruption without addition of more complex replication technologies such as Data Guard, Streams/Advanced Replication, or third party products. <hr/> <p><u>New for 10g:</u></p> <p>The following new features make it even simpler to recover from a variety of logical and even physical corruption scenarios, substantially decreasing MTTR and increasing availability during object restoration (e.g. online Flashback Table feature):</p> <ul style="list-style-type: none"> • Now supports recovery at database, table, row, and transaction levels. • Flashback Database: Effectively REWINDS entire database to previous state (timestamp or SCN). <ul style="list-style-type: none"> ➔ Data Guard can be used in conjunction with Flashback Database to optimize MTTR: To provide protection against logical corruption, many standby databases are configured to delay application of the archive logs (by lagging physical recovery or logical SQL). To failover to a point just prior to failure therefore requires additional recovery time. In 10g, you can use flashback database as an alternative, so MTTR would be fastest for most recovery scenarios and slower in the less likely scenario of rewinding to a point before failure or logical corruption. • Flashback Table: Online object level restoration — includes sub-objects such as indexes, constraints, and triggers. After flashback is complete, can even revert table to original state — good for troubleshooting and diagnosing logical corruption. • Flashback Drop: Can UNdrop a variety of objects until “PURGE” command is used or automatic reclamation event (e.g. when user exceeds quota or tablespace needs to auto-extend). Handles tables, indexes, constraints, triggers, etc. Can see and use dropped objects, e.g. look in RECYCLEBIN and access via BIN\$\$tablename (BIN\$\$ is prepended to dropped object names). • Flashback Query can now query data older than 5 days by setting UNDO_RETENTION init parameter. (Should also set UNDO_MANAGEMENT = AUTO and, to guarantee that unexpired undo is not overwritten, set RETENTION_GUARANTEE for the undo tablespace.) • Flashback Versions Query: AMAZING diagnostic/auditing feature which facilitates recovery from logical corruption or construction of audit trail information. Use new VERSIONS BETWEEN clause in regular SQL SELECT query to retrieve all committed versions of rows that existed in a table during a specific time period or within a specific SCN range. Using VERSION* metacolumns (similar to ROWID metacolumn), a Flashback Versions Query also lists the transaction id which instantiated each row version, the DML type that manipulated the row version (e.g. insert/update/delete), and the first and last SCNs of each row version. • Flashback Transaction Query: Once Flashback Versions Query has identified one or more responsible transaction ids, you can use Flashback Transaction Query to further investigate the suspect transaction(s). Use the FLASHBACK_TRANSACTION_QUERY view to show the full impact of a single transaction across multiple tables. The view also lists one or more specific SQL statements required to repair or undo all or part of a transaction.
<p>Weaknesses</p>	<ul style="list-style-type: none"> • Protection is limited to individual databases (although restoration of partial data such as individual tables or queries). • Database/storage failure or corruption of flashback media obstructs usefulness of this feature since restorable data is collocated in and dependent on the subject database. <ul style="list-style-type: none"> ➔ Use in conjunction with RMAN and possibly more advanced HA/DR replication oriented technologies such as Data Guard and/or third party products.

HA/DR FEATURE	BASIC FUNCTIONALITY (and Specific HA/DR Strengths & Weaknesses)
REAL APPLICATION CLUSTERS (RAC)	Real Application Clusters (RAC) enables near instantaneous instance and host failover. Combined with a connection failover implementation, RAC enables clients to transparently failover in seconds.
Strengths	<ul style="list-style-type: none"> • High availability to provide near continuous access to data with minimal interruption in case of system/server failure or database instance crash. • Additional availability protection as follows: <ul style="list-style-type: none"> ○ Provides application resilience through Transparent Application Failover (TAF) support. ○ Maximum, unrestricted uptime when “scaling out” within a cluster (when adding horsepower by adding new server nodes to an existing cluster). This is primarily because Oracle uses a “Shared Disk” clustering architecture versus “Shared Nothing” approach which requires data to be redistributed as additional server nodes are added to the database cluster. Shared Nothing scaling may even require modification of certain application code (planned downtime). • High scalability and load balancing across clustered servers connected by Oracle’s Cache Fusion technology and appropriate clusterware provided by Oracle or third party (depending on platform and preferences). <hr/> <p><u>New for 10g:</u></p> <ul style="list-style-type: none"> • Zero Downtime Rolling Upgrades. Starting with Oracle 10g, Oracle Corporation patches are now labeled as certified or not certified for rolling upgrades for RAC environments. For certified rolling upgrades, you can shutdown one or more nodes for patch application while other node(s) remain running. • Fast Connection Failover automates the coordinated failover of application server sessions. Initial support is only for JDBC Implicit Connection Cache and Oracle Application Server 10g. (Support for additional mid-tier products and components in the future.) • Integrated Clusterware improves cost-benefit (ROI). Oracle Clusterware (OCR) replaces or can optionally complement third party clusterware. If you are using non-ethernet cluster interconnect or other applications are dependent on third party clusterware, you may want to continue using it. • Automatic Workload Management can be used to enhance performance through more effective resource distribution. Application workloads can be defined as named services, which can be managed in groups by Resource Manager. System resources can be allocated as needed across the named services. • Improved monitoring ability helps prevent reliability and performance problems. As part of automated workload management (above), alerts can be set at various thresholds for different named services. • Miscellaneous performance enhancements, e.g. via minimized messaging traffic, and reduced utilization of memory and other resources. • Miscellaneous functionality improvements (not directly related to HA/DR): <ul style="list-style-type: none"> ○ Enhanced diagnostic and troubleshooting tools can improve MTTR, although their focus is primarily for initial installation/configuration (e.g. improved cluster verification configuration tool). If failing over long term (e.g. role reversal) from RAC to single instance and then re-building a clustered environment to accommodate necessary performance load, these would be considered HA/DR improvements. ○ Simplified GUI administration — OEM’s Cluster Database Page provides a single system status view spanning multiple server nodes. The DBA can drill down to see details for individual instances.
Weaknesses	<ul style="list-style-type: none"> • Protects only server computers and their individual connectivity — no protection from media/disk failure or larger scale network or site failure (see next bullet). • Protection focus is within same site only, although geographically disperse clustering is possible under limited scenarios (most significant restriction is throughput requirements across cluster interconnect). • Even with geographically disperse clustering (aka global clusters), there is dependence on central shared disk storage. So complete HA/DR solution would require additional advanced architecture, e.g. Data Guard and/or third party replication products.

HA/DR FEATURE	BASIC FUNCTIONALITY (and Specific HA/DR Strengths & Weaknesses)
<p>DATA GUARD</p>	<p>Oracle Data Guard sustains a synchronized copy of the production database at one or more secondary sites for protection from server/site outages and optionally for offloading of read-only functionality such as reporting. Data Guard manages the two databases by providing remote archiving, managed recovery, failover, and planned switchover.</p> <p>Data Guard may be maintained using three different Oracle provided tools: SQL*Plus (of course), Data Guard Broker command-line interface which simplifies management using higher level commands (dgmgrl), and OEM's GUI Database Control web console which depends on Data Guard Broker under the hood.</p>
<p>Strengths</p>	<ul style="list-style-type: none"> • Disaster Protection from site destruction. • Protection from data failures. • Capability to use the standby database for reporting. <ul style="list-style-type: none"> ○ Unrestricted 24x7 access using Logical Standby SQL Apply technology. ○ Physical Standby has time restrictions (cannot keep database open 24x7), and read-only restriction may impede reports which require creation of temporary work or staging tables. • No data loss or minimum data loss options with primary tradeoffs balancing increased availability versus performance, potential data loss, and cost of inter-site communications throughput: <ul style="list-style-type: none"> ○ Maximum Protection provides Zero Data Loss with Double/Multi-Site Failover Protection via Synchronous redo log shipping to two or more sites. LGWR simultaneously commits to both local and standby redo log (SRL) for at least one standby instance. <ul style="list-style-type: none"> ➔ IMPORTANT: Primary database activity <i>ceases</i> (shuts down) if last standby is unavailable. Also see <i>Improved RAC Support</i> below for increased protection when combining Maximum Protection mode with a clustered standby database. Command: ALTER DATABASE SET STANDBY TO MAXIMIZE PROTECTION ○ Maximum Availability provides Zero Data Loss with Single Failover Protection via Synchronous redo log shipping to one site. LGWR simultaneously commits to both local and standby redo log (SRL) for at least one standby instance, but the primary database does NOT fail if a standby redo log is unavailable. <ul style="list-style-type: none"> Command: ALTER DATABASE SET STANDBY TO MAXIMIZE AVAILABILITY ○ Maximum Performance provides Minimal Data Loss with Single <i>or</i> Multi-Site Failover Protection via Asynchronous redo log shipping to one or more sites. Using an intermediate (asynchronous) Logwriter Network Server (LNS) process, writes to the standby redo log (SRL) occur independently from writes to local redos, so the primary database LGWR runs without the direct performance impact of synchronous propagation. <ul style="list-style-type: none"> Command: ALTER DATABASE SET STANDBY TO MAXIMIZE PERFORMANCE • Data protection tolerance to disk level block corruption (which is a classic susceptibility of remote disk mirroring technologies). • Can offload backups from production by running against standby database. <ul style="list-style-type: none"> ➔ WARNING: Not necessarily true for logical standby if you have customized indexes and other supporting infrastructure, e.g. to improve reporting performance. <hr/> <p><u>New for 10g:</u></p> <ul style="list-style-type: none"> • Real Time Apply feature now scans shipped redo logs continually, versus waiting for log shift. Significantly improve MTTR for failover scenarios. Real Time Apply is supported for both Physical <i>and</i> Logical Standby, although the commands to instantiate are different for each: <ul style="list-style-type: none"> ○ Physical Standby (Redo Apply): ALTER DATABASE RECOVER MANAGED STANDBY DATABASE USING CURRENT LOGFILE ○ Logical Standby (SQL Apply): ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE • Flashback Database Integration -- see Flashback Technology Strengths (in this section) for methods of decreasing MTTR when Flashback Database is used in conjunction with Data Guard.

HA/DR FEATURE	BASIC FUNCTIONALITY (and Specific HA/DR Strengths & Weaknesses)
---------------	--



- **BEST PRACTICE:** Improved RAC Support for Broker-enabled configurations (preferable to administering Data Guard through SQL*Plus). Remember that the OEM web console leverages Data Guard Broker (transparently), so the following benefits are achieved when using either interface — but not when using only SQL*Plus.
 - If failover database is RAC and one standby apply instance fails, log shipping and apply services automatically failover to a different standby RAC instance. DMON, the Data Guard Monitor background process for broker-managed instances, detects the failure and catalyzes the Broker to make the changes after waiting for a configurable timeout period (current default is 2 minutes). This intelligent failover works generally the same regardless of the Data Guard protection mode, with one notable exception. When using Maximum Protection and the apply instance failure causes the primary database's LGWR process to fail, the Broker does not wait for the timeout period before switching log services.
 - ➔ **TIP:** If Data Guard Broker is not enabled, some of this intelligence can be configured by other means. The tnsnames.ora file should contain multiple host/server nodes in the ADDRESS_LIST section. LGWR will then failover to the next node in the list after a timeout period (NET_TIMEOUT attribute of the LOG_ARCHIVE_DEST_N init parameter). Unfortunately, you must manually start the Redo/SQL Apply process on the failover node using SQL*Plus.
 - RAC now integrates with Data Guard Broker and OEM's Database Control web console. (In 9i, if using Data Guard Broker or OEM web console for administration, write order across redo threads was not ensured and therefore not supported.)
- Logical standby now officially supports rolling upgrades for database patch sets, major releases and even for hardware, OS, and (where applicable) clusterware.
- Zero Downtime Initial Standby Instantiation can be achieved without quiescing or shutting down the primary database to enable Resource Manager when hot backup is made. Don't even need to record SCN. All this info is now built-in to the standby control file instead.
- Fail/Switchover using Flashback Technology:
 - FailBACK: Flashback Database can be used to rewind primary database and then apply archive logs from the now-activated standby database.
 - SIDE NOTE about Flashback Database: Not propagated to standby even if operation forced logging (init parameter) enabled, so must manually replicate when relevant.
 - Switchover = *Planned* Role Reveal (primary changes to standby and vice versa).
 - Switchover/Switchback: Zero Downtime Switchover Instantiation can be achieved using Switch Database feature:
 - alter database prepare switchover to primary/standby
 - alter database commit to switchover... (must perform asap after prepare!)
 - alter database start logical standby apply
 For more, see *RMAN Fast Recovery* in Scenarios section of this paper (Media/Storage Failure).
- Increased data protection — additional data types now supported for Logical Standby (were already supported for Physical Standby):
 - E.g. Multi-byte CLOB, NCLOB, LONG, LONG RAW, BINARY_FLOAT, BINARY_DOUBLE, IOTs without overflows and without LOB columns.
- Improved security — Data Guard now utilizes authenticated network sessions to propagate redo log shipments. Oracle Advanced Security may optionally be used to implement encryption and integrity checksums when transferring logs across the network.
- Improved monitoring and diagnostic views:
 - See all tables with columns that cannot be propagated by logical standby:

```
SELECT DISTINCT TABLE_NAME, ATTRIBUTES (new column)
FROM DBA_LOGSTDBY_UNSUPPORTED WHERE OWNER = 'SCOTT';
```
 - See which log files have been applied to the standby database:

```
SELECT THREAD#, SEQUENCE#, APPLIED (new column)
FROM DBA_LOGSTDBY_LOG ORDER BY SEQUENCE#;
```
 - Monitor ongoing progress of SQL Apply down to the SCN level:

```
SELECT APPLIED_SCN, APPLIED_THREAD#, NEWEST_SCN, NEWEST_THREAD#
FROM DBA_LOGSTDBY_PROGRESS;
```

HA/DR FEATURE		BASIC FUNCTIONALITY (and Specific HA/DR Strengths & Weaknesses)
		<ul style="list-style-type: none"> Improved reliability and resilience to un-appliable transactions. Can now simply add the SKIP FAILED TRANSACTION CLAUSE to Alter Database Start Logical Standby Apply statement to automatically skip the last failed transaction and restart SQL Apply processing. (Used to require issuing explicit skip transaction processing and manual restart of SQL Apply.) Improved usability — helps improve overall reliability when things are easier to maintain: <ul style="list-style-type: none"> Data Guard Broker command line interface (DBMMGRL) has significantly expanded, refined, and simplified command set. Also note terminology changes, e.g. Enable/Disable/Alter <i>Resource</i> replaced with Enable/Disable/Edit <i>Database</i>. Simplified web browser interface (Database Control console within OEM).
Weaknesses		<ul style="list-style-type: none"> Replication is only unidirectional — does not support bidirectional (multi-master). Does not propagate anything outside the database, e.g. product binaries (executables), configuration files for database/os/etc, application source code, DDL, batch process scripts, etc. Logical standby does not propagate tables containing any column with unsupported certain data types, e.g. BFILE, RowID, URowID, User-Defined Types (UDTs), Object Type REFS, VARRAYs, nested tables, tables using data segment compression, and IOTs with overflows or LOB columns.
STREAMS AND ADVANCED REPLICATION		<p>Oracle advanced replication has traditionally <i>not</i> been viewed as HA/DR technology, although it has been Oracle's only method of fulfilling bidirectional replication and multi-master requirements involving conflict avoidance and resolution, etc. (Technically it is possible and often preferable to have bidirectional replication without multi-master complexities of updating the same rows in the same tables at approximately the same time across multiple servers.)</p> <p>Even when used for unidirectionally, advanced replication tends to yield unacceptable data loss and maintenance complexity which often sacrifices both quantity and quality of failover data.</p> <p>➔ With the growing maturity of newer, faster Streams technology (significantly improved in 10g), using bidirectional (multi-master) replication for high availability purposes becomes more feasible. Multi-master replication maximizes the utility (ROI) of expensive, redundant DR computers and related infrastructure by enabling pure load balancing across multiple sites (versus Oracle Data Guard, for example, which at best is read-only for the secondary databases).</p> <p>Streams leverages log scraping technology (aka log mining or log sniffing) versus much more intrusive internalized triggers used by conventional advanced replication. Unlike third party alternatives, however, Streams puts captured data back into the Oracle database for propagation to target(s) using Oracle Advanced Queuing mechanisms (AQ). While this at first seems more intrusive on the production database, and possibly more time consuming versus externalized queuing, Oracle places Streams data in special memory resident queues versus the slower disk based, persistent queues used for conventional AQ messages.</p>
Strengths		<ul style="list-style-type: none"> May improve ROI. Multi-master load balancing across sites can improve cost benefit of redundant disaster recovery computers and related equipment. (Primary tradeoff is added complexity.) Can actually SIMPLIFY environment complexity if already using replication for other business requirements. May be easier to use the same technology versus adding heterogeneous architecture for HA/DR purposes. Advanced Replication 9i (and 10g) can replicate the following objects and data types: <ul style="list-style-type: none"> Tables, Indexes, Views, Synonyms, Triggers Packages, Procedures, Functions Advanced Data Types: <ul style="list-style-type: none"> User-Defined Types (UDTs), Index Types Tables with column objects, object tables Nested Tables, VARRAYs Streams 9i can replicate the following data types: CHAR, NCHAR, VARCHAR2, NVARCHAR2, NUMBER, DATE, RAW, BLOB, CLOB (fixed width character sets only), TIMESTAMP, TIMESTAMP WITH TIME ZONE, TIMESTAMP WITH LOCAL TIME ZONE, INTERVAL YEAR TO MONTH, INTERVAL DAY TO SECOND. <ul style="list-style-type: none"> ➔ Also see Streams 10g additions to this list further below. Both Streams and Advanced Replication offers automatic conflict detection and resolution, with ability to customize. Out of the box routines include latest or earliest timestamp, maximum or minimum value, and site priority. Values can be overwritten or discarded. Unresolved conflicts are recorded in exception

HA/DR FEATURE	BASIC FUNCTIONALITY (and Specific HA/DR Strengths & Weaknesses)
	<p>queue.</p> <ul style="list-style-type: none"> • Can tap into Streams capture/staging/apply data flows using open programming interfaces, e.g. JDBC, C/C++, PL/SQL (of course), JMS, SOAP (XML/HTTP). • <u>BEST PRACTICES for 9i Release 2:</u> <ul style="list-style-type: none"> ○ Ideally, avoid 9i Streams (even 9iR2). If you are compelled to use it, follow the additional bullets below. ○ Use separate queues for capture and apply (separate queues for each database). ○ Increase shared_pool_size init parameter. You'll probably have to waste a lot of memory since Streams uses only a maximum of 10 percent of the general purpose shared pool. This ratio is not overrideable. ○ Move the Streams dictionary from the system tablespace, e.g. to a SYSAUX tablespace as in 10g. ○ Implement flow control — see Oracle MetaLink paper 259609.1. ○ Eliminate duplicate rules in a rule set. (NOTE: An empty rule set is not equivalent to having no rule set defined.) ○ Use same method to remove rules as was used to create them. Don't switch between DBMS_STREAMS_ADM and DBMS_RULE_ADM when administering the same rules. ○ To ensure that objects with unsupported components or data types are not touched by Streams: When using Global rules, add the entire Streams administrator schema (which contains queues) to the negative rule set. Do the same for application schema tables containing unsupported data types. ○ Before deleting old archive logs, make sure they are not needed for Streams capture process which may have a backlog. ○ If replicating DDL, modify any manual hot backup scripts to set an apply tag before starting the backup. Also, configure the instantiation SCN at next higher level. (This entry contains exact wording provided by Oracle Corporation.) ○ Implement a "heartbeat" table and update it periodically. (This entry contains exact wording provided by Oracle Corporation. @TBD if this is to avoid network timeouts or for some other reason such as measuring latency.) <hr style="border-top: 1px dashed black;"/> <p><u>New for 10g Streams:</u></p> <ul style="list-style-type: none"> • Zero Impact Performance using Downstream Capture: Allows a separate Streams processing database to capture (mine) logs from your primary database, so there is no performance impact on the primary database. • Improved integration with RAC: Active-passive implementation with automatic failover from primary to secondary database instance. • New data dictionary views enable better monitoring, to help increase availability and reduce data loss, e.g. by detecting queue build-up which would result in lost in-flight transactions in the event of database failure. • Transportable tablespaces and RMAN are now supported instantiating Streams target databases, decreasing planned downtime during initial configuration as well as some re-synchronization scenarios. • Improved scalability via new STREAMS_POOL_SIZE init parameter which separates Streams shared memory data from Oracle's general purpose shared pool. (Streams used to be limited to 10 percent of the shared pool, often requiring overallocation of shared pool memory in order to provide adequate Streams support.) • Data type support has been expanded to include: NCLOB, BINARY_FLOAT, BINARY_DOUBLE, LONG, and LONG_RAW. → Also see Streams 9i supported types further above. • Expanded capture and apply functionality now supports index-organized tables (IOTs), function-based indexes, and descending indexes. • Hot mining of current redo log (not just archive logs). • Directed and Cascaded Replication: Directed Networks feature enables data to cascade through a pass-through site to one or more final targets based on rules based publish and subscribe routing instructions which operate on the staging queues.

HA/DR FEATURE	BASIC FUNCTIONALITY (and Specific HA/DR Strengths & Weaknesses)
	<p>→ This feature provides a means of provisioning data across a GRID :-).</p> <ul style="list-style-type: none"> Improved rules and queue configuration support. For example, subset inclusion and exclusion rules can now be defined for all queuing levels: capture, staging/propagation, and apply. Subsetting provides a means of segmenting or distributing data across one or more target databases. Enhanced segmentation means wider, albeit more complex, DR scenarios can be supported. While this practice is more replication versus HA/DR focused, it can be useful particularly when budget constraints preclude a one-to-one failover site architecture. That is, when a single DR site cannot support the entire load of a one primary site, the data (and corresponding load) may be partitioned across multiple DR sites. Numerous useability improvements. Improved manageability can help improve reliability and possibly hasten MTTR for recovery scenarios depending on Streams replication. BEST PRACTICES for 10g Release 1: <ul style="list-style-type: none"> Use separate queues for capture and apply (separate queues for each database). Configure streams_pool_size init parameter so that Oracle's general purpose shared pool is not used (default is 10 percent of shared pool). Make sure you don't run out of SGA memory. If sga_max_size init parameter is set, increase it as needed. <hr/> <p><u>New for 10g Conventional Replication:</u></p> <ul style="list-style-type: none"> While there are improvements in materialized view functionality and supplemental logging, there are no significant improvements in conventional advanced replication. Streams replication, on the other hand has undergone serious upgrades (see above).
Weaknesses	<ul style="list-style-type: none"> Data latency and, therefore, likely data loss can be high in advanced replication environments with high transaction throughput, e.g. more than 600 to 800 transactions per second throughput (extremely rough rule of thumb which depends on many factors such as server/CPU power, network bandwidth, distance between sites, record sizes, data types being replicated, etc). <ul style="list-style-type: none"> → Streams is generally faster than advanced replication so tends to have less data latency/loss. Unfortunately, while Streams is maturing, it is still much more difficult to administer, monitor, and maintain. Data latency for advanced replication is still technically dependent on job_queue_interval init parameter minimum allowable value of 60 seconds. Although UNdocumented _job_queue_interval allows 1 second intervals between data transfers, practical experience indicates intervals less than 2 to 5 minutes result in system thrashing (latency backlog) for moderate to high throughput environments. Advanced Replication does not replicate LONG or LONG RAW data types, and probably never will :-). Streams does not replicate tables containing any column with unsupported data types including. BFILE, Variable width multibyte character set CLOB, RowID, URowID, Simple and nested User-Defined Types (UDTs), Collections (e.g. REFs, VARRAYs, and Nested Tables), Object Refs, and XML Type. Streams does not replicate the following kinds of database objects: <ul style="list-style-type: none"> Index Organized Tables (IOTs), function based indexes, temporary objects, external tables, Create Table As Select of a table with a clustered key, Oracle workflow tables and workflow queues, Transportable Tablespaces, Materialized View Tables (Hint: Create the MV using PREBUILT TABLE clause so Streams will capture data from the MV's base table.). Moreover, Streams will not capture changes to objects in the following schemas: SYS, SYSTEM, and CTXSYS Restrictions on which data types can be replicated, e.g. no "long" data types or abstract data types (yet). While long data types have basically been replaced by LOBs, they are still used in many older systems. Abstract data types are not abundantly used yet, but are growing in popularity. Oracle will probably evolve more support for abstract data types. Replicated tables <i>must</i> have a primary key (PK) with corresponding unique index. While this is documented as a weakness, most if not all applications tables with logical business content should have a PK. Some third party products, like SharePlex, automatically compare all table columns (except LOBs and LONGs) if a PK is not present. The primary compromise for this flexibility is performance. Complex administration/maintainability — requires advanced (expensive) skills and monitoring or can lead to excessive out-of-sync conditions, sacrificing integrity of failover environment(s) at both the database infrastructure level (e.g. tablespaces, etc) and application data level (table rows).

HA/DR FEATURE	BASIC FUNCTIONALITY (and Specific HA/DR Strengths & Weaknesses)
GRID CONTROL	Oracle Grid Control Panel provides centralized management console support for a variety of functions spanning the core database and supporting infrastructure such as application servers.
Strengths	<ul style="list-style-type: none"> Improved usability, especially for large environments with many databases and N-Tier environments with multiple technical components working in conjunction with each database. Ease of use and centralized management helps improve overall system reliability for daily maintenance. Also improves diagnostics ability and resolution turnaround time in crisis mode, indirectly improving DR metrics such as MTTR, RTO, and RPO. <p>-----</p> <p><u>New for 10g:</u></p> <ul style="list-style-type: none"> Grid is an entirely new feature set for Oracle 10g.
Weaknesses	<ul style="list-style-type: none"> <i>N/A. No major weakness at this time</i> since use of Grid is an added convenience which may or may not be used, at DBA's discretion. Cost is free for enterprise licensed installations. There is no major competition to Oracle Grid. The manner in which it complements other technologies will grow as more products are supported via Grid Control mechanisms.

BEST PRACTICES FOR BASIC INFRASTRUCTURE

While this paper focuses on advanced HA/DR technologies, utilizing basic technology in specific ways can drastically improve the reliability of your overall environment. This section lists some vital best practices which will help you do just that.

- Note that other sections of this paper elaborate on more advanced HA/DR functionality, some of which are built on top of the basic technologies explained here. For example, Remote Disk Mirroring is a sophisticated add-on for disk storage replication across geographically disperse sites. Relevant best practices for those advanced technologies are listed in those sections.

A FEW SIMPLE RULES FOR HIGH AVAILABILITY DBAS AND SYSTEM ADMINISTRATORS

There are a few basic yet essential guidelines worth bundling together for IT people working on the daily grind of technical activity. These simple rules are essential for both DBAs and system administrators to follow in order to maintain stability and serviceability in just about any highly available environment. They are also quite useful for conventional environments where matters such as performance are a prime imperative even if availability is not. While being primarily database focused, much of the information in this paper applies to all areas of the technical environment, including the vital tips below.

BEST PRACTICES for Performing Planned Maintenance Activity — Four Simple Rules

1. Perform planned maintenance work during windows of low system utilization.
2. Use dedicated maintenance resources to separate or minimize the impact of maintenance work from online production activity.
3. Make sure your capacity plans allow for spikes in both regular production activity as well as maintenance activity — preferably during mid-high utilization periods even though we plan to perform them only during low utilization.
 - NOTE: Many HA environments, especially worldwide applications which run across global time zones, lack a low utilization period.
4. Always monitor critical system resources more attentively during planned maintenance, regardless of the automated monitoring processes you have in place.

Some of these rules may sound rather obvious, but all of these practices — or technically lack thereof — bite many people *much* more often than you would ever imagine. Regarding (1), remember that even simple, untested activity performed in a production environment can severely degrade performance or even bring down the system in an environment which seems generally well tuned.

I personally have been party to a situation where we installed new disk storage in production with an obvious performance boost for online OLTP DBMS support. In short, the new disks were *smokin!* On the same server with the older, “slower” disks, we used to run medium size database imports with no discernable impact on production. Using our new, blazing fast disk storage system, we tried the same thing using file sizes and tuning identical to the past. We did NOT TEST the data load before running it in production during a mid-high utilization period as we always had. It brought system performance to its knees, acutely affecting online user sessions for a brief period until we noticed the impact — thanks to (4) above — and aborted the data load. It turns out there was an inefficiency in the underlying disk geometry configuration and an anomaly (why not just call it a bug!) in the associated firmware level. Bottom line is that the system choked whenever a demanding sequential I/O process was run. Although we didn’t have a real test environment, we could have run the load off-hours.

Lesson learned: whenever any component in your environment changes, do not let speculative, interpolated, or even

seemingly empirical evidence serve as a substitute for testing the *exact* steps and configuration, or at least running the exact steps and configuration off-hours before introducing into a more risky production timeframe.

A simpler example of simple rule (1) is to avoid copying or transferring medium to large files around a production environment, particularly during mid-high utilization periods. It can drastically impede both disk I/O and network performance, unless you have dedicated maintenance infrastructure in place.

This brings us to simple rule (2). Sometimes you just can't wait for off hours or maybe you run a worldwide application which has no discernable off hours utilization window. In these cases and even under simpler circumstances when performing maintenance activity while most users sleep, you should use separate, dedicated resources to help insulate your activity from production bandwidth. This means, for instance, using isolated disk volumes and Virtual LANs (VLANs) or even separate physical networks to copy files around. A database internal example is creating at least one separate temporary tablespace dedicated to Oracle for DBA activity. While you're at it, create another one for intensive batch processes, again, to help insulate users from performance degradation. Take these principles and sprinkle them around your environment.

Simple rule (3) is self explanatory, and (4) already came to our rescue in the disk I/O mishap story above. So I'll leave these simple rules alone for now.

DISK STORAGE

At the most basic level of any highly available framework, redundancy is key — and disk storage is no exception. Before evaluating pricey add-ons, you should make sure that your basic infrastructure is configured for dependability. Consider the following best practices.

BEST PRACTICES for Basic Disk Storage HA/DR

- **RAID:**
Without digressing into the many pros and cons of various levels of RAID (Redundant Array of Inexpensive Disks) which is far below the scope of this paper, it is worth mentioning that RAID 1+0 (aka RAID 10), RAID 0+1, or at least RAID 5 is essential for even the simplest HA environments.
- **Other Redundancy:**
Redundant disk controllers should be used. Most mid to high level disk storage architectures these days involve active-active load balancing and failover, where all controllers are actively used to distribute I/O processing load, and the work of one controller is taken over by the others in the event of failure. At a minimum, you want active-passive configuration for most HA environments. Additional redundant disk storage components include Host Bus Adapters, e.g. for Storage Area Networks (SANs) or redundant Network Interface Cards (NICs) for Network Attached Storage (NAS). For NAS, it is always preferable to separate NAS traffic from other network activity by using separate NICs and either separate physical network paths or router-controlled Virtual LANs (VLANs) which can minimize network contention by directing different kinds of traffic away from one another. Dispersing network activity does more than improve performance. It can also protect your most concentrated data streams from corruption. For example, some modern network monitoring software actually attaches supplemental data to in-stream data packets. For some kinds of traffic, this is harmless, but for data streams bound for inter-site replication or even local NAS, certain parameters (e.g. MTU) may need to be tuned differently in order to avoid incidental corruption.
- **OFA and SAME Guidelines & Enhancements:**
In the beginning there was the Oracle Flexible Architecture (OFA) which provided guidelines for distributing I/O of different kinds of Oracle files (e.g. data, indexes, undo, temporary space, and redo/archive logs)

across separate disks. Moreover, some file types such as redo/archive should be placed on non-striped partitions due to the sequential nature they are written to. The primary goal of OFA as presented by Oracle Corporation is centered on performance. As mass storage units have become more sophisticated, the idea of Stripe and Mirror Everything (SAME) has evolved as an attempt to simplify DBMS file dispersal. SAME essentially groups all file types together in one huge logical volume with many underlying disks in a highly available RAID configuration (typically RAID 5 or RAID 10). For large databases, multiple filesystems may be used, but the meticulous mapping of different kinds of files across different volumes is said to be superfluous. The reasoning is twofold. First, economy of scale suggests that inefficiencies are outweighed by raw horsepower of the massive storage arrays (e.g. huge numbers of extremely fast disks and gigabytes of cache). Second, with the increased complexity of the underlying disk geometry (breaking up the disks into physical partitions, mapping them to logical partitions and volumes, setting the stripe depth and stripe width, etc), it is extremely difficult to know for a fact that placing files on different volumes or filesystems actually means separating them across different physical disks or at least different spindles or partitions of those disks. Separation is crucial for both performance and availability. Making this happen is usually a painful process involving many hours or even days of intense “discussion” between DBAs, system administrators, and disk storage administrators — all of whom have varying perspectives of the system. This pain is one of the reasons Oracle Corporation has introduced Automatic Storage Management (ASM), discussed elsewhere in this paper.

Here we list certain OFA related practices which have vital relevance to highly available environments. We are not mandating a strict departure from SAME, just proposing a middle ground between OFA and SAME to satisfy special demands of highly available environments.

- Separate archive logs from other DBMS files so their relatively fast disk consumption does not compromise availability should they fill up the filesystem.
- There is another major reason to separate archive logs from other files: to accommodate current or even potential (future) Remote Disk Mirroring related technology. It is common in RDM environments to propagate archive logs in a manner which minimizes data latency and consequential data loss, versus other files which we may want to replicate using less expensive communications channels. For example, we usually want to use synchronous or at least semi-synchronous propagation for archive logs. For other data files, we may choose asynchronous propagation which can leverage less expensive (lower speed/bandwidth) communications while minimizing the impact on database performance by queuing up data that lags behind the communications throughput rate (typically during peak utilization periods). Queued up data could be lost should the primary site fail, but since the archive logs would have zero, all the data can be recovered (or almost all the data if using semi-synchronous propagation which de-escalates from synchronous to asynchronous under some conditions). So by combining these communications modes, instead of sacrificed performance or data loss, the net loss becomes recovery time (MTTR or at least RPO) — the time it takes to apply the missing transactions to the database from the archive logs. The bottom line is that you must typically place the archive logs in a separate disk volume in order to configure them for a different synchronization level than the other database files.
- Separate filesystems for backups, data imports/exports, and other large data staging areas. As with archive logs, the growth of any kind of large or fast growing data should not compete with online database files which could bring down all or part of the database if they run out of disk space.
- Separate storage for different applications on distinct disk volumes. If you run more than one database instance on the same server in support of multiple applications, for example, you do not want the failure of one disk volume to impact the availability of multiple applications. While you may have logical interdependencies between the applications, keep the planned and unplanned downtime between them in your control.

NETWORK COMMUNICATIONS

Redundancy of network infrastructure is another elementary given for most highly available environments, although beyond the more data-centric scope of this paper. The following techniques will help you attain basic availability goals, although for higher fault tolerance they are generally complemented by advanced technologies involving clustering and application resiliency (reviewed elsewhere in this paper).

BEST PRACTICES for Basic Network Communications HA/DR

- **Redundancy:**

There are numerous network related components which improve availability through redundancy. Typical examples include multiple DNS servers to route within and possibly between primary and secondary sites; routers, load balancers, and switches to redirect application servers as they gain or lose availability; and load balancers or switches to route across clustered database servers.

- **Communications Channel Isolation:**

When running more than one database instance on the same server, make sure to separate communications so that failure of one channel or port does not affect the availability of another. At a minimum, there should be differently named database listeners on distinct ports, e.g. 1521, 1526, and so on. Other communications dependent processes should likewise be evaluated for fault tolerance, such as XML DB and Oracle listeners for http, https, and iSQL. Just as for separation across disk volumes, if there are logical interdependencies between database applications, their respective downtime should be individually controlled.

When architecting advanced, network intensive HA/DR solutions, not only should communications supporting different applications be isolated, but infrastructural sessions should ideally be separated from regular business activity. If replicating from a primary to a secondary database, for instance, use a different port for replication than for the listener(s) which support online users. In fact, you may even want to separate intensive batch operations from online user sessions. Even though they are both at the logical, business level, they undergo different degrees of stress and therefore have different risk levels for failure. Channel separation can also facilitate performance tuning. For example, using a bequeath protocol for locally connected batch processes can offload significant and totally unnecessary network traffic from mainstream TCP/IP communications.

- **Fault Tolerant Configuration:**

This may seem obvious, but proper configuration is often overlooked, particularly when it comes to basic network and server redundancy. For example, even when Oracle RAC is not implemented, you can configure Oracle Network Services (e.g. tnsnames.ora) to connect to one or more secondary “failover” servers should the primary server be unreachable. This can be useful for replication environments. Of course TNS configurations only handle thick driver connectivity to the database (also called OCI or type 2 drivers). When using thin JDBC driver (type 4) connectivity, you can dramatically increase MTTR by programming hardware switches to failover to a secondary server — instead of reconfiguring manually upon failure. In fact, as of Oracle 9i, failover support can be automated at the database listener level even for thin driver connections.

APPLICATION SERVER AND MID-TIER SUPPORT

Oracle and mainstream third party application servers basically offer similar HA/DR functionality. Other than coverage of clustering software, the detailed analysis of mid tier architecture is outside the scope of this paper. Some elementary best practices, however, are outlined below.

BEST PRACTICES for Basic Mid-Tier HA/DR

- **Application Server Redundancy:**

Just as Oracle Real Application Clusters (RAC) enables fault tolerance across database servers, redundancy at the application server tier should also be provided. This may be accomplished either through built-in clustering software (within the application server software) or via external hardware switching. Either way, application servers can be configured as active-passive with failover from primary to standby servers as needed, or they may be load balanced in an active-active configuration which may help justify the extra cost of redundancy (the cost of redundant servers are more noticeable than the cost of redundant network switches and routers).

Note that the nature of the supported user applications may help determine your HA application server architecture. Stateless middle tier programs may be managed by just about any architecture (software clustering or hardware managed balancing and failover). Database applications, however, tend to be stateful, so you may be dependent on clustering within the application server software if you need to guarantee zero interruption should one of the servers — or connectivity to one of the servers — fail.

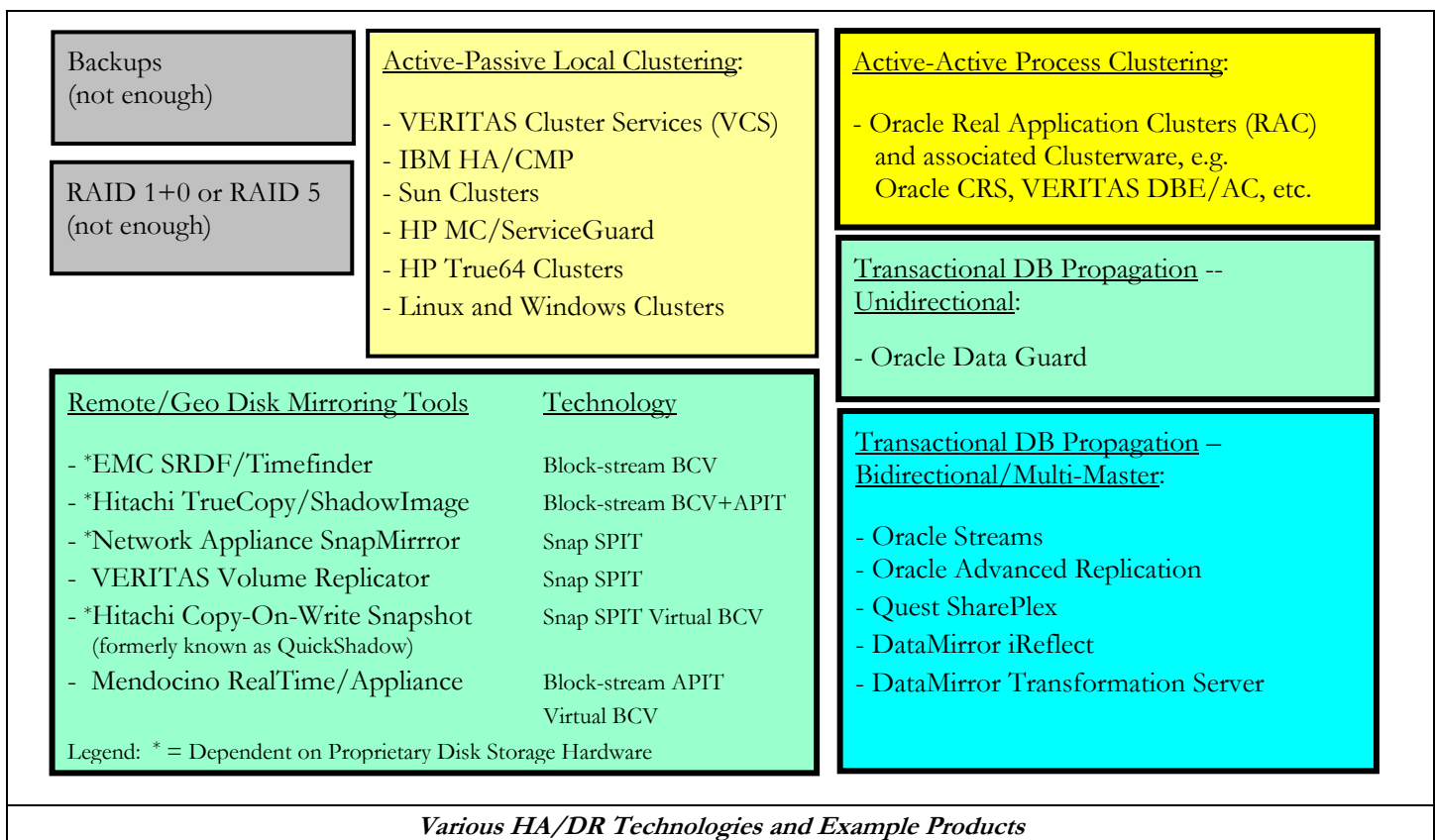
- **Other Mid-Tier Processing:**

Remember that application servers are not the only kind of middle tier services. Make a checklist for all mission critical middleware components including business rule engines, workflow servers, and messaging brokers. For data warehousing environments, you may even consider redundant data extraction, loading, and transformation (ETL) processes if real time or near-real time analysis and reporting is required.

ADVANCED THIRD PARTY HA/DR SOLUTIONS

There are numerous technologies which contribute to overall IT fault tolerance and recovery. The two most basic methods as far as databases are concerned are disk redundancy (e.g. RAID) and plain old backups — see the two small boxes in the top left corner of the figure below. For high availability environments, you need more advanced technology to assure your HA/DR requirements are met (e.g. MTTR, RPO, RTO) for a wide variety of planned and unplanned failure scenarios, and also to help assure that performance is not significantly impacted as a tradeoff for increasing availability and recoverability metrics.

There are three fundamental types of advanced third party HA/DR products: remote mirroring, transactional propagation, and process clustering. The colored boxes in the figure below (everything except the two small boxes in the upper left corner) represent these more advanced technologies and list some example products for each. Acronyms and special terms are explained later in this section (and also covered in other sections of this paper). Note that transactional propagation can occur at the application level, in addition to the database level which is the only level shown in the diagram. Application level propagation is not a primary focus of this paper, although it is explained. Application awareness within data-centric HA/DR technologies is also explained.



@TODO:

ADD DIAGRAM HERE to MAP major HA/DR Technologies from prior diagram against the 7 OSI Layers or perhaps just generic IT Infrastructure Layers such as **Disk Subsystem, Operating System, Network, Database, Application, Business/End Users** (from lowest physical to highest logical layers).

Relating HA/DR Technologies to generic IT Infrastructure Layers

IMPORTANT NOTE AND DISCLAIMER ABOUT PRODUCT CLASSIFICATION AND TECHNICAL ACCURACY

Many products do not fall exclusively into a single functional or technological category, although in this paper we use categorization to distinguish products to some extent. Our observations may conflict with those of other people. As the HA/DR industry has matured, many products which used to be firmly rooted in one category have broadened their capabilities and encroached one or more other categories. This de-specialization process is likely to continue where it makes sense. For example, under the Remote Disk Management (RDM) umbrella, the various recovery point methods (sub-technologies if you will) referred to as SPIT, APIT, and NPIT are slowly merging. On the other hand, higher level categorization is more likely to remain constant. RDM replication, as a case in point, should remain quite distinct from transactional propagation despite the sharing of certain operations at a conceptual level, e.g. Oracle's Flashback technology and the virtual cloning ability of some RDM tools.

To minimize the possibility of misclassification or inaccuracy in this paper, we first describe from a generic technical perspective the various capabilities and methodologies found in third party tools. Then we drill down into example products which utilize these specific technologies. For example, the principles of Remote Disk Mirroring and the various forms it takes are, for the most part, described before listing the various products which fall into this broad HA/DR category or the various underlying sub-technologies. Technology is extremely abundant and often transitory in nature. It usually moves faster than pen and paper. With all of this in mind, the reader is advised to verify the content provided here with vendors of the represented technologies before finalizing your HA/DR architecture and corresponding solutions. Understanding the generic principles will help you ask the right questions at the right time. In fact, some points in this paper actually provide specific questions for you, to facilitate obtaining meaningful versus fluff responses.

REMOTE MIRRORING EXPLAINED

Remote disk mirroring (RDM), sometimes referred to as Geo-Mirroring, is basically an extension of more common local disk mirroring technologies such as RAID, and is usually conceptually closest to RAID 1 in particular. RAID 1 involves the simultaneous or near simultaneous propagation of disk I/O across multiple similar disk units generally all physically collocated within the same chassis, a nearby rack, or at least close proximity within the same server room. Remote disk mirroring extends RAID 1's geographical boundaries to allow high availability failover to one or more remote sites which may reside tens, hundreds, or even thousands of miles away.

While it is possible to mirror computer server processing, for the purposes of this paper, we focus on remote *disk* mirroring which is currently the more prevalent and practical technology for most business and scientific computing environments. Note that remote process (server) mirroring is different than process clustering (discussed separately). Process clustering is a

form of distributed processing which automates the allocation of various sessions across multiple computers — much like Symmetric Multi-Processing (SMP) across CPUs within the same machine, but on a larger scale across different computers which are more loosely coupled. The allocation may be active-passive with failover from primary to secondary servers occurring upon system failure, or active-active as in the load balanced process partitioning of Oracle RAC and peer-to-peer computing technologies. Process clustering, while advanced technologically, is relatively prevalent and affordable for many HA environments. True process mirroring, on the other hand, involves running the exact same processes on multiple computers at the same time. Whereas load-balancing divides the work across multiple computers, process mirroring *duplicates* the work in order to minimize or eliminate interruption (RPO and RTO metrics) under the widest variety of failure scenarios. Process mirroring is in general exorbitantly expensive, currently more apt for NASA space shuttles than the fault tolerant computing necessary for most HA/DR environments.

To better understand remote disk mirroring technology and its strengths and weaknesses, let's define the core characteristics and fundamental configuration options.

- RDM operates at the disk level. Disk block changes are tracked and replicated either in periodic spurts (often called “snap technology”) or in a continuous “block-stream” fashion. Regardless of snap or block-stream technology, the propagation of data from source (primary) to target (secondary) servers is generally referred to as a data stream.
- Data is usually pushed from one or more sets of primary/source disks called disk groups or volume groups to one or more corresponding sets of target disks.
- Major Strength: RDM is comprehensive in that it replicates *everything* from primary to secondary server disk storage. This means all database files (including redo/archive logs) as well as all supporting infrastructure files (e.g. configuration files, DDL, batch/cron process files) and N-Tier application source code as well.
- Major Strength: Once set up, RDM is extremely low maintenance as far as the assured data propagation to secondary (target) server storage. Adding or removing disk volumes from replication requires special attention and possible downtime, but this is true when scaling many alternative HA/DR architectures.
- Weakness: A traditional argument against RDM technology is that block level disk corruption is propagated to the failover environment's disk storage. Transactional level propagation such as Oracle Data Guard, Streams, and advanced replication, on the other hand, would either choke on the corrupted blocks before propagating them over a clean secondary database — or the corruption may actually be overlooked. For example, in the case of Data Guard and Streams which both utilize log scraping technology, if the block corruption occurs in a data file but the redo/archives remain viable, a problem will not occur until the *next time* the corrupt blocks are accessed for replication.
- Potential Weakness: Another well known but usually incidental disadvantage of RDM is that it propagates significantly more data than transactional replication technologies which operate at a higher, almost logical level. It takes more resources to capture, transmit, and apply thousands of data blocks versus a lesser amount of data represented by SQL commands in a transaction log, or even compared to DBMS row data itself. Transactional replication is often estimated to be 30 percent more efficient for database propagation. Or course, remember that transactional replication cannot handle any files outside of the database.
- → TIP – Application Level Replication (ALR): While on the topic of replication efficiency in terms of volume, let's abstract one more level to understand the whole picture. Replication technology also exists at the application or procedural level, transmitting true business transactions between database sites. It is obviously more efficient to transmit a single command that says “Post Payroll for All Employees” than to replicate thousands of rows over a short period of time (indeed, millions of rows if you process payroll for DOD!). Oracle provides mechanisms to help automate procedural replication, e.g. the DBMS_DEFER() package which leverages Oracle Advanced Queuing beneath the hood. Third party products are also available for application level replication, and are particularly popular in the ERP arena (e.g.

for Oracle Applications/Financials, PeopleSoft, and SAP). The reason for this popularity is because automating replication at the application level requires technically interfacing with certain parts of the application code, e.g. after certain events when it makes sense to synchronize data across primary and secondary databases. ERP applications are well known, not particularly inexpensive :-), and have traditionally large market share — so developing application specific replication programs for ERP is generally cost effective.

Incidentally, ALR or procedural replication can also be done manually. For instance, instead of procedurally replicating a data load of millions of rows into a data warehouse, you could simply perform the load on the primary database, and perform the load again on a secondary instance. The databases should remain in sync provided that they were in sync before the load, and that the same data and ETL logic is used on both sides.

- Potential Strength — Application Aware RDM Tools: RDM will always remain distinct from Application Level Replication (ALR, see separate bullet). However, RDM products are evolving a degree of application awareness. The first generation of application awareness allows applications to interface with RDM via batch interface or API. For example, after processing payroll, a custom API call from the application could insert a Named Point In Time (NPIT) recovery bookmark in the RDM data stream and in a corresponding recovery dictionary which tracks all NPITs for use in DR scenarios. A quiesce (split mirror) or replication checkpoint may also occur at the NPIT marker to ensure data integrity at that known, meaningful state of the application, e.g. just after payroll processing or after the next batch of 100 loans is approved. The recovery dictionary can also record Specific Points in Time (SPITs) which occur at regular time intervals, independent of NPITs. (More on SPITs later.)

The second generation of application aware RDM tools are not here yet, as far as this author knows. They will provide the intelligence from the RDM product side — similar to Application Level Replication tools (see separate bullet), but only for the generation of NPIT recovery markers in the RDM data stream.

- Weakness: Replication is technically restricted to unidirectional, active-passive support — although some products have mechanisms to provide limited access to the target data (typically involving some kind of data cloning feature, discussed below). There are some filesystem replication products which allow target environments to remain online and updateable, but these products generally use higher level propagation schemes (e.g. automated FTP, SFTP, or proprietary file level propagation) and are not suitable for live DBMS support. One such product is SharePlex FS by Quest Software (a spinoff product of SharePlex for Oracle which is a transactional replication product made by the same company). For the purposes of this paper, RDM technology does not include these products.

Filesystems which use underlying RDM target disks are generally offline until failover, so MTTR is not great, but typically acceptable given the fact that other major infrastructure must also failover. Application servers, for example, must either have their connection pools recycled in place or must themselves be failed over to the secondary site. It can take over 20 minutes for the whole standby environment to become ready. For RDM alone, you must wait until all relevant filesystems come online, and then wait longer until the database itself is open and available.

Incidentally, RDM is intrinsically unidirectional because physical disk blocks must be reconciled into logical transactions by the DBMS before the data is usable, but the database cannot be open for processing while underlying physical disk blocks are changing. It's kind of a chicken and egg situation.

- Strength and Weakness: In order to instantiate a usable copy of the primary database without actually onlining and failing over to the secondary filesystems and database, many RDM products offer some level of data cloning. Cloning allows the primary failover environment to remain up to date, while a clone is in a limited capacity for other purposes such as reporting, testing, disaster recovery analysis, access to clean data for logical recovery (similar to recovering data using Oracle Flashback queries), and optionally for enabling failover to specific points in time (SPIT, often referred to as just PiT). Regarding the last benefit listed here (SPIT recovery), it is faster to failover to a database clone than to wait for the filesystems and ultimately the database itself to come online. Clone failover is also usually faster than in-place database recovery and roll forward, unless the recovery requirement is very small, as in database block-level recovery (an Oracle RMAN feature introduced in version 9i).

EMC provides a traditional data cloning feature called TimeFinder, available as an add-on for SRDF (EMC's core remote mirroring product). TimeFinder allows you to create one or more Business Continuance Volumes (BCVs) which are complete copies of the primary database (full clones). BCVs can be maintained to the current point in time, or they may intentionally lag behind the primary database to support recovery from logical corruption. For instance, if a table is accidentally dropped or updated in the wrong way, an hour-lagged BCV can provide ample time to restore good data, without resorting to more complex recovery schemes such as database or tablespace restore, or even recovery from an export file if one is available.

Comprehensive HA/DR architectures involving multiple full database clones (e.g. BCVs) used to be commonplace, in order to provide both current and intentionally lagged recovery both onsite and at one or more secondary DR sites. Oracle's Flashback functionality has been obsoleting the need for multiple BCVs, however, due to drastic disk space savings, cost savings (Flashback is free!), and infrastructure savings — the extra disk space requirements are almost negligible, depending on how far back you need to “rewind” the database and the rate of transactional changes over time. Oracle Flashback was first introduced in Oracle 9i and significantly enhanced in 10g — it is explained in more detail elsewhere in this paper where Oracle specifics are covered.

- Strength and Weakness: Oracle Flashback is not a viable DR option in the event of full database or site failure. Full database clones (e.g. EMC BCVs), while costly in disk storage, can drastically reduce MTTR since it is usually much faster to failover to a full, open standby database than to wait for the main standby disk volumes to come online and finally for the database to open. The tradeoff is data loss. Once a clone is open for full access, propagation to the clone is suspended. This is a technical necessity in RDM technology since data is replicated at the block level instead of at the transactional level (see transactional replication discussed elsewhere). Full database clones typically alternate between open and closed states. They must periodically close to play replication catch-up. By maintaining more than one clone, you can configure your environment so that at any point in time at least one online clone is available for fast failover — again, at the expense of data loss compared to a main standby whose data is either completely current (if replicating synchronously) or almost current (if using semi-synchronous or asynchronous propagation). It is possible but less common in HA/DR environments to configure RDM with a single failover instance which doubles as the main standby when replicating and as a full clone when it is open. Depending on how often the clone is open, however, it would be highly likely that RDM itself could not provide failover to the most current viable point in time. This data loss can be worked around by applying archive logs to the failover database. While this adversely impacts MTTR, it is still a viable DR option.
- Strength — Dynamic or Virtual RDM Cloning (aka Copy-On-Write Snapshots): While Oracle Flashback supports only database level rewind, RDM technologies can provide similar back-in-time views of *all* kinds of files, e.g. such as binary executables, configuration files, alert and error log files, source code, DDL, and batch process scripts. Older RDM products such as EMC TimeFinder BCVs require expense disk space and related infrastructure to maintain one or more full copies of the database. You need an entire database copy for each interval in time you'd like the option of rewinding to, e.g. a current copy, plus one copy which lags behind by 5 minutes, and another copy which lags by 60 minutes (to allow ample recovery time from accidental, logical corruption). Newer RDM technologies offer real rewind functionality similar to Oracle's Flashback feature. They come in three basic flavors: continuous block-stream (CBS) replication, periodic snap (PS) replication, and hybrid replication. See separate bullet explaining the differences between these technologies and their corresponding abilities to restore data to Any Point in Time (APIT), Specific Points in Time (SPIT), and Named Points in Time (NPIT).

Some RDM products support a conceptual rewind feature similar to Oracle Flashback. They create a virtual clone of the database as it existed at a user selectable point in time by logically projecting block change “deltas” in a backwards fashion over the secondary database. The database files are not actually overwritten — instead, a virtual picture of the database is constructed as if the failover database were rewound to a previous state. The virtual clone can be mounted and opened as if it were the real database. As more deltas are propagated from source to target, the virtualization process consumes more and more offline journal space which is maintained separate from both the failover target and the rewound clone image. Eventually, due to journal disk space limitations and the need to have more current data, the virtual clone must be closed. Updates made to the clone during the open period are discarded since RDM technology is not bidirectional. So

RDM clones are best used for reporting, testing, recovery of accidentally deleted or changed data (logical corruption), and for providing an extremely convenient way of verifying that a previous point in time would indeed be a viable point in time for recovery purposes. Once verified, you can then rewind the failover target itself to that point in time. Some RDMs allow you to conceptually convert the clone into a permanent image, which can appear to reduce recovery time (RPO and MTTR). Under the hood, however, they are hiding the two step process and, therefore, typically require a brief database outage for processes already using the clone to switchover to the fully restored failover database.

RDM rewind journaling is fundamentally similar to the way Oracle uses undo and flashback area. It is worth noting, however, that RDM rewind journaling technology preceded Oracle Flashback in the commercial marketplace. It's anyone's guess who conceived of the technology first. Oracle might argue that the ability was always there and implemented at a smaller level in their patented rollback/undo processing. Because RDM and DBMS replication methods operate at two completely different levels, there is probably (hopefully?) no legal contention over the matter ;-).

Some RDM products create virtual clones in an opposite but logically equivalent manner. They utilize reverse journaling instead of forward journaling. Instead of projecting the block change deltas backwards from secondary site storage, they project the differential data over primary storage in a roll forward fashion. While reverse and forward journaling techniques achieve the same logical result, the scenarios in which they excel or underperform are extremely different. First, let it be said that virtual cloning is often used more for providing temporary reporting, test, or analysis environments versus supplementing your DR scenarios. If you have most of your reporting and such application servers and other infrastructure at the primary site, forward journaling will save valuable network traffic since the components are local to each other. On the other hand, those secondary processes potentially compete for local network bandwidth (depending on how you have configured both physical and VLAN aspects of your environment). They may compete for expensive server resources as well. By creating clones using reverse journaling from a secondary database, more inter-site workload partitioning can occur, and these issues go away.

There are additional considerations when comparing forward and reverse journaled clone generation. Clones are always useful for retrieving data from a past point in time, to recover a dropped table or logically corrupted data for instance (as you could similarly leverage Oracle Flashback). Some RDMs allow virtual clones to play a more significant part in disaster recovery — more like Oracle Flashback database but better in that you can restore the complete environment to a remote site (database and all supporting infrastructure). Traditional full-copy business continuance volumes (BCVs) have served this purpose for many years (e.g. EMC Timefinder and Hitachi ShadowImage). RDMs which support this feature accomplish it in a more complicated, but more flexible manner. We say more complicated since the database content of a clone is initially stored in two places: the actual primary or secondary data files and the journal pool which contains the transactions that logically overlay the actual image. For example, we might want to failover to 10 minutes behind the last changes propagated to a secondary site so the recovered environment precedes the point of corruption. Reverse journal rewinding provides a fast way to satisfy this requirement — typically much faster than restoring a hot backup and rolling forward. Of course, compared to a full-copy BCV which is constantly maintained with a 10 minute lag, rewind journaling would be comparatively slower. (On the other hand, the BCV consumes incredible disk space by comparison, and does not provide the flexibility of dynamically choosing the lag value at recovery time. The best you can do is maintain multiple BCVs.)

Now consider forward journaling under the same scenario. If the primary site or connectivity to the primary site has failed — which is most often the case with remote failover — forward journaling is useless. Even in a planned switchover where the primary site remains available, forward journaling to a remote clone will consume substantially more network traffic versus reverse journaling. If your RDM vendor implements forward journaling for virtual cloning purposes, make sure they use reverse journaling for rewind instantiation of an intentionally lagged failover environment.

Note that the virtual clones we've been discussing are popularly referred to in the industry as "copy-on-write snapshots". The term copy-on-write goes back to operating system memory management internals, and is perhaps less intuitive. Hitachi Data Systems' Copy-On-Write Snapshot product (formerly called QuickShadow) is a popular example of a reverse journal cloning tool. Mendocino Software provides reverse journal cloning technology which allows the replication journal to restore to *any* local or remote destination. If you choose a target on which no failover image has ever been

created, it will automatically propagate all data and infrastructure tracked in the journal pool. This from-scratch propagation is referred to as a “super transaction”. Products from both Hitachi and Mendocino are explained further else where in this paper.

- **Potential Weakness:** Some RDM products require specific hardware from the same vendor that supplies the RDM software or appliance. Be careful — if you’re going to move into more proprietary corner, make sure you are aware of it and doing it for the right reasons.
- **Potential Weakness:** Some RDM products require potentially intrusive, periodic processing to guarantee “write order fidelity” (WOF) — so that data is always written to secondary site(s) in exactly the same order as on primary storage. Write order in the data stream is required to avoid integrity problems across various data including configuration files, source code files, flat data files, and complex DBMS data. A primary issue in evaluating different RDM technologies is whether write order enforcement is automatic across all propagated blocks, or if it requires special quiescing or checkpoint processing which can impact database performance, especially when performed during peak utilization. Since recovery to points in time during peak utilization are as important, if not more important, than recovery to points in time during low (off hours) system usage, the WOF method may become a deciding factor for your environment.
- **Potential Weakness:** Due to the co-dependency of all tablespace data/index files on centralized, persistent DBMS infrastructure such as Oracle’s system tablespace and control files, there are often limitations on dividing parts of a database into different replication groups, e.g. to satisfy HA/DR requirements for different application schemas within the database or to help optimize performance by partitioning the load across different data streams. On the other hand, a common best practice for HA/DR environments is to separate logically disparate schemas into different database instances, which would preclude the dependency restrictions — see details and related options in the next *Best Practices* box (after the *Important Concept* box below). Moreover, you can usually increase propagation performance by multiplexing the data stream of a single database across multiple, fault tolerant communications channels.

IMPORTANT CONCEPTS: DBMS Sensitivity to Write Order Fidelity and Fractured Blocks

- DBMSs such as Oracle, DB2, SQL Server, etc. are most sensitive to write order fidelity (WOF) since ongoing I/O typically affects multiple tables spanning multiple areas across multiple large files, and integrity must be maintained in the physical infrastructure as well as the logical, business level. While WOF is crucial, it does not necessarily need to be enforced continuously in the data stream — and even when it is, it may not be enough, depending on the technology being used.

RDM propagation occurs at a physical layer far below database and corresponding application logic. OS level block I/O is typically smaller than DBMS blocking factors, e.g. 512 bytes to 2K or 4K compared to typical Oracle block sizes of at least 8K for production databases. Even when WOF is guaranteed, it is possible for failure to occur at some level (media, network, server, database) before RDM has finished propagating all the physical blocks which constitute a logical databaseblock involved in a data transaction. This partial result is manifested on secondary server storage as “fractured blocks” or “split blocks”. Technically, if you were to change Oracle’s block size to the same as the OS block size (at the expense of significant performance impact), fractured blocks would not occur.

Most people stop at the fractured block level when considering the kinds of propagation damage that RDM technology must work around. Keep in mind that RDM is generally unaware of complex logical level transactions affecting multiple database blocks. Even if the fractured block problem were to go away, it is still possible at point of failure for RDM to have transmitted only some of the database blocks associated with a complex logical transaction. As with fractured blocks, the result is an incomplete, corrupted state on secondary server storage. Since the corruption is at a higher data blocking level, the term “fractured

transactions” is more appropriate. When data files are affected by fractured transactions, the damage is correctable at instance startup (automatic crash recovery) since redo/archive logs can reconcile the differences. Left alone, fractured transactions — unlike fractured blocks — will probably not cause database errors, although the damage is likely to affect some point of your application. To minimize the possibility of fractured transactions from affecting redo/archive logs, redos and archives can be propagated synchronously (even if other data files are propagated asynchronously). As with fractured blocks, recovering to a point in time shortly before point of failure can also prevent fractured transactions from corrupting your failover environment, even if that point is just a minute earlier. For both fractured blocks and fractured transactions, performing a periodic split mirror or related replication checkpoint process is usually the most practical, 100 percent certain way of avoiding these kinds of RDM corruption scenarios. For purposes of simplicity, when we refer to the concept of fractured blocks, we also imply fractured transactions unless otherwise stated.

Upon failover to and consequential startup of a secondary server database, most DBMSs such as Oracle will recognize that the database failed without being shutdown. Automatic crash recovery services will be performed and partial transactions will be backed out (aka undone or rolled back). Crash recovery will usually take care of everything necessary to allow the database to start and work again properly. For Oracle, crash recovery depends on the completeness and integrity of redo log propagation, so it is preferable to propagate redo and archive logs via synchronous communication even if other data files are propagated asynchronously (see next *Best Practices* box). Fractured blocks may occur in the redo log itself, however; or a split block in a database file may not exactly match the logical level backout instructions in the redo log. Consequently, it is possible that some parts of the failover database may be corrupted, even when using synchronous log replication. Worse, that corruption may be unrelated to other corruption you may have been trying to recover from, since corruption from fractured blocks and transactions is caused by the DR framework itself. If the corruption occurs in a database control file, the entire instance may become unusable until the control file is recreated (assuming adequate control file backups were created and available on secondary server sites).

Some RDM products are more susceptible to fractured blocks than others, although this vulnerability may arguably be due to technical philosophy versus technical inferiority. To work around the potential for fractured blocks (no matter how unlikely they are), many RDM products recommend or mandate periodic data quiescing or checkpointing. Both of these methods involve adjusting the RDM’s view of continually changing DBMS data by providing specific points in time when *both* write order and block integrity are guaranteed.

➔ **Warning:** Be careful with terminology since vendors may use the same term to mean something different. For example, an RDM vendor may use the term checkpointing but really mean quiescing — or vice versa. Or they may use completely different jargon. Understanding the concepts and architecture is unfortunately critical these days to choosing the right HA/DR solutions for your environment. You may consider asking your vendor to specify their functionality using your more generic terminology, or at least provide a cross reference of some sort.

- **Quiescing** has been around for many years and is perhaps most reminiscent of first generation EMC SRDF technology. SRDF was one of the first popular RDM products and is still in wide use today, although its market share is giving way to more modern technologies (more on specific third party products later). Quiescing is essential for RDMs which do not guarantee write order fidelity. Worst case you would shutdown the database as if you were performing a cold backup — not suitable at all for high availability environments. More typically it involves telling the database to go into conventional hot backup state, basically suspending writes to data file header blocks (versus RMAN backups which obtain a consistent hot backup view using other means). For Oracle, quiescing starts with the ALTER TABLESPACE BEGIN BACKUP or the newer ALTER DATABASE BEGIN BACKUP commands, and quiescing ends with the corresponding END commands. There is usually more logic to a quiescing script, including RDM specific commands to perform the copying. This overall process is often referred to as “Split Mirror”

implementation.

Common misconception about Oracle's hot backup mode leads people to believe that split mirrors are more intrusive on database performance than they actually are. When hot backup mode starts, Oracle writes pending (dirty) data to disk (via DBWR process) and updates time timestamp in each affected file's header block (via CKPT process). This activity causes a small spike in overhead but is no different than a regular checkpoint which occurs at least as often as redo log switches. During hot backup mode, whenever DML writes the first time to a block (in an affected tablespace's data file), the *entire* block is written to redo so there is no confusion (corruption) due to the difference in os blocking and database blocking factors. This admittedly causes extra overhead, although subsequent writes to affected blocks (during hot backup) write the changes only, since Oracle already has a pristine baseline. Header blocks of changed data are not updated in the actual data files — the SCN remains at the point in time when hot backup began — so Oracle knows during recovery that redo must be used instead of data file content, and a clean and complete copy of each block is always available. NOTE that writes to the data file BODY (not the header) continues normally during hot backup, so when hot backup mode ends, there is no intensive catch-up process. There is, however, extra time involved upon failover (MTTR) to perform crash recovery as explained above. Oracle uses the SCN marker made at Begin Backup to determine the starting point for recovery processing in the DR database (just as it would if the primary database were recovered in-place from a real hot backup). MTTR is worsened by the fact that disk volumes must be mounted before database startup/recovery can begin, although this is true for most RDM products due to their active-passive nature.

- **Checkpointing** is another means of suspending the RDM view of constantly changing DBMS data. It is generally less intrusive than quiescing and usually simpler to configure and maintain, but may still be susceptible to some forms of fractured transactions. Checkpointing usually involves periodically suspending the data stream at some point in the propagation from source to target (primary to secondary) storage — without freezing or altering DMBS activity in any way. Checkpointing provides a means of guaranteeing write order fidelity but, again, in rare circumstances may still result in fractured transactions.

BEST PRACTICES: Assuring RDM Write Order and Avoiding or Minimizing Fractured Blocks

- Even if your RDM product guarantees write order fidelity, ask the vendor about their official policy on requirements for periodic full or partial quiescing and/or checkpointing. Do they assure 100% DBMS integrity on the secondary (target) servers under all circumstances?
- If the answer to the previous question is not a solid 100% reinforced with some kind of certified support statement, then you should count on extra RTO, MTTR, and possibly RTO costs in your HA/DR plan.
- If your RDM vendor says that quiescing is optional but their product supports it, you should probably do it. Negative tradeoffs typically occur only in very high volume, high throughput environments where database activity is so high almost all the time that they cannot tolerate the extra performance (and sometimes disk capacity) demands of quiescing.
- When quiescing a database for split mirror purposes, a traditional best practice is to *not* place all tablespaces in hot backup mode at the same time. This is in spite of Oracle 10g's new Alter Database Begin/End Backup commands which are provided as simple administration options for small databases with low transactional volume. Placing only one or two tablespaces in hot backup mode at a time alleviates concern regarding excessive redo activity during the backup (particularly at the beginning of the backup when all dirty blocks are flushed to disk). See *Quiescing* above for more information.
- Use one database instance per application schema whenever possible. This will prevent the planned or unplanned downtime of one application from affecting other applications. This can also facilitate the distribution of communications load across multiple communications channels, although this can be done

through physical multiplexing as well, perhaps with greater efficiency. Nonetheless, due to potential corruption problems (as the third reason), you should implement only one logical schema per instance. Here is the rationale:

Most RDM products allow you to bind together one or more disk volumes into a single data stream or replication group, commonly referred to by multiple vendors as a “consistency group”. Only within a consistency group can write order be assured. Fractured blocks cannot occur between consistency groups (only within one). The simplest way to ensure database integrity is to place all DBMS files in the same consistency group. For performance purposes, you may be inclined to separate into different groups certain files associated with logically disparate tablespaces, e.g. tablespaces for different schemas. This can be dangerous since the system tablespace and other persistent infrastructure such as control files are tied to all tablespaces, and placing them in separate groups can cause logical and perhaps even physical database corruption. Separating redo and archive logs into their own consistency group is usually acceptable, on the other hand, particularly if it is required by the RDM product in order to propagate the logs synchronously (to minimize data latency and consequential data loss) versus propagating other database files semi-synchronously or asynchronously to help save communication pipeline costs. See next *Best Practices* box for deeper coverage on RDM communications modes.

- If your database is very large (e.g. a VLDB with one or more TeraBytes), you should consider having more frequent but smaller, partial quiesce points. This means staggering the quiesce intervals across different logical parts of the database (e.g. tablespaces). Make sure this approach is supported by your RDM and database vendors (yes in the case of Oracle, but keep in mind data integrity across tables, whether or not the tables have physical foreign key constraints). Staggering can substantially reduce the amount of continuous time spent in quiesce mode, distributing the extra stress over a longer period of time. Notable tradeoffs include added complexity and possibly a reduced number of recoverability points over, say, a 24 hour time span.

RDM products support one or more kinds of physical recovery levels: Specific Point in Time (SPIT), Any Point in Time (APIT), and Named Point in Time (NPIT). SPIT technology has been around the longest, and is usually referred to as just “PIT”. To eliminate confusion between SPIT and the other forms of point-in-time recovery, we use the term SPIT in this paper, despite the unattractive biological connotation.

SPIT products utilize periodic snap (PS) replication technology, performing repeated “split mirror” quiescing or checkpointing at regular time intervals which determine the safest recovery points — often the only recovery points when it comes to handling DBMS’s. The snapshot (or snap) frequency may vary in granularity, possibly just minutes apart or as infrequently as a few times per day — any less often and you may find difficulty justifying the price of this or any advanced DR solution versus simply exporting daily hot backups to a failover site. Failover recovery is available at each SPIT with guaranteed integrity. For guidelines on setting SPIT interval, see the next *Best Practices* box in this section.

Block changes which occur between SPITs are usually tracked as cumulative overlays, so if a block undergoes more than one change in a single SPIT interval, only the most recent content is propagated. This has the advantage of reducing network traffic at the expense of providing less recoverable points in time. In fact, most SPIT products have explicit restrictions on the frequency of SPITs performed for each set of replicated disk volume(s), so make sure you ask what the magic number is before settling on a specific product. If the product offers a dynamic cloning feature such as providing a virtual copy of the database as of a user selectable point in time, the spin-off granularity of the clone will likewise be affected.

EMC SRDF/TimeFinder, Network Appliance’s SnapMirror, and VERITAS’ Volume Replicator (VVR) are examples of SPIT technology with different underlying approaches and dependencies.

EMC’s SRDF comes in multiple flavors, e.g. regular SRDF and newer SRDF/A which supports asynchronous replication.

EMC also has plans to expand their RDM offerings. In this paper, we loosely use the term EMC to refer collectively to all kinds of SRDF, with or without the TimeFinder option which allows you to create spinoff full-copy clones of the replicated data called Business Continuance Volumes (BCVs). EMC can copy data continuously, but it *guarantees* recoverable SPITs only at regular time intervals when quiescing is performed. It is dependent on proprietary (though admittedly very fast) storage from EMC, e.g. their Symmetrix series of mass storage units.

Hitachi Data Systems (HDS) offers a variety of RDM products which require proprietary HDS disk storage. Hitachi is particularly strong in the mainframe arena (OS/390 and newer z/OS) as well as open systems (Unix, Linux, and Windows platforms). TrueCopy and ShadowImage are roughly equivalent to EMC's SRDF and TimeFinder BCVs (respectively) — one product responsible for the primary replication framework and the other supporting full database cloning. One of TrueCopy's major strengths is its ability to guarantee write order, provided that you adhere to some basic guidelines when selecting synchronous versus asynchronous replication modes, and configuring continuance volume(s). A continuance volume is essentially a grouping of disk volumes into a single set of replication stream data. The equivalent of TrueCopy on the mainframe is called NanoCopy. Hitachi also makes Copy-on-Write Snapshot (formerly called QuickShadow) for open systems — supporting dynamic virtual cloning of a database and all supporting infrastructure with far less expensive disk space requirements versus full-copy BCV cloning. Virtual RDM cloning is discussed in more detail earlier in this paper. Like the EMC products, HDS TrueCopy, ShadowImage, and Copy-on-Write Snapshot products all require proprietary disk storage systems from their respective vendor (Hitachi).

Net Appliance (aka NetApp) SnapMirror propagates data only at periodic SPIT intervals, and also requires proprietary mass storage. SnapMirror is often complemented with other NetApp snap related products, e.g. SnapManager. Through the use of the Filer disk operating system which generally accompanies NetApp's proprietary storage systems, innovative version control features are added to an otherwise standard looking filesystem. For example, one or more copies of past versions of user selectable files and directories can be made available immediately online, reminiscent to an old and treasured operating system which is quite hard to find these days: VMS (made by Digital Equipment Corporation which was acquired by Compaq, and subsequently Compaq was acquired by HP).

VVR provides snapshot (aka snap) technology as well, but has the advantage of being hardware independent. VERITAS is foremost a software company, so VVR works with storage systems from a variety of mainstream vendors. You can change storage vendors without incurring the extra material and labor costs of re-architecting your replication framework or reeducating your IT staff.

For a longer list of RDM products and technologies, see the diagram called *Various HA/DR Technologies and Example Products* near the beginning of the *Advanced Third Party HA/DR Solutions* section of this paper.

APIT products implement continuous block-streaming (CBS) to emit an ongoing flow of disk block changes from primary to secondary servers, except during network outages or other scenarios when disk “journal” space for queued-up block changes becomes scarce and granularity may de-escalate to send only the most current image for each changed block. CBS products have the core ability to provide virtually unlimited recovery granularity; hence the term Any Point in Time (APIT). Hitachi and Mendocino Software make RDM products which leverage continuous block-stream, APIT technology. It is important to ask your CBS vendor whether they officially support APIT recovery. Refer to other areas in this paper which explain the concepts of fractured blocks and the need to periodically quiesce database activity via hot backup and propriety split mirror commands.

A good way to understand the significant difference between APIT and SPIT is to envision a stream of sand falling in an hourglass. Place your hand next to the stream, vertically as if you are going to shake someone's hand. Now connect your index finger (pointer) with your thumb to make an O around the stream of sand. This is APIT, basically viewing and recording the stream of sand in a continuous, streaming fashion. Replace this image with another to understand the time sliced nature of SPIT technology. Hold your hand flat, palm-side up and parallel to the ground, right next to the stream of sand. Now, to maintain data consistency, you will need to periodically perform two extra tasks: 1) cup your hand and stick it under the stream of sand, basically causing a mess unless accompanied by 2) stop the stream of sand for a short yet arguably intrusive period while your hand processes all the grains. Extra task (1) here is similar to a split mirror process, and (2)

conceptually resembles the database quiesce process, although placing Oracle in hot backup mode does not actually inhibit physical writes to the data files (details on this covered elsewhere in this paper). Keep in mind that while APIT technology can assure write order fidelity without the “extra” quiesce/split mirror activity, preventing fractured blocks may mandate these “extra” steps, particularly if you don’t want to spend extended recovery time (MTTR) waiting for the database to repair fractured blocks during automated crash recovery upon starting the failover instance. Also note that DBMS’s can be more vulnerable to block fractures than you would think — see Oracle bug examples under *Verifying Failover Integrity* in the next *Best Practices* box.

Mendocino Software makes APIT software which has a distinct advantage of working with virtually any disk storage system. Like Hitachi’s Copy-on-Write Snapshot (QuickShadow), it supports virtual RDM cloning technology. Moreover, Mendocino can spin-off a database clone at virtually any point in time (not restricted to snapshot intervals as SPIT products are). Mendocino started out with a software-only product called RealTime, which supported only some Unix platforms (e.g. IBM AIX and Sun Solaris). It has never been available for Linux or Windows. Mendocino’s next generation of this software will be sold as an appliance — a hardware component which plugs into your network without the need to install on a separate server. Of course configuration is still required. You can access the tool for both configuration and operational purposes by both web GUI and command line. The appliance will support heterogeneous platforms including Unix (IBM, Sun, and HP), Linux, and Windows, and maintain its independence from disk storage platforms as well. Mendocino has executives and engineers which come from VERITAS and other competing technology firms. They have innovative visions and are pioneering some of the application aware principles inherent in NPIT technology discussed below.

NPIT propagation is basically an intelligent kind of SPIT replication. In addition to or instead of creating recovery points at regular time intervals, NPIT creates recovery points at named points which correlate to meaningful states in an application — typically a specific business event such as the completion of invoicing for the day, just before posting to accounts receivable. Of course it would be more useful to bookmark each completed batch of 100 invoices. To minimize confusion after physical disaster recovery, when the business needs to answer the question “Where do we pick up?”, NPIT technology provides incredible value since true recovery time depends on business operations as well physical restoration.

In a typical multi-user application, it is generally impossible for all database sessions to be in a 100 percent known state at any single point in time, even if you were to optionally suspend all database transactions while recording the NPIT snapshot. So even NPIT is not perfect. By combining NPIT with periodic SPIT and or continuous APIT replication, however, you achieve ultimate flexibility as far as RDM technology is concerned. Combining RDM at a higher level with complementary technologies can provide even more HA/DR power or reduce licensing costs (since most Oracle HA/DR add-ons are free). Combining technologies are discussed in various places throughout this paper. Also see bullet on *Application Aware RDM Tools* for more information on NPIT technology.

BEST PRACTICES: More tips for Remote Disk Mirroring

- **Verifying Failover Integrity:**

Use Oracle’s DBVERIFY utility after failover to an RDM database, to detect and repair fractured blocks.

→ NOTE: There are documented bugs in Oracle regarding problems identifying and dealing with fractured blocks. The following are examples for version 9i:

- Bug 2373145: “DBVERIFY does not identify fractured blocks”. (This affected Oracle 8.1.7.4 through 9.2.0.1.)
- Bug 2649033: When there are fractured blocks in control file, “LMON may crash instance with ORA-227 errors”. (Oracle 9.2.0.3.)
- Bug 2354965: “False fractured block messages possible during Direct Path IO”. (Oracle 9.2.0.3.)

- **Synchronous, Semi-synchronous, and Asynchronous Communications Modes:**

The highest degree of data protection is achieved with synchronous propagation (zero data loss), then comes semi-synchronous (minimal data loss), and finally asynchronous propagation (least intrusive on performance

of the primary database but most prone to data loss). Synchronous and asynchronous communications modes are self explanatory (beyond the scope of this paper to define). Semi-synchronous replication normally functions synchronously, but de-escalates to asynchronous operation upon reaching certain predefined thresholds such as delays due to network bottleneck. EMC uses the term “adaptive sync” when referring to semi-synchronous propagation.

The tradeoff for data protection is the expense of a faster communications pipeline to allow synchronous or near real time writes across both sides of the pipe. If communications throughput is not sufficient, performance will degrade.

→ **TIP:** If cost is a factor — and it usually is — consider using synchronous communications for the redo and archive logs but asynchronous communications for all other data files. Synchronous redo/archive propagation will help eliminate the occurrence of fractured blocks and fractured transactions (see below) and has a variety of other benefits. For more information on this architectural option, see *OFA and SAME Guidelines* under *Best Practices for Basic Disk Storage HA/DR* in the *Best Practices for Basic Infrastructure* section of this paper.

- **Oracle versus OS Block Size:**

Set the Oracle block size as a multiple of the OS block size. This is a common best practice for many reasons, and can also help minimize the occurrence of fractured blocks. If the OS and Oracle block sizes are the same, Oracle says the fractured block issue goes away, only to be unfortunately replaced with bad performance, so few if any people have actually tested this theory ;-). Consider another form of data splitting called “fractured transactions”. RDM replicates data at the physical block level and is unable to identify all blocks associated with each complex logical transaction. It is therefore possible to propagate partial transactions within a data file, resulting in “fractured transactions” on secondary server storage. Propagating redo and archive logs synchronously (even if other data files are propagated asynchronously) can offset the occurrence of fractured transactions and fractured blocks in general, provided that crash recovery is performed on the secondary database before making it fully available. Of course, performing a periodic split mirror or related replication checkpoint process would be the only 100 percent certain way of avoiding these kinds of RDM corruption scenarios.

- **SPIT Snapshot Frequency :**

While some products allow you to configure intervals as low as every minute, if your database is sizeable and experiences high transaction rates, you need to make sure that the delta tracking logic does not thrash and either cause data propagation delay (e.g. if using synchronous communication) or data latency and consequential data loss in the event of system failover. For example, a well performing 1 TB Oracle database I know of generates about 350 MB of redo activity every 2-5 minutes. It typically takes about 4-5 minutes to place the database in hot backup mode, split the mirror, and propagate all block changes since the last SPIT to a single passive failover system approximately 100 miles away. Communication resources are reasonably available, so acquiring sufficient bandwidth is less of a problem than in many places. In a situation like this, a SPIT interval of less than 6 or 7 minutes does not seem practical, although setting a granularity of 3 minutes would probably provide more benefit than you would at first think. If a snap process extends into the next time window, most SPIT products will simply skip the next snapshot. In our example, setting the frequency to every 3 minutes would result in some snapshots occurring at the 3 minute interval, and others waiting 6 minutes. You can see that SPIT technology does not necessarily produce completely predictable results, particularly if you try pushing the limits.

REMOTE MIRRORING PRODUCTS VERSUS ORACLE

Oracle 9i/10g is plentiful in mirroring technology, but not in the remote disk mirroring flavor. The oldest conceptual similarities are the mirroring of redo logs and Data Guard's effective mirroring of the entire database. ASM also offers mirroring features which are closer to RDM technology, although ASM does not (yet) propagate between geographically distant sites.

Instead of comparing pros and cons between RDM and DBMS availability and replication mechanisms, you may consider complementing the technologies. There is nothing provided by Oracle, for instance, that automatically replicates supporting infrastructure outside the database, e.g. binary executables, configuration files, alert and error log files, source code, DDL, and batch process scripts. Obviously you'll need this infrastructure in your failover environment, and it must be kept synchronized with the database. Based on your budget, HA/DR metrics requirements, functional requirements, and desire to homogenize HA/DR targeted technologies, you might consider using only RDM — or you could save a lot of money and fulfill additional requirements by combining RDM with transactional propagation.

Coupling different technologies into your complete HA/DR solution architecture is quite commonplace. Heterogeneity is tolerated more these days than in the past because products are better at interoperating or at least coexisting with one another. In the classic example above, the products complement each other with minimal overlap, and therefore, minimal operational contention. A word of caution, though. Some combined scenarios are much more challenging. Another classic example uses technology outside the scope of RDM, but is covered elsewhere in this paper. To support active-active database clustering, depending on the operating system you are using, you may need to mix Oracle RAC with the OS vendor's or yet another vendor's clusterware (e.g. IBM HA/CMP or VERITAS DBE/AC). The clusterware may operate at the OS and/or filesystem levels (use of clustered filesystems is optional). There is a high degree of interaction and dependency between Oracle RAC, the clusterware, the operating system, and shared disk storage with its specialized OS and firmware. Obtaining just the right versions and patch levels of each product so everything works together seamlessly is a challenging task, and even more painful when it comes time to upgrade. Some products may require an important security patch, for example, before the complementary products are re-certified to work with the new version. Worse, diagnosing problems across the array of technologies can lead to vendors finger pointing at each other, with you in the middle :-(. This is bad enough with a vanilla DBMS and operating system dependency — it is potentially excruciating once you add additional layers of HA/DR technology.

Oracle Corporation wants your end goal of maintaining a healthy, reliable, and secure database to be as easy as possible. They also would like to make more money :-). In recognition that cross-vendor contention may be more noticeable than cross-vendor collaboration, Oracle has been augmenting their offerings with HA/DR solutions such as Cluster Ready Services (CRS) and Automatic Storage Management (ASM), both discussed elsewhere in this paper. Even better, many new features are available at no extra cost for enterprise licensed installations.

TRANSACTIONAL PROPAGATION EXPLAINED

For the purposes of HA/DR framework, transactional propagation (TP) equates to the replication of changes within the database. Oracle has three transactional replication tools: Data Guard, Streams Replication, and Advanced Replication. There are many third party replication tools on the market. In the Oracle community there are a few that stand out, including Quest SharePlex and multiple tools from DataMirror, such as iReflect which supports homogenous multi-master replication, and Transformation Server which supports heterogeneous replication with some limitations.

TRANSACTIONAL PROPAGATION PRODUCTS VERSUS ORACLE

Before version 9i Release 2, comparing Oracle to third party TP tools was relatively easy.

- 8i/9iR1 Data Guard (DG):

If all you needed were active-passive database replication with support for opening a target database for only brief periods

of time (e.g. for offloading reports), and no need to replicate anything except database transactions, Data Guard physical standby was the way to go. (Physical standby was the only flavor offered at the time.) Oh yeah, there was another major functional limitation. Data Guard posed significant MTTR delay since the current standby redo log could not be applied until the log shifted automatically or propagation ceased from the primary database (e.g. due to failover). Data Guard also added notable complexity to database maintenance. You couldn't beat the free price tag, though, since it came with Oracle Enterprise.

- 8i/9iR1 Advanced Replication (AR):

Advanced replication was adequate for handling multi-master replication requirements, if you could afford the trigger based performance encumbrance and extra maintenance complexity. If you had a low tolerance for data loss in the event of failover, combined with high transactional throughput, you'd need to resort to faster third party technology such as Quest SharePlex or DataMirror iReflect. Long and Long Raw data types have never been supported in Advanced replication (and probably never will be) — another factor that could have driven you elsewhere, especially when these older data types were more popular (before LOBs effectively obsoleted them). Oracle 9i did, however, introduce support for replicating User-Defined Types (UDTs).

When evaluating data loss, latency is naturally a key contributor. Advanced Replication, even in 10g, technically depends on `job_queue_interval` (init parameter for `DBMS_JOB`) which has a minimum value of 60 seconds. This means that data propagation cannot occur more frequently than every 60 seconds. As of at least v9i, Oracle has offered an undocumented parameter `_job_queue_interval` (with an underscore as the first character) which allows more frequent data transfers to be configured. In practice, however, most mid-to-high throughput environments thrash uselessly when setting the interval lower than 2 to 5 minutes. The net effect is latency backlog, so it is usually better to configure the interval more realistically at less frequent levels.

High transactional throughput requirements are more difficult to gauge than data latency. What constitutes “high” can vary drastically depending on many factors such as the horsepower of your servers (e.g. CPU and memory), communications bandwidth, distance between sites, record sizes, data types being replicated, and more. An extremely rough rule of thumb still seems to work in Oracle 9i/10g — for environments with more than 600 to 800 transactions per second, it is time to consider Oracle 10g Streams (don't bother with Streams in 9i) or evaluate third party alternatives.

For details on third party TP products, see comparisons with Oracle 9.2 and 10g further below.

- 8i/9iR1 Streams Replication:

Streams replication was actually introduced in Oracle 8i, but was effectively half vaporware. In 9i it became usable for very simple replication requirements having few replicated objects, due to cumbersome maintenance, lack of tuneability, and inability to effectively monitor Streams processing. It was certainly not mature enough in 9i to depend on for highly available and scaleable environments. There was almost no way to monitor critical replication metrics such as queue backlog (10g introduced new Streams data dictionary views). Streams also used Oracle's general purpose shared pool for in transit queuing, but was limited to only 10 percent of the shared pool size. Consequently, you would need to overallocate shared memory in order to provide adequate Streams support. General useability and maintainability was far below that of advanced replication, which still is below the sophistication of mainstream third party products from Quest and DataMirror. The list of inadequacies went on.

In 9i Streams performance was comparable to advanced replication, despite its log scraping technology which is theoretically faster than advanced replication's internalized triggers. Stream's log scraping technology is similar to that used for many years by much faster, mainstream third party replication products such as Quest SharePlex. But Streams still had significant infrastructural limitations in 9i, and advanced replication had maturity and years of performance tweaking on its side.

With the advent of versions 9.2 and 10g, Oracle certainly narrowed the gap between Oracle and third party TP technologies. There's a lot more to say on the Oracle side, especially since Data Guard, Advanced Replication, and Streams all come free

with Enterprise level licensing. There must now be more specific reasons for resorting to expensive third party products. Let's reexamine the picture with various HA/DR improvements in mind for both camps. Let's start with the Oracle perspective, but we've added another bullet in this section to more formally cover the outside vendors.

- 10g Data Guard (DG):

Oracle Data Guard introduced logical standby functionality (aka SQL apply) in 9i Release 2. For the first time, a standby database could remain open 24x7, like the competing products, improving the cost-benefit (ROI) of the millions of dollars typically spent on failover computers and related infrastructure. In 10g, a Real Time Apply feature was finally added, allowing transactions to be applied to secondary databases directly from the current standby redo log. Real Time Apply, available for both physical and logical standby, finally leveled the playing field of Data Guard recovery time (MTTR) versus the competition. 10g includes two new availability enhancements involving Zero Downtime Instantiation. First is zero downtime of the primary database while building the initial standby database. The standby database can now be created from the primary database's hot backup without quiescing or shutting down the primary database to enable Resource Manager or record the SCN (this info is now recorded in the standby control file). Second is zero downtime of the primary database during planned switchover to standby using prepare and commit switchover commands. 10g also added significant security and usability enhancements. See *Oracle HA/DR Features, Strengths, and Weaknesses* section for more information.

Before settling on Data Guard for high throughput environments, however, there are still some negative considerations. First, although data type support has been improved in 10g, logical standby still does not replicate tables containing any column with unsupported data types, e.g. BFILE, RowID, URowID, User-Defined Types (UDTs), Object Type REFS, VARRAYs, nested tables, tables using data segment compression, and IOTs with overflows or LOB columns. Second, measure Data Guard performance compliance against your particular requirements. Under the hood, Data Guard leverages Oracle log miner to translate redo logs on the target databases into SQL statements. Somehow, though, the performance of log miner seems to lag behind third party log scraping technology.

Data Guard's logical standby mode is most similar to the TP technology offered by third party products (e.g. products mentioned above). There are even similarities to Streams and advanced replication in that SQL transactions instead of native archive log recovery are run on the secondary databases. Consequently, all these TP technologies except for Data Guard physical standby allow the target databases to the restrictions of logical standby, all those products and even Oracle Streams/Advanced Replication supports minor infrastructural differences between primary and secondary databases. Indexes, for instance, can be fashioned for standby databases to improve reporting performance. For the same reasons, rolling upgrades it is possible to perform rolling upgrades to optimize availability across primary and secondary environments. Database patches and major releases can usually be upgraded in a rolling fashion, and the same principals generally work for upgrades in hardware, operating system, and, where applicable, clusterware.

For more information regarding third party TP product strengths and weaknesses, see the *Third Party TP Products* bullet further below.

There are a few distinct advantages for Data Guard physical standby. First, all data types are supported. Second, it is probably the simplest Oracle-provided TP framework to implement and maintain. Finally, it is the only Oracle or third party replication vehicle which preserves exact rowids from primary to secondary databases. Well behaved applications should not depend on rowid, but some older environments may still be rowid dependent for some behavior.

- 10g Advanced Replication (AR):

While 9i and 10g introduced improvements in materialized view functionality, supplemental logging, and some replication enhancements (mostly performance and usability related), conventional advanced replication has remained largely unchanged in 10g. Advanced replication 10g is still easier to maintain than Streams replication 10g, although advanced replication remains more cumbersome versus third party products. While AR still does not support LONG and LONG RAW data types (and probably never will), that weakness becomes less significant every year since LOBS were effectively obsoleted by LOBs. The performance limitations of AR's trigger based architecture continues to limit its usefulness in low latency, high throughput environments. The days of advanced replication are effectively numbered. Knowing this might

incline some people toward third party replication tools such as Quest SharePlex, DataMirror iReflect, or DataMirror Transformation Server — even though Streams will eventually become mature enough to offset their expensive licensing.

- 10g Streams Replication:

Oracle 10g has made it obvious that Streams is the new replication vehicle of choice for Oracle in the future. For its relevant strengths and weaknesses (and list of major 10g improvements), please refer to *Streams and Advanced Replication* in the *Oracle HA/DR Features, Strengths, and Weaknesses* section of this paper. Since we've bashed the usability of Streams a bit in this paper (especially compared to third party products), it is worth noting here that overall usability of Streams in your high availability environment may actually be simpler if you are already using it for other business requirements. It is often easier to use the same technology for multiple purposes, versus adding heterogeneous architecture solely for HA/DR purposes. Just make sure to keep your critical HA/DR requirements metrics in mind before making final decisions.

- Third Party TP Products:

Transactional propagation products from third party vendors have led Oracle replication technology for many years in most if not all major categories *other than cost*: performance, functionality, scalability, reliability/availability, and usability. Oracle started closing the gap in 9i, and has almost leveled the playing field in 10g. For HA/DR high throughput environments, there are still some compelling reasons to consider paying for expensive third party licensing compared to the *free* price tags of Data Guard, Streams, and Advanced Replication which all come with Oracle Enterprise.

The mainstream TP products considered in this paper include Quest SharePlex and two DataMirror products: iReflect for Oracle and Transformation Server for heterogeneous (or homogenous) replication between diverse data sources such as Oracle DB2, SQL Server, and Sybase. Transformation Server is the only third party product here with heterogeneous support. There are several mainstream third party products which support heterogeneous sources, e.g. IBM Information Integrator and Virtuoso which support more data sources than Transformation Server (including native XML, IBM MQ, and even spreadsheets) — but those products concentrate more on data virtualization and integration (aka federated support), although they do support bidirectional propagation. Also note that Oracle Streams has some heterogeneous support through optional gateways you can purchase. To date, heterogeneity is unidirectional only — outbound replication — with database gateway targets such as DB2 and Sybase, and message gateway targets such as IBM MQ and TIBCO. This paper remains replication focused, and heterogeneity is considered as a side benefit only.

All these TPs support bidirectional, multi-master replication with configurable conflict detection and resolution support. Both SharePlex and iReflect currently support Oracle RAC (clustered) databases, a focal requirement in many highly available environments, although SharePlex's implementation is truly comprehensive compared to iReflect which currently requires cumbersome redundant data piping configuration across which write order of SCNs across the RAC instances is not guaranteed. If you want to use RAC with iReflect, you'll need to vertically partition your processing so SCN ordering is not dependent between the nodes. Note that Oracle Streams, advanced replication, and Data Guard are all fully RAC compliant, particularly as of 10g when RAC awareness was added to Data Guard Broker and, consequently, to OEM's Database Control console which issues broker commands under the hood.

In contrast to the performance limitations of Oracle advanced replication, performance sluggishness of Oracle Data Guard's log mining functionality (all documented elsewhere in this paper), Quest SharePlex routinely propagates data from primary to secondary databases with only 4 second latency under moderate-to-high system load. This author has no DataMirror timing information as of the time of this writing, but iReflect is reportedly comparable to SharePlex with respect to performance. Transformation Server is a bit slower with the offset benefit of heterogeneous translation services.

SharePlex and DataMirror products easier to administer than Oracle advanced replication, and much easier than Oracle Streams administration, although Streams has made improvements in 10g. Essentially, the command line and reporting/GUI interfaces of the third party products are easier to use and more comprehensive. Usability here refers to both regular and exception operations. Detecting and repairing out-of-sync data conditions is a classic example of exception operations. SharePlex's "compare-repair" functionality, for example, is incredibly easy to use against a single table as well as an entire set of replicated tables. It is multi-threaded and even shows percent complete progress for each

table being processed. As of this writing, however, it does not work on LOB columns (it ignores them). Moreover, it does not support multi-master replication based on most-current (time oriented) priority, which happens to be the most common conflict resolution scheme. SharePlex resynchronization supports only site priority, and while online conflict resolution can be customized to handle other schemes, the compare-repair facility cannot... short of creating an entirely new resynchronization tool which this author has done (see separate paper *Bridging the Gap in Multi-Master Replication — An Advanced Synchronization Toolkit*). Nonetheless, SharePlex's compare-repair resynchronization functionality is light years ahead of Oracle's DBMS_RECTIFIER_DIFF() package which works on only one table at a time, does not support LOBs, and does not support a most-current (time oriented) resynchronization scheme for multi-master replication. DataMirror iReflect and Transformation Server resynchronization features are basically comparable with SharePlex — far ahead of Oracle but definitely shy on the multi-master side. You'd wonder why this would be the case when all these products support time priority online conflict resolution for multi-master replication. They all also support site priority resynchronization, which is used primarily for active-readonly replication. The need to resynchronize in multi-master replication is more crucial than for active-readonly configurations, but it is likewise more complex.

PROCESS CLUSTERING EXPLAINED

➔ This paper is over 60 pages so far compared to most IOUG papers of about 7 to 10 pages. Due to space and time considerations, this version bound for the conference Proceedings Manual/CD is incomplete for this section of the paper. Look for subsequent releases of this paper at www.ioug.org (IOUG Live 2005 conference), or contact the authors directly — see *About the Authors* at the end.

PROCESS CLUSTERING PRODUCTS VERSUS ORACLE

➔ This paper is over 60 pages so far compared to most IOUG papers of about 7 to 10 pages. Due to space and time considerations, this version bound for the conference Proceedings Manual/CD is incomplete for this section of the paper. Look for subsequent releases of this paper at www.ioug.org (IOUG Live 2005 conference), or contact the authors directly — see *About the Authors* at the end.

BASIC HA/DR SCENARIOS AND EXAMPLES

➔ This paper is over 60 pages so far compared to most IOUG papers of about 7 to 10 pages. Due to space and time considerations, this version bound for the conference Proceedings Manual/CD is incomplete for this section of the paper.

IMPORTANT

— **Examples of enhancements scheduled for subsequent releases of this section include the following:**

- **Addition of a REQUIREMENTS METRICS GAUGE** for each of the solutions under each basic scenario. The gauge can be useful for **proactive** strategy, evaluation, and architecture as well as for **retroactive** crisis management in quickly determining the most appropriate recovery method based on available infrastructure. It's nice to have a quick decision guide available when you're under fire, especially if your DR plan does not cover a variety of options for each type of crisis. Some examples you'll be able to see at a glance (using the Requirements Metrics Gauge)...

Plain old Oracle RMAN is one of the fastest (MTTR, RPO, and RTO) and least expensive (free COST) recovery methods for minor logical block corruption. Site Failover using Oracle Data Guard or RDM technology is fastest for more extensive damage, particularly catastrophic events. While Data Guard cost is unbeatable (free) compared to RDM, only RDM automates replication of NON-database files such as source code, configuration files, product binaries, etc, so its convenience (EASE OF USE metric) is the high scorer in this case.

- More coverage in this section for THIRD PARTY PRODUCTS. Meanwhile, refer to the section *Advanced Third Party HA/DR Solutions* which includes a "...Versus Oracle" comparison for each of the products covered in this paper.

Look for subsequent releases of this paper at www.ioug.org (IOUG Live 2005 conference), or contact the authors directly — see *About the Authors* at the end of the paper.

SCENARIO 1: COMPUTER FAILURE

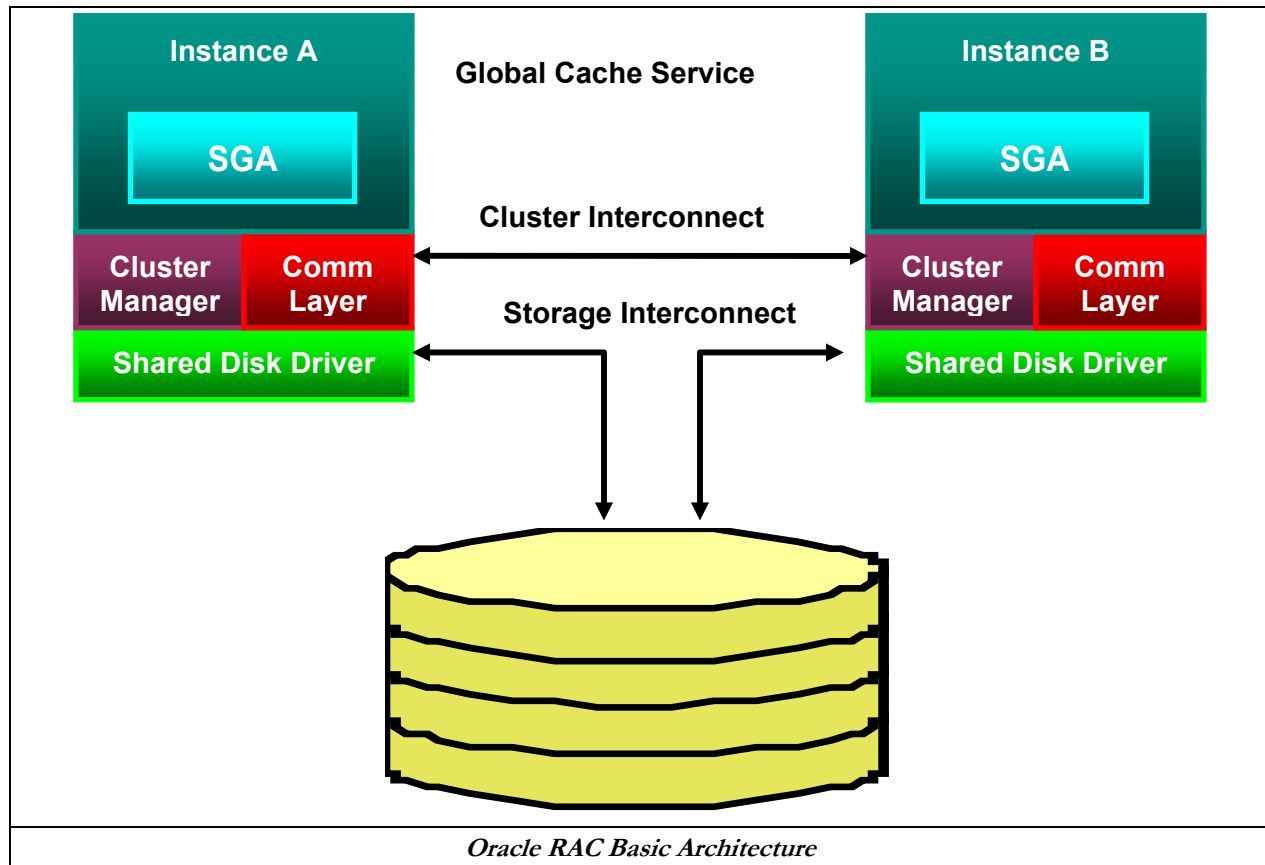
Most cases of failure are due to hardware breakdown, such as the CPU, memory, and other circuit components failures. In addition, corrupt software within a computer is another major reason, which includes corrupt service drivers, operating system (OS), Firmware, and virus. Sure, there are various solutions for computer system failures. However, these types of failures are best remedied by taking advantage of fast database crash recovery and cluster technology, which can achieve the highest availability.

ARCHITECTURE OF ORACLE 9I/10G REAL APPLICATION CLUSTER (RAC)

The central point for the solution of system failures is to provide redundant hardware and software system, which can protect system failure from individual computer. Oracle released Oracle Parallel Server (OPS) from version 7, and due to the performance issue, with Oracle9i, Oracle released Real Application Clusters (RAC), which is the multi-node extension to Oracle database server. Powered with Cache Fusion and Transparent Application Failover (TAF), Oracle9i RAC enables e-business application with guaranteed continuous availability and horizontal scalability.

The following diagram depicts a basic architecture of Oracle9i Real Application Clusters. The components of Oracle9i Real Application Clusters include servers (usually SMPs), a cluster interconnect and a shared disk subsystem. Each node in a cluster environment is independent, but inter-connected; the cluster management software provided by the hardware vendor provides the structure transparency. The shared database files (data files, control file, and redo log files) are physically or logically connected to each node and placed in the shared disks for read and write by cluster nodes. Oracle9i RAC software manages the data access from interconnected servers.

The performance of a cluster database environment is based on the database effectively utilizing a fast interconnect between cluster nodes. Oracle9i RAC is the first successfully shared-disk cluster architecture and utilizes Cache Fusion fused cache among cluster nodes to allow high performance by getting rid of slow disk I/O.



Cache Fusion technology is introduced by Oracle, which enables RAM caches are shared among multiple Oracle instances running against a single database. It is a breakthrough in high availability clustered solutions. Oracle9i RAC enables instances to find global resources and database blocks that are currently in use by one or more instances. The Global Cache Services are responsible for transferring information like database block images among instances using the cluster interconnect. Therefore, the implementation of Cache Fusion enhances the performance of cluster system by avoiding the slow I/O and enables continuous cluster availability.

The “fusing of cache” for all the cluster instances is very exciting and promising. It provides users on any nodes of the clustering system an extended database cache both for read and write. It is the underlying technology that enables Oracle9i RAC. A test was performed and the results showed that the cache block shipping greatly enhances the performance as following: local DB cache < 1ms, remote DB cache < 5ms and hard disk unit < 20-80ms. Beyond the performance improvement, feature of high scalability is also achieved. For example, whenever the processing load in a clustered database system becomes excessive, additional nodes can be added to reach the highest throughput for production. Therefore, the transparent scalability with Oracle 9i/10g RAC enables continuous increases in throughput in the presence of limited increases in the processing capability. To achieve this, Oracle Net services provides client load balancing and connection load balancing, where client load balancing features randomizing the client connections among all listeners serving a database to distribute the load, and connection load balancing features improves connection performance by balancing number of active connections

among multiple instances and multiple dispatches if shared server configuration is employed. Remote listener is configured to listen to all instances, and to be aware of the load information of all instances and dispatches. For dedicated server configuration, a listener selects an instance in the order: 1) least loaded node, and 2) least loaded instance. For a shared server configuration, a listener selects a dispatcher in the following order: 1) least loaded node, 2) least loaded instance, and, 3) least loaded dispatcher for the instance. Round robin load balancing is also configurable (ironically, by setting load balance OFF in listener/tnsnames configurations).

TRANSPARENT APPLICATION FAILOVER (TAF)

Obviously, the central feature of Oracle9i RAC database is to provide system availability solutions. However, if the requirement is for the system to survive disasters, Oracle9i RAC cluster database alone is not enough. Applications combining the Oracle Net8 Transparent Application Failover option and Oracle9i RAC database can fail over transparently the crashed instances to the surviving ones in the cluster system. However, for a connecting application using TAF, Oracle Call Interface (OCI) is required running on it because the client must use the Oracle Net OCI libraries to take advantage of TAF functionality. This includes OCI programs, JDBC thick drivers (OCI drivers), ODBC connections and SQL*PLUS connections.

When an Oracle9i RAC node (instance) fails (such as hardware failure), the connecting applications are automatically rerouted to surviving node (instance) and continue the processing. Generally, multiple Oracle instances are concurrently active on multiple nodes and these instances synchronize access to the same database. All nodes also have concurrent ownership and access to all disks. When one node fails, all other nodes in the cluster maintain access all the disks; there is no disk ownership to transfer, and database application binaries are already loaded in to memory. This makes failover process in Oracle9i RAC rapid in server side. The duration of failover depends on the database size.

FAST-START FAULT RECOVERY

The Oracle Database provides very fast recovery from system faults and crashes. However, equally important to being fast is being predictable. The Fast-Start Fault Recovery Fault technology included in the Oracle Database automatically bounds database crash recovery time and is unique to the Oracle Database. The database will self-tune checkpoint processing to safeguard the desired recovery time objective. This makes recovery time fast and predictable, and improves the ability to meet service level objectives. Oracle's Fast-Start Fault Recovery can reduce recovery time on a heavily loaded database from tens of minutes to less than 10 seconds.

SCENARIO 2: MEDIA (STORAGE) FAILURE

In most cases, the disaster recovery is with the storage or media failures, which include disk failures caused by failed storage components, such as Disk Controller, HBA, RAID, disk-array, lost of disks, data files corruption or loss, etc. As discussed above, computer or system failures can be easily resolved by the Oracle RAC, or Fast-Start Fault Recovery Fault, however, the media failure is not that easy and fast for the recovery process. There are several solutions to storage or media failure within Oracle Maximum Availability Architecture. In Oracle 10g, Automatic Storage Management (ASM) provides superior data protection from media failure. Absolutely, Recovery Manager (RMAN) is also another fast and effective method for data recovery to the point of failure for media failure.

AUTOMATIC STORAGE MANAGEMENT (ASM)

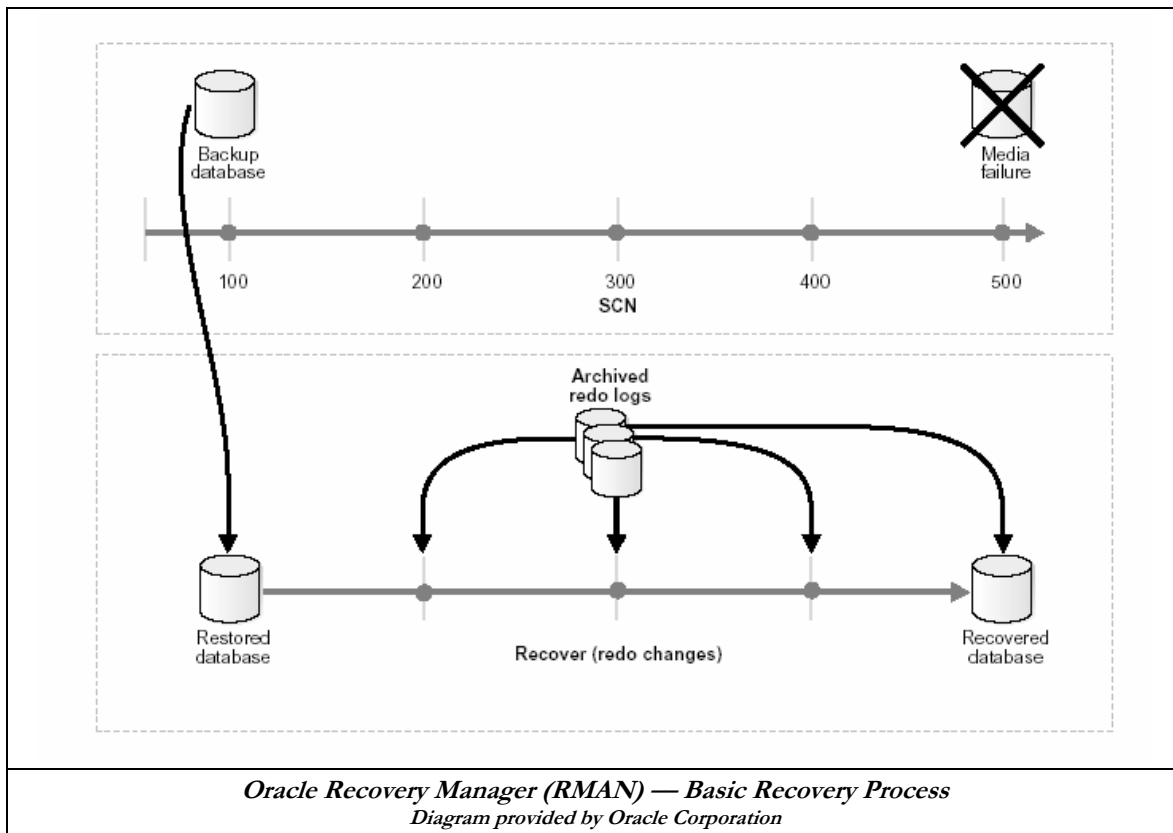
ASM is an exciting feature for Oracle 10g database. Why ASM? Consider following scenario: you are the Oracle DBA, and you often spend a significant of time in defining and implementing how you will protect and mirror your data when provisioning the storage, and monitoring the performance and identifying the hotspots (areas of disk I/O). ASM provides a vertically integrated filesystem and volume manager directly in the Oracle kernel, resulting in much less work to provision database storage, with a higher level of availability, without the expense, installation and maintenance of specialized storage products, and provides unique capabilities for database applications. ASM can provide storage management for single SMP machine or

across multiple nodes of a cluster for RAC support. ASM distributes I/O load across all available disks to optimize the performance while removing the need for I/O tuning, and absolutely and more importantly, ASM eliminates the requirement and complexity for manual management of data and disks by automating the database file naming, layout and management, setting up mirroring, adding disks, and removing disks. For some database and data warehouse system, there are hundreds, even thousands of database files. With ASM, only a large-grained object called disk group, which is a collection of disks managed as a logical unit. This will save a significant amount of time.

Another unique and significant advantage of ASM for data protection is that it makes the concept of SAME (stripe and mirror everything) more reasonable and maintain redundant copies by applying mirroring of data on a file basis, rather than on a volume basis or at the entire disk level, to provide fault tolerance. ASM provides three types of mirroring disk group: normal redundancy, high redundancy and external redundancy. With external redundancy, ASM does not provide any redundancy for the disk group. It could be used if the data loss can be tolerated as the results of a disk failure, or third party hardware mirroring is implemented. In external redundancy, the underlying disks in the disk group must provide redundancy (for example, using a RAID storage array). With normal and high redundancy, two-way or triple mirroring is provided with an additional higher level of data protection with the use of failure groups. A failure group is a set of disks sharing a common resource (disk controller or an entire disk array) whose failure can be tolerated. For example, a failure group of disks is a string of SCSI disks connected to a common SCSI controller. A failure of controller leads to all of the disks on its SCSI bus becoming unavailable, although each of the individual disks is still functional. Once defined, when a disk fails, ASM automatically reconstructs the contents of the failed disks on the surviving disks in the disk group by reading the mirrored contents from the surviving disks to ensure that the data will be available and transparently protected against the failure of any component in the storage subsystem.

ORACLE RECOVERY MANAGER (RMAN) AND CONVENTIONAL RECOVERY

The recovery process from media corruption includes the complete recovery or incomplete recovery depending on the failure type and business requirement. Traditionally, the complete recovery is pretty easy to handle. However, the incomplete recovery is often more complex, more error prone, and sometimes even more time consuming. The recovery process consists of two basic activities which may be used independently or in conjunction with one another: rolling back and rolling forward. The archived log files and online redo logs are used to apply committed transactions to the database and the undo or rollback segments are used to rollback uncommitted transactions. This process is depicted in following diagram:



WHAT IS RMAN?

Oracle Recovery Manager (RMAN) is a well known utility for Oracle database backup and recovery. It was first introduced with Oracle8. Via Oracle 8i, 9i and 10g, this utility is more efficient, easier, and more reliable for disaster recovery process. With RMAN, you can perform hot backup, including incremental, accumulative incremental and full backup. The backup strategy is based on the business requirement, data volatility, and data volume size; also with RMAN you can perform complete and incomplete recovery. There are several different scenarios of DR process with RMAN. For example, for what reason, you found lost or corrupted control files or data files. In this situation, using RMAN is a best solution to recover the database to the point of failure.

MAJOR RMAN FEATURES WITH ORACLE9I/10G

RMAN is a powerful, reliable and robust tool. It manages the processes of creating backups of database files and restoring or recovering from backups in an easy way, just with a single command or a few lines of scripts. Very importantly, it can detect many types of corruption blocks, and makes sure that the backup does not include corrupted blocks. RMAN provides true incremental backups and automatic parallelization of backups and restores. With the perspective of high availability (HA), never use RMAN for a complete restore and recovery of the database, in contrast, tablespace or data files recovery take much less time, sometime in minute level if increment backup strategy is implemented. Oracle releases some new RMAN features with Oracle 10g, one improvement of RMAN with Oracle 10g is that incremental backups can be applied to data file image copies, this means that with the recovery method, the image copy of data file can be rolled forward to the specified time by applying the incremental backups to the image copy. The benefits of applying incremental backups to data file image copies are that the media recovery time is greatly reduced because only archive logs are needed since last incremental backups. If recovery fails, restart it and RMAN automatically determine which incremental backup is needed to apply. Another excellent feature of 10g RMAN is the optimized incremental backup, Oracle introduce a new change tracking file, which is used to track the changed blocks as redo is generated. RMAN uses the change tracking file to determine the required blocks to read for incremental backups. This greatly enhances the incremental backup speed because RMAN need not to read the entire data

files.

We think it is worth to mention that with Oracle10g RMAN, we can recover data files without a backup. RMAN use the control file to retrieve the information, and all archive log files for recovery to get the lost data files back.

RMAN AND FLASH RECOVERY AREA — DISK IS FASTER THAN TAPE

Without a doubt, the recovery process is error prone and can be time consuming. In most cases, SBT tape is used as the backup media. Oracle offers a media management/proxy copy that enables third-party backup and recovery software to integrate with Oracle RMAN to facilitate the backup. The recovery time is the sum of data restore from backup tape, applying archived logs and online redo logs and recovering database. Reducing the recovery process time is to reduce the MTTR, and enhance the availability. Also, industry analysts have long noted that tape backups fail to fully restore as much as 50% of the time. This level of avoidable risk is indefensible. One of the solutions to reduce the recovery time and reduce recovery risk is to use disk-based online backup. With Oracle 10g, a more efficient, fast and reliable feature of RMAN backup and recovery is introduced, called Flash Recovery Area, which is a disk location in which the database can store and manage files related to backup and recovery. The Flash Recovery Area, a single directory on disk or ASM disk group, consolidates all recovery-related files, including control, data, archive logs, RMAN backup set files; everything needed by RMAN upon recovery can be retrieved from this directory. With a rapidly dropping price, disk has become a more attractive option for primary backup storage, in addition to faster read/write performance than tape. In addition, Flash Recovery Area can automatically warn administrators of disk capacity issues and obsolete outdated backup sets to reclaim space. Using the flash recovery area for backup to disk provide significantly faster recovery than from tape.

RMAN FAST RECOVERY — SWITCHING DATABASES IS FASTEST

Prior to version 10g, RMAN offered a hot restore feature which helped automate the process of editing the control file (trace dump) to point to an existing image backup of all data files and then restarting the database and resetting logs as needed. You could perform the process manually in just a bit more time using a regular hot backup and text editor. Oracle 10g RMAN introduced a new SWITCH DATABASE feature which decreases MTTR even more (no need to restart the database as a separate step). Of course, both the 9i and 10g methods are drastically faster than most any other kind of restoration involving copying backup pieces back to the original or to a cloned database. Note that using a Flash Recovery Area for storing backups can complicate things. For example, once you switch to the backup set, the Flashback Recovery Area now contains online/open files which can no longer be backed-up to tape without intervention and cannot be used in an incremental backup aggregation scheme (see *Oracle HA/DR Features, Strengths, and Weaknesses* section).

SCENARIO 3: HUMAN ERROR

One of the most central roles of DBA/architect is to define sound and efficient strategy and plan for high availability (HA), data protection (DP) and disaster recovery (DR). To provide powerful HA/DR solutions, it is imperative to analyze the business continuity requirements, and present the most likely data loss scenarios. As mentioned above, human errors are primary causes of the majority of outages in daily IT operations, and they are hard to avoid, then it attracts much attention of database vendors to provide a technology to recover from these human errors or logical errors in a easy and fast way, such like that a user drops the wrong tables, commits data updated with an incorrect “WHERE” clause, and so forth, you can easily get the table or data back without performing any point in time data recovery. Following is an example of these scenarios:

The current time is 5:30 PM on Friday. You are ready to leave office and looking forward to spending time with kids on weekend. You just stepped out the office, and your pager was ringing. A message from your human resource department told you that the employee (EMP) table in your HR database couldn't be found. You rush back to office, and you felt the blood pressure start to rise.... The problem was caused by your training DBA, who by accident dropped the table. The table was dropped around 4:00 PM. The database activity is minimal because most the company staffs are leaving for home. Absolutely, the table has to be recovered...

There are various cases caused by human errors. This example is just one of them. There are a few traditional solutions for this scenario. Firstly, the IMPORT utility can be used to bring back the dropped table, because you have full EXPORT data last Sunday. However, this is not a good choice at all for the data recovery! The table dropped is prerry volatile and pretty much change happened with the HR data last week, you will loss trsnasctions. Important consideration using import utility is that import process is time consuming, you had better perform an import testing on separate database before applying the same

onto the production, and depending on the size of the database, these procedures can consume hours or days of precious recovery time.

A well known methodology for data recovery in Oracle database is recovery manager (RMAN), and possibly, this is one of the best choice. The basic requirement for this solution is that the database run with archive log mode, and you have full backup or incrementall backup implemented within your database system. The advantages of RMAN is that you can recover data prior to the time of failure, such as above case (dropped tables) or deleted data erroneously in important tables, when immediately notified of failure. RMAN can provide data recovery in situation where complete recovery is not possible. However, the disadvantages are obvious, (1) during recovery, the database is taken back in time, so data after this point is lost and must be reentered, (2) the total recovery time is significant, because time to recover the database is the length of time your hardware can backup the existing database, restore all data files from disk or tape, and roll the database forward to a point-in-time before the error, the whole process can take many hours, depending on the size of the database, and (3) since the database must be started in MOUNT until recovery process is completes, the period of downtime might be unacceptable in environments that require a high degree of availability. If the logical errors can be confined to certain tablespaces, then a tablespace point-in-time recovery (TSPITR) can be performed while the rest of the database remains accessible. However, in TSPITR, the appropriate datafiles are still taken offline, restored from backup, rolled forward to the required point-in-time, and finally put back online. Note that the entire tablespace is recovered; included tables that were not affected by logical errors are also recovered to the point-in-time.

ORACLE 10G FLASHBACK TECHNOLOGY FEATURES AND CAPABILITIES

Leveraged several technology features, Oracle 10g flashback features can perform the query to retrieve the past data, show the detailed history of data changes in row or table level, and recover the tables or rows or database to the point in time. Due to the capability of supporting viewing and rewinding data back and forth in time, Oracle Flashback technology provides the fastest and most efficient method of data recovery from potential disasters. The following tables contains the Oracle 10g flashback new features and the capabilities they offer to query past versions of schema objects, query historical data, analyze database changes, or perform self-service repair to recover from logical corruptions while the database is online.

Flashback Features	Capability
Flash Query	This feature lets you specify a target time or SCN in the past and then run queries against your database to view any data at that time using the AS OF clause of the SELECT statement. Can be used to recover from any unwanted changes like an erroneous update to a table.
Flashback Version Query	Provides a mechanism to review the changed versions of all rows made to the database in a specified time interval. With this feature, You can also retrieve metadata about the differing versions of the rows, including start time, end time, operation, and transaction ID of the transaction that created the version. Therefore, you can recover lost data values and audit any changes made to the tables queried.
Flashback Transaction Query	Lets you view changes made by a single transaction, or by all the transactions during a period of time.
Flashback Table	This feature can return a table to its state at a previous point in time. You can easily restore table data while the database is online, undoing changes only to the specified table.

Flashback Features	Capability
Flashback Drop	Undo the effects of a DROP TABLE statement.
Flashback Database	This is a new and more efficient strategy for doing point-in-time recovery. It likes a “rewind button” embedded within the database to let you rewind your database to a point in time and correct any problems caused by human errors, data corruption or any other logical data errors.

The features of Flashback Table, Flashback Query, Flashback Transaction Query and Flashback Version Query all rely on the Automatic Undo Management, which was introduced in Oracle9i. Used primarily for such purposes as providing read consistency for SQL queries and rolling back transactions, these undo segments should contain sufficient information required to reconstruct data as it stood at a past time and examine the record of changes since that past time. Flashback Drop uses a mechanism of recycle bin, which is a virtual container for dropped objects. The feature of Flashback database uses a different architecture and a mechanism for quickly rewinding the data back and forth, which will be discussed in next section.

UNDO MANAGEMENT AND FLASHBACK TECHNOLOGY

Oracle introduces concepts of UNDO management starting with 9i Release 1 (aka 9iR1). Leveraged with this, Oracle Flashback features provides a mechanism to rewind data or specific objects to the snapshot in the past, and make it possible to query past data based on date, time, and System Change Number (SCN). You may ask, how far can you flash back into the past? of course, this depends on how much UNDO information retained in the database's UNDO segments, and is bounded by the time frame specified by the UNDO_RETENTION initialization parameter.

The implementation is pretty simple. There are a few UNDO related initialization parameters needed to be assigned appropriate values so that a database is enabled to use the Flashback Query and other flashback features. These parameters are:

- UNDO_MANAGEMENT – setting to AUTO, which ensures that the database is using an undo tablespace.
- UNDO_TABLESPACE – Define which undo tablespace to use. The size of UNDO tablespace is another key factor for flashback features, that determine how much information retains within the UNDO space. For Oracle Real Application Cluster (RAC) environment, each instance has its own UNDO space.
- UNDO_RETENTION - setting this initialization parameter to a value that causes UNDO to be kept for how far you can flashback in time.
- RETENTION GUARANTEE - This is a statement clause used for UNDO tablespace to guarantee that unexpired undo will not be overwritten.

It is very important to understand the effect of UNDO_RETENTION Setting. This parameter obviously will define the time windows the row versions are available. By default, this setting is 900 seconds (15 minutes); for some scenarios, if you want to flashback longer time into the past, of course, you need to set UNDO_RETENTION to a higher value, such as 10800 (3 hours). Absolutely, some databases needs longer UNDO retention durations.

FLASHBACK DATABASE ARCHITECTURE

Oracle10g Flashback Database feature provides another way of easier, faster, and more efficient point-in-time recovery in case of data corruption or data loss in database level. It is like a rewind button for your database. Architecturally, Oracle 10g introduces a few new concepts for Flashback Database:

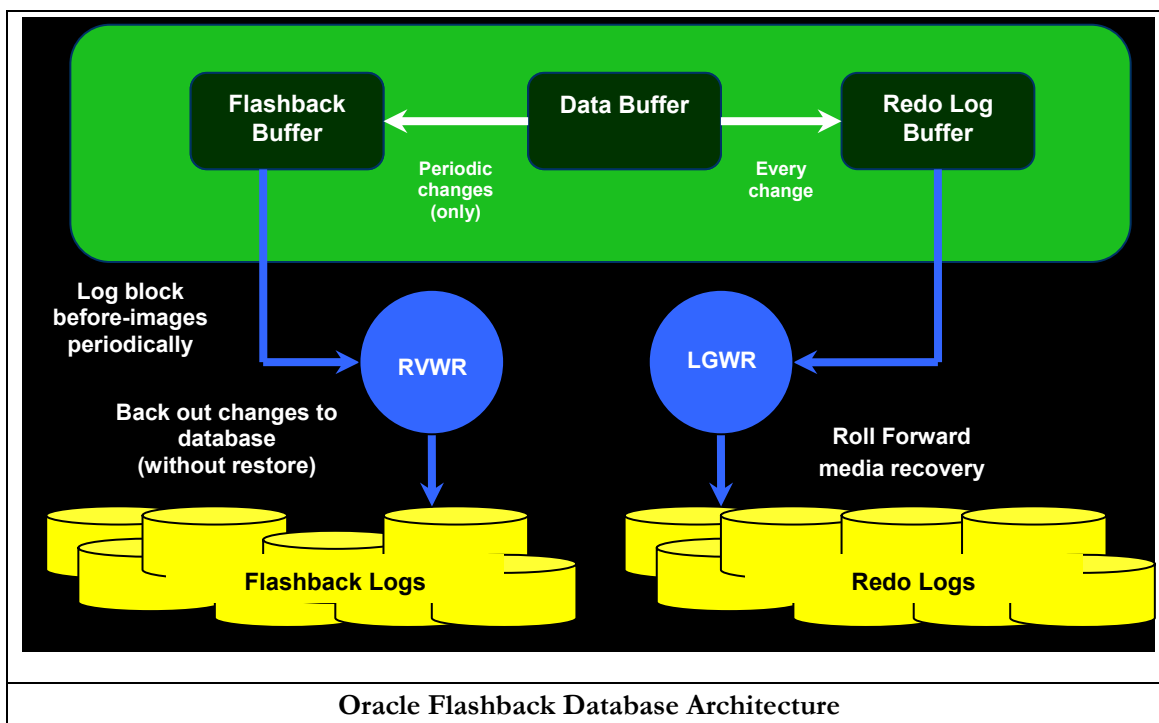
- (1) *Flash recovery area* provides a unified storage location for all recovery related files (flashback database logs, archived redo logs, and RMAN backups) within the database. It can be managed via Oracle Managed Files (OMF) or by Oracle

Automated Storage Management (ASM). Also, the Flash Recovery Area can be configured to be shared by multiple database instances.

- (2) *Flashback Database log* is another new concept. It is the old versions of changed blocks of the database, and these flashback logs are located at Flash Recovery area. You can think the flashback logs as a continuous backup or storage snapshot.
- (3) *Flashback buffer* is a new cache within SGA. It is used for caching the snapshot of changed data blocks.
- (4) *RVWR (Recovery Writer)* is a new background process, which is started whenever Flashback Database is enabled. Similar to the LGWR (log writer) process, RVWR periodically and sequentially writes the old images of changed data blocks within the flashback buffer as flashback logs onto the Flash Recovery Area.

Flashback database logs are used to quickly back out the data files to any time at which the log is captured just before the desired target time. Then, the redo logs are applied to fill in the gap. This is why Flashback Database reduces the time required to recover the database greatly, because it doesn't need to restore backups from tape, and no lengthy downtime, and no complicated recovery procedures are required. Therefore, using flashback database as the point in time recovery strategy is just extremely fast and easy to use.

The following diagram illustrates the architecture of Oracle Flashback Database.



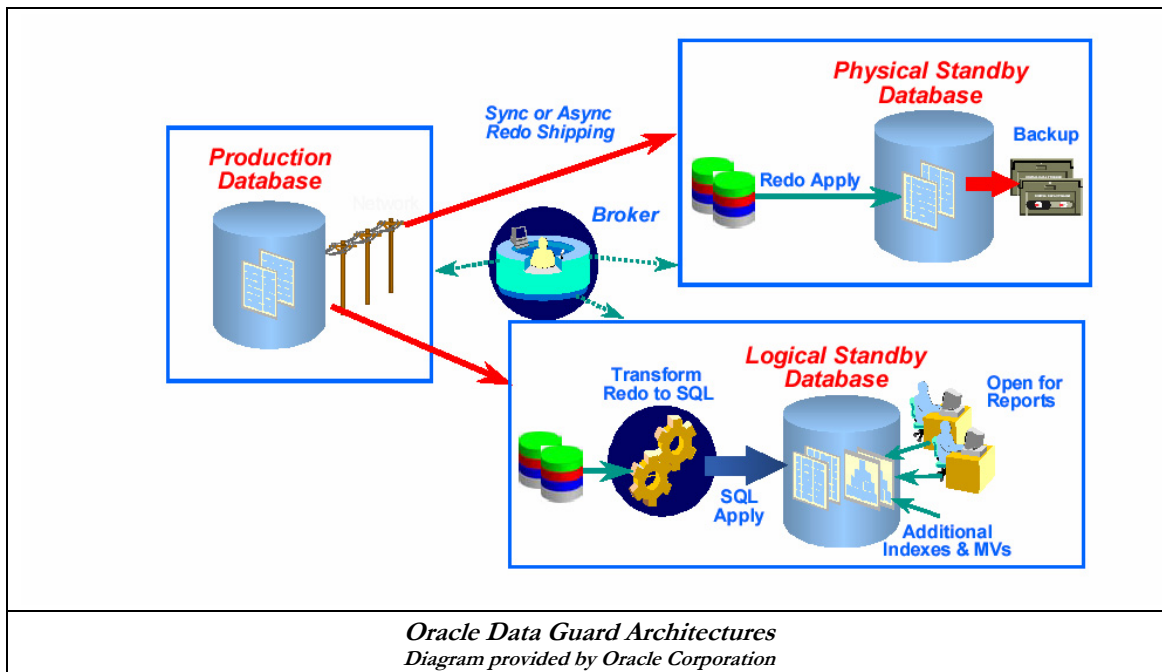
SCENARIO 4: SITE FAILURE

Oracle MAA blueprint uses Oracle Real Application Cluster (RAC) and Oracle Data Guard to provide the most effective solution to protect the core asset of any enterprise – its data and disaster recovery features in helping business survive any disasters such as hardware or system failures, data/storage failures, human errors, computer virus, software glitches, natural disaster and malicious act with a fast mean time to recovery (MTTR). Oracle RAC solution, using the methodology of redundant hardware and software, addresses local system failures and provides rapid recovery from computer failure or instance crashes. But it does not offer data protection from site failure, storage failure, and other logical data corruption. The scenario is that no matter how good a High Availability or Fault Tolerant solution you have, if the data center it resides in is

destroyed, it can no longer support your business. Oracle 9i and Oracle 10g data guard provides not only the data protection strategy, but also the data center site disaster recovery.

WHAT IS DATA GUARD?

Oracle Data Guard was first introduced in Oracle 8.0 as Oracle Standby. Data Guard is Oracle’s disaster recovery solution for the Oracle database. It is the management, monitoring, automation software infrastructure that creates, maintains, and monitoring one or more standby databases. In case site failure or other disaster, Data Guard ensure automatic switch over from primary production database to one of the standby databases, which are transactionally consistent copies of the primary production data. The standby databases at disaster recovery site can be thousands miles away from the production data center, or just within the same city and campus, across TCP/IP network. Without requiring roll forward and rollback, the failover process is fast, reliable, and efficient to minimize the downtime, and ensure the database survive a data center disaster. Following diagram presents Data Guard architecture.



PHYSICAL STANDBY DATABASE AND LOGICAL STANDBY DATABASE

As depicted in above diagram, there are two types of implementations with Oracle Data Guard: Physical Standby database and Logical standby database. Physical standby database uses Data Guard Redo Apply technology and has on-disk database structures that are identical to the primary database on a block-for-block basis, and is updated by the shipped redo data from the primary database using Oracle media recovery. Physically, both the primary and standby databases are exactly same, even the schemas and all objects with in it.

A logical standby database is an independent database, which uses Data Guard SQL Apply technology and updated by the SQL statements. SQL Apply technology transforms the redo data received from the primary database into the SQL statements, and then execute the SQL statements on the standby database to synchronize the standby database with the primary database. The physical database structure can be different, but it contains the same logical data as the primary database.

With both implementations, Data Guard automatically resynchronizes in case of network problem or corruption detected. You can also configure with a delay to ensure that corruption and logical errors are not propagated to the standby databases.

Oracle Data Guard offers centralized and simple managements for primary database and all standby databases, as well as various interactions. You can use command line SQL*Plus or Data Guard Broker, which is distributed management framework offered by Data Guard. The Data Broker automates the management and operational tasks for both primary and standby databases. It can also monitors all the systems within a single Data Guard configuration. To take the advantage of this integrated management framework, Oracle Enterprise Manager (OEM) and Broker's command line interface (DGMGRL) may be used.

DATA GUARD BENEFITS

Benefits of Data Guard and Standby Databases are obvious as following:

- Recovery Time Objectives and High Availability – both physical and logical standby databases enable robust and efficient disaster recovery and high availability solution. It facilities switchover and failover operation between primary database and standby databases, and minimize both unplanned and planned downtime.
- Data Protection – Using standby database strategy, you can ensure no data loss. It can safeguard data corruption, media failure, and logical data errors.
- Load Balance – Physical/logical standby databases are available for queries and reporting. Also, physical standby database can be used to create data backup using RMAN. These activities can save significant CPU, Memory and I/O usages on the primary site.
- Easy Use and Management – Data Guard Broker in combination with Oracle OEM provides an easy to use configuration and management framework. Also, SQL*Plus can be used for Data Guard management.
- No extra cost – Data Guard is part of the Oracle Database Enterprise License, and no need to buy additional software to implement.

Data Guard configuration consists of primary database and standby database. Both primary database and standby database can run on a single node or Real Application Cluster environment. However, Data Guard requires the operating system architecture on the primary and standby sites to be same. In addition, the same release of Oracle Enterprise Database edition must be installed and configured.

SCENARIO 5: NETWORK FAILURE

See *Redundant Network Infrastructure* earlier in this paper.

SCENARIO 6: APPLICATION FAILURE

- From an Oracle perspective, Flashback Technology, Real Application Clusters, Data Guard, and RMAN would provide various ways of recovering from application failure. Depending on circumstances and extent of damage, some of these methods would be inappropriate or even overkill.
- In consideration of third party tools, most of the advanced tools listed here would be overkill for typical application failure situations, although the invoking a full scale failover to a remote site, for example, may be safer with faster MTTR than analyzing and attempting simpler forms of recovery.

WRAPPING UP

➔ This paper is over 60 pages so far compared to most IOUG papers of about 7 to 10 pages. Due to space and time considerations, this version bound for the conference Proceedings Manual/CD is incomplete for this section of the paper. Look for subsequent releases of this paper at www.ioug.org (IOUG Live 2005 conference), or contact the authors directly – see *About the Authors* at the end.

ABOUT THE AUTHORS

Jeffrey Bertman is CTO and a principal consultant for DataBase Intelligence Group (DBIG), an IT consultancy focusing on large scale business intelligence, data center management, business continuity, and overall data engineering at both strategic and operational levels. Specializing in mission-critical services and decision support for the enterprise and targeted systems, Mr. Bertman has over 18 years of hands-on experience and has been designing and building comprehensive database environments and application systems since 1987. He has published articles and developed courseware on a wide range of technical and management topics including data warehousing, database administration (DBA), high availability, disaster recovery, business process management/engineering (BPM), IT organization, lifecycle optimization, and project management. Business disciplines include accounting, telecom, health care, marketing, and manufacturing applications. Mr. Bertman can be reached via e-mail at jefflit@dbigusa.com.

Jeffrey Bertman DataBase Intelligence Group (DBIG) Reston Town Center 11921 Freedom Drive, Reston, VA 20190	Phone: (703) 405-5432 Fax: (703) 726-0603 E-mail: jefflit@dbigusa.com
--	--

Ran Pan, PhD, OCP (8/8i/9i/10g). Senior Oracle DBA and data engineer for PCAOB. Has a decade of architecture and data management experiences in Oracle technologies and multiple relational database systems and data warehouse engineering. Taught Database Design and Implementation, Database System and Management, Oracle Programming, Oracle DBA. Has published 8 papers in peer-reviewed scientific computation journals. Regular presenter at SEOUC, OraTech and IOUG.

Dr. Ran (Ronald) Pan Office of Information Technology Public Company Accounting Oversight Board (PCAOB) 45945 Center Oak Plaza Sterling, VA 20166	Phone: (703) 547-6420 E-mail: panr@pcaobus.org
---	---